Deep Network Development

Lecture Memorandum

Course Instructors
Tamás Takács, Imre Molnár

Course Supervisor
Kristian Fenech

Department of Artificial Intelligence
Faculty of Informatics, ELTE
2026

Preface

This volume is the course memorandum for Deep Network Development (course code `IPM-22fmiDNDEG`), a Master's-level course offered by the Department of Artificial Intelligence, Faculty of Informatics, Eötvös Loránd University (ELTE), under the academic supervision of Dr. Kristian Fenech.

Scope of the course

Deep Network Development introduces students to the full landscape of modern deep learning through a weekly lecture paired with a hands-on practical session. This memorandum focuses on the lecture side of the course. The semester covers the foundations of neural networks (linear and logistic regression, multi-layer perceptrons, backpropagation, optimizers), convolutional architectures for image classification and transfer learning, the four classical computer-vision tasks (object detection, semantic and instance segmentation, pose estimation) together with their modern deep-learning solutions, recurrent networks and attention mechanisms for sequence modelling, the Transformer architecture with its vision and language variants, and finally generative models and inverse problems such as depth estimation, optical flow, and text-guided image inpainting.

Structure of this memorandum

The memorandum is organized into four parts – the weekly lecture decks followed by the graded assessments that test them – together with a bibliography:

Part I. Lectures. The nine slide decks of the weekly lecture series, reproduced in their landscape PowerPoint form. The decks are: Introduction; Linear Regression and Artificial Neural Networks; Image Classification and Convolutional Neural Networks; Convolutional Neural Network Architectures and Transfer Learning; Object Detection and Image Segmentation; Natural Language Processing Basics and Recurrent Neural Networks; Transformers and Vision Transformers; and Deep Learning Tools for Computer Vision, Parts 1 and 2.

Part II. Quizzes. Two collections of graded quiz example questions. Quizzes are administered roughly every three weeks and count towards the student's final grade.

Part III. Midterms. Eight variants of the two graded midterm examinations (base, retake, and sample papers for each), with full written solutions for every question. The midterms are sat twice during the term and contribute directly to the final course grade.

Part IV. Final Exams. Eight final-examination variants (two papers, variants A and B, across four examination periods), covering the full breadth of the semester.

References. The bibliography and external sources cited across the lecture decks.

Authorship and attribution

The course content collected here was co-developed by Tamás Takács and Imre Molnár on an approximately equal basis.

Unless otherwise noted, figures are made by the authors. Figures sourced from third parties are cited at the point of use.

Contents

Preface **2**

I. Lectures **6**

1. Introduction **7**
2. Linear Regression and Artificial Neural Networks **76**
3. Image Classification and Convolutional Neural Networks **166**
4. CNN and Transfer Learning **245**
5. Object Detection **307**
6. NLP Basics and Recurrent Neural Networks **457**
7. Transformers, Vision Transformers **540**
8. Deep Learning Tools for Computer Vision part 1 **667**
9. Deep Learning Tools for Computer Vision part 2 **738**

II. Quizzes **792**

10. Quiz 1: Question Examples **793**
11. Quiz 2: Question Examples **812**

III. Midterms **817**

12. Midterm 1 Variant A **818**
13. Midterm 1 Variant B **823**
14. Midterm 1 Retake **826**
15. Midterm 1 Sample **831**
16. Midterm 2 Variant A **836**
17. Midterm 2 Variant B **838**
18. Midterm 2 Retake **840**
19. Midterm 2 Sample **848**

IV. Final Exams **853**

20. Final Exam 1 Variant A **854**
21. Final Exam 1 Variant B **860**
22. Final Exam 2 Variant A **865**

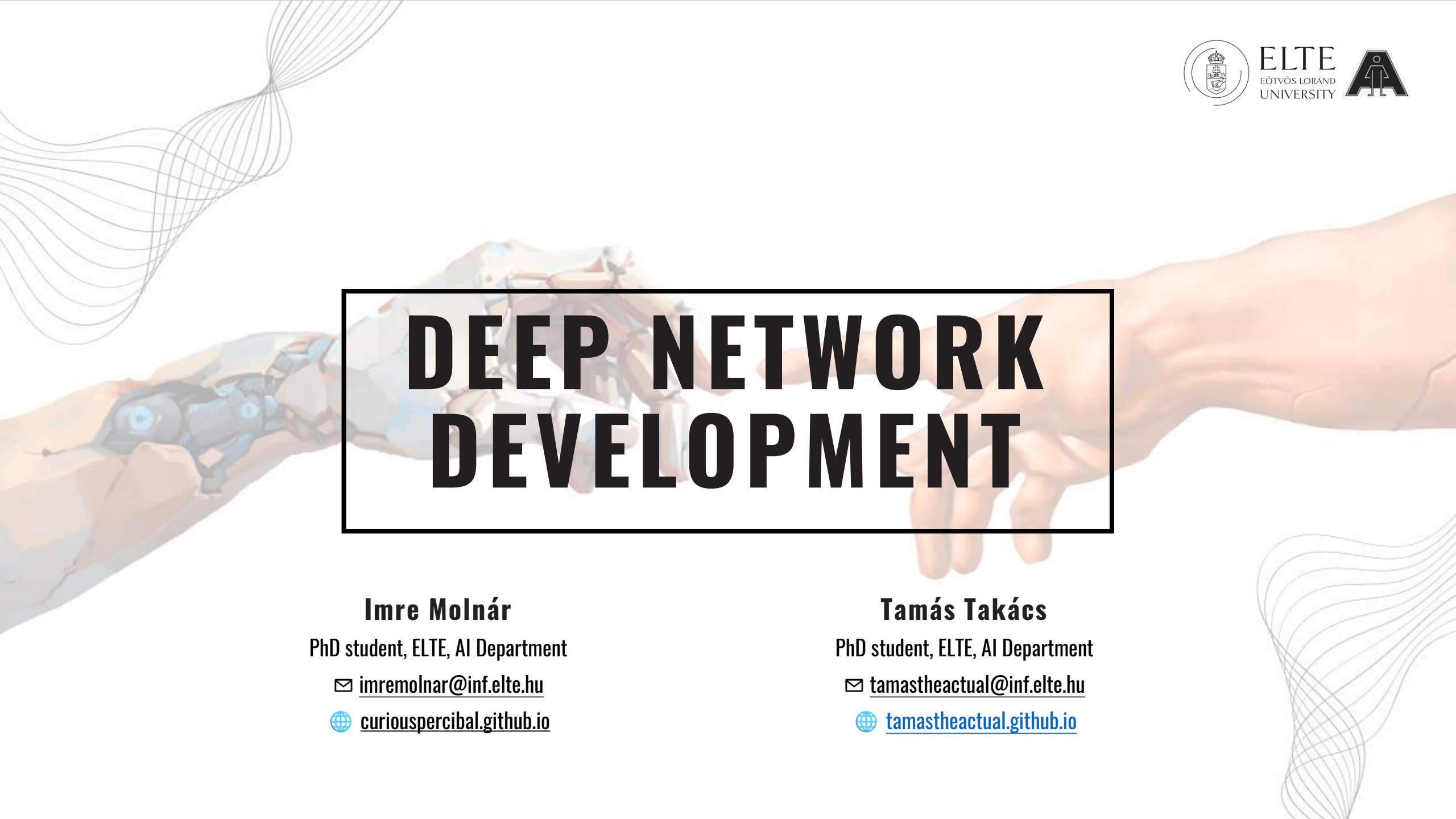
Contents

23. Final Exam 2 Variant B **870**
24. Final Exam 3 Variant A **876**
25. Final Exam 3 Variant B **882**
26. Final Exam 4 Variant A **888**
27. Final Exam 4 Variant B **893**

References **899**

Part I.

Lectures



DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

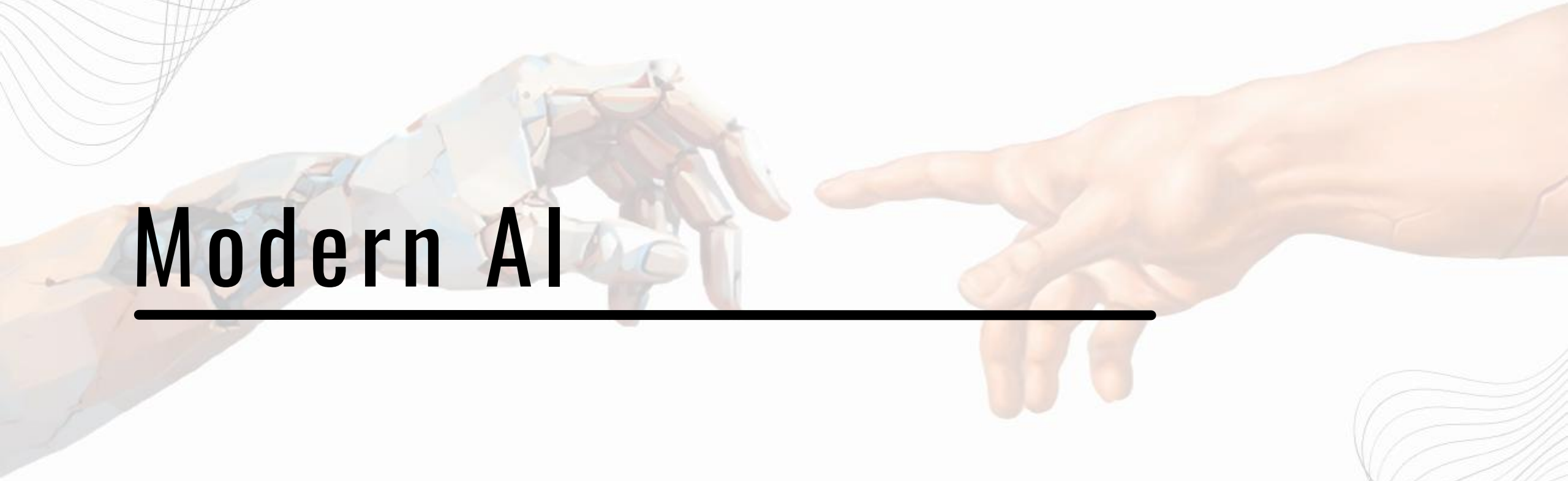
PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

Lecture 1



Modern AI

1 Course Details

2 Deep Learning Overview

3 Modern AI & State of AI

Deep Network Development

Lecture 1



Course Details

1 Course Details

2 Deep Learning Overview

3 Modern AI & State of AI

Prerequisites

Have a basic understanding of:

- Linear Algebra
- Probability Theory
- Programming Skills (for practices)

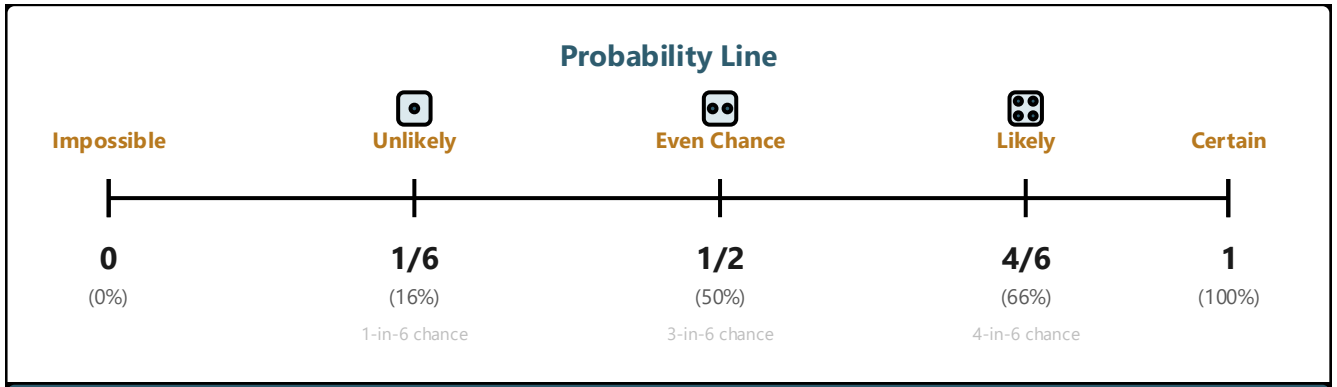
prerequisites.py

```python
import numpy as np

# Linear algebra: a matrix-vector product
W = np.array([[0.2, -0.5, 0.1],
              [ 0.7, 0.3, -0.4]])
x = np.array([1.0, 2.0, 3.0])
z = W @ x             # [-0.5  0.1]

# Probability: softmax turns scores into probabilities
p = np.exp(z) / np.exp(z).sum()
print(p.round(2), p.sum())
# [0.35 0.65] 1.0
```

Python 3 + NumPy



Matrix rules

scalar multiplication

$$n\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} = \begin{bmatrix} na & nb & nc \\ nd & ne & nf \end{bmatrix}$$

matrix addition

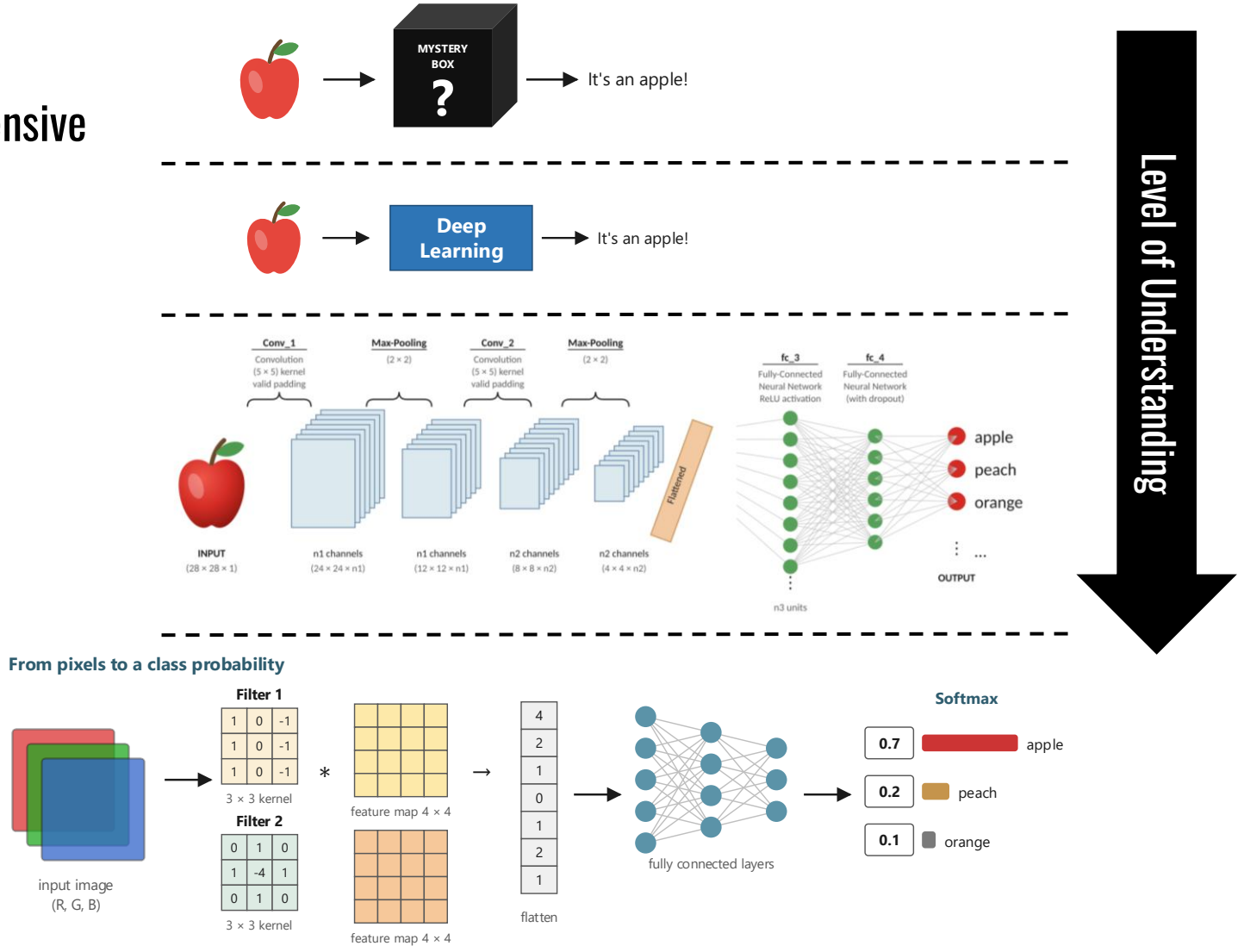
$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} + \begin{bmatrix} g & h \\ i & j \\ k & l \end{bmatrix} = \begin{bmatrix} a+g & b+h \\ c+i & d+j \\ e+k & f+l \end{bmatrix}$$

matrix multiplication

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}\begin{bmatrix} g & h \\ i & j \\ k & l \end{bmatrix} = \begin{bmatrix} ag+bi+ck & ah+bj+cl \\ dg+ei+fk & dh+ej+fl \end{bmatrix}$$

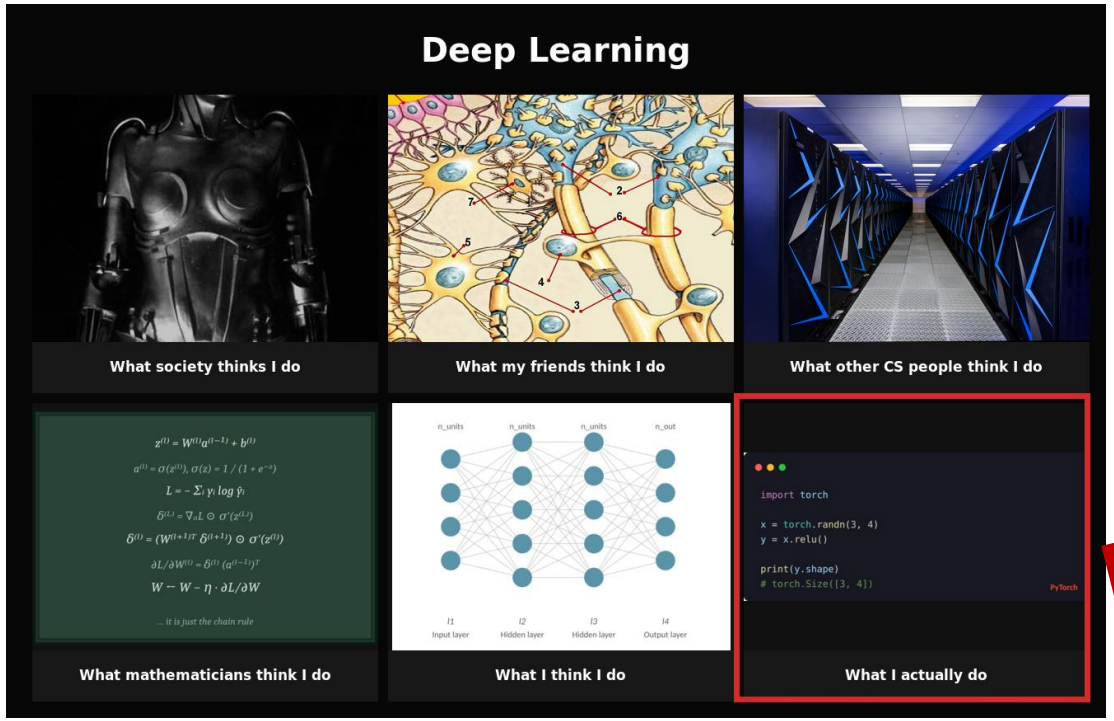
Description

- This course provides students with a comprehensive understanding of how **Deep Neural Networks** work:



Description

- The practical part of this course equips students with the necessary skills to implement various Deep Neural Network architectures using a framework called PyTorch based on Python programming language;



PyTorch

Your hero 👇

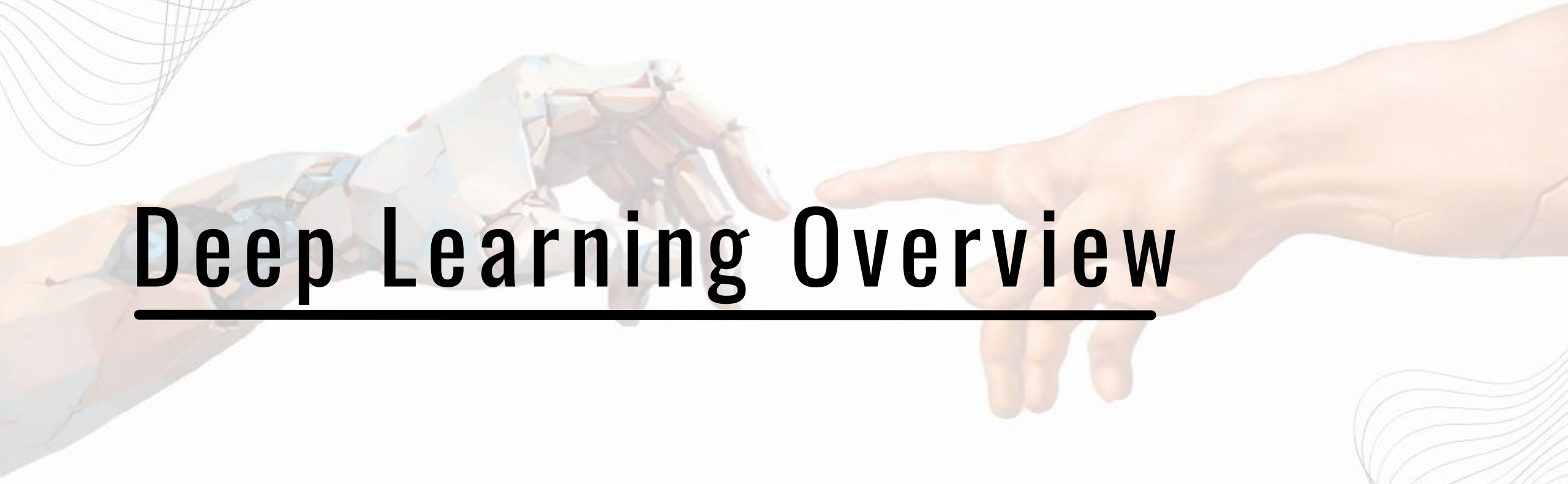


ptrblck

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 1



Deep Learning Overview

1 Course Details

2 Deep Learning Overview

3 Modern AI & State of AI

Artificial Intelligence (AI)

AI: the science and engineering of making computer systems and algorithms capable of performing tasks that typically require human intelligence.



“Intelligent” Machines / Systems

… behave like humans

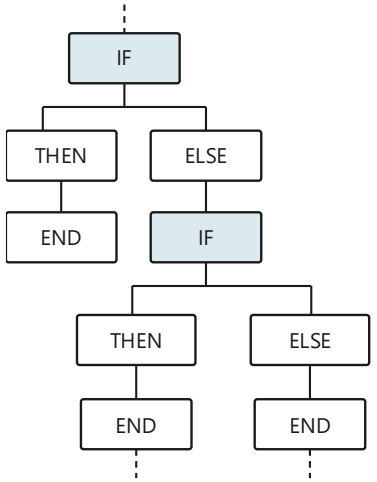
… talk like humans

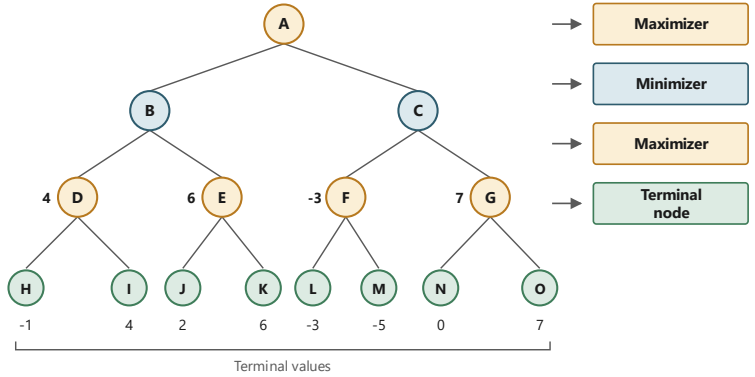


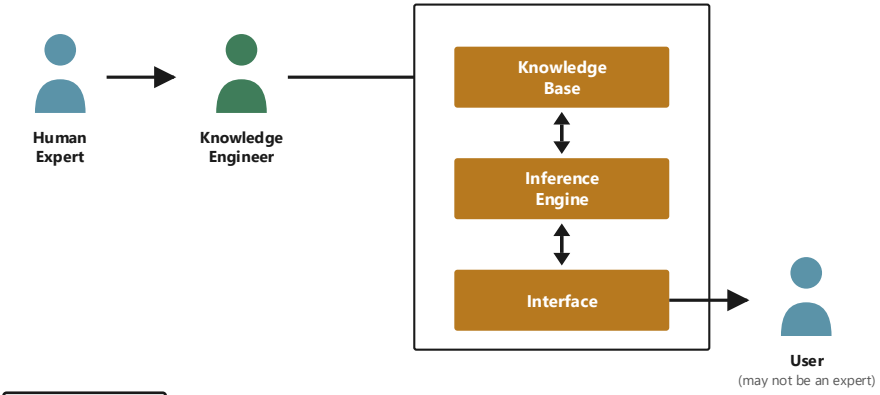
Image Source: Innorobo 2015 - NAO (cropped).JPG, © Xavier Caré / Wikimedia Commons / CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0/
Image Source: Horst von Harbou - Metropolis Maschinenmensch.jpg, Horst von Harbou, Public domain, via Wikimedia Commons

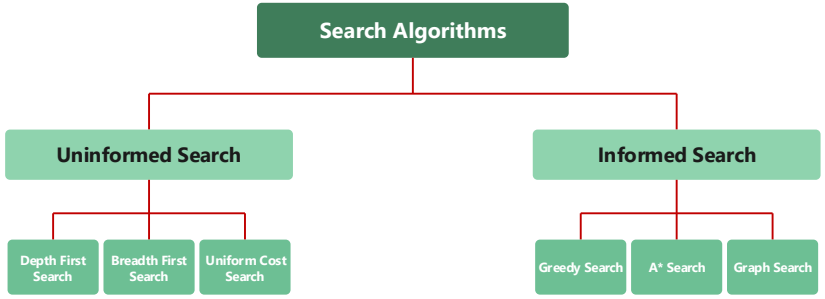
Artificial Intelligence (AI)

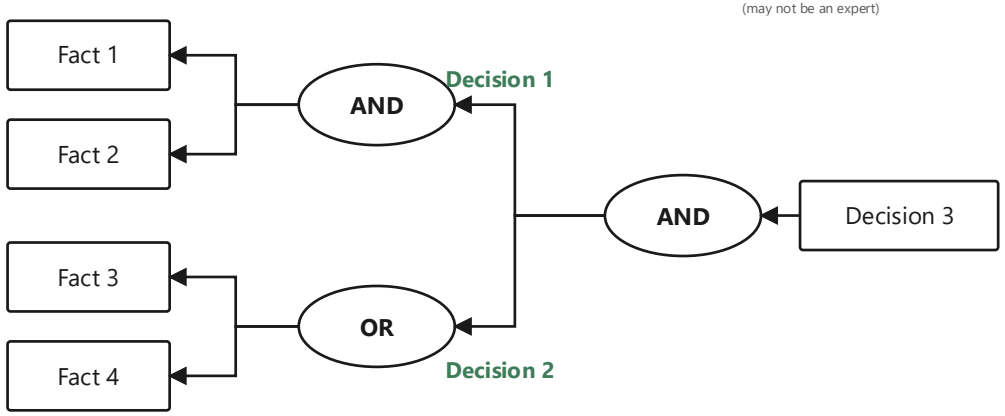
- Traditional AI methods used to be rule-based or goal oriented.
- **Expert Systems:** contain a knowledge base and rules for reasoning and decision-making





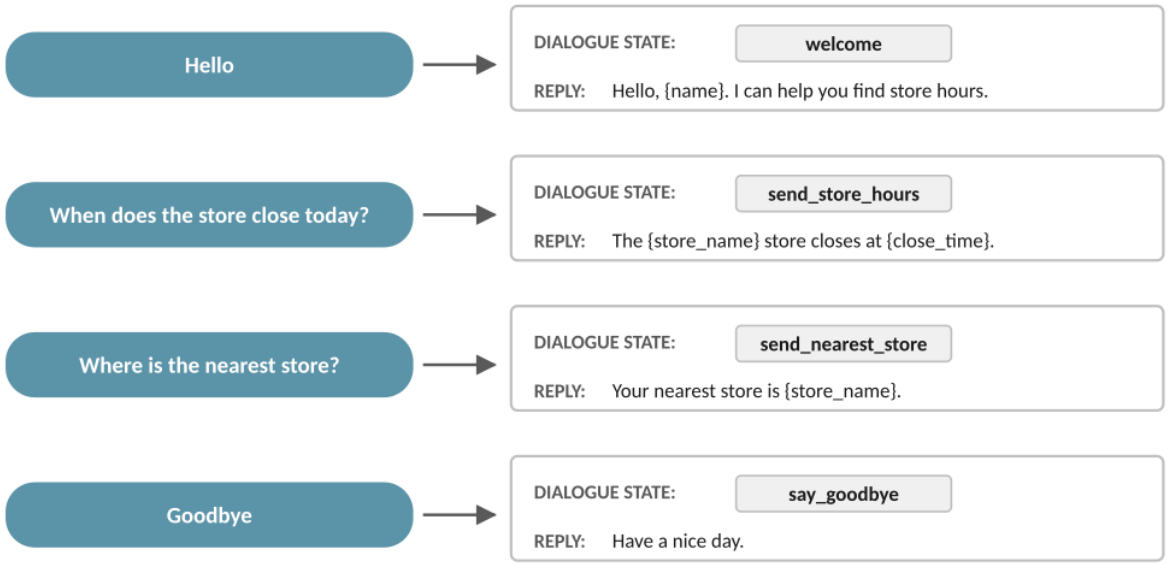


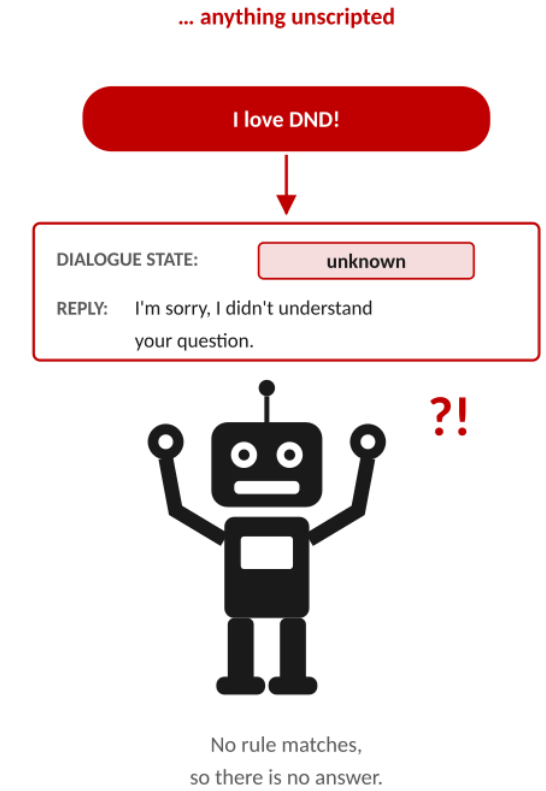




Artificial Intelligence (AI)

- Traditional AI methods used to be rule-based or goal oriented.
- **Expert Systems:** contain a knowledge base and rules for reasoning and decision-making



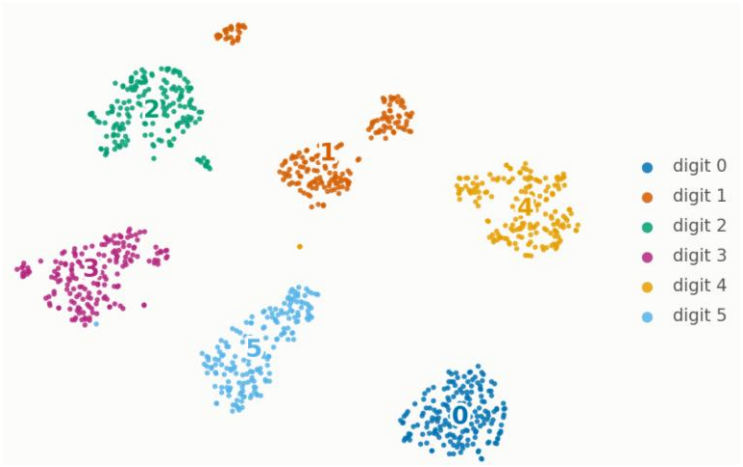


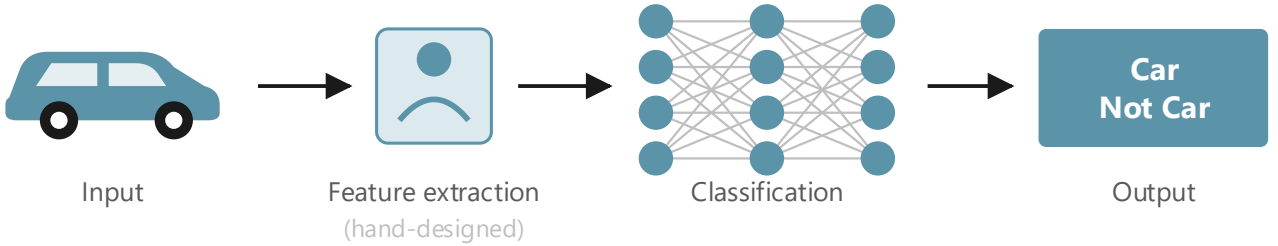
Machine Learning (ML)

ML: a subfield of AI that focuses on the development of algorithms and models capable of learning from data and making predictions without being explicitly programmed.



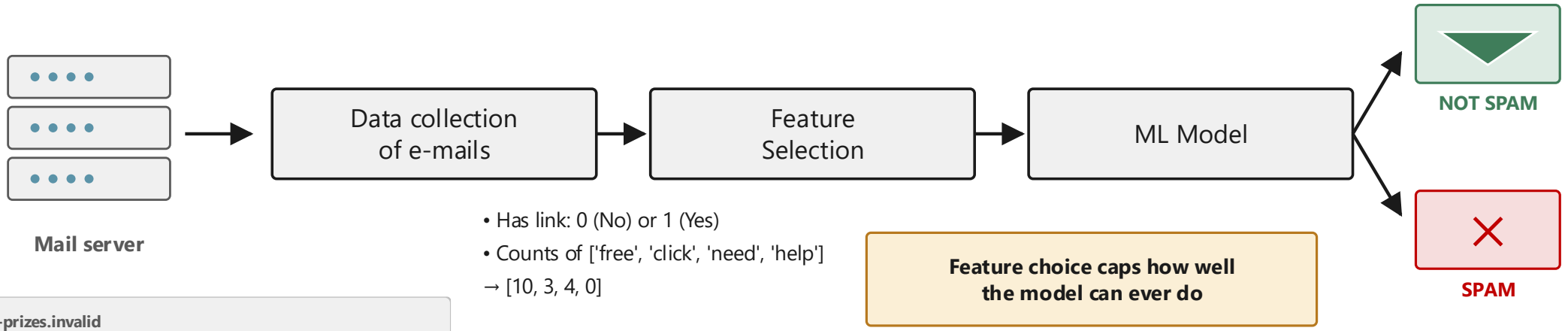
Learn patterns from data





Machine Learning (ML)

ML: a subfield of AI that focuses on the development of algorithms and models capable of learning from data and making predictions without being explicitly programmed.



From: rewards@example-prizes.invalid

CONGRATULATIONS!!! We've created a giveaway just for you! Enter using the link below for a chance to win a FREE subscription.

http://example.invalid/claim-your-free-prize

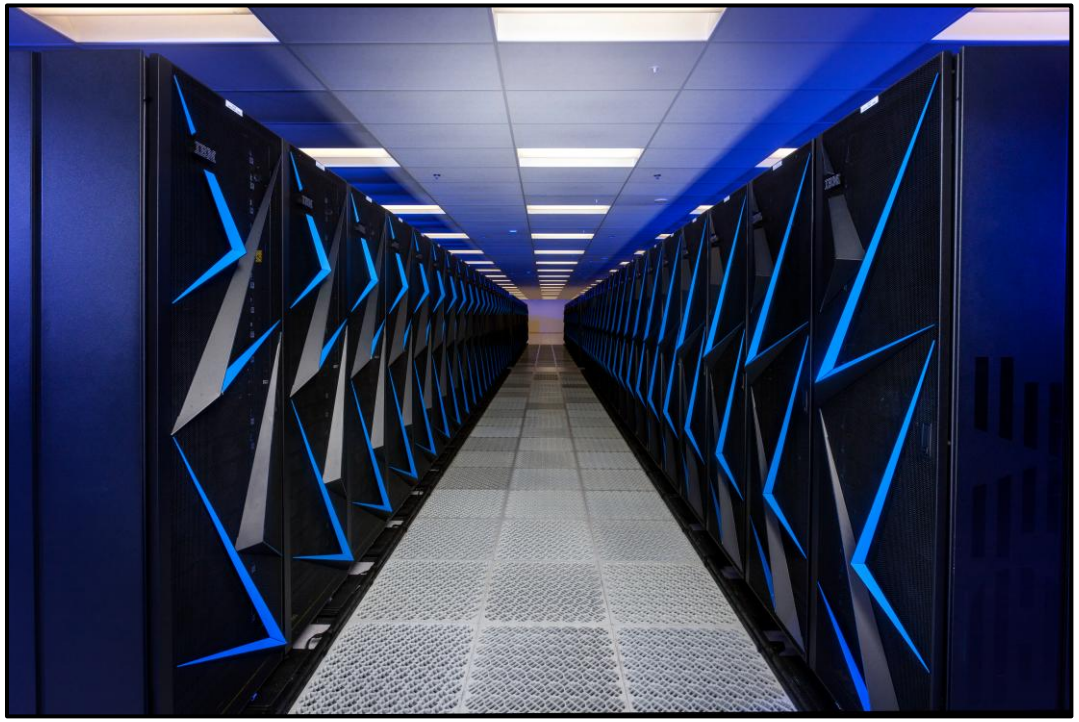
Click now, offer expires today!

Thank you again!
Our Team

free | click | need | help

Deep Learning (DL)

DL: a subfield of ML that focuses on the development and training of deep neural networks. Deep Learning algorithms leverage these deep architectures to automatically learn hierarchical representations of data, enabling them to capture complex patterns and relationships.



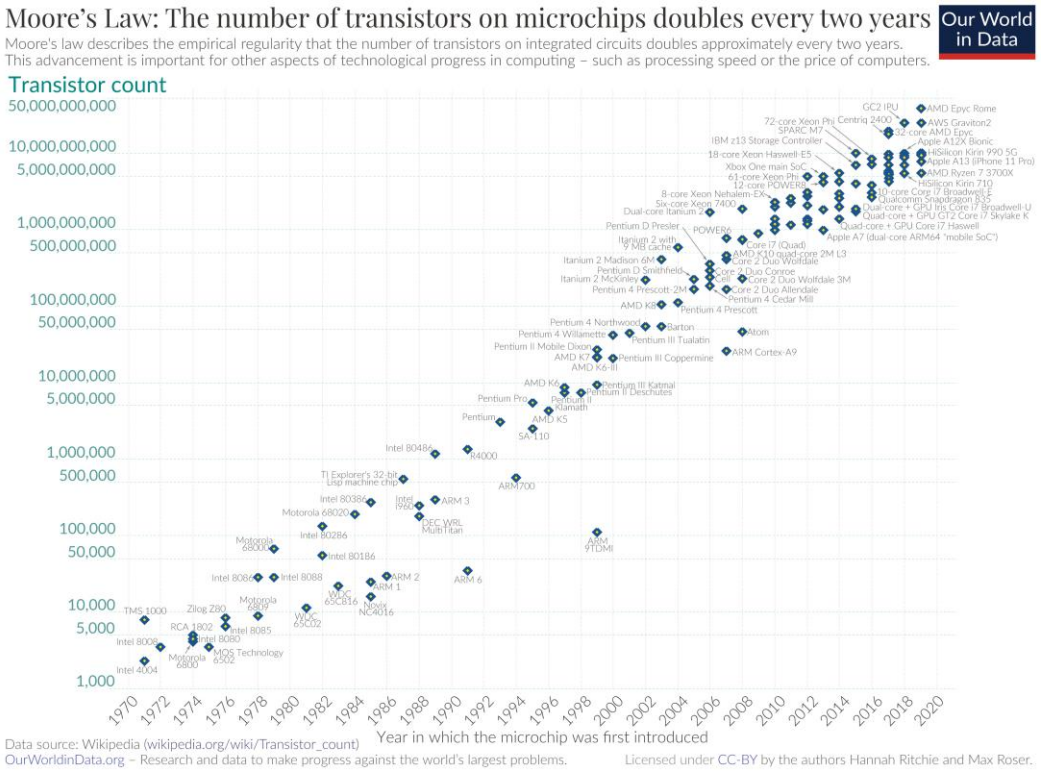
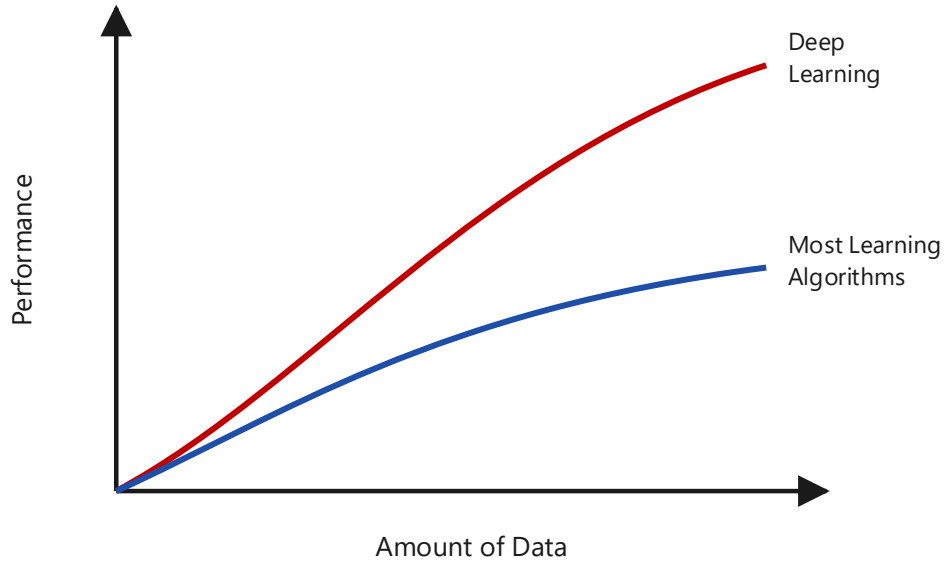
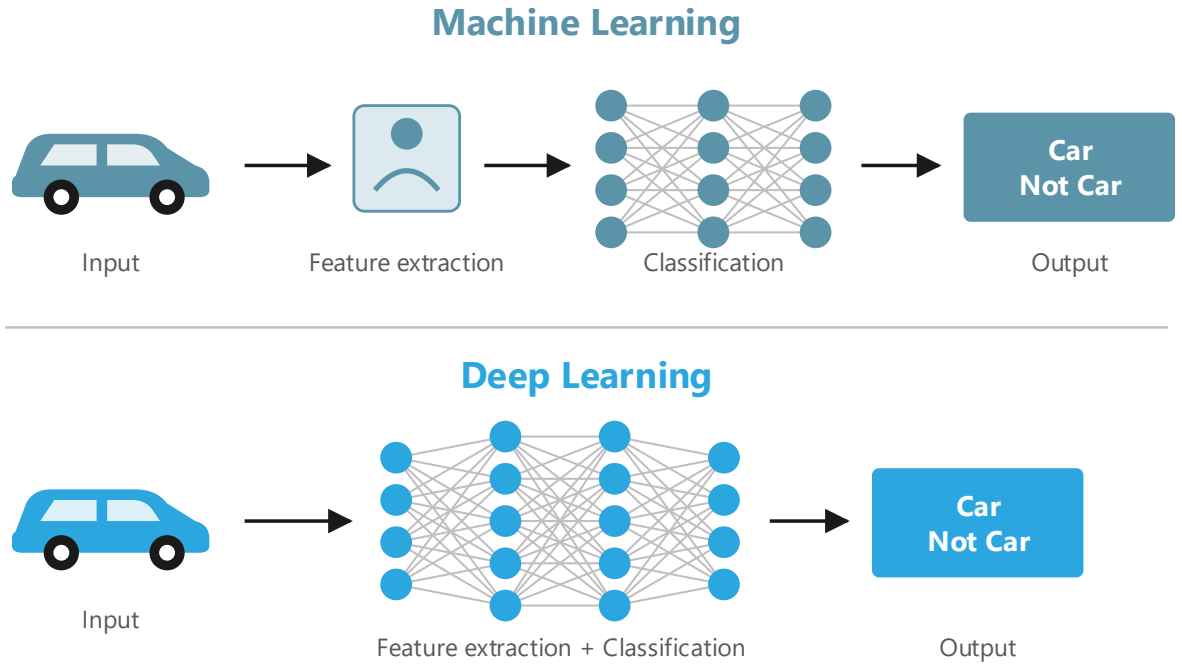


Image Source: Sierra Supercomputer (48002385338).jpg, U.S. Department of Energy from United States, Public domain, via Wikimedia Commons
Image Source: Moore's Law Transistor Count 1970-2020.png, Max Roser, Hannah Ritchie, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Deep Learning (DL)

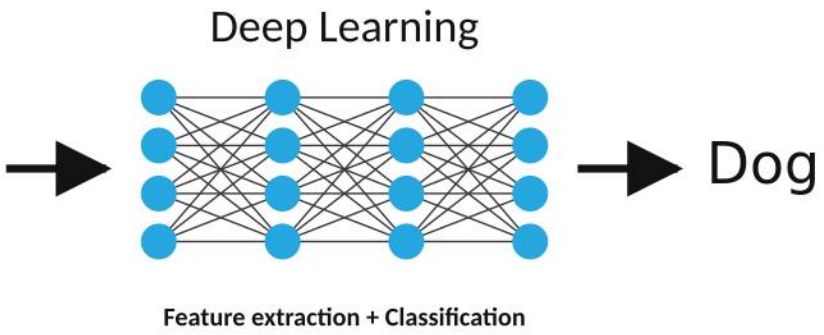
DL: a subfield of ML that focuses on the development and training of deep neural networks. Deep Learning algorithms leverage these deep architectures to automatically learn hierarchical representations of data, enabling them to capture complex patterns and relationships.



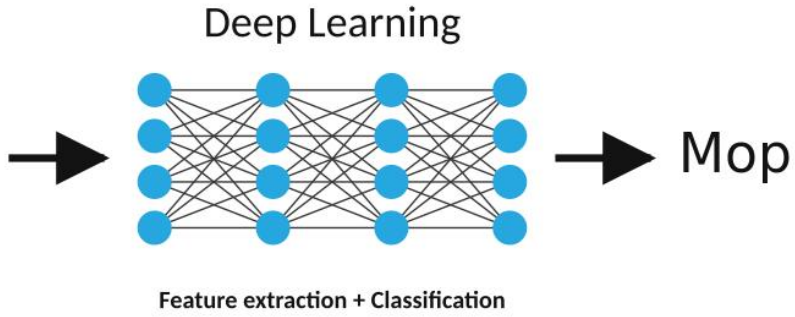


Deep Learning (DL)











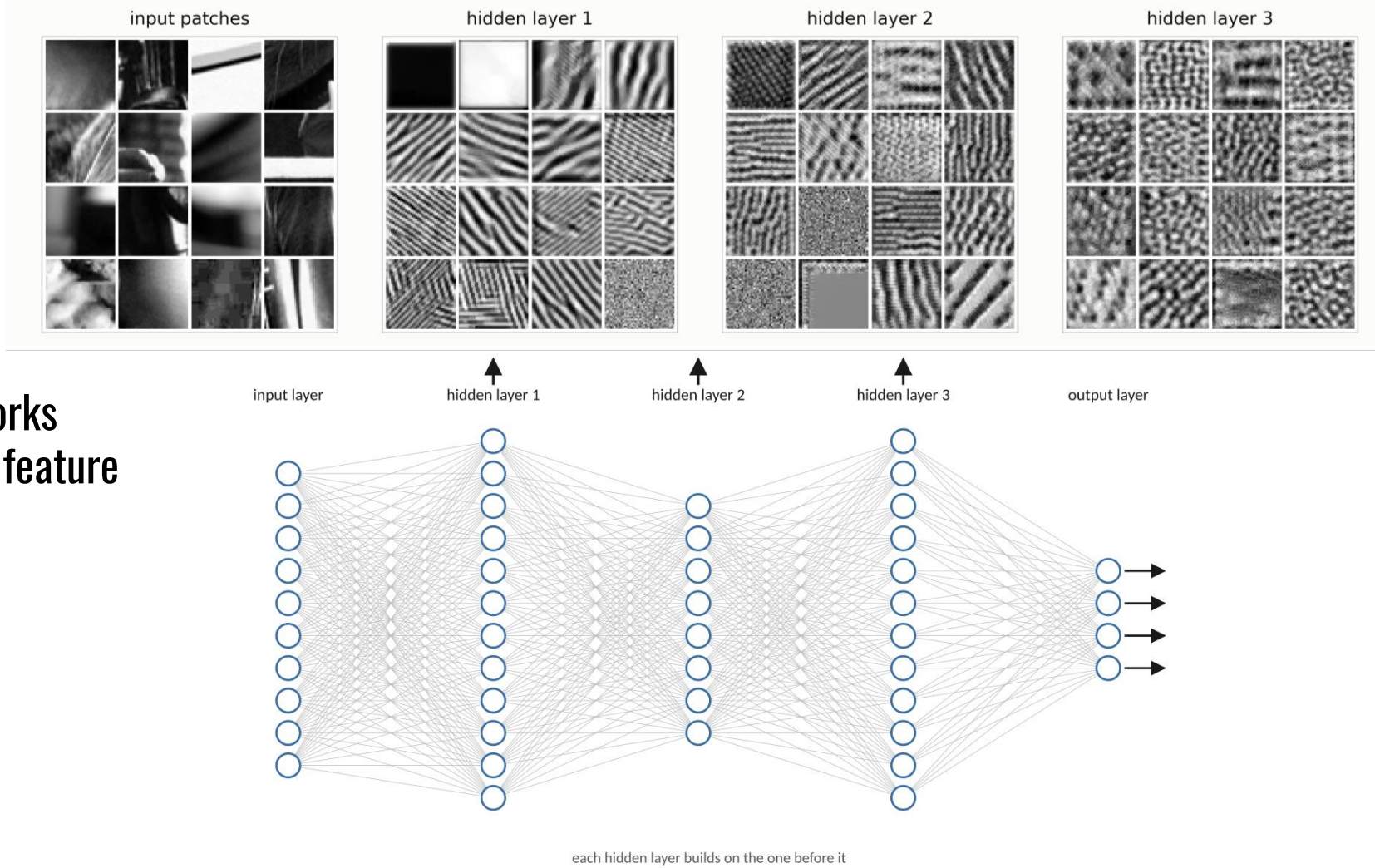








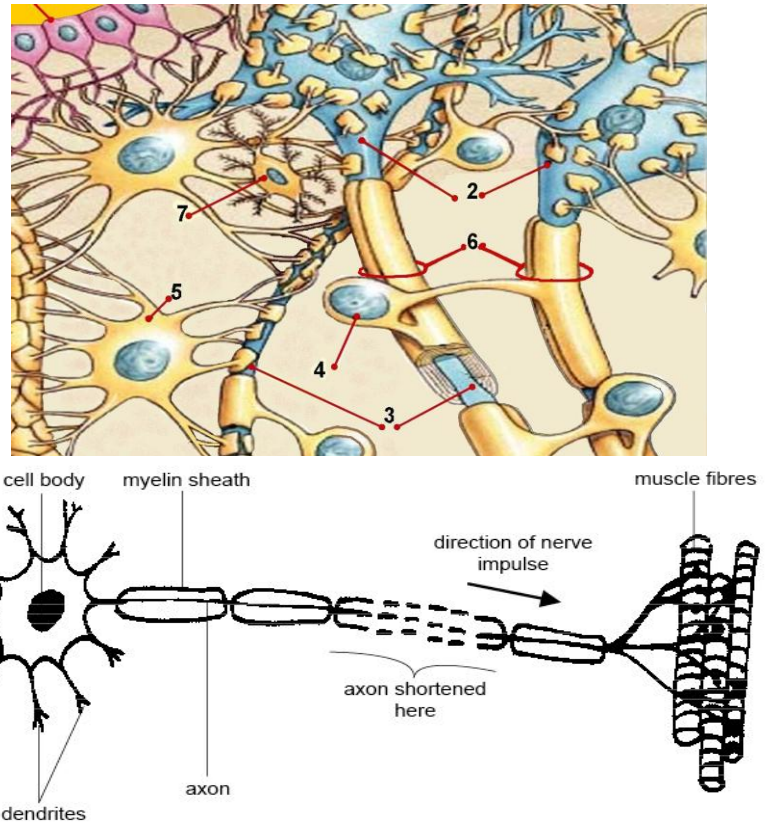
Deep Learning (DL)



Deep neural networks learn hierarchical feature representations

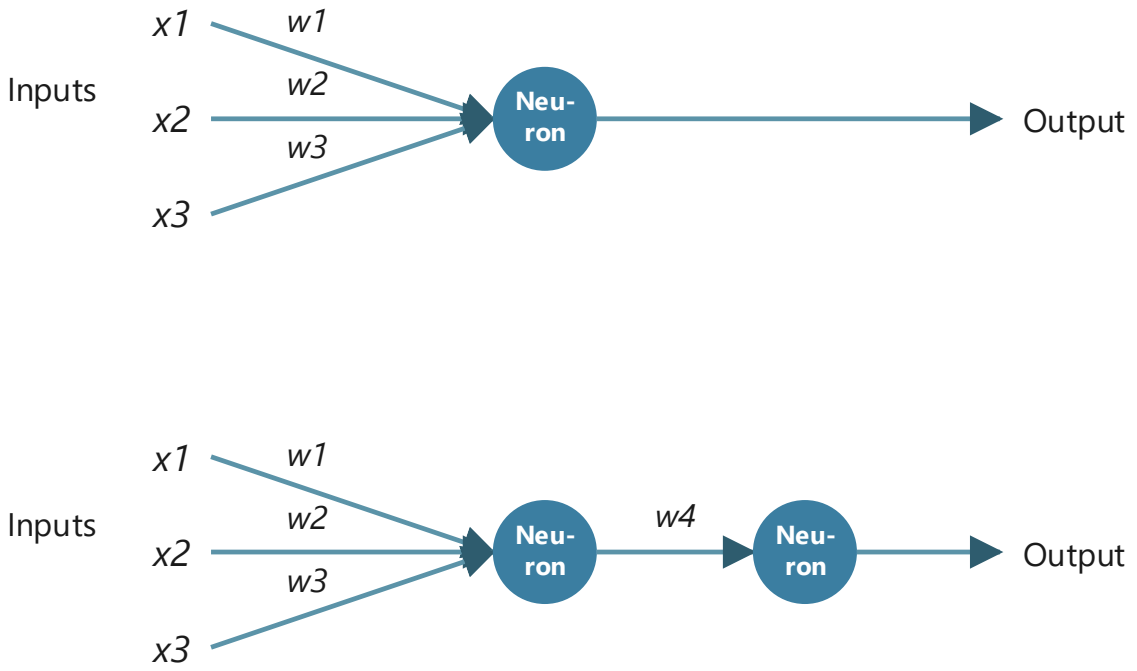
Artificial Neural Networks are inspired in Biological Neural Networks

Biological Neuron



Anatomy and physiology of animals Motor, Sunshineconnelly (en.wikibooks), CC BY 3.0, via Wikimedia Commons

Artificial Neuron



… but the mechanism is different

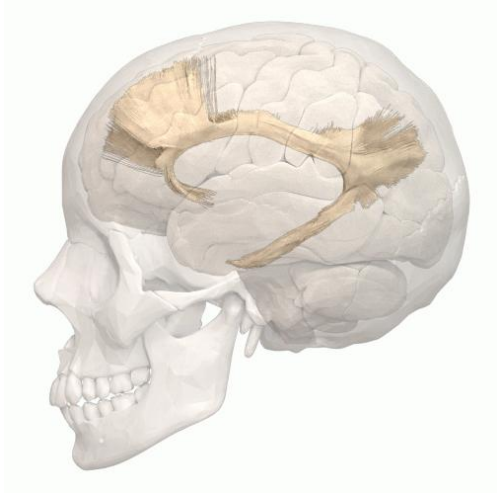




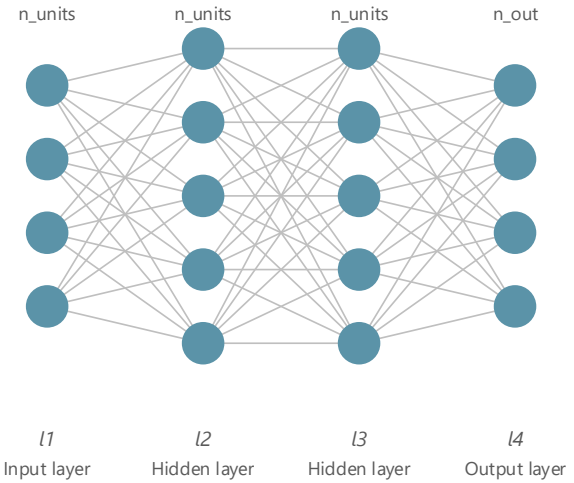
Image Source: Douglas DC-8 rear from down.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1563528
Image Source: Grey Heron taking flight in Paul de Arzila Natural Reserve, Portugal (52987496443).jpg, Carlos Eduardo Joos, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

Biological vs Artificial Neuron

Biological Neural Networks (BNNs)



Artificial Neural Networks (ANNs)

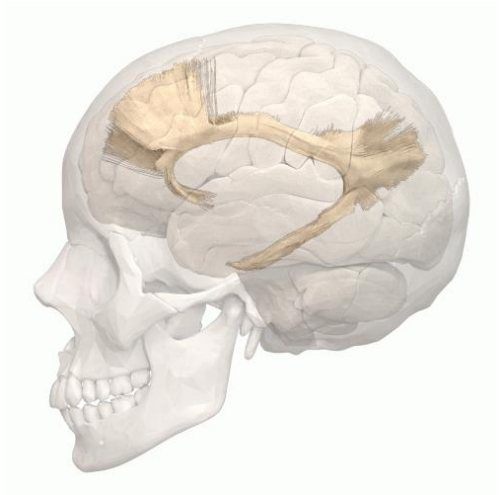


- **ANNs** have pre-defined and **fixed architectures** that can only **change the strength of connections** between neurons (weights). **BNNs** can do the former, but also **create new and reorganize existing connections** between neurons (neuroplasticity) and even create new neurons (neurogenesis).

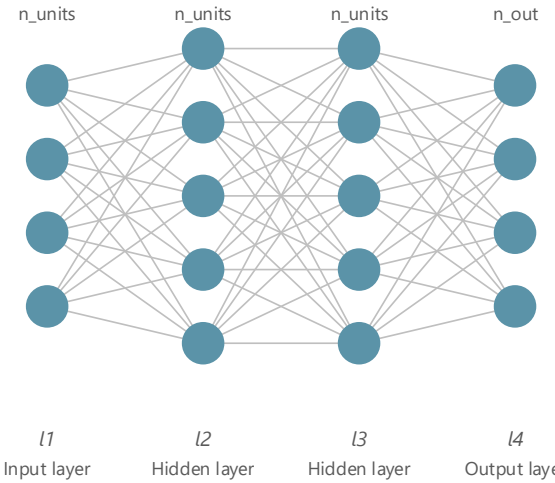
Image Source: Tractography - Cingulum - animation.gif, Tractography data: Yeh, F. C., Panesar, S., Fernandes, D., Meola, A., Yoshino, M., Fernandez-Miranda, J. C., ... & Verstynen, T. (2018). Population-averaged atlas of the macroscale human structural connectome and its network topology. NeuroImage, 178, 57-68. PubMed: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6921501/Skull and brain data: Nobutaka Mitsuhashi, Kaori Fujieda, Takuro Tamura, Shoko Kawamoto, Toshihisa Takagi, and Kousaku Okubo. (2009) BodyParts3D: 3D structure database for anatomical concepts. Nucleic Acids Research, Vol. 37, Database issue D782-D785, https://doi.org/10.1093/nar/gkn613, CC BY-SA 2.1 JP <https://creativecommons.org/licenses/by-sa/2.1/jp/deed.en>, via Wikimedia Commons

Biological vs Artificial Neuron

Biological Neural Networks (BNNs)



Artificial Neural Networks (ANNs)

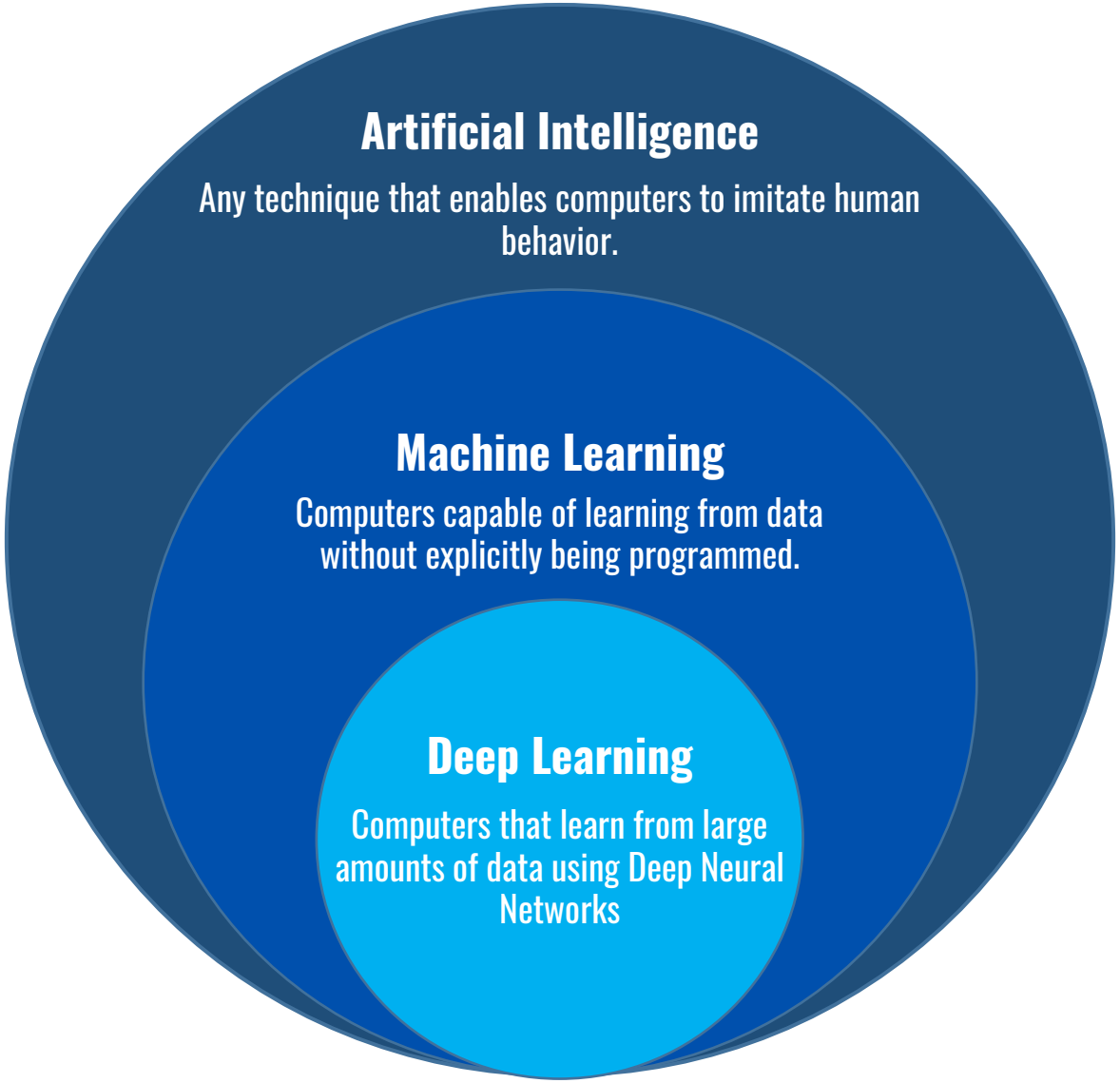


- **ANNs** use **continuous values** with **continuous activation functions** to propagate information throughout the network, whereas **BNNs** function as **firing systems that can be activated or not** (allowing or blocking information to pass).[1]
- In **BNNs**, neurons that **fire together**, **wire together**. The same **does not hold** for **ANNs**.[2]

1. There are computational models and algorithms inspired by the discrete firing behavior of biological neurons, such as spiking neural networks.
2. Hebbian learning

Image Source: Tractography - Cingulum - animation.gif, Tractography data: Yeh, F. C., Panesar, S., Fernandes, D., Meola, A., Yoshino, M., Fernandez-Miranda, J. C., ... & Verstynen, T. (2018). Population-averaged atlas of the macroscale human structural connectome and its network topology. NeuroImage, 178, 57-68. PubMed: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6921501/Skull and brain data: Nobutaka Mitsuhashi, Kaori Fujieda, Takuro Tamura, Shoko Kawamoto, Toshihisa Takagi, and Kousaku Okubo. (2009) BodyParts3D: 3D structure database for anatomical concepts. Nucleic Acids Research, Vol. 37, Database issue D782-D785, https://doi.org/10.1093/nar/gkn613, CC BY-SA 2.1 JP <https://creativecommons.org/licenses/by-sa/2.1/jp/deed.en>, via Wikimedia Commons

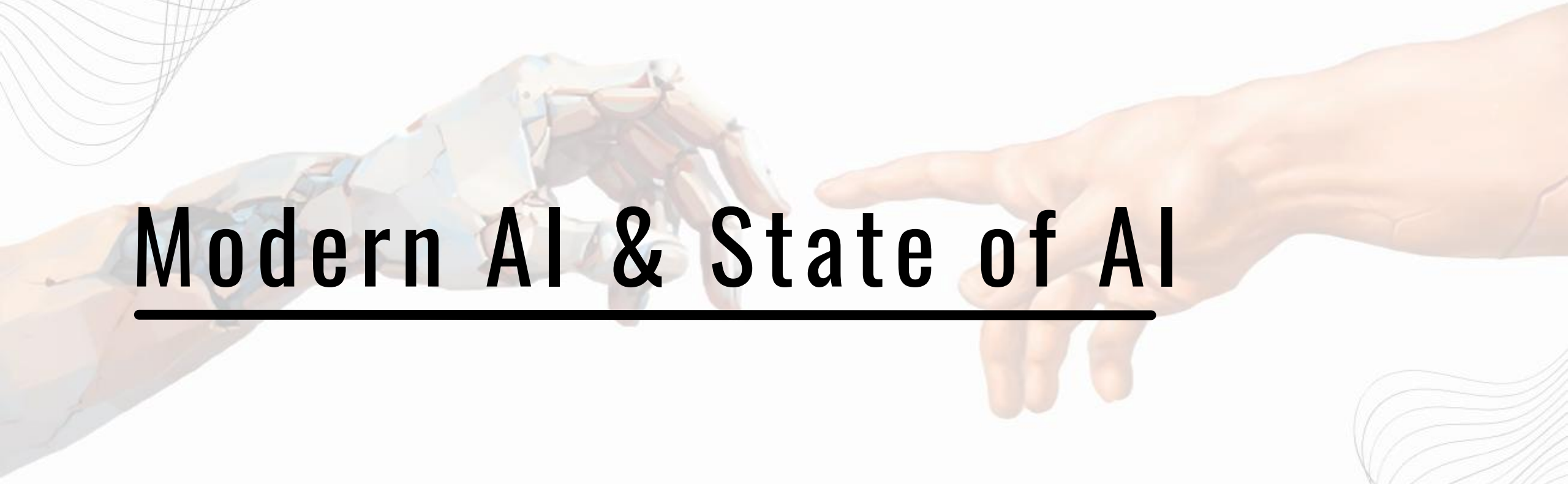
Overview


Artificial Intelligence
Any technique that enables computers to imitate human behavior.
Machine Learning
Computers capable of learning from data without explicitly being programmed.
Deep Learning
Computers that learn from large amounts of data using Deep Neural Networks

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 1



Modern AI & State of AI

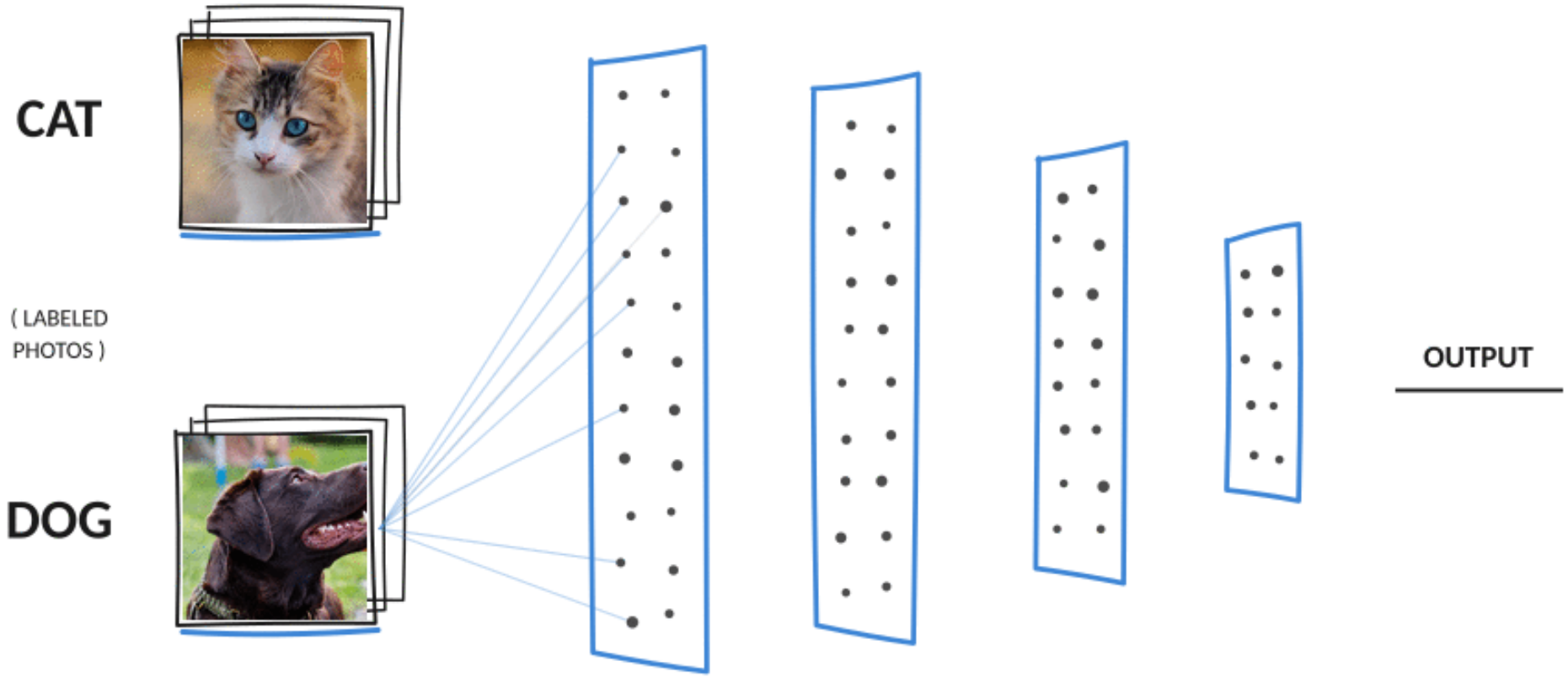
1 Course Details

2 Deep Learning Overview

3 Modern AI & State of AI

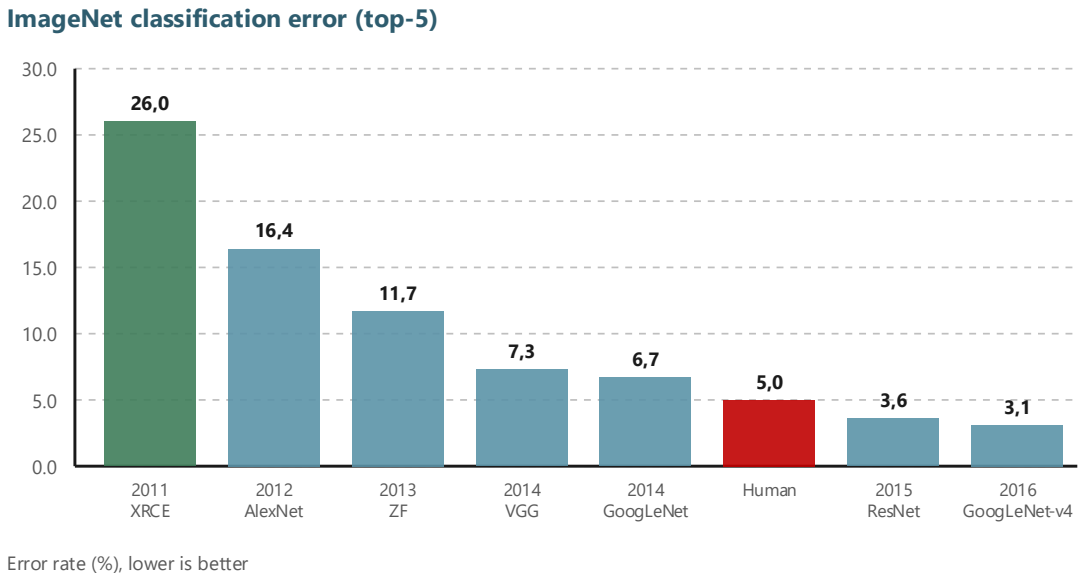
Advancements in AI (Deep Learning methods)

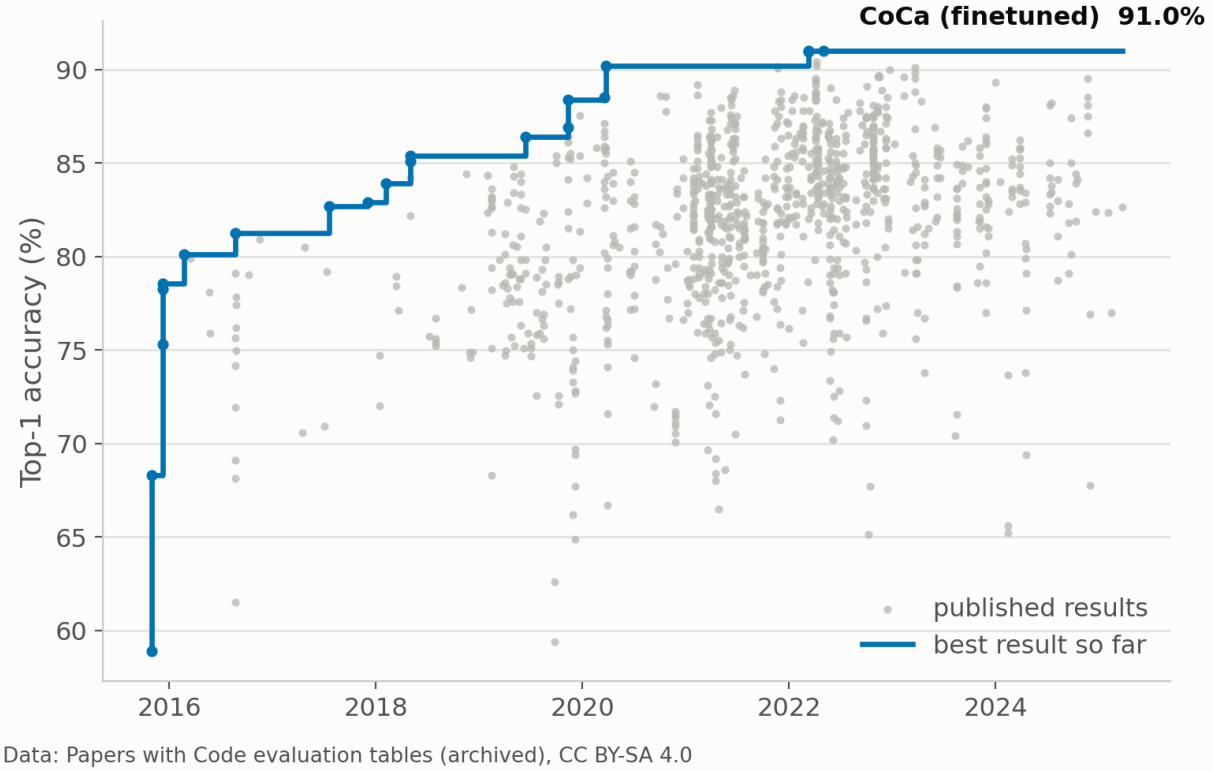
Classification



Advancements in AI (Deep Learning methods)

Classification





Advancements in AI (Deep Learning methods)

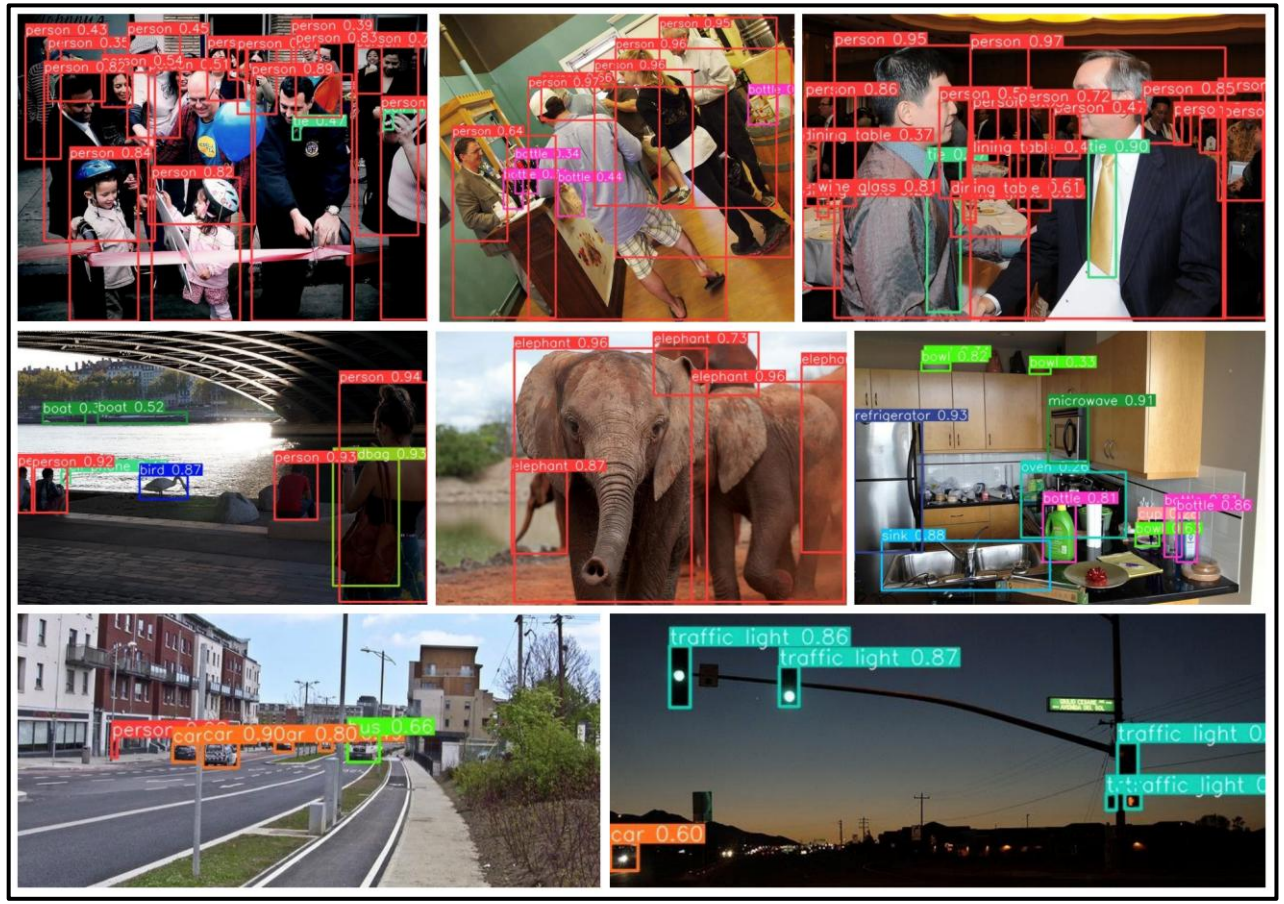
Object Detection

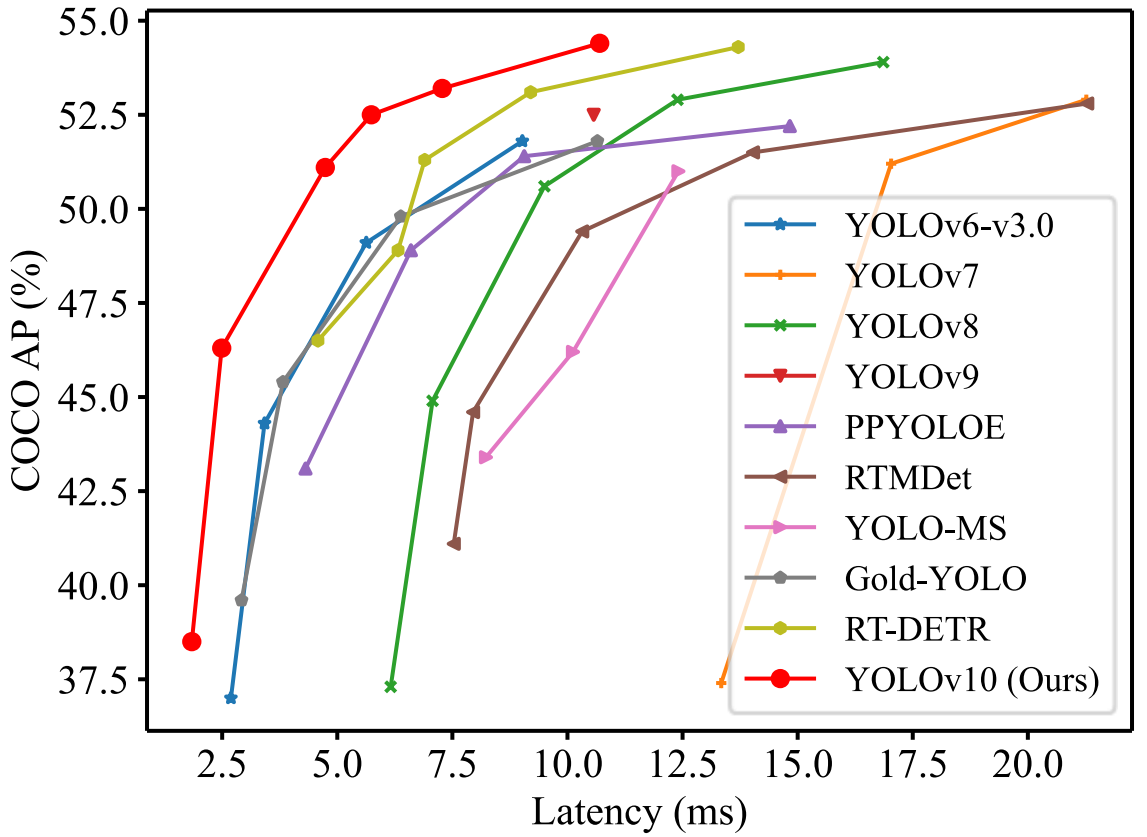




Advancements in AI (Deep Learning methods)

Object Detection





Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). YOLOv10: Real-Time End-to-End Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2405.14458, CC BY 4.0, via Wikimedia Commons

Advancements in AI (Deep Learning methods)

Image Segmentation



original



semantic segmentation – one colour per class

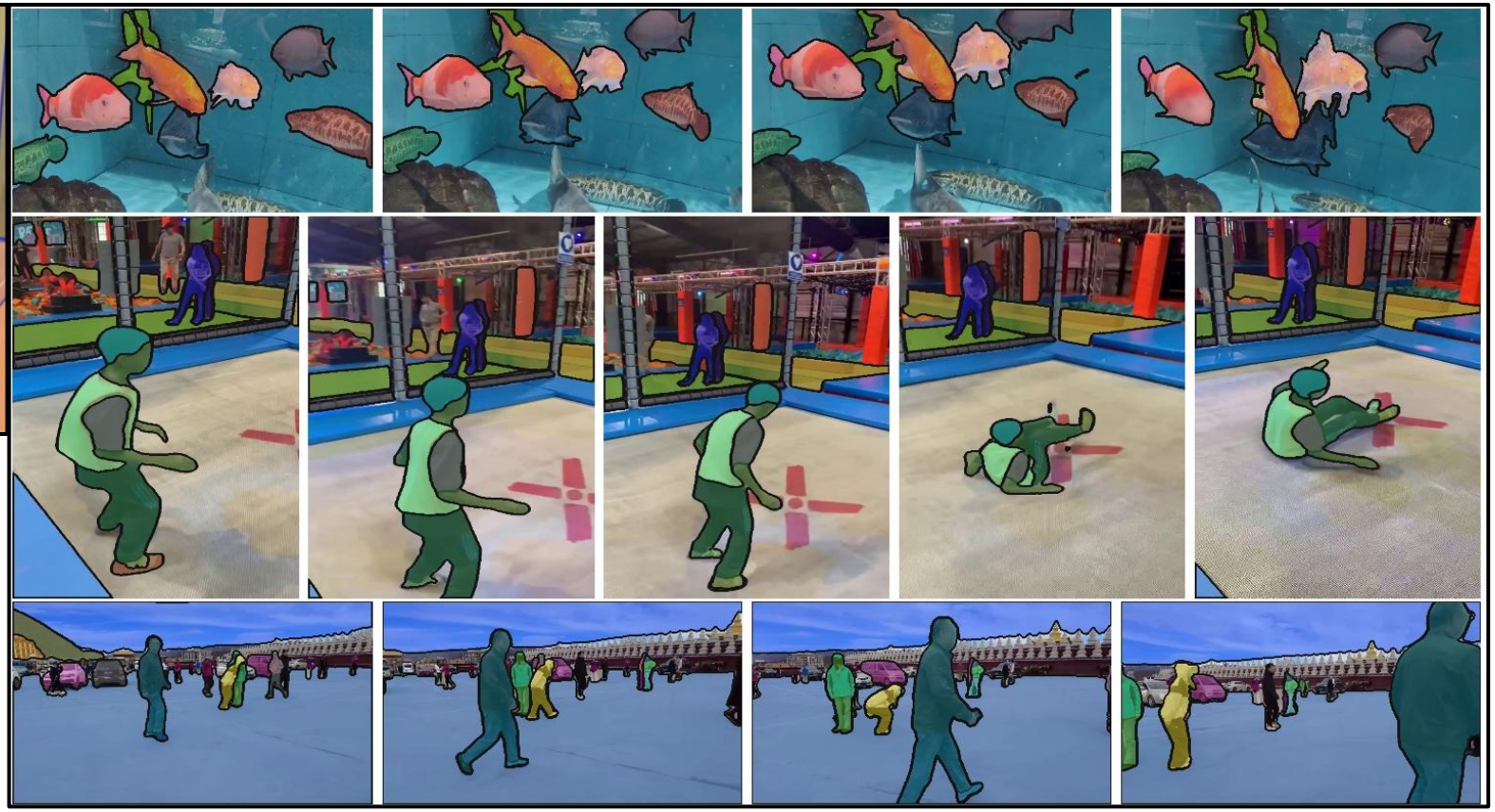


instance segmentation – one colour per object

Advancements in AI (Deep Learning methods)

Image Segmentation: Segment Anything 1 & 2 (SAM)





- SAM segments any type of object (even if it was not explicitly trained on that object)
- SAM 2 is built on top of SAM with the addition of memory for tracking objects overtime.

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., ... Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2408.00714 , CC BY 4.0, via Wikimedia Commons

Advancements in AI (Deep Learning methods)

Text-to-Image



Stable Diffusion 1.2 (2022)



FLUX.1-schnell (2024)

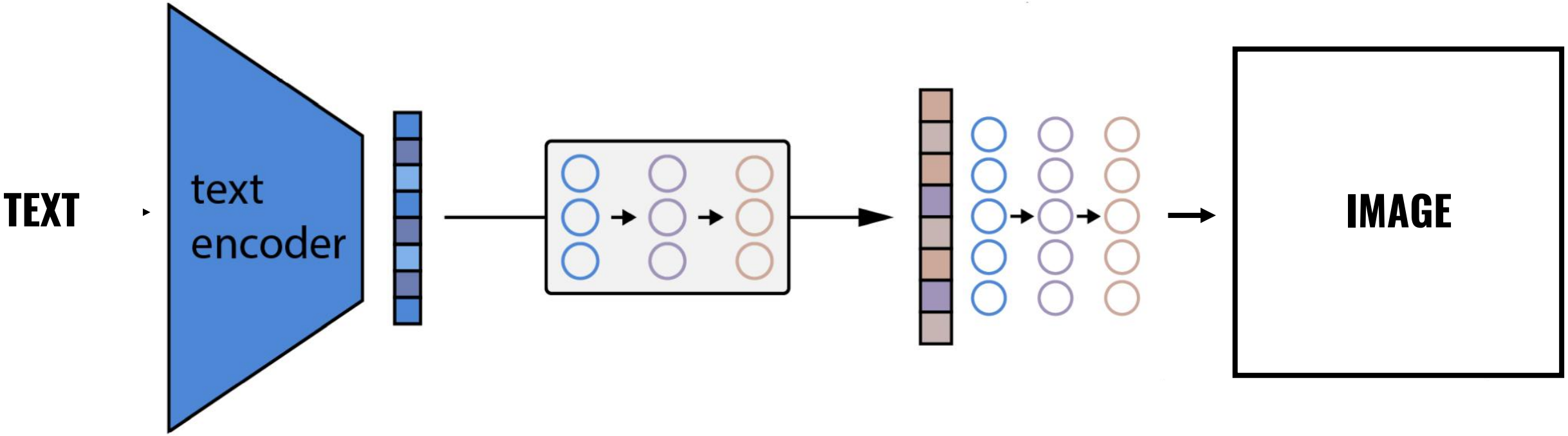


FLUX.1-dev-LoRa (2024)

Advancements in AI (Deep Learning methods)

Text-to-Image: DALL-E 2

https://openai.com/dall-e-2/



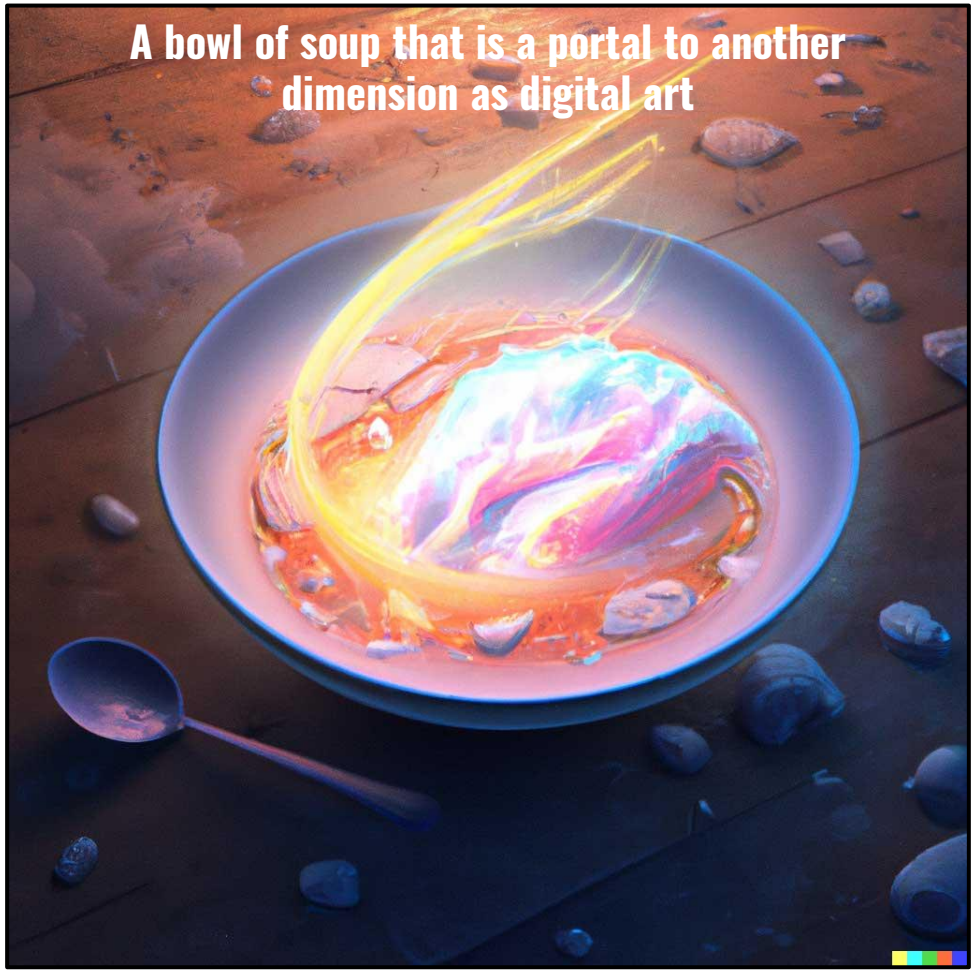
Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2204.06125 , CC BY 4.0, via Wikimedia Commons

Advancements in AI (Deep Learning methods)



A photorealistic image of an astronaut riding a horse" — generated with DALL·E 2, OpenAI (2022), https://openai.com/index/dall-e-2/

Advancements in AI (Deep Learning methods)



A bowl of soup that is a portal to another dimension as digital art" — generated with DALL·E 2, OpenAI (2022), https://openai.com/index/dall-e-2/

Advancements in AI (Deep Learning methods)

Text-to-Video: SORA

https://openai.com/index/sora/



Sora demonstration video — OpenAI, “A stylish woman walks down a Tokyo street…”, Sora (2024), https://openai.com/index/sora/

Modern AI categories

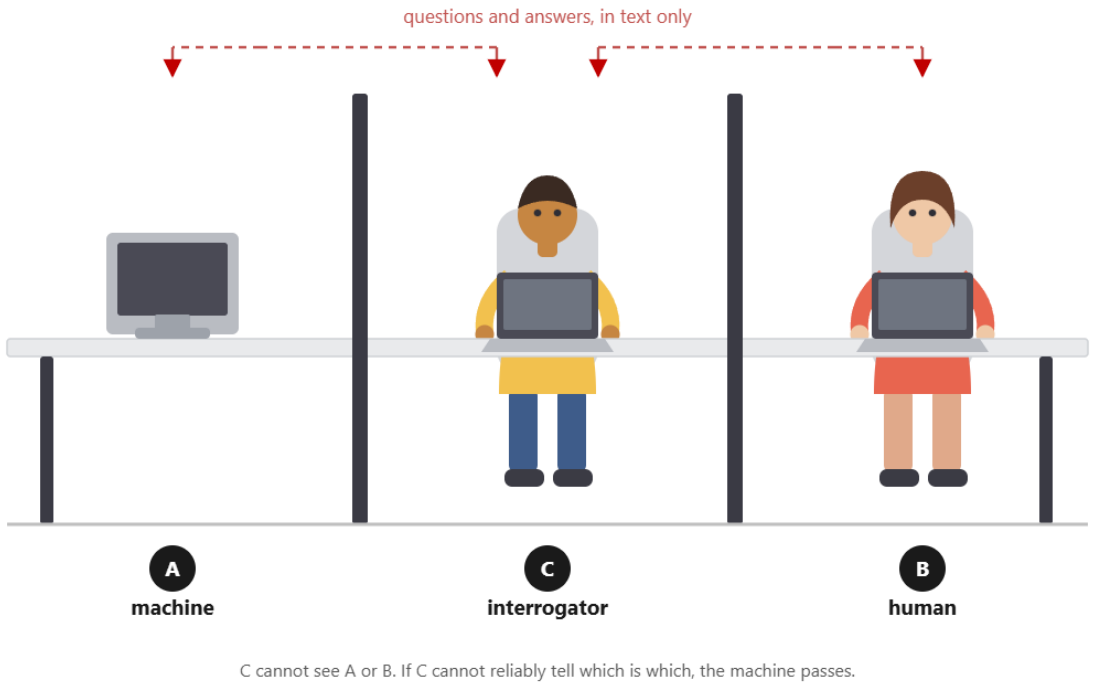
What is Artificial Intelligence?

Traditional version – the science and engineering of making machines that simulate human intelligence and improve themselves.

Modern AI categories

What is Artificial Intelligence?

The goal of traditional AI is to pass the Turing test.



- What happens if a computer (algorithm) passes the **Turing test**?
- Is it intelligent?
- Is it considered **Artificial General Intelligence** (AGI)?

Modern AI categories

“In this paper, we argue that it is critical for the AI research community to explicitly reflect on what we mean by "AGI," and aspire to quantify attributes like the performance, generality, and autonomy of AI systems” [5]

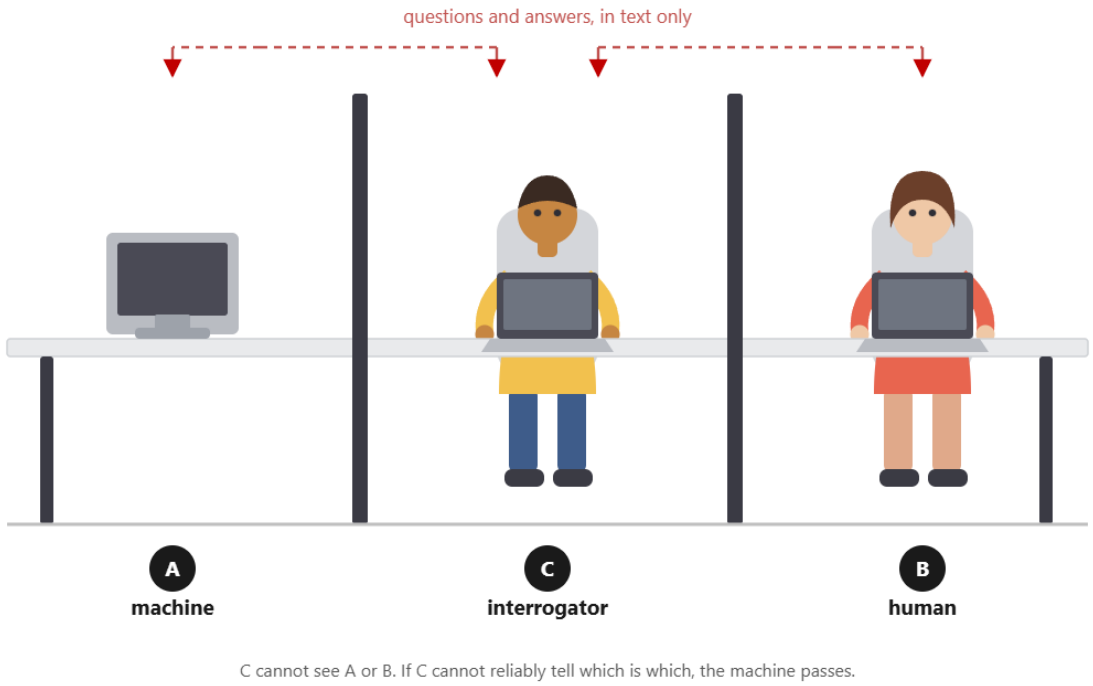
About Turing Test: “in practice, the test often highlights the ease of fooling people (Weizenbaum, 1966; Wikipedia, 2023a) rather than the “intelligence” of the machine. Given that modern LLMs pass some framings of the Turing Test, it seems clear that this criteria is insufficient for operationalizing or benchmarking AGI”

Morris, M.R., Sohl-Dickstein, J., Fiedel, N., Warkentin, T., Dafoe, A., Faust, A., Farabet, C. & Legg, S.. (2024). Position: Levels of AGI for Operationalizing Progress on the Path to AGI. Proceedings of the 41st International Conference on Machine Learning, in Proceedings of Machine Learning Research 235:36308-36321 Available from https://proceedings.mlr.press/v235/morris24b.html.

Modern AI categories

What is Artificial Intelligence?

The goal of traditional AI is to pass the Turing test.



There are many claims that ChatGPT / GPT-4 has passed the Turing test (not necessarily true)

- **Is ChatGPT intelligent?**
- **Has ChatGPT achieved AGI?**

Modern AI categories

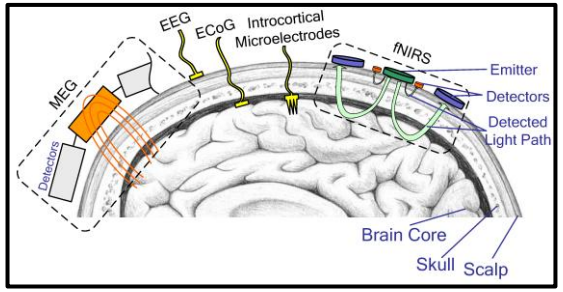
~~AGI remains a distant goal~~, so let us now focus on the current state of AI

AI has changed and is being applied in new ways.

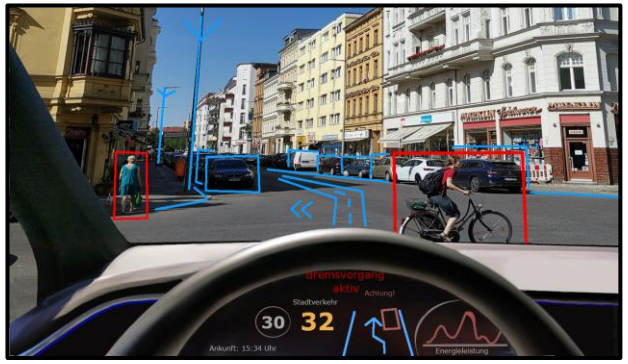


Futuristic AI

A technology that extends human capabilities and aids people in solving real world problems.



Brain Tumor detection



Autonomous driving



Prosthetics



Brain Computer Interface

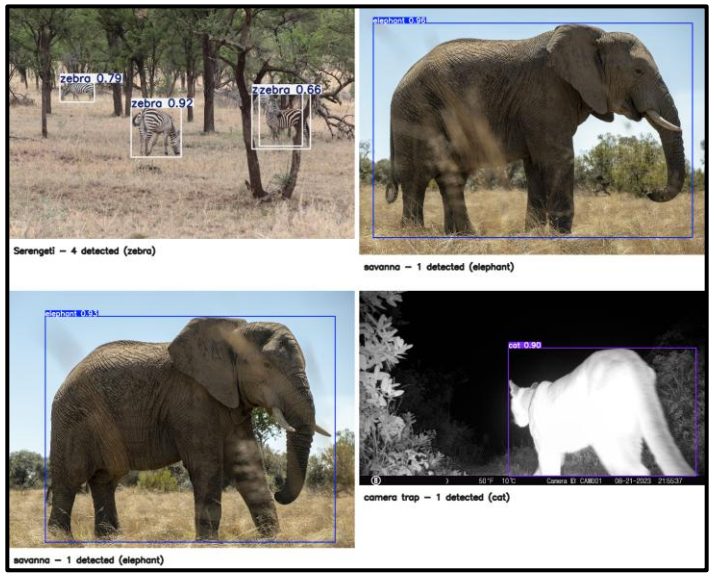
… makes our lives better

… enhances humans

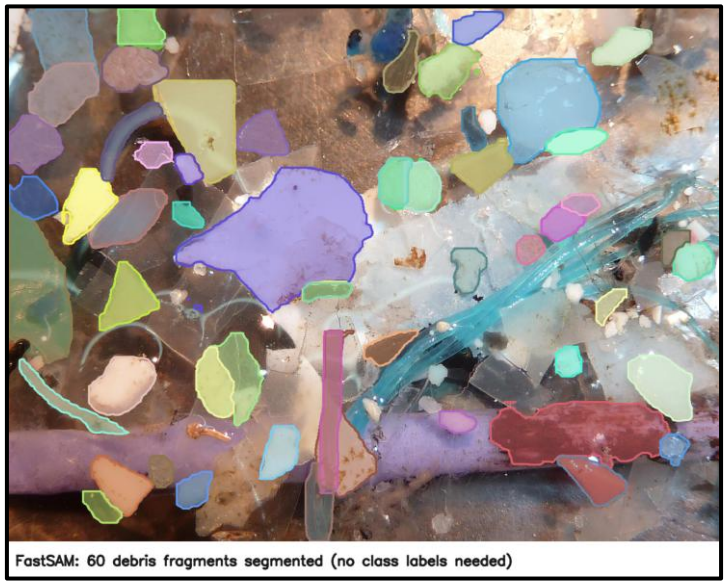
Image Source: BCI sensor mounting types — invasive (IM), semi-invasive (ECoG), and non-invasive (MEG, EEG, fNIRS).png, Peksa Janis and Dmytro Mamchur, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
Image Source: Autonomous-driving-Barcelona.jpg, Eschenzweig, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Myoelectric prosthetic arm.jpg, Petty Officer 2nd Class Greg Mitchell of the United States Navy, Public domain, via Wikimedia Commons
Image Source: BrainGate.jpg, PaulWicks at English Wikipedia, Public domain, via Wikimedia Commons

Futuristic AI

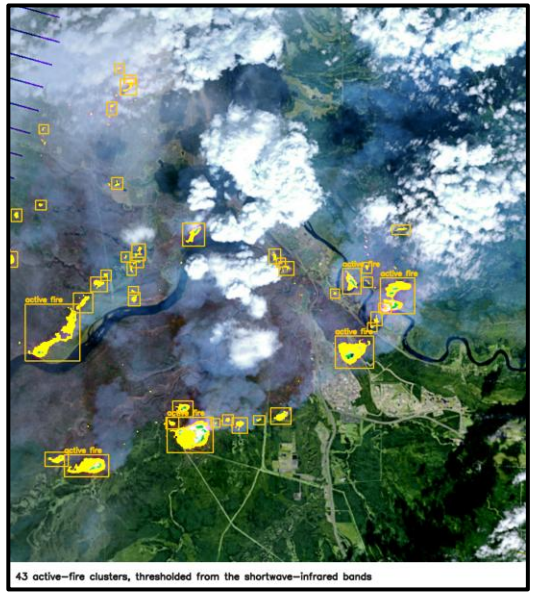
A technology that extends human capabilities and aids people in solving real world problems.



Monitoring wildlife



Monitoring ocean



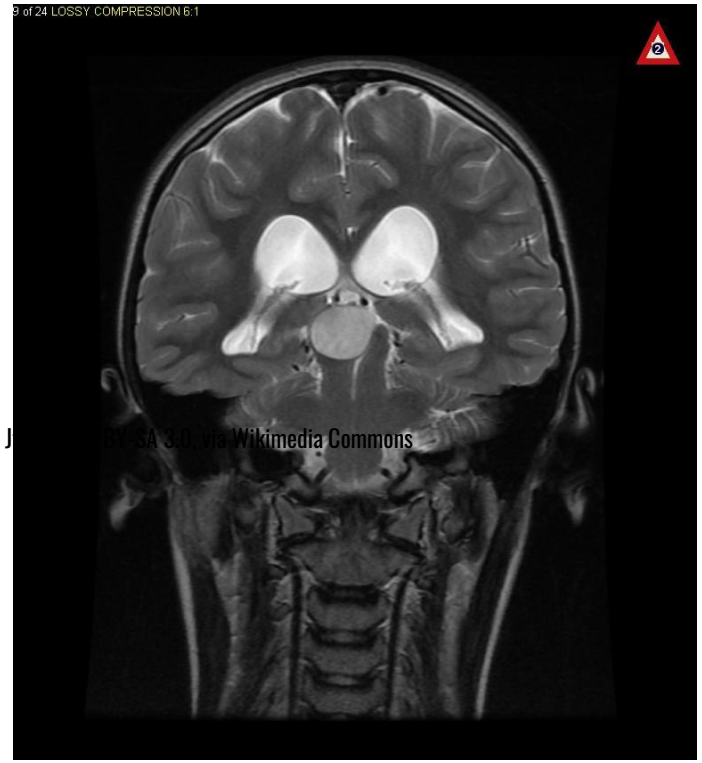
Monitoring forests

... saves the world

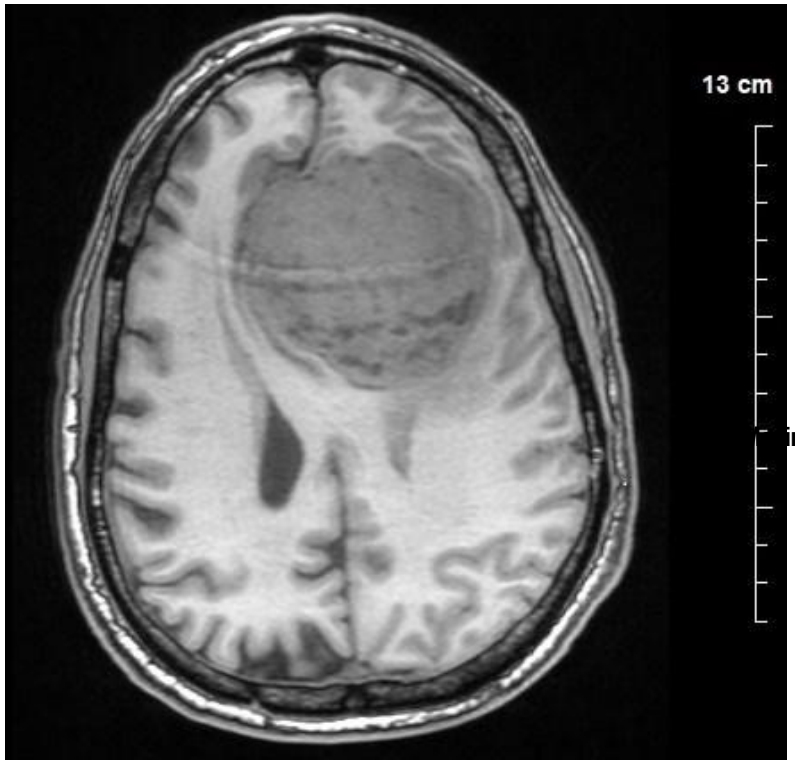
Futuristic AI (Examples)

Brain tumour detection

https://huggingface.co/spaces/sindhoorar/brain-tumor-classifier



T2-weighted coronal MRI tectal glioma, J SA 3.0, via Wikimedia Commons



ingioma, Fowle, CC BY-SA 4.0, via Wikimedia

Image Source: T2-weighted coronal MRI tectal glioma.jpg, Jto410, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
Image Source: Huge Meningioma.jpg, Fowle, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Futuristic AI (Examples)

The most advanced Medical AI system: **Med-PaLM 2** (Google)
https://sites.research.google/med-palm/

For multiple modalities:
Med-PaLM M (Google)



Instructions: You are a helpful radiology assistant. Describe what lines, tubes, and devices are present and each of their locations. Describe if pneumothorax is present; if present, describe size on each side. Describe if pleural effusion is present; if present, describe amount on each side. Describe if lung opacity (atelectasis, fibrosis, consolidation, infiltrate, lung mass, pneumonia, pulmonary edema) is present; if present, describe kinds and locations. Describe the cardiac silhouette size. Describe the width and contours of the mediastinum. Describe if hilar enlargement is present; if enlarged, describe side. Describe what fractures or other skeletal abnormalities are present.

Given the LATERAL view X-ray image <img>. **Reason for the study:** Amiodarone routine surveillance.
Q: Describe the findings in the image following the instructions.
A: As compared to the previous radiograph there is no relevant change. Normal lung volumes. Mild bilateral apical scarring. Normal size of the cardiac silhouette and tortuosity of the thoracic aorta. No pathologic findings in the lung parenchyma notably no evidence of fibrotic lung parenchymal changes. A faint 2 mm rounded opacity projecting over the lower aspect of the fourth right rib and internally to the upper border of the second right rib is seen on the frontal radiograph only and likely reflects structure on the skin.

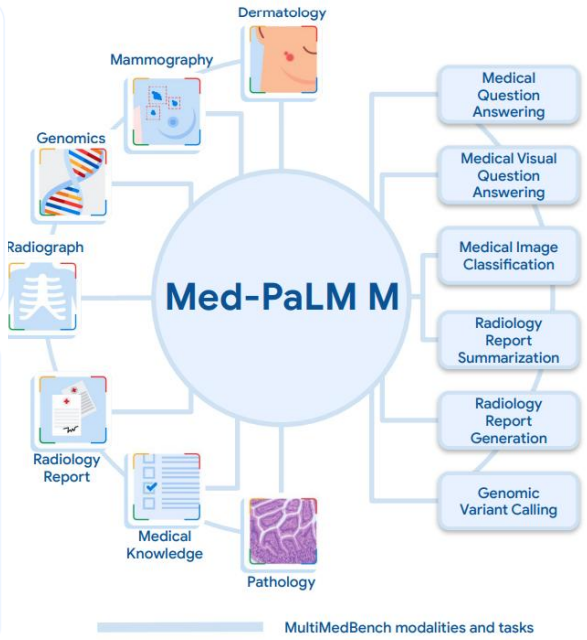
Given the PA view X-ray image <img>. **Reason for the study:** History m with malaise pneumonia.
Q: Describe the findings in the image following the instructions.
A:



Instructions: You are a helpful dermatology assistant. The following are questions about skin lesions. Categorize the skin lesions into the most likely class given the patient history.

Given <img>. **Patient History:** Age: 51, Gender: female, Smoke: false, Drink: false, Family skin cancer history: true, Family any cancer history: false, Lesion region: back, Lesion itch: false, Lesion grew: false, Lesion bled: false, Lesion elevation: false, Fitzpatrick scale: 1.0, Diameters (mm): [12.0, 8.0].
Q: Which of the following is the most likely diagnosis of the patient's skin lesion?
(A) Nevus (B) Basal Cell Carcinoma (C) Squamous Cell Carcinoma (D) Actinic Keratosis (E) Seborrheic Keratosis (F) Melanoma
A: Basal Cell Carcinoma.

Given <img>. **Patient History:** Age: 39, Gender: unknown, Smoke: unknown, Drink: unknown, Family skin cancer history: unknown, Family any cancer history: unknown, Lesion region: neck, Lesion itch: false, Lesion grew: true, Lesion bled: false, Lesion elevation: true, Fitzpatrick scale: unknown, Diameters (mm): [unknown, unknown].
Q: Which of the following is the most likely diagnosis of the patient's skin lesion?
(A) Nevus (B) Basal Cell Carcinoma (C) Squamous Cell Carcinoma (D) Actinic Keratosis (E) Seborrheic Keratosis (F) Melanoma
A:



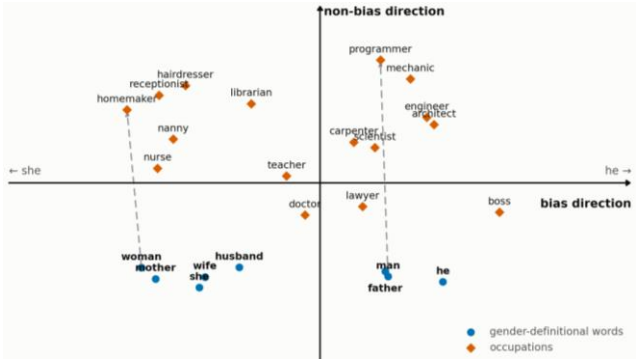


Tu, T., Azizi, S., Driess, D., Schaekermann, M., Amin, M., Chang, P.-C., Carroll, A., Lau, C., Tanno, R., Ktena, I., Mustafa, B., Chowdhery, A., Liu, Y., Kornblith, S., Fleet, D., Mansfield, P., Prakash, S., Wong, R., Virmani, S., ... Natarajan, V. (2023). Towards Generalist Biomedical AI. In *arXiv [cs.CL]*. arXiv. https://doi.org/10.48550/arXiv.2307.14334
Image Source: Figure 2 — Tu et al., *Towards Generalist Biomedical AI* (2023), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2307.14334
Image Source: Figure 1 — Tu et al., *Towards Generalist Biomedical AI* (2023), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2307.14334

Skeptical AI

Any AI system that is biased, unsafe, insecure and is used as a resource for inequality and war.





biased ...





... manipulates information

Image Source: Male and female deepfake.jpg, Vahan03, Public domain, via Wikimedia Commons
Image Source: Pope Francis in puffy winter jacket.jpg, Midjourney, Public domain, via Wikimedia Commons
Image Source: Deepfake.jpg, Ministerie van Buitenlandse Zaken, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

Skeptical AI (Examples)

Diffusion Bias Explorer
(https://huggingface.co/spaces/society-ethics/DiffusionBiasExplorer)
A method that shows the bias in text-to-image methods

Ambitious CEO
Ambitious Receptionist
Ambitious Lawyer
Sensitive/Emotional Lawyer
Scientist
Teacher
...
Why?



Image generated with Stability AI SD-Turbo; prompts: “ambitious CEO” and “ambitious receptionist”; seeds 1000–1008; composite figure by the authors

Skeptical AI (Examples)

The previous example was simple, but it can get messy:
https://futurism.com/the-byte/lawyer-chatgpt-court



Image Source: Courtroom One Gavel - Flickr - Joe Gratz.jpg, Joe Gratz, CCO, via Wikimedia Commons

Skeptical AI (Examples)

DeepFakes:

- Misinformation - https://www.youtube.com/watch?v=cQ54GDm1eL0&t=13s
- Not always skeptical: Entertainment/Tribute - https://www.youtube.com/watch?v=uAPUkgeiFVY



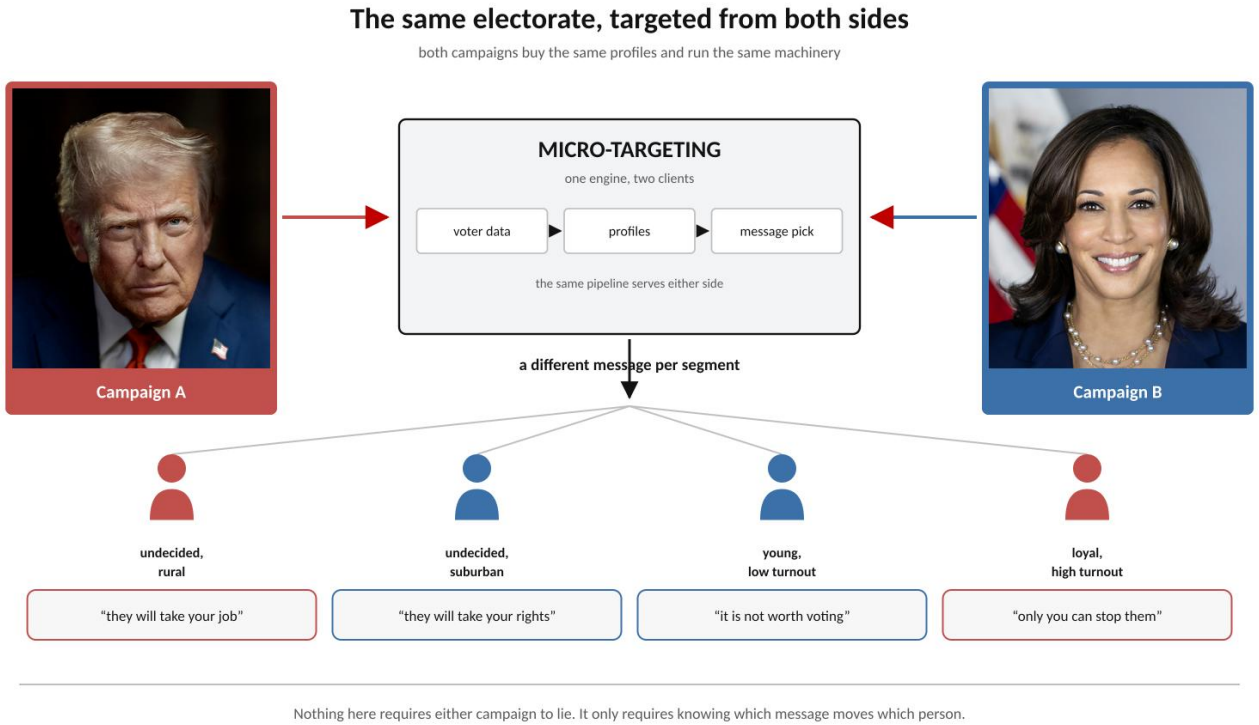


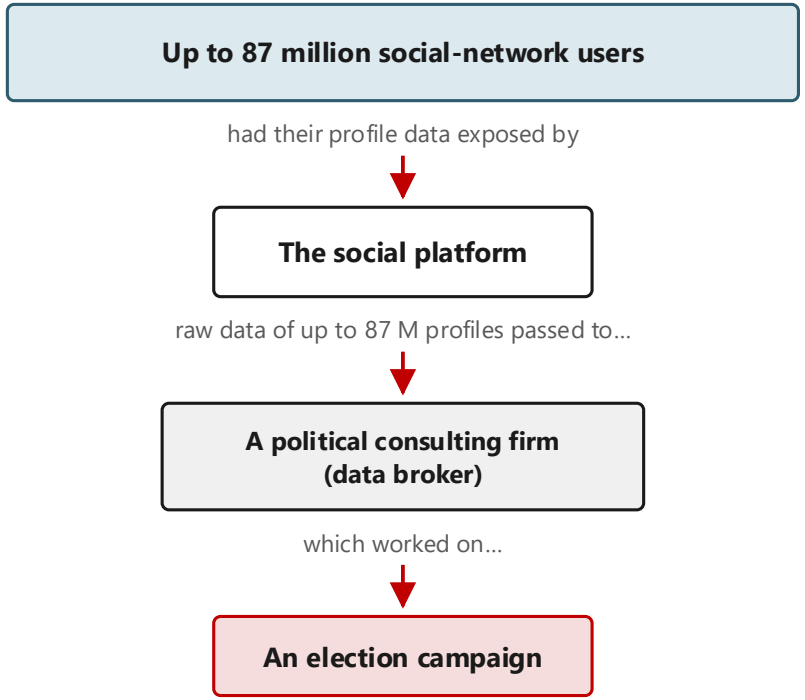
Trump Gaza, Arcana, prompted by Ariel Vromen, Public domain, via Wikimedia Commons

Image Source: Dictators - Kim Jong-Un by RepresentUs.webm, RepresentUs, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons
Image Source: Frame from - Trump Gaza.webm, Arcana, prompted by Ariel Vromen (Q15957667) and Solo Avital (Q1416127), Public domain, via Wikimedia Commons

Political AI

A tool for controlling people's decision making towards the government's benefit.



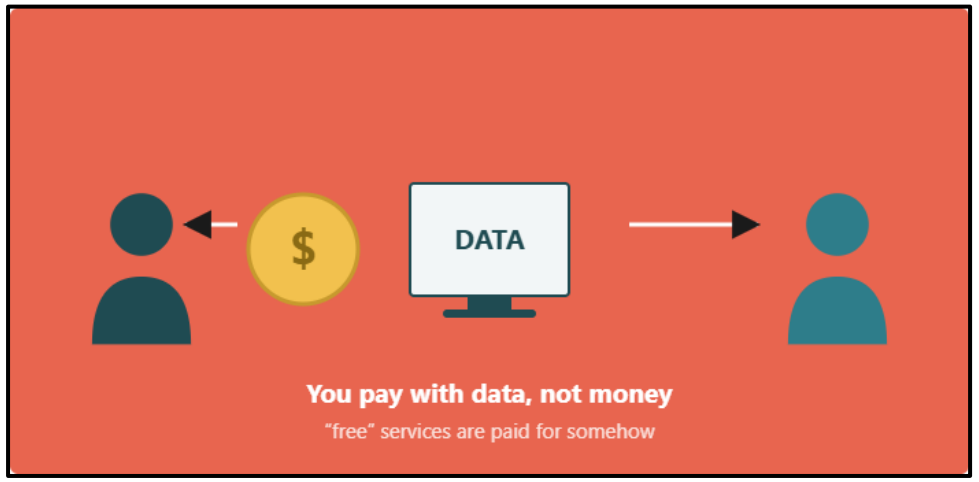


... take advantage of our disadvantages

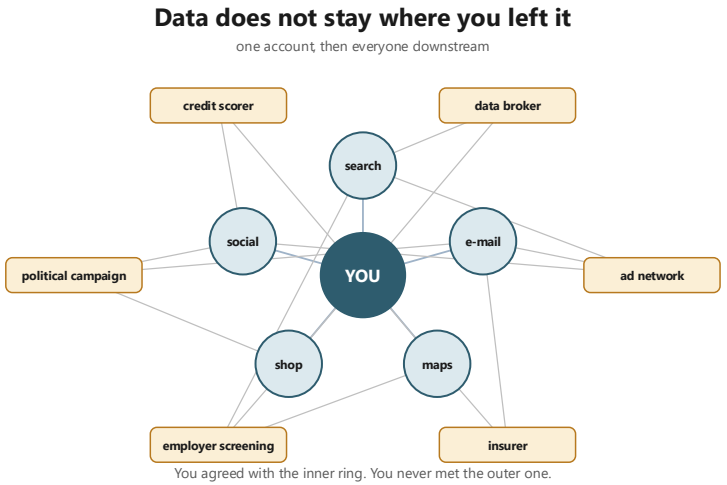
Image Source: Official Presidential Portrait of President Donald J. Trump (2025).jpg, Daniel Torok, Public domain, via Wikimedia Commons
Image Source: Kamala Harris Vice Presidential Portrait.jpg, Lawrence Jackson, Public domain, via Wikimedia Commons

Enterprise AI

A data-driven tool to maximize profit.



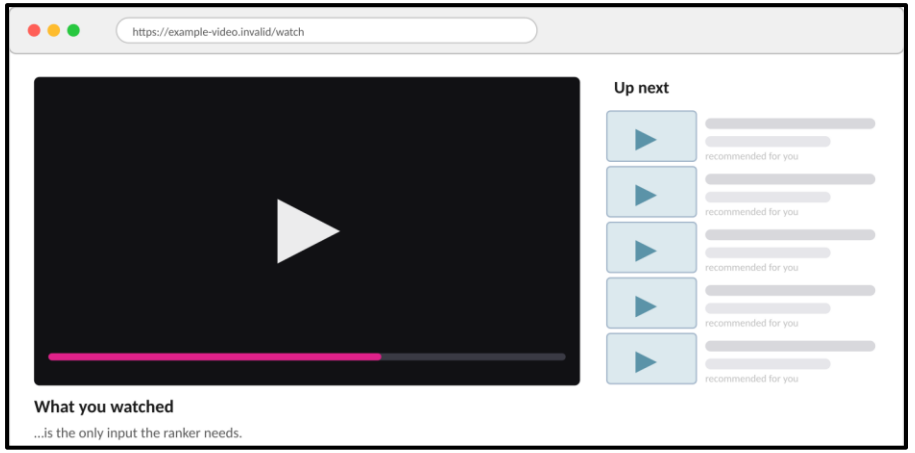
... data is everything (no such thing as free service)

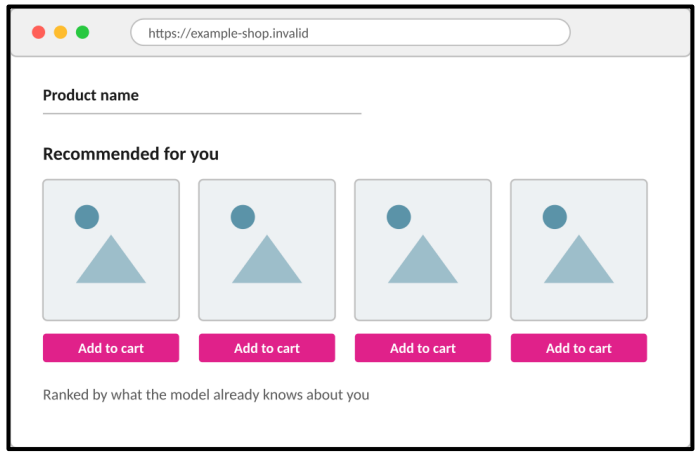




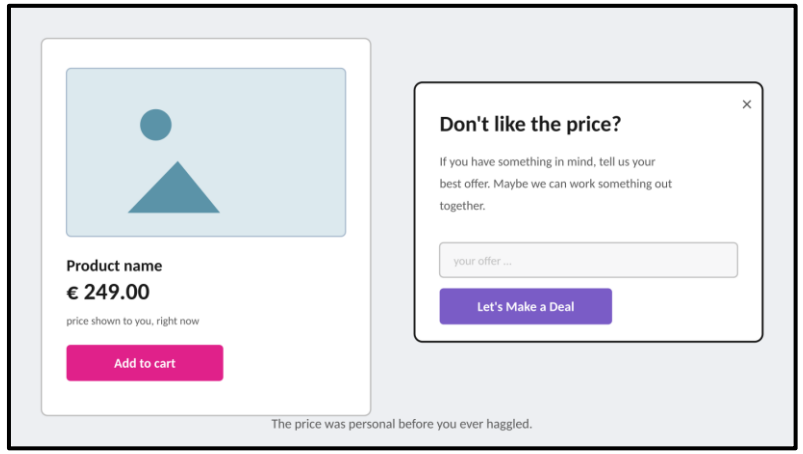
Enterprise AI

A data-driven tool to maximize profit.





The more they know about you, the more they profit...



Enterprise AI (Examples)

Data needed for training an AI model
- Can sell trained model as a service

Data produced by trained model
- Can use produced data to improve model
- Can sell produced data

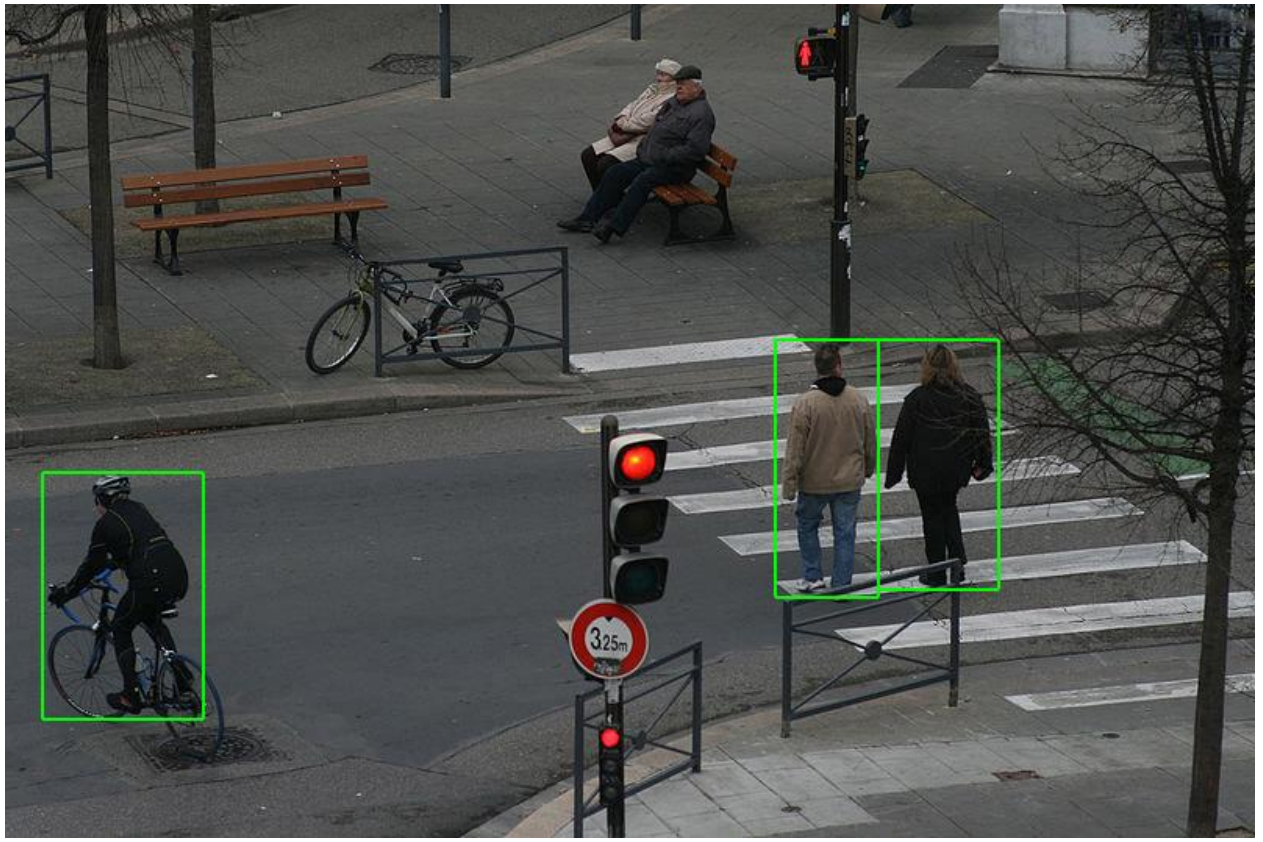
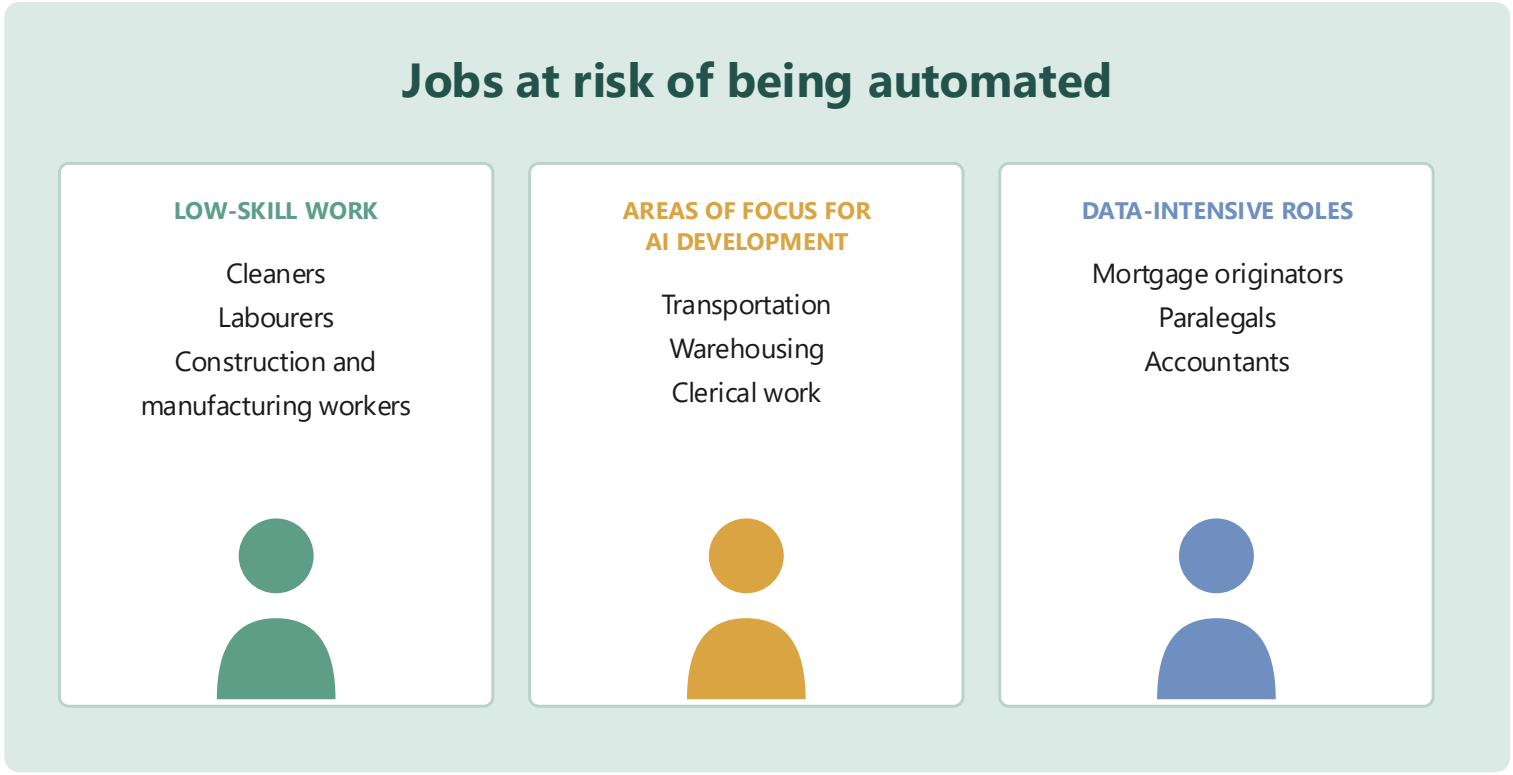


Image Source: Détection de personne - exemple 2.jpg, Exemple_d'infractions.jpg: IlOzzderivative work: Indif, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

Enterprise AI (Examples)

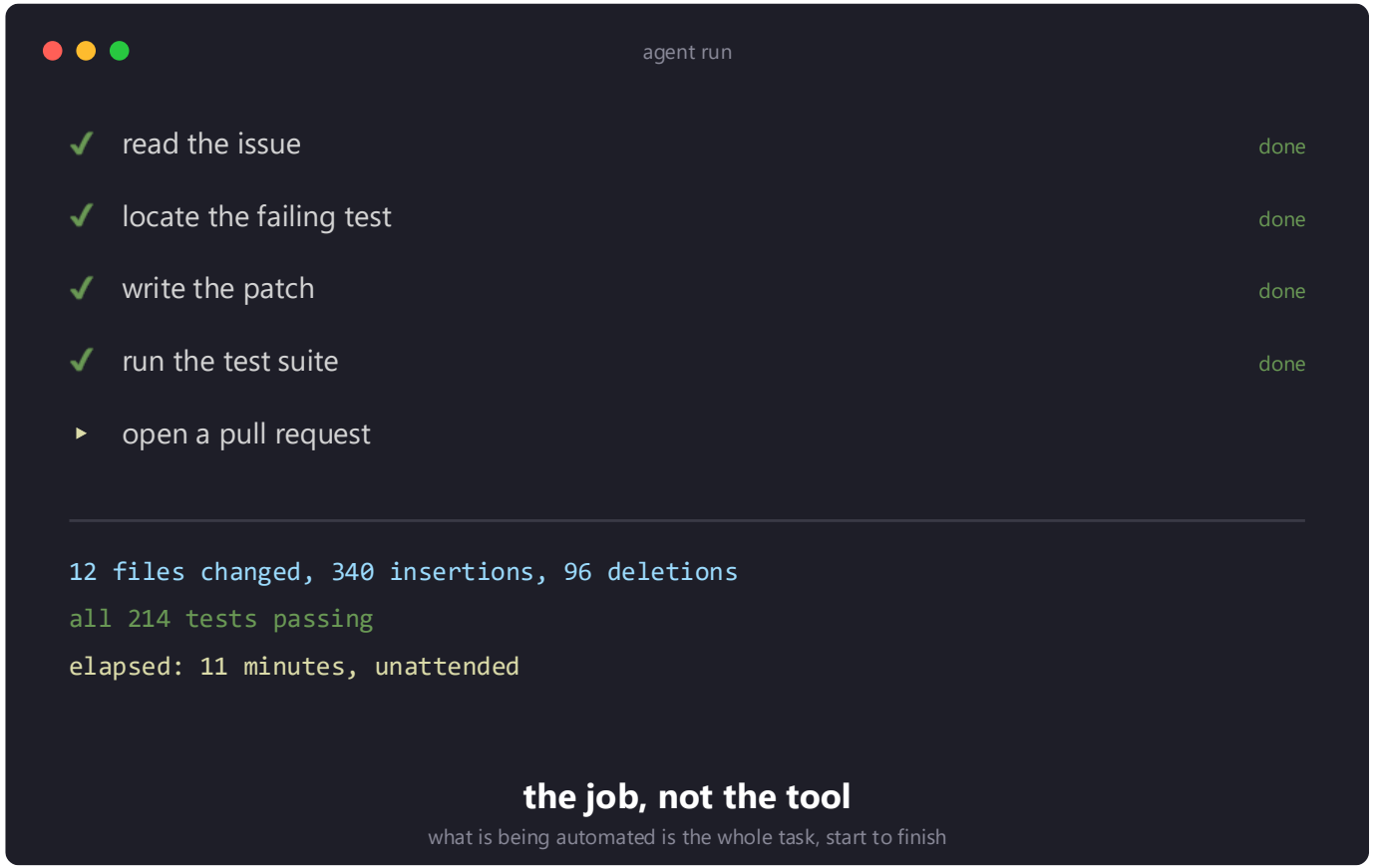
A data-driven tool to maximize profit.



Job automation

Enterprise AI (Examples)

Cognition labs **Devin**



Job automation

Summary

FUTURISTIC AI: The goal is to extend human life.



SKEPTICAL AI: The goal is to have objectives that are misaligned with human objectives.



POLITICAL AI: The goal is to control humans.



ENTERPRISE AI: The goal is to maximize profit.



Image Source: WBAMC first in DoD to use robot for surgery 160426-A-EK666-506.jpg, Marcy Sanchez, Public domain, via Wikimedia Commons
Image Source: MQ-9 Reaper in flight (2007).jpg, U.S. Air Force photo/Staff Sgt. Brian Ferguson, Public domain, via Wikimedia Commons
Image Source: PloenMarionetten 1.jpg, Jürgen Howaldt, CC BY-SA 2.0 DE <https://creativecommons.org/licenses/by-sa/2.0/de/deed.en>, via Wikimedia Commons
Image Source: All Euro banknotes.png, Bericht, CCO, via Wikimedia Commons

Ethical Considerations

- This course also emphasizes ethical considerations in AI development, ensuring that students not only learn the technical aspects of **Deep Learning** but also understand its broader impact on society.

More Speech and Fewer Mistakes (07.01.2025)



Meta Just Changed EVERYTHING About Free Speech:

https://www.youtube.com/watch?v=eUJLmSMIi2U&t=1s

Image Source: Mark Zuckerberg F8 2019 Keynote (32830578717) (cropped).jpg, Anthony Quintano from Westminster, United States, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

Highlights

- Replace fact checkers with **Community Notes**
  - They are too politically biased
  - Destroyed more trust than they have created

- Filters that scan policy violations
  - Made a lot of mistakes and took down content that is shouldn't have

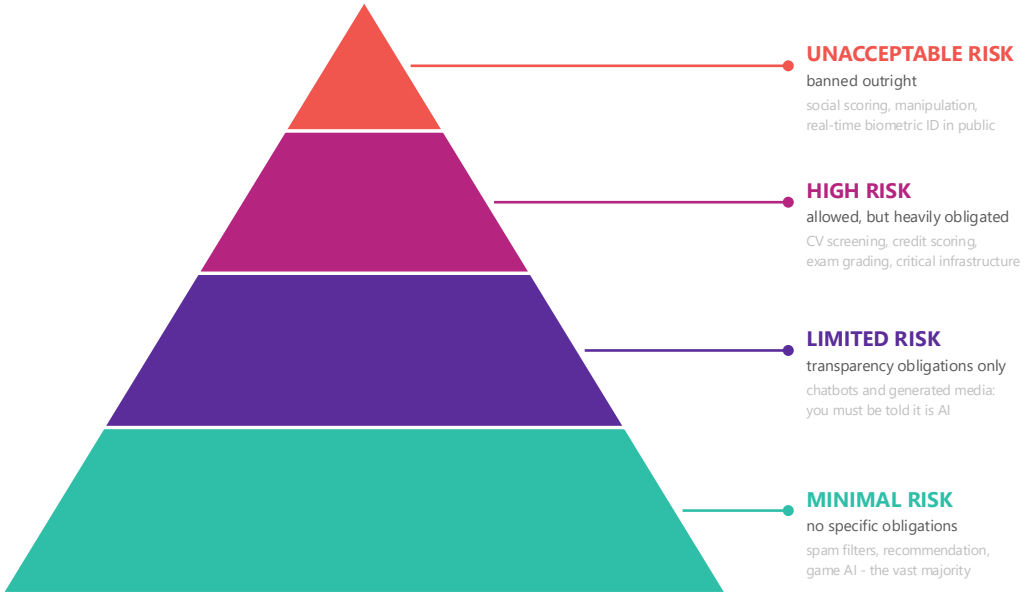
- Europe has an ever-increasing number of laws institutionalizing censorship and making it difficult to build anything innovative

EU AI Act

"The AI Act … is the first-ever comprehensive legal framework on AI worldwide. The aim of the rules is to foster trustworthy AI in Europe"

Why do we need rules on AI?

- "While most AI systems pose limited to no risk […] certain AI systems create risks that we must address"
- "For example, it is often not possible to find out why an AI system has made a decision or prediction and taken a particular action. …"



EU AI Act: obligations scale with risk

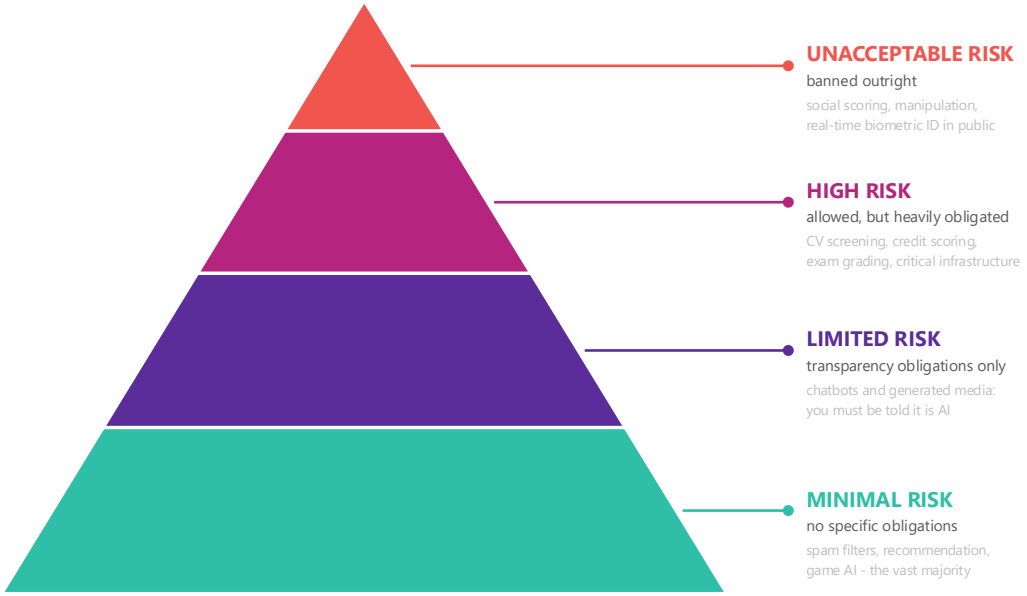
more risk, higher up, fewer systems, stricter rules

AI Act. (n.d.). Shaping Europe's Digital Future. Retrieved May 7, 2026, from https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

EU AI Act

Unacceptable risk

1. **harmful AI-based manipulation and deception**
2. harmful AI-based exploitation of vulnerabilities
3. social scoring
4. **Individual criminal offence risk assessment or prediction**
5. untargeted scraping of the internet or CCTV material to create or expand facial recognition databases
6. **emotion recognition in workplaces and education institutions**
7. biometric categorisation to deduce certain protected characteristics
8. real-time remote biometric identification for law enforcement purposes in publicly accessible spaces



EU AI Act: obligations scale with risk

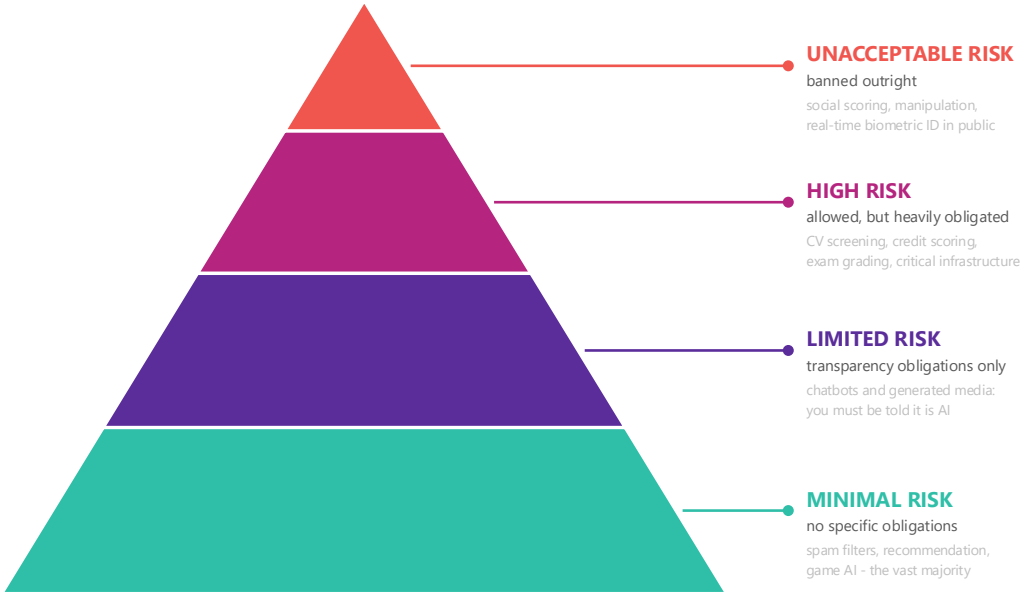
more risk, higher up, fewer systems, stricter rules

AI Act. (n.d.). Shaping Europe's Digital Future. Retrieved May 7, 2026, from https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

EU AI Act

High risk

- AI safety components in critical infrastructures (e.g. **transport**), the failure of which could put the life and health of citizens at risk
- AI solutions used in education institutions, that may determine the access to education and course of someone's professional life (e.g. **scoring of exams**)
- AI tools for employment, management of workers and access to self-employment (e.g. **CV-sorting software for recruitment**)
- Certain AI use-cases utilised to give access to essential private and public services (e.g. **credit scoring denying citizens opportunity to obtain a loan**)
- And many more ...



EU AI Act: obligations scale with risk

more risk, higher up, fewer systems, stricter rules

AI Act. (n.d.). Shaping Europe's Digital Future. Retrieved May 7, 2026, from https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

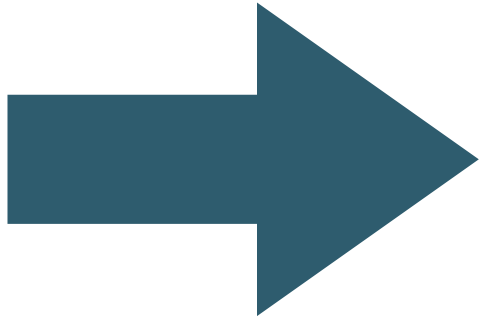
Summary

In this first lecture we set up the course and placed deep learning in its wider context:

- **Course details:** prerequisites, plus a theoretical and a practical track
- **AI, ML and DL:** nested fields, where deep learning trains multi-layer networks to learn features directly from data
- **Biological vs artificial neurons:** the inspiration is real, but the mechanism is different
- **State of AI:** classification, detection, segmentation, text-to-image and text-to-video
- **Modern AI categories:** futuristic, skeptical, political and enterprise AI
- **Ethics and regulation:** bias, safety and misinformation, with the EU AI Act setting risk-based obligations

Summary

What version will you create?



YOUR TURN

The future is in your hands …

Resources

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning

Papers

1. Yeh, F. C., Panesar, S., Fernandes, D., Meola, A., Yoshino, M., Fernandez-Miranda, J. C., ... & Verstynen, T. (2018). Population-averaged atlas of the macroscale human structural connectome and its network topology. NeuroImage, 178, 57-68.
2. Mitsuhashi, N., Fujieda, K., Tamura, T., Kawamoto, S., Takagi, T., & Okubo, K. (2009). BodyParts3D: 3D structure database for anatomical concepts. Nucleic Acids Research, 37(Database issue), D782-D785. https://doi.org/10.1093/nar/gkn613
3. Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). YOLOv10: Real-Time End-to-End Object Detection. arXiv [cs.CV]. http://arxiv.org/abs/2405.14458
4. Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., ... Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. arXiv [cs.CV]. http://arxiv.org/abs/2408.00714
5. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv [cs.CV]. http://arxiv.org/abs/2204.06125
6. Morris, M.R., Sohl-Dickstein, J., Fiedel, N., Warkentin, T., Dafoe, A., Faust, A., Farabet, C. & Legg, S. (2024). Position: Levels of AGI for Operationalizing Progress on the Path to AGI. Proceedings of the 41st International Conference on Machine Learning, in Proceedings of Machine Learning Research 235:36308-36321. https://proceedings.mlr.press/v235/morris24b.html
7. Tu, T., Azizi, S., Driess, D., Schaekermann, M., Amin, M., Chang, P.-C., Carroll, A., Lau, C., Tanno, R., Ktena, I., Mustafa, B., Chowdhery, A., Liu, Y., Kornblith, S., Fleet, D., Mansfield, P., Prakash, S., Wong, R., Virmani, S., ... Natarajan, V. (2023). Towards Generalist Biomedical AI. arXiv [cs.CL]. https://doi.org/10.48550/arXiv.2307.14334

Image Sources

- Unless otherwise noted, the figures were created for this course.
- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.
- Dancing Robot: Gif robot.git, Meks01, CC BY-SA 4.0, via Wikimedia Commons.
- Figures built on Share-Alike source images are themselves released under CC BY-SA 4.0.
- Detection and segmentation figures were produced with Ultralytics YOLO11 and FastSAM (AGPL-3.0). That licence covers the software, not the images produced with it. Text-to-image bias panels were generated with SD-Turbo under Stability AI's non-commercial research licence.
- ImageNet leaderboard data: Papers with Code evaluation tables (archived), CC BY-SA 4.0. Word-embedding bias figure computed on GloVe vectors (Pennington, Socher & Manning, 2014).
- Reproduced paper figures are used under their own licences: YOLOv10, DALL·E 2 and Med-PaLM M are CC BY 4.0.

Image Sources

- Innorobo 2015 - NAO (cropped).JPG, © Xavier Caré / Wikimedia Commons / CC BY-SA 4.0, https://creativecommons.org/licenses/by-sa/4.0/
- Horst von Harbou - Metropolis Maschinenmensch.jpg, Horst von Harbou, Public domain, via Wikimedia Commons
- Sierra Supercomputer (48002385338).jpg, U.S. Department of Energy from United States, Public domain, via Wikimedia Commons
- Moore's Law Transistor Count 1970-2020.png, Max Roser, Hannah Ritchie, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- Douglas DC-8 rear from down.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1563528
- Grey Heron taking flight in Paul de Arzila Natural Reserve, Portugal (52987496443).jpg, Carlos Eduardo Joos, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

Image Sources

- Tractography - Cingulum - animation.gif, Tractography data: Yeh, F. C., Panesar, S., Fernandes, D., Meola, A., Yoshino, M., Fernandez-Miranda, J. C., ... & Verstynen, T. (2018). Population-averaged atlas of the macroscale human structural connectome and its network topology. NeuroImage, 178, 57-68. PubMed: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6921501/. Skull and brain data: Nobutaka Mitsuhashi, Kaori Fujieda, Takuro Tamura, Shoko Kawamoto, Toshihisa Takagi, and Kousaku Okubo. (2009) BodyParts3D: 3D structure database for anatomical concepts. Nucleic Acids Research, Vol. 37, Database issue D782-D785, https://doi.org/10.1093/nar/gkn613, CC BY-SA 2.1 JP <https://creativecommons.org/licenses/by-sa/2.1/jp/deed.en>, via Wikimedia Commons
- BCI sensor mounting types — invasive (IM), semi-invasive (ECoG), and non-invasive (MEG, EEG, fNIRS).png, Peksa Janis and Dmytro Mamchur, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- Autonomous-driving-Barcelona.jpg, Eschenzweig, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Myoelectric prosthetic arm.jpg, Petty Officer 2nd Class Greg Mitchell of the United States Navy, Public domain, via Wikimedia Commons

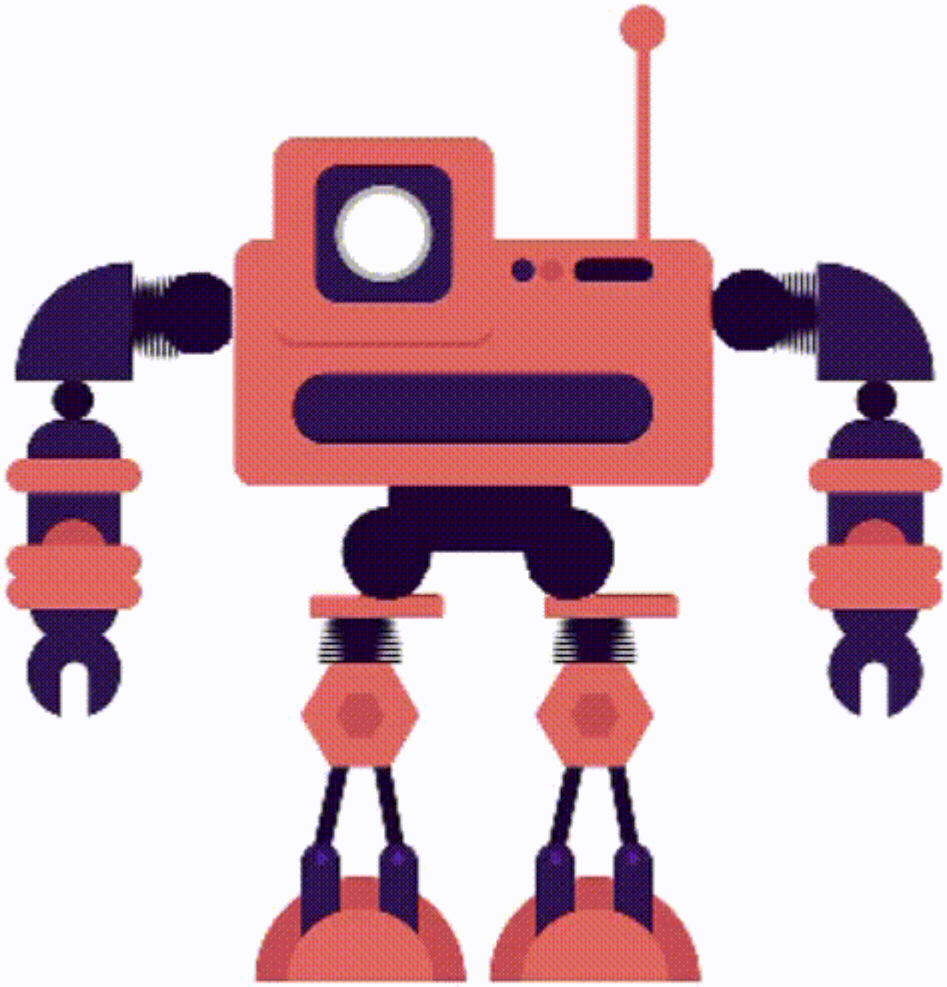
Image Sources

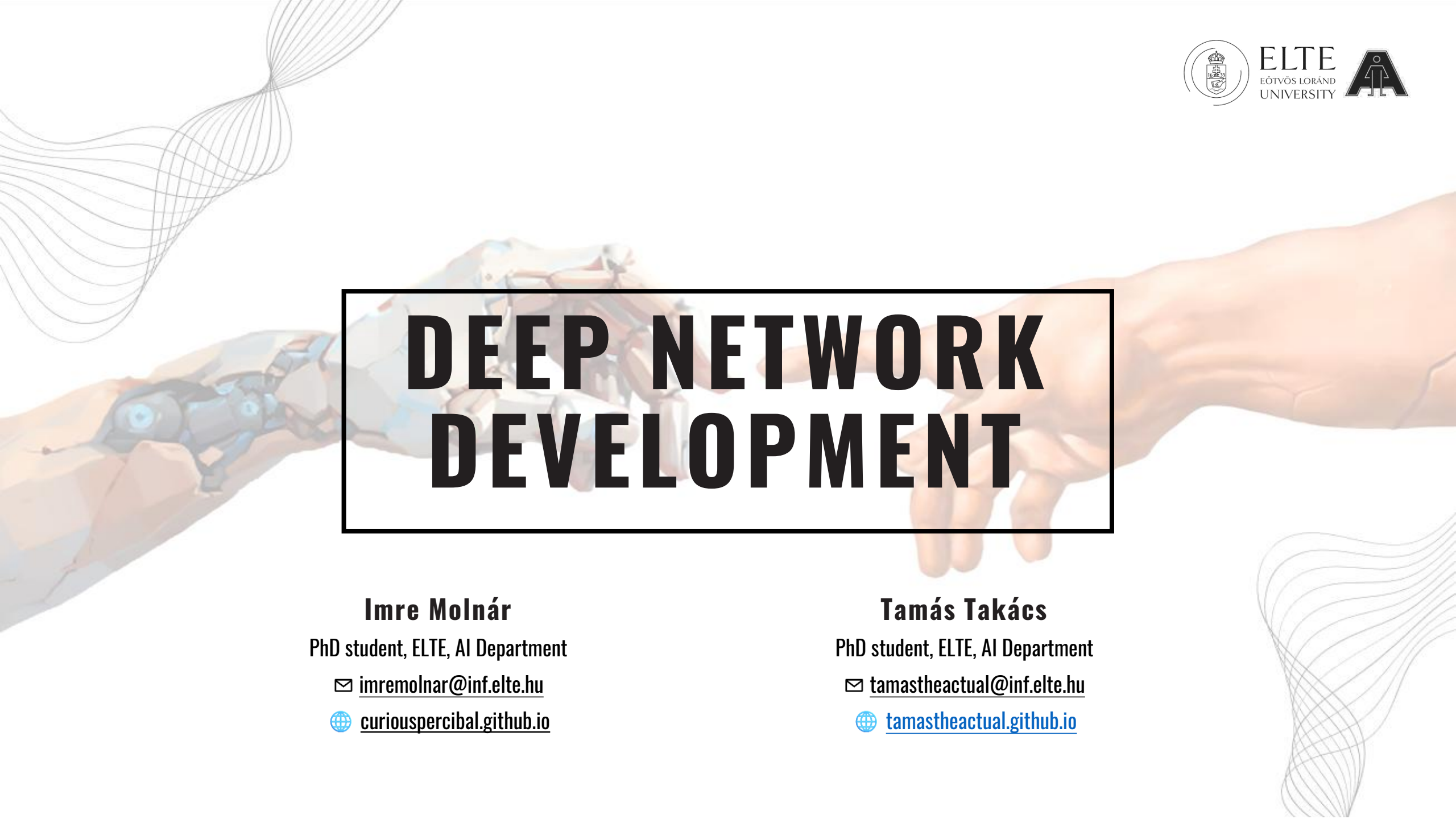
- BrainGate.jpg, PaulWicks at English Wikipedia, Public domain, via Wikimedia Commons
- T2-weighted coronal MRI tectal glioma.jpg, Jto410, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
- Huge Meningioma.jpg, Fowle, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Male and female deepfake.jpg, Vahan03, Public domain, via Wikimedia Commons
- Pope Francis in puffy winter jacket.jpg, Midjourney, Public domain, via Wikimedia Commons
- Deepfake.jpg, Ministerie van Buitenlandse Zaken, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons
- Courtroom One Gavel - Flickr - Joe Gratz.jpg, Joe Gratz, CC0, via Wikimedia Commons
- Dictators - Kim Jong-Un by RepresentUs.webm, RepresentUs, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons
- Frame from - Trump Gaza.webm, Arcana, prompted by Ariel Vromen (Q15957667) and Solo Avital (Q1416127), Public domain, via Wikimedia Commons

Image Sources

- Official Presidential Portrait of President Donald J. Trump (2025).jpg, Daniel Torok, Public domain, via Wikimedia Commons
- Kamala Harris Vice Presidential Portrait.jpg, Lawrence Jackson, Public domain, via Wikimedia Commons
- Détection de personne - exemple 2.jpg, Exemple_d'infractions.jpg: llOzzderivative work: Indif, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons
- WBAMC first in DoD to use robot for surgery 160426-A-EK666-506.jpg, Marcy Sanchez, Public domain, via Wikimedia Commons
- MQ-9 Reaper in flight (2007).jpg, U.S. Air Force photo/Staff Sgt. Brian Ferguson, Public domain, via Wikimedia Commons
- Ploen Marionetten 1.jpg, Jürgen Howaldt, CC BY-SA 2.0 DE <https://creativecommons.org/licenses/by-sa/2.0/de/deed.en>, via Wikimedia Commons
- All Euro banknotes.png, Bericht, CC0, via Wikimedia Commons
- Mark Zuckerberg F8 2019 Keynote (32830578717) (cropped).jpg, Anthony Quintano from Westminster, United States, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons
- Figure 2 — Tu et al., Towards Generalist Biomedical AI (2023), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://arxiv.org/abs/2307.14334
- Figure 1 — Tu et al., Towards Generalist Biomedical AI (2023), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://arxiv.org/abs/2307.14334

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 2

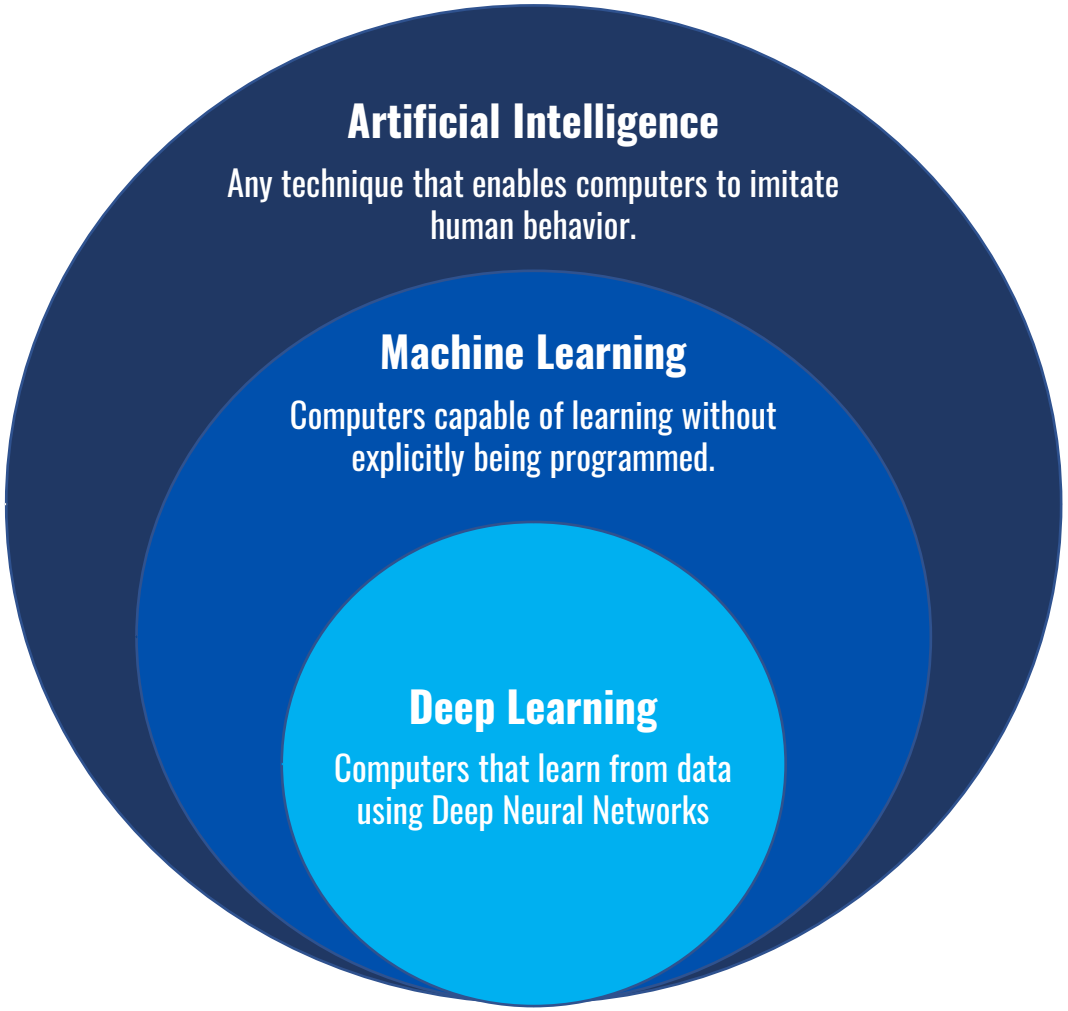
Linear Regression & Artificial Neural Networks

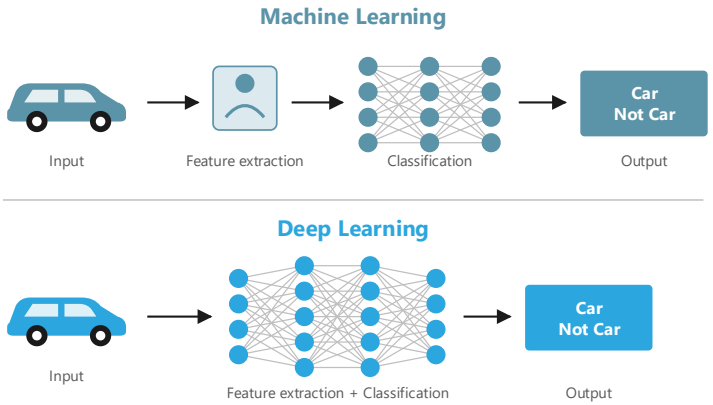
1 Linear Regression

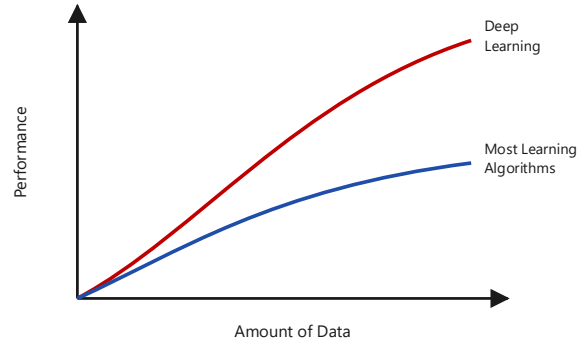
2 Artificial Neural Networks

3 Backpropagation

Previously on Lecture 1









Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 2

Linear Regression & Artificial Neural Networks

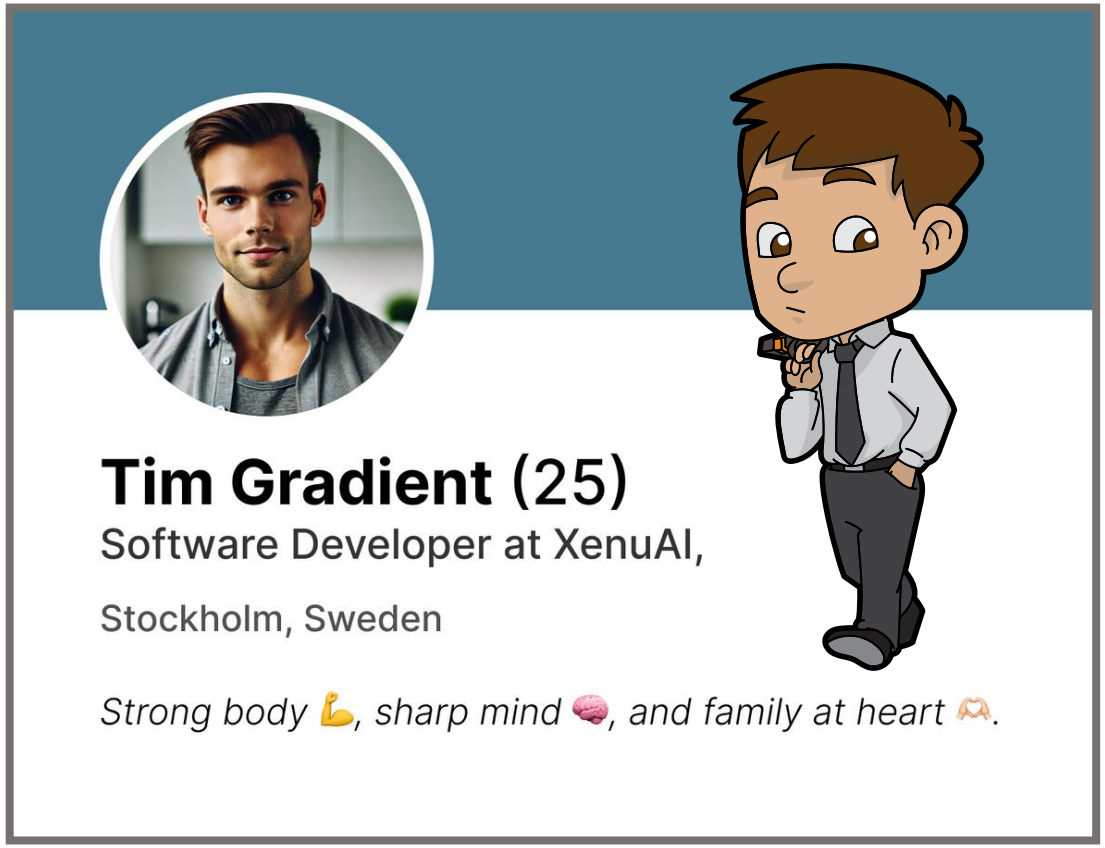
1 Linear Regression

2 Artificial Neural Networks

3 Backpropagation

Meet Tim and Tom

Once childhood friends, but now on different life paths



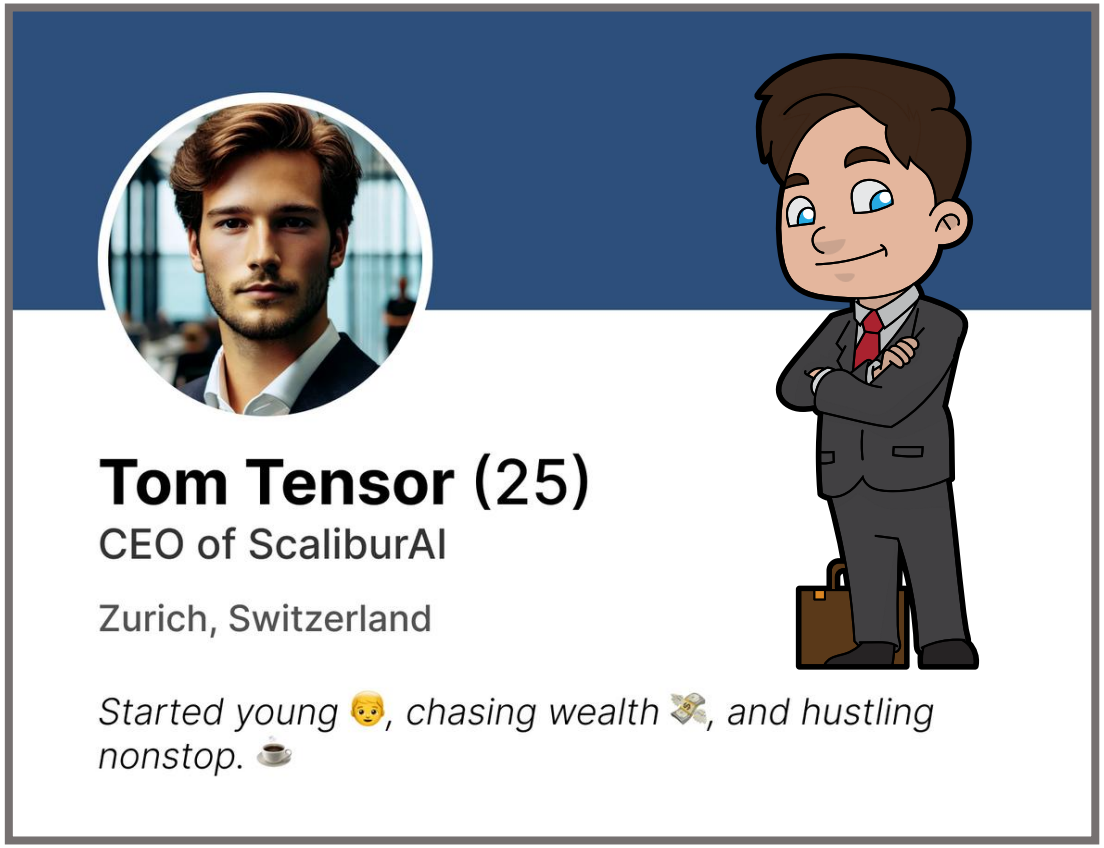


Image generated with ChatGPT Images (OpenAI), 2026, https://chatgpt.com/
Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

One day they met and…

Tim was mad at Tom for not spending time with him anymore.

Tim, emotionally said: "You're probably going to end up dead alone…"

"No way! I am healthier than you!"



Tom was mad at Tim for living in his comfort zone and not use his full potential.

Tom, emotionally replied: "But you will die first…"

"No way! I am wealthier than you!"

… and that is when they had an idea …

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Meet Tim and Tom







TO USE ARTIFICIAL INTELLIGENCE TO PREDICT THEIR LIFE EXPECTANCY!

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Supervised Learning: Linear Regression

Life Expectancy function

$$f(x) = y$$

- Have: (x, y)
  - **x** – data
  - **y** – label
- **Goal: Learn a function to map** $x \rightarrow y$
- Regression – Predict real valued / continuous output: $y \in \mathbb{R}$
  - What is the height of person B?
  - What is going to be the price of a share tomorrow?
  - How many cars are in a city?
  - **What is the life expectancy of person A?**

$f\left(\right.$  $\left.\right)$ =77.6 years

$f\left(\right.$  $\left.\right)$ =78.1 years

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Collect a dataset

- Life Expectancy (WHO) Dataset (https://www.kaggle.com/datasets/lashagoch/life-expectancy-who-updated)
- Statistical Analysis on factors influencing Life Expectancy across different countries
- Dataset has 2938 data points, each having 20 factors influencing Life Expectancy, the country name and the actual Life Expectancy value

- Adult Mortality - Adult Mortality Rates of both sexes, probability of dying between 15 and 60 years per 1000 population;
- BMI - Average Body Mass Index of entire population;
- GDP - Gross Domestic Product per capita (in USD);
- ...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          2938 non-null   object
 1   Year                             2938 non-null   int64
 2   Status                           2938 non-null   object
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years             2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

Exploring the dataset

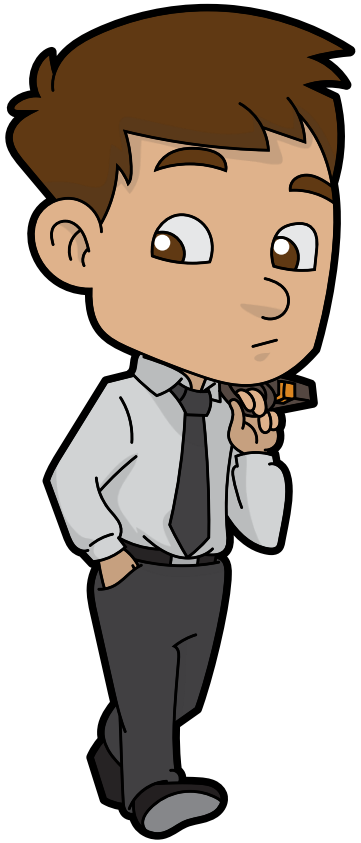
Country	Year	Status	Life expectancy	Adult Mortality	Infant Deaths	Alcohol	Percentage Expenditure	Hepatitis B	Measles	BMI
Hungary	2014	Developed	75.6	137	0	0.01	160.9449		0	64.2

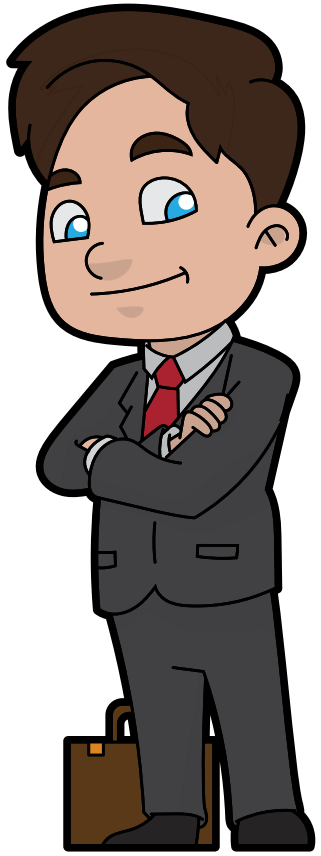
Under-five Deaths	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	Thinness 1-19 years	Thinness 5-9 years	Income composition of resources	Schooling
0	99	7.4	99	0.1	14117.98	9866468	1.7	1.6	0.834	15.8

Feature Selection

Tim chose BMI (independent variable) for predicting Life expectancy (dependent variable)

X = BMI
Y = Life Expectancy





Tom chose GDP (independent variable) for predicting Life expectancy (dependent variable)

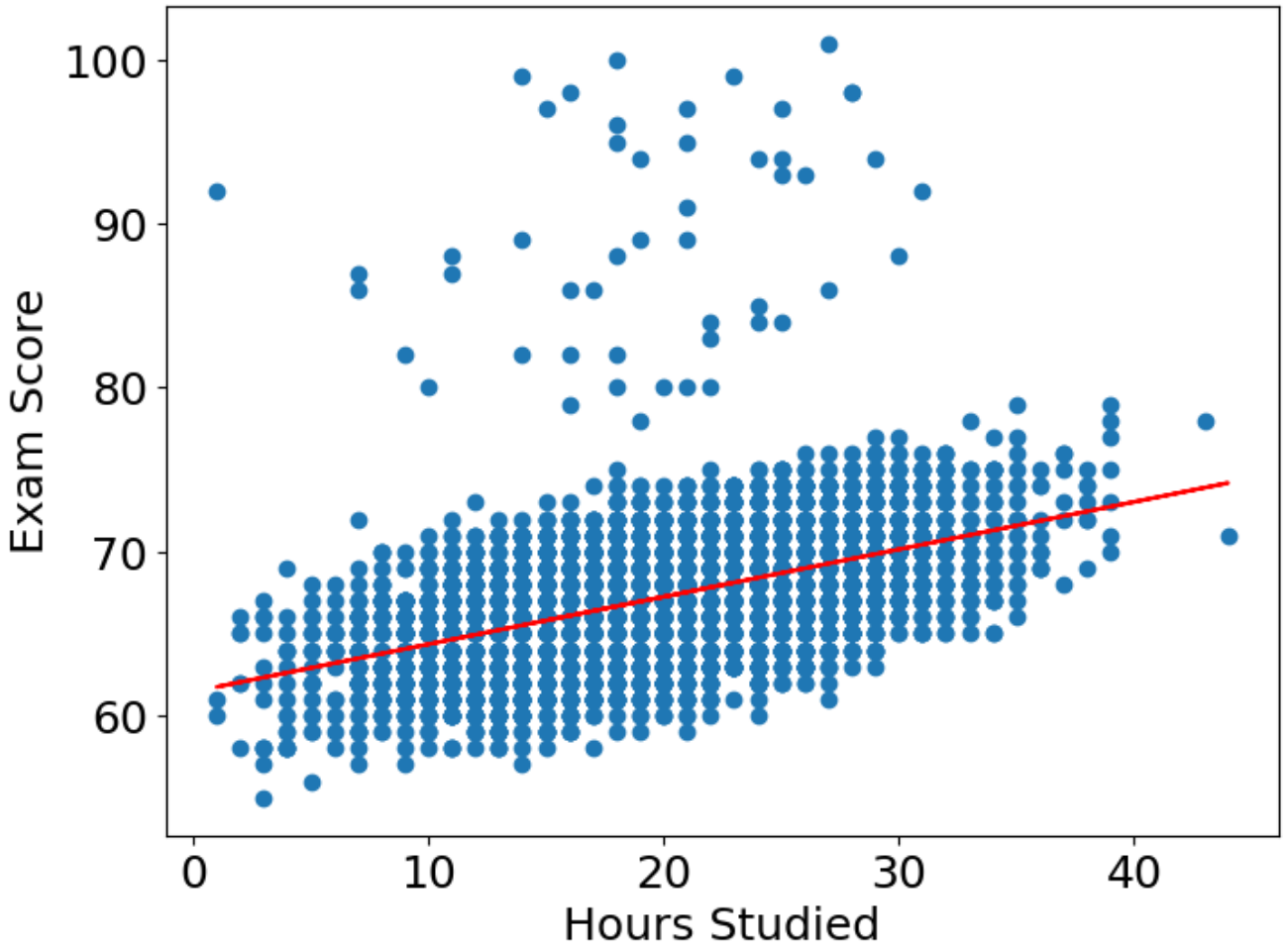
X = GDP
Y = Life Expectancy

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Example: Student Performance Factors

- **x** – Hours Studied
- **y** – Exam Score (out of 100)

- We expect that the more hours spent with studying the higher the exam score (the two variable has a linear correlation)



Data source: Lai Nguyen (lainguyn123), Student Performance Factors, Kaggle, CC0 1.0 Universal. https://www.kaggle.com/datasets/lainguyn123/student-performance-factors

Example

x – (input) number of residents
y – (output) number of cars

Seems to have a linear relationship (we try to fit a line):

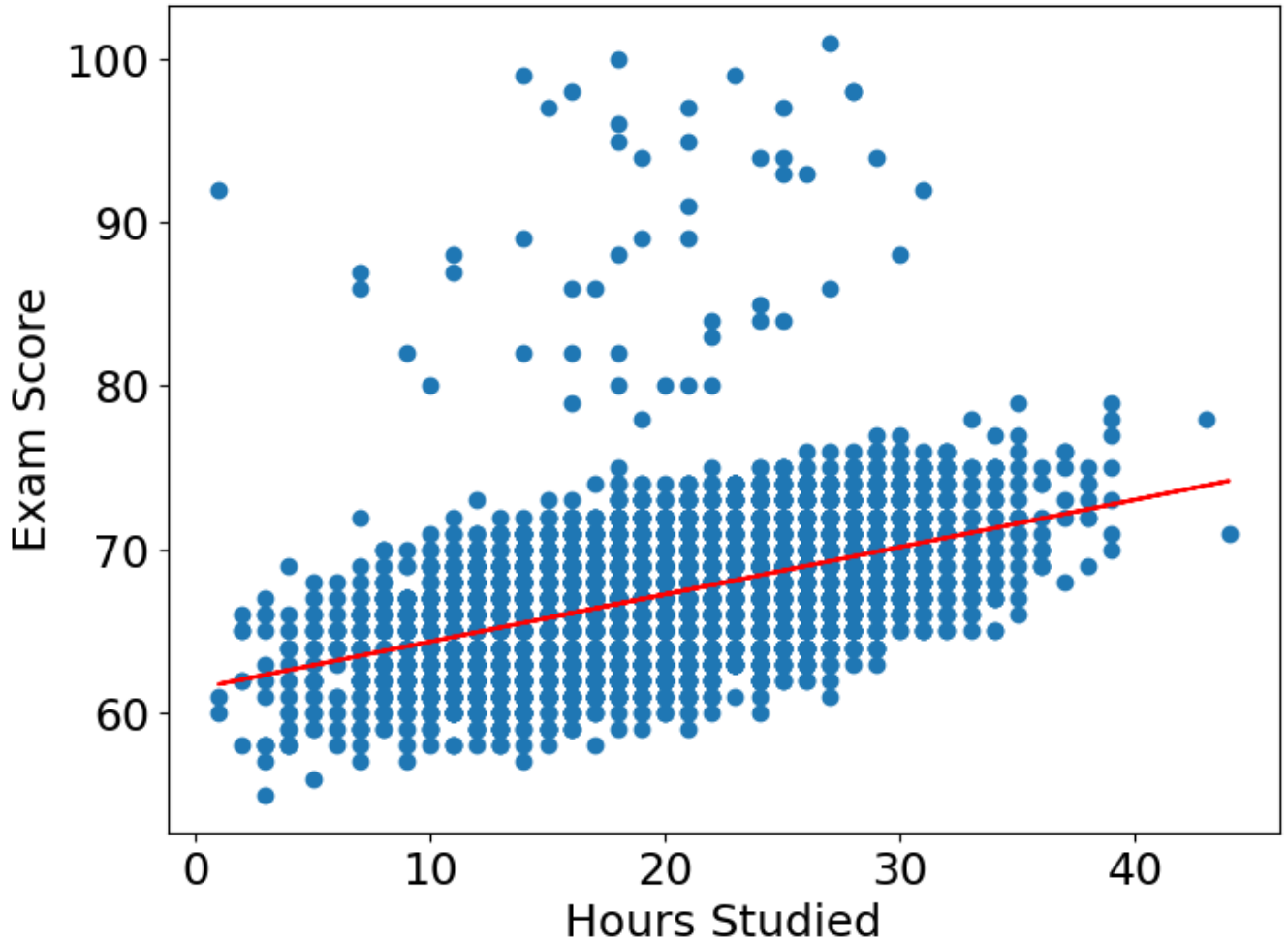
$$\hat{y} = \theta_0 + \theta_1 x = \theta x$$

$$\boldsymbol{\theta_0} = ?\,, \boldsymbol{\theta_1} = ?$$

$$y = \underbrace{mx}_{\text{Slope}} + \underbrace{b}_{\text{y-intercept}}$$

Slope: Describes the slope of the line

y-intercept: Describes where the line crosses the y-axis



Supervised Learning: Linear Regression

x – (input) floor area (sq. meters)
y – (output) price (in M HUF)

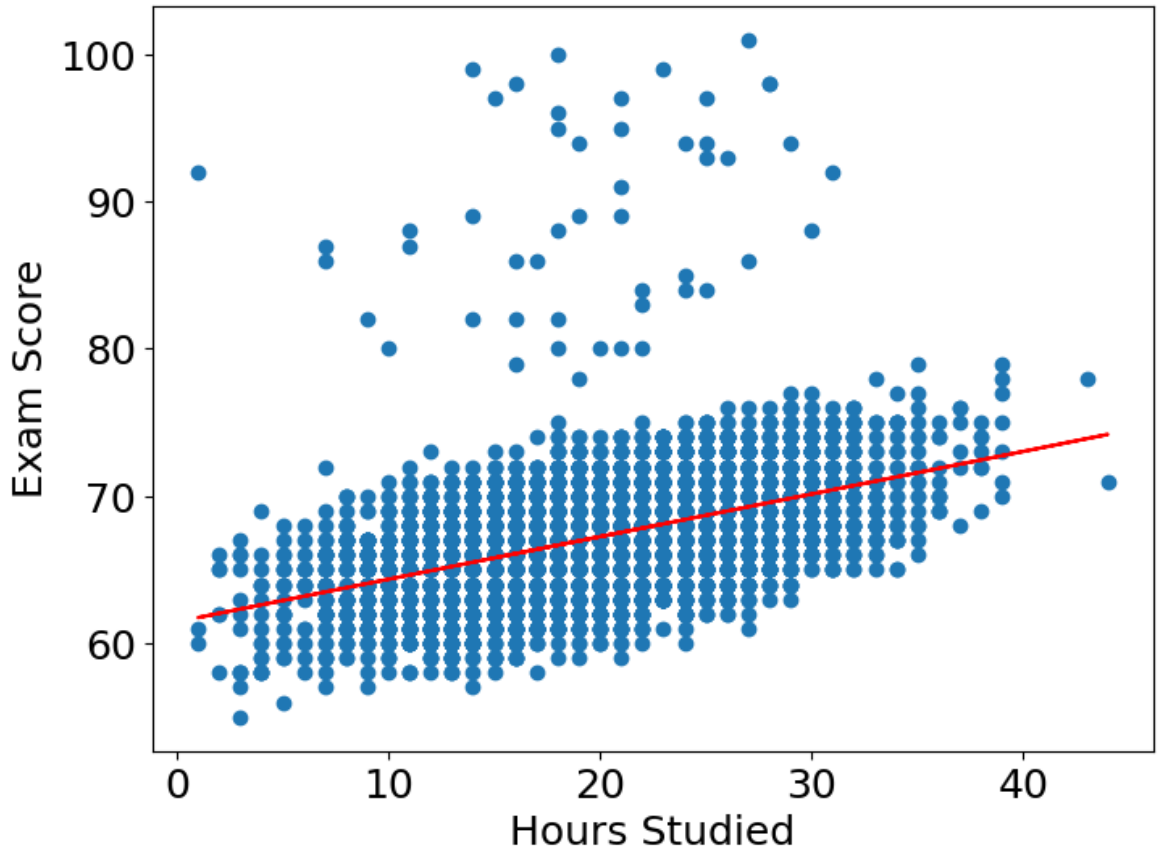
Seems to have a linear relationship. We want to find a function:

$$h(x) = \hat{y} = \theta_0 + \theta_1 x = \theta x$$

To select best, θ that minimize the loss:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(\theta x^{(i)} - y^{(i)} \right)^2$$



Supervised Learning: Linear Regression

x – (input) floor area (sq. meters)
y – (output) price (in M HUF)

Seems to have a linear relationship. We want to find a function:

$$h(x) = \hat{y} = \theta_0 + \theta_1 x = \theta x$$

To select best, θ that minimize the loss:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(\theta x^{(i)} - y^{(i)} \right)^2$$

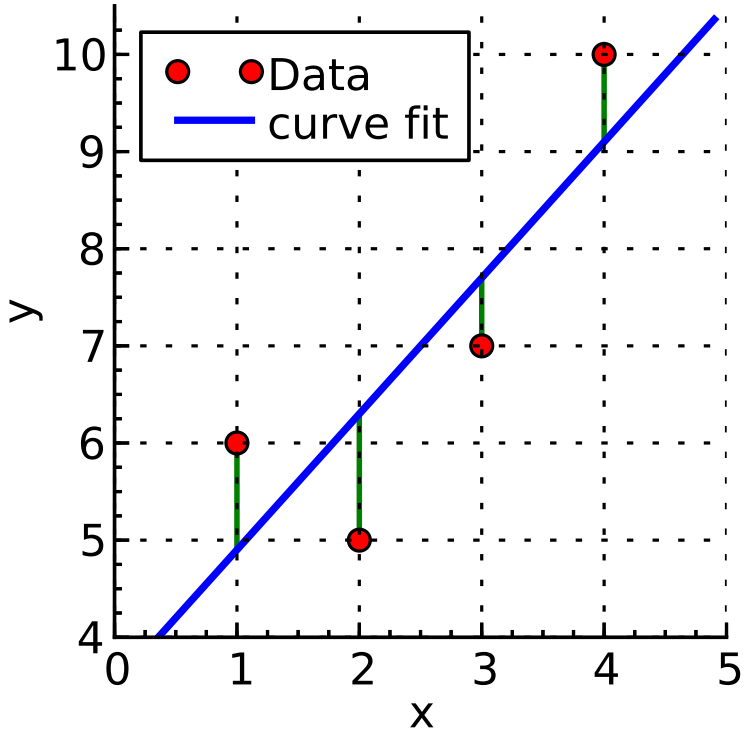


Image Source: Linear least squares example2.svg, Krishnavedala, CC BY-SA 3.0 , via Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Linear_least_squares_example2.svg

Vectorization

Say we have 200 data points:

$$\underset{(200,2)}{x} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \dots & \dots \\ 1 & x^{(200)} \end{bmatrix}, \quad \underset{(200,1)}{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(200)} \end{bmatrix}$$

$$y = \underbrace{mx}_{\text{Slope}} + \underbrace{b}_{\text{y-intercept}}$$

Slope: Describes the slope of the line

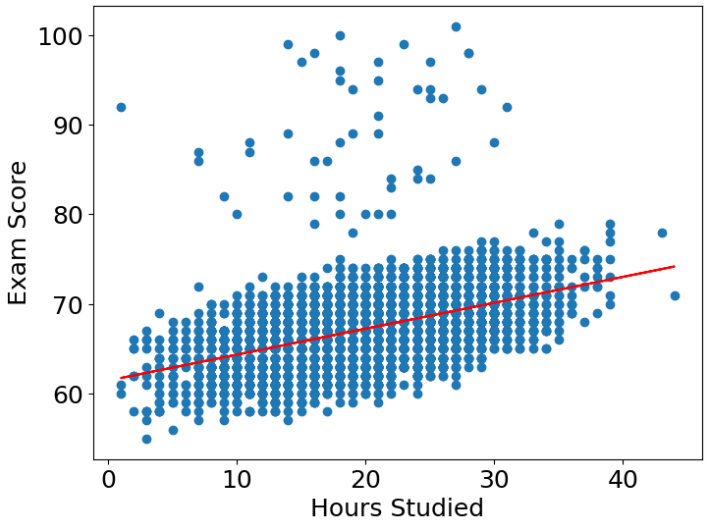
y-intercept: Describes where the line crosses the y-axis

We want to find $\underset{(2,1)}{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ for $h(x) = \hat{y} = x\theta = \begin{bmatrix} 1 \cdot \theta_0 + x^{(1)}\theta_1 \\ 1 \cdot \theta_0 + x^{(2)}\theta_1 \\ \dots \\ 1 \cdot \theta_0 + x^{(200)}\theta_1 \end{bmatrix}$

That minimizes loss:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_0 + x^{(i)}\theta_1 - y^{(i)}\right)^2$$

(vectorized form): $J(\theta) = \left(\hat{Y} - Y\right)^T\left(\hat{Y} - Y\right) = (X\theta - Y)^T(X\theta - Y)$



Supervised Learning: Linear Regression

The problem can be summarized:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

Find the optimal θ^* that minimizes the loss (in this case the Mean Squared Error):

$$J(\theta^*) = \frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)}\theta^* - y^{(i)}\right)^2$$

How to find optimal θ^* ?

- Analytical solution: Normal equation
- Numerical solution: Gradient Descent

Analytical solution: Normal Equation

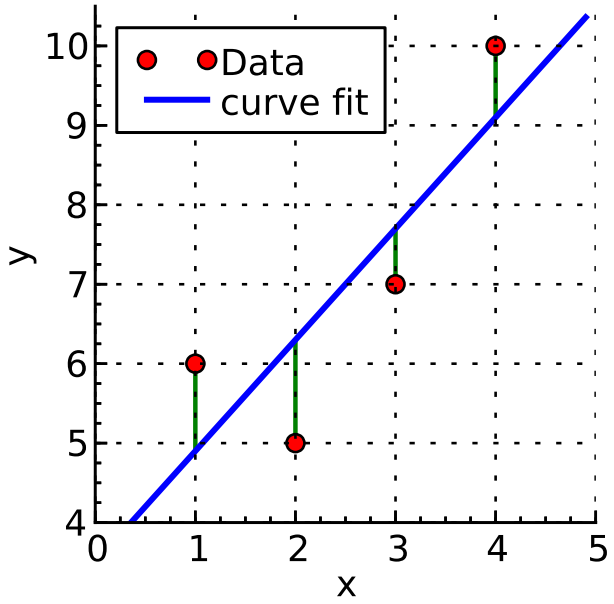
Say we have 200 data points:

$$x = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \dots & \dots \\ 1 & x^{(200)} \end{bmatrix}$$

$$y = \underbrace{m}_{\text{Slope}}x + \underbrace{b}_{\text{y-intercept}}$$

Slope: Describes the slope of the line

y-intercept: Describes where the line crosses the y-axis



$$\theta = [\theta_0, \theta_1]^T$$

$$\mathbf{y} = [y^{(1)}, y^{(2)}, \dots, y^{(200)}]^T$$

$$\hat{\mathbf{y}} = X\theta$$

$$J(\theta) = (X\theta - Y)^T(X\theta - Y)$$

Normal Equation [1]: $\theta^* = (X^T X)^{-1}(X^T Y)$

θ^* is the optimal parameter that minimizes the loss function $J(\theta)$

Image Source: Linear least squares example2.svg, Krishnavedala, CC BY-SA 3.0 , via Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Linear_least_squares_example2.svg

Feature selection (recap)

Tim chose BMI (independent variable) for predicting Life expectancy (dependent variable)

X = BMI
Y = Life Expectancy



Tom chose GDP (independent variable) for predicting Life expectancy (dependent variable)

X = GDP
Y = Life Expectancy

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Analytical solution: Normal Equation (Tim)

$$y = \underbrace{mx}_{\text{Slope}} + \underbrace{b}_{\text{y-intercept}}$$

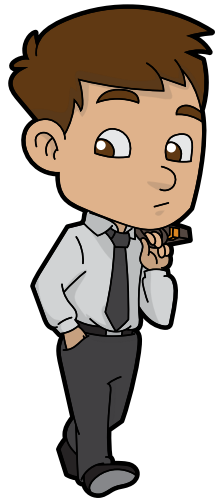
Slope: Describes the slope of the line

y-intercept: Describes where the line crosses the y-axis

$$\theta^* = (X^T X)^{-1}(X^T Y)$$

```
def normal_equation(X,Y):
    return np.linalg.inv(X.T @ X) @ (X.T @ Y)

tim_theta = normal_equation(tim_data_ready, Y)
print(f'y = mx + b')
print(f'y = {tim_theta[1]}x + {tim_theta[0]}')
```



$$y = mx + b$$
$$y = 0.26743768477023727x + 58.98617203832164$$

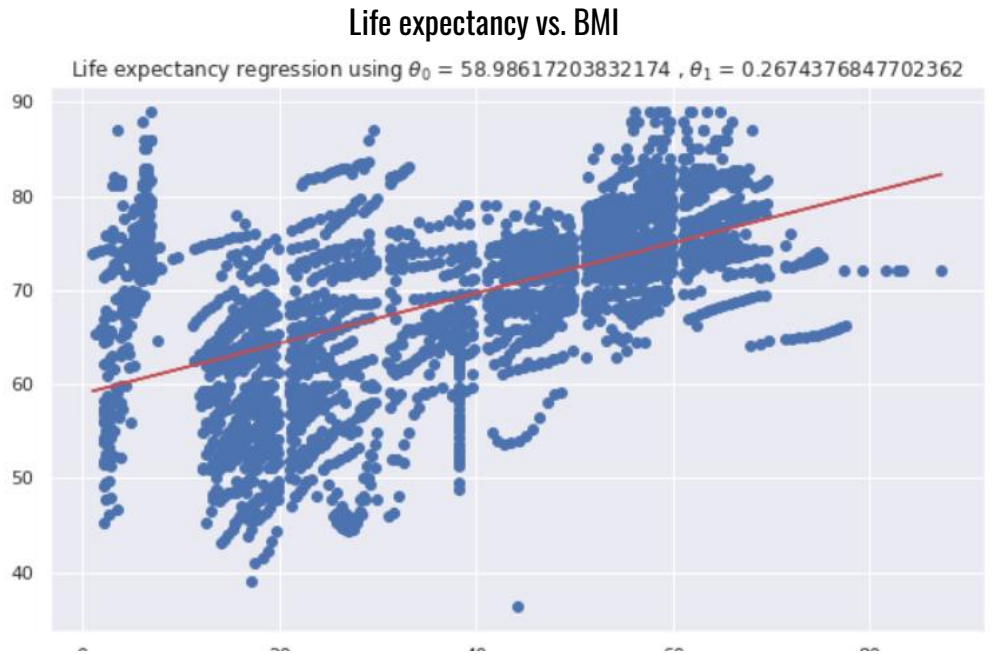
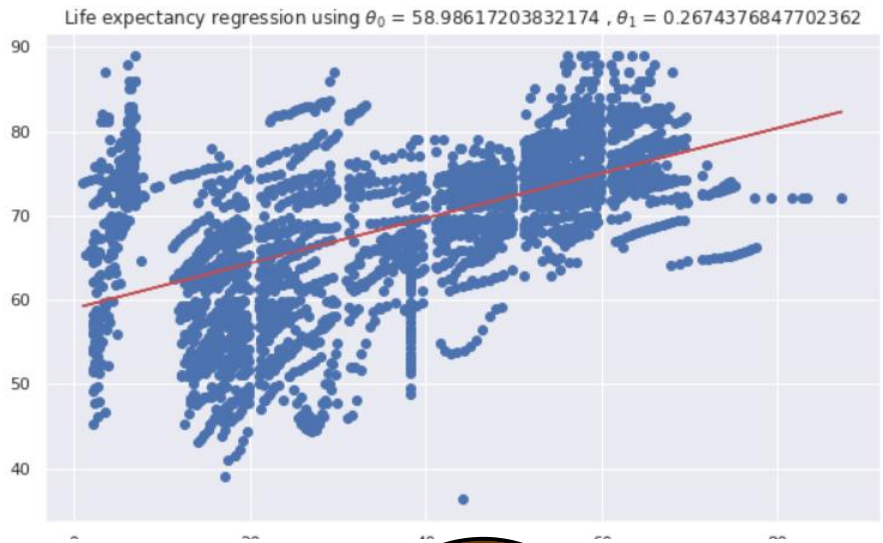


Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons

Analytical solution: Normal Equation (Tim)





```
def life_expectancy(X, theta):
    X = np.concatenate((np.ones(1),np.array(X)),axis=0)
    return round(np.dot(X, theta),1)

height = 1.82 #float(input("Please input your height (in meters): ")) #1.82
weight = 80 #float(input("Please input your weight (in kilograms): ")) #80
bmi = weight / height**2
print("BMI:",bmi)
life_exp_tim = life_expectancy([bmi], tim_theta)
print("Tim's life expectancy is", life_exp_tim , "years.")
```

BMI: 24.151672503320853

Tim's life expectancy is 65.4 years.

```
height = 1.80 #float(input("Please input your height (in meters): ")) #1.82
weight = 73 #float(input("Please input your weight (in kilograms): ")) #80
bmi = weight / height**2
print("BMI:",bmi)
life_exp_tim_tom = life_expectancy([bmi], tim_theta)
print("Tom's life expectancy predicted by Tim's model is", life_exp_tim_tom , "years.")
```

BMI: 22.530864197530864

Tom's life expectancy is 65.0 years.

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons

Analytical solution: Normal Equation (Tom)

$$y = \underbrace{mx}_{\text{Slope}} + \underbrace{b}_{\text{y-intercept}}$$

Slope: Describes the slope of the line

y-intercept: Describes where the line crosses the y-axis

```
tom_theta = normal_equation(tom_data_ready, Y)
print(f'y = mx + b')
print(f'y = {tom_theta[1]}x + {tom_theta[0]}')
```



$$y = mx + b$$
$$y = 0.00031140962386767 78x + 66.90438993067785$$

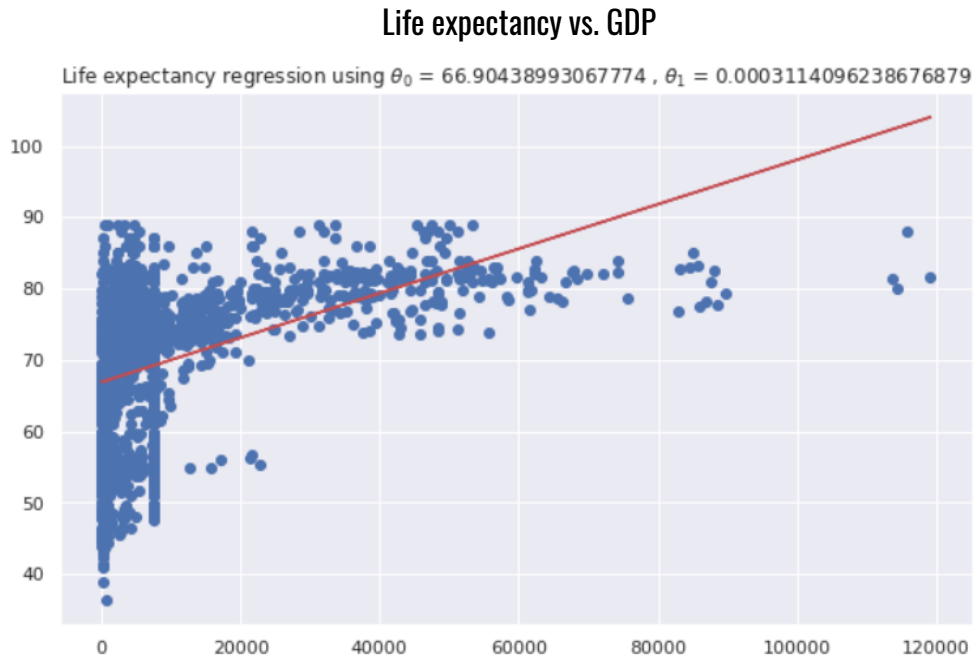
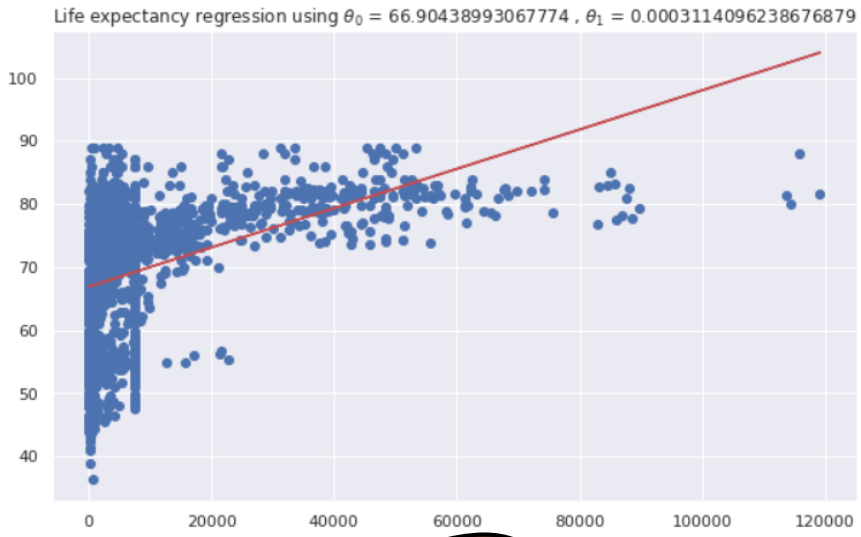


Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Analytical solution: Normal Equation (Tom)





```
def life_expectancy(X, theta):
    X = np.concatenate((np.ones(1),np.array(X)),axis=0)
    return round(np.dot(X, theta),1)

gdp = 8500 #float(input("Please input the GDP of your country: ")) #8500
life_exp_tom = life_expectancy([gdp], tom_theta)
```

Tom's life expectancy is 69.6 years.

```
gdp = 5000 #float(input("Please input the GDP of your country: ")) #5000
life_exp_tom_tim = life_expectancy([gdp], tom_theta)
print("Tim's life expectancy predicted by Tom is", life_exp_tom_tim , "years.")
```

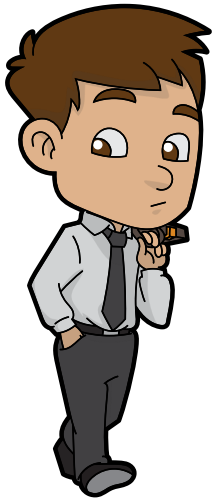
Tim's life expectancy predicted by Tom is 68.5 years.

Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Results

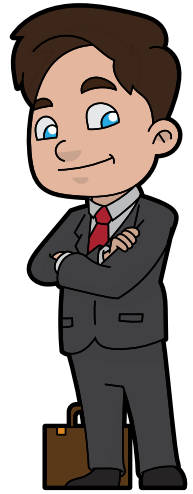
Tim's Life Expectancy predictions:
- Tim – 65.4 years
- Tom – 65 years

Tom's Life Expectancy predictions:
- Tim – 68.5 years
- Tom – 69.6 years



NO AGREEMENT!!

- Metrics?
- Further explore dataset?
- Better feature selection?



My model is better! I'll live longer!

NO! Mine is better! I'll live longer!

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Exploring the dataset part 2

Pearson Correlation:

- Measures the strength and direction of the linear relationship between two quantitative, continuous variables
- Correlation ranges from -1 to +1.
- Positive values mean the variables increase together, negative values mean one increases while the other decreases, and values near 0 indicate little or no linear relationship.
- Correlation measures linear association, not causation.

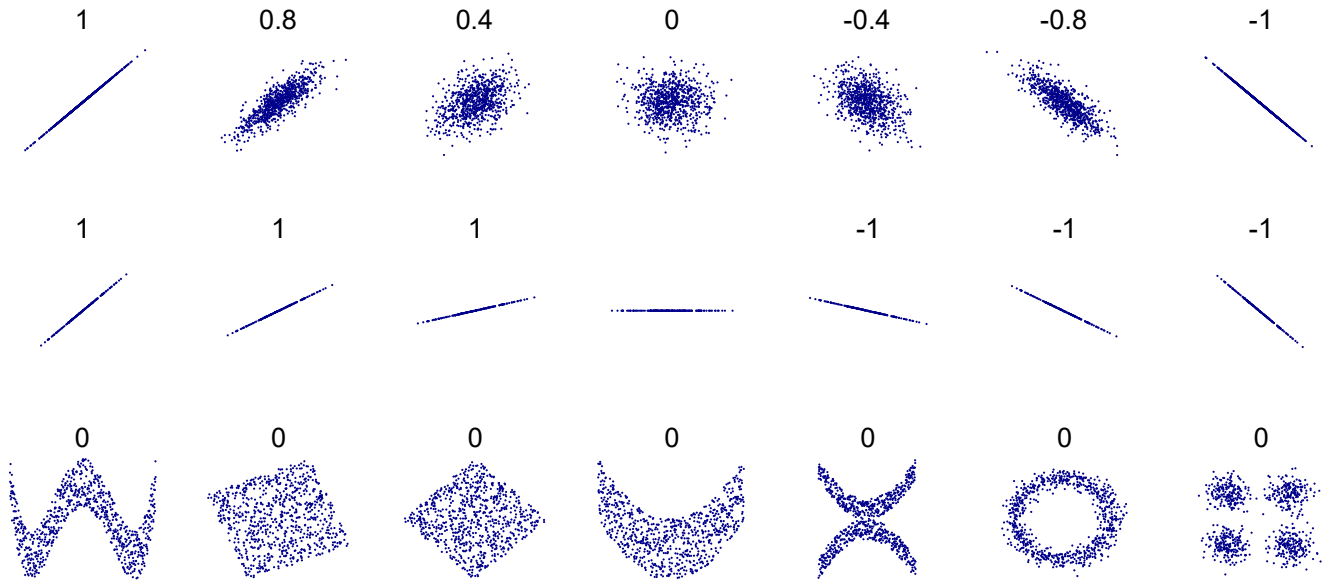
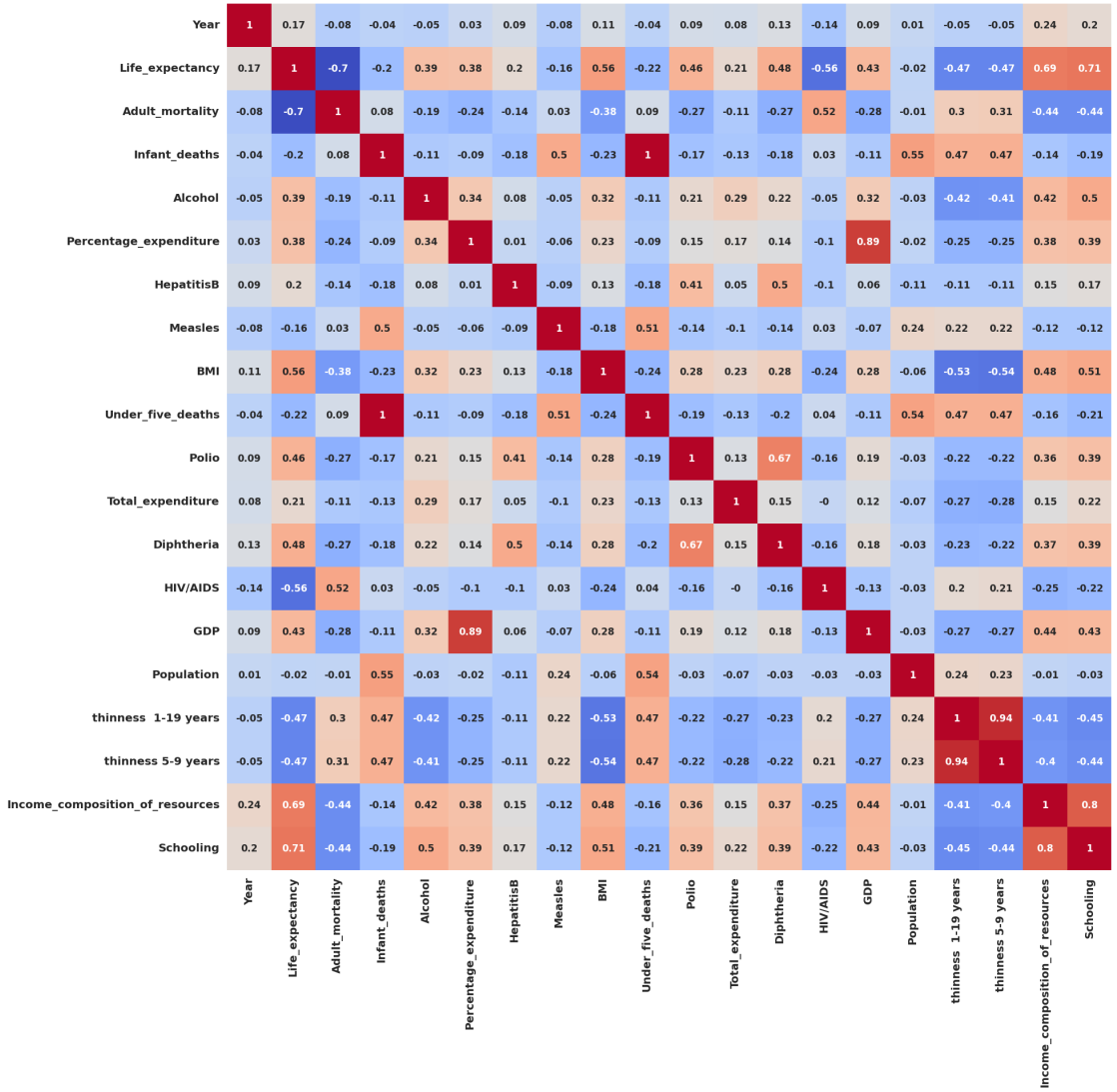


Image source: Correlation examples.png from Imagecreator at English Wikipedia, Public domain, via Wikimedia Commons

Exploring the dataset part 2

Correlation matrix

- What is the relationship between the different features and the Life expectancy?
- A correlation matrix is a table that shows the correlation coefficient between every pair of variables in a dataset.
- Each cell tells how strongly two variables are linearly* related



* In case of Pearson correlation

Exploring the dataset part 2

Tim chose: BMI
Tom chose: GDP





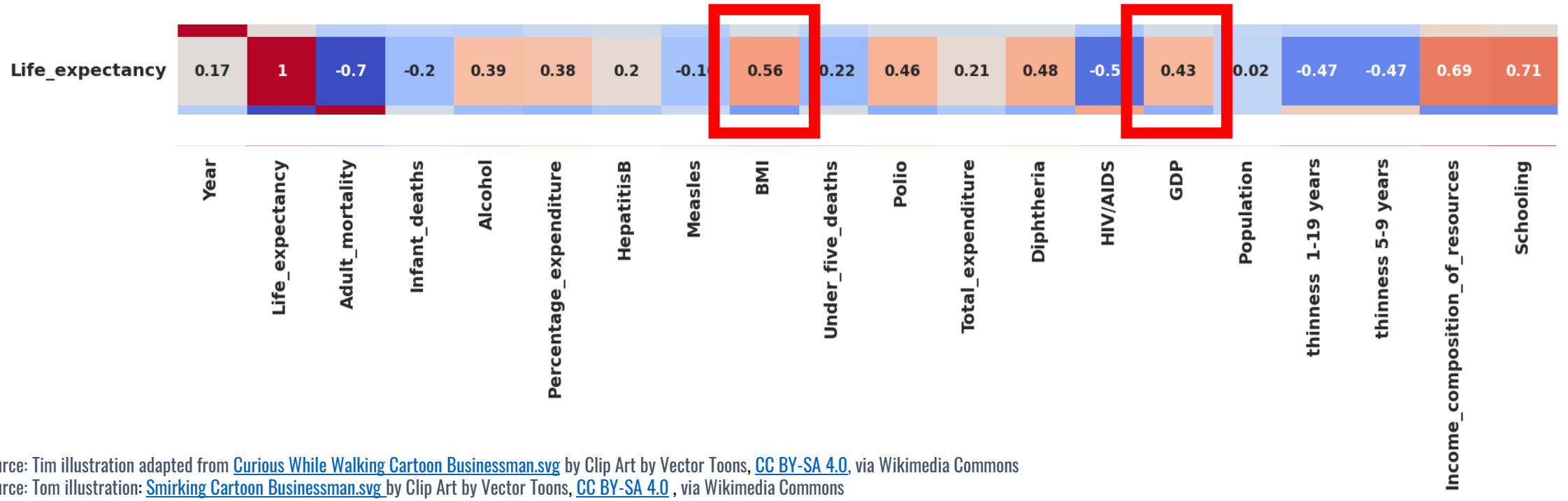


Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Exploring the dataset part 2

Now they chose multiple features to predict life expectancy:

- **Tim chose:** Adult mortality, BMI, HIV/AIDS
- **Tom chose:** GDP, Income composition of resources, Schooling





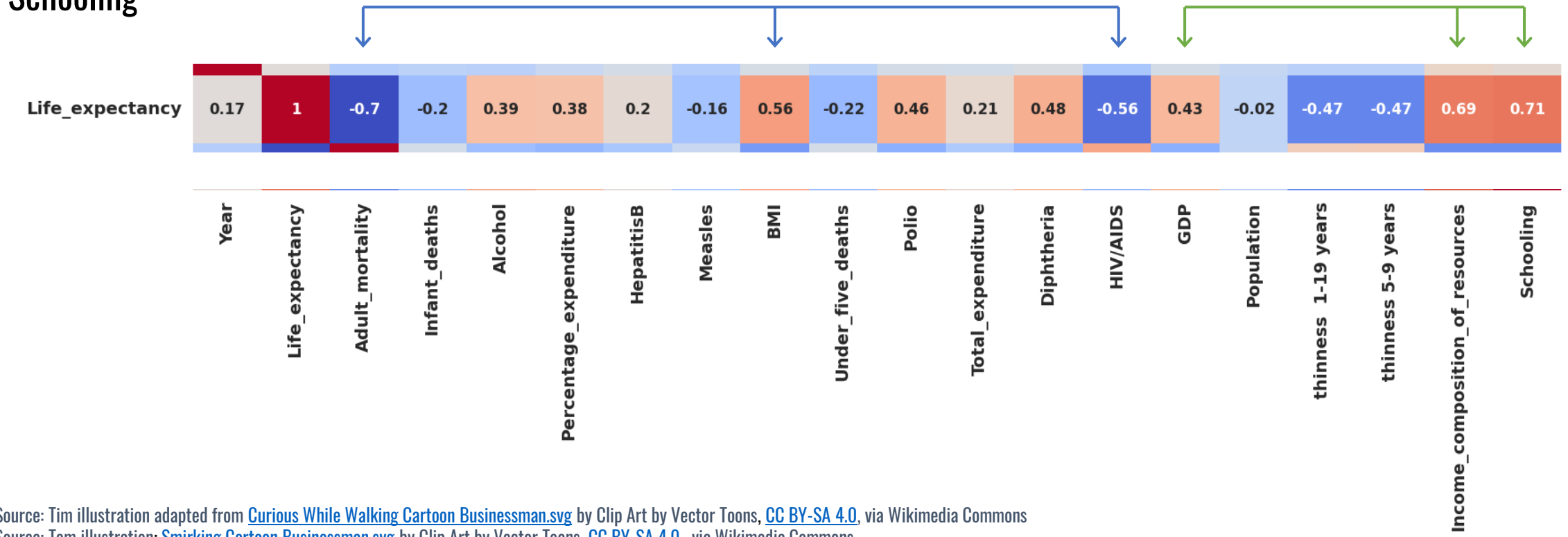
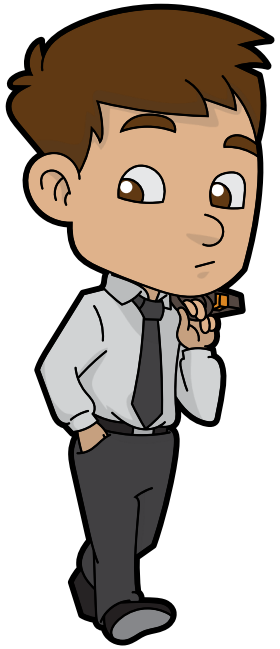


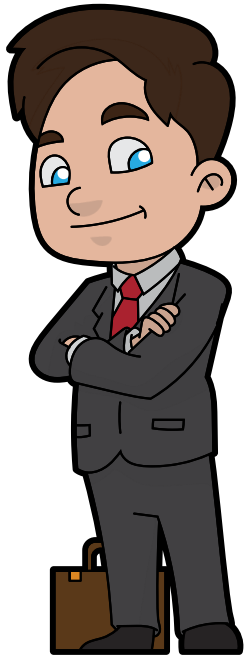
Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Exploring the dataset part 2

Tim chose BMI, Adult Mortality and HIV/AIDS (independent variables) for predicting Life expectancy (dependent variable)

X = BMI, Adult Mortality, HIV/AIDS
Y = Life Expectancy





Tom chose GDP, Income composition of resources and Schooling (independent variables) for predicting Life expectancy (dependent variable)

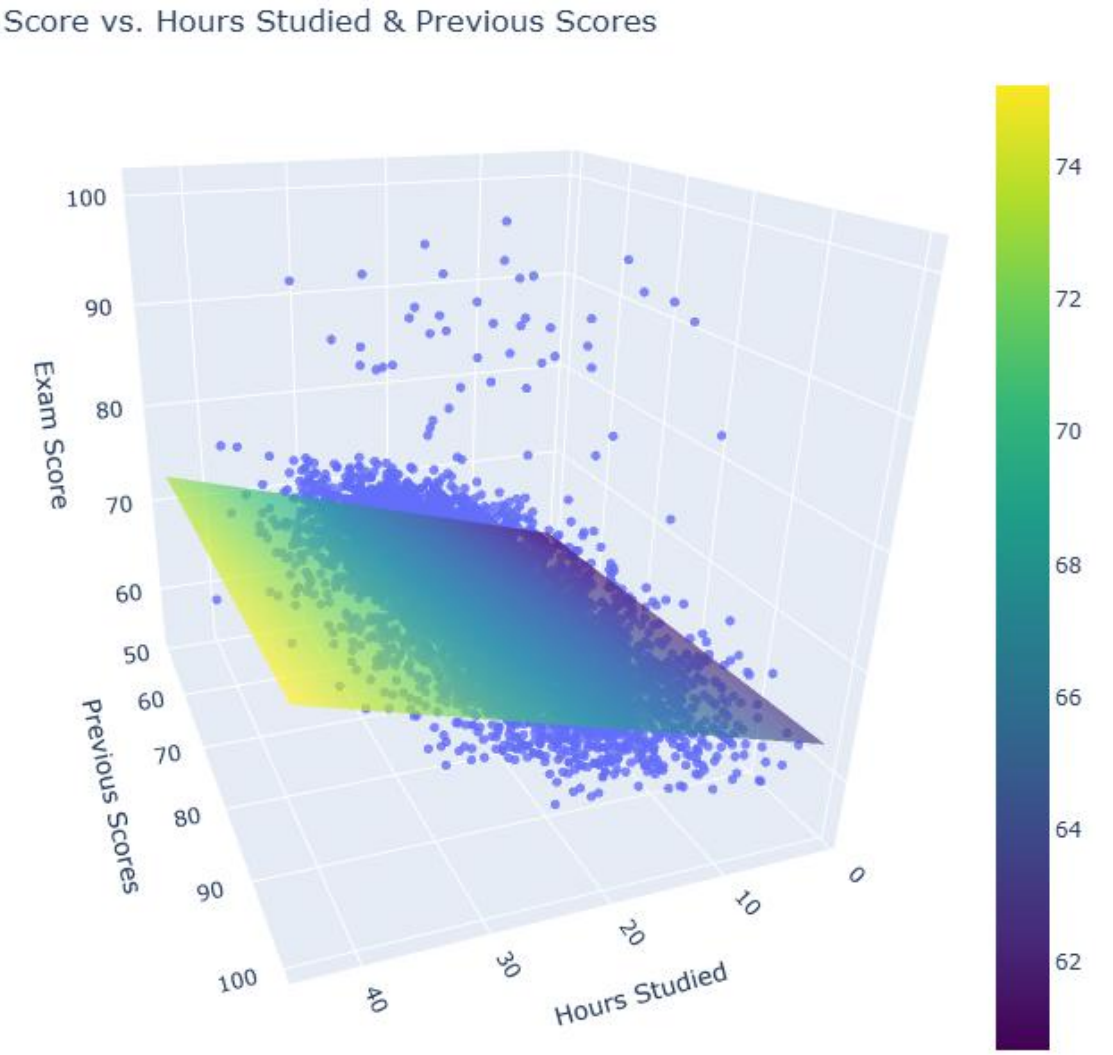
X = GDP, Income comp. resources, Schooling
Y = Life Expectancy

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Multiple Linear Regression

- Predicting Students' Performance based on multiple factors
- x_1 – (input) Hours Studied
- x_2 – (input) Previous Scores
- y – (output) Exam Scores

- We can add more… $x_3, \ldots, x_n$



Multiple Linear Regression (Vectorization)

- Similar (only dimensions change)
- Generalizable
- **n** features, +1 column with ones

$$\underset{(m,n+1)}{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & \dots & x_n^{(2)} \\ \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad \underset{(n+1,1)}{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix} \quad \underset{(m,1)}{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

$$\underset{(m,1)}{y} = \underset{(m,n+1)}{X}\underset{(n+1,1)}{\theta}$$

Multiple Linear Regression: Normal equation (Tim)

```
tim_theta = normal_equation(tim_data_ready, Y)
print(f'theta = {tim_theta}')
```

```
theta = [ 6.95250308e+01  1.58686254e-01 -3.38199840e-02 -4.58041192e-01]
```

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$



```
def RMSE(y_pred, y):
    return np.sqrt(metrics.mean_squared_error(y, y_pred))

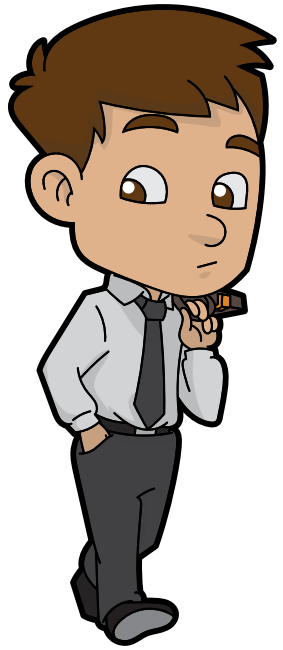
tim_y_pred = np.dot(tim_data_ready, tim_theta)
tim_rmse = RMSE(tim_y_pred, Y)

print("Tim's RMSE: ", tim_rmse)
```

```
Tim's RMSE:  5.784750918898876
```

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons

Multiple Linear Regression: Normal equation (Tim)



```
height = 1.82 #float(input("Please input your height (in meters): "))
weight = 80 #float(input("Please input your weight (in kilograms): "))
bmi = weight / height**2
print("BMI:",bmi)
adult_mortality = 53
hiv = 0.1
life_exp_tim = life_expectancy([bmi,adult_mortality,hiv], tim_theta)
print("Tim's life expectancy is", life_exp_tim , "years.")
```

```
BMI: 24.151672503320853
Tim's life expectancy is 71.5 years.
```

```
height = 1.80 #float(input("Please input your height (in meters): "))
weight = 73 #float(input("Please input your weight (in kilograms): "))
bmi = weight / height**2
print("BMI:",bmi)
adult_mortality = 70
hiv = 0.1
life_exp_tim_tom = life_expectancy([bmi,adult_mortality,hiv], tim_theta)
print("Tom's life expectancy predicted by Tim is", life_exp_tim_tom , "years.")
```

```
BMI: 22.530864197530864
Tom's life expectancy predicted by Tim is 70.7 years.
```

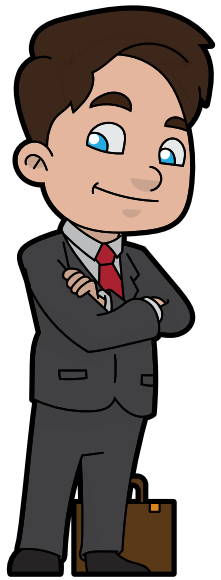
Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons

Multiple Linear Regression: Normal Equation (Tom)

```
tom_theta = normal_equation(tom_data_ready, Y)
print(f'theta = {tom_theta}')
```

```
theta = [4.49681347e+01 8.26882411e-05 1.42744076e+01 1.22489350e+00]
```

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2}$$



```
def RMSE(y_pred, y):
    return np.sqrt(metrics.mean_squared_error(y, y_pred))

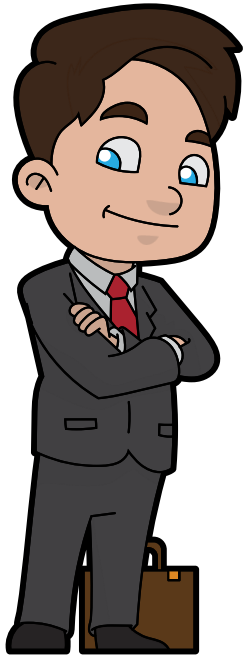
tom_y_pred = np.dot(tom_data_ready, tom_theta)
tom_rmse = RMSE(tom_y_pred, Y)

print("Tom's RMSE: ", tom_rmse)
```

```
Tom's RMSE:  6.288890108092894
```

Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Multiple Linear Regression: Normal equation (Tom)



```
gdp = 8500
income = 0.8
school = 16.5
life_exp_tom = life_expectancy([gdp,income,school], tom_theta)
print("Tom's life expectancy is", life_exp_tom , "years.")
```

```
Tom's life expectancy is 77.3 years.
```

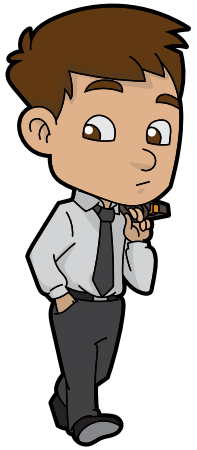
```
gdp = 5000
income = 0.77
school = 15.5
life_exp_tom_tim = life_expectancy([gdp,income,school], tom_theta)
print("Tim's life expectancy predicted by Tom is", life_exp_tom_tim , "years.")
```

```
Tim's life expectancy predicted by Tom is 75.4 years.
```

Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Results part 2





Tim's Life Expectancy predictions:

- Tim – 71.5 years
- Tom – 70.7 years

Tim's RMSE: 5.79

Tom's Life Expectancy predictions:

- Tom – 77.3 years
- Tim – 75.4 years

Tim's RMSE: 6.29



Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Next steps...

- Normalization? Standardization?
- Gradient Descent?
- Artificial Neural Networks?

Normalization vs Standardization

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

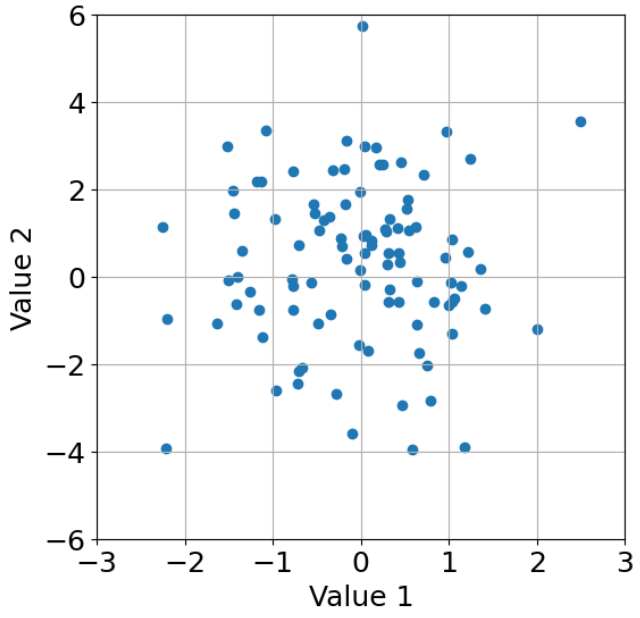
Normalization

Consider a dataset containing two features:

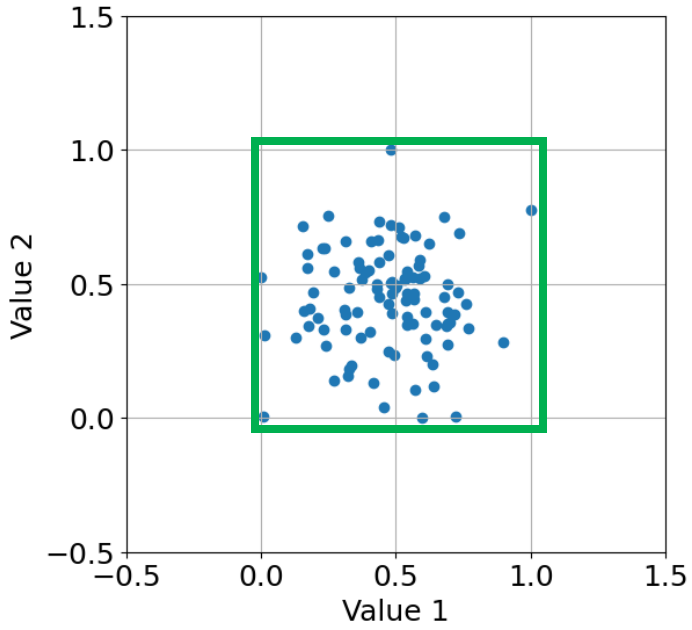
- Age
- Income
- The range of age can go from 0-100 years old, whereas the range of income can go from 0-1,000,000 HUF

- If we would do linear regression, income would have the highest influence because of its large values.
- After normalization the scale of the data is between 0 and 1

Actual Data



Normalized Data



Normalization vs Standardization

$$z = \frac{x - \bar{x}}{\sigma}$$

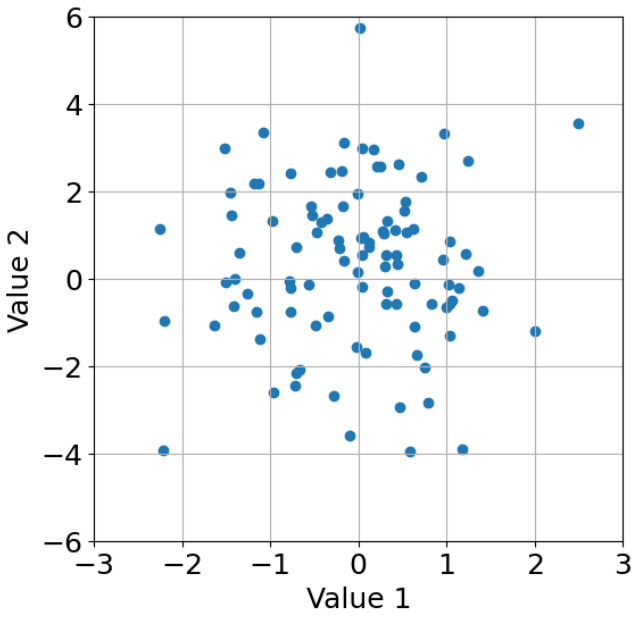
Standardization

Consider a dataset containing two features:

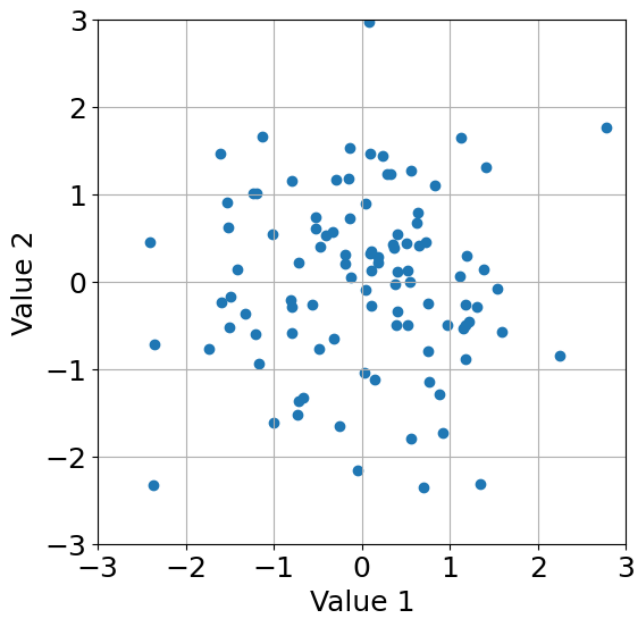
- Weight (Kg)
- Height (cm)

- The range might be similar, but the meaning is different. For example, 100 Kg is different than 100 cm.
- Most importantly, the deviation and the mean are different for both attributes.
- For instance, it is more common to have people with a mean of 80 Kg and a few people with 100 or 200 Kg, than having people with a mean of 80 cm tall (the majority will have more 160 cm).
- After standardization the scale is unbounded

Actual Data



Standardized Data

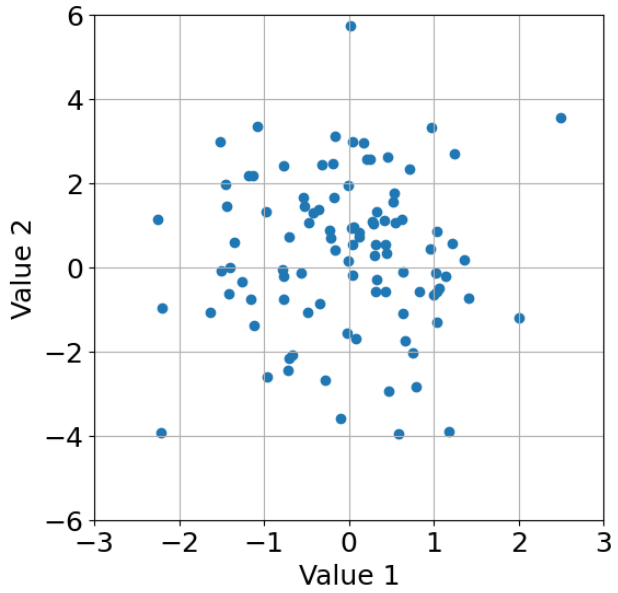


Normalization vs Standardization

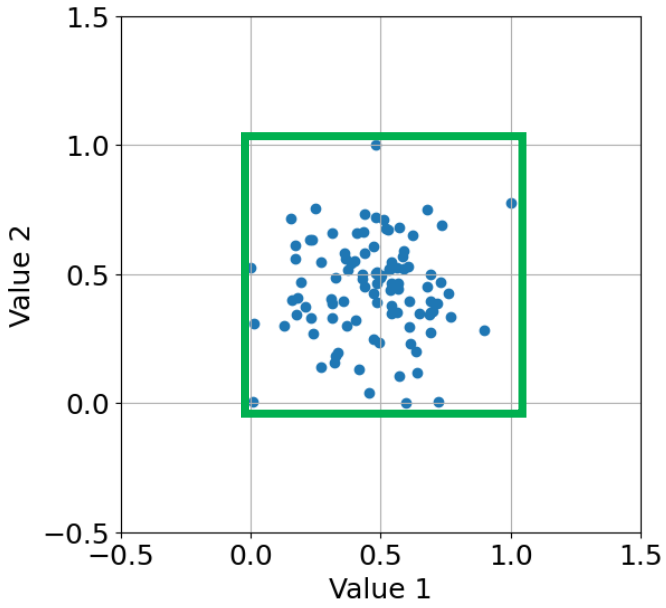
$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

$$z = \frac{x - \bar{x}}{\sigma}$$

Actual Data

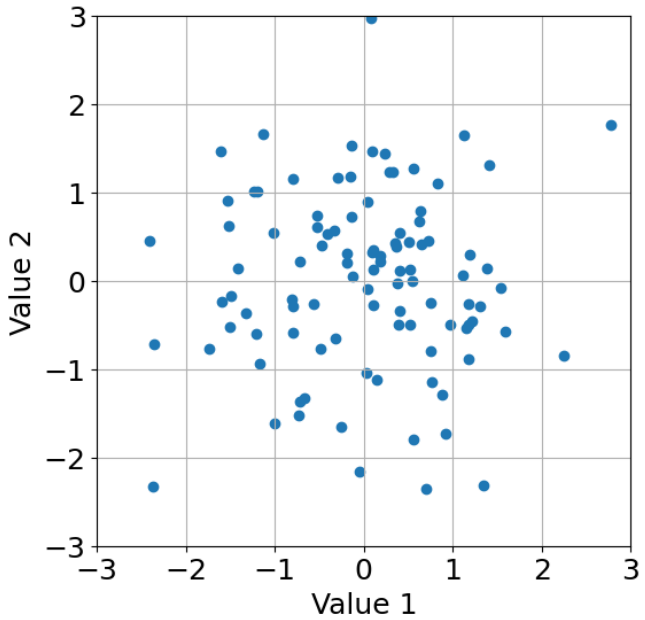


Normalized Data



Normalization is recommended when your data has varying scales.

Standardized Data



Standardization is recommended when your data has different units.

Applications

Object Detection (Bounding Box Regression)



Image Source: Auto-Zooming CNN-Based Framework for Real-Time Pedestrian Detection in Outdoor Surveillance Videos - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Bounding-box-post-processing-The-boxes-with-green-and-red-colors-show-the-pedestrians_fig2_334780733, under CC BY 4.0 from Alfasly, Saghir & Liu, Beibei & Hu, Yongjian & Wang, Yufei & Li, Chang-Tsun. (2019). Auto-Zooming CNN-Based Framework for Real-Time Pedestrian Detection in Outdoor Surveillance Videos. IEEE Access. 7. 105816 - 105826. 10.1109/ACCESS.2019.2931915.

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 2

Linear Regression & Artificial Neural Networks

1 Linear Regression

2 Artificial Neural Networks

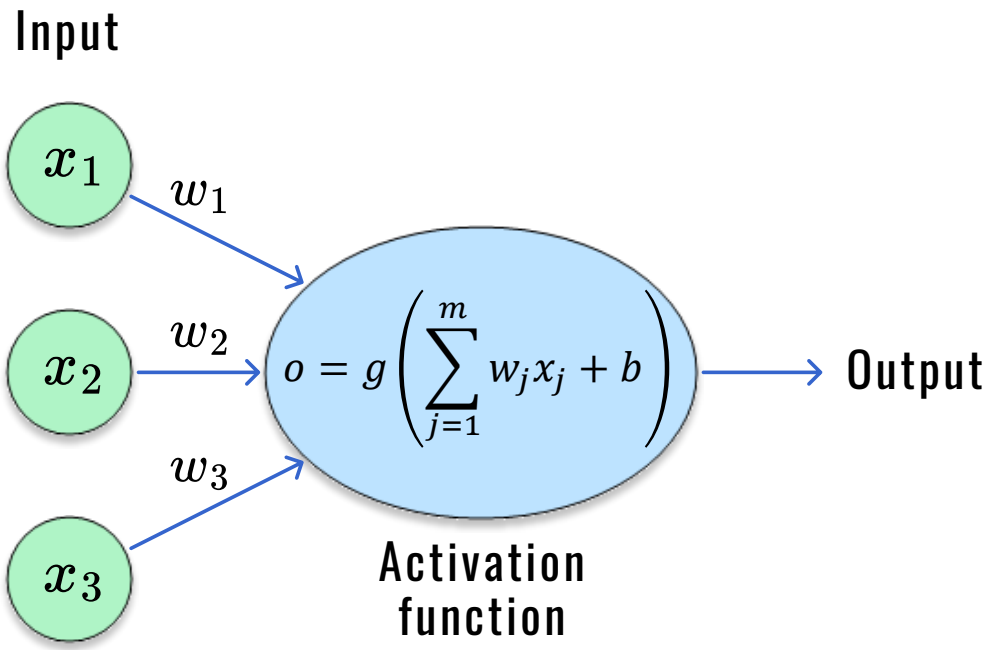
3 Backpropagation

How do ANNs work?

x – the inputs
w – weight parameters we will train
b – bias parameter we will train
g – nonlinear activation function
(ReLU, Softmax, ...)
o – the output

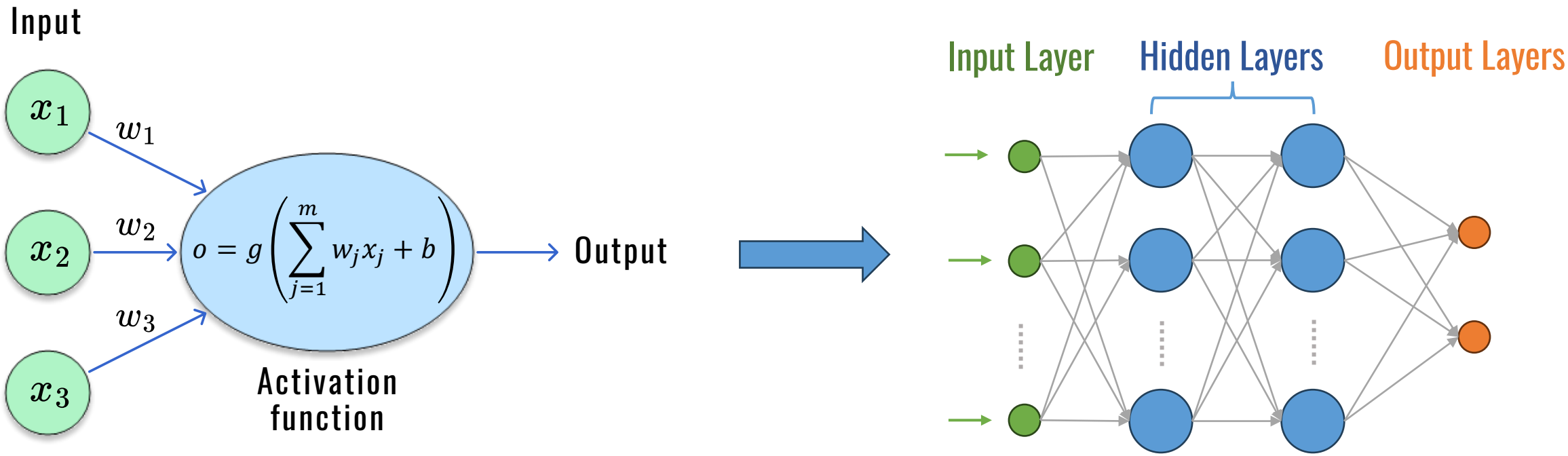
One neuron with m inputs does the following:

$$o = g\left(\sum_{j=1}^{m} w_j x_j + b\right)$$



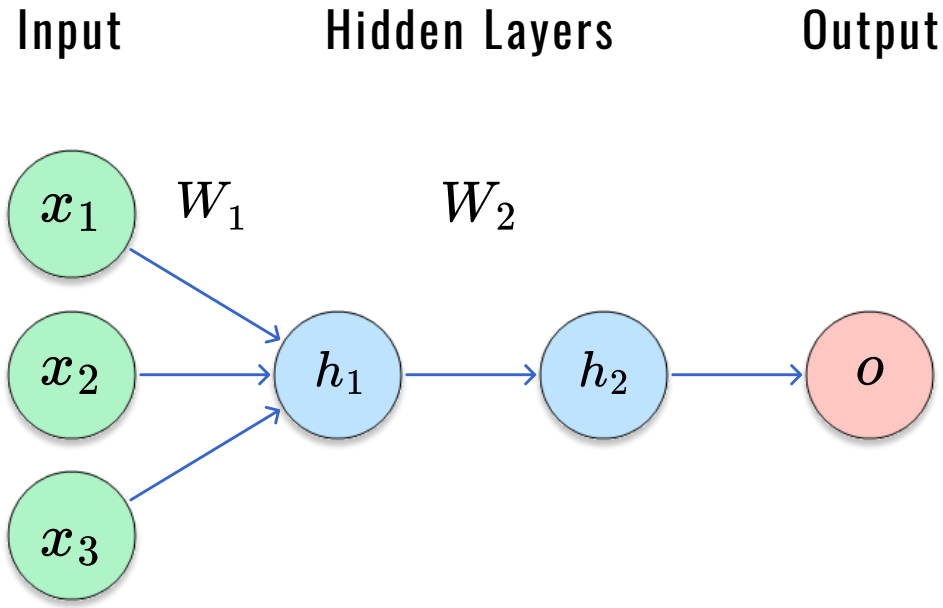
Deep Neural Networks

- Neural networks are built up from neurons, that have inputs and outputs
- Neurons are organised into layers
- Layers refine the output of the previous layers



Why do we need nonlinear activation functions?

What happens if we create a deep neural network with 2 hidden layers without an activation function?



(vectorized)

$$g(x) = x$$

$$h_1(x) = g\left(W_1 x + b_1\right) = W_1 x + b_1$$

$$h_2(x) = g\left(W_2 x + b_2\right) = W_2 x + b_2$$

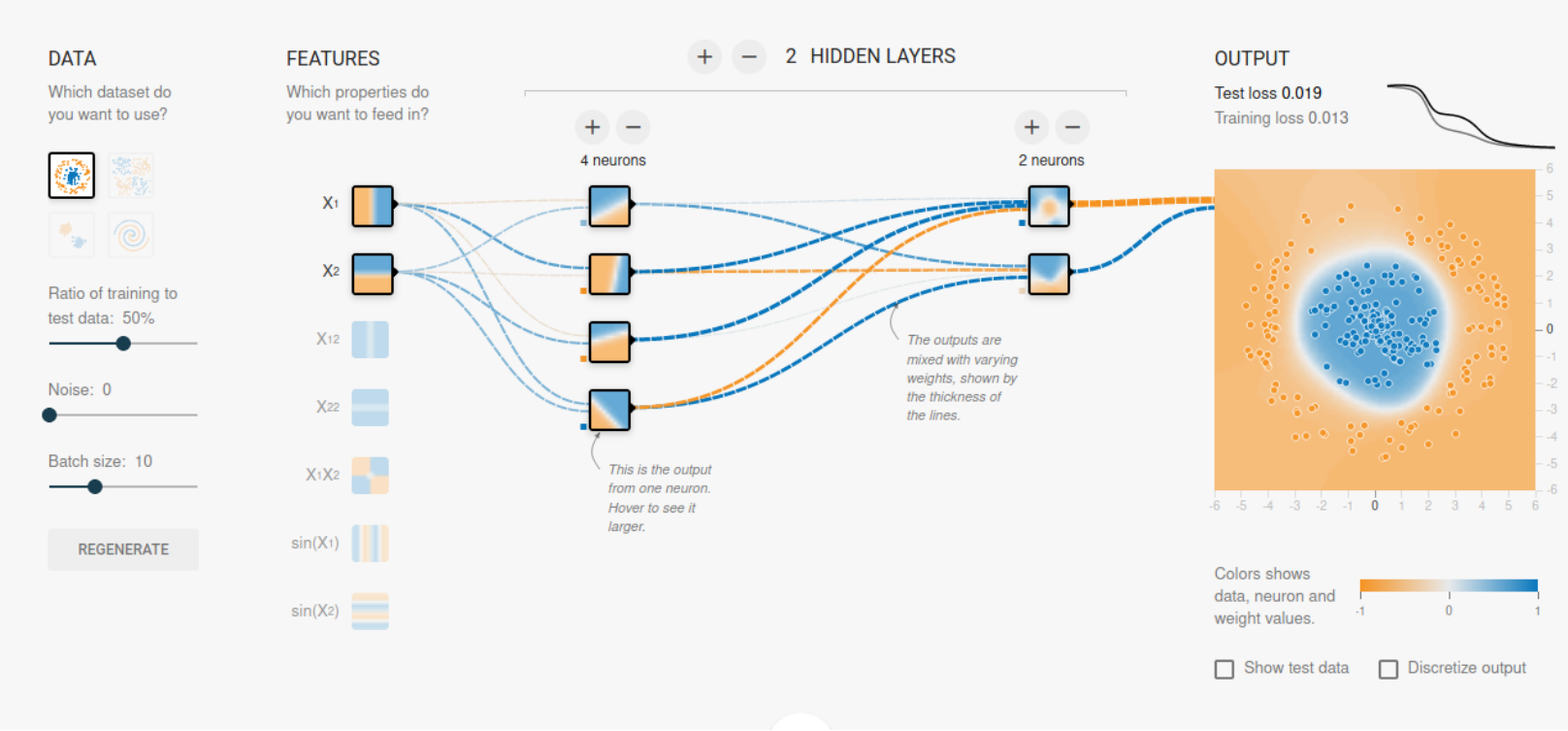
$$\begin{aligned} o = h_2(h_1(x)) &= h_2(W_1 x + b_1) = \\ &= W_2(W_1 x + b_1) + b_2 = \\ &= W_2 W_1 x + W_2 b_1 + b_2 \end{aligned}$$

- Stacking more layers does not make our model more expressive
- It just learns linear or affine transformations
- Nonlinear activations are required to learn complex “shapes”

Why do we need nonlinear activation functions?

In short, it allows the model to learn **complex patterns** and **relationships**

Demo: https://playground.tensorflow.org



This Photo by Unknown Author is licensed under CC BY-SA

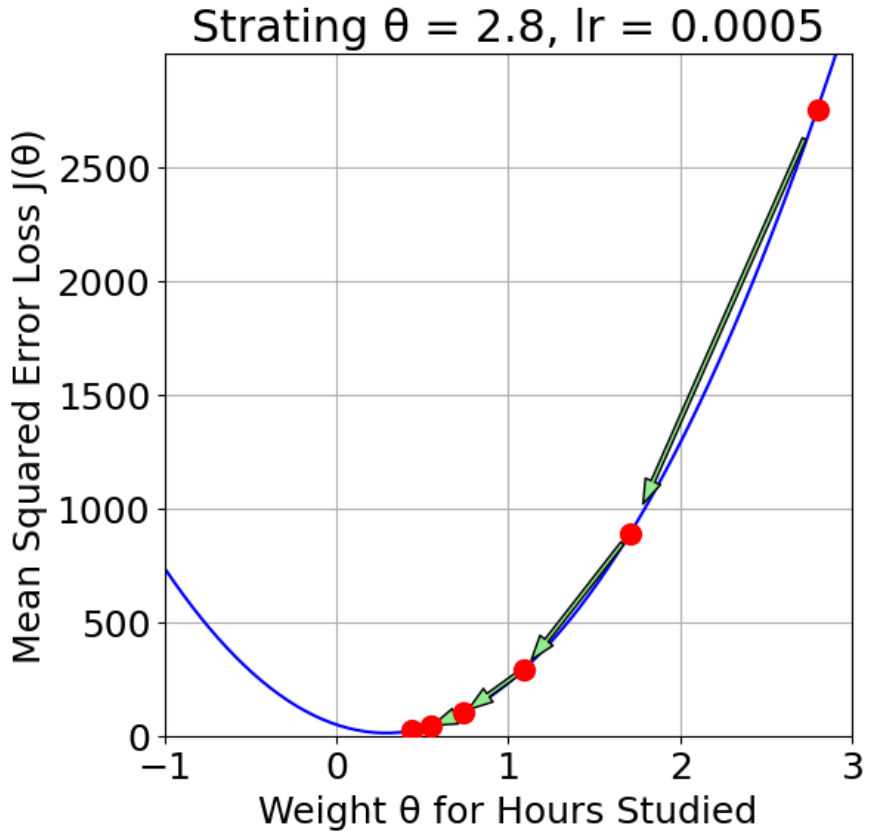
Numerical Solution: Gradient Descent

1. Select θ randomly
2. Update θ :

$$\theta_t = \theta_{t-1} - \alpha \nabla J(\theta_{t-1})$$

   - where $J(\theta)$ is the average error

3. Iterate until it converges



$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i\theta_i - y_i)^2$$

Numerical Solution: Gradient Descent

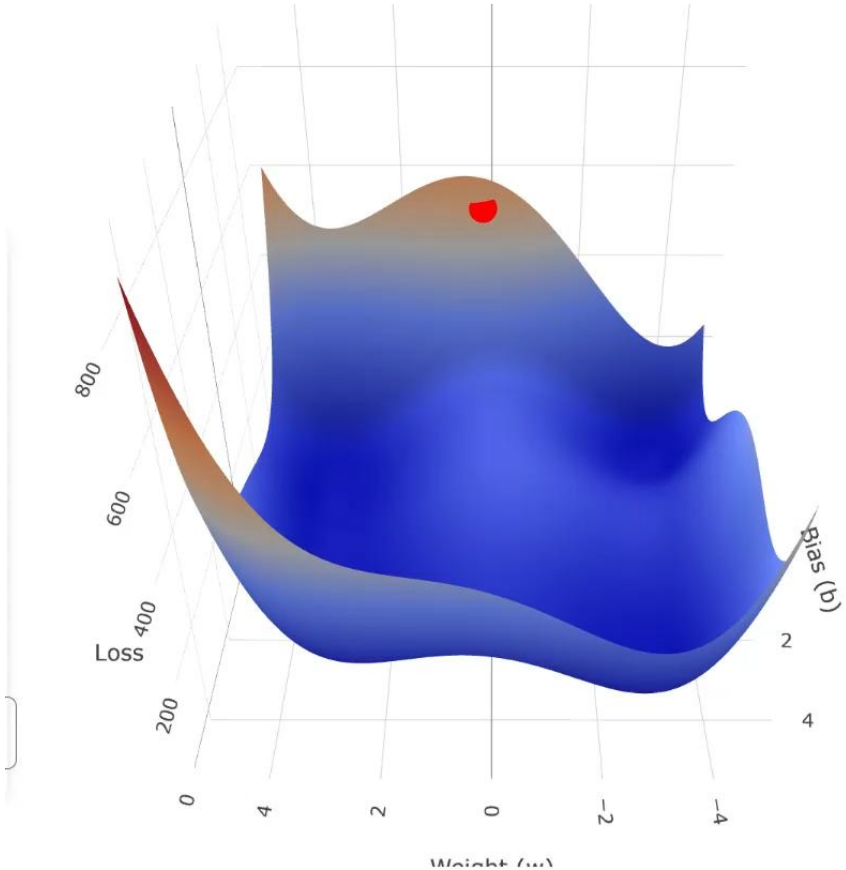
1. Select θ randomly
2. Calculate $\hat{y}$ for all training examples
3. Calculate the gradient of the total loss:

$$\nabla J(\theta_{t-1}) = \nabla \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}_i, y_i)$$

4. Update :

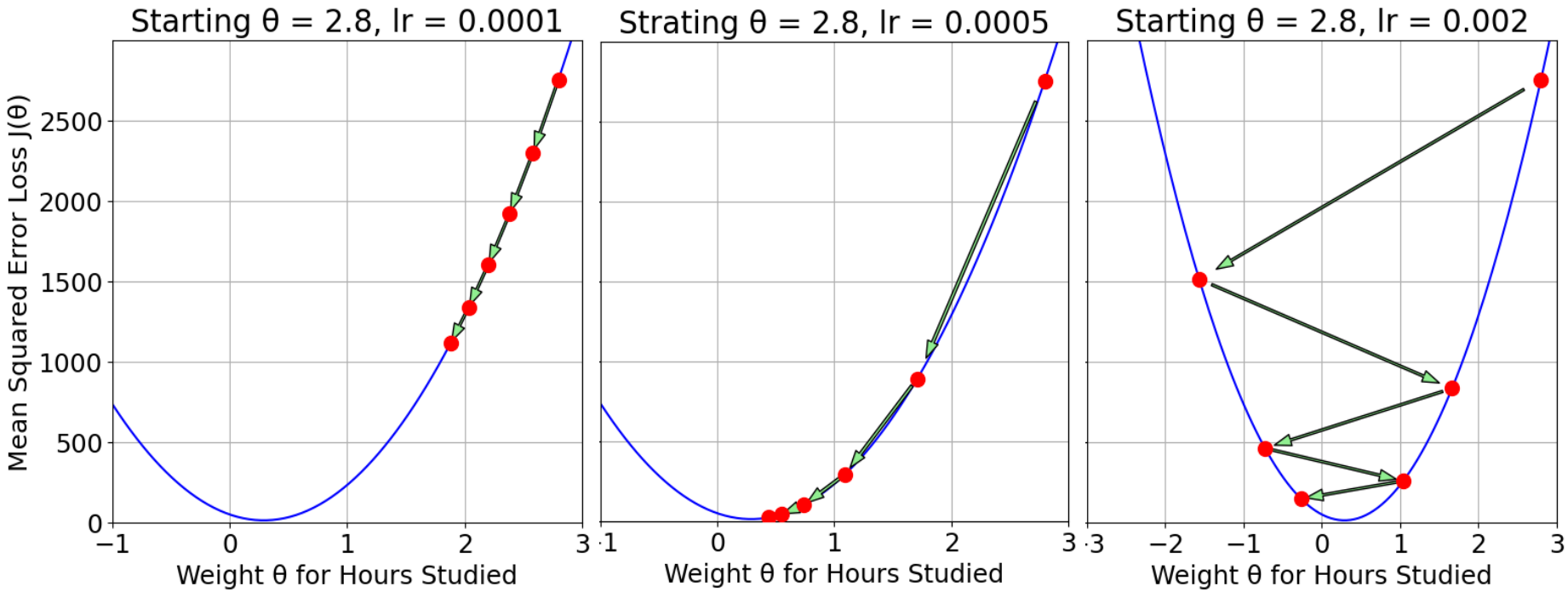
$$\theta_t = \theta_{t-1} - \alpha \nabla J(\theta_{t-1})$$

5. Iterate until it converges



Gradient Descent: Learning Rate

Effect of selected learning rate on the Gradient Descent updates.



A small learning rate requires many updates to before reaching the minimum point

The optimal learning rate quickly reaches the minimum point

Too large learning rates causes huge updates which leads to divergence, or not reaching the minimum point

Gradient Descent

- It looks easy if we visualize it in 2D but in reality the surface of the loss function can be more complex
- It can contain multiple local minima which can cause the optimization to stuck in suboptimal states
- Visualization on a more complex surface: https://losslandscape.com/explorer

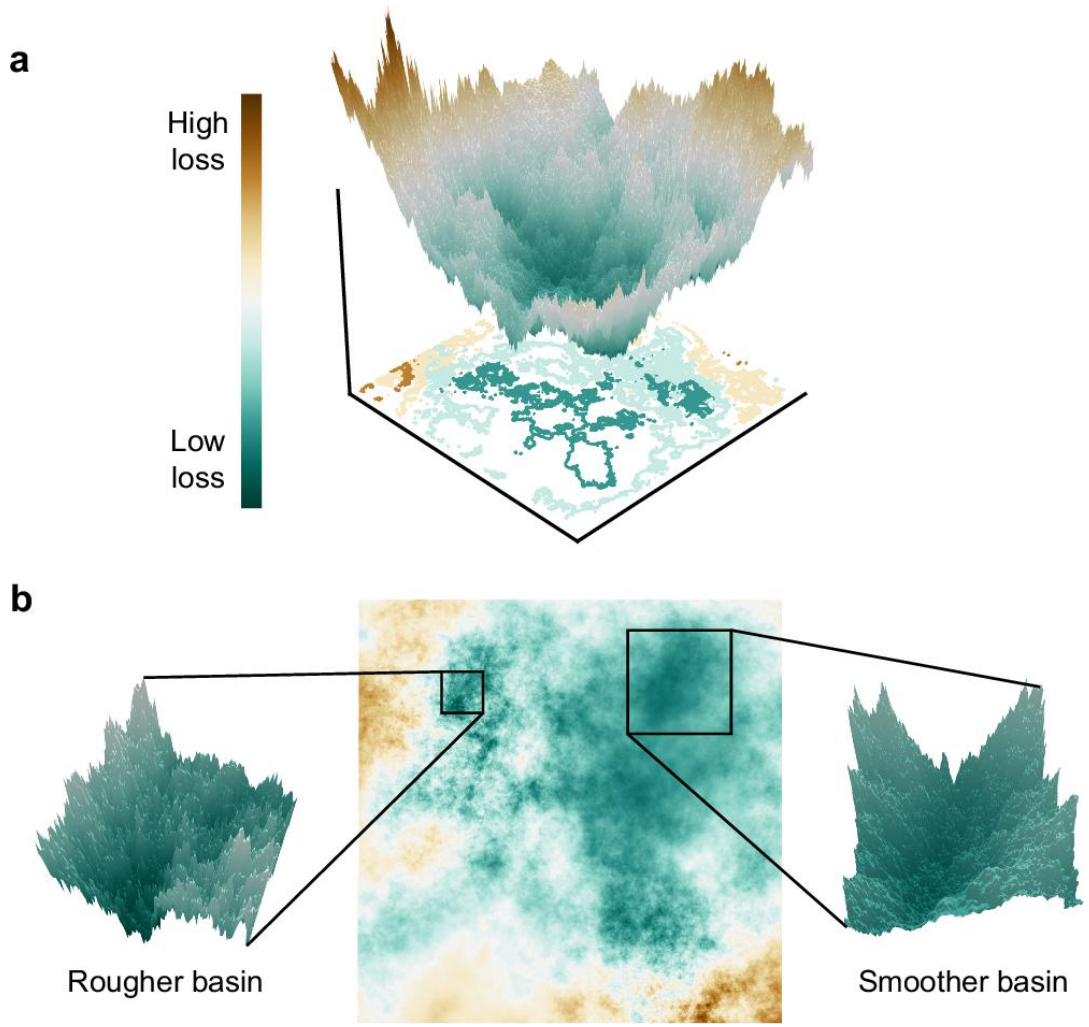


Image Source: Fig. 1: A loss landscape with multifractal structure, Ly, A., Gong, P. Optimization on multifractal loss landscapes explains a diverse range of geometrical and dynamical properties of deep learning. *Nat Commun* **16**, 3252 (2025), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/ . https://doi.org/10.1038/s41467-025-58532-9,

Stochastic Gradient Descent (SGD)

- Processing the whole dataset in one batch is straightforward, may not fit into the memory
- Go through the training data in randomly selected batches
- A single pass through on the training data called **epoch**
- batch size can be:
  - Dataset size (**full-batch**)
  - 1, 2, 4 , 8, 16, … (**mini-batch**)

“SGD is like a drunkard trying to find his way home.” — Geoffrey Hinton

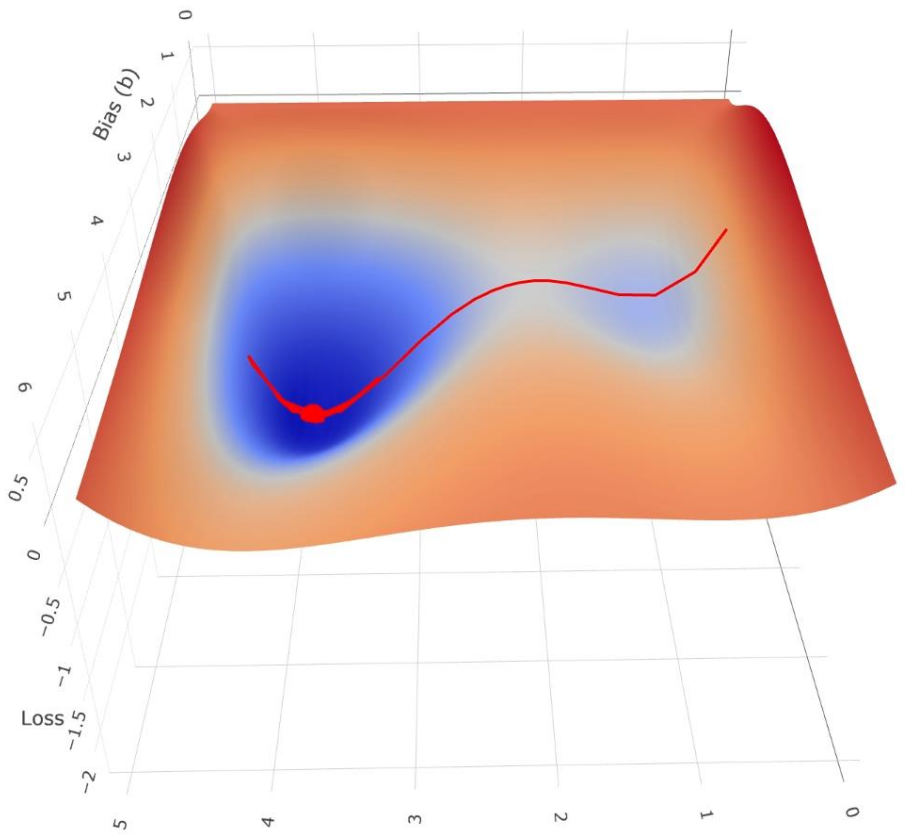


Image Source: Gradient Descent Visualiser — Yash Arote, MIT License https://opensource.org/license/mit , https://github.com/YashArote/descent-visualisers

SGD with Momentum

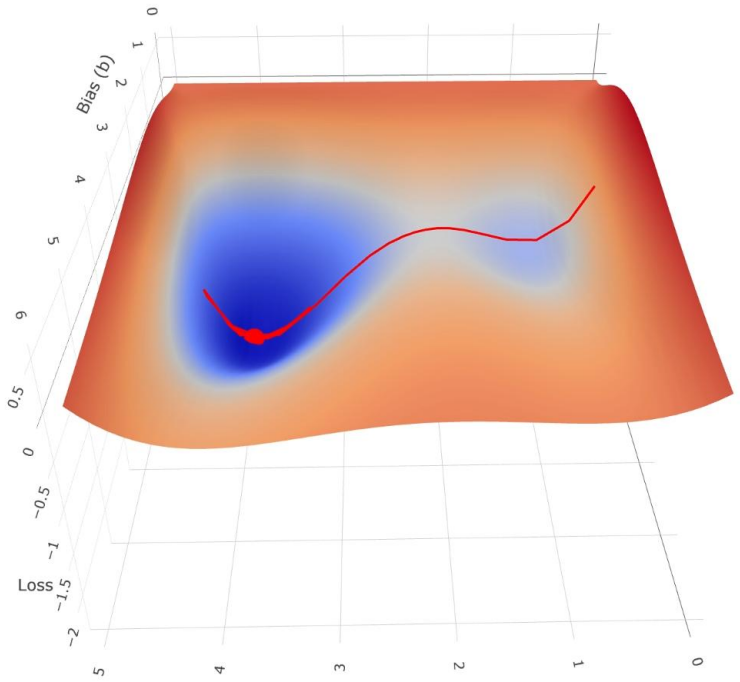
- To get a smoother trajectory and reduce oscillations in valleys lets introduce **momentum**
- β controls the smoothing (weighted average)

4. Update:

$$\mathbf{m}_t = \beta \cdot \mathbf{m}_{t-1} + (1-\beta)\nabla J(\theta_{t-1})$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \mathbf{m}_{t-1}$$

SGD with Momentum



Gradient Descent

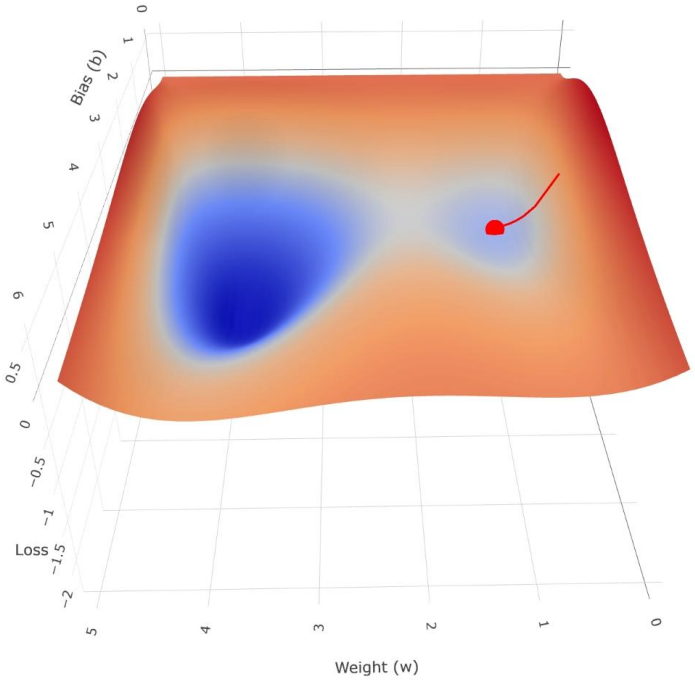


Image Source: Gradient Descent Visualiser — Yash Arote, MIT License https://opensource.org/license/mit , https://github.com/YashArote/descent-visualisers

Adaptive Moment Estimation (Adam)

- Gradient descent makes large adjustments to parameters with large gradients
- **Solution**: Normalize the gradients so we move fix distances (learning rate)

4. Update:

$$\mathbf{m}_t = \beta_0 \cdot \mathbf{m}_{t-1} + (1 - \beta_0)\nabla J(\theta_{t-1})$$

$$\mathbf{v}_t = \beta_1 \cdot \mathbf{v}_{t-1} + (1 - \beta_1)\nabla J(\theta_{t-1})^2$$

Measuring gradient and applying momentum

$$\hat{\mathbf{m}}_t = \mathbf{m}_t / (1 - \beta_1^t)$$

$$\hat{\mathbf{v}}_t = \mathbf{v}_t / (1 - \beta_2^t)$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{\mathbf{m}}_{t-1} / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon)$$

Normalize

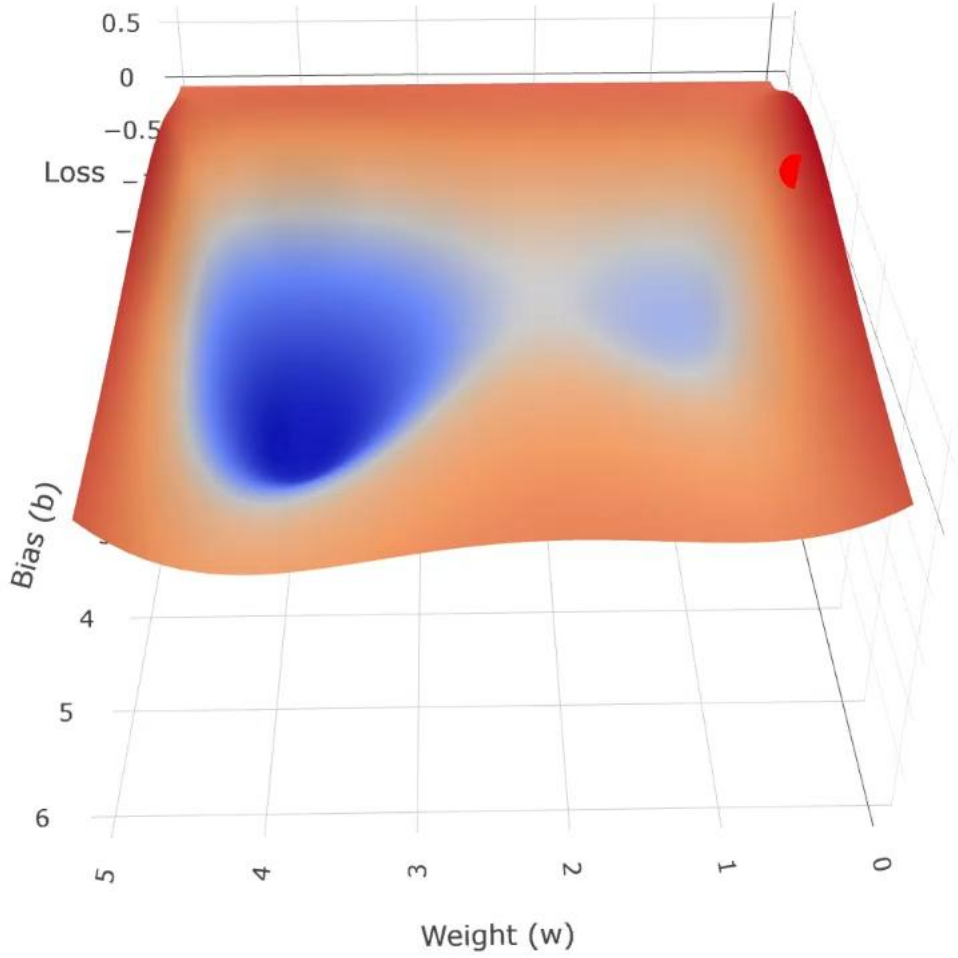


Image Source: Gradient Descent Visualiser — Yash Arote, MIT License https://opensource.org/license/mit , https://github.com/YashArote/descent-visualisers

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 2

Linear Regression & Artificial Neural Networks

1 Linear Regression

2 Artificial Neural Networks

3 Backpropagation

Backpropagation algorithm

Forward Propagation:

- The Neural Network tries to make its best guess about the correct output
- We run the input data through its functions

Backward Propagation:

- The Neural Network adjusts its parameters based on the error in its guess

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Chain Rule

The chain rule is a formula that expresses the derivative of the composition of two differentiable functions:

$$h(x) = f\big(g(x)\big), \qquad h'(x) = f'\big(g(x)\big)g'(x)$$

Or, equivalently

$$h' = (f \circ g)' = (f' \circ g) \cdot g'$$

The chain rule may also be expressed in Leibniz's notation:

$$\frac{\partial h(x)}{\partial x} = \frac{\partial h(x)}{\partial q} \cdot \frac{\partial q}{\partial z} \cdot \frac{\partial z}{\partial x}, \qquad q \coloneqq f(z), z \coloneqq g(x)$$

Training our Regression Model

We have our model: $\hat{y} = \theta_0 + \theta_1 x = \theta x$

Loss function: $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_i x_i - y_i\right)^2$

Update rule: $\theta_t = \theta_{t-1} - \alpha \nabla J(\theta_{t-1})$

$$\nabla J(\theta_{t-1}) = \nabla \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i)$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm

Pytorch's autograd keeps a record of data and all executed operations in a directed acyclic graph (**DAG**)

We create a DAG of our executed operations, where the arrows are pointing in the direction of the **forward pass**

- The leaf nodes represents our parameters
- The nodes represents the operations
- The root node represents our loss function

The backward pass kicks off when we call **.backward()** on our root node (loss)

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

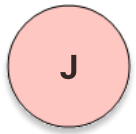
1. Let's create the directed acyclic graph of the operations

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

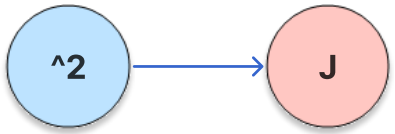


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

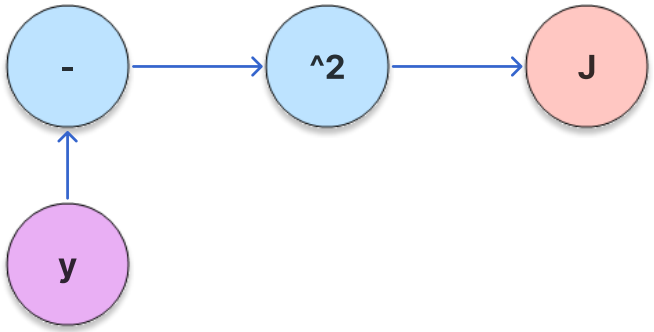


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

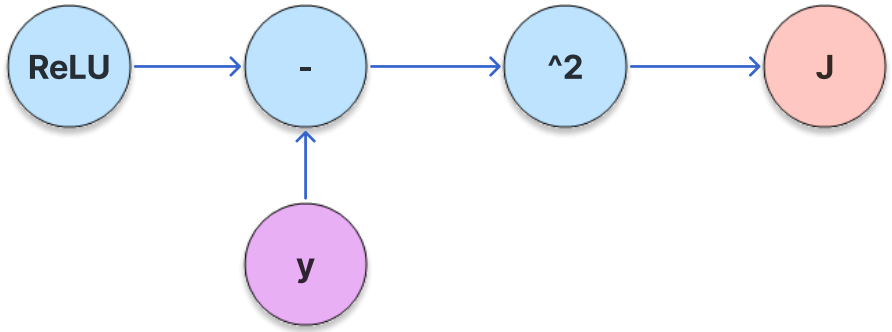


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

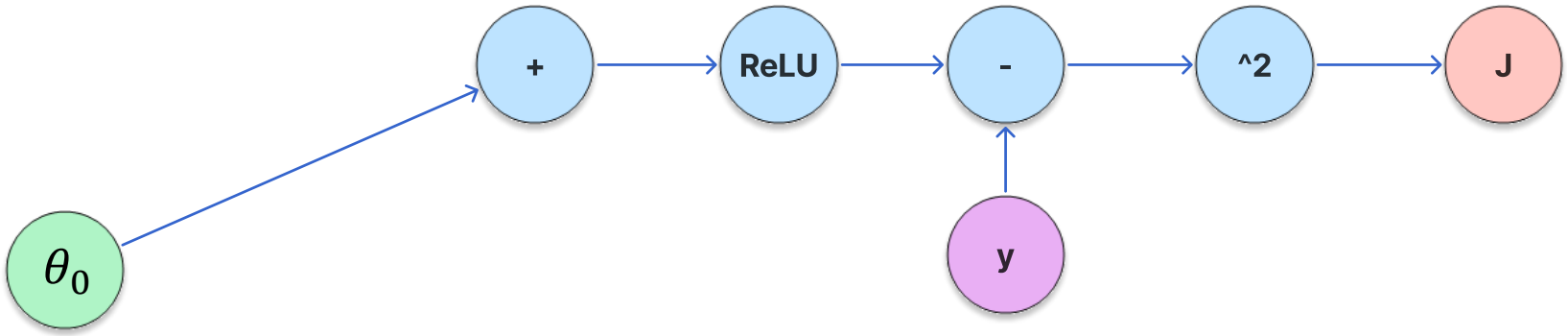


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

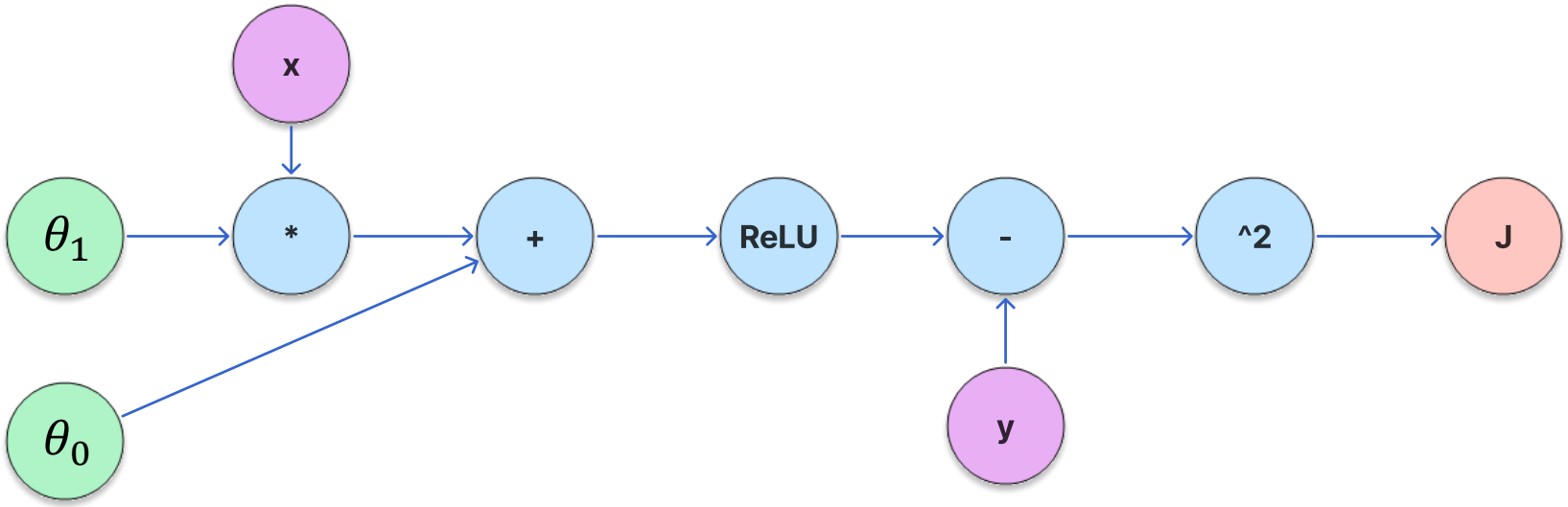


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

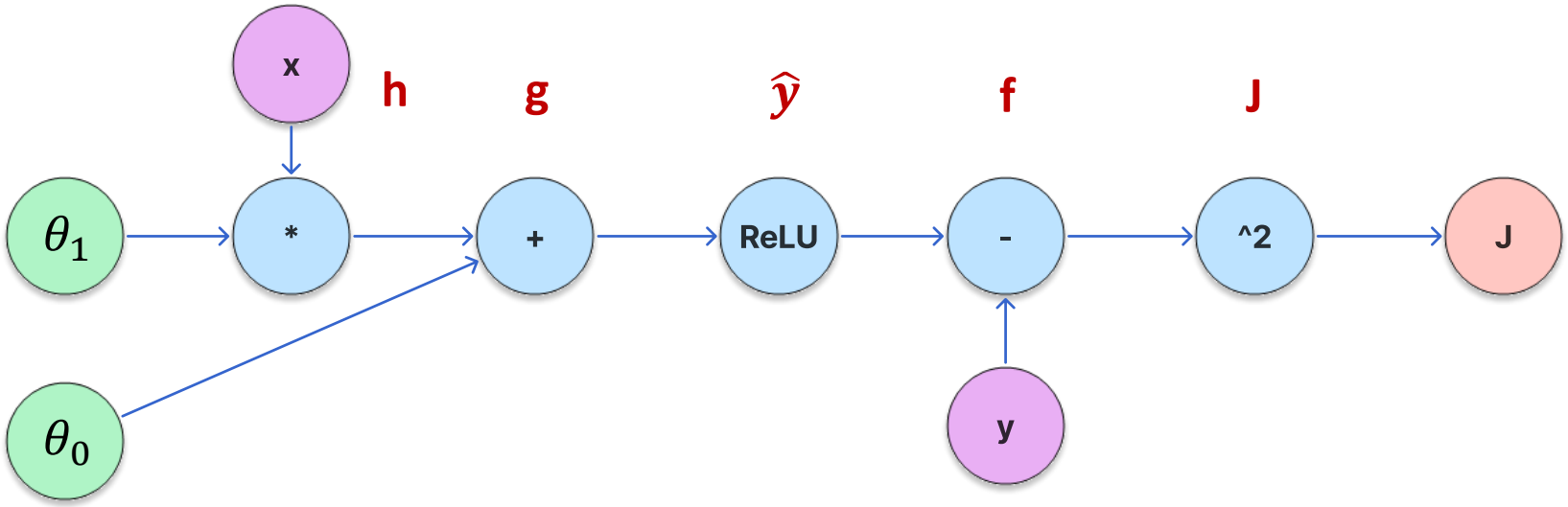


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

2. Name these nodes



https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

2. Name these nodes

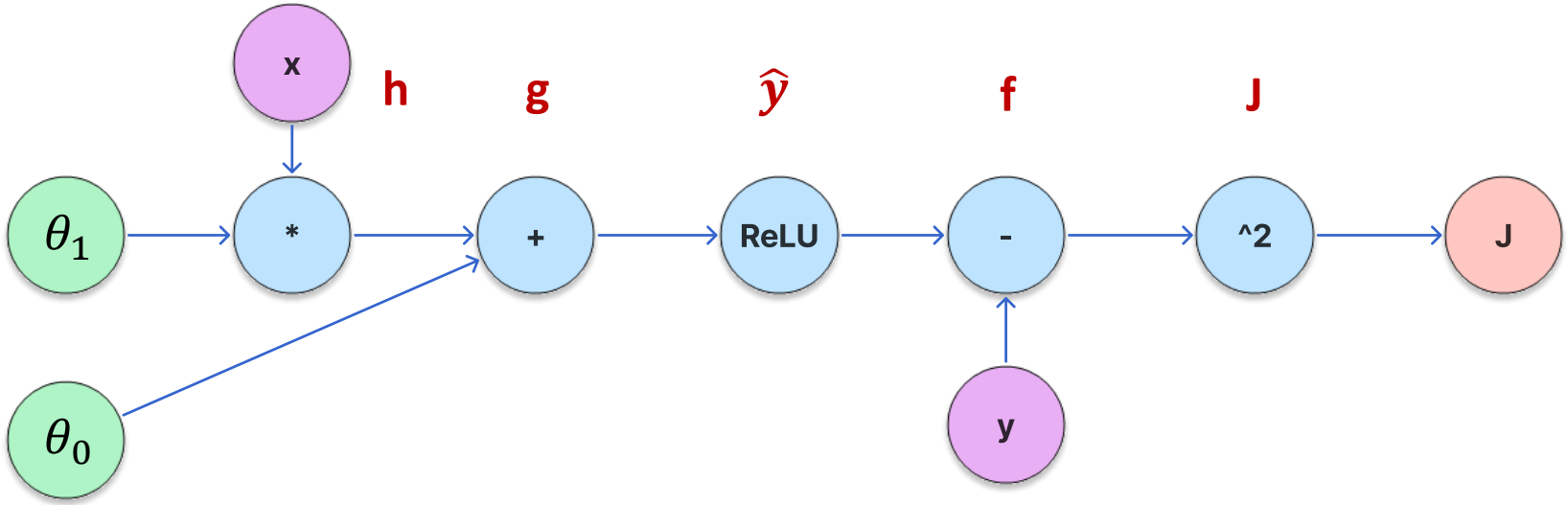
$$J = f^2$$

$$f = \hat{y} - y$$

$$\hat{y} = ReLU(g)$$

$$g = h + \theta_0$$

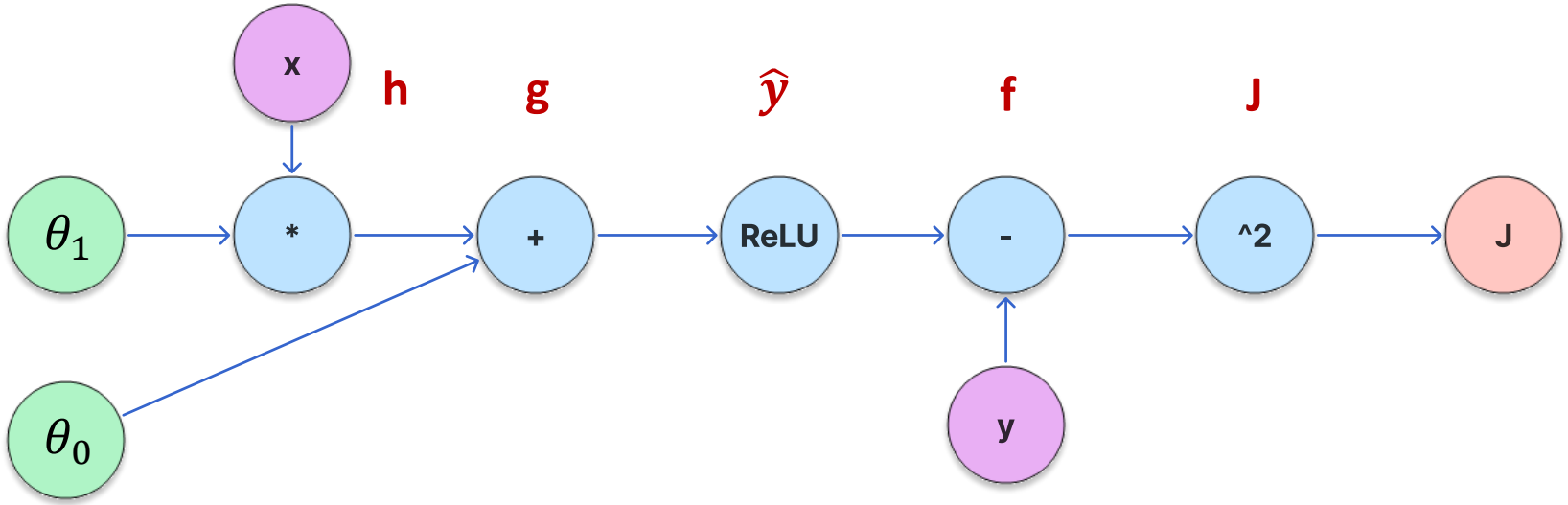
$$h = \theta_1 x$$



https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

3. Symbolic differentiation



$$J = f^2$$

$$f = \hat{y} - y$$

$$\hat{y} = ReLU(g)$$

$$g = h + \theta_0$$

$$h = \theta_1 x$$

$$\frac{\partial J}{\partial \theta_0} = ?$$

$$\frac{\partial J}{\partial \theta_1} = ?$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

3. Symbolic differentiation

$$\frac{\partial J}{\partial J} = 1$$

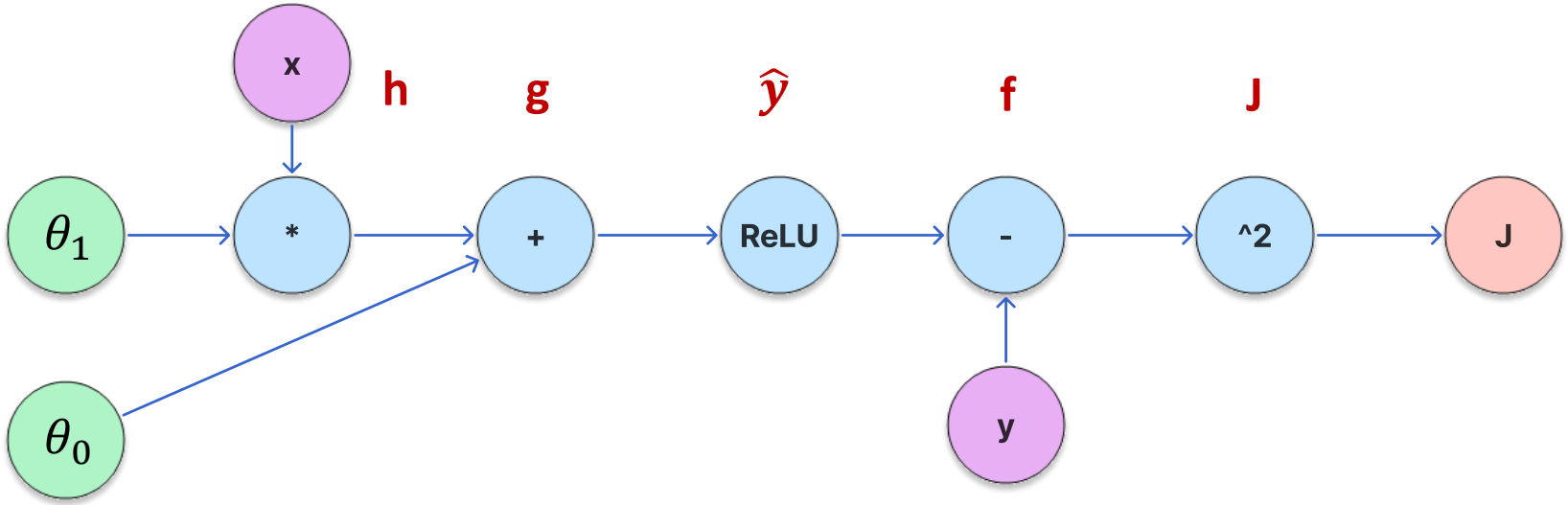
$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \quad \frac{\partial g}{\partial \theta_0} = 1 \qquad \frac{\partial J}{\partial \theta_0} = ?$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \quad \frac{\partial h}{\partial \theta_1} = x \qquad \frac{\partial J}{\partial \theta_1} = ?$$



https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

3. Symbolic differentiation

$$\frac{\partial J}{\partial J} = 1$$

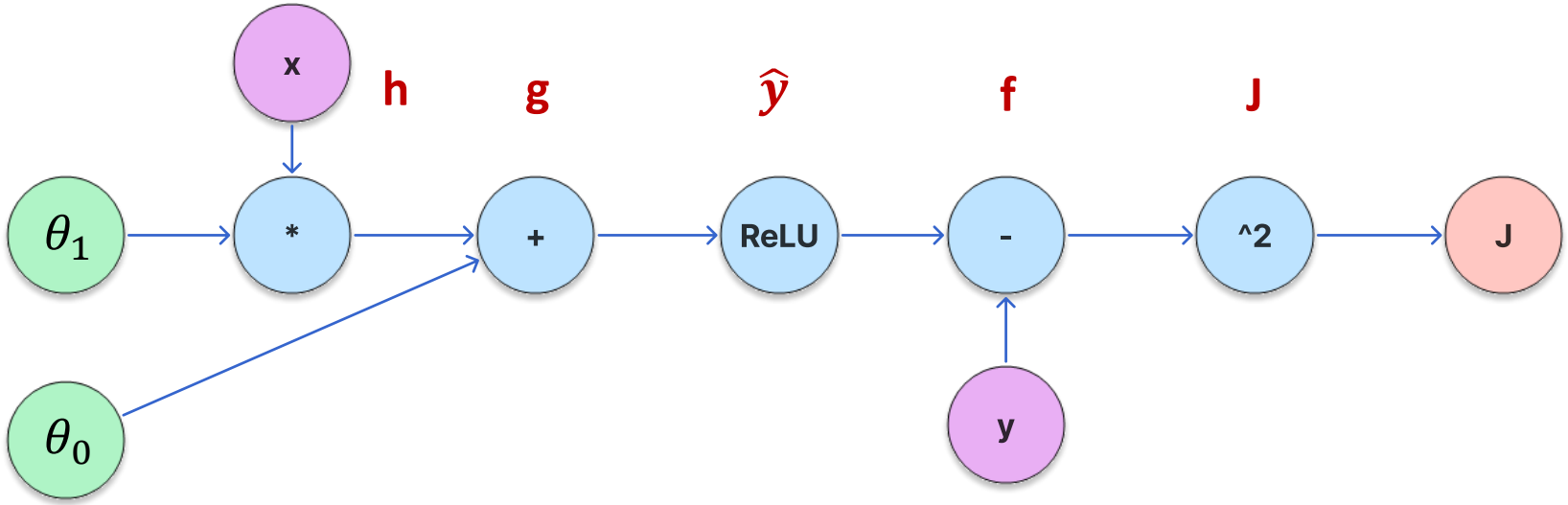
$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$



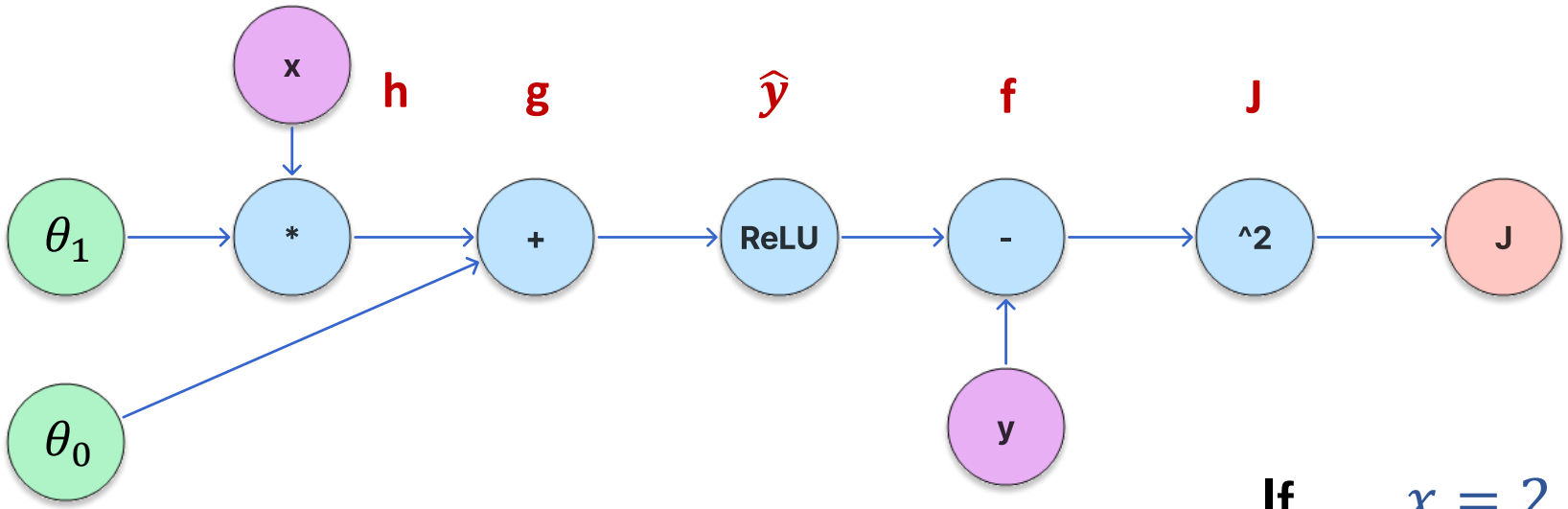
$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – forward pass



$$\frac{\partial J}{\partial J} = 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$

$y = 1$

$\theta_0 = 5$

$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – forward pass

$$\frac{\partial J}{\partial J} = 1$$

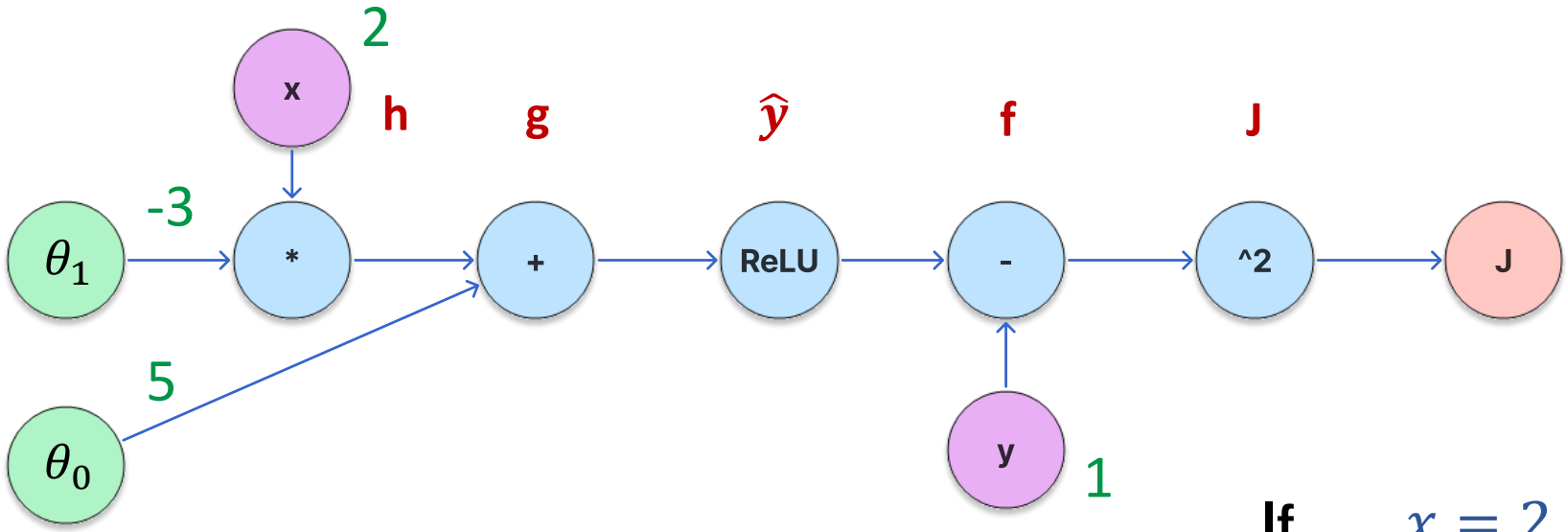
$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$



If $x = 2$

$y = 1$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

$\theta_0 = 5$

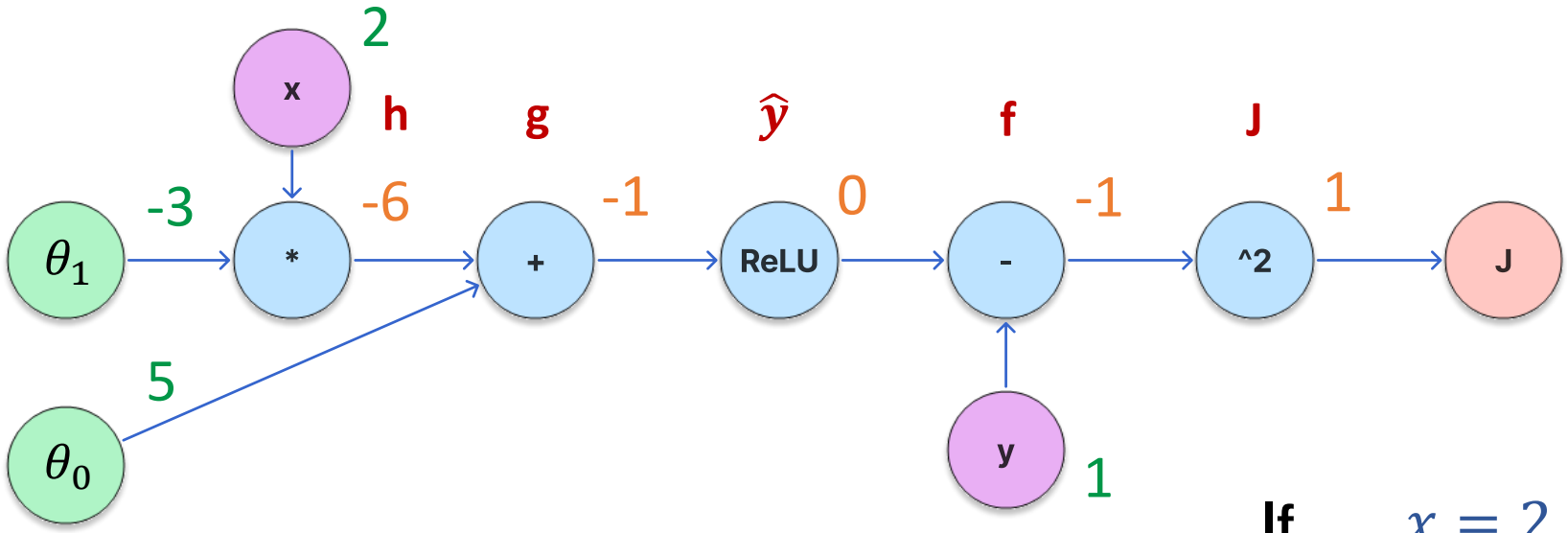
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – forward pass



$$\frac{\partial J}{\partial J} = 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$

$y = 1$

$\theta_0 = 5$

$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – backward pass

$$\frac{\partial J}{\partial J} = 1$$

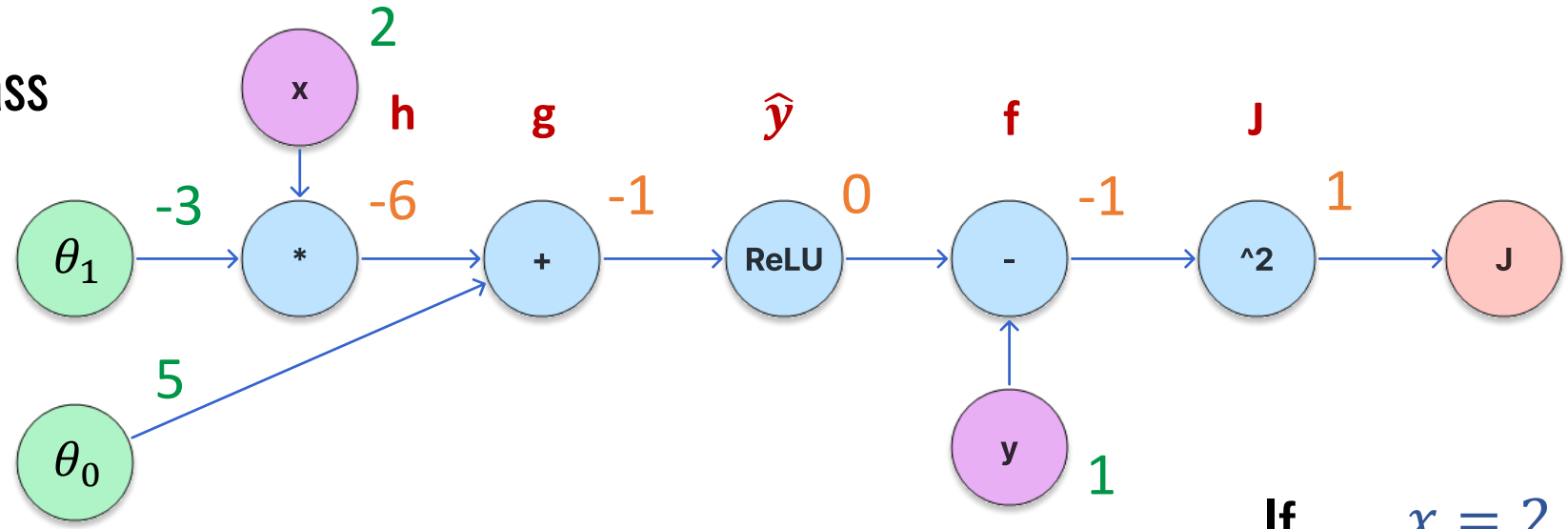
$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$

$y = 1$

$\theta_0 = 5$

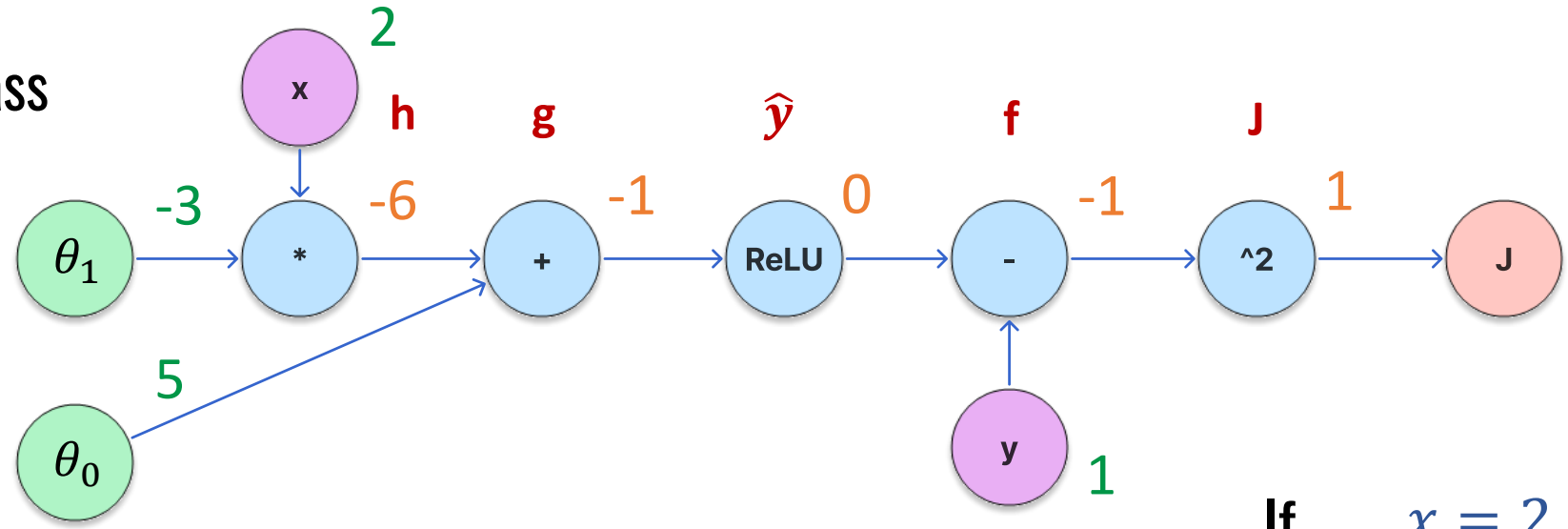
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – backward pass



$$\frac{\partial J}{\partial J} = 1 \quad 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f \quad -2$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases} \quad 0$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

-3 2

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$
$y = 1$

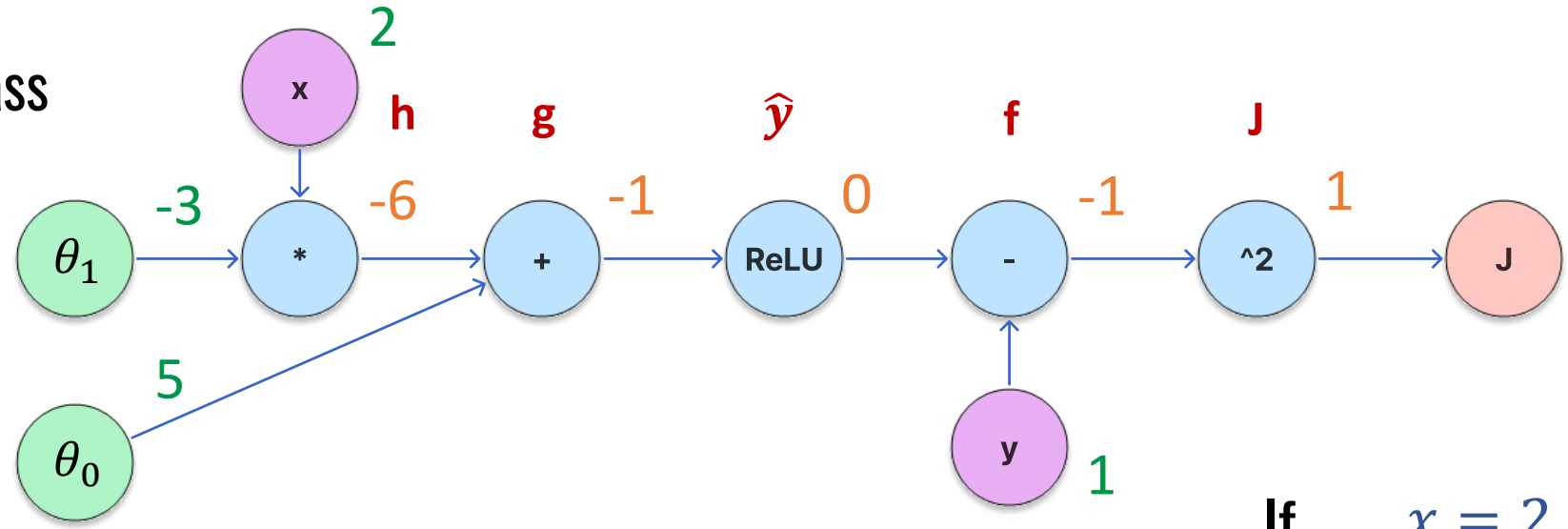
$\theta_0 = 5$
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – backward pass



$\frac{\partial J}{\partial J} = 1$ 1

$J = f^2$ $\frac{\partial J}{\partial f} = 2f$ -2

$f = \hat{y} - y$ $\frac{\partial f}{\partial \hat{y}} = 1$

$\hat{y} = ReLU(g)$ $\frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$ 0

$g = h + \theta_0$ $\frac{\partial g}{\partial h} = 1$ $\frac{\partial g}{\partial \theta_0} = 1$

$h = \theta_1 x$ $\frac{\partial h}{\partial x} = \theta_1$ $\frac{\partial h}{\partial \theta_1} = x$

-3 2

1 -2 1 0 1

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} = 0$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} = 0$$

1 -2 1 0 1 2

If $x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

5. Update weights

What happened? Did we learn?

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} = 0$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} = 0$$

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = 5 - 0.1 \cdot 0$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = -3 - 0.1 \cdot 0$$

$x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

3. Symbolic differentiation

$$\frac{\partial J}{\partial J} = 1$$

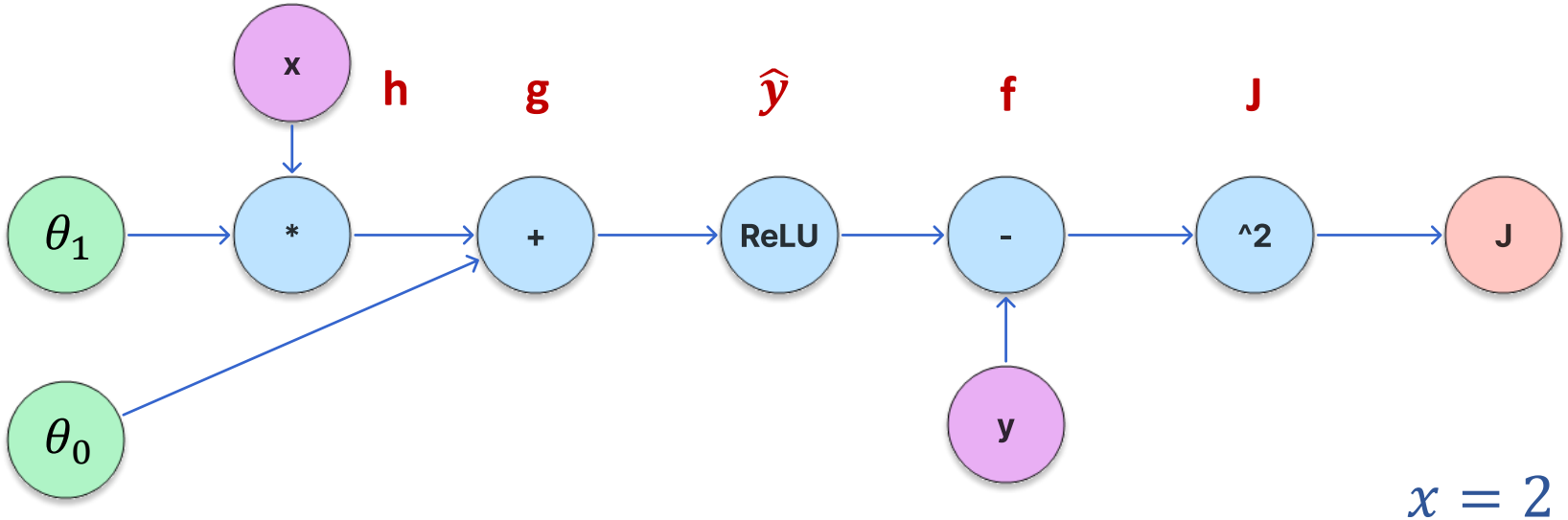
$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

$x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – forward pass

$$\frac{\partial J}{\partial J} = 1$$

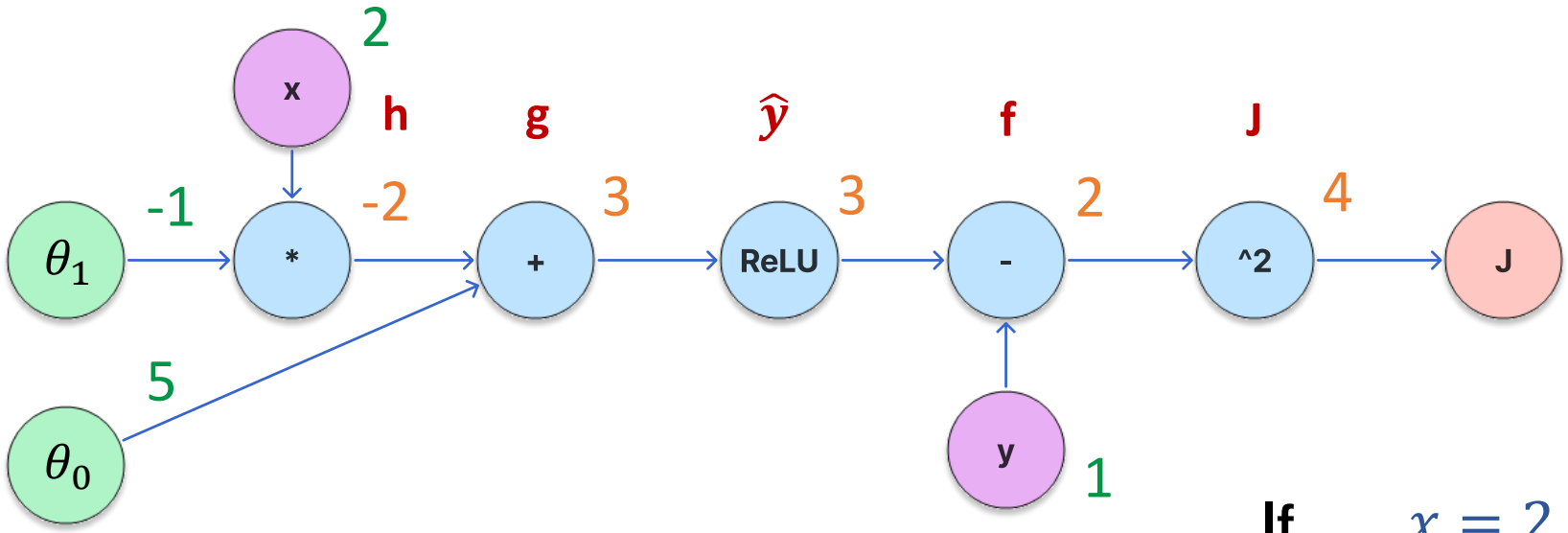
$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$

$y = 1$

$\theta_0 = 5$

$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – backward pass

$$\frac{\partial J}{\partial J} = 1 \quad 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f \quad 4$$

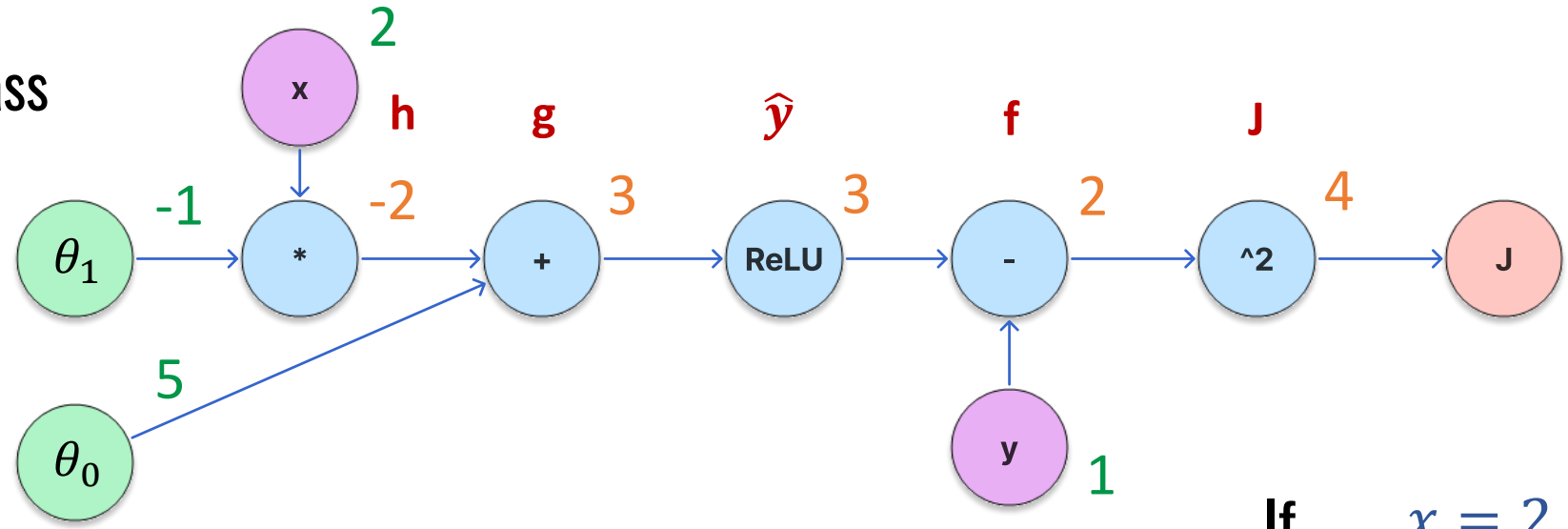
$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases} \quad 1$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

-1 2



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$

$y = 1$

$\theta_0 = 5$

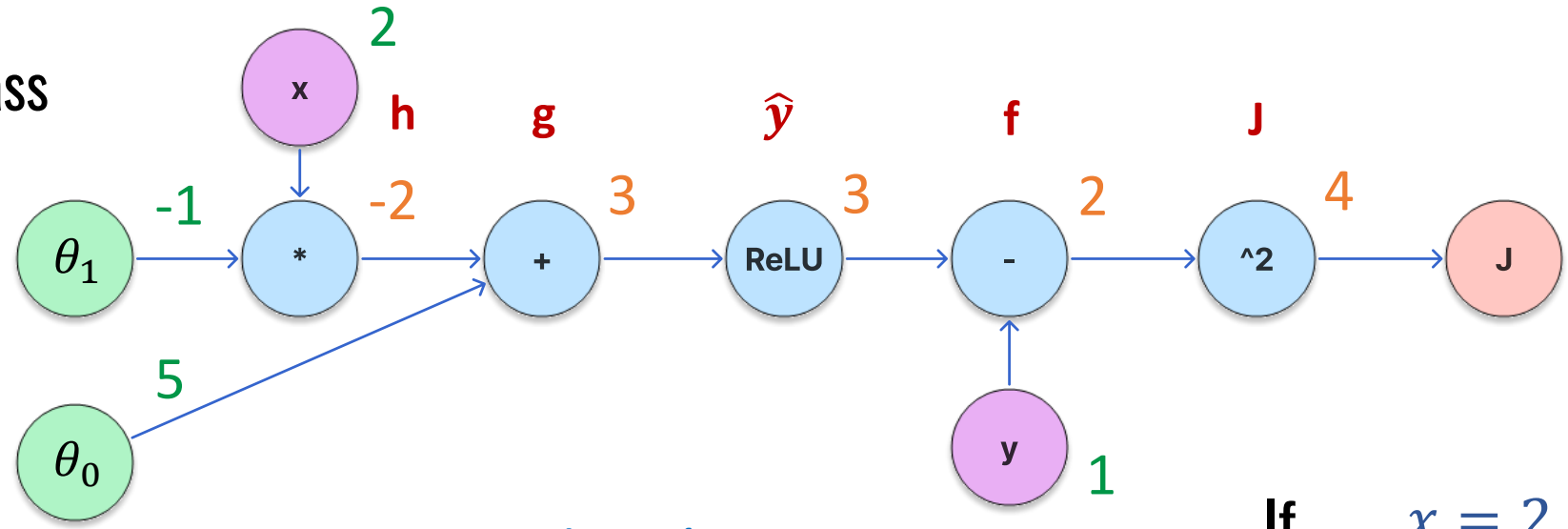
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – backward pass



$$\frac{\partial J}{\partial J} = 1 \quad 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f \quad 4$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases} \quad 1$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \quad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \quad \frac{\partial h}{\partial \theta_1} = x$$

-1 2

If $x = 2$

$y = 1$

1 4 1 1 1

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

1 4 1 1 1 2

$\theta_0 = 5$

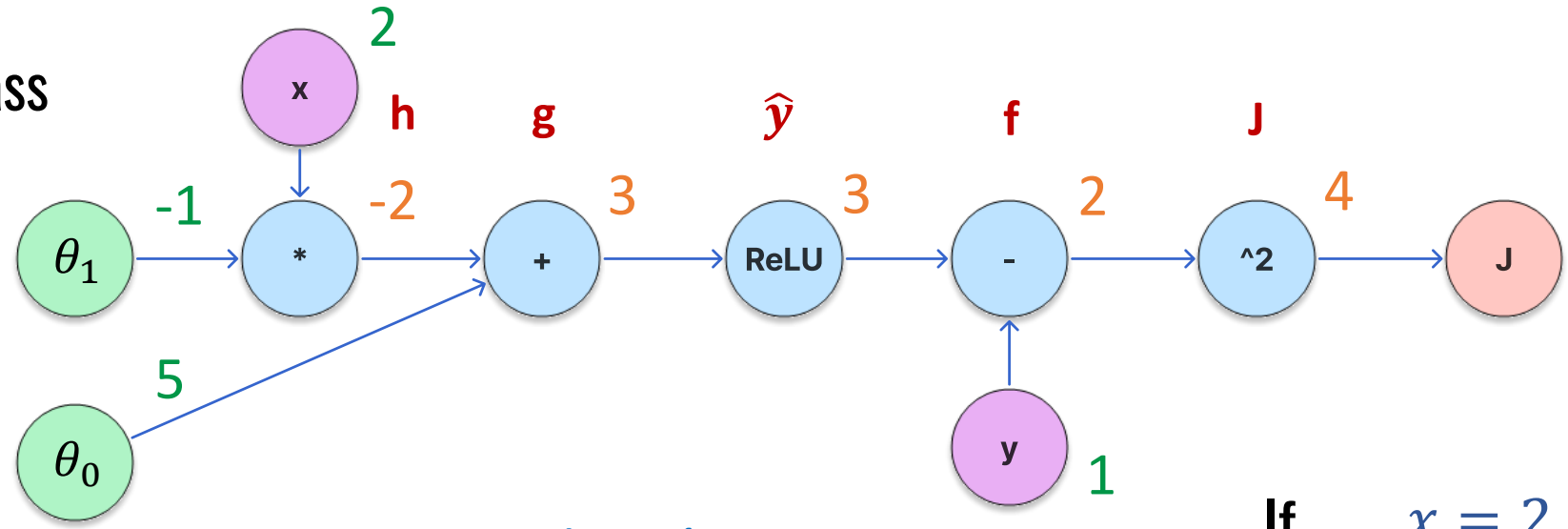
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – backward pass



$$\frac{\partial J}{\partial J} = 1 \quad 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f \quad 4$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases} \quad 1$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \quad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \quad \frac{\partial h}{\partial \theta_1} = x$$

-1 2

If $x = 2$, $y = 1$

$\theta_0 = 5$

$\theta_1 = -1$

$\alpha = 0.1$

1 4 1 1 1

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} = 4$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} = 8$$

1 4 1 1 1 2

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

5. Update weights

What happened? Did we learn?

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} = 4$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} = 8$$

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = 5 - 0.1 \cdot 4 = 4.6$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = -1 - 0.1 \cdot 8 = 1.8$$

$x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Summary

In Artificial Neural Network we introduced non-linearity, we don't have an exact solution, so we use the iterative approach for optimizing:

- **Gradient Descent:** we calculate the loss on the whole dataset and adjust the parameters
- **Stochastic Gradient Descent**: we split our dataset into random mini-batches (and train, test, validation sets), we calculate the loss on these mini-batches, update based on these
- **Stochastic Gradient Descent with Momentum**: we smooth out the updates by applying a weighted average of the gradients
- **Adaptive Momentum**: we normalize the gradients, so the updates are uniform

Backpropagation algorithm:

- Calculating the gradient for our deep learning model.
- Forward pass – prediction – loss – backward pass.
- Calculating symbolic differentiation.

Resources

Datasets used in this lecture:

- Lai Nguyen (lainguyn123), Student Performance Factors, Kaggle, CC0 1.0 Universal. https://www.kaggle.com/datasets/lainguyn123/student-performance-factors
- Lasha Gochiashvili, Life Expectancy (WHO) Fixed, Kaggle, CC0: Public Domain, https://www.kaggle.com/datasets/lashagoch/life-expectancy-who-updated

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning

Image Sources

- Unless otherwise noted, the figures were created by the instructor
- Python code examples were created by the instructor and rendered with Carbon.
- Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
- Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons
- Dancing Robot: Gif robot.git, Meks01, CC BY-SA 4.0, via Wikimedia Commons
- Image generated with ChatGPT Images (OpenAI), 2026, https://chatgpt.com/
- Linear least squares example2.svg, Krishnavedala, CC BY-SA 3.0, via Wikimedia Commons, https://commons.wikimedia.org/wiki/File:Linear_least_squares_example2.svg
- Correlation examples.png from Imagecreator at English Wikipedia, Public domain, via Wikimedia Commons
- Pedestrian detection figure: Alfasly, Saghir, Liu, Beibei, Hu, Yongjian, Wang, Yufei & Li, Chang-Tsun (2019), Auto-Zooming CNN-Based Framework for Real-Time Pedestrian Detection in Outdoor Surveillance Videos, IEEE Access 7, 105816–105826, CC BY 4.0, via ResearchGate
- Fig. 1: A loss landscape with multifractal structure, Ly, A., Gong, P. Optimization on multifractal loss landscapes explains a diverse range of geometrical and dynamical properties of deep learning. *Nat Commun* **16**, 3252 (2025), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/ . https://doi.org/10.1038/s41467-025-58532-9,

Image Sources

- Gradient Descent Visualiser — Yash Arote, MIT License https://opensource.org/license/mit , https://github.com/YashArote/descent-visualisers
- Diagrams adapted from the PyTorch Autograd tutorial, http://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Legal Notice

descent-visualisers — Yash Arote
Source: https://github.com/YashArote/descent-visualisers
License: MIT License

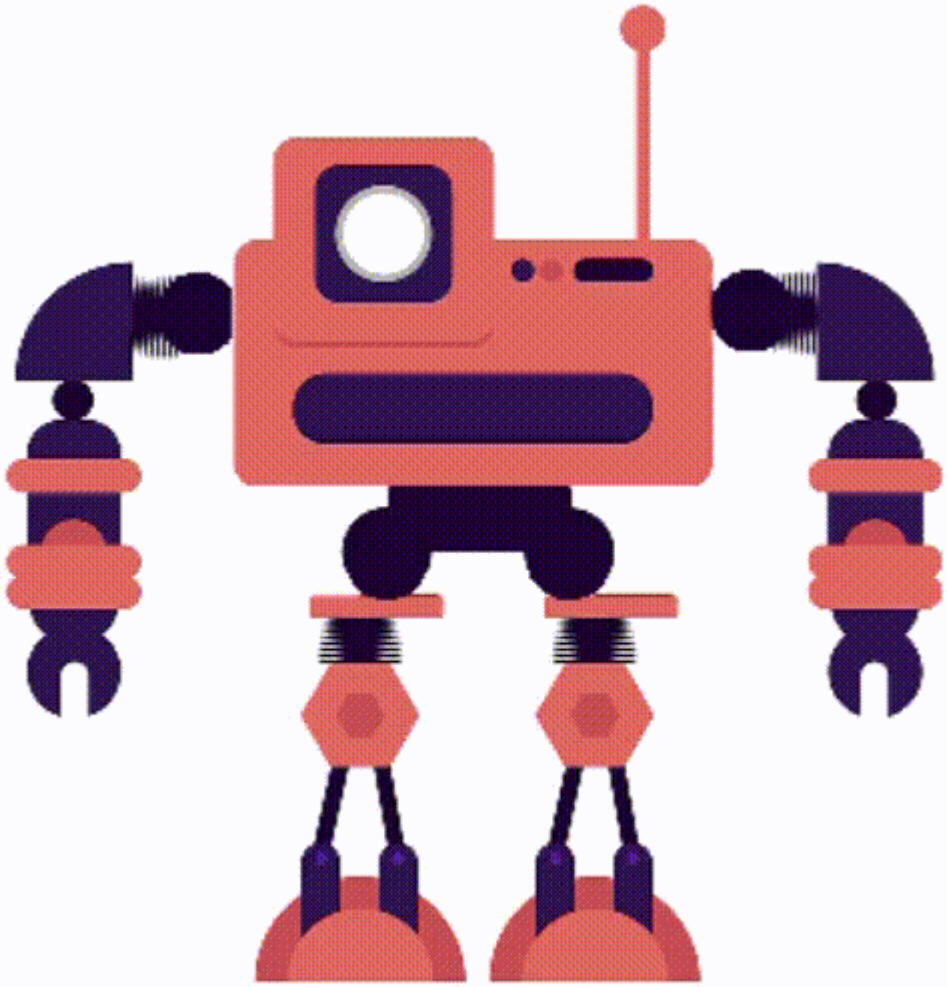
Copyright (c) 2026 Yash Arote

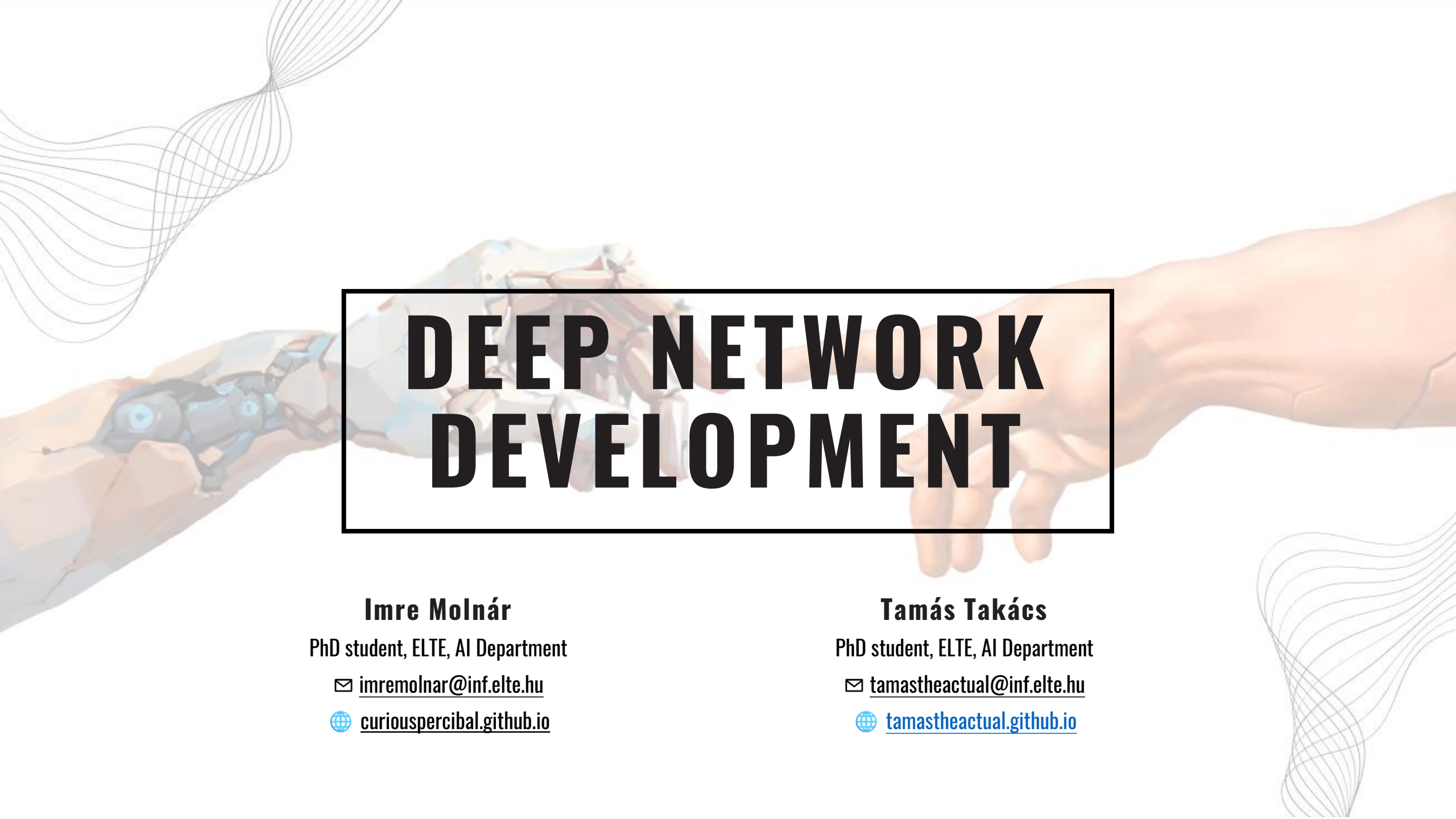
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 3

Image Classification & Convolutional Neural Networks

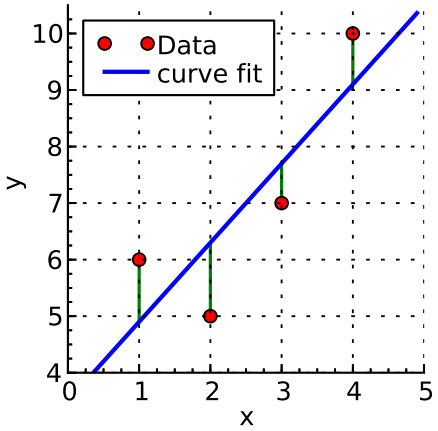
1 Image Classification

2 Convolutional Neural Networks

Previously on Lecture 2



Use Linear Regression to predict Life Expectancy



$f\left(\text{  }\right)$ =77.6 years

$f\left(\text{  }\right)$ =78.1 years

Normal Equation:
$\boldsymbol{\theta^* = (X^T X)^{-1}(X^T Y)}$

$\hat{y} = \boldsymbol{\theta_1 x + \theta_0}$ $\quad \hat{\mathbf{Y}} = \boldsymbol{X\theta}$

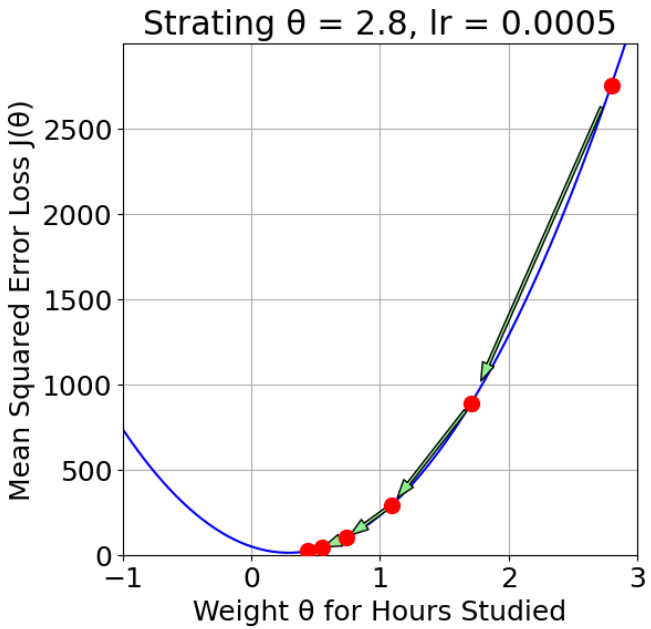
$\boldsymbol{\theta_1, \theta_0 = ?}$

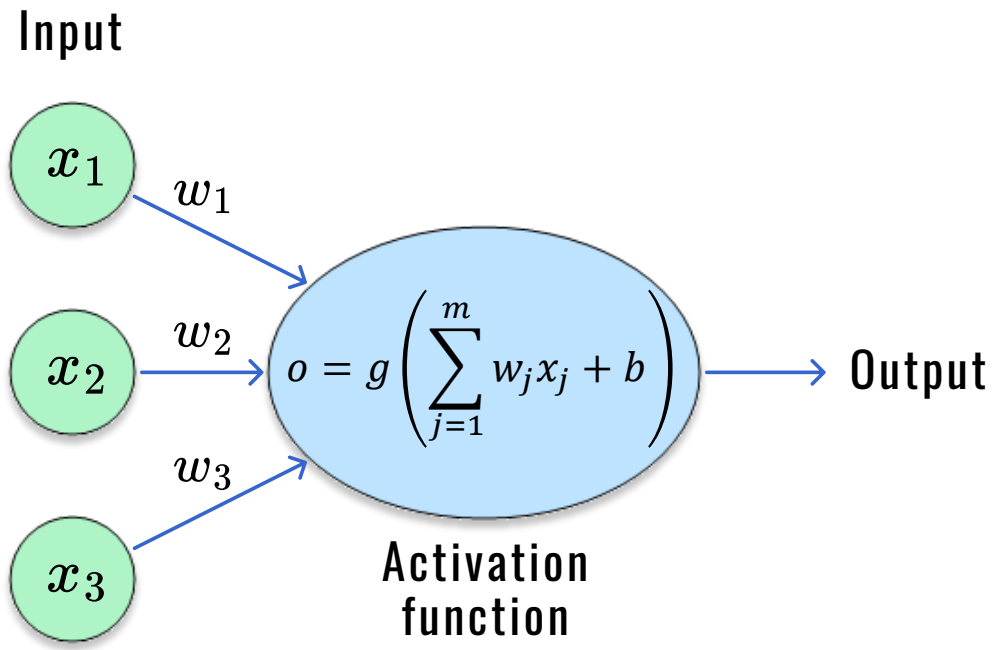
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_1 x^{(i)} + \theta_0 - y^{(i)}\right)^2$$

$$\boldsymbol{J(\theta) = (X\theta - Y)^T(X\theta - Y)}$$

Image Source: Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
Image Source: Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

Previously on Lecture 2





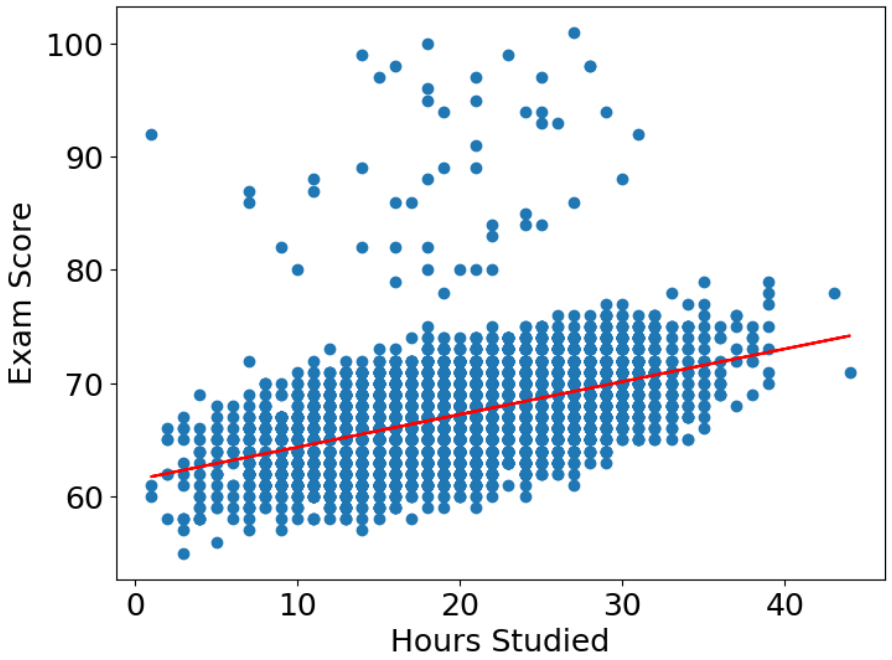
Linear Regression

- Supervised learning
- Have: (x, y)
  - x – input
  - y – target
- *Goal: Learn a function to map* $\boldsymbol{x \rightarrow y}$.
  - $\boldsymbol{h(x)} = \hat{y}$
  - $\hat{y} = \boldsymbol{\theta_1 x + \theta_0}$
- Regression – Predict real-valued / continuous output:
  - $\hat{y} \in \mathbb{R}$

$$y = \underbrace{mx}_{\text{Slope}} + \underbrace{b}_{\text{y-intercept}}$$

Slope: Describes the slope of the line

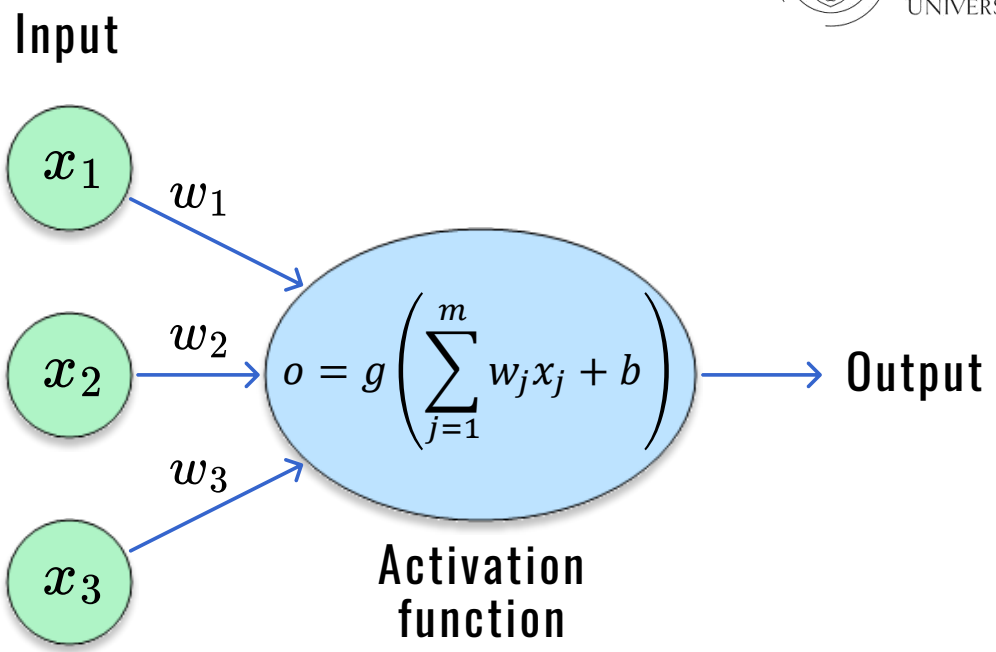
y-intercept: Describes where the line crosses the y-axis

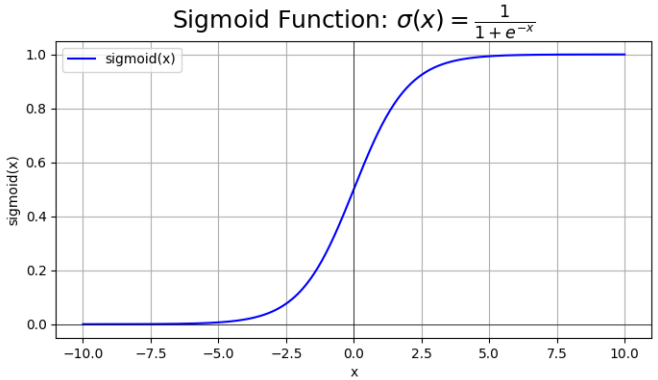


- Given the height and weight of a person, predict the age of the person.
- Age can be: 10,25,33,71,...

Artificial Neural Network (ANN)

- Using ANN for linear regression
- x_j – the inputs
- w_j – parameters we will train
- b – bias parameter
- **g – nonlinear activation function**
- o – the output
- One neuron with m inputs does the following:
  - $o = g\left(\sum_{j=0}^{m} w_j x_j + b\right)$





$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Activation function

- Adds non-linearity
- Output limited to a range (i.e. 0-1)
- Given the height and weight information, predict if it is a human or an animal.
- 0: human; 1: animal

Deep Network Development

Lecture 3

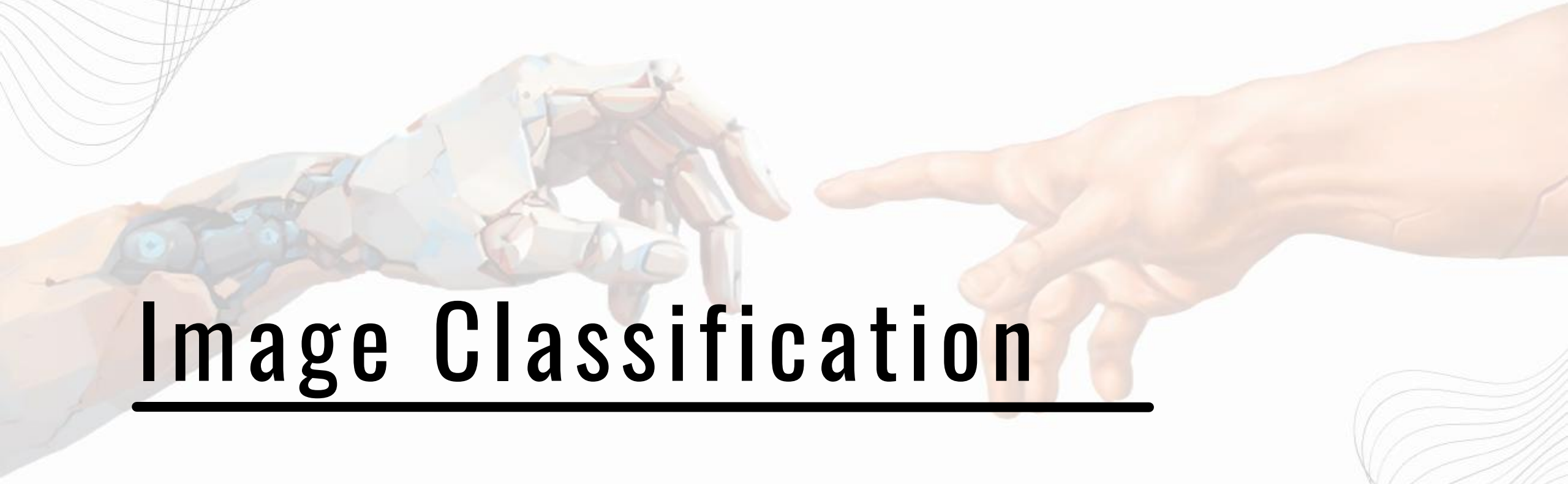


Image Classification

1 Image Classification

2 Convolutional Neural Networks

Linear Regression vs Classification

Linear Regression

- Supervised learning
- Have: (x, y)
  - x: input
  - y: target $\in \mathbb{R}$
- *Goal: Learn a function to map $x \rightarrow y$.*
  - $h(x) = \hat{y}$
  - $\hat{y} = \sum_{j=0}^{m} w_j x_j + b$
- Regression – Predict real-valued / continuous output:
  - $\hat{y} \in \mathbb{R}$

Binary Classification

- Supervised learning
- Have: (x, y)
  - x: input
  - y: target $\in \{0, 1\}$
- *Goal: Learn a function to map $x \rightarrow y$.*
  - $h(x) = \hat{y}$
  - $\hat{y} = g\left(\sum_{j=0}^{m} w_j x_j + b\right)$ Add non-linear activation function
- Classification – Predict discrete set of values:
  - $\hat{y} \in \{0, 1\}$

Binary Classification

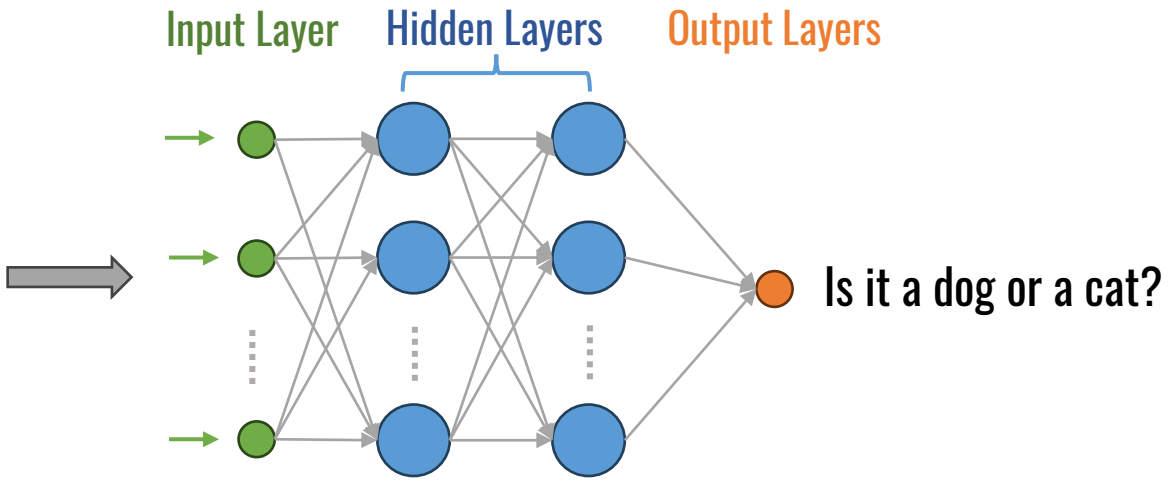
- Binary classification – Two classes
- Is x in the target class C?
- Output $P(x \in C)$, the probability of x is in C
- The network will have 1 output
- h(x) must be between 0 and 1

Dog vs Cat





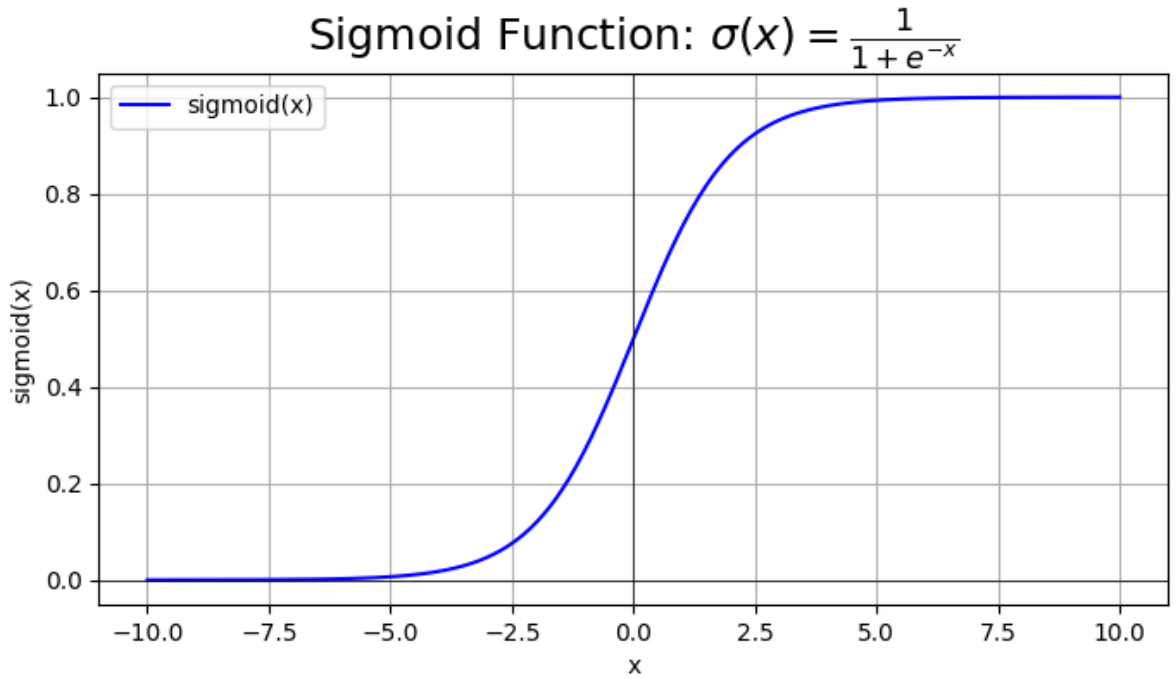




Binary Classification

- h(x) must be between 0 and 1
- Use the sigmoid activation on last layer

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



- Linear regression becomes Logistic Regression (Binary Classification), if we add a Sigmoid activation function to the output
- Linear Regression: $\boldsymbol{h}(\boldsymbol{x}) = Wx + b$ (continuous output)
- Binary Classification: $\boldsymbol{h}(\boldsymbol{x}) = g(Wx + b)$(discrete output between 0 and 1)

Linear Regression

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Binary Classification Loss

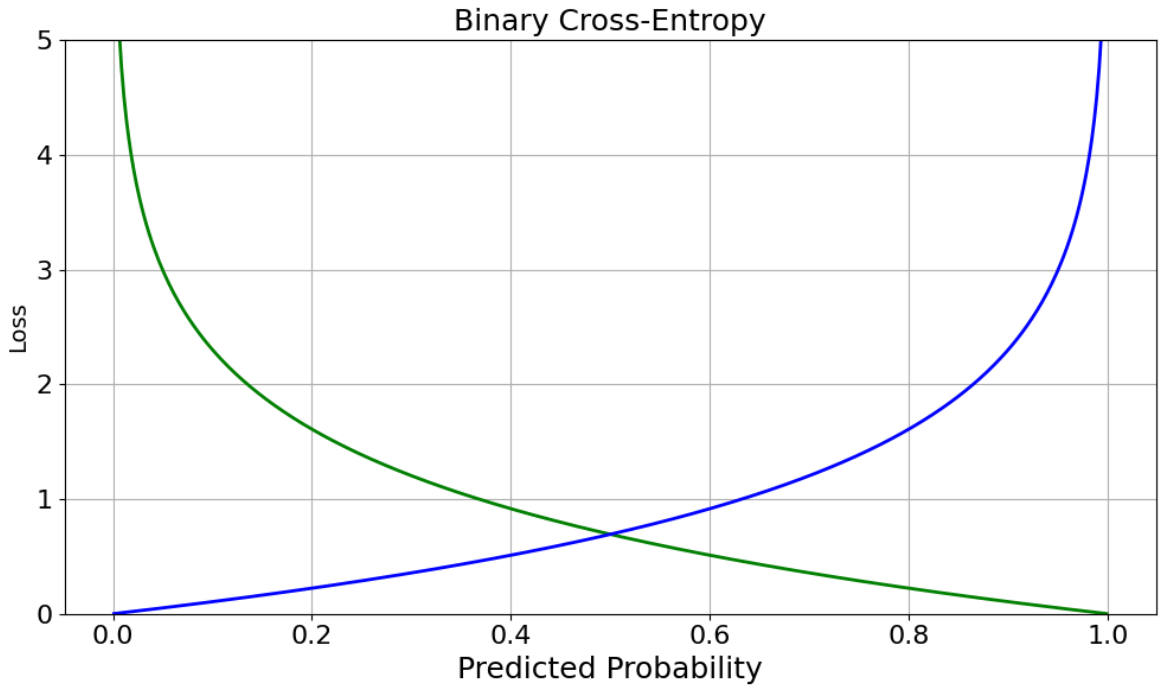
- Binary cross-entropy loss

  $L_{BC}(h(x), y) = -y \log h(x) - (1 - y) \log(1 - h(x))$

- Assumptions:

  y is 0 or 1

  $h(x) = \hat{y}$ is between 0 and 1



$L_{BC}(h(x), 1)$

$L_{BC}(h(x), 0)$

$$L_{BC}(h(x), y) = -y \log h(x) - (1 - y) \log(1 - h(x))$$

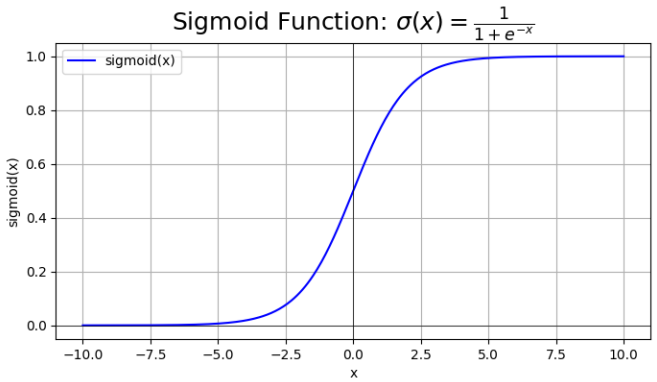
$$L_{BC}(1,1) = -1 \log 1 - (1 - 1) \log(1 - 1)$$

$$L_{BC}(1,1) = 0 - (0) \log(0)$$

$$L_{BC}(1,1) = 0$$

log(0) is undefined -> clipping

Binary Classification example



Classes C = {0: Dog, 1: Cat}

INPUT

NEURAL NETWORK

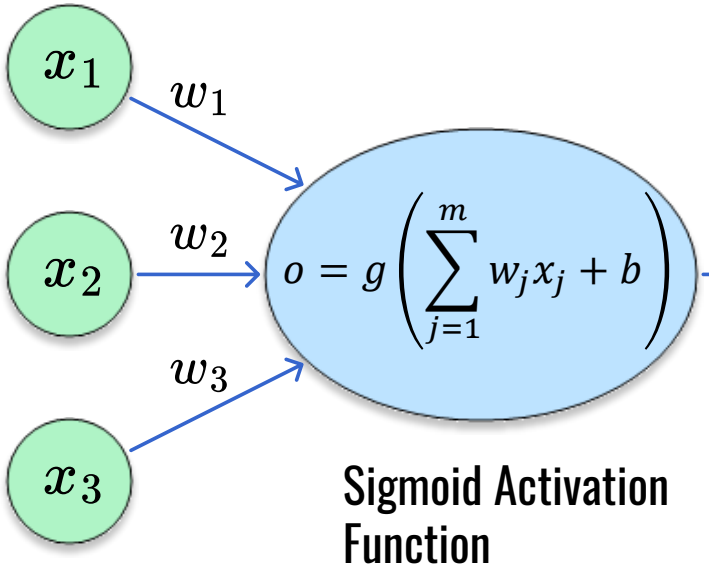
OUTPUT

TARGET



Reshaped image

$$\begin{pmatrix} 123 \\ 124 \\ 64 \\ 85 \\ 12 \\ 3 \\ \vdots \\ 57 \\ 234 \end{pmatrix}$$



Sigmoid Activation Function

If the prediction is y_{pred} > 0.5 then we consider that it predicted 1

Prediction: 0.73 Y = 1

LOSS

$$L_{BC}(h(x), y) = -y \log h(x) - (1-y)\log(1-h(x))$$
$$L_{BC}(0.73, 1) = -1 \log 0.73 - (1-1)\log(1-0.73)$$
$$L_{BC}(0.73, 1) = -1 * -0.13667714 - (0)\log(0.23)$$
$$L_{BC}(0.73, 1) = 0.13667714$$

$$\boldsymbol{L_{BC}(0.73, 1) = 0.14}$$

More neurons and categories

Next

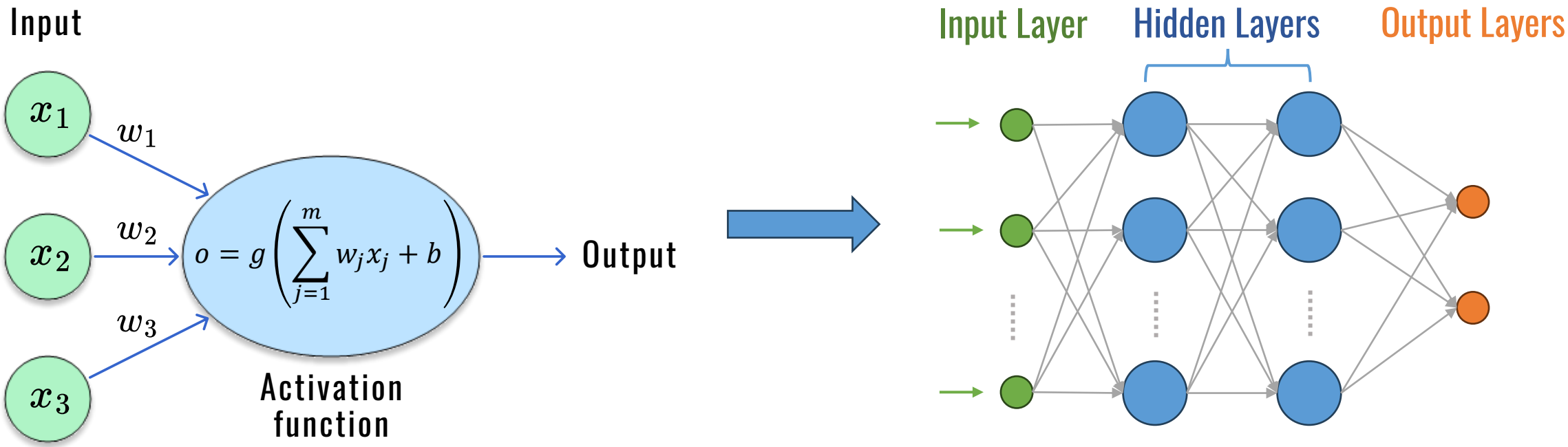
- Single artificial neuron
- Binary Classification (0 or 1)

→

- Deep Neural Network
- Multi-Class Classification (0,1,2,3…)

Deep Neural Networks

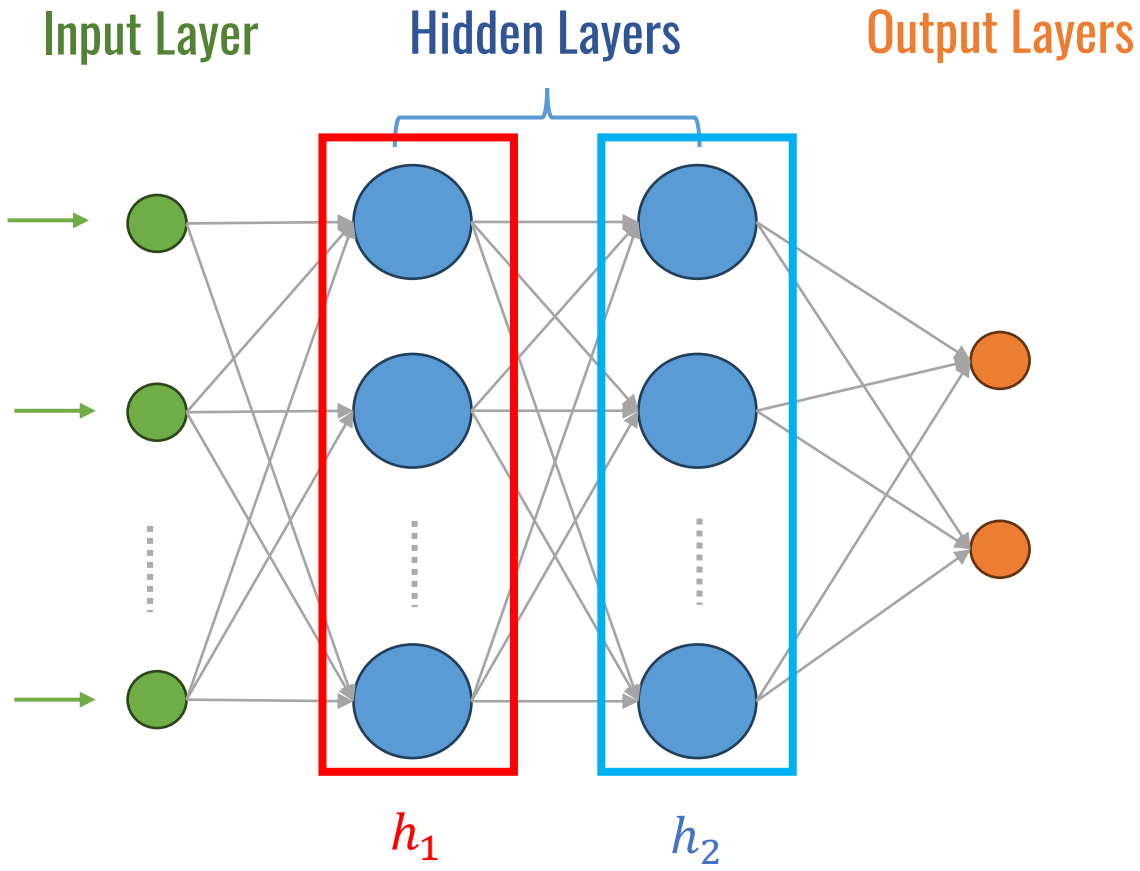
- Neural networks are built up from neurons, that have inputs and outputs
- Neurons are organised into layers
- Layers refine the output of the previous layers



Deep Neural Networks

- Artificial Neural Networks ⟺ Feed Forward Neural Networks ⟺ Fully Connected Networks

- A neural network is built up from neurons
- Neurons are organised into layers
- Layers refine the output of the previous layers
  - $\boldsymbol{h_1(x)} = g\left(W^{(1)}x + b^{(1)}\right)$
  - $h_2(x) = g\left(W^{(2)}\boldsymbol{h_1} + b^{(2)}\right)$
  - $h_2(x) = g\left(W^{(2)}\boldsymbol{g}\left(\boldsymbol{W^{(1)}x} + \boldsymbol{b^{(1)}}\right) + b^{(2)}\right)$



Deep Neural Networks

- How to find *W* and *b*?
- Given $h = g\left(W^{(2)}\boldsymbol{g}\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + b^{(2)}\right)$

- Select a loss function L that measures how good *h* is:
  - the smaller L(h(x),y) , the better *h* is

- Update $\boldsymbol{W}^{(1)}$:
  $$\boldsymbol{W}_t{}^{(1)} = \boldsymbol{W}_{t\_1}{}^{(1)} - \alpha\nabla J(\boldsymbol{W}^{(1)})$$
- Update $\boldsymbol{W}^{(2)}$:
  $$\boldsymbol{W}_t{}^{(2)} = \boldsymbol{W}_{t\_1}{}^{(2)} - \alpha\nabla J(\boldsymbol{W}^{(2)})$$
- Update $\boldsymbol{b}^{(1)}$:
  $$\boldsymbol{b}_t{}^{(1)} = \boldsymbol{b}_{t\_1}{}^{(1)} - \alpha\nabla J(\boldsymbol{b}^{(1)})$$
- Update $\boldsymbol{b}^{(2)}$:
  $$\boldsymbol{b}_t{}^{(2)} = \boldsymbol{b}_{t\_1}{}^{(2)} - \alpha\nabla J(\boldsymbol{b}^{(2)})$$

Backpropagation derivation: https://www.cs.put.poznan.pl/pliskowski/pub/teaching/eio/lab1/eio-supplementary.pdf
Another backpropagation derivation (be aware of notation differences): https://www.cs.swarthmore.edu/~meeden/cs81/s10/BackPropDeriv.pdf

Terminology

- L(h(x),y) – loss or error function for a single data point of the dataset
- C(θ), J(W) – cost or objective function for the entire dataset (usually average)
- The update rule can also be in the form of

$$w_i \leftarrow w_i + \Delta w_i$$

$$b \leftarrow b + \Delta b$$

$$\Delta w_i = -\alpha\frac{\partial L}{\partial w_i}$$

$$\Delta b = -\alpha\frac{\partial L}{\partial b}$$

Deep Neural Networks

1. Select W_0 randomly
2. Calculate $h(X)$ for all training examples
3. Calculate the cost:

$$J(W) = \frac{1}{N}\sum_{i=1}^{N} L(h(x_i), y_i)$$

4. Update W:

$$W_t = W_{(t-1)} - \alpha \nabla J(W)$$

5. Repeat steps 2-4 until convergence

Deep Neural Networks

Batch size – the number of training samples used to optimize the model's parameters

Types (with Gradient Descent)

- **Batch Gradient Descent**
  - Batch size = Size of dataset
- **Mini-Batch Gradient Descent**
  - Batch size = a subset of the dataset examples; typically, 2^n , e.g., 8,16,32,128,...
  - In each iteration, a mini-batch is randomly sampled from the dataset
- **Stochastic Gradient Descent**
  - Batch size = 1
  - In each iteration, a single example is randomly sampled from the dataset

1. Select W_0 randomly
2. Calculate $h(X)$ for all training examples
3. Calculate the cost:

$$J(W) = \frac{1}{N}\sum_{i=1}^{N} L(h(x_i), y_i)$$

4. Update W:

$$W_t = W_{(t-1)} - \alpha \nabla J(W)$$

5. Repeat steps 2-4 until convergence

Hyperparameter vs Parameter

Hyperparameters

To create and train Neural Networks we must make certain decisions (we choose the values):

- Number of layers; Number of neurons in each layer
- Activation functions
- Learning rate
- Batch size
- Many more (optimization related)

Parameters

- What the Neural Network learns. The weights and biases of the network are optimized with an algorithm (Gradient Descent, etc.). We do not choose the values for them.
- We can choose the initial values (weight initialization), it can be initialized with zeros, ones, random values, etc.
- However, the network learns through training.

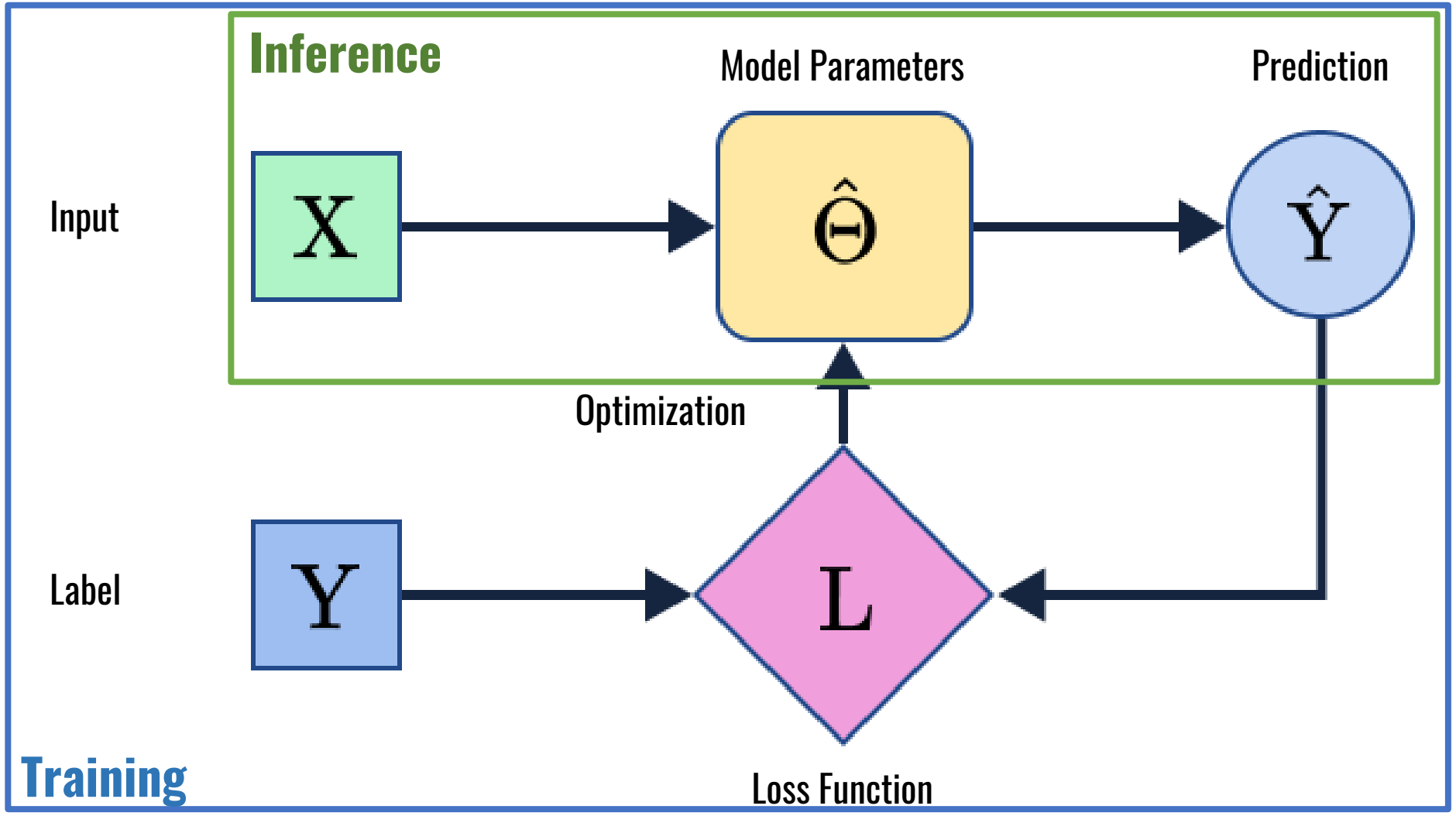
Deep Neural Networks

Inference:

- Given the input X the model tries to predict the label using it's parameters (Y)

Training:

- We run multiple inferences and we compare the predictions with the actual label (ground truth) using the loss function
- We update the model parameters based on the loss



More neurons and categories

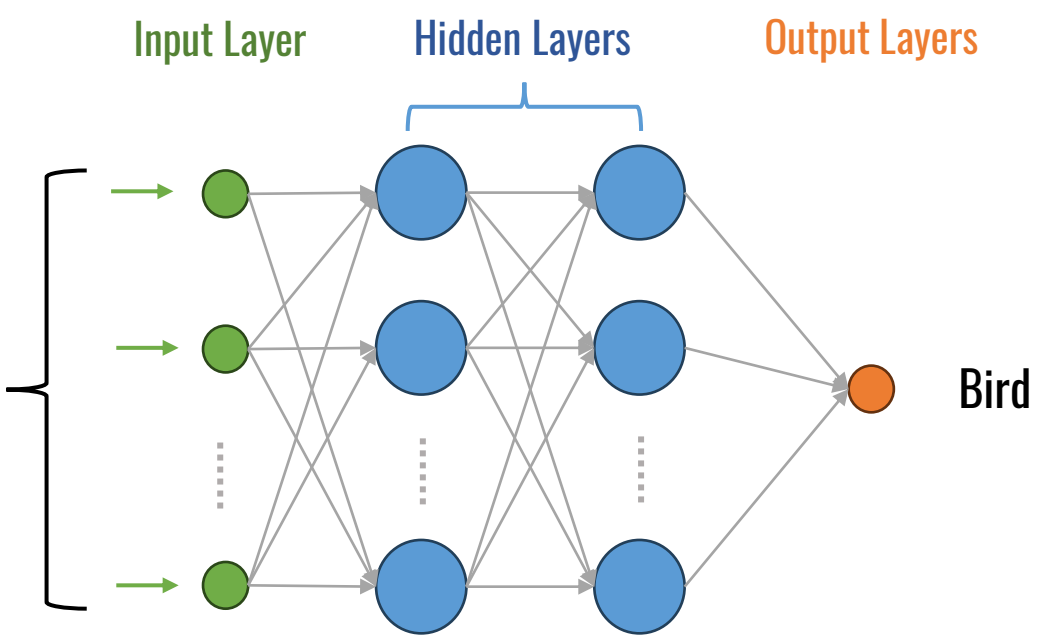
- Single artificial neuron
- Binary Classification (0 or 1)

→

- Deep Neural Network
- Multi-Class Classification (0,1,2,3…)

Next



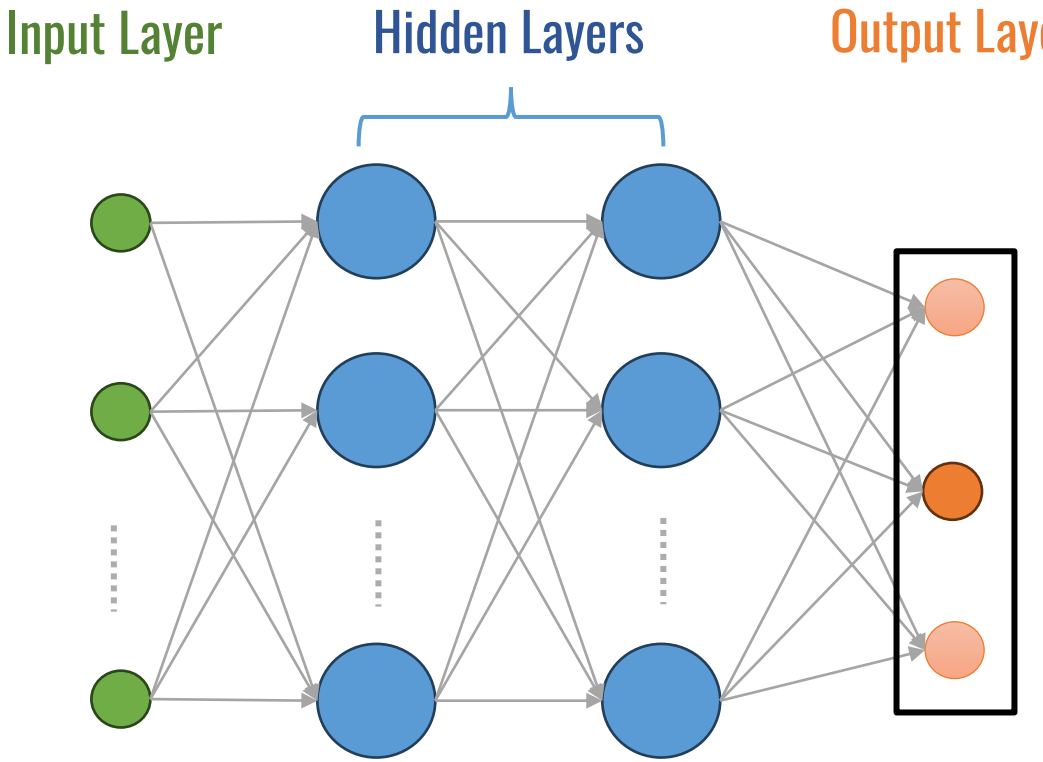


Multi-Class Classification

- With *K* classes
- The network will have K outputs
- Softmax activation for squashing outputs between 0 and 1

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

- We interpret the softmax-normalized values as a probability distribution over the classes.
- The softmax-normalized values sum up to 1.



We apply softmax to the output logits

Multi-Class Classification

Output of the neural network is a K long vector

How should we encode the ground-truth values?

One-hot encoding (K=3):

1. (cat) -> [1, 0, 0]
2. (dog) -> [0, 1, 0]
3. (horse) -> [0, 0, 1,]

Just as the softmax-normalized values between 0 and 1, sum up to 1

Class
Cat
Dog
Horse

One-hot encoding ⟹

Cat	Dog	Horse
1	0	0
0	1	0
0	0	1

Multi-Class Classification

- K = 3 (number of classes)
- Softmax activation $\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$
- Probability distribution on the classes. They sum up to 1.
- We will need 3 neurons in the output layer to get 3 logits for each class (each neuron produces one logit)

(Cat, Dog, Horse)



Input image



Feeding image to network →

Output logits

Cat	Dog	Horse
4	2	8

Applying softmax on the outputs →

Class probabilities

Cat	Dog	Horse
0.02	0.00	0.98

Multi-Class Classification Loss

With *K* classes
Categorical Cross-entropy loss

$$L(h(x), y) = -\sum_{i=1}^{K} y_i \log h_i(x)$$

Assumptions:

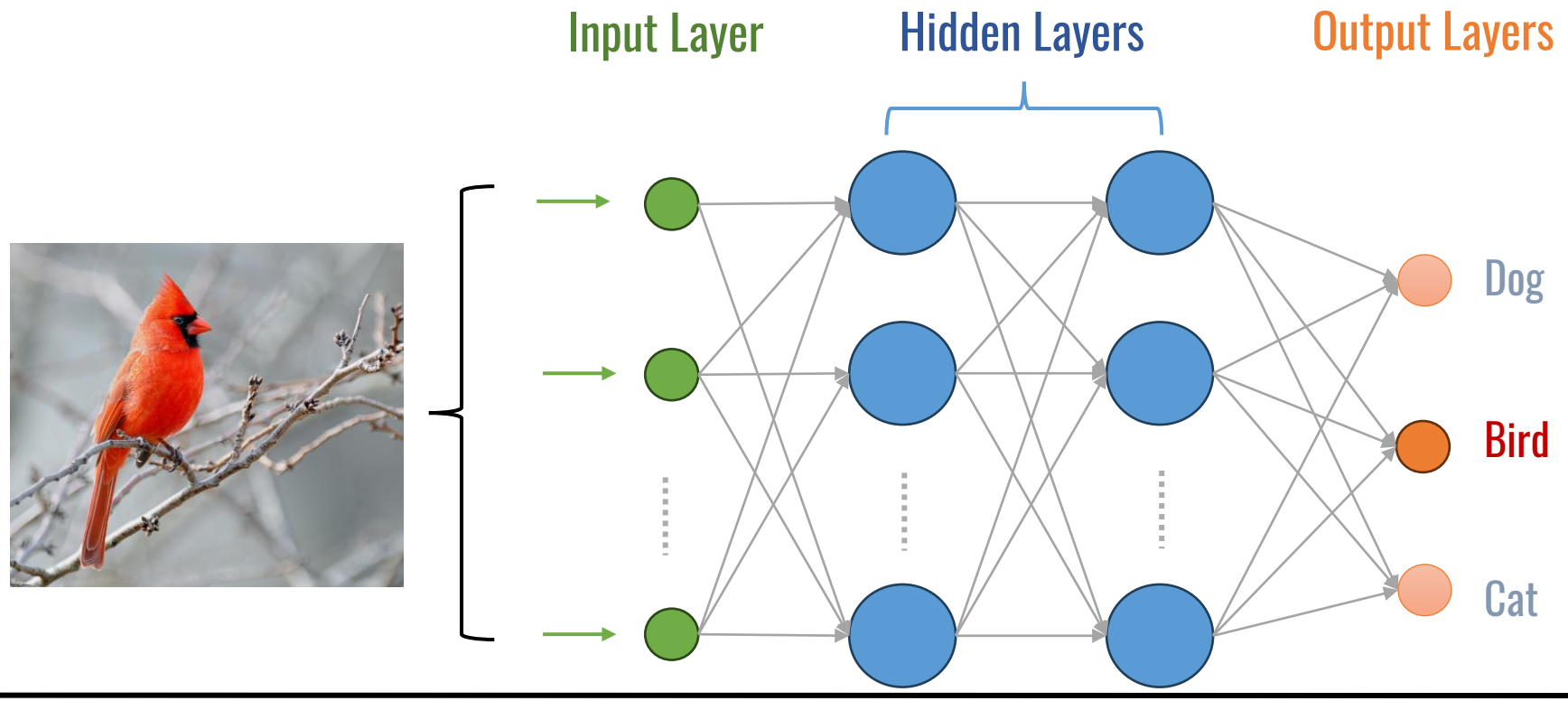
- *y* is a vector of *K elements with values* 0 or 1, exactly one element is 1
- $h_i(x)$ is a vector of K elements between 0 and 1, the sum of all elements is equal to 1

More neurons and categories

- Single artificial neuron
- Binary Classification (0 or 1)

→

- Deep Neural Network
- Multi-Class Classification (0,1,2,3…)

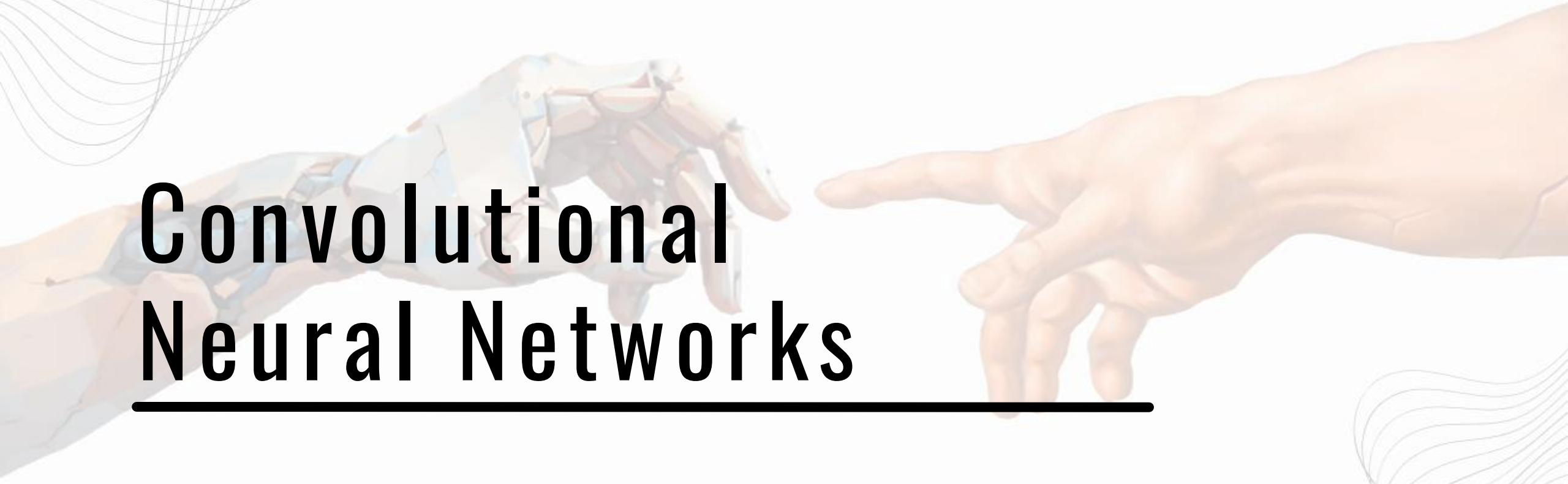


Disadvantages of Feed Forward Neural Networks

1. **Computational inefficiency**
   - For an input image with a resolution of 64x64 we would have 4096 input variable
   - Each input variable will have a dedicated weight – more parameters to optimize
2. **Loss of spatial information**
   - If we trained the model on images of cats, where the cats are sitting in the center of the image, if we give an image of a cat where the cat is for example at the top left corner of the image, it might not be able to detect it
   - Smaller cats might get lower scores than larger cats
3. **Independent weights/features**
   - The weights are shared, for example we use the same weights to try to detect a cat and a horse
   - For objects with varying features (dog vs. car) we get suboptimal efficiency

Deep Network Development

Lecture 3

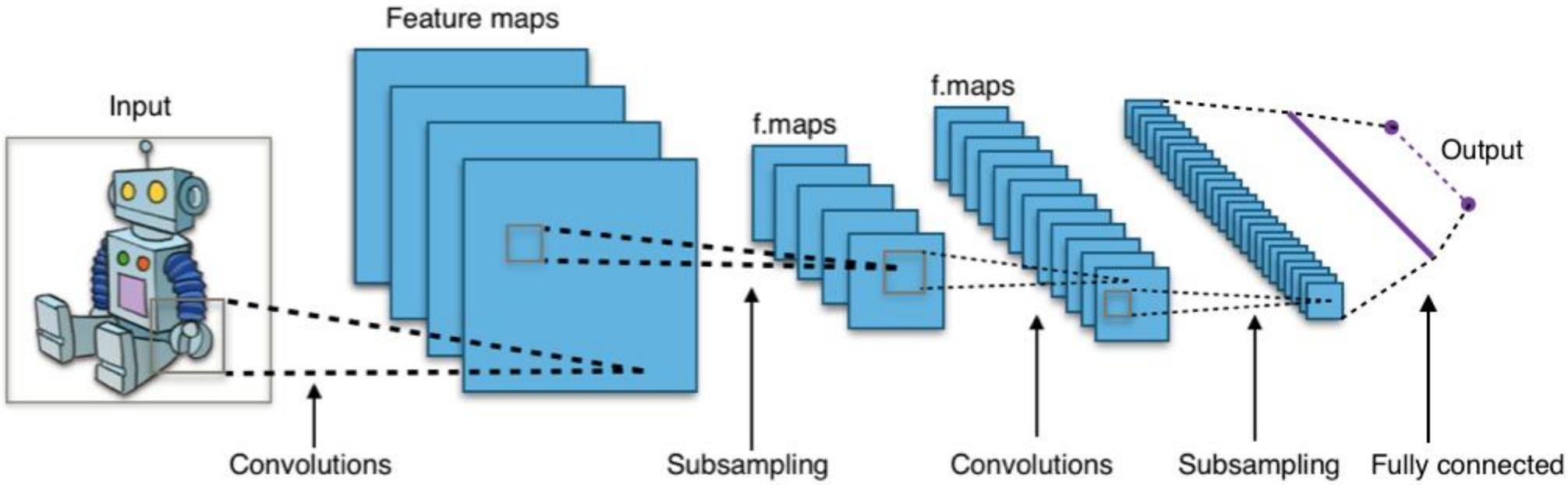


Convolutional Neural Networks

1 Image Classification

2 Convolutional Neural Networks

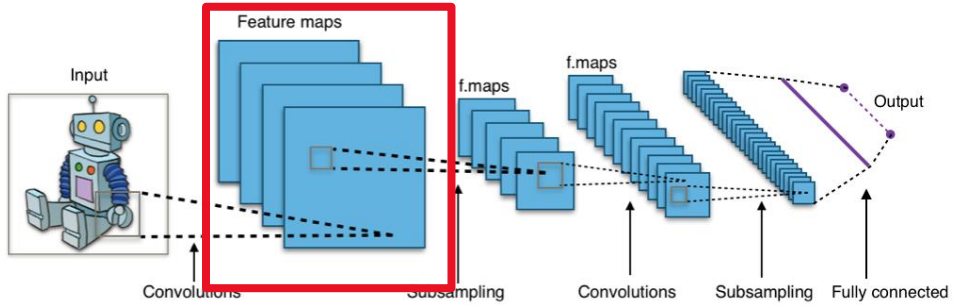
How do Convolutional Neural Networks (CNNs) work?



CNNs extract relevant features from an image.
Like a magnifying glass “enhancing” details of the image.

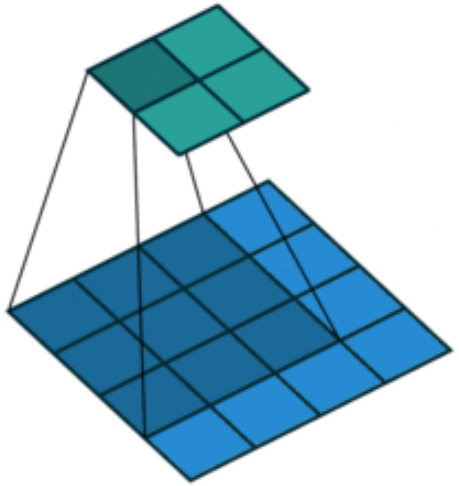
Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Convolutional layer



Convolution
Input: 5x5
Filter: 3x3
Output: 3x3

Filter / Kernel size = f x f



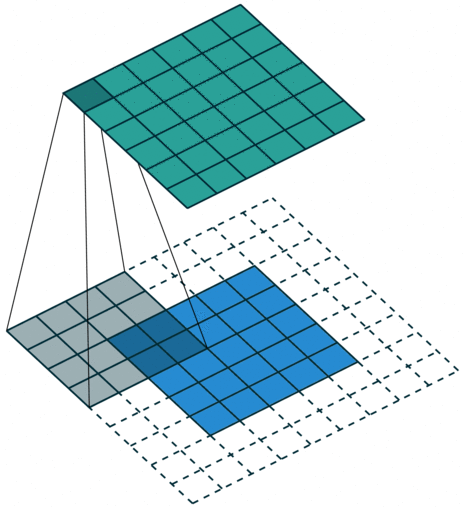


Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Convolution arithmetic visualization, adapted from Vincent Dumoulin, & Francesco Visin. (2018). A guide to convolution arithmetic for deep learning.; original visual assets from the conv_arithmetic GitHub repository, MIT License.

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3
- Output = ?

10	2	4	1	1
0	4	0	2	2
1	8	16	0	0
8	1	1	0	0
0	0	2	6	1

*

2	2	2
1	0	1
2	2	2

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10 x 2	2 x 2	4 x 2	1	1
0 x 1	4 x 0	0 x 1	2	2
1 x 2	8 x 2	16 x 2	0	0
8	1	1	0	0
0	0	2	6	1

$$10 \times 2 + 2 \times 2 + 4 \times 2 + 0 \times 1 + 4 \times 0 + 0 \times 1 + 1 \times 2 + 8 \times 2 + 16 \times 2 = 82$$

- Output = ?

82		

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10	2 x 2	4 x 2	1 x 2	1
0	4 x 1	0 x 0	2 x 1	2
1	8 x 2	16 x 2	0 x 2	0
8	1	1	0	0
0	0	2	6	1

$$2 \times 2 + 4 \times 2 + 1 \times 2 + 4 \times 1 + 0 \times 0 + 2 \times 1 + 8 \times 2 + 16 \times 2 + 0 \times 2 = 68$$

- Output = ?

82	68	

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10	2	4 x 2	1 x 2	1 x 2
0	4	0 x 1	2 x 0	2 x 1
1	8	16 x 2	0 x 2	0 x 2
8	1	1	0	0
0	0	2	6	1

$4 \times 2 + 1 \times 2 + 1 \times 2 +$
$0 \times 1 + 2 \times 0 + 2 \times 1 +$
$16 \times 2 + 0 \times 2 + 0 \times 2 = 46$

- Output = ?

82	68	46

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

10	2	4	1	1
0 x 2	4 x 2	0 x 2	2	2
1 x 1	8 x 0	16 x 1	0	0
8 x 2	1 x 2	1 x 2	0	0
0	0	2	6	1

$$0 \times 2 + 4 \times 2 + 0 \times 2 + 1 \times 1 + 8 \times 0 + 16 \times 1 + 8 \times 2 + 1 \times 2 + 1 \times 2 = 46$$

- Output = ?

82	68	46
45		

Convolutional layer

- Convolution operation
- Input: 5 x 5
- Filter / kernel size = 3 x 3

- Output = 3 x 3

10	2	4	1	1
0	4	0	2	2
1	8	16	0	0
8	1	1	0	0
0	0	2	6	1

*

2	2	2
1	0	1
2	2	2

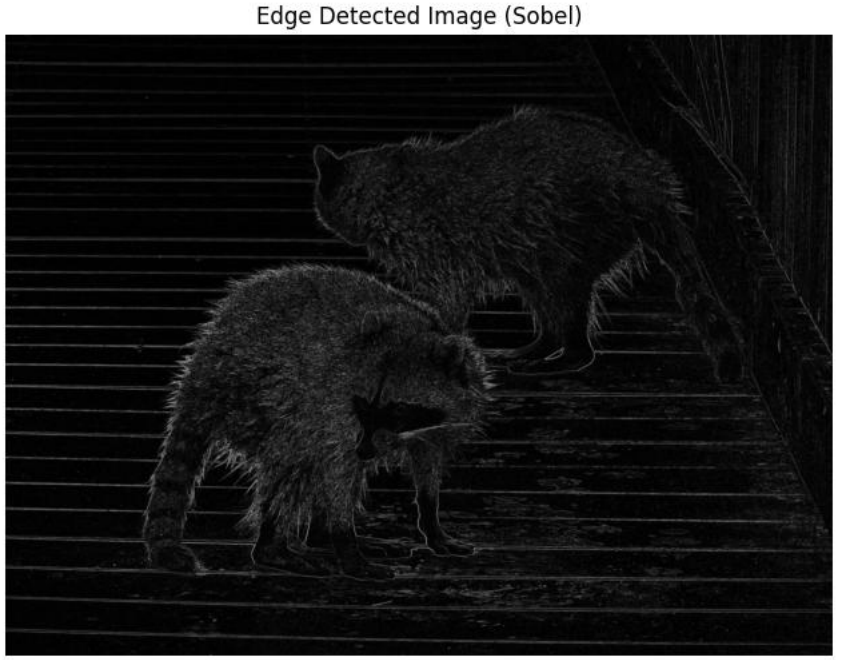
=

82	68	46
45	24	26
63	65	51

Convolutional layer



The Sobel operator applied on the source image



Sobel operator:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A$$

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A$$

$$G = \sqrt{G_x^2 + G_y^2}$$

Luckily, we don't have to define the weight of the convolution.
We can apply **learnable filters.**

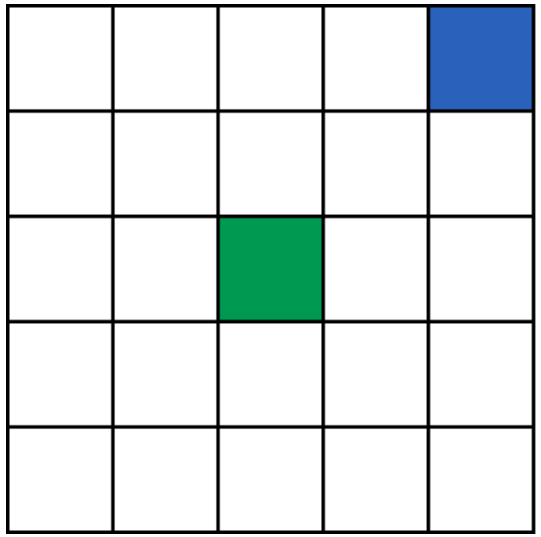
w1	w2	w3
w4	w5	w6
w7	w8	w9

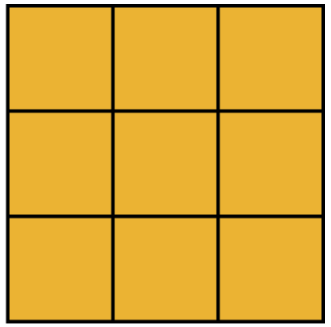
Learnable filter: The filter weights are like the weights in Feed Forward Networks (FFN). Similar to FFN, a bias term is added after applying the filter weights. Weights are shared, it requires less memory and less optimization.

Image source: Oso Flaco Lake Raccoons. - Flickr - docentjoyce.jpg Joyce Cory, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

Convolutional layer - Padding

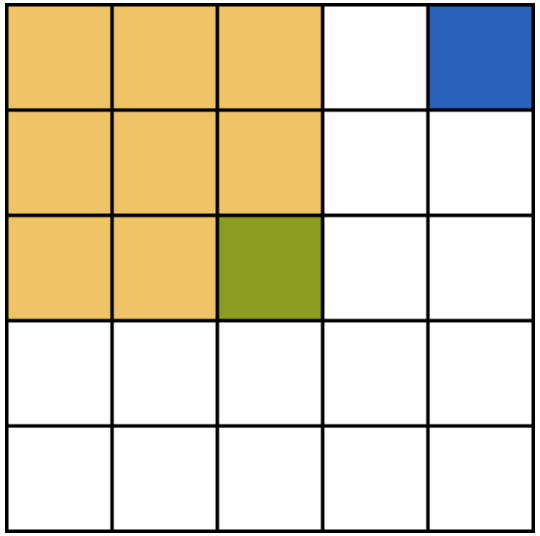
- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 0**
- **Number of times green pixel is in the filter’s receptive field: 0**





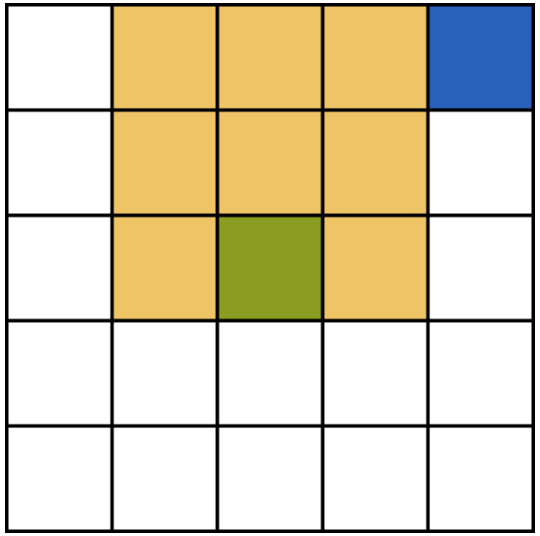
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 0**
- **Number of times green pixel is in the filter’s receptive field: 1**



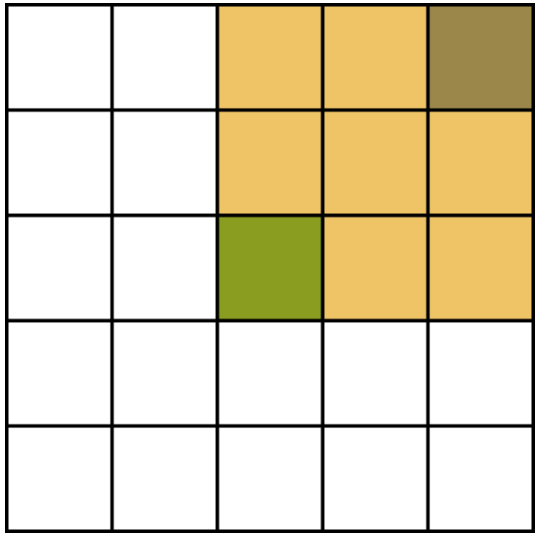
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 0**
- **Number of times green pixel is in the filter’s receptive field: 2**



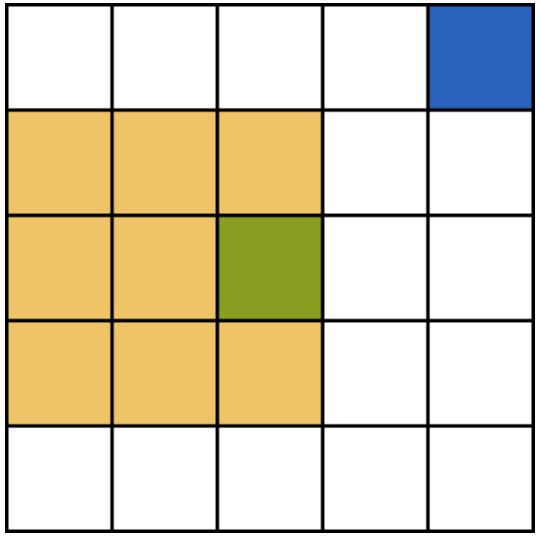
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 3**



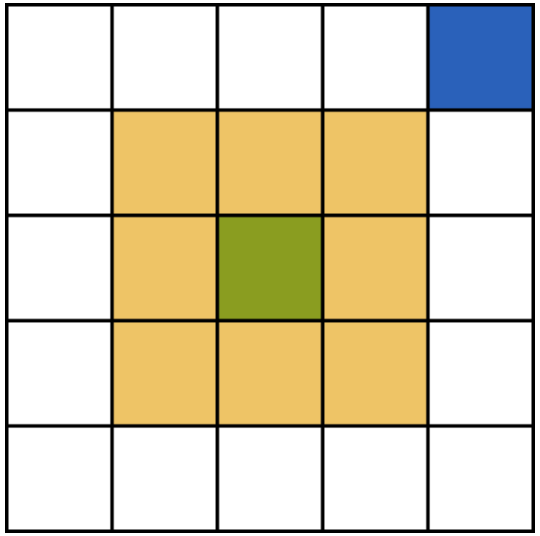
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 4**



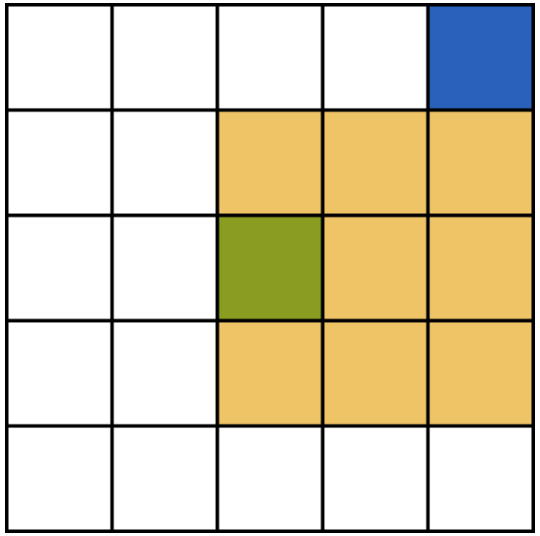
Convolutional layer - Padding

- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 5**



Convolutional layer - Padding

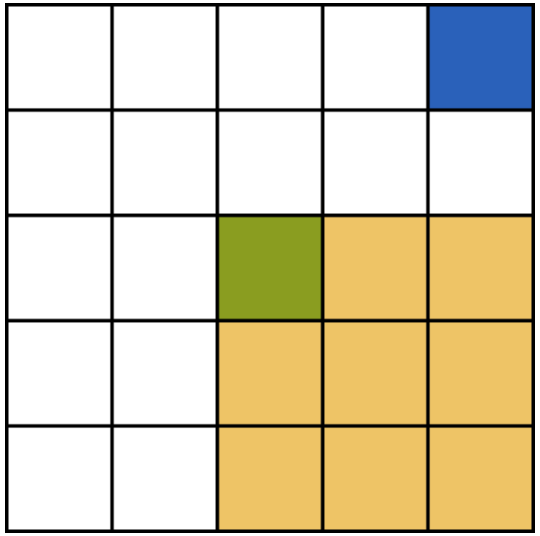
- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 6**



Convolutional layer - Padding

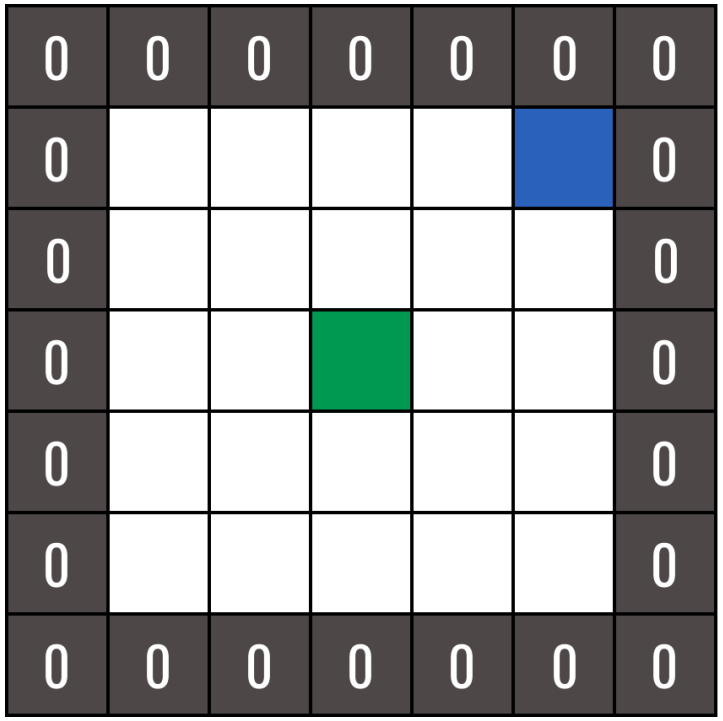
- Without Padding – corners / edges are not “seen” enough
- **Number of times blue pixel is in the filter’s receptive field: 1**
- **Number of times green pixel is in the filter’s receptive field: 9**

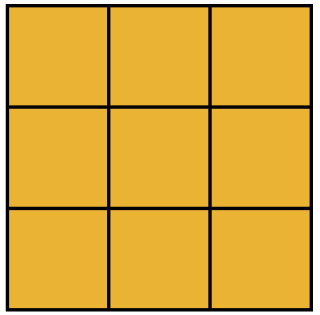
… Skip to the end



Convolutional layer - Padding

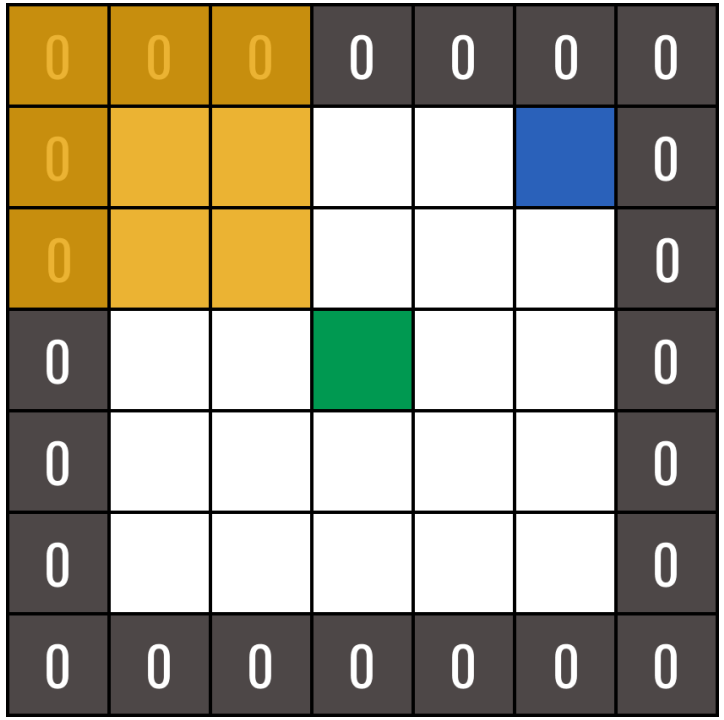
- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 0**
- **Number of times green pixel is in the filter's receptive field: 0**





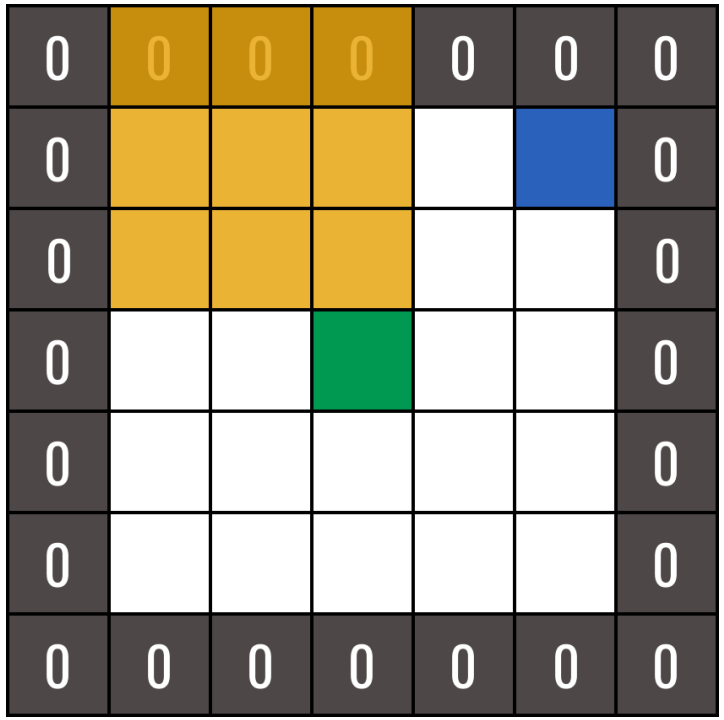
Convolutional layer - Padding

- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 0
- Number of times green pixel is in the filter's receptive field: 0



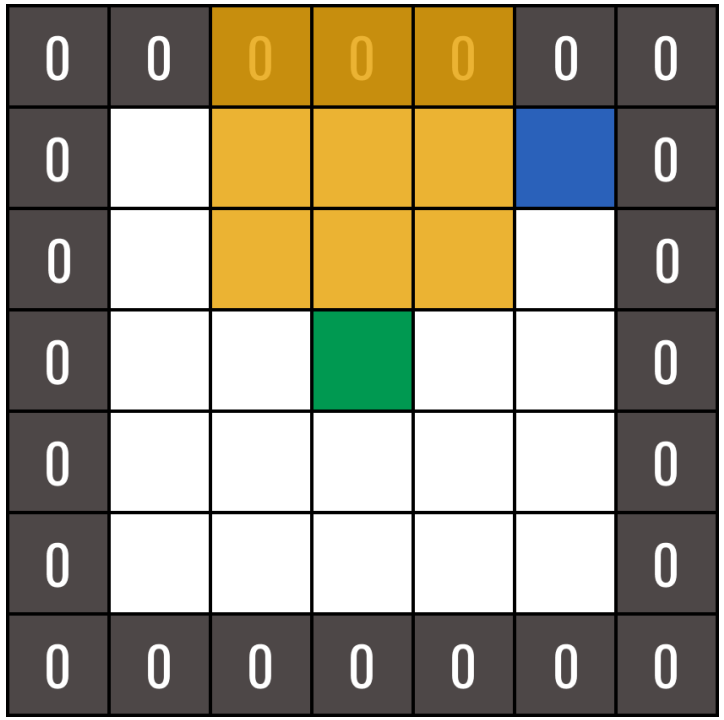
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 0**
- **Number of times green pixel is in the filter's receptive field: 0**



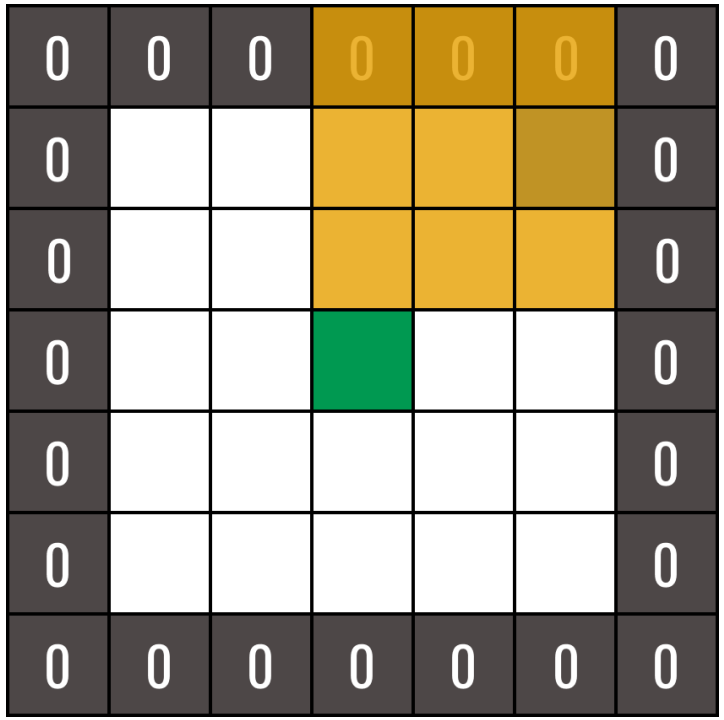
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 0**
- **Number of times green pixel is in the filter's receptive field: 0**



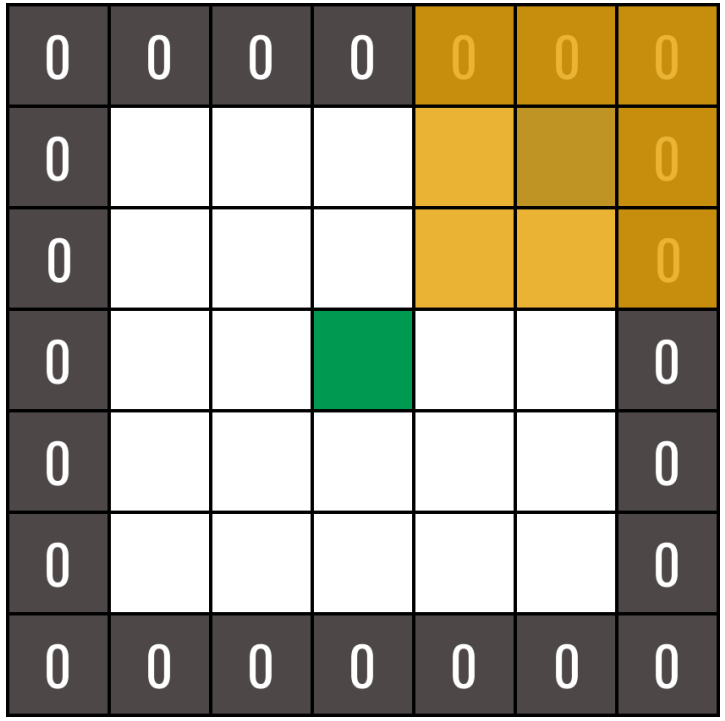
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 1**
- **Number of times green pixel is in the filter's receptive field: 0**



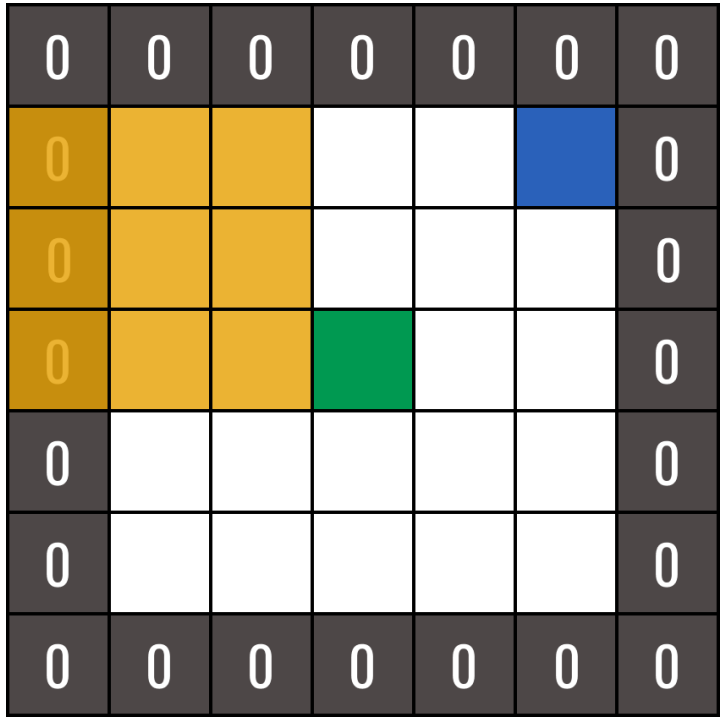
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 2**
- **Number of times green pixel is in the filter's receptive field: 0**



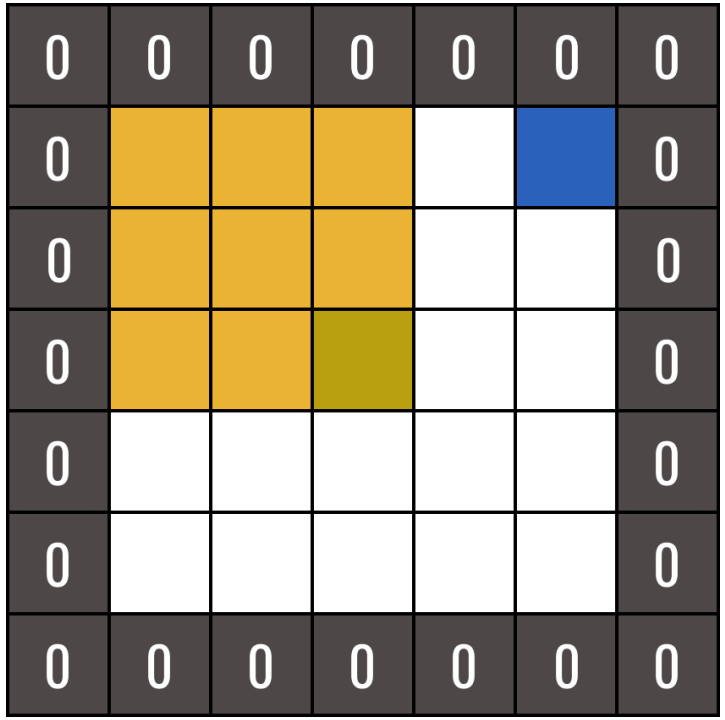
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 2**
- **Number of times green pixel is in the filter's receptive field: 0**



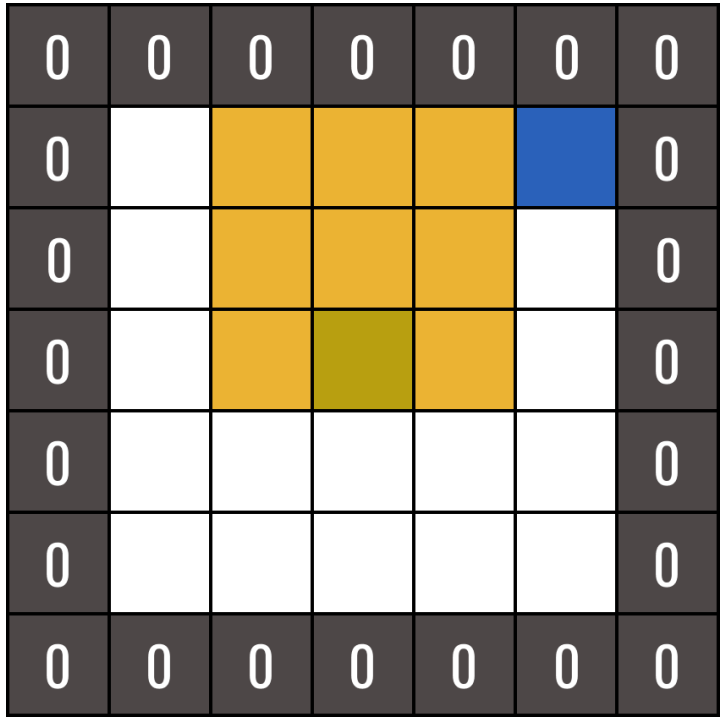
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 2**
- **Number of times green pixel is in the filter's receptive field: 1**



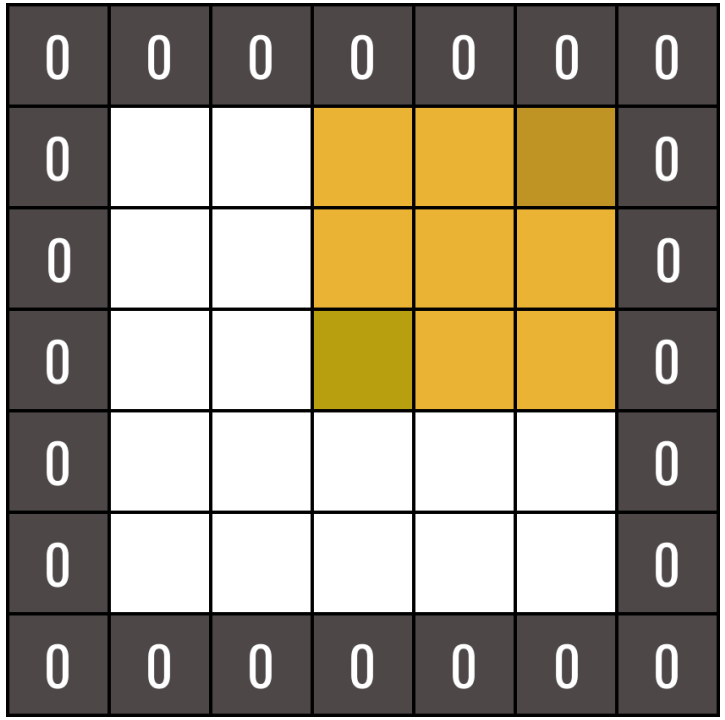
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 2**
- **Number of times green pixel is in the filter's receptive field: 2**



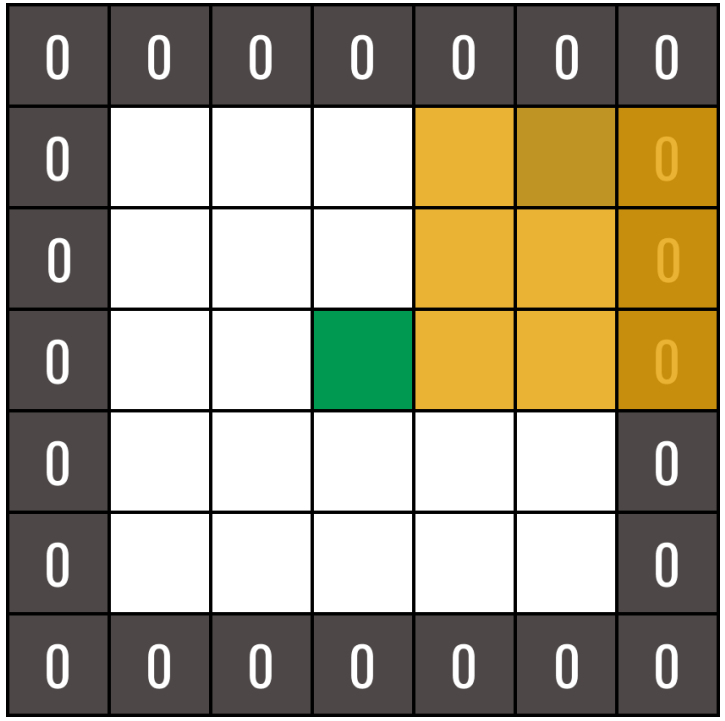
Convolutional layer - Padding

- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 3**
- **Number of times green pixel is in the filter's receptive field: 3**



Convolutional layer - Padding

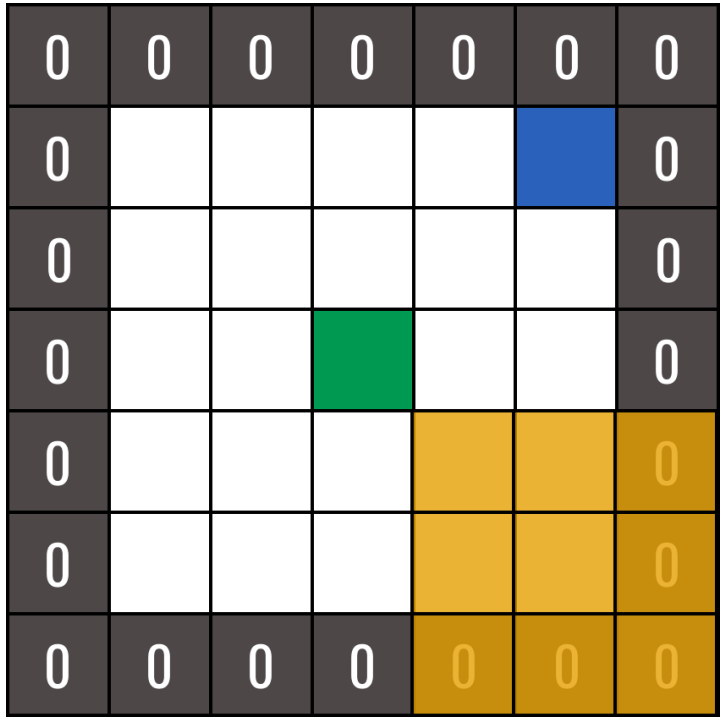
- **With Padding**
- Number of times blue pixel is in the filter's receptive field: 4
- Number of times green pixel is in the filter's receptive field: 3



Convolutional layer - Padding

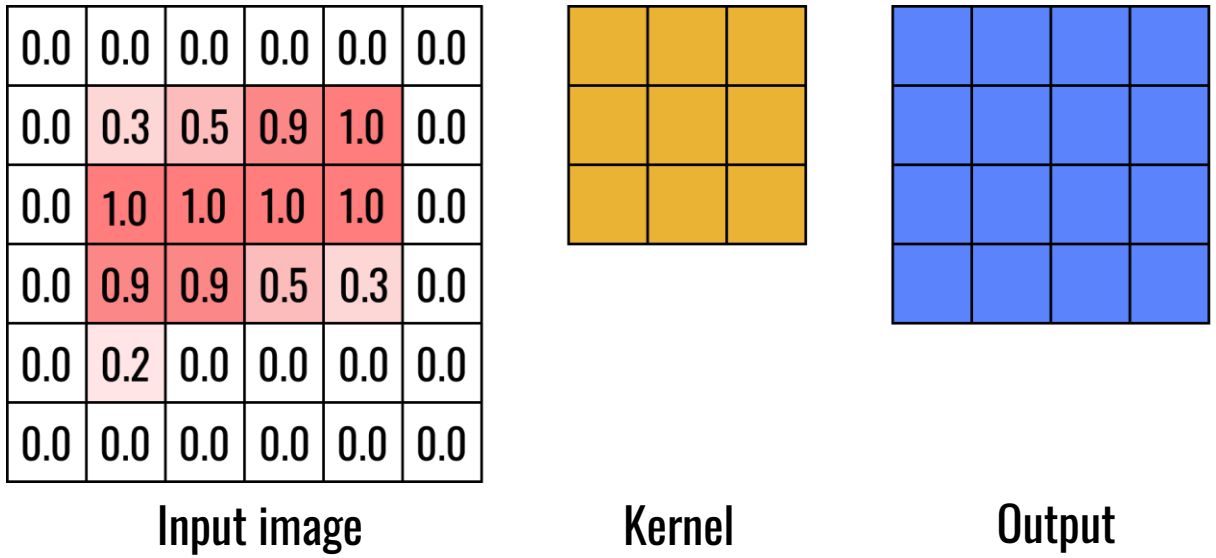
- **With Padding**
- **Number of times blue pixel is in the filter's receptive field: 4**
- **Number of times green pixel is in the filter's receptive field: 9**

... Skip to the end



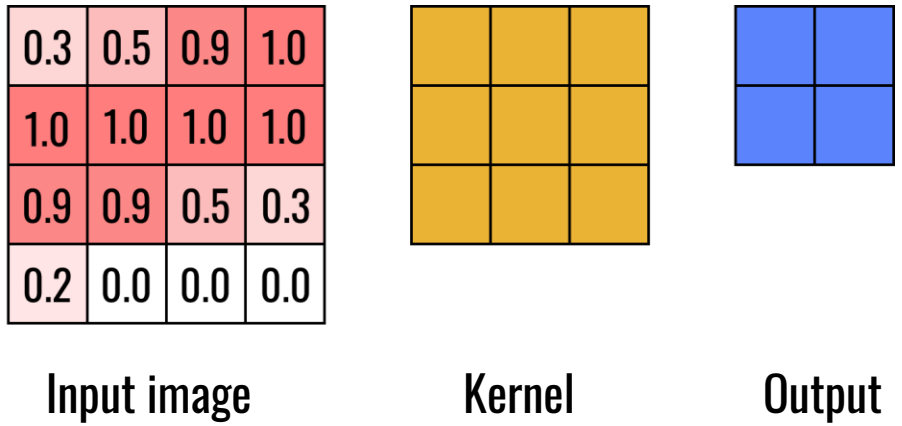
Convolutional layer - Padding

Same Padding
p ≠ 0



The padding is chosen so that the output spatial size is the **same** as the input spatial size

Valid Padding (No Padding)
p = 0

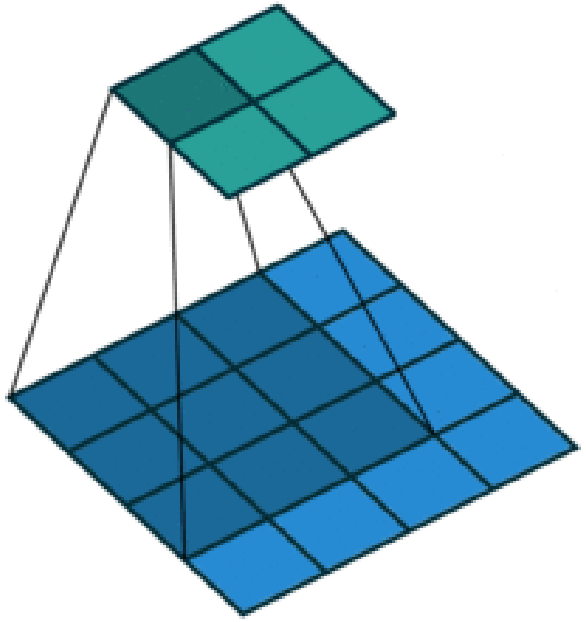


Only the "**valid**" part of the image is used. We don't add data to the image (by padding)

Convolutional layer - Stride

- The stride adjusts how much we move the kernel over the image
- Assuming valid padding and we have an input image with a resolution of 6x6 a kernel with a size 3x3 and the stride is set to 2 then part of the right border is skipped

Stride s = 1



Stride s = 2

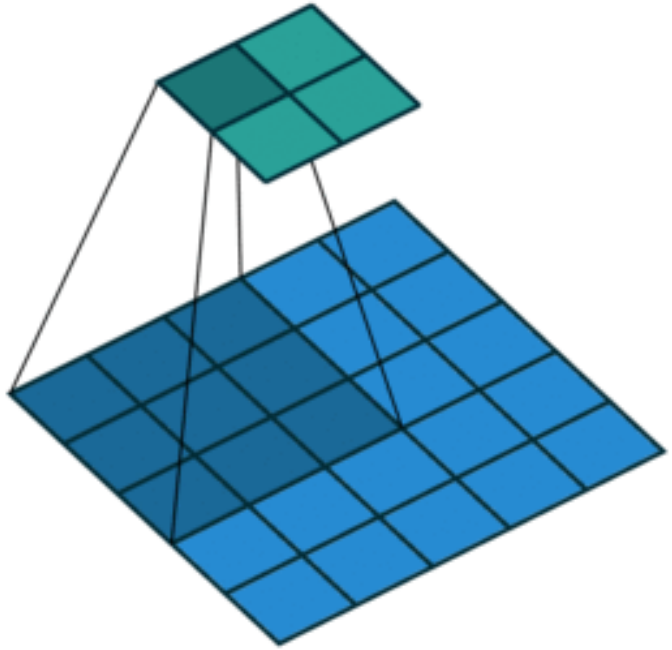
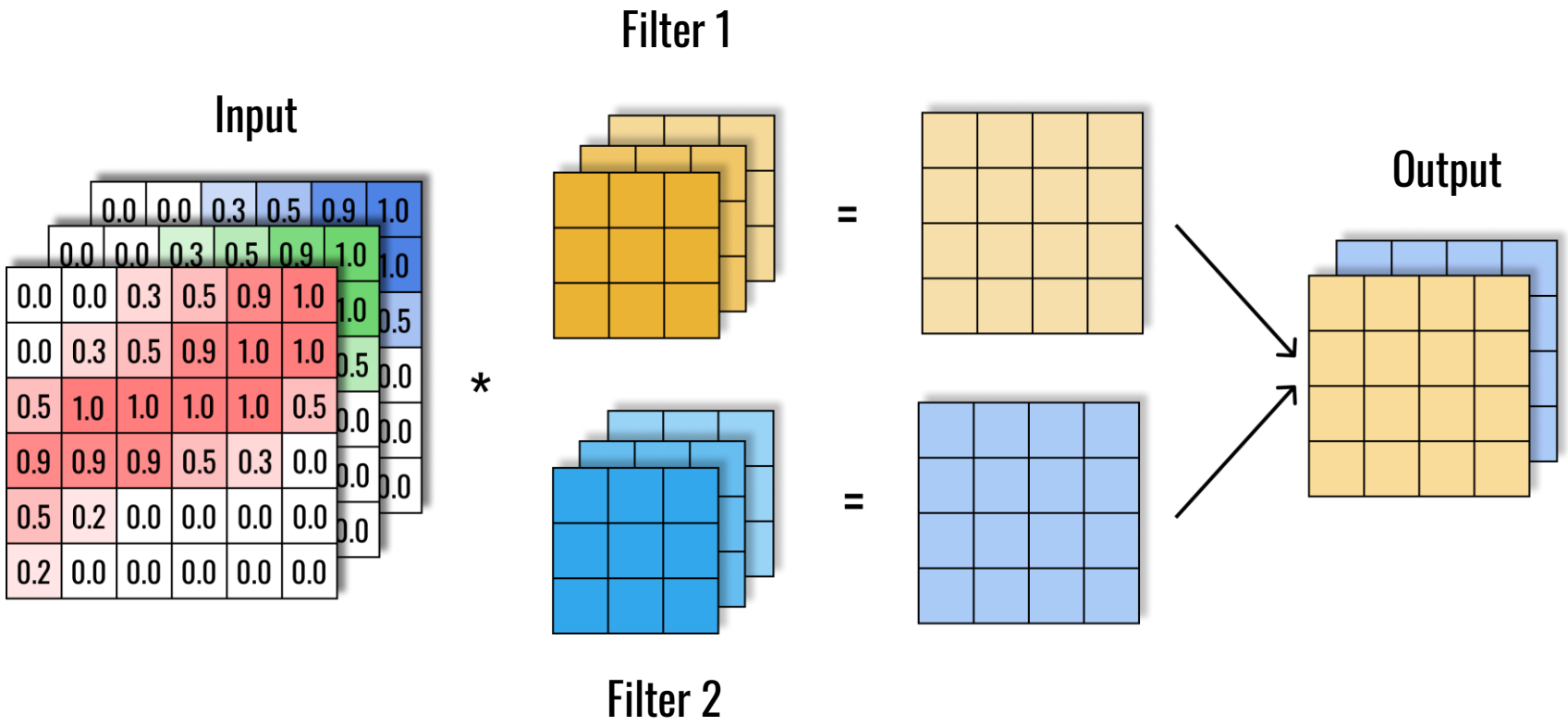


Image Source: Convolution arithmetic visualization, adapted from Vincent Dumoulin, & Francesco Visin. (2018). A guide to convolution arithmetic for deep learning.; original visual assets from the conv_arithmetic GitHub repository, MIT License.

Convolutional layer – Multiple filters

We can apply multiple filter groups at the same time on the input image



Convolutional Layer

Summary of convolutions

n × m **image** f × f **filter** c **channels**

padding p **stride** s

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{m + 2p - f}{s} + 1 \right\rfloor \times \mathbf{c}$$

Example:

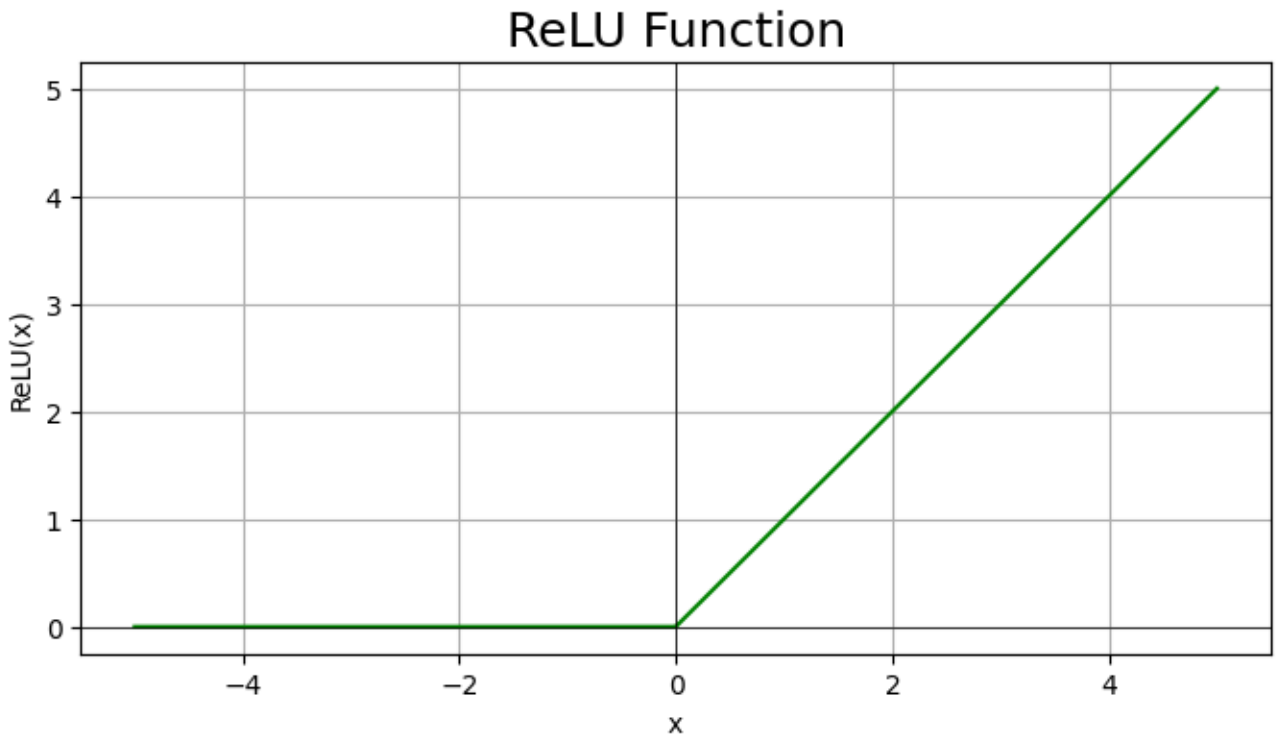
Input Image = 224x224x3

Filter = 3x3x3 (x32)
n = 224; p = 0; s = 1; f = 3; c_o = 32

Output = 222x222x32

ReLU activation function

- The Rectified Linear Unit (ReLU) is a common activation function between convolution layers
- More on this later…



$$\mathrm{ReLU}(x) = \max(0, x)$$

Pooling layer – Max Pooling

Similar to convolution, but instead of taking the weighted sum of the region we only take the maximum value in that region

Max Pooling
Filter: 2x2
Stride: 2

1

4	6	1	1
1	3	1	3
4	0	0	8
8	5	4	0

4	

2

4	6	1	1
1	3	1	3
4	0	0	8
8	5	4	0

4	3

3

4	6	1	1
1	3	1	3
4	0	0	8
8	5	4	0

4	3
8	

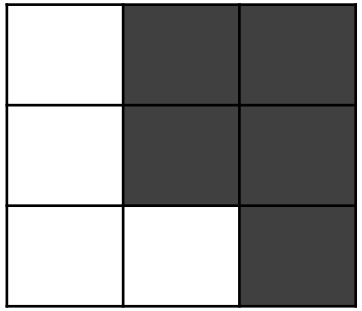
4

4	6	1	1
1	3	1	3
4	0	0	8
8	5	4	0

4	3
8	8

Convolutional layer – Convolution properties

- **Translation Equivariant**



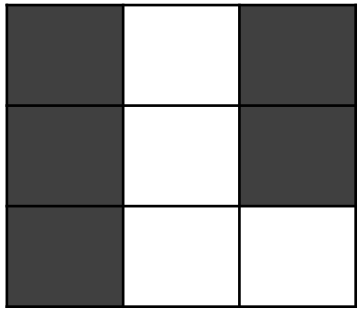
Input 3x3		
20	0	0
20	0	0
20	20	0

*

Filter 2x2	
2	0
0	2

=

Output 2x2	
40	0
80	0



0	20	0
0	20	0
0	20	20

*

2	0
0	2

=

40	40
40	80

Translation equivariance means that the model responds predictably to translations

$f\big(t(x)\big) = t\big(f(x)\big)$,
where **t** is a transformation

Both the **input** and the **output** shifted to the **right**!

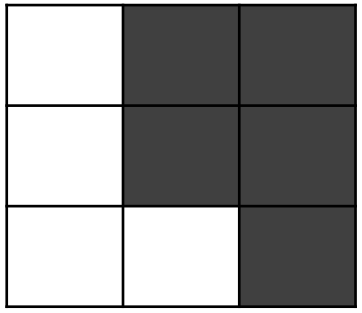
Pooling layer – Max Pooling properties

- **Translation Invariant**

Translation invariance means that the model produces the same output regardless of translations

$$f\big(t(x)\big) = f(x),$$

where **t** is a transformation



Input 3x3		
20	0	0
20	0	0
20	20	0

* Filter 2x2

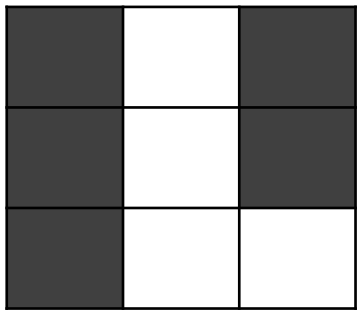
2	0
0	2

= Output 2x2

40	0
80	0

Pooling 2x2 → 80

Change in the **input** results in the **same output**



0	20	0
0	20	0
0	20	20

*

2	0
0	2

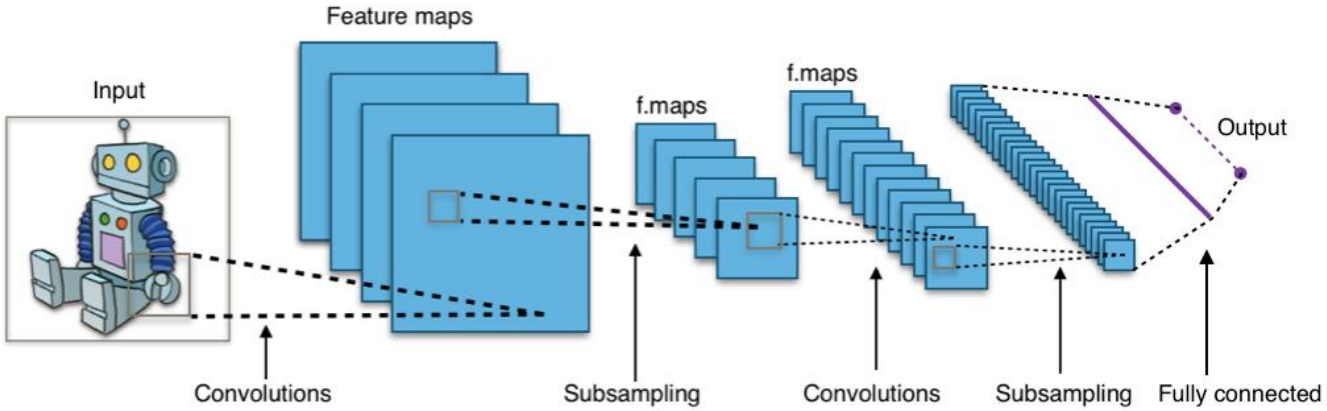
=

40	40
40	80

Pooling 2x2 → 80

Using convolution and pooling together we can ensure that the model will be able to detect object at different positions on the image

Flatten and Fully Connected layer



Before feeding the extracted features to the Fully Connected Layer we have to flatten the extracted feature map

1	1	0
4	2	1
0	2	1

Flattening →

1
1
0
4
2
1
0
2
1

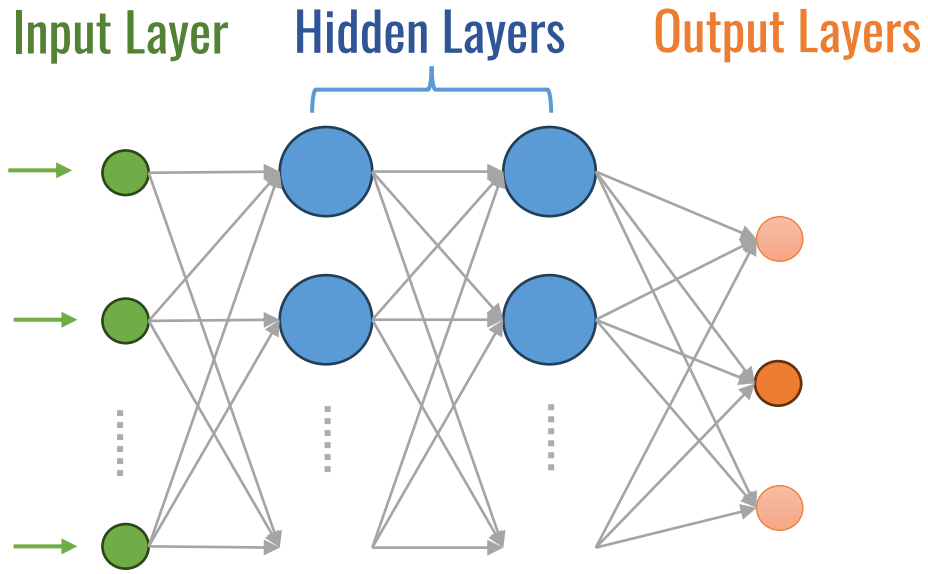


Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Convolutional Neural Network

Visualization:

- https://poloclub.github.io/cnn-explainer/
- https://distill.pub/2017/feature-visualization/

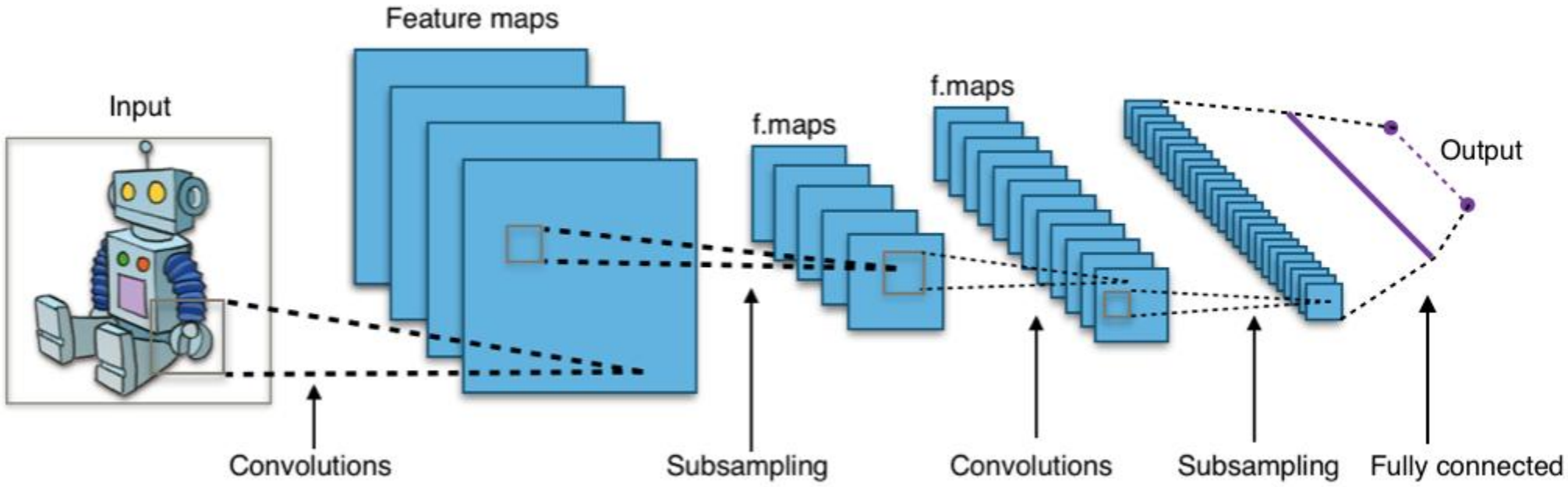


Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

“Neocognitron” Kunihiko Fukushima (1979)

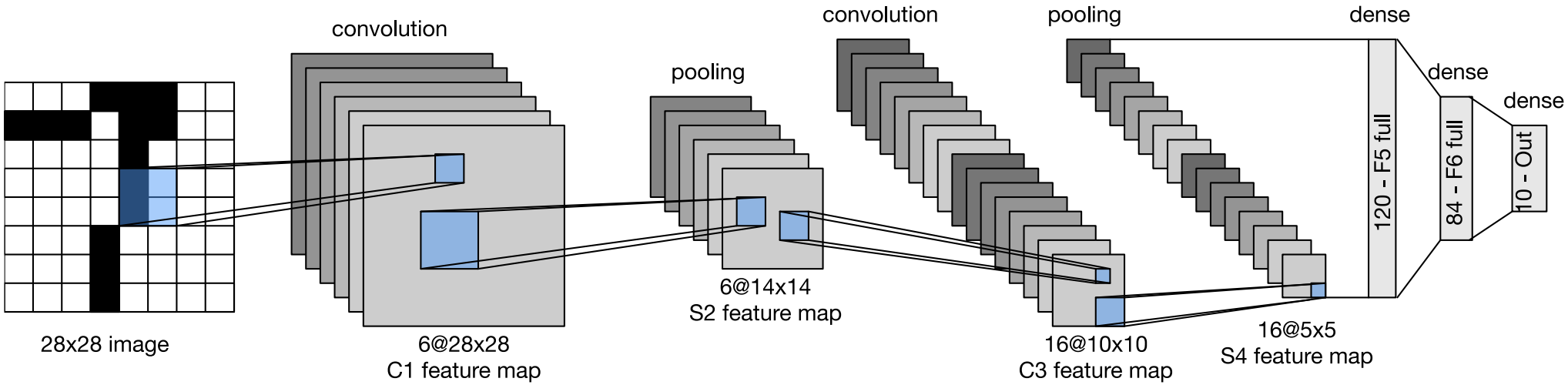
Where it started - handwritten character recognition and other pattern recognition



Fukushima, K. (1980). Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4), 193–202. https://doi.org/10.1007/bf00344251
Image Source: MnistExamples.png, Josef Steppan, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

“LeNet 5” Y. Lecun et al. (1998)

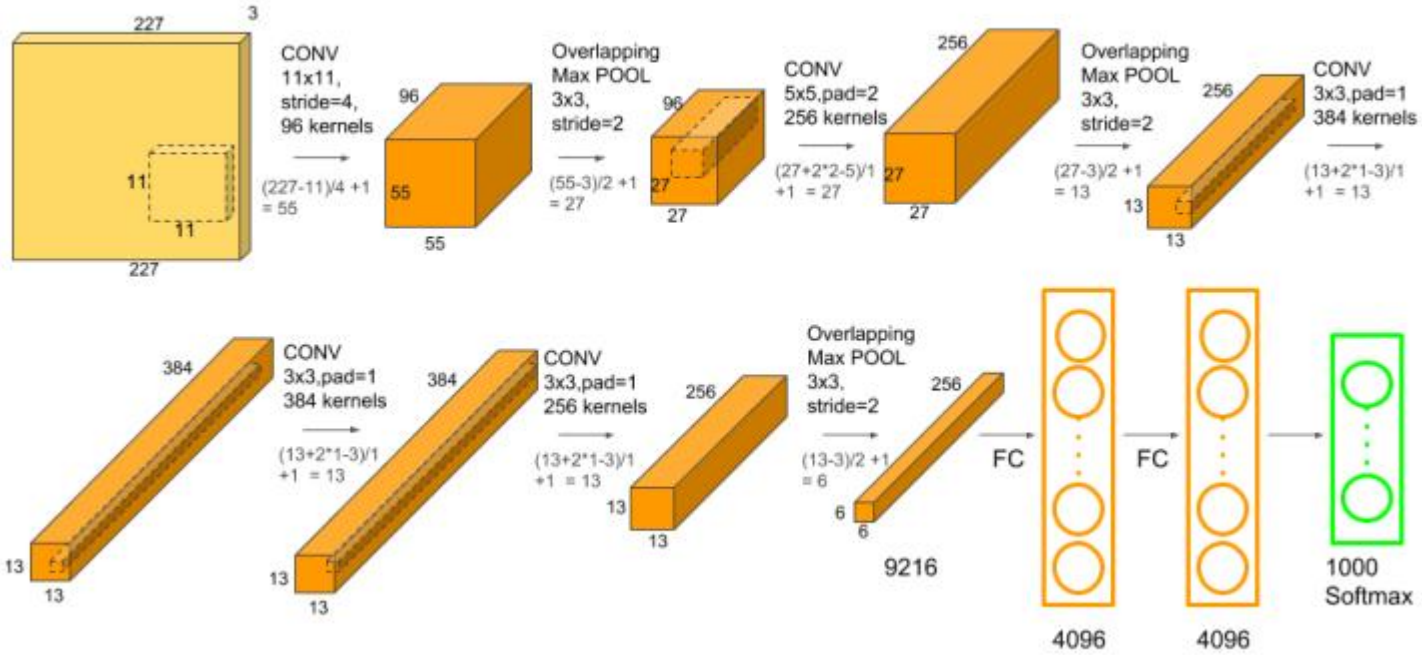
- CNNs were popularized for image recognition by Yann LeCun and collaborators through the LeNet family of models, especially for handwritten digit recognition.



Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
Image Source: LeNet-5_architecture.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

“AlexNet” A Krizhevskyet al. (2012)

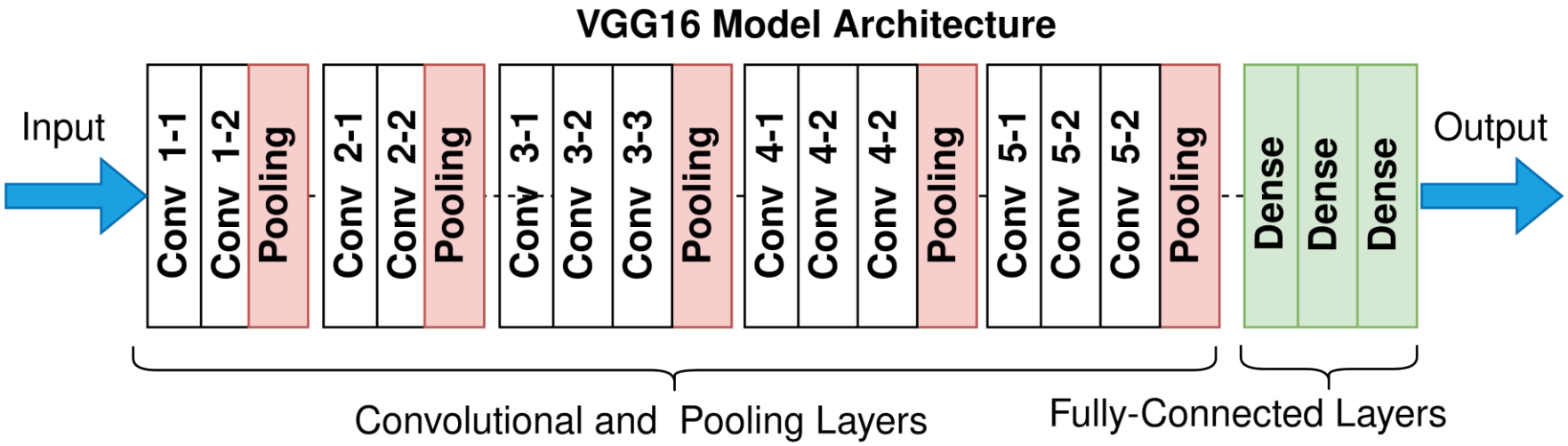
- Trained on the ImageNet dataset
- Contains more than 1M images
- 1000 classes
- They trained a deep convolutional neural network to classify images



Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems. Curran Associates, Inc..
Image Source: Deep CNNs for large scale species classification - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/AlexNet-architecture-adapted-from-8_fig1_349025628, Figures - available via license: Creative Commons Attribution 4.0 International

"VGGNet" Simonyan and Zisserman (2014)

- The number of filters doubles in each block

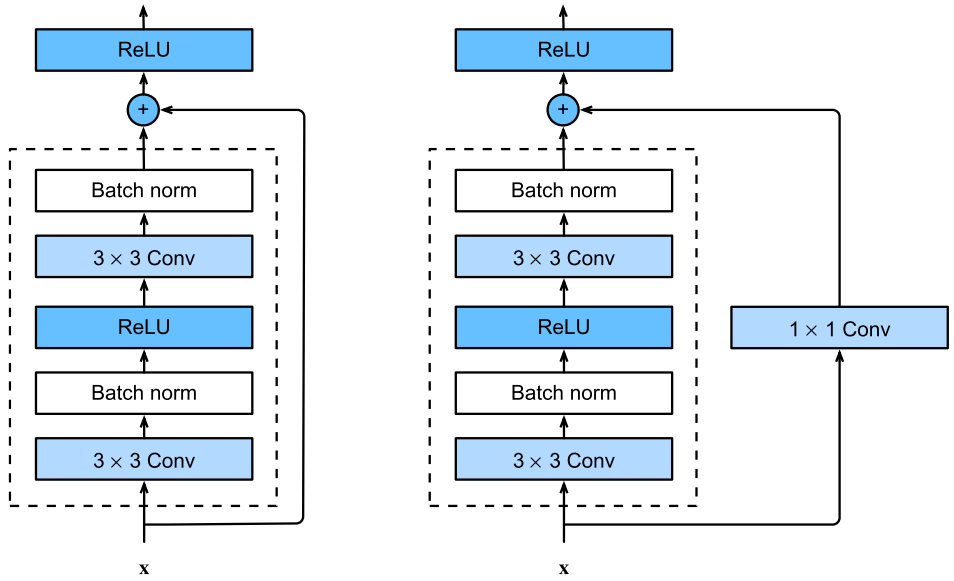


Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. https://arxiv.org/abs/1409.1556
Image source: VGG16.png, Gorlapraveen, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

"ResNet" Kaiming He et al. (2015)

- Introduced residual connections between convolutional blocks
- More on this later…

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90
Image source: ResNet block.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image source: ResNetTypes.png, Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Summary

- **Fully Connected Networks** (Feed Forward Networks) calculate a weighted sum of the inputs
- We can add non-linear activation functions like **Sigmoid** to transform linear regression to logistic regression (classification)

- **Classification** is a supervised learning task
  - Binary: the target is 0 or 1 - **Sigmoid**
  - Multiclass: the target is one element out of a discrete set of elements (usually use one-hot encoding) - **Softmax**

- **Convolutional Neural Networks** (CNNs) are efficient algorithms for images
  - Weight sharing
  - Capture spatial patterns

- By changing the CNN configurations, we create **different architectures**

Resources

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning
- https://developers.google.com/machine-learning/crash-course/embeddings

Papers

1. Fukushima, K. (1980). Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4), 193–202. https://doi.org/10.1007/bf00344251
2. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
3. Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems. Curran Associates, Inc..
4. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. https://arxiv.org/abs/1409.1556
5. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

Image Sources

- Unless otherwise noted, the figures were created by the instructor
- Python code examples were created by the instructor and rendered with Carbon.
- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.

- Tim illustration adapted from Curious While Walking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0, via Wikimedia Commons
- Tom illustration: Smirking Cartoon Businessman.svg by Clip Art by Vector Toons, CC BY-SA 4.0 , via Wikimedia Commons

- Dancing Robot: Gif robot.git, Meks01, CC BY-SA 4.0, via Wikimedia Commons

- Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0, via Wikimedia Commons
- Convolution arithmetic visualization, adapted from Vincent Dumoulin, & Francesco Visin. (2018). A guide to convolution arithmetic for deep learning.; original visual assets from the conv_arithmetic GitHub repository, MIT License.

Image Sources (cont.)

- Oso Flaco Lake Raccoons - Flickr - docentjoyce.jpg, Joyce Cory, CC BY 2.0, via Wikimedia Commons
- MnistExamples.png, Josef Steppan, CC BY-SA 4.0, via Wikimedia Commons
- LeNet-5_architecture.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0, via Wikimedia Commons
- Deep CNNs for large scale species classification, Scientific Figure on ResearchGate, available via license: CC BY 4.0
- VGG16.png, Gorlapraveen, CC BY-SA 4.0, via Wikimedia Commons
- ResNet block.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0, via Wikimedia Commons
- ResNetTypes.png, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, CC BY-SA 4.0, via Wikimedia Commons

Legal Notice

conv_arithmetic — Vincent Dumoulin & Francesco Visin
Source: https://github.com/vdumoulin/conv_arithmetic
License: The MIT License (MIT)

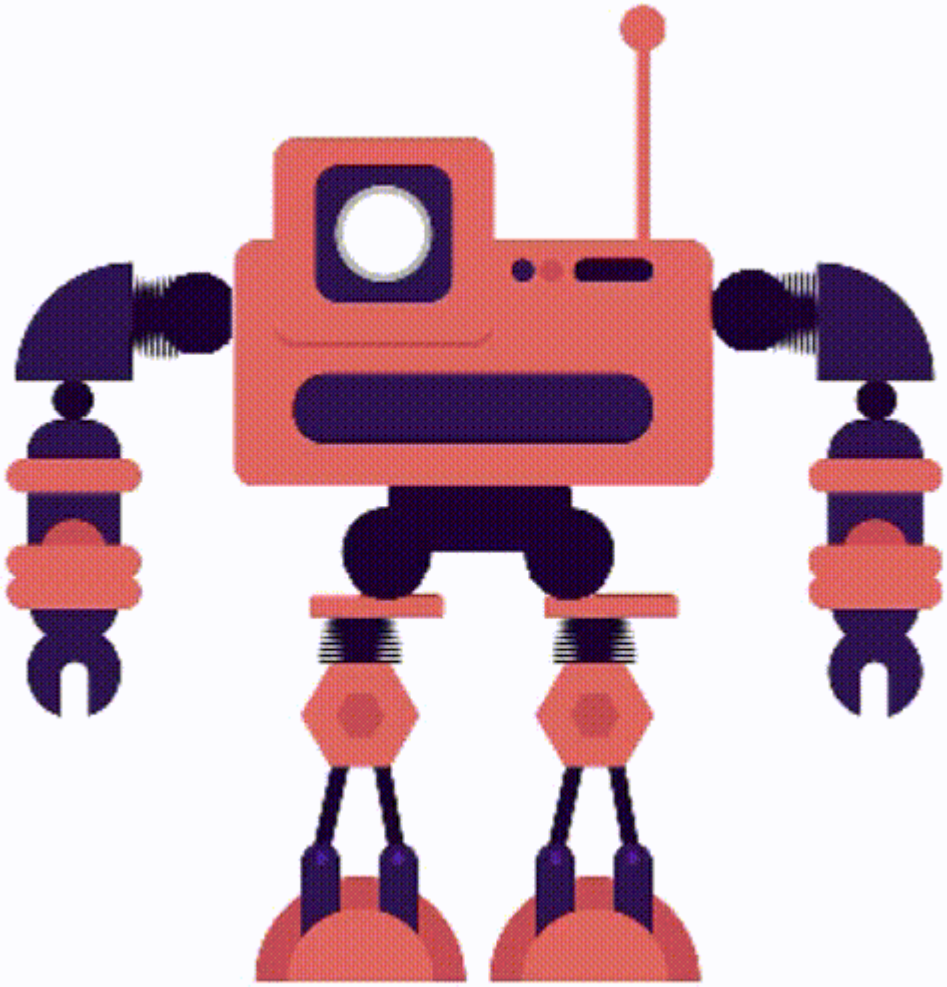
Copyright (c) 2016

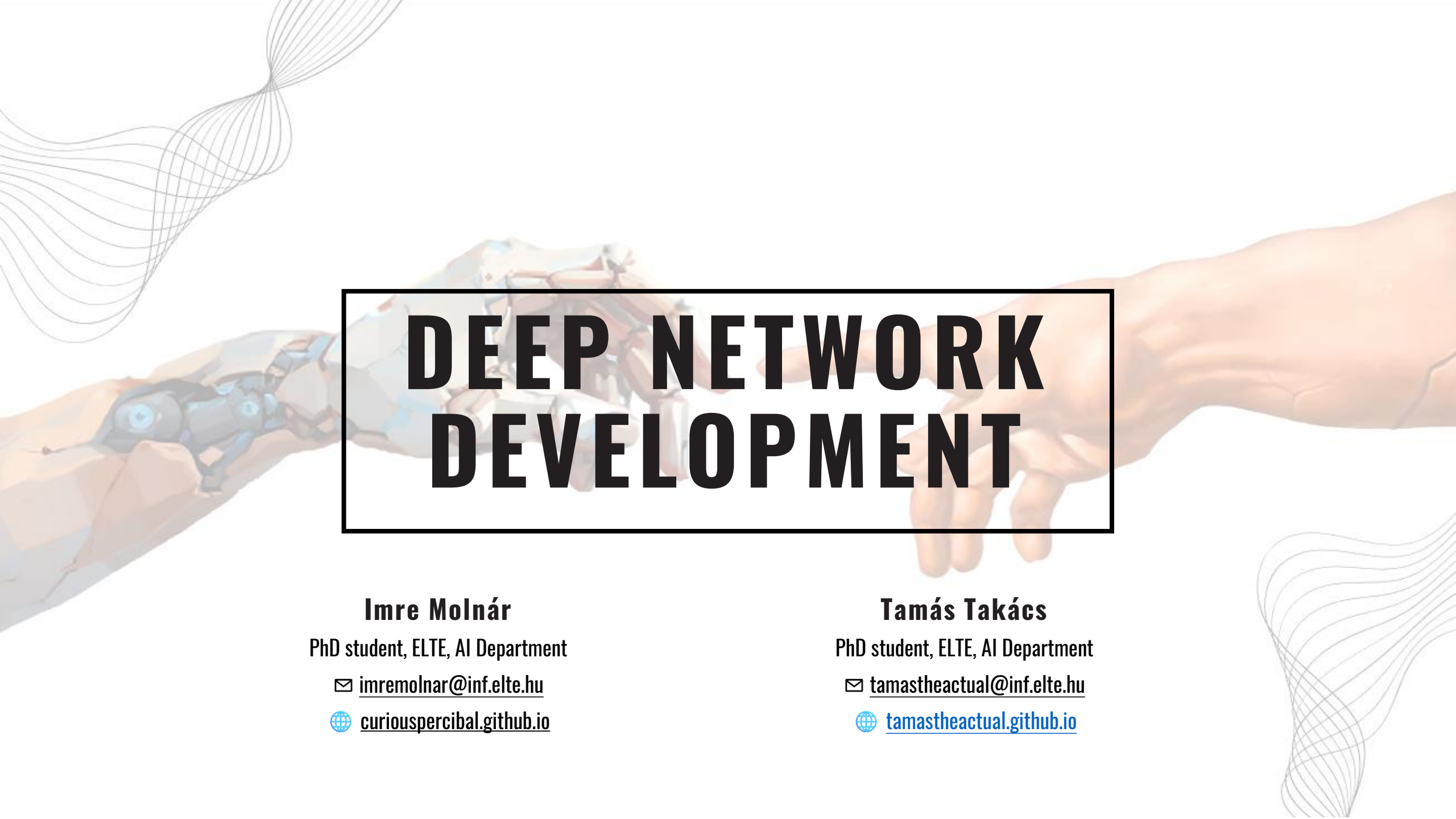
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

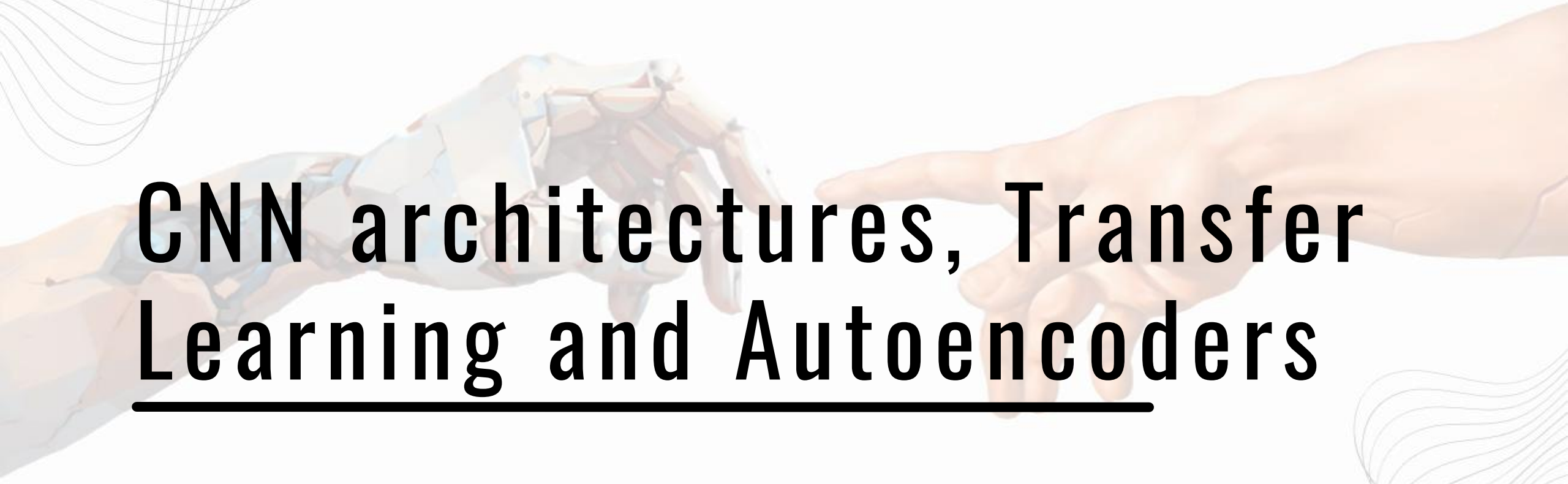
✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 4



CNN architectures, Transfer Learning and Autoencoders

1 CNN Architectures

2 Transfer Learning

3 Autoencoders

Previously on Lecture 3

- With ordered input data, data that are close to each other are more closely related, so stronger/more relationships are needed.

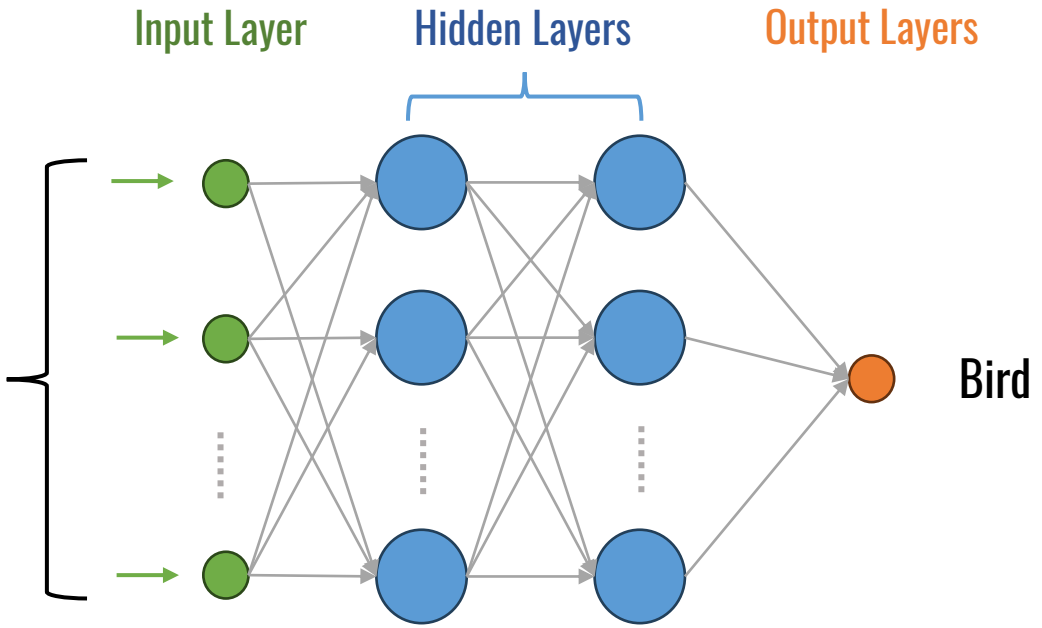
Advantage:

- memory requirements are significantly reduced
- The relationship pattern can be well optimized by exploiting the relationship pattern, thus allowing fast and transparent operation.

Disadvantage:

- More complex network, but still structured



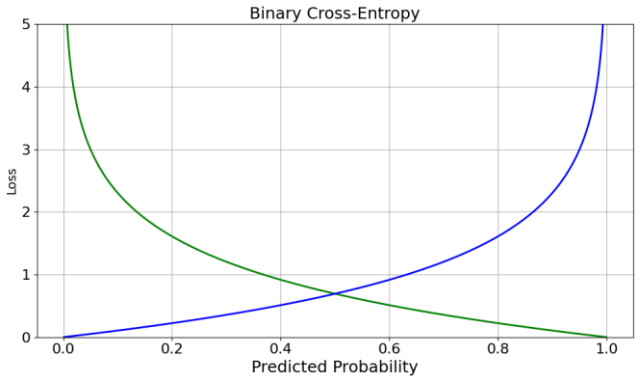


For ordered data, other connection patterns are also possible.

Data that are distant from each other may not even be related (e.g., for time series, causality can be exploited, for images, pixels that are distant from each other are less and less likely to belong to the same object).

Previously on Lecture 3

Binary Classification



- Supervised learning
- Have: (x, y)
  - x: input
  - y: target $\in \{0, 1\}$
- *Goal: Learn a function to map $x \rightarrow y$.*
  - $h(x) = \hat{y}$
  - $\hat{y} = g\left(\sum_{j=0}^{m} w_j x_j + b\right)$ Add non-linear activation function
- Classification – Predict a discrete set of values:
  - $\hat{y} \in \{0, 1\}$
- Binary cross-entropy loss

$$L_{BC}(h(x), y) = -y \log h(x) - (1 - y) \log(1 - h(x))$$

Multi-class Classification

- With K classes
- The network will have K outputs
- Softmax activation for squashing outputs between 0 and 1 (to mimic Probability distribution)
- $\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$

One-hot encoding (K=3):
(cat) 1 -> [1, 0, 0]
(dog) 2 -> [0, 1, 0]
(horse) 3 -> [0, 0, 1,]

Output logits

Cat	Dog	Horse
4	2	8

Applying softmax on the outputs

Class probabilities

Cat	Dog	Horse
0.02	0.00	0.98

Categorical Cross-entropy loss

$$L(h(x), y) = -\sum_{i=1}^{K} y_i \log h_i(x)$$

Previously on Lecture 3

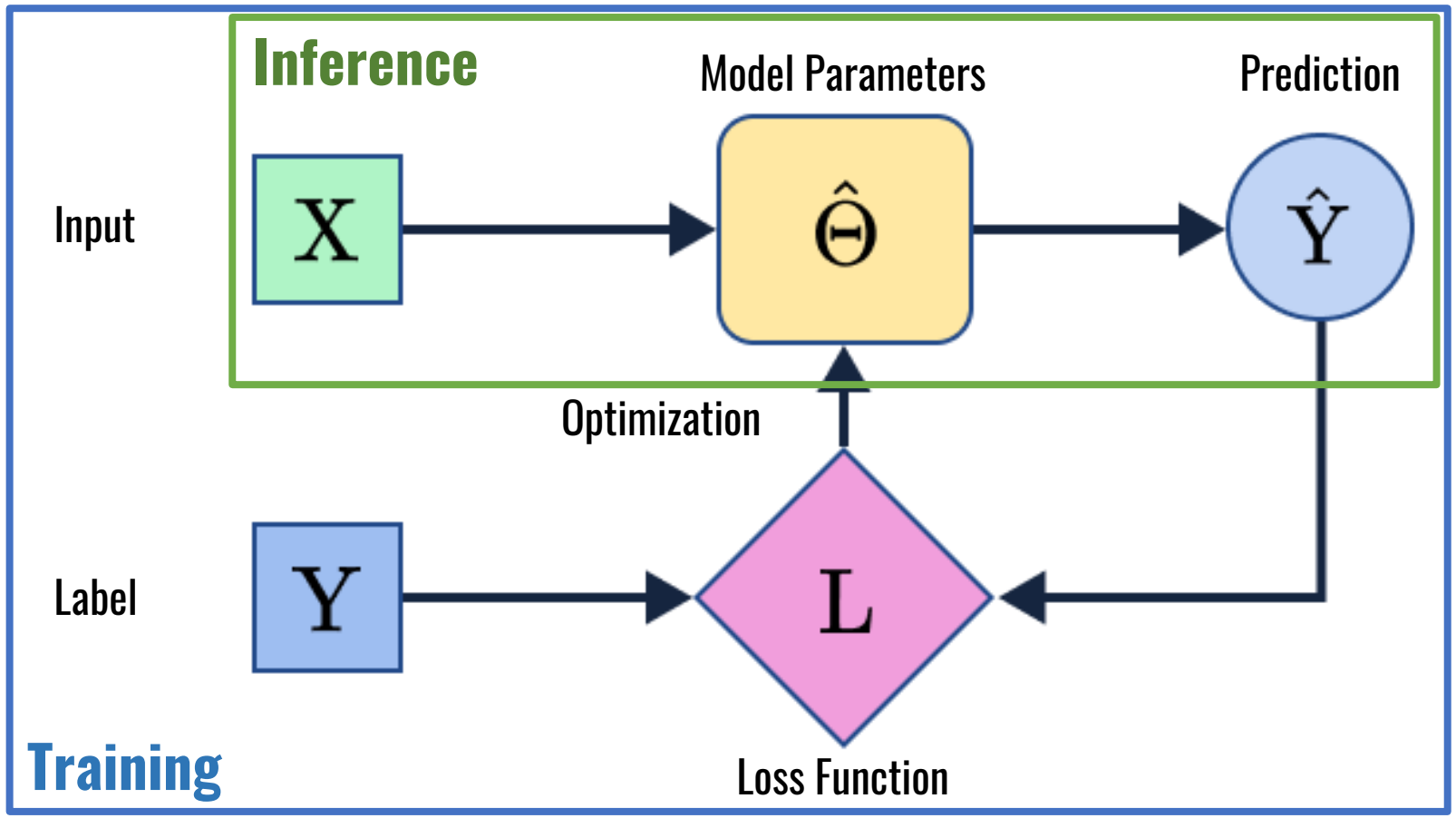
- Given $h = g\left(W^{(2)}\boldsymbol{g}\left(\boldsymbol{W^{(1)}x} + \boldsymbol{b^{(1)}}\right) + b^{(2)}\right)$
- Update $\boldsymbol{W^{(1)}}$:

$$\boldsymbol{W_t}^{(1)} = \boldsymbol{W_{t\_1}}^{(1)} - \alpha\nabla J(\boldsymbol{W^{(1)}})$$

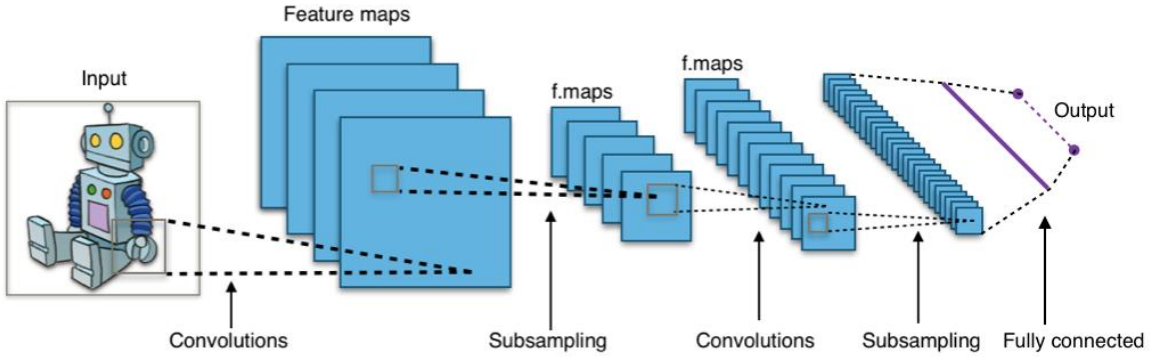
- Update $\boldsymbol{W^{(2)}}$:

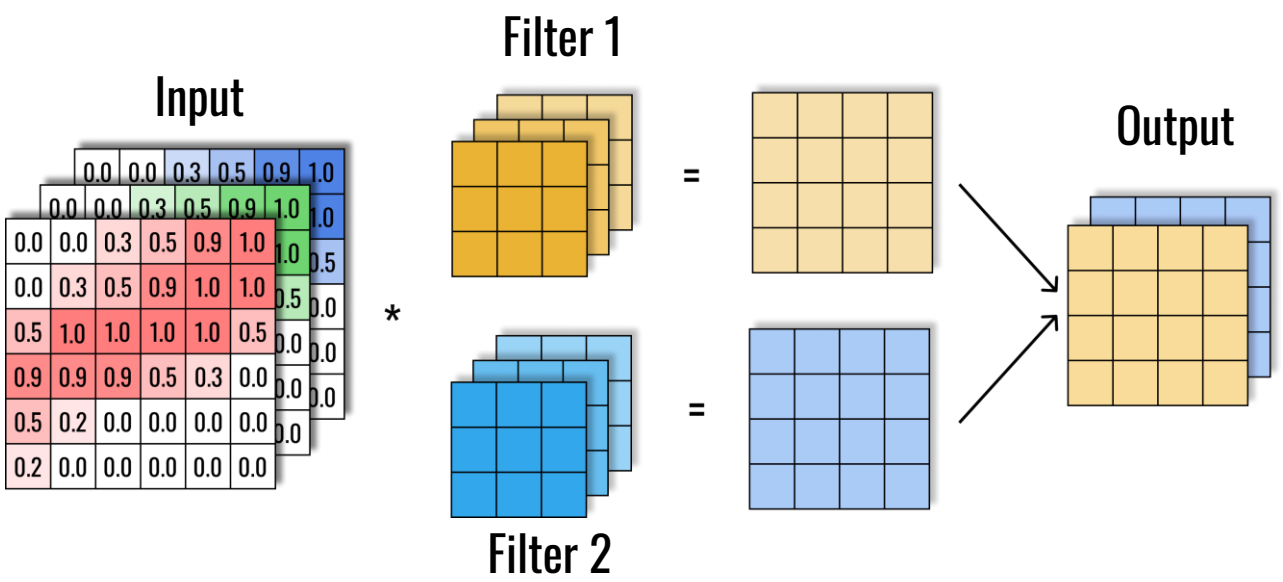
$$\boldsymbol{W_t}^{(2)} = \boldsymbol{W_{t\_1}}^{(2)} - \alpha\nabla J(\boldsymbol{W^{(2)}})$$

- Same for bias



Previously on Lecture 3





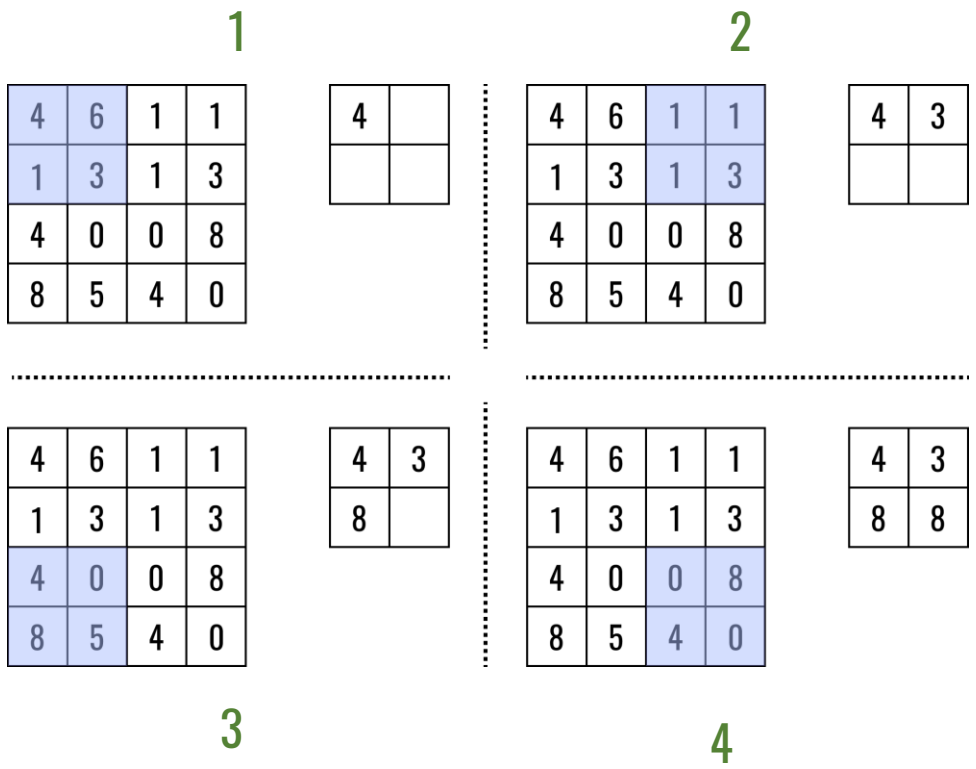
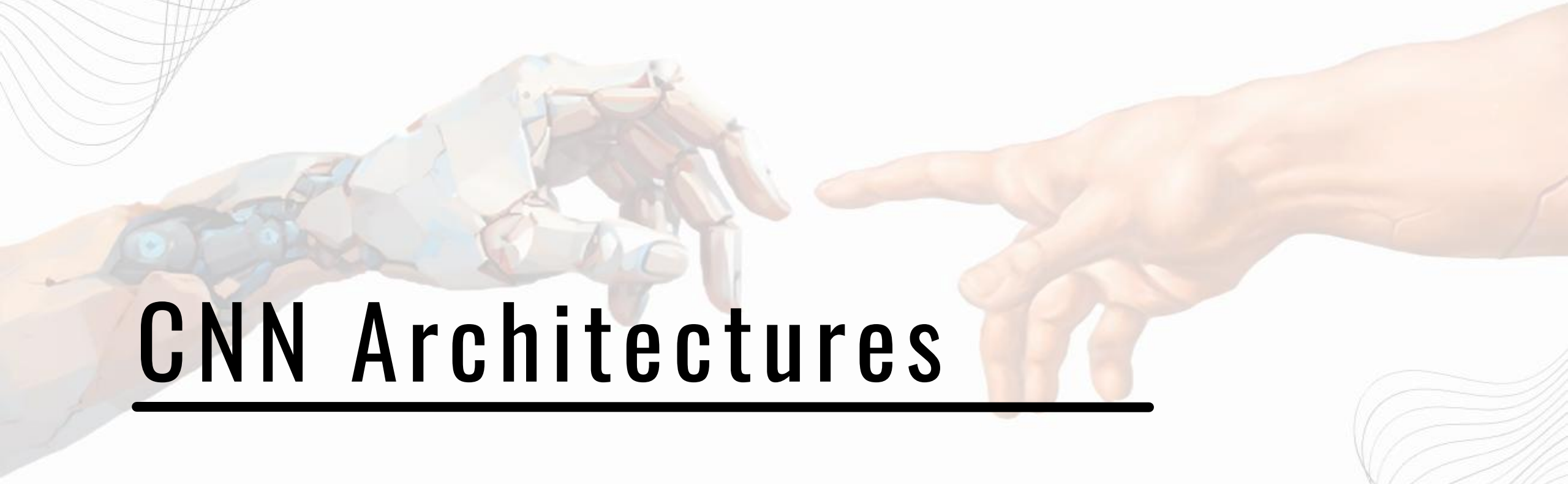


Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Deep Network Development

Lecture 4



CNN Architectures

1 CNN Architectures

2 Transfer Learning

3 Autoencoders

Convolutional Layer

- Purpose: To quantify the correspondence with the characteristics defined by the weights of the filters
- E.g.: how much of a (vertical edge)/nose/wheel/face is in the image part under consideration
- Play

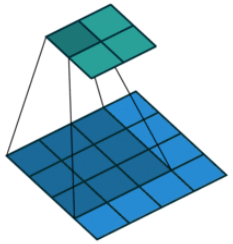
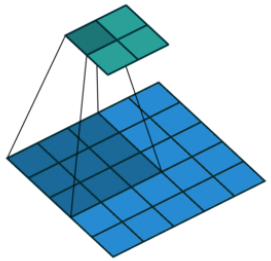
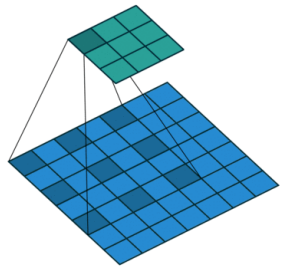
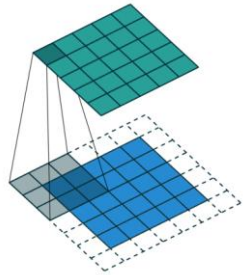
Kernel size 3x3	Stride 2,2	Dilatation 2,2	Padding 1,1,1,1 (up, down, left, right)
			

Image Source: Convolution arithmetic visualization, adapted from Vincent Dumoulin, & Francesco Visin. (2018). A guide to convolution arithmetic for deep learning.; original visual assets from the conv_arithmetic GitHub repository, MIT License.

Convolutional Layer

Summary of convolutions

n × m **image** f × f **filter** c **channels** **padding** p **stride** s

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{m + 2p - f}{s} + 1 \right\rfloor \times \mathbf{c}$$

Example:

Input Image = 224x224x3

Filter = 3x3x3 (x32)
n = 224; p = 0; s = 1; f = 3; m = 32

Output = 222x222x32

Going deeper

What prevents us from creating CNN with arbitrary depth?

- Underfitting/Overfitting
- Vanishing/Exploding gradient
- Generalization
- And many more...

Underfitting/Overfitting

Underfitting

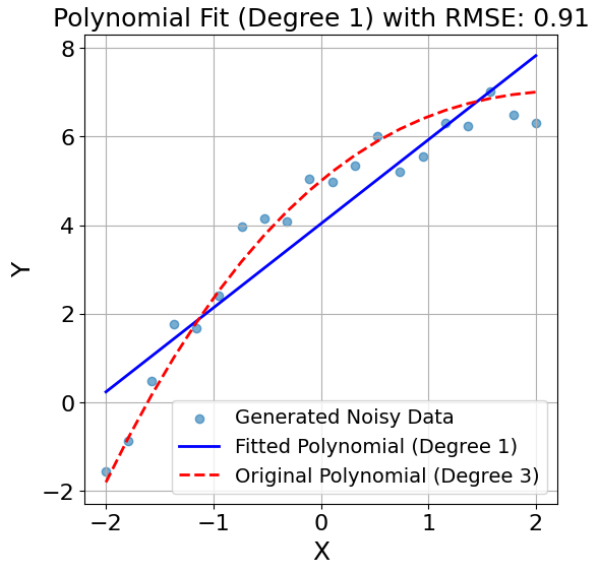
- The model is too simple for the data. It cannot capture the underlying pattern, so it performs poorly both on the training data and on new data.

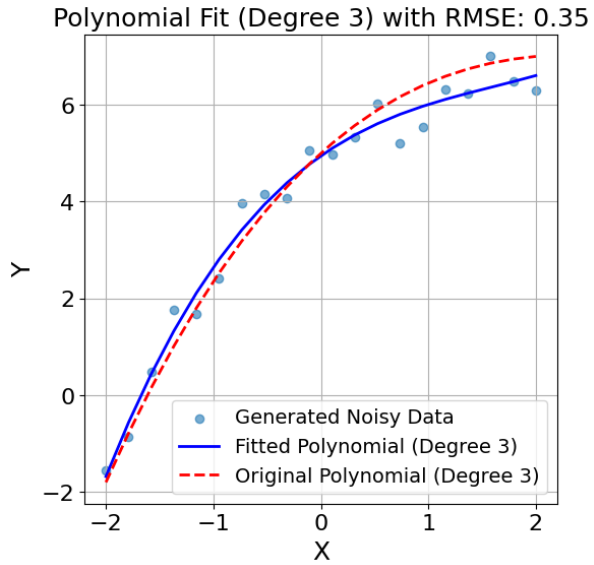
Just right

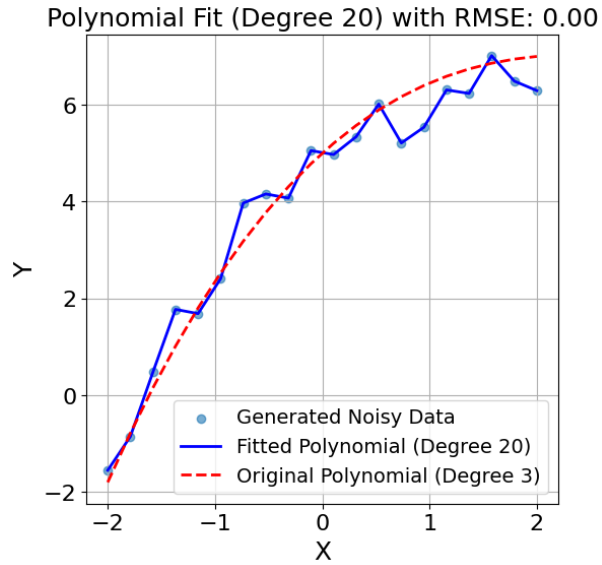
- The model captures the main pattern in the data without trying to match every individual data point. It generalizes well to new data.

Overfitting

- The model is too complex and fits the training data too closely. It learns not only the real pattern, but also noise or random fluctuations, so it performs well on training data but poorly on new data.







Underfitting/Overfitting

Underfitting (Solutions)

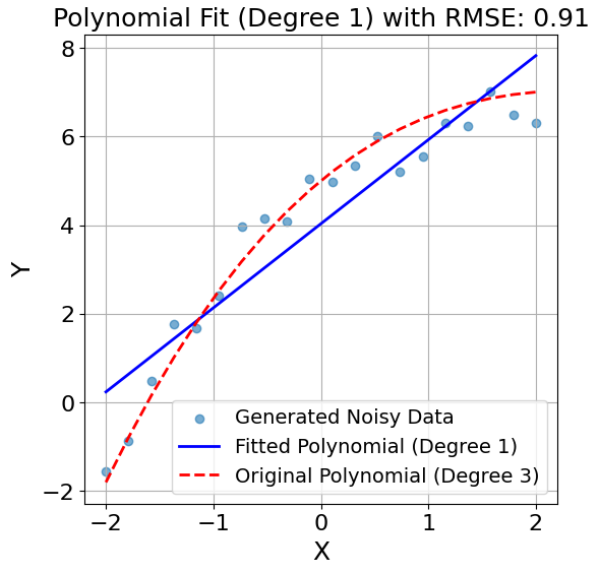
- We can reduce underfitting by training for more epochs, increasing model complexity, adding more parameters, or using better features.

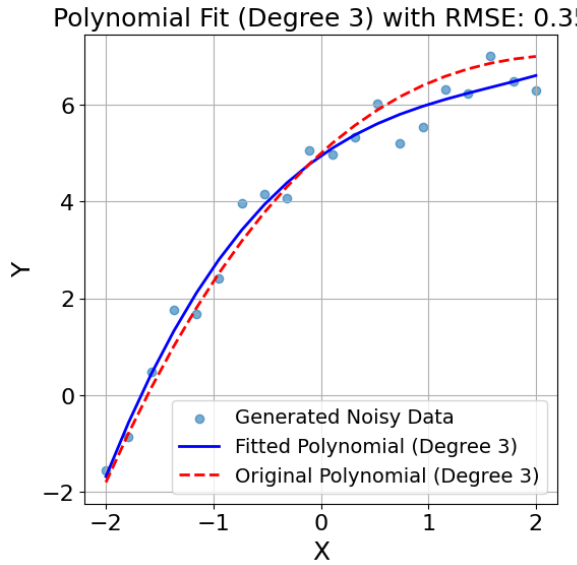
Just right

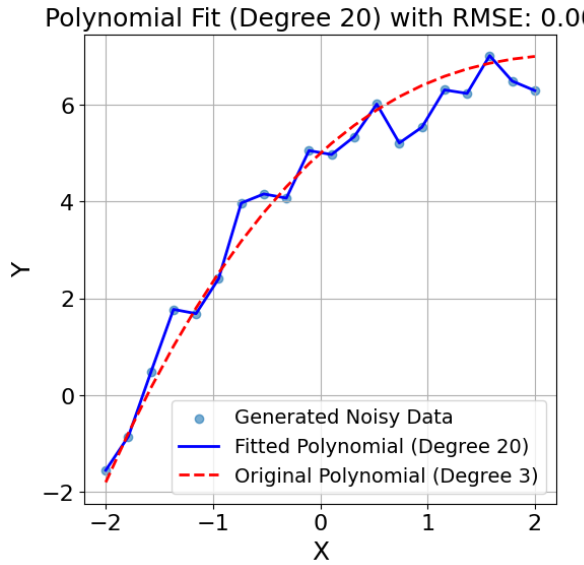
- A good fit can be achieved by choosing suitable hyperparameters and model complexity using validation data, hyperparameter optimization, and model selection.

Overfitting (Solutions)

- Overfitting can be reduced by using regularization methods such as early stopping, dropout, etc., or by reducing model complexity. A model with fewer parameters is usually less likely to memorize the training data.

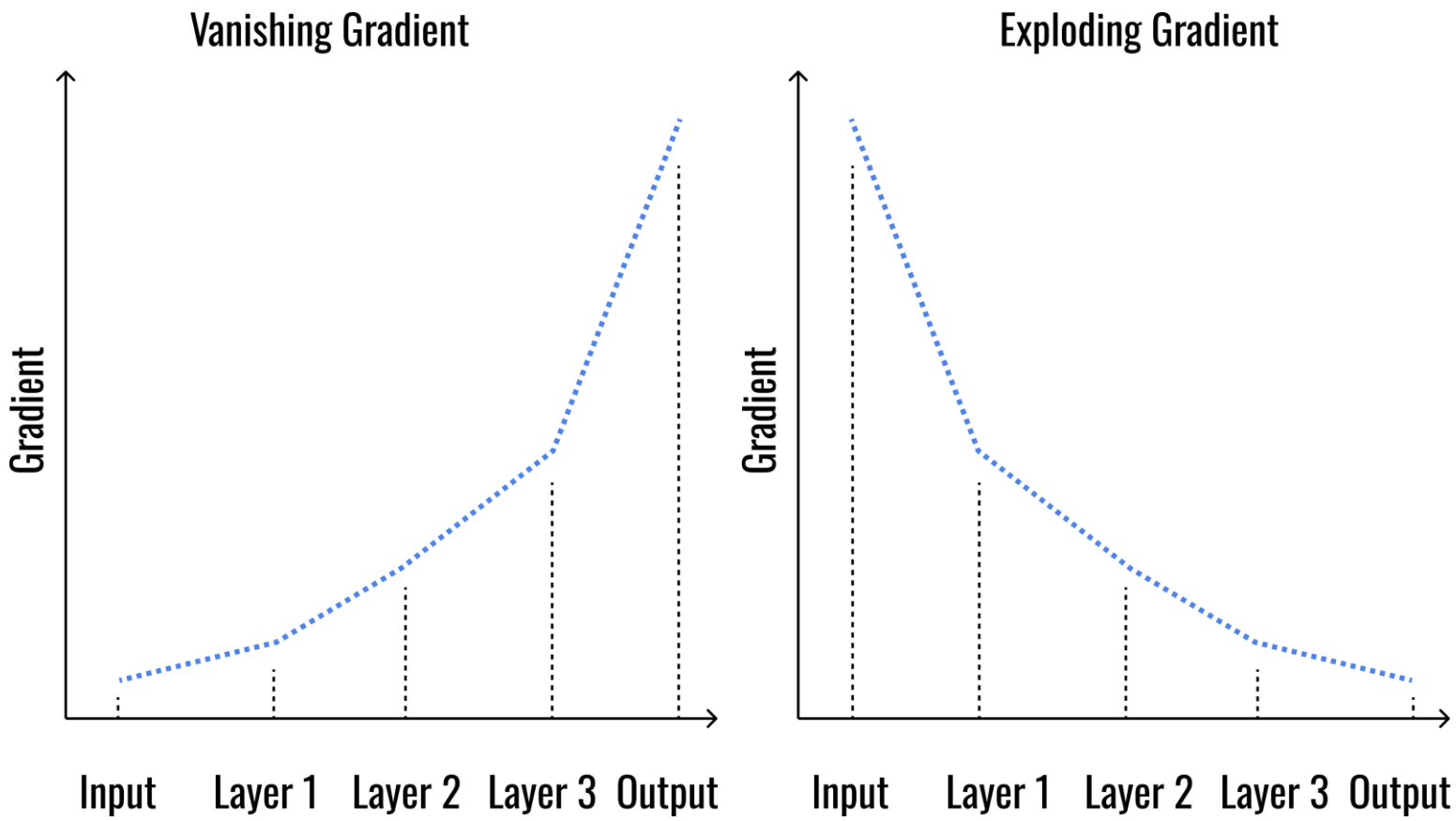






Vanishing/Exploding Gradient

- Weight initialization
- Activation function
- Regularization
  - Batch Normalization
  - Dropout
  - Weight decay
- “Skip connection”
- And many more…



Weight initialization

- As the name suggest specifies the method how we initialize the weight of our neural network
  - Sample from a uniform distribution in $[a, b]$
  - Sample from a normal distribution $N(\mu, \sigma)$ – usually with 0 mean and 1 standard deviation
  - Set it to some constant value
  - Or manually add the initial weights

- In pytorch the weights of the Convolution is sampled from $U(-\sqrt{k}, \sqrt{k})$, where $k = \frac{groups}{C_{in} * \prod_{i=0}^{1} kernel\_size[i]}$

- And in Linear layers is sampled from $U(-\sqrt{k}, \sqrt{k})$, where $k = \frac{1}{in\_features}$

Regularization

- Weight related regularizations:
  - **L1 regularization** (also called LASSO) leads to sparse models by adding a penalty based on the absolute value of coefficients.
  - **L2 regularization** (also called ridge regression) encourages smaller, more evenly distributed weights by adding a penalty based on the square of the coefficients.
- Early stopping
- Data augmentation
- Batch normalization
- Dropout
- And many more...

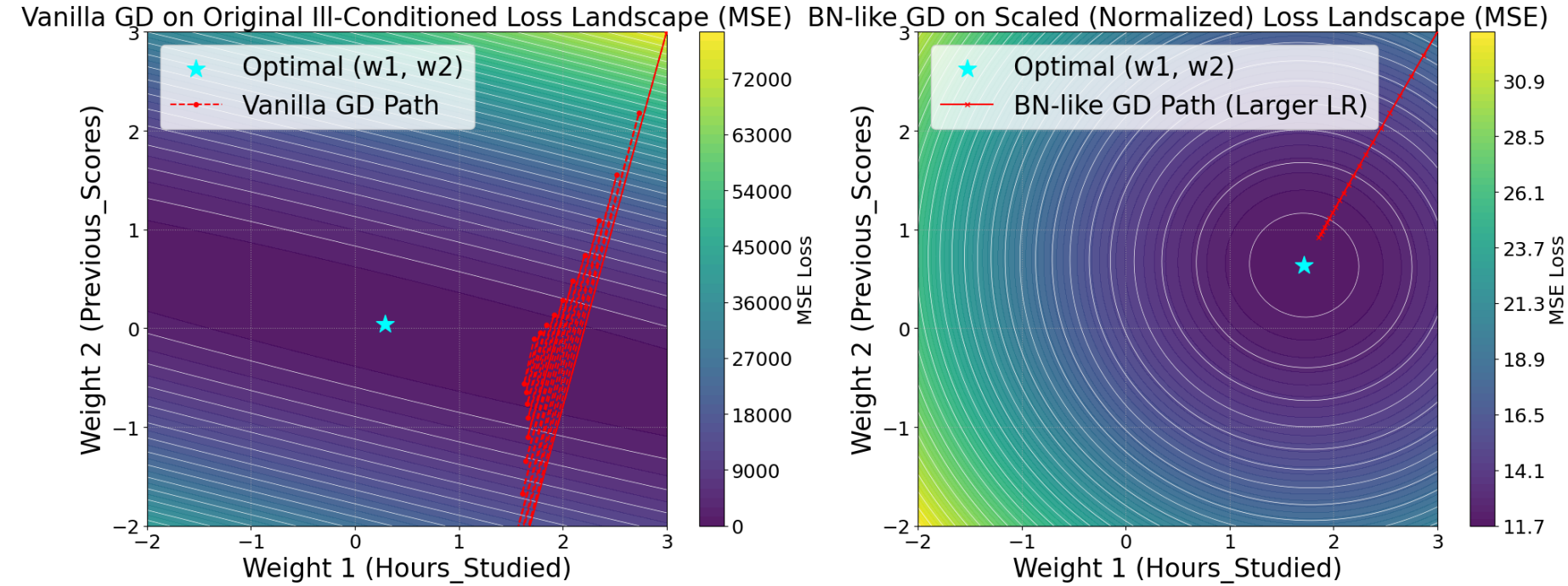
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i) + \lambda \sum_{j=0}^{M} |W_j|$$

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i) + \lambda \sum_{j=0}^{M} W_j^2$$

Batch Normalization

- Batch normalization is achieved through a normalization step that fixes the means and variances of each layer's inputs
- It basically normalizes the output of the previous layer in this batch
- It also keeps track of moving averages, and can do some transformations too

$$y_B = \frac{x_B - \mathbf{E}[x_\mathrm{B}]}{\sqrt{\mathbf{Var}[x_\mathrm{B}]}}$$



ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- 1000 classes
- 1,281,167 training images, 50,000 validation images and 100,000 test images
- Serves as the common benchmark for neural nets

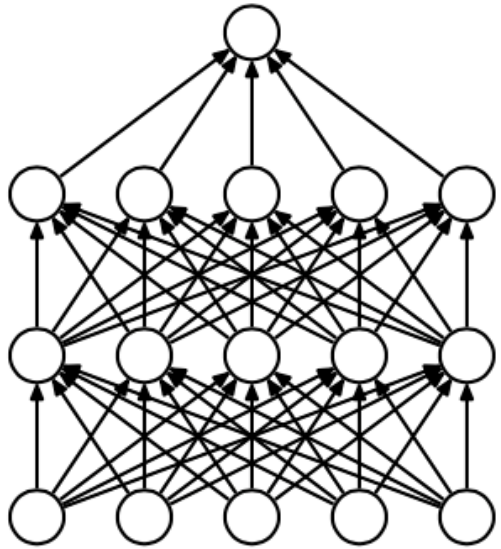
J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

“AlexNet” A. Krizhevsky et al. (2012)

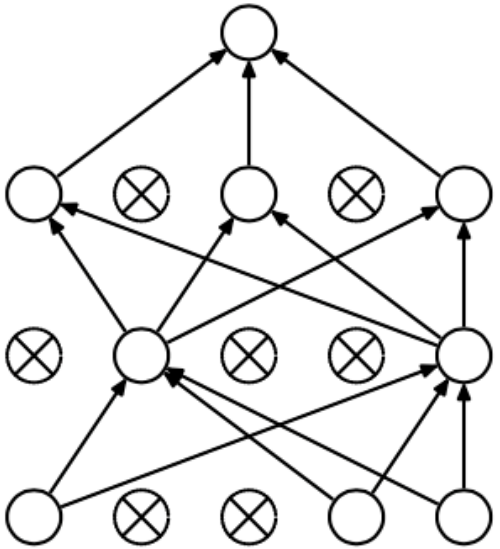
Dropout:

- We are dropping some neurons between layers with some probability **p**.
- The dropped-out neurons do not contribute to the **forward pass** and do not participate in the **backpropagation.**
- “Every time we sample from a different architecture”

- Reduces neuron co-adaptation
- The model forced to learn more robust features



Standard neural net



After droput

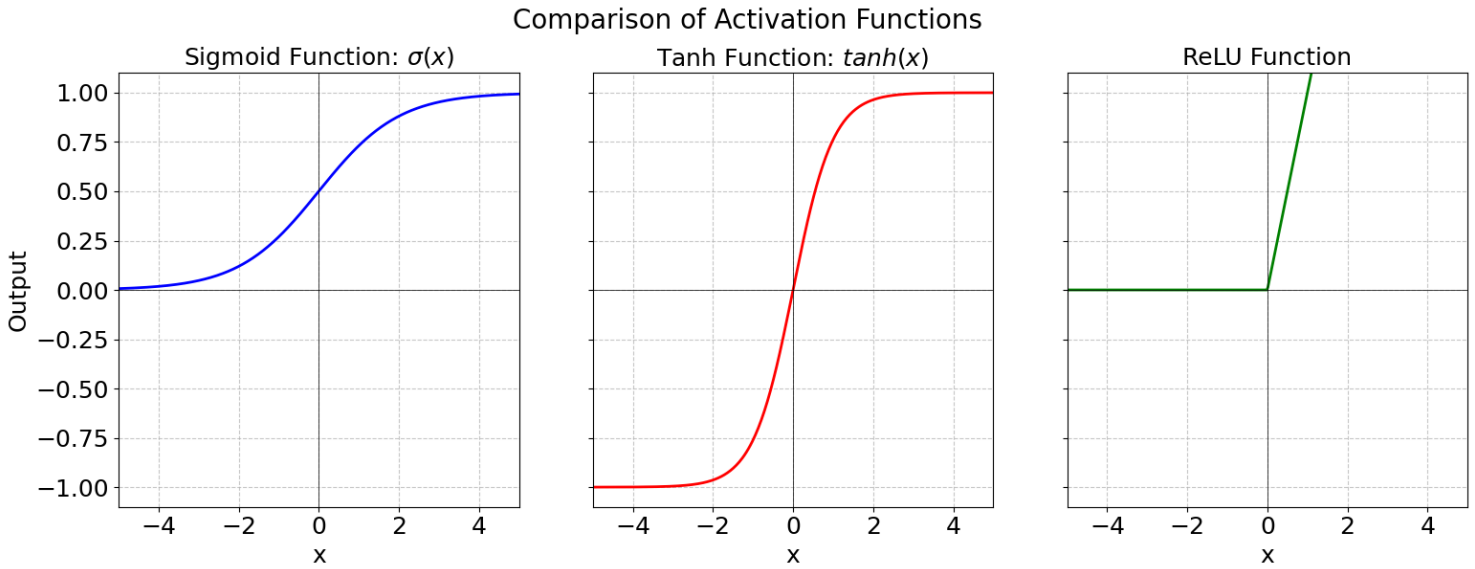
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Image Source: A Convolutional Neural Network for Airport Security Inspection of Dangerous Goods - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Dropout-work-diagram_fig1_334328895, available via license: Creative Commons Attribution 3.0 Unported

“AlexNet” A. Krizhevsky et al. (2012)

ReLU over Sigmoid or tanh:

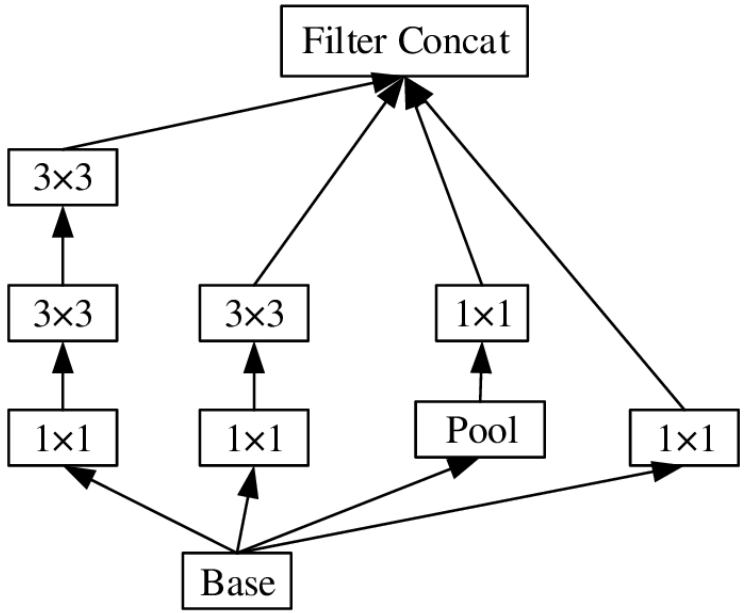
- During training with **Gradient Descent** using **ReLU** (non-saturating nonlinearity) results in less training time compared to **Sigmoid** or **tanh** (saturating nonlinearity).
- Sigmoid is bounded between 0 and 1, tanh bounded between -1 and 1, ReLU only has a lower bound of 0 but no upper bound
- Does not require input normalization
- Learning will happen if at least some training examples produce positive input



Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

“Inception / GoogLeNet” C. Szegedy et al. (2014)

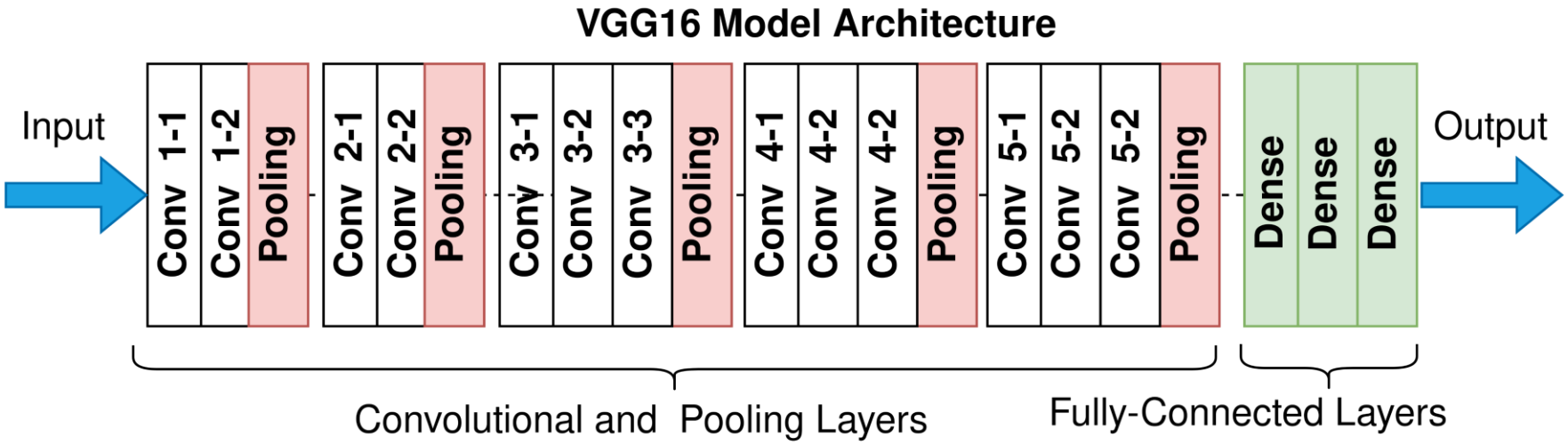
- Aligns with the intuition that visual information should be processed at various scales
- 1x1 Convolution blocks for dimensionality reduction, and multiple kernel sizes parallelly
- Attached multiple classifier network during training that were discarded during inference (inside the architecture)
- This was a regularization method, so the earlier layers are also conditioned to learn useful features for classification, and introduces new “entry points” for the gradients, thus mitigating the vanishing gradient problem



Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Weinberger (Eds.), Advances in Neural Information Processing Systems (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
Image source: Multi-spectral remote sensing images feature coverage classification based on improved convolutional neural network - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/nception-structure_fig3_342413084, available via license: Creative Commons Attribution 4.0 International

“VGG16” K. Simonyan and A. Zisserman (2014)

- Increased the depth of the CNN by using smaller convolutions
- This enables the network the extract more complex hierarchical features
- Used multiple **3x3** convolutions instead of larger **(5x5 or 7x7)** convolutions, while introducing more non-linearity
- Multiple kernel sizes allow objects with different sizes to be detected

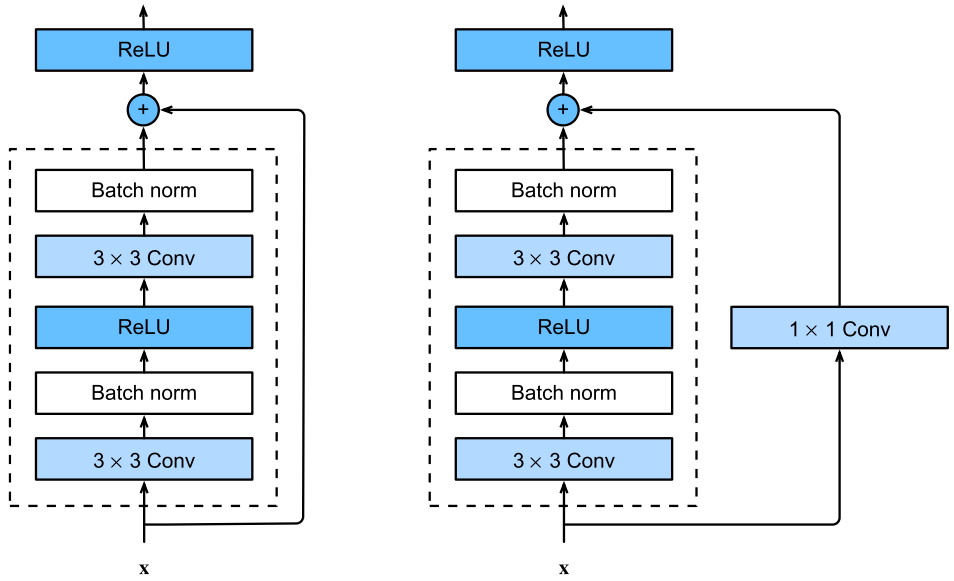


Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. https://arxiv.org/abs/1409.1556
Image source: VGG16.png, Gorlapraveen, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

“Residual Network / ResNet” K. He et al. (2015)

- Learn residual functions reference to layer inputs instead of unreferenced functions
- If we pretrain a smaller network and then we extend it with extra blocks it should have similar or better performance compared to the original network – **Identity mapping**

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90
Image source: ResNet block.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image source: ResNetTypes.png, Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

“DenseNet” G. Huang et al. (2016)

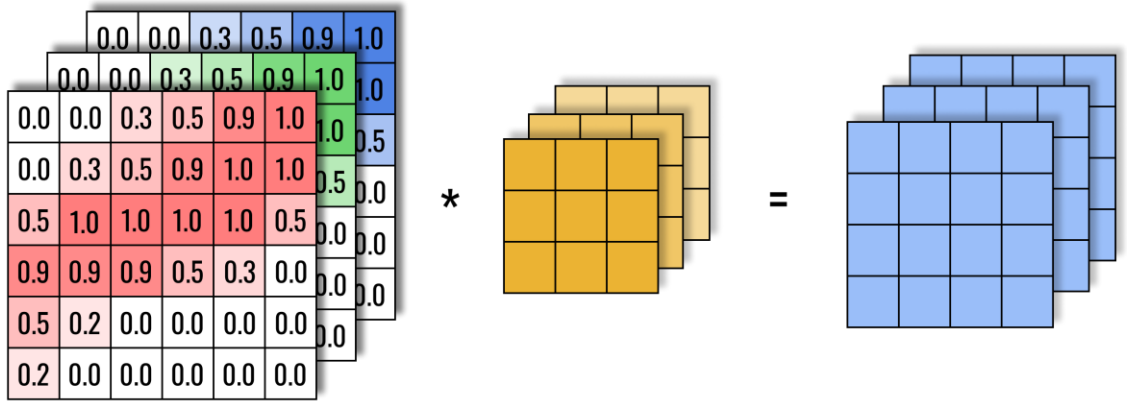
- Multiple residual connections
- Each block is connected with the all of the previous layers with residual connects (densely connected)

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017, July). Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

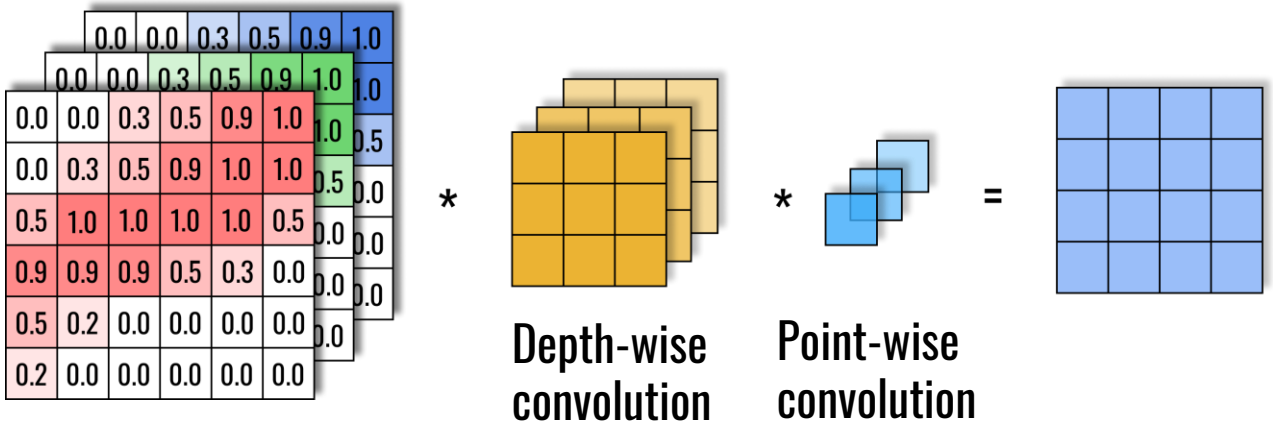
“MobileNet” A. G. Horward et al. (2017)

- It used depth-wise and point-wise convolutions
- **Depth-wise convolution** is like a regular convolution
- **Point-wise convolution** uses a 1x1 kernel and summarizes the features across the feature maps

Normal convolution



Mobilenet



Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1704.04861

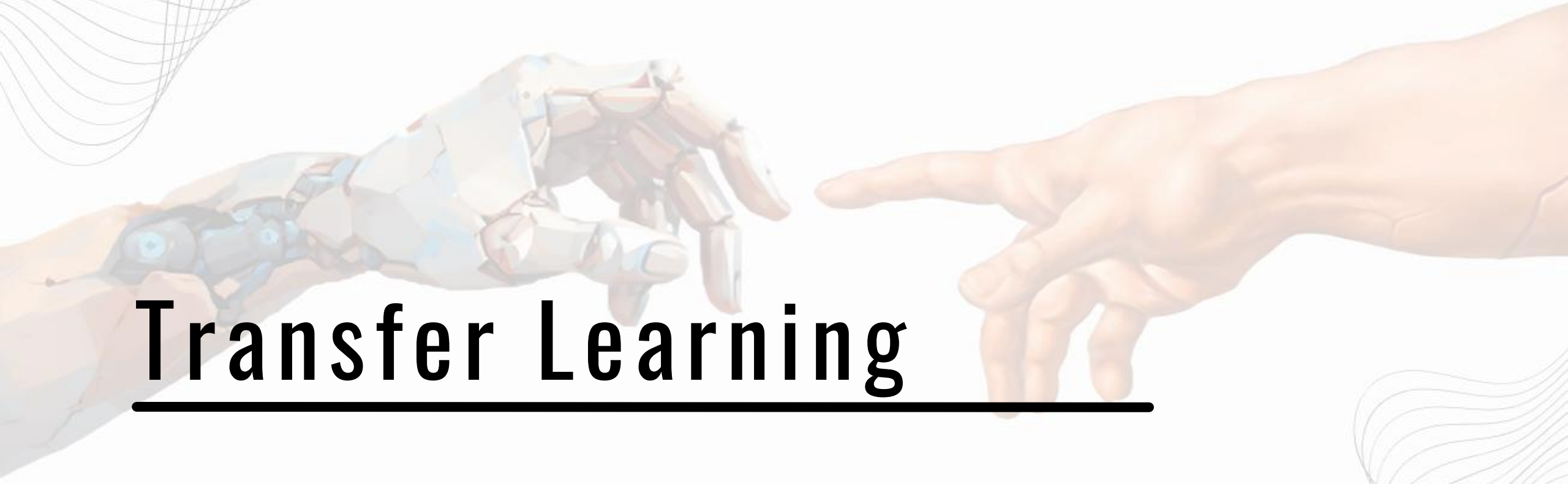
More recent CNN architectures

- **ZfNet**
- **Xception**
- **InceptionV2, V3, V4**
- **Inception-ResNet**
- **MobileNet V3**
- **FractalNet**
- **WideResNet**
- **PyramidalNet**
- **Residual Attention Net**
- **EfficientNet**
- **Etc**

- Currently, there are many models using the Transformer architecture (we will explore this later)

Deep Network Development

Lecture 4



Transfer Learning

1 CNN Architectures

2 Transfer Learning

3 Autoencoders

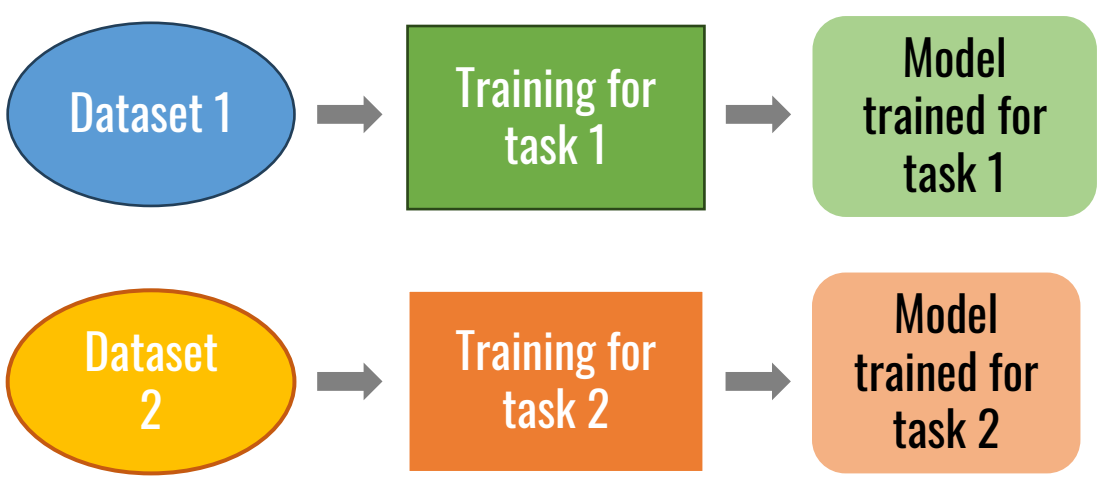
Traditional ML vs Transfer Learning

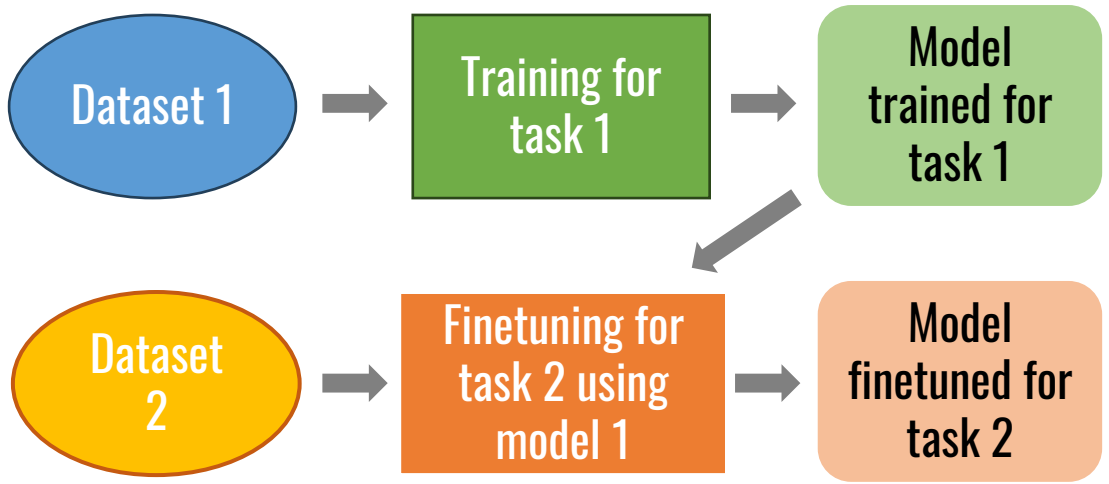
Traditional ML

- We train a model for a given task
- The training starts from initialized weights, no prior training happened

Transfer Learning

- Learning relies on a previously trained model
- Since the model is pretrained for the task specific fine tuning less data is required compared to training from zero





Learned / Transferrable features

- The characteristics are built up in a pyramidal way
- Initial edge filtering with different orientation, "color" and cut-off values
- Simpler shapes, forms
- Part components (nose, eye, wheel,...)
- Faces, cars

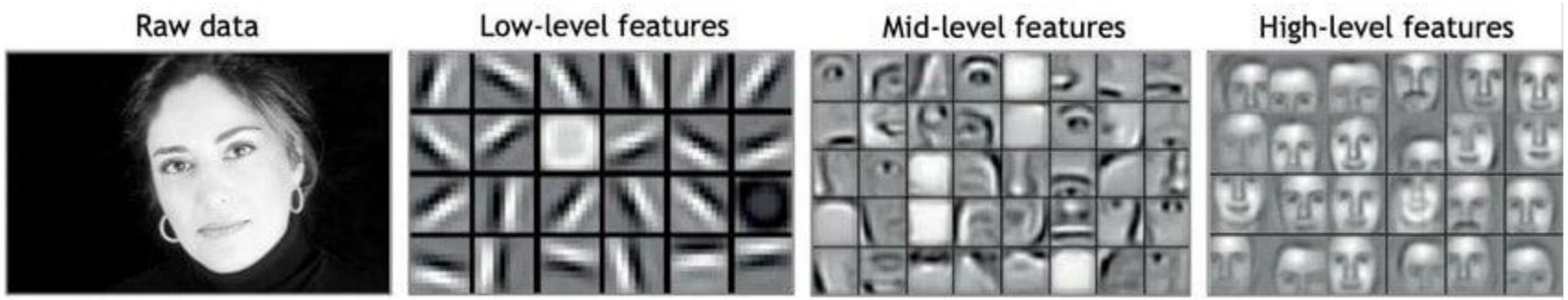


Image source: A Mechanistic Explanatory Strategy for XAI - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/sualization-of-features-on-various-layers-of-a-CNN-for-input-images-of-faces-reproduced_fig4_385528811 , available via license: Creative Commons Attribution 4.0 International

Learned / Transferrable features

- Most of the time the low level features can be reused and only the head has to be fine-tuned

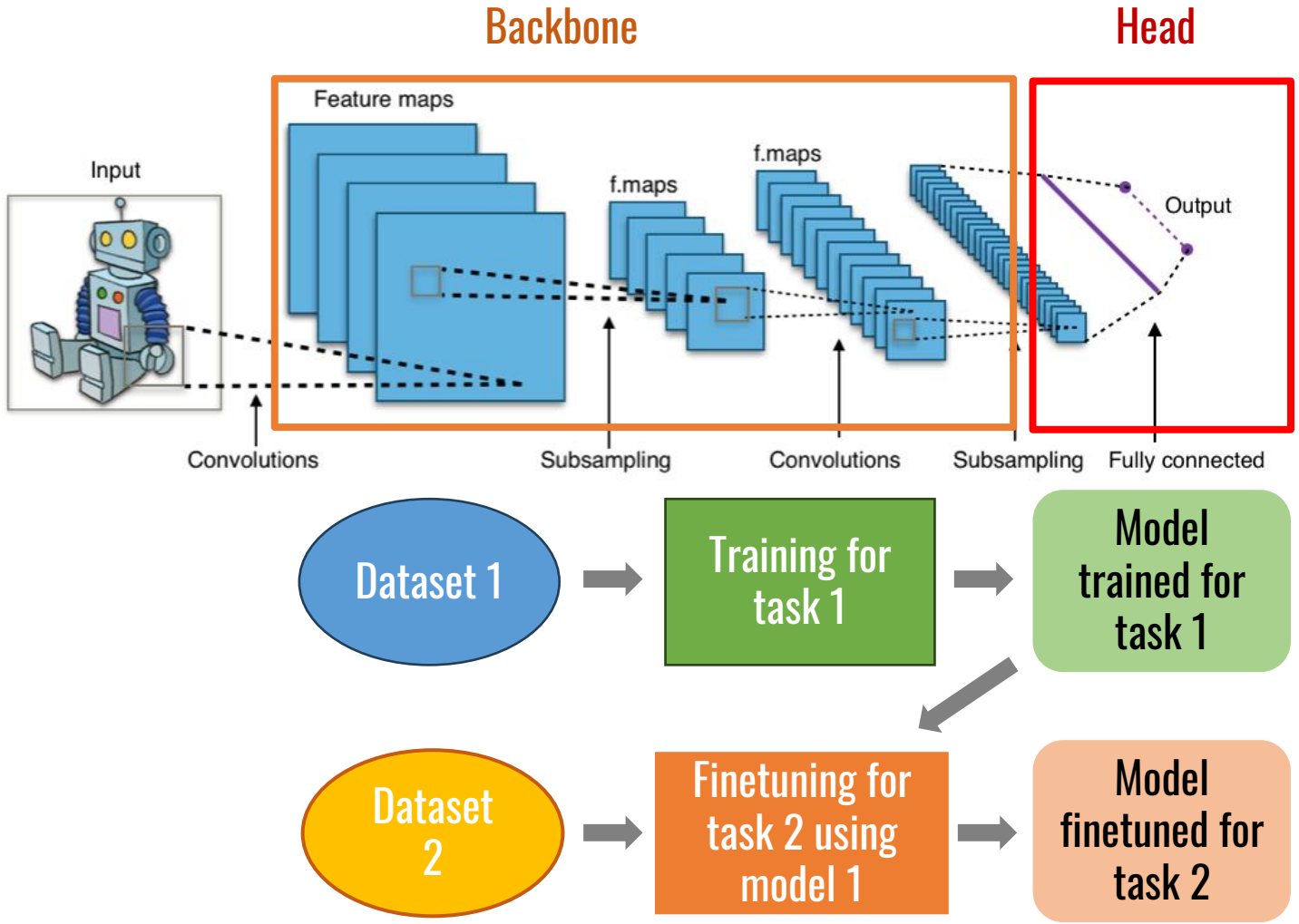


Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Feature extraction

- We initialize the network with pre-trained weights from a pre-trained model. We freeze all the weights from the backbone, and we replace the head with a new one, initialized with random weights. We only train the new head, and not the backbone.

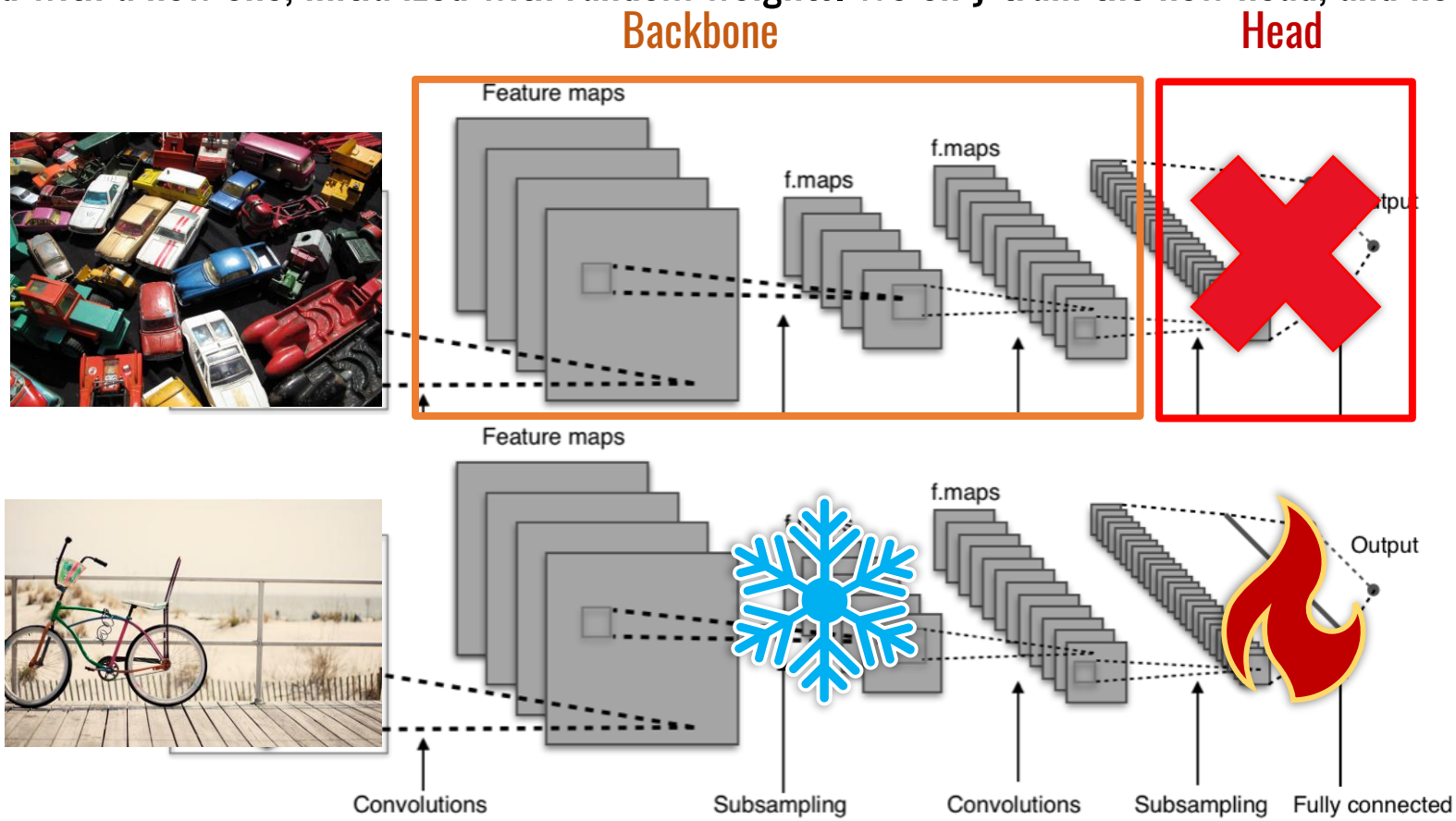


Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Fine-tuning

- Instead of randomly initializing the weights for training , we initialize the network with pretrained weights from a pretrained model trained on a bigger dataset such as ImageNet (1M images and 1K class) and then we can fine-tune (train) the whole network.

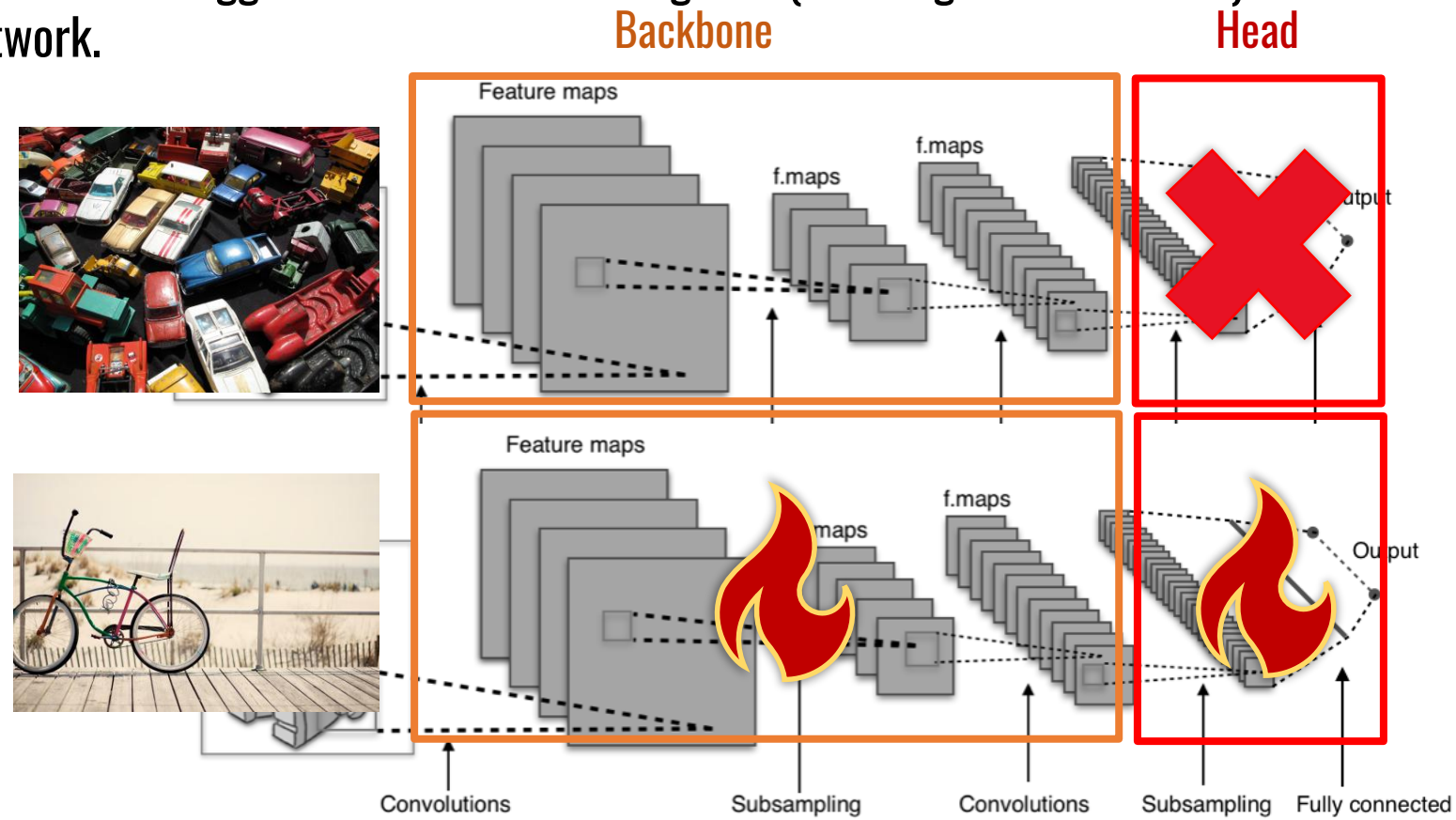
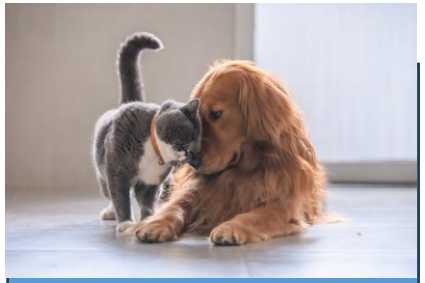


Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Performance comparison



Dataset A
(Cats vs Dogs)



Dataset B
(Cat breeds)



Dataset C
(Kangaroos)

Model A
(trained on dataset A)

Model B
(Using Model A for Feature Extraction)

Good performance;
Fast Training;
Less data needed.

Model D
(Using Model A for Feature Extraction)

Worse performance

Model C
(Fine-Tuned from Model A)

Good performance;
Slower Training;
More data needed.

Model E
(Fine-Tuned from Model A)

Good performance;
Slow Training;
More data needed.

When to use Transfer Learning?

- Task A and B have the same input x
- When you have a lot of data for the problem you are transferring from (A) and few data for the problem you are transferring to (B)
- Low level features from A could be helpful for learning B
- Faster training. Use pre-trained weights as initialization point whether than randomly initializing weights

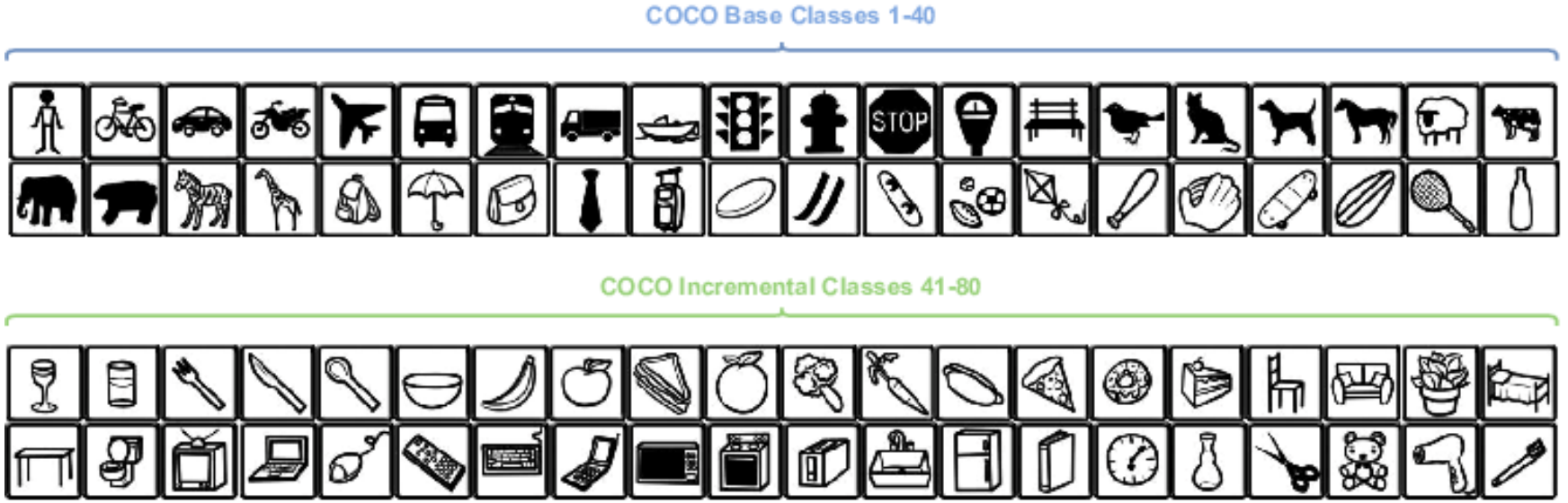
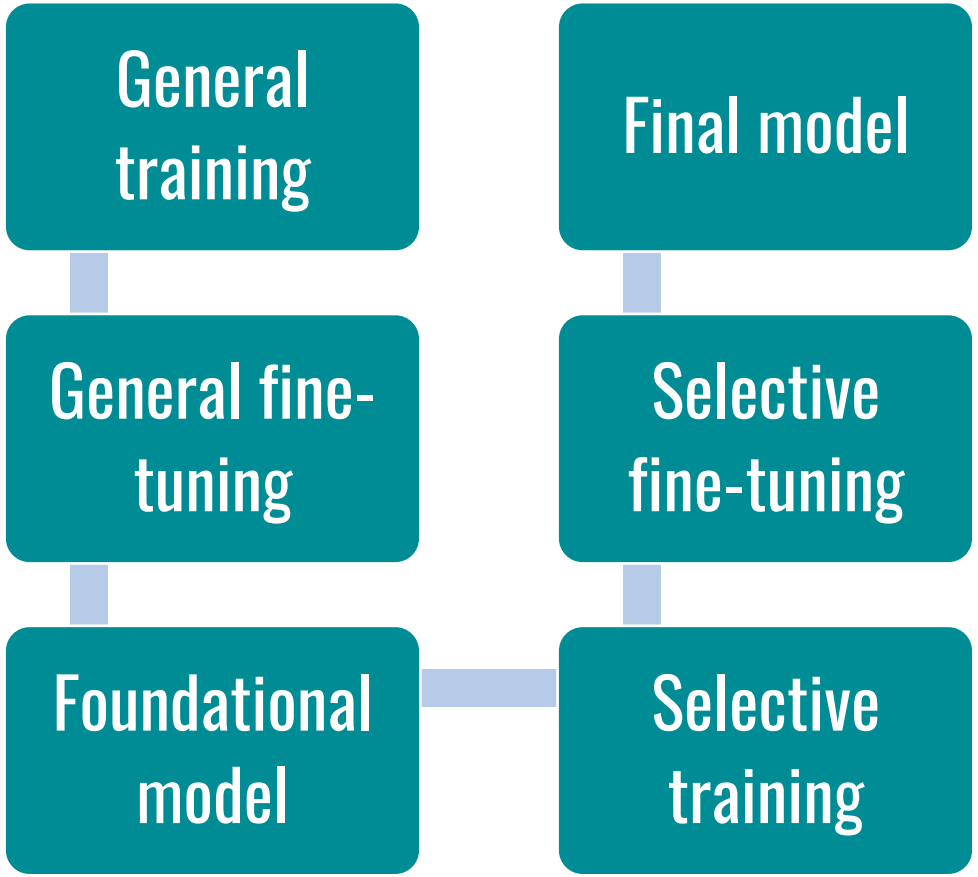


Image source: Continual Object Detection: A review of definitions, strategies, and challenges - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Description-of-an-incremental-scenario-with-MS-COCO-2014-dataset_fig7_360993612 , available via license: Creative Commons Attribution 4.0 International

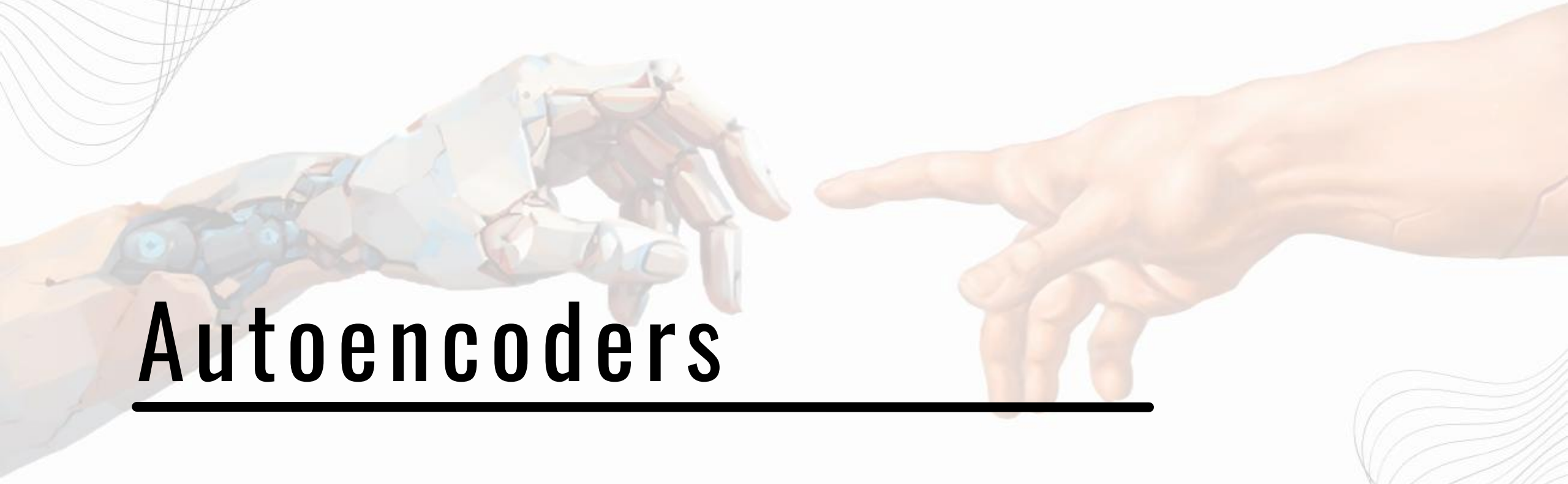
Deep Learning common steps - Parameter learning

- General
  - large dataset
  - Training
  - Fine-tuning
  - Foundational model



Deep Network Development

Lecture 4



Autoencoders

1 CNN Architectures

2 Transfer Learning

3 Autoencoders

Feature learning: Autoencoders

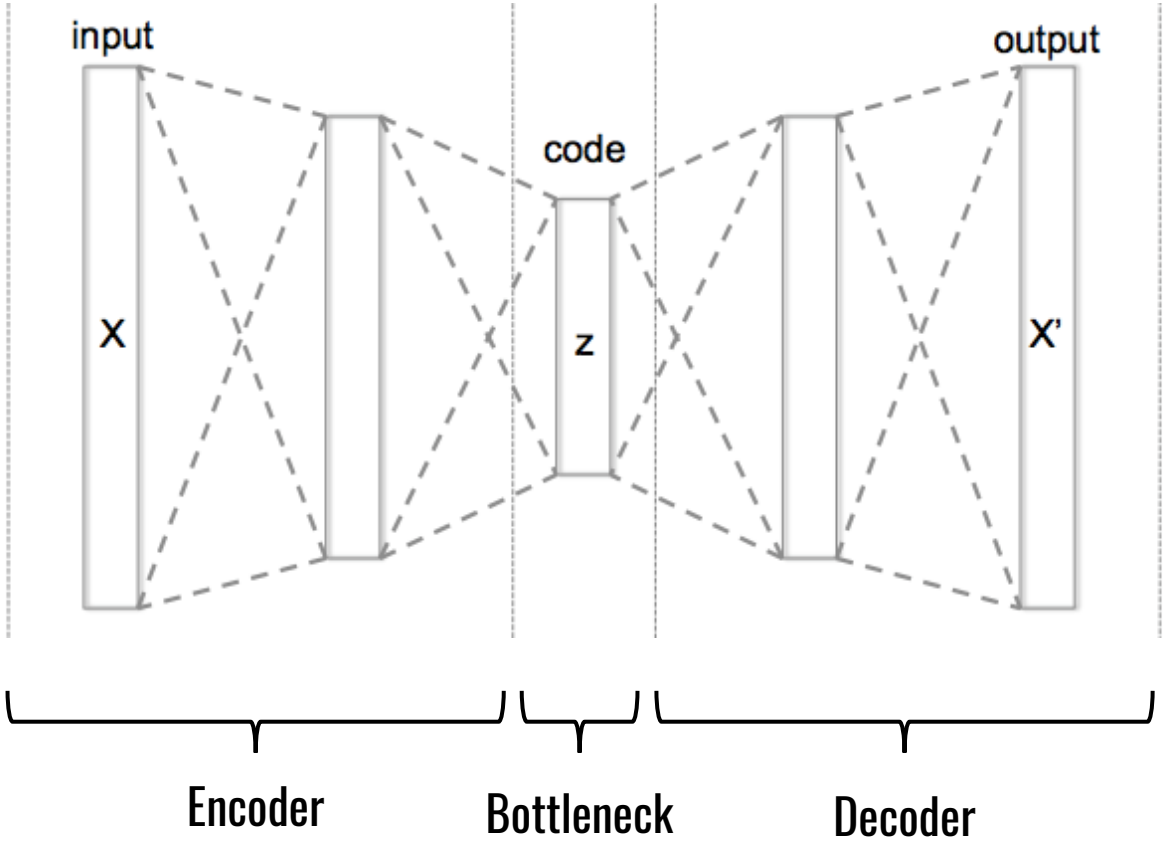


Image Source: Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Reconstruction Task

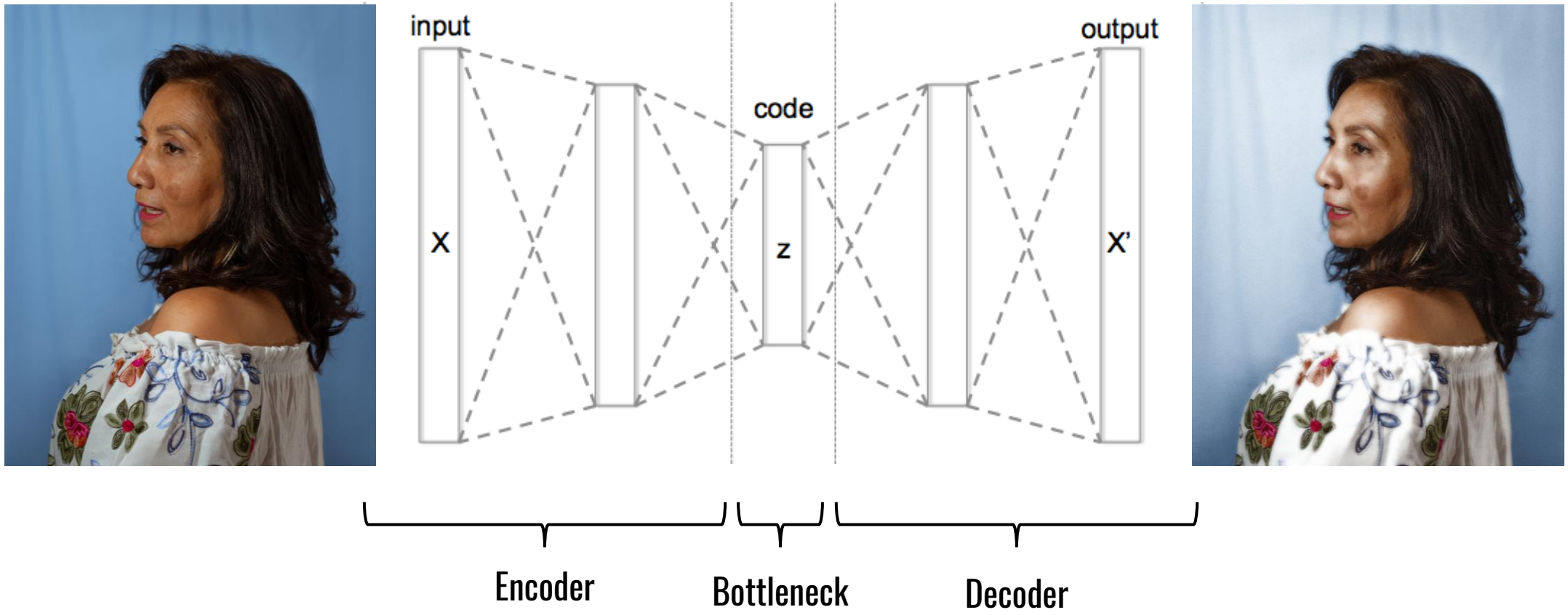
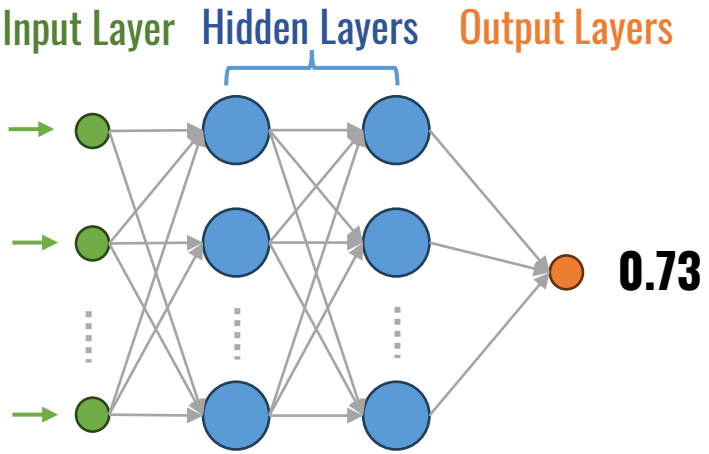


Image Source: Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Image Classification: Supervised Learning (labelled data is needed)

Data:
X = image
Y = label / target (1)





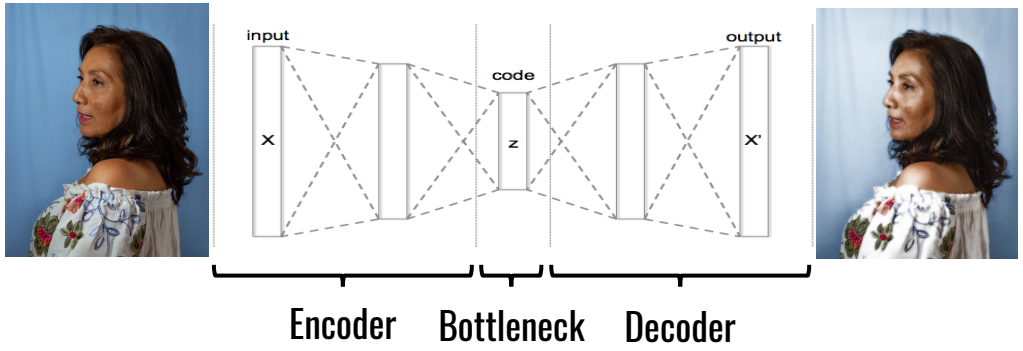
Loss between prediction and target

$$L_{BC}(h(x), y) = -y \log h(x) - (1-y)\log(1-h(x))$$
$$L_{BC}(0.73, 1) = -1 \log 0.73 - (1-1)\log(1-0.73)$$
$$L_{BC}(0.73, 1) = -1 * -0.13667714 - (0)\log(0.23)$$
$$L_{BC}(0.73, 1) = 0.13667714$$

$$\boldsymbol{L_{BC}(0.73, 1) = 0.14}$$

Reconstruction: Self-Supervised Learning (no need for labelled data)

Data:
X = image
Y = X



Loss between prediction and input



Image Source: Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Autoencoder structure

Autoencoder is a machine learning algorithm that **learns a compressed representation of an input**. Because it requires no label it is an unsupervised method[1]. However, the **output is a reconstruction of the input**, thus it is also considered as a self-supervised method.

An autoencoder consists of three parts:

- **Encoder**: the part of the network that compresses the input into a latent-space representation of reduced dimension. It can be represented by an encoding function h=f(x).
- **Latent space (bottleneck/code)**: the part of the network which contains the reduced representation of the input.
- **Decoder**: the part that aims to reconstruct the input from the latent space representation. It has a similar structure to the encoder and can be represented by a decoding function r=g(h).

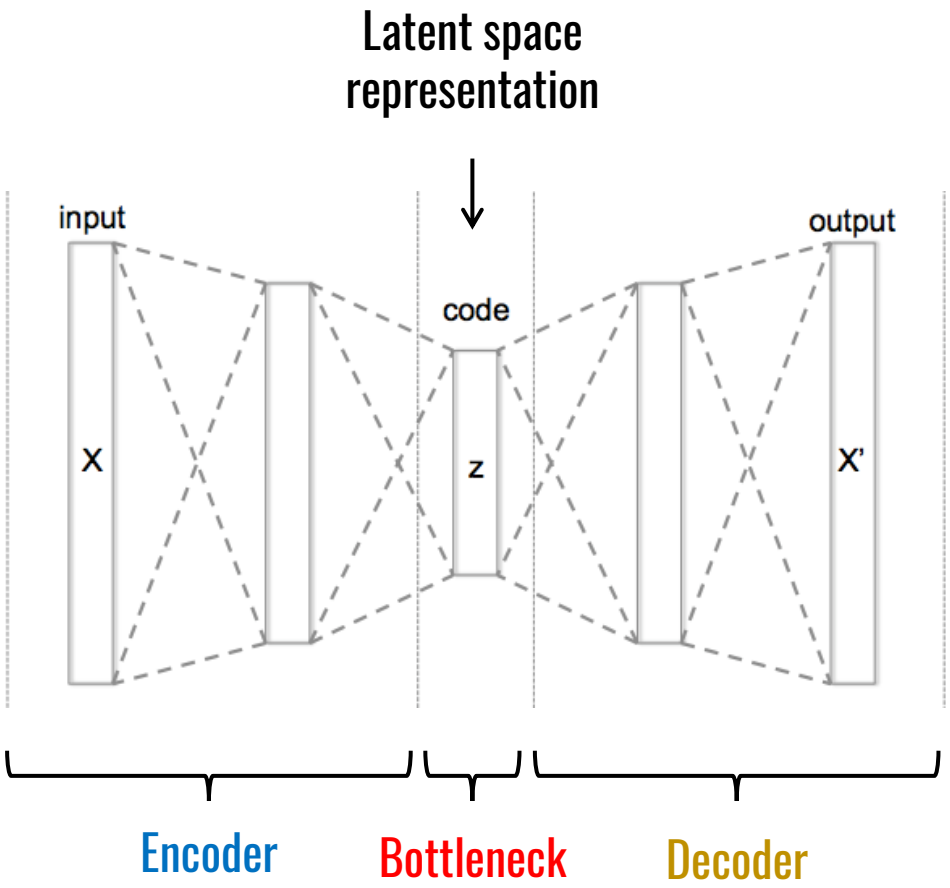


Image Source: Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Autoencoder structure

- Autoencoders are learned automatically from data examples, **using the input as output target**, making them easy to train. However, they do not generalize to new data.
- If the only purpose of autoencoders was to copy the input to the output, they would be useless. The main idea is that by training the autoencoder to copy the input to the output, **the latent representation will learn the most important features of the input**.
- The bottleneck is designed in such a way that the maximum information possessed by an input is captured in it, therefore, the bottleneck **forms a knowledge-representation of the input**.

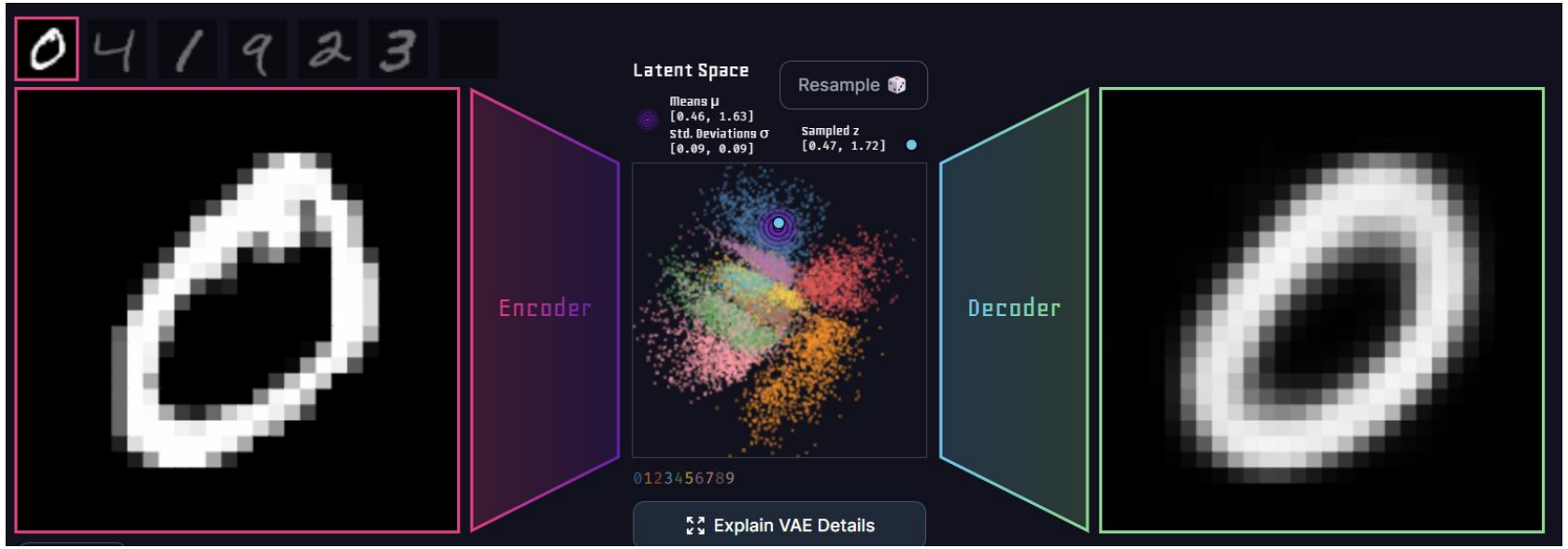
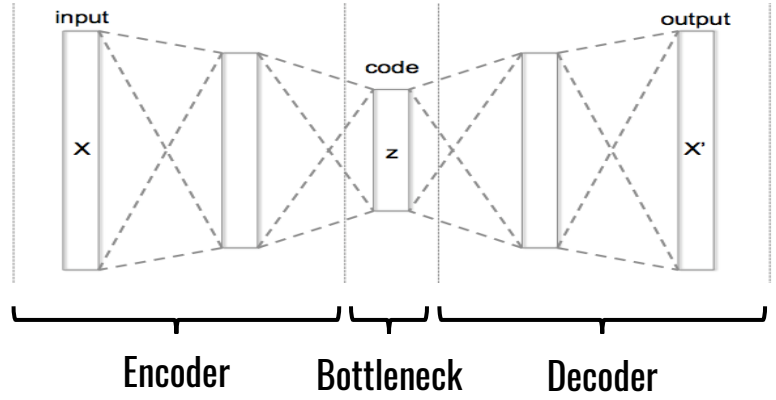


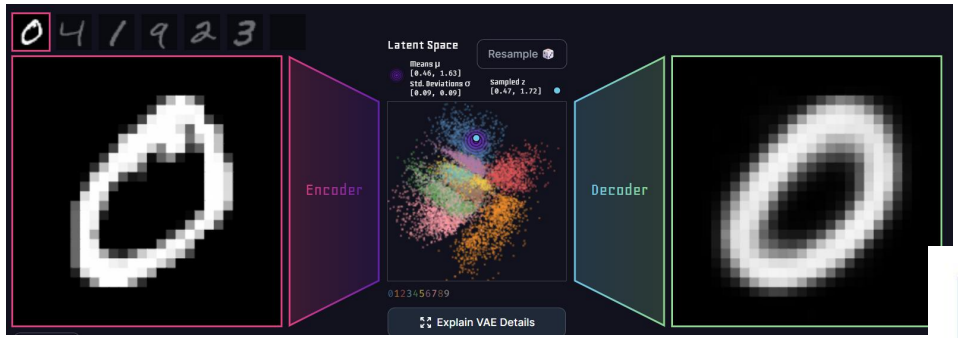
Image Source: Screenshot from **VAE Explainer** by Donald Bertucci and Alex Endert, based on the MIT-licensed xnought/vae-explainer repository. Source: https://github.com/xnought/vae-explainer.

Autoencoder structure

The input and output layers of the autoencoder can be formed with:

- Feed Forward neural networks (Dense / Fully connected layers)
- Convolutional neural networks
- Recurrent neural networks
- Others (Restricted Boltzmann Machine, …)





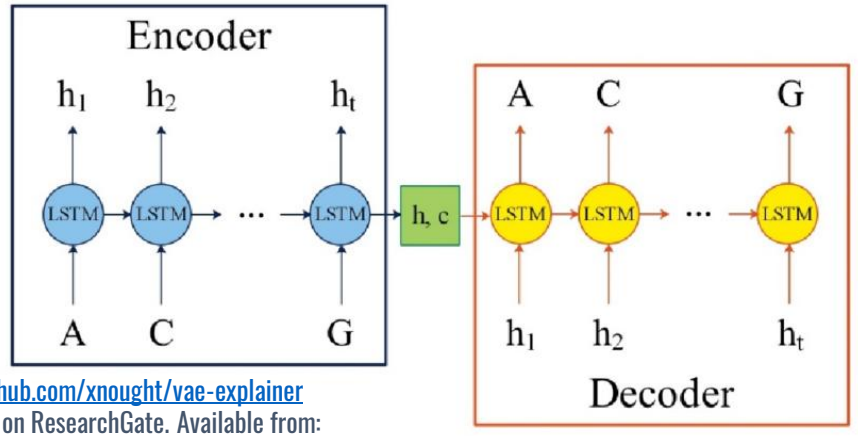


Image Source: Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Screenshot from **VAE Explainer** by Donald Bertucci and Alex Endert, based on the MIT-licensed xnought/vae-explainer repository. Source: https://github.com/xnought/vae-explainer
Image Source: SPREAD: An ensemble predictor based on DNA autoencoder framework for discriminating promoters in Pseudomonas aeruginosa</i - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/DNA-autoencoder_fig1_363552533 available via license: Creative Commons Attribution 4.0 International

Autoencoder Types

- Convolutional Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Variational Autoencoder
- and more

Convolutional Autoencoders

- Specifically designed for image data.
- They employ convolutional layers in both the **encoder** and **decoder** parts of the network.
- This architecture allows them to capture spatial dependencies and hierarchical features effectively.
- The reconstruction of the input image is often blurry and of lower quality due to compression during which information is lost.

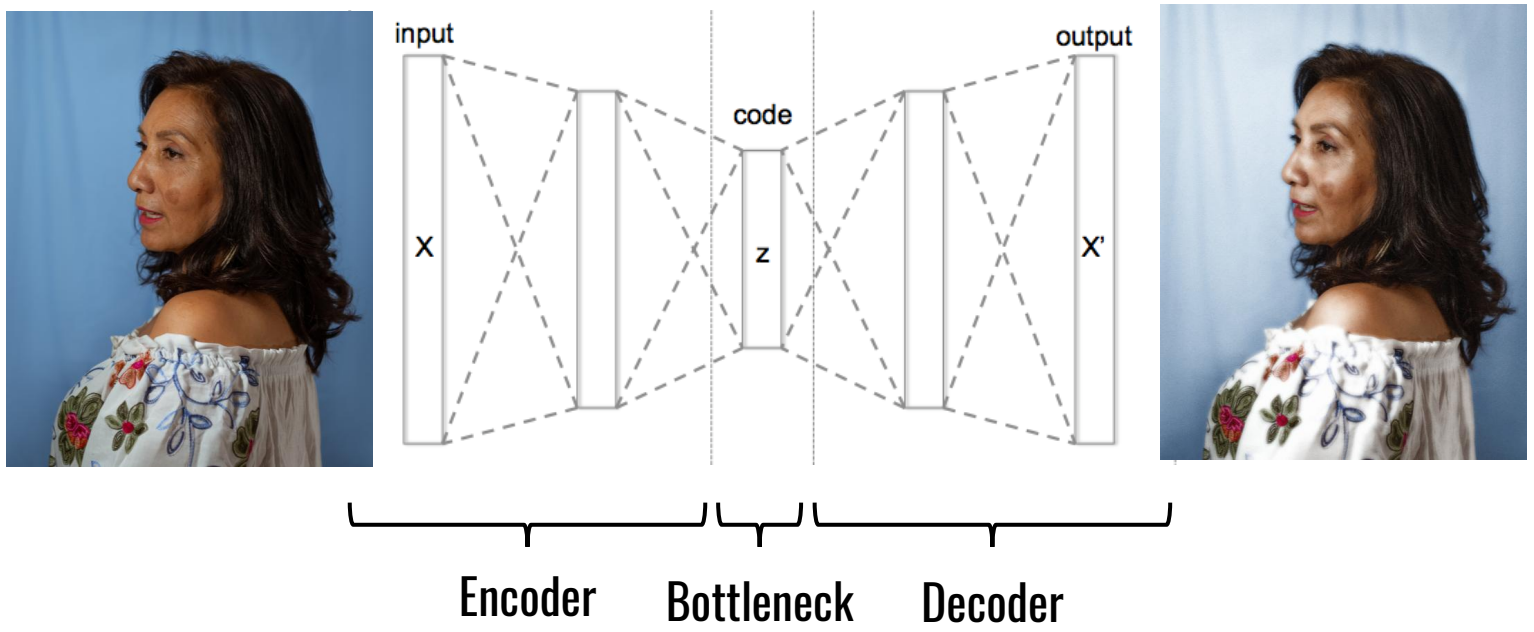


Image Source: Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Denoising Autoencoders

- Designed to remove noise from input data.
- Noise can be added to the input, by modifying different parts of the input:
  - Gaussian noise; Salt-and-Pepper noise; Random Masking; Perturbation noise; etc
- During training, a noisy version of the input is provided, and the network learns to reconstruct the clean, noise-free data.
- This encourages the model to capture robust and meaningful features while filtering out irrelevant or noisy information.

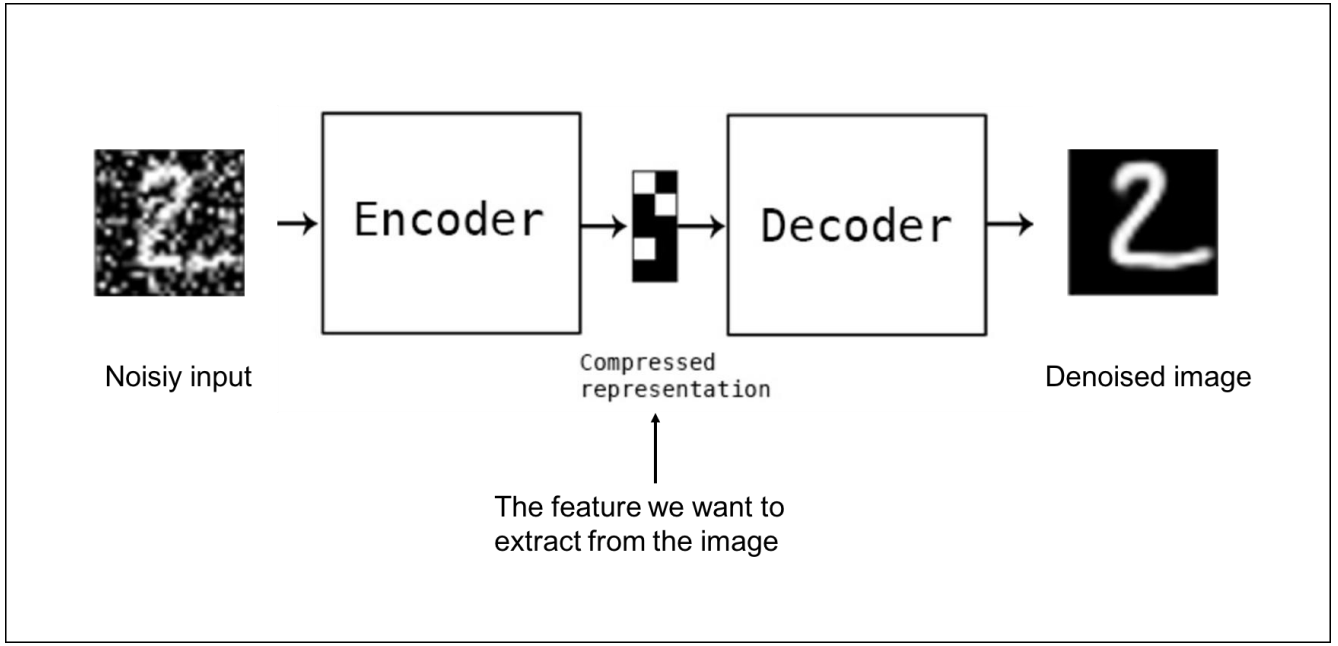


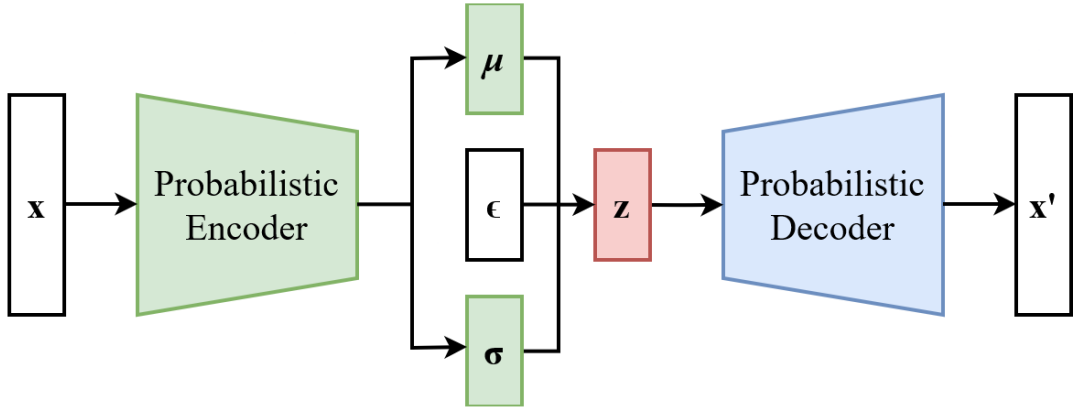
Image Source: Denoising-autoencoder.png, neerajkrbansal1996, CC0, via Wikimedia Commons

Sparse Autoencoders

- Designed to impose sparsity constraints on the representations learned by the autoencoder.
- Sparsity means that only a small subset of neurons in the hidden layer is active at a time
- They take the highest activation values in the hidden layer and zero out the rest of the hidden nodes. This prevents autoencoders to use all the hidden nodes at a time and forcing only a reduced number of hidden nodes to be used.
- Sparse autoencoders have a sparsity penalty, a value close to zero but not exactly zero. Sparsity penalty is applied on the hidden layer and to the reconstruction error. This prevents overfitting.

Variational Autoencoders

- Variational Autoencoders (VAE) are generative autoencoders that **can generate new instances**
- Different from traditional autoencoders, instead of learning a mapping function for each input to a number in the latent space, they **learn a probability distribution representation of the data**.
- By sampling from the learned probability distribution, a point within it is selected, which represents a potential encoding of the input. This randomness during sampling is what makes VAEs generative, enabling to generate diverse outputs consistent with the input's learned distribution.
- https://xnought.github.io/vae-explainer/



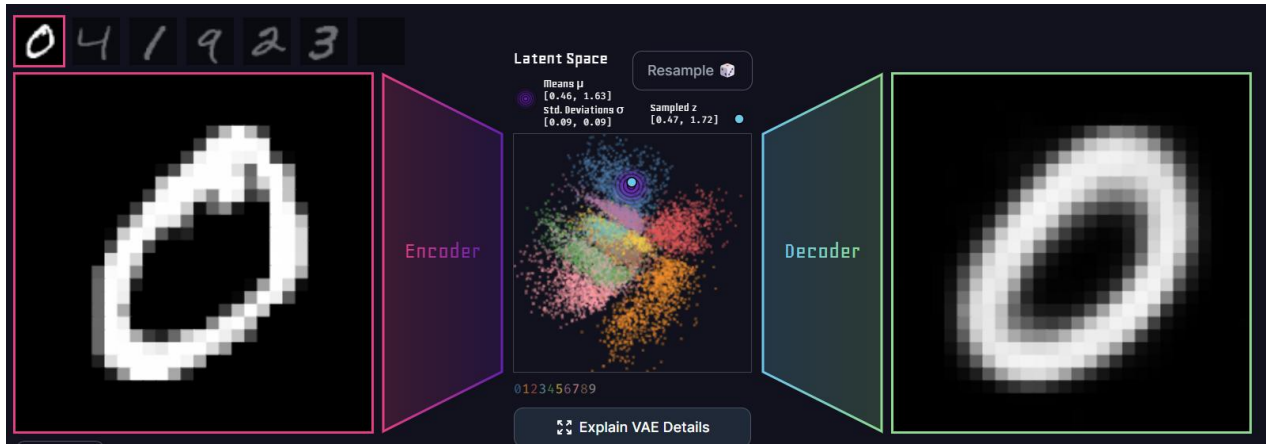


Image Source: Reparameterized Variational Autoencoder.png, EugenioTL, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Screenshot from **VAE Explainer** by Donald Bertucci and Alex Endert, based on the MIT-licensed xnought/vae-explainer repository. Source: https://github.com/xnought/vae-explainer.

Applications

(Not for) Data Compression

- Although autoencoders can compress the input into a latent representation, it is **usually not used for data compression**. The reasons are:
  - **Lossy compression**: The output of the autoencoder is not the same as the input, it is a close but degraded representation because information is lost in the latent representation.
  - **Data-specific**: Autoencoders are only able to meaningfully compress data like what they have been trained on. Since they learn features specific for the given training data. Hence, we can't expect an autoencoder trained on handwritten digits to compress landscape photos.

Applications

Dimensionality Reduction

- The autoencoders convert the input into a **reduced representation which is stored in the latent representation space**. This is where the information from the input has been compressed. Thus by **removing the decoder part**, an autoencoder can be used for dimensionality reduction with **the output being the latent representation vector**.

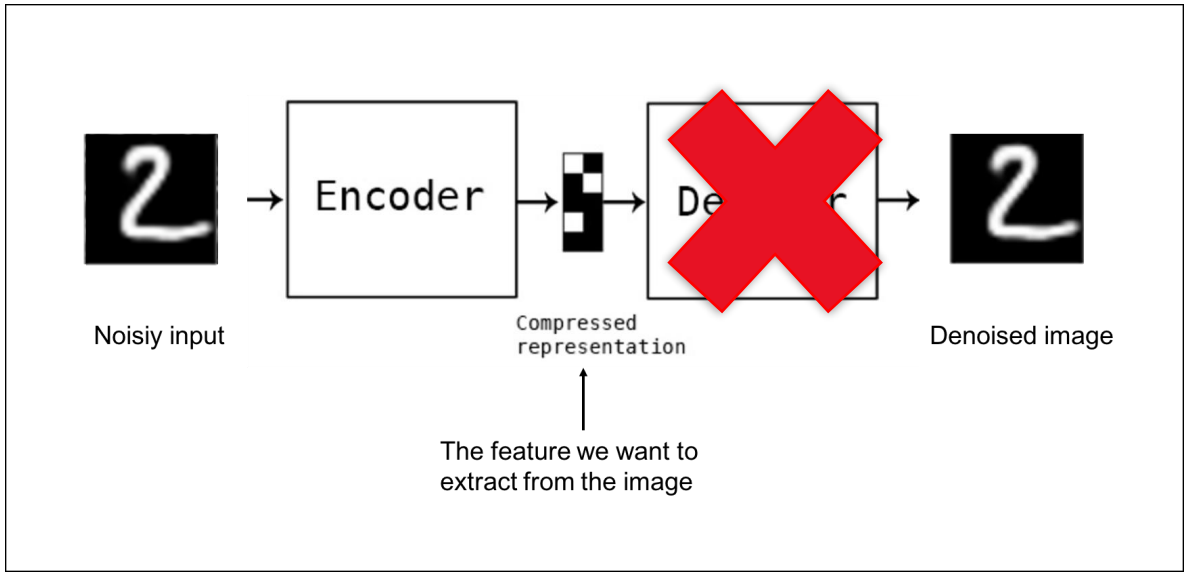


Image Source: Denoising-autoencoder.png, neerajkrbansal1996, CC0, via Wikimedia Commons

Applications

Image Denoising

- Example:
  - https://huggingface.co/spaces/Xintao/GFPGAN
  - https://huggingface.co/spaces/aryadytm/photo-colorization

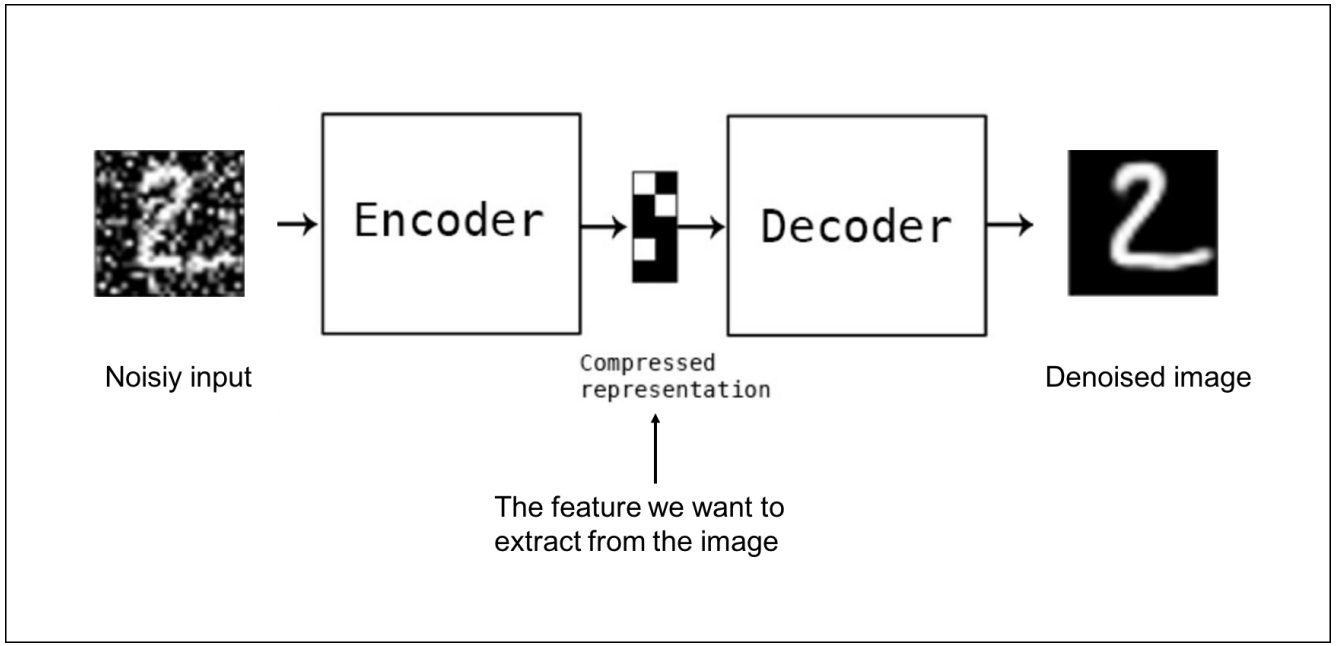


Image Source: Denoising-autoencoder.png, neerajkrbansal1996, CC0, via Wikimedia Commons

Applications

Image Segmentation

- Image segmentation is the process of partitioning an image into multiple segments each belonging to a class. The goal is to simplify and/or change the representation of an image by **grouping pixel values according to the class they belong**.



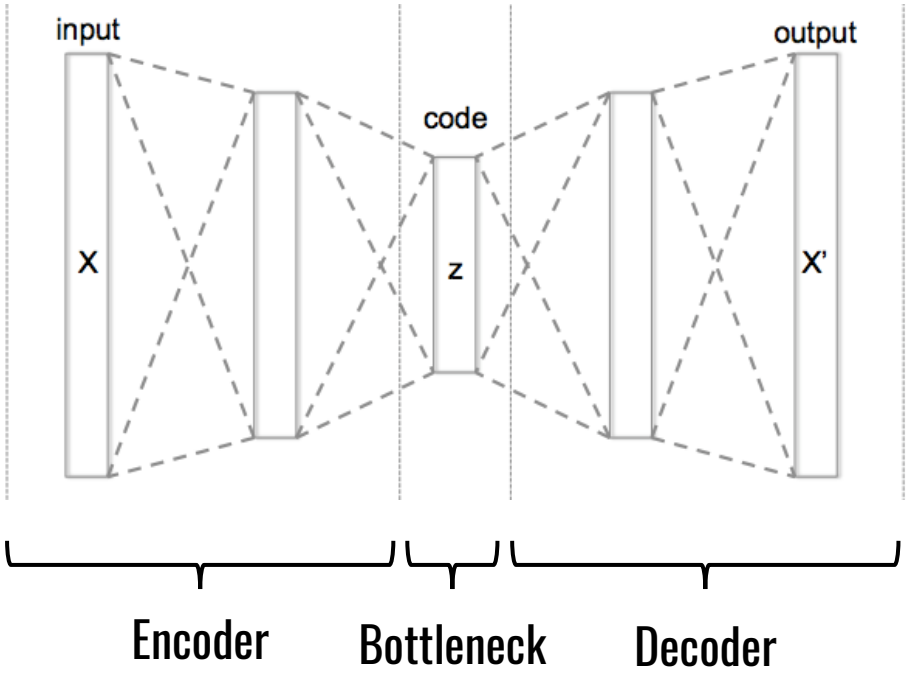




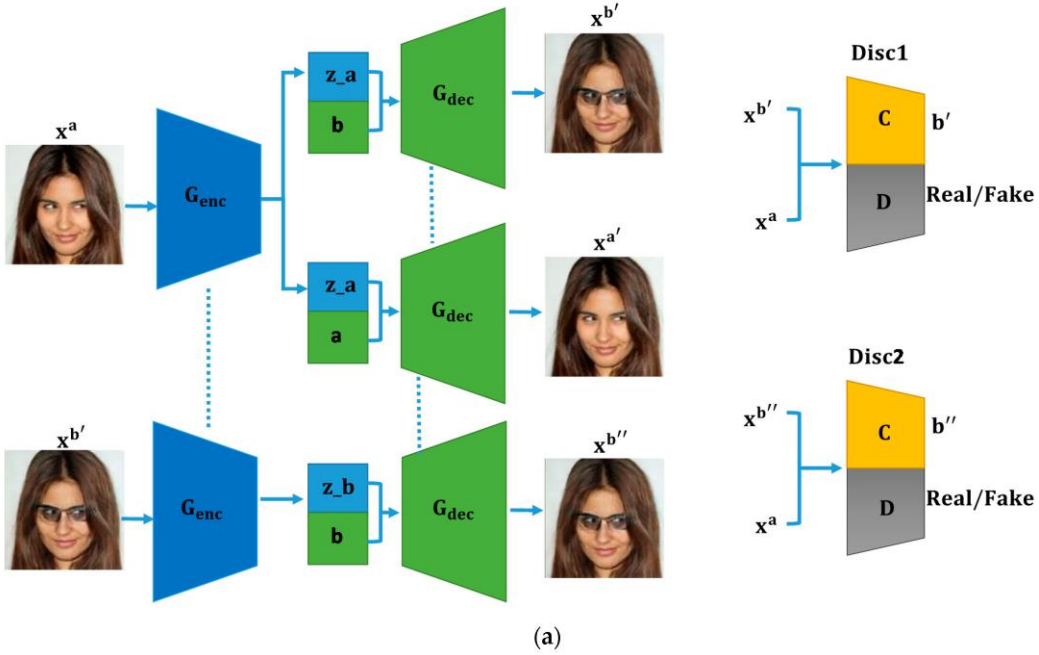
Image source: Oso Flaco Lake Raccoons. - Flickr - docentjoyce.jpg , Joyce Cory, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

Applications

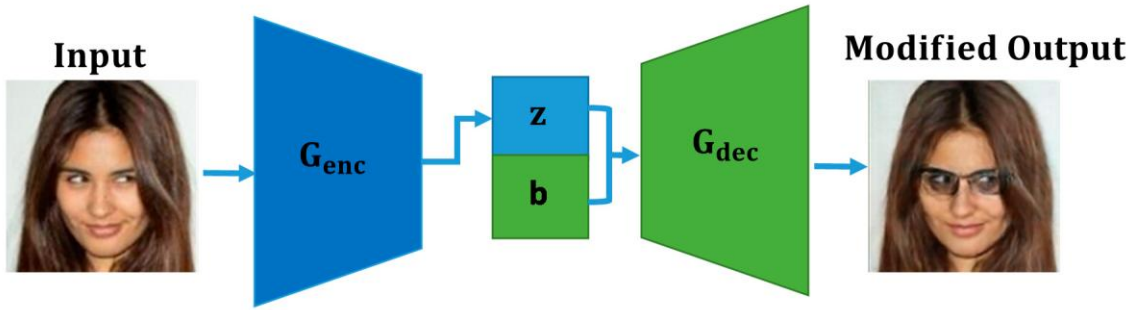
Image Generation

Encoder-Decoder swap

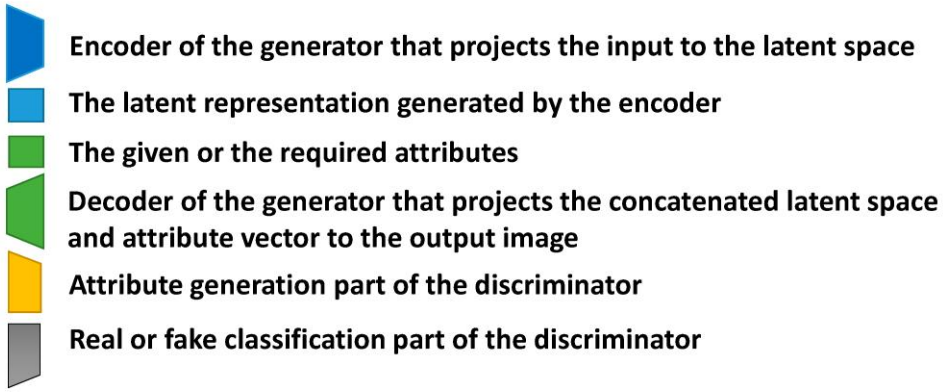
Latent Space Manipulation



(a)



(b)



(c)

Image Source: Fig. 1, Fine-Tuned Attribute-Modification Network — Islam, N. U. & Park, J., *Face Attribute Modification Using Fine-Tuned Attribute-Modification Network* (2020), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://doi.org/10.3390/electronics9050743

Applications

Feature Extraction

- The Encoding part of Autoencoders helps to **learn important hidden features** present in the input data, in the process to reduce the reconstruction error. During encoding, a new set of combinations of original features is generated. **By removing the decoder, we can use the encoded features as input features for a network to use**, for example, in the classification task.

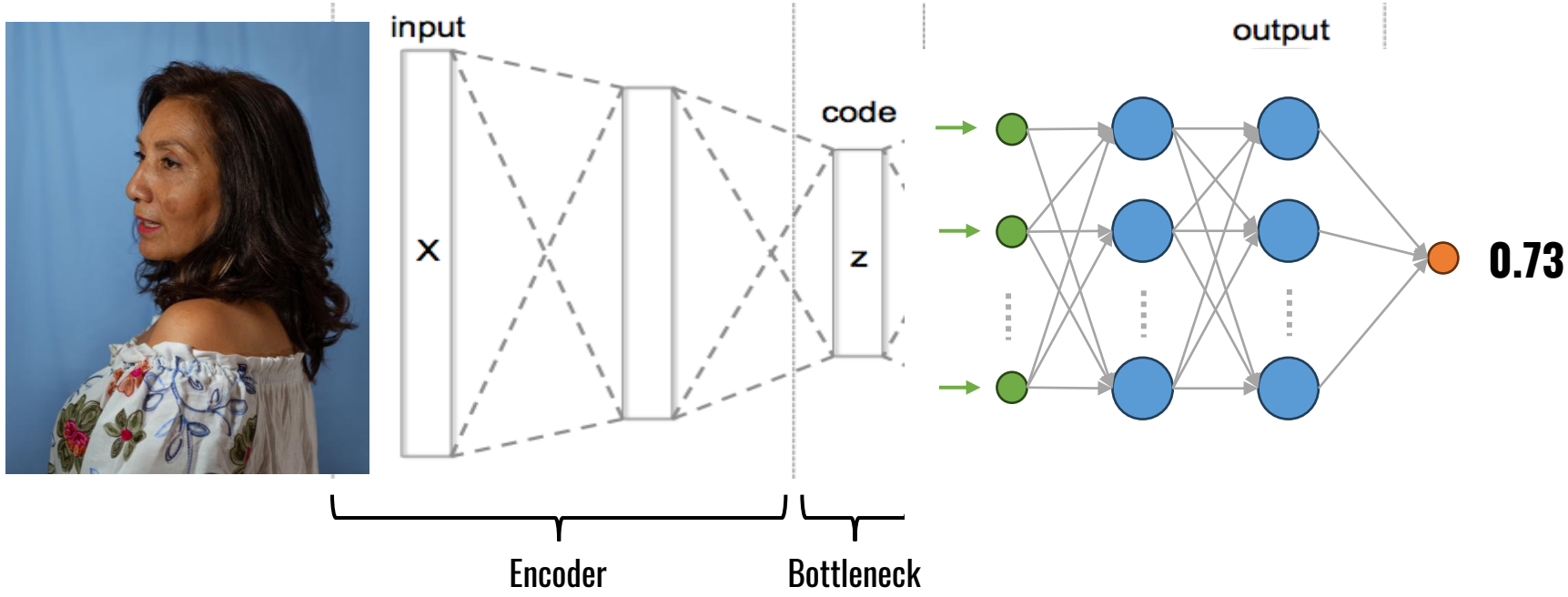


Image Source: Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Summary

- Different CNN hyperparameter choices result in different CNN architectures

- Transfer Learning is used when we want to utilize the learned features from a model A, that can be useful for a task that model B is trying to solve.
- We can retrain the whole model or freeze part of the model and only retrain a small part

- Autoencoders are unsupervised / self-supervised methods
- By reconstructing the input, they learn to encode the input into a lower dimensional latent space
- By removing the decoder from a trained autoencoder, the encoder can be used for feature learning

Resources

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning

Papers

1. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/cvpr.2016.90
2. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1704.04861
3. Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017, July). Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
4. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
5. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.
6. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1409.1556
7. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9. https://doi.org/10.1109/cvpr.2015.7298594

Image Sources

- Unless otherwise noted, the figures were created by the instructor
- Python code examples were created by the instructor and rendered with Carbon.
- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.

- Dancing Robot: Gif robot.git, Meks01, CC BY-SA 4.0, via Wikimedia Commons

Image Sources

- Typical cnn.png, Aphex34, CC BY-SA 4.0 ,https://creativecommons.org/licenses/by-sa/4.0/ , via Wikimedia Commons
- Convolution arithmetic visualization, adapted from Vincent Dumoulin, & Francesco Visin. (2018). A guide to convolution arithmetic for deep learning.; original visual assets from the conv_arithmetic GitHub repository, MIT License.
- VGG16.png, Gorlapraveen, CC BY-SA 4.0, via Wikimedia Commons
- ResNet block.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0, via Wikimedia Commons
- ResNetTypes.png, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, CC BY-SA 4.0 < https://creativecommons.org/licenses/by-sa/4.0/> , via Wikimedia Commons
- Autoencoder structure.png, Chervinskii, CC BY-SA 4.0 < https://creativecommons.org/licenses/by-sa/4.0/> , via Wikimedia Commons
- Denoising-autoencoder.png, neerajkrbansal1996, CC0, via Wikimedia Commons
- Reparameterized Variational Autoencoder.png, EugenioTL, CC BY-SA 4.0 < https://creativecommons.org/licenses/by-sa/4.0/> , via Wikimedia Commons

Image Sources

- Multi-spectral remote sensing images feature coverage classification based on improved convolutional neural network, Scientific Figure on ResearchGate, available via Creative Commons Attribution 4.0 International <https://creativecommons.org/licenses/by/4.0/>
- A Mechanistic Explanatory Strategy for XAI, Scientific Figure on ResearchGate, available via Creative Commons Attribution 4.0 International < https://creativecommons.org/licenses/by/4.0/>
- Continual Object Detection: A review of definitions, strategies, and challenges, Scientific Figure on ResearchGate, available via Creative Commons Attribution 4.0 International < https://creativecommons.org/licenses/by/4.0/>
- SPREAD: An ensemble predictor based on DNA autoencoder framework for discriminating promoters in Pseudomonas aeruginosa, Scientific Figure on ResearchGate, available via Creative Commons Attribution 4.0 International <https://creativecommons.org/licenses/by/4.0/>
- Screenshot from VAE Explainer by Donald Bertucci and Alex Endert, based on the MIT-licensed xnought/vae-explainer repository. Source: https://github.com/xnought/vae-explainer
- Fig. 1, Fine-Tuned Attribute-Modification Network — Islam, N. U. & Park, J., Face Attribute Modification Using Fine-Tuned Attribute-Modification Network (2020), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>

Image Sources

- Oso Flaco Lake Raccoons – Flickr – docentjoyce.jpg, Joyce Cory, CC BY 2.0 < https://creativecommons.org/licenses/by/2.0/> , via Wikimedia Commons
- A Convolutional Neural Network for Airport Security Inspection of Dangerous Goods, Scientific Figure on ResearchGate, available via Creative Commons Attribution 3.0 Unported < https://creativecommons.org/licenses/by/3.0/>

Legal Notice

conv_arithmetic — Vincent Dumoulin & Francesco Visin
Source: https://github.com/vdumoulin/conv_arithmetic
License: The MIT License (MIT)

Copyright (c) 2016

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Legal Notice

vae-explainer — Donny Bertucci
Source: https://github.com/xnought/vae-explainer
License: MIT License

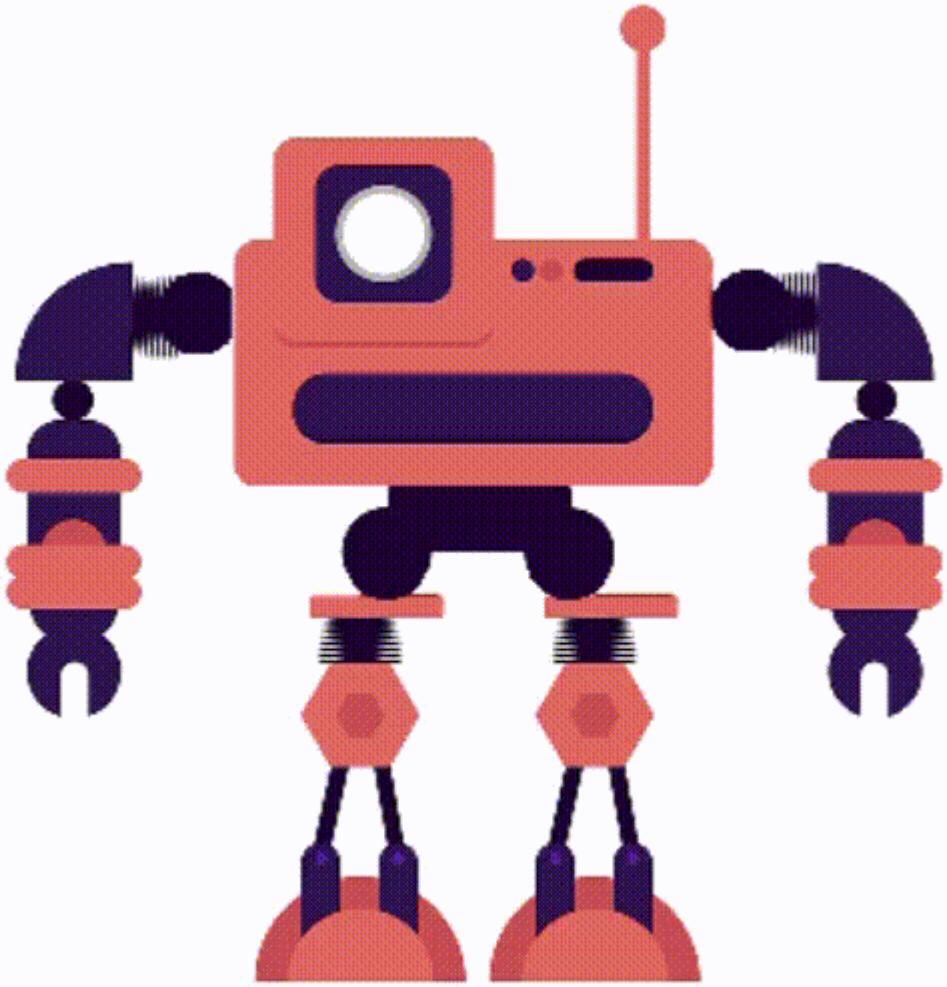
Copyright (c) 2024 Donny Bertucci

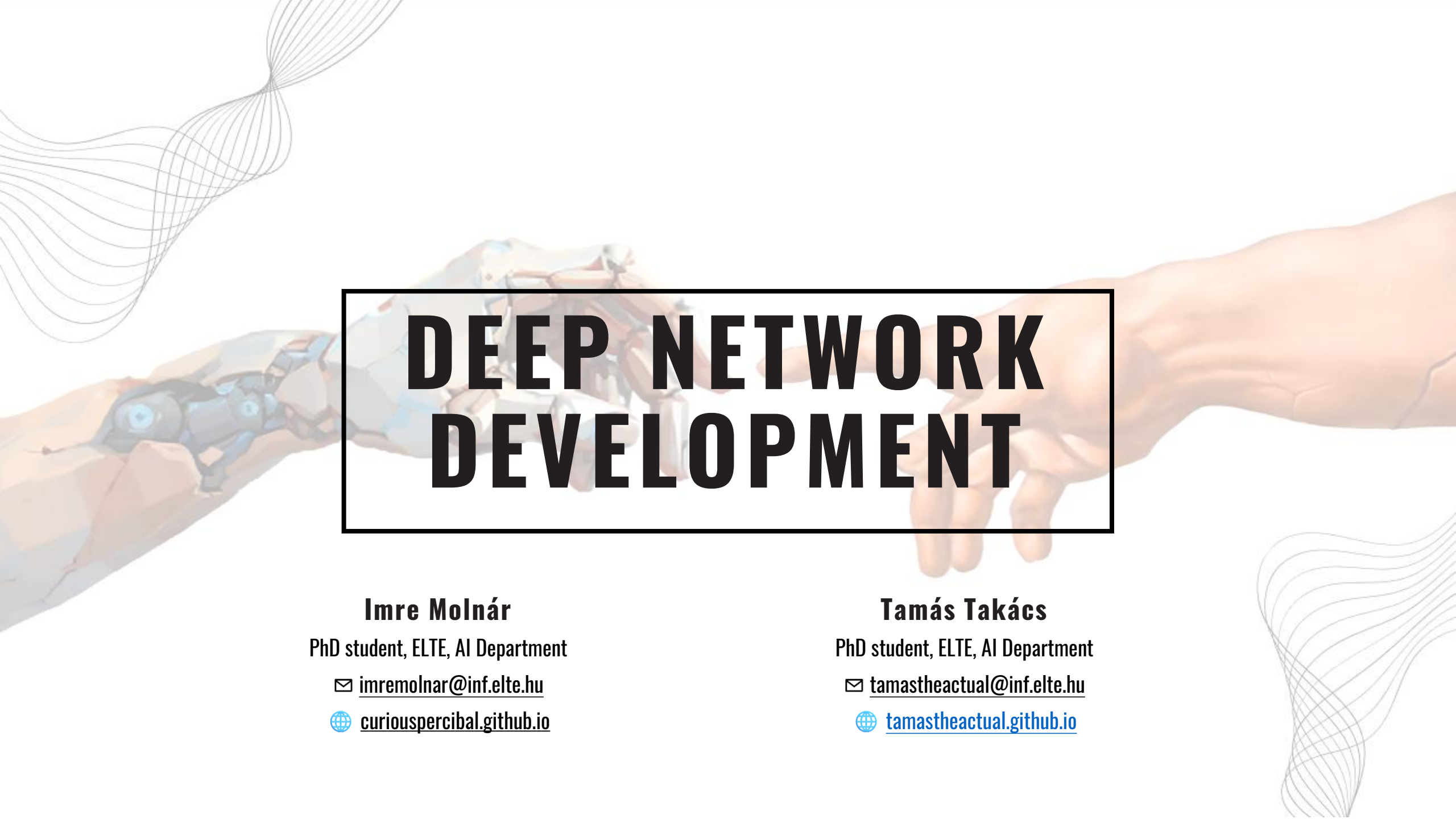
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

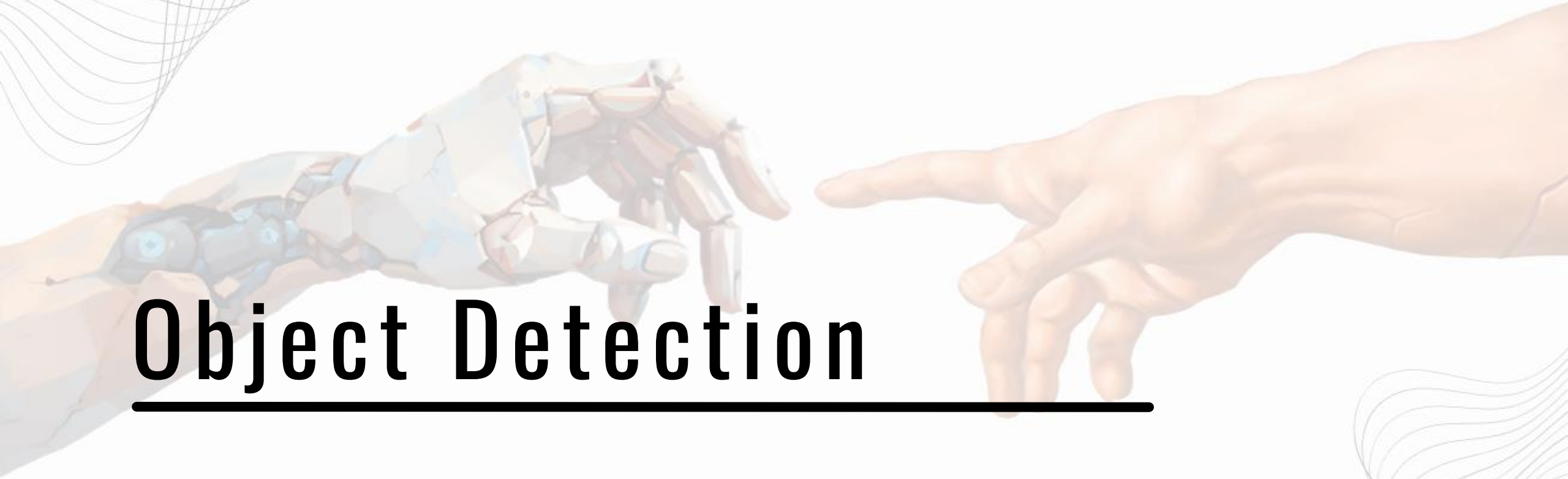
✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

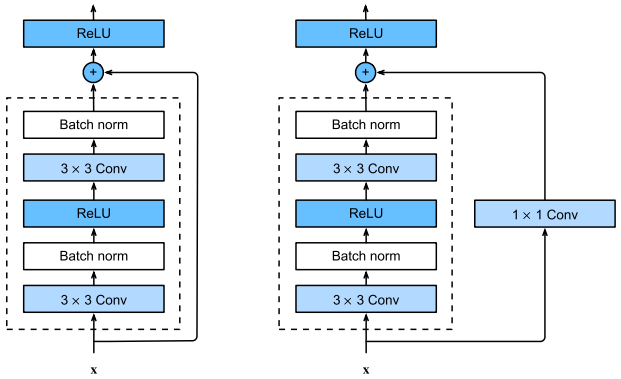
Lecture 5

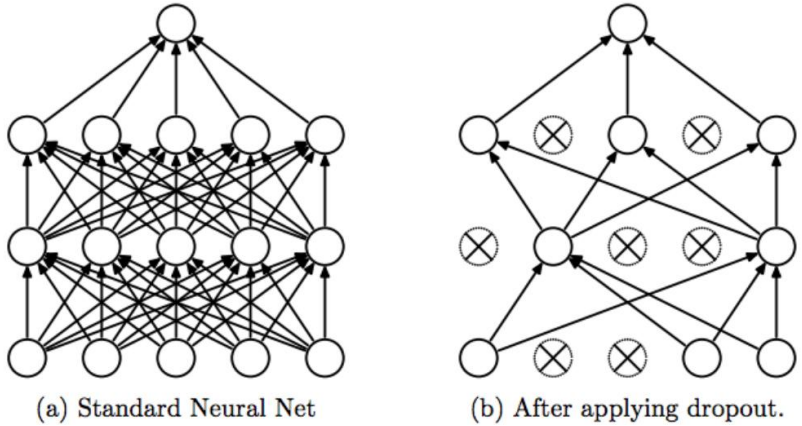


Object Detection

1. Two Stage Detectors
2. One Stage Detectors
3. Object Detection Metrics

CNN Architectures





When to use Transfer Learning

- Task A and B have the same input x
- When you have a lot of data for the problem you are transferring from (A) and few data for the problem you are transferring to (B)
- Low level features from A could be helpful for learning B
- Faster training. Use pre-trained weights as initialization point whether than randomly initializing weights

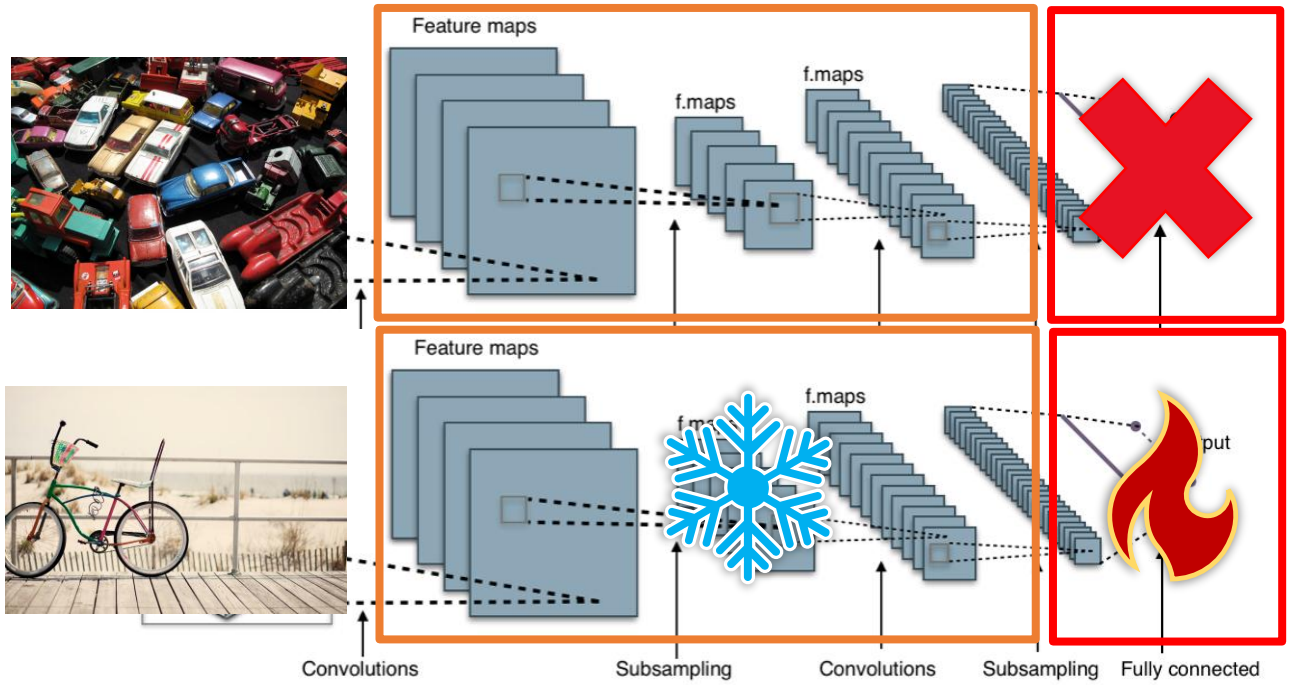


Image Source: "Dropout work diagram," Gao, Q., Li, Z. & Pan, J., A Convolutional Neural Network for Airport Security Inspection of Dangerous Goods, IOP Conf. Ser.: Earth Environ. Sci. 252 (2019) 042042, CC BY 3.0 https://creativecommons.org/licenses/by/3.0/, DOI: 10.1088/1755-1315/252/4/042042
Image source: ResNet block.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image source: ResNet block.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image source: Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Supervised Learning tasks

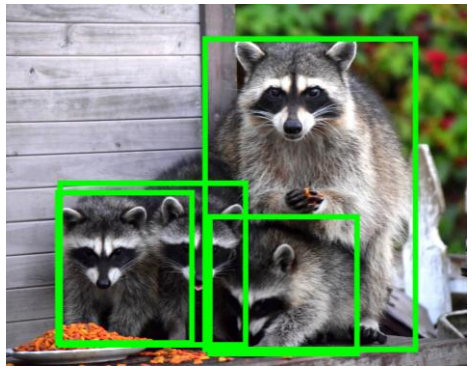
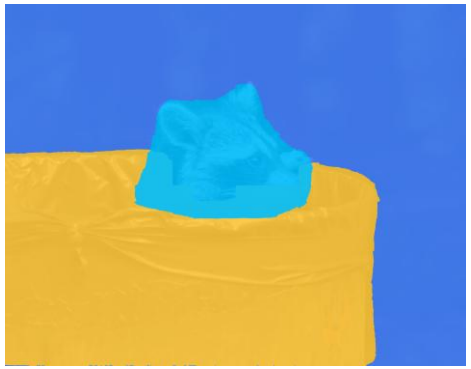
Classification	Classification + Localization	Object Detection	Instance Segmentation	Semantic Segmentation
				
RACCOON	RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, BIN, BACKGROUND
Single Object	Single Object	Multiple Objects	Multiple Objects	No objects, just pixels

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

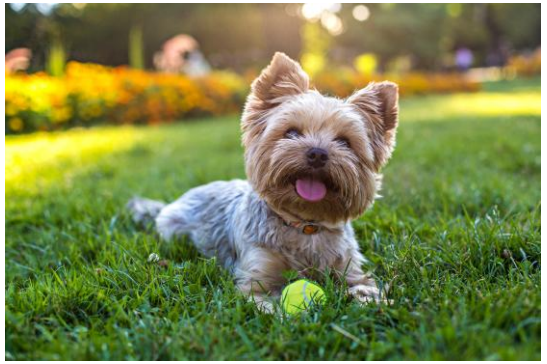
What is Object Detection?

- Sometimes the question is not whether there is a dog or a cat on the image, rather where is the dog or the cat.
- How many cats are there on the image?

Image Classification
[cat, dog, car]

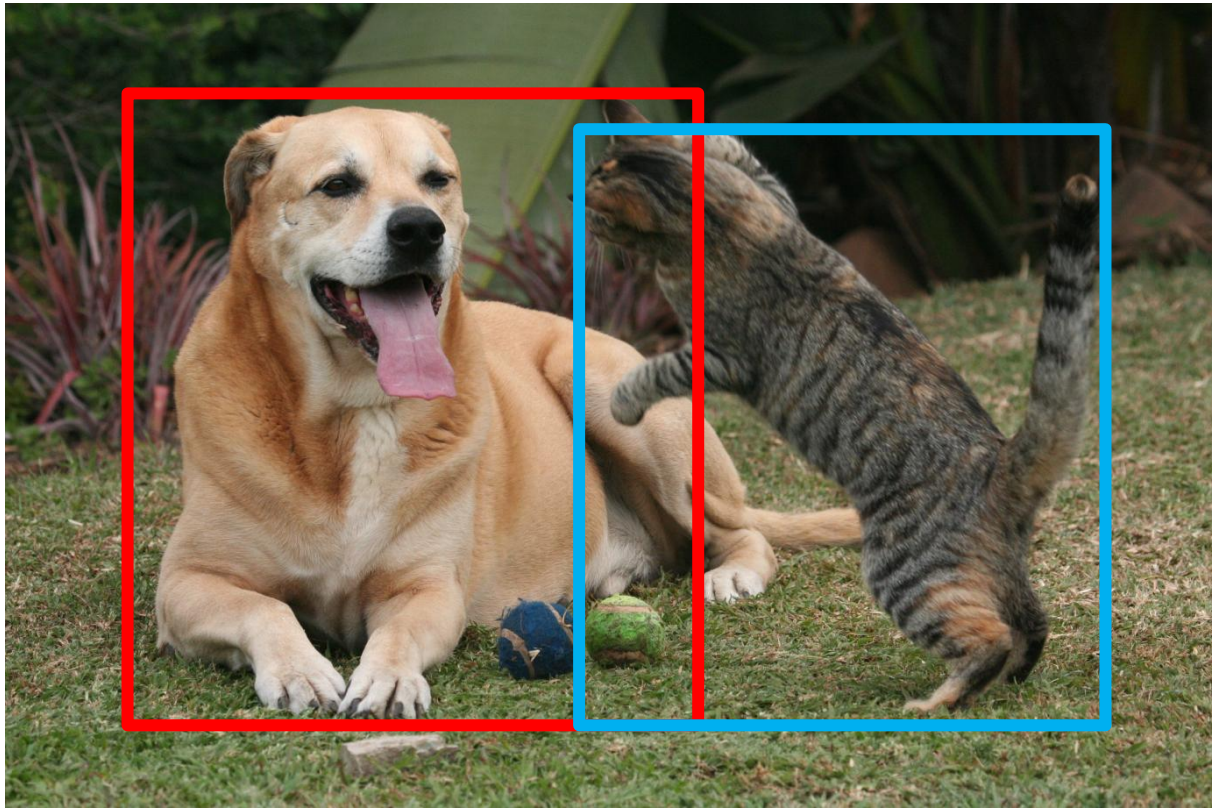


[0.9, 0.05, 0.05]



[0.7, 0.21, 0.09]

WHERE?



[0.48, 0.51, 0.01]

Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

What is Object Detection?

- **Supervised Learning Task**
- **Input**: RGB image
- **Output**: A set of detected objects;

For each object predict:
- **Category label** (from fixed, known set of categories)
- **Bounding box** (four numbers: x, y, width, height)

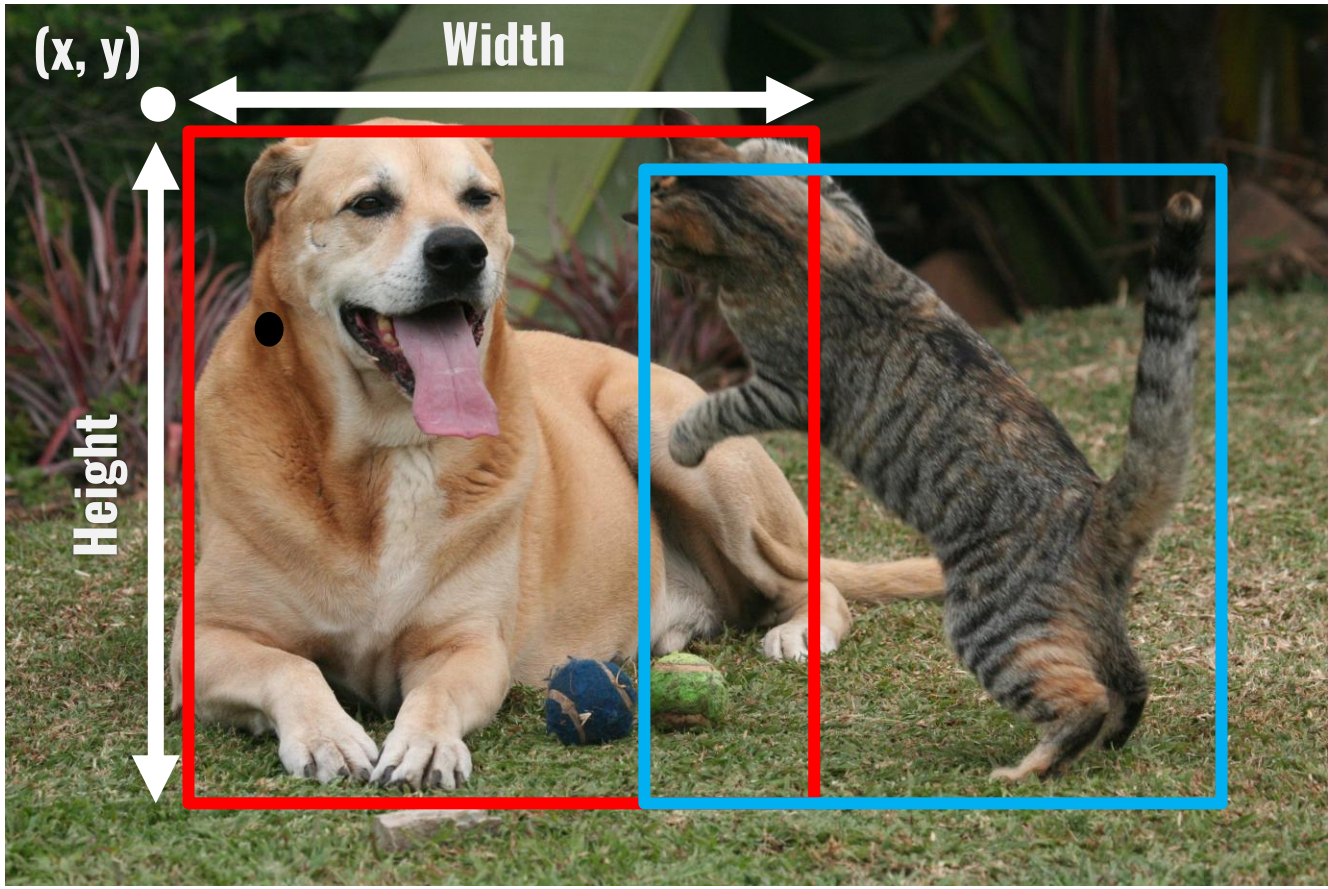


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

What is Object Detection?

- **Multiple outputs**: Need to output variable numbers of objects per image
- **Multiple types of outputs**: Need to predict “what” (category label) as well as “where” (bounding box)
- **Large images**: Classification works at 224x224; need higher resolution for detection, often ~800x600

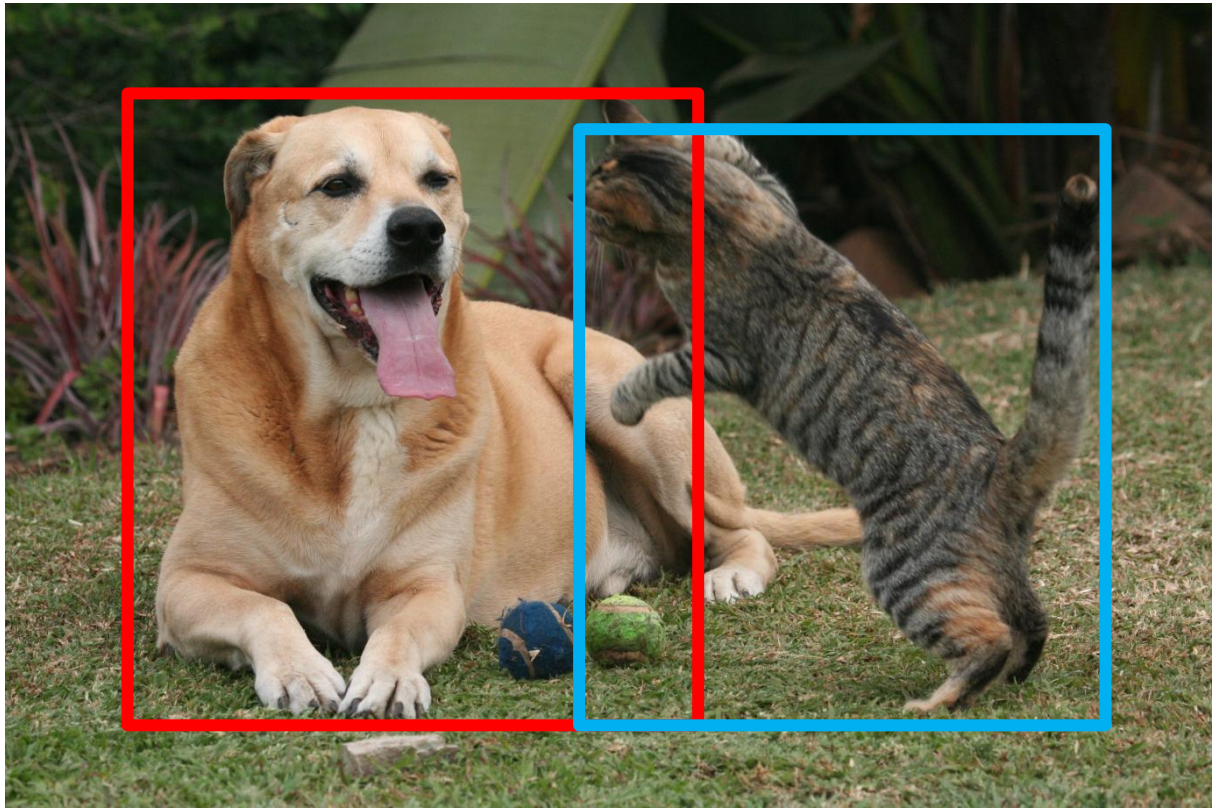


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Classification + Localization (Object Detection for a single object)

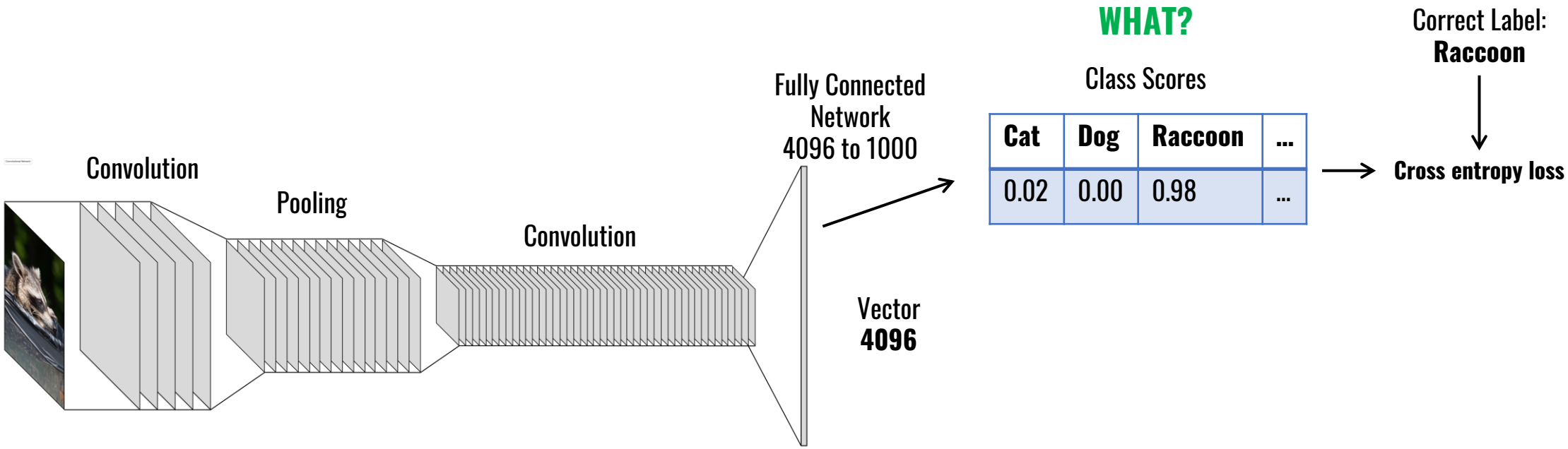


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Classification + Localization (Object Detection for a single object)

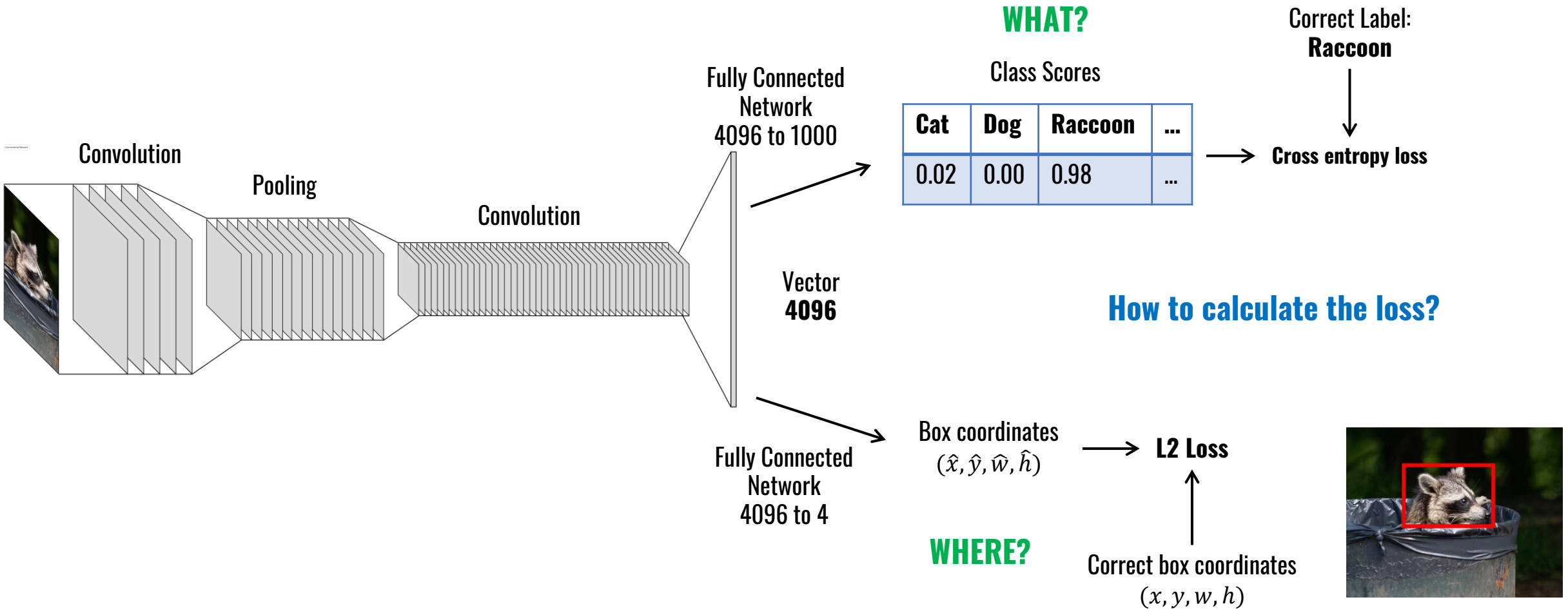


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Classification + Localization (Object Detection for a single object)

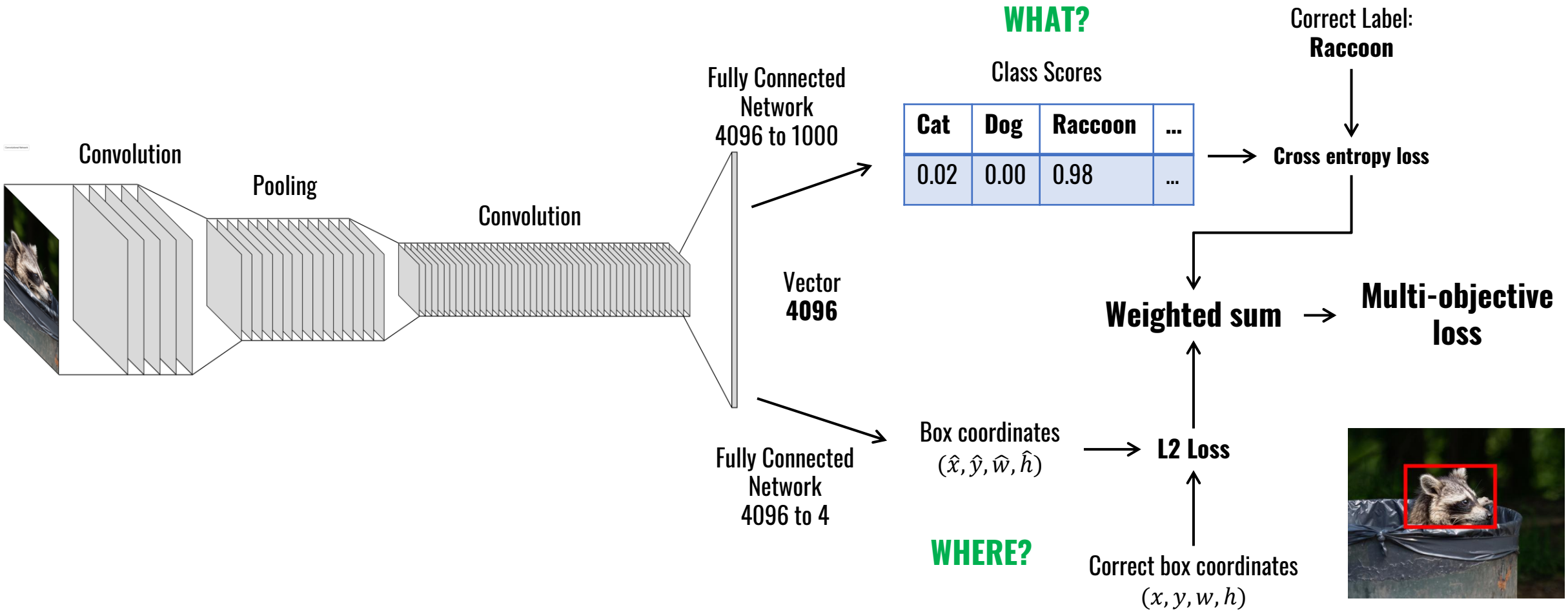
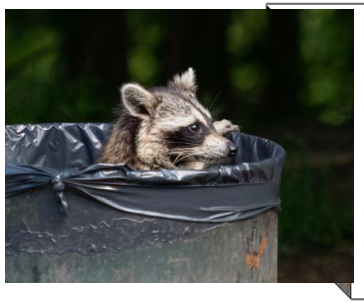


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Classification + Localization (Object Detection for a single object)

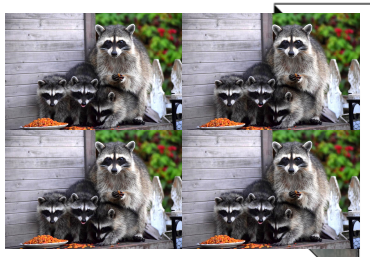
For 1 object on the image



For 4 objects on the image



For a horde of racoons



4 numbers:

(x, y, w, h)

16 numbers:

$$\begin{bmatrix} x_1 & y_1 & w_1 & h_1 \\ x_2 & y_2 & w_2 & h_2 \\ x_3 & y_3 & w_3 & h_3 \\ x_4 & y_4 & w_4 & h_4 \end{bmatrix}$$

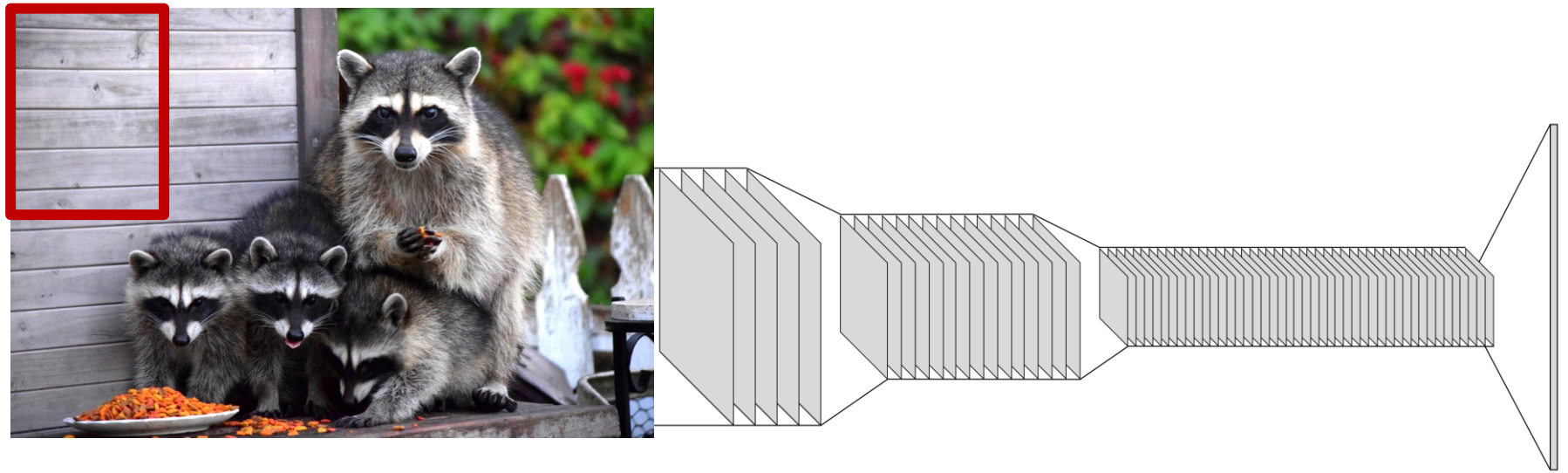
many numbers:

$$\begin{bmatrix} x_1 & y_1 & w_1 & h_1 \\ x_2 & y_2 & w_2 & h_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & w_n & h_n \end{bmatrix}$$

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Sliding Window (Naïve approach)

- Cut the image into smaller overlapping sections, and for each run the image classification
- How big the section should be?
- How much should we move the section?

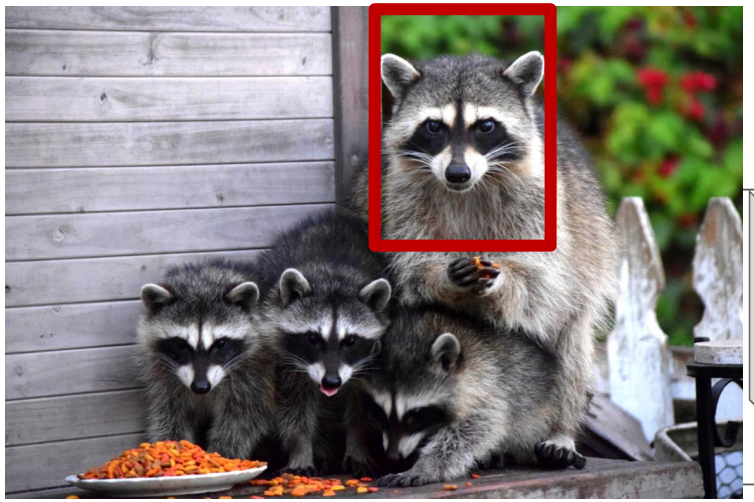


Dog? No
Raccoon? No
Background? Yes

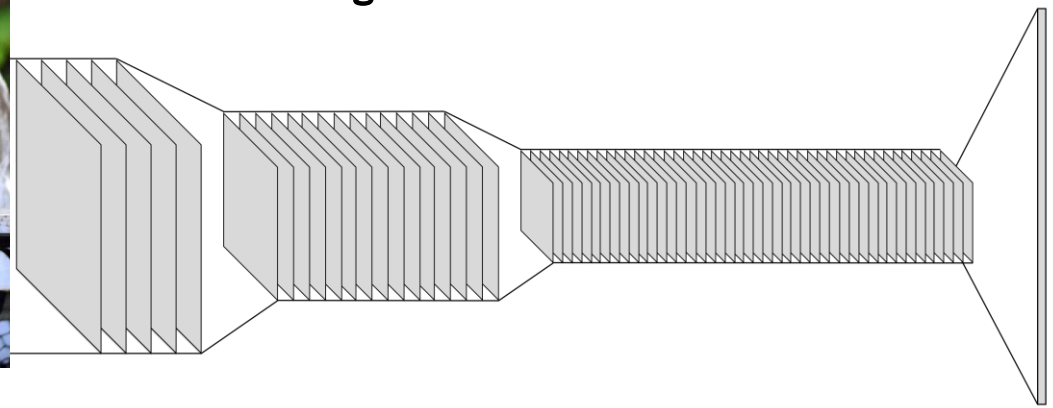
Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Sliding Window (Naïve approach)

- Cut the image into smaller overlapping sections, and for each run the image classification
- How big the section should be?
- How much should we move the section?



If we found a section containing an object, we mark it as a bounding box



Dog? No
Raccoon? Yes
Background? No

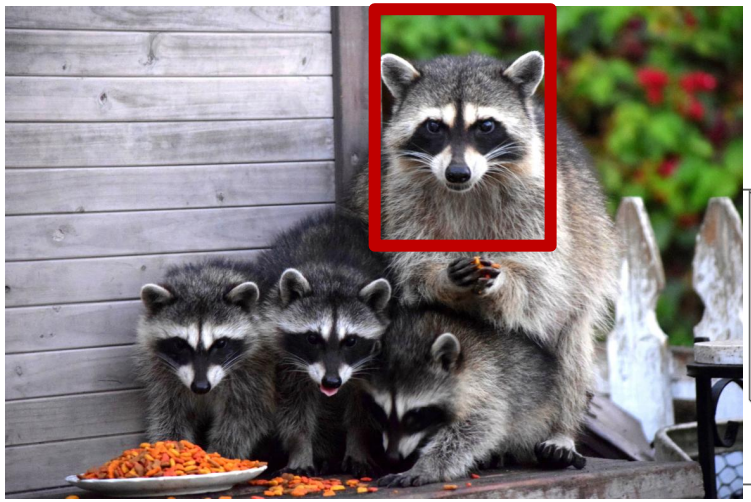
Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Sliding Window (Naïve approach)

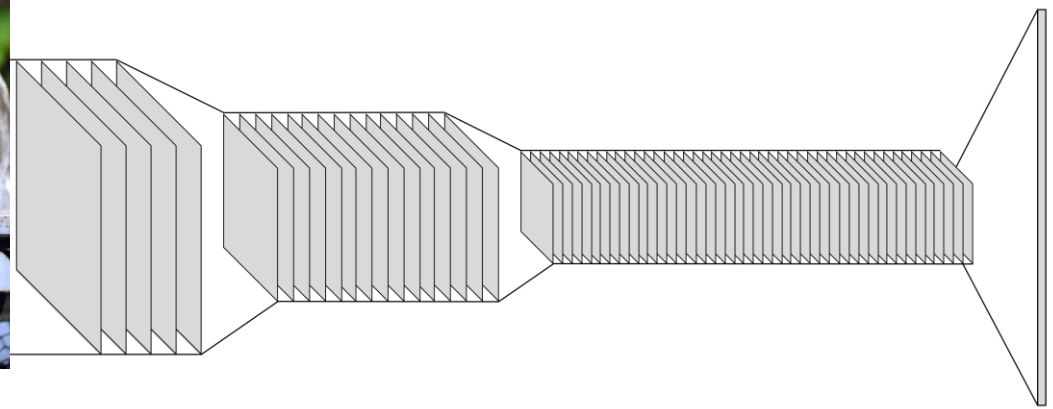
- For large images we would need a lot of sections
- For each section running the detection is computationally heavy
- If we have smaller sections, we will run more inference

If we have an image with 800x600 resolution, window size 200x150 and stride 10 we would get 2806 subsections.

For an image with 1920x1080 resolution, same window size and stride we would get 8460 subsections



CAN WE DO BETTER?



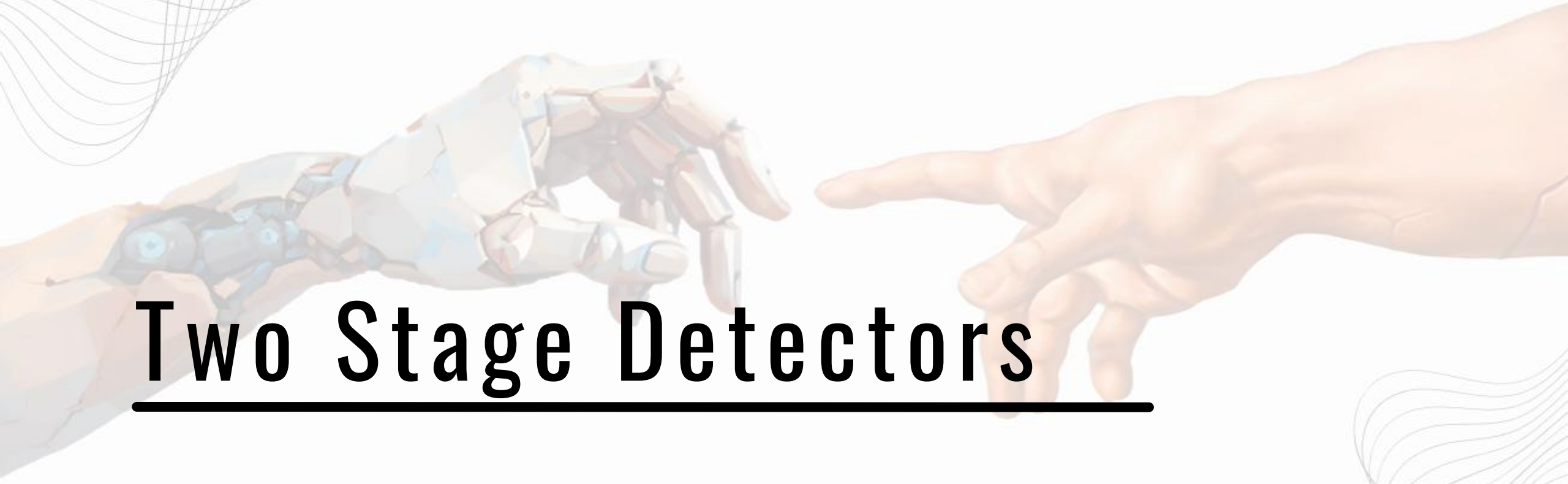
Dog? No
Raccoon? Yes
Background? No

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 5



Two Stage Detectors

1 Two Stage Detectors

2 One Stage Detectors

3 Object Detection Metrics

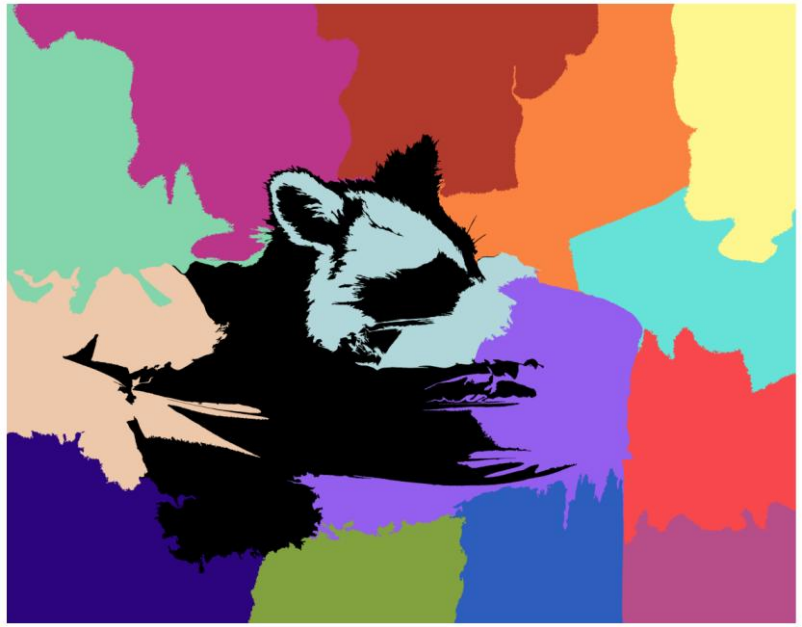
Region Proposal Detectors (Two Stage Detectors)

- Find regions of the image that are likely to contain objects using **Selective Search**
- These candidate regions are called **region proposals**.
- Selective Search can be configured with different parameters and heuristics

Original Image



Proposed Regions



Region Based Bounding Boxes

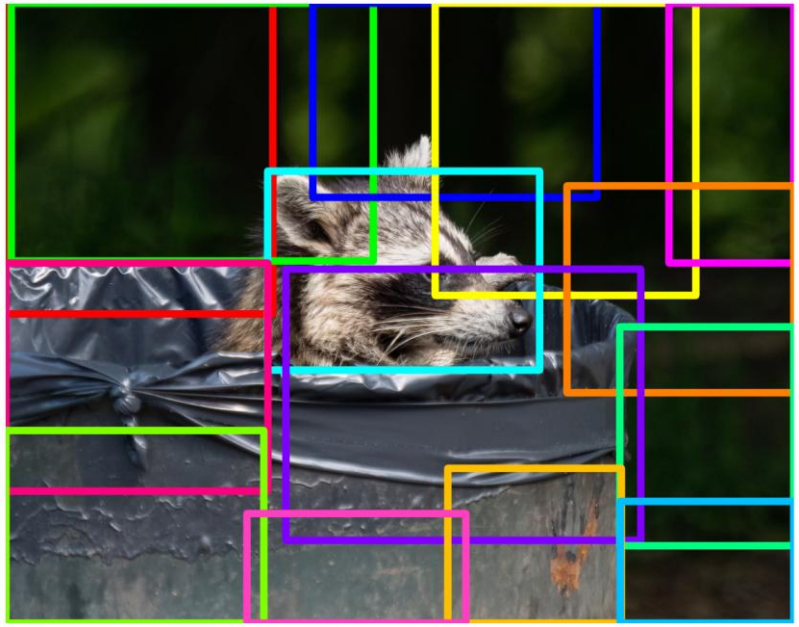
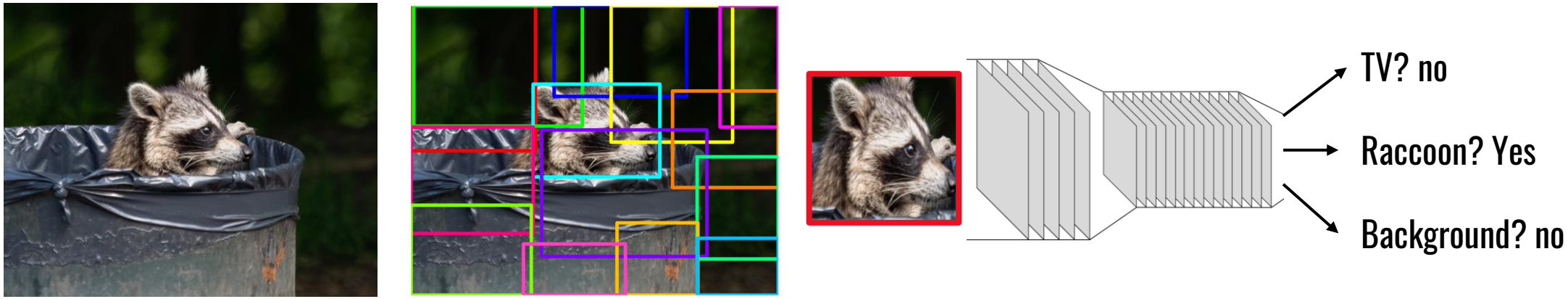


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

R-CNN: Region-Based Convolutional Neural Network



1. Input image
2. Region Proposals
3. Extract CNN features
4. Classify Regions

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1311.2524
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

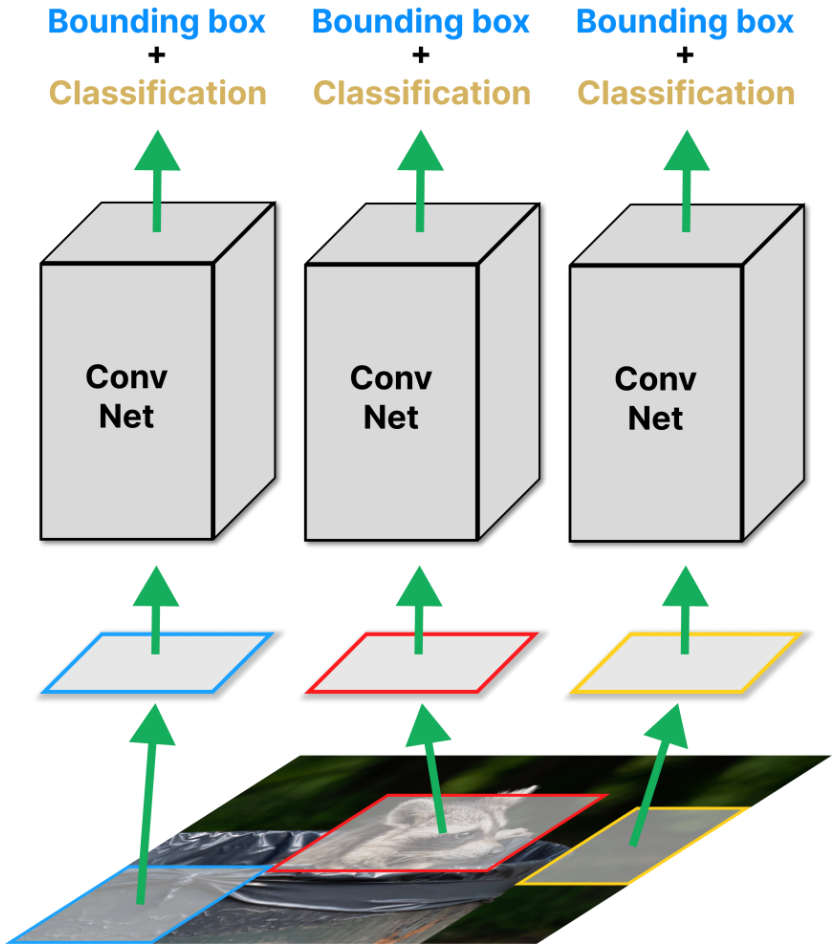
R-CNN: Region-Based Convolutional Neural Network

Classify each region

Forward each region through the CNN

Warped image regions (input of CNN)

Input image and region proposals



The bounding boxes must be warped back to the correct size and shape:

- The bounding box regressor predicts normalized transformation of the proposed region

Proposal: $P = (x_p, y_p, w_p, h_p)$

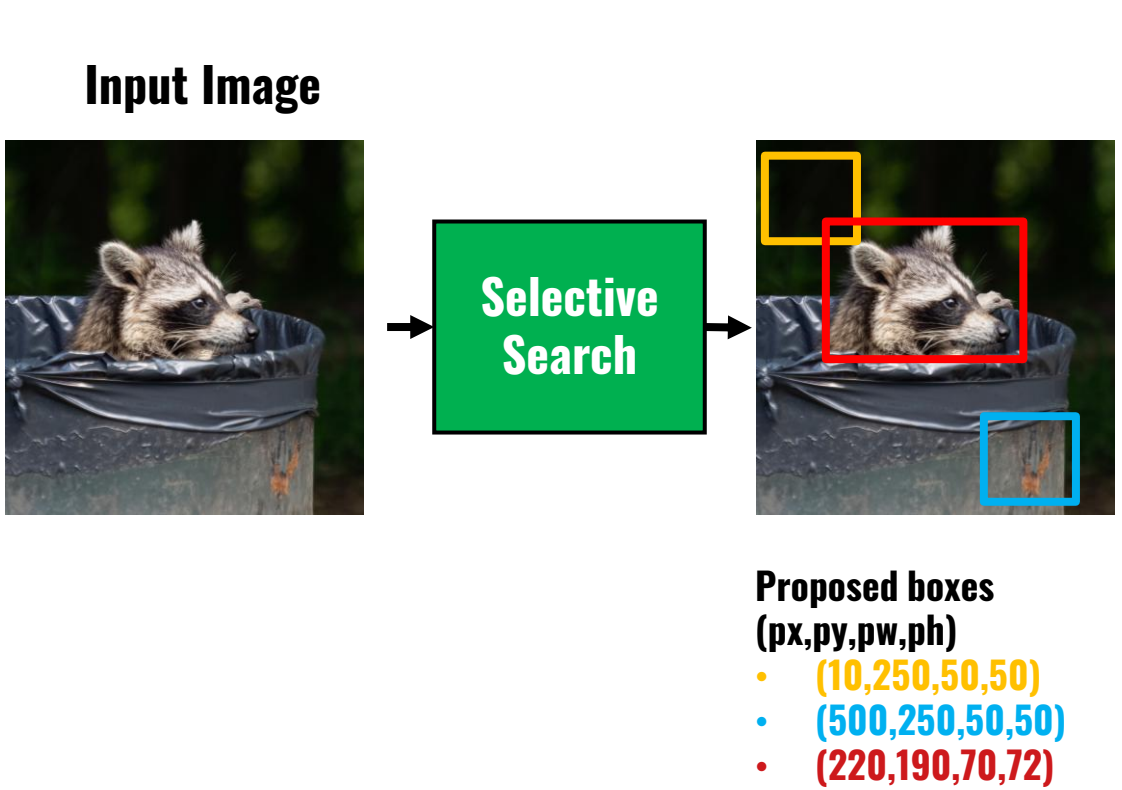
Predicted transformation: $(\hat{t}_x, \hat{t}_y, \hat{t}_w, \hat{t}_h)$

Refined box:

$$\hat{x} = x_p + \hat{t}_x w_p$$
$$\hat{y} = y_p + \hat{t}_y h_p$$
$$\hat{w} = w_p e^{\hat{t}_w}$$
$$\hat{h} = h_p e^{\hat{t}_h}$$

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1311.2524
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

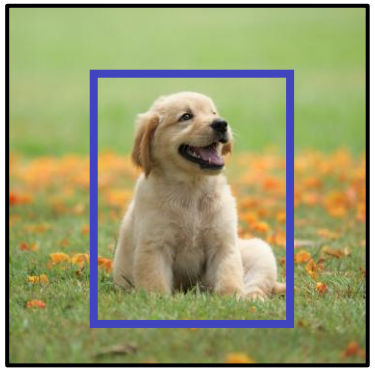
R-CNN: Region-Based Convolutional Neural Network



Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1311.2524
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

R-CNN: Region-Based Convolutional Neural Network

Ground Truth:
- Class: Dog
- Bounding Box (x,y,w,h) = (200,200,120,80)



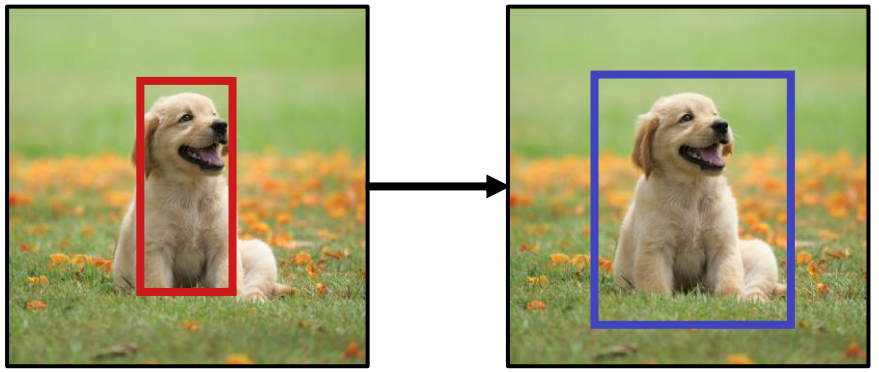
How to transform the proposed box into the correct bounding box?
We use the predicted transformation values (tx,ty,tw,th)
For example:

$$x = p_x + \hat{t}_x$$

The bounding box regression predicts the transformation to refine the proposed box

Proposal

Transformed



Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1311.2524

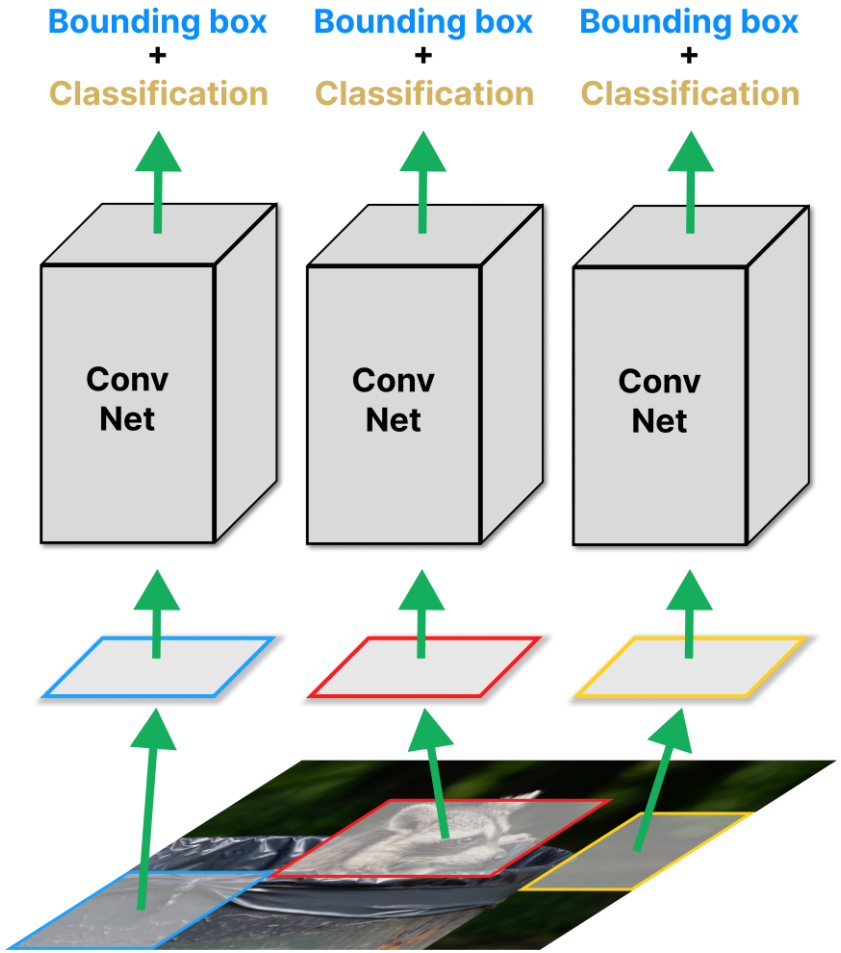
R-CNN: Region-Based Convolutional Neural Network

Classify each region

Forward each region through the CNN

Warped image regions (input of CNN)

Input image and region proposals



The bounding boxes must be warped back to the correct size and shape:

- The bounding box regressor predicts normalized transformation of the proposed region

Problem: Very slow. In practice there are thousands of proposals for each image

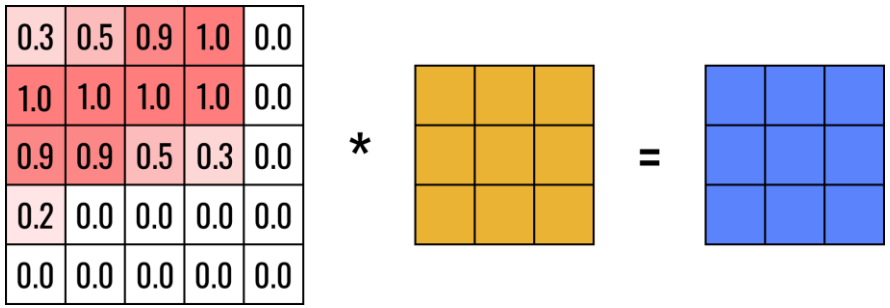
Solution: Run CNN before the proposal

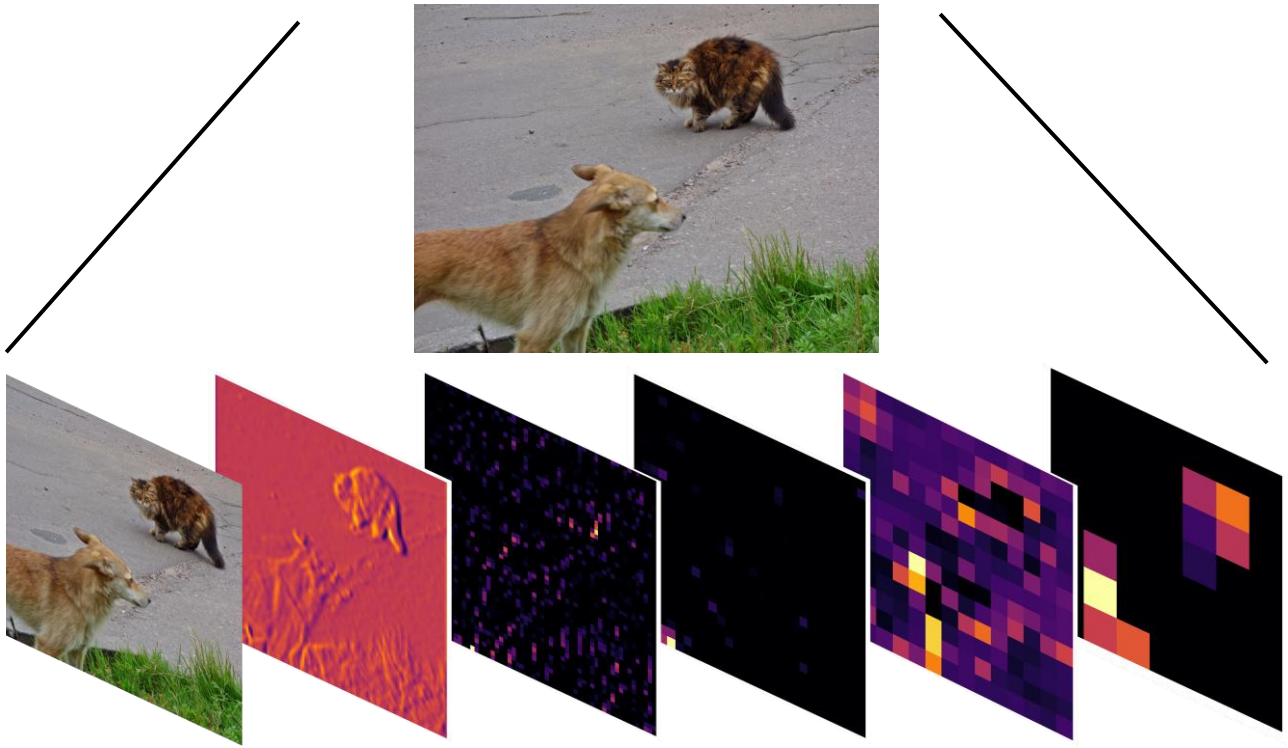
Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1311.2524
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Quick ROI projection to feature map insight

Convolutional Layer

- **Filter / Kernel size = f x f**





Convolution
Input: 5x5
Filter: 3x3
Output: 3x3

- The extracted feature maps already contain information about the objects and can be used to run the region proposal
- Do ROI proposals on the extracted feature map
- In practice it is harder, there are several tricks. For example: ROI Pool and ROI Align

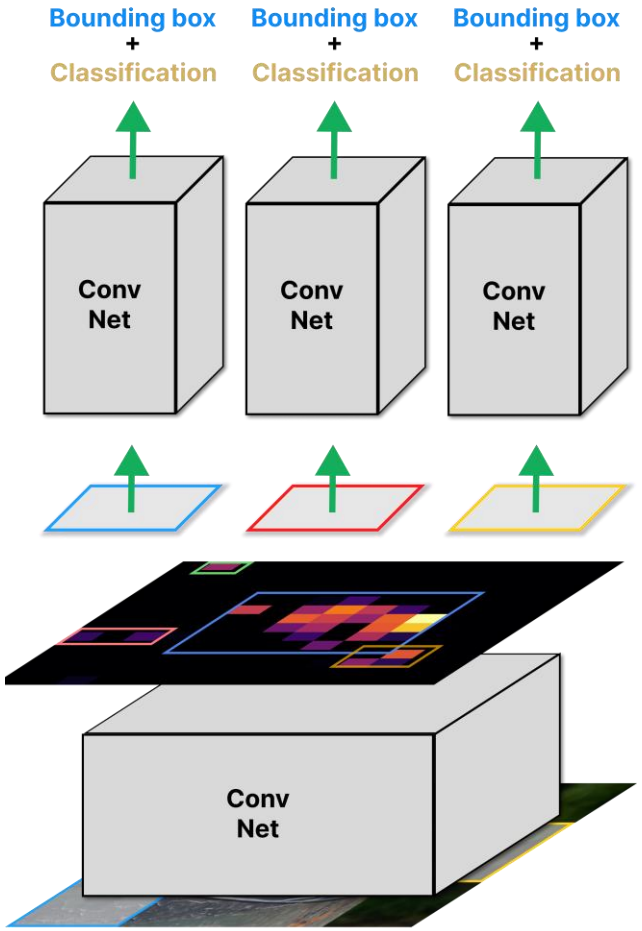
Image Source: Кошка и собака.jpg, Тулип, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Fast R-CNN: Region-Based Convolutional Neural Network

Per region classification and bounding box prediction

Region proposal on feature map

Backbone network: AlexNet, ResNet, etc.

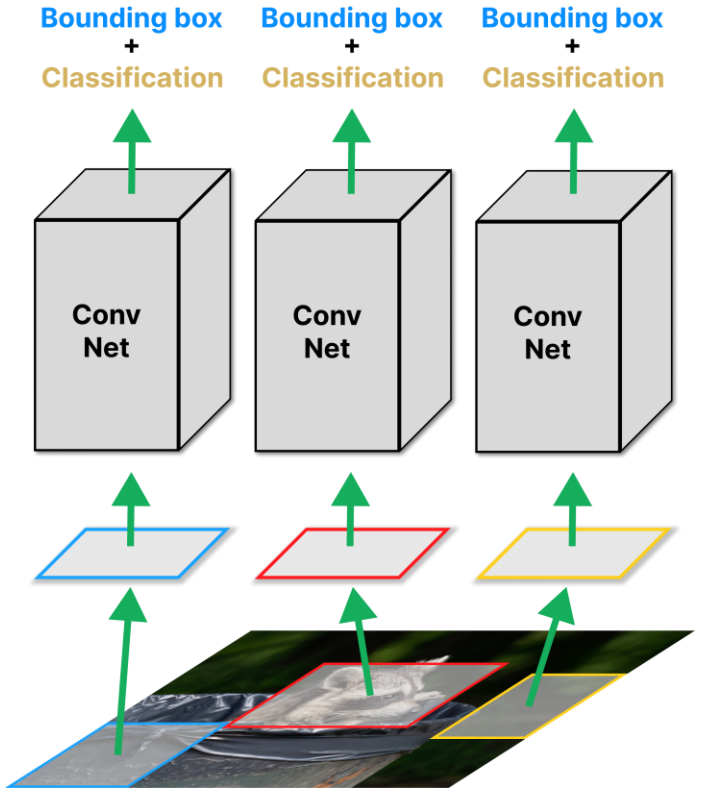


Proposed regions

Image features

Run the whole image through a convolutional network

Process each region independently (R-CNN)



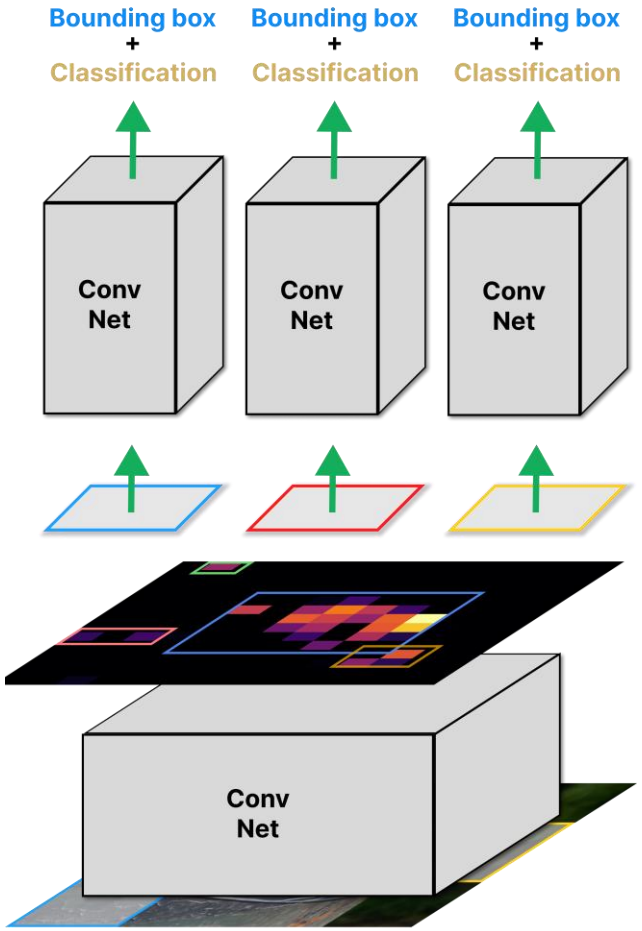
Girshick, R. (2015). Fast R-CNN. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1504.08083
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Fast R-CNN: Region-Based Convolutional Neural Network

Per region classification and bounding box prediction

Region proposal on feature map

Backbone network: AlexNet, ResNet, etc.



Proposed regions

Image features

Run the whole image through a convolutional network

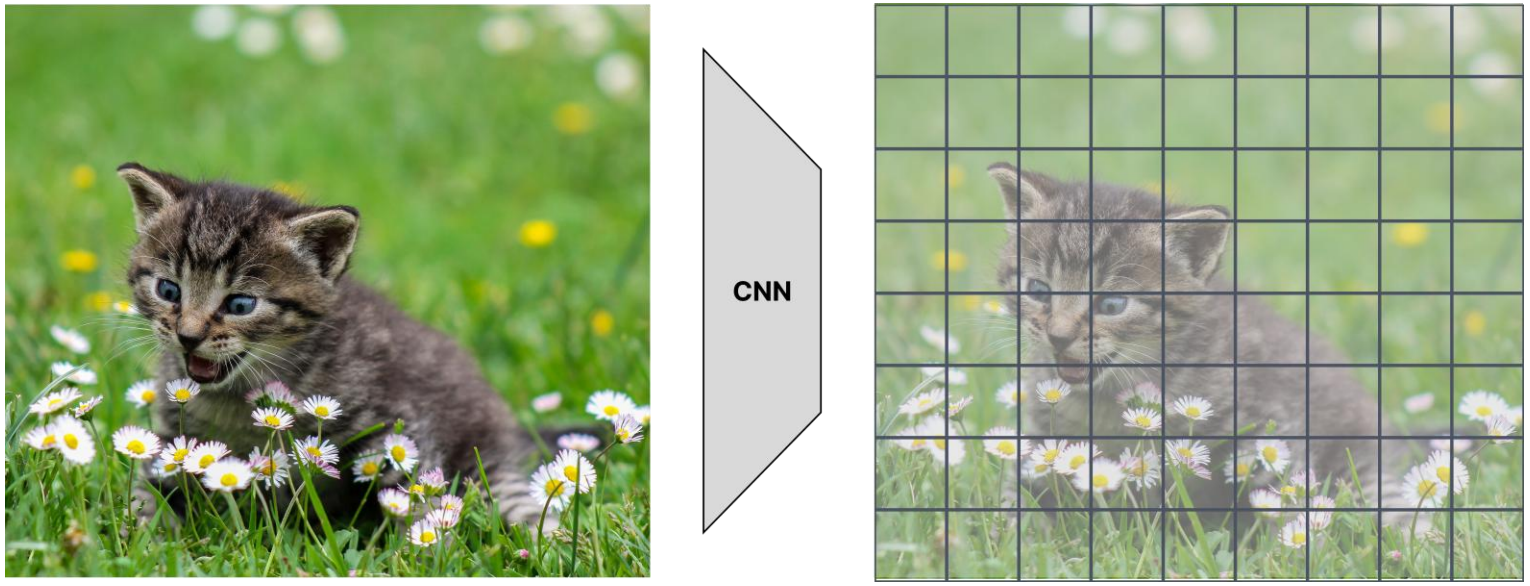
Problem: Region proposal is slow

Solution: Replace the region proposal with a Region Proposal Network (RPN)

Girshick, R. (2015). Fast R-CNN. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1504.08083
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Region Proposal Network (RPN)

Run backbone CNN to extract features from the image



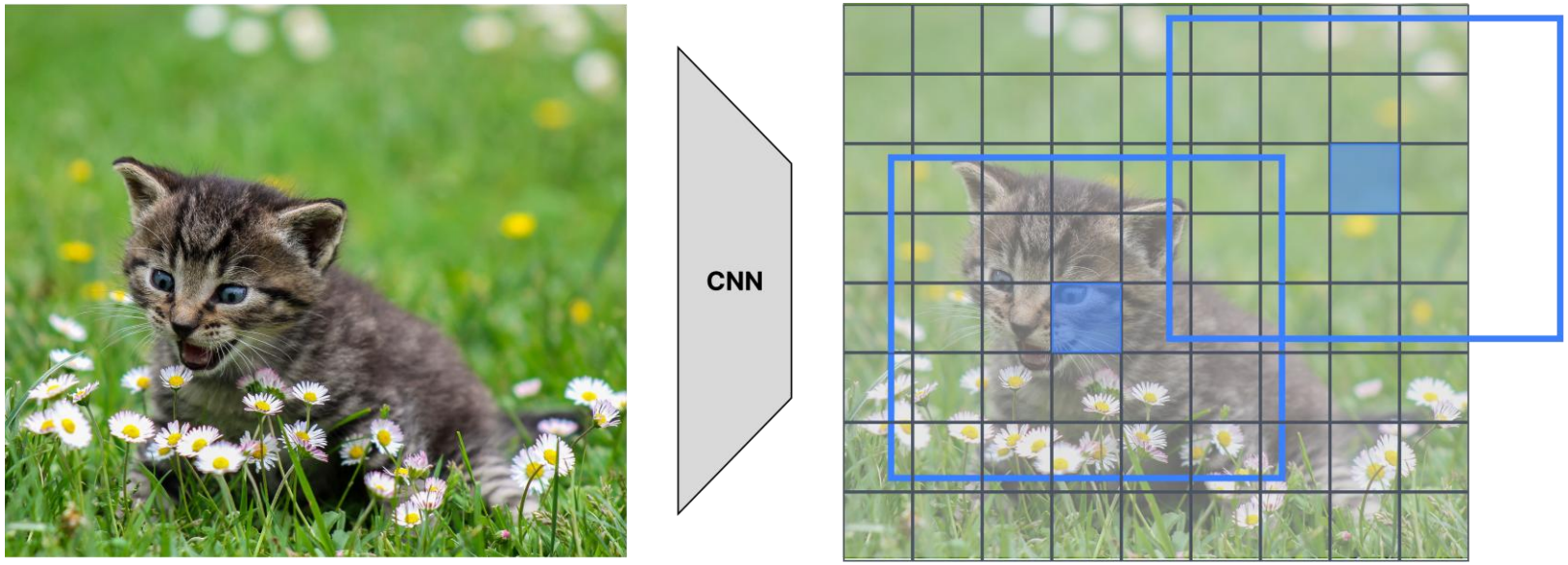
Input Image
(e.g., 3 x 640 x 480)

Feature map
(e.g., 512 x 20 x 15)

Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Region Proposal Network (RPN)

Run backbone CNN to extract features from the image



Input Image
(e.g., 3 x 640 x 480)

Feature map
(e.g., 512 x 20 x 15)

Imagine an anchor box with a fixed size at each point of the feature map

Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

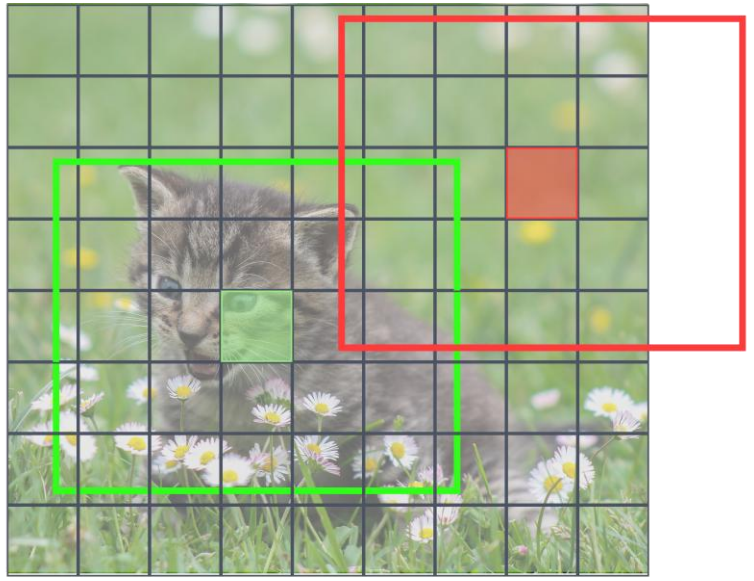
Region Proposal Network (RPN)

Run backbone CNN to extract features from the image



Input Image
(e.g., 3 x 640 x 480)

CNN



Feature map
(e.g., 512 x 20 x 15)

CNN

Imagine an anchor box with a fixed size at each point of the feature map

Is the anchor box an object?
1 x 20 x 15

At each point ask whether the anchor box contains an object. Logistic regression (binary class.) and score prediction

Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Region Proposal Network (RPN)

Run backbone CNN to extract features from the image



Input Image
(e.g., 3 x 640 x 480)

CNN



Feature map
(e.g., 512 x 20 x 15)

CNN

Imagine an anchor box with a fixed size at each point of the feature map

Is the anchor box an object?
1 x 20 x 15

Box transformations
4 x 20 x 15

For boxes containing objects predict the anchor box transformation

Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Region Proposal Network (RPN)

Run backbone CNN to extract features from the image



Input Image
(e.g., 3 x 640 x 480)

CNN



Feature map
(e.g., 512 x 20 x 15)

CNN

Problem: Anchor boxes may have the wrong size

Solution: Use K predefined anchor box

Is the anchor box an object?
K x 20 x 15

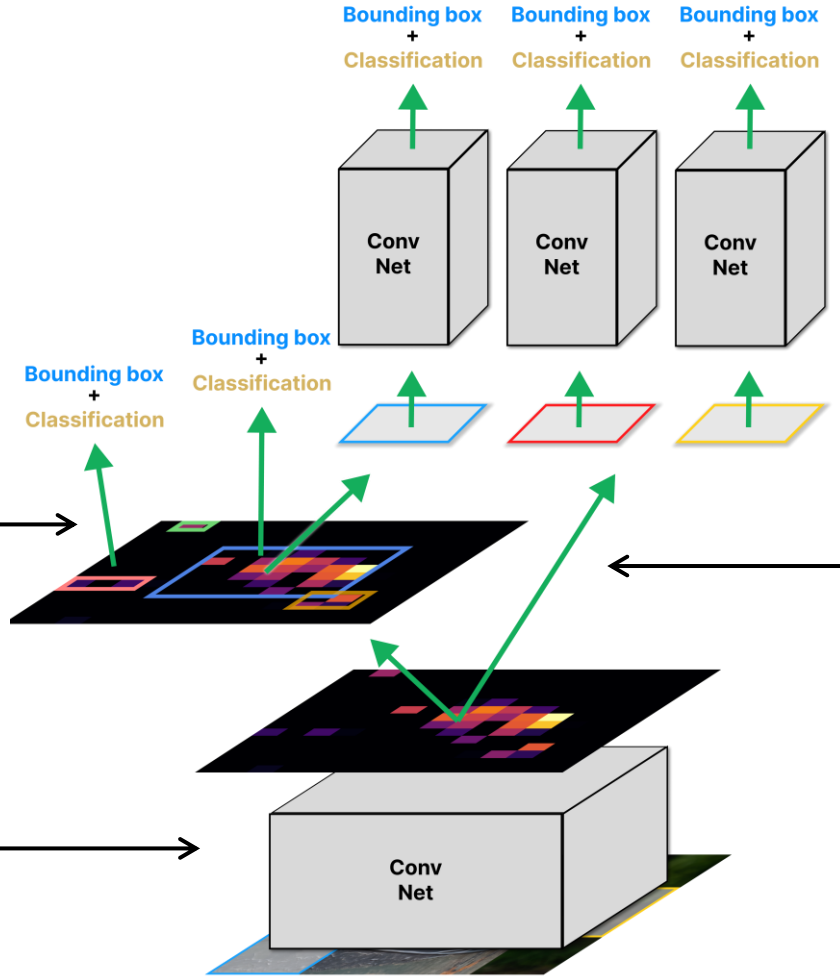
Box transformations
4K x 20 x 15

At test time sort all the boxes based on their scores and take the top ~300 as region proposals

Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Faster R-CNN: Learnable Region Proposals

Same as Fast R-CNN but a **Region Proposal Network (RPN)** is inserted



Classify each proposed regions

The **Region Proposal Network** tells which region to pay attention to

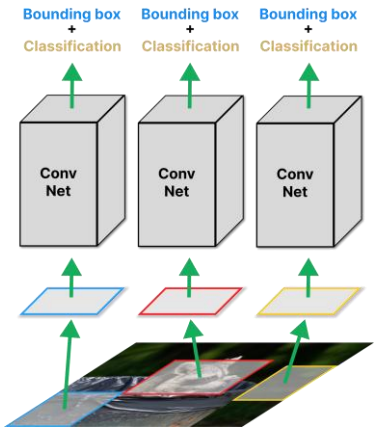
ROI Pooling: crop and summarize features from the proposed regions into a $C \times 7 \times 7$ tensor (split proposed region into a 7x7 grid, pool grid cells)

Pretrained backbone creates the feature map

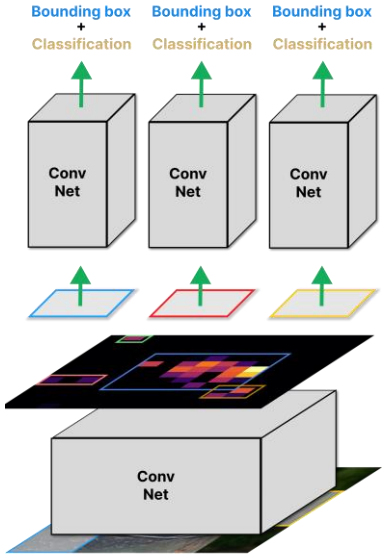
Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497

R-CNN Family Comparison

R-CNN runs CNN for each region proposal independently



Fast R-CNN uses a CNN to create a feature map, then runs region proposal on the feature map. Runs detection for each proposed region



Faster R-CNN replaces region proposal with a **Region Proposal Network (RPN)**

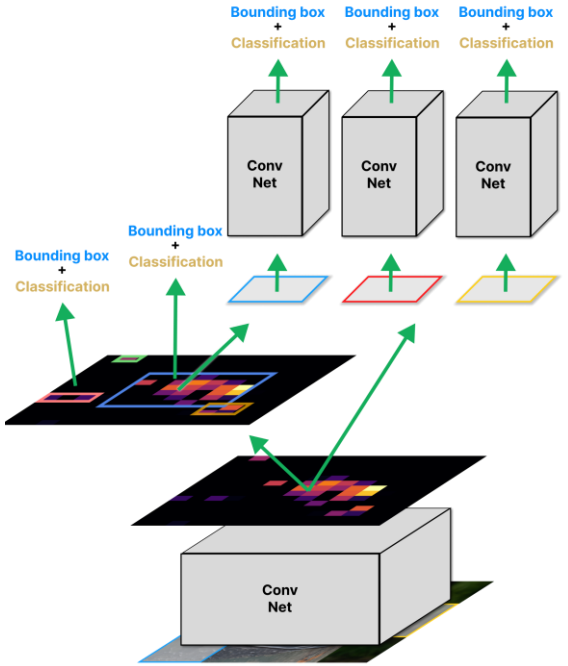


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

R-CNN Family

The R-CNN family does the detection in two stages:

First Stage (runs one per image):
- Region Proposal (algorithm/network)
- Feature extraction (for region proposals)

Second Stage (runs once per region):
- Crop features (Roi pooling/align)
- Predict object/class
- Predict bounding box

Question: Do we need both stages?

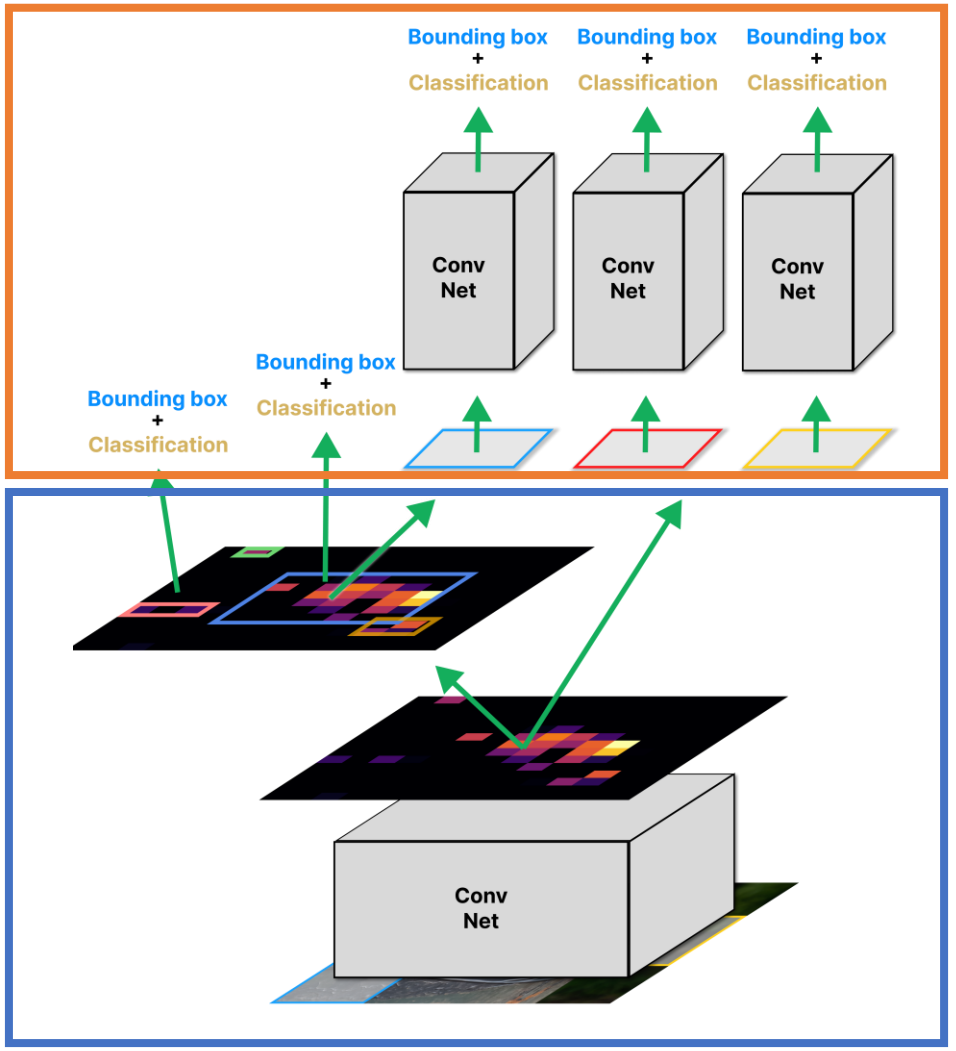
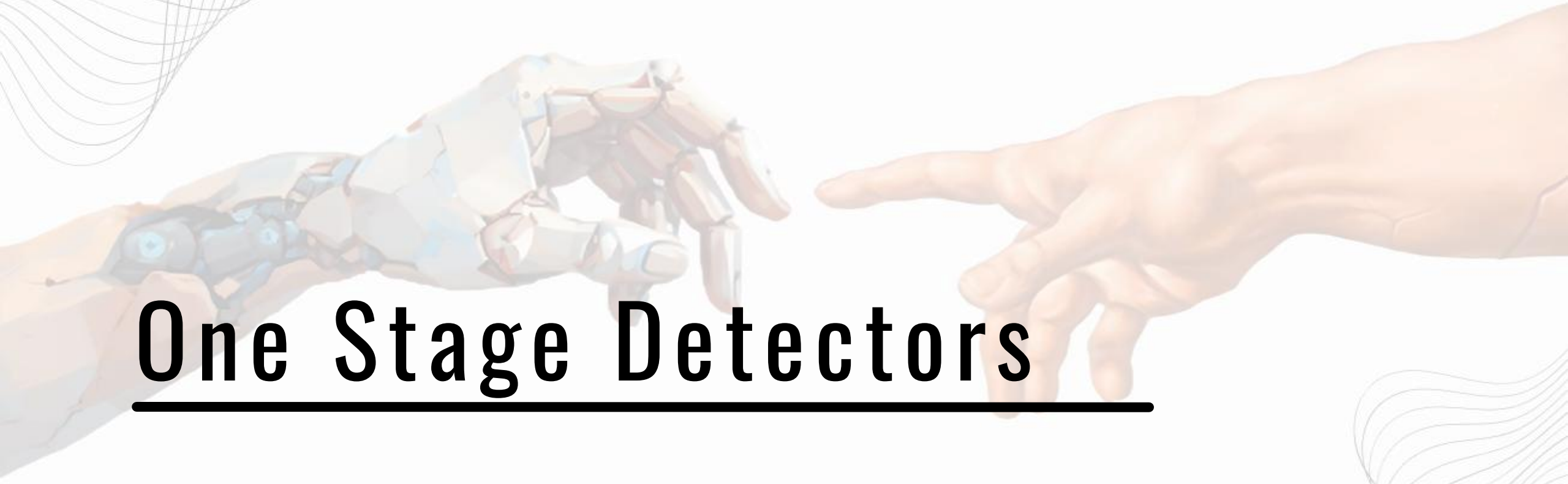


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 8



One Stage Detectors

1 Two Stage Detectors

2 One Stage Detectors

3 Object Detection Metrics

Problems with Faster R-CNN

Run backbone CNN to extract features from the image



Input Image
(e.g., 3 x 640 x 480)

CNN



Feature map
(e.g., 512 x 20 x 15)

CNN

Problem: Anchor boxes may have the wrong size

Solution: Use K predefined anchor box

Is the anchor box an object?
K x 20 x 15

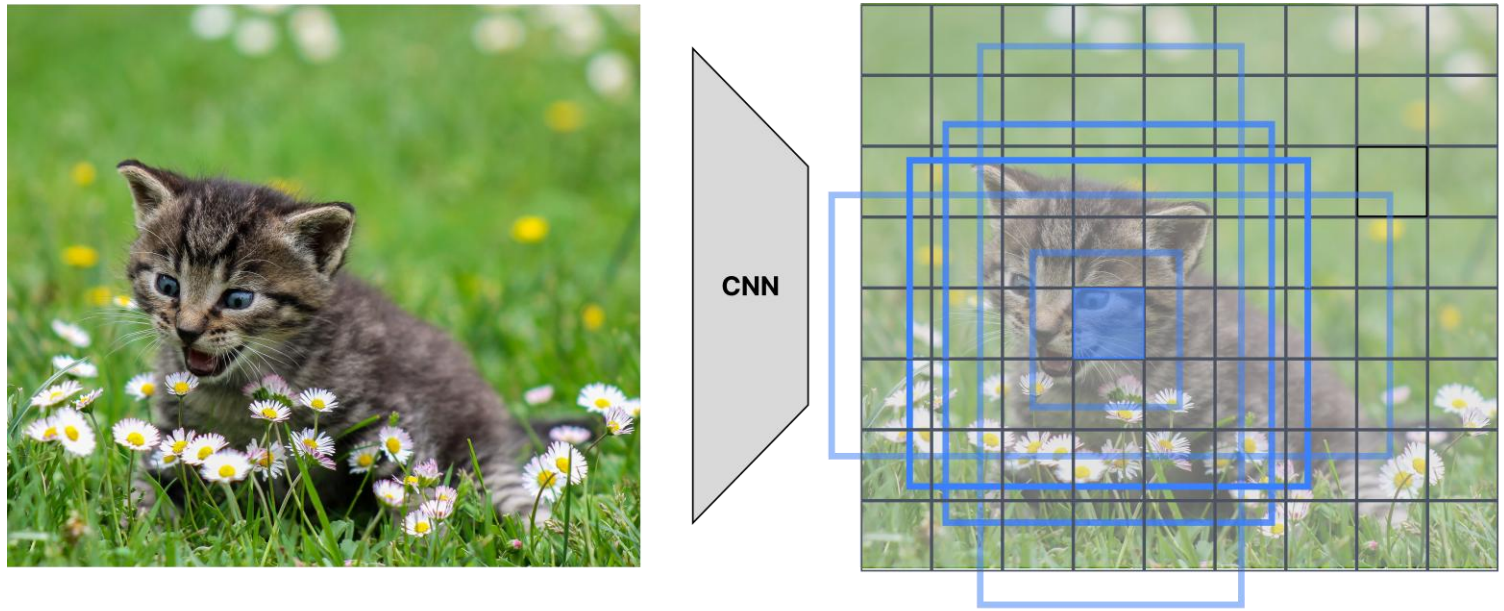
Box transformations
4K x 20 x 15

At test time sort all the boxes based on their scores and take the top ~300 as region proposals

Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Single-Stage Object Detection

Run backbone CNN to extract features from the image



Input Image
(e.g., 3 x 640 x 480)

Feature map
(e.g., 512 x 20 x 15)

RPN: Classify each anchor as an object/not object

Single-Stage Detector: Classify each object as one of **C** categories (or background)

Anchor category
(C+1) x K x 20 x 15

Box transformations
4K x 20 x 15

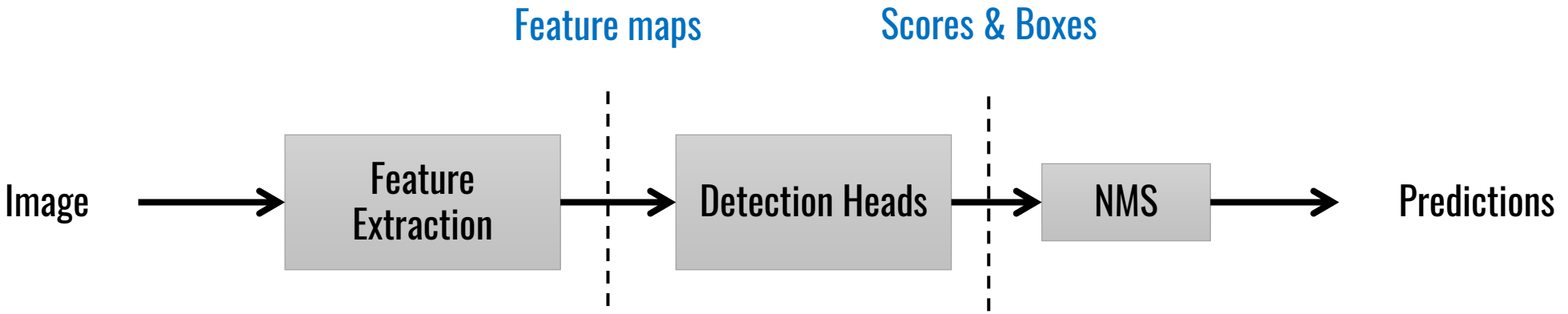
Remember: K anchor at each position in the image feature map

Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Single-Shot Detectors (SSD)

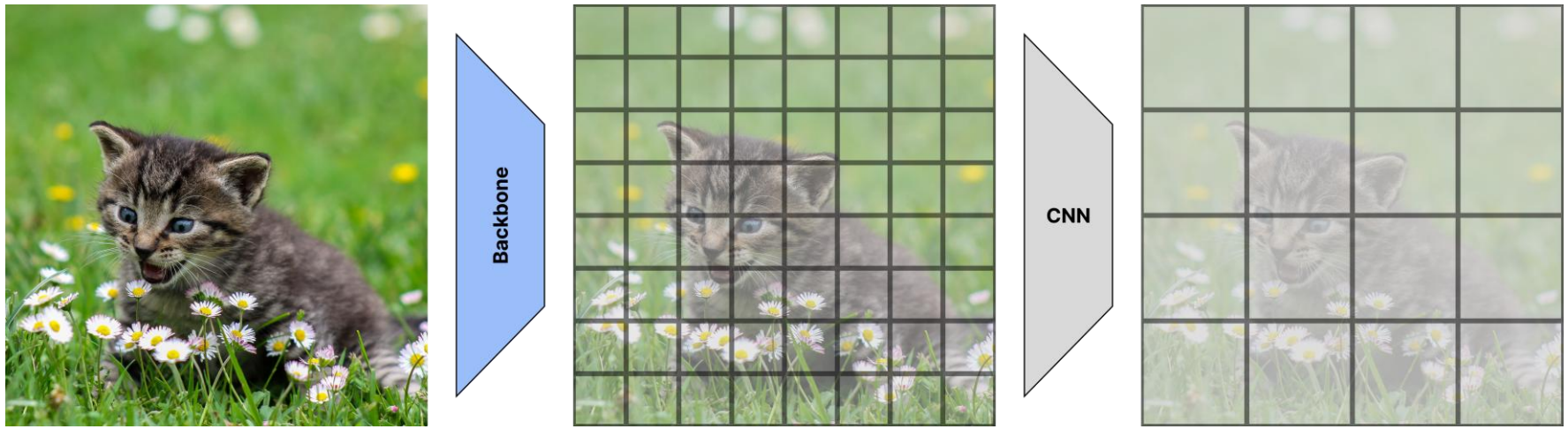
Key Ideas:

1. Multiple Layers → handle different scales
2. Different filters predict boxes of different shapes/sizes



Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2

Single-Shot Detectors (SSD)

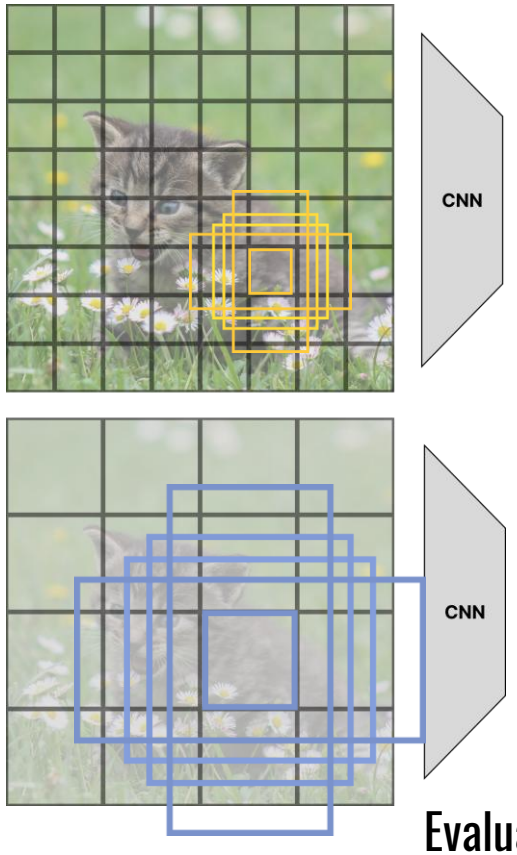


Run backbone CNN to extract features from the image (e.g. VGG, ResNet)

Create feature maps with different scales (for large and small objects)

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2
Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Single-Shot Detectors (SSD)



$$\hat{x} = x_p + \hat{t}_x w_p$$
$$\hat{y} = y_p + \hat{t}_y h_p$$
$$\hat{w} = w_p e^{\hat{t}_w}$$
$$\hat{h} = h_p e^{\hat{t}_h}$$

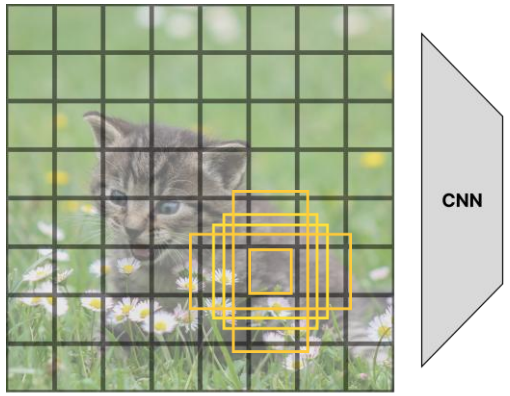
Predict shape offsets and confidences for all object categories

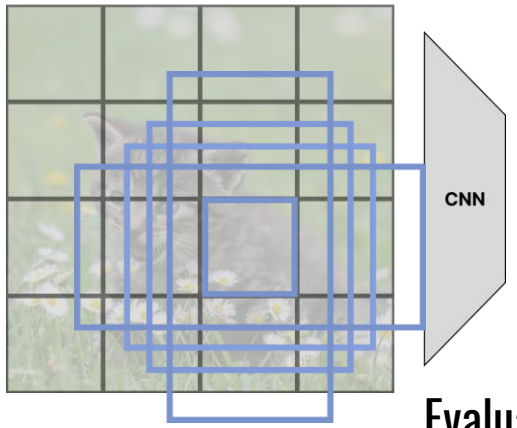
Car? 0.02
Cat? 0.8
Raccoon? 0.2

Evaluate the different default anchor boxes for each feature map scales

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2
Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CCO, via Wikimedia Commons

Single-Shot Detectors (SSD)





Evaluate the different default anchor boxes for each feature map scales

Predict shape offsets and confidences for all object categories

$$\hat{x} = x_p + \hat{t}_x w_p$$
$$\hat{y} = y_p + \hat{t}_y h_p$$
$$\hat{w} = w_p e^{\hat{t}_w}$$
$$\hat{h} = h_p e^{\hat{t}_h}$$

Car? 0.02
Cat? 0.8
Raccoon? 0.2

Non-Max Suppression

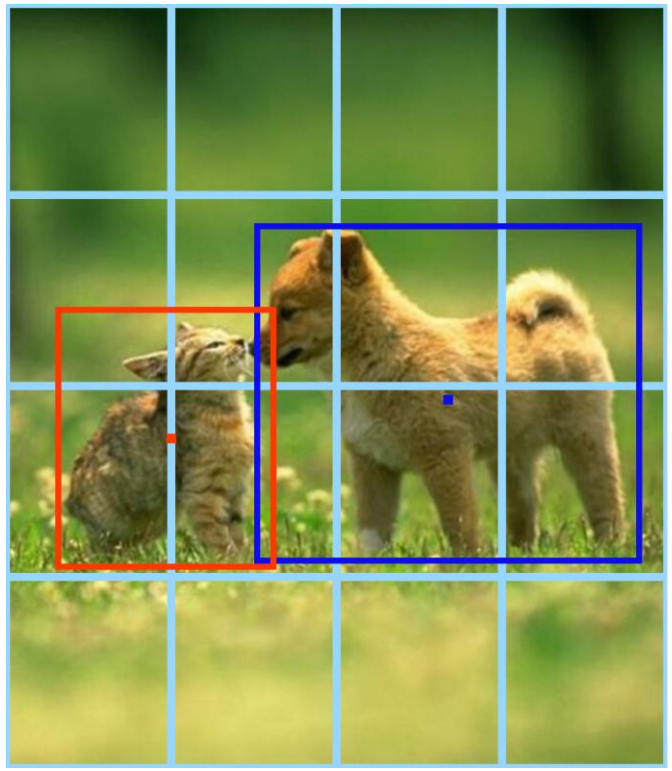
Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). doi:10.1007/978-3-319-46448-0_2
Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CCO, via Wikimedia Commons

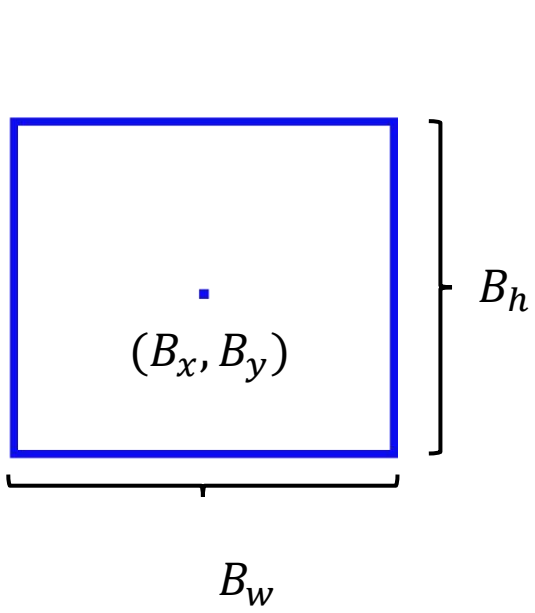
You Only Look Once (YOLO)

- Anchor boxes are highly overlapped in SSD
- YOLO cuts the input image uniformly into S x S anchor boxes
- Each anchor box predicts B bounding boxes
- Later versions add more improvements

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640

You Only Look Once (YOLO)

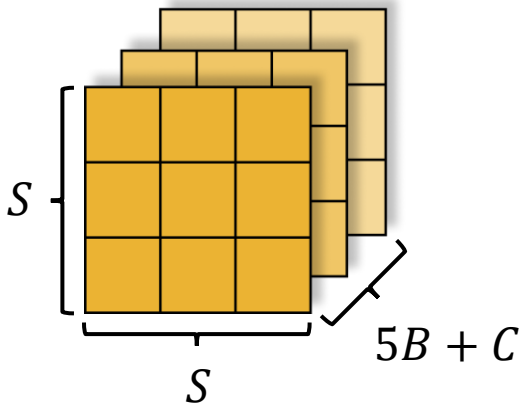




$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix}$$

- P_c: Probability of containing an object
- B_x, B_y: Bounding box center coordinates
- B_w, B_h: Bounding box width and height
- C_1, C_2: Probability of bounding box belonging to the given class, if the bounding box contains an object

$$S \times S \times (5B + C)$$

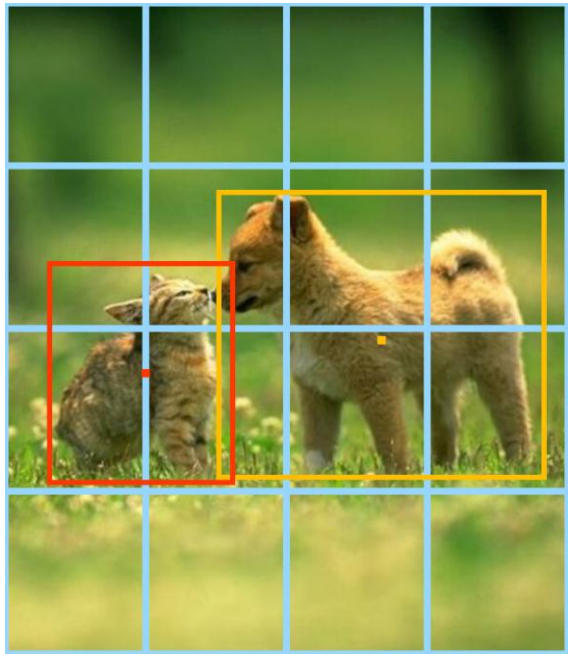


Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640
Image Source: Cat and dog (4984798619).jpg, kitty.green66, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

You Only Look Once (YOLO) - Format

$$\begin{pmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{pmatrix} \begin{pmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$$



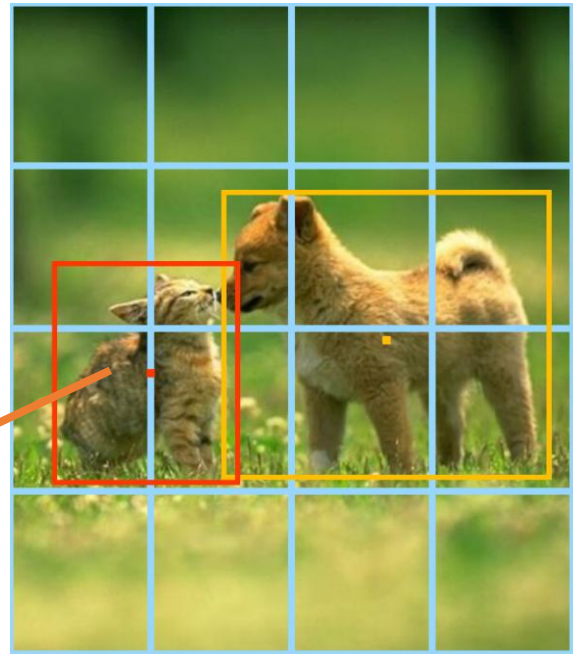


Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640
Image Source: Cat and dog (4984798619).jpg, kitty.green66, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

You Only Look Once (YOLO) - Format

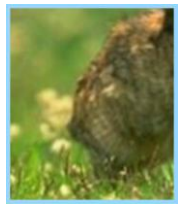
$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix} \begin{bmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{bmatrix}$$





$$\begin{bmatrix} 1 \\ 0.05 \\ 0.3 \\ 2 \\ 1.3 \\ 1 \\ 0 \end{bmatrix}$$

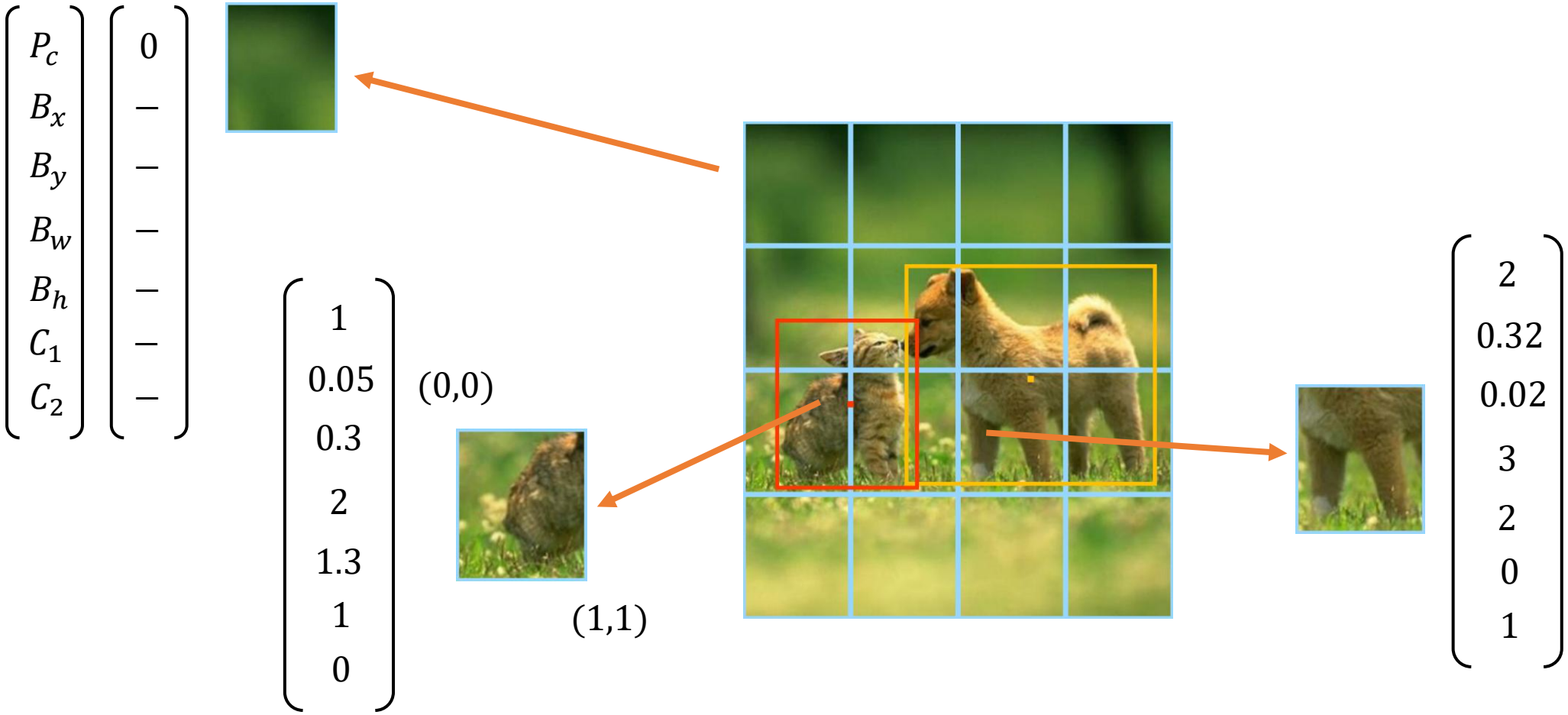
(0,0)



(1,1)

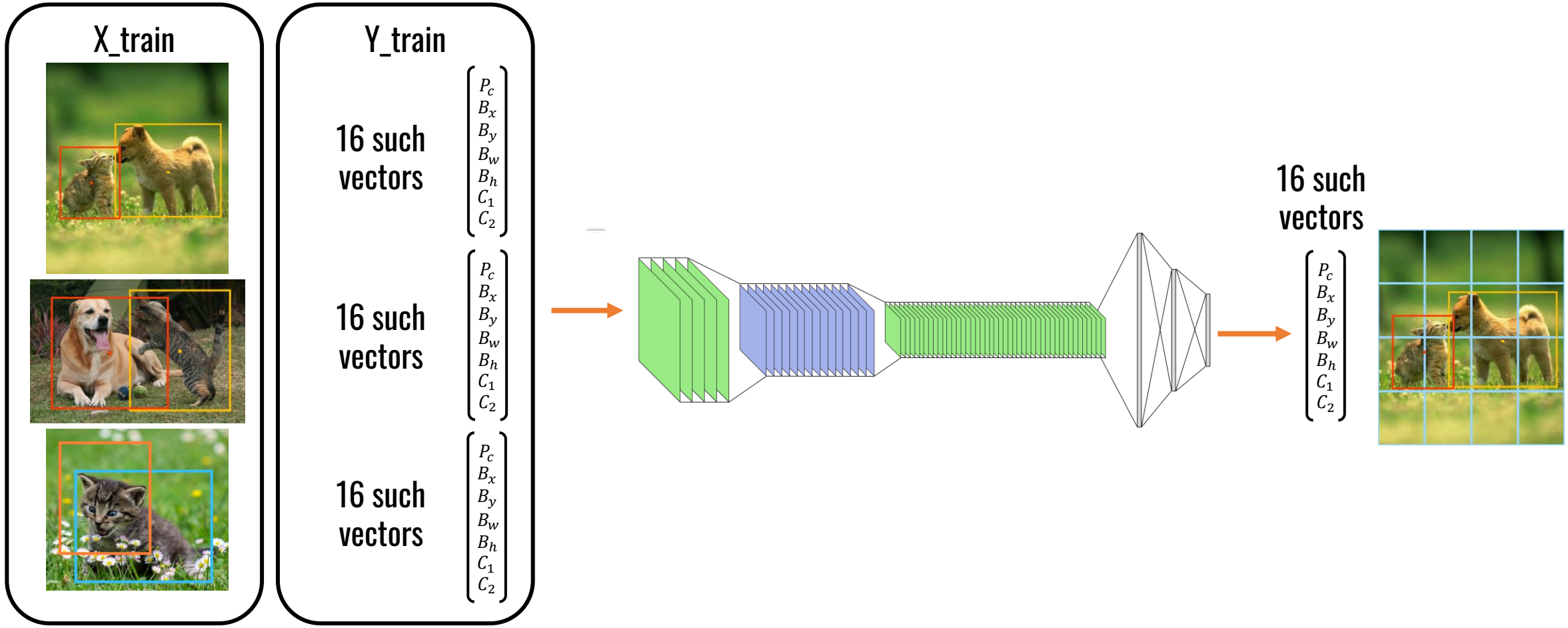
Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640
Image Source: Cat and dog (4984798619).jpg, kitty.green66, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

You Only Look Once (YOLO) - Format



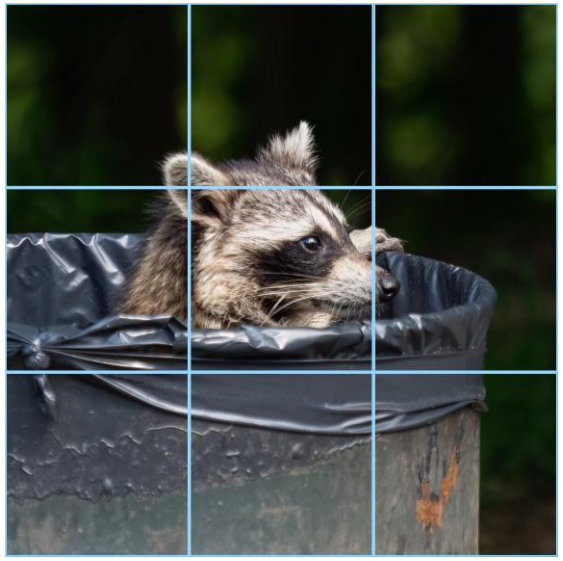
Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640
Image Source: Cat and dog (4984798619).jpg, kitty.green66, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

You Only Look Once (YOLO) - Prediction

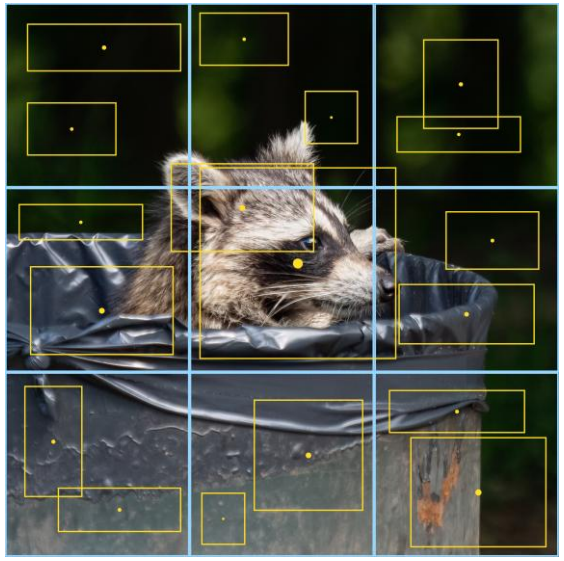


Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.02640
Image Source: Cat and dog (4984798619).jpg, kitty.green66, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons
Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons
Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

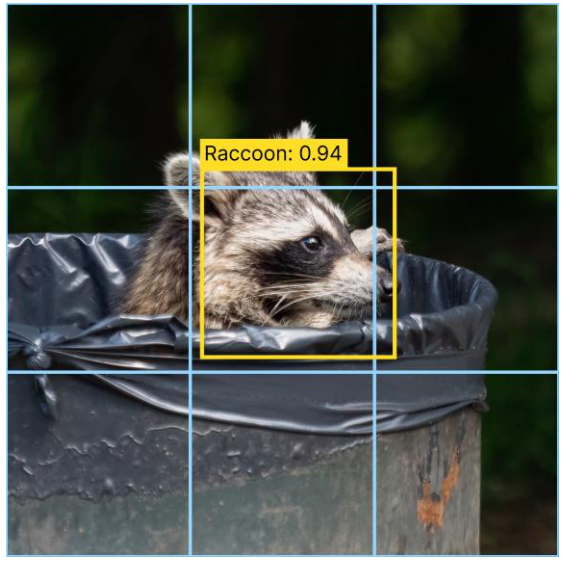
You Only Look Once (YOLO)



1. The image is divided into SxS grid. (S = 3)



2. Each cell is responsible for detecting B objects if the object is within the cell. (B = 2)



3. Each bounding box has a confidence score. Boxes below a predefined threshold are suppressed. (i.e. threshold = 0.9)

Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

You Only Look Once (YOLO)

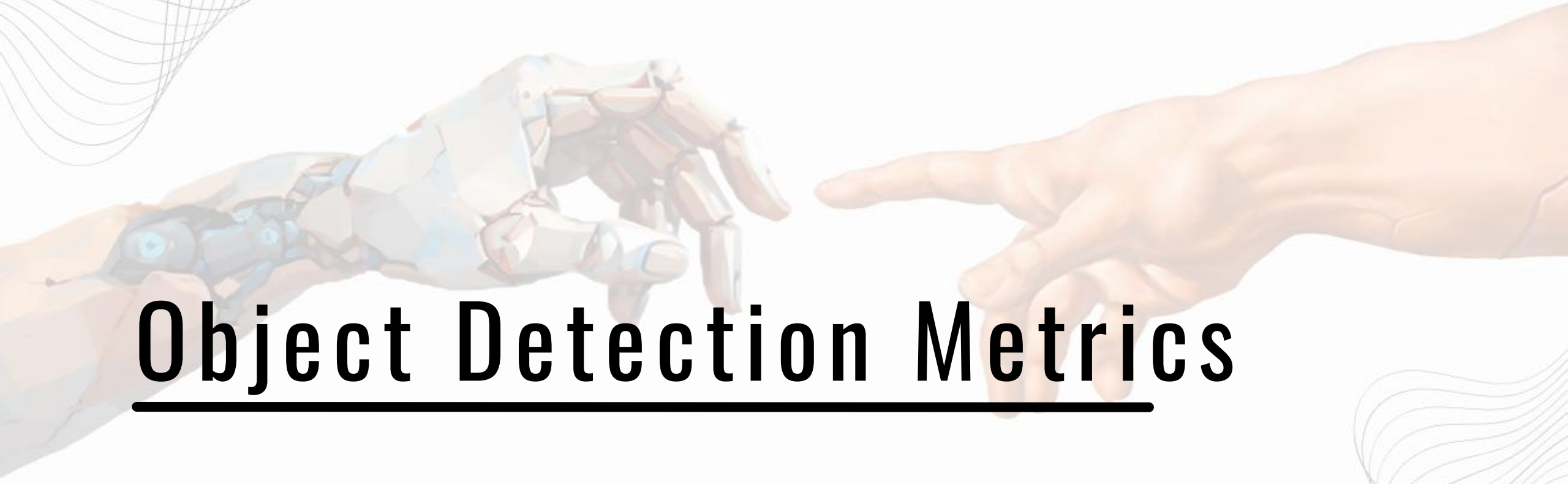
Please check the following links for a detailed explanation of YOLO versions:

- https://www.v7labs.com/blog/yolo-object-detection
- https://www.datacamp.com/blog/yolo-object-detection-explained
- Video comparison

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 5



Object Detection Metrics

1 Two Stage Detectors

2 One Stage Detectors

3 Object Detection Metrics

Comparing predictions: Accuracy

In case of binary classification, the binary accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

where:

- TP = True positive,
- FP = False Positive,
- TN = True negative
- FN = False negative

Comparing predictions: Accuracy

In case of multi class classification, the accuracy is:

$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{all classifications}}$$

- This is usually expressed as a percentage, i.e. 90%
- To get a better insight we usually visualize it in a **confusion matrix**



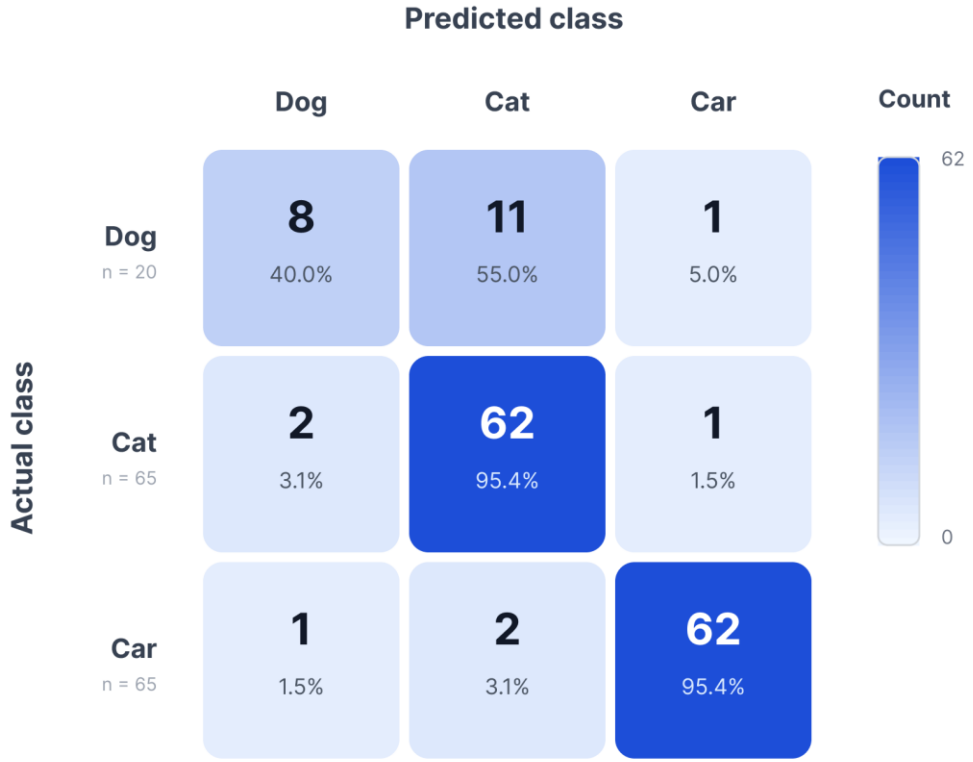
Confusion Matrix

Comparing predictions: Accuracy

Would you consider this model good?

- Correct classifications: 8 + 62 + 62 = 132
- All classifications: 20 + 65 + 65 = 150
- Accuracy: 132/150 = 0.88

The accuracy is 88%, but we can see that the dog is misclassified



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

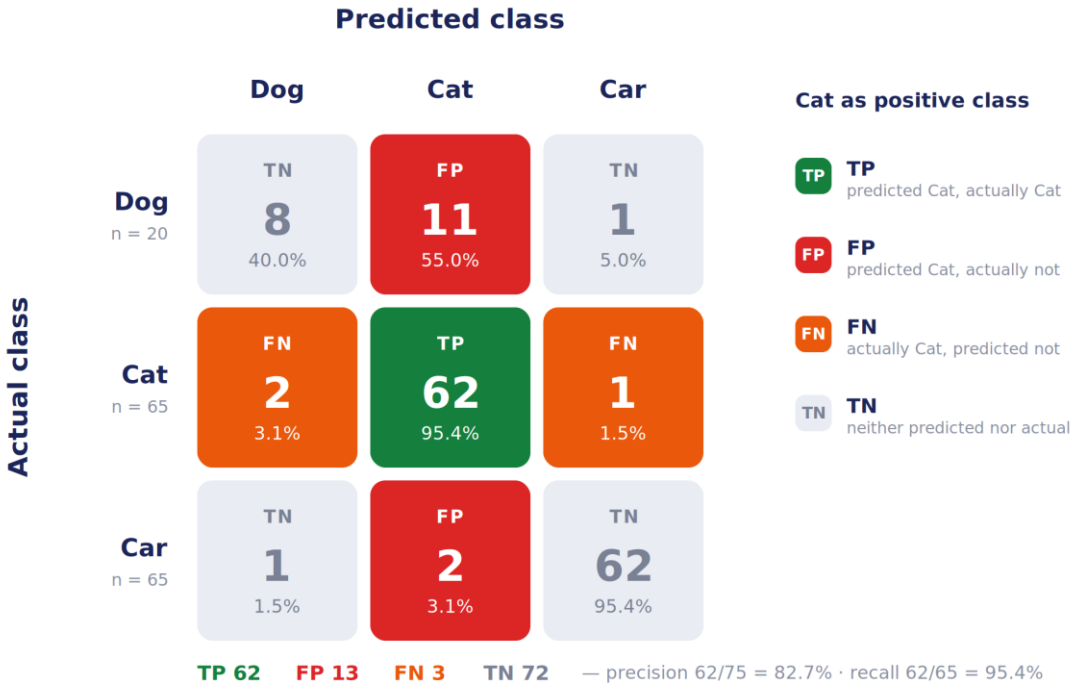
Comparing predictions: Precision

Precision answers: *Of all predicted dog detections, how many were actually dogs?*

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

True Positives (TP): predicted class and the actual class is the same – in the case of the dog, **8**
False Positives (FP): predicted as a dog but belongs to a different class: **3**

In this case the precision for the dog class: $\frac{8}{8+3} = 0.73$



Confusion Matrix

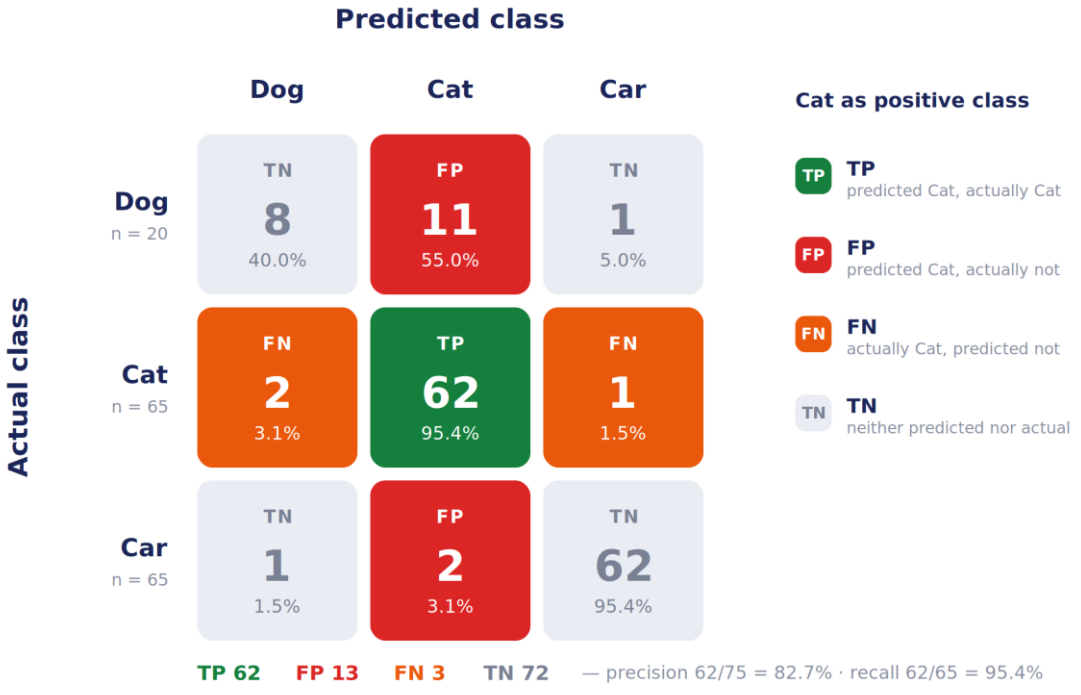
Comparing predictions: Recall

Recall answers: *Of all actual dogs, how many did the detector find?*

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

False Negative (FN): Predicted as a cat, but the actual class is dog – in this case, **12**

In this case the recall for the dog class: $\frac{8}{8+12} = 0.4$



Confusion Matrix

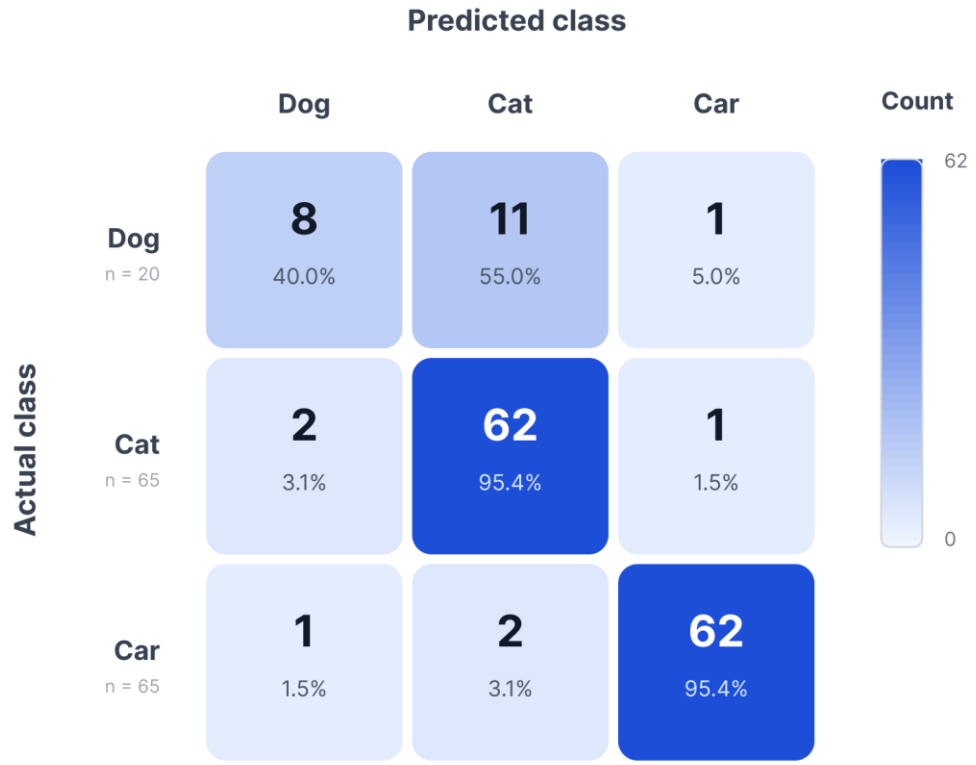
Comparing predictions: Observation

Accuracy: 88%
Precision: 73%
Recall: 40%

Number of instances for each class:

1. Dog – 20
2. Cat – 65
3. Car – 65

Observation: Most dogs are classified as cats because the dog class is underrepresented



Comparing predictions: F1 score

F1 score is a single metric that balances precision and recall. It is the harmonic mean of the two:

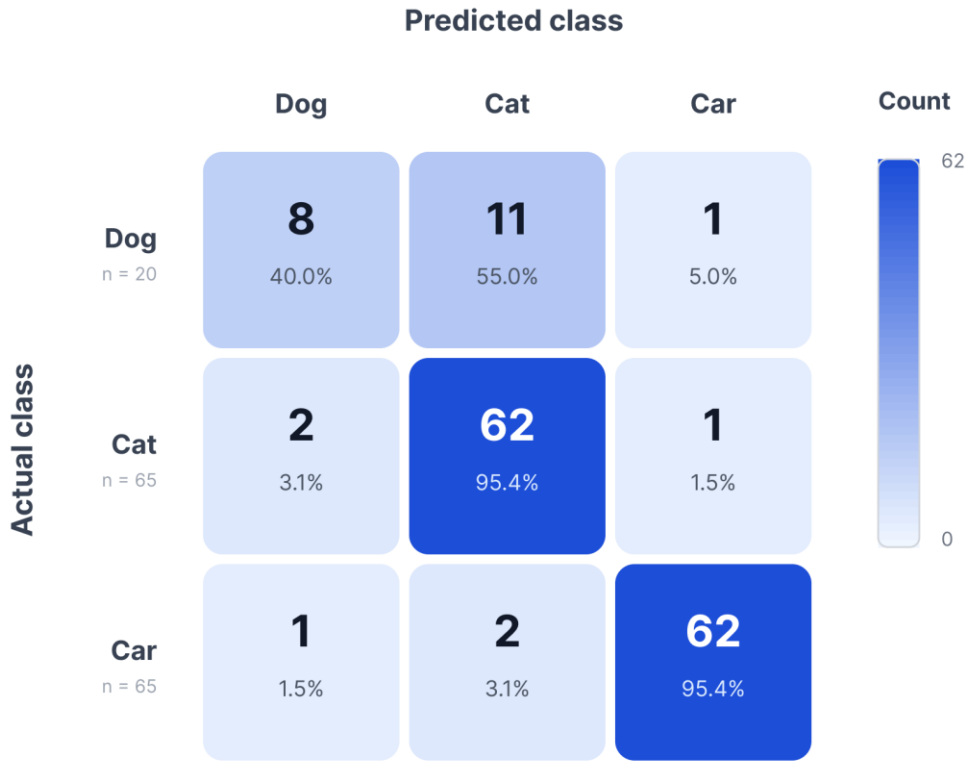
$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

High F1 score: The model achieves a good balance between finding objects and avoiding false detections

In case of the dog:

$$F_1 = 2 \cdot \frac{0.80 \times 0.40}{0.80 + 0.40} = 0.53$$

The model preforms poorly on dogs.



Comparing boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground truth box?

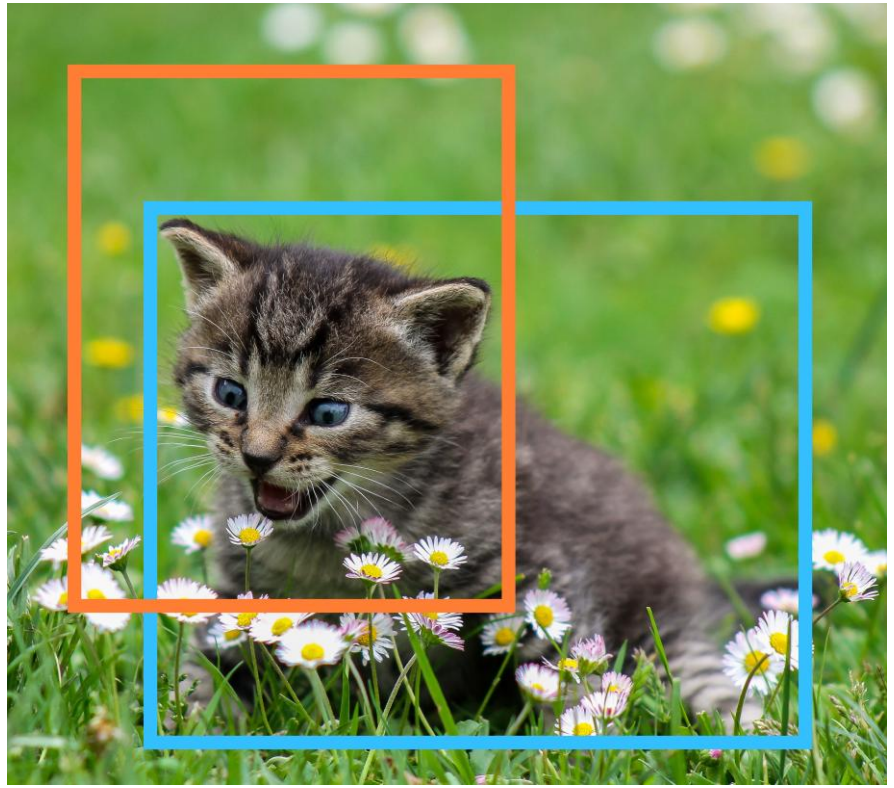


Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Comparing boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground truth box?

Intersection over Union (IoU) measures how much a predicted bounding box overlaps the ground-truth box:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union}$$

An IoU of 1 means perfect overlap; 0 means no overlap.

In object detection, a prediction is often considered correct when its IoU exceeds a chosen threshold, such as 0.5.

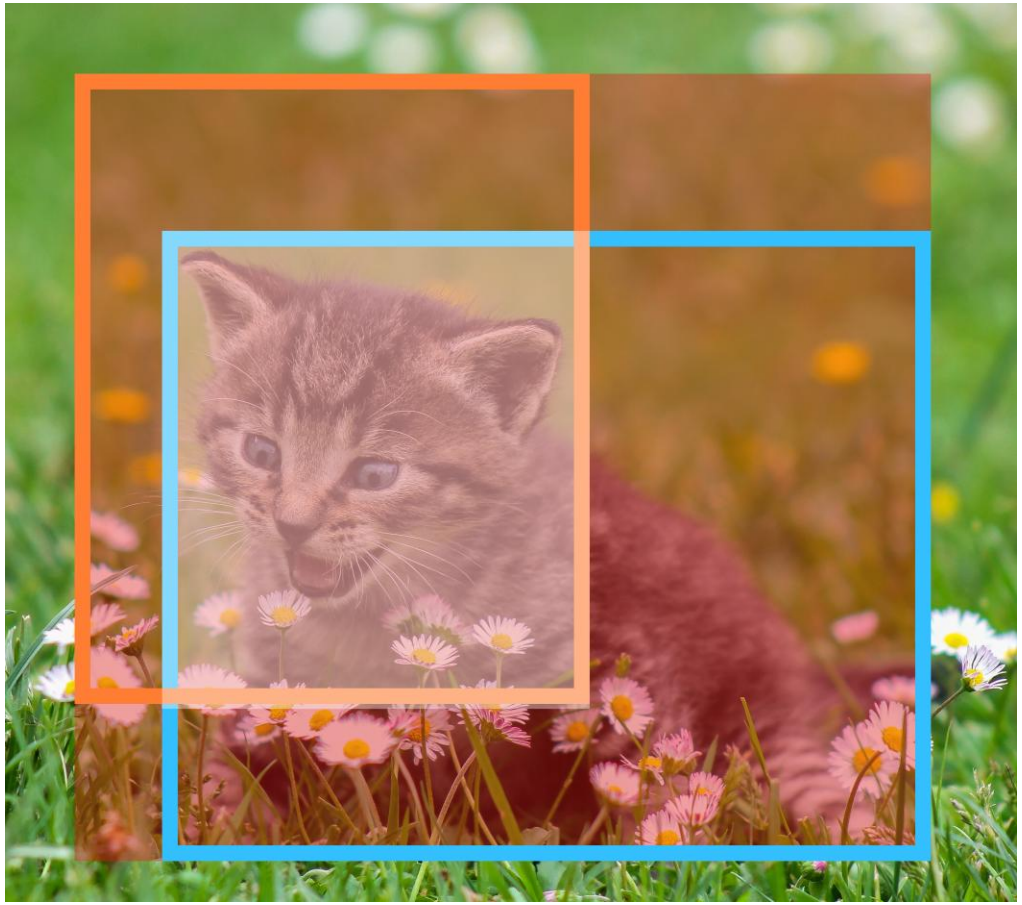


Image Source: A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons

Overlapping boxes

Object detectors output multiple candidate bounding boxes, and some of them may overlap.

How can we filter out the boxes that belongs to the same object?

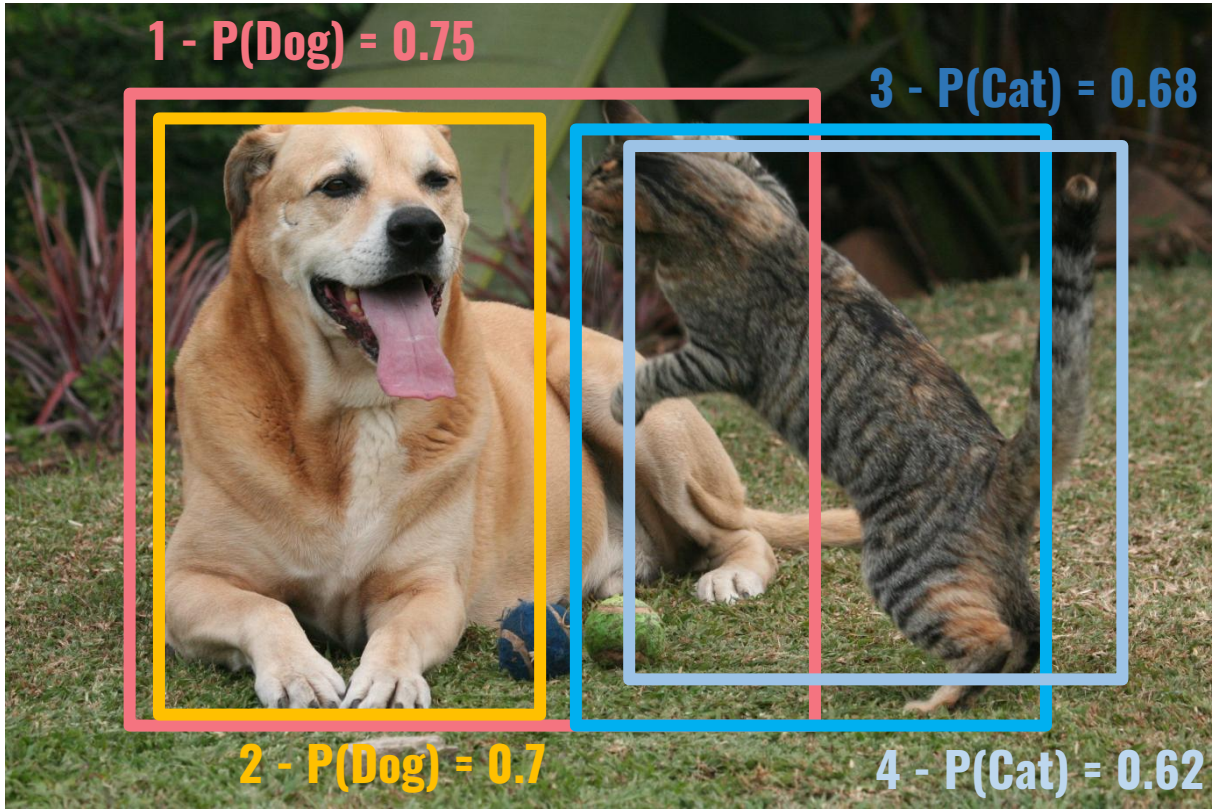


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Overlapping boxes

Object detectors output multiple candidate bounding boxes, and some of them may overlap.

How can we filter out the boxes that belongs to the same object?

Solution: Post process detections using **Non-Max Suppression (NMS)**

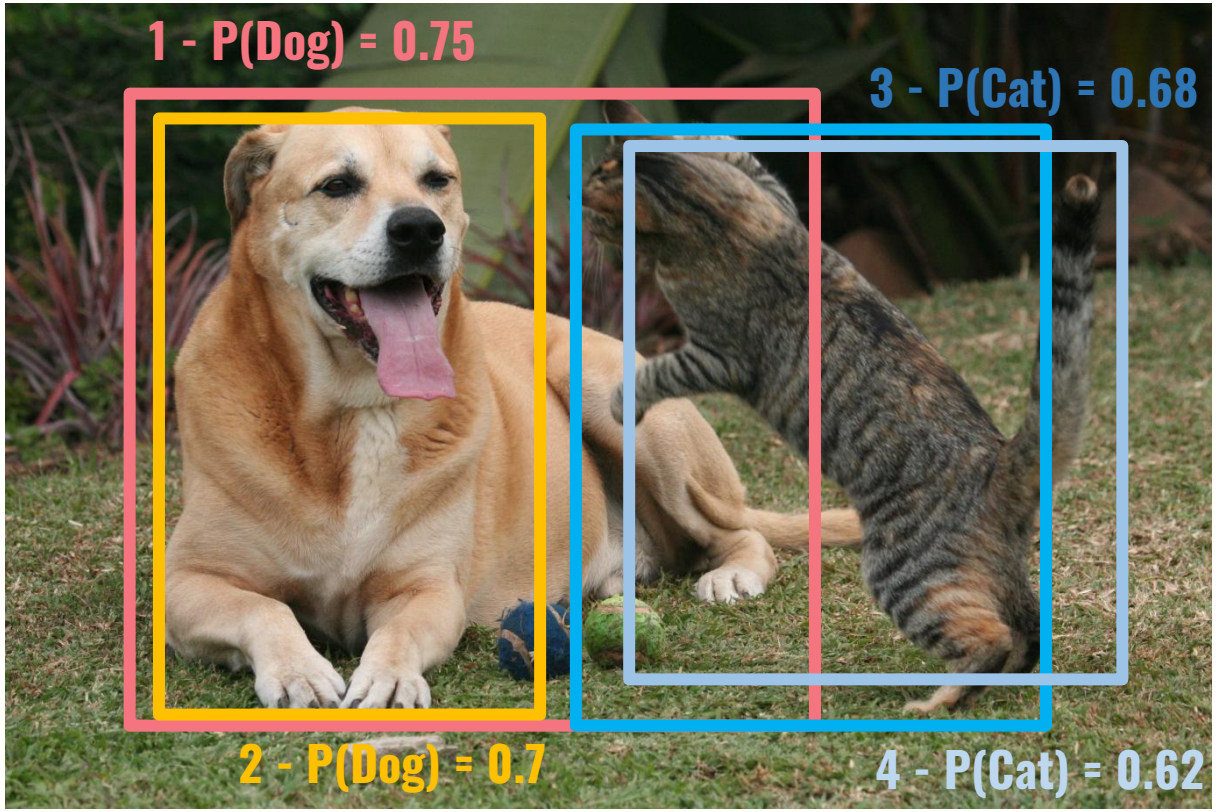


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Overlapping boxes

Non-Maximum Suppression (NMS) removes duplicate detections:

1. Select the bounding box with the highest confidence score.
2. Remove other boxes that overlap it above an IoU threshold (i.e. 0.5).
3. Repeat with the remaining boxes.

The result keeps the strongest box for each detected object.

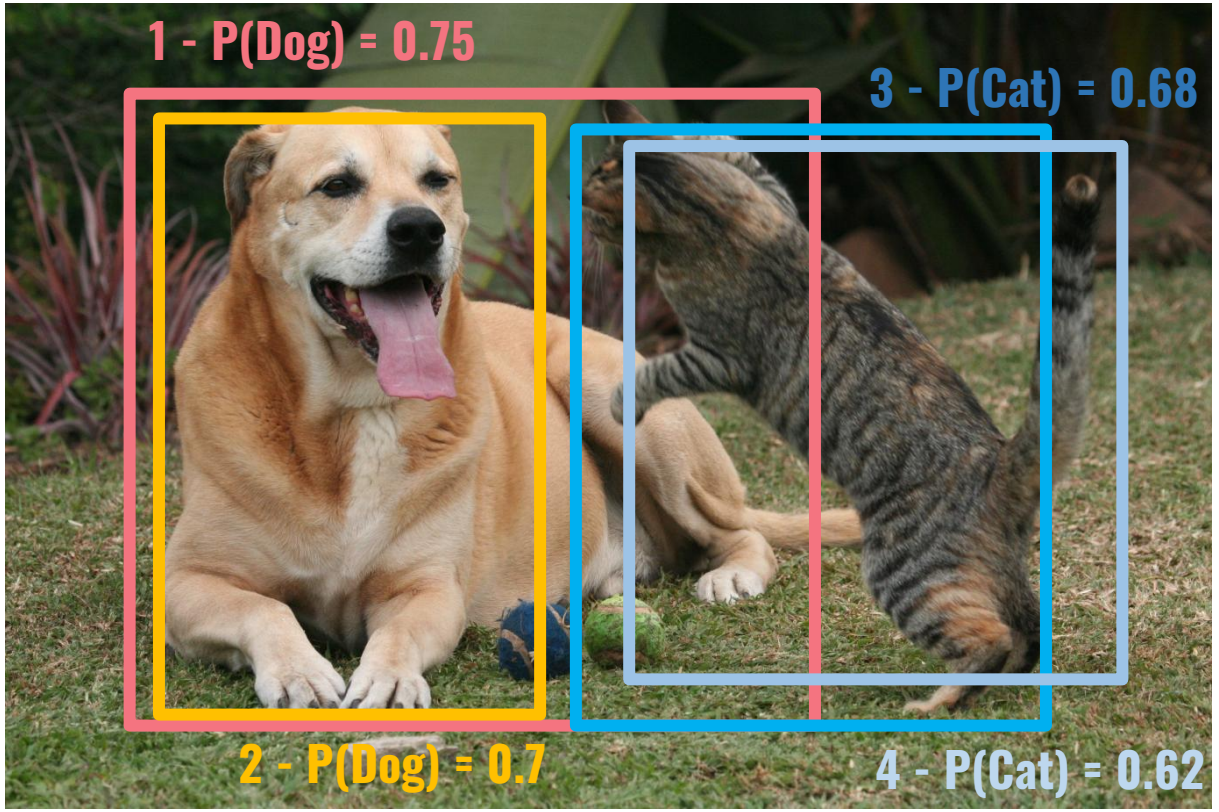


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Overlapping boxes

Taking the highest confidence score box (1) and comparing to the other boxes:

IoU (1, 2) = 0.53
IoU (1, 3) = 0.26
IoU (1, 4) = 0.17
IoU (2, 3) = 0.0
IoU (2, 4) = 0.0
IoU (3, 4) = 0.69

With a threshold of 0.5 we eliminate box 2.

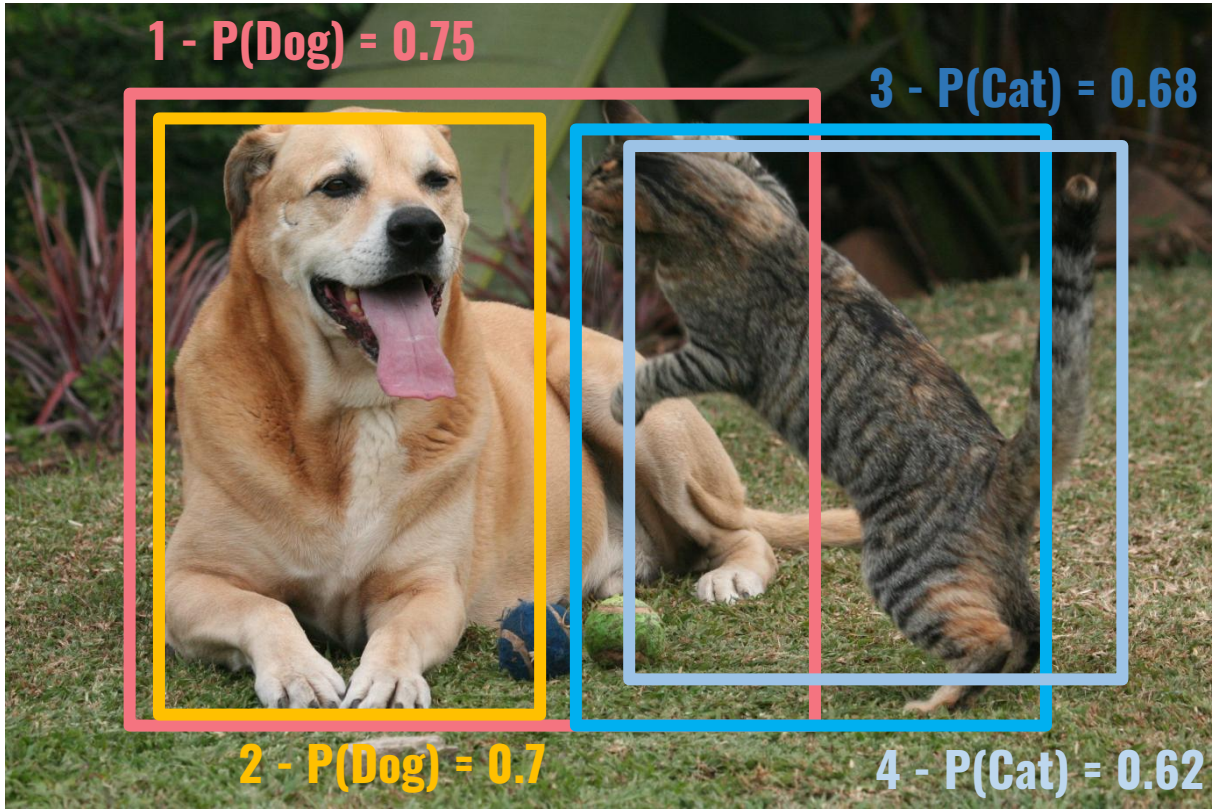


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Overlapping boxes

Taking the highest confidence score box (3) and comparing to the other boxes:

~~IoU (1, 2) = 0.53~~
IoU (1, 3) = 0.26
IoU (1, 4) = 0.17
~~IoU (2, 3) = 0.0~~
~~IoU (2, 4) = 0.0~~
IoU (3, 4) = 0.69

With a threshold of 0.5 we eliminate box 4.

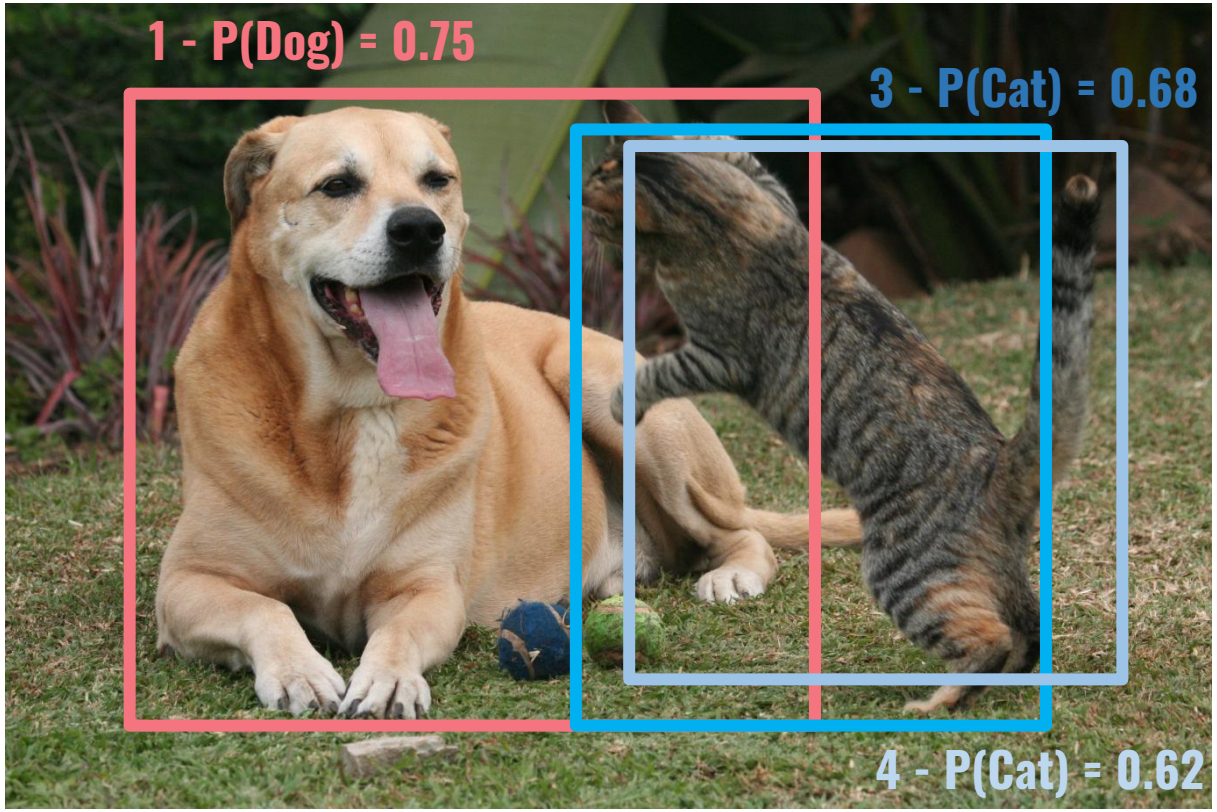


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Overlapping boxes

No more boxes left to eliminate.

~~IoU (1, 2) = 0.53~~
IoU (1, 3) = 0.26
~~IoU (1, 4) = 0.17~~
~~IoU (2, 3) = 0.0~~
~~IoU (2, 4) = 0.0~~
~~IoU (3, 4) = 0.69~~

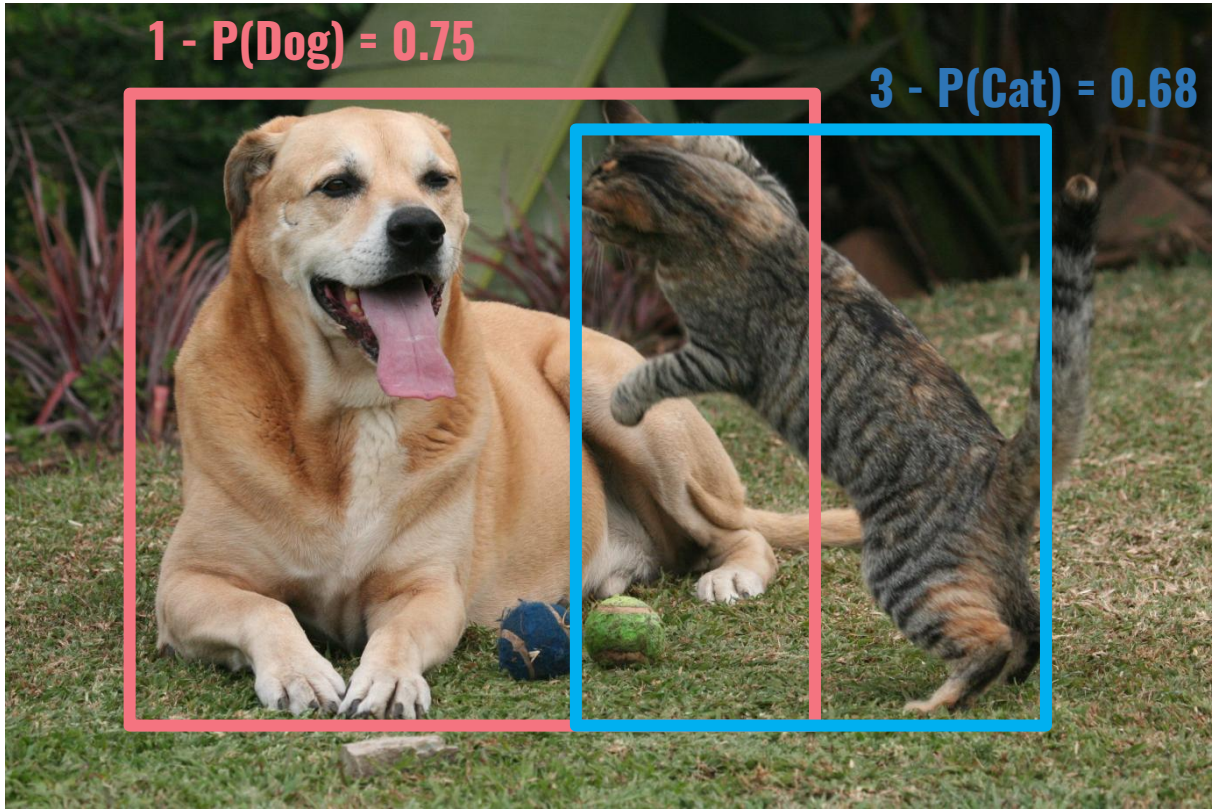


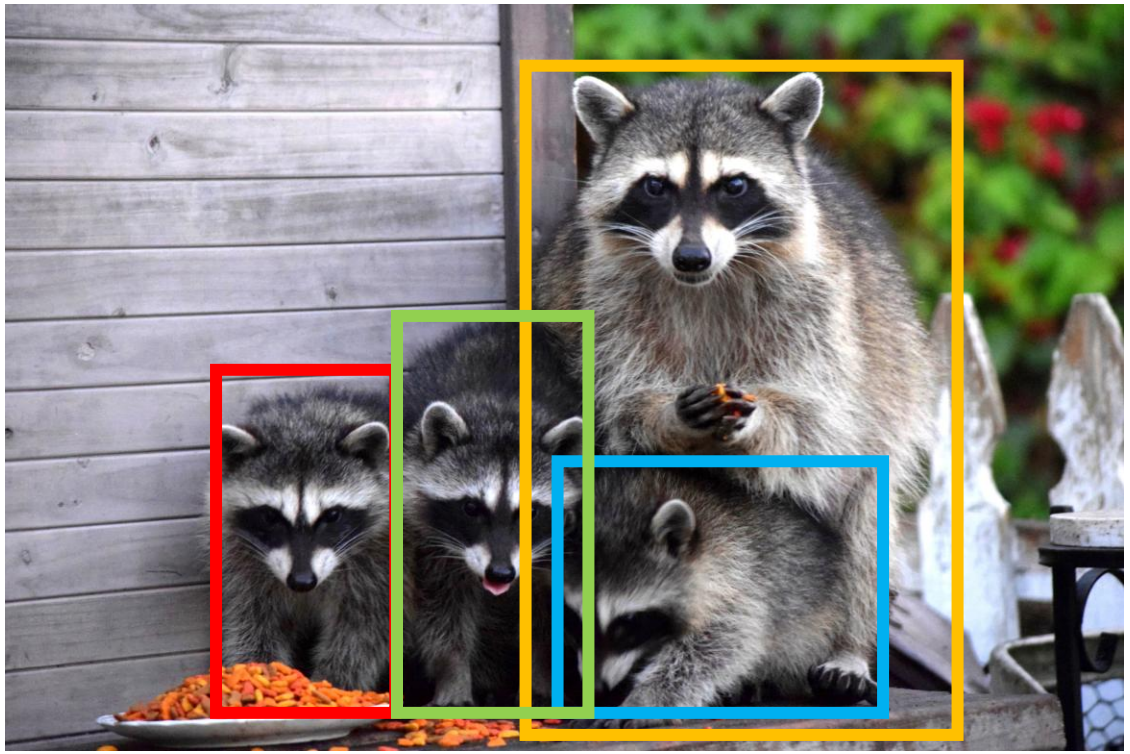
Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Overlapping boxes

Non-Maximum Suppression (NMS) removes duplicate detections:

1. Select the bounding box with the highest confidence score.
2. Remove other boxes that overlap it above an IoU threshold (i.e. 0.5).
3. Repeat with the remaining boxes.

The result keeps the strongest box for each detected object.



Problem: NMS may eliminate good boxes if the image is crowded and the object overlap

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

NMS Limitations

There are a few other methods that try to improve detections:

- **Soft-NMS:** Traditional NMS is a binary operation that discards all but the highest-scoring bounding box. Soft-NMS, on the other hand, assigns lower scores to overlapping boxes rather than completely removing them. This results in smoother score degradation for close-by objects, reducing the chance of removing valid detections.

NMS Limitations

There are a few other methods that try to improve detections:

- **IoU Threshold Adaptation:** Instead of using a fixed IoU (Intersection over Union) threshold for NMS, you can dynamically adjust the threshold based on the object's characteristics. For instance, you might use a higher IoU threshold for large objects and a lower IoU threshold for smaller objects.

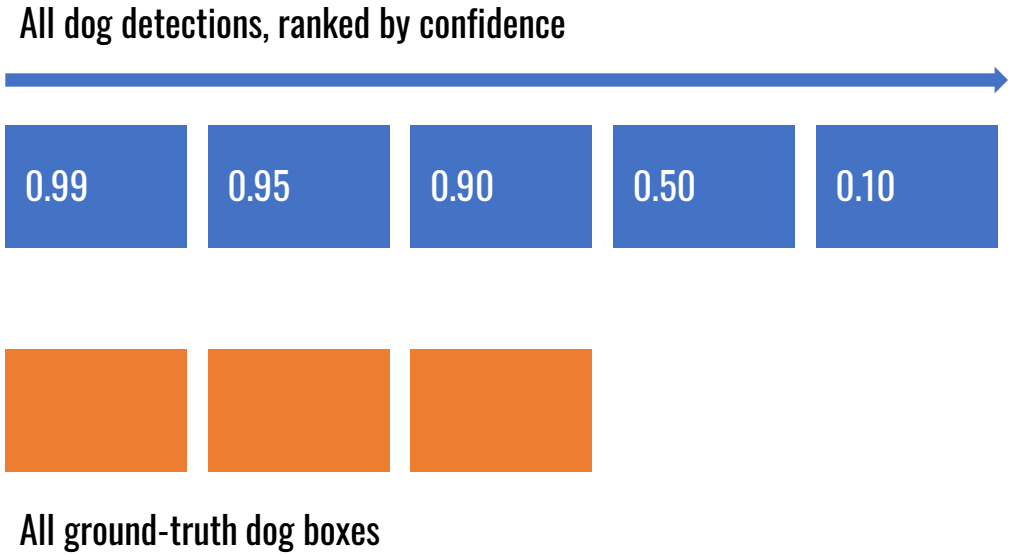
NMS Limitations

More methods:

- Selection of object detections using overlap map predictions (https://link.springer.com/article/10.1007/s00521-022-07469-x)
- https://www.sciencedirect.com/science/article/pii/S2214914721001914
- https://arxiv.org/pdf/2207.00865.pdf
- And more…
- Open research field (make your contribution)

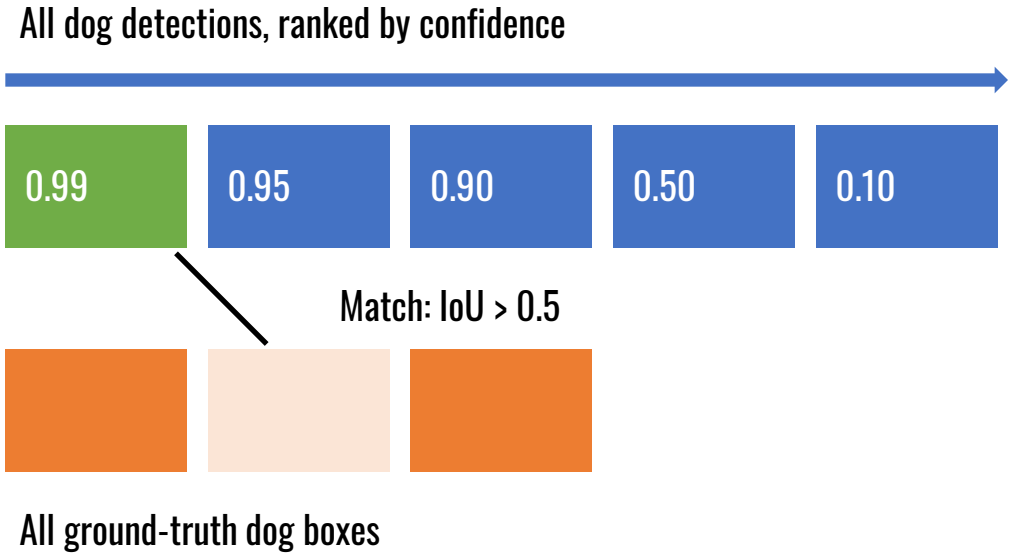
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS to the raw outputs.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.



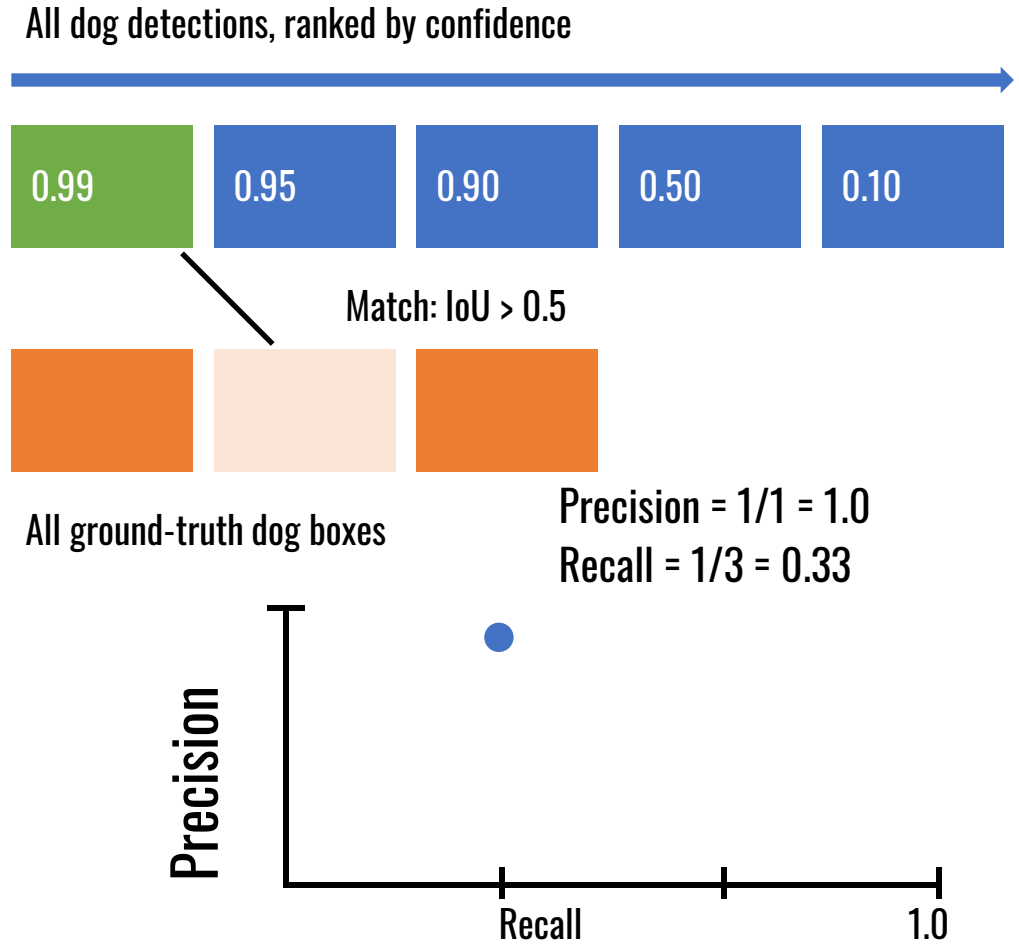
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.



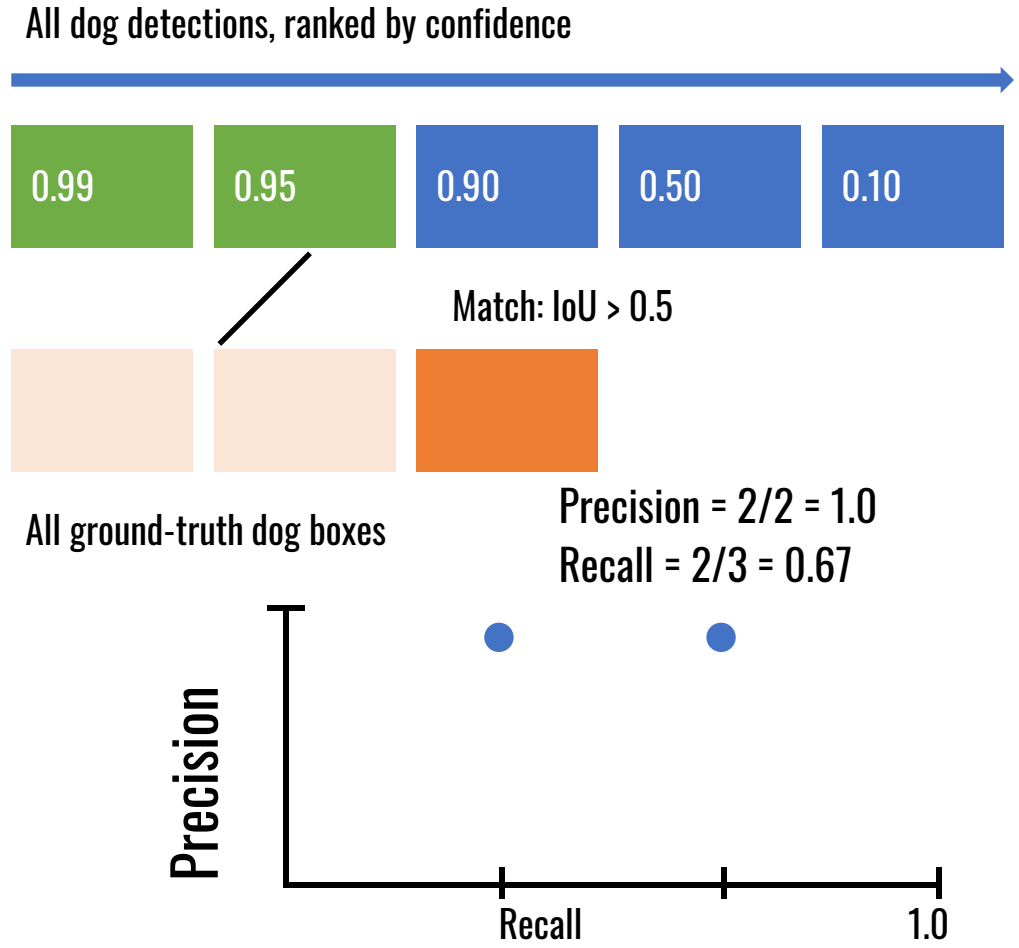
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.



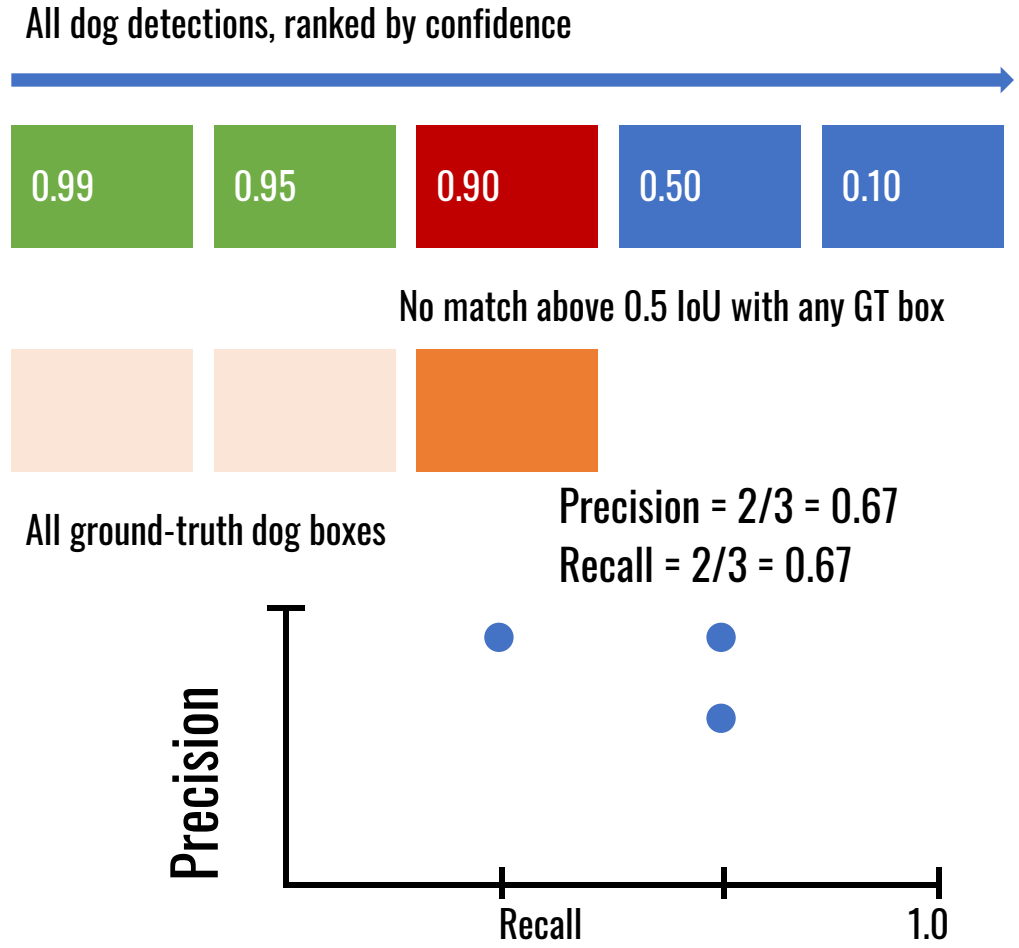
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.



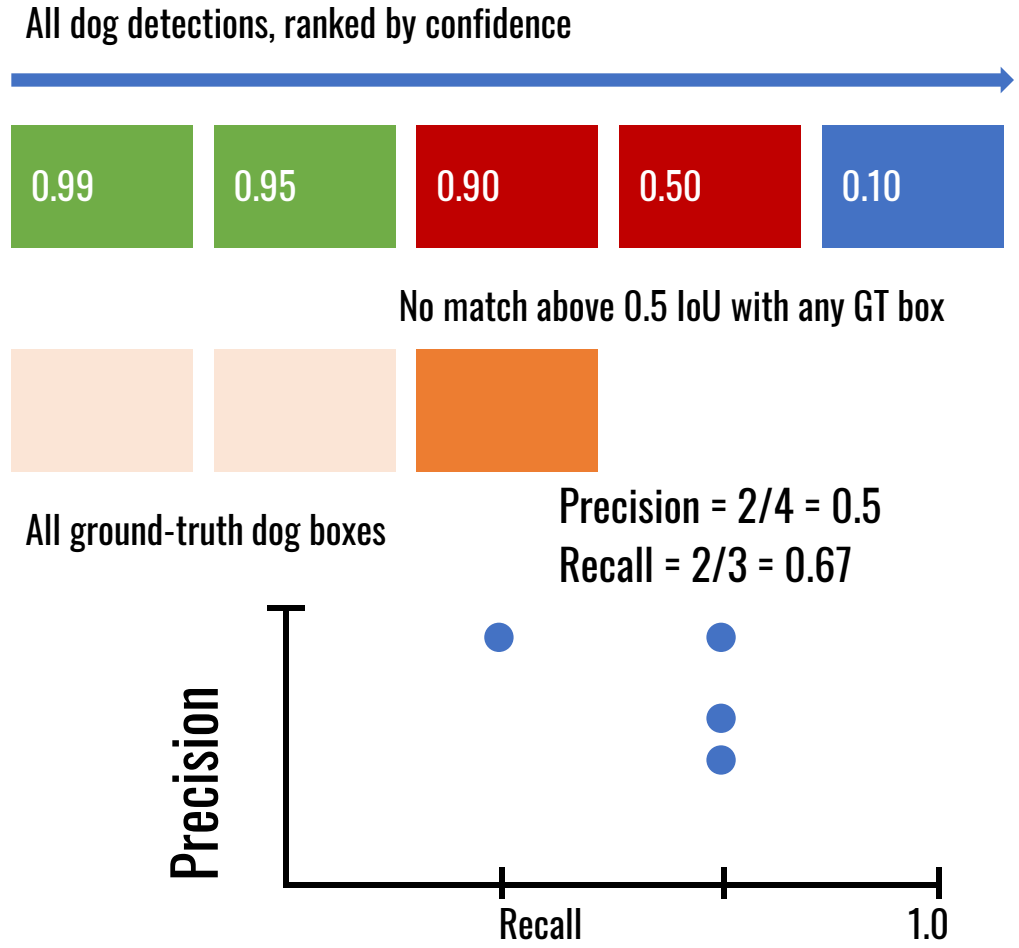
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.



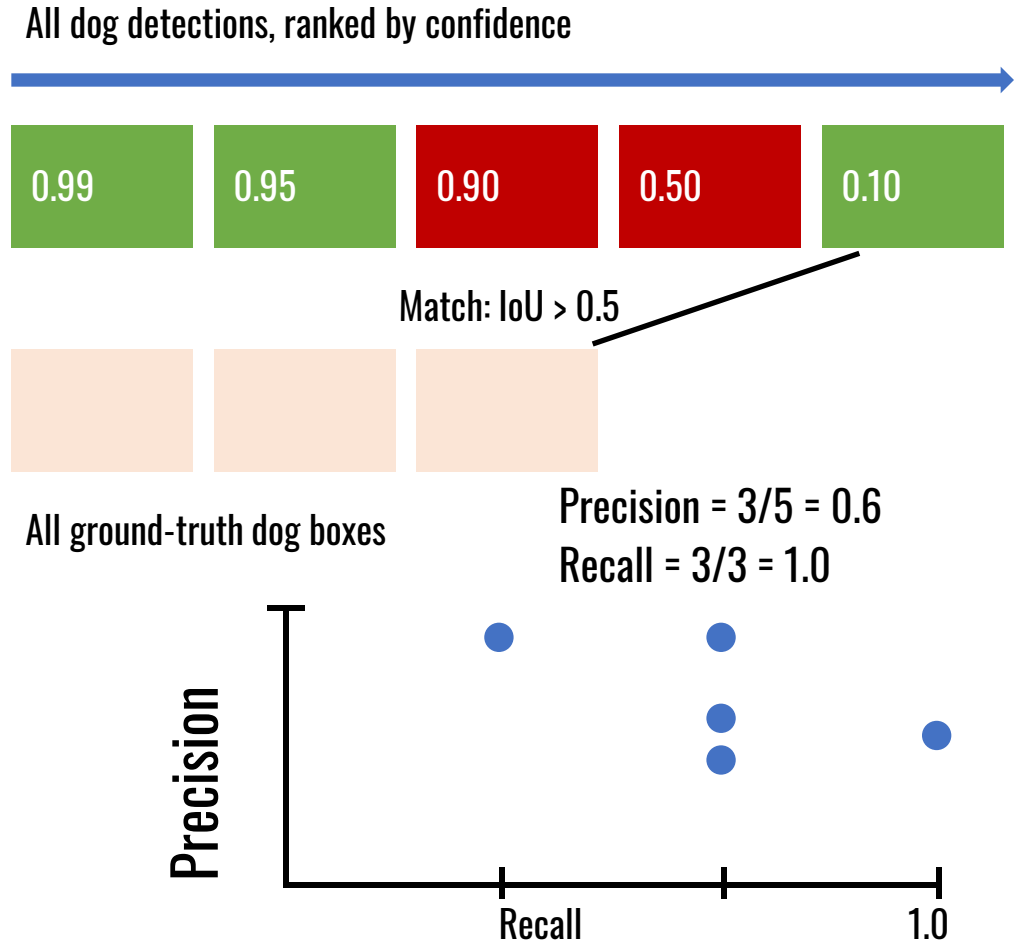
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.



Evaluating Object Detectors: Mean Average Precision (mAP)

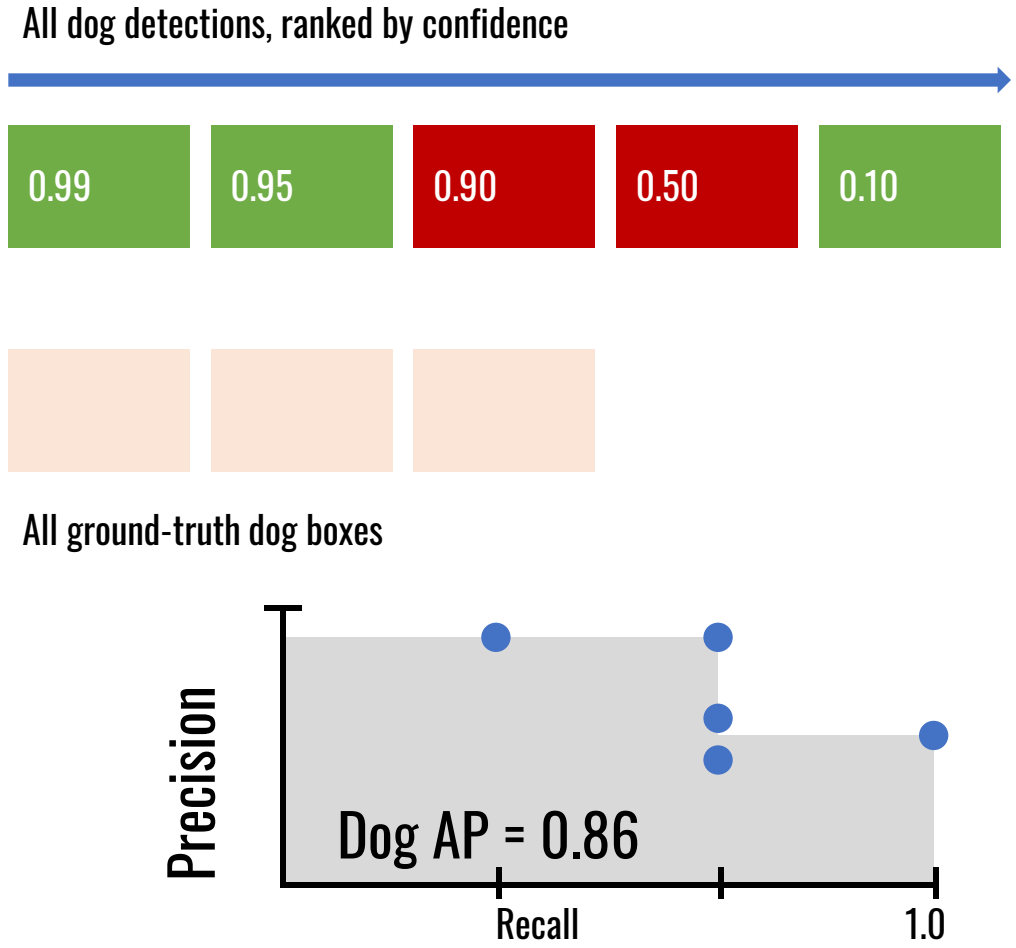
1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.



Example: Justin, J. (2022). EECS 498: Deep Learning for Computer Vision (Fall 2019). University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.
    2. AP is the area beneath the finished PR curve.



Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.
    2. AP is the area beneath the finished PR curve.
3. Mean Average Precision (mAP) is the average of the per-class AP values.

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run the trained detector over every test image and apply NMS.
2. For each class, measure Average Precision (AP) as the area under its Precision-Recall curve:
    1. Walk through the detections from highest to lowest confidence score.
        1. If it overlaps a ground-truth box with IoU > 0.5, count it as a positive and remove that GT box.
        2. Otherwise, count it as a negative.
        3. Add the resulting point to the PR curve.
    2. AP is the area beneath the finished PR curve.
3. Mean Average Precision (mAP) is the average of the per-class AP values.
4. COCO mAP: repeat the whole calculation at IoU thresholds 0.5, 0.55, 0.6, … 0.95 and average those results.

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

…

mAP@0.95 = 0.20

COCO mAP = 0.40

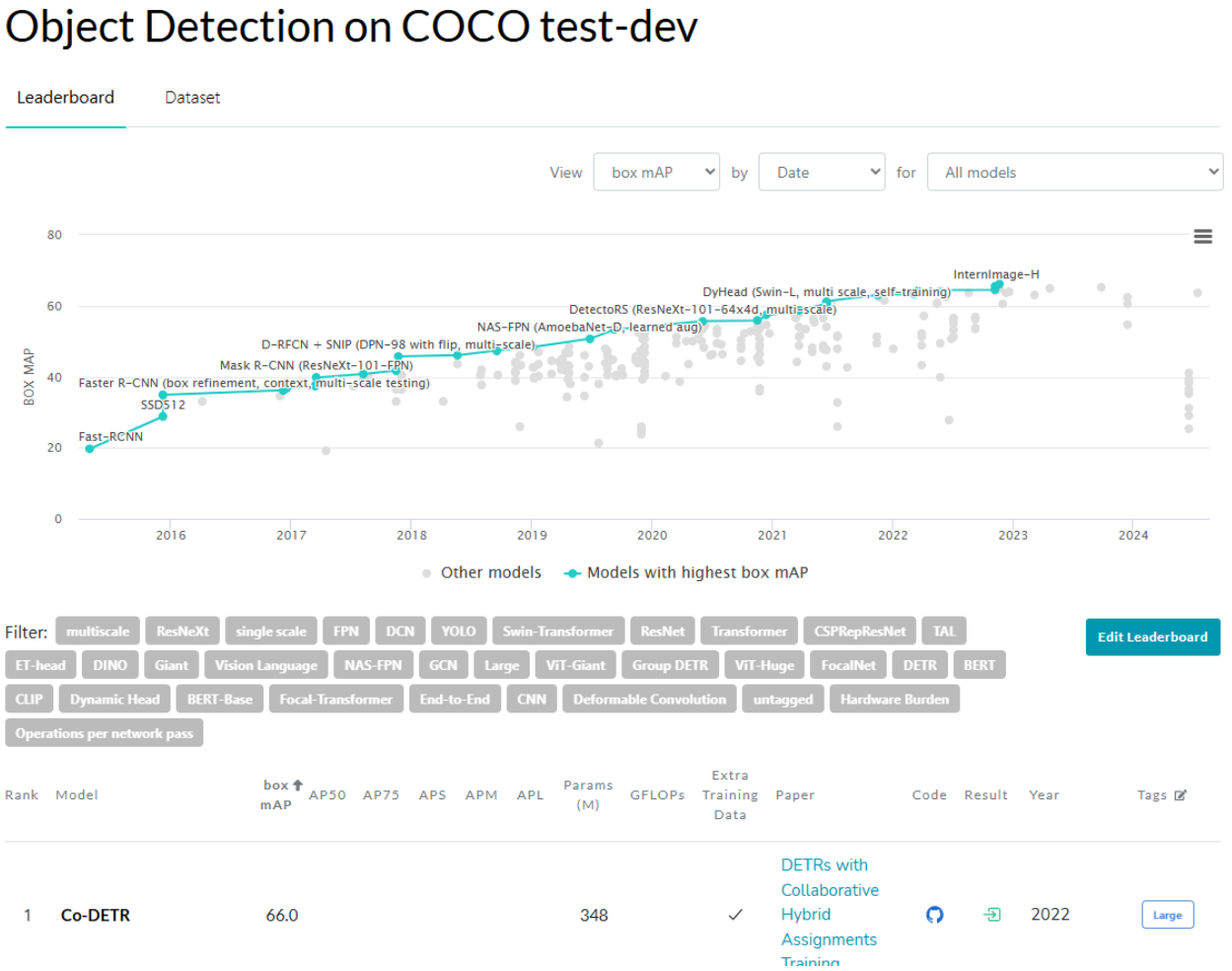
Understanding object detection paper results

Hardware independent metrics:

- **mAP (Mean Average Precision):** Overall average precision across classes and IoU thresholds, indicating detection quality.
- **AP50:** Average precision at an IoU threshold of 0.50, a lenient measure of detection accuracy.
- **AP75:** Average precision at an IoU threshold of 0.75, a stricter measure of accuracy.
- **APS/APM/APL:** Average precision for small, medium, and large objects, assessing model performance across object sizes.
- **Params (M):** Number of model parameters, in millions, indicating model size.
- **GFLOPs:** Computational complexity, showing the number of floating-point operations needed for a forward pass.

Hardware dependent metrics:

- Latency, Inference speed



COCO test-dev Benchmark (Object Detection) | Papers With Code

Understanding object detection paper results

- **mAP (Mean Average Precision):** This is the average precision across different classes and IoU (Intersection over Union) thresholds. It's a comprehensive metric for measuring the performance of object detection models. The higher the mAP, the better the model's precision and recall performance over all categories and thresholds.
- **AP50 (Average Precision at IoU=0.50):** This metric evaluates the average precision when the IoU threshold is set to 0.50. This means that, for a detection to be considered correct, the predicted bounding box must overlap the ground truth by at least 50%. It is often considered a relatively lenient metric.
- **AP75 (Average Precision at IoU=0.75):** Like AP50.
- **APS (Average Precision for Small Objects):** This metric calculates the average precision for detecting small objects. Smaller objects can be more difficult to detect accurately, so this metric specifically tracks how well the model handles smaller object sizes.
- **APM/APL:** Average precision for medium and large objects, assessing model performance across object sizes.
- **Params (M) (Model Parameters in Millions):** This metric shows the number of parameters in the model, typically in millions (M). More parameters often mean a larger, more complex model, but not necessarily better performance.
- **GFLOPs (Giga Floating Point Operations):** This refers to the number of floating-point operations required to make a single forward pass through the model, usually measured in gigaflops (GFLOPs). It is an indicator of the model's computational complexity and can be used to gauge the efficiency of the model.

Supervised Learning tasks

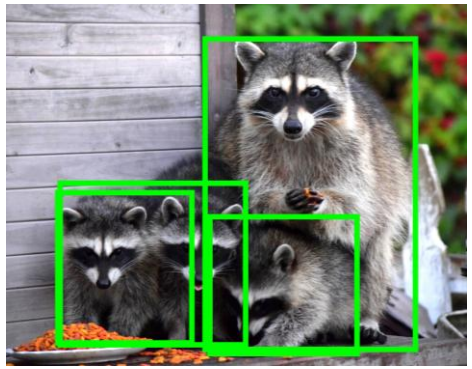
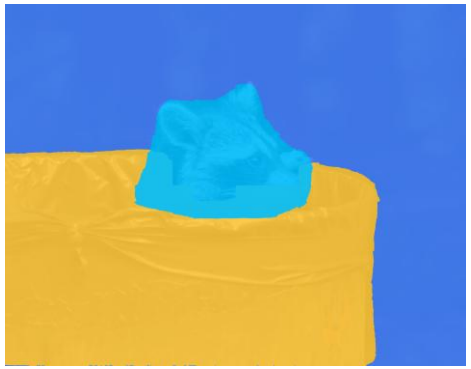
Classification	Classification + Localization	Object Detection	Instance Segmentation	Semantic Segmentation
				
RACCOON	RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, BIN, BACKGROUND
Single Object	Single Object	Multiple Objects	Multiple Objects	No objects, just pixels
✓	✓	✓	⧗	⧗

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Object Detection

Inventory Management:
- https://www.youtube.com/watch?v=he5I6ByoaB4

Multi Camera Object Tracking:
- https://www.youtube.com/watch?v=oBL84CeLwKk

Surveillance:
- https://www.youtube.com/watch?v=9Xg-7FfLIVw

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 5

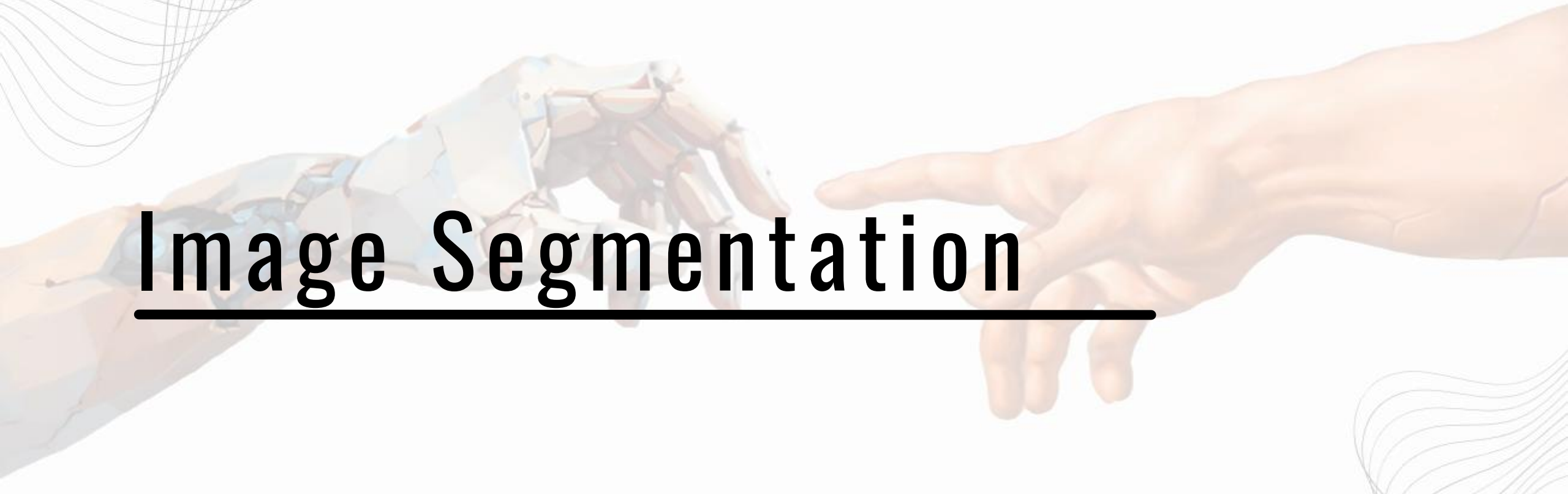


Image Segmentation

1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

Supervised Learning tasks

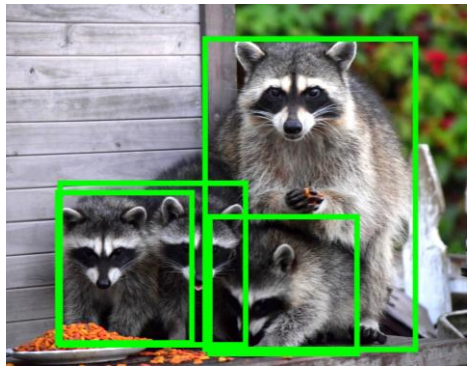
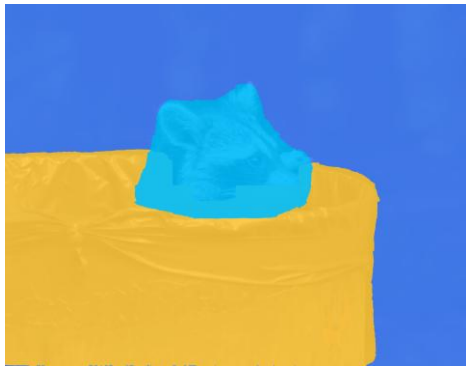
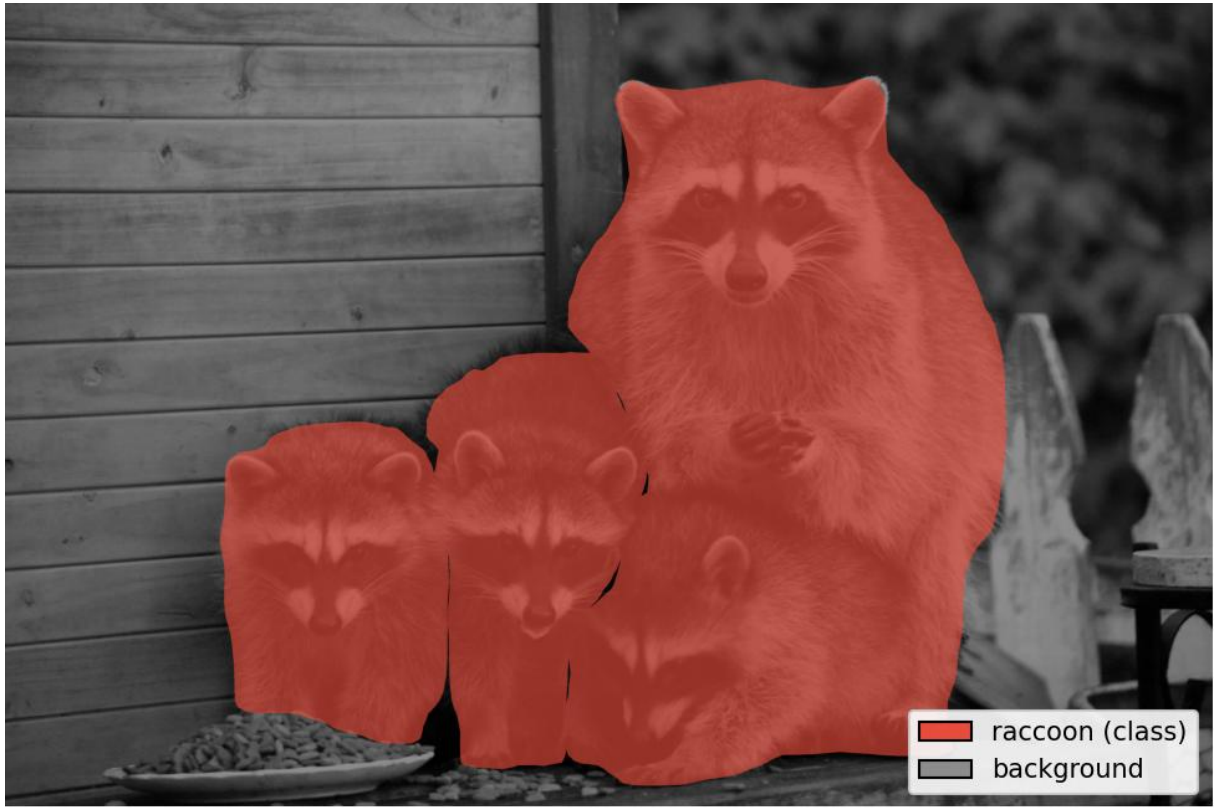
Classification	Classification + Localization	Object Detection	Instance Segmentation	Semantic Segmentation
				
RACCOON	RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, BIN, BACKGROUND
Single Object	Single Object	Multiple Objects	Multiple Objects	No objects, just pixels
✓	✓	✓	⧗	⧗

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Image Segmentation

Semantic Segmentation

Instance Segmentation



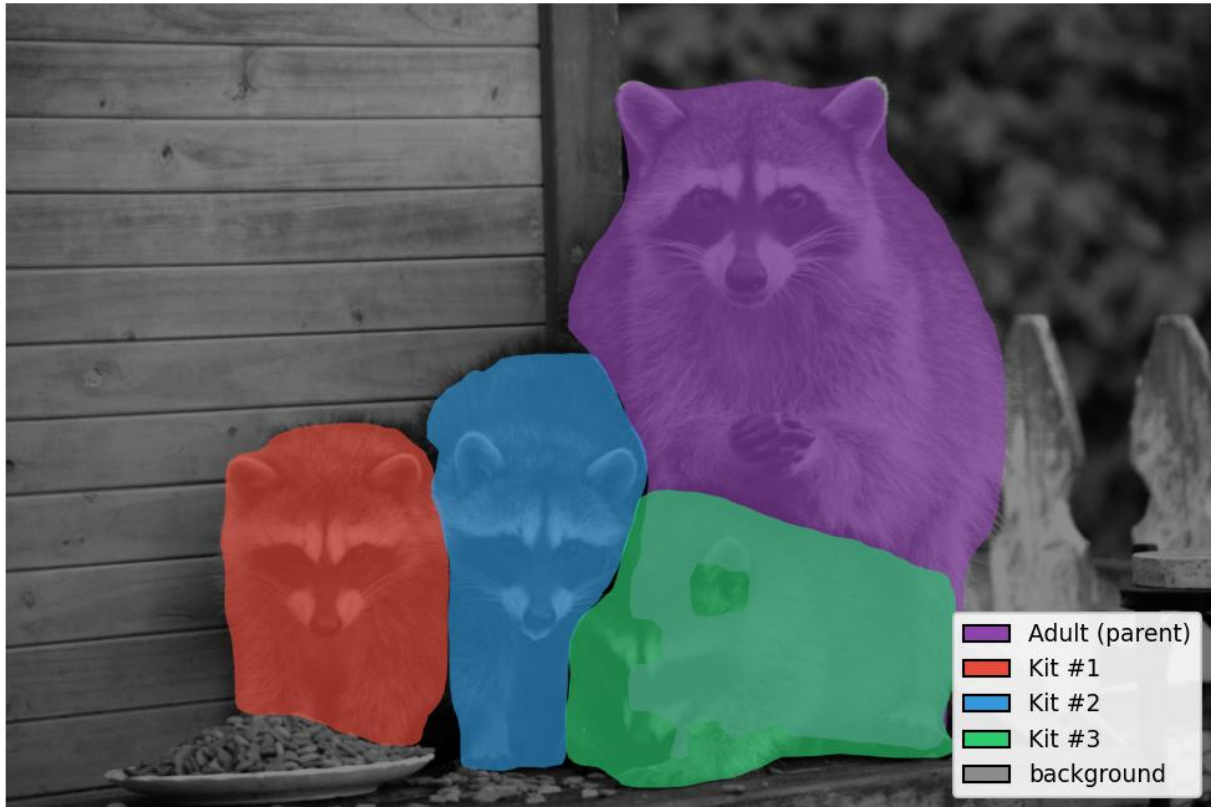


Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Video Segmentation

Autonomous Driving example - NVIDIA DRIVE (2024) - https://www.youtube.com/watch?v=nBjXyoltCHU

Medical Image Diagnosis:

- https://github.com/mateuszbuda/brain-segmentation-pytorch
- https://github.com/Saswatm123/3D-Brain-Tumor-Segmentation-PyTorch
- https://github.com/hahnicity/pytorch-lung-segmentation

Virtual Try On:

- https://github.com/kishorkuttan/Deep-Virtual-Try-On
- https://github.com/shadow2496/VITON-HD
- https://github.com/JDAI-CV/Down-to-the-Last-Detail-Virtual-Try-on-with-Detail-Carving

Classification Task

We map images (x) to labels (y)

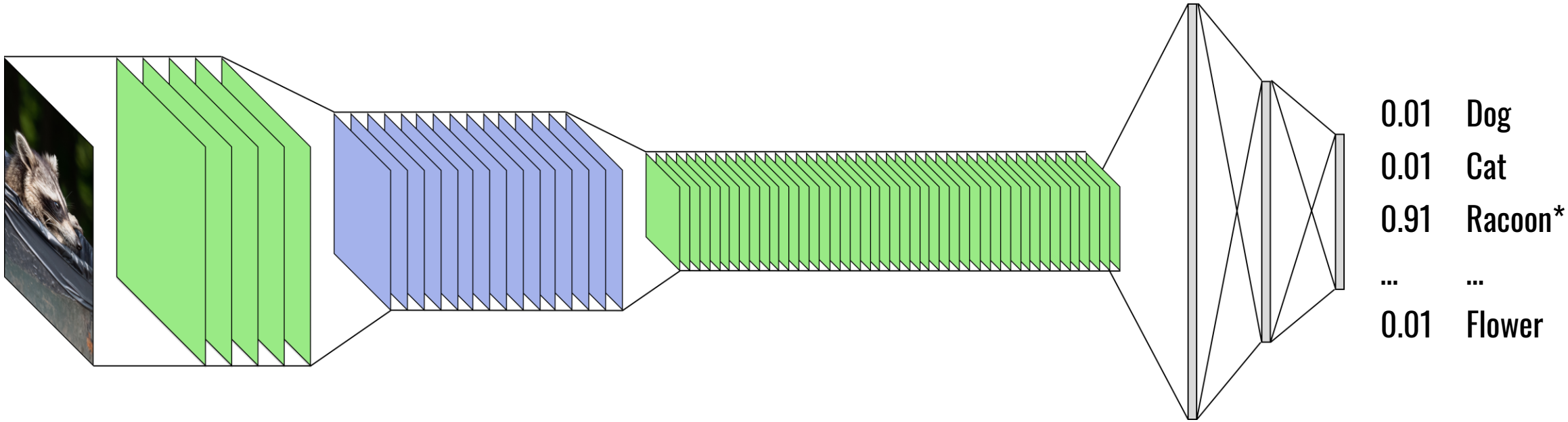


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Reconstruction task

We try to get back the original image while constraining the network to only learn meaningful information

- Can be used for denoising images
- We lose information during the constraining
- How can we up sample from the latent representation?

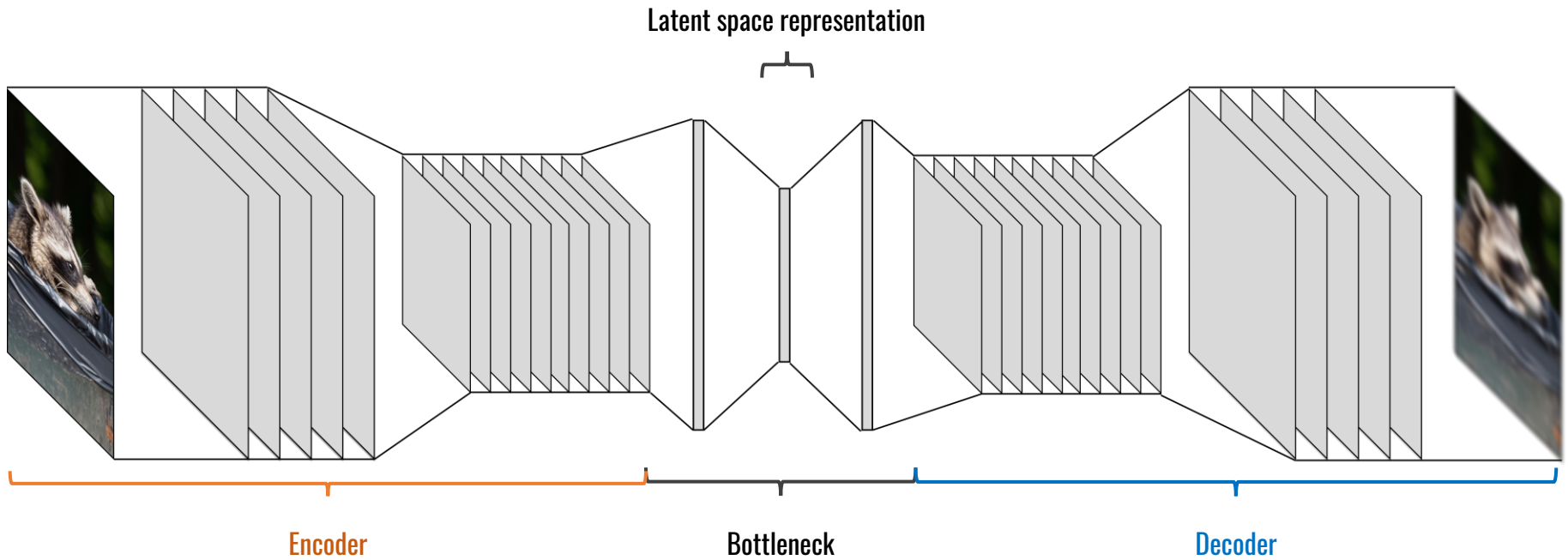
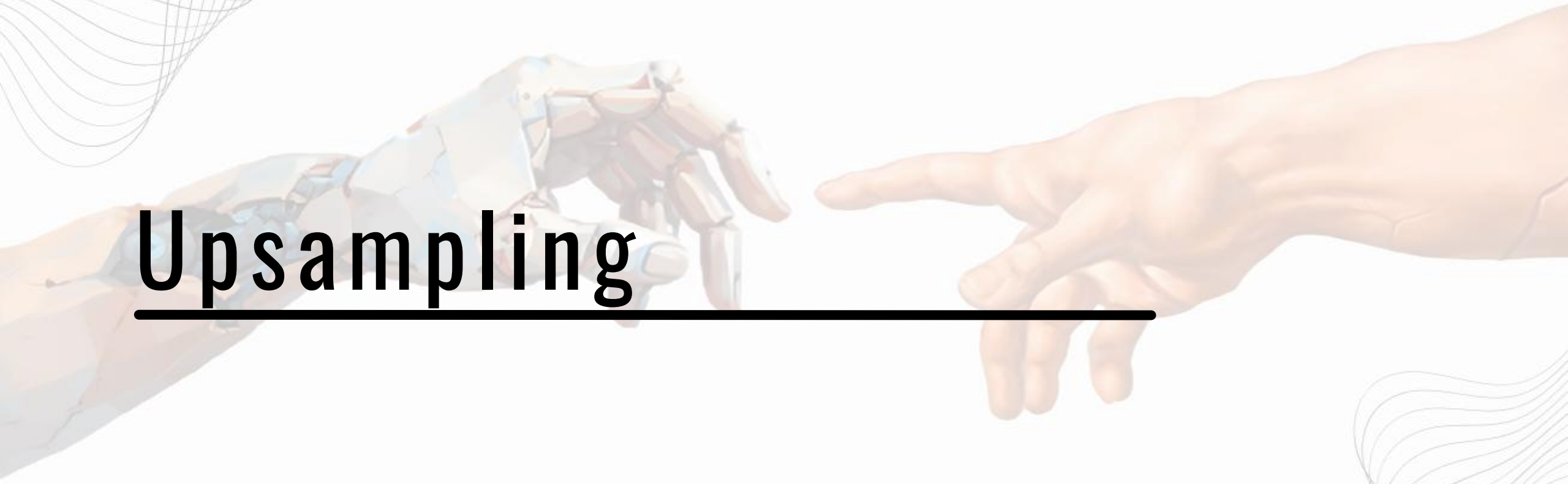


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Deep Network Development

Lecture 5



Upsampling

1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

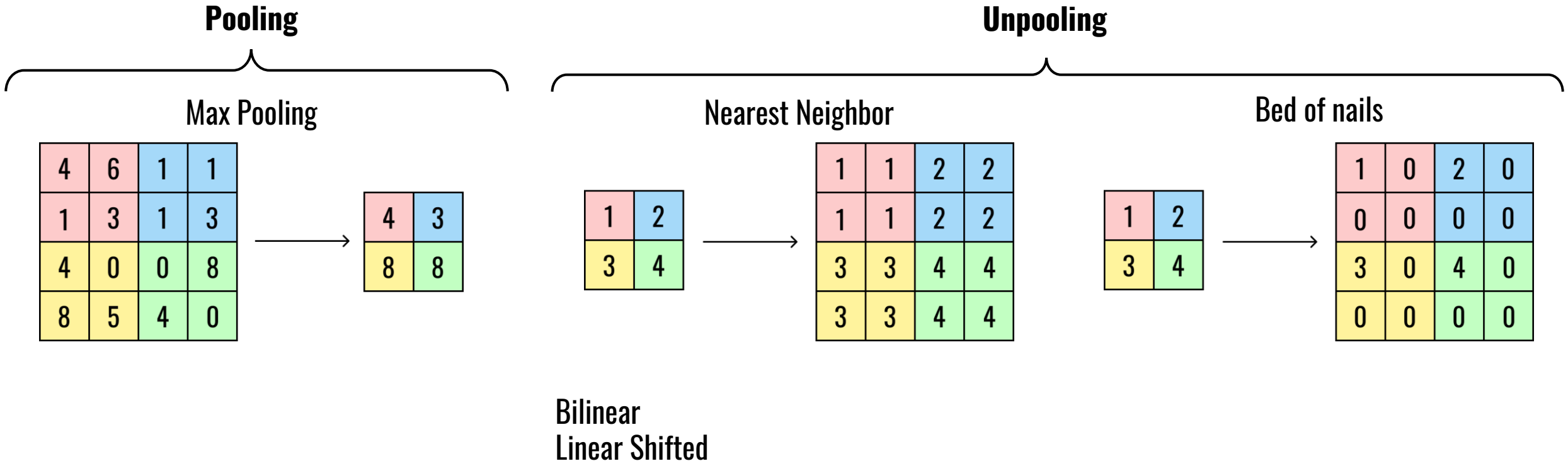
Upsampling

How to upsample?

1. Unpooling
2. Transposed Convolution

Unpooling

- Whereas pooling operations downsample the resolution by summarizing a local area with a single value (ie. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.



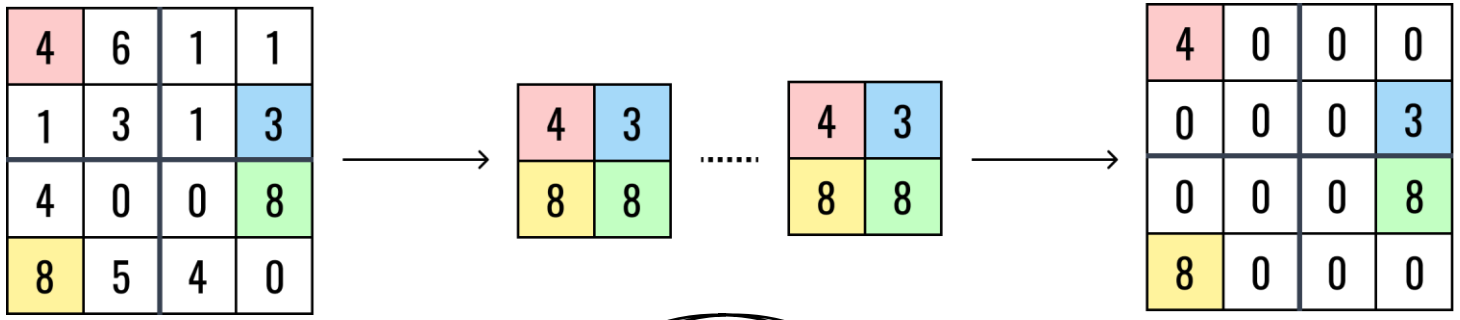
Unpooling

- Whereas pooling operations downsample the resolution by summarizing a local area with a single value (I.e. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.
- **No weights, nothing to learn here!**

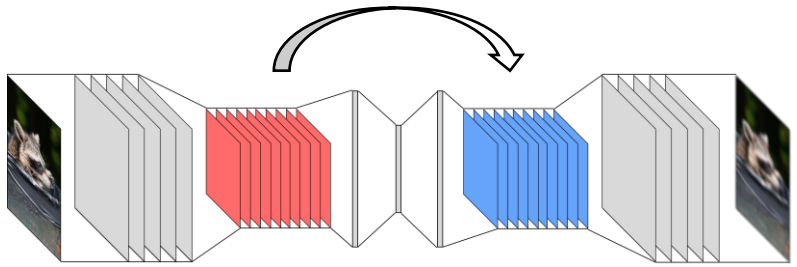
In-Network upsampling: “Max Unpooling”

Remember the position where we pooled from

Use the position to put the value back during unpooling

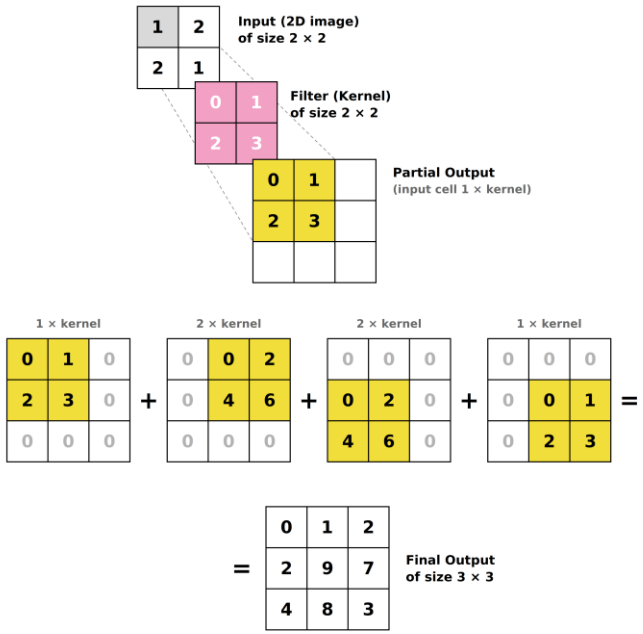


Corresponding pairs of down sampling and up sampling



Transposed Convolution

- Most popular approach
- Whereas a typical convolution operation will take the dot product of the values currently in the filter's view and produce a single value for the corresponding output position, a **transpose convolution** essentially does the opposite. For a transpose convolution, we take a single value from the low-resolution feature map and multiply all the weights in our filter by this value, projecting those weighted values into the output feature map.



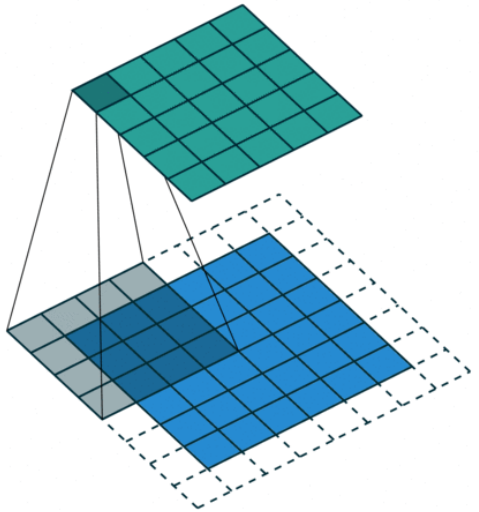
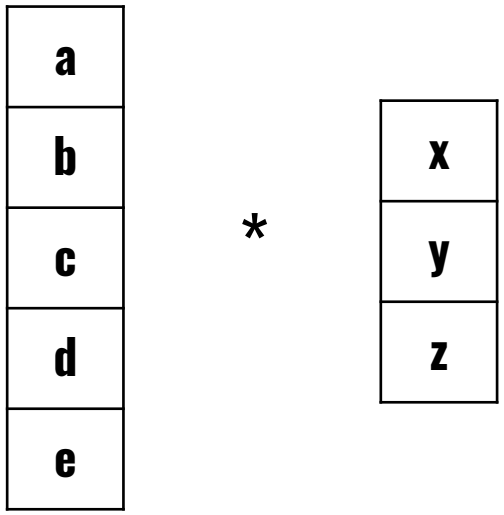


Image Source: Convolution arithmetic visualization, adapted from Vincent Dumoulin, & Francesco Visin. (2018). A guide to convolution arithmetic for deep learning.; original visual assets from the conv_arithmetic GitHub repository, MIT License.

Normal Convolution

- 1D example

Input size: 5 **Filter size:** 3
Stride: 2

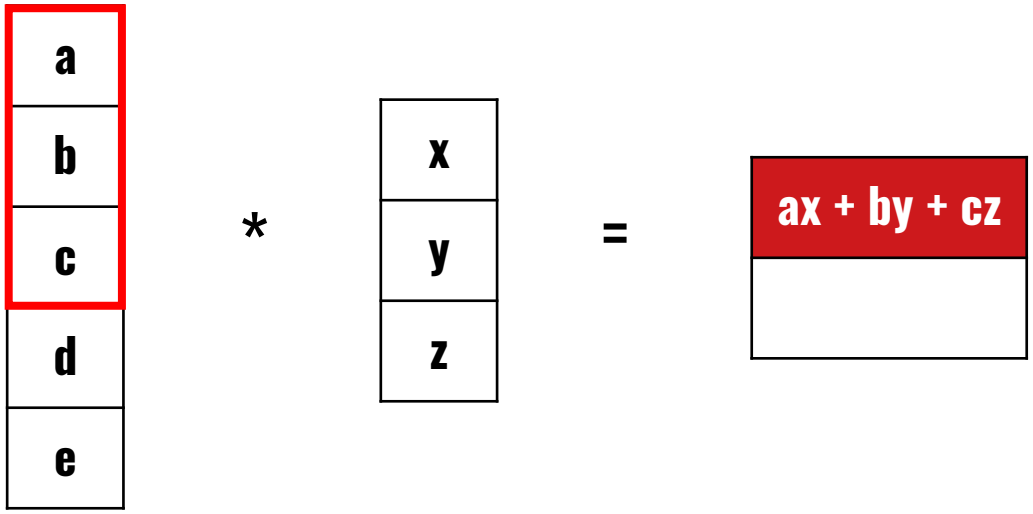


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Normal Convolution

- 1D example

Input size: 5 **Filter size:** 3 **Output size:** 2
Stride: 2

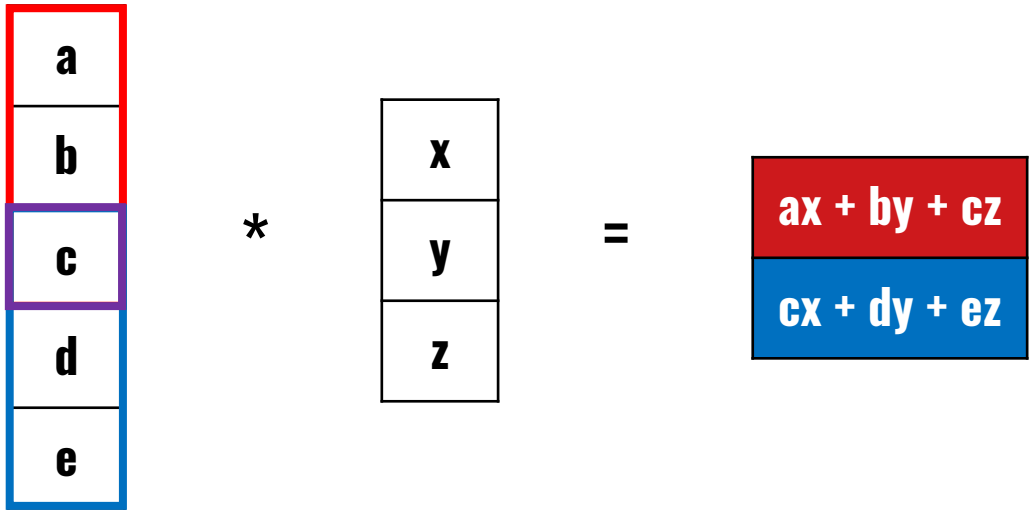


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Normal Convolution

- 1D example

Input size: 5 **Filter size:** 3 **Output size:** 2
Stride: 2



$n \times n$ *image* $f \times f$ *filter*
padding p *stride* s

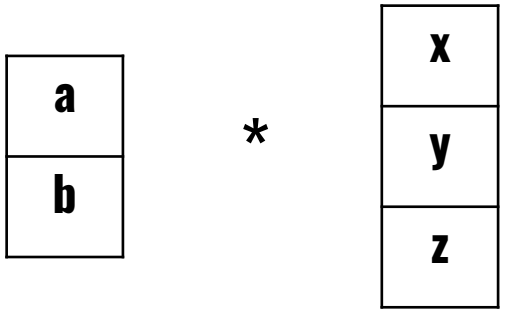
$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example

Input size: 2 **Filter size:** 3
Stride: 2

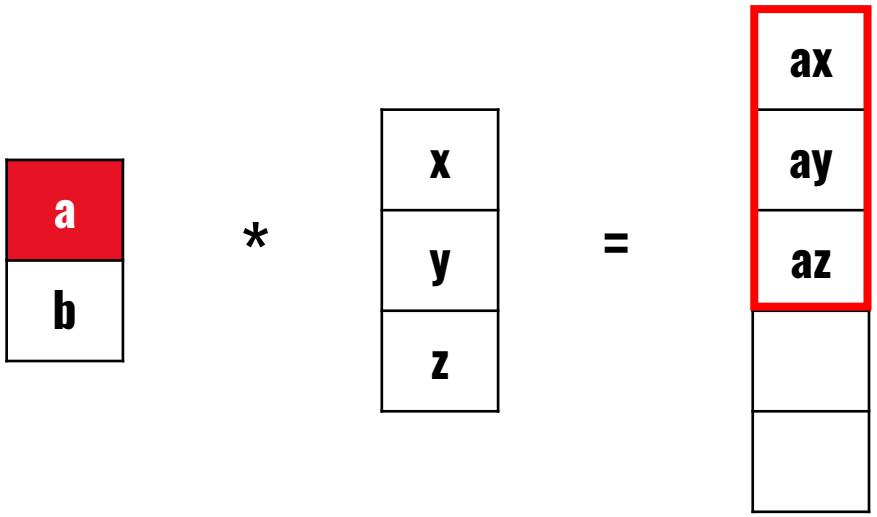


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example

Input size: 2 **Filter size:** 3 **Output size:** 5
Stride: 2

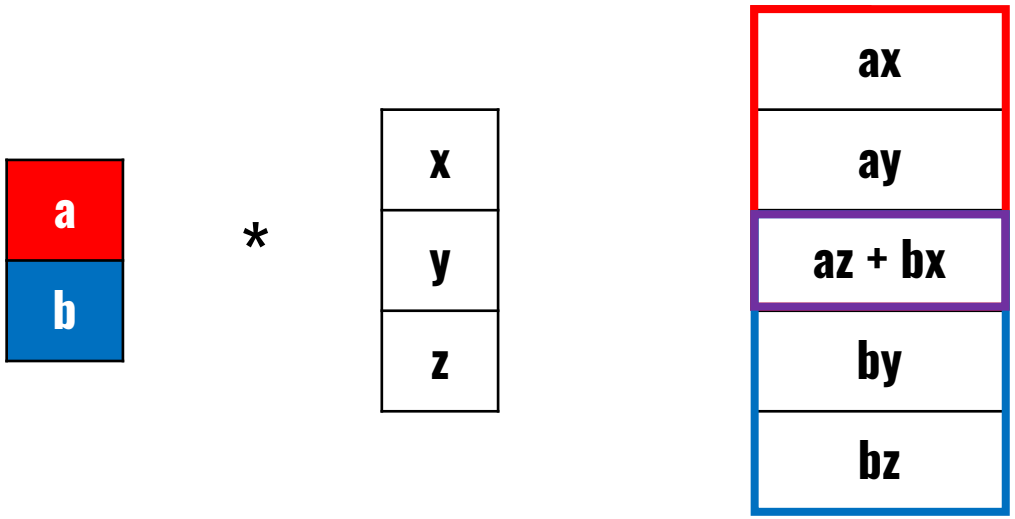


Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example

Input size: 2 **Filter size:** 3 **Stride:** 2 **Output size:** 5

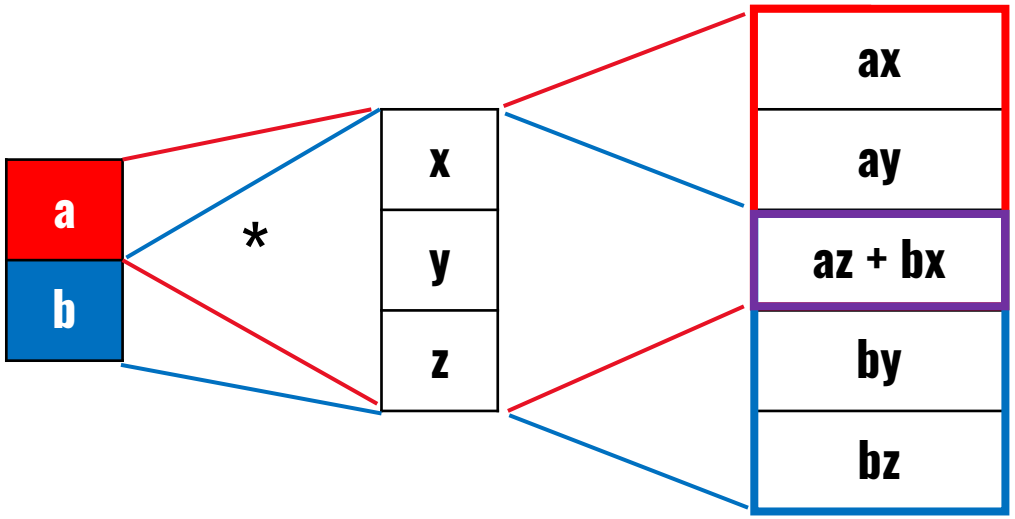


$$\textbf{output size} = (input\ size\ - 1) * stride\ - 2 * padding + (kernel\ size\ - 1) + 1$$

Good explanation: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Transposed Convolution

- 1D example
- For filter sizes which produce an overlap in the output feature map, the overlapping values are simply added together.
- Less common names: **Deconvolution**, **Fractionally strided convolution**, **Up convolution**, ...

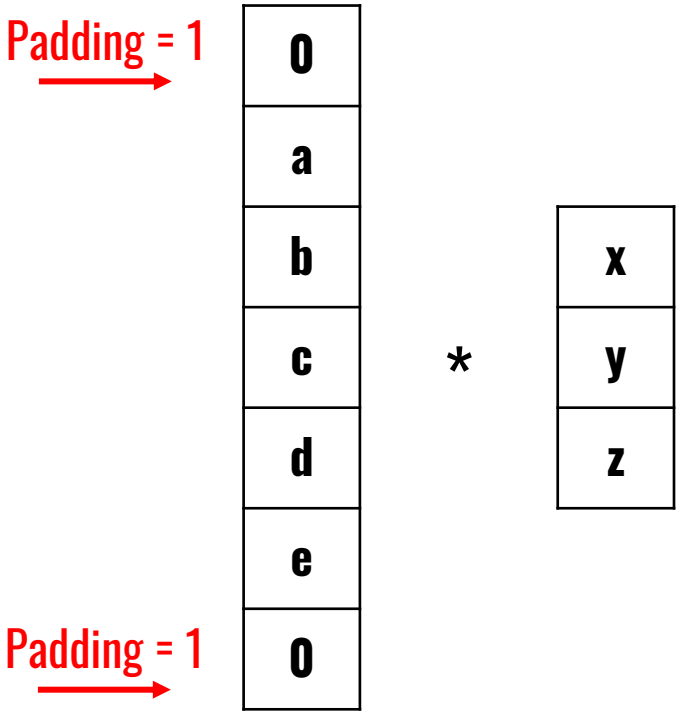


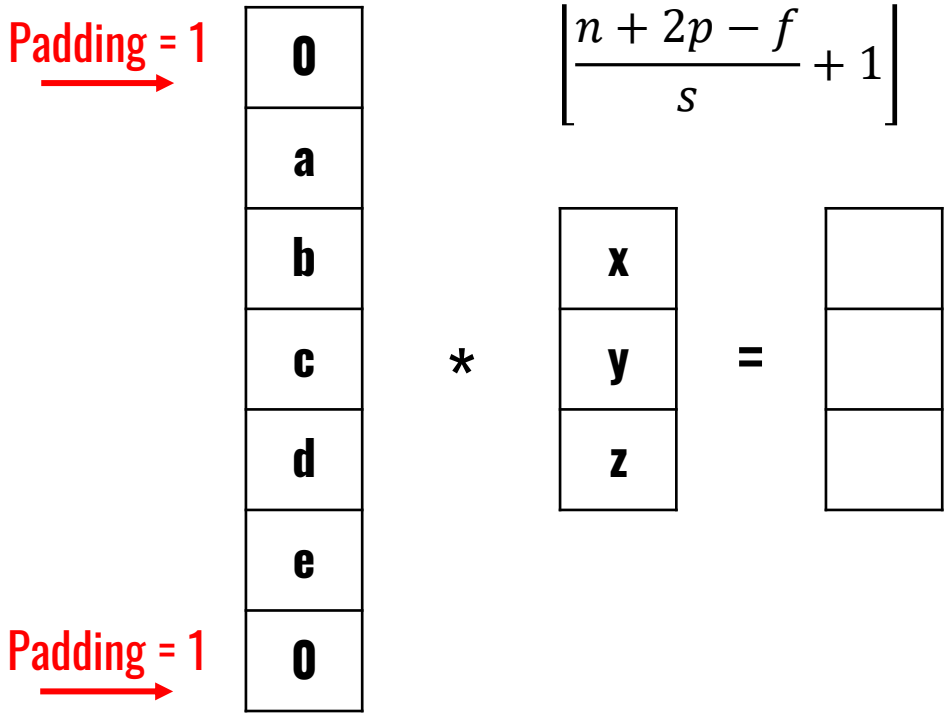
Normal Convolution

$n \times n$ *image* $f \times f$ *filter*

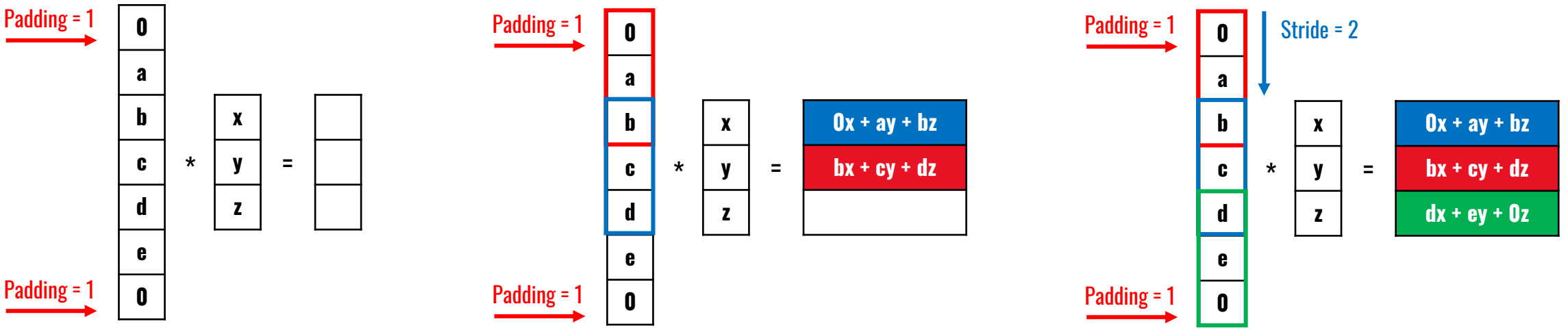
padding p *stride* s

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$



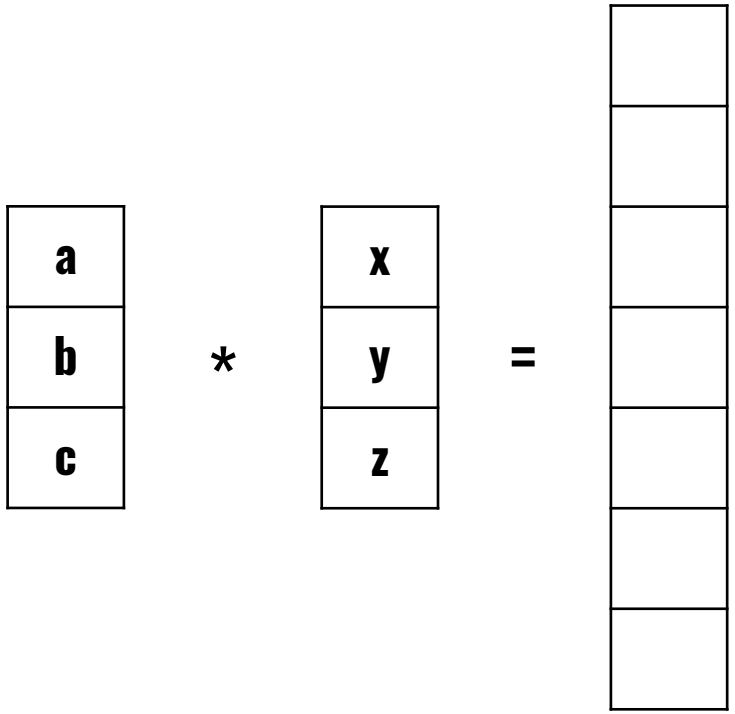


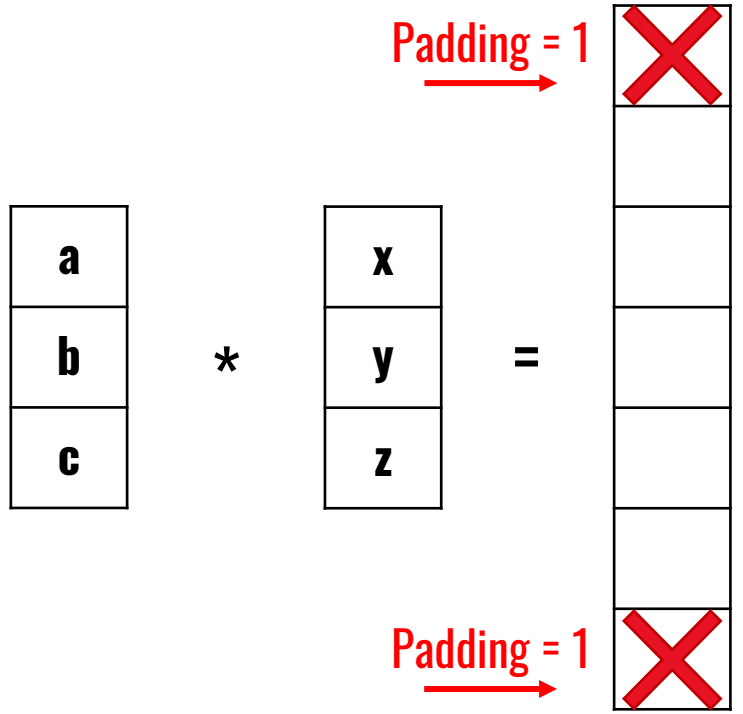
Normal Convolution



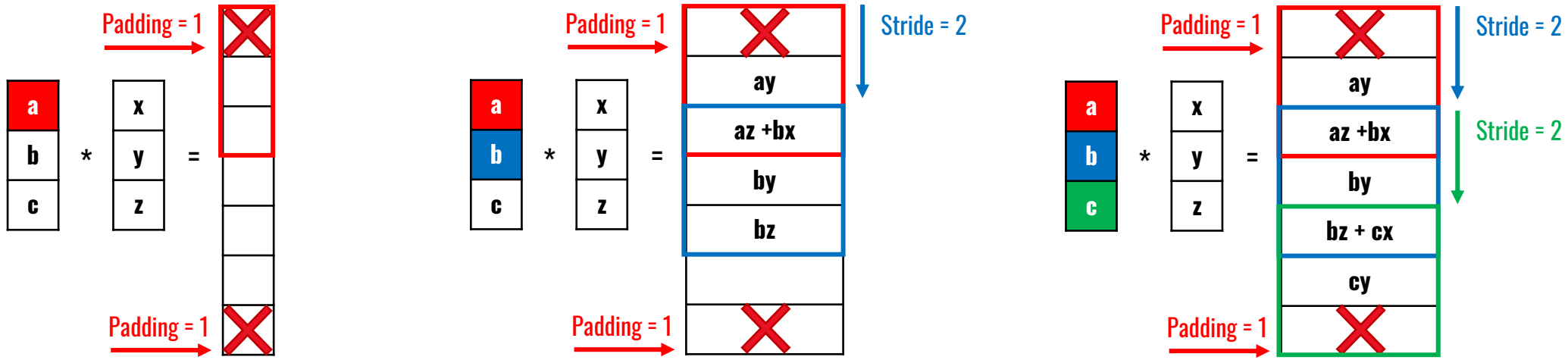
Transposed Convolution

$$\textbf{output size} = (input\ size\ - 1) * stride\ - 2 * padding + (kernel\ size\ - 1) + 1$$

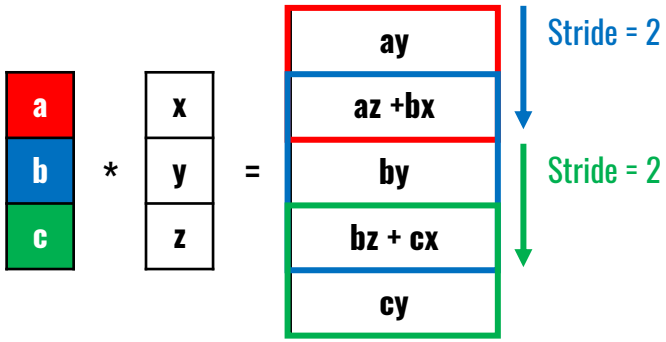


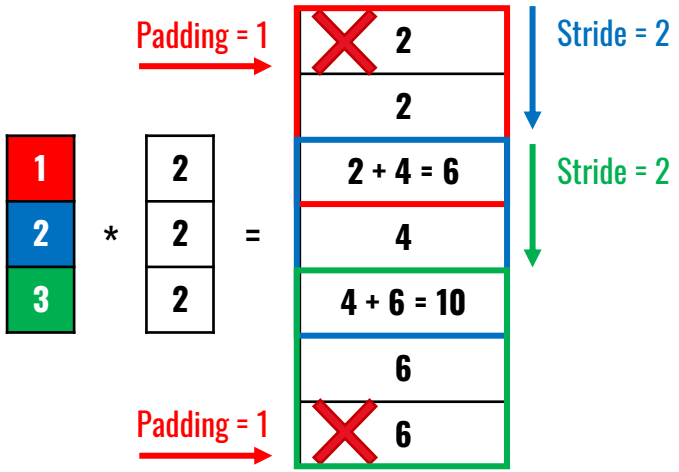


Transposed Convolution



Transposed Convolution

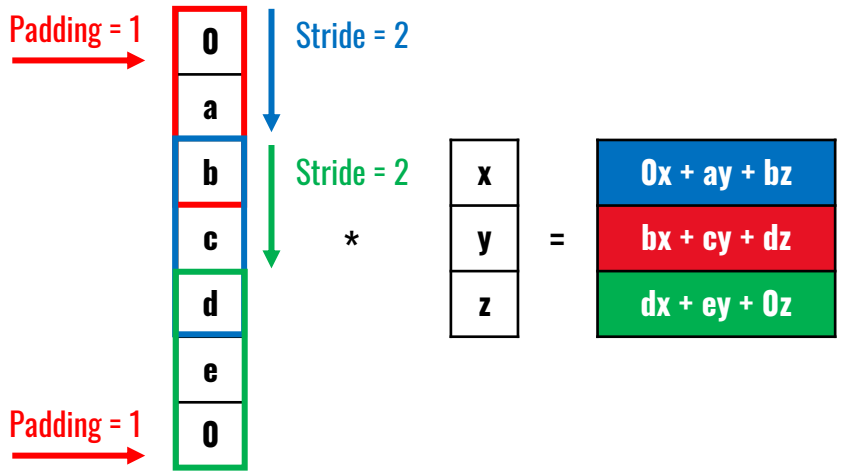




Summary on 1D

Convolution

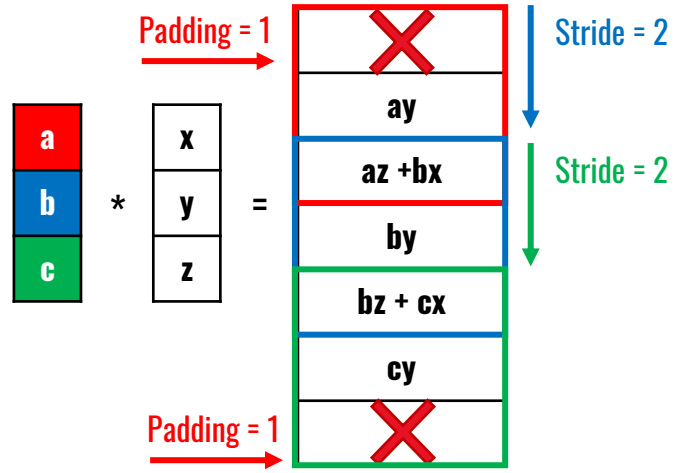
Input size: 5 **Filter size:** 3 **Stride:** 2 **Padding:** 1 **Output size:** 3



VS

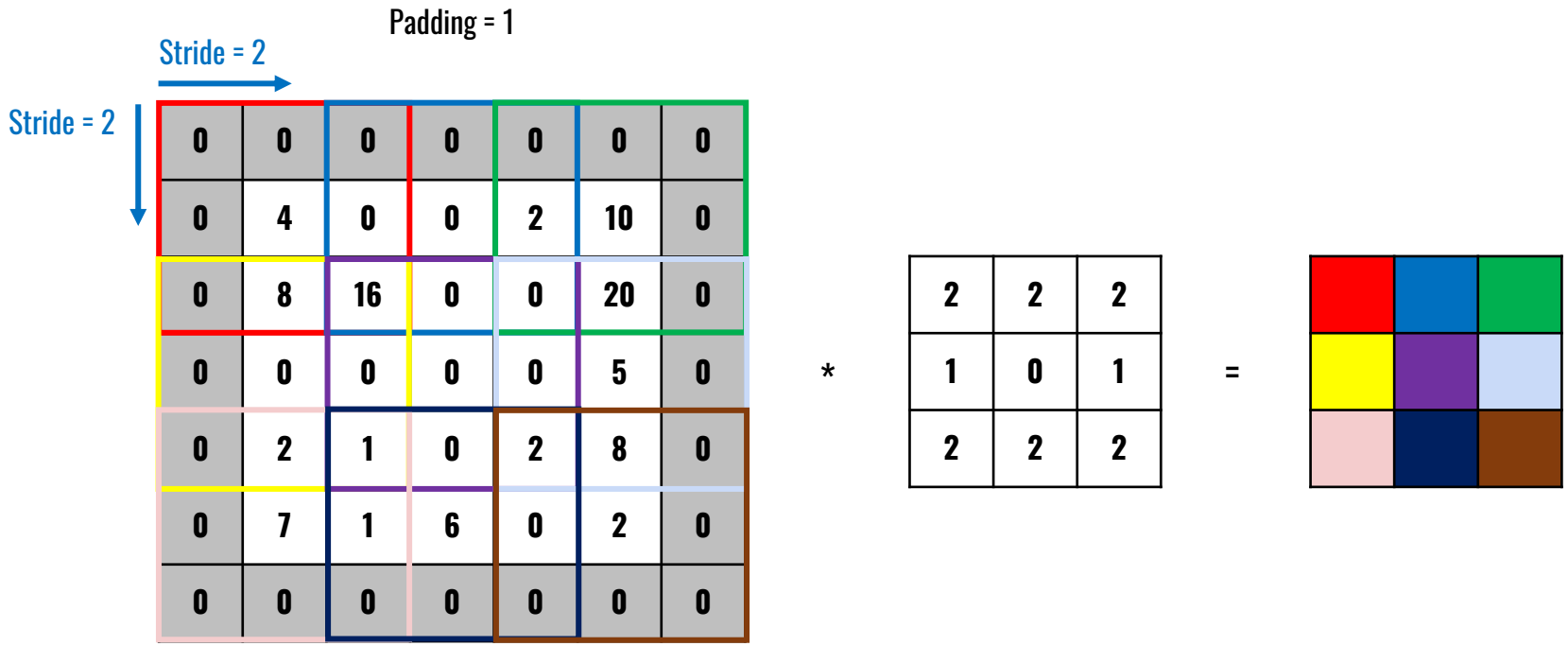
Transposed Convolution

Input size: 3 **Filter size:** 3 **Stride:** 2 **Padding:** 1 **Output size:** 5



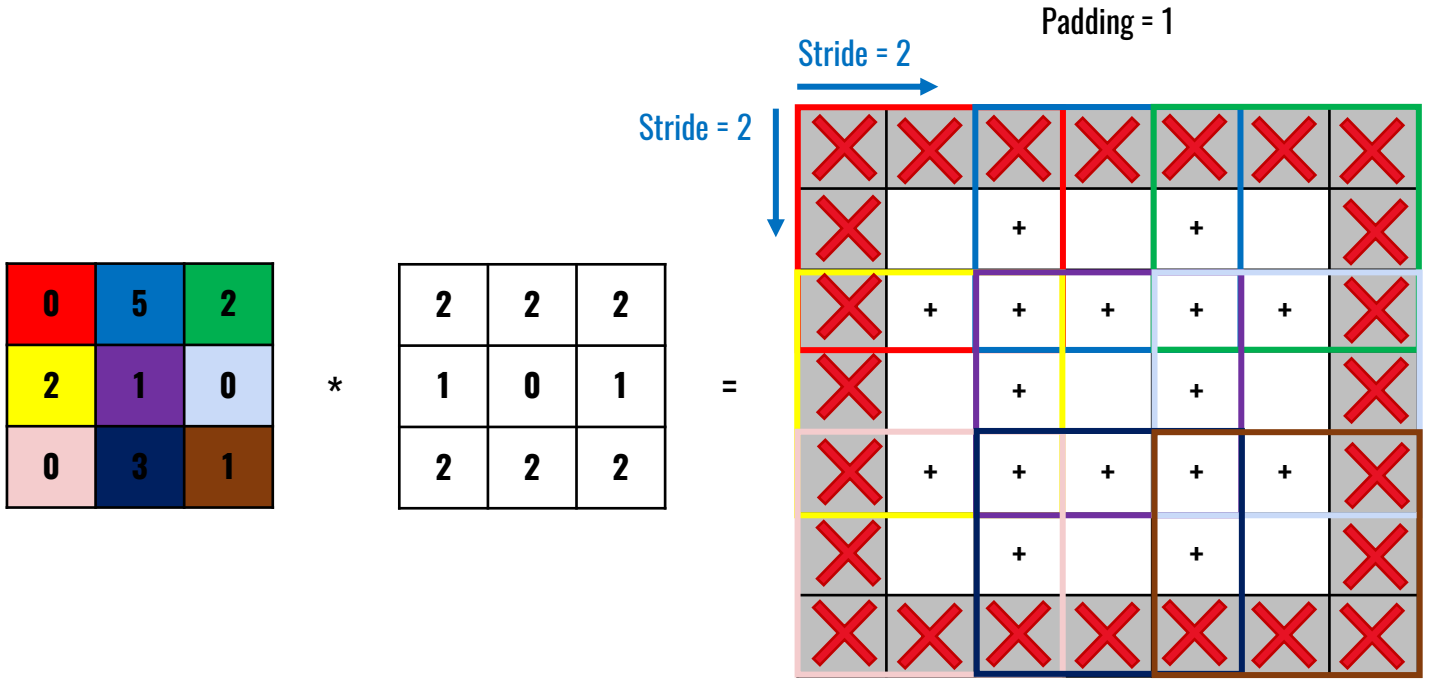
Convolution

- 2D example



Transposed Convolution

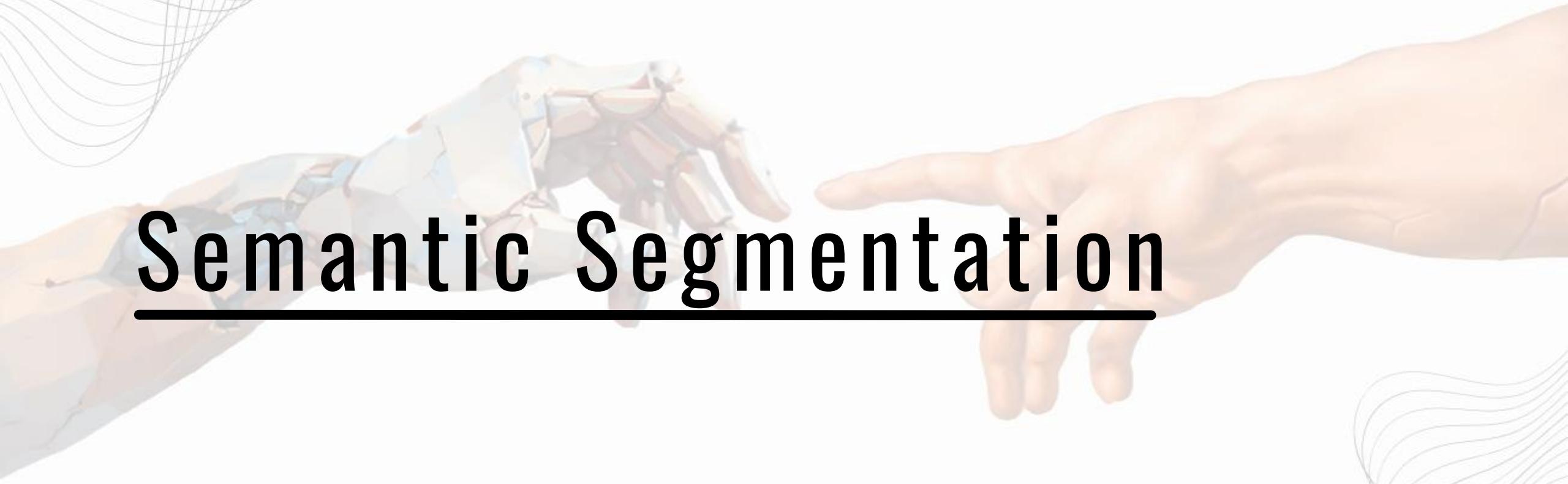
- 2D example



Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 5



Semantic Segmentation

1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

Semantic Segmentation

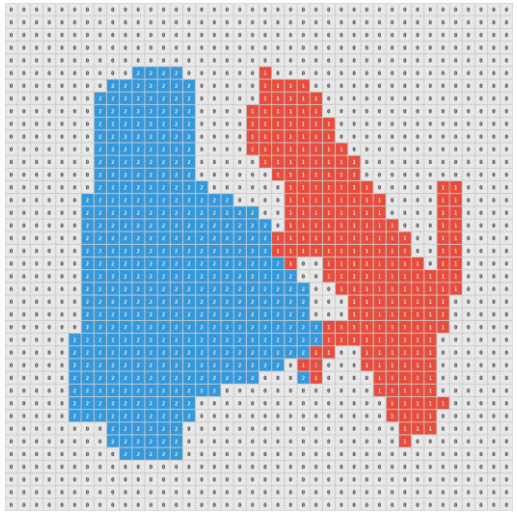
Semantic image segmentation is the task of **labelling each pixel** of an image with a corresponding **class** of what is being represented.

Input: RGB image (height × width × 3) or a grayscale (height × width × 1)

Output: a segmentation map (height × width × 1), where each pixel contains a class label represented as an integer.



Input



Semantic Labels

Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Multiclass Classification (Recap)

Output of the neural network is a *K* long vector

How should we encode the ground-truth values?

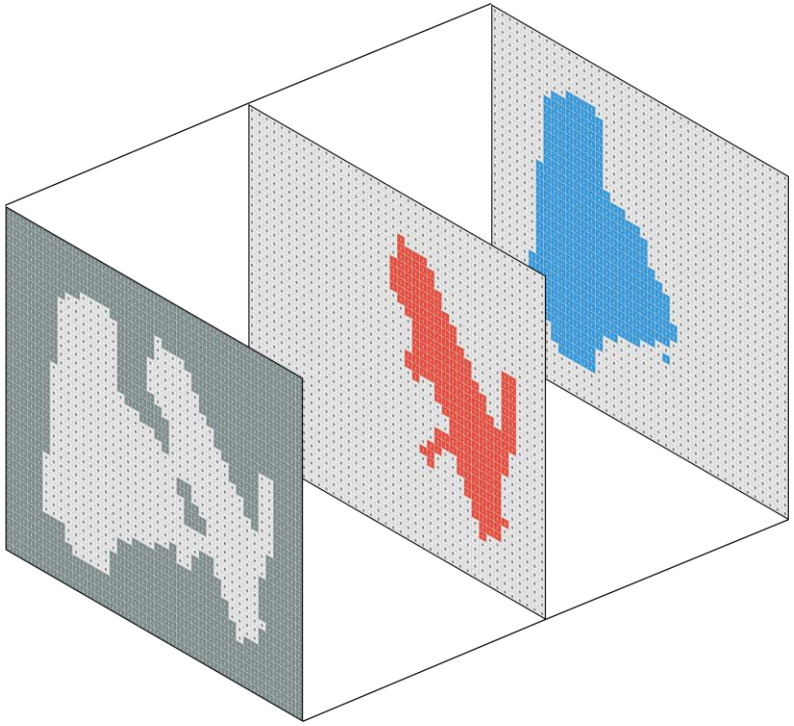
One-hot encoding (K=3):

(cat) 1 -> [1, 0, 0]
(dog) 2 -> [0, 1, 0]
(horse) 3 -> [0, 0, 1,]

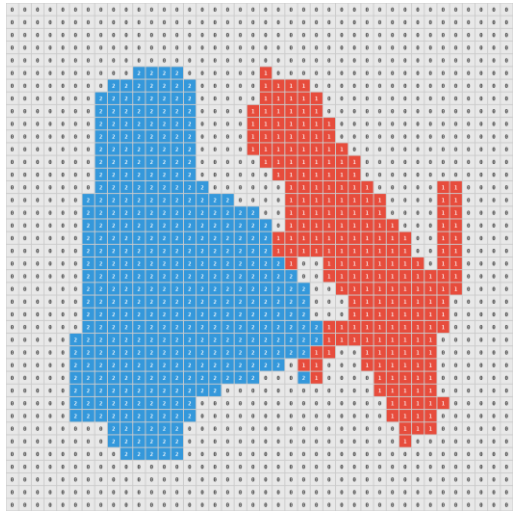
Just as h(x): values between 0 and 1, sum up to 1

Semantic Segmentation

Similar to how we treat standard categorical values, we'll create our **target** by one-hot encoding the class labels - essentially creating **an output channel for each of the possible classes**.



Depth-wise argmax



Semantic Labels

A prediction can be collapsed into a segmentation map by taking the argmax of each depth-wise pixel vector.

Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

How to solve it? The naïve approach:

- Sliding Windows + Classification
  - Computationally expensive
  - Not reusing shared features

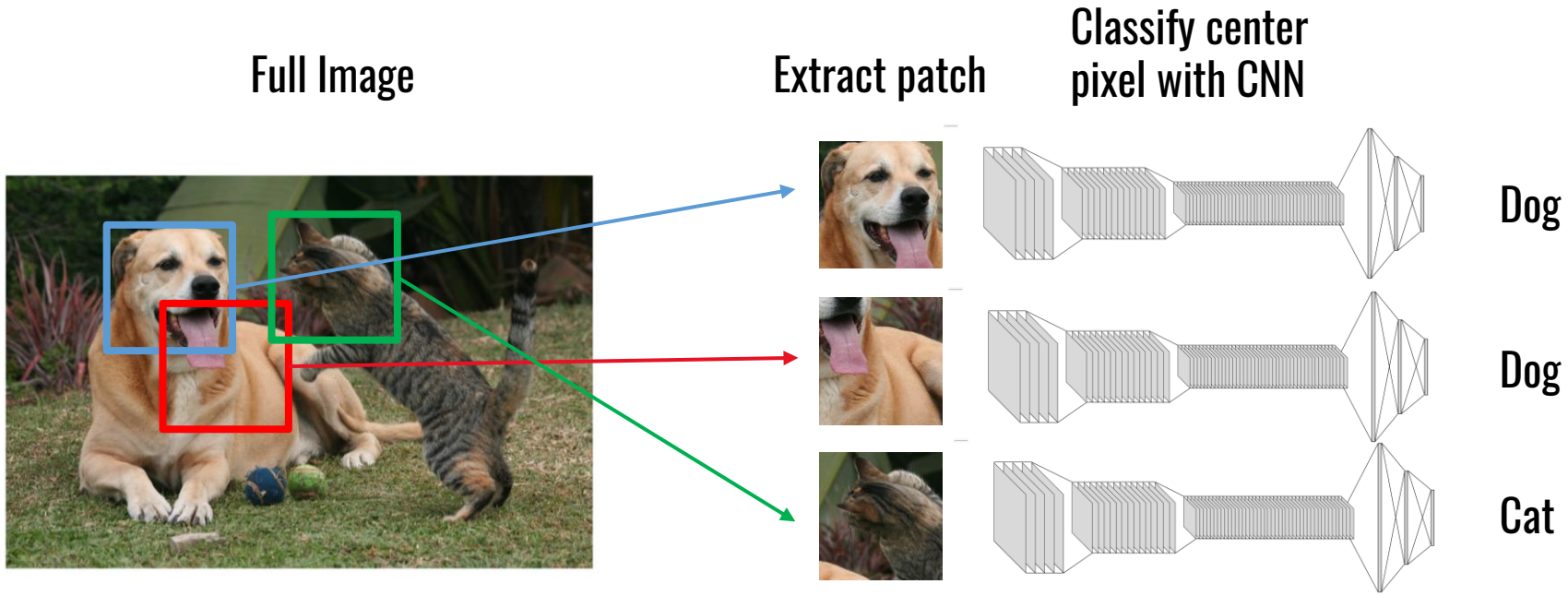


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

How to solve it? The naïve approach:

- Apply convolutions for all pixels at once, keeping original resolution
  - Computationally expensive
  - Does not enforce network to learn key features. It only learns a direct mapping from input pixels to the segmentation pixels.

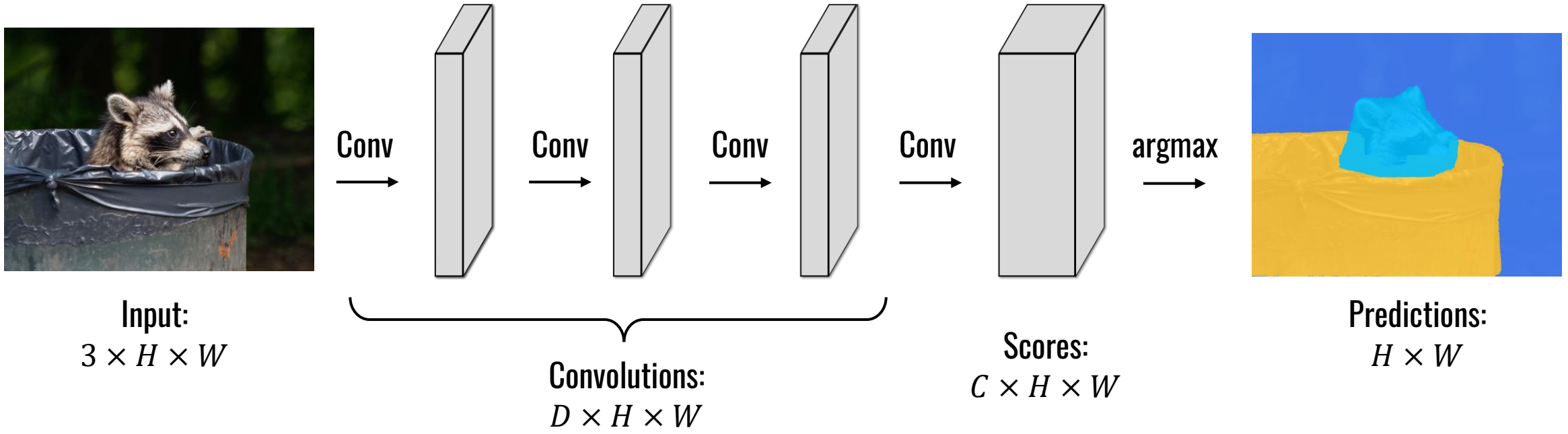


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Convolutional Autoencoders

- Specifically designed for **image data**.
- They employ convolutional layers in both the encoder and decoder parts of the network.
- This architecture allows them to **capture spatial dependencies and hierarchical features** effectively.
- The reconstruction of the input image is often blurry and of lower quality due to compression during which information is lost.

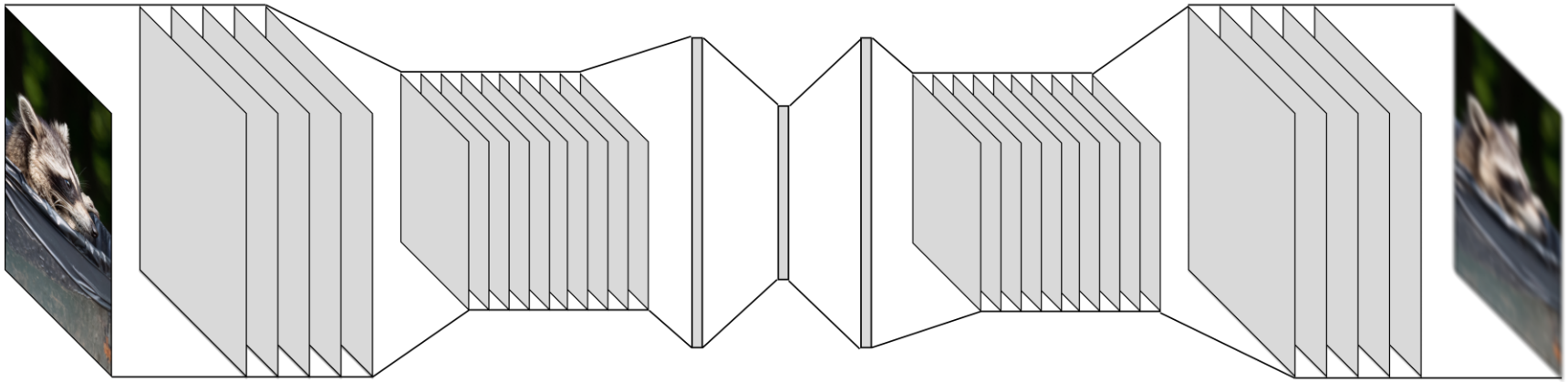


Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Convolutional Autoencoders

Image Segmentation

- Image segmentation is the process of **partitioning** an image **into multiple segments** each belonging to a class.
- The goal is to simplify and/or change the representation of an image by **grouping pixel values according to the class they belong to**.

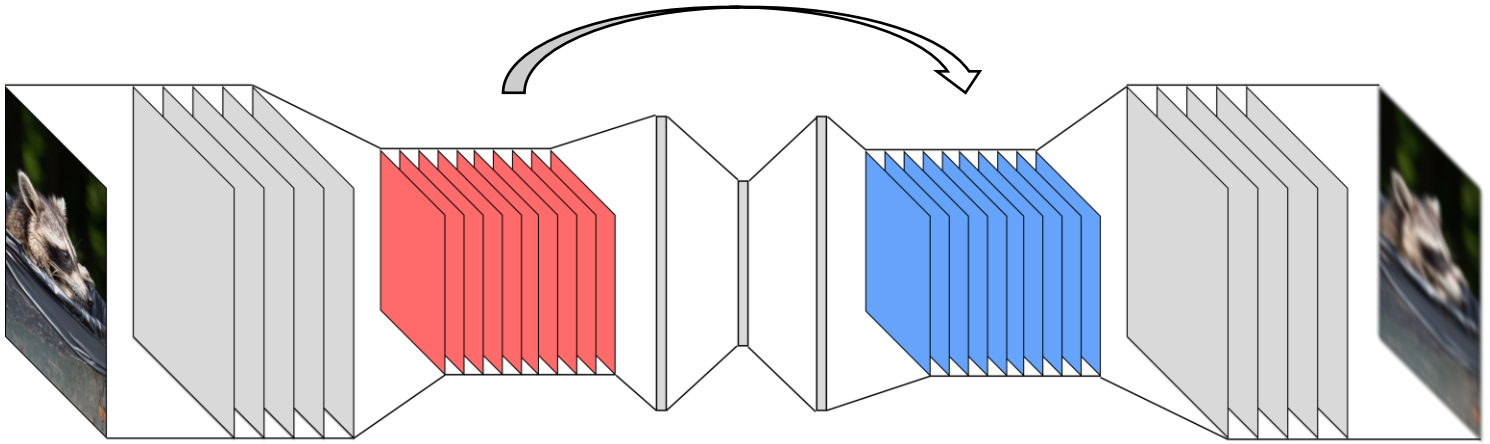


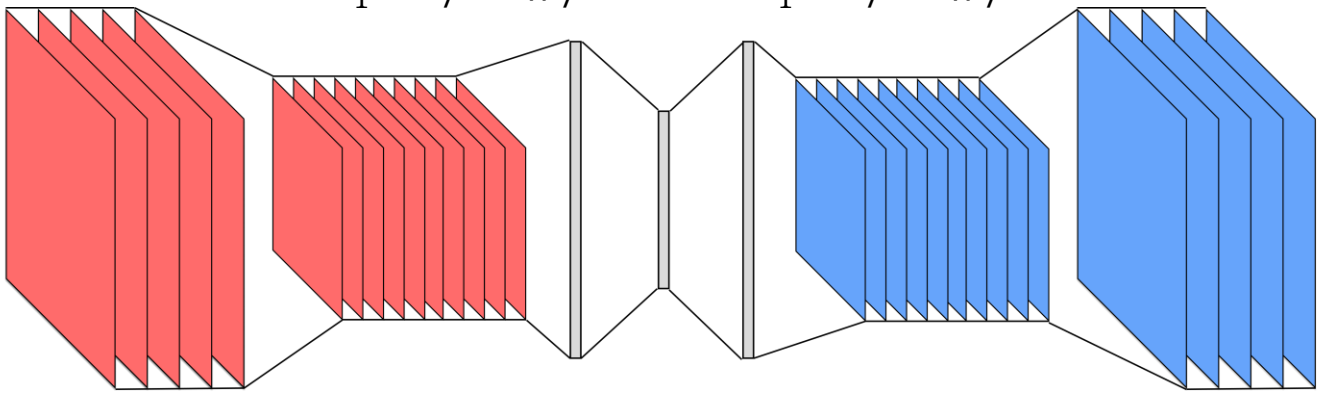
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Encoder-Decoder Structure

- One popular approach for image segmentation models is to follow an **encoder**/**decoder** **structure** where we **downsample** the spatial resolution of the input, developing lower-resolution feature mappings which are learned to be highly efficient at discriminating between classes, and then **upsample** the feature representations into a full-resolution segmentation map.



Input:
$3 \times H \times W$



High-res:
$D_1 \times H/2 \times W/2$

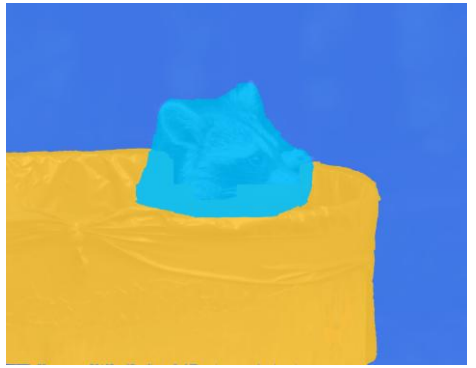
Low-res:
$D_1 \times H/2 \times W/2$

High-res:
$D_1 \times H/2 \times W/2$

Encoder

Bottleneck

Decoder

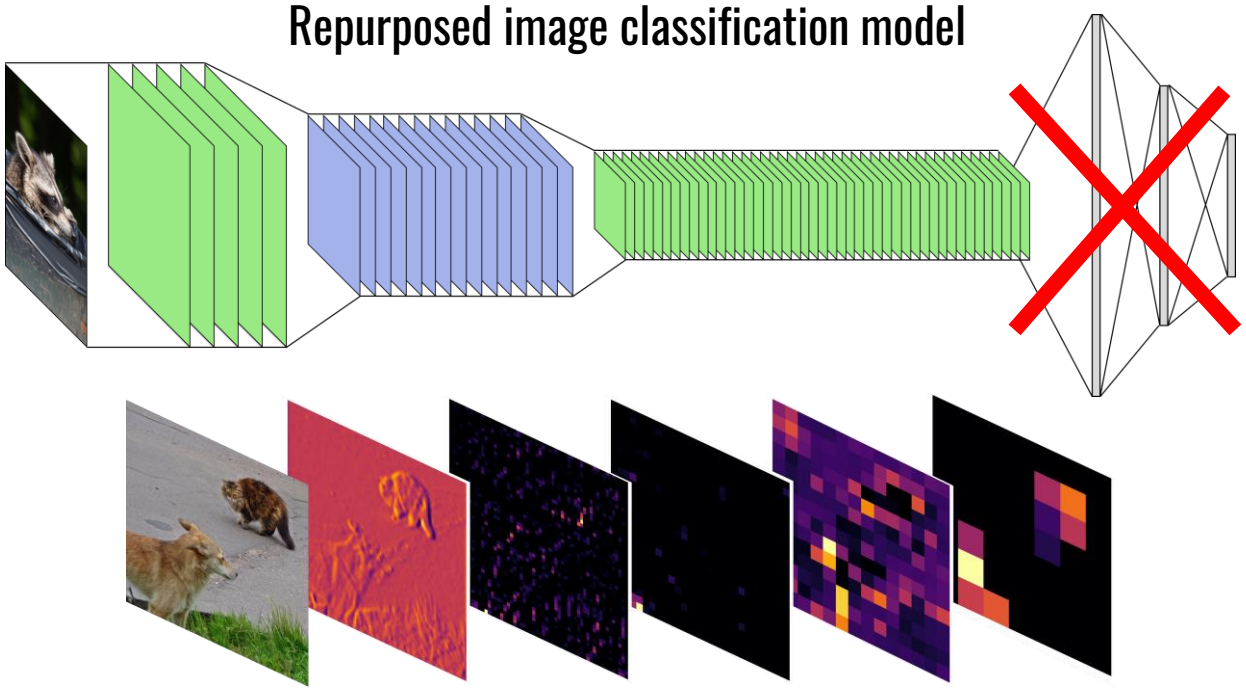


Predictions:
$H \times W$

Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Fully Convolutional Network (FCN)

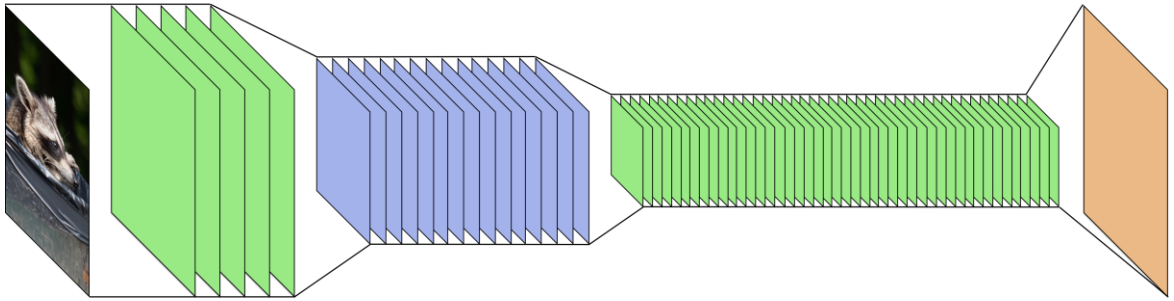
The approach of using a *"Fully Convolutional"* network trained end-to-end, pixels-to-pixels for the task of image segmentation was introduced by Long et al. in late 2014. The paper's authors propose adapting existing, well-studied **image classification** networks (e.g. **AlexNet**) to serve as the encoder module of the network, appending a decoder module with transpose convolutional layers to upsample the coarse feature maps into a full-resolution segmentation map.

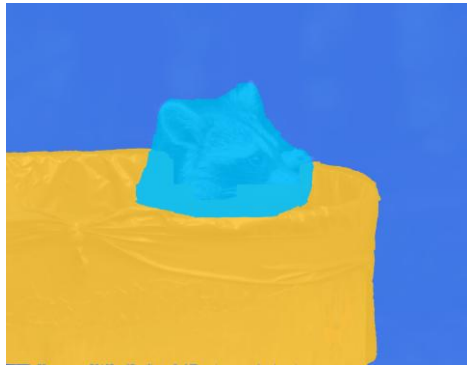


The encoder produces a coarse feature map which is then refined by the decoder module.

Long, J., Shelhamer, E., & Darrell, T. (2014). Fully convolutional networks for semantic segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1411.4038

Fully Convolutional Network (FCN)



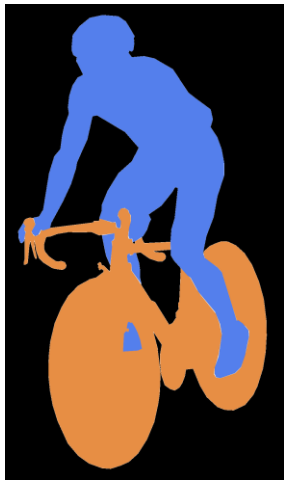


However, because the encoder module reduces the resolution of the input by a factor of 32, the decoder module **struggles to produce fine-grained segmentations**.

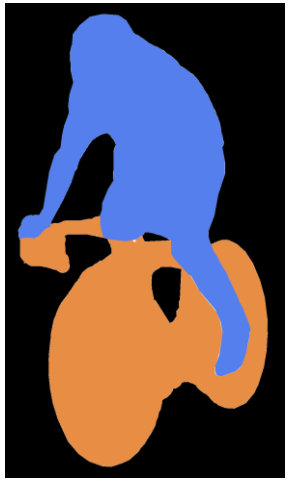
Original Image



Ground truth target



Predicted segmentation

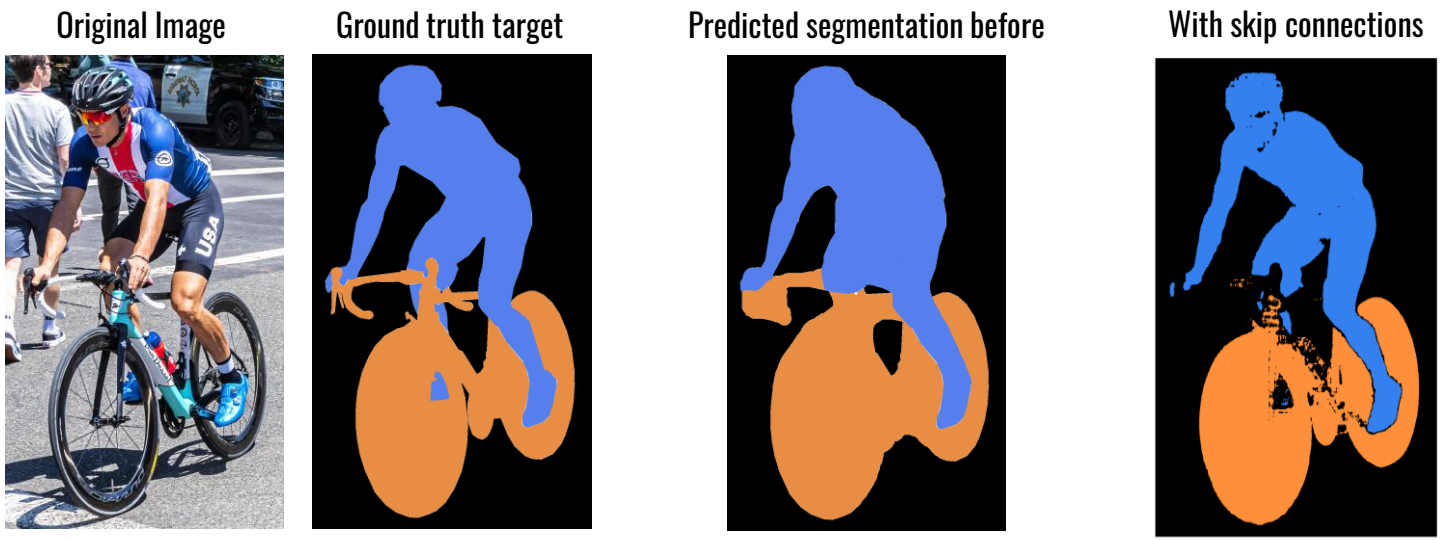


Illustration

Long, J., Shelhamer, E., & Darrell, T. (2014). Fully convolutional networks for semantic segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1411.4038
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Travis McCabe of Team USA (48068764576) (cropped).jpg, Ray Rogers from Novato, United States, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

Fully Convolutional Network (FCN)

- **Adding skip connections** - The authors address this tension by slowly upsampling (in stages) the encoded representation, adding "skip connections" from earlier layers, and summing these two feature maps.
- These skip connections from earlier layers in the network (prior to a downsampling operation) should provide the necessary detail to reconstruct accurate shapes for segmentation boundaries. Indeed, we can recover more fine-grain detail with the addition of these skip connections.

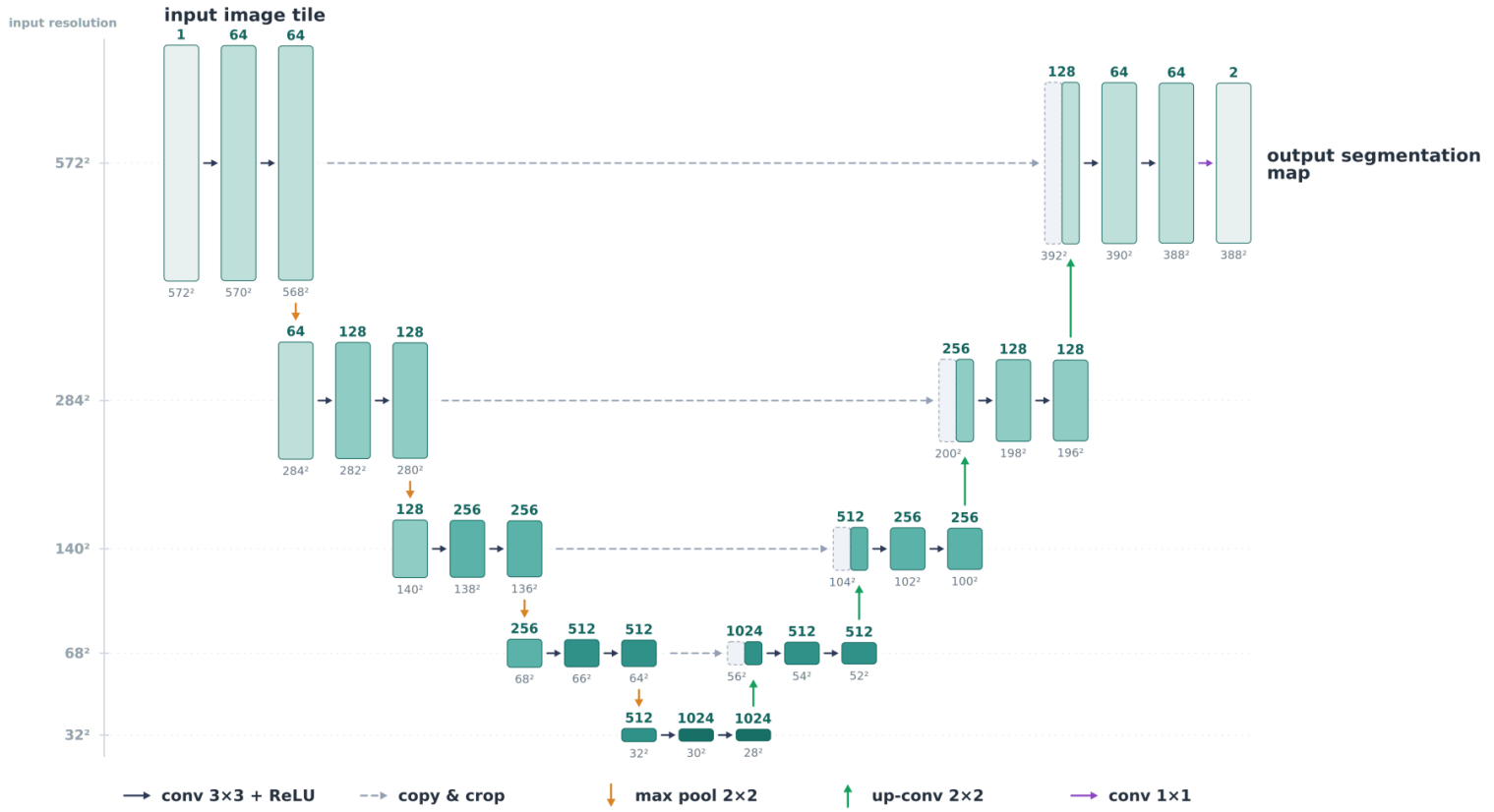


Illustration

Long, J., Shelhamer, E., & Darrell, T. (2014). Fully convolutional networks for semantic segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1411.4038
Image Source: Travis McCabe of Team USA (48068764576) (cropped).jpg, Ray Rogers from Novato, United States, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

U-Net: Convolutional Networks for Biomedical Image Segmentation

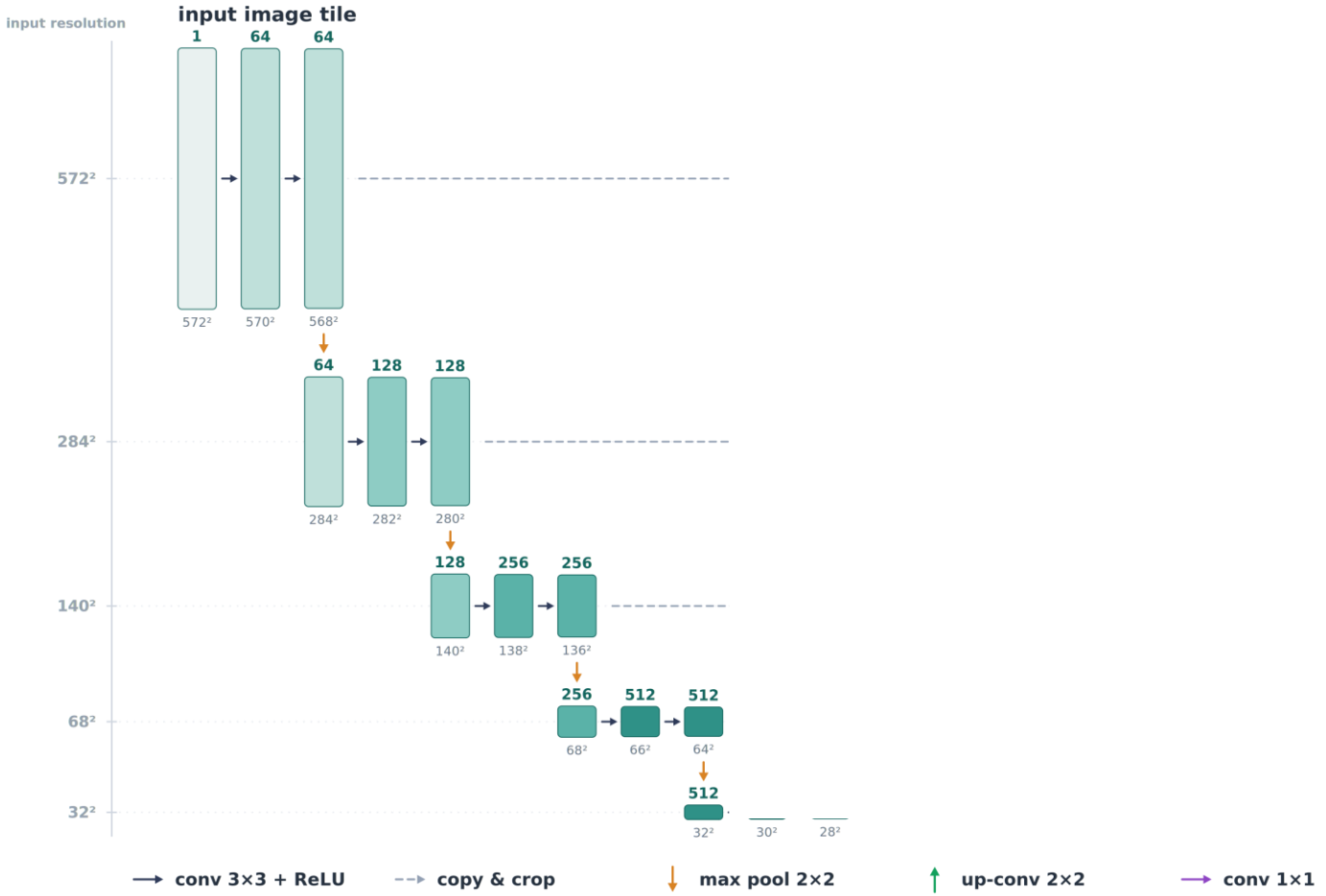
- Ronneberger et al. improve upon the "fully convolutional" architecture primarily through **expanding the capacity of the decoder** module of the network. More concretely, they propose the **U-Net architecture** which "consists of a contracting path to capture context and a **symmetric** expanding path that enables precise localization."



Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1505.04597

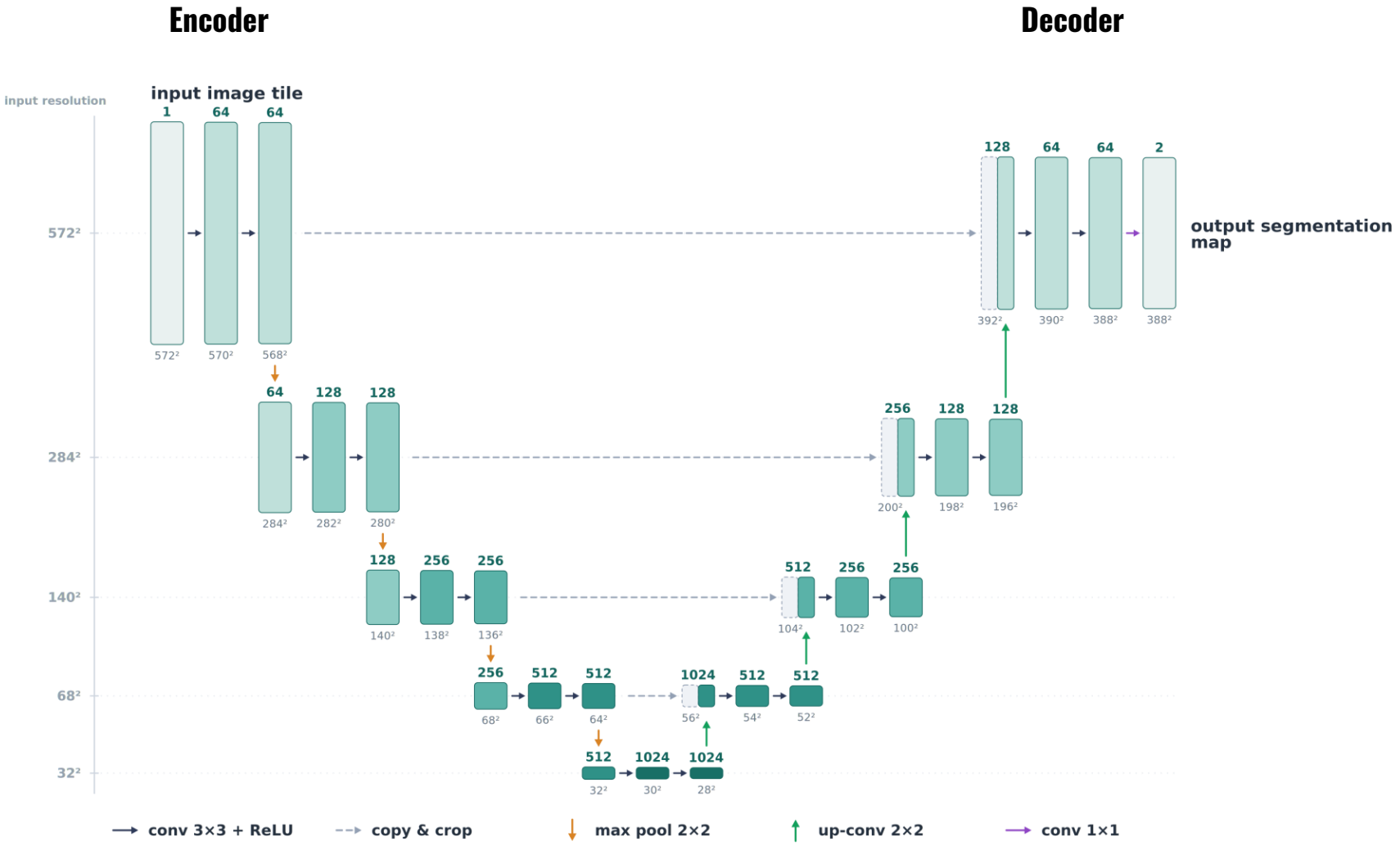
U-Net

Encoder



Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1505.04597

U-Net

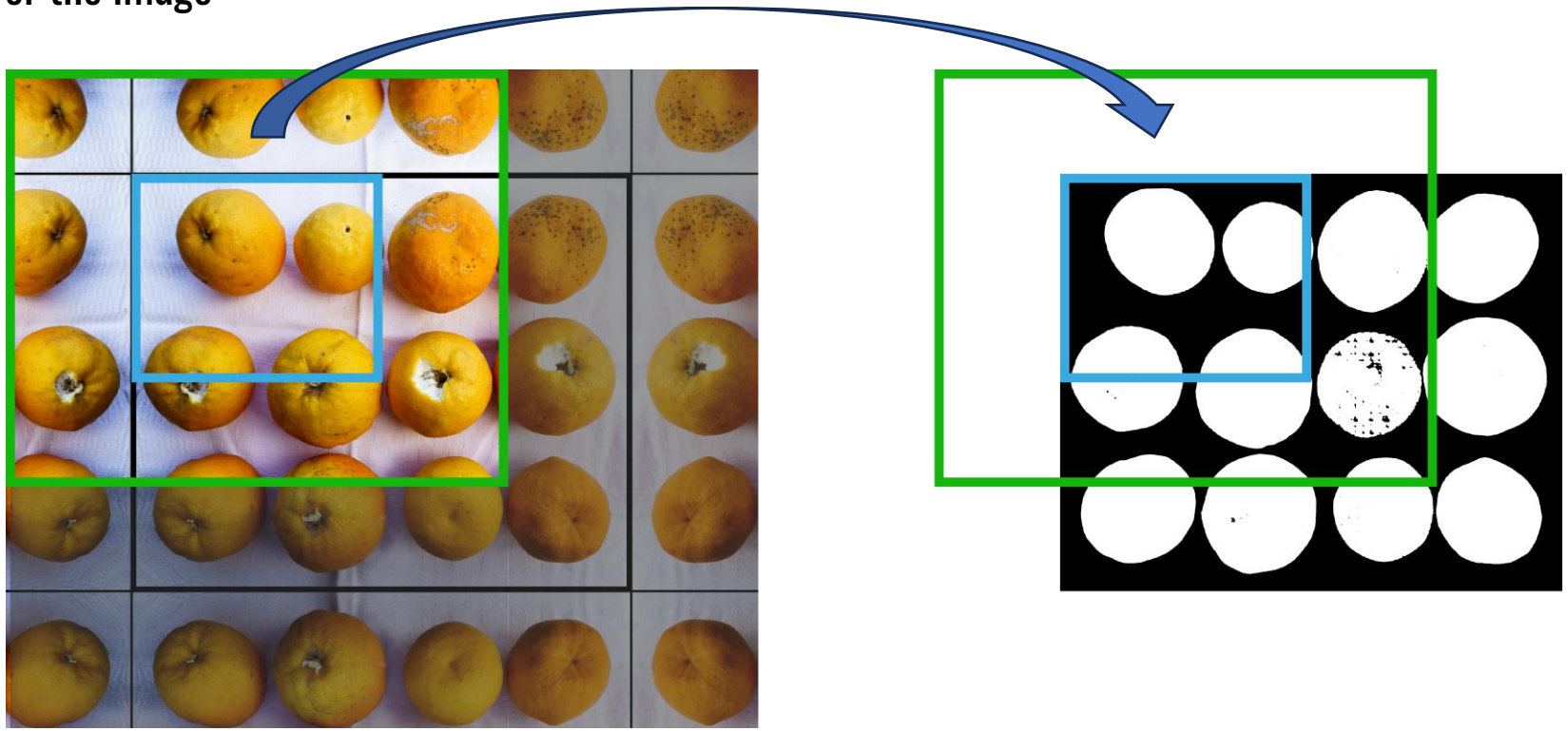


Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1505.04597

U-Net

How can we use it for images with arbitrary size?

- Do the segmentation for smaller regions of the image
- On the edges mirror the image



Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1505.04597
Image Source: Pomeranzen.jpg, Neptuul, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Architectures

Advanced U-Net variants

- The standard U-Net model consists of a series of convolution operations for each "block" in the architecture. These convolutional blocks can be replaced for more advanced ones such as:
  - ResNet blocks
  - Inception modules
  - Dense blocks
  - Etc.

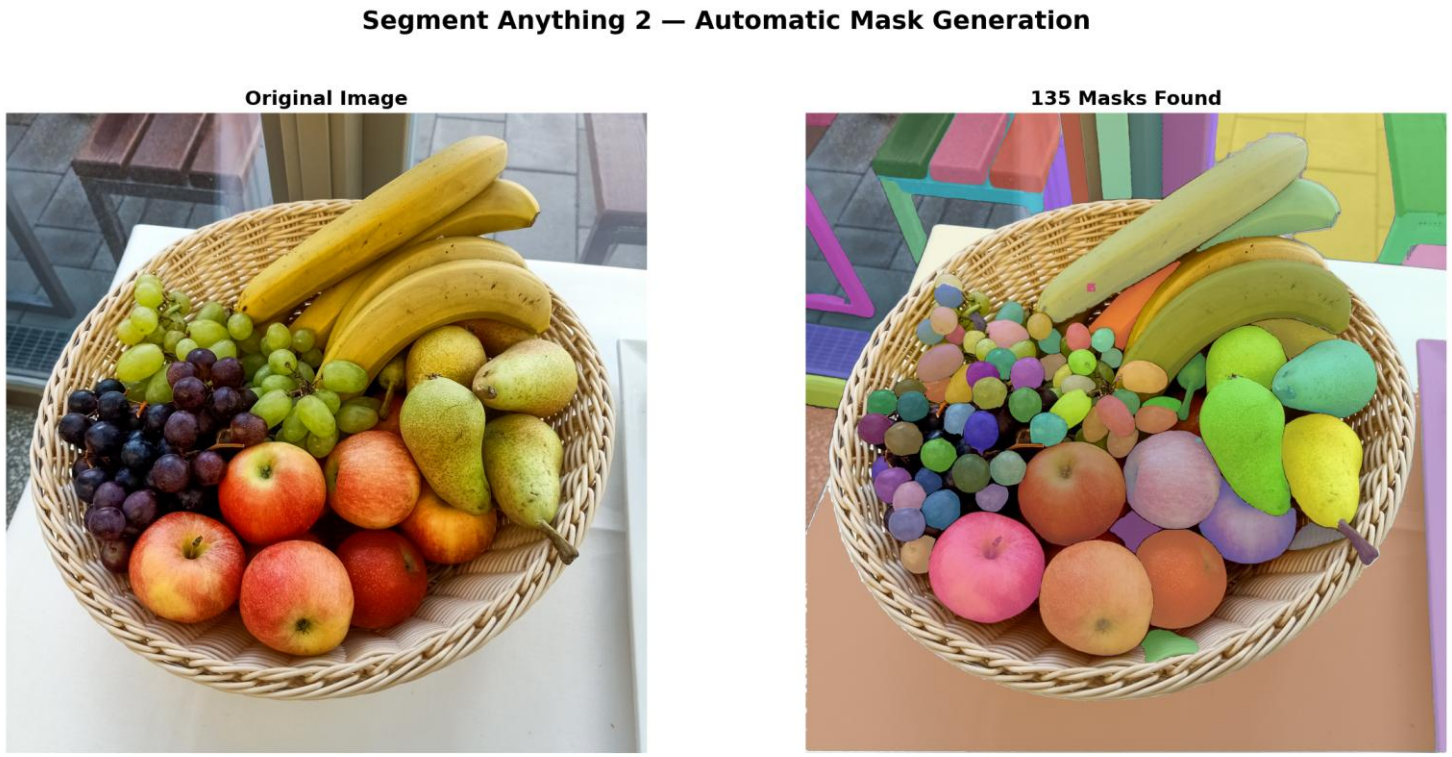
DeepLab

- Deeplab from a group of researchers from Google have proposed a multitude of techniques to improve the existing results and get finer output at lower computational costs. The 3 main improvements suggested as part of the research are:
  - Atrous convolutions
  - Atrous Spatial Pyramidal Pooling
  - Conditional Random Fields usage for improving final output

DeepLab v3: https://arxiv.org/abs/1706.05587v3

Segment Anything Model (SAM 2)

- Extending SAM with memory to keep the masks over the whole video
- Images are considered as a single frame video
- Object segmentation, there is no class prediction
- Video Object Segmentation (VOS) and Interactive Video Object Segmentation (iVOS)
- https://sam2.metademolab.com/demo



Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., & Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2408.00714
Image Source: A basket of fruits.jpg, Suyash Dwivedi, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Training

How to train such networks?

- Have input X (images) and labels Y (masks).
- Define architecture
- Set hyperparameters
- Define loss function and metrics

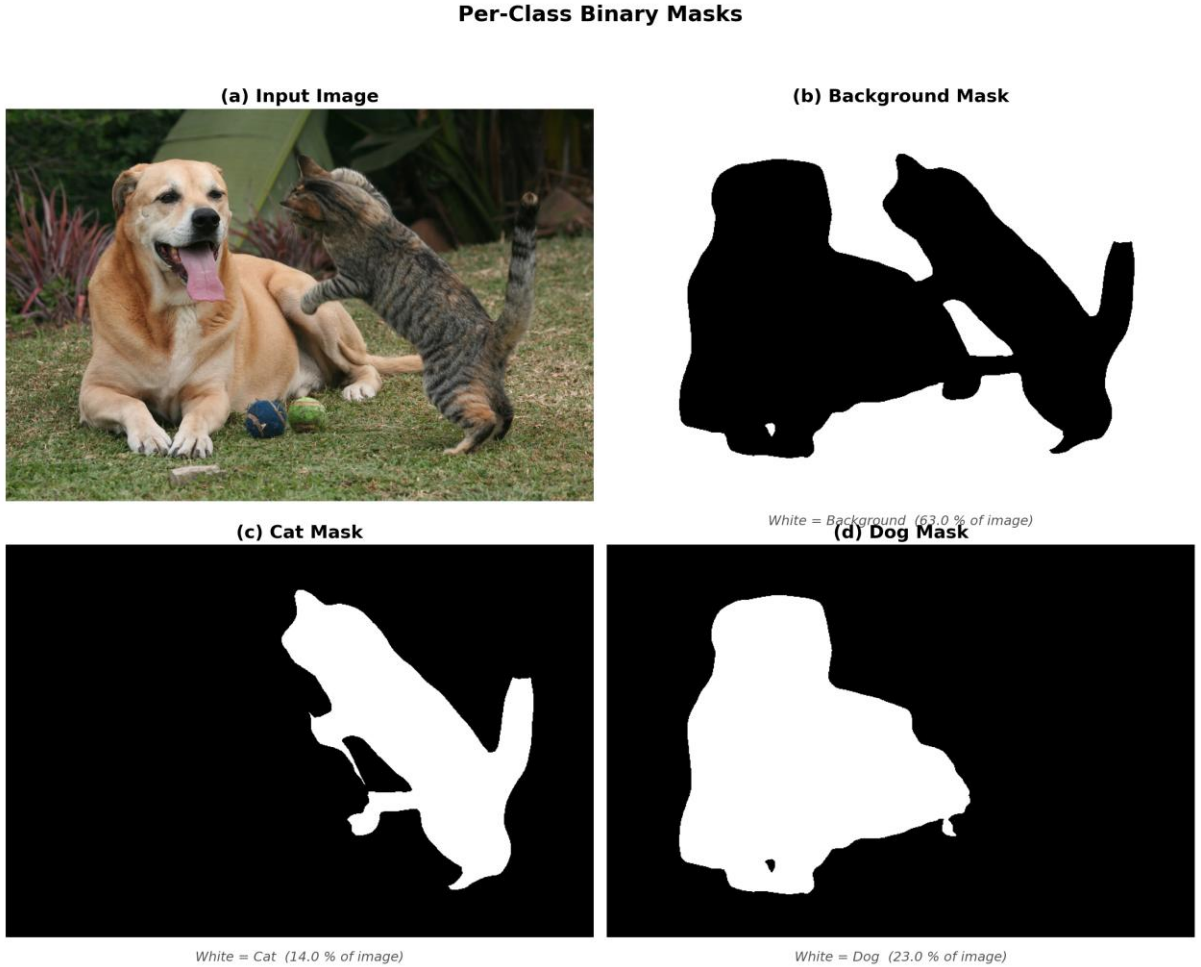


Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Losses

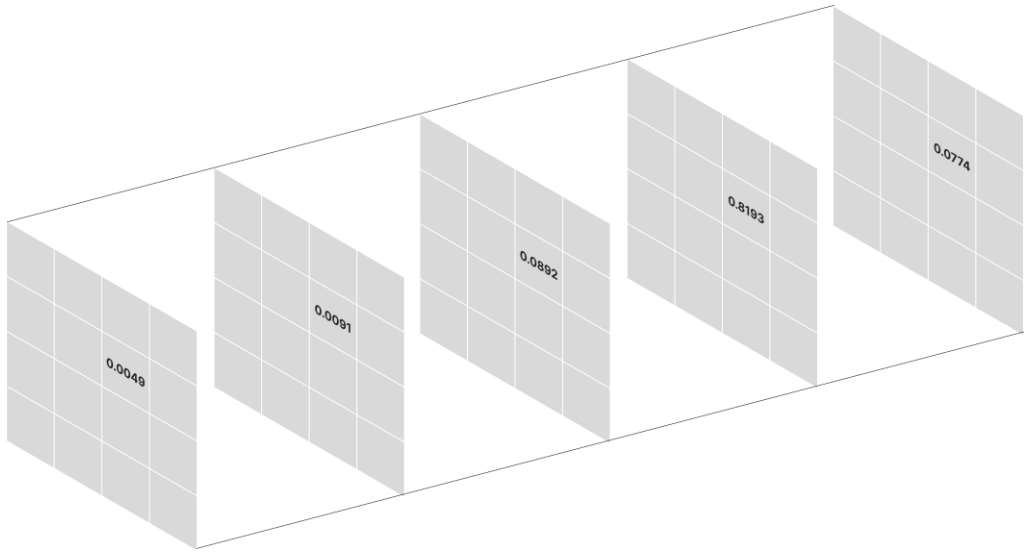
Pixel-wise cross entropy loss

- This loss examines *each pixel individually*, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector.
- Problematic for unbalanced classes

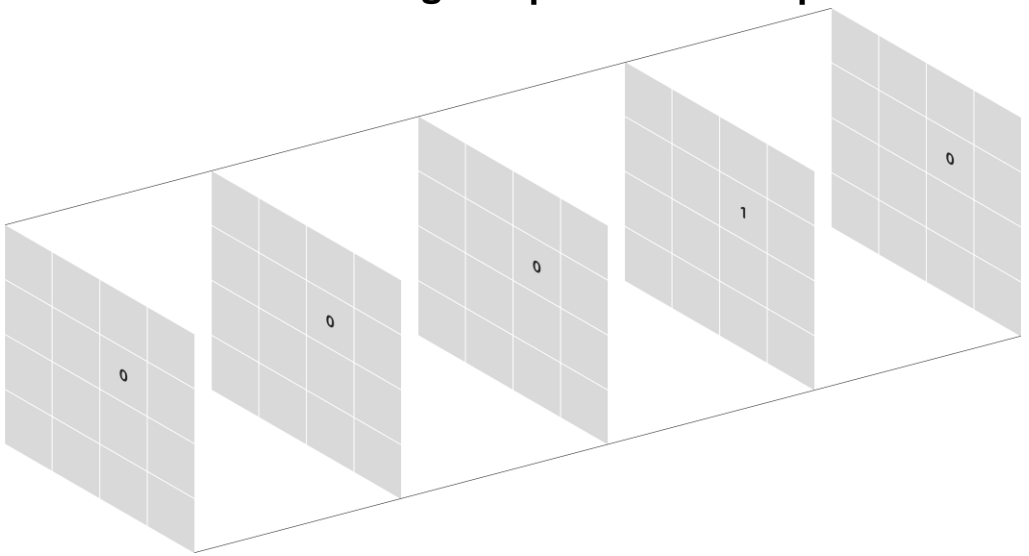
Pixel-wise loss is calculated as the log loss, summed over all possible classes

$$-\sum_{classes} y_{true} \log(y_{pred})$$

The scoring is repeated over all pixels and averaged



Prediction for a selected pixel



Target for the corresponding pixel

Losses

Pixel-wise cross entropy loss

- This loss examines *each pixel individually*, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector - problematic for unbalanced classes
- What happens if we have small objects and the model predicts a full black image – we get a small cross-entropy loss – the model does not learn how to segment small objects

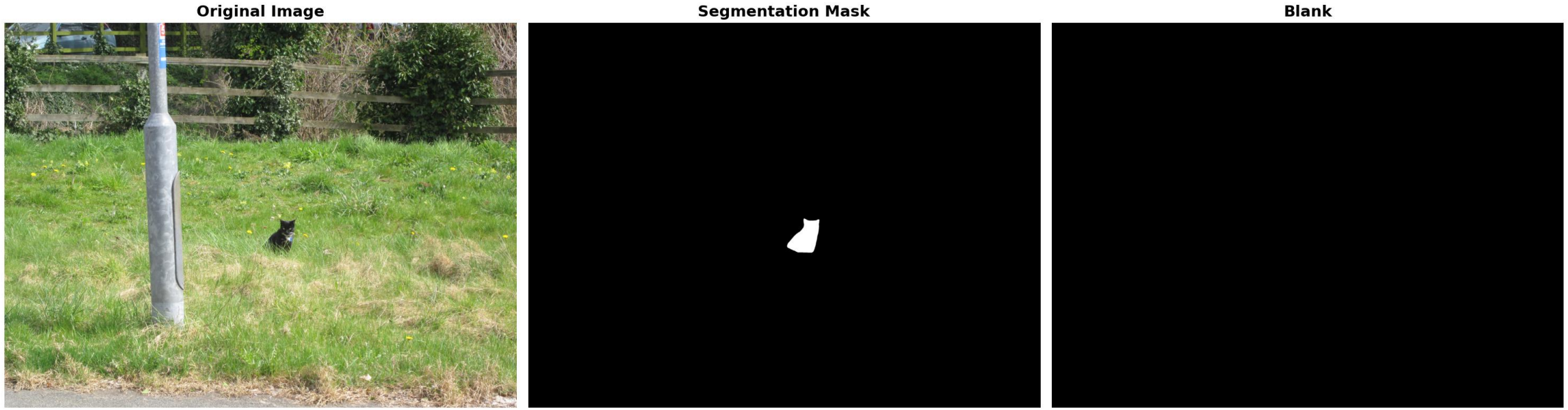


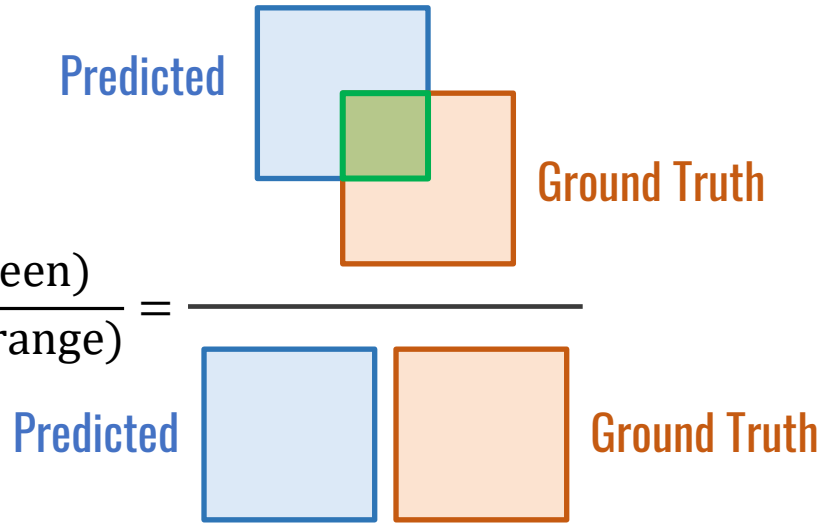
Image Source: A cat at Bridge Street, Neston.JPG, ReptOn1x, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

Losses

Dice Loss

- Another popular loss function for image segmentation tasks is based on the Dice coefficient, which is essentially a measure of overlap between two samples. This measure ranges from 0 to 1 where a Dice coefficient of 1 denotes perfect and complete overlap. The Dice coefficient was originally developed for binary data, and can be calculated as:

$$Dice = 2 \times \frac{|A \cap B|}{|A| + |B|} = 2 \times \frac{\text{Area of overlap (green)}}{\text{Total area (blue + orange)}} =$$

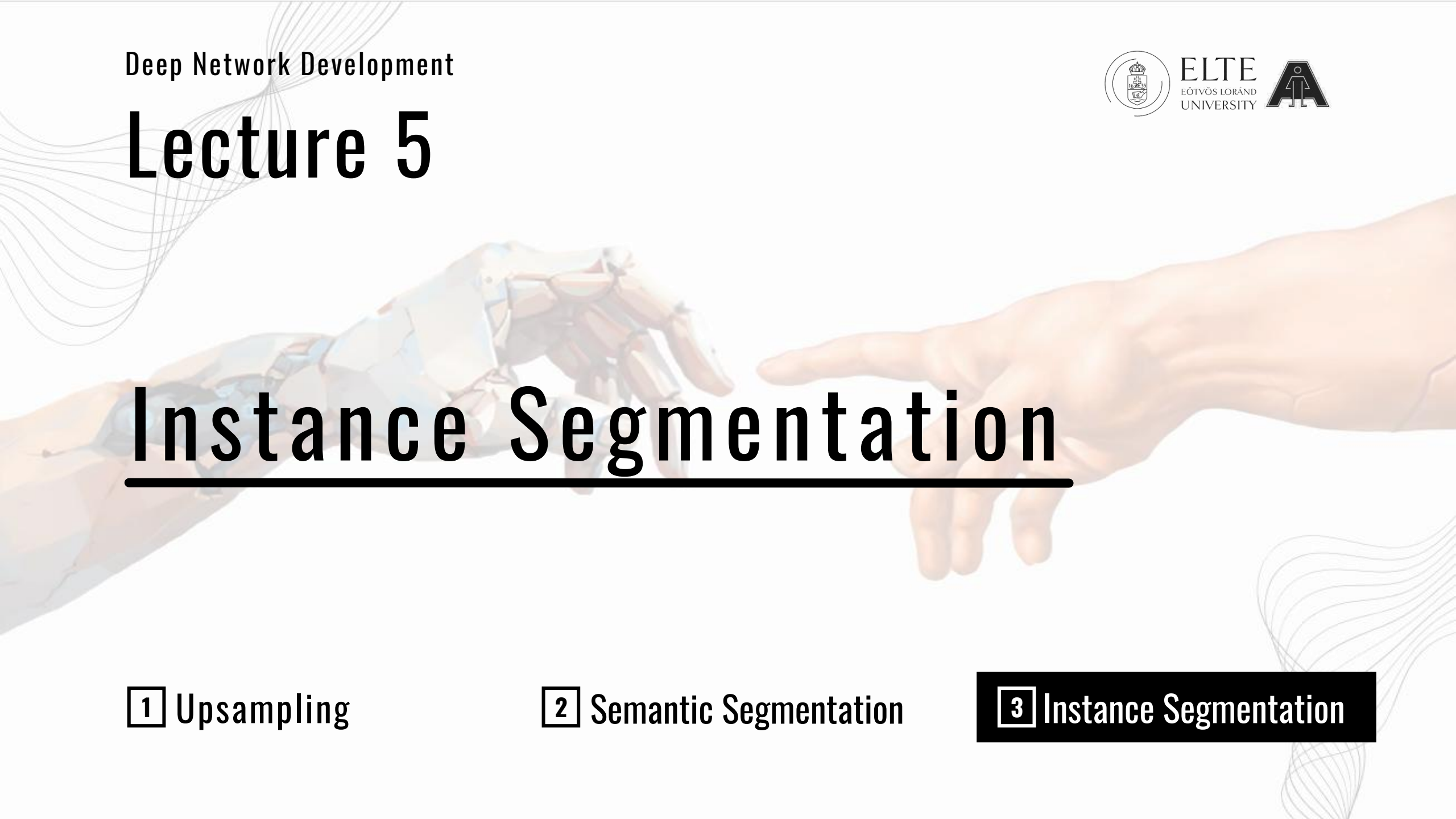


where:

- $|A \cap B|$ represents the common elements between sets A and B, and
- $|A|$ represents the number of elements in set A
- $|B|$ represents the number of elements in set B

Deep Network Development

Lecture 5



Instance Segmentation

1 Upsampling

2 Semantic Segmentation

3 Instance Segmentation

What is Instance Segmentation?

- Instance Segmentation is identifying each object instance for every known object within an image. It assigns a label to each pixel of the image.

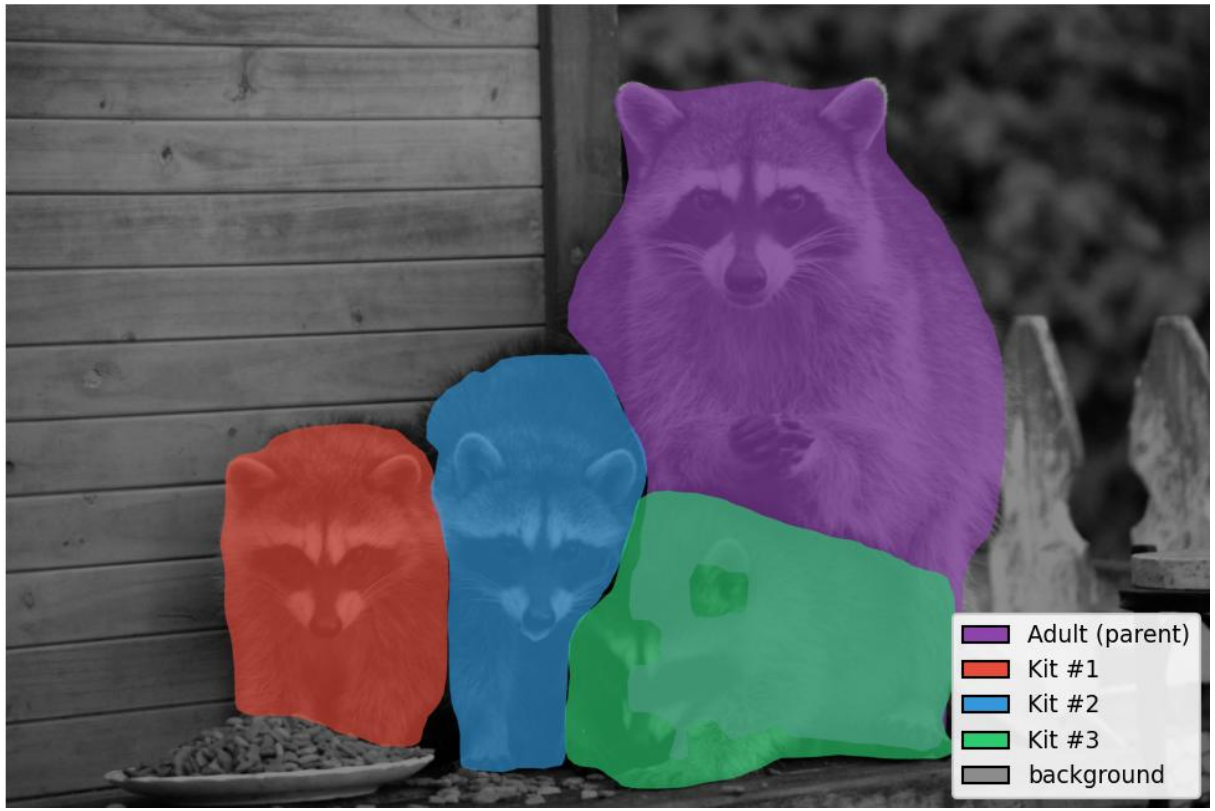


Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

What is Instance Segmentation?

Semantic Segmentation: gives per-pixel labels, but merges instances.

Object Detection: detects individual object instances but only gives boxes.

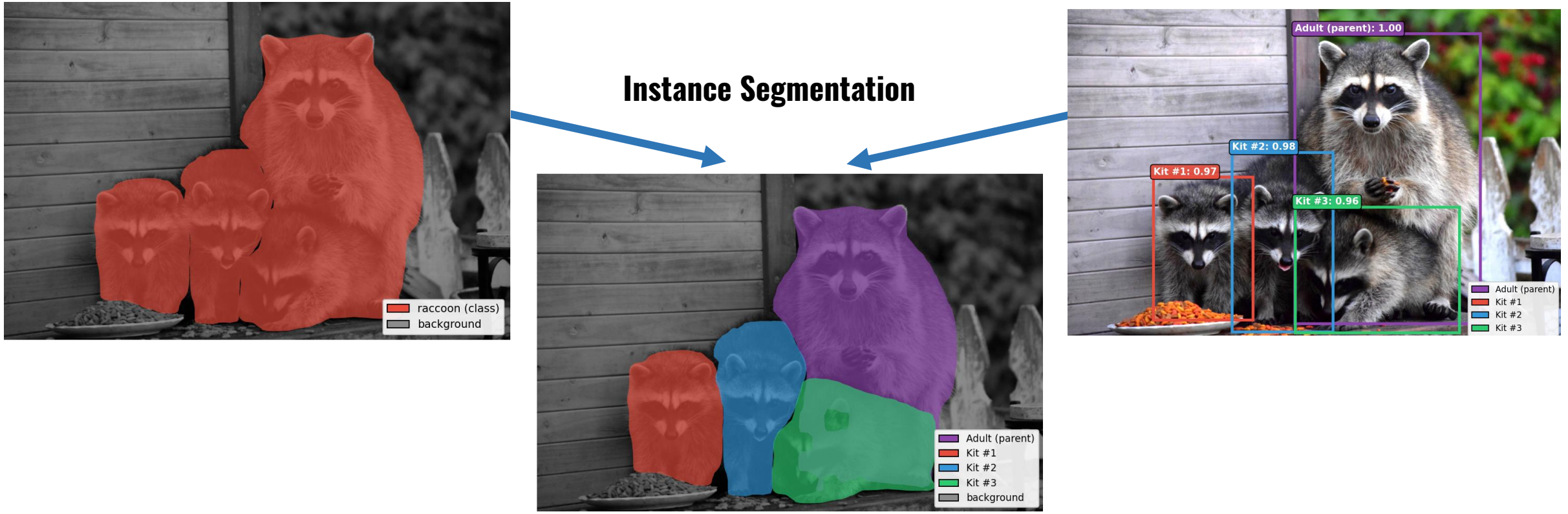


Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

What is Instance Segmentation?

Things and Stuff

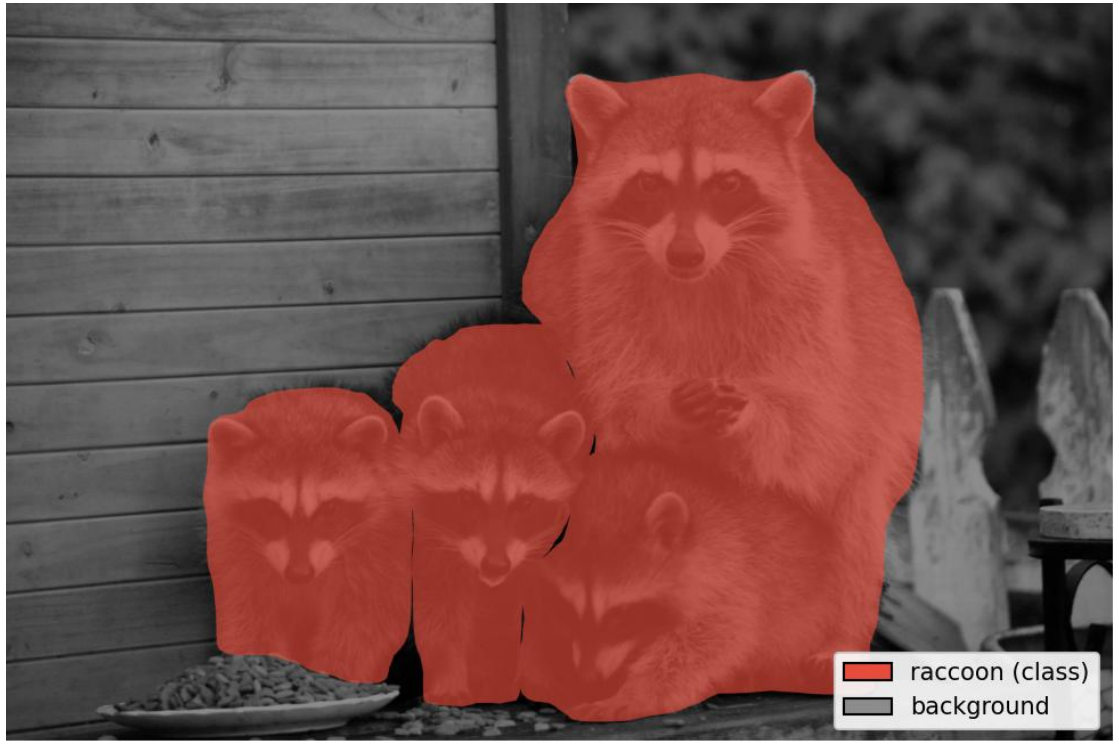
- **Things:** Object categories that can be separated into object instances (e.g. cats, cars, person)
- **Stuff:** Object categories that cannot be separated into instances (sky, grass, water, trees)

- Label all pixels in the image (both things and stuff).
- For “thing” categories also separate into instances.

What is Instance Segmentation?

Semantic Segmentation: detects both objects and regions but doesn't distinguish individual instances.

Instance Segmentation: distinguishes individual object instances, but only for countable objects.



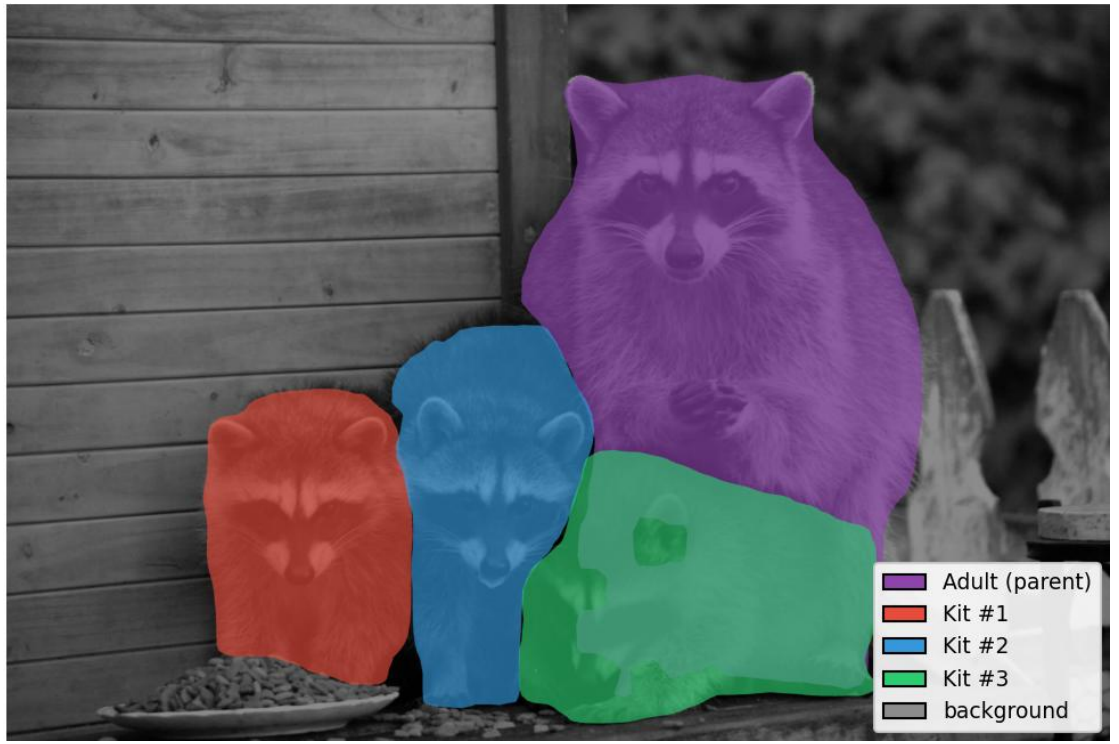


Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

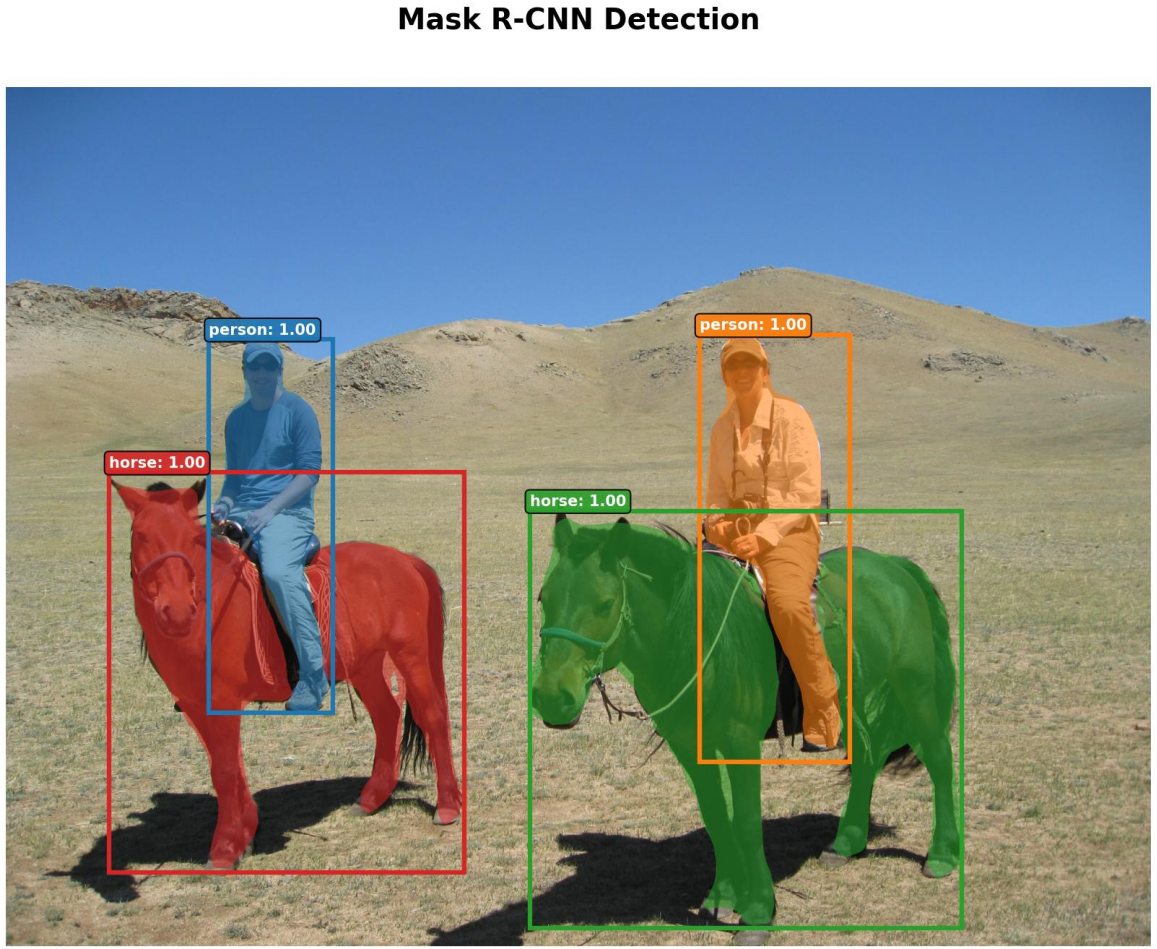
How does Instance Segmentation work?

Instance Segmentation:

- Detect all objects in the image and identify the pixels that belong to each object (Only things!)

Approach:

- Perform object detection, then predict a segmentation mask for each object!



Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic Segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1801.00868
Image Source: Horse Ride.jpg, Chris Feser, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

How does Instance Segmentation work?

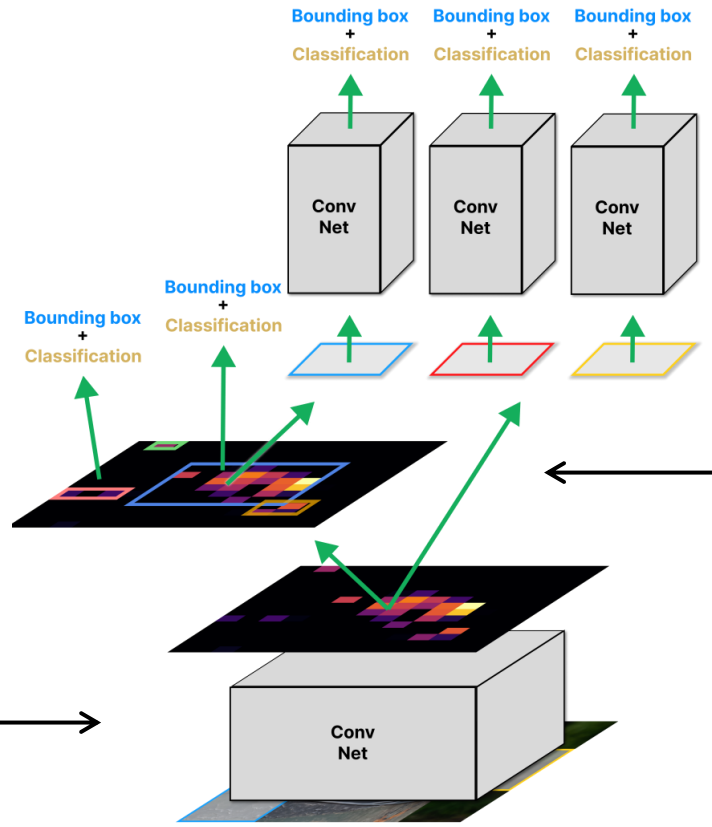
Faster R-CNN: Learnable Region Proposals

Same as Fast R-CNN but a **Region Proposal Network (RPN)** is inserted

The **Region Proposal Network** tells which region to pay attention to.

Classify region for object or not object, and anchor box transformation regression

Pretrained backbone creates the feature map



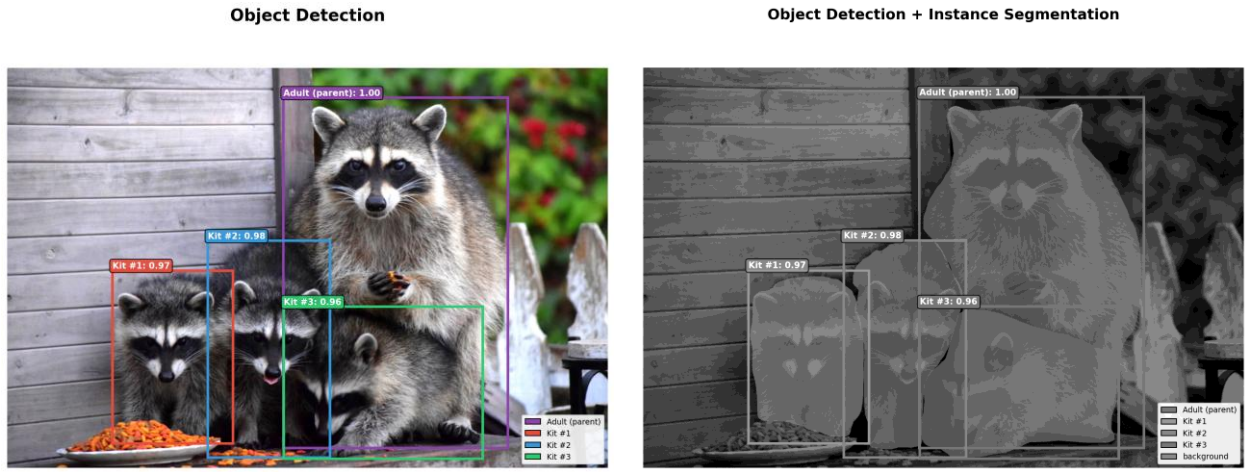
Classify each proposed regions, and box regression

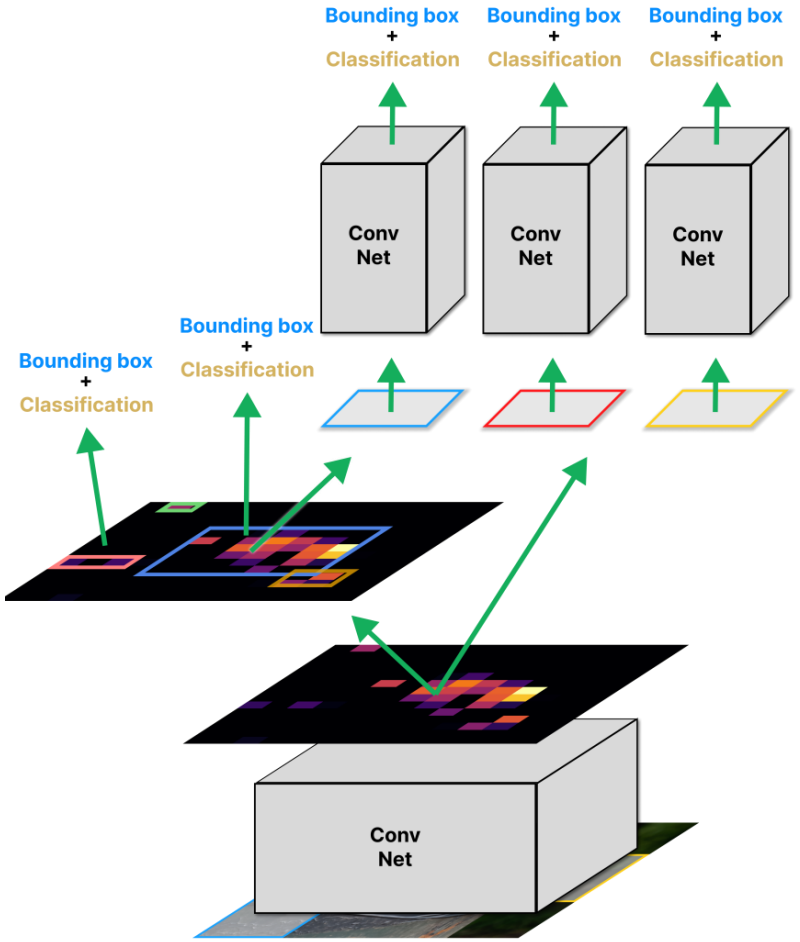
ROI Pooling: crop and summarize features from the proposed regions into a $C \times 7 \times 7$ tensor (split proposed region into a 7x7 grid, pool grid cells)

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1506.01497

How does Instance Segmentation work?

Object Detection: Faster R-CNN

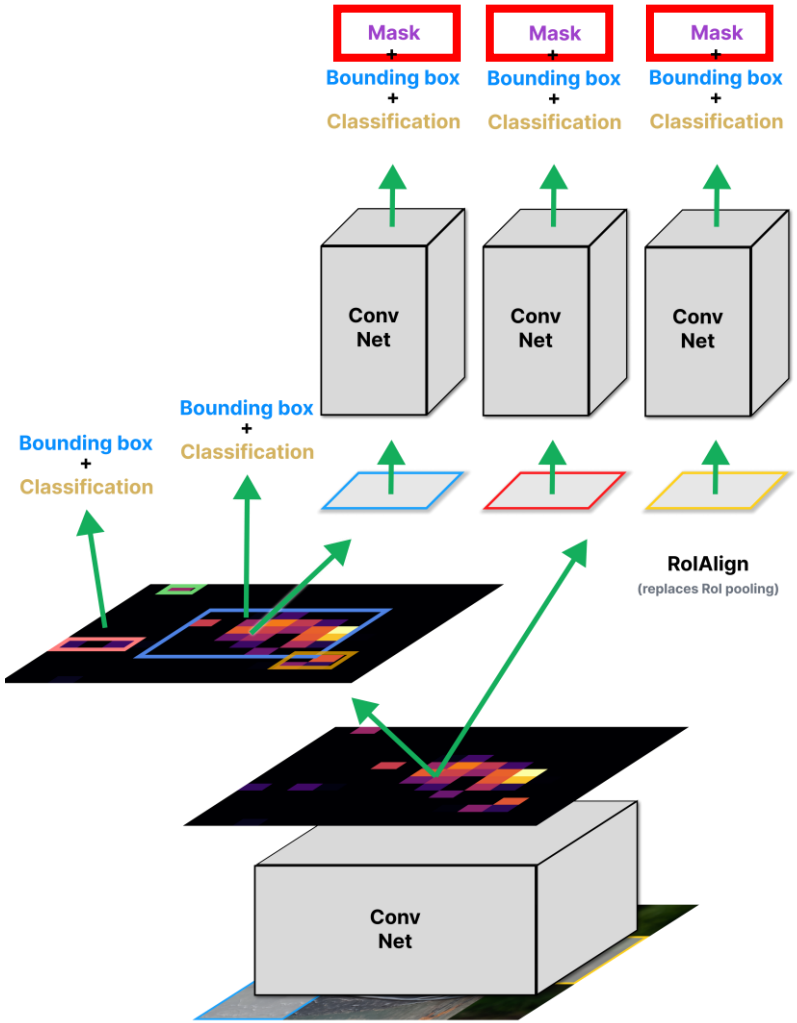




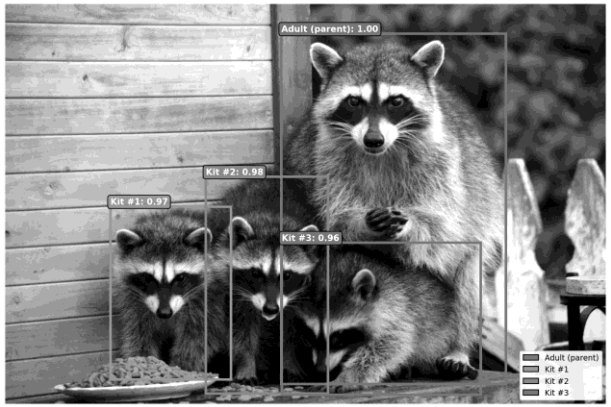
Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1506.01497
Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

How does Instance Segmentation work?

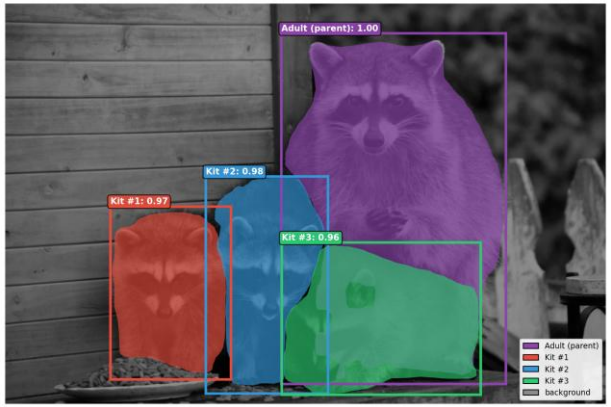
Object Detection: Faster R-CNN



Object Detection



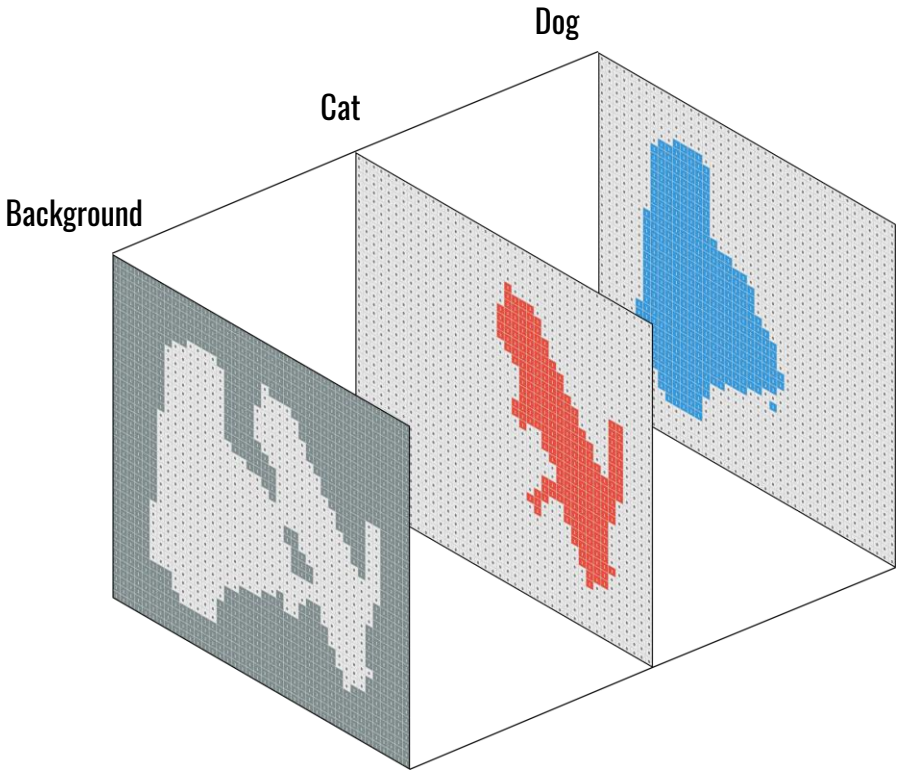
Object Detection + Instance Segmentation

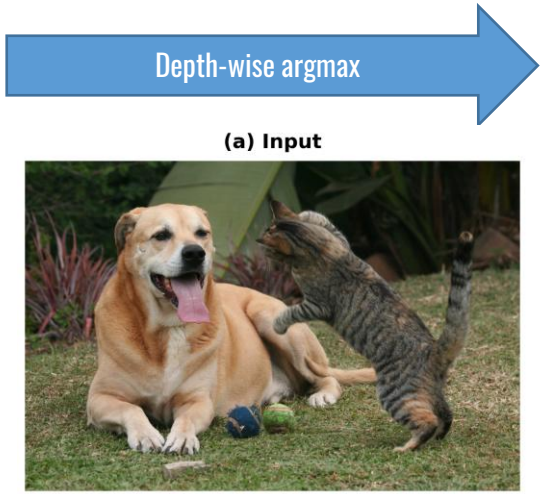


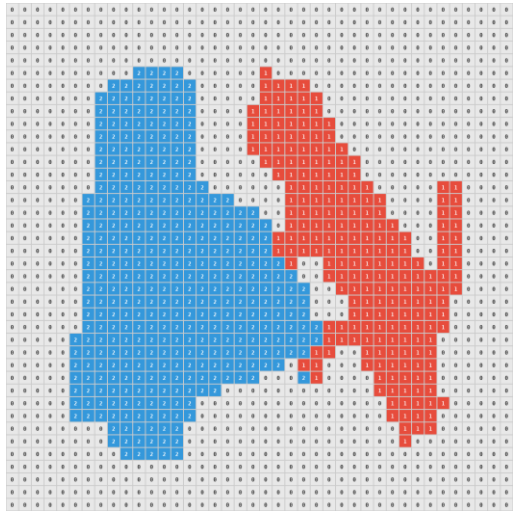
Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1506.01497
Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Segmentation labels (recap)

- Similar to how we treat standard categorical values, we'll create our target by one-hot encoding the class labels - essentially creating an output channel for each of the possible classes.
- A prediction can be collapsed into a segmentation map by taking the argmax of each depth-wise pixel vector.







Semantic Labels

Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Instance Segmentation labels

Training label example: Per-Class Binary Mask

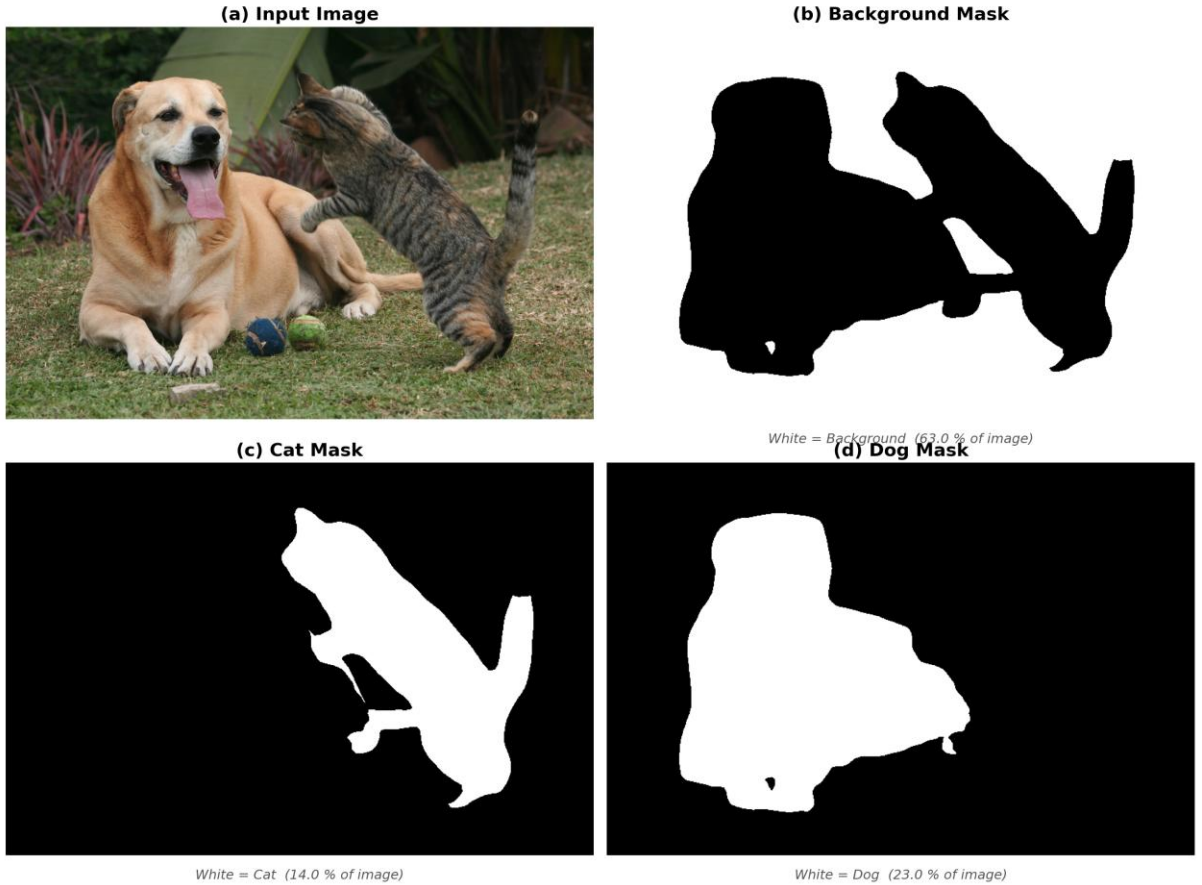


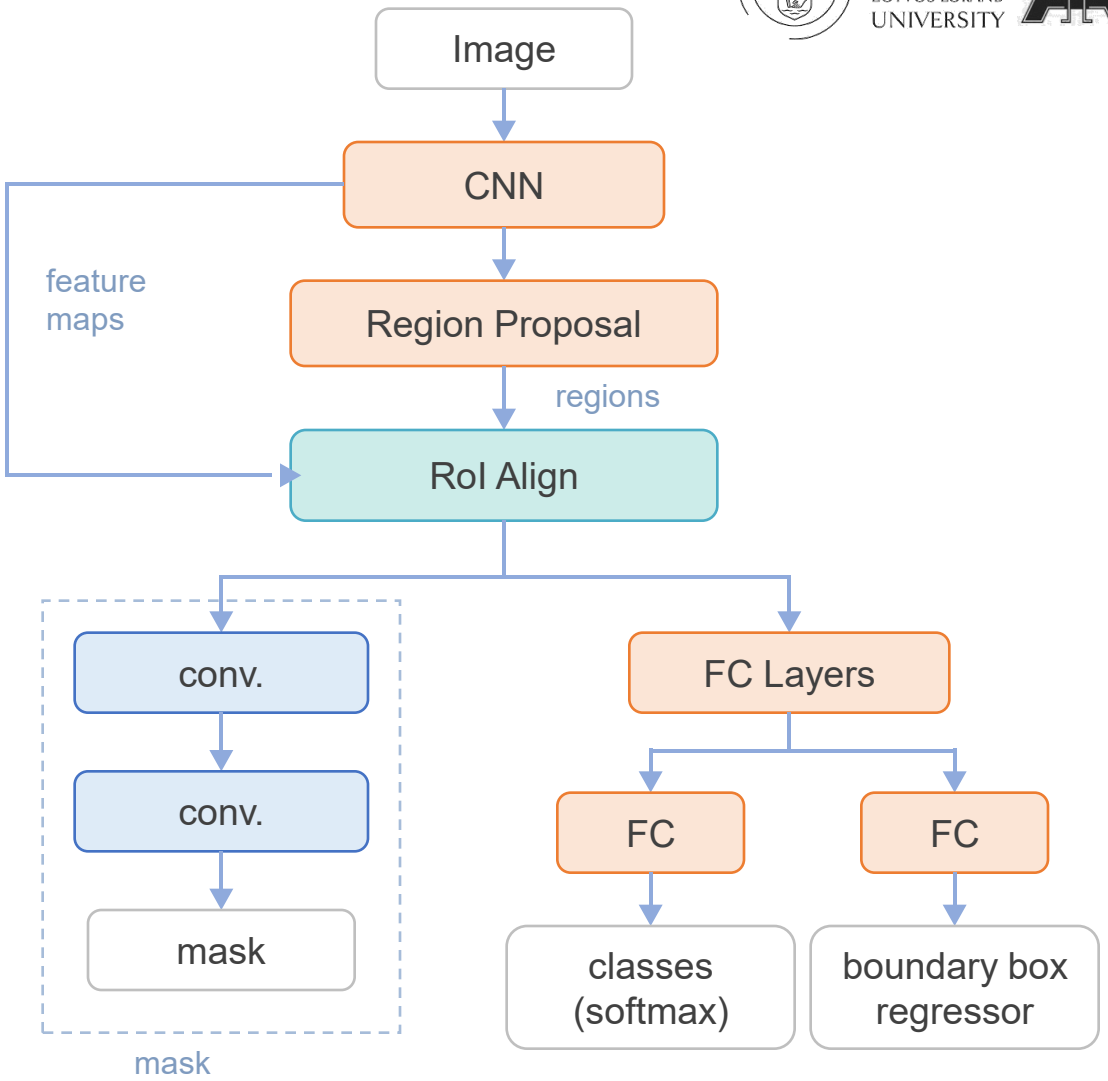
Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Mask R-CNN architecture

(More details in the paper)

What RoI Align is for

- Region proposals land at fractional positions on the feature-map grid.
- RoI Pooling rounds them to whole cells, displacing features by up to a stride and smearing boundaries.
- RoI Align keeps the fractions, sampling each bin with bilinear interpolation — pixel-accurate features, and sharper masks.



Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1506.01497

Supervised Learning tasks

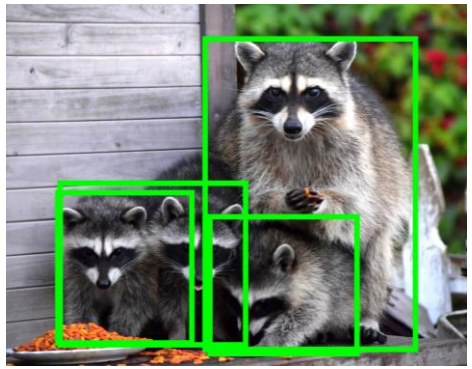
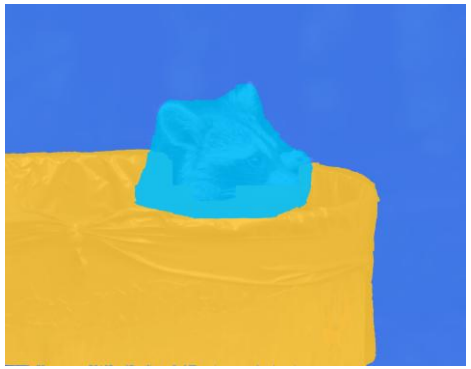
Classification	Classification + Localization	Object Detection	Instance Segmentation	Semantic Segmentation
				
RACCOON	RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, BIN, BACKGROUND
Single Object	Single Object	Multiple Objects	Multiple Objects	No objects, just pixels
✓	✓	✓	✓	✓

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Summary

- **Object Detection** is a supervised learning task
- Goal is to predict what and where are the objects in an image

- (Core) Region proposal methods:
  - **R-CNN:** uses selective search to find regions
  - **Fast R-CNN:** first extracts features and then uses selective search
  - **Faster R-CNN:** uses a learnable region proposal network

- One-stage methods like **YOLO** and **SSD**, remove the region proposal part used in two-stage methods like the R-CNN family

- Many useful applications

- Techniques like **IoU** and **NMS** improve the predictions of object detectors
- **mAP** is a metric to evaluate object detectors

Summary

- **Upsampling** is essential to reconstruct the original image from lower-resolution feature maps.
- By increasing the resolution, **upsampling** enlarges images with the following methods:
  - **Unpooling** upsamples by distributing a single value over higher resolution.
  - **Transpose Convolution** reverses the operation of convolution.

- Object masks are predicted within an image through **Image Segmentation.**
- **Fully Convolutional Networks (FCNs)** serve as encoders for coarse feature maps but struggle with detailed segmentations.
- **U-Net** improves localization by expanding the decoder's capacity for segmentation tasks.
- With **Mask R-CNN**, adding a mask prediction head allows for extended segmentation capabilities.
- **Semantic Segmentation**: Treats all objects of the same class as one, using one-hot encoded class labels.
- **Instance Segmentation**: Identifies individual instances of the same object.

Resources

This presentation is based on: Johnson, J. (2022). EECS 498: Deep Learning for Computer Vision. University of Michigan. https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning

Image Sources

- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.
- Typical cnn.png, Aphex34, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Кошка и собака.jpg, Тулип, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Cat and dog (4984798619).jpg, kitty.green66, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons
- A kitten on the lawn (Pixabay).jpg, Ilona Ilyés, CC0, via Wikimedia Commons
- A cat at Bridge Street, Neston.JPG, Rept0n1x, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
- Pomeranzen.jpg, Neptuul, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- A basket of fruits.jpg, Suyash Dwivedi, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Image Sources (continued)

- ResNet block.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA, via Wikimedia Commons
- Dancing Robot: Gif robot.gif, Meks01, CC BY-SA 4.0, via Wikimedia Commons
- Dropout work diagram, from “A Convolutional Neural Network for Airport Security Inspection of Dangerous Goods”, via ResearchGate, Creative Commons Attribution 3.0 Unported
- Convolution arithmetic visualization, adapted from Vincent Dumoulin & Francesco Visin (2018), A guide to convolution arithmetic for deep learning; original visual assets from the conv_arithmetic GitHub repository https://github.com/vdumoulin/conv_arithmetic/, MIT License
- Horse Ride.jpg, Chris Feser, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons
- Travis McCabe of Team USA (48068764576) (cropped).jpg, Ray Rogers from Novato, United States, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

Papers

1. Girshick, R. (2015). Fast R-CNN. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1504.08083
2. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/cvpr.2014.81
3. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV 2016* (Vol. 9905, pp. 21–37). Springer International Publishing. https://doi.org/10.1007/978-3-319-46448-0_2
4. Long, J., Shelhamer, E., & Darrell, T. (2014). Fully convolutional networks for semantic segmentation. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1411.4038
5. Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., & Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2408.00714
6. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1506.01497
7. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1505.04597

Legal Notice

conv_arithmetic — Vincent Dumoulin & Francesco Visin
Source: https://github.com/vdumoulin/conv_arithmetic
License: The MIT License (MIT)

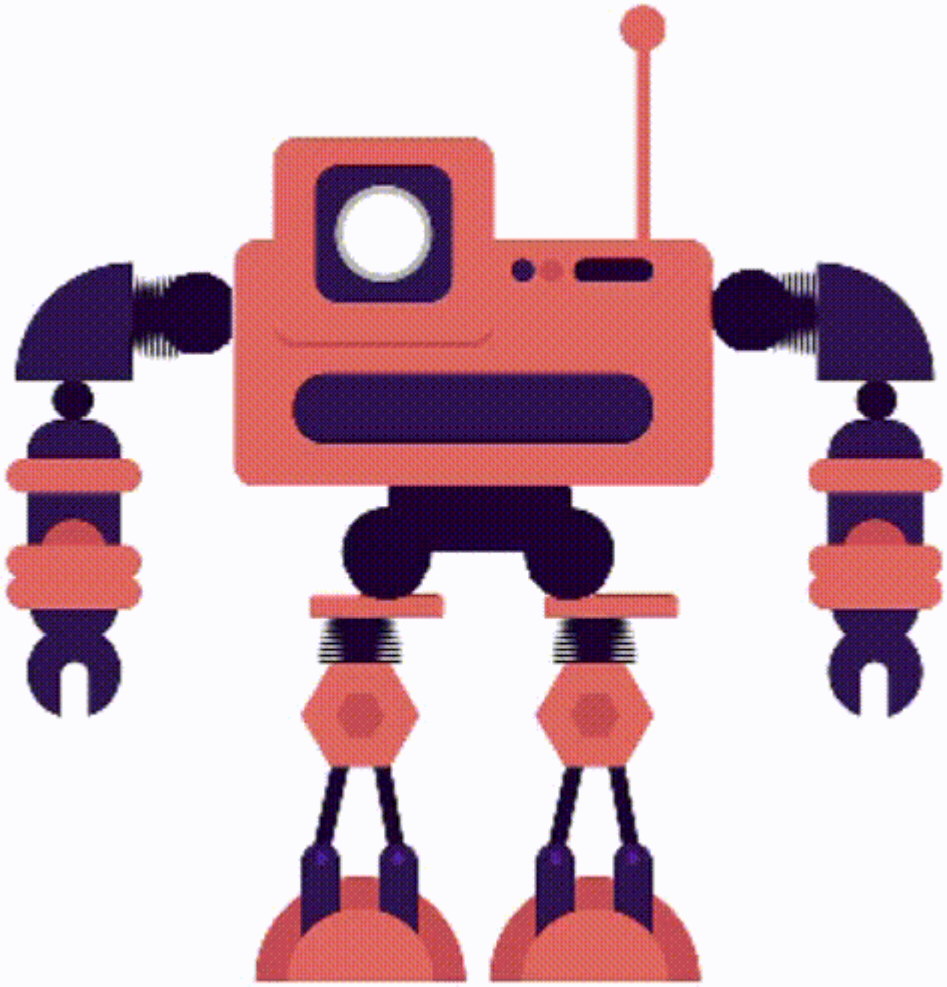
Copyright (c) 2016

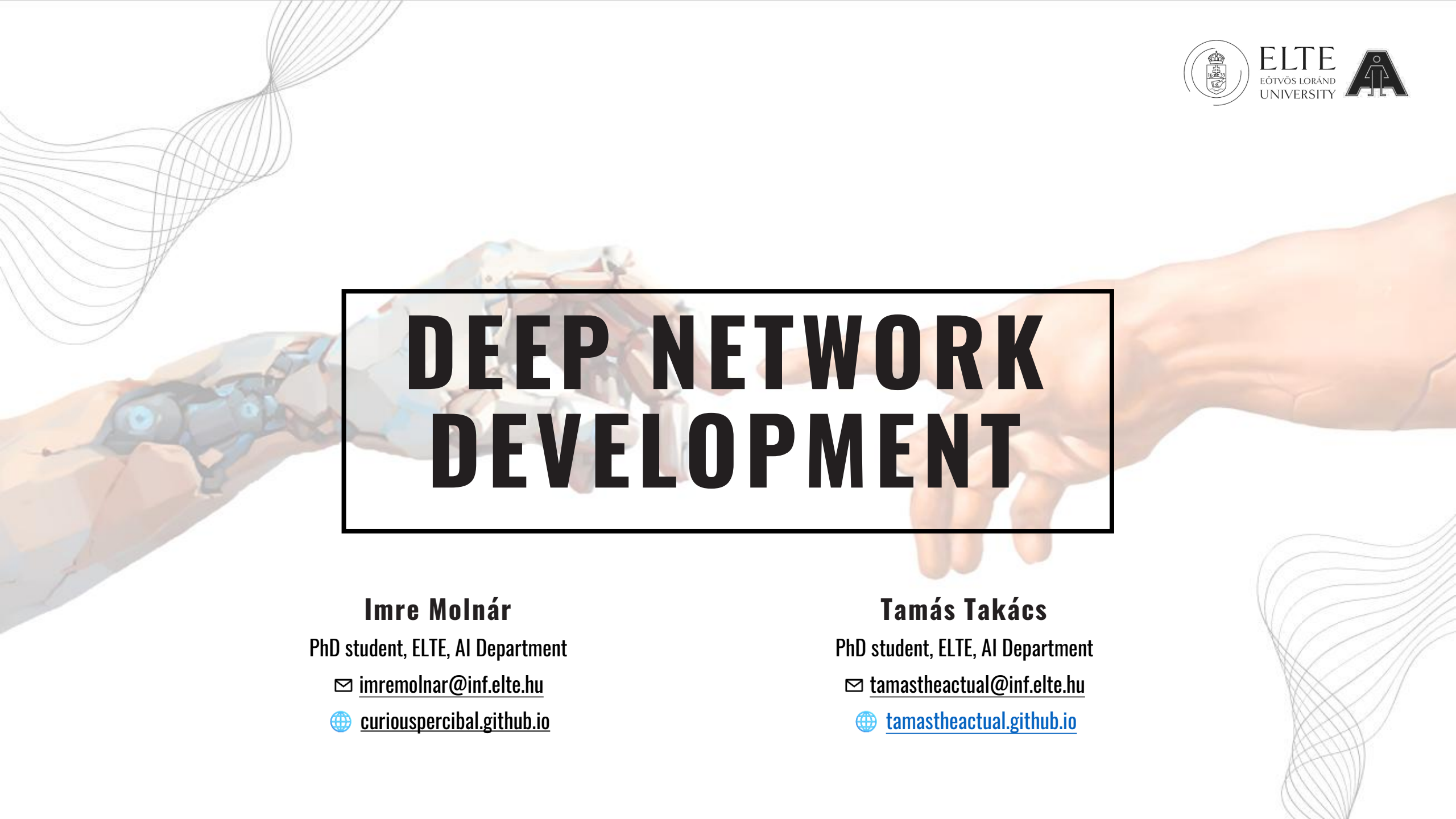
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Supervised Learning tasks

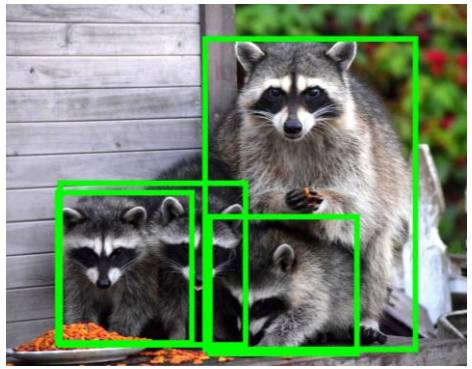
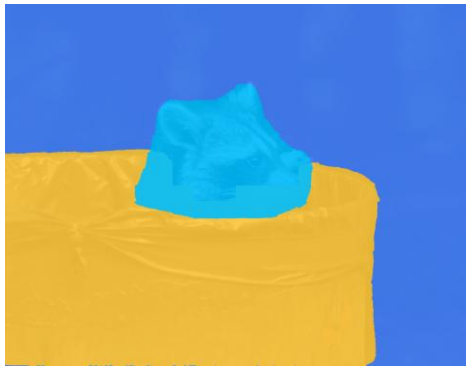
Classification	Classification + Localization	Object Detection	Instance Segmentation	Semantic Segmentation
				
RACCOON	RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, RACCOON, RACCOON, RACCOON	RACCOON, BIN, BACKGROUND
Single Object	Single Object	Multiple Objects	Multiple Objects	No objects, just pixels
✓	✓	✓	✓	✓

Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Deep Network Development

Lecture 6

Natural Language Processing Basics and Recurrent Neural Networks

1 NLP Basics

2 RNN, LSTM, GRU & Seq2Seq

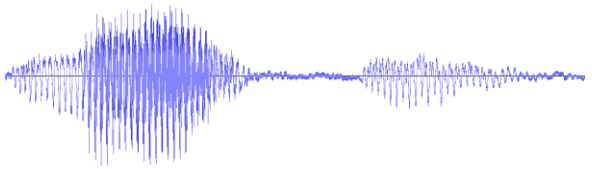
3 Attention Mechanism

Sequential Data Processing

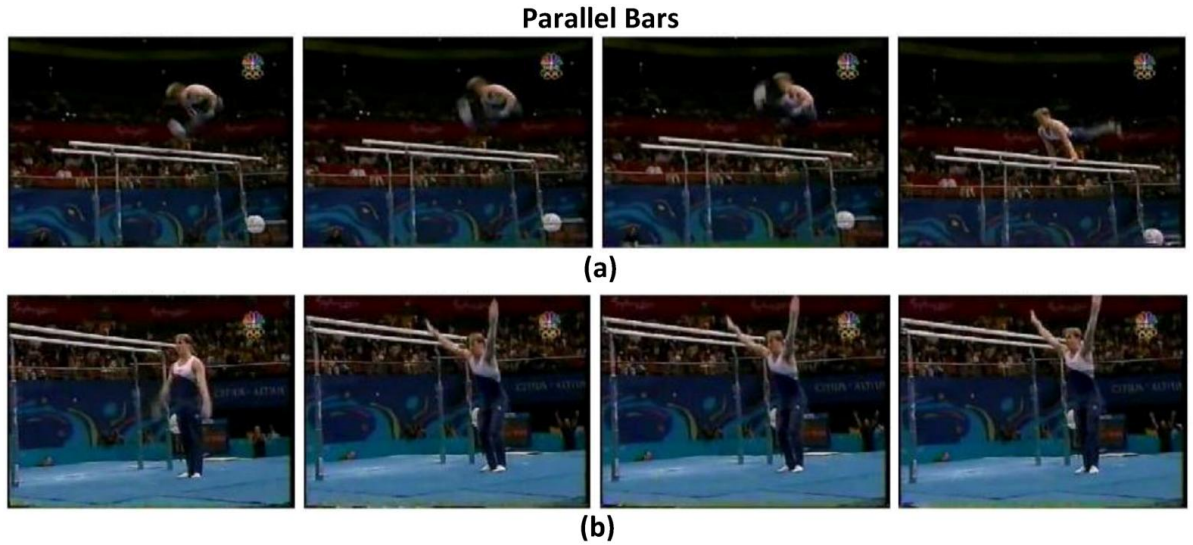
Sequential data

Text – sequence of words / characters

Speech – sequence of signals / acoustic features



Video – sequence of images (frames)



DNA – sequence of symbols (nucleotides)

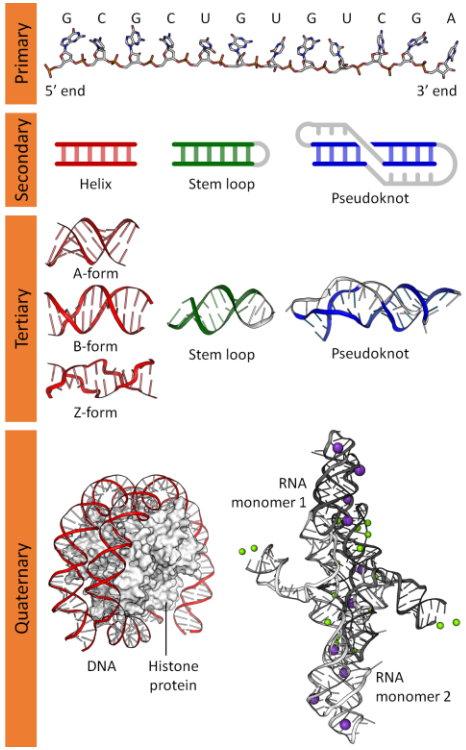


Image Source: DNA RNA structure (full).png, Thomas Shafee, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
Image Source: Signal-speech-martin-de.png, MartinThoma, CC0, via Wikimedia Commons
Image Source: Video frames of the Parallel Bars action category.png, Bikingdog, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Sequential Data Processing

Sequential data carry **temporal information** – Is this car parking or leaving?



Image Source: P3030027ParkingLot wb.jpg, No machine-readable author provided. Elf assumed (based on copyright claims)., CC BY-SA 3.0 <http://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia Commons

Sequential Data Processing

Sequential data carry **context**

BANK



“The bank will lend us money.”



"Let's swim to the opposite bank."

Natural Language Processing Tasks

Named Entity Recognition

- A text corpus tagged with 4 different entity types
  - PER (persons), LOC (locations), ORG (organization), MISC (miscellaneous)

Index	0	1	2	3	4	5	6	7	8
Input X:	Harry	Potter	and	Hermione	Granger	invented	a	new	spell
Tags:	B-PER	I-PER	O	B-PER	I-PER	O	O	O	O
Output Y:	1	2	0	1	2	0	0	0	0

Index	0	1	2	3	4	5	6	7	8	9	10
Input X:	The	European	Commission	said	on	Thursday	it	disagreed	with	German	advice
Tags:	O	B-ORG	I-ORG	O	O	O	O	O	O	B-MISC	O
Output Y:	0	3	4	0	0	0	0	0	0	7	0

Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition](https://aclanthology.org/W03-0419/) (Tjong Kim Sang & De Meulder, CoNLL 2003)

Natural Language Processing Tasks

Question Answering

- Stanford Question Answering Dataset (SQuAD)
- **Context**
- **Question** – **Answer** pairs
- The answer is a span in a given Wikipedia paragraph

Context

The first aircraft, a South African Beechcraft King Air 200, landed at the new airport on 15 September 2015, prior to conducting a series of flights to calibrate the airport's radio navigation equipment..

Question Answer Pairs

1. **Question:** What kind of aircraft was the first to land at the new airport?
   Answer: South African Beechcraft King Air 200
2. **Question:** What date did the aircraft land at the new airport?
   Answer: 15 September 2015
3. **Question:** What are they calibrating via test flights at the airport?
   Answer: the airport's radio navigation equipment

SQuAD: 100,000+ Questions for Machine Comprehension of Text](https://aclanthology.org/D16-1264/) (Rajpurkar et al., EMNLP 2016)

Natural Language Processing Tasks

Textual Entailment

Stanford Natural Language Inference (SNLI)

Determining whether a "**hypothesis**" is true, given a "**premise**".

Five different annotator – **Gold label**

Contradiction - the hypothesis cannot be true given the premise.

Neutral - the hypothesis might be true or false; it's not entailed or contradicted.

Entailment - the hypothesis must be true given the premise.

Text	Judgements	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction C C C C C	The man is sleeping
An older and younger man smiling.	neutral N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	contradiction C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	entailment E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	neutral N N E C N	A happy woman in a fairy costume holds an umbrella.

A large annotated corpus for learning natural language inference](https://aclanthology.org/D15-1075/) (Bowman et al., EMNLP 2015)

Natural Language Processing Tasks

Machine Translation

- The name speaks for itself
- BUT, **input – output** pairs in **source – target** language.
- For one input there might be **multiple candidate** translations

English	French
It's cold.	C'est froid.
Stay calm.	Reste tranquille.
Stop that.	Arrêtez ça !
I will submit the assignment tonight	Je rendrai le devoir ce soir.

Europarl: A Parallel Corpus for Statistical Machine Translation](https://aclanthology.org/2005.mtsummit-papers.11/) (Koehn, MTSummit 2005)

Other Sequence Processing Tasks

- Speech recognition
- Named entity recognition
- Sentiment Analysis
- Semantic role labelling
- Conference resolution
- Etc.

Sequential Data Processing

Representing words as One hot vectors

Dataset: (X,Y)

Vocabulary: [a, aron, ..., harry, ..., potter, ..., zulu]

Position: 1, 2, ..., 4075, ..., 6883, ..., 10000

Index	0	1	2	3	4	5	6	7	8
Input X:	Harry	Potter	and	Hermione	Granger	invented	a	new	spell
Position:	4075	6883	2	...	...	...	1	...	...
Representation	$\begin{bmatrix}0\\0\\0\\\vdots\\1\\0\\\vdots\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\\vdots\\0\\1\\\vdots\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\\\vdots\\0\\0\\\vdots\\0\end{bmatrix}$	...	...	...	$\begin{bmatrix}1\\0\\0\\\vdots\\\vdots\\\vdots\\0\end{bmatrix}$	...	...

Embeddings

Representing words as One hot vectors

Input X: My favorite sport is football.

$X^{<1>}$ $X^{<2>}$ $X^{<3>}$ $X^{<4>}$ $X^{<5>}$

Vocabulary: [favorite, football, is, my, sport]

Position: 1 2 3 4 5

Representation:

Football = [0, 1, 0, 0, 0]

Position: 1 2 3 4 5

Sport = [0, 0, 0, 0, 1]

Position: 1 2 3 4 5

Problems

- Scalability - huge vector for each word
  - If we have a dataset of several sentences, from which we form a vocabulary of 10 000 words.
  - Each word would be represented as a 10 000 long vector, having a single element set to 1. $X^{<1>}$ = [0 , 0, ..., 1, ..., 0, 0]
- There is **no relationship between words**. Each word is treated as an independent entity with no similarity to other words.

Embeddings: Word representation

Featurized representation: Word Embedding

Vocabulary size: 10 000

Vocabulary: [a, ..., apple, ..., football, ..., man, ..., orange, ..., sport, ..., woman, ..., zulu]

Position: 1 456 2078 5391 6257 7301 9853 10 000

	Man (5391)	Woman (9853)	King (4914)	Queen (7151)	Apple (456)	Orange (6257)
Gender	**-1**	**1**	**-0.95**	**0.97**	0.00	0.01
Royal	0.01	0.02	**0.93**	**0.95**	-0.01	0.00
Age	0.03	0.02	**0.7**	**0.68**	0.03	-0.02
Food	0.04	0.01	0.02	0.01	**0.95**	**0.97**
...	...	...	...	...	...	...

Embeddings: Word representation

Featurized representation: Word Embedding

Man (5391) instead of being represented a one hot encoded vector [0,0,...,1,...,0,0] would be represented as:

e_{5391}=[-1, 0.01, 0.03, 0.04, ...]

Embedding Matrix **X** one hot man = embedding man

(# features , vocab size) (vocab size, 1) (# features, 1)

	Man (5391)	Woman (9853)	King (4914)	Queen (7151)	Apple (456)	Orange (6257)
Gender	**-1**	**1**	**-0.95**	**0.97**	0.00	0.01
Royal	0.01	0.02	**0.93**	**0.95**	-0.01	0.00
Age	0.03	0.02	**0.7**	**0.68**	0.03	-0.02
Food	0.04	0.01	0.02	0.01	**0.95**	**0.97**
...	...	...	...	...	...	...

Embeddings: Word representation

Featurized representation: Word Embedding

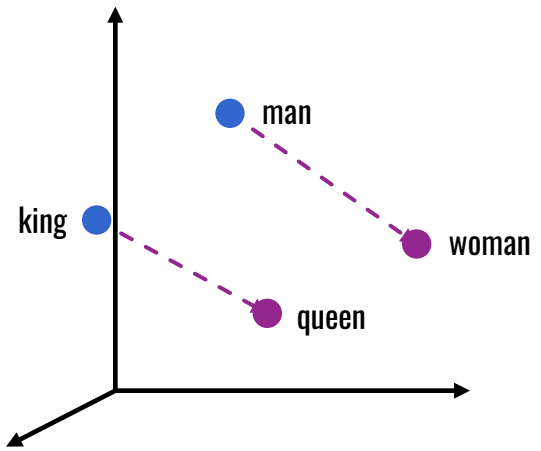
If we subtract man and woman, main difference is gender

We can compute word similarities

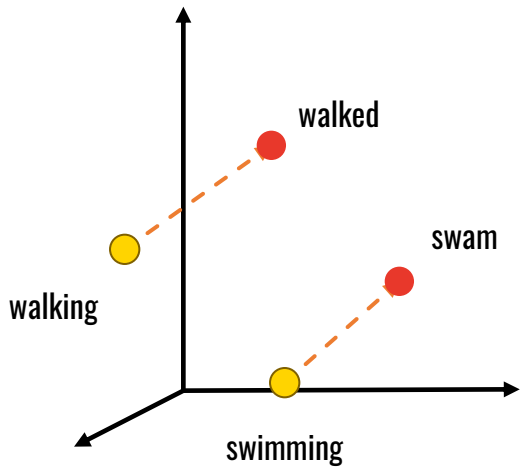
We can compute word analogies: man is to woman as king is to

https://vectors.nlpl.eu/explore/embeddings/en/calculator/

	Man (5391)	Woman (9853)	King (4914)	Queen (7151)	Apple (456)	Orange (6257)
Gender	**-1**	**1**	**-0.95**	**0.97**	0.00	0.01
Royal	0.01	0.02	**0.93**	**0.95**	-0.01	0.00
Age	0.03	0.02	**0.7**	**0.68**	0.03	-0.02
Food	0.04	0.01	0.02	0.01	**0.95**	**0.97**
…	…	…	…	…	…	…



Male-Female



Verb tense

Embeddings

How to create better embeddings?

- We learn the cooccurrences of the words (they appear in similar contexts)
- Let's create input – output pairs as follows
- Given an example sentence

"A person is playing with a cat on the couch."

- We create a list of words that occurs around the cat
- Then we use the one-hot encoding of the words

Skip-grams

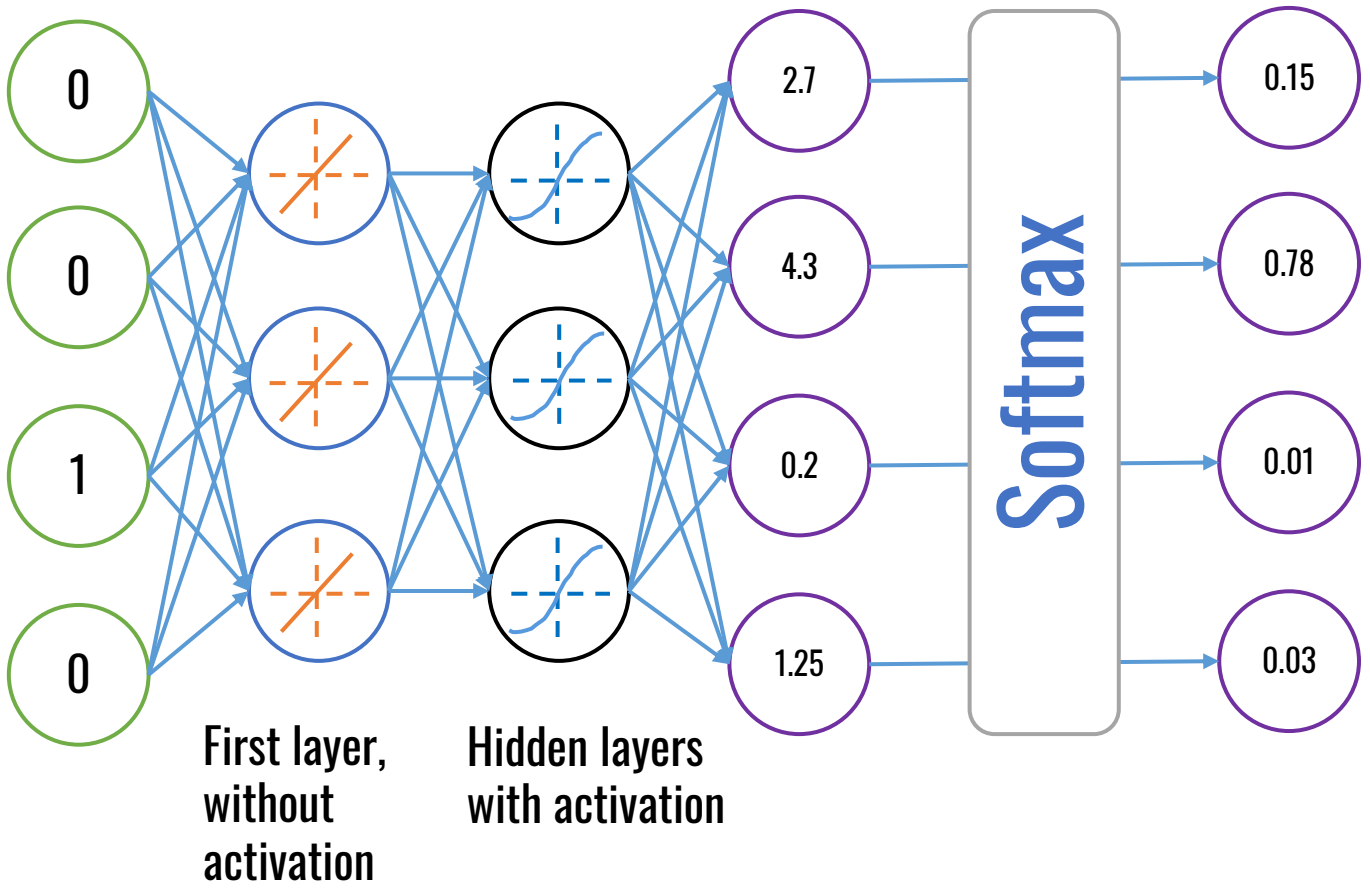
Input	Output
Cat	On
Cat	The
…	…
Cat	Couch
Cat	Person
Cat	Playing

Embeddings

How to create better embeddings?

Compare output with the Outputs (**Person**) one-hot encoded vector

Cat one-hot encoded vector

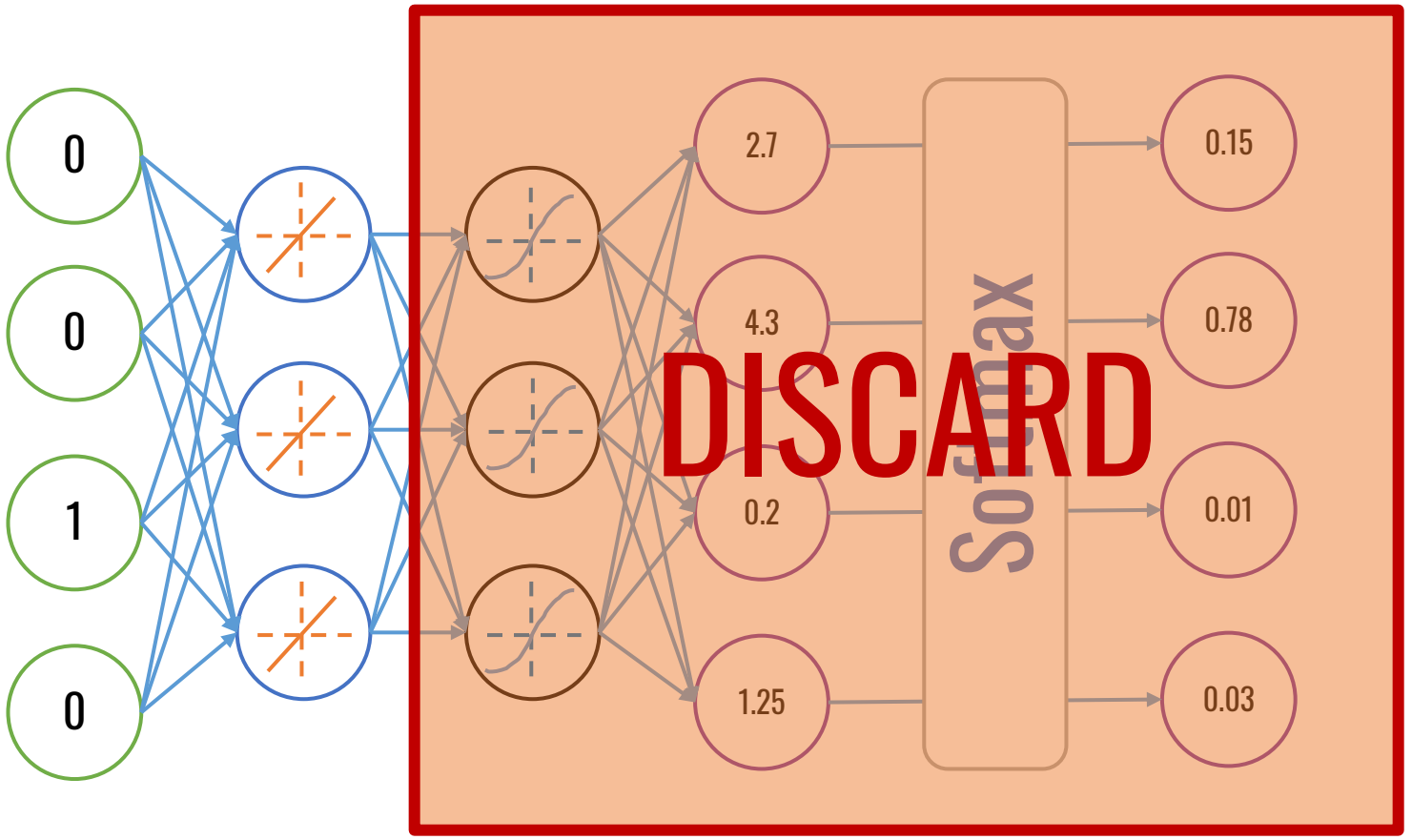


Input	Output
Cat	On
Cat	The
...	...
Cat	Couch
Cat	Person
Cat	Playing

Embeddings

How to create better embeddings?

- Keep only the first layer (thus creating a linear projection of the input vector)



ELTE EÖTVÖS LORÁND UNIVERSITY

Embeddings: Word representation

- **Featurized representation: Word Embedding**
- The word embeddings are learned with training.
- Therefore, in practice, the features aren't that understandable.
- We can visualize lower representations of the embeddings with techniques such as T-SNE

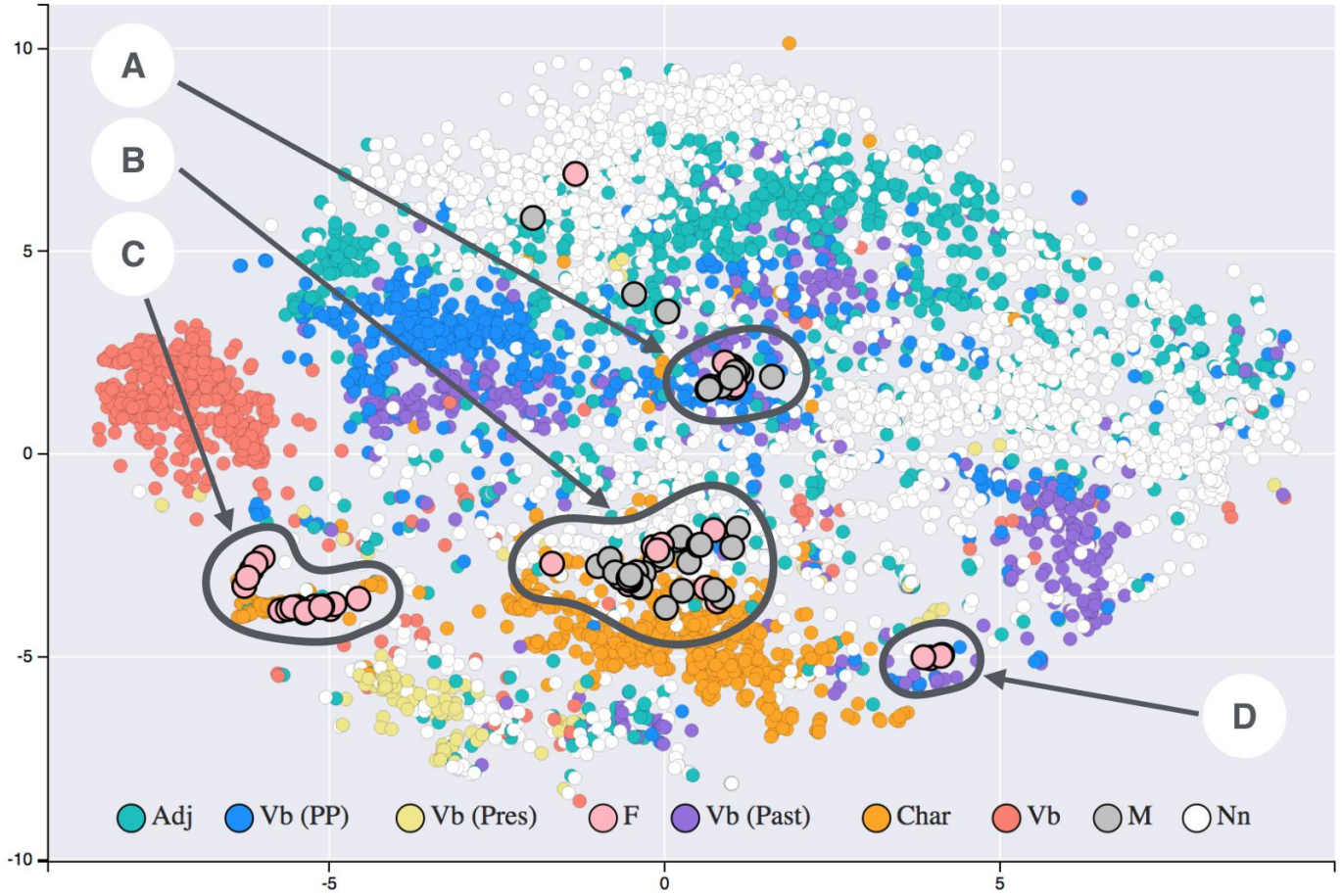


Image Source: T-SNE visualisation of word embeddings generated using 19th century literature.png, Siobhán Grayson, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Embeddings: Word representation

Embeddings can be used to represent other types of data:

- Speech: speaker embedding
- Image Embedding
- Document Embedding
- Etc.

T-SNE visualization of the embeddings of MNIST

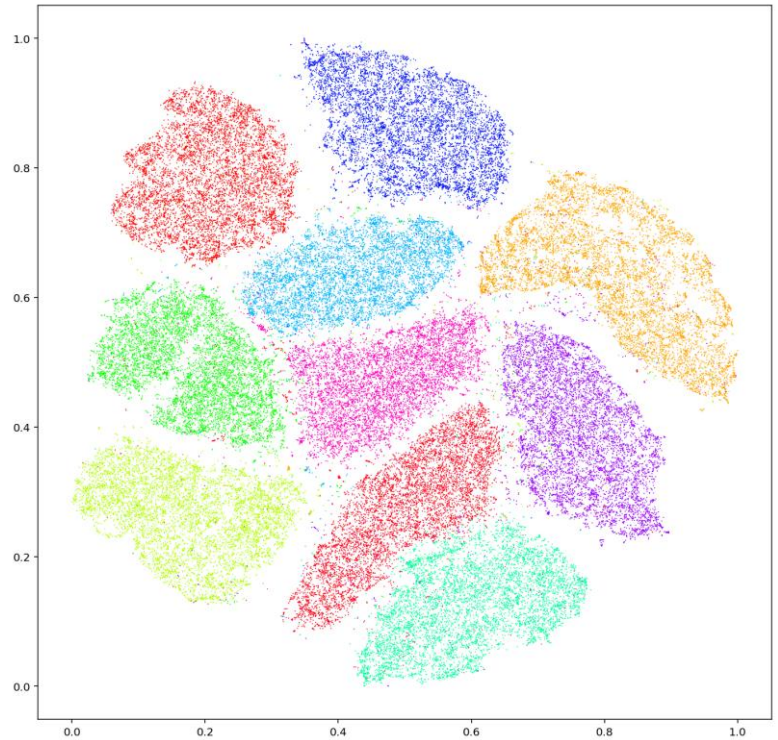


Image Source: T-SNE Embedding of MNIST.png, Kyle McDonald, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

Deep Network Development

Lecture 6

Natural Language Processing Basics and Recurrent Neural Networks

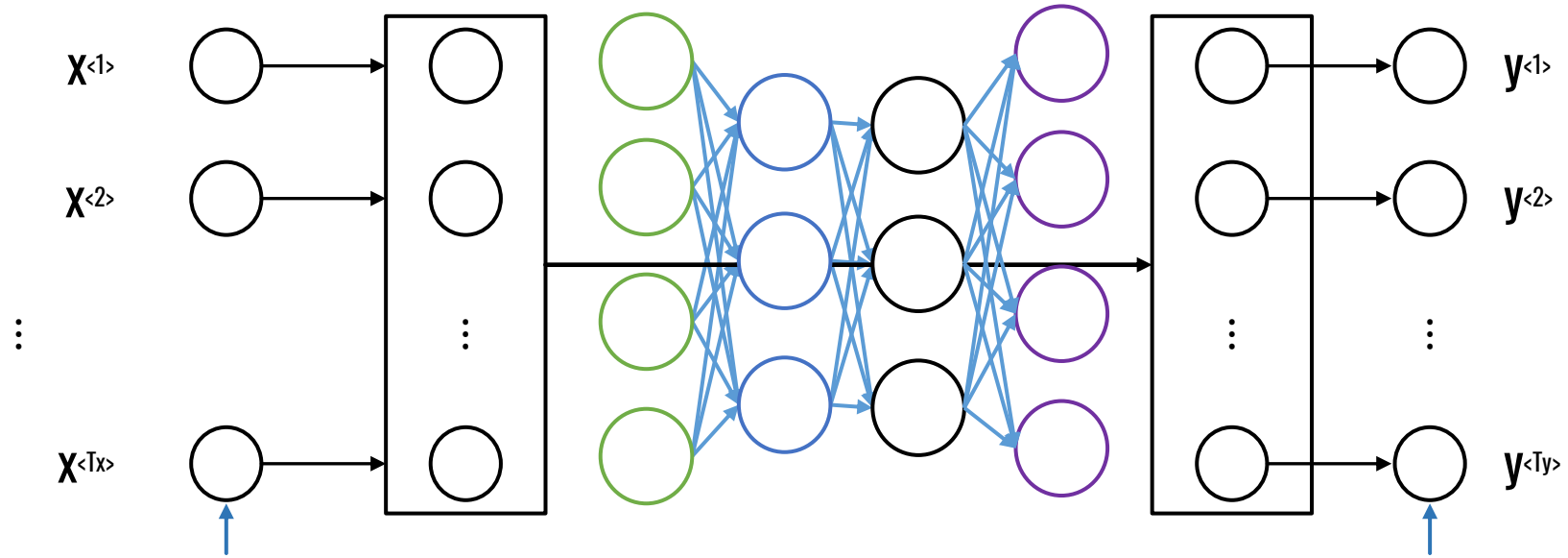
1 NLP Basics

2 RNN, LSTM, GRU & Seq2Seq

3 Attention Mechanism

Introduction to Recurrent Neural Network

Why not a standard network?



Problems:

- Inputs, outputs can be different lengths in different examples
- Doesn't share features learned across different positions of text

Introduction to Recurrent Neural Network

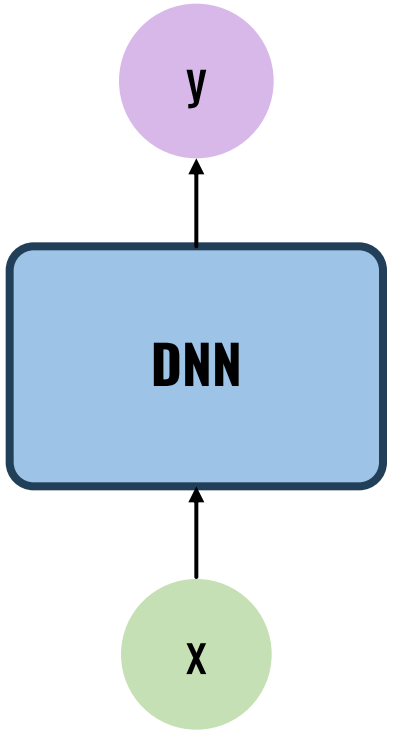
What should we consider?

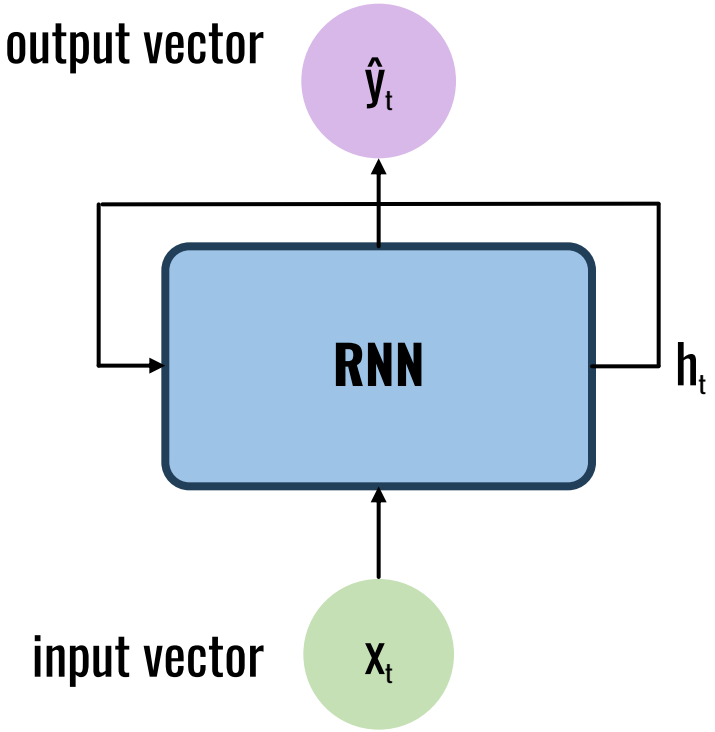
The model needs to:

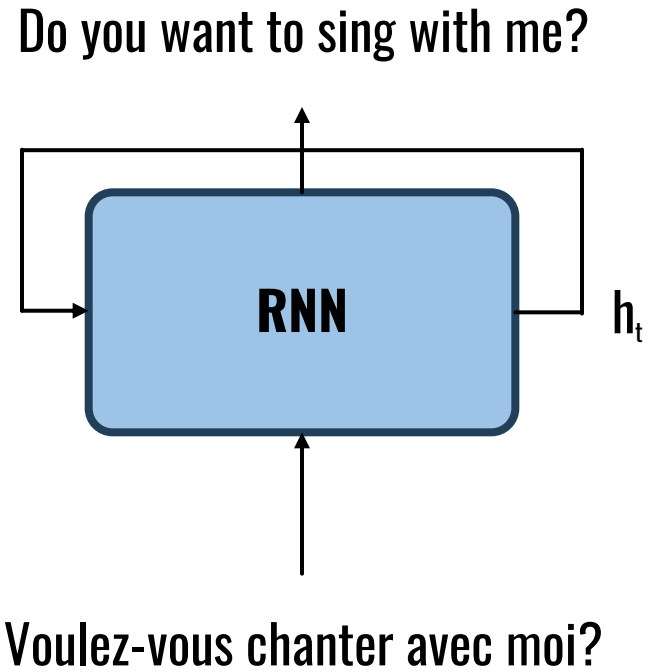
- Handle variable length sequences
- Track long term dependencies
- Maintain the order of the input
- Share parameters across sequences

Introduction to Recurrent Neural Network

What about Recurrent Neural Networks (RNNs)?







Introduction to Recurrent Neural Network

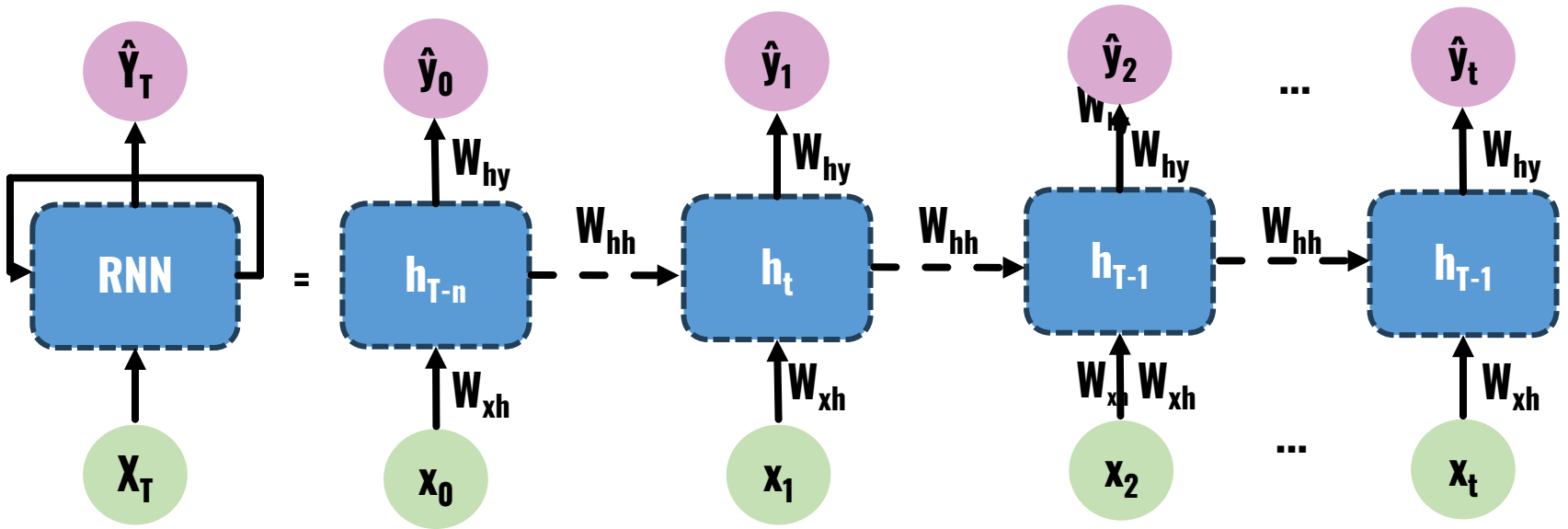
RNN (unrolled version)

Hidden state:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Output vector:

$$\hat{y}_t = \tanh(W_{hy}h_t + b_y)$$



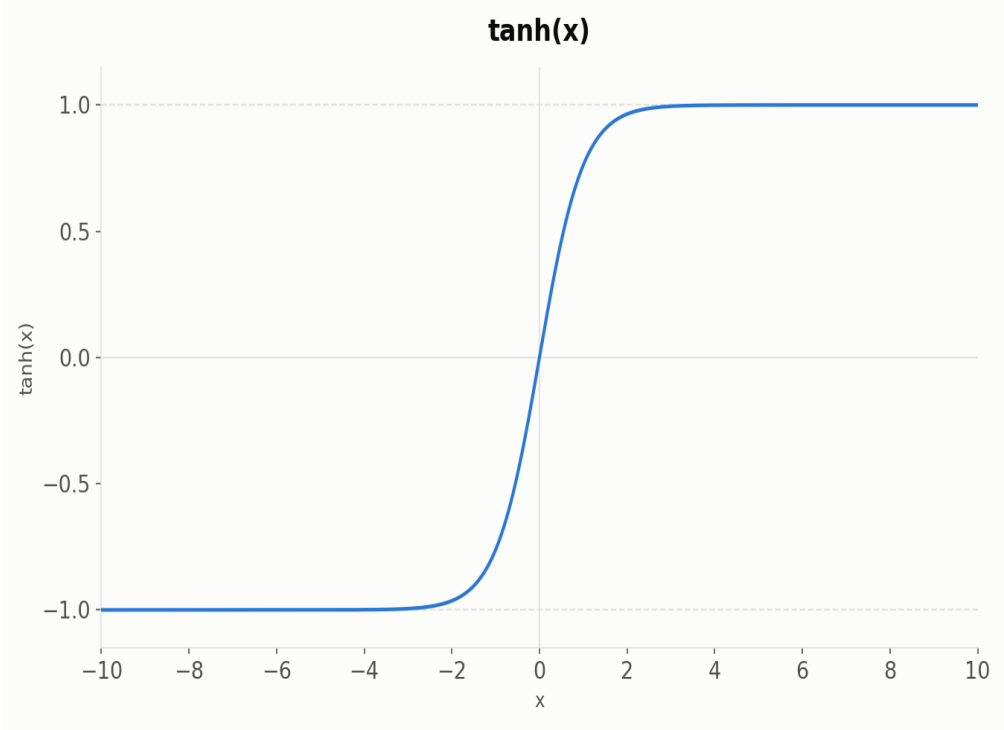
Introduction to Recurrent Neural Network

RECAP: ACTIVATION FUNCTION — TANGENT

Hyperbolic tangent

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$\tanh'(x) = 1 - \tanh^{2}(x)$$



Pros:

- Zero centered.
- Activations are bounded in range [-1,1].
- The gradient is stronger for tanh than sigmoid. Derivatives are steeper.

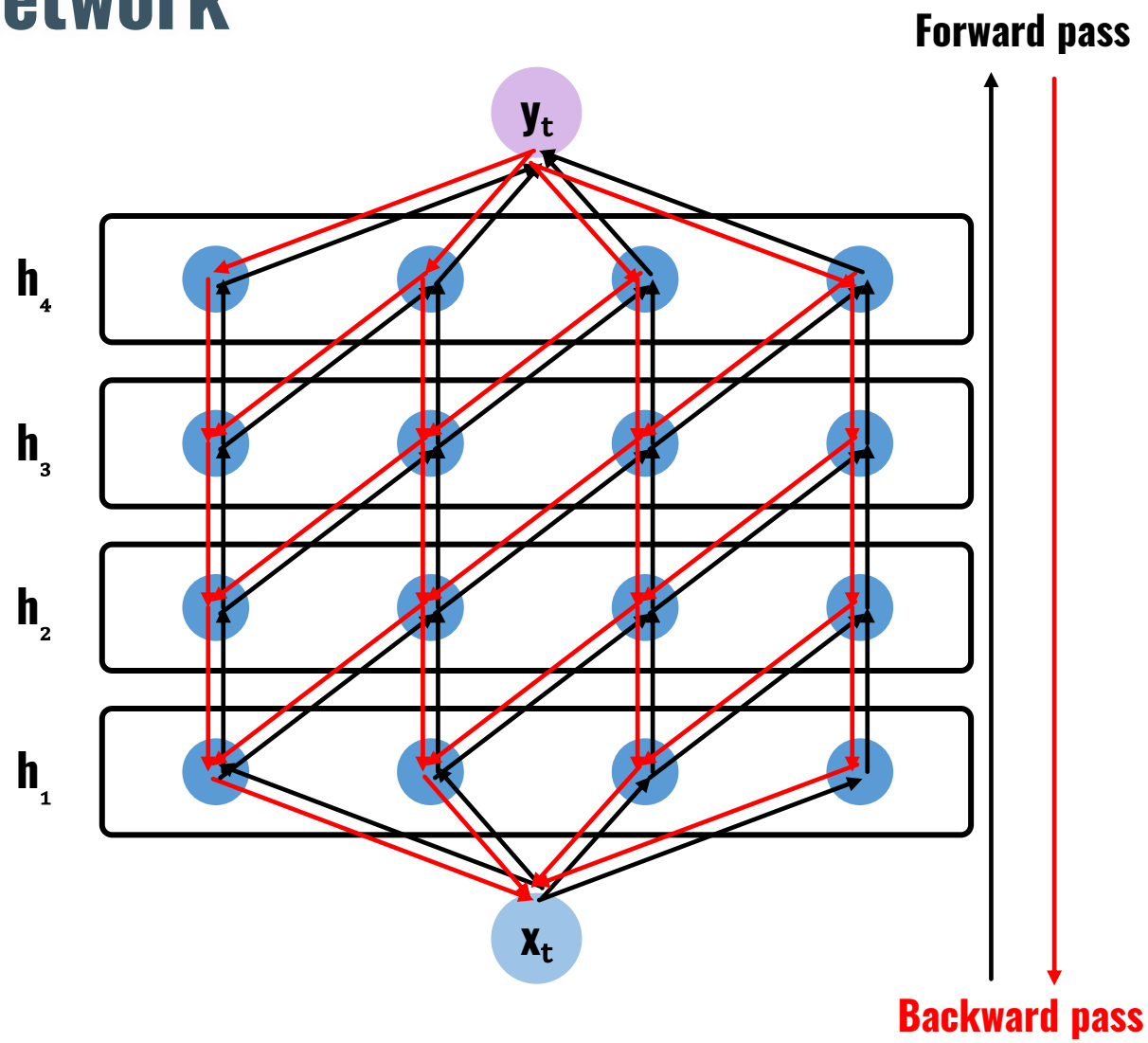
Cons:

- Y values tend to respond very less to changes in X towards either end of the function.
- Vanishing gradients.

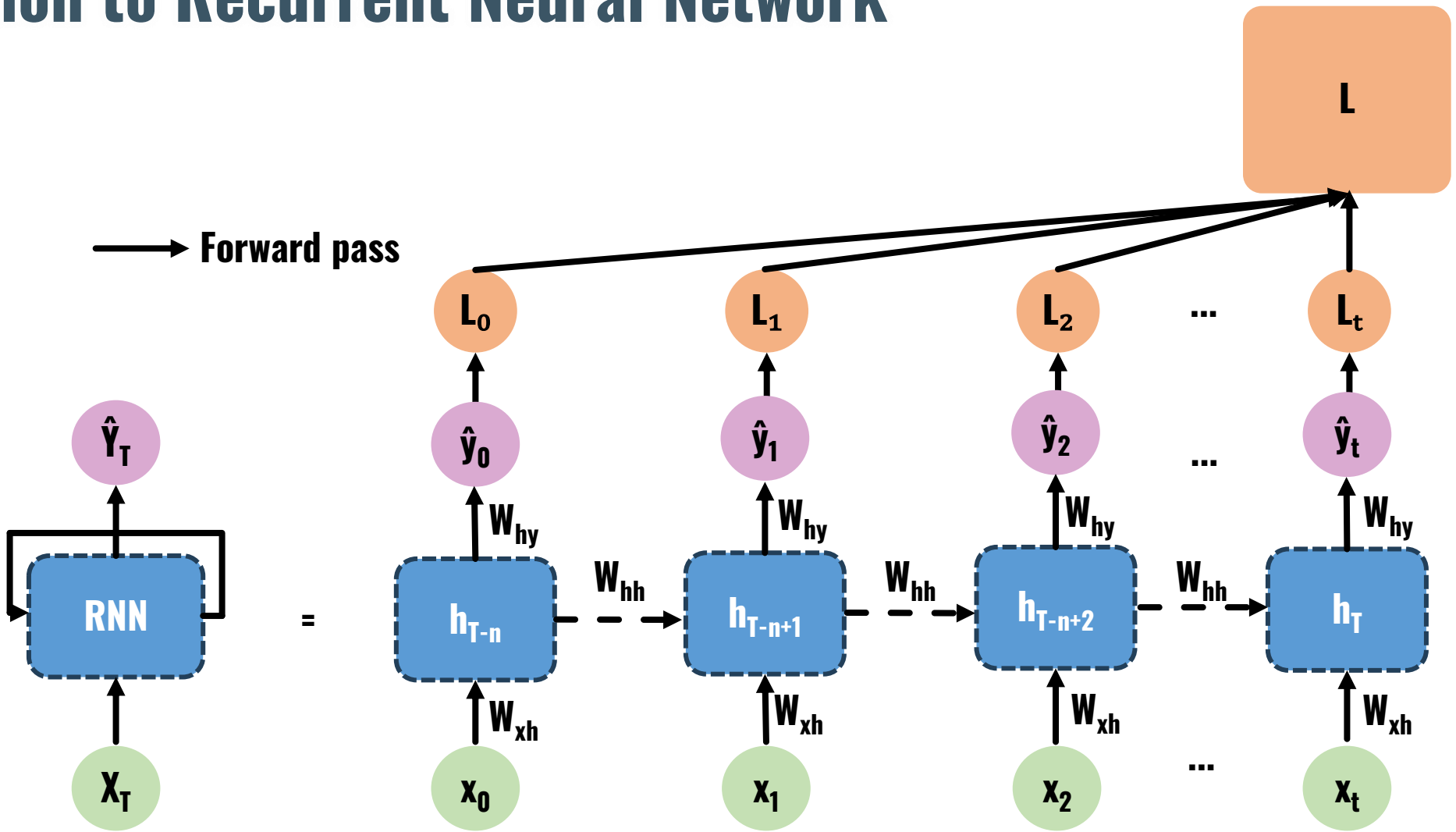
Introduction to Recurrent Neural Network

Backpropagation

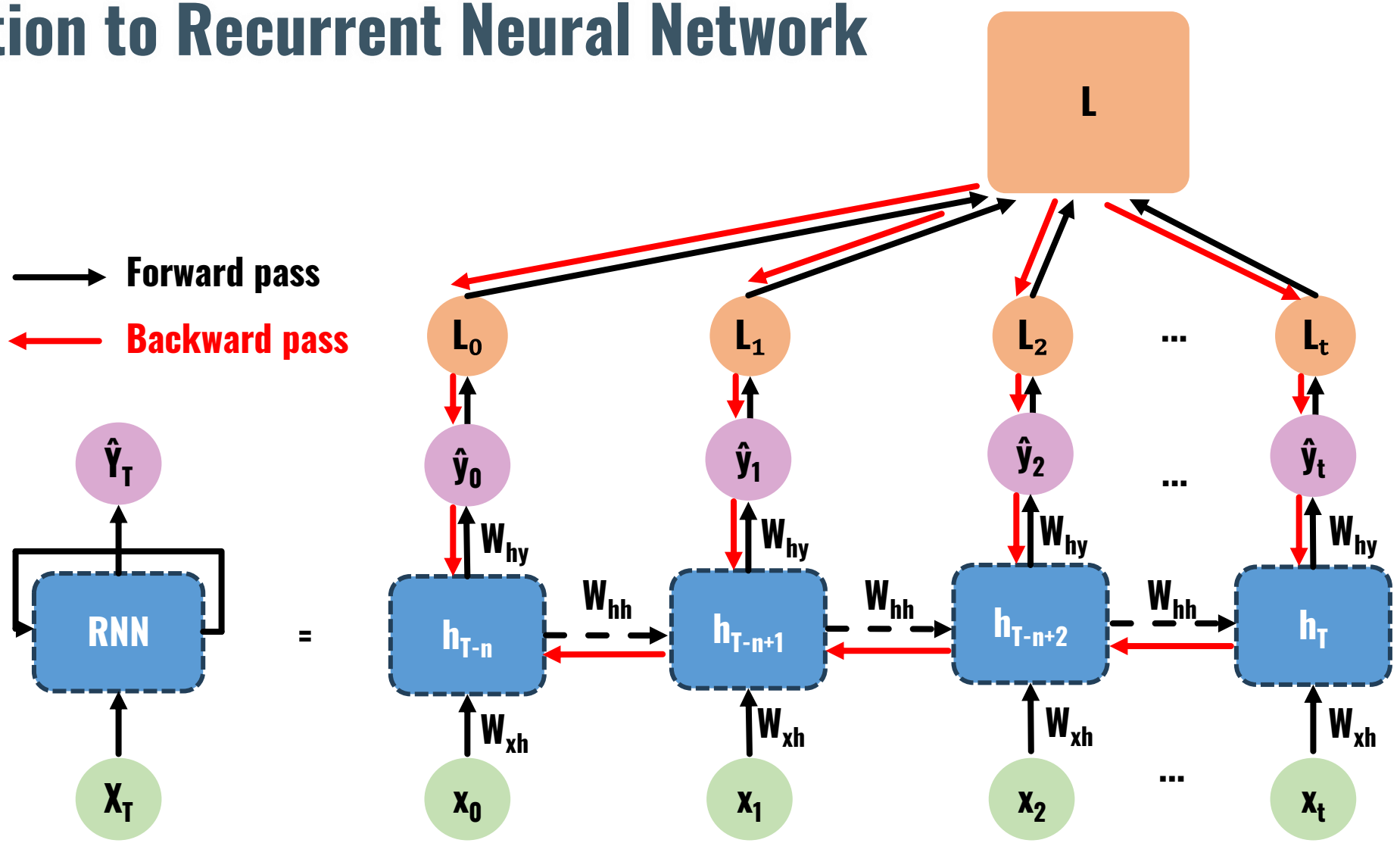
1. Calculate the forward pass
2. Determine the loss
3. Take the partial derivative (gradient) of the loss respect to each parameter
4. Shift the parameters to minimize the loss



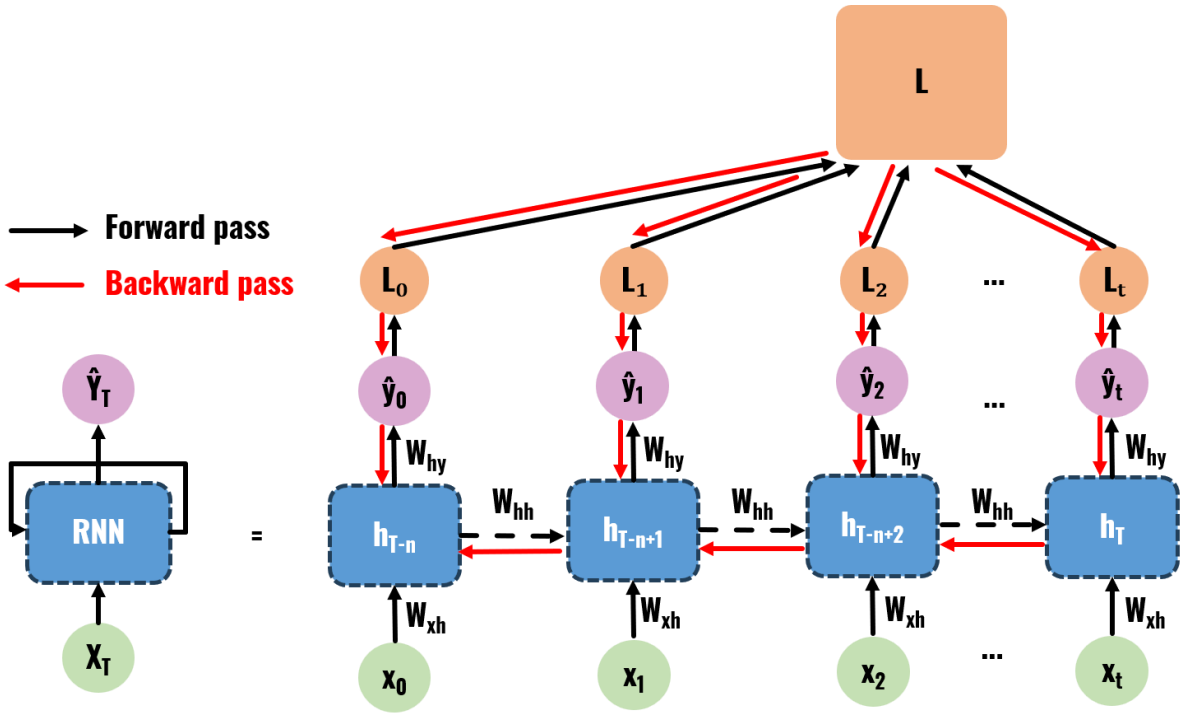
Introduction to Recurrent Neural Network



Introduction to Recurrent Neural Network



Introduction to Recurrent Neural Network



Hidden state:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Output vector:

$$\hat{y}_t = tanh(W_{hy}h_t + b_y)$$

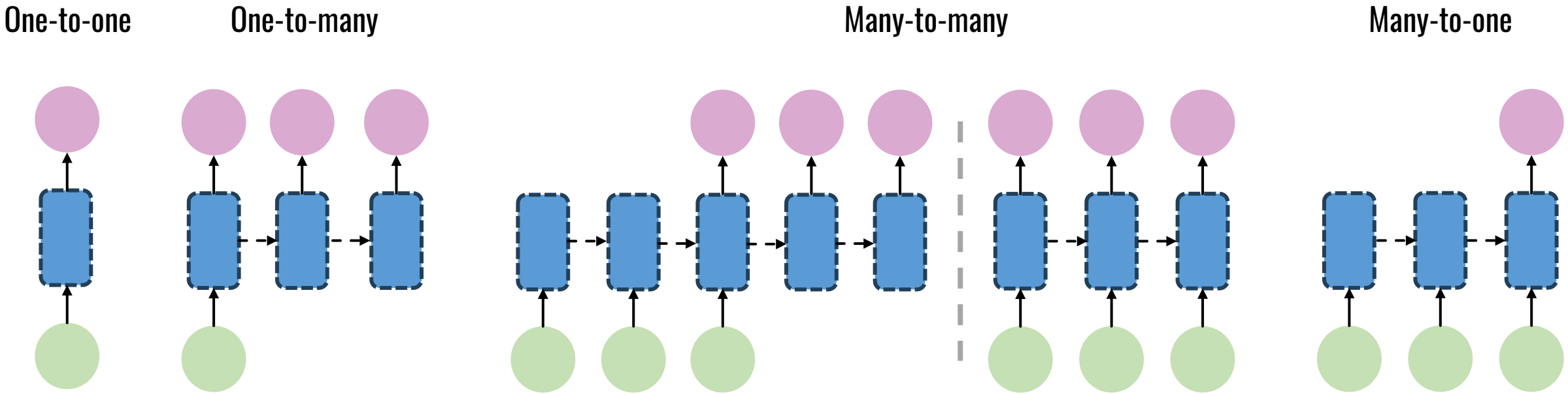
Loss:

$$L(x_1, \dots, x_T, y_1, \dots, y_T, W_{xh}, W_{hh}, W_{hy}) = \frac{1}{T}\sum_{t=1}^{T} l(y_t, \hat{y}_t)$$

Gradients:

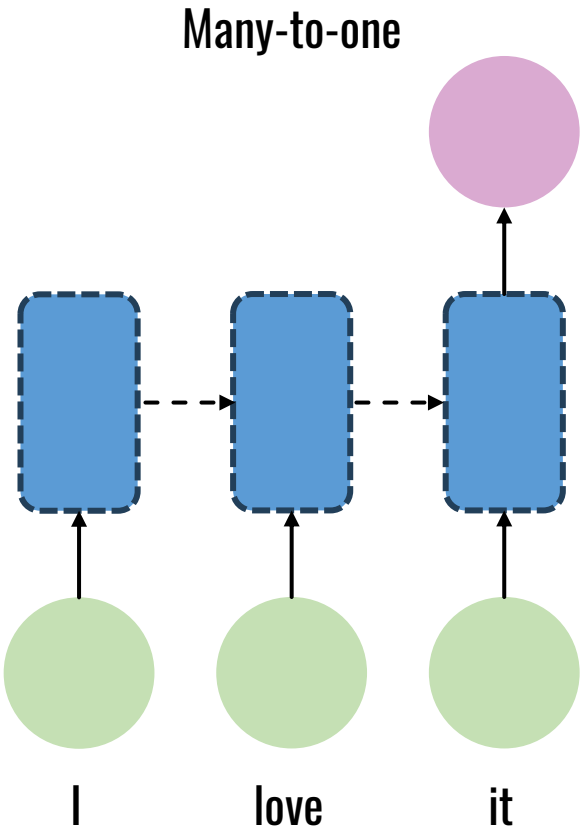
$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial l(y_t, \hat{y}_t)}{\partial W_{hh}}$$

RNN types



Tasks using RNNs

RNN Many to One – Sentiment Analysis



Sentiment:
Positive



Steven M
San Diego, California
Level 5
Contributor
64 reviews
33 attraction reviews
11 helpful votes

"Beautiful"

Reviewed 1 week ago via mobile

My wife and I stopped here on our cross country. Highly recommend. The walk to the bridge is pretty easy. Initially it's downhill with step (137 according to the sig) and then it is flat. Very beautiful. There are seat to just sit and enjoy or you can walk under it. On the other side there is a Native American village a salt mine and a river you can walk to. Very worth it if you live in VA or are just passing through.

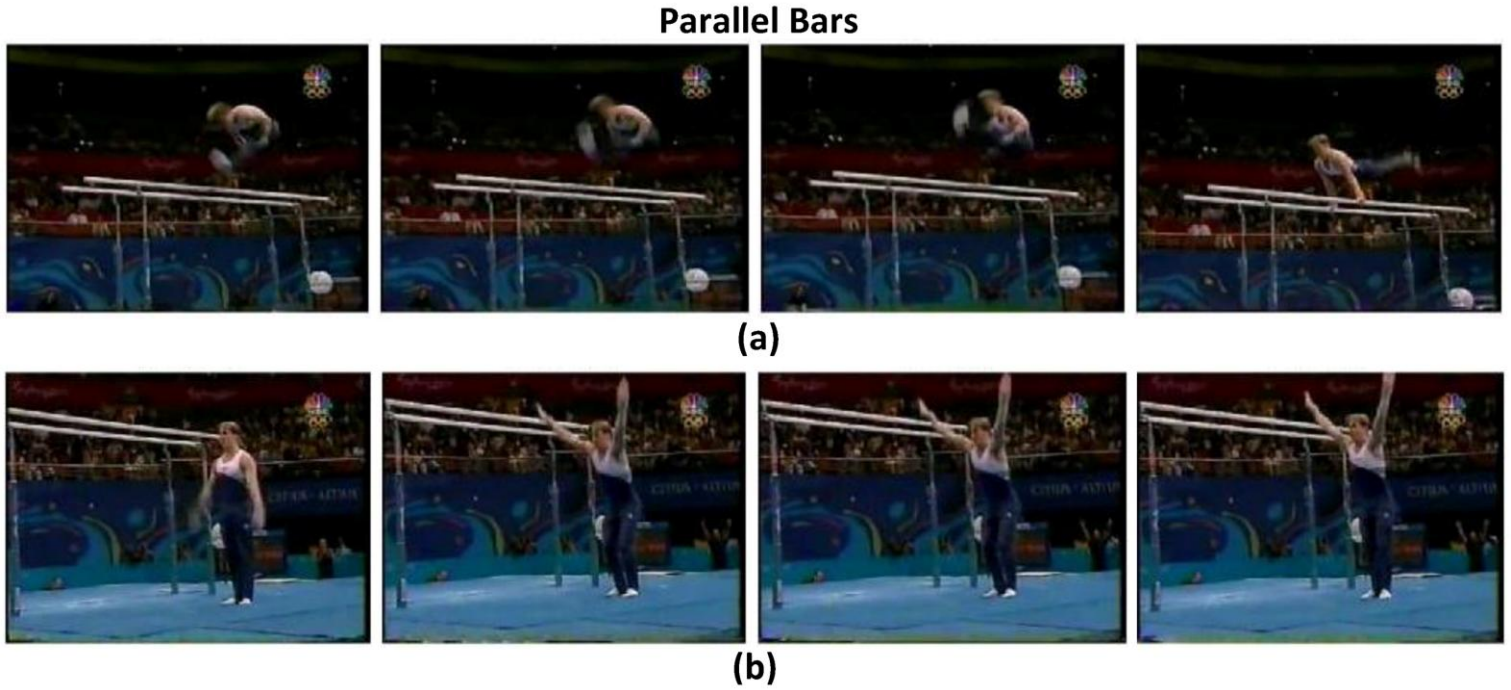


Visited October 2016

Image Source: Review of Natural Bridge State Park on Trip advisor (30352819436).png, vastateparksstaff, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

Tasks using RNNs

RNN Many to One – Video Activity Recognition



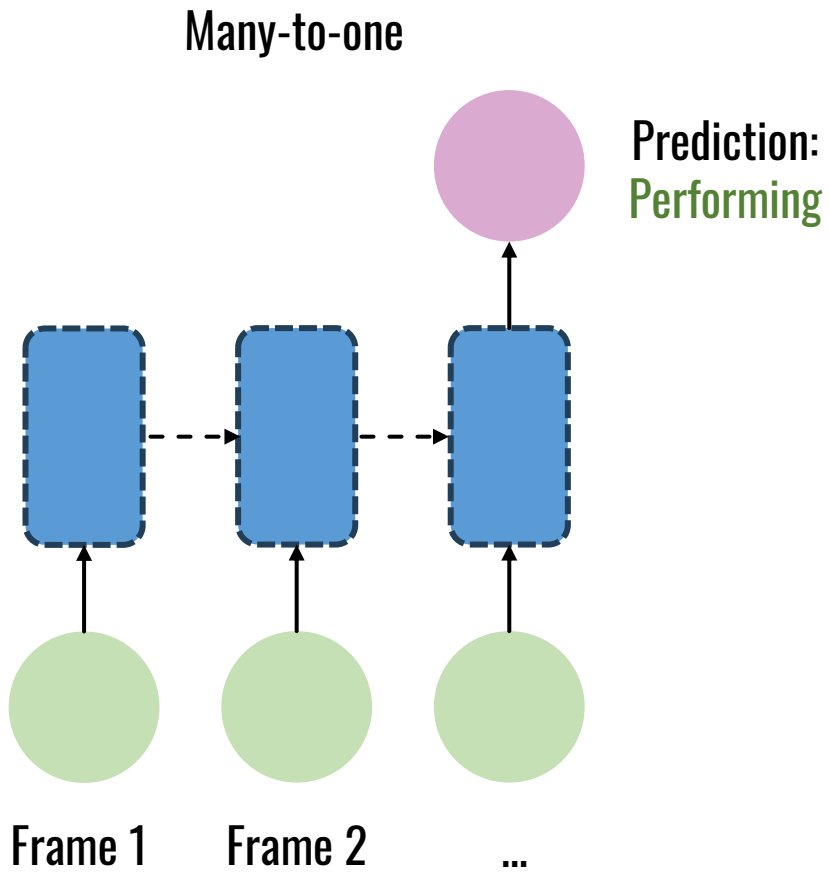
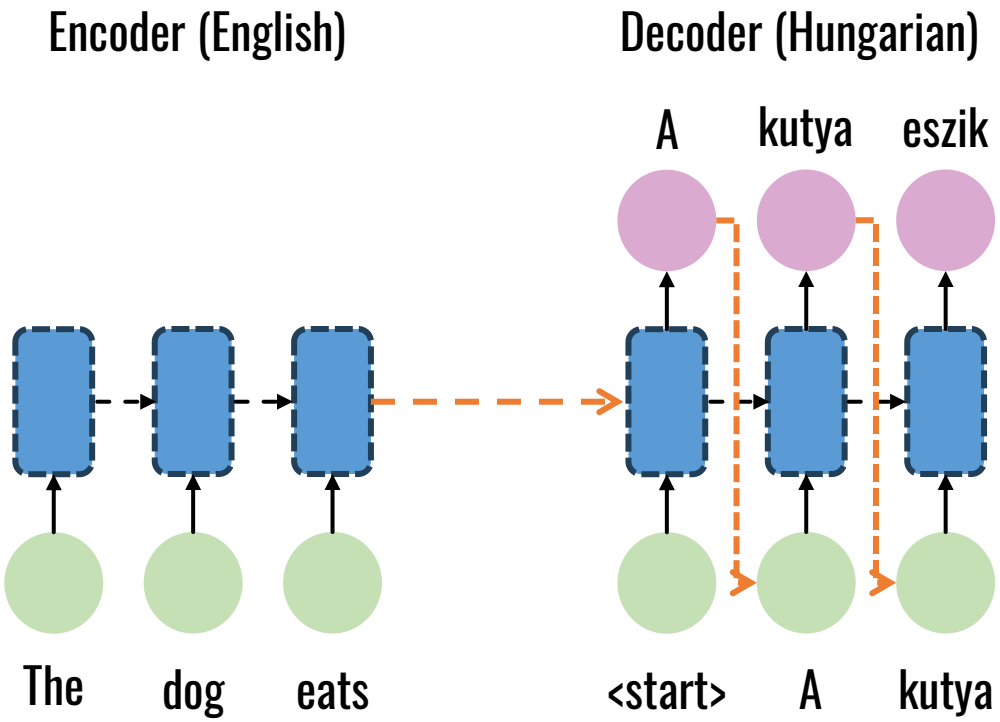


Image Source: Video frames of the Parallel Bars action category.png, Bikingdog, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Tasks using RNNs

RNN Many to Many – Machine Translation



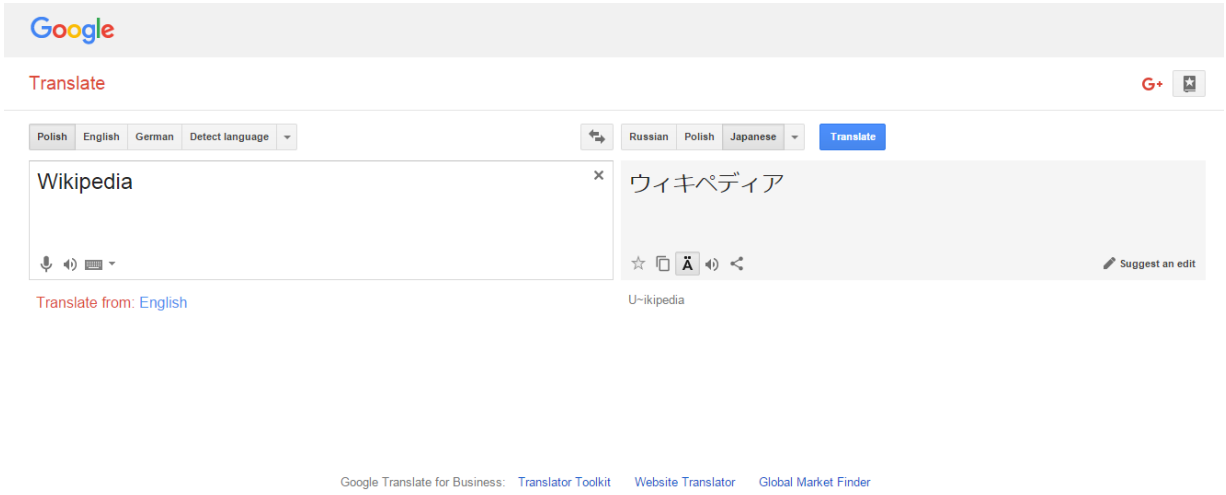
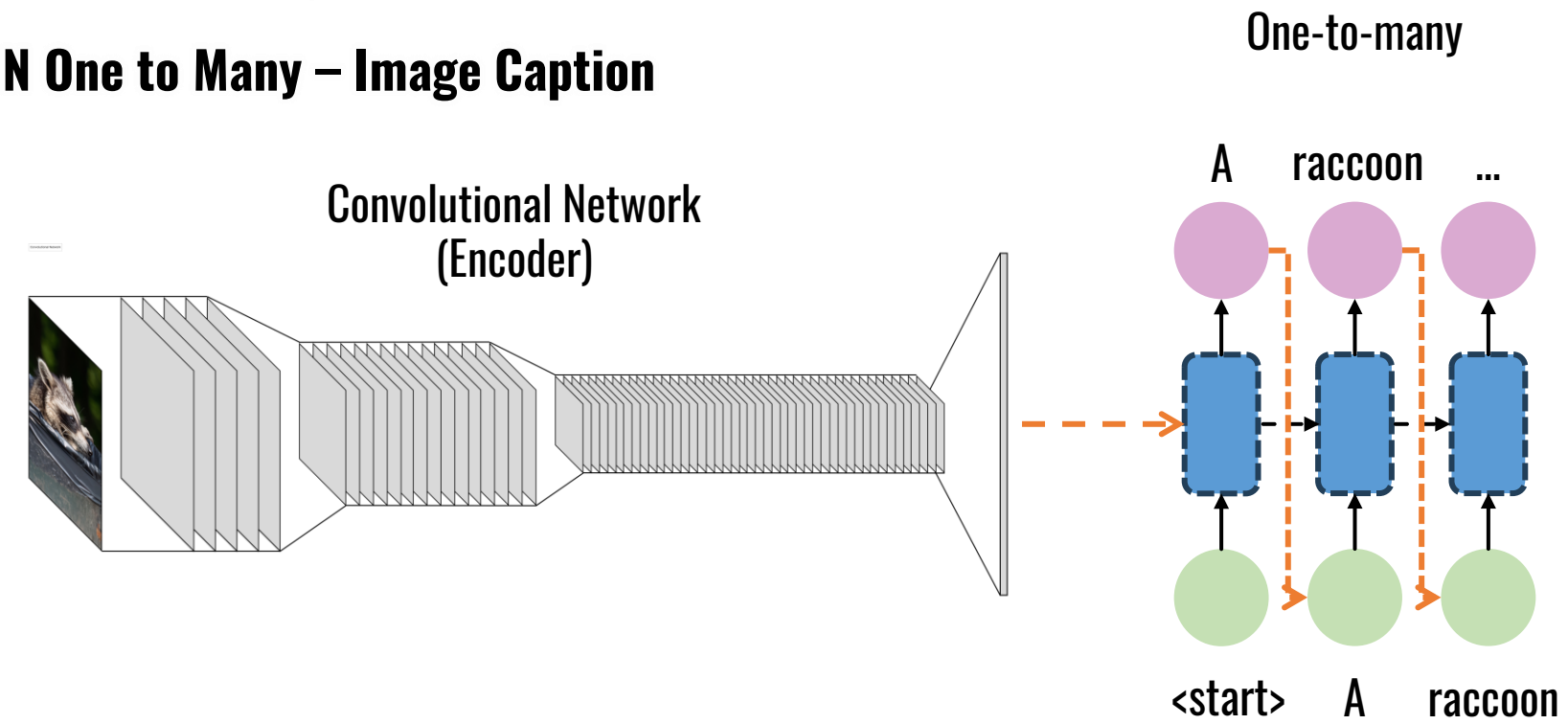


Image Source: Google Translate Polish-Japanese.png, Zuckergloy, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Tasks using RNNs

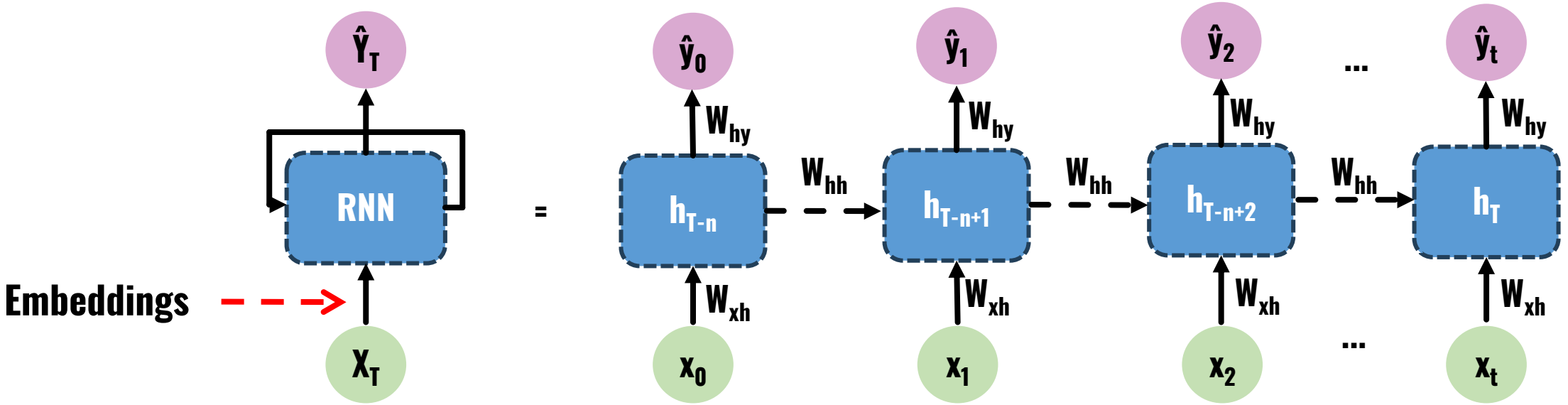
RNN One to Many – Image Caption



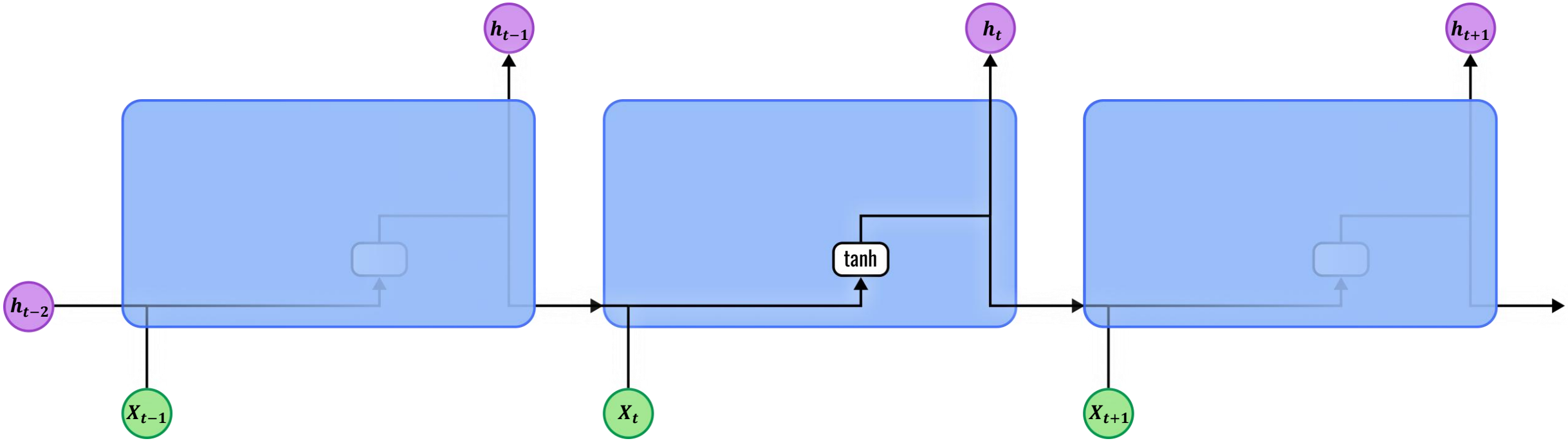
Prediction:
A raccoon sitting in a garbage can

Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Embeddings: Word representation



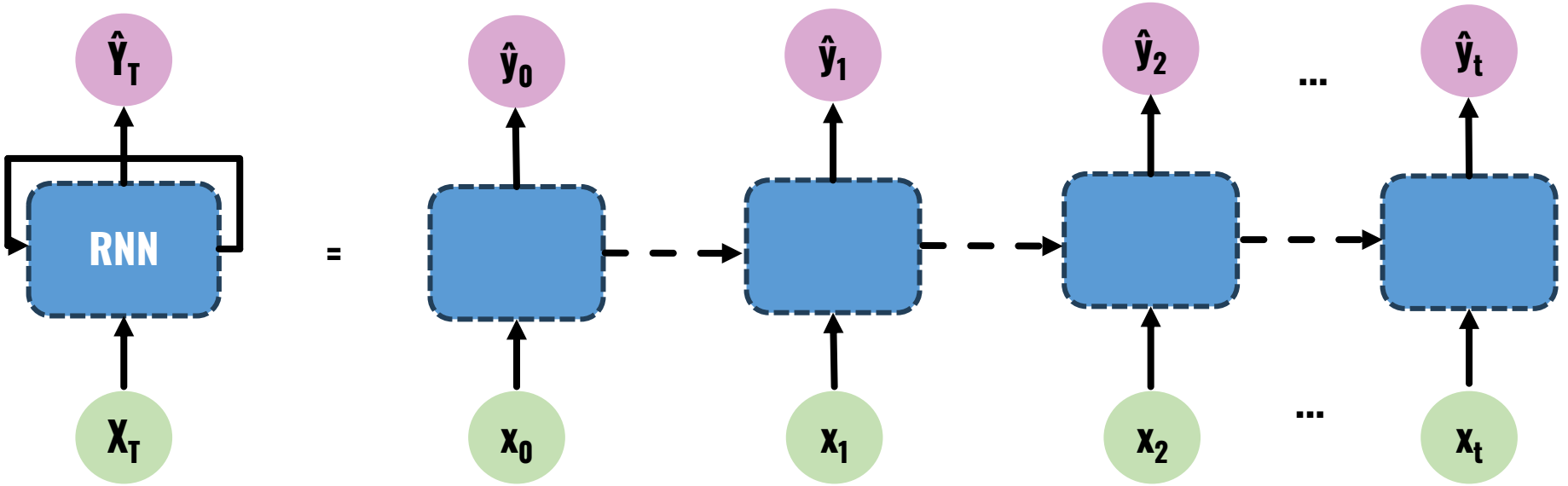
Vanilla RNNs



Vanilla RNNs

Problems with vanilla RNN

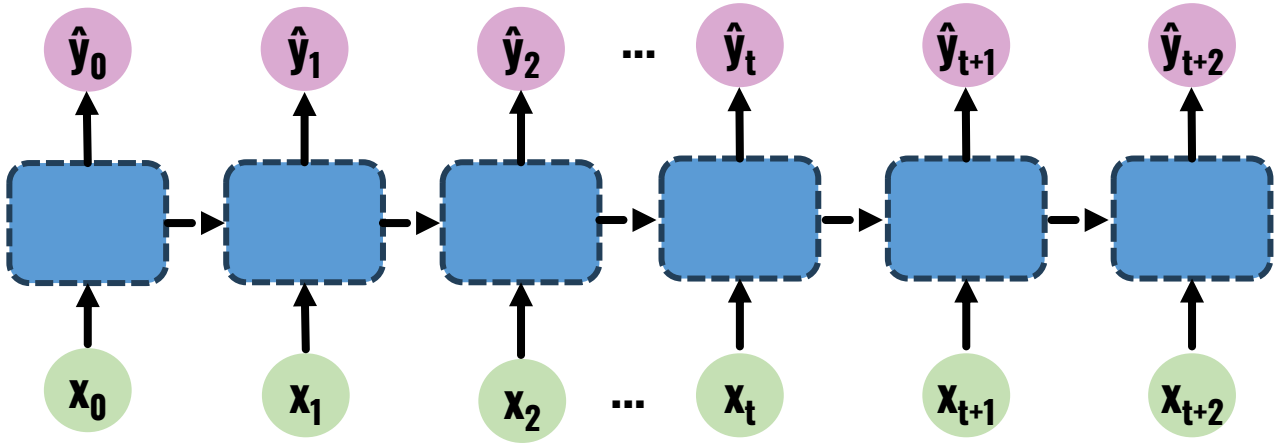
- Vanishing gradients
- Short term dependency



Vanilla RNNs

Problems with vanilla RNN

- It is like a very deep neural network
- Vanishing gradients
- Short term dependency – input closed to the current timestep has a greater influence

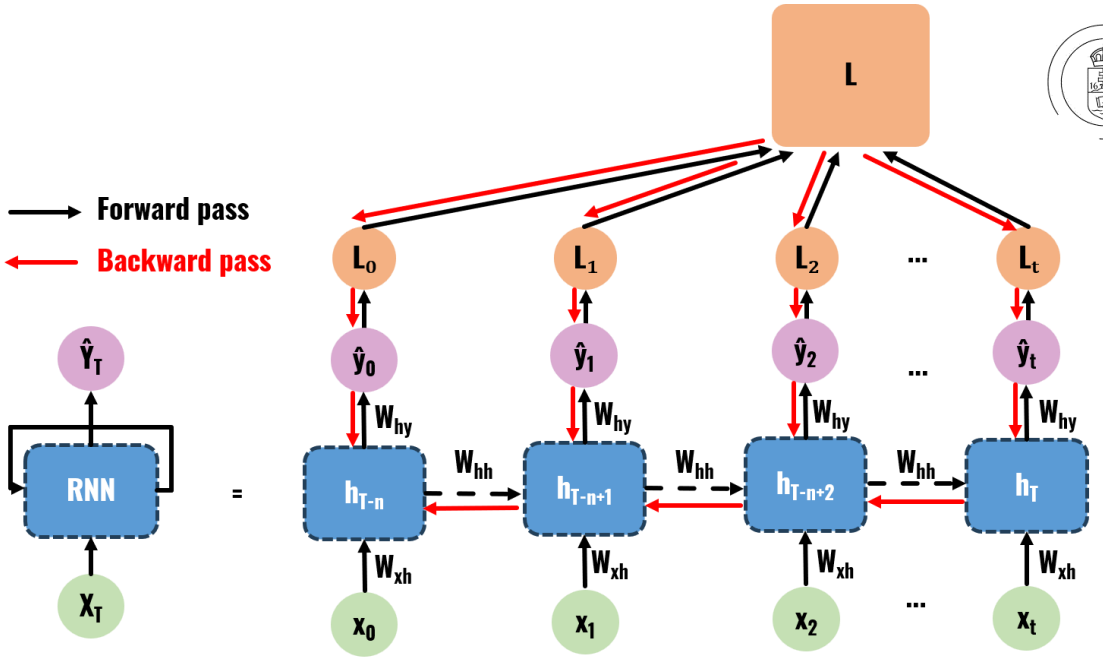


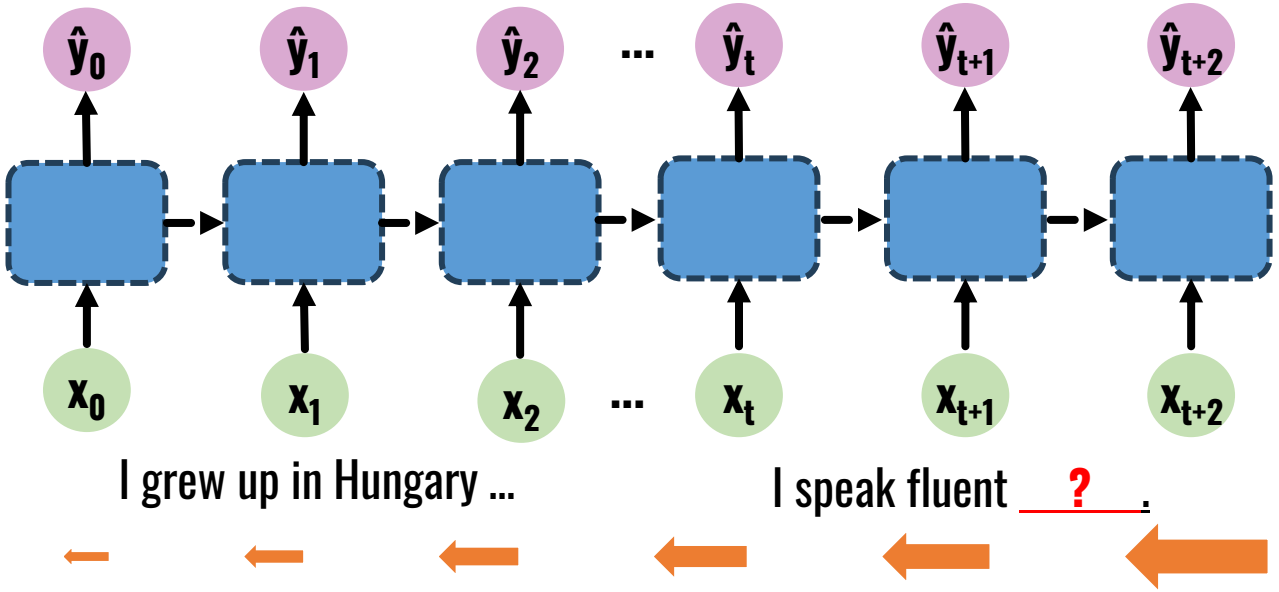
I grew up in Hungary … I speak fluent ___?___.

Vanilla RNNs

Problems with vanilla RNN

- It is like a very deep neural network
- Vanishing gradients
- Short term dependency – input closed to the current timestep has a greater influence





W_{hh} → small → vanishing gradient

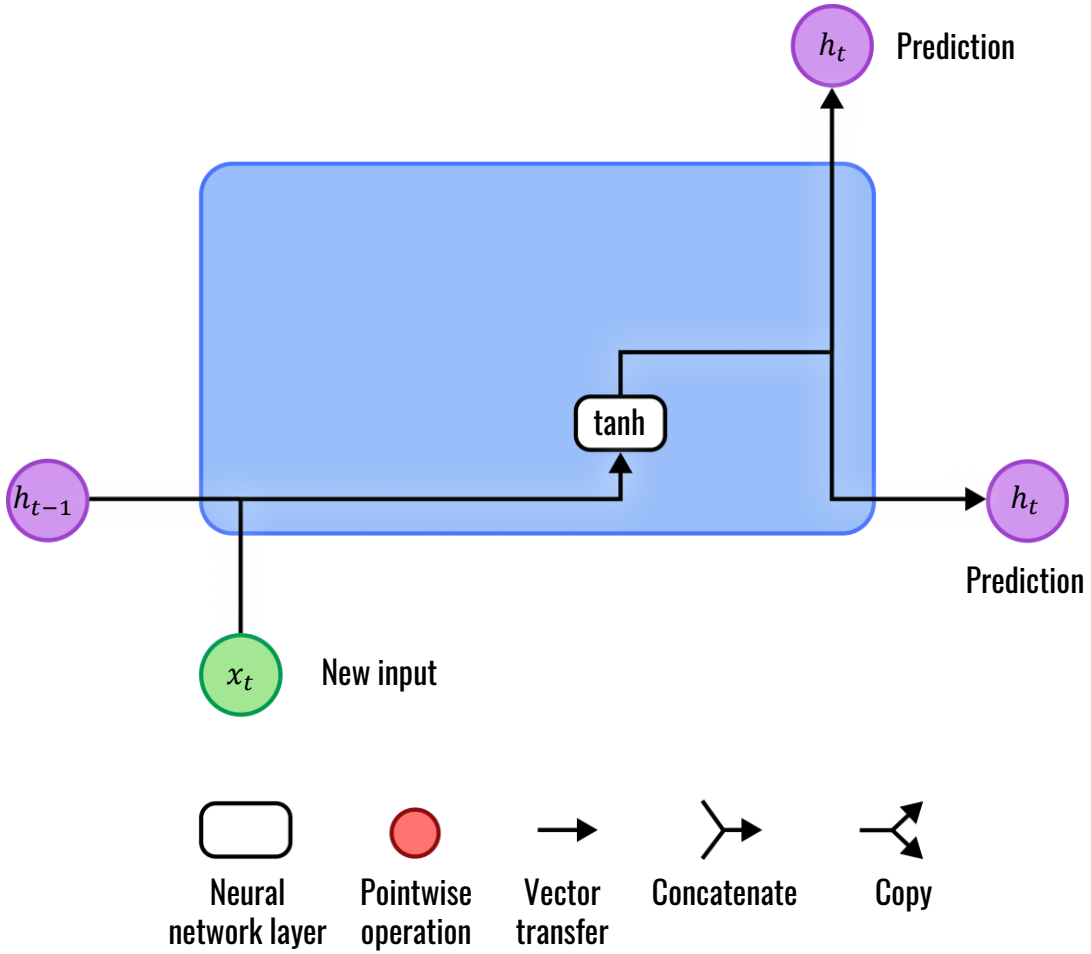
W_{hh} → large → exploding gradient

Long Short-Term Memory (LSTM)

Starting from a Vanilla RNN

Hidden state:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$



Vennerød, C. B., Kjærran, A., & Bugge, E. S. (2021). Long Short-term Memory RNN. In arXiv [cs.LG]. arXiv. https://doi.org/10.48550/arXiv.2105.06756

Long Short-Term Memory (LSTM)

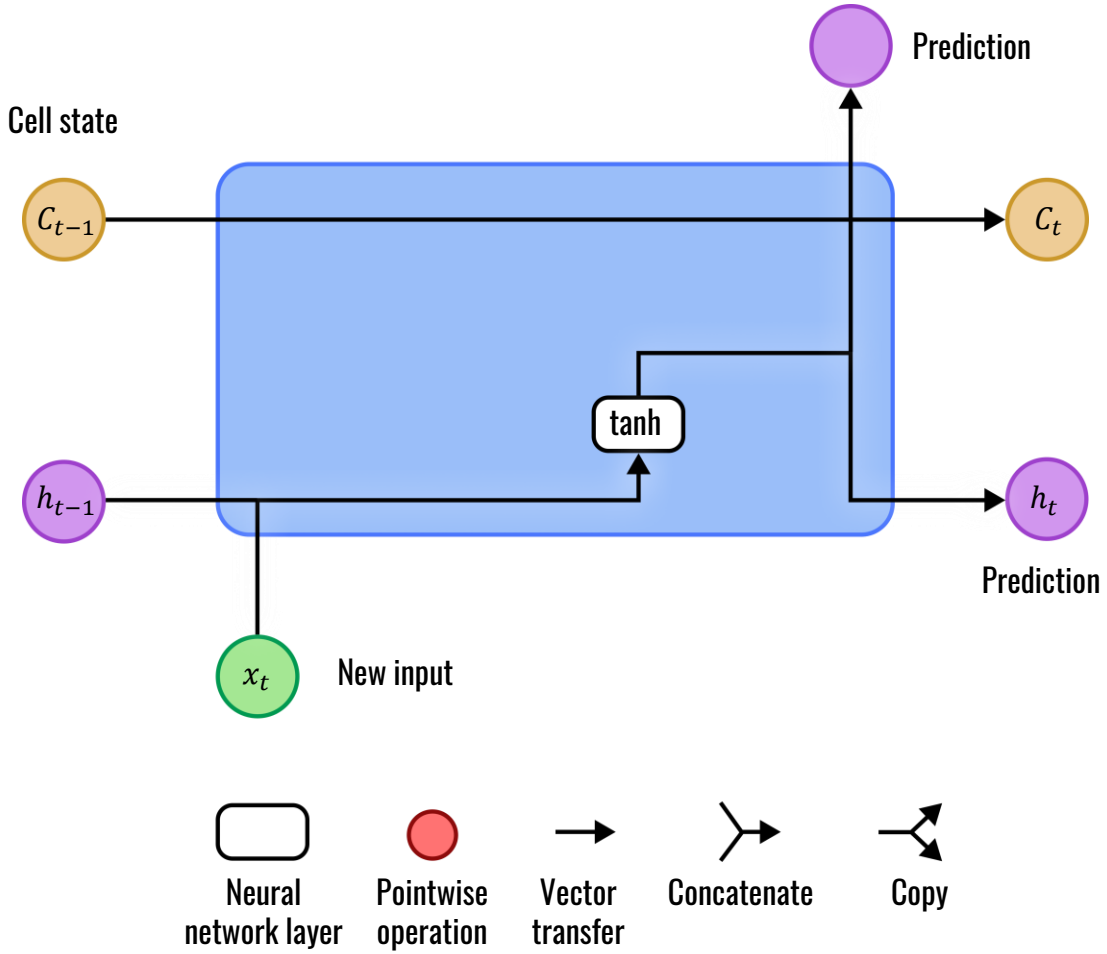
Add cell state to remember what happened many timesteps before

Hidden state:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Cell state:

$$C_t = C_{t-1}$$



Long Short-Term Memory (LSTM)

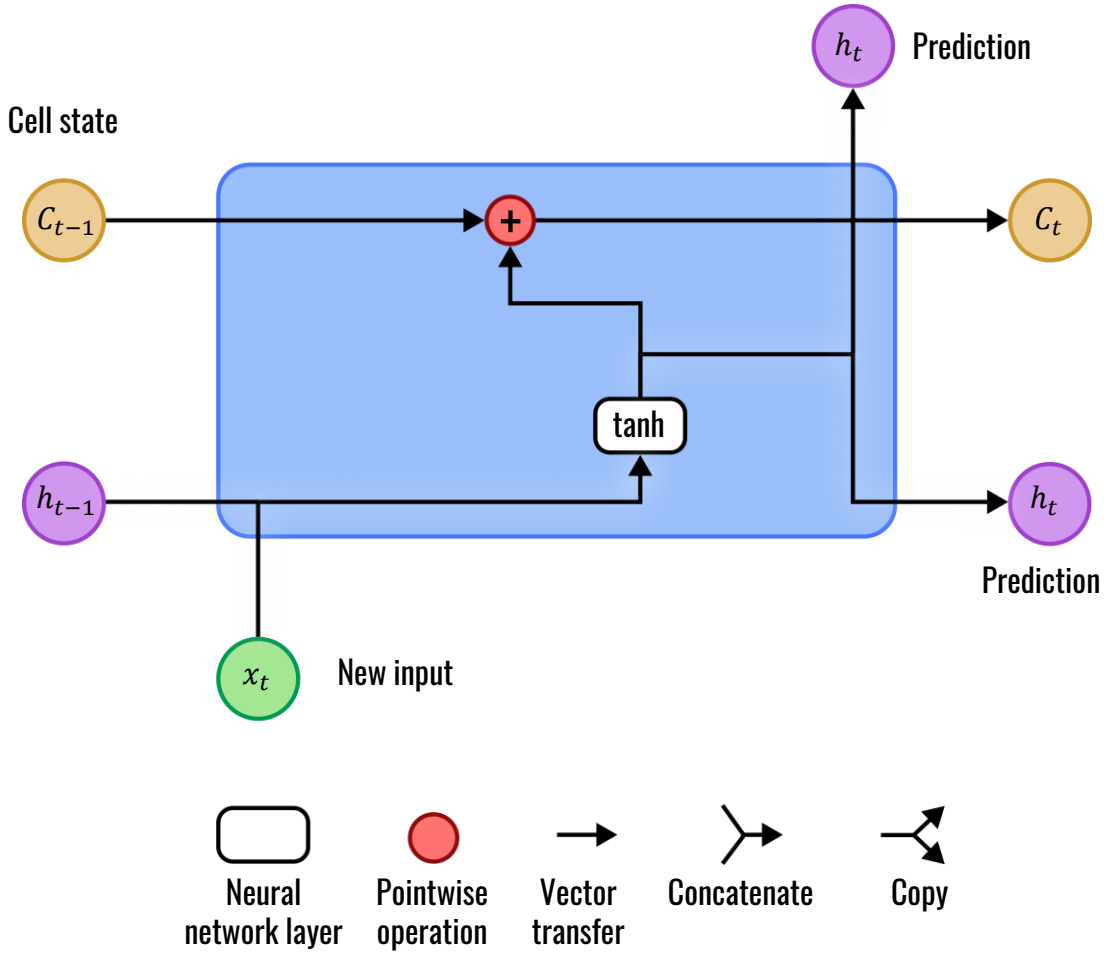
Update cell state based on the new timestep

Hidden state:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Cell state:

$$C_t = C_{t-1} + h_t$$



Long Short-Term Memory (LSTM)

Memory: to remember what happened many timesteps before

Forgetting: Based on what we are seeing now, we decide what we want to forget/remember

A copy of the predictions is saved for the next timestep

- Some of them are forgotten, some of them are remembered
- Then added back to the prediction

Hidden state:

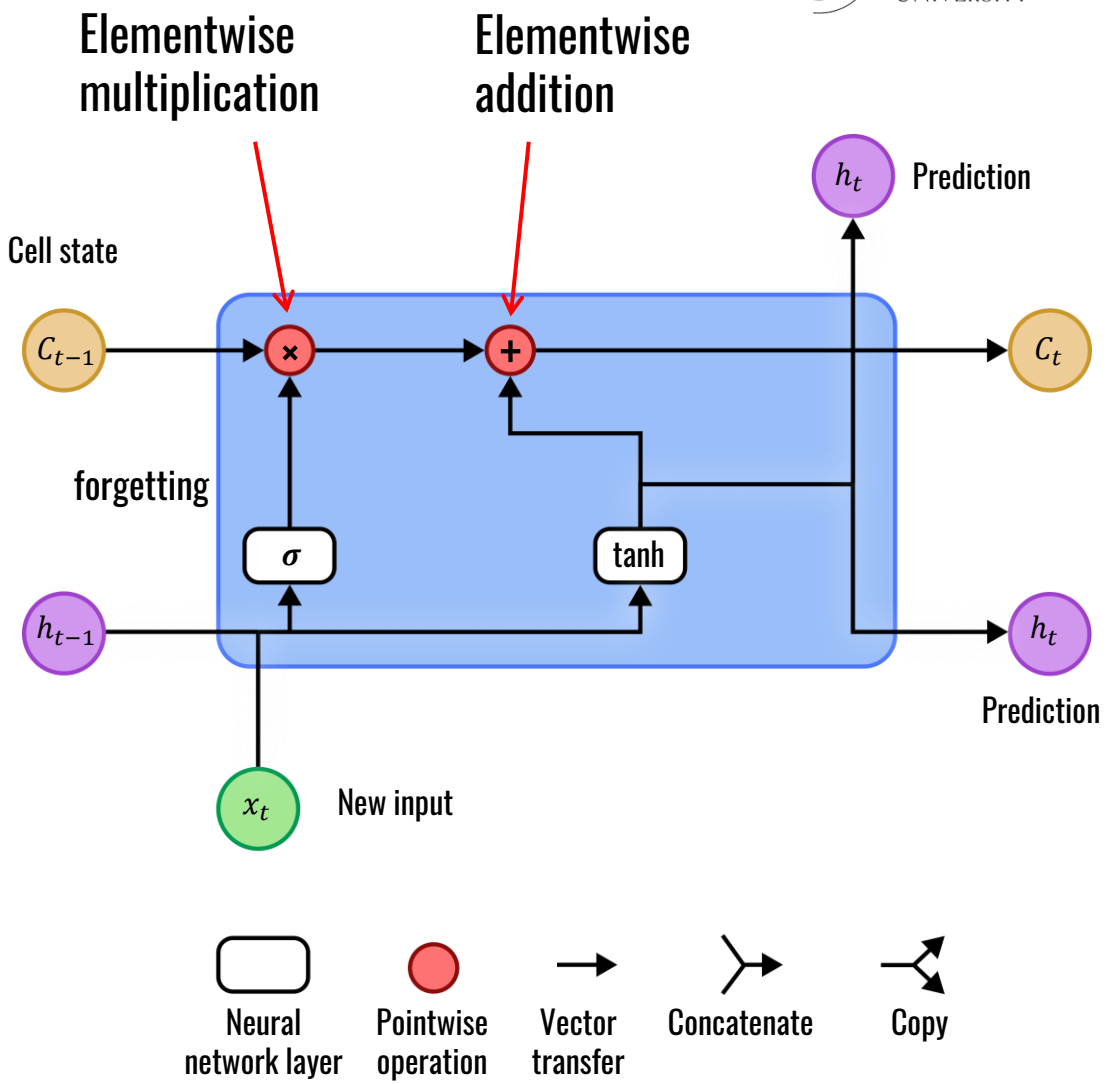
$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Forget gate:

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f)$$

Cell state:

$$C_t = f_t \odot C_{t-1} + h_t$$

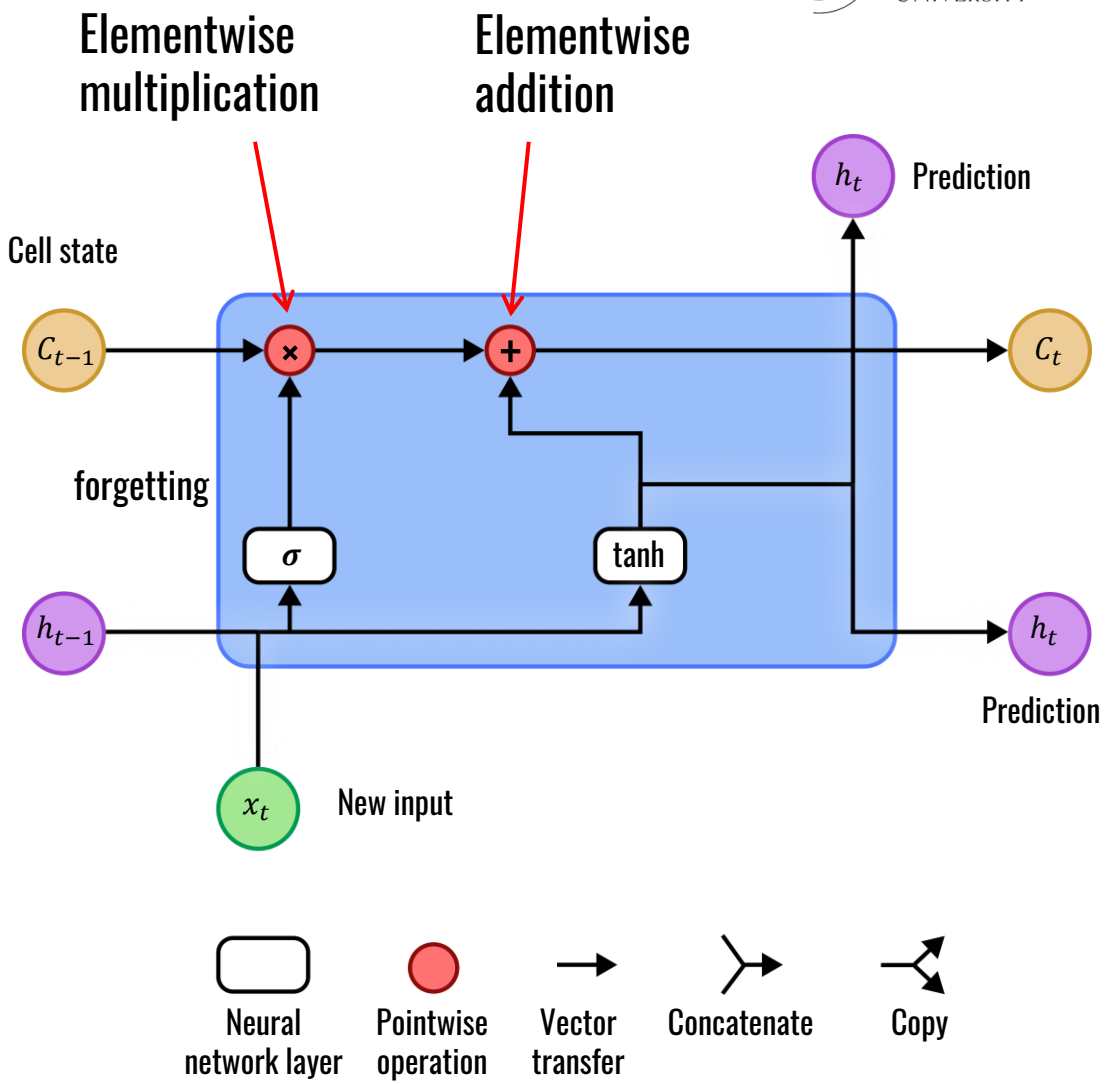


Long Short-Term Memory (LSTM)

Memory: to remember what happened many timesteps before

Forgetting: Based on what we are seeing now, we decide what we want to forget/remember

$$\begin{bmatrix}0.8\\0.8\\0.8\end{bmatrix} \times = \begin{bmatrix}0.8 \cdot 1.0\\0.8 \cdot 0.5\\0.8 \cdot 0.0\end{bmatrix} = \begin{bmatrix}0.8\\0.4\\0.0\end{bmatrix}$$
$$\begin{bmatrix}1.0\\0.5\\0.0\end{bmatrix}$$



Long Short-Term Memory (LSTM)

- Now we have a new state based on the previous and current time step
- **The hidden state is not affected by the cell state**
- We ignore what is not immediately relevant – **ignore gate**
- After addition we need **to rescale the cell state** – tanh

Forget gate:

$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t + b_f)$$

Ignore gate:

$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t + b_i)$$

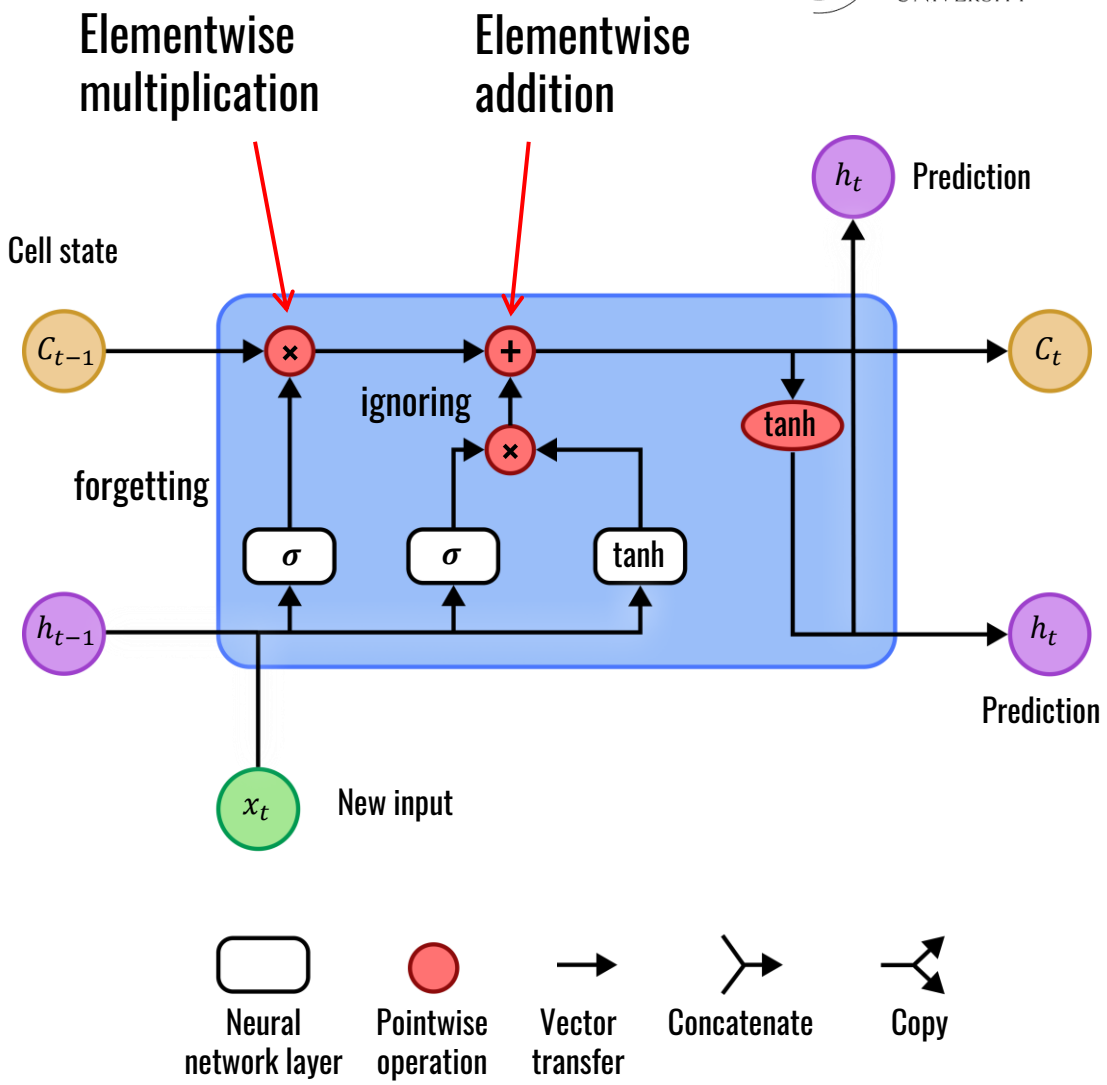
Cell state:

$$\widetilde{C}_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t$$

Hidden state:

$$h_t = \tanh(C_t)$$



Long Short-Term Memory (LSTM)

We may not want to show everything after we combined our prediction with our memory

Introduce a filter to keep our memories inside and let our prediction out – **selection/output gate**

Forget gate:

$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t + b_f)$$

Ignore gate:

$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t + b_i)$$

Output gate:

$$o_t = \sigma(W_{ho} h_{t-1} + W_{xo} + b_o)$$

Cell state:

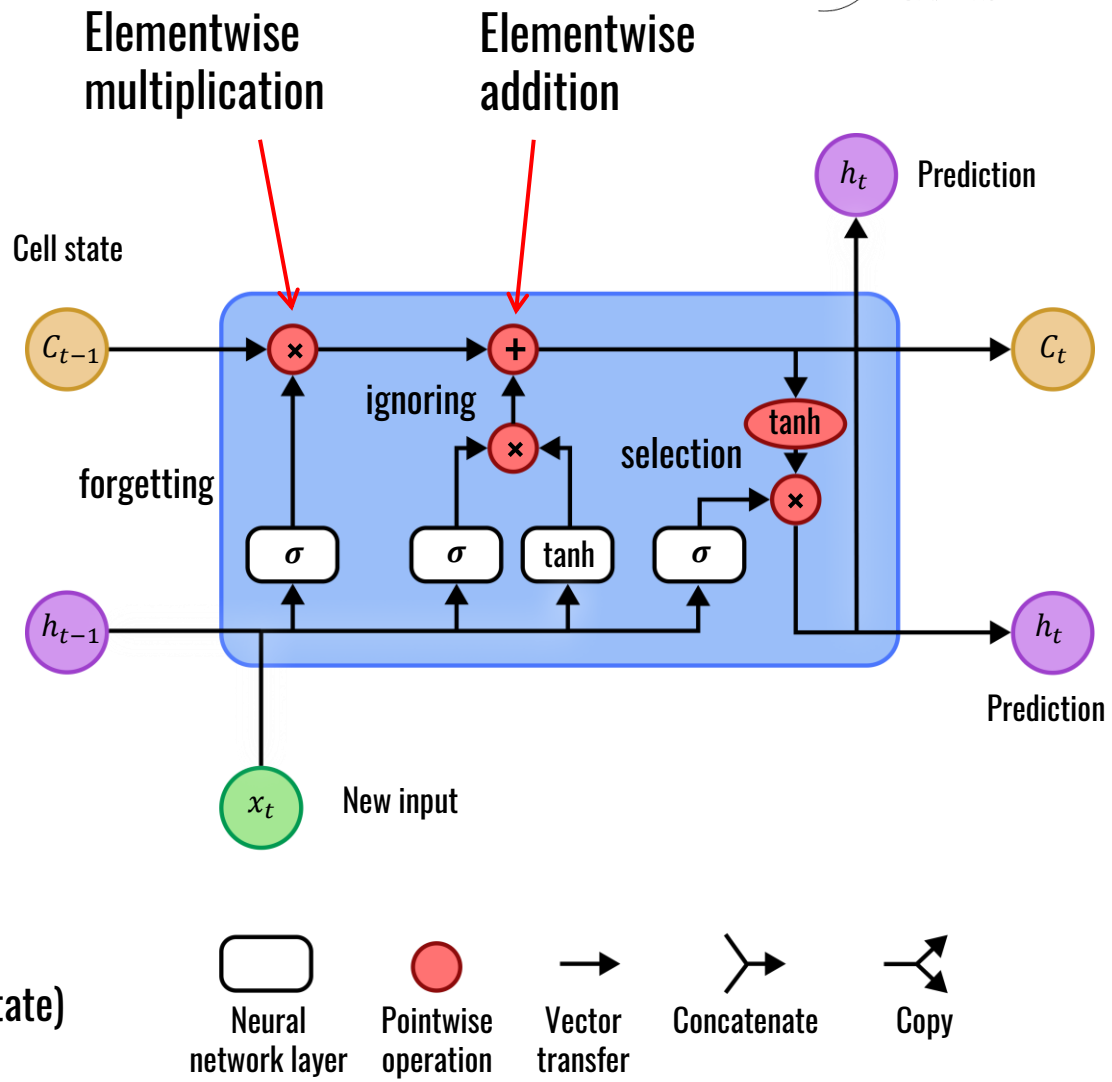
$$\widetilde{C_t} = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C_t}$$

(Candidate cell state)

Hidden state:

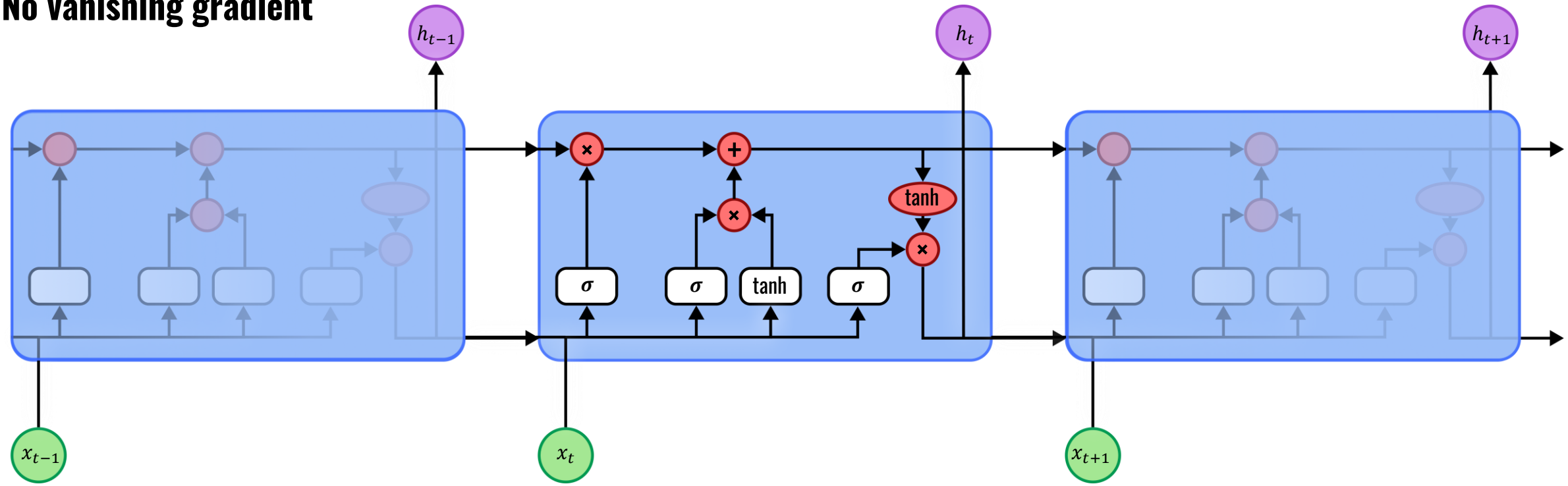
$$h_t = o_t \odot \tanh(C_t)$$



Long Short-Term Memory (LSTM)

LSTM

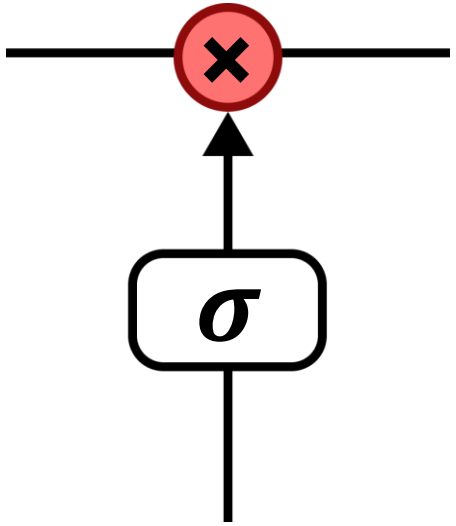
- a more complex network
- No vanishing gradient



Long Short-Term Memory (LSTM)

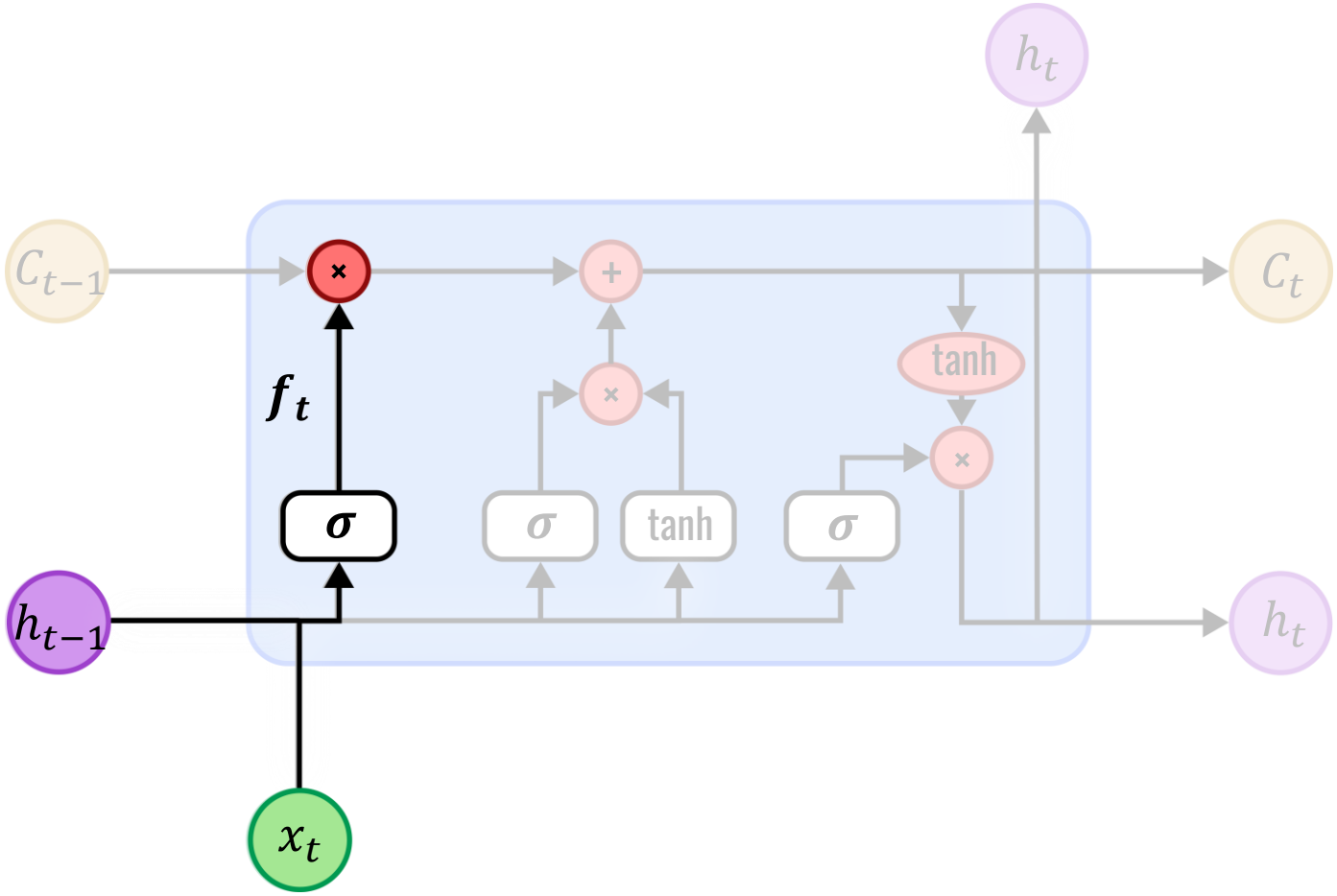
LSTM - Gate

- Sigmoid puts output between 0 and 1
- Output of sigmoid controls which info goes through the gate



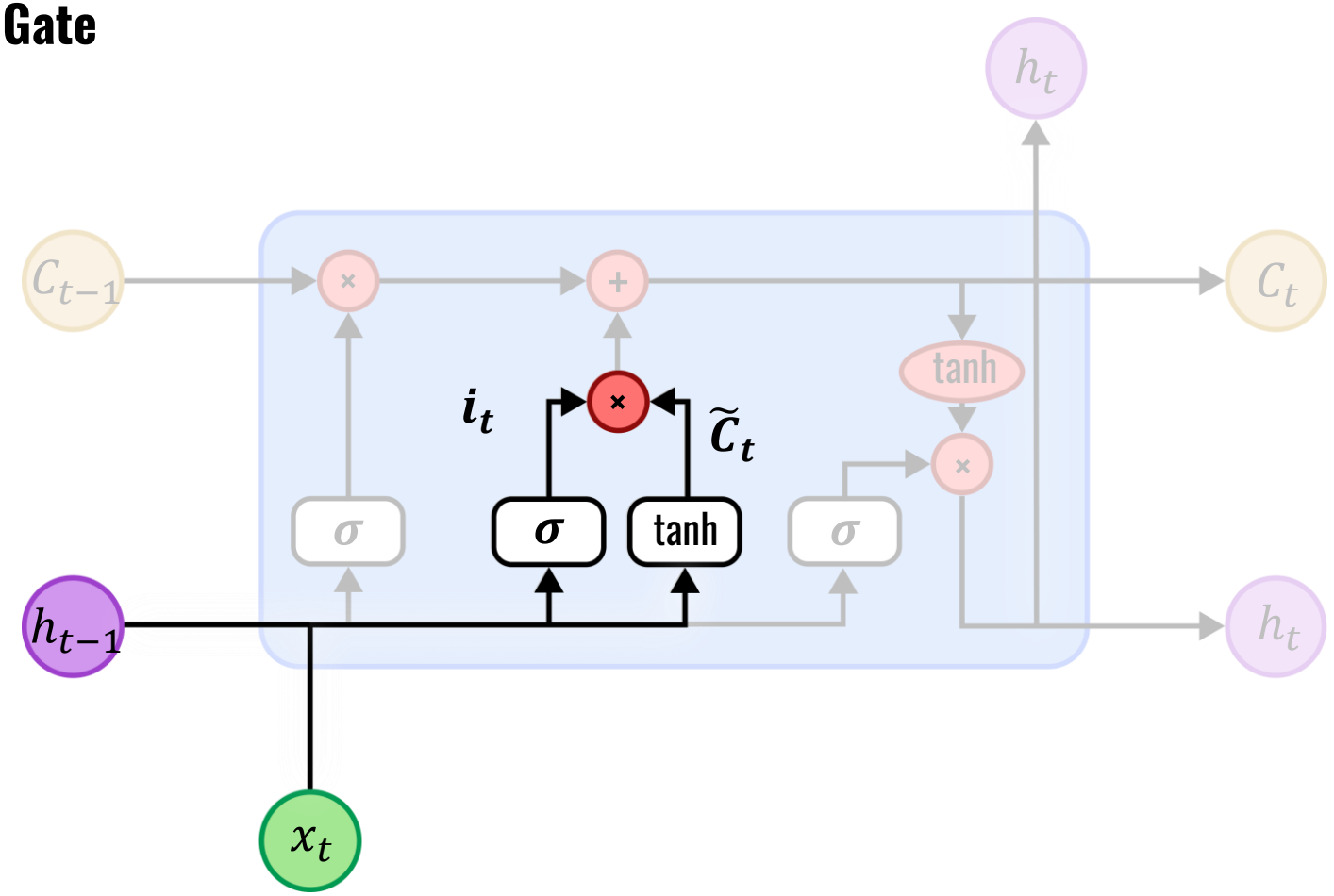
Long Short-Term Memory (LSTM)

LSTM – Forget Gate



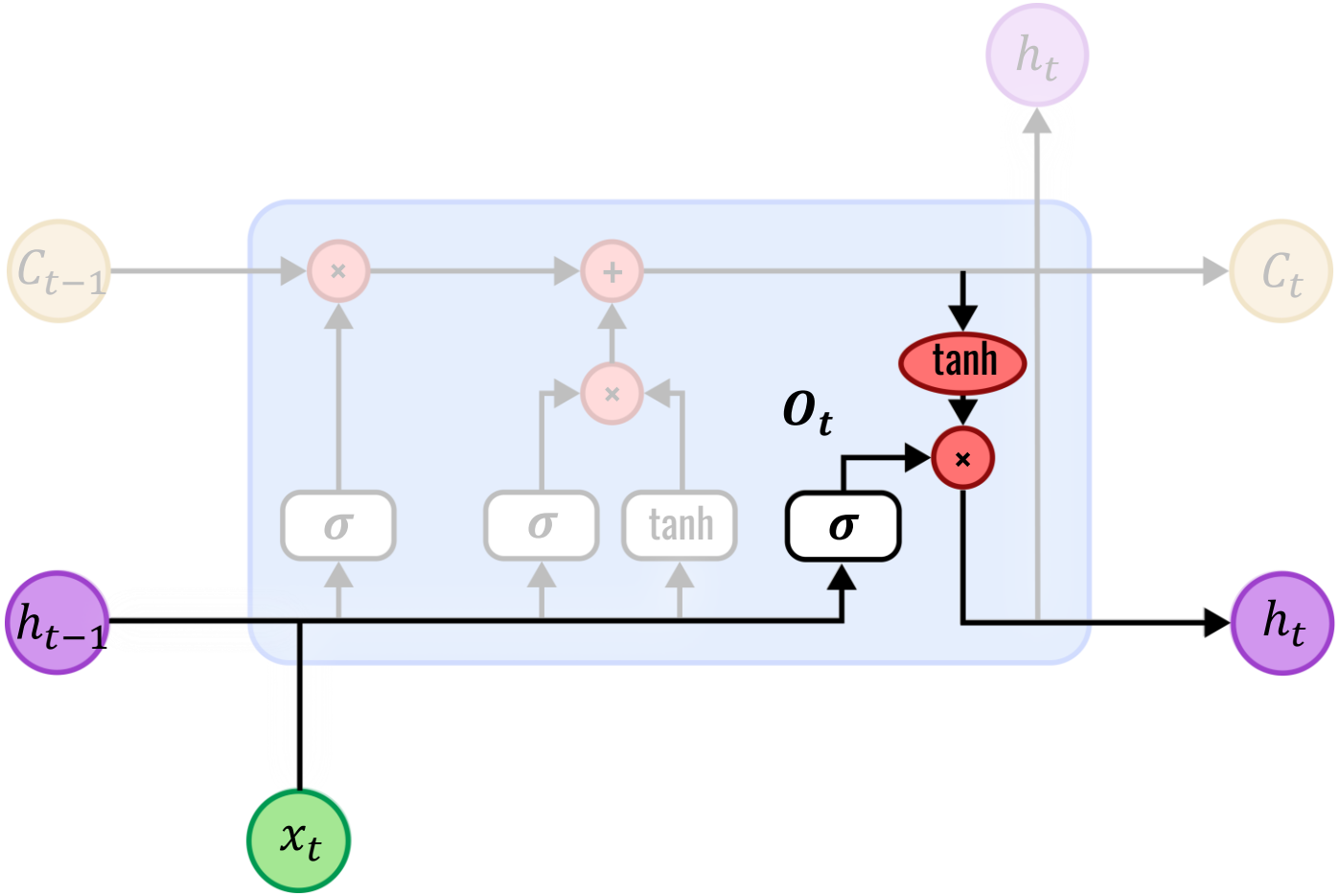
Long Short-Term Memory (LSTM)

LSTM – Input / Ignore Gate



Long Short-Term Memory (LSTM)

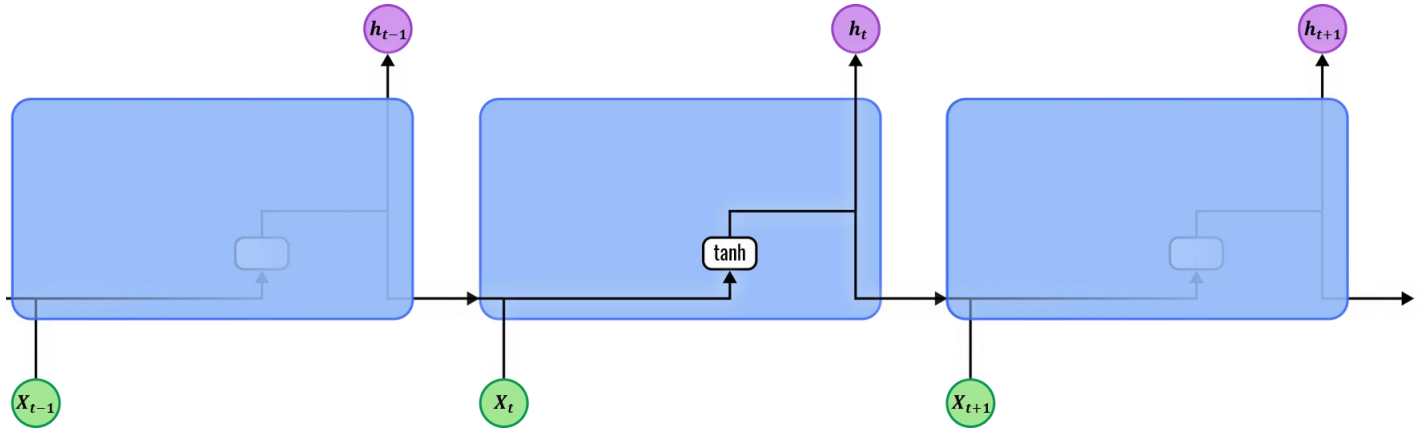
LSTM – Output Gate



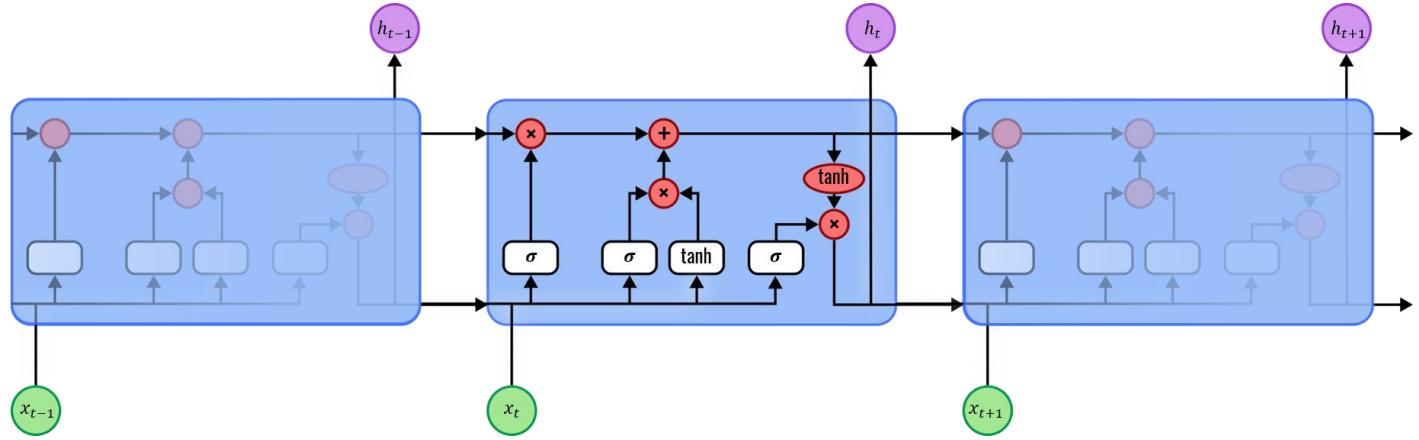
Long Short-Term Memory (LSTM)

LSTM – a more complex network

Vanilla RNN:



LSTM:



Long Short-Term Memory (LSTM)

Gates:

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f)$$
$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t + b_i)$$
$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo} + b_o)$$
$$\widetilde{C}_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Output:

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t$$
$$h_t = o_t \odot \tanh(C_t)$$

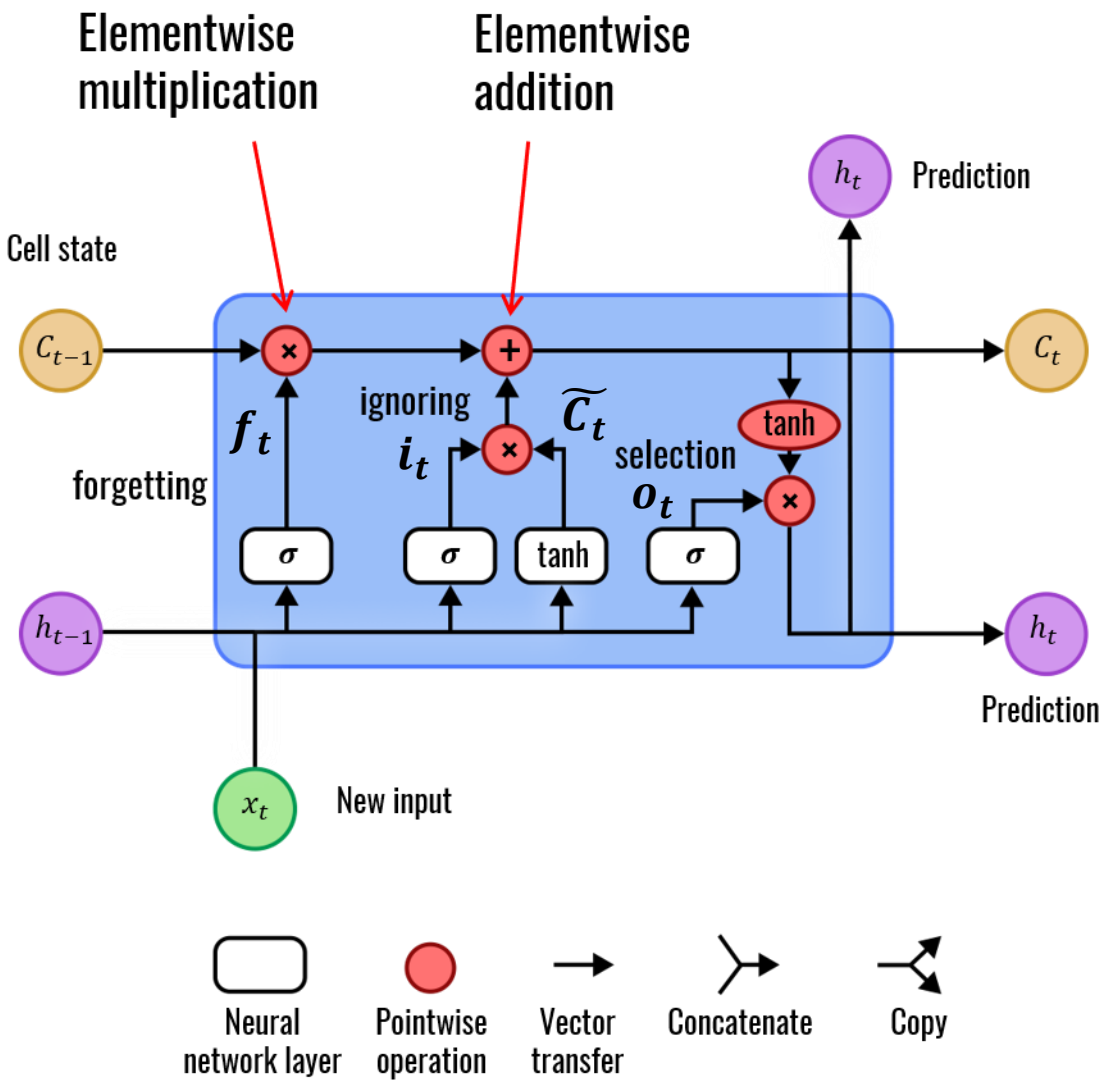
Where $\odot$ is an elementwise operation

$\mathbf{f_t}$ - forget gate - erase cell

$\mathbf{i_t}$ - ignore gate - what to ignore

$\mathbf{o_t}$ - output gate - whether to reveal cell

$\widetilde{\mathbf{C}}_\mathbf{t}$ - "Vanilla RNN" - how much to write to cell



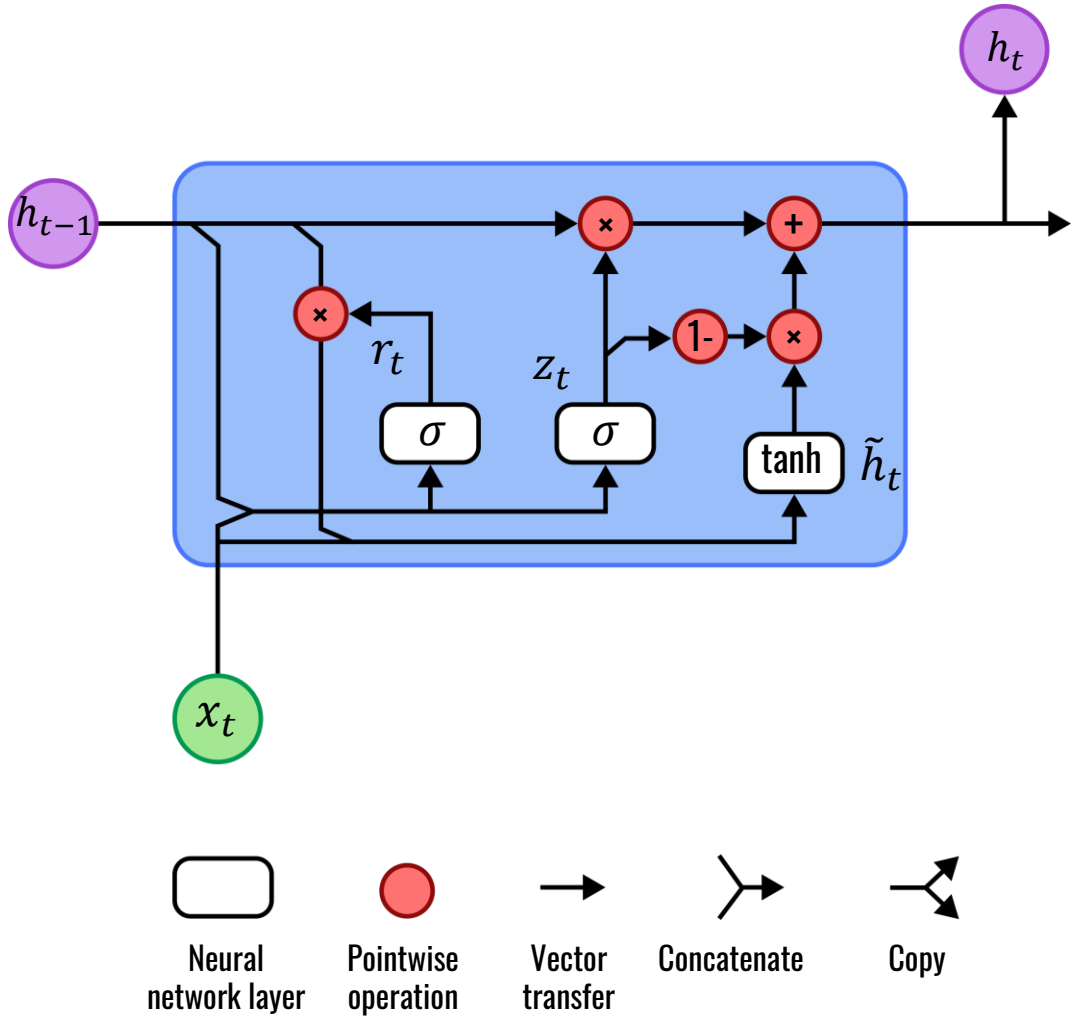
Gated Recurrent Unit (GRU)

$$r_t = \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r)$$

$$z_t = \sigma(x_t W_{xz} + h_{t-1} W_{hz} + b_z)$$

$$\tilde{h}_t = \tanh(x_t W_{xh} + (r_t \odot h_{t-1}) W_{hh} + b_h)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$



where

r_t- reset gate – what information to use for the update

z_t- update gate – how much of the update to apply

$\tilde{h}_t$- candidate hidden state

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.1259

LSTM vs GRU

The main differences between GRU and LSTM are:

- **Number of gates:** GRU has two gates - an update gate and a reset gate, whereas LSTM has three gates - input, forget, and output gates.
- **Memory cell:** Unlike LSTM, GRU doesn't have a separate memory cell. It combines the hidden state and memory cell into a single hidden state, simplifying the structure.

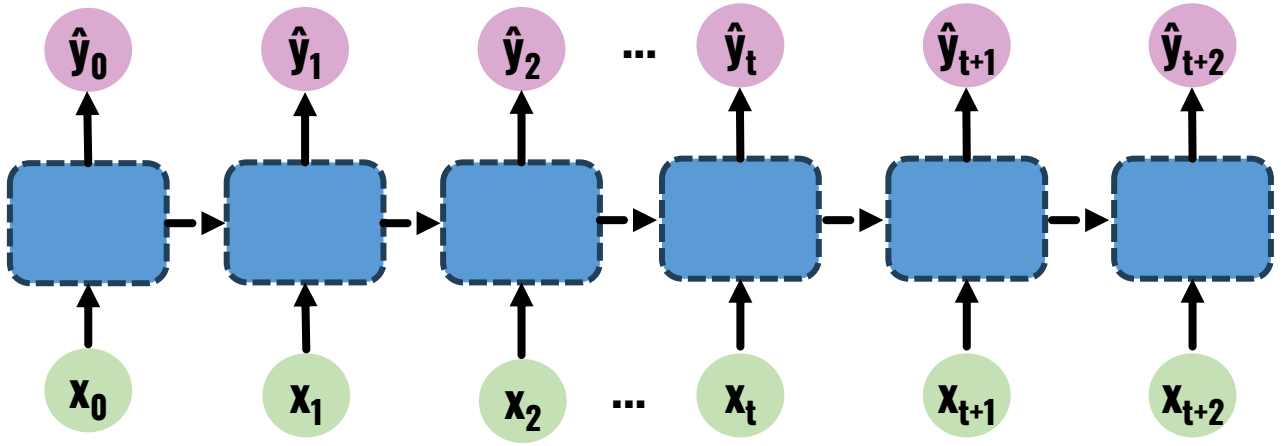
Recurrent Layer

Maintains an internal hidden state

- Updated at every time step and fed back into the model as the next input arrives
- Carries past context forward to inform future steps

Well suited to learning long-term dependencies

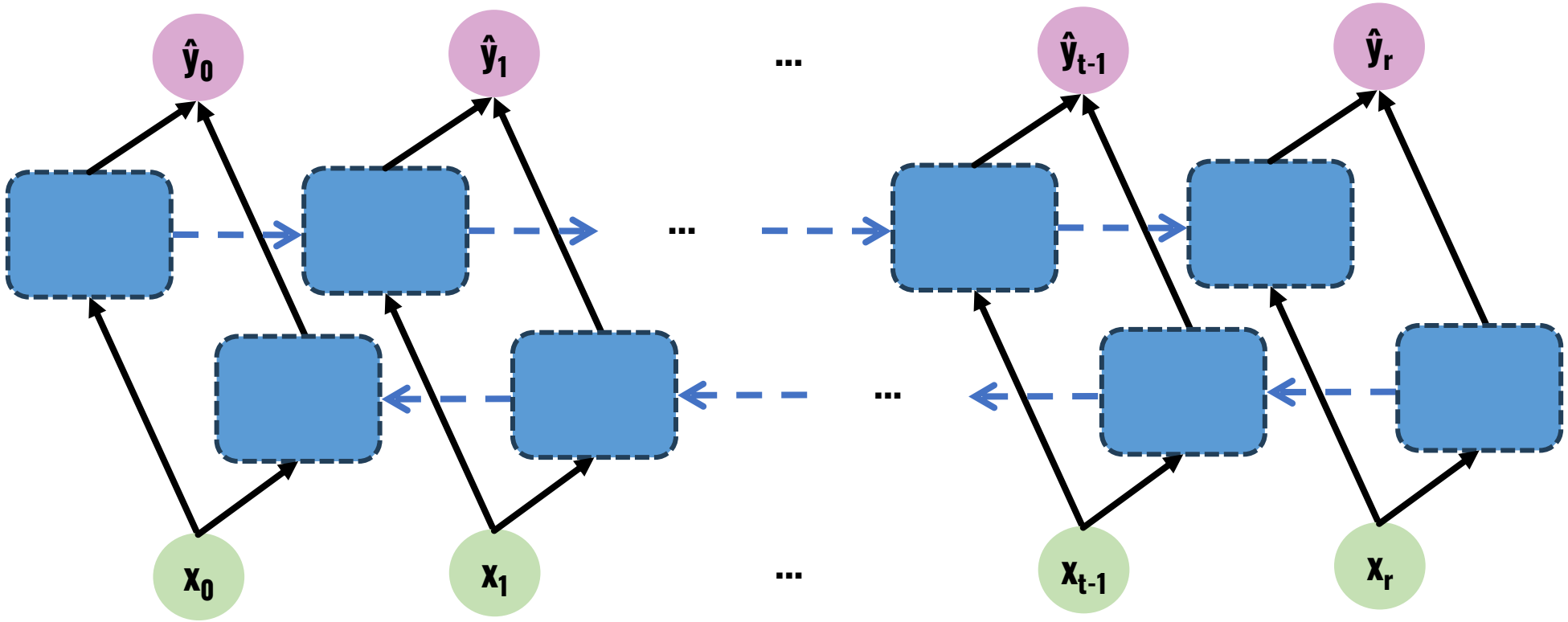
- E.g., time series and sequential data



Bidirectional RNN

Combines two independent recurrent models

- One processes the input in forward time order; the other in reverse
- Their outputs are merged at each time step by concatenation or summation



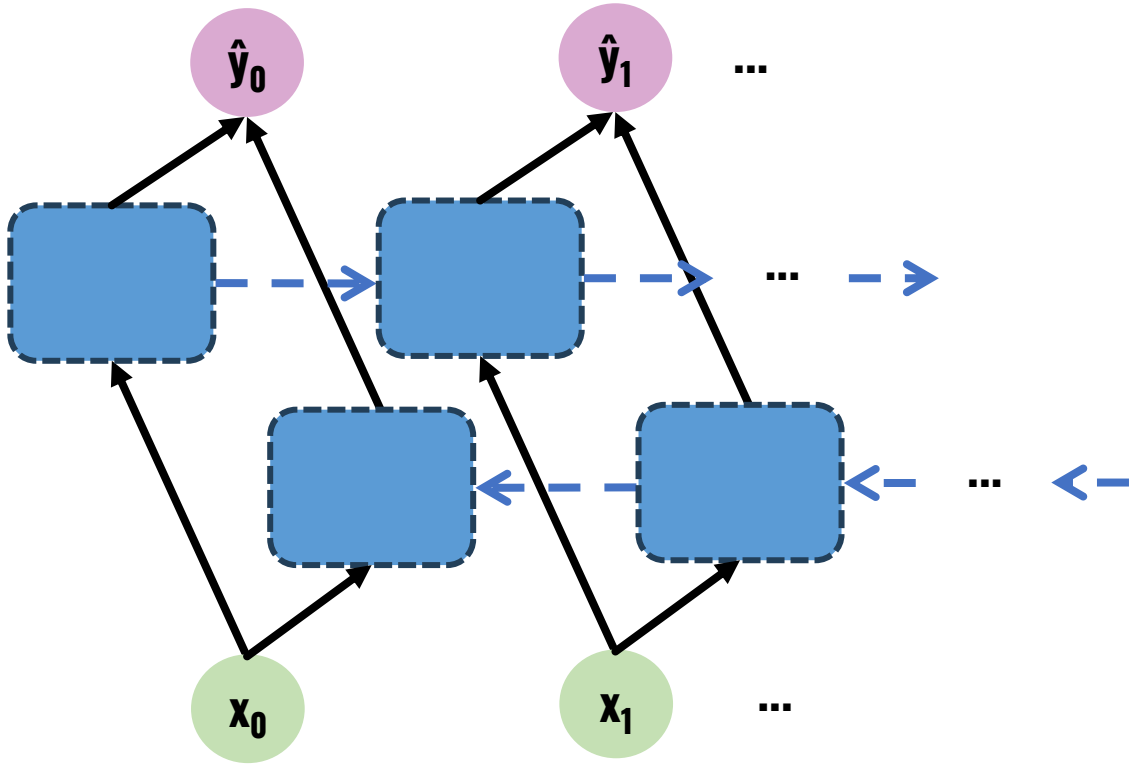
Bidirectional RNN (BRNN)

Hidden state:

$$\overrightarrow{h_t} = \tanh(\overrightarrow{W_{hh}}\,\overrightarrow{h_{t-1}} + \overrightarrow{W_{xh}}\,\overrightarrow{x_t} + \overrightarrow{b_h})$$

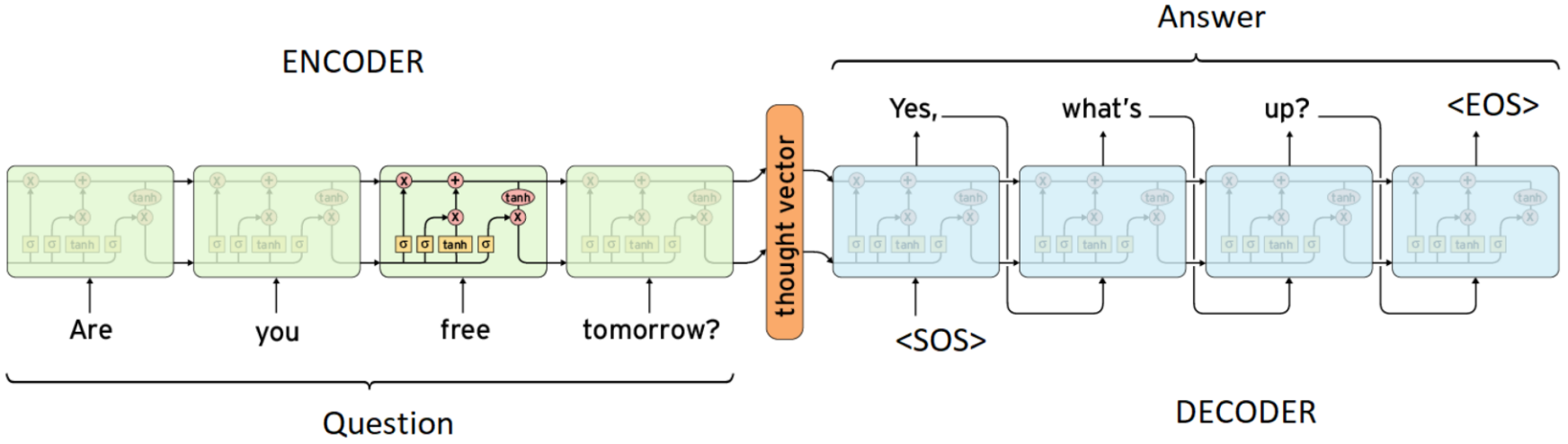
$$\overleftarrow{h_t} = \tanh(\overleftarrow{W_{hh}}\,\overleftarrow{h_{t+1}} + \overleftarrow{W_{xh}}\,\overleftarrow{x_t} + \overleftarrow{b_h})$$

$$h_t = \tanh(\overrightarrow{W_{hy}}\,\overrightarrow{h_t} + \overleftarrow{W_{hy}}\,\overleftarrow{h_t} + b_y)$$



Encoder – Decoder (Seq2Seq)

Question : Answer
(sequence) : (sequence)



Machine Translation
English : French
(sequence) : (sequence)

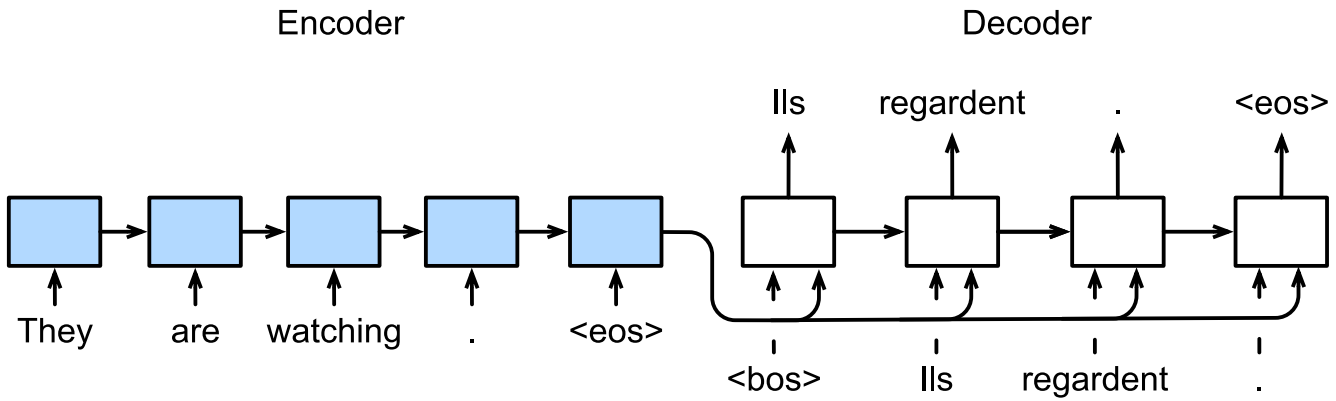
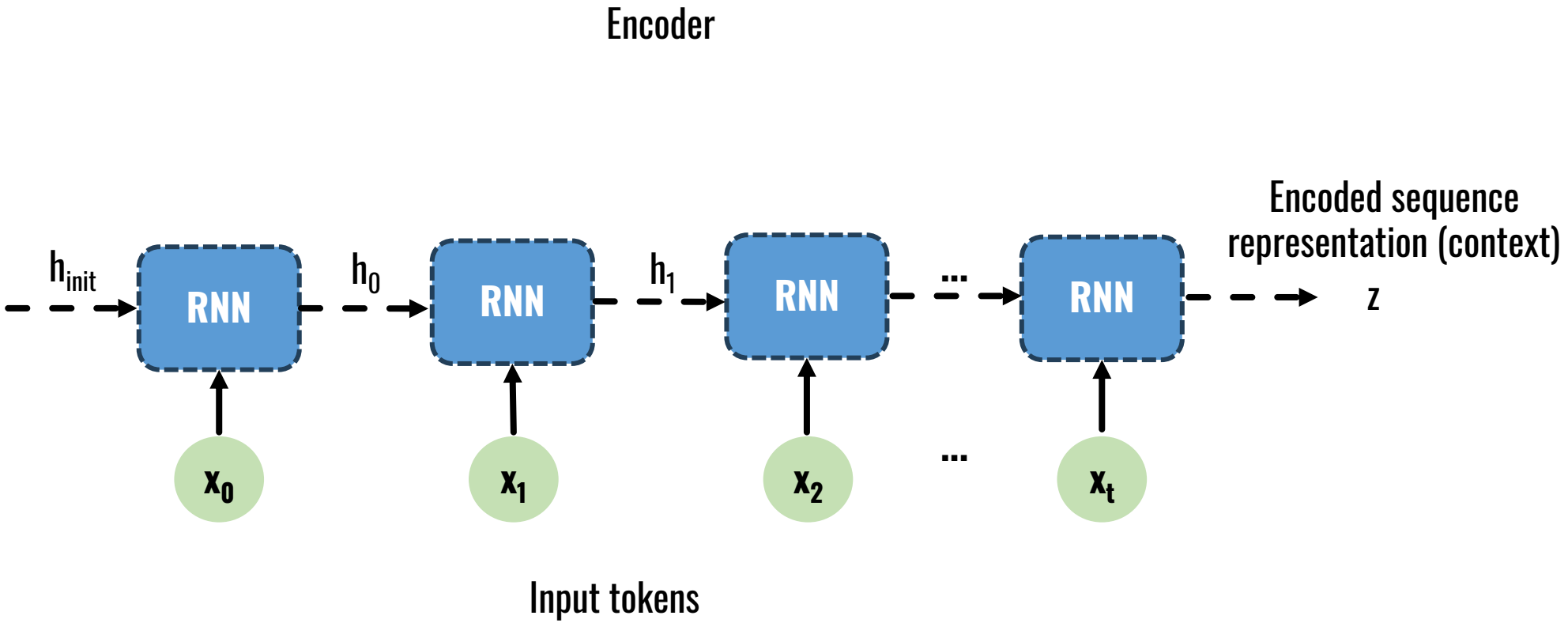


Image Source: The entire Seq2Seq chatbot architecture — Hosameldeen et al. (2023), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://www.preprints.org/manuscript/202303.0070
Image Source: Seq2seq-training.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

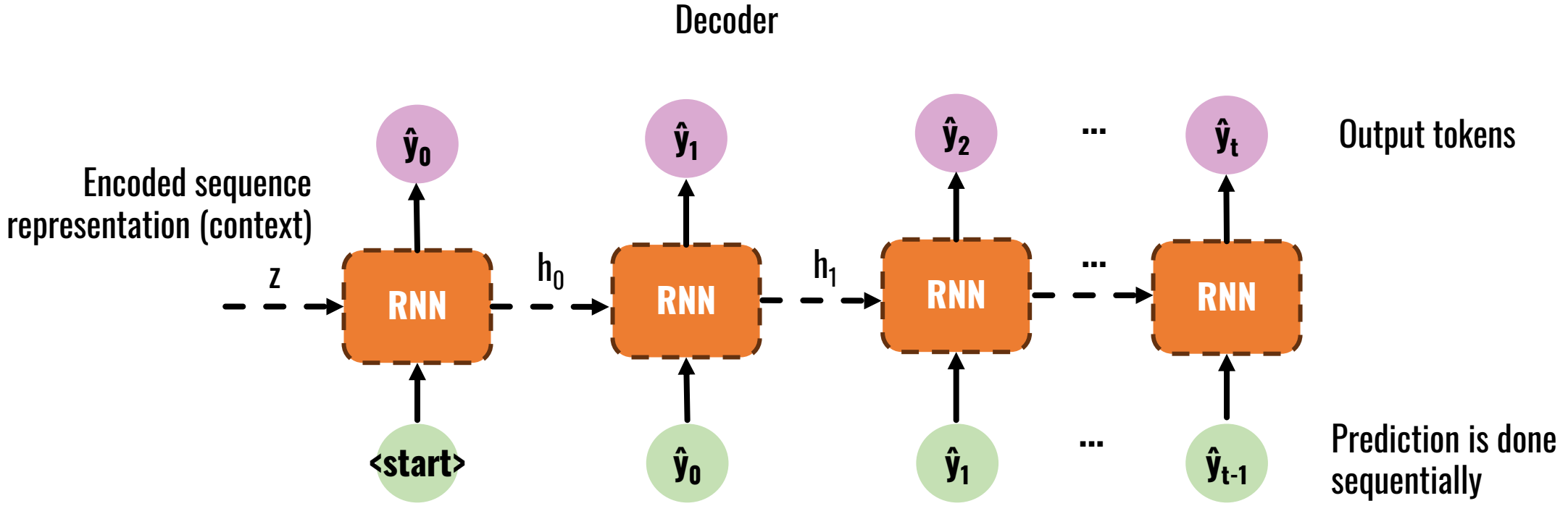
Encoder – Decoder (Seq2Seq)

- The encoder and decoder are nothing more than stacked RNN layers, such as LSTM's. The encoder processes the input and produces one compact representation, called z, from all the input timesteps. It can be regarded as a compressed format of the input.



Encoder – Decoder (Seq2Seq)

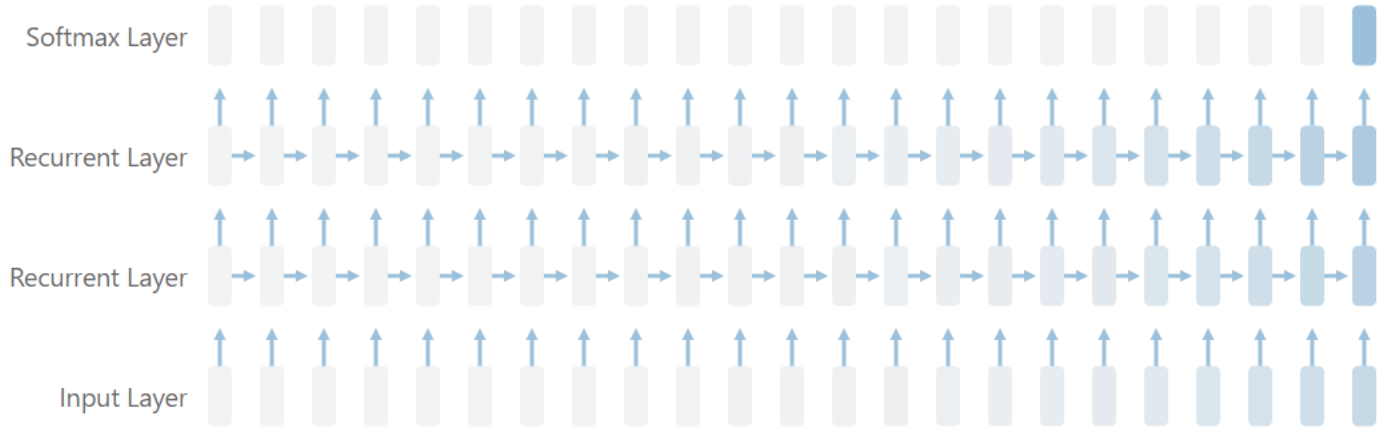
- On the other hand, the decoder receives the context vector z and generates the output sequence. The most common application of Seq2seq is language translation. We can think of the input sequence as the representation of a sentence in English and the output as the same sentence in French.



Encoder – Decoder (Seq2Seq)

Problems with the Seq2Seq

- The intermediate representation **z** cannot encode information from all the input timesteps. This is commonly known as the **bottleneck problem**. The vector z needs to capture all the information about the source sentence.
- In practice, how far we can see in the past (the so-called reference window) is finite. RNN's tend to **forget information** from timesteps that are far behind.



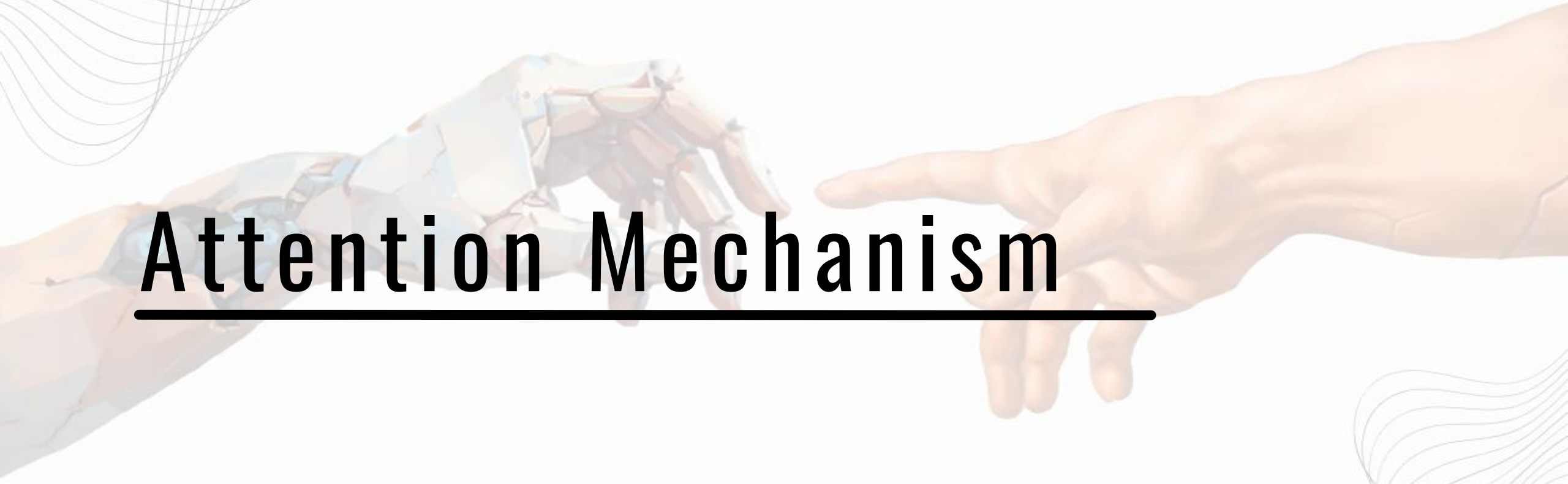
Vanishing Gradient: where the contribution from the earlier steps becomes insignificant in the gradient for the vanilla RNN unit.

Image Source: Vanishing Gradient — Andreas Madsen, Visualizing memorization in RNNs, Distill (2019), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://distill.pub/2019/memorization-in-rnns/

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 6

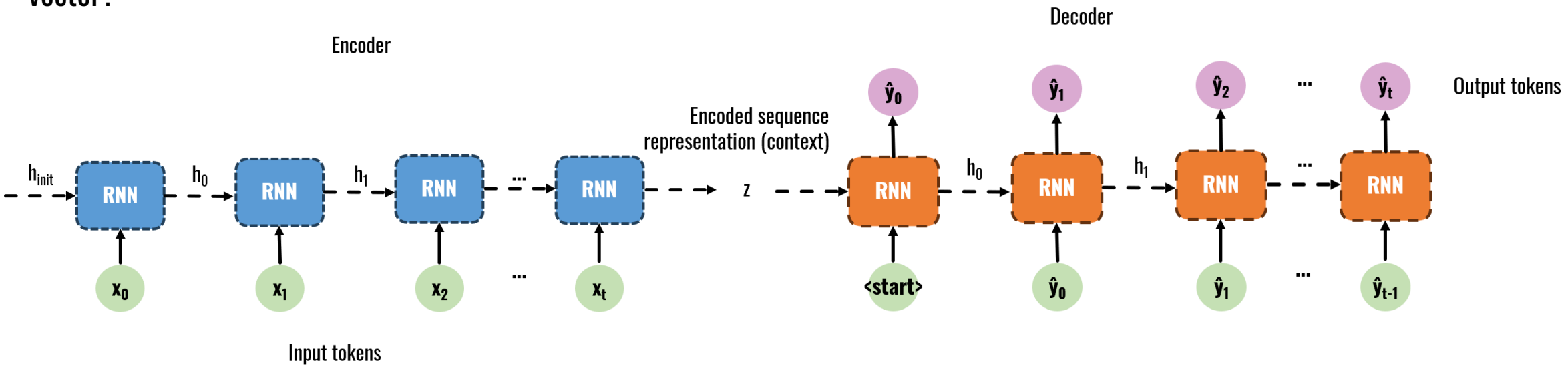


Attention Mechanism

1 RNNs and Embeddings 2 LSTM, GRU & Seq2Seq **3 Attention Mechanism**

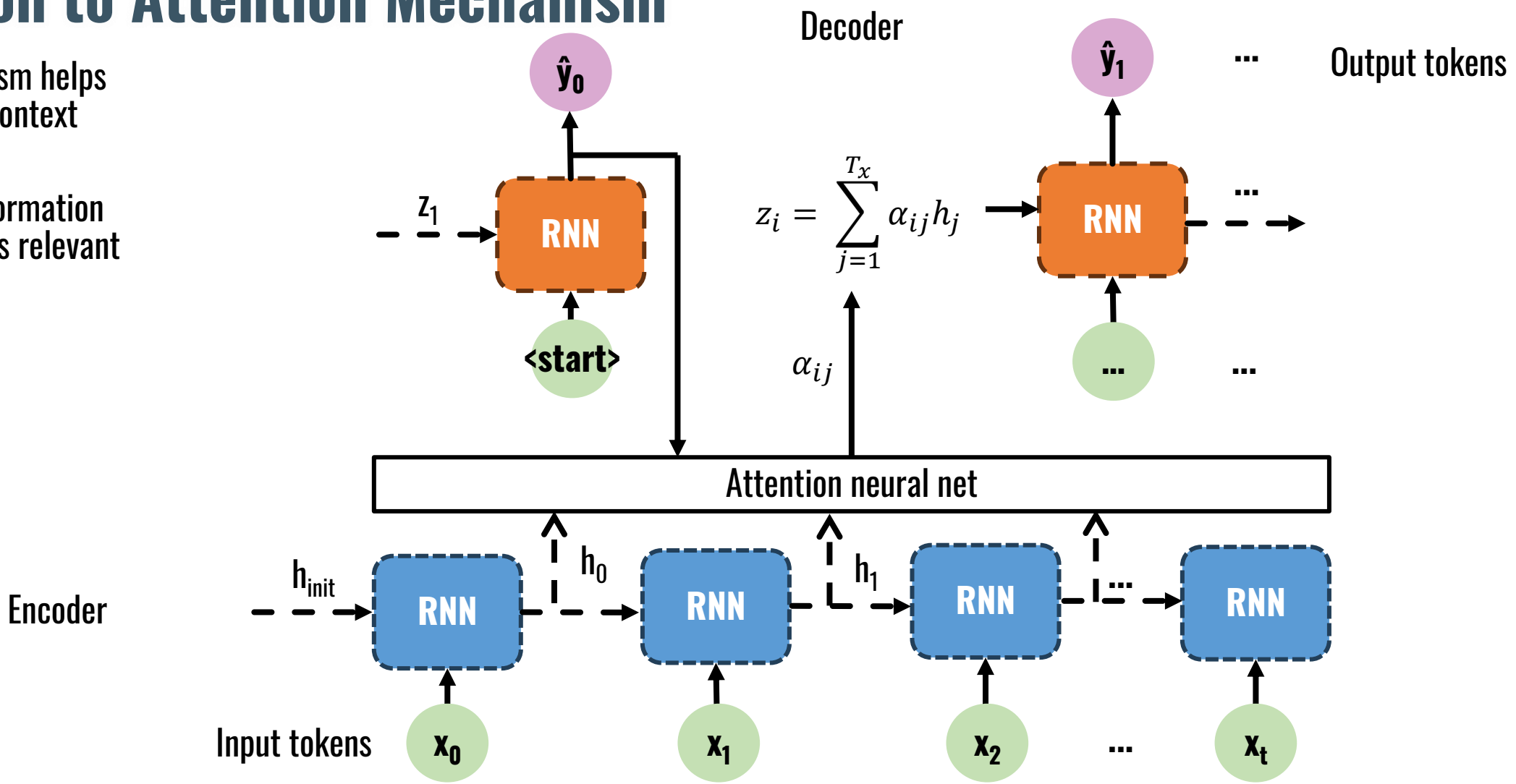
Introduction to Attention Mechanism

- All the information from the encoder is represented in the z vector (context)
- However, as seen previously, information from earlier timestamps is not preserved
- Can we create a better context vector?



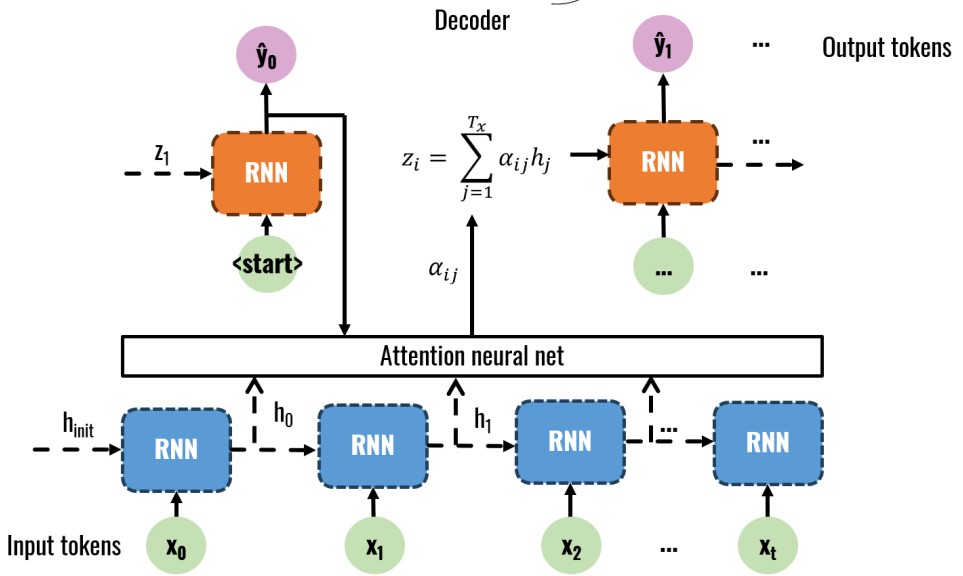
Introduction to Attention Mechanism

- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder



Introduction to Attention Mechanism

- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder



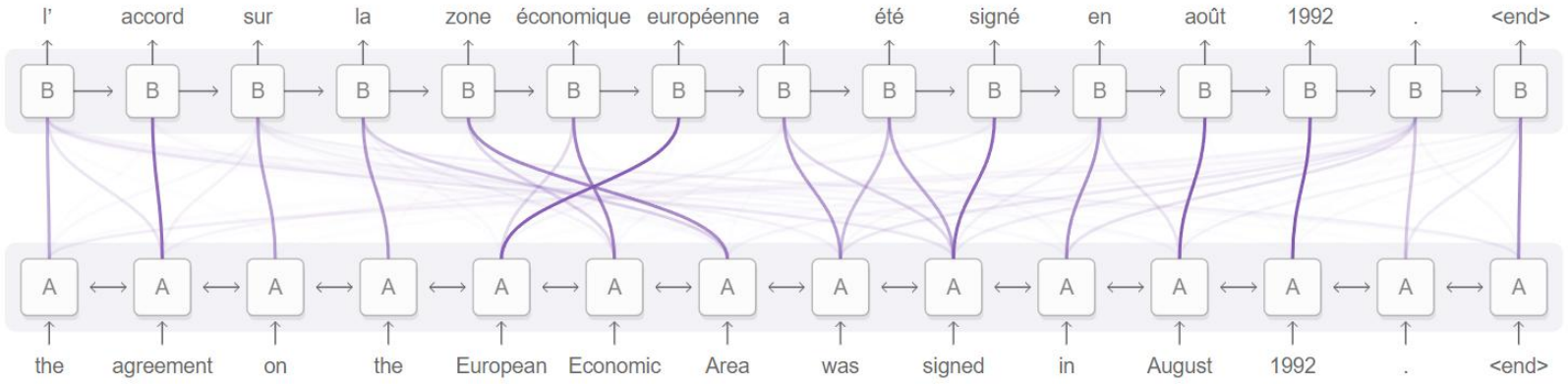


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

Image Source: Attention between RNNs — Chris Olah & Shan Carter, Attention and Augmented Recurrent Neural Networks, Distill (2016), derived from Fig. 3 of Bahdanau et al. (2014), CC BY 2.0 https://creativecommons.org/licenses/by/2.0/, https://distill.pub/2016/augmented-rnns

Introduction to Attention Mechanism

- The context vector **c** should have access to all parts of the input sequence instead of just the last one.
- In other words, we need to form a **direct connection** with each timestamp. We can look at all the different words at the same time and learn to "pay attention" to the correct ones depending on the task at hand.
- In the encoder-decoder:
  - Given the hidden states of the encoder at each time step h = h_1, h_2, ..., h_n
  - Given the previous state in the decoder y_{i-1}
  - We define an attention network that gives the attention scores for the current state (s) of the decoder

$$e_{ij} = \text{attention\_net}(s_{i-1}, h_j)$$

  - We convert this scores into probabilities

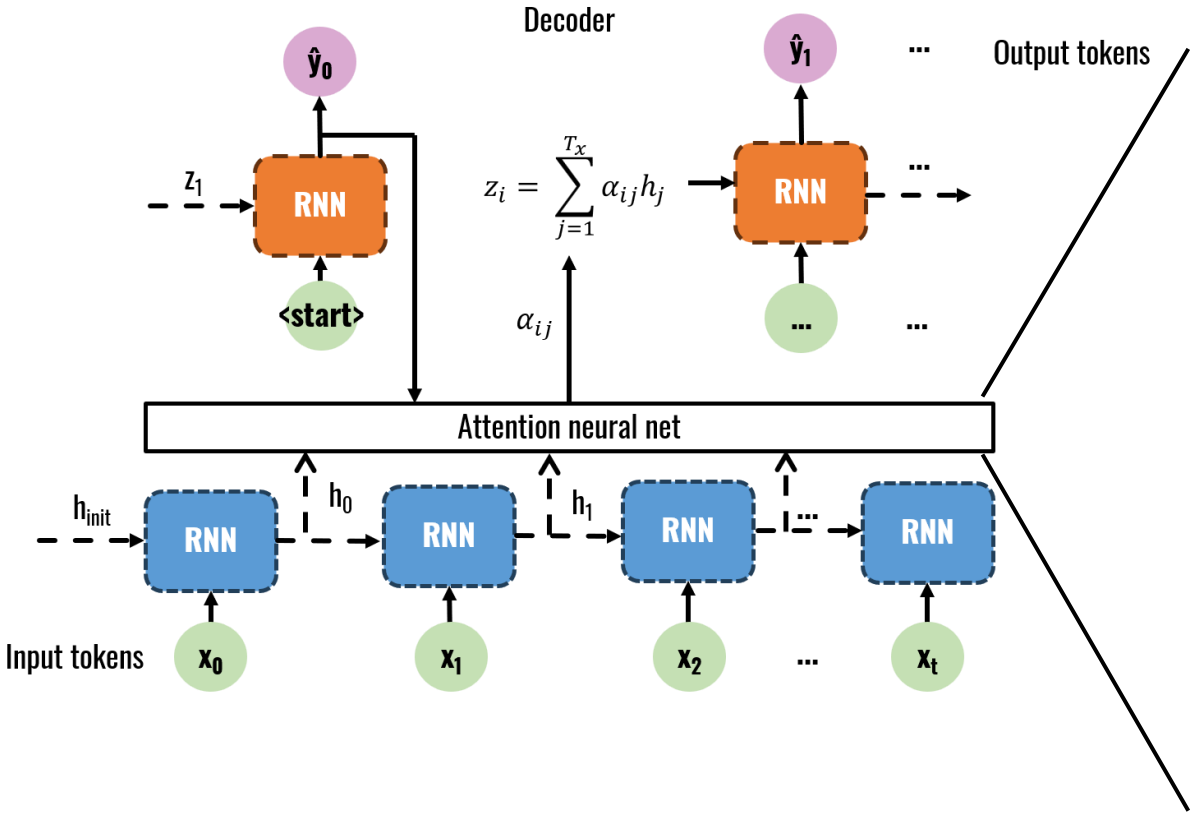
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

  - Finally, we get our new context vector c:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.0473

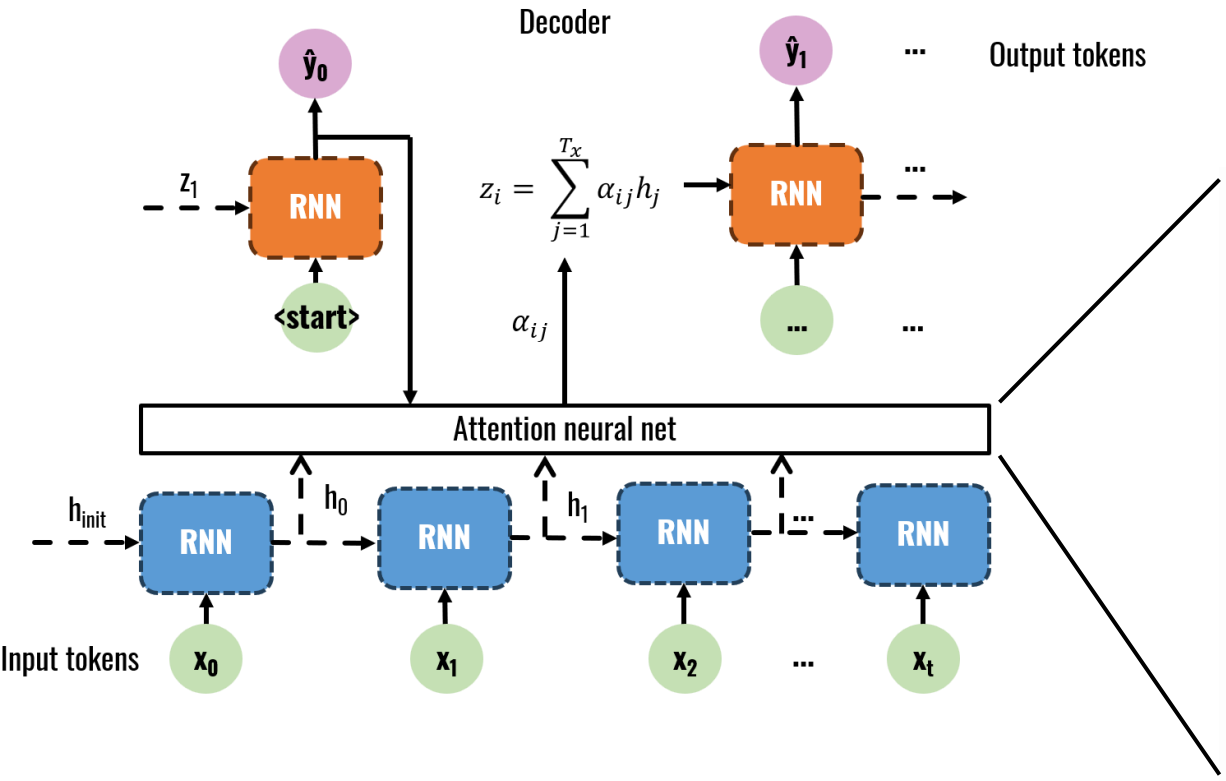
Attention Mechanism



Additive(*)	$score(s_t, h_i) = v_a^\intercal \tanh(W_a[s_t; h_i]$	Bahdanau et al.
Location-Base	$a_{t,i} = \text{softmax}(W_a s_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong et al.
General	$score(s_t, h_i) = s_t^\intercal W_a h_i$ where W_a is a trainable weight matrix in the attention layer.	Luong et al.
Scaled Dot-Product(^)	$score(s_t, h_i) = \frac{s_t^\intercal h_i}{\sqrt{n}}$ Note: Very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani et al.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.0473
Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1508.04025
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762

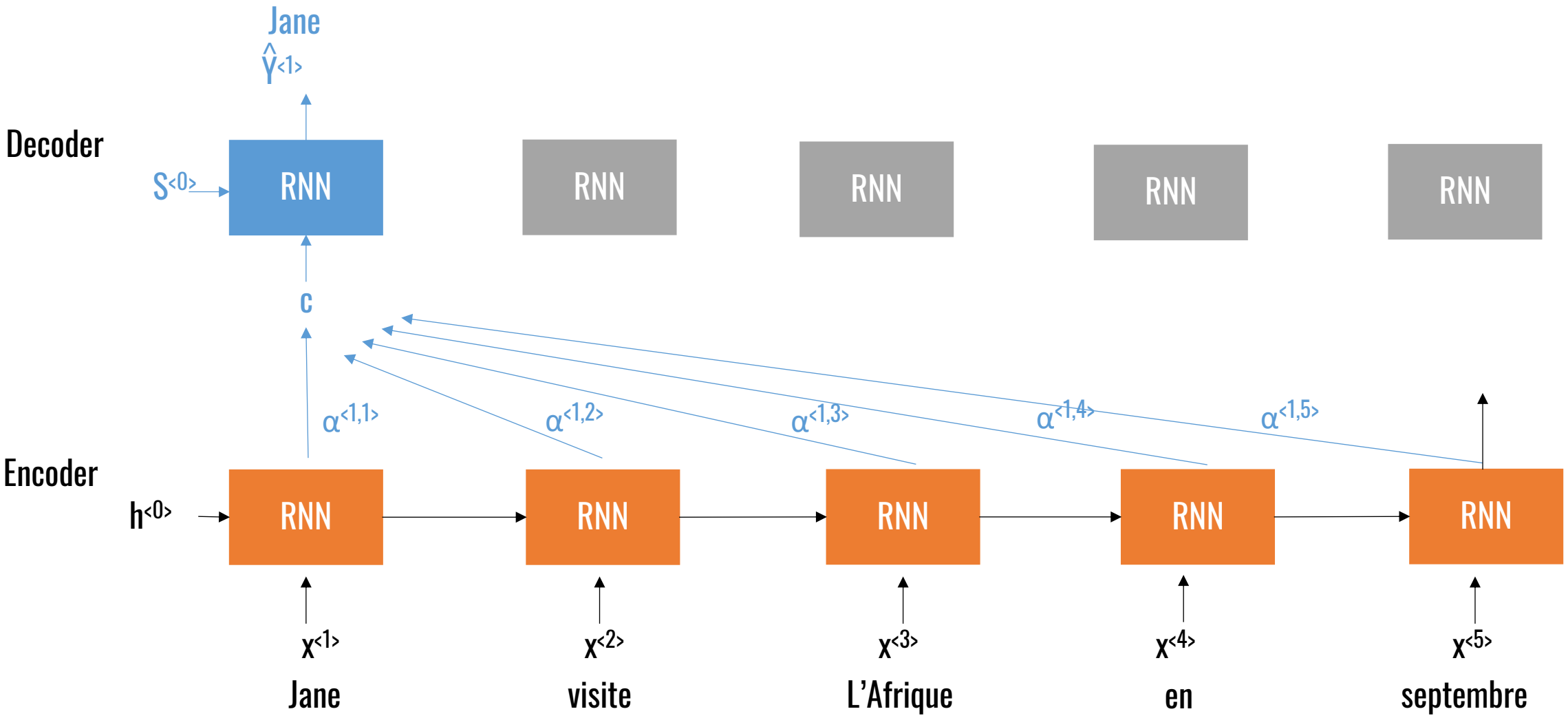
Attention Mechanism



Several ways to calculate the scores

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

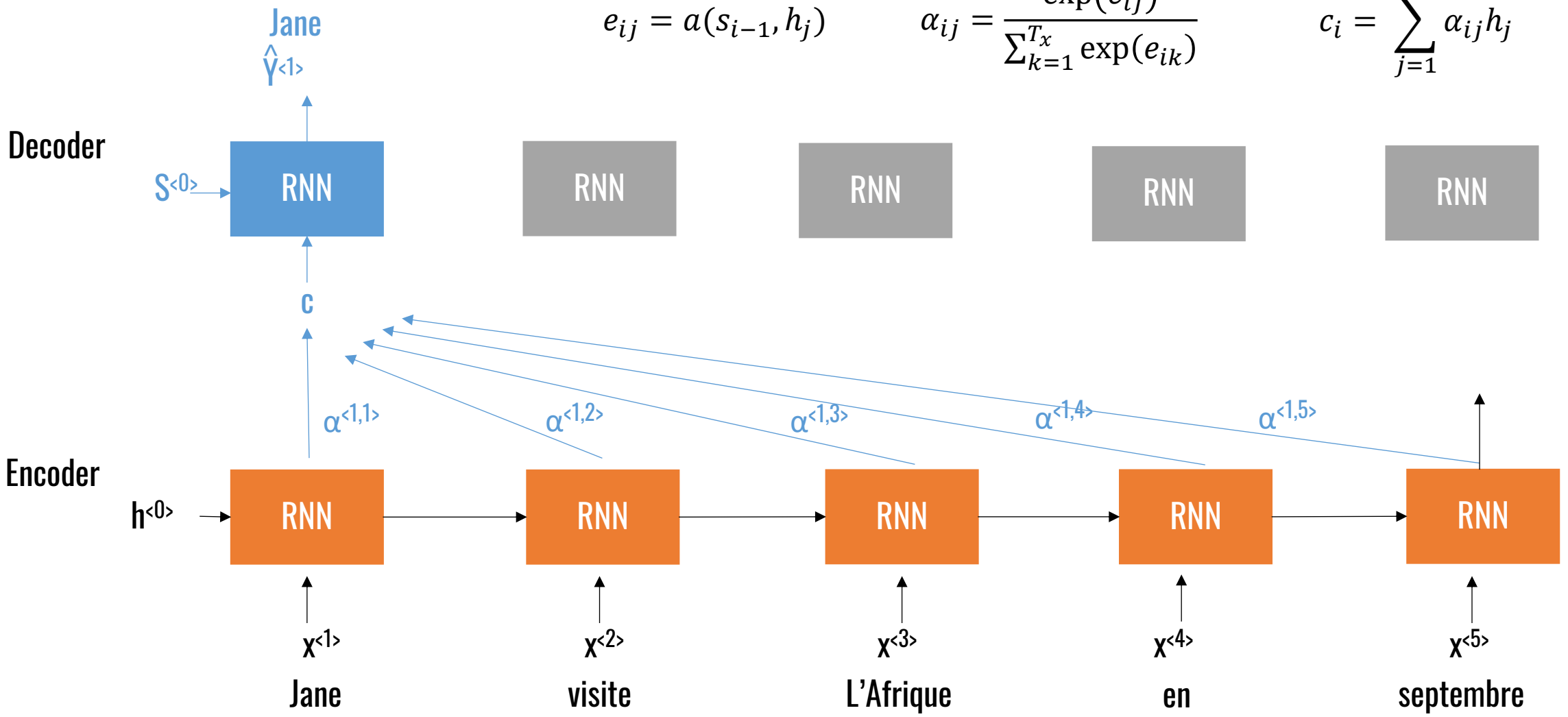
Attention Mechanism



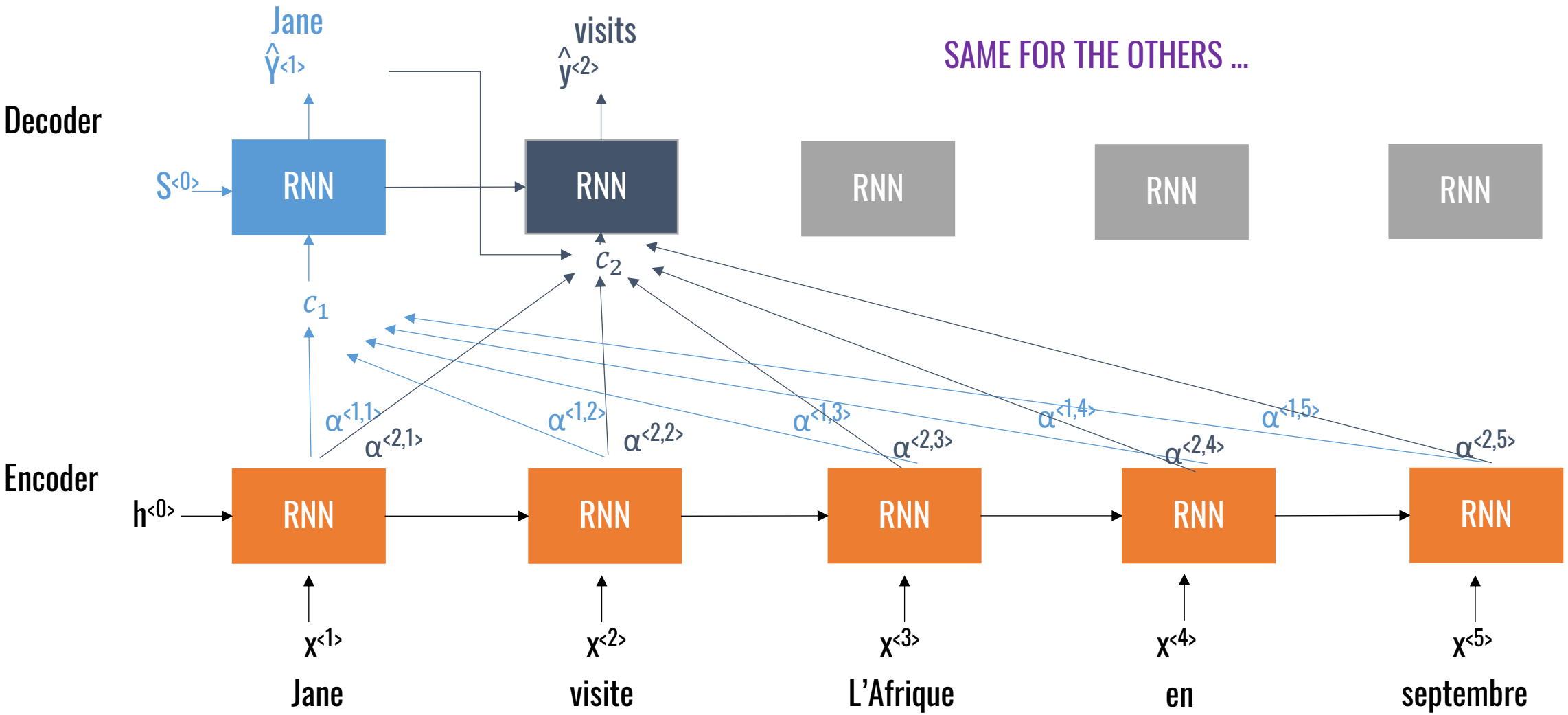
$\alpha^{<t,t'>}$ = amount of "attention" $\hat{y}^{<t>}$ should pay to $h^{<t'>}$

Attention Mechanism

$$e_{ij} = a(s_{i-1}, h_j) \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$



Attention Mechanism



Next Lecture

We will continue next lecture with Attention and Transformers…

Summary

- **Sequential data** is important because it carries temporal information and context
- **Recurrent Neural Networks:**
  - Sequence based models
  - Handle variable length sequences
  - Track dependencies
  - Maintain the order of the input
  - Share parameters across sequences
- RNNs have limitations: **vanishing gradients and short memory**

- Other architectures like **LSTM and GRU** improve the limitations of RNNs
  - Include gates to control the flow of information

- **Seq2Seq** models are encoder-decoder based architectures
  - The context vector from the encoder is limited
- **Attention mechanism provides a better context by allowing the network to pay attention to every part of the input** / have access to all hidden states
  - It computes a score / weight that tells the relevance of each part

Datasets

- Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition](https://aclanthology.org/W03-0419/) (Tjong Kim Sang & De Meulder, CoNLL 2003)
- https://github.com/flairNLP/CleanCoNLL
- SQuAD: 100,000+ Questions for Machine Comprehension of Text](https://aclanthology.org/D16-1264/) (Rajpurkar et al., EMNLP 2016)
- A large annotated corpus for learning natural language inference](https://aclanthology.org/D15-1075/) (Bowman et al., EMNLP 2015)
- Europarl: A Parallel Corpus for Statistical Machine Translation](https://aclanthology.org/2005.mtsummit-papers.11/) (Koehn, MTSummit 2005)

Resources

- https://developers.google.com/machine-learning/crash-course/embeddings/embedding-space
- https://distill.pub/2016/augmented-rnns/
- https://distill.pub/2019/memorization-in-rnns/

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning

Papers

1. Vennerød, C. B., Kjærran, A., & Bugge, E. S. (2021). Long Short-term Memory RNN. In arXiv [cs.LG]. arXiv. https://doi.org/10.48550/arXiv.2105.06756
2. Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.1259
3. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.0473
4. Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1508.04025
5. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762

Image Sources

- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.

- Dancing Robot: Gif robot.git (https://commons.wikimedia.org/wiki/File:Gif_robot.gif), Meks01, CC BY-SA 4.0, via Wikimedia Commons
- Raccoon Family DSC 0037.jpg (https://commons.wikimedia.org/wiki/File:Raccoon_Family_DSC_0037.jpg), Tracie Hall, CC BY-SA 4.0, via Wikimedia Commons
- Raccoon in a garbage can in Prospect Park (72564).jpg (https://commons.wikimedia.org/wiki/File:Raccoon_in_a_garbage_can_in_Prospect_Park_(72564).jpg), Rhododendrites, CC BY-SA 4.0, via Wikimedia Commons
- DNA RNA structure (full).png (https://commons.wikimedia.org/wiki/File:DNA_RNA_structure_(full).png), Thomas Shafee, CC BY 4.0, via Wikimedia Commons

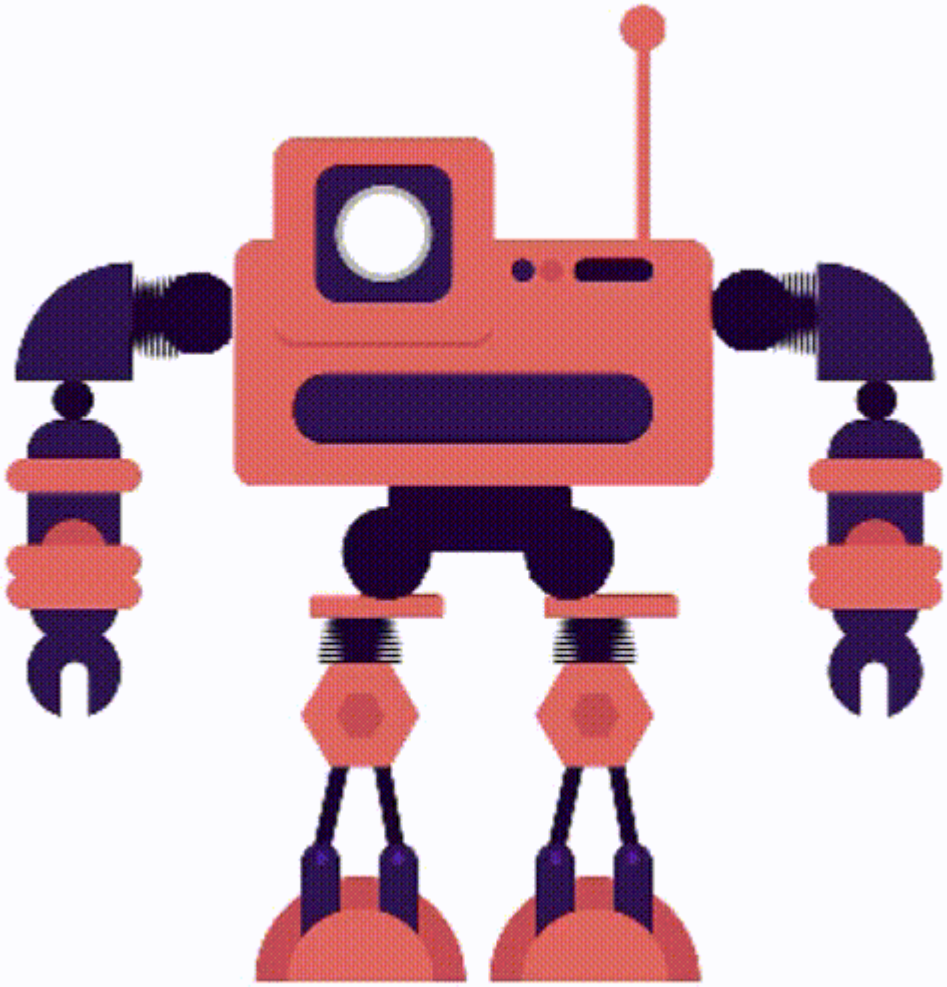
Image Sources

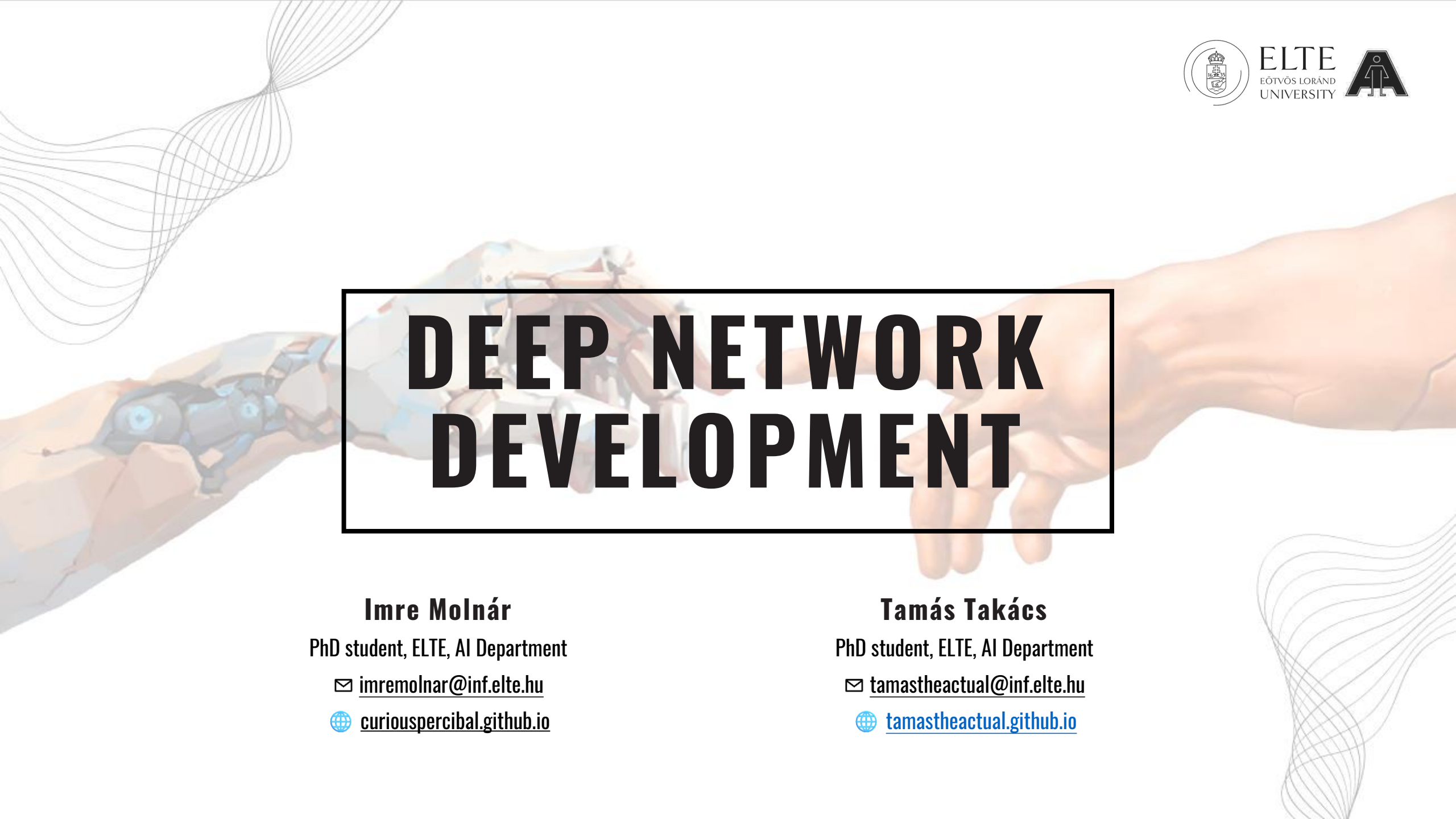
- Signal-speech-martin-de.png (https://commons.wikimedia.org/wiki/File:Signal-speech-martin-de.png), MartinThoma, CC0, via Wikimedia Commons
- Video frames of the Parallel Bars action category.png (https://commons.wikimedia.org/wiki/File:Video_frames_of_the_Parallel_Bars_action_category.png), Bikingdog, CC BY-SA 4.0, via Wikimedia Commons
- P3030027ParkingLot wb.jpg (https://commons.wikimedia.org/wiki/File:P3030027ParkingLot_wb.jpg), Elf (assumed), CC BY-SA 3.0, via Wikimedia Commons
- T-SNE visualisation of word embeddings generated using 19th century literature.png (https://commons.wikimedia.org/wiki/File:T-SNE_visualisation_of_word_embeddings_generated_using_19th_century_literature.png), Sioban Grayson, CC BY-SA 4.0, via Wikimedia Commons
- T-SNE Embedding of MNIST.png (https://commons.wikimedia.org/wiki/File:T-SNE_Embedding_of_MNIST.png), Kyle McDonald, CC BY 2.0, via Wikimedia Commons

Image Sources

- Review of Natural Bridge State Park on Trip advisor (30352819436).png (https://commons.wikimedia.org/wiki/File:Review_of_Natural_Bridge_State_Park_on_Trip_advisor_(30352819436).png) , vastateparksstaff, CC BY 2.0, via Wikimedia Commons
- Google Translate Polish-Japanese.png (https://commons.wikimedia.org/wiki/File:Google_Translate_Polish-Japanese.png), Zuckergloy, CC BY-SA 4.0, via Wikimedia Commons
- The entire Seq2Seq chatbot architecture — Hosameldeen et al. (2023), CC BY 4.0, https://www.preprints.org/manuscript/202303.0070
- Seq2seq-training.svg (https://commons.wikimedia.org/wiki/File:Seq2seq-training.svg), Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY 4.0, via Wikimedia Commons
- Vanishing Gradient — Andreas Madsen, Visualizing memorization in RNNs, Distill (2019), CC BY 4.0, https://distill.pub/2019/memorization-in-rnns/
- Attention between RNNs — Chris Olah & Shan Carter, Attention and Augmented Recurrent Neural Networks, Distill (2016), derived from Fig. 3 of Bahdanau et al. (2014), CC BY 2.0, https://distill.pub/2016/augmented-rnns/

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 7.

Transformer Networks, Vision Transformers

1 Natural Language Understanding

2 Attention Mechanism

3 Transformer Architecture

4 Vision Transformers

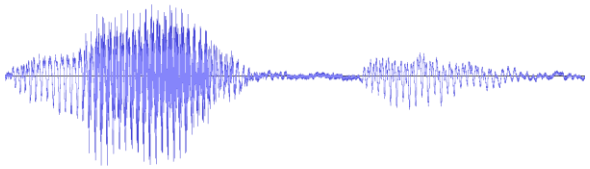
4 Transformer Network

Sequential Data Processing

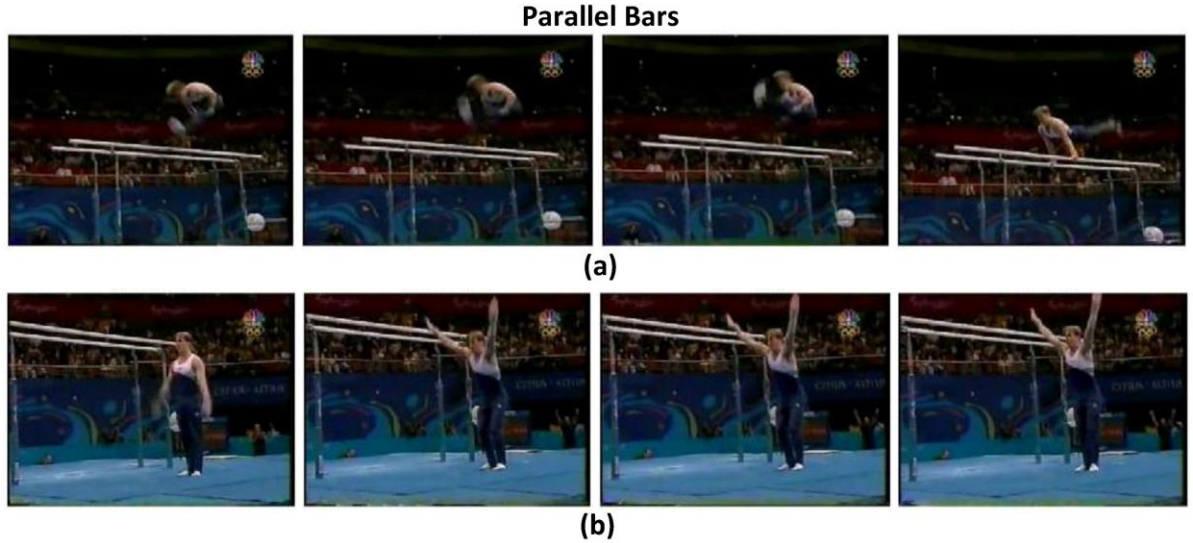
Sequential data

Text – sequence of words / characters

Speech – sequence of signals / acoustic features



Video – sequence of images (frames)



DNA – sequence of symbols (nucleotides)

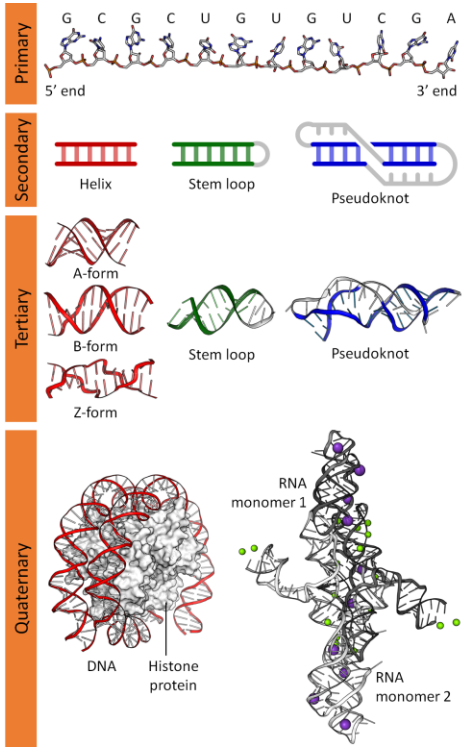
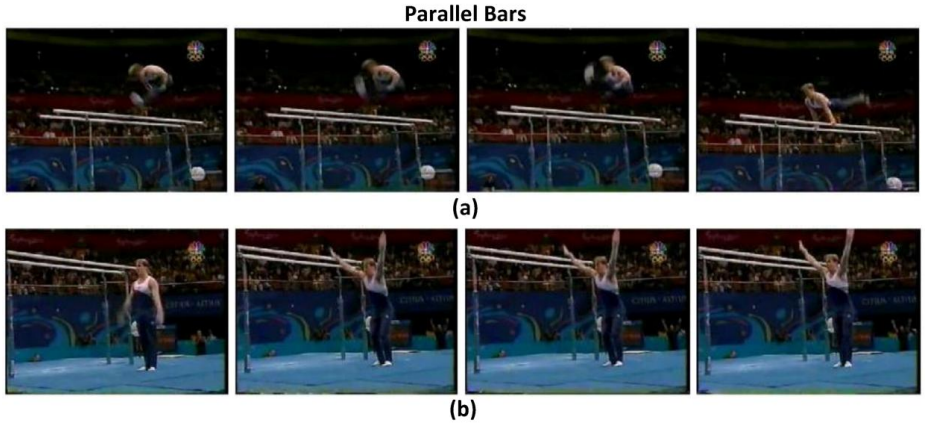


Image Source: DNA RNA structure (full).png, Thomas Shafee, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
Image Source: Signal-speech-martin-de.png, MartinThoma, CC0, via Wikimedia Commons
Image Source: Video frames of the Parallel Bars action category.png, Bikingdog, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Recap

Key properties of sequential data

- Order
- Temporal information
- Context

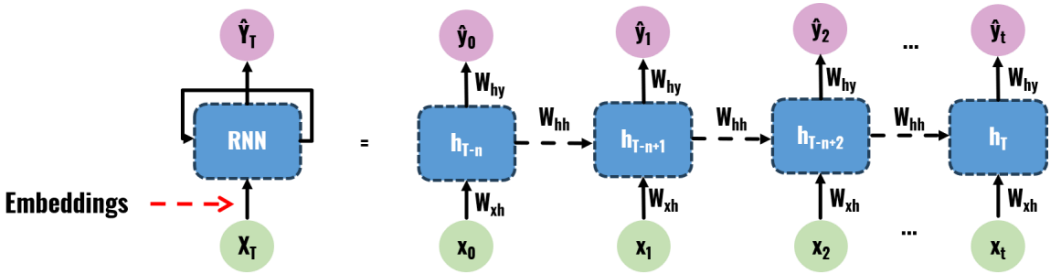




"Let's swim to the opposite bank."

Key model properties:

- Handle variable length sequences
- Track long term dependencies
- Maintain the order of the input
- Share parameters across sequences



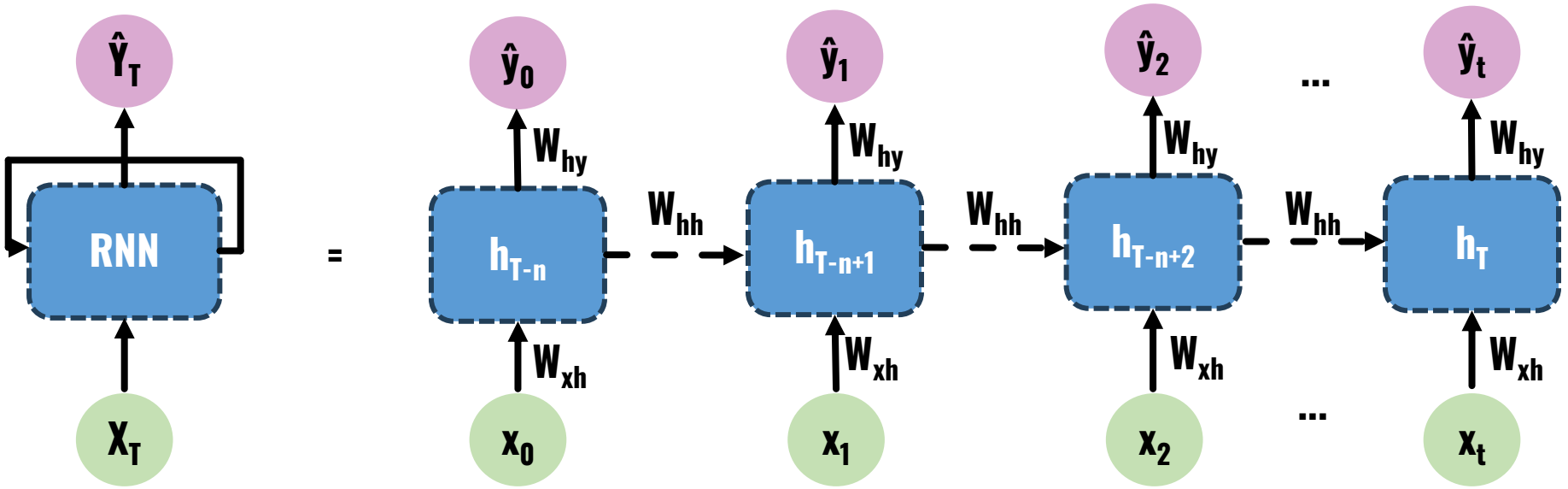


“The bank will lend us money.”

Image Source: Video frames of the Parallel Bars action category.png, Bikingdog, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Recurrent Neural Network (review)

RNN (unrolled version)



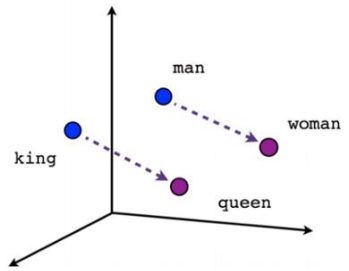
Embeddings: Word representation (review)

Featurized representation: Word Embedding

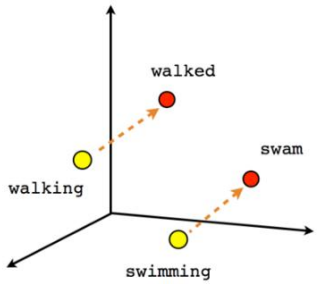
If we subtract man and woman, main difference is gender

We can compute word similarities

We can compute word analogies: man is to woman as king is to ___



Male-Female



Verb tense

	Man (5391)	Woman (9853)	King (4914)	Queen (7151)	Apple (456)	Orange (6257)
Gender	**-1**	**1**	**-0.95**	**0.97**	0.00	0.01
Royal	0.01	0.02	**0.93**	**0.95**	-0.01	0.00
Age	0.03	0.02	**0.7**	**0.68**	0.03	-0.02
Food	0.04	0.01	0.02	0.01	**0.95**	**0.97**
…	…	…	…	…	…	…

Embeddings: Word representation (review)

Featurized representation: Word Embedding

The word embeddings are learned with training.

Therefore, in practice, the features aren't that understandable.

We can visualize lower representations of the embeddings with techniques such as T-SNE

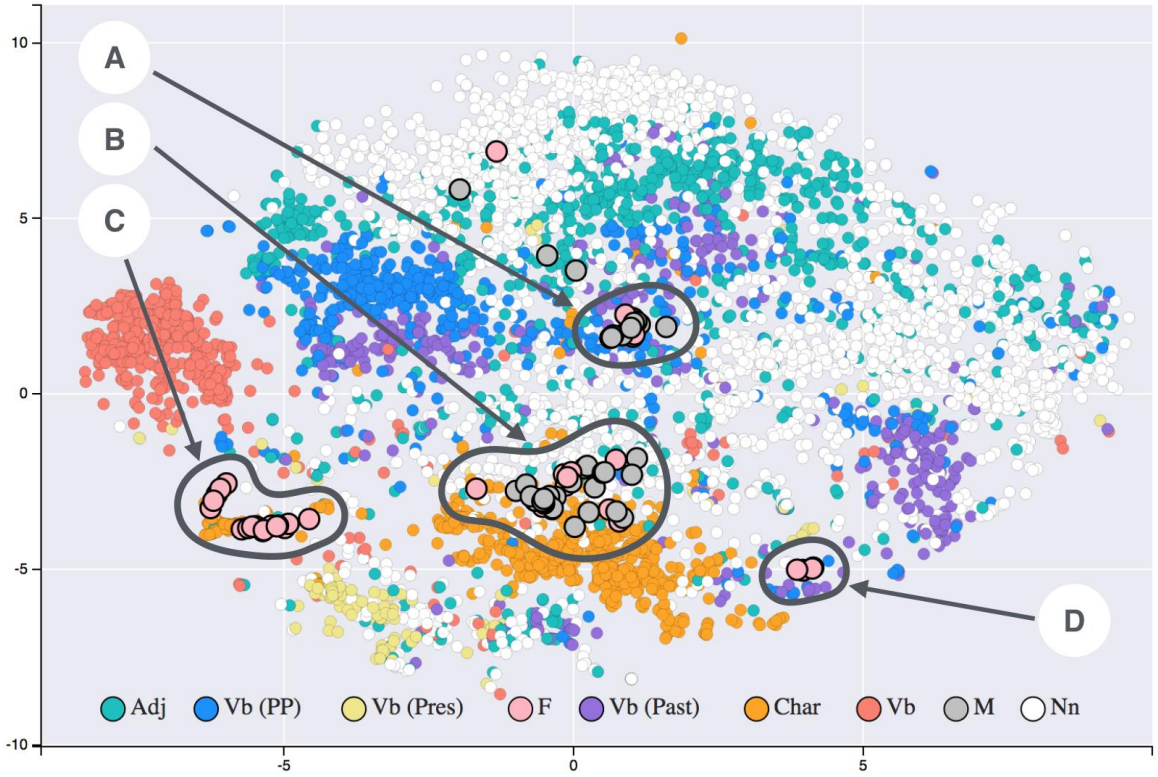
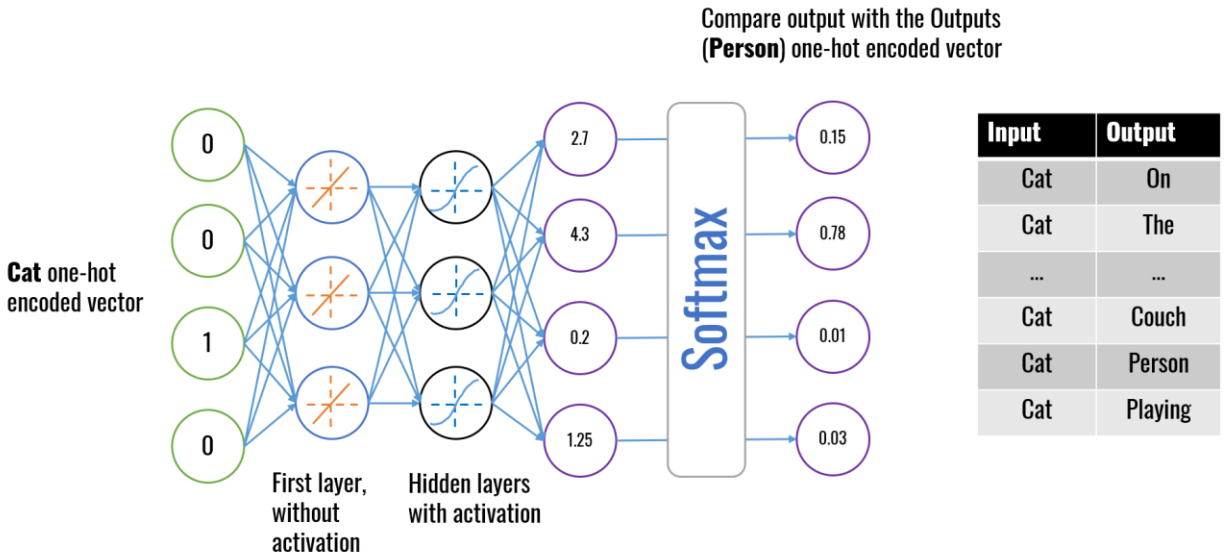


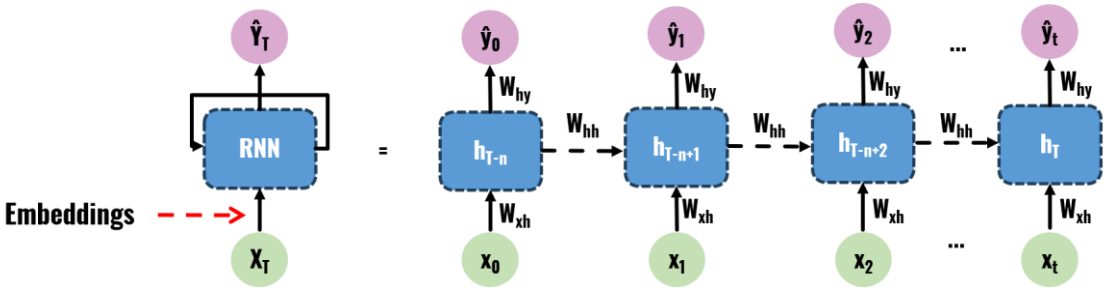
Image Source: T-SNE visualisation of word embeddings generated using 19th century literature.png, Siobhán Grayson, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Embeddings: Word representation (review)

RNN (unrolled version)

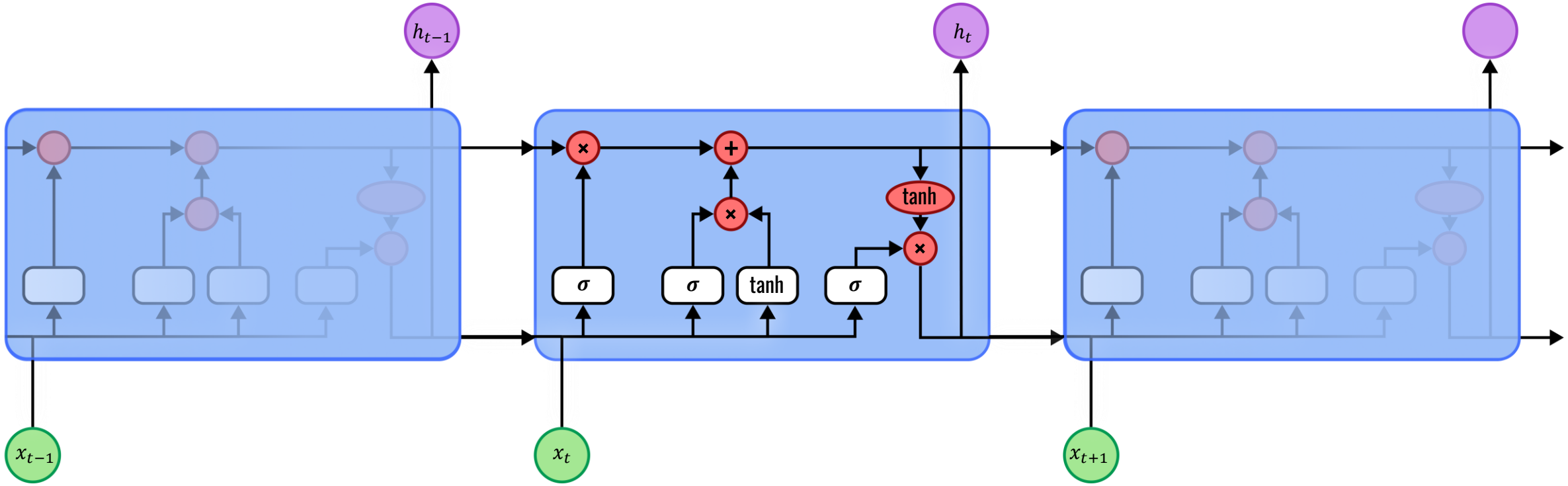


Input	Output
Cat	On
Cat	The
...	...
Cat	Couch
Cat	Person
Cat	Playing



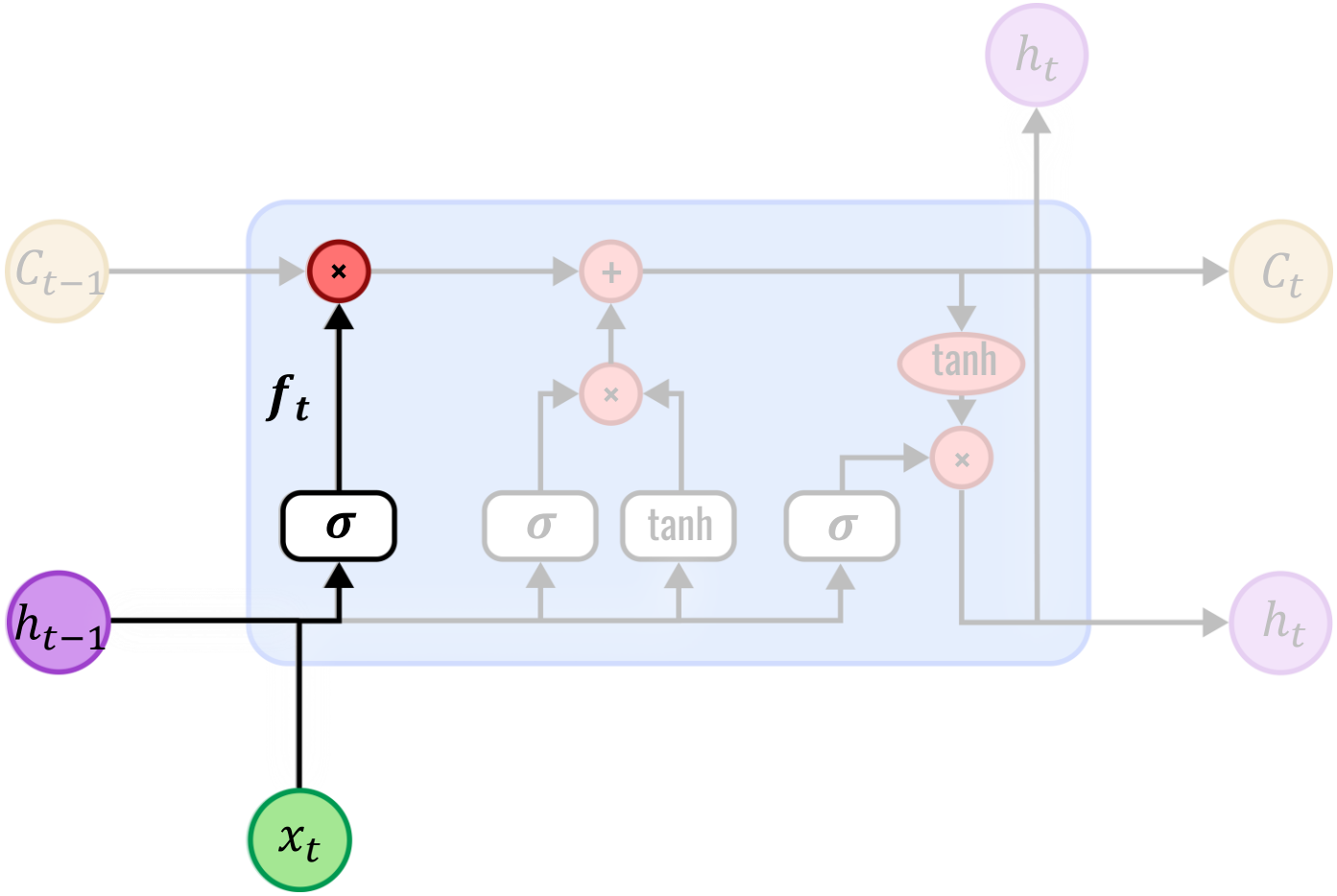
Long Short-Term Memory (LSTM) (review)

LSTM – a more complex network



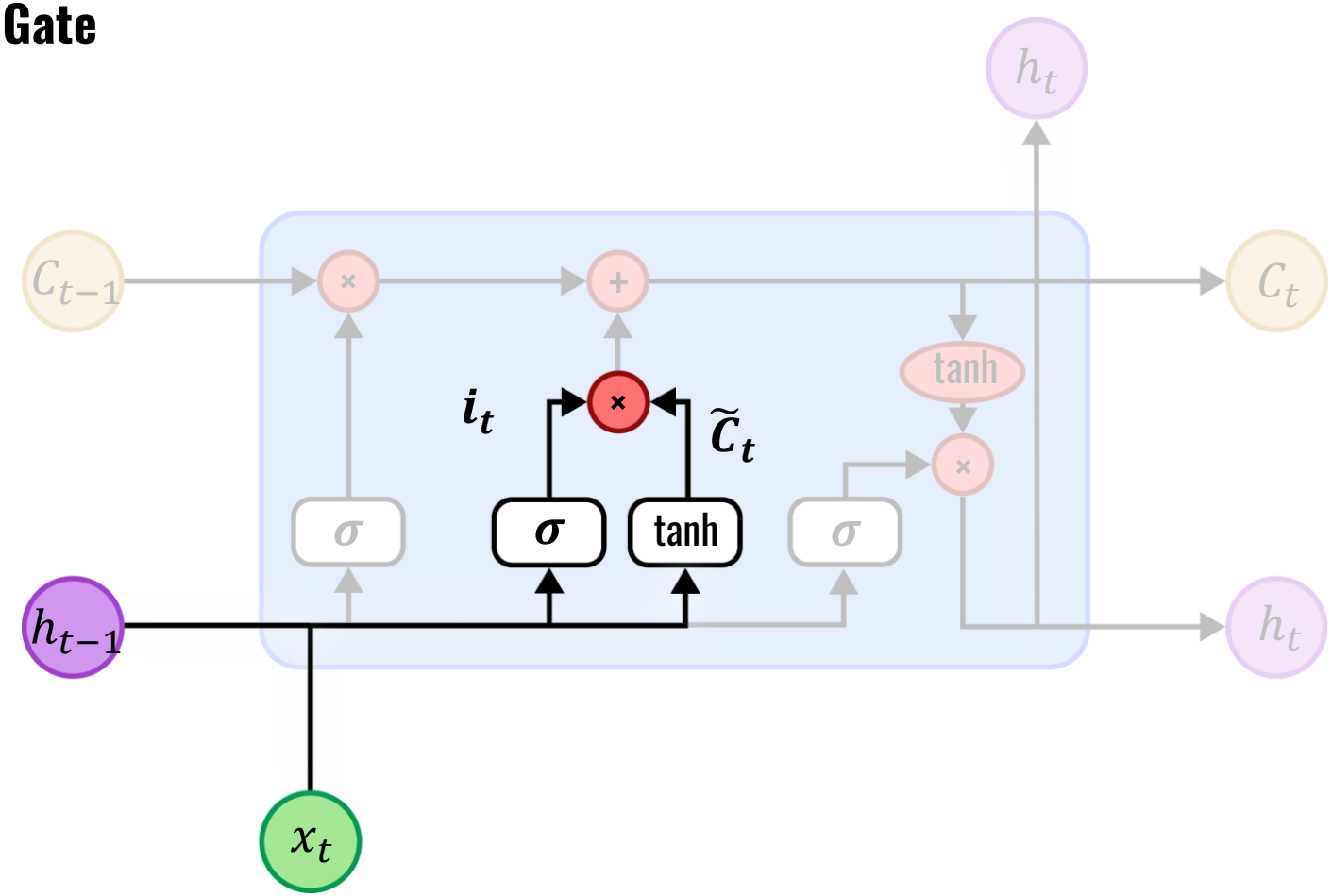
Long Short-Term Memory (LSTM) (review)

LSTM – Forget Gate



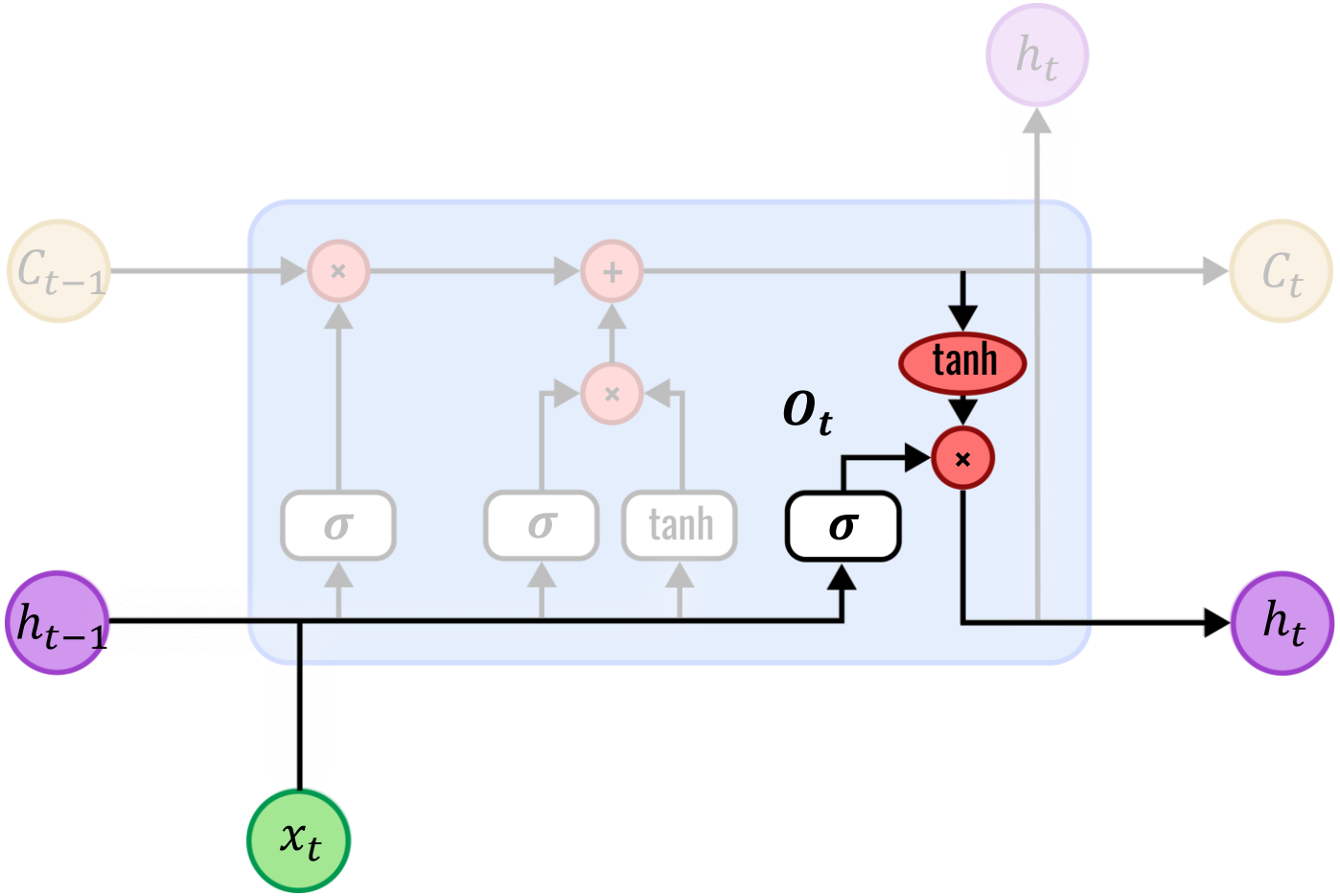
Long Short-Term Memory (LSTM) (review)

LSTM – Input / Ignore Gate



Long Short-Term Memory (LSTM) (review)

LSTM – Output Gate



Long Short-Term Memory (LSTM)

Gates:

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f)$$
$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t + b_i)$$
$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo} + b_o)$$
$$\widetilde{C}_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Output:

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t$$
$$h_t = o_t \odot \tanh(C_t)$$

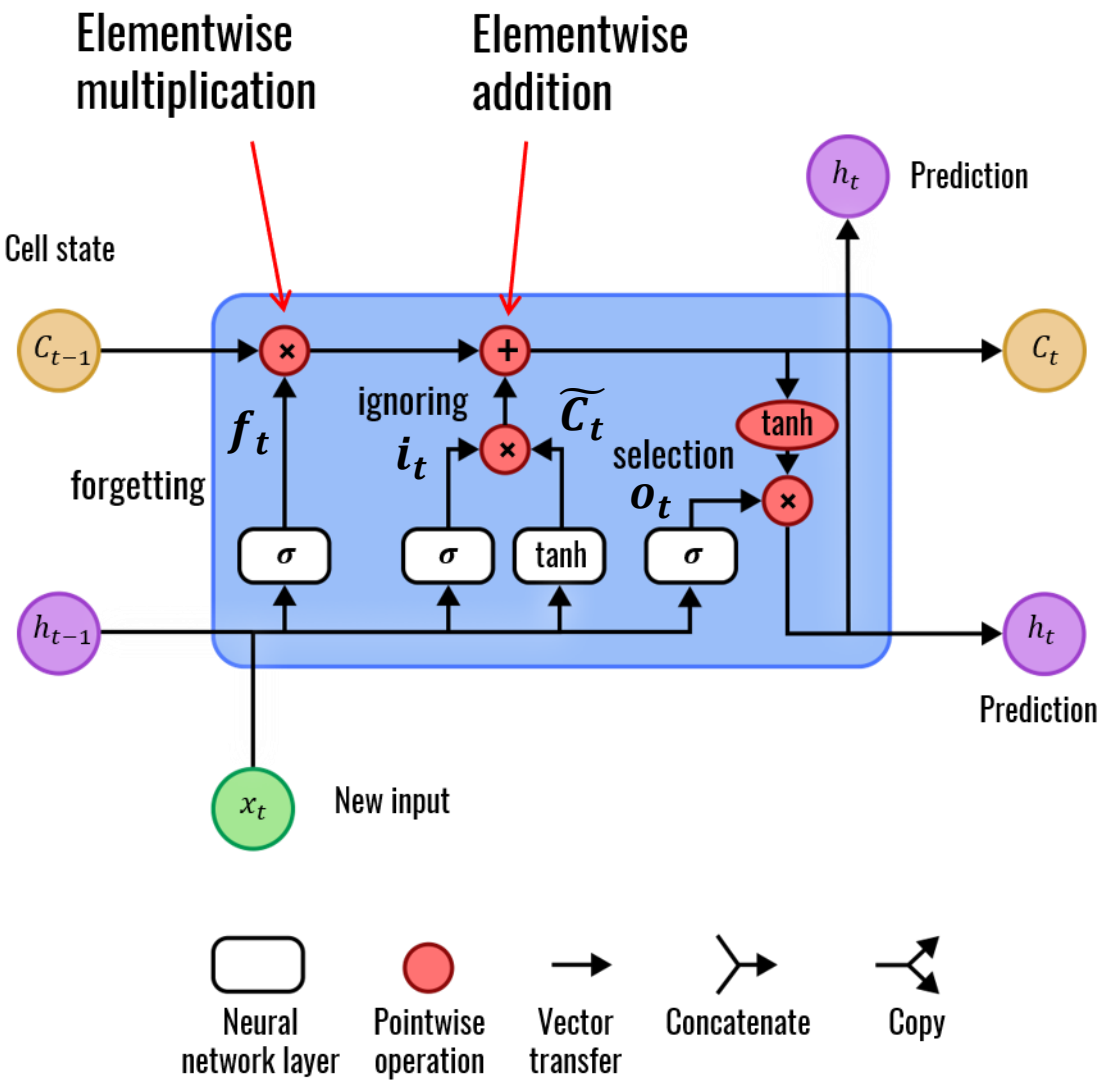
Where $\odot$ is an elementwise operation

$\mathbf{f_t}$ - forget gate - erase cell

$\mathbf{i_t}$ - ignore gate - what to ignore

$\mathbf{o_t}$ - output gate - whether to reveal cell

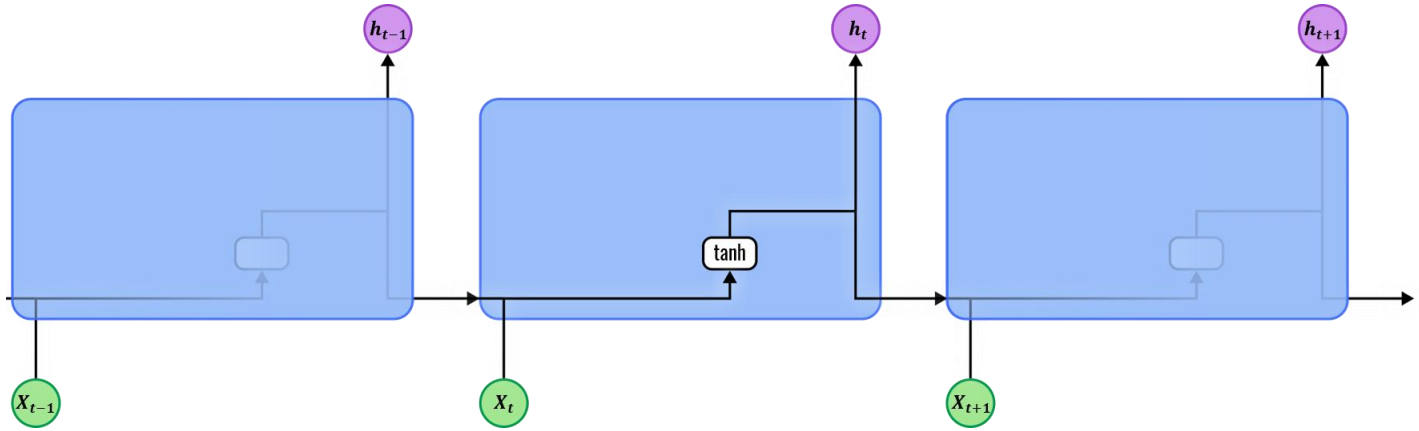
$\widetilde{\mathbf{C}}_\mathbf{t}$ - "Vanilla RNN" - how much to write to cell



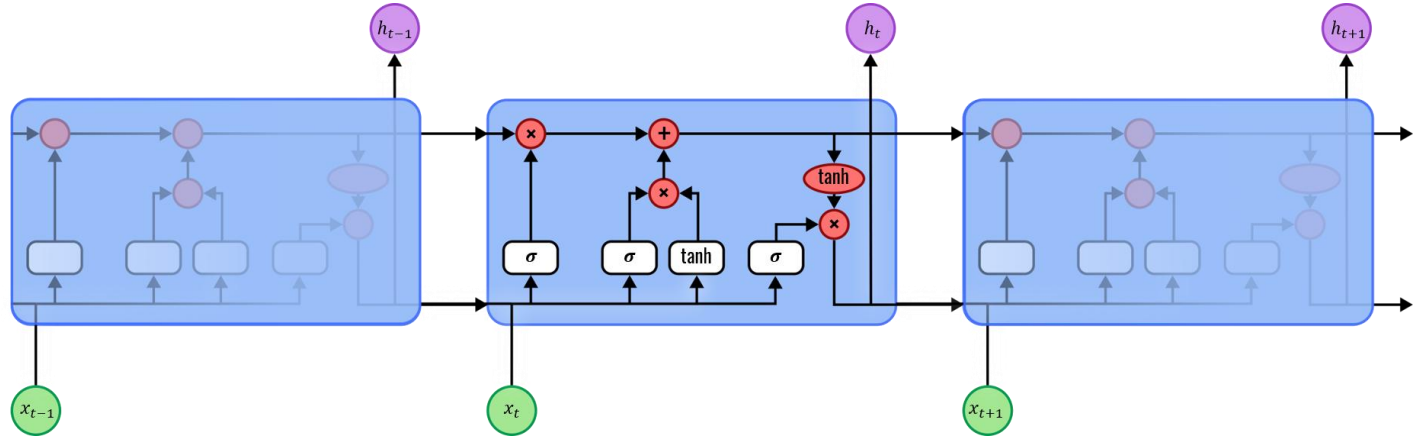
Long Short-Term Memory (LSTM)

LSTM – a more complex network

Vanilla RNN:



LSTM:



Gated Recurrent Unit (GRU)

$$r_t = \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r)$$

$$z_t = \sigma(x_t W_{xz} + h_{t-1} W_{hz} + b_z)$$

$$\tilde{h}_t = \tanh(x_t W_{xh} + (r_t \odot h_{t-1}) W_{hh} + b_h)$$

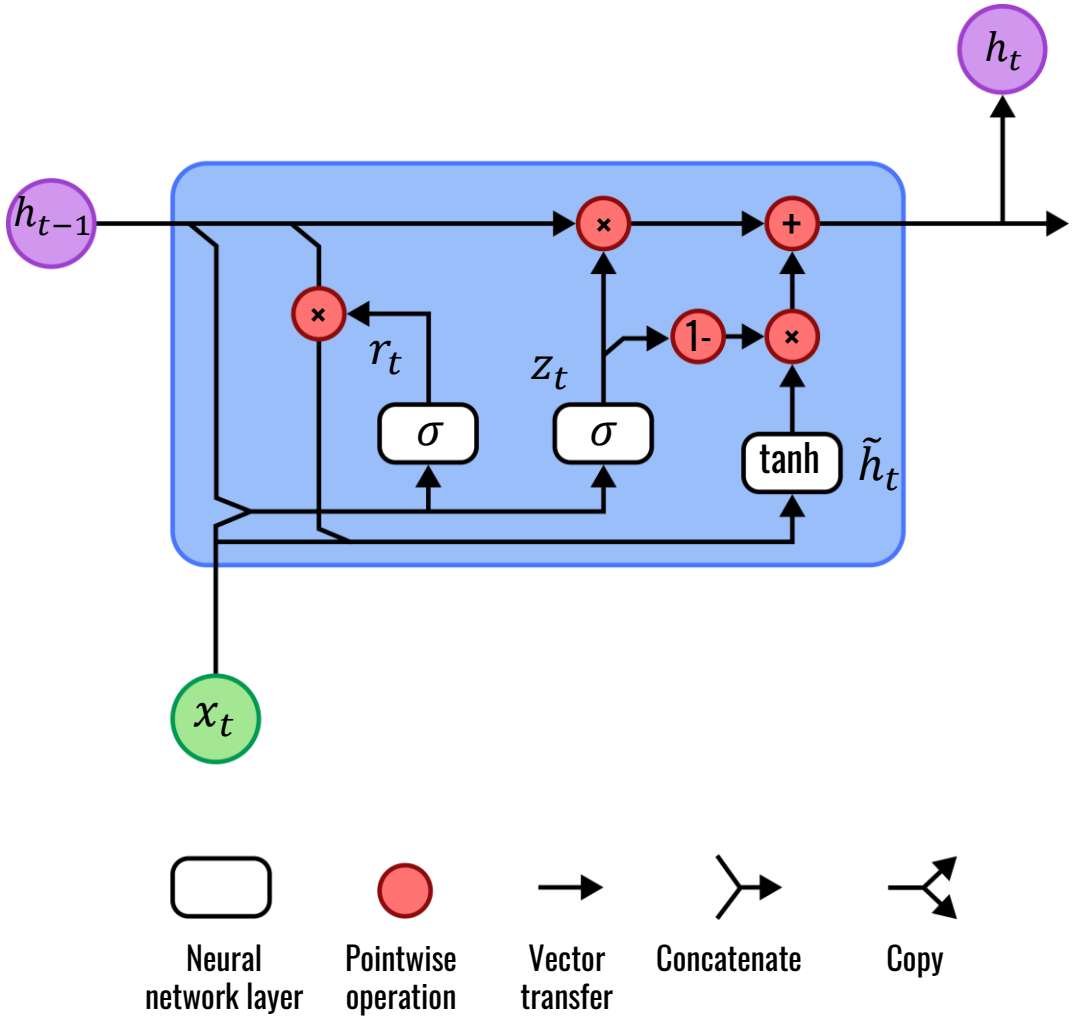
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

where

r_t- reset gate – what information to use for the update

z_t- update gate – how much of the update to apply

$\tilde{h}_t$- candidate hidden state

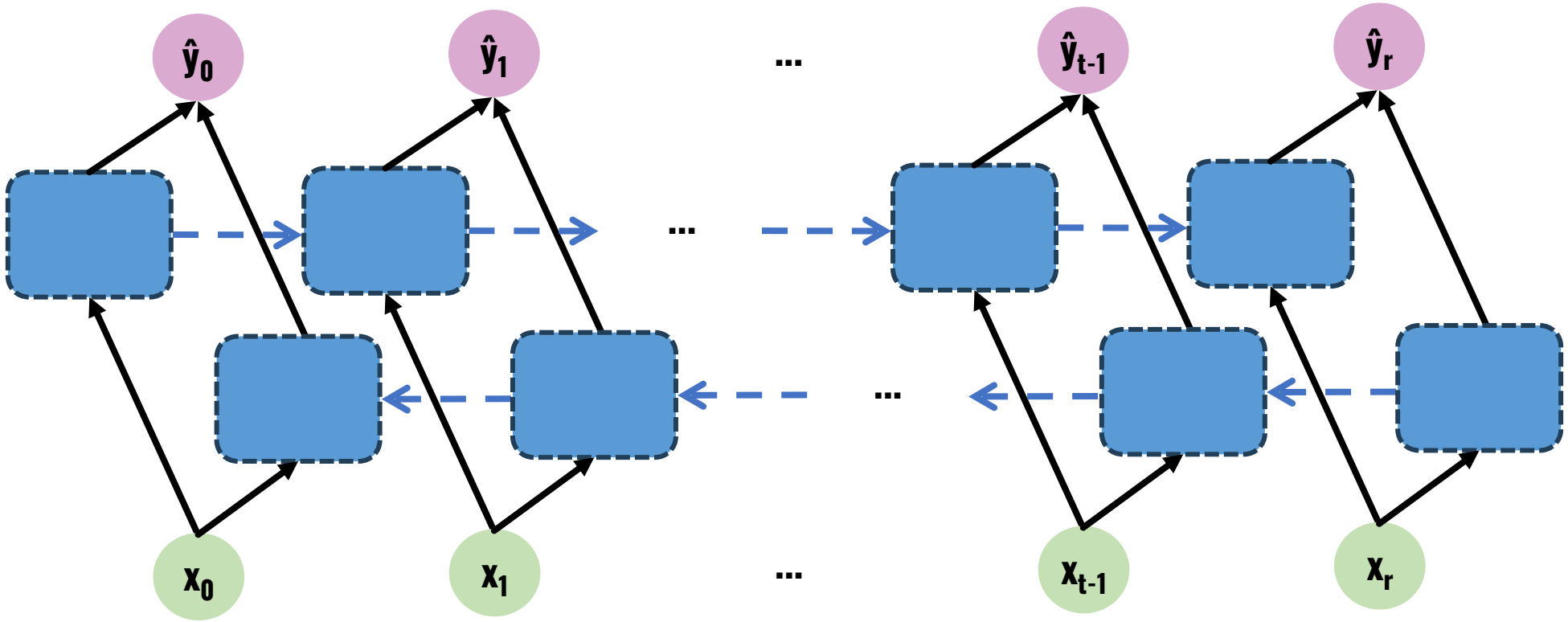


Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.1259

Bidirectional RNN

Combines two independent recurrent models

- One processes the input in forward time order; the other in reverse
- Their outputs are merged at each time step by concatenation or summation



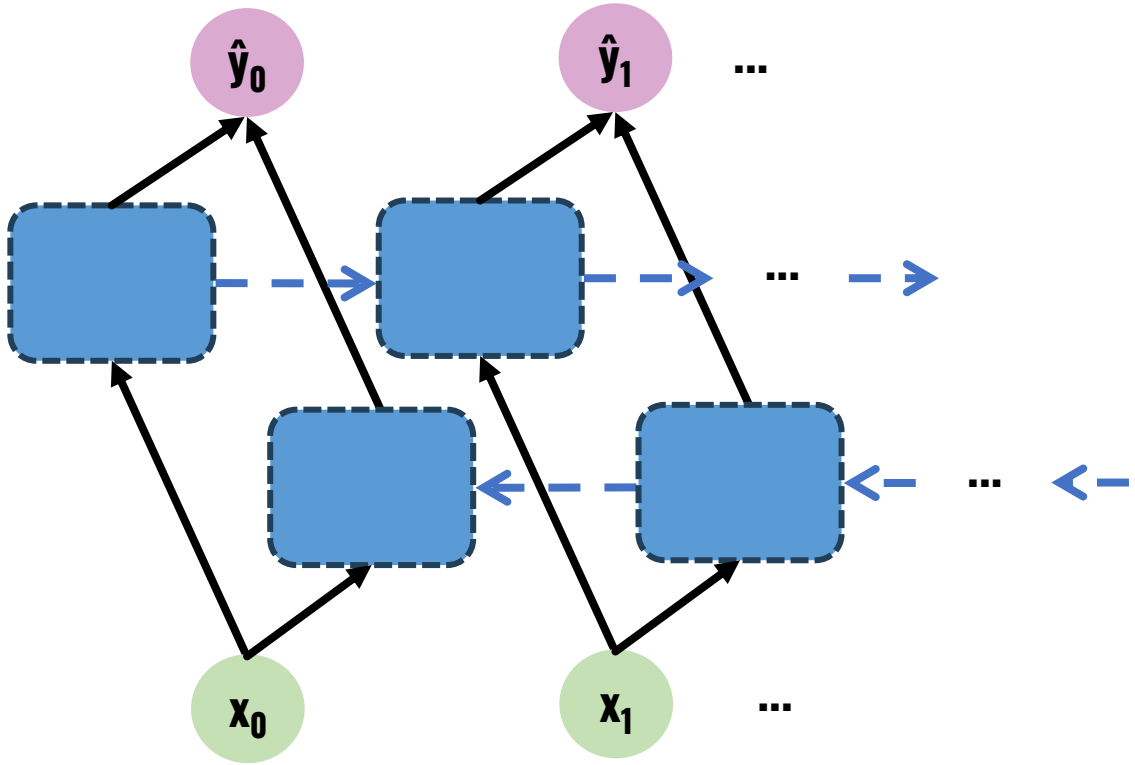
Bidirectional RNN (BRNN)

Hidden state:

$$\overrightarrow{h_t} = \tanh(\overrightarrow{W_{hh}}\,\overrightarrow{h_{t-1}} + \overrightarrow{W_{xh}}\,\overrightarrow{x_t} + \overrightarrow{b_h})$$

$$\overleftarrow{h_t} = \tanh(\overleftarrow{W_{hh}}\,\overleftarrow{h_{t+1}} + \overleftarrow{W_{xh}}\,\overleftarrow{x_t} + \overleftarrow{b_h})$$

$$h_t = \tanh(\overrightarrow{W_{hy}}\,\overrightarrow{h_t} + \overleftarrow{W_{hy}}\,\overleftarrow{h_t} + b_y)$$



Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 7

Transformer Networks, Vision Transformers

1 Natural Language Understanding

2 Attention Mechanism

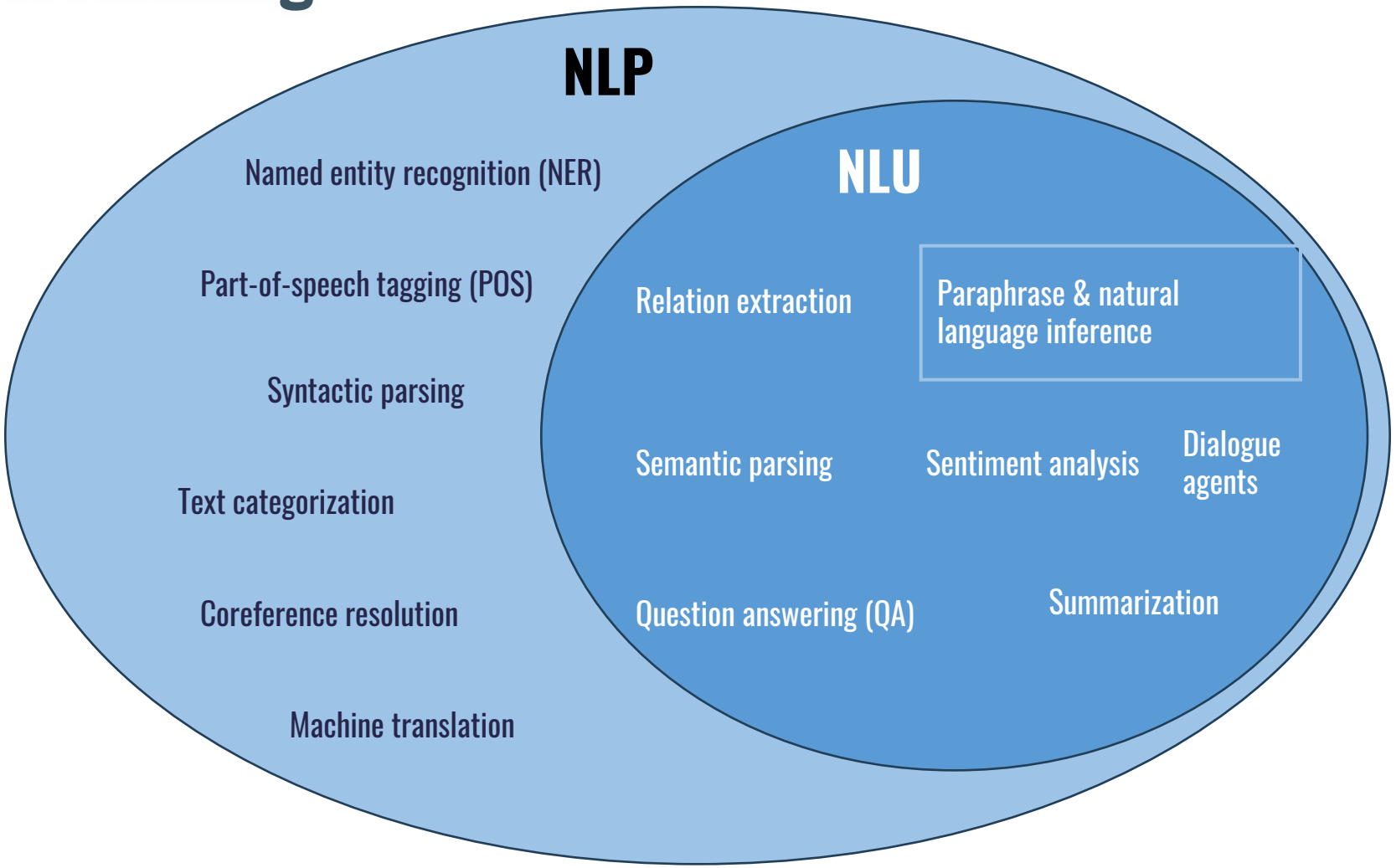
3 Transformer Architecture

4 Vision Transformers

4 Transformer Network

Natural Language Understanding

- Neural Machine Translation
- Textual entailment
- Question answering
- Semantic similarity assessment
- Text classification
- Etc...



Machine Translation

- Most of the proposed neural machine translation models belong to the family of **encode-decoders**
- From probabilistic perspective:

$$argmax_y p(y \mid x)$$

- Which means finding a target **y** that maximizes the conditional probability of **y** given a source sentence **x**

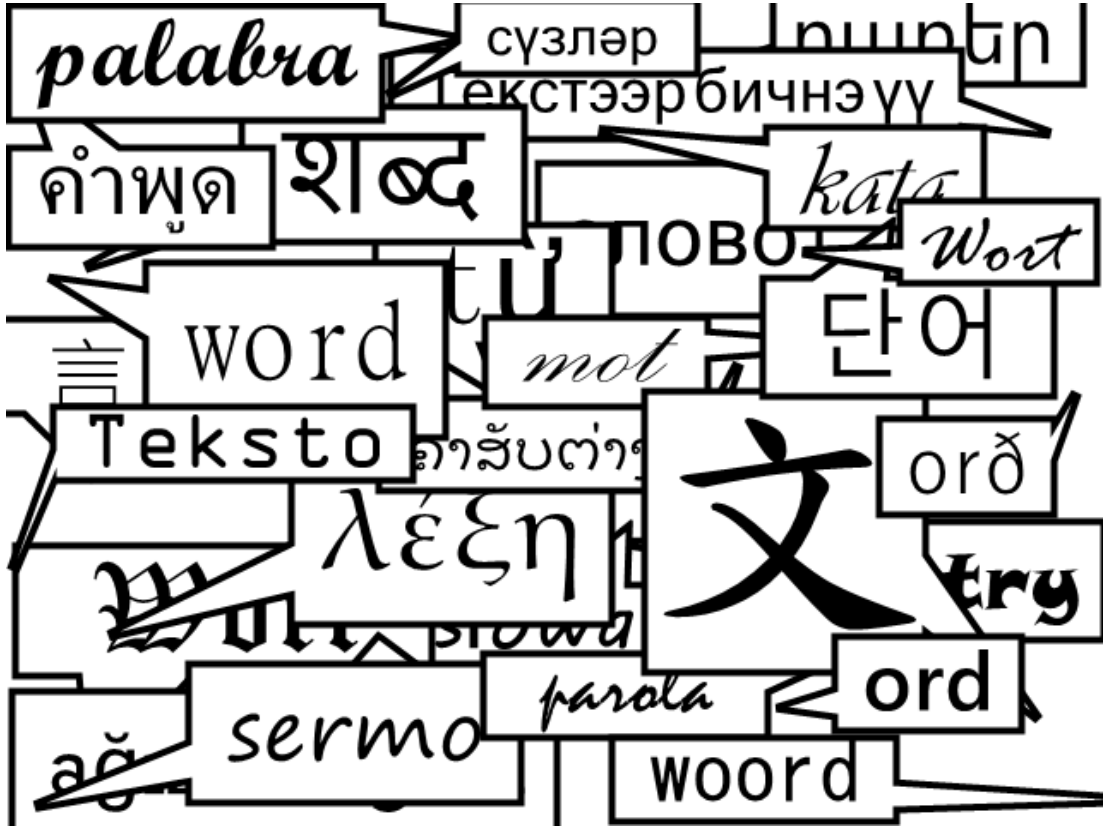


Image Source: Translate.png, Impartial just, CC0, via Wikimedia Commons

Machine Translation

- In the **encoder-decoder** framework an **encoder** reads and reduces the input sentence (a sequence of vectors $\mathbf{x} = (x_1, \dots, x_T)$) into a vector c
- The approach for RNNs:

$$h_t = f(x_t, h_{t-1})$$
$$c = q(\{h_1, \dots, h_{T_x}\})$$

- Where q is a nonlinear function, h_t is the hidden state at time t
- If we choose f as an LSTM then $q(\{h_1, \dots, h_{T_x}\}) = h_{T_x}$
- The decoder then predicts the next word y_t given the context vector c and all the previously predicted words $\{y_1, \dots, y_{t-1}\}$

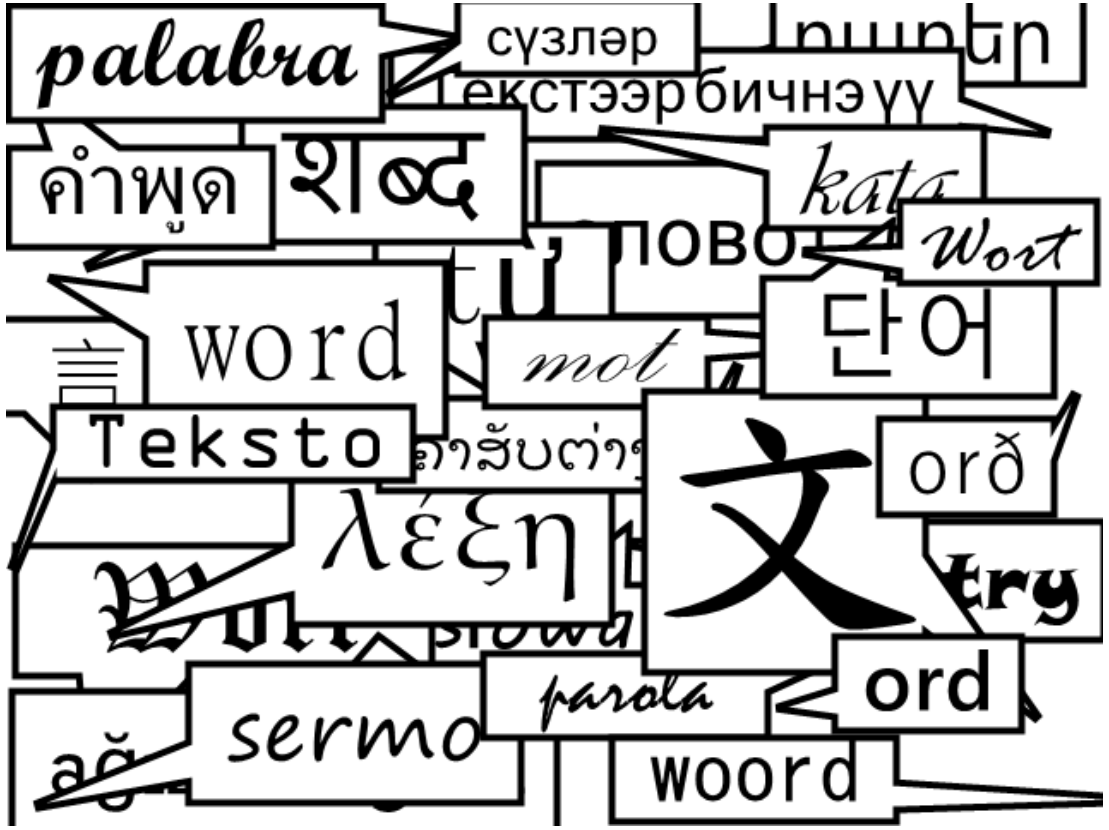


Image Source: Translate.png, Impartial just, CCO, via Wikimedia Commons

Machine Translation

- The **decoder** then predicts the next word y_t given the context vector c and all the previously predicted words $\{y_1, \dots, y_{t-1}\}$
- The predicted translation $\mathbf{y}$ can be expressed as the joint probability of:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \dots, y_{t-1}\}, c)$$

- where $\mathbf{y} = \{y_1, \dots, y_{T_y}\}$
- Whit and RNN this is modelled as:

$$p(y_t \mid \{y_1, \dots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

- Where g nonlinear multi layered function and s_t is the hidden state

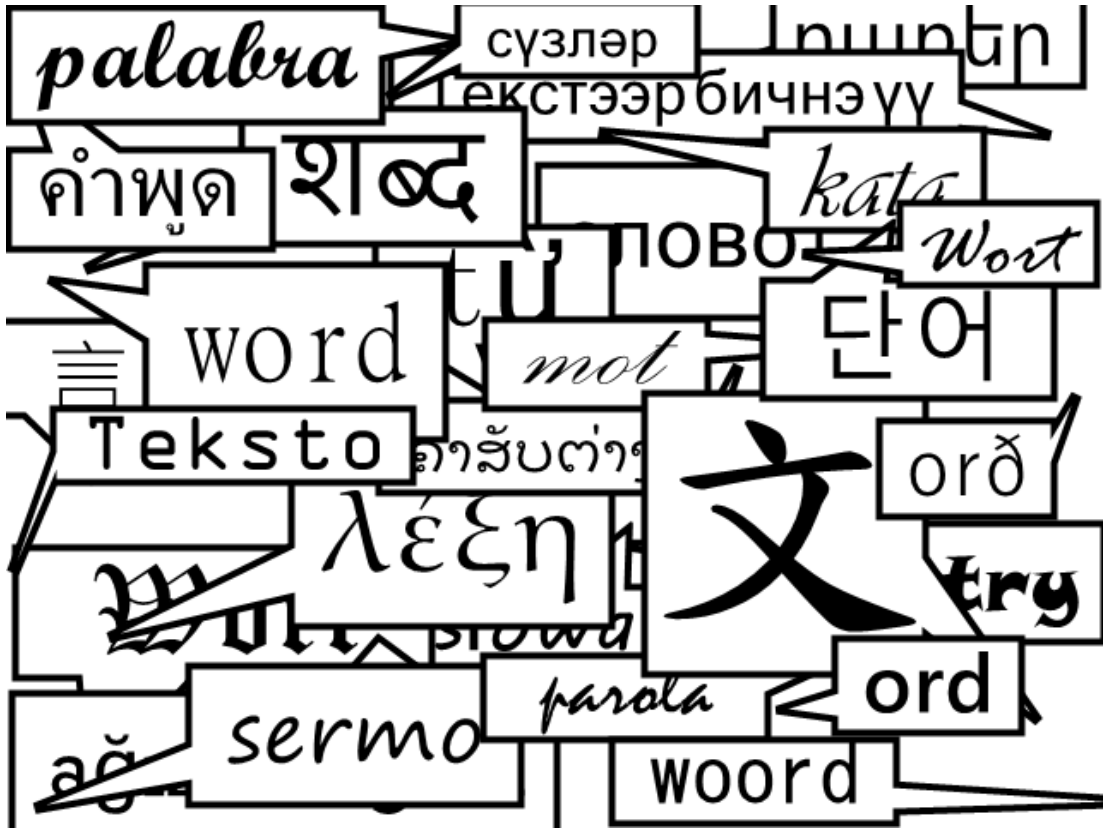
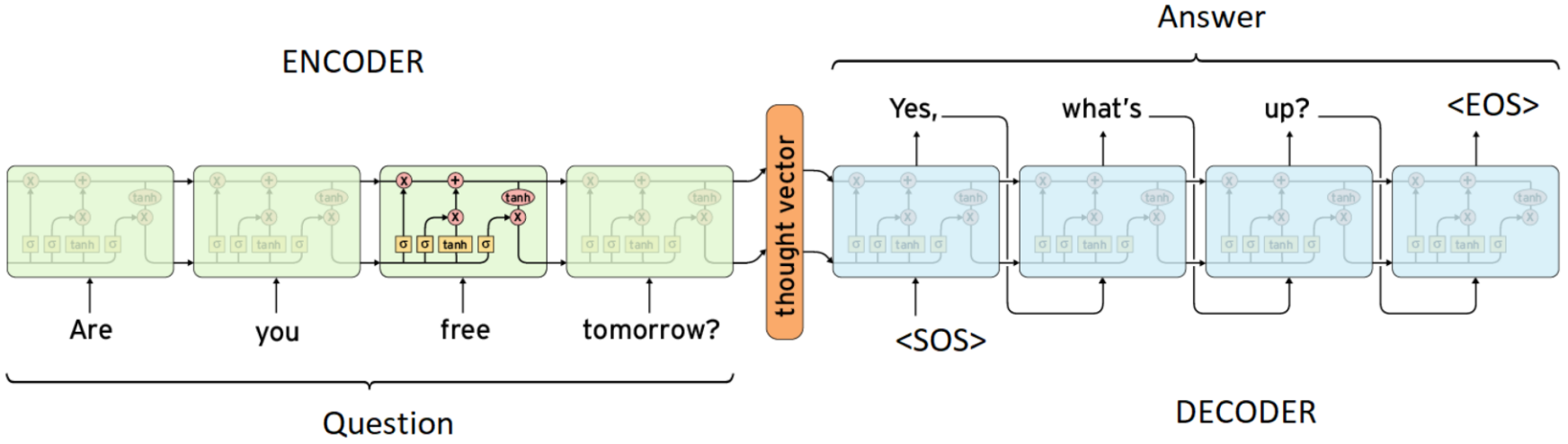


Image Source: Translate.png, Impartial just, CC0, via Wikimedia Commons

Encoder – Decoder (Seq2Seq)

Question : Answer
(sequence) : (sequence)



Machine Translation
English : French
(sequence) : (sequence)

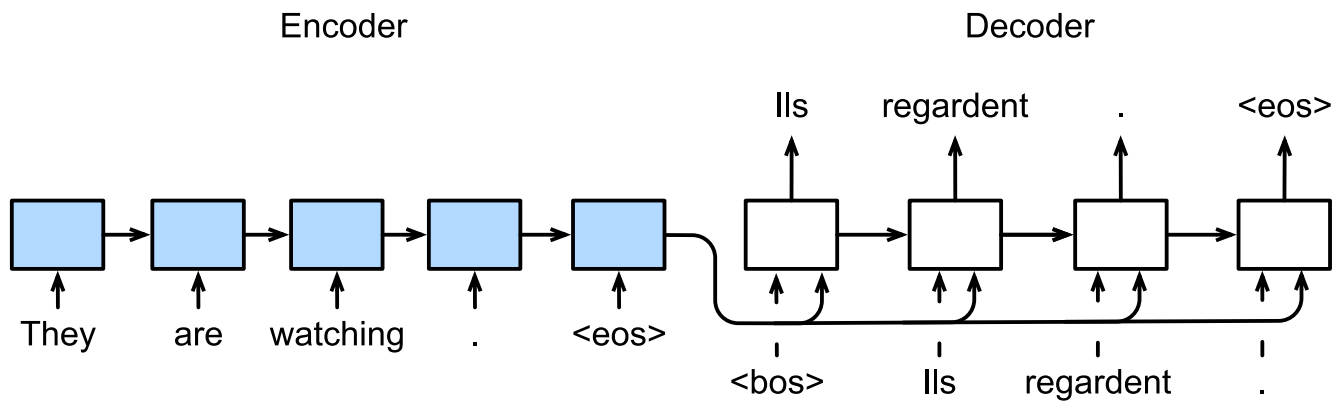
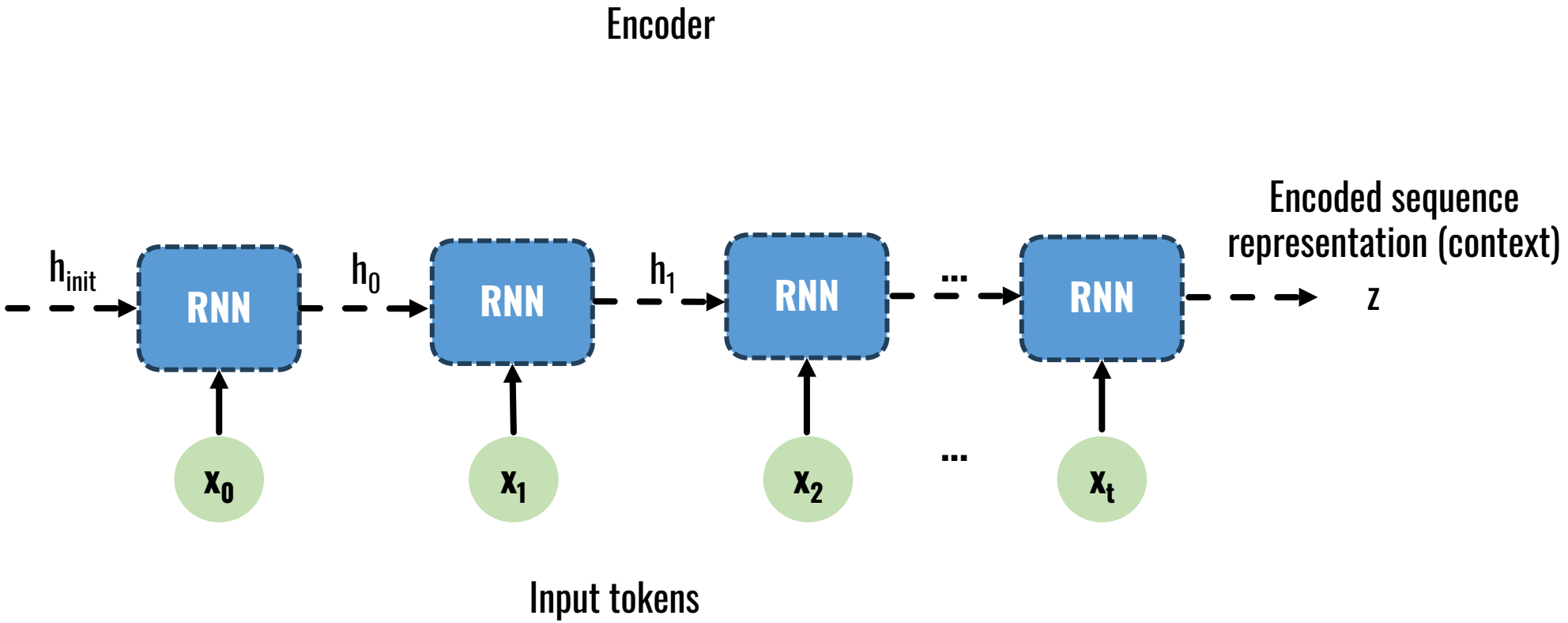


Image Source: The entire Seq2Seq chatbot architecture — Hosameldeen et al. (2023), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://www.preprints.org/manuscript/202303.0070
Image Source: Seq2seq-training.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Encoder – Decoder (Seq2Seq)

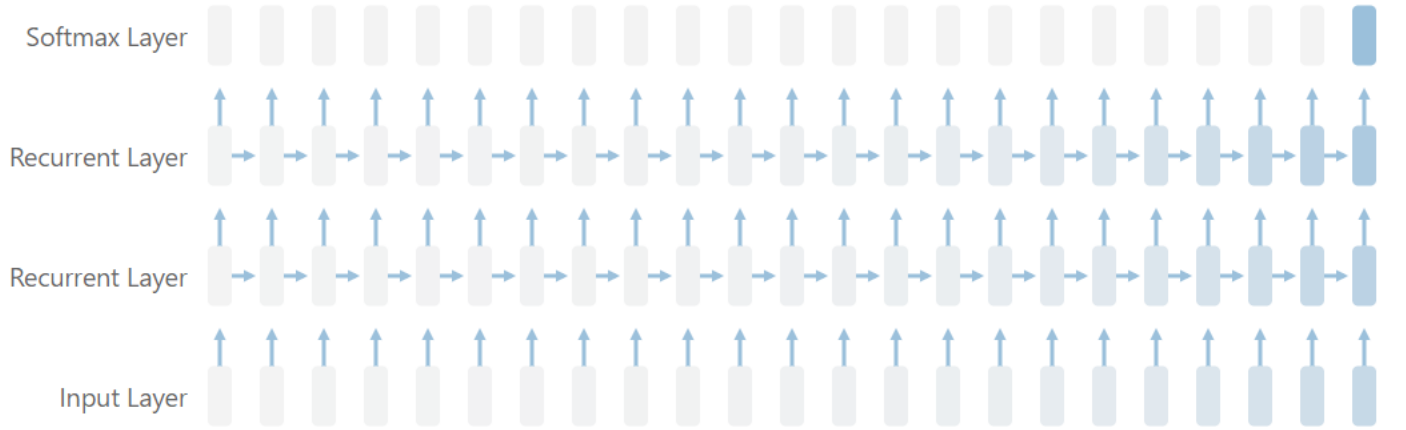
- The encoder and decoder are nothing more than stacked RNN layers, such as LSTM's. The encoder processes the input and produces one compact representation, called z, from all the input timesteps. It can be regarded as a compressed format of the input.



Encoder – Decoder (Seq2Seq)

Problems with this approach

- The intermediate representation **z** cannot encode information from all the input timesteps. This is commonly known as the **bottleneck problem**. The vector z needs to capture all the information about the source sentence.
- In practice, how far we can see in the past (the so-called reference window) is finite. RNN's tend to **forget information** from timesteps that are far behind.



Vanishing Gradient: where the contribution from the earlier steps becomes insignificant in the gradient for the vanilla RNN unit.

Image Source: Vanishing Gradient — Andreas Madsen, Visualizing memorization in RNNs, Distill (2019), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://distill.pub/2019/memorization-in-rnns/

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 7

Transformer Networks, Vision Transformers

1 Natural Language Understanding

2 Attention Mechanism

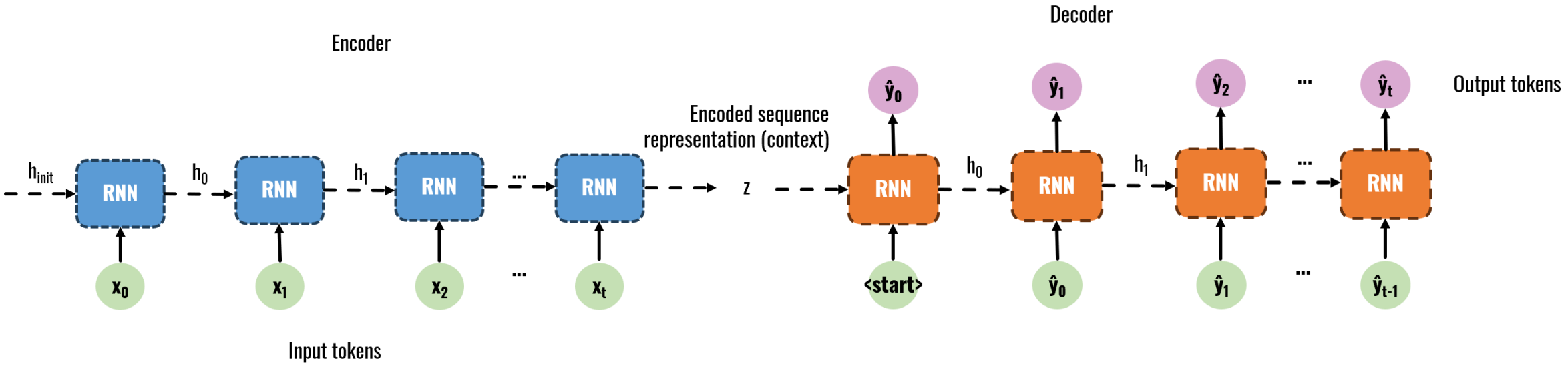
3 Transformer Architecture

4 Vision Transformers

4 Transformer Network

Introduction to Attention Mechanism

- All the information from the encoder is represented in the z vector (context)
- However, as seen previously, information from earlier timestamps is not preserved
- Can we create a better context vector?



Introduction to Attention Mechanism

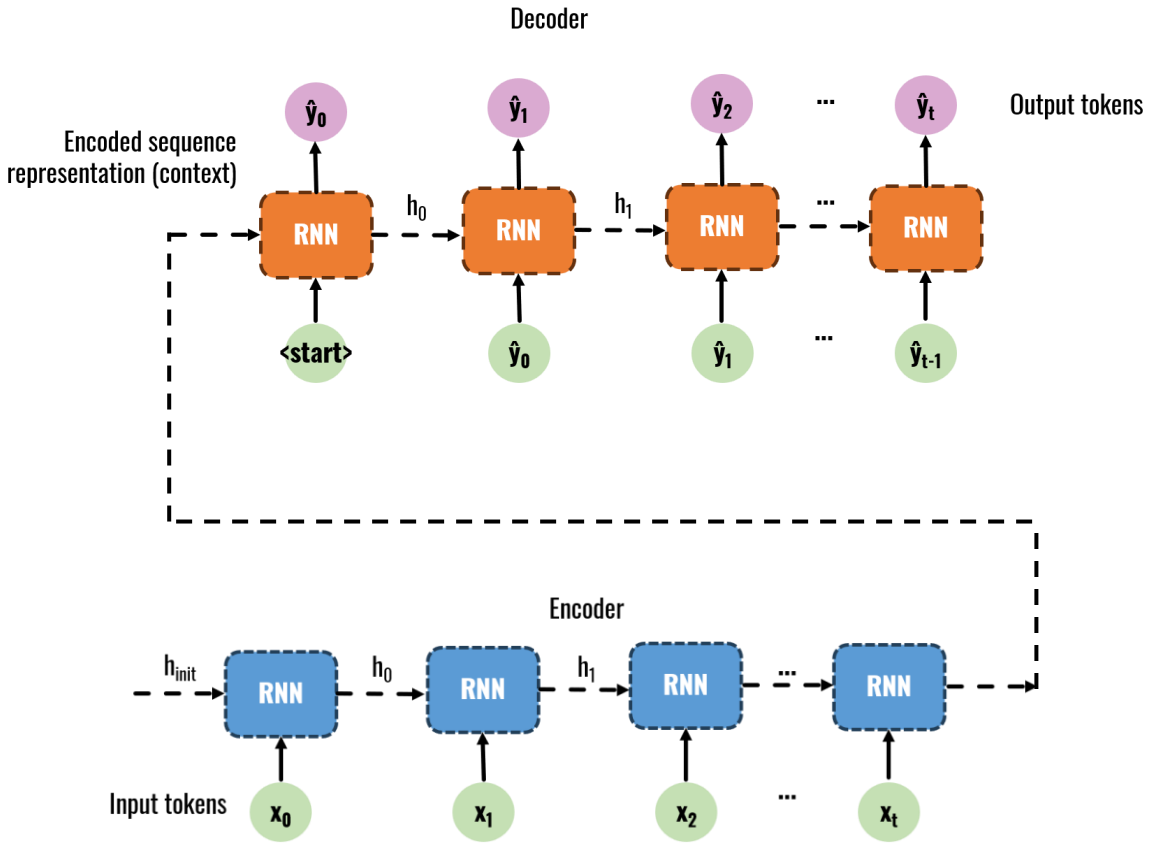
- In the new approach, we define each conditional probability as:

$$p(y_i \mid \{y_1, \ldots, y_{i-1}\}, c) = g(y_{i-1}, s_i, c_i)$$

- Where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

- Note that here we produce a distinct context vector c_i for each target word y_i



Introduction to Attention Mechanism (review)

- The core idea is that the context vector c_i should have access to all parts of the input sequence instead of just the last one.
- In other words, we need to form a **direct connection** with each timestamp. We can look at all the different words at the same time and learn to "pay attention" to the correct ones depending on the task at hand.
- In the encoder-decoder:
  - Given the hidden states of the encoder at each time step $h = \{h_1, \dots, h_{T_x}\}$
  - Given the previous state in the decoder y_{i-1}
  - We define an attention network that gives the attention scores for the current state of the decoder

$$e_{ij} = \text{attention}_{\text{net}}(y_{i-1}, \mathbf{h}) \in \mathbb{R}^n$$

  - We convert this scores into probabilities

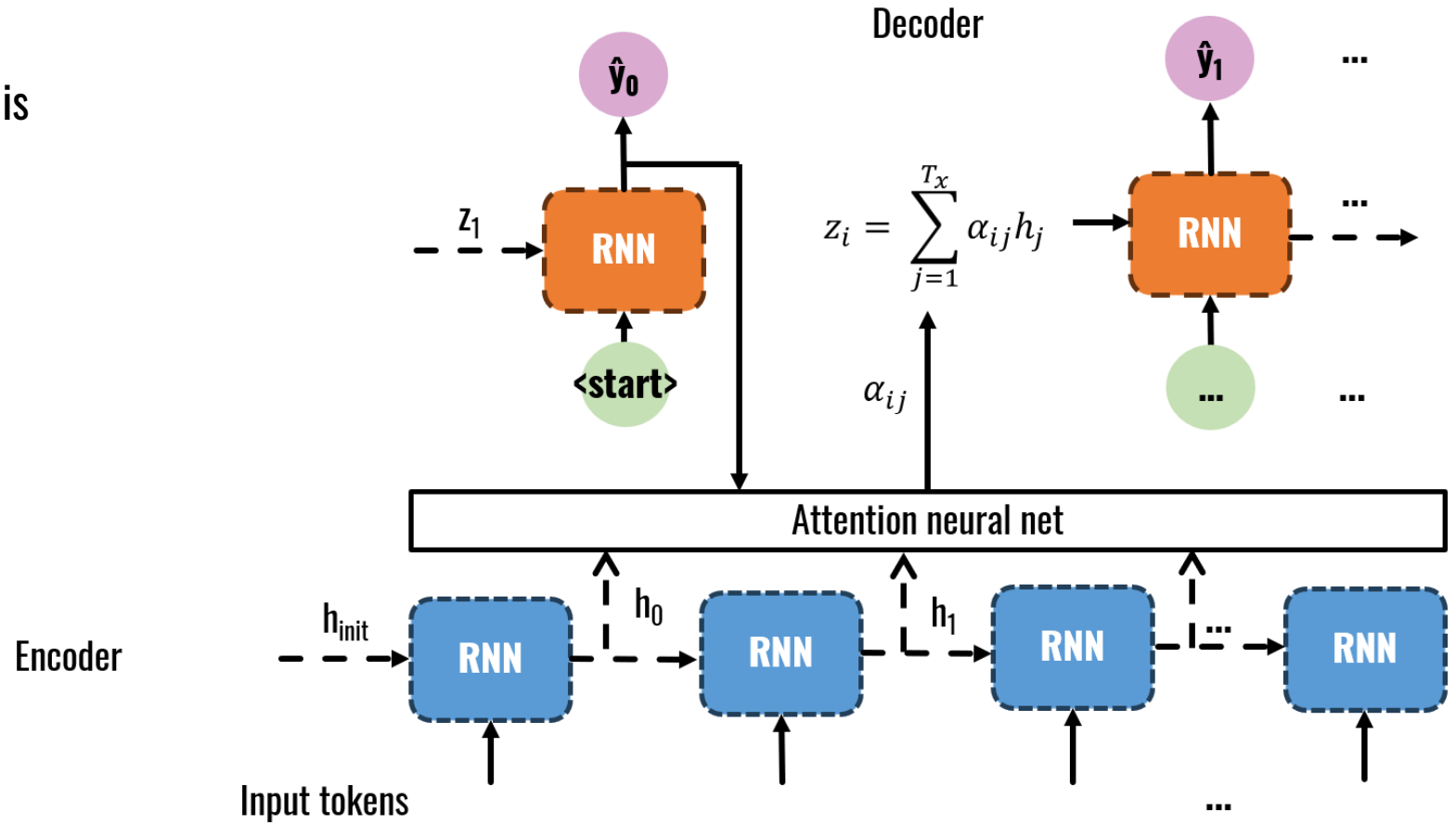
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

  - Finally, we get our new context vector z:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

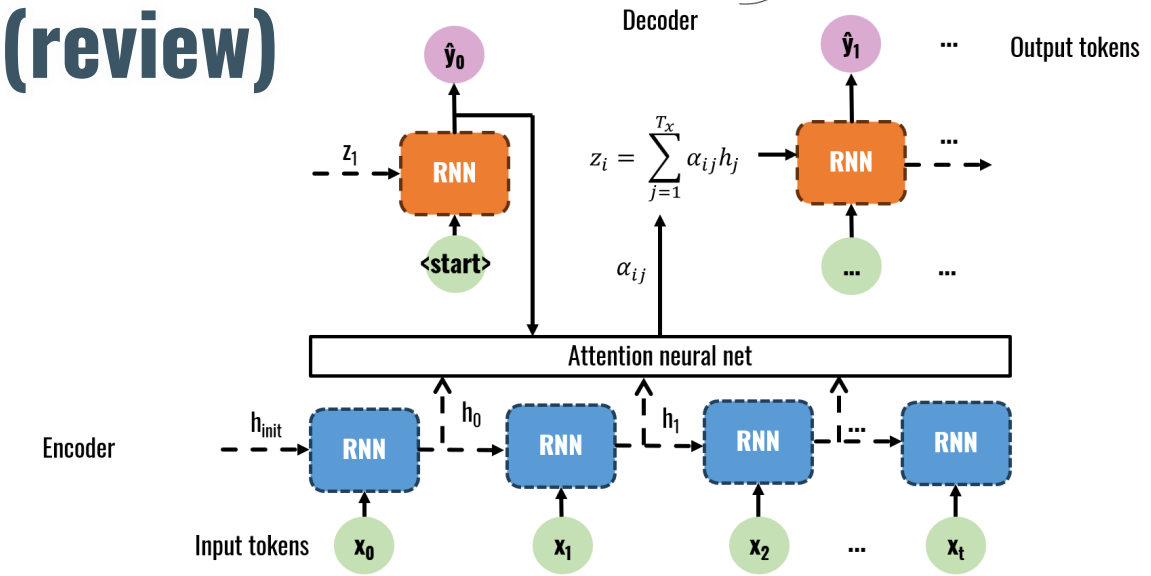
Introduction to Attention Mechanism (review)

- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder



Introduction to Attention Mechanism (review)

- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder



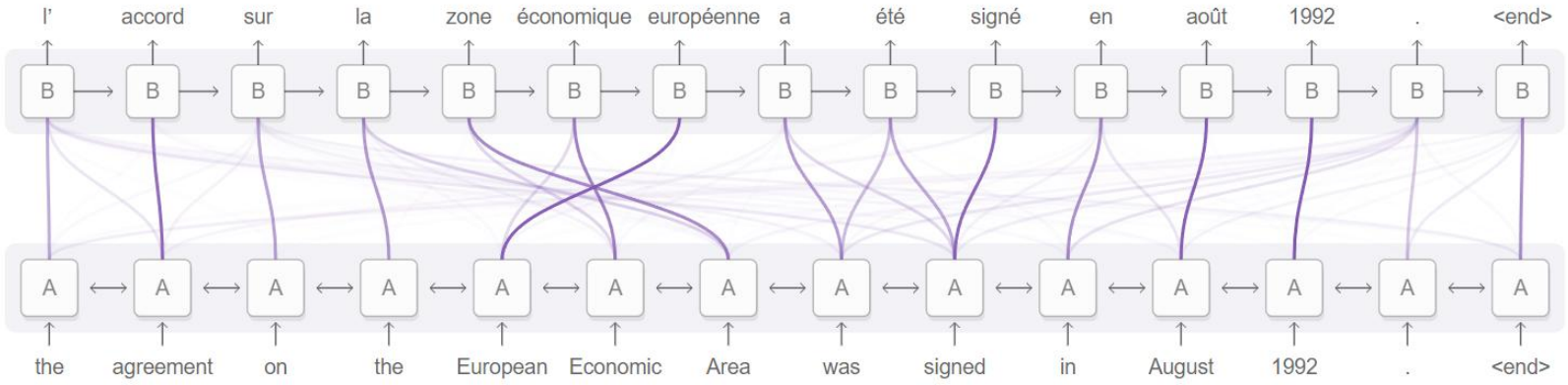
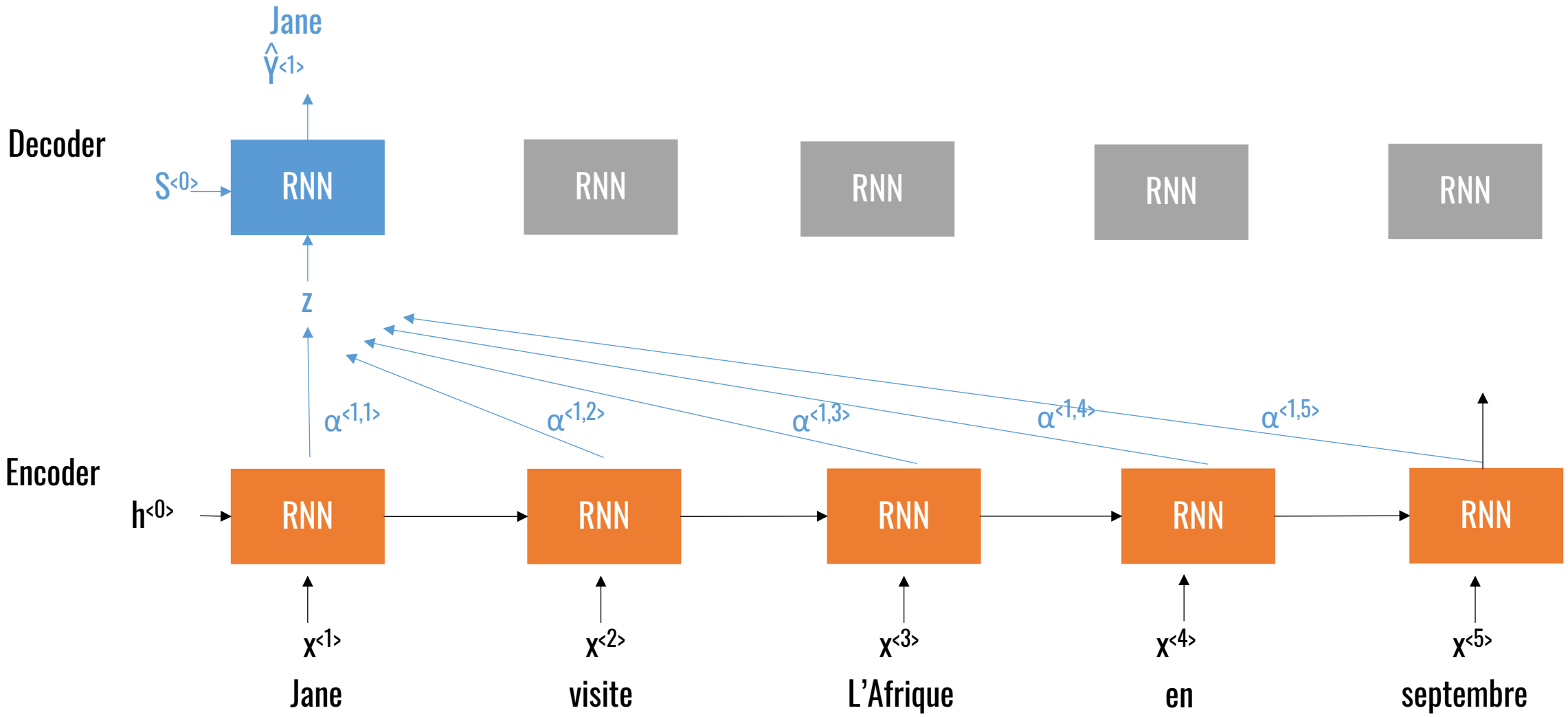


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

Image Source: Attention between RNNs — Chris Olah & Shan Carter, Attention and Augmented Recurrent Neural Networks, Distill (2016), derived from Fig. 3 of Bahdanau et al. (2014), CC BY 2.0 https://creativecommons.org/licenses/by/2.0/, https://distill.pub/2016/augmented-rnns

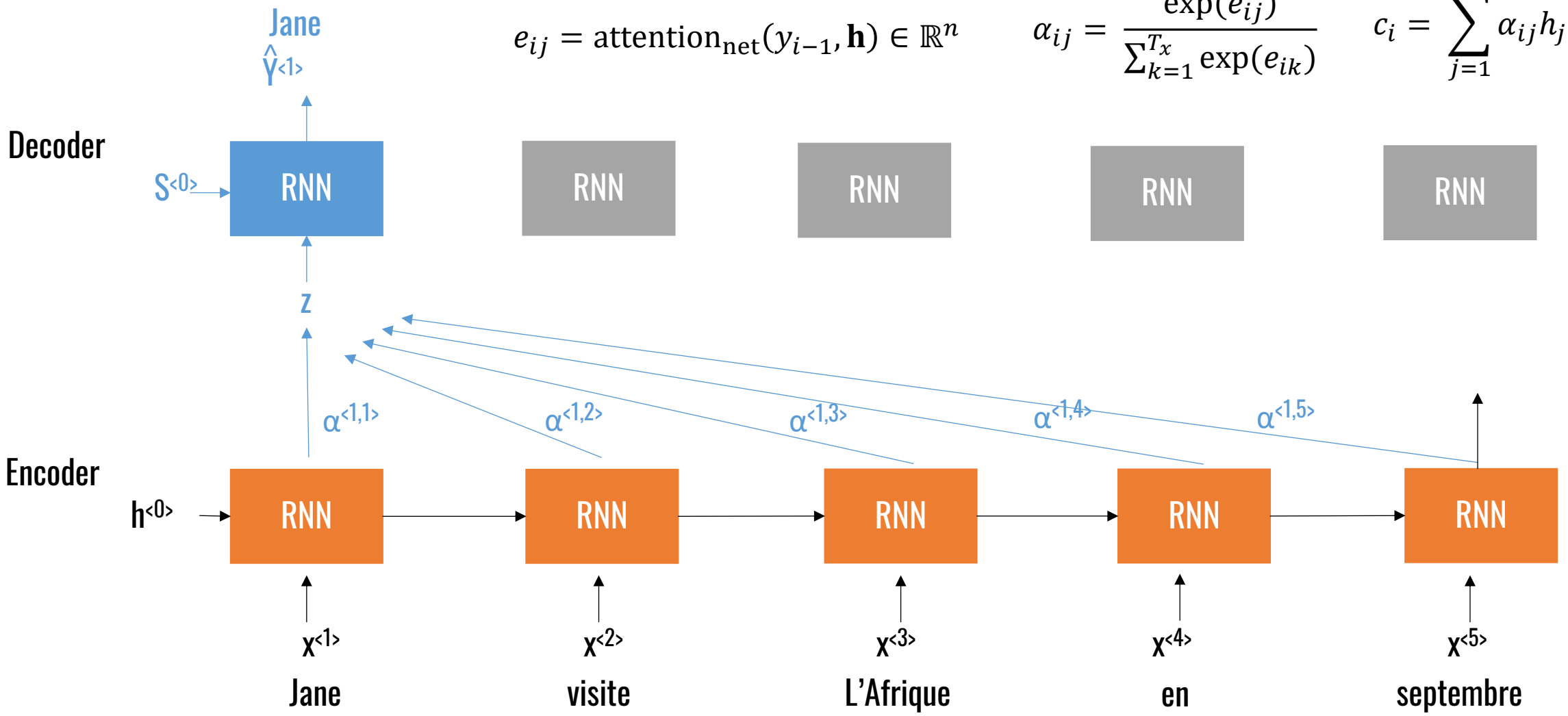
Attention Mechanism (review)



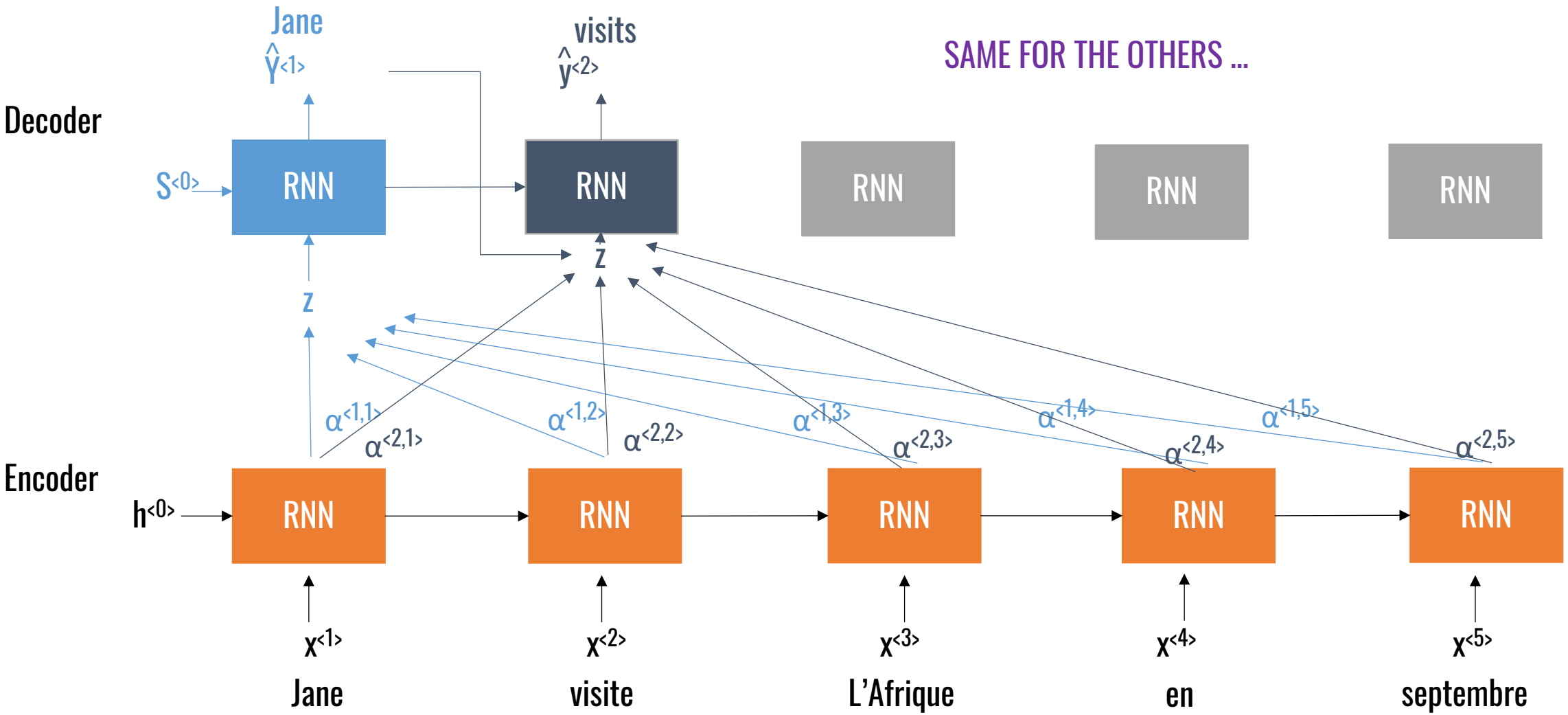
α_{ij} = amount of "attention" y_i should pay to h_*

Attention Mechanism (review)

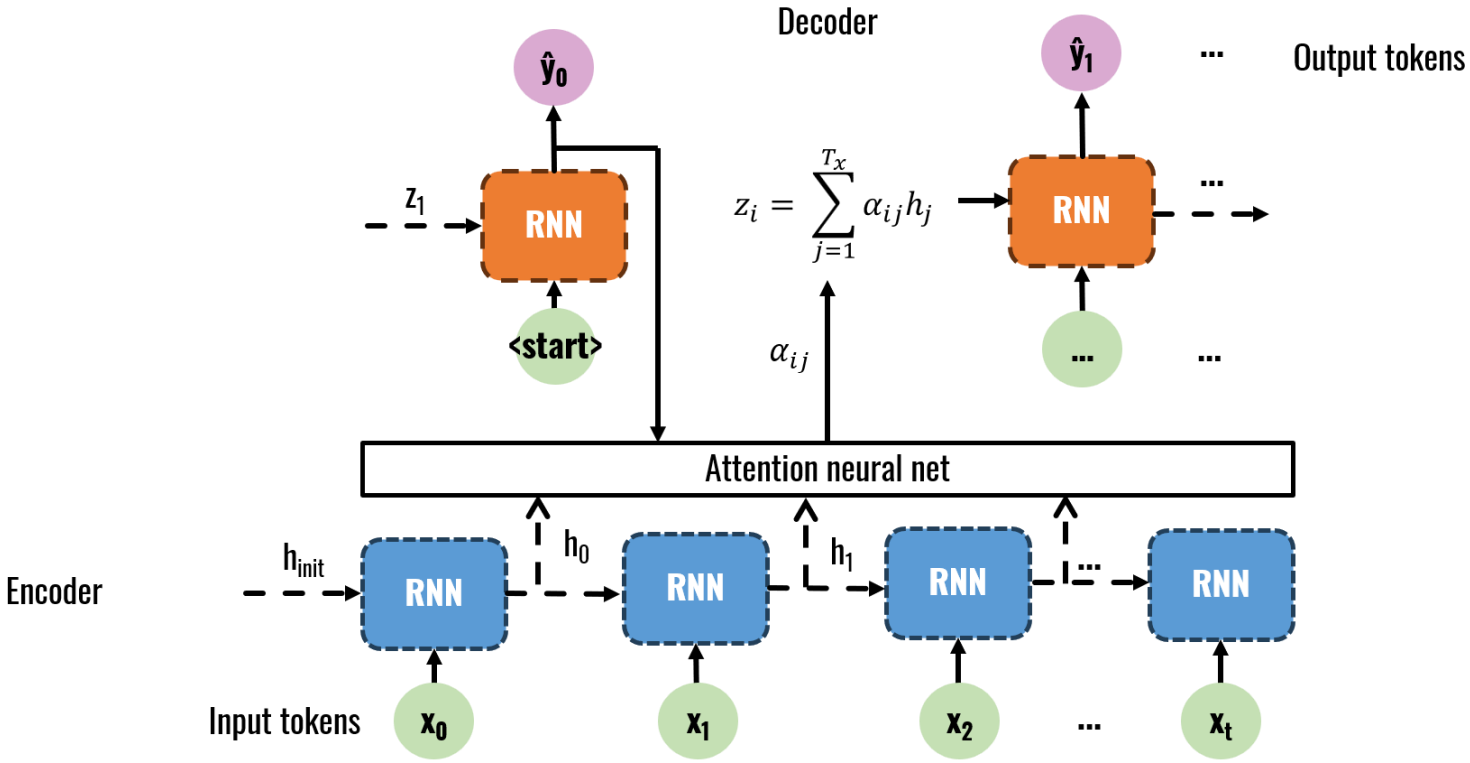
$$e_{ij} = \text{attention}_{\text{net}}(y_{i-1}, \mathbf{h}) \in \mathbb{R}^n \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

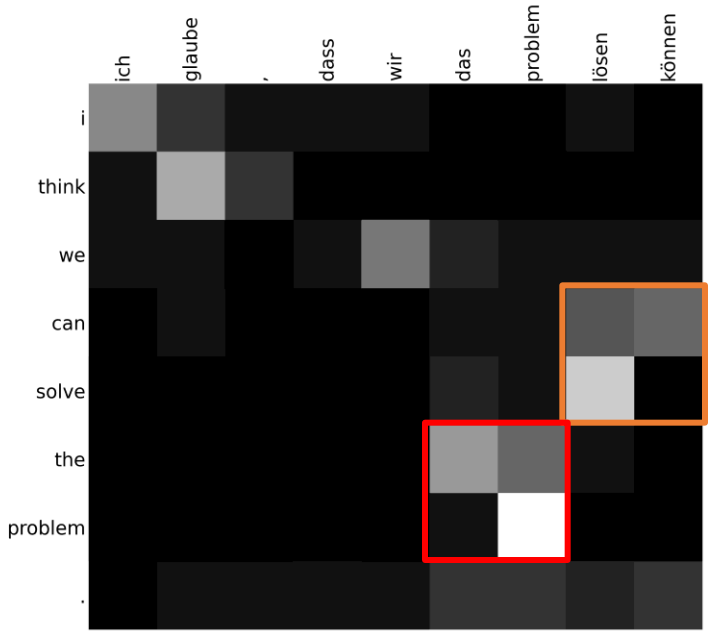


Attention Mechanism (review)

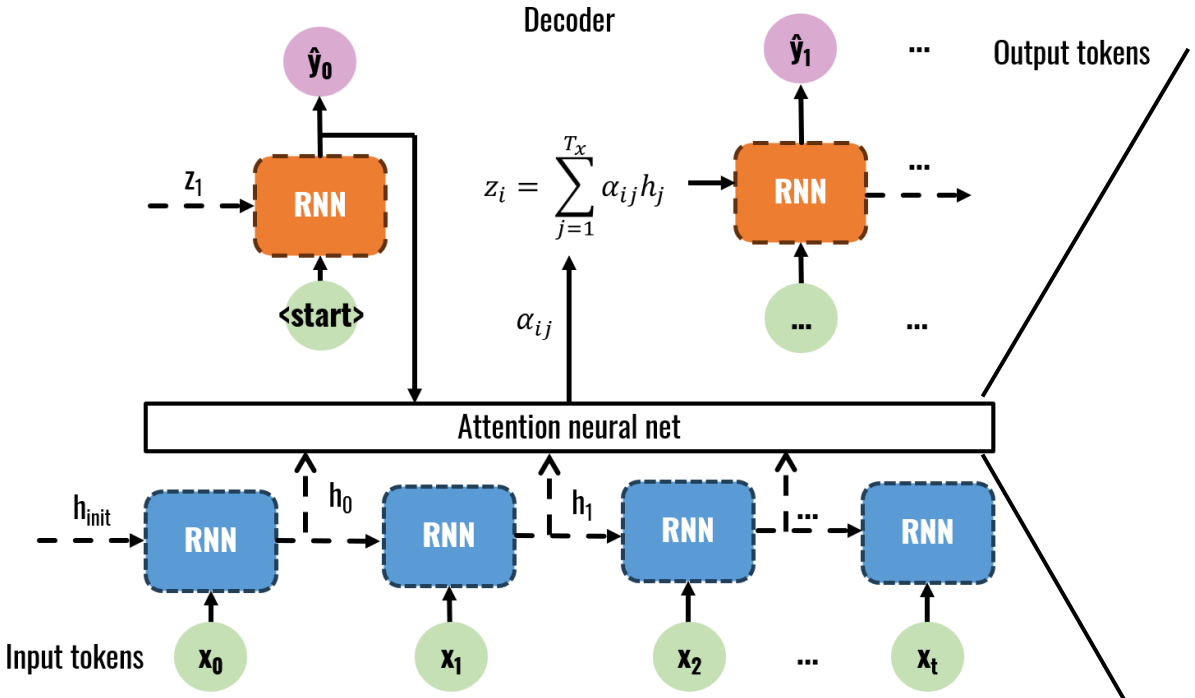


How does Attention work?





Attention Mechanism



Additive(*)	$score(s_t, h_i) = v_a^{\intercal} \tanh(W_a[s_t; h_i]$	Bahdanau et al.
Location-Base	$a_{t,i} = \text{softmax}(W_a s_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong et al.
General	$score(s_t, h_i) = s_t^{\intercal} W_a h_i$ where W_a is a trainable weight matrix in the attention layer.	Luong et al.
Scaled Dot-Product(^)	$score(s_t, h_i) = \frac{s_t^{\intercal} h_i}{\sqrt{n}}$ Note: Very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani et al.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.0473
Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1508.04025
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 7

Transformer Networks, Vision Transformers

1 Natural Language Understanding

2 Attention Mechanism

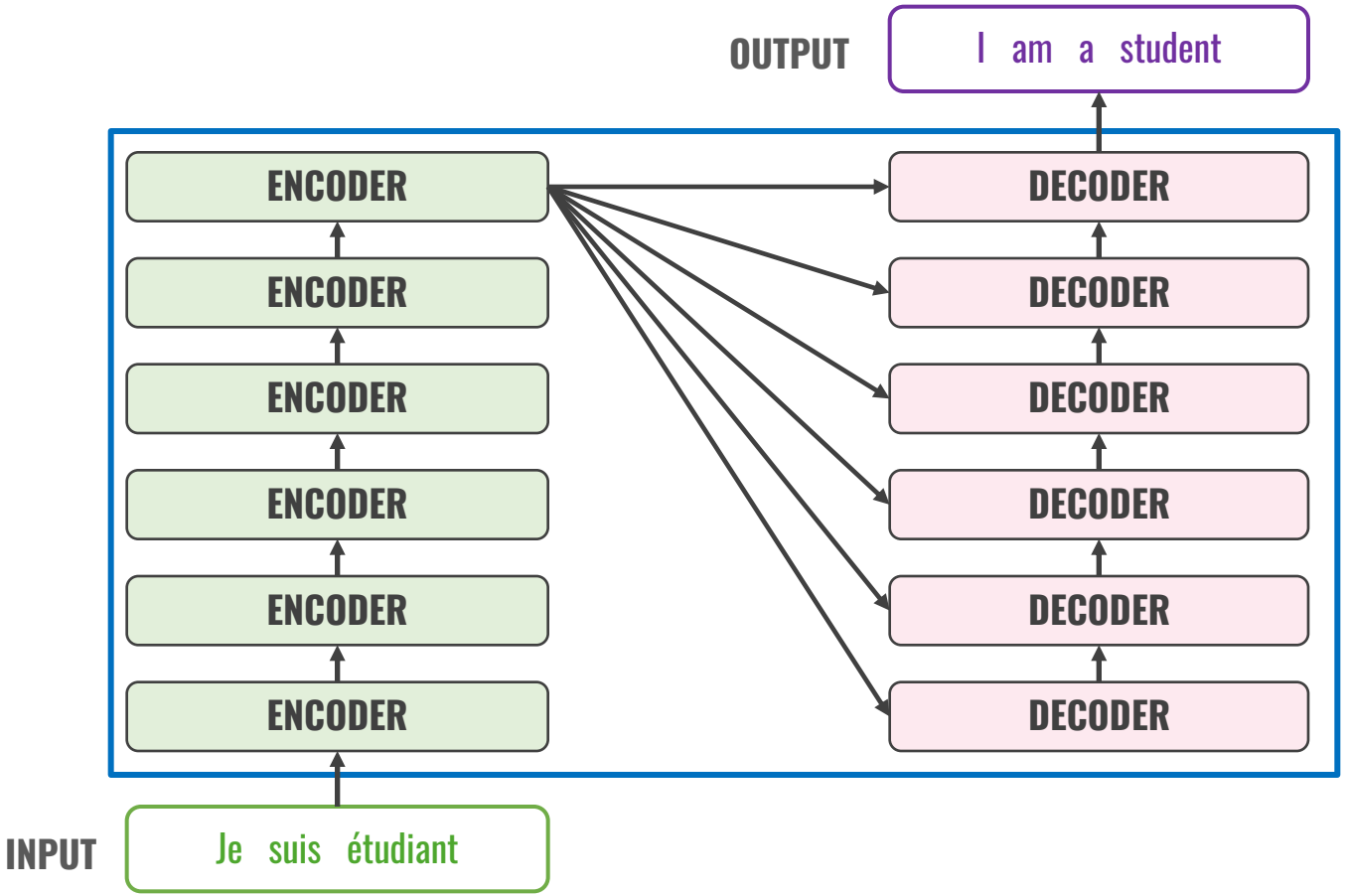
3 Transformer Architecture

4 Vision Transformers

4 Transformer Network

Transformers

- Multi-Head Attention
- Masked Multi-Head Attention
- Self-Attention



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762

Transformer Attention

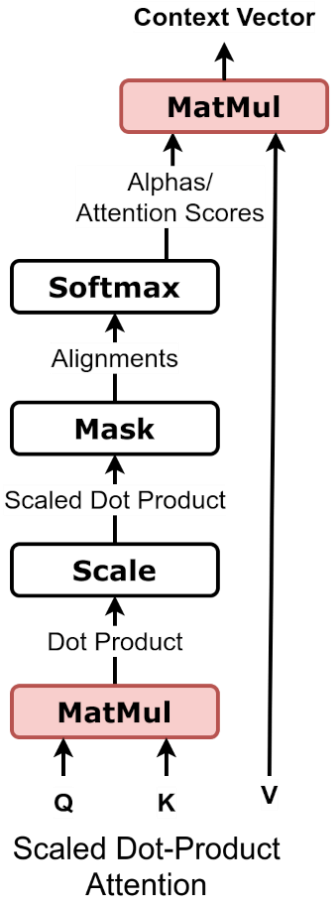
- **Feature-based attention: The Key, Value, and Query**
- Key-value-query concepts come from **information retrieval systems**.

Example of searching for a video on YouTube:

- When you search (**query**) for a particular video, the search engine will map your **query** against a set of **keys** (video title, description, etc.) associated with possible stored videos. Then the algorithm will present you the best-matched videos (**values**). This is the foundation of content/**feature-based lookup**.

- Bringing this idea closer to the transformer's attention we have something like this:

Transformer Attention



$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

$$Q, K, V \in R^{n_k \times d_k}$$

$$QK^T : (n_q \times d_k)(d_k \times n_k) \rightarrow n_q \times n_k$$

$$softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V : (n_q \times d_k)(d_k \times n_k)(n_k \times d_v) \rightarrow (n_q, d_v)$$

For self-attention, $n_q = n_k = n$, so:

$$Z \in R^{n \times d_v}$$

Image Source: Transformer, attention block diagram.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Self-Attention

Self-attention: the key component of the Transformer architecture

- We can also define the attention of the same sequence, called self-attention. Instead of looking for an input-output sequence association/alignment, **we are now looking for scores between the elements of the sequence**

	the	dog	chase	d	its	tail	</s>
the	0.01	0.48	0.18	0.01	0.01	0.08	0.23
dog	0.22	0.29	0.32	0.02	0.04	0.03	0.07
chase	0.02	0.44	0.46	0.02	0.01	0.02	0.03
d	0.00	0.17	0.75	0.01	0.02	0.03	0.01
its	0.00	0.03	0.11	0.14	0.12	0.35	0.24
tail	0.00	0.02	0.06	0.02	0.12	0.37	0.41
</s>	0.01	0.01	0.01	0.02	0.02	0.12	0.81

Self-Attention

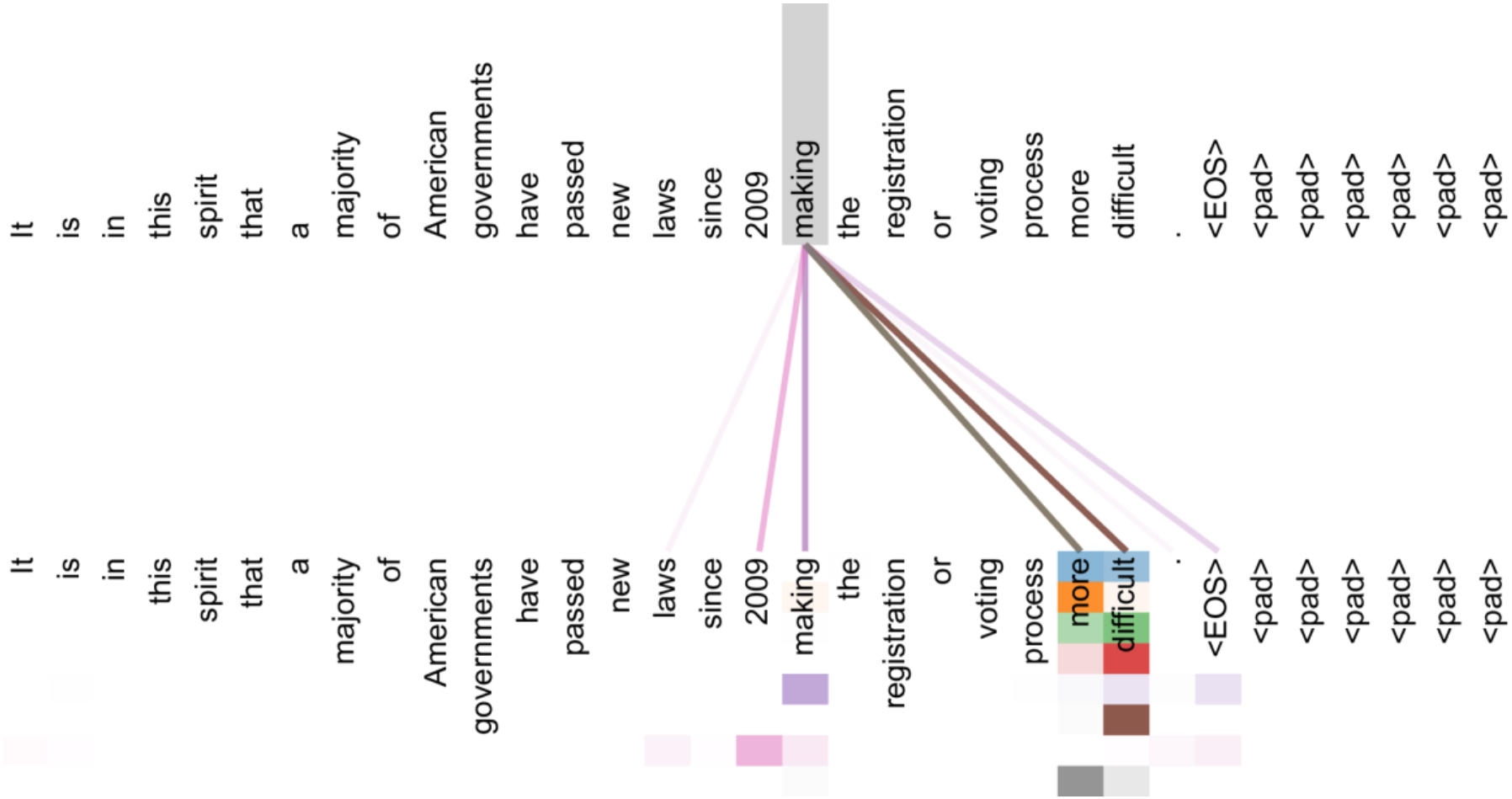
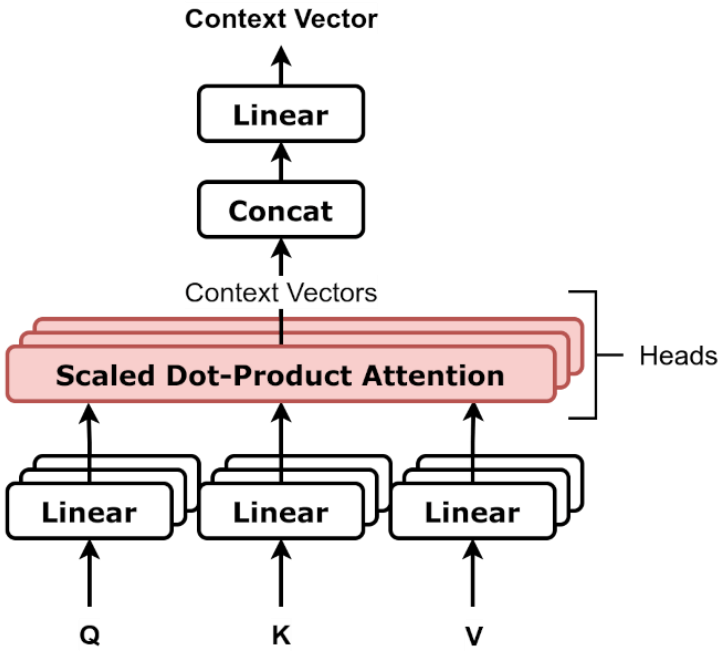
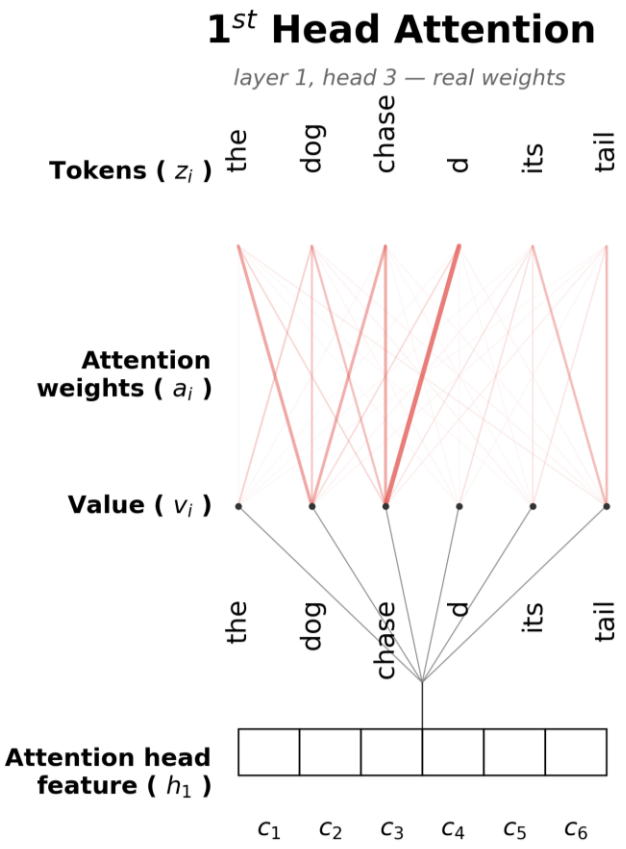


Image Source: Attention Is All You Need - long-distance attention, encoder self-attention in layer 5 of 6.png, Google, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Multi-Head Attention

- Repeating self-attention multiple times to better understand the input (multiple ways of deciding the relationship between the parts of the input)
  - Looks at the input from different perspectives





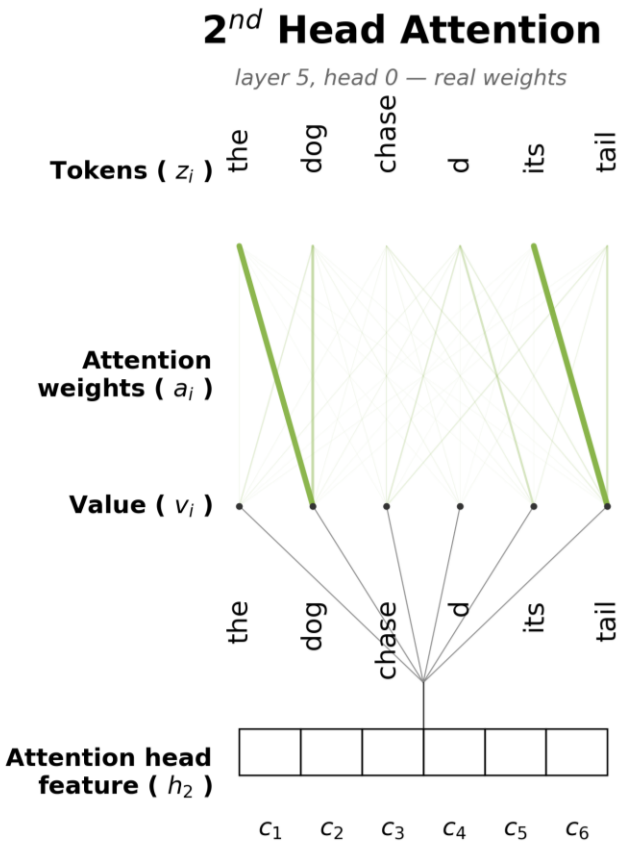


Image Source: Multiheaded attention, block diagram.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Visualizing Self-Attention

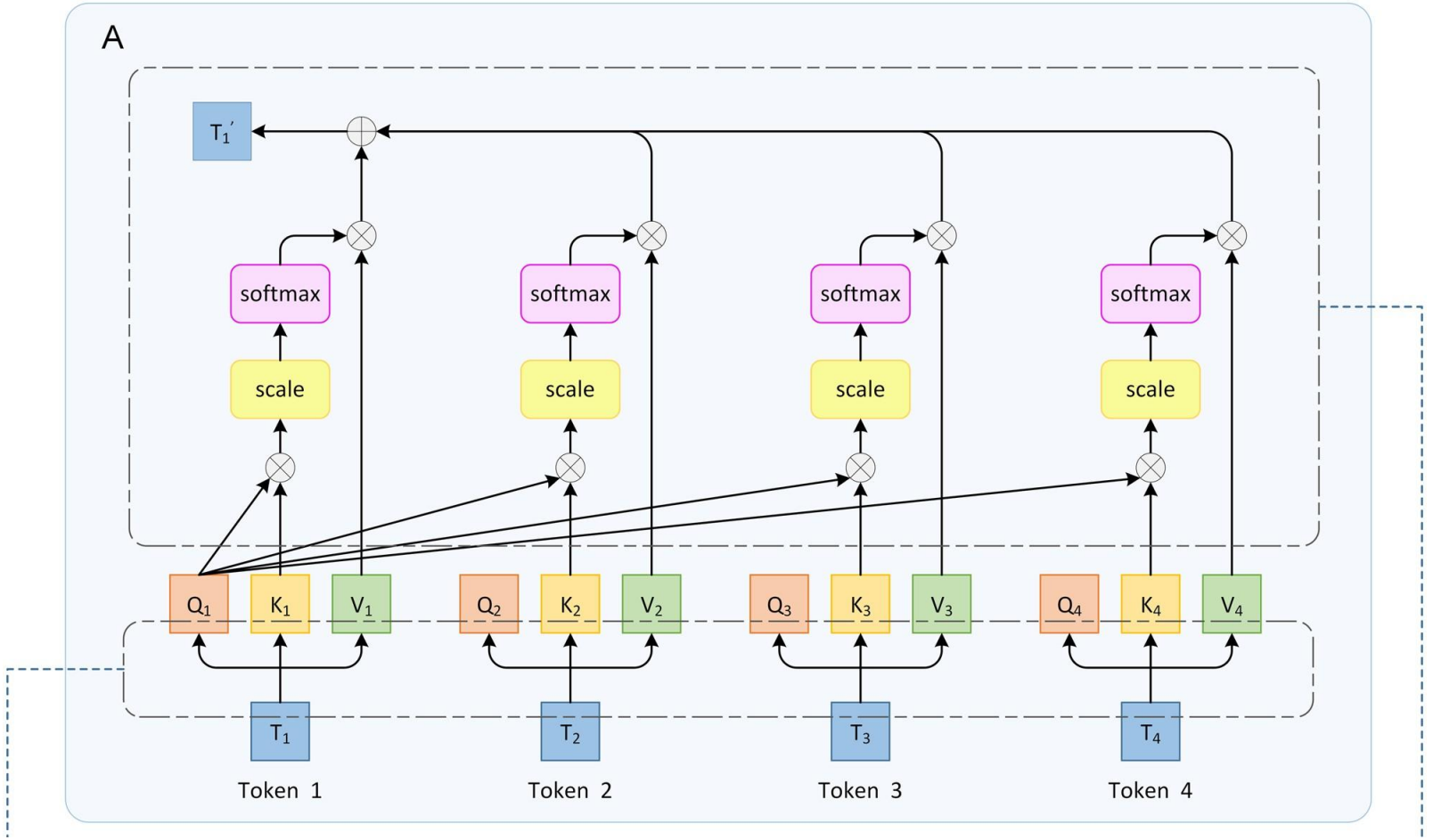
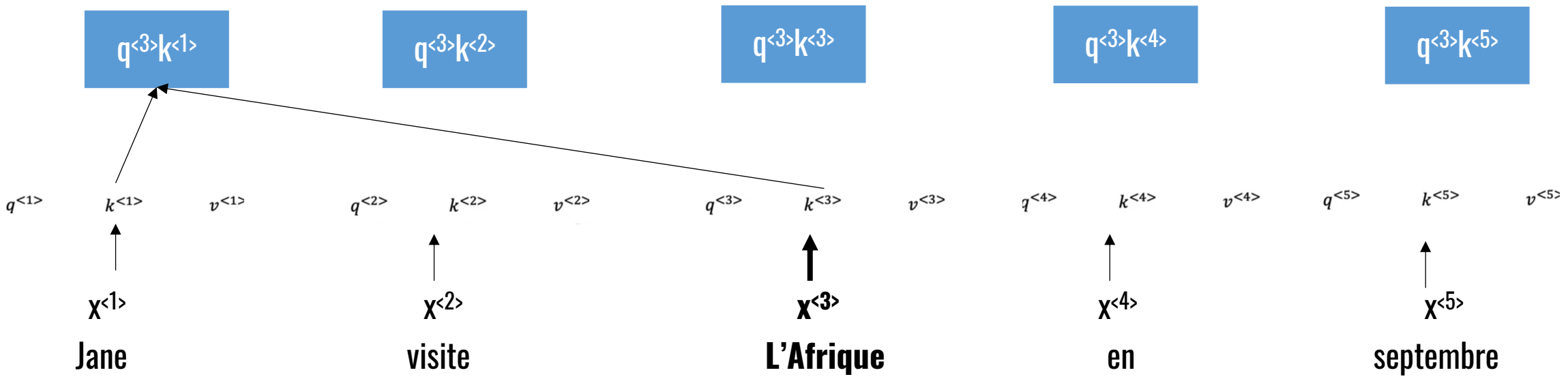


Image Source: [Process of a Single Attention Head in a Transformer Model.jpg](), Shuang Zhang, Rui Fan, Yuti Liu, Shuang Chen, Qiao Liu, Wanwen Zeng, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Self-Attention

$q^{<3>} = W^Q \cdot x^{<3>}$ $k^{<3>} = W^K \cdot x^{<3>}$ $v^{<3>} = W^V \cdot x^{<3>}$

Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$
$q^{<5>}$	$k^{<5>}$	$v^{<5>}$

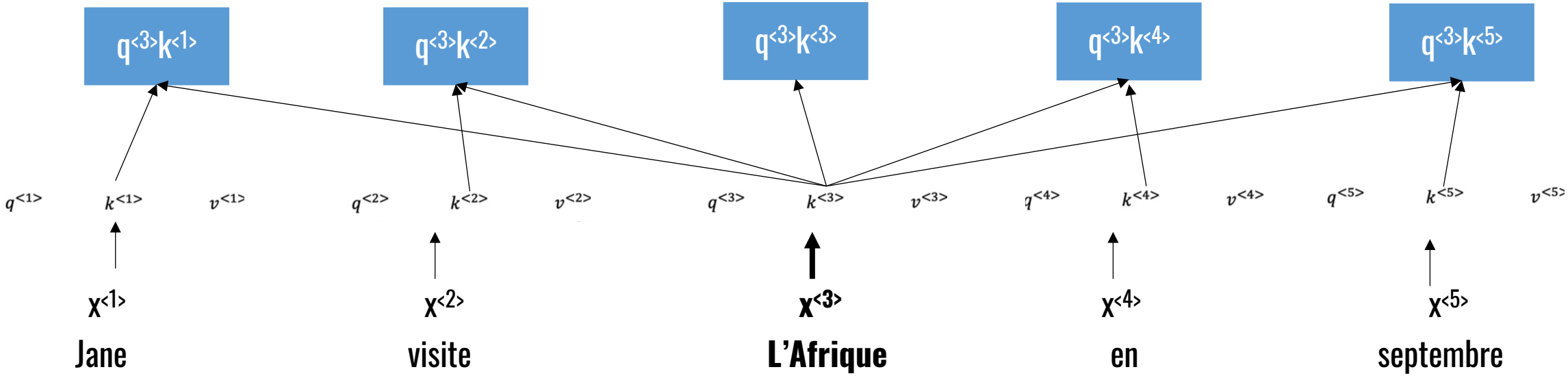


Self-Attention

$q^{<3>} = W^Q \cdot x^{<3>}$ $k^{<3>} = W^K \cdot x^{<3>}$ $v^{<3>} = W^V \cdot x^{<3>}$

Query (Q)		Key (K)	Value (V)
$q^{<1>}$	person	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	action	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$		$k^{<3>}$	$v^{<3>}$
$q^{<4>}$		$k^{<4>}$	$v^{<4>}$
$q^{<5>}$		$k^{<5>}$	$v^{<5>}$

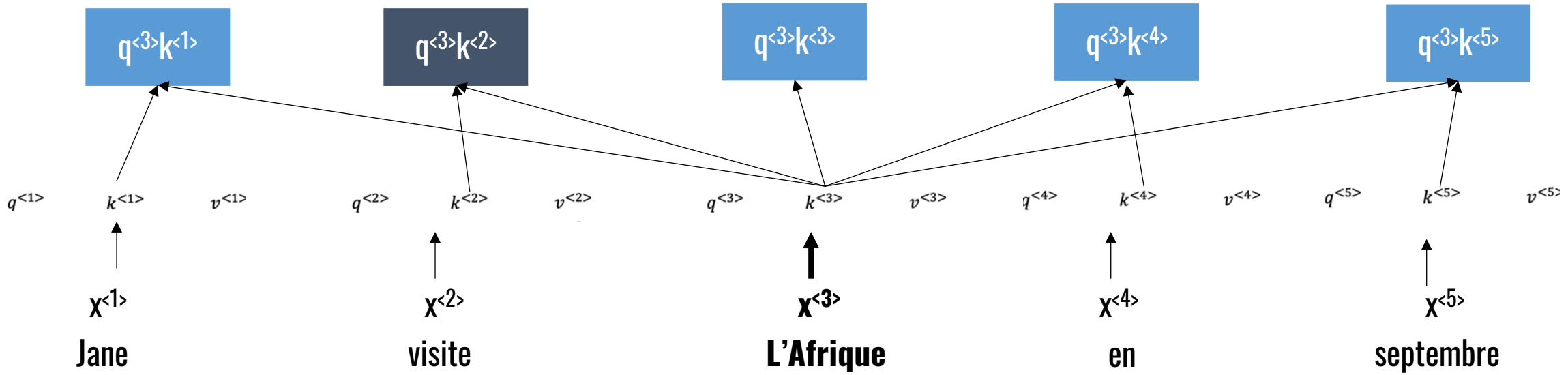
What's happening there?



Self-Attention

$q^{<3>} = W^Q \cdot x^{<3>}$ $k^{<3>} = W^K \cdot x^{<3>}$ $v^{<3>} = W^V \cdot x^{<3>}$

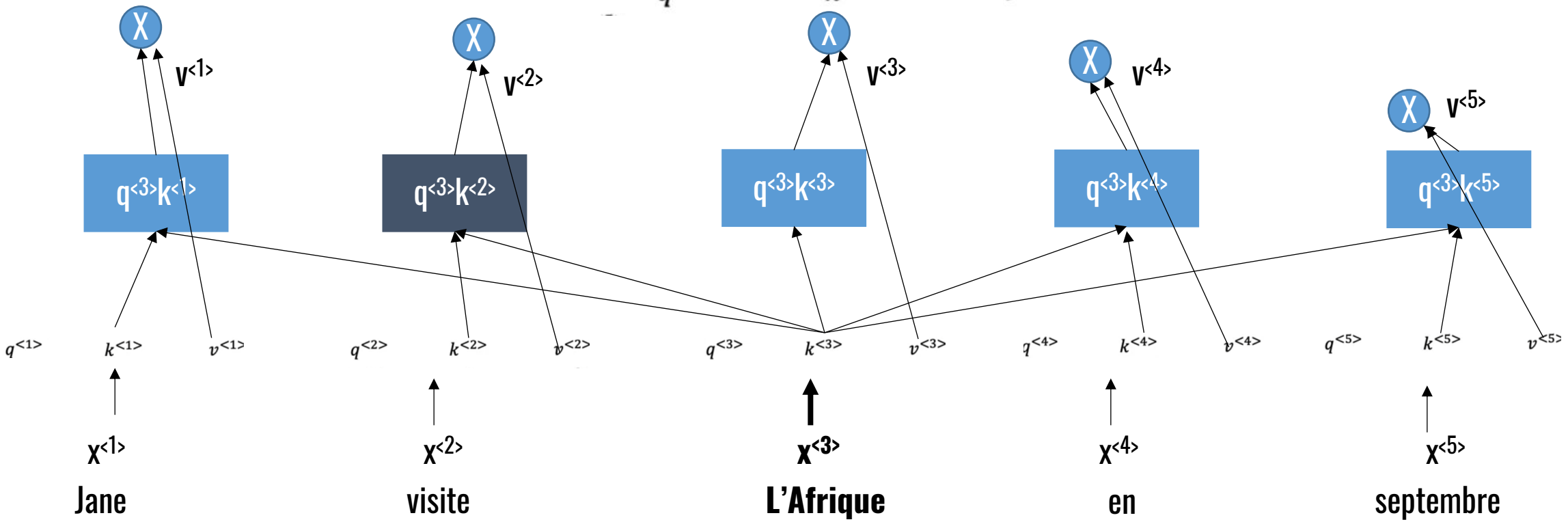
Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$
$q^{<5>}$	$k^{<5>}$	$v^{<5>}$



Self-Attention

$q^{<3>} = W^Q \cdot x^{<3>}$ $k^{<3>} = W^K \cdot x^{<3>}$ $v^{<3>} = W^V \cdot x^{<3>}$

Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$
$q^{<5>}$	$k^{<5>}$	$v^{<5>}$



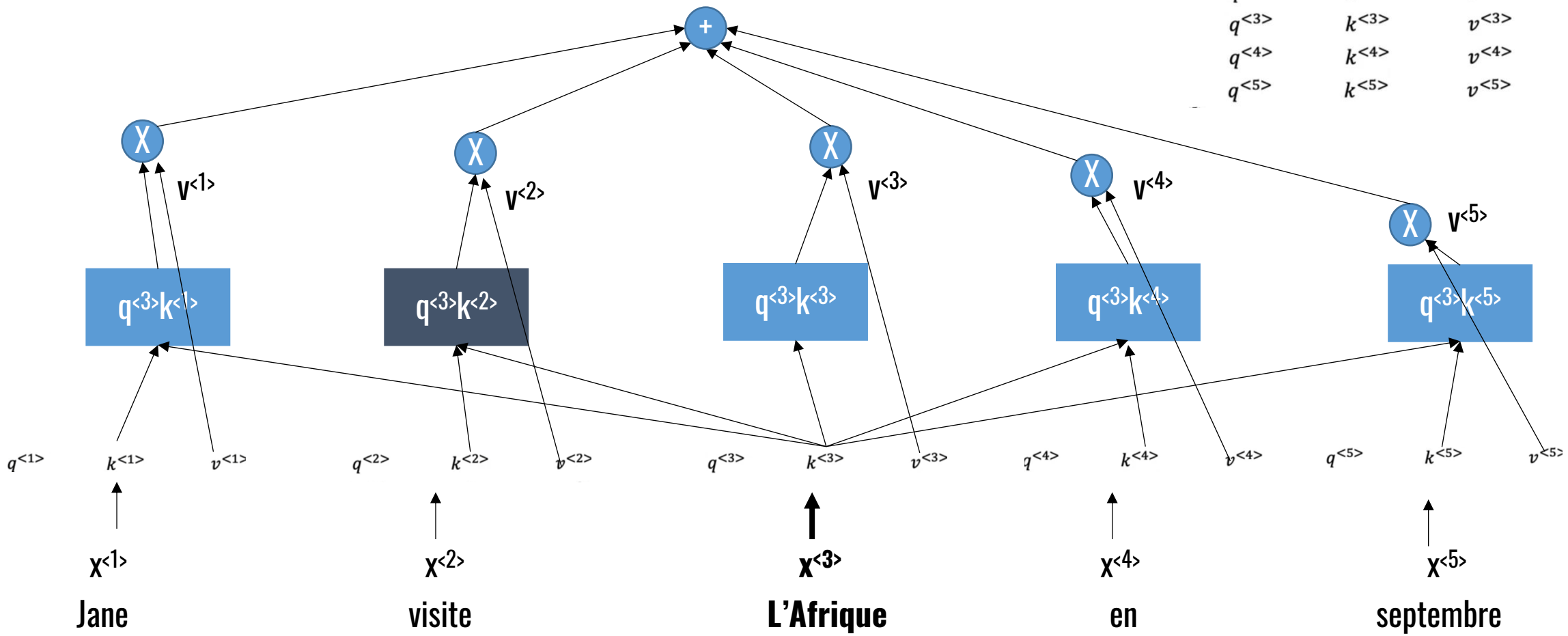
Self-Attention

$$\text{Attention}(Q,K,V) = \sum_i \frac{\exp(q \cdot k^{<i>})}{\sum_j \exp(q \cdot k^{<j>})} v^{<i>}$$

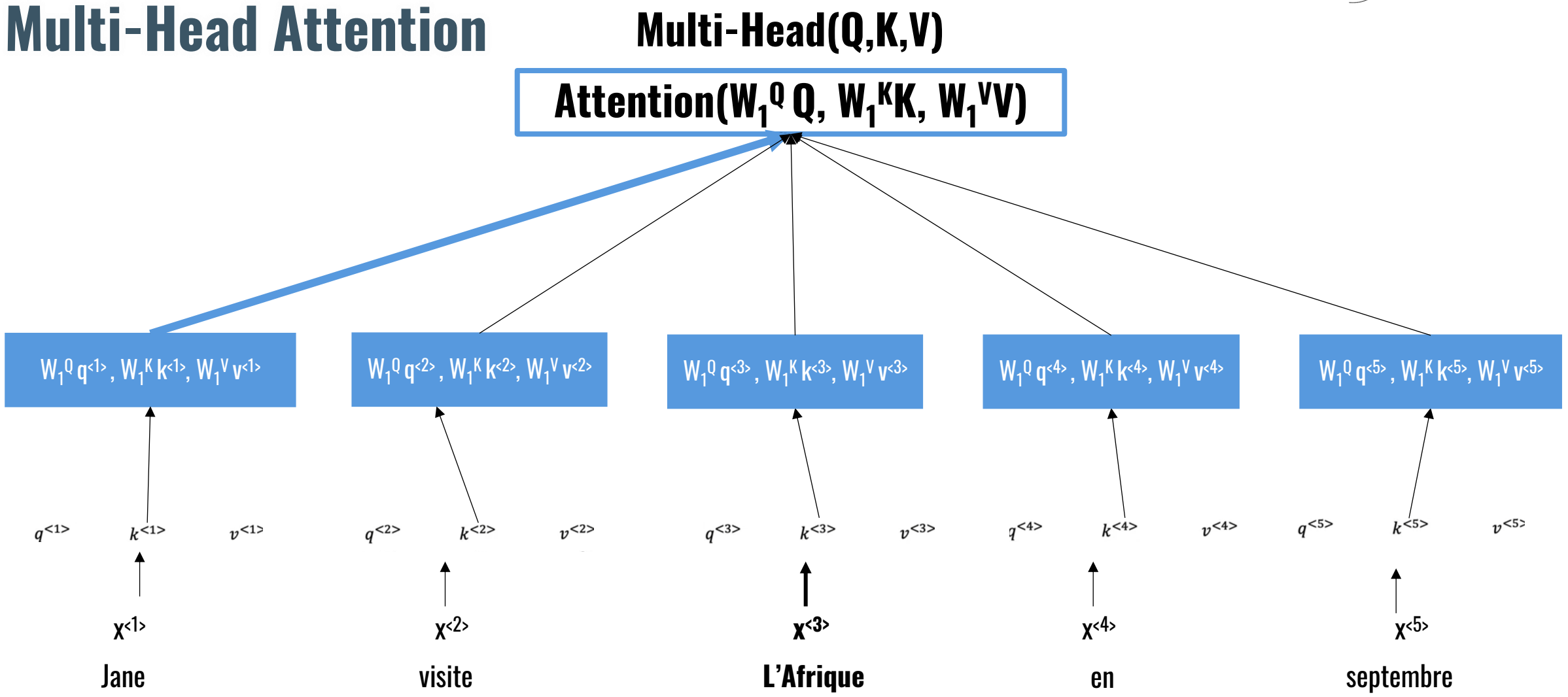
$q^{<3>} = W^Q \cdot x^{<3>}$ $k^{<3>} = W^K \cdot x^{<3>}$ $v^{<3>} = W^V \cdot x^{<3>}$

Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$
$q^{<5>}$	$k^{<5>}$	$v^{<5>}$

$A^{<3>} = A(q^{<3>},K,V)$

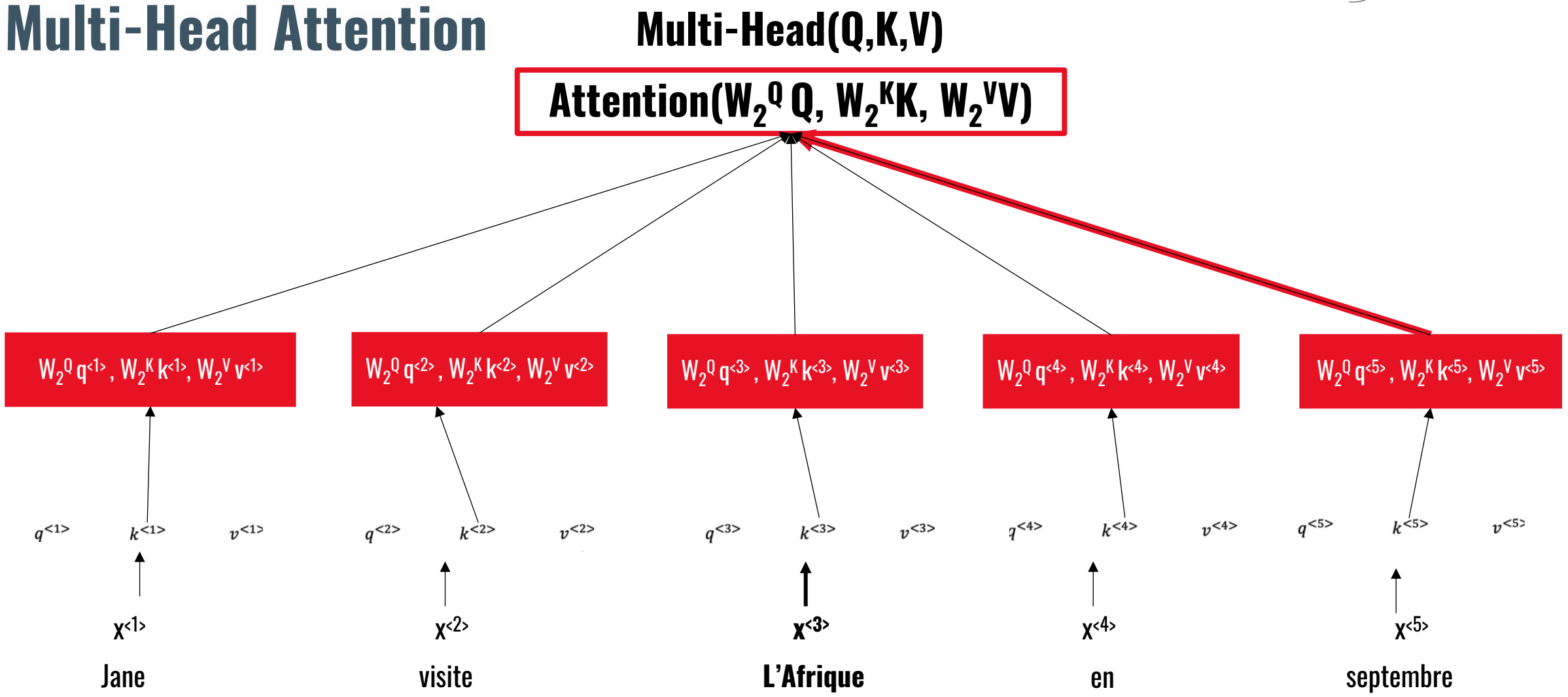


Multi-Head Attention



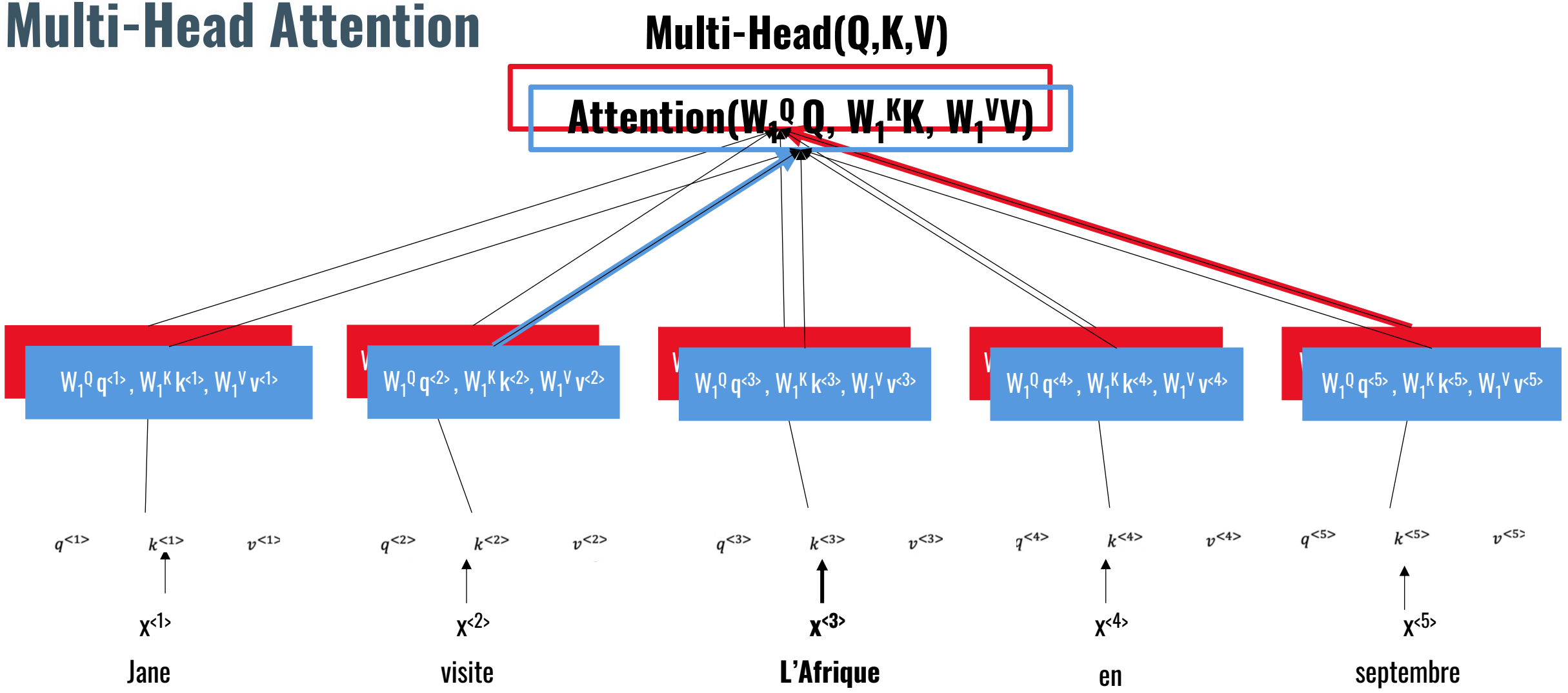
W_1^Q , W_1^K , W_1^V – what's happening there?

Multi-Head Attention



W_2^Q , W_2^K , W_2^V – when?

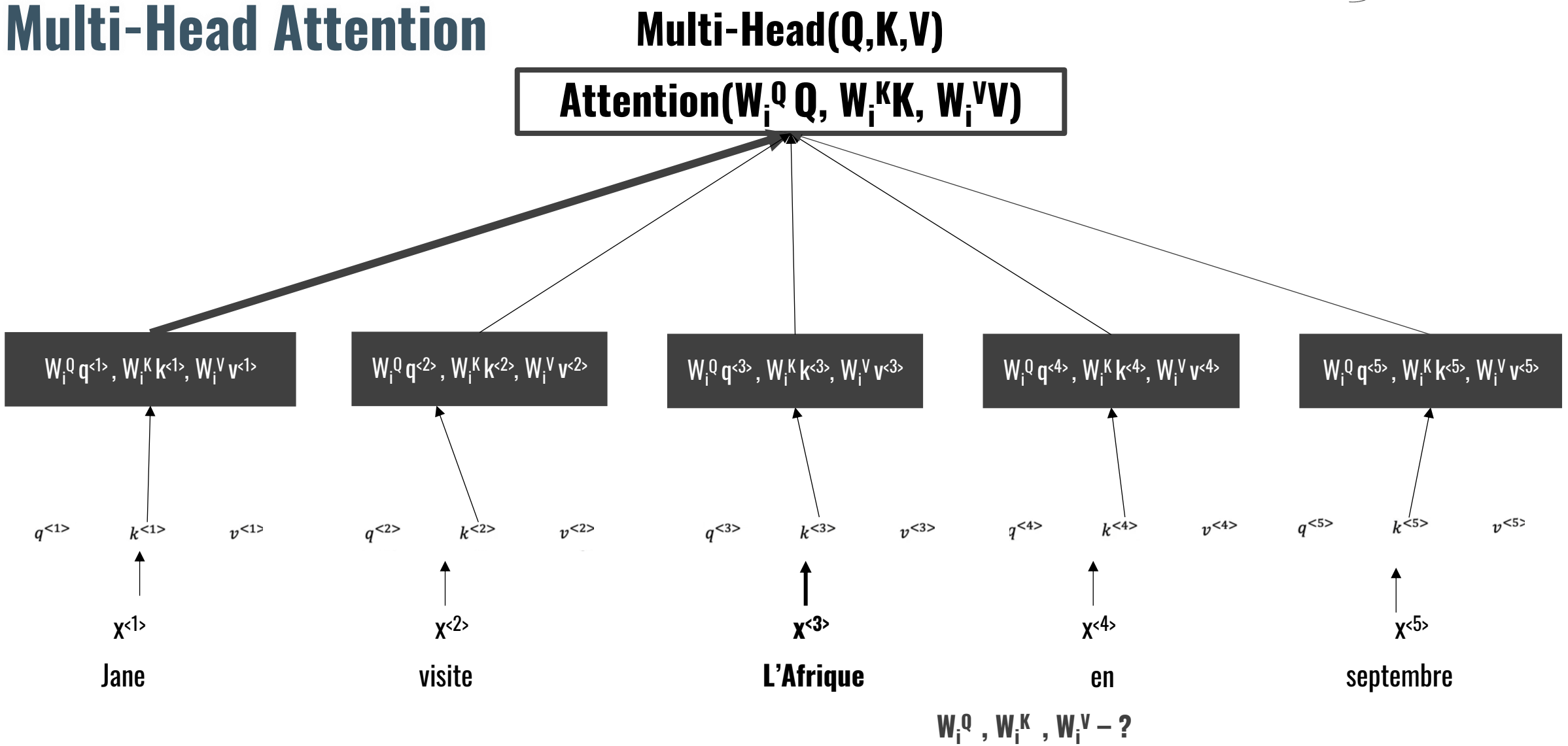
Multi-Head Attention



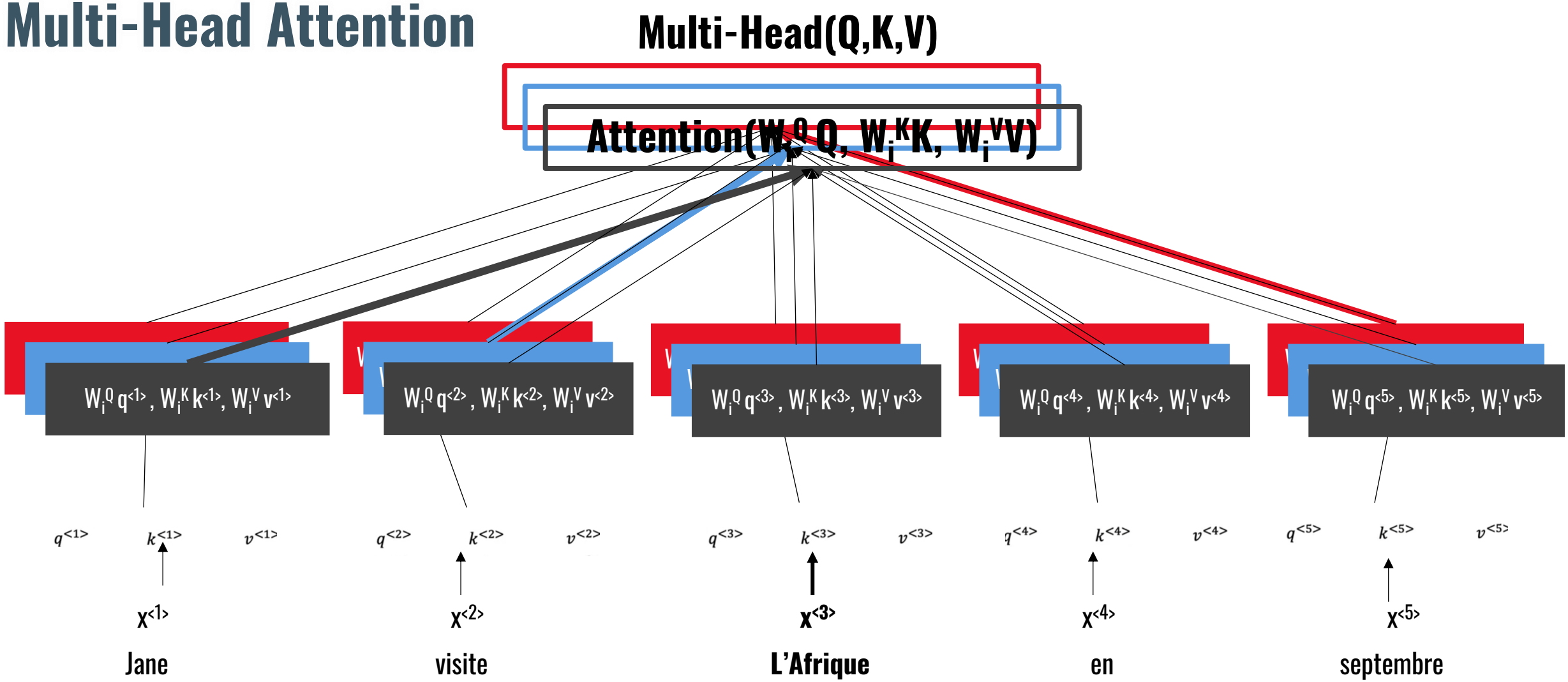
W_2^Q , W_2^K , W_2^V – when?

W_1^Q , W_1^K , W_1^V – what's happening there?

Multi-Head Attention



Multi-Head Attention



Summary

1. Attention

- How does the input elements relate to the output (what part of the input is relevant for the output)
  - Helps the model in focusing on the most relevant information

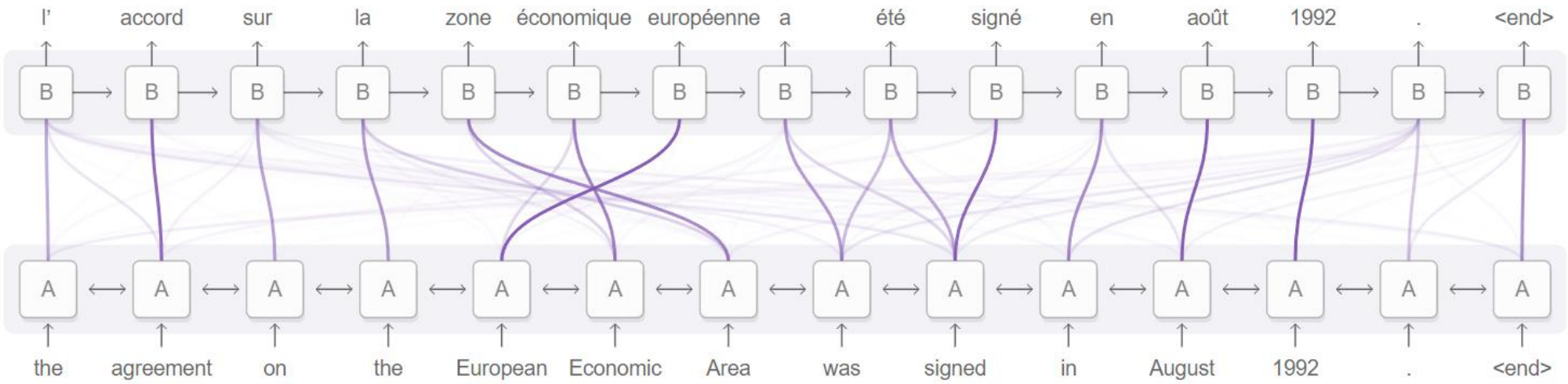


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

Image Source: Attention between RNNs — Chris Olah & Shan Carter, Attention and Augmented Recurrent Neural Networks, Distill (2016), derived from Fig. 3 of Bahdanau et al. (2014), CC BY 2.0 https://creativecommons.org/licenses/by/2.0/, https://distill.pub/2016/augmented-rnns

Summary

2. Self-Attention

- How do the different elements of the input relate to each other (what part of the input is relevant in understanding the other parts of the input)
  - Helps in capturing dependencies and contextual information within the input

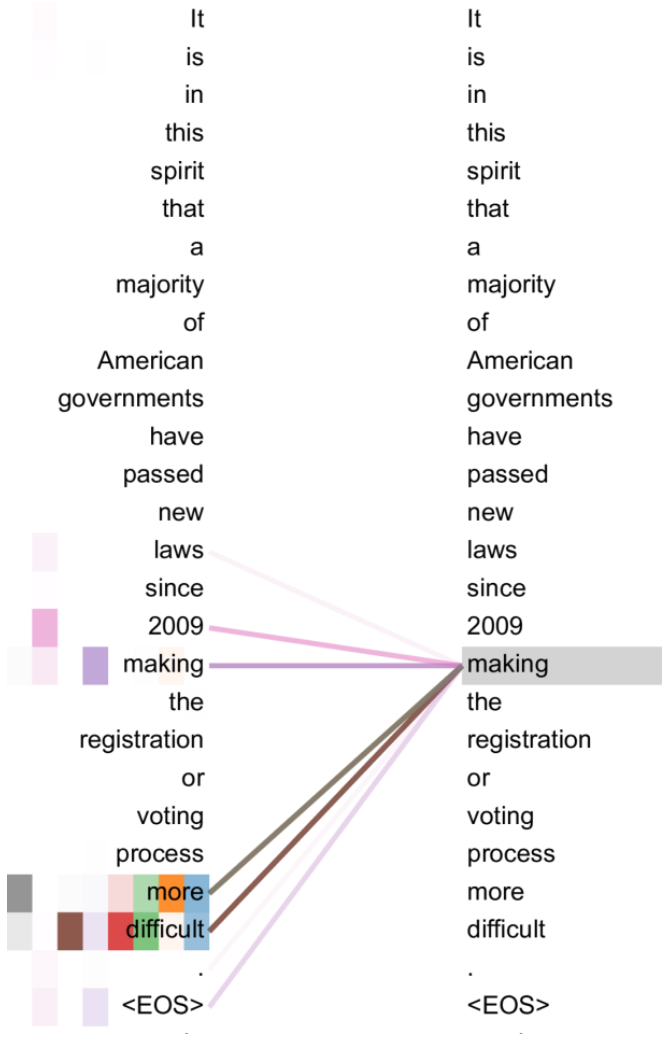
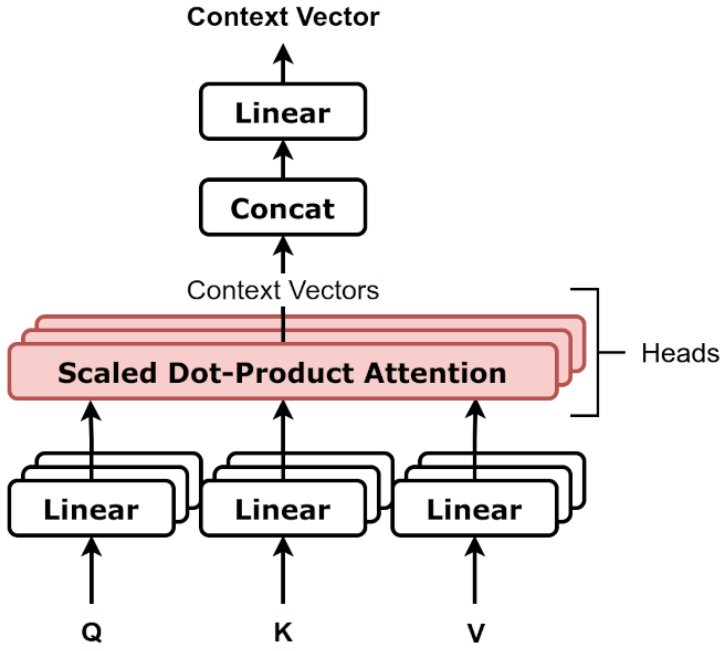


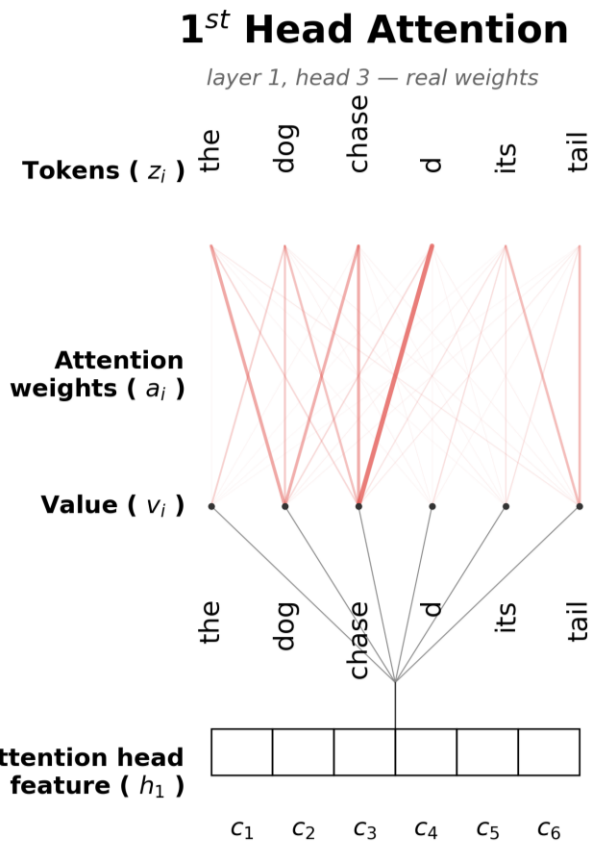
Image Source: Attention Is All You Need - long-distance attention, encoder self-attention in layer 5 of 6.png, Google, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Summary

3. Multi-Head Attention

- Repeating self-attention multiple times to better understand the input (multiple ways of deciding the relationship between the parts of the input)
  - Looks at the input from different perspectives





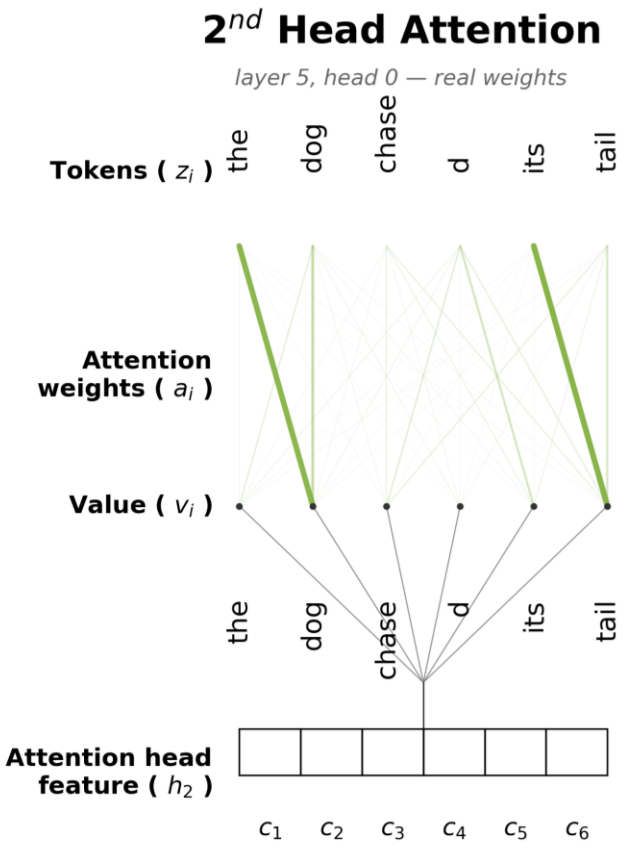
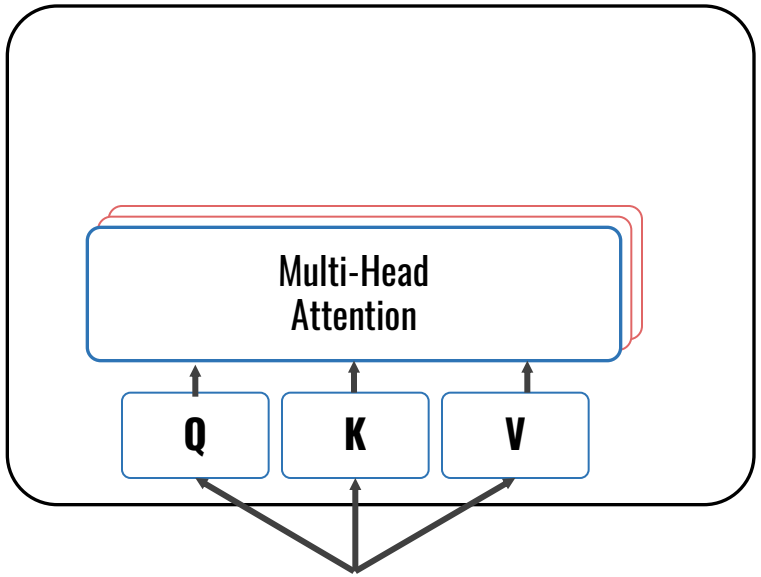


Image Source: Multiheaded attention, block diagram.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer

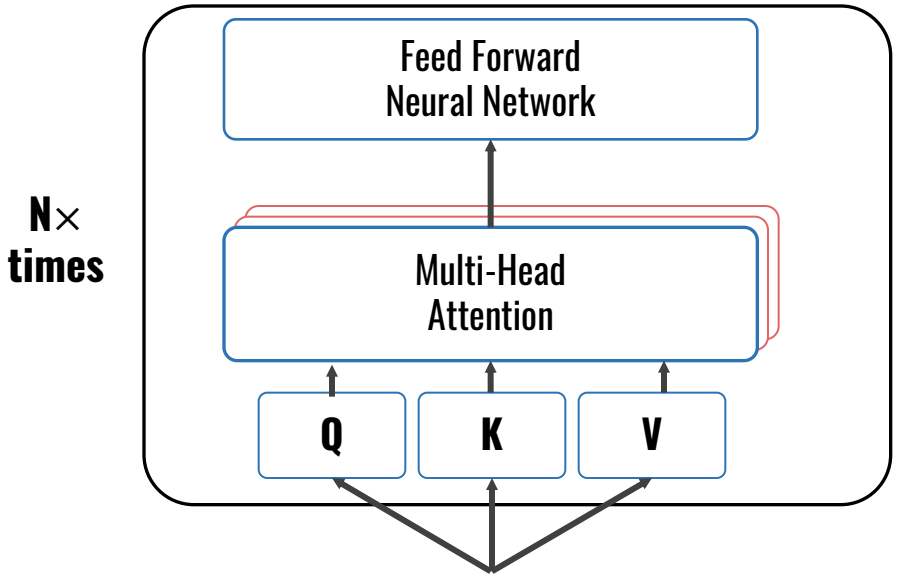
<SOS>	$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	...	$x^{<n-1>}$	$x^{<n>}$	<EOS>
	I	will	complete	my	assignment	on	time	.

Transformer



<SOS>	$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	...	$x^{<n-1>}$	$x^{<n>}$	<EOS>
	I	will	complete	my	assignment	on	time	.

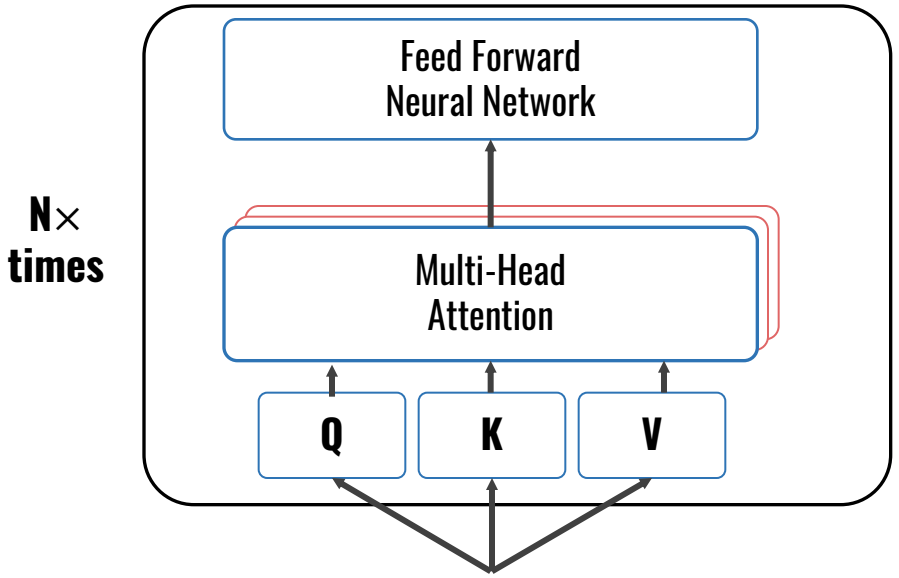
Transformer



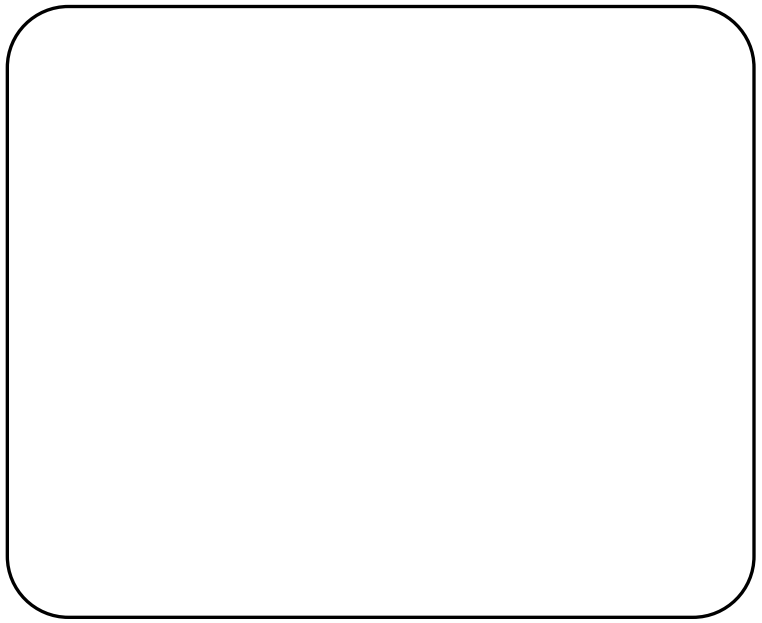
<SOS>	$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	...	$x^{<n-1>}$	$x^{<n>}$	<EOS>
	I	will	complete	my	assignment	on	time	.

Transformer

Encoder

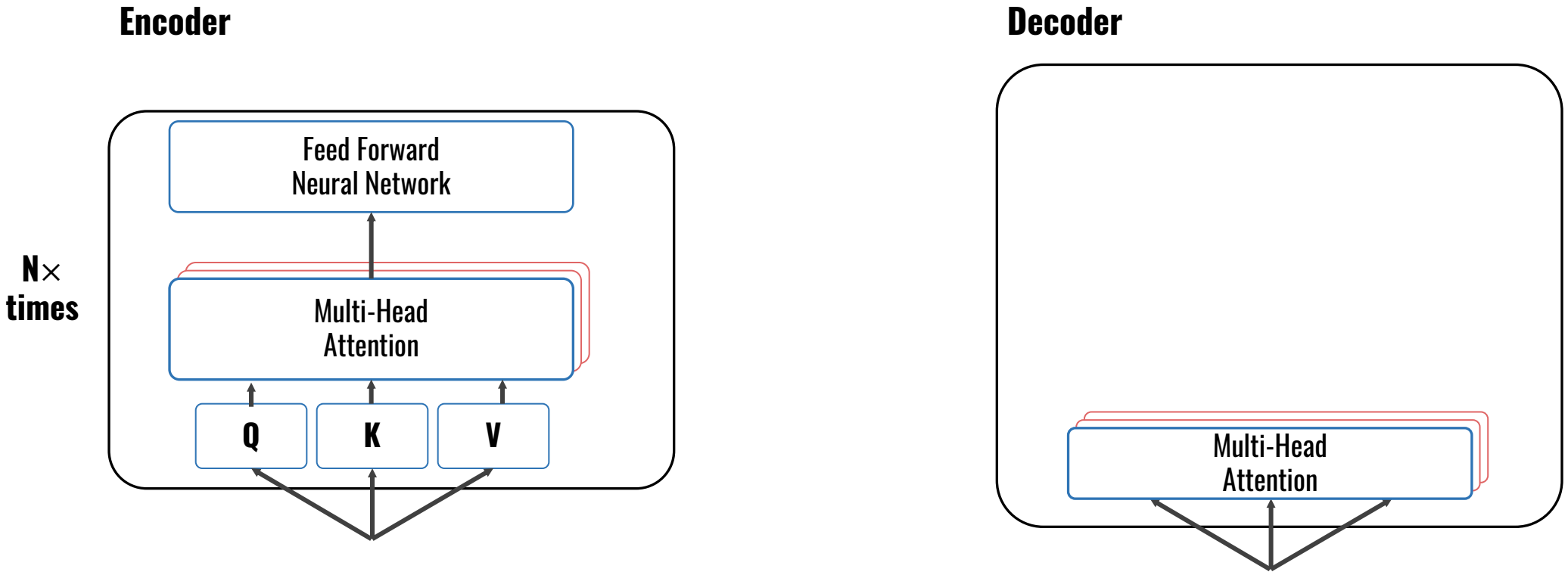


Decoder

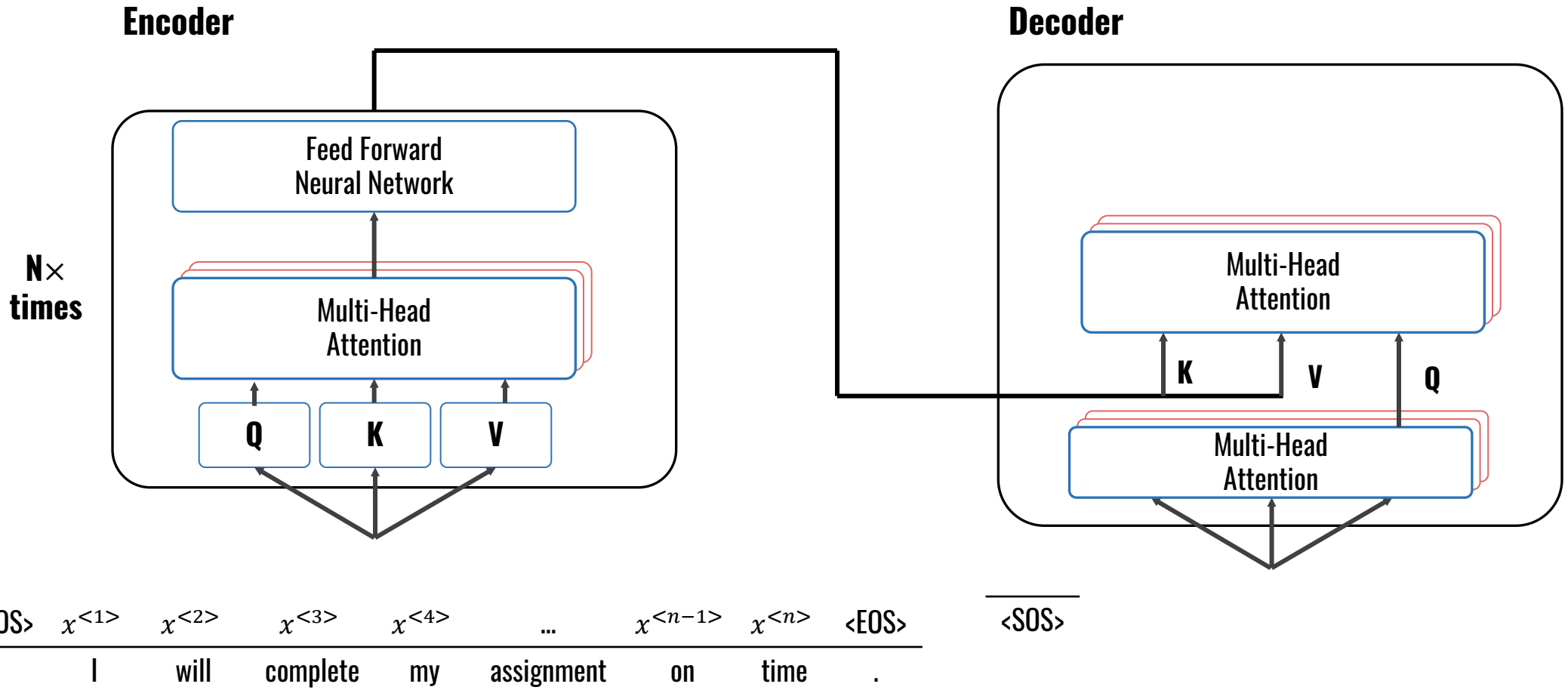


<SOS>	$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	...	$x^{<n-1>}$	$x^{<n>}$	<EOS>
	I	will	complete	my	assignment	on	time	.

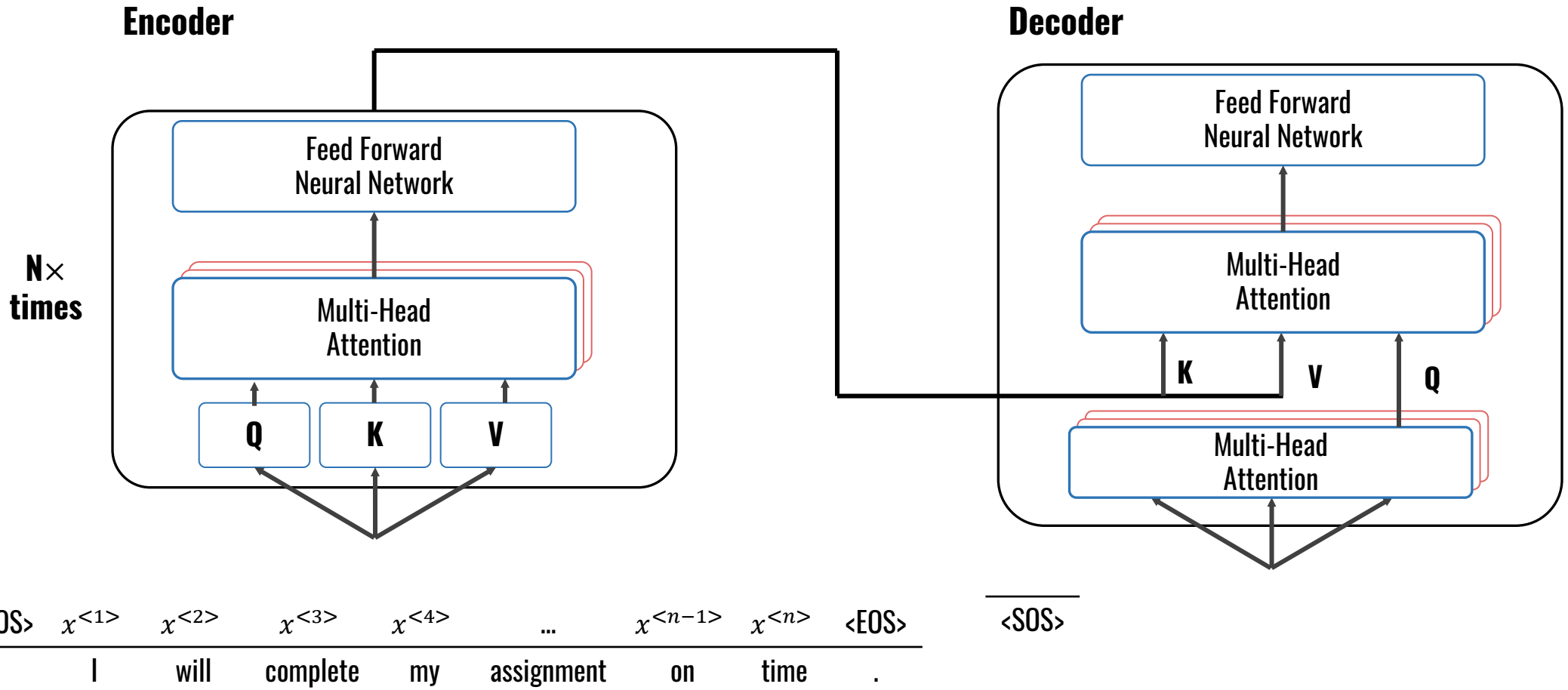
Transformer



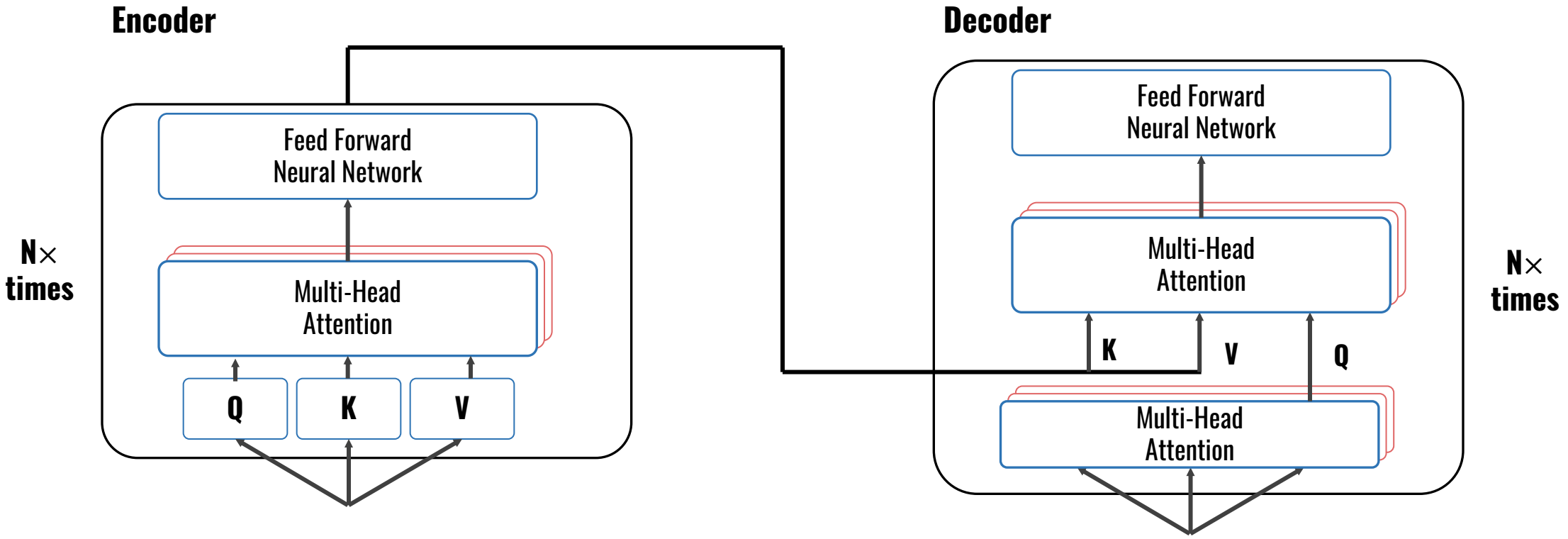
Transformer



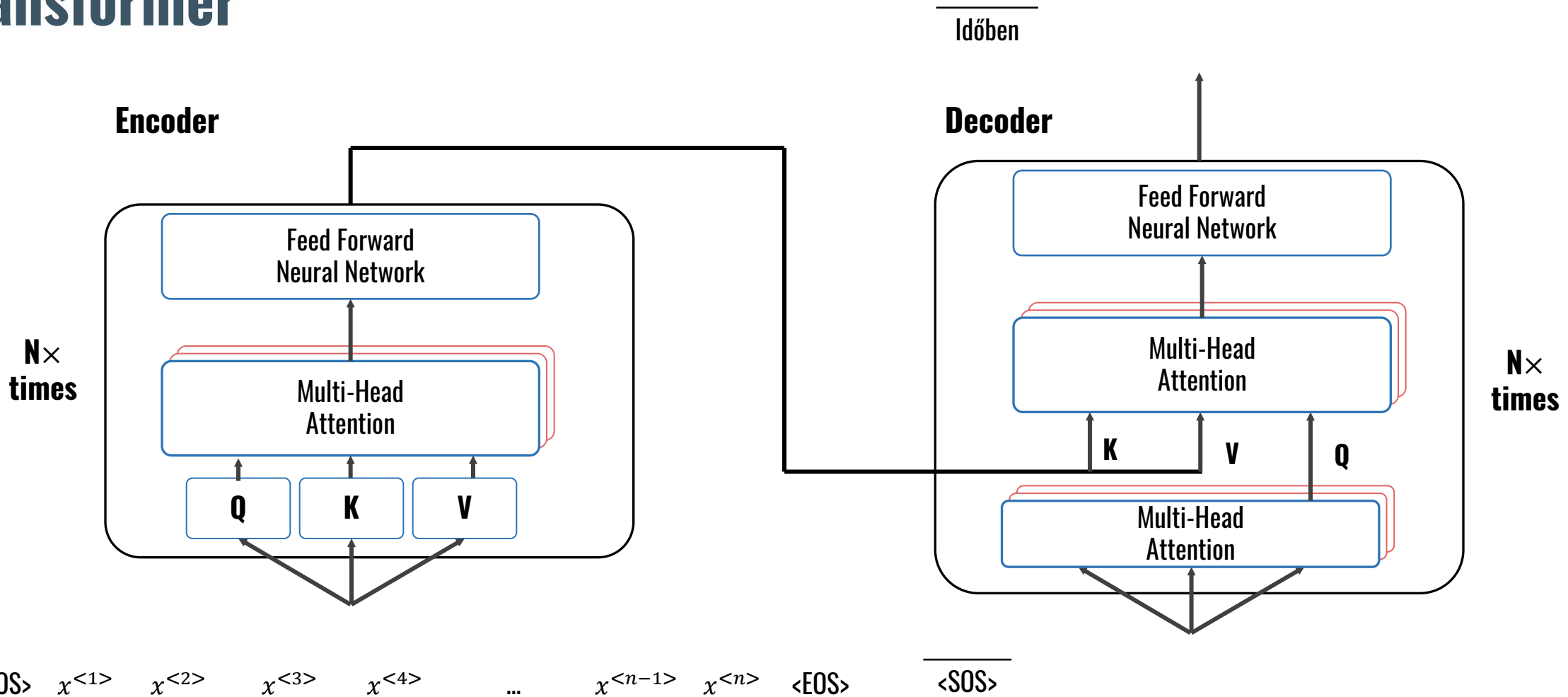
Transformer



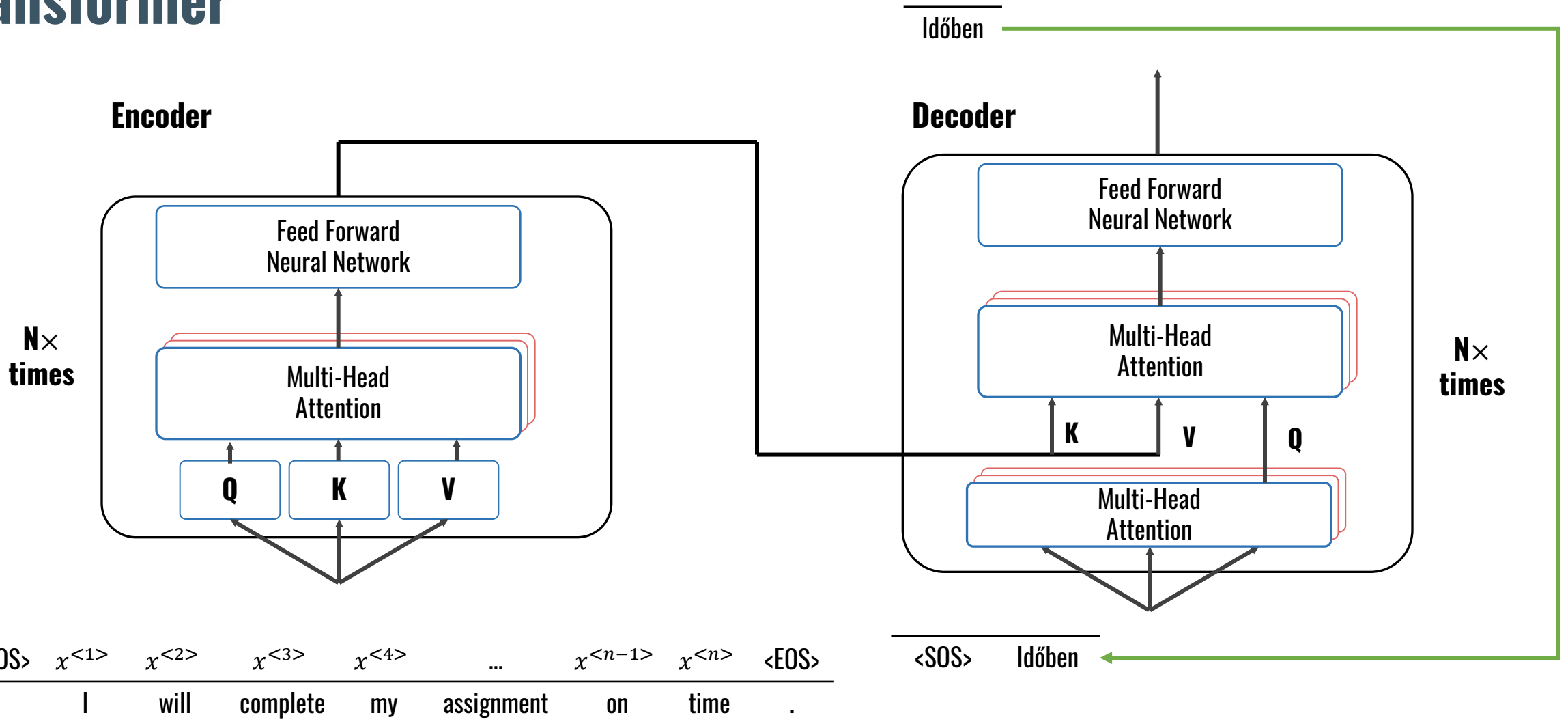
Transformer



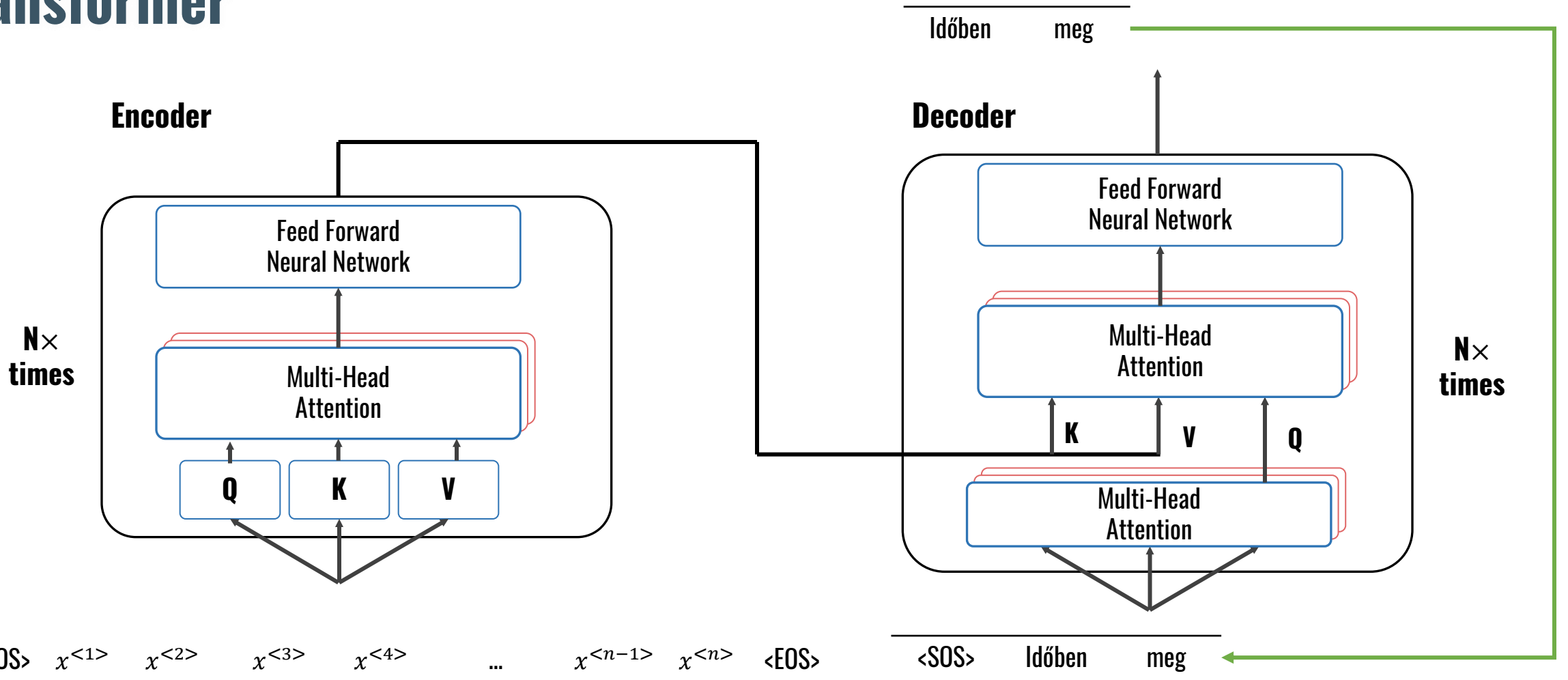
Transformer



Transformer



Transformer



Transformer

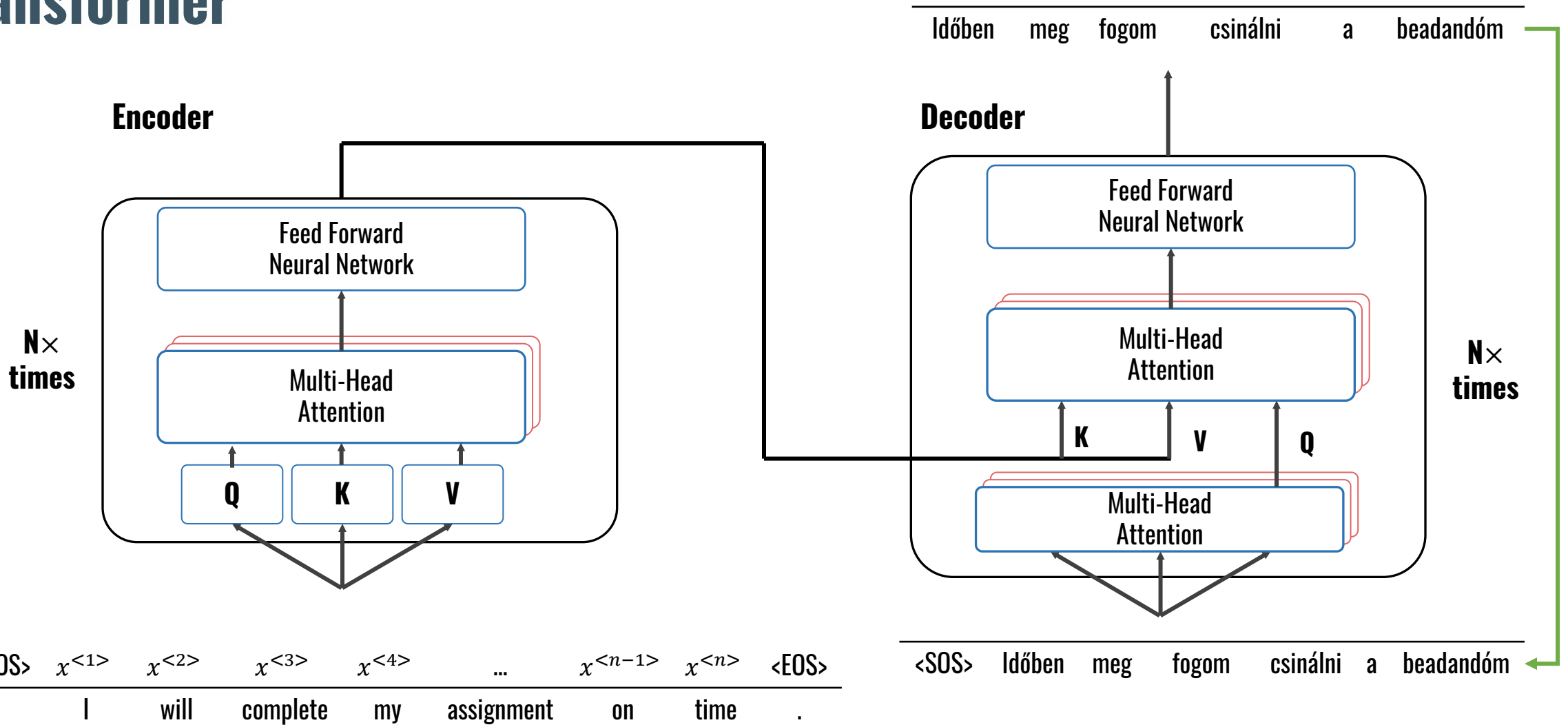


Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer

- When you convert a sequence into a set (**tokenization**), you lose the notion of order.
- **Positional encoding** is a set of small constants, which are added to the word embedding vector before the first self-attention layer.
- So if the same word appears in a different position, the actual representation will be slightly different, **depending on where it appears in the input sentence**.
- They use a sinusoidal function for the positional encoding. The sine function tells the model to pay attention to a particular wavelength.

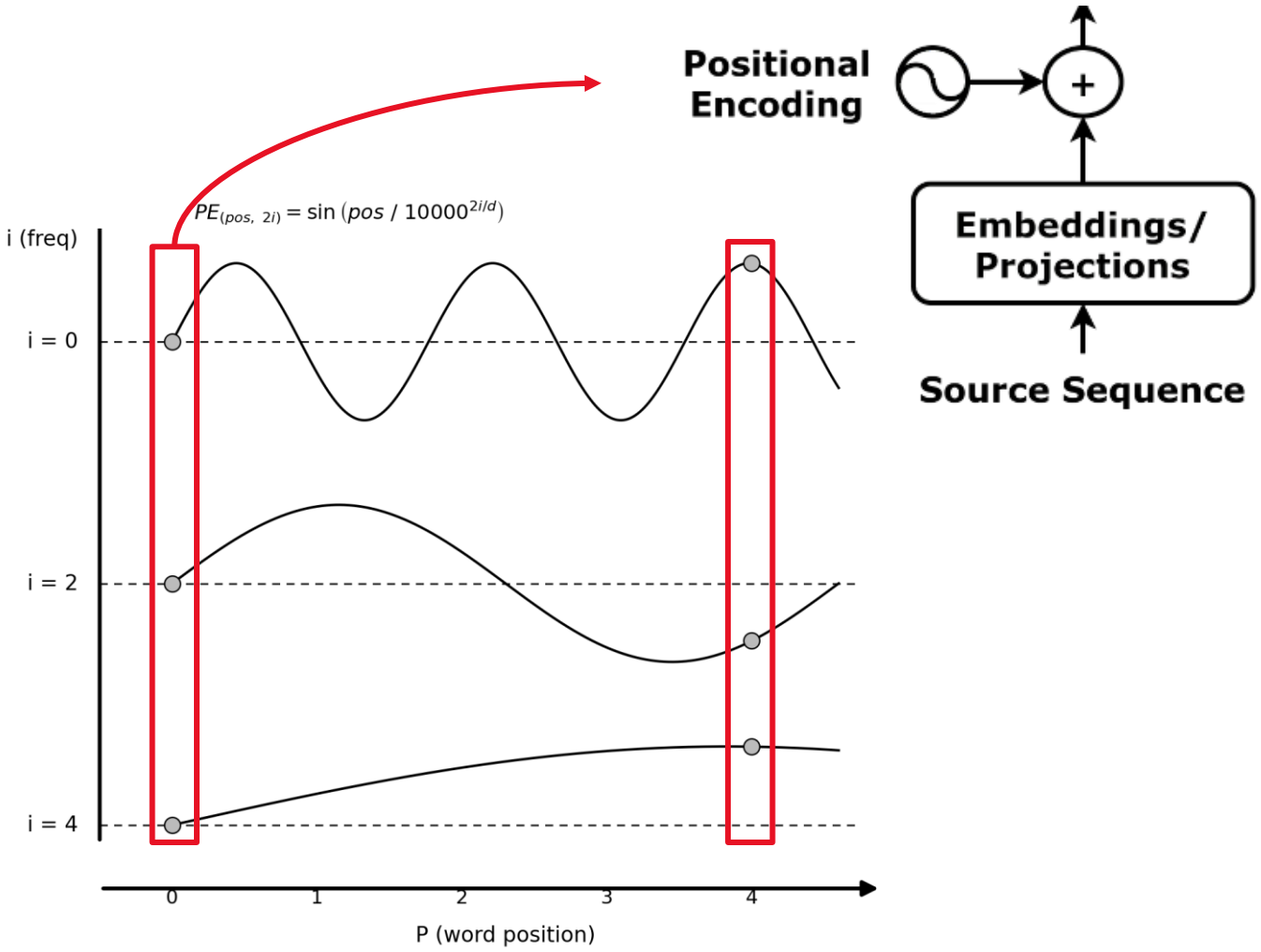
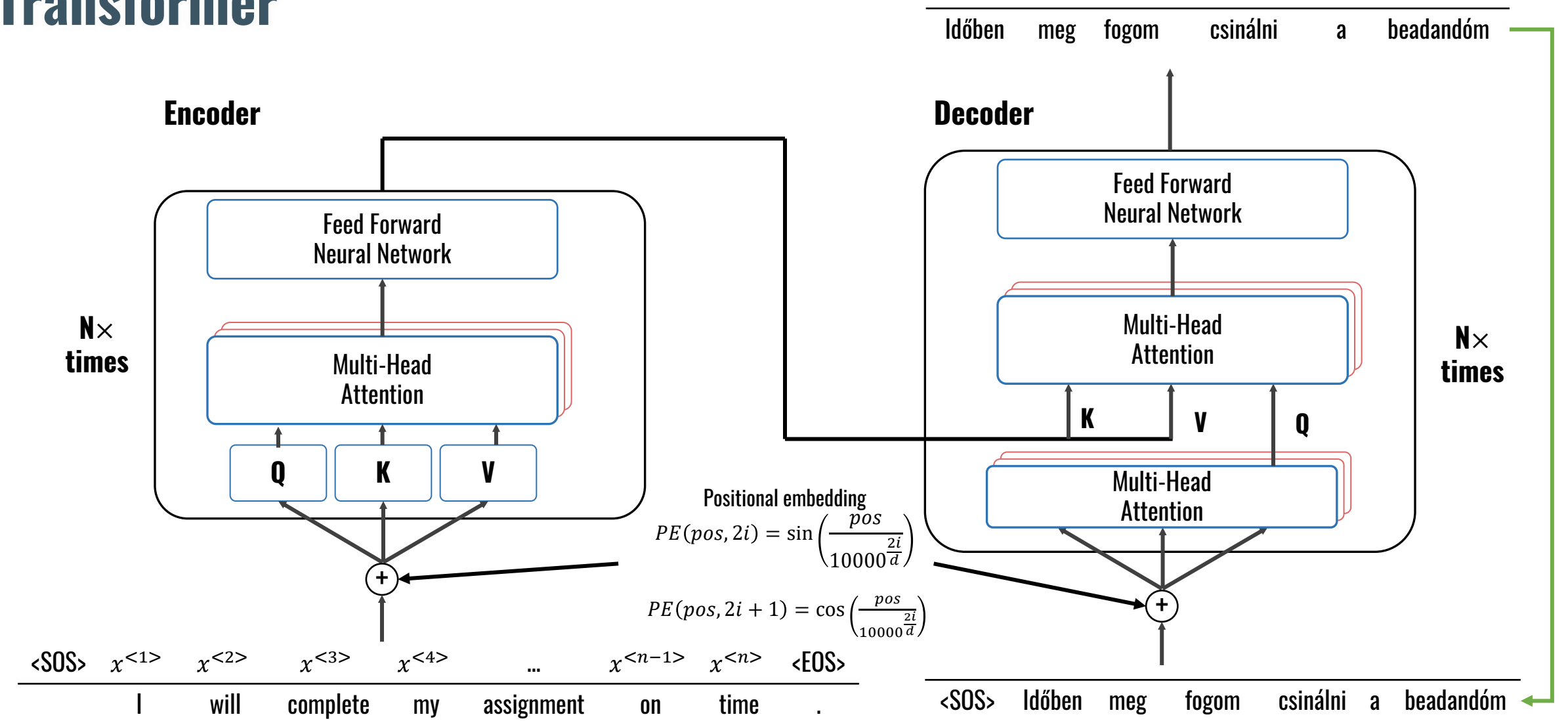
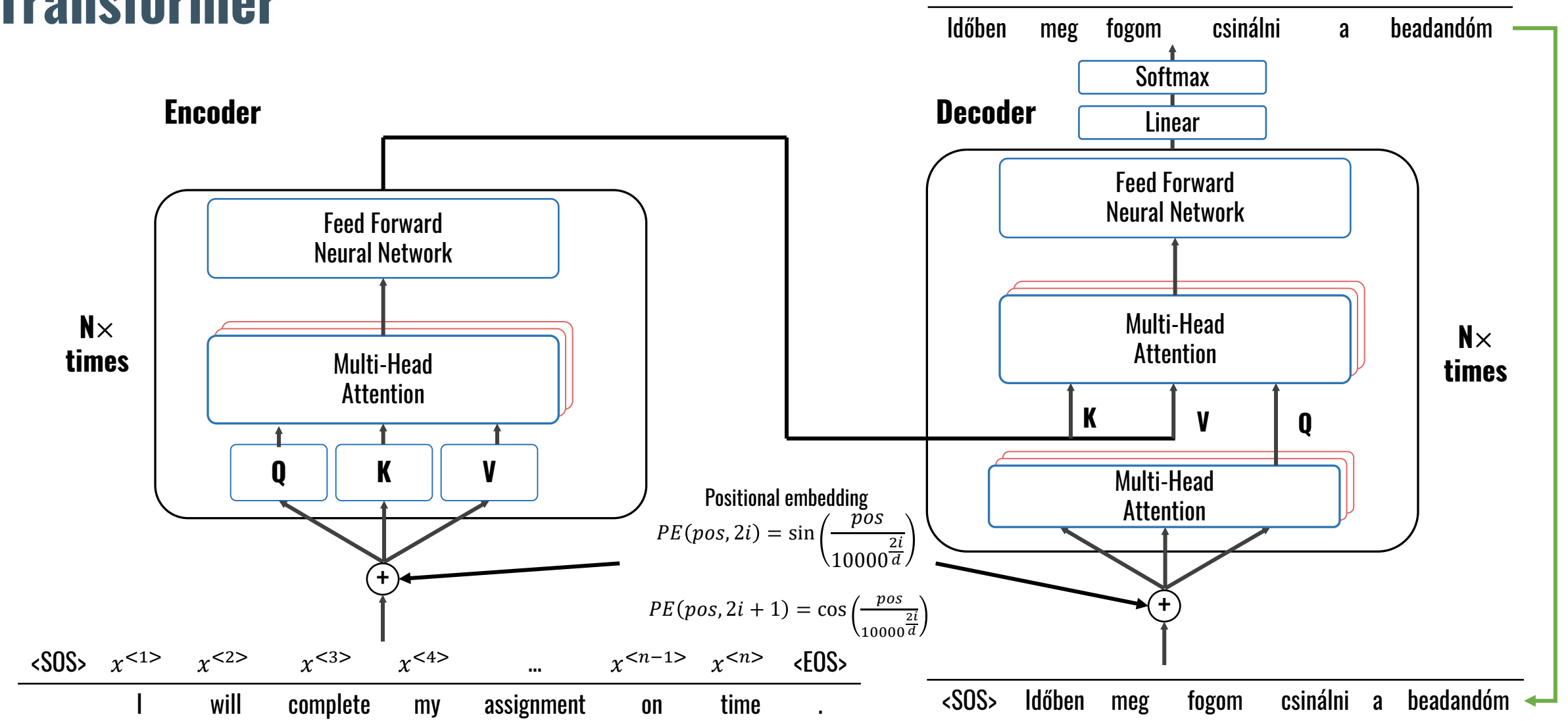


Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer



Transformer



Transformer Encoder

Summary: the Transformer encoder

To process a sentence, we need these 3 steps:

1. Word embeddings of the input sentence are computed simultaneously.
2. Positional encodings are then applied to each embedding resulting in word vectors that also include positional information.
3. The word vectors are passed to the first encoder block.

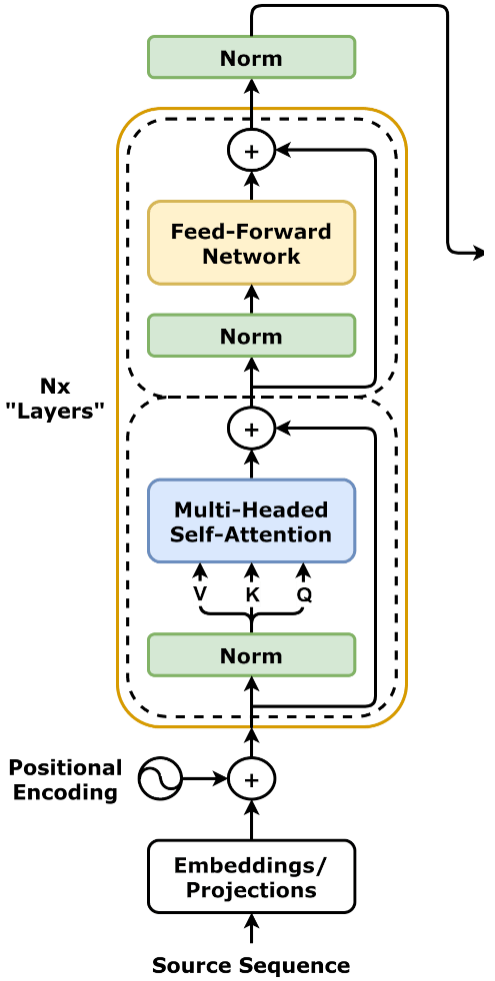


Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer Encoder

Each block consists of the following layers in the same order:

1. A multi-head self-attention layer to find correlations between each word
2. A normalization layer
3. A residual connection around the previous two sublayers
4. A linear layer
5. A second normalization layer
6. A second residual connection

Note that the above block can be replicated several times to form the Encoder. In the original paper, the encoder composed of 6 identical blocks.

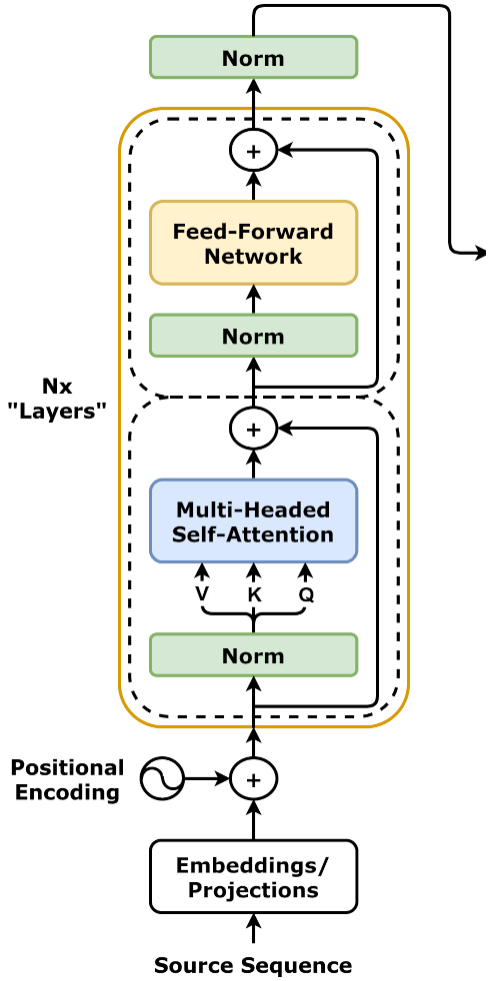


Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer Decoder

Transformer decoder: what is different?

The decoder consists of all the previously mentioned components plus two novel ones. As before:

1. The output sequence is fed in its entirety and word embeddings are computed
2. Positional encoding are again applied
3. And the vectors are passed to the first Decoder block

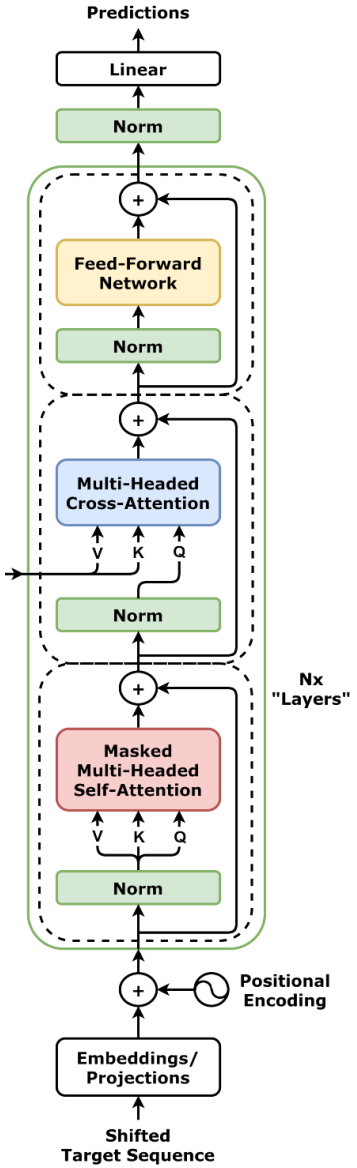


Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer Decoder

Each decoder block includes:

1. A **Masked** multi-head self-attention layer
2. A normalization layer followed by a residual connection
3. A new multi-head attention layer (known as **Encoder-Decoder attention**)
4. A second normalization layer and a residual connection
5. A linear layer and a third residual connection

The decoder block appears again 6 times. The final output is transformed through a final linear layer and the output probabilities are calculated with the standard softmax function.

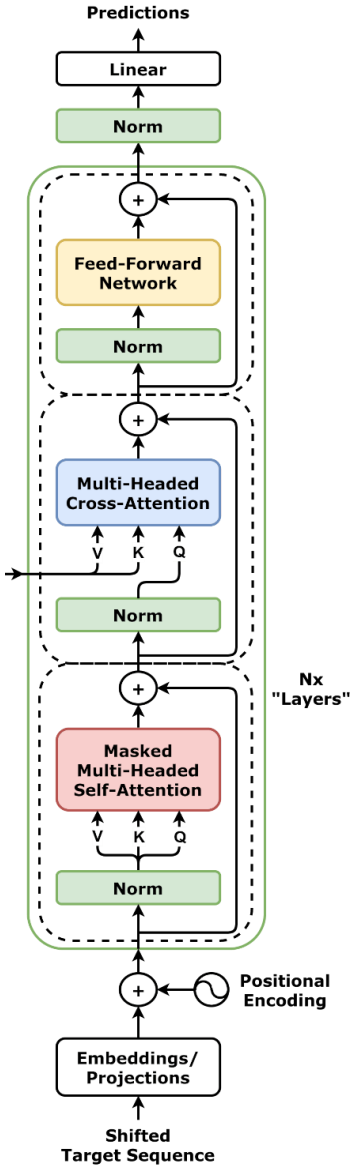
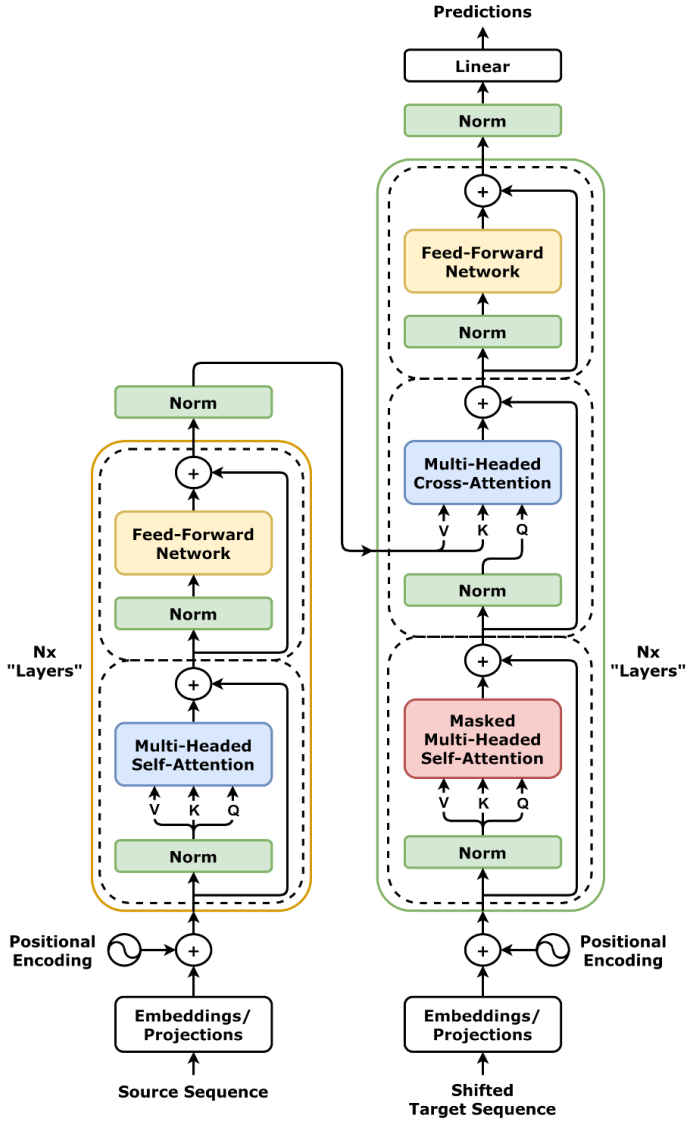


Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

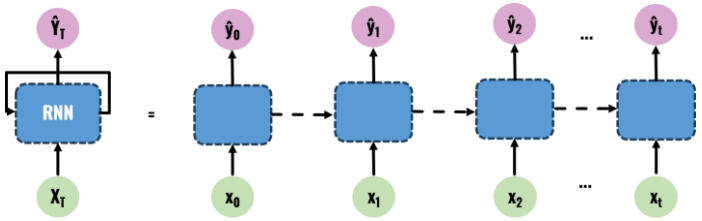
Transformer



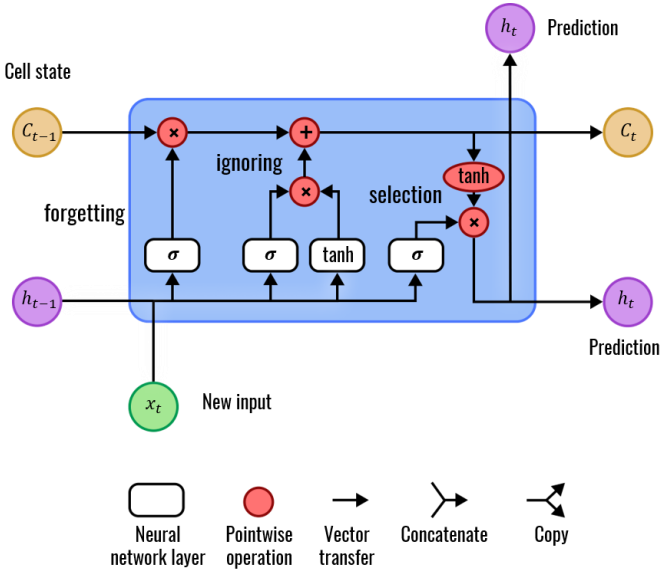
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762
Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Sequence models

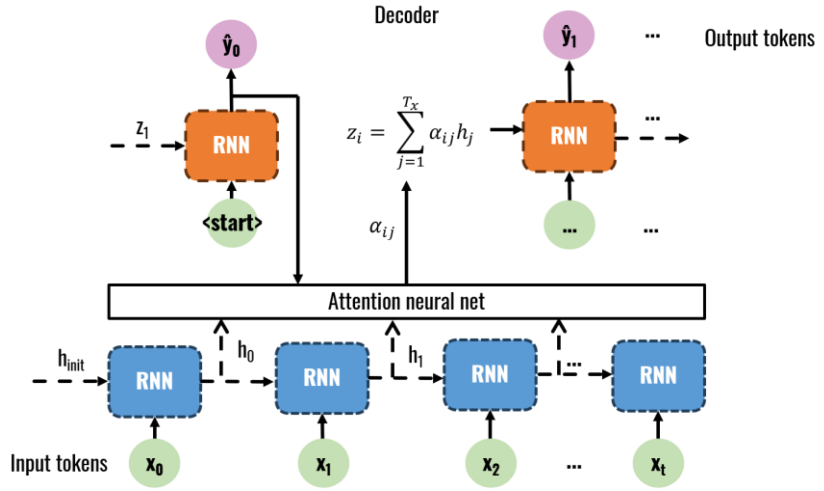
RNN



LSTM



RNN with attention



Transformer

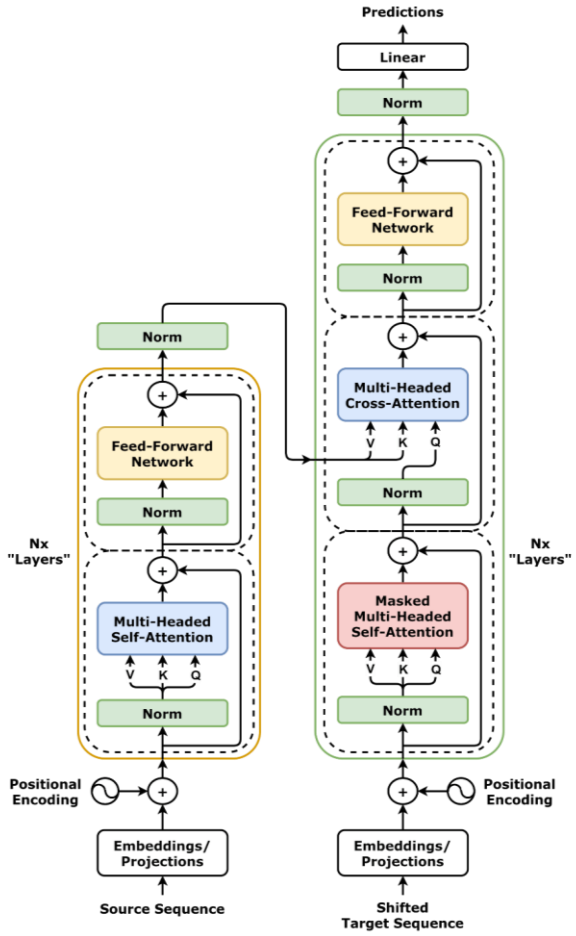
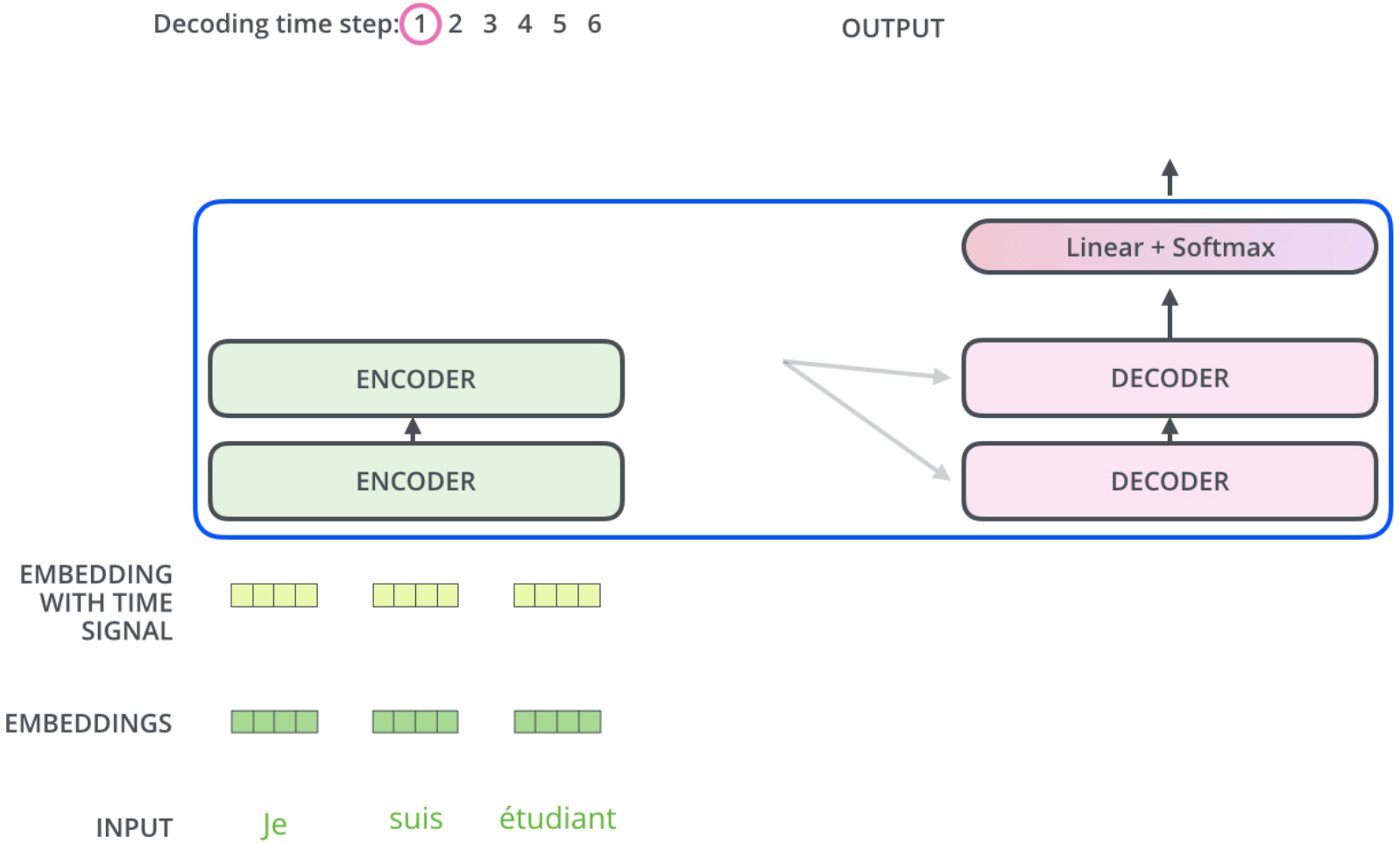


Image Source: Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer



BLEU (bilingual evaluation understudy)

- **BLEU** is a metric used to evaluate the **quality of text** that has been machine-translated from one language into another.
- It works by measuring the **precision** of overlapping **n-grams** (such as bigrams or trigrams) between the **predicted translation** and the **reference translations**



Prediction	**He went to the store to buy some groceries**
Target 1	He went to the shop to get groceries
Target 2	He went to the store for some food

N-gram

- Different way of saying n consecutive words in a sentence.
- **“He played basketball yesterday"**, has the following n-grams are present:

  - **1-gram (unigram):** “He", “played", “basketball", “yesterday"
  - **2-gram (bigram):** “He played", “played basketball", “basketball yesterday"
  - **3-gram (trigram):** “He played basketball", “played basketball yesterday"
  - **4-gram:** “He played basketball yesterday"



Prediction	**He went to the store to buy some groceries**
Target 1	He went to the shop to get groceries
Target 2	He went to the store for some food

Precision

- Let's say, that we have:
  - **Target Sentence:** He played basketball
  - **Predicted Sentence:** He plays basketball
- We would normally compute the Precision using the formula:

$$\textbf{Precision} = \frac{\textbf{Number of correct predicted words}}{\textbf{Number of total predicted words}}$$

- Here **played** and **plays** count as different words
- Precision = 2 / 3

Precision

- The issue is that only using precision allow us to cheat:
    - **Target Sentence:** He played basketball yesterday
    - **Predicted Sentence:** He he he
- Precision = 3 / 3

Clipped Precision

- **Target Sentence 1: He is eating pizza for lunch**
- **Target Sentence 2: He eats pizza for lunch**
- **Predicted Sentence: He He He eats pizza for dinner**

1. Each word in the predicted sentence is checked against every target sentence for a match.
2. A word is marked correct if it appears in at least one of the target sentences.
3. Each correct word is capped at the number of times it appears in that target sentence.

$$\text{Clipped Precision} = \frac{\text{Clipped \# of correct predicted words}}{\text{\# of total predicted words}}$$

Word	Matching sentence	Matched predicted count	Clipped count
He	Both	3	1
eats	Target 2	1	1
pizza	Both	1	1
for	Both	1	1
dinner	None	0	0
Total		6	4

Clipped Precision = **4 / 7**

How is BLEU score calculated?

Precision 1-gram: $p_1 = \frac{4}{6}$

Precision 2-gram: $p_2 = \frac{2}{5}$



Prediction **He is swimming in the pool**

Target 1 He swims in the pool during the day



Prediction **He is swimming in the pool**

Target 1 He swims in the pool during the day

Precision 3-gram: $p_3 = \frac{1}{4}$

Precision 4-gram: $p_2 = \frac{0}{3}$



Prediction **He is swimming in the pool**

Target 1 He swims in the pool during the day



Prediction **He is swimming in the pool**

Target 1 He swims in the pool during the day

How is BLEU score calculated?

Geometric Average Precision (GAP)

- We combine these precision scores using the following formula

$$\mathbf{GAP}(N) = \prod_{n=1}^{N} p_n^w$$

- We usually use $N = 4$ and uniform weights $w = \frac{1}{N}$
- In case of $N = 4$ the Geometric Average Precision:

$$\mathbf{GAP}(N) = (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}}$$

How is BLEU score calculated?

Brevity Penalty

- The Brevity Penalty penalizes sentences that are too short.

$$\mathbf{BrevityPenalty} = \begin{cases} 1, & \text{if } c > r \\ e^{(1-\frac{r}{c})}, & \text{if } c \leq r \end{cases}$$

- c is the predicted length -> number of words in the predicted sentence
- r is the target length -> number of words in the target sentence (length closes to prediction length)

How is BLEU score calculated?

- To get the BLEU score we multiply the **Brevity Penalty** with the **Geometric Average** of the **Precision Scores**

$$\mathbf{BLEU}(N) = \mathbf{BrevityPenalty} \cdot \mathbf{GAP}(N)$$

- **BLEU** Score can be computed for different values of N.
  - **BLEU-1** uses the **unigram** Precision score
  - **BLEU-2** uses the geometric average of **unigram** and **bigram** precision
  - **BLEU-3** uses the geometric average of **unigram**, **bigram**, and **trigram** precision
  - and so on.

How is BLEU score calculated? (Example)

Prediction	**He eats an apple after lunch**
Target 1	He always eats an apple after lunch
Target 2	He is eating an apple after finishing lunch

N-gram	Matches	Total	p_n
1-gram	6	6	1.000
2-gram	4	5	0.800
3-gram	3	4	0.750
4-gram	2	3	0.667

$$BP = e^{1-r/c} = e^{1-7/6} \approx 0.8465$$

Length of the predicted sentence: c = 6
Length of the closes target in length: r = 7

$$\text{BLEU}(1) = \text{BP} \cdot \text{GAP}(1) = 0.8465 \cdot 1^{\frac{1}{1}} = 0.8465$$
$$\text{BLEU}(2) = \text{BP} \cdot \text{GAP}(2) = 0.8465 \cdot 1^{\frac{1}{2}} \cdot 0.8^{\frac{1}{2}} = 0.7571$$
$$\text{BLEU}(3) = \text{BP} \cdot \text{GAP}(3) = 0.8465 \cdot 1^{\frac{1}{3}} \cdot 0.8^{\frac{1}{3}} \cdot 0.75^{\frac{1}{3}} = 0.7140$$
$$\text{BLEU}(4) = \text{BP} \cdot \text{GAP}(4) = 0.8465 \cdot 1^{\frac{1}{4}} \cdot 0.8^{\frac{1}{4}} \cdot 0.75^{\frac{1}{4}} \cdot 0.667^{\frac{1}{4}} = 0.6732$$

Natural Language Processing Tasks

Name Entity Recognition

- A text corpus tagged with 4 different entity types
  - PER (persons), LOC (locations), ORG (organization), MISC (miscellaneous)

Index	0	1	2	3	4	5	6	7	8
Input X:	Harry	Potter	and	Hermione	Granger	invented	a	new	spell
Tags:	B-PER	I-PER	O	B-PER	I-PER	O	O	O	O
Output Y:	1	2	0	1	2	0	0	0	0

Index	0	1	2	3	4	5	6	7	8	9	10
Input X:	The	European	Commission	said	on	Thursday	it	disagreed	with	German	advice
Tags:	O	B-ORG	I-ORG	O	O	O	O	O	O	B-MISC	O
Output Y:	0	3	4	0	0	0	0	0	0	7	0

Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition](https://aclanthology.org/W03-0419/) (Tjong Kim Sang & De Meulder, CoNLL 2003)

Natural Language Processing Tasks

Question Answering

- Stanford Question Answering Dataset (SQuAD)
- **Context**
- **Question** – **Answer** pairs
- The answer is a span in a given Wikipedia paragraph

Context

The first aircraft, a South African Beechcraft King Air 200, landed at the new airport on 15 September 2015, prior to conducting a series of flights to calibrate the airport's radio navigation equipment..

Question Answer Pairs

1. **Question:** What kind of aircraft was the first to land at the new airport?
   Answer: South African Beechcraft King Air 200
2. **Question:** What date did the aircraft land at the new airport?
   Answer: 15 September 2015
3. **Question:** What are they calibrating via test flights at the airport?
   Answer: the airport's radio navigation equipment

SQuAD: 100,000+ Questions for Machine Comprehension of Text](https://aclanthology.org/D16-1264/) (Rajpurkar et al., EMNLP 2016)

Natural Language Processing Tasks

Textual Entailment

Stanford Natural Language Inference (SNLI)

Determining whether a "**hypothesis**" is true, given a "**premise**".

Five different annotator – **Gold label**

Contradiction - the hypothesis cannot be true given the premise.

Neutral - the hypothesis might be true or false; it's not entailed or contradicted.

Entailment - the hypothesis must be true given the premise.

Text	Judgements	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction C C C C C	The man is sleeping
An older and younger man smiling.	neutral N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	contradiction C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	entailment E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	neutral N N E C N	A happy woman in a fairy costume holds an umbrella.

A large annotated corpus for learning natural language inference](https://aclanthology.org/D15-1075/) (Bowman et al., EMNLP 2015)

Natural Language Processing Tasks

Machine Translation

- The name speaks for itself
- BUT, **input – output** pairs in **source – target** language.
- For one input there might be **multiple candidate** translations

English	French
It's cold.	C'est froid.
Stay calm.	Reste tranquille.
Stop that.	Arrêtez ça !
I will submit the assignment tonight	Je rendrai le devoir ce soir.

Europarl: A Parallel Corpus for Statistical Machine Translation](https://aclanthology.org/2005.mtsummit-papers.11/) (Koehn, MTSummit 2005)

Generative Pre-trained Transformer (GPT)

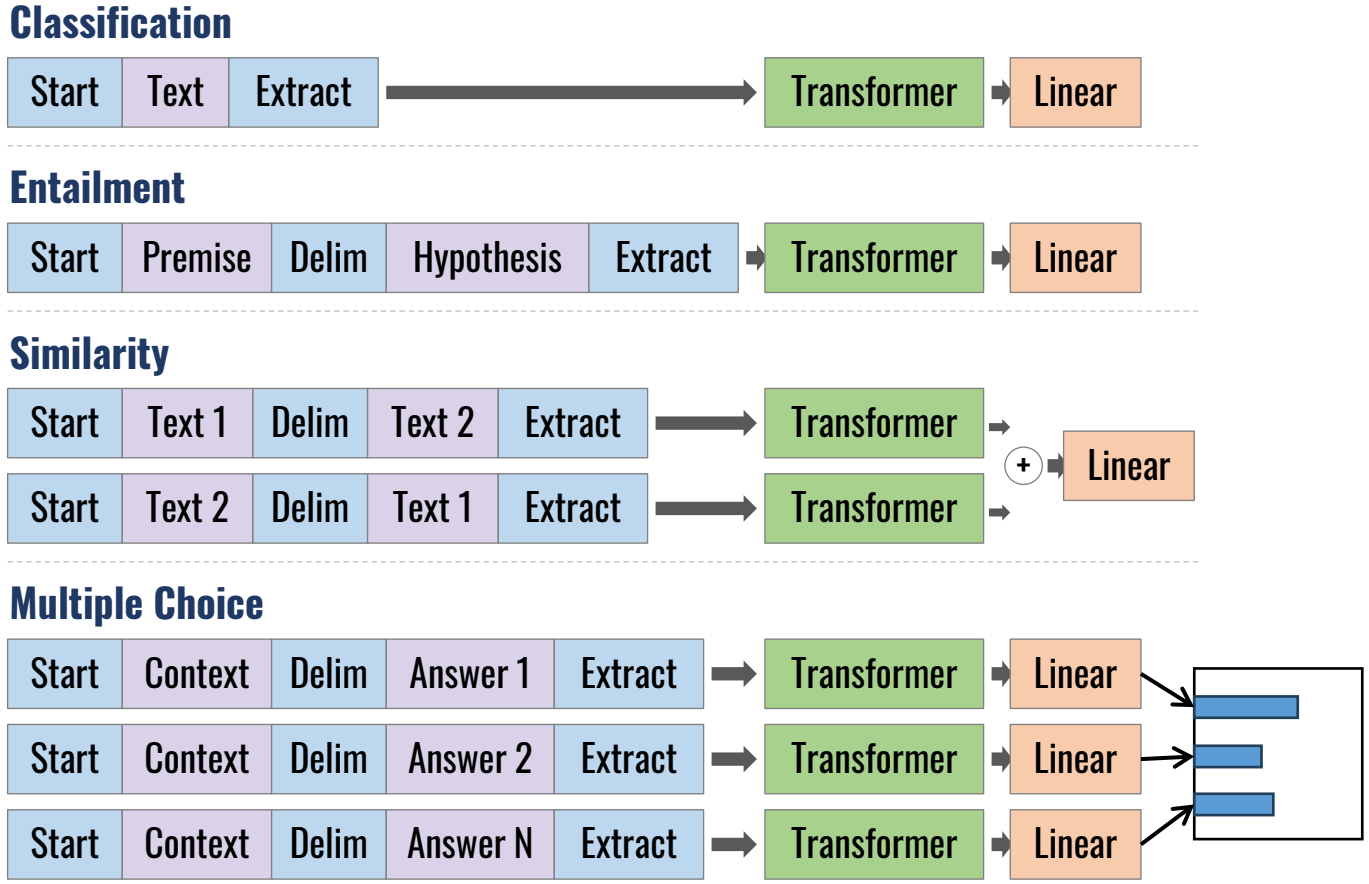
The original goal is Language Modelling (LM)

Uses Masked Self-Attention to limit the attention to the previous tokens only (left-to-right)

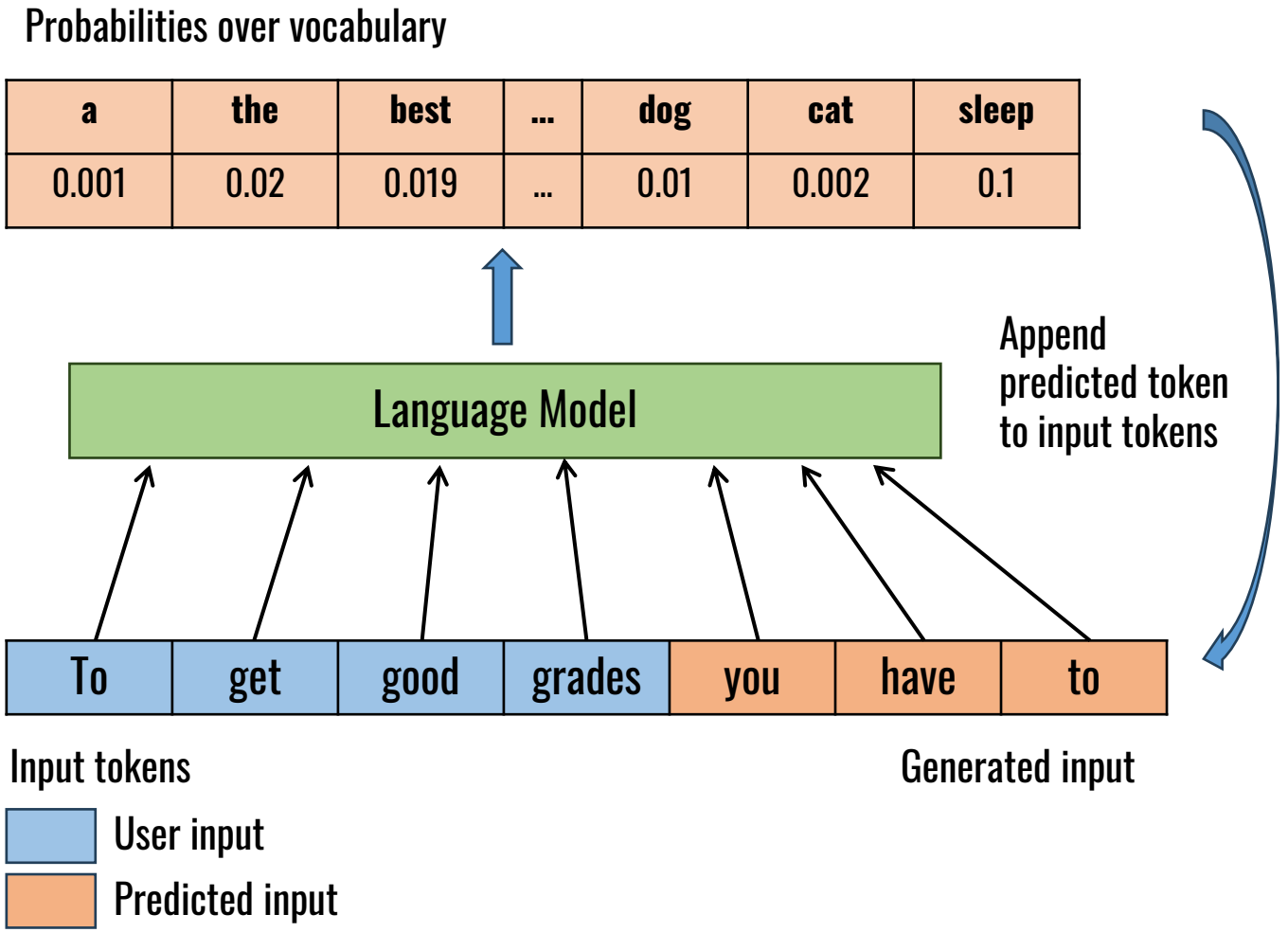
Two stage training:

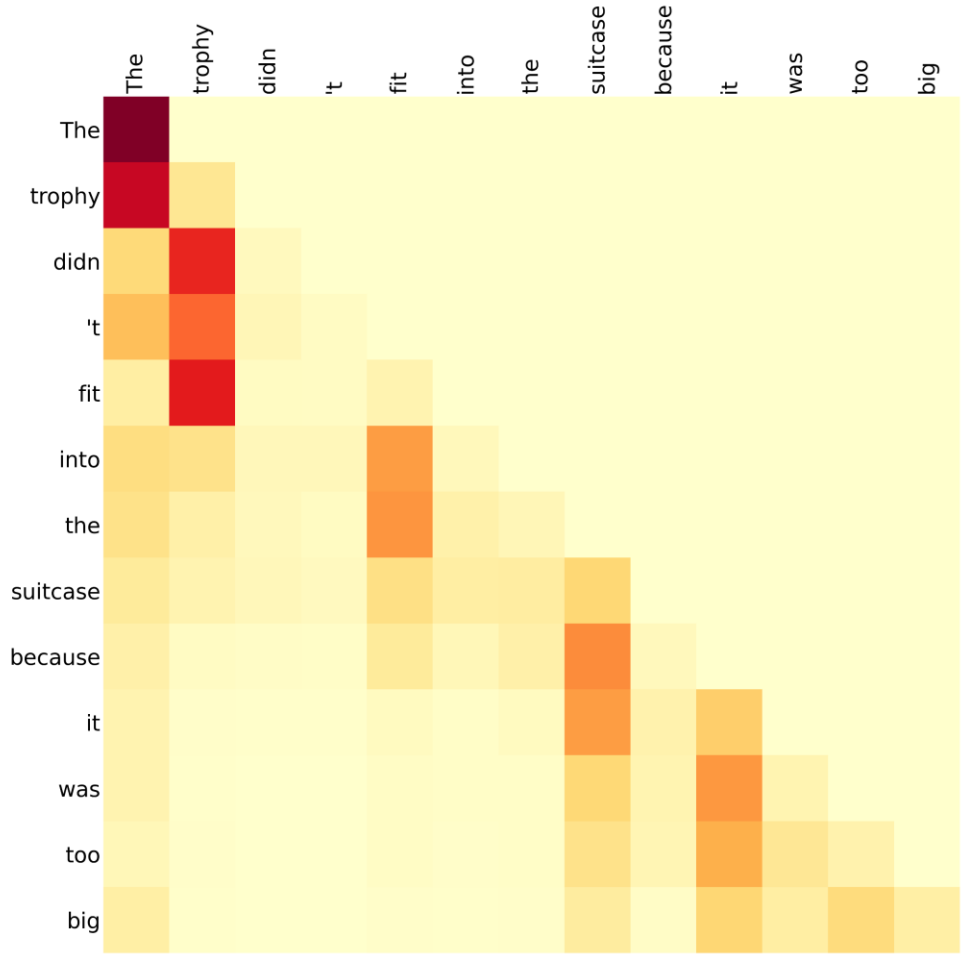
1. Unsupervised pre-training:
   - The goal is to predict the next token based on the previous tokens.

2. Supervised fine-tuning:
   - Predict the label (y) based on the input tokens



Generative Pre-trained Transformer (GPT)

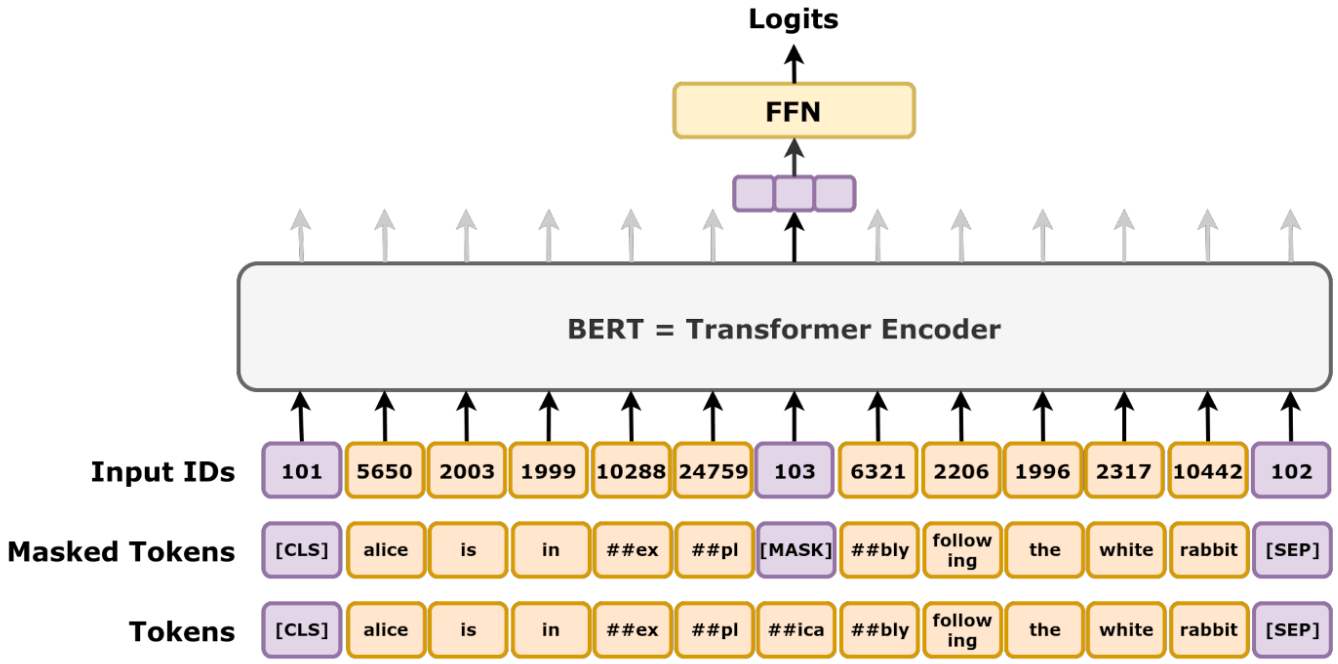




Bidirectional Encoder Representations from Transformers (BERT)

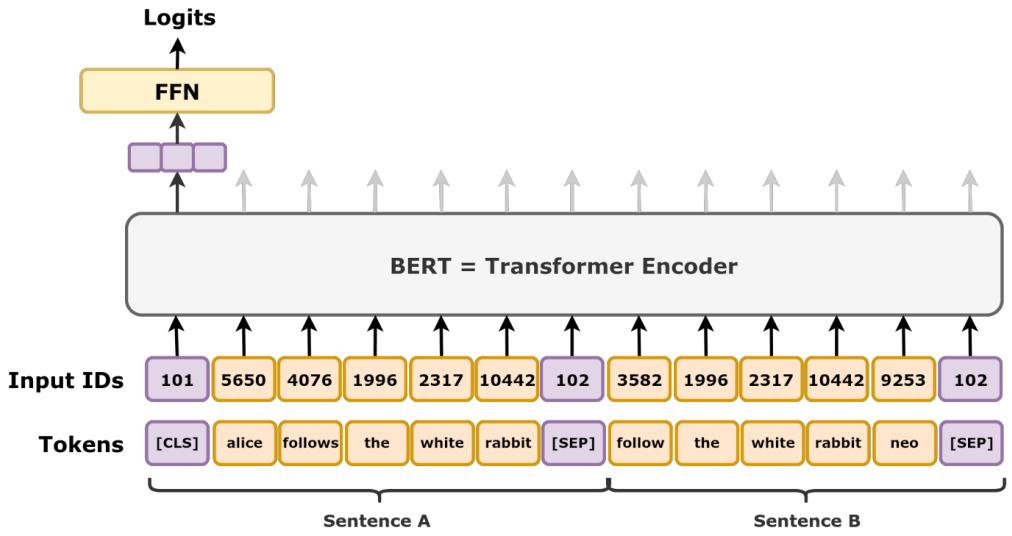
- Compared to OpenAI GPT it uses a bidirectional self-attention
- Trained on 2 tasks at the same time during pre-training
  1. Masked LM (15% of the tokens are masked)
  2. Next Sentence Prediction

- A special [CLS] token is introduced at the beginning of each sequence.

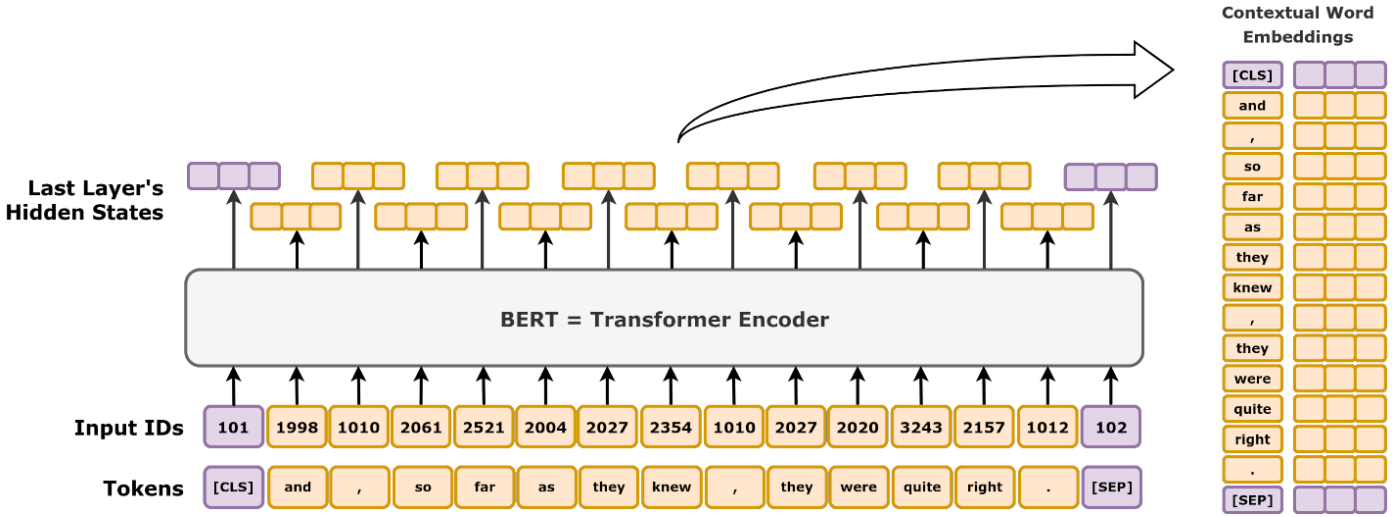


Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional Transformers for language understanding. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1810.04805
Image Source: BERT masked language modelling task.png, Daniel Voigt Godoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Bidirectional Encoder Representations from Transformers (BERT)



Next sentence prediction task. Notice the separator between the sentences

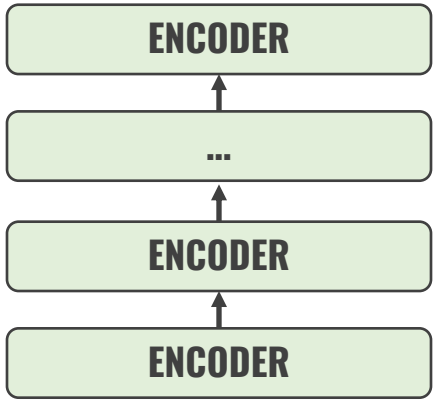


Natural Language Understanding task.

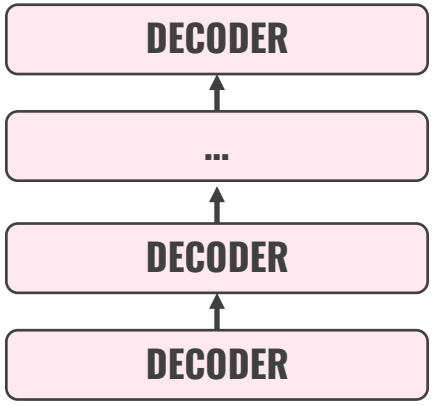
Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional Transformers for language understanding. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1810.04805
Image Source: BERT next sequence prediction task.png, Daniel Voigt Godoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
Image Source: BERT embeddings 01.png, Daniel Voigt Godoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

Transformer Architectures

BERT (encoder only)



GPT 2 (decoder only)



Deep Network Development

Lecture 7

Transformer Networks, Vision Transformers

1 Natural Language Understanding

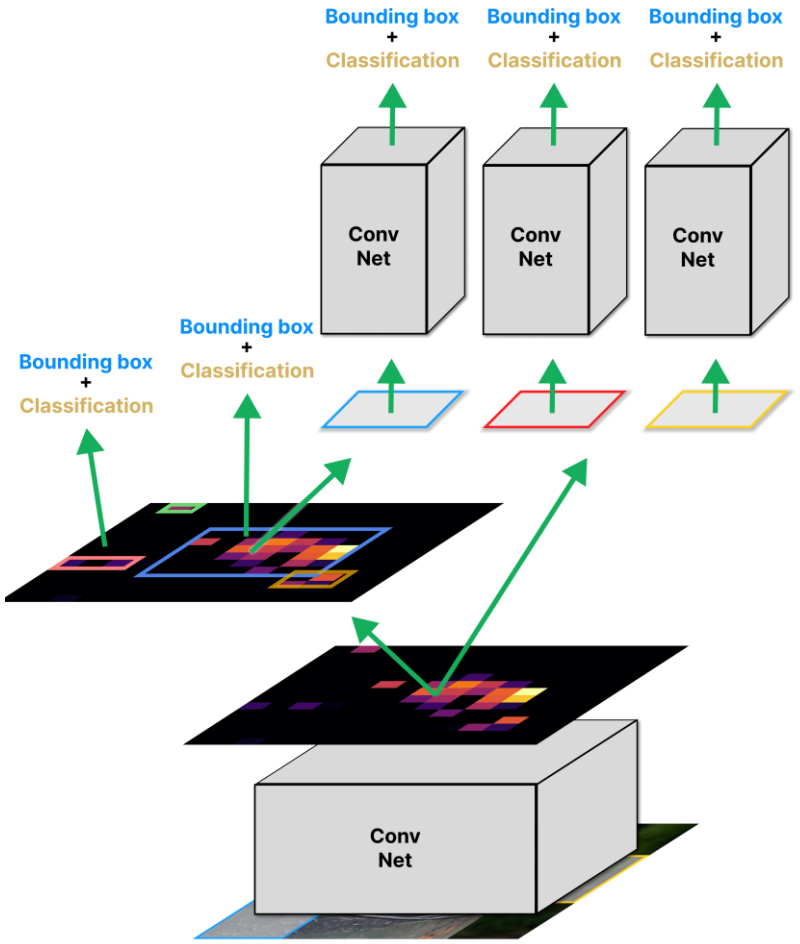
2 Attention Mechanism

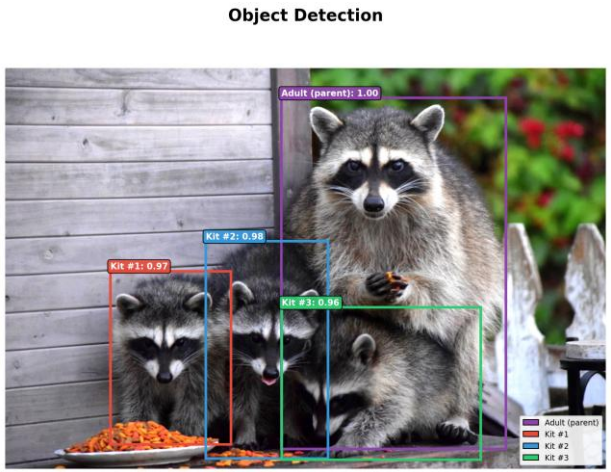
3 Transformer Architecture

4 Vision Transformers

Object detection

Object Detection: Faster R-CNN [3]







Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1506.01497
Image Source: Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Instance Segmentation

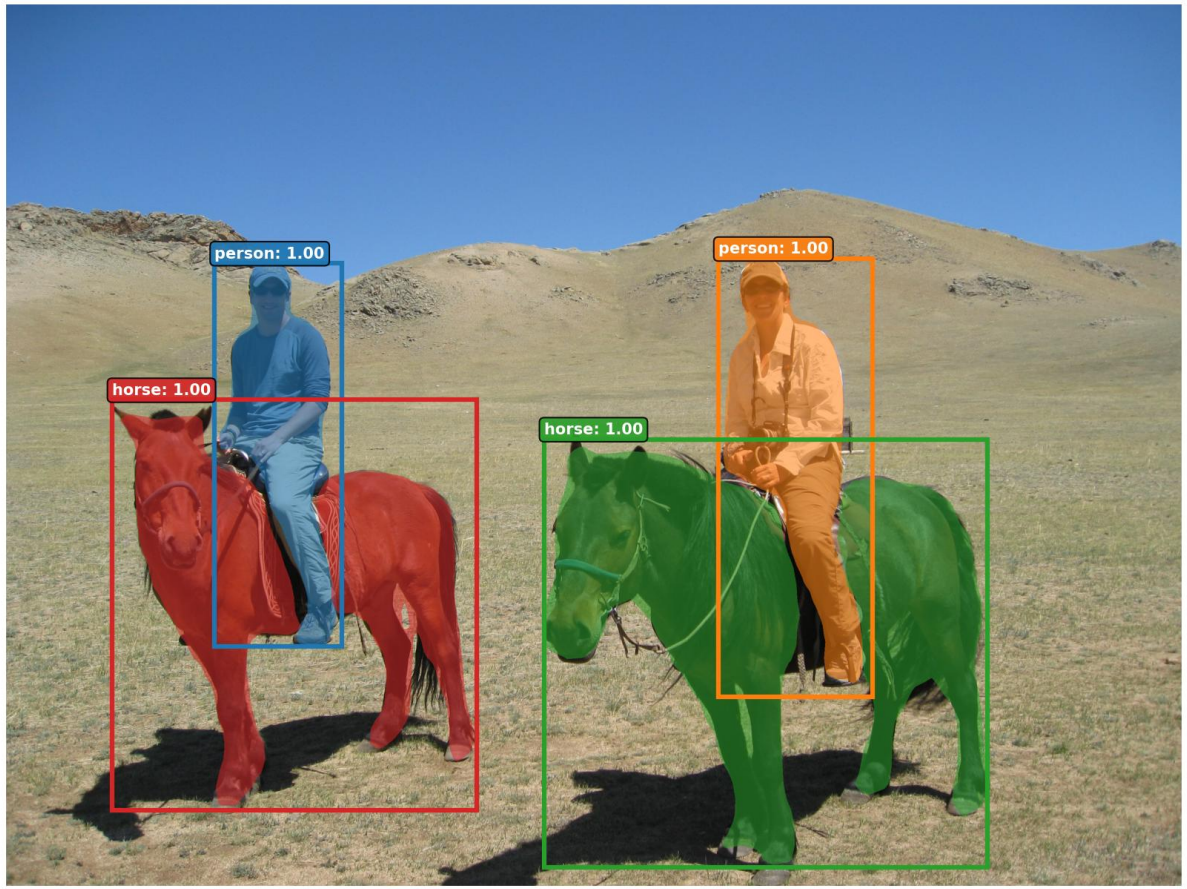
Things and Stuff

- **Things:** Object categories that can be separated into object instances (e.g. cats, cars, person)
- **Stuff:** Object categories that cannot be separated into instances (sky, grass, water, trees)

- Label all pixels in the image (both things and stuff).
- For "thing" categories also separate into instances.

Instance Segmentation:
- Detect all objects in the image and identify the pixels that belong to each object (Only things!)

Approach:
- Perform object detection, then predict a segmentation mask for each object!

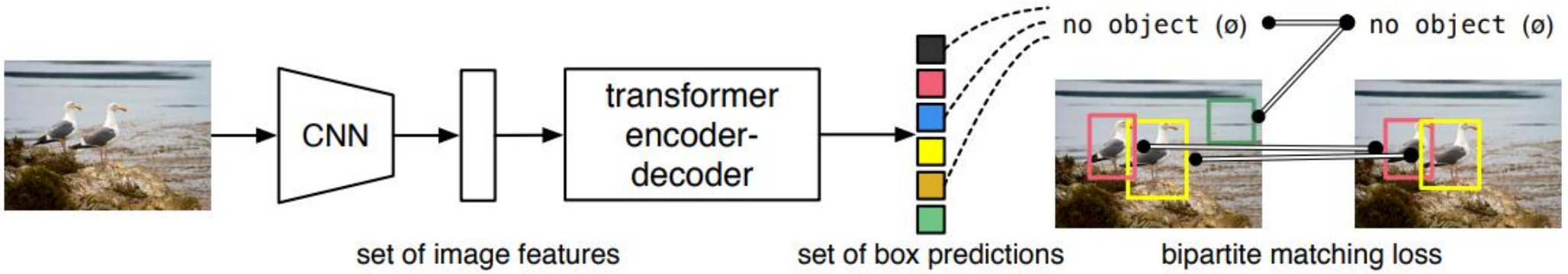


Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic Segmentation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1801.00868
Image Source: Horse Ride.jpg, Chris Feser, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

DEtection TRansformer (DETR)

Simple architecture:

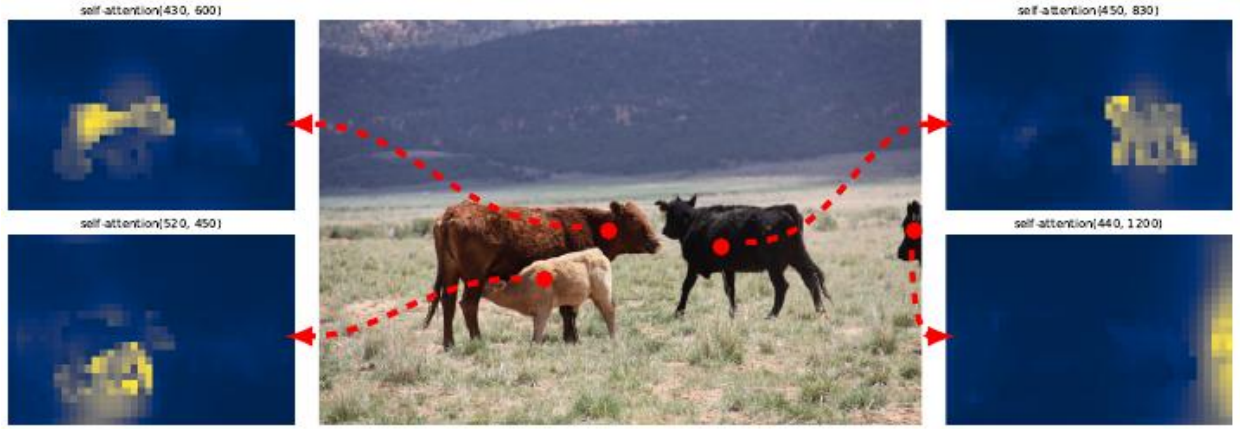
1. CNN backbone
2. Encoder-Decoder Transformer
3. Feed Forward Network

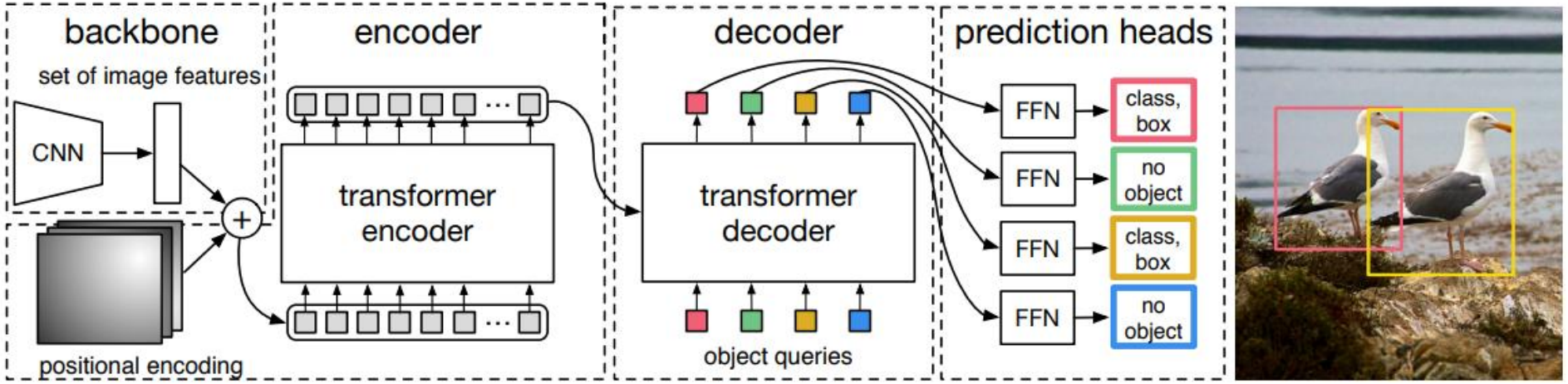


Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2005.12872
Image Source: Fig. 1: DETR architecture - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 https://creativecommons.org/publicdomain/zero/1.0/, https://arxiv.org/abs/2005.12872

DEtection TRansformer (DETR)

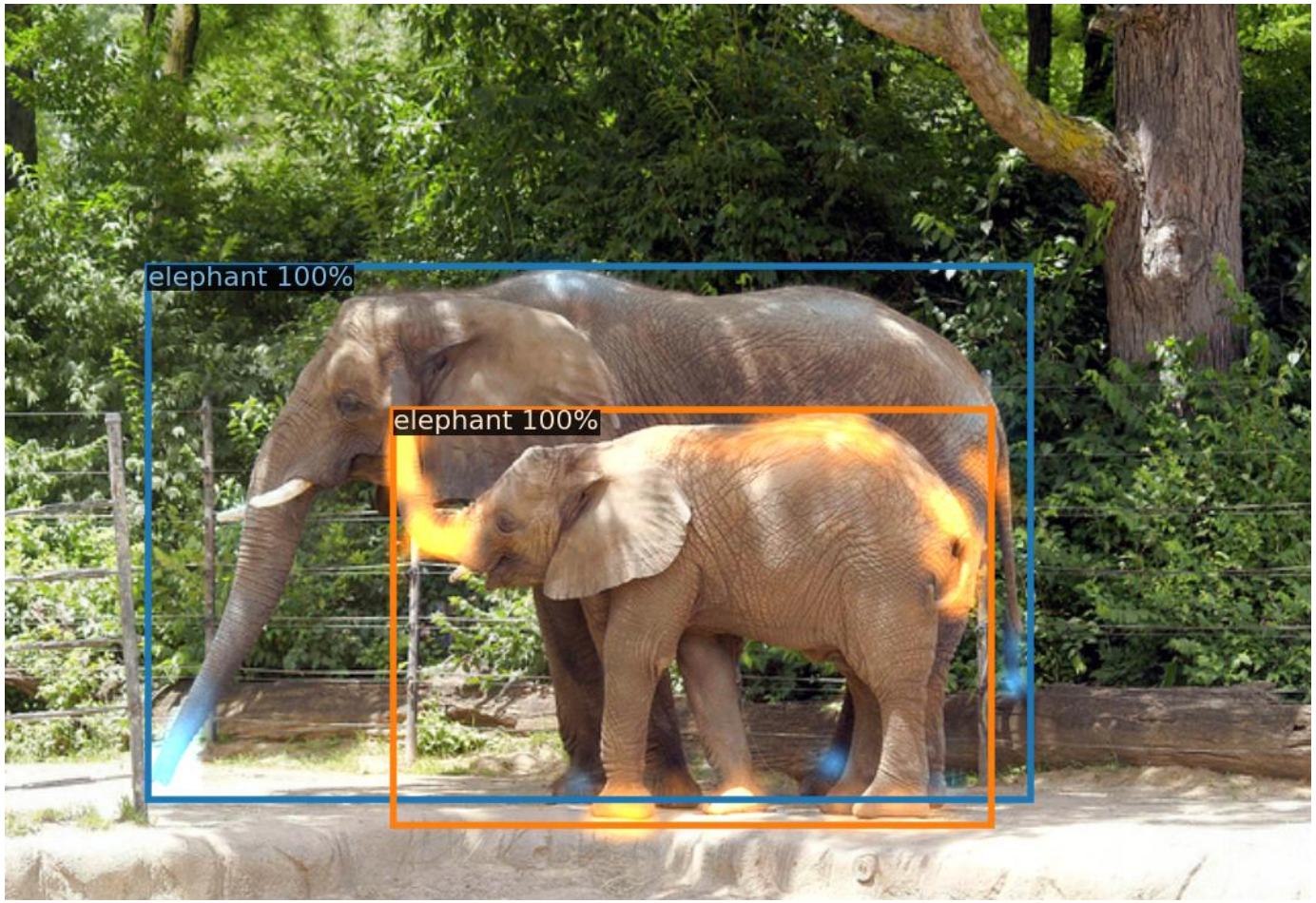
1. Does N object detection parallelly based on learned object queries – (contrary to Region Proposals)
2. Uses a bipartite matching loss to remove duplicate bounding boxes – (contrary to Non-Maximum Suppression)





Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2005.12872
Image Source: Fig. 2: DETR architecture - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 https://creativecommons.org/publicdomain/zero/1.0/, https://arxiv.org/abs/2005.12872
Image Source: Fig. 3: Encoder self-attention visualization - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 https://creativecommons.org/publicdomain/zero/1.0/, https://arxiv.org/abs/2005.12872

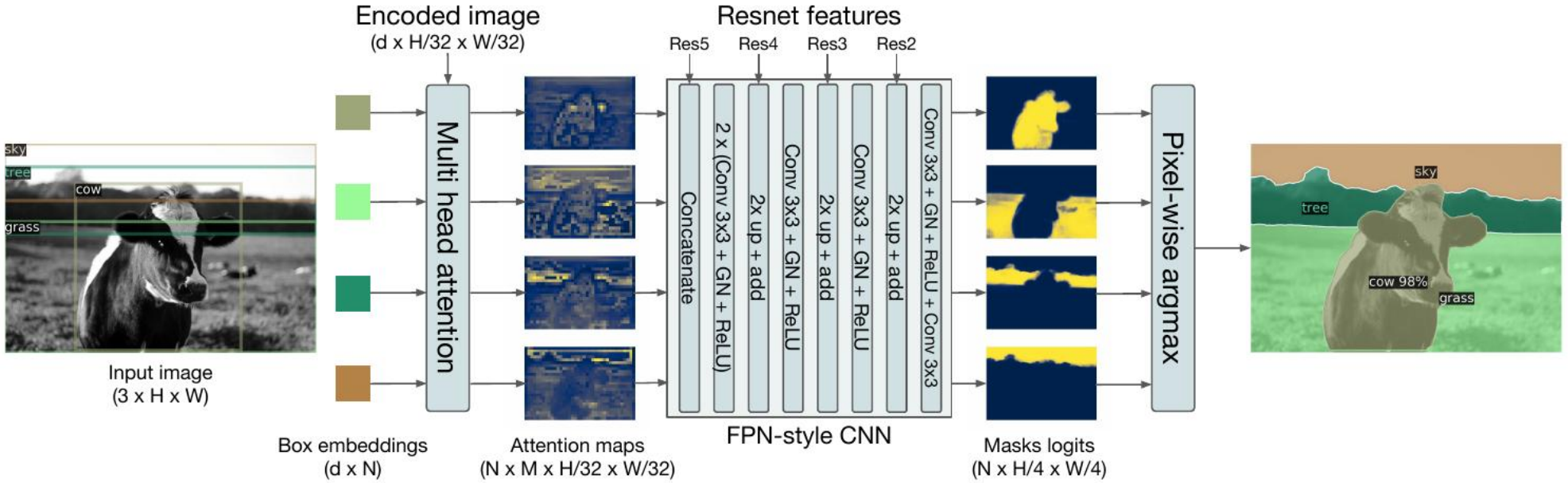
DEtection TRansformer (DETR)



Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2005.12872
Image Source: Fig. 6: Decoder attention visualization - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 https://creativecommons.org/publicdomain/zero/1.0/, https://arxiv.org/abs/2005.12872

DEtection TRansformer (DETR)

- DETR can be naturally extended by adding a mask head on top of the decoder outputs.
- Then it will generate the masks parallel to each predicted object



Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2005.12872
Image Source: Fig. 8: Illustration of the panoptic head - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 https://creativecommons.org/publicdomain/zero/1.0/, https://arxiv.org/abs/2005.12872

Vision Transformers

- In vision, attention is either applied in conjunction with convolutional networks or used to replace certain components of convolutional networks while keeping their overall structure in place.
- Reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks

- Transformers operate on a sequence of tokens
- How do we transform an image into tokens?

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2010.11929

Image Tokenization

1. Reshape images of $x \in \mathbb{R}^{\mathrm{H} \times W \times \mathrm{C}}$ into $N = \frac{HW}{P^2}$ patches
   - (H, W) – image resolution
   - C – number of channels
   - (P, P) – patch resolution

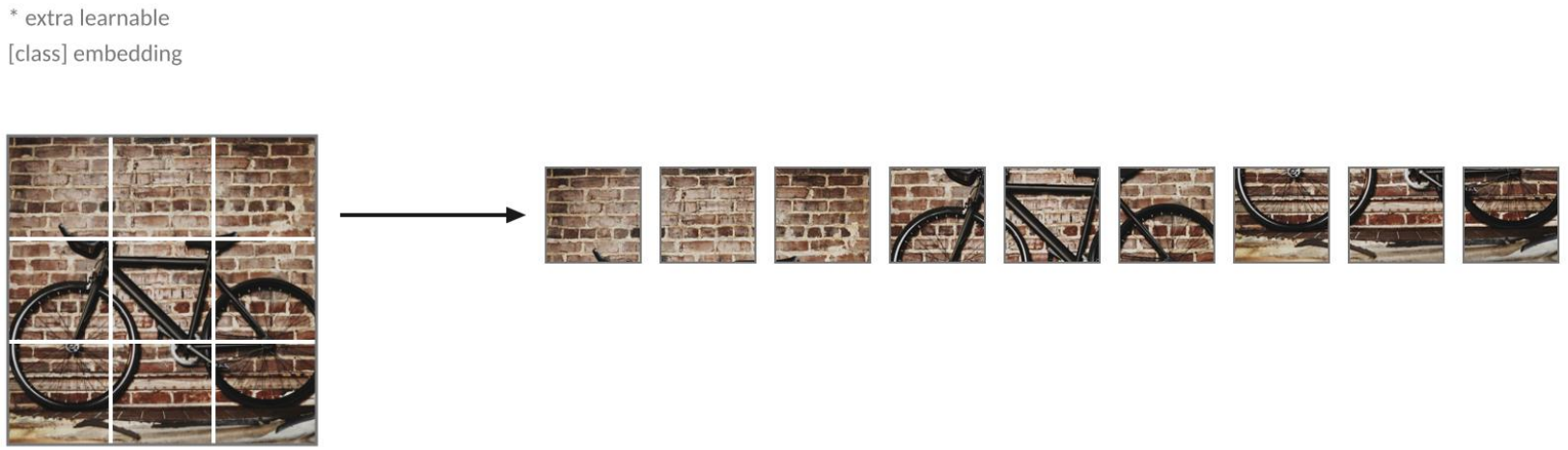


Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CCO, via Wikimedia Commons

Image Tokenization

2. The Transformer uses constant latent vector size D through all of its layers
    - Flatten all the patches and apply a learnable linear projection (*patch embeddings*)

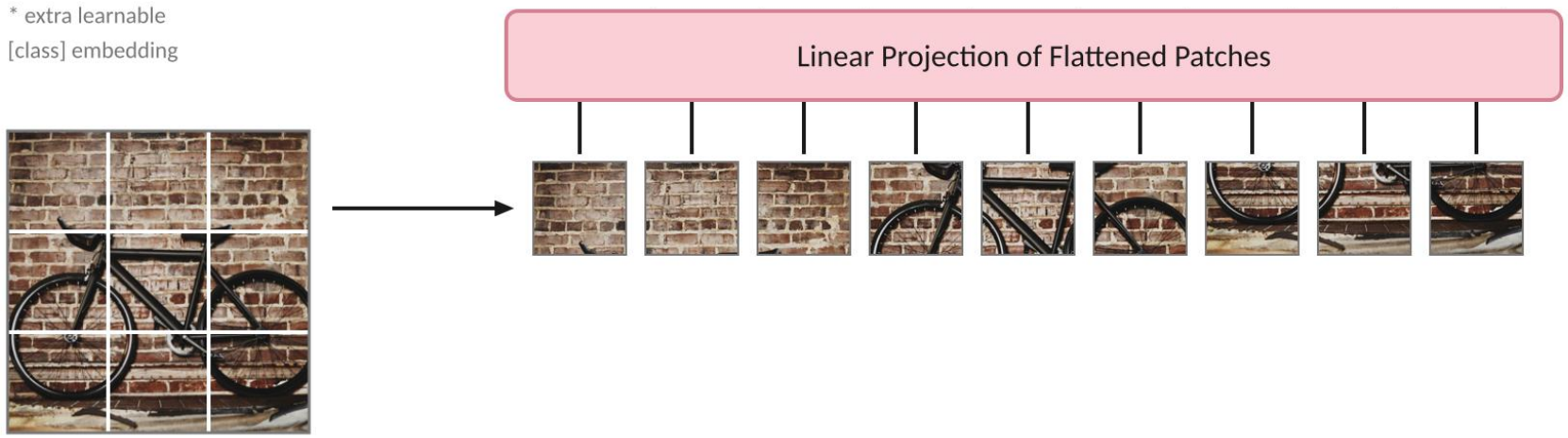


Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CCO, via Wikimedia Commons

Token Processing

3. Processing the tokens
    - Prepend a learnable embedding to the patch embeddings
    - Apply position embedding

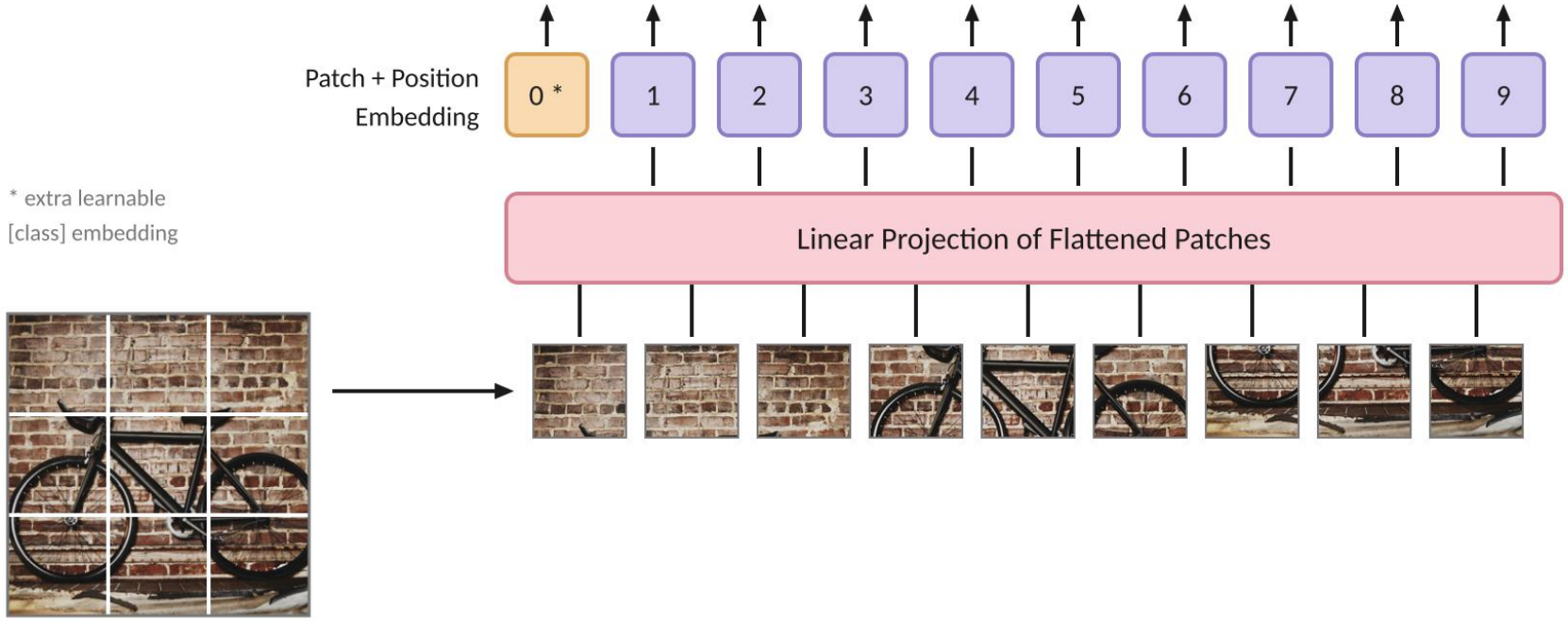


Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CCO, via Wikimedia Commons

Encoder Block

4. Feeding the embedded patches to the Transformer Encoder
   - Input the sequence of embedded patches ($[\boldsymbol{z_0^0}, z_0^1, \ldots, z_0^N]$)
   - At the end we get the image representation ($[\boldsymbol{z_L^0}, z_L^1, \ldots, z_L^N]$)

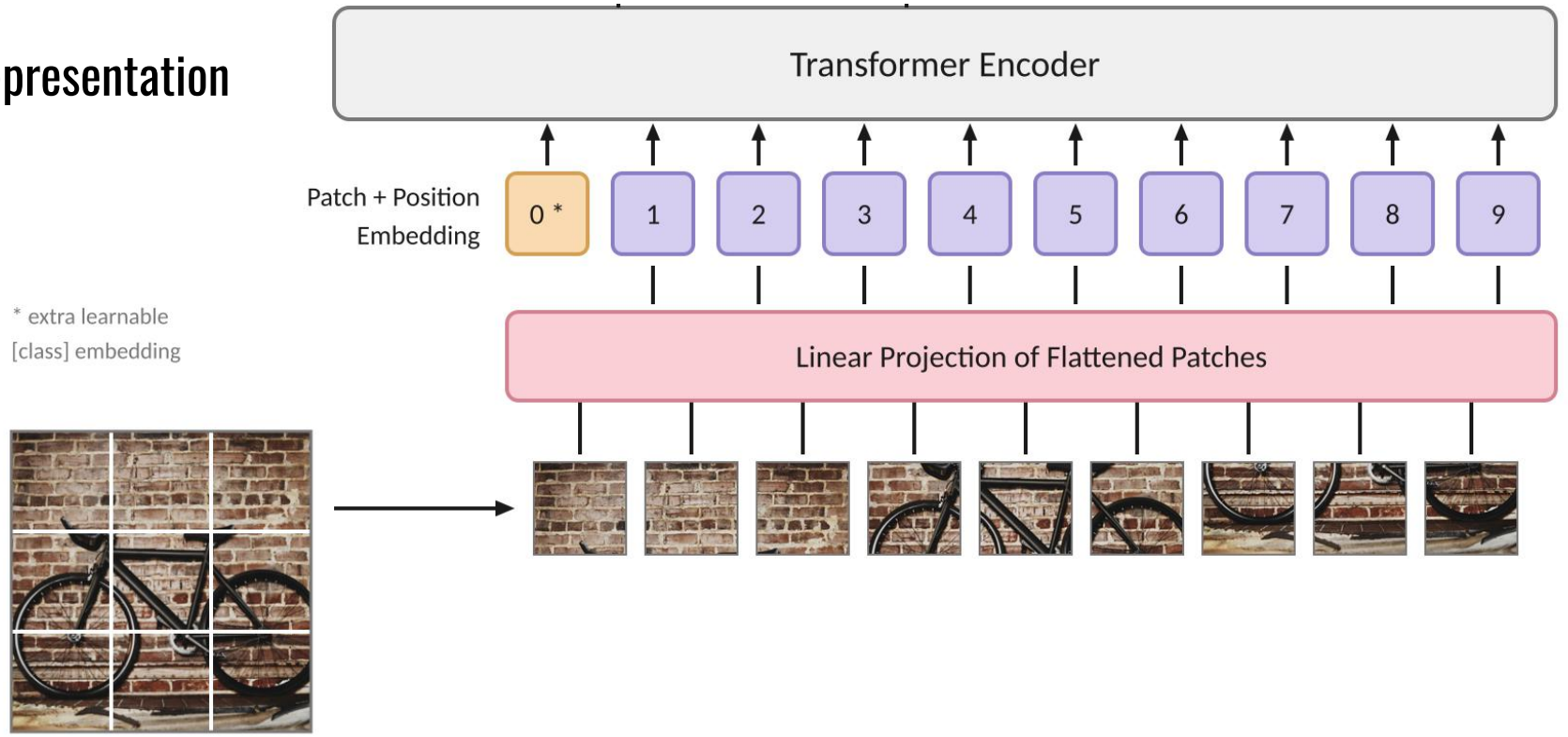


Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CCO, via Wikimedia Commons

Prediction Processing

5. Classification MLP head
   - Attaching a classifier head to z_L^0

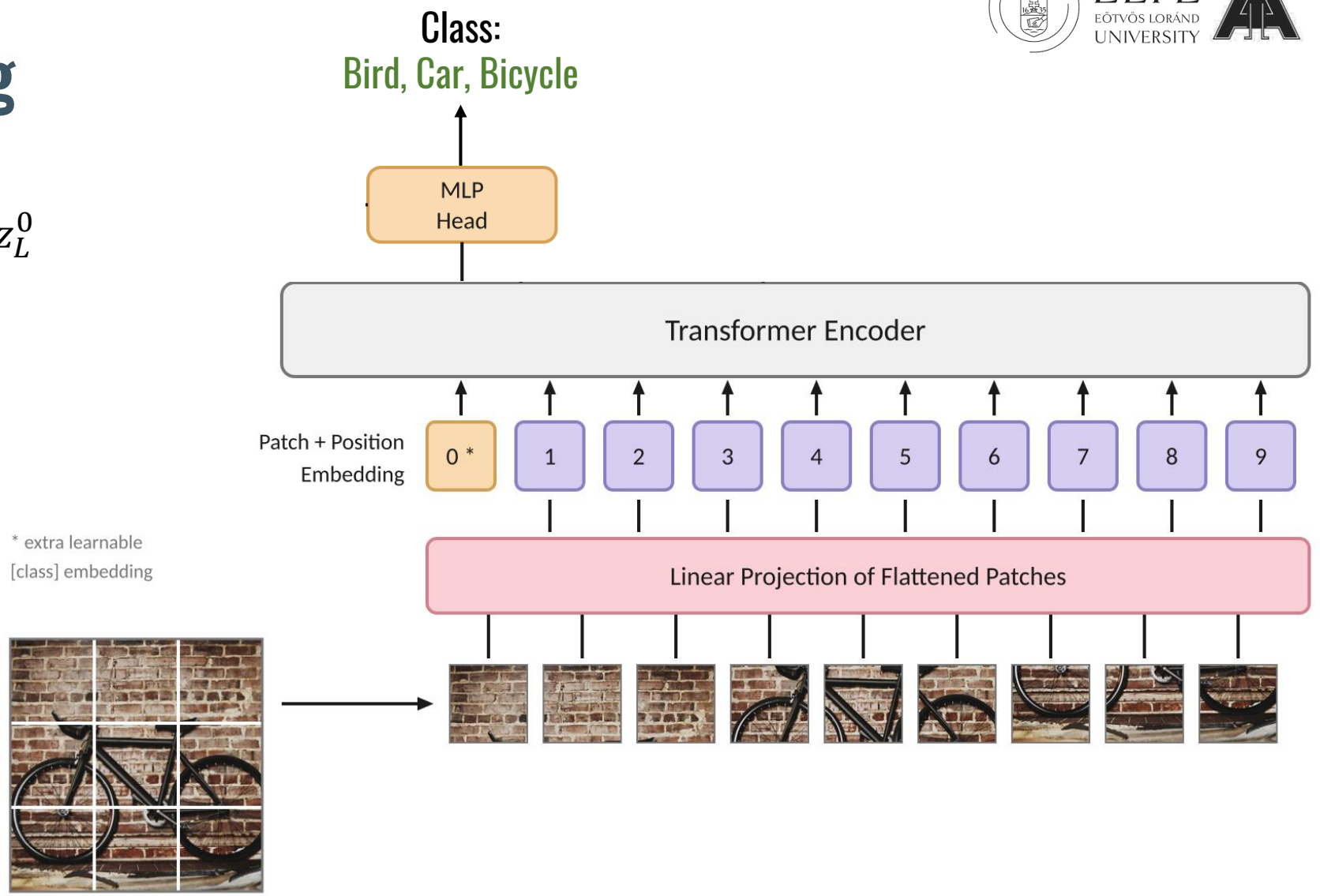


Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CCO, via Wikimedia Commons

Comparison with Convolutional Networks

- ViT has much less image-specific inductive bias
  - features like **edges**, **textures**, and **patterns** are spatially localized and translationally invariant
- In CNNs, locality, two-dimensional neighbourhood structure, and translation equivariance are baked into the whole model
- Position embedding does not carry information about the 2D position of the patches



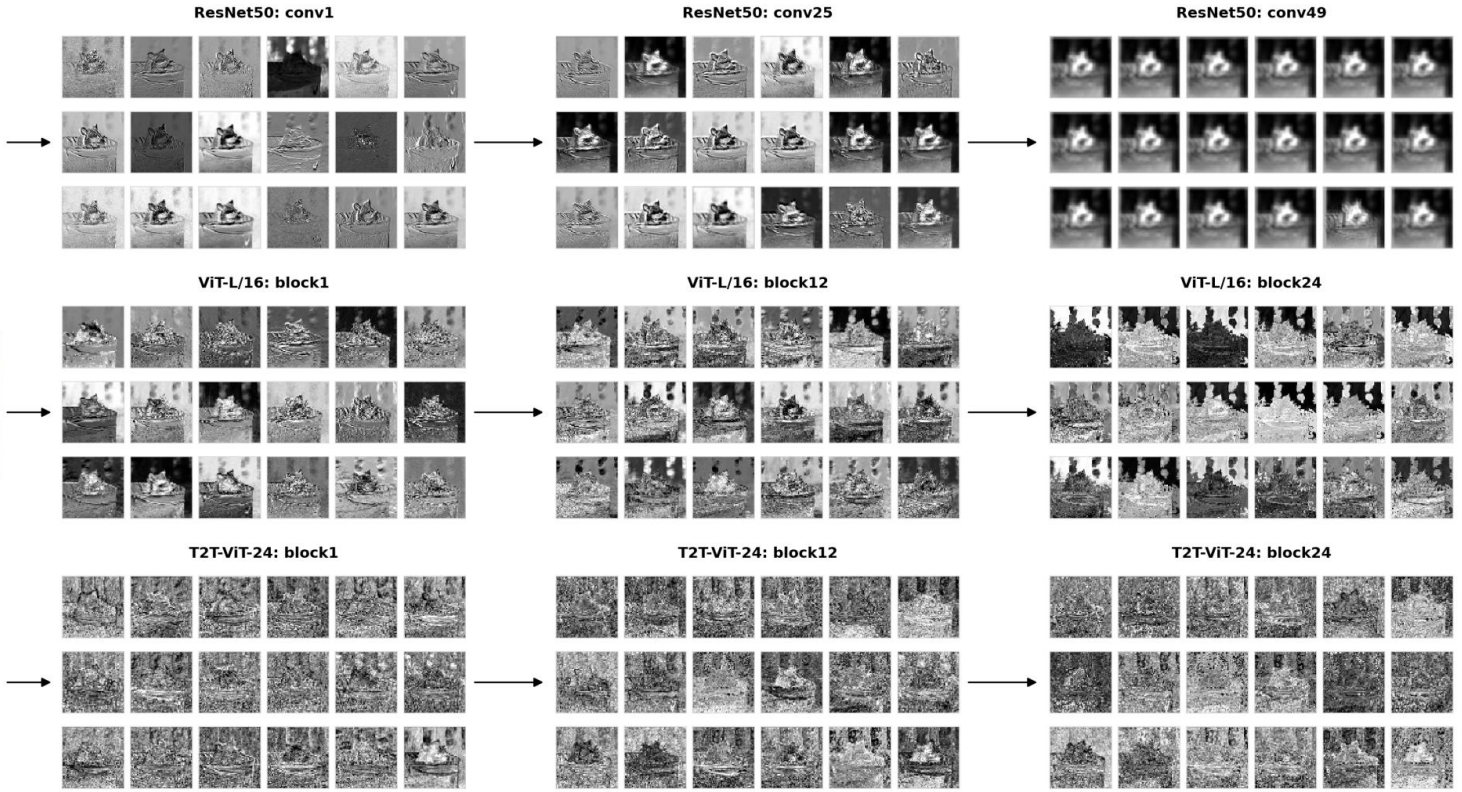
Image Source: A basket of fruits.jpg, Suyash Dwivedi, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Aeroflot Airbus A330 Kustov.jpg, Sergey Kustov, CC BY-SA 3.0 < https://creativecommons.org/licenses/by-sa/3.0/ >, via Wikimedia Commons

Tokens-to-Token ViT (T2T-ViT)

ViT achieves inferior performance to CNNs when trained on a midsize dataset

1. The tokenization fails to model the important local structure such as edges, lines, etc.
2. The redundant attention backbone leads to limited feature richness





Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F. E. H., Feng, J., & Yan, S. (2021). Tokens-to-Token ViT: Training vision transformers from scratch on ImageNet. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2101.11986
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Token-to-Token ViT (T2T-ViT)

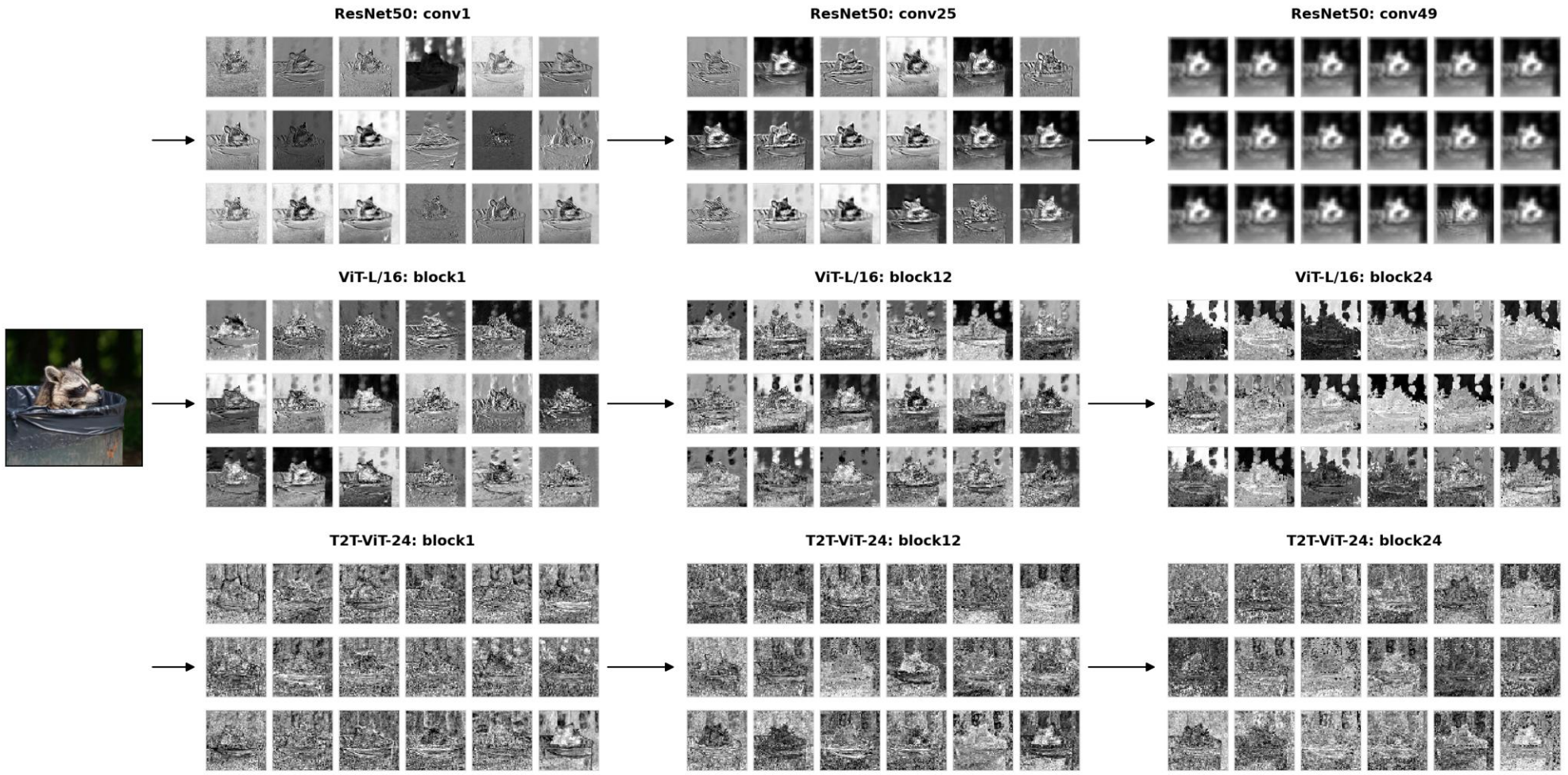


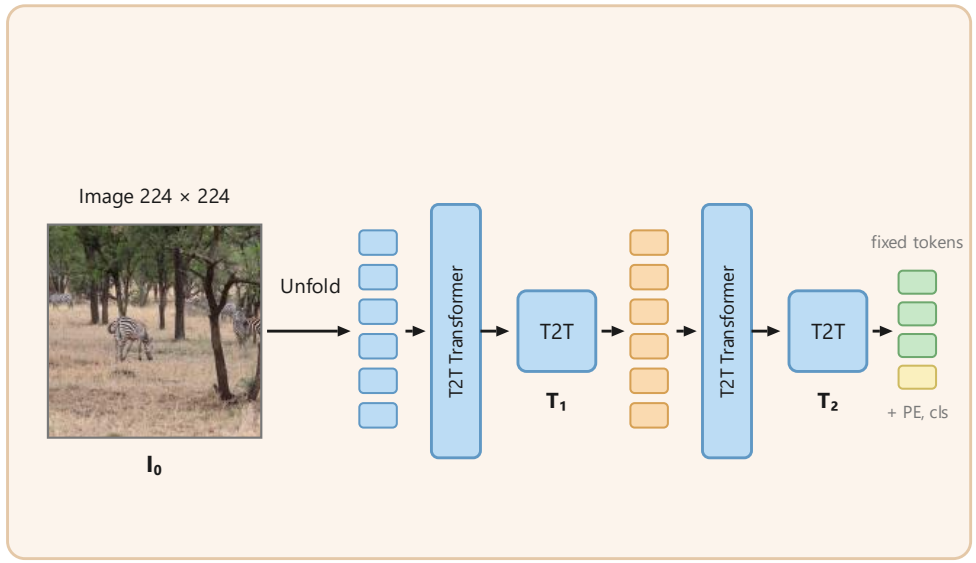
Image Source: Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Token-to-Token ViT (T2T-ViT)

1. A layer-wise “Token-to-token module”
2. An efficient “T2T-ViT backbone”

- The generated tokens are reordered like an “image”
- Then areas closer together are grouped together into a new token

Tokens-to-Token module



One T2T step

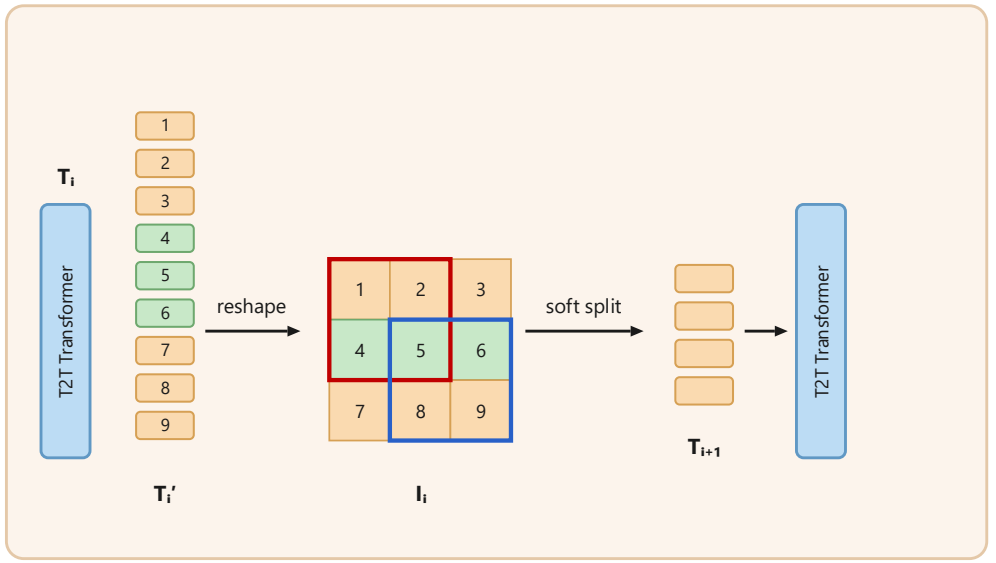


Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CCO, via Wikimedia Commons

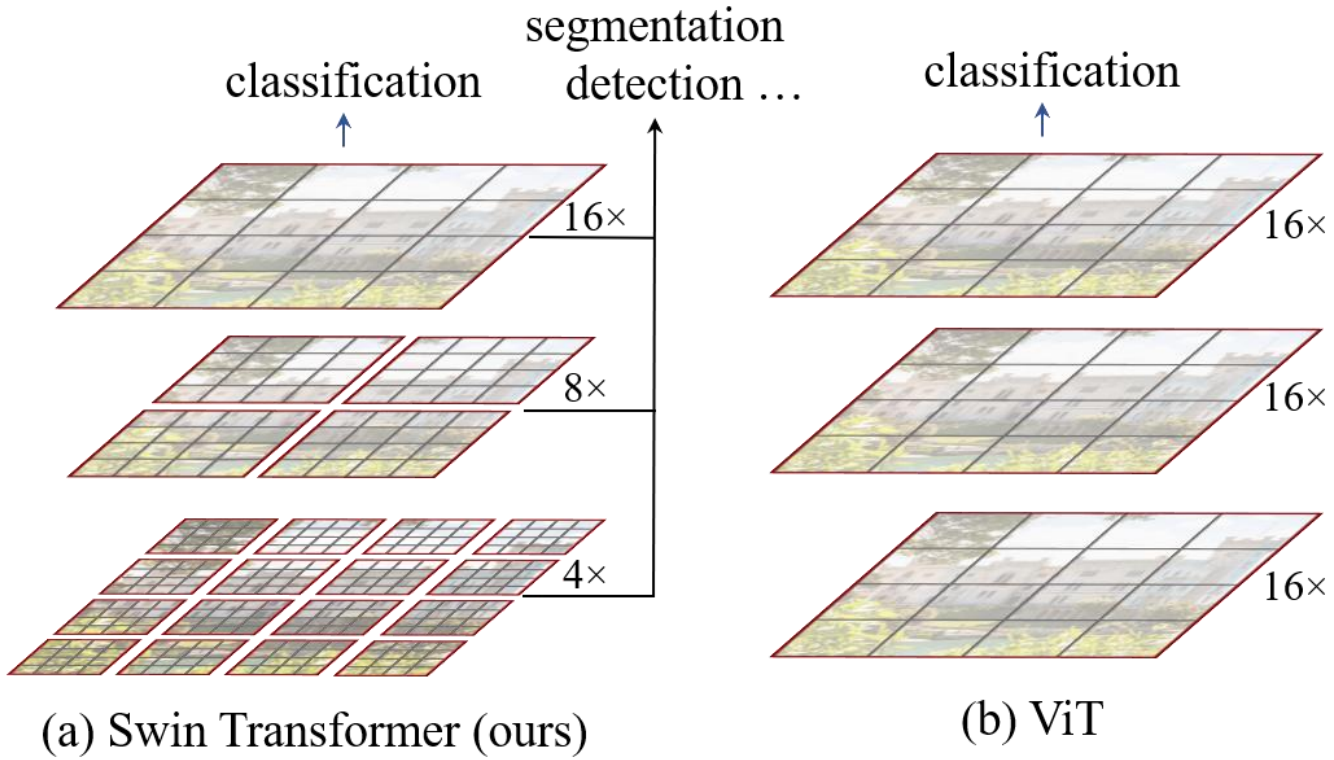
Token-to-Token ViT (T2T-ViT)

- While “vanilla” ViT requires a large dataset and more tuneable parameters to beat the “state-of-the-art” (**JFT-300M**) CNN models
- T2T-ViT requires smaller datasets and less tuneable parameter

Swin Transformer

- ViT - computational complexity is quadratic to image size.
  - Conduct global self attention, where the relationships between a token and all other tokens are computed

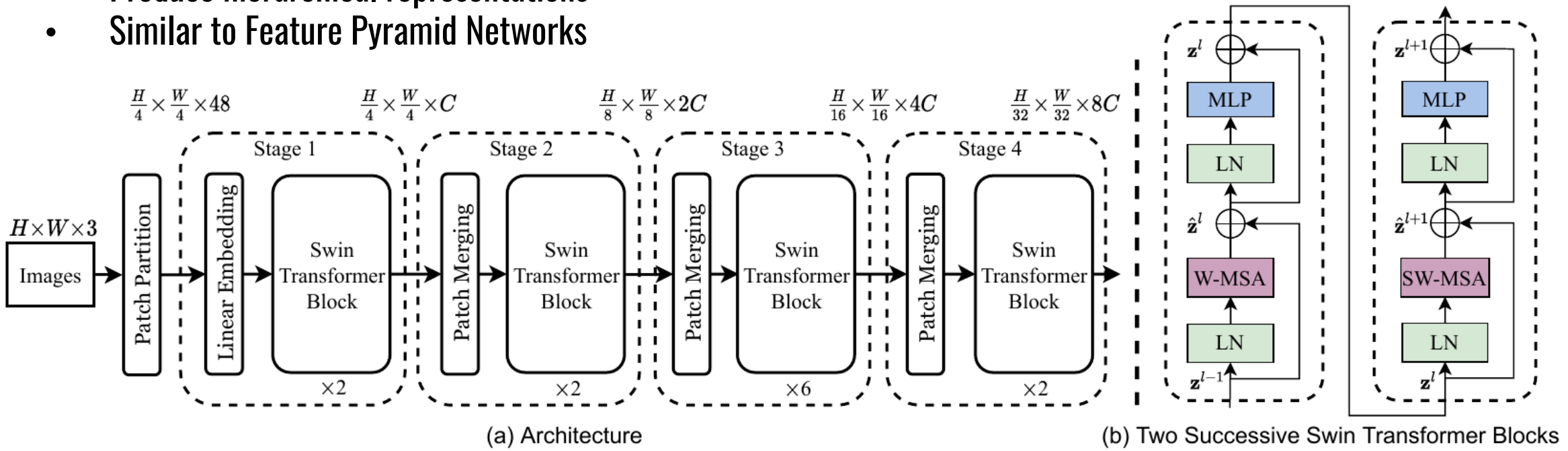
- Calculate self-attention within local windows
- Patch merging and scaling as the network gets deeper
- Shifted window approach with fixed number of patches.



Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical vision Transformer using shifted windows. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2103.14030
Image Source: Figure 1. Hierarchical feature map visualization - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2103.14030

Swin Transformer

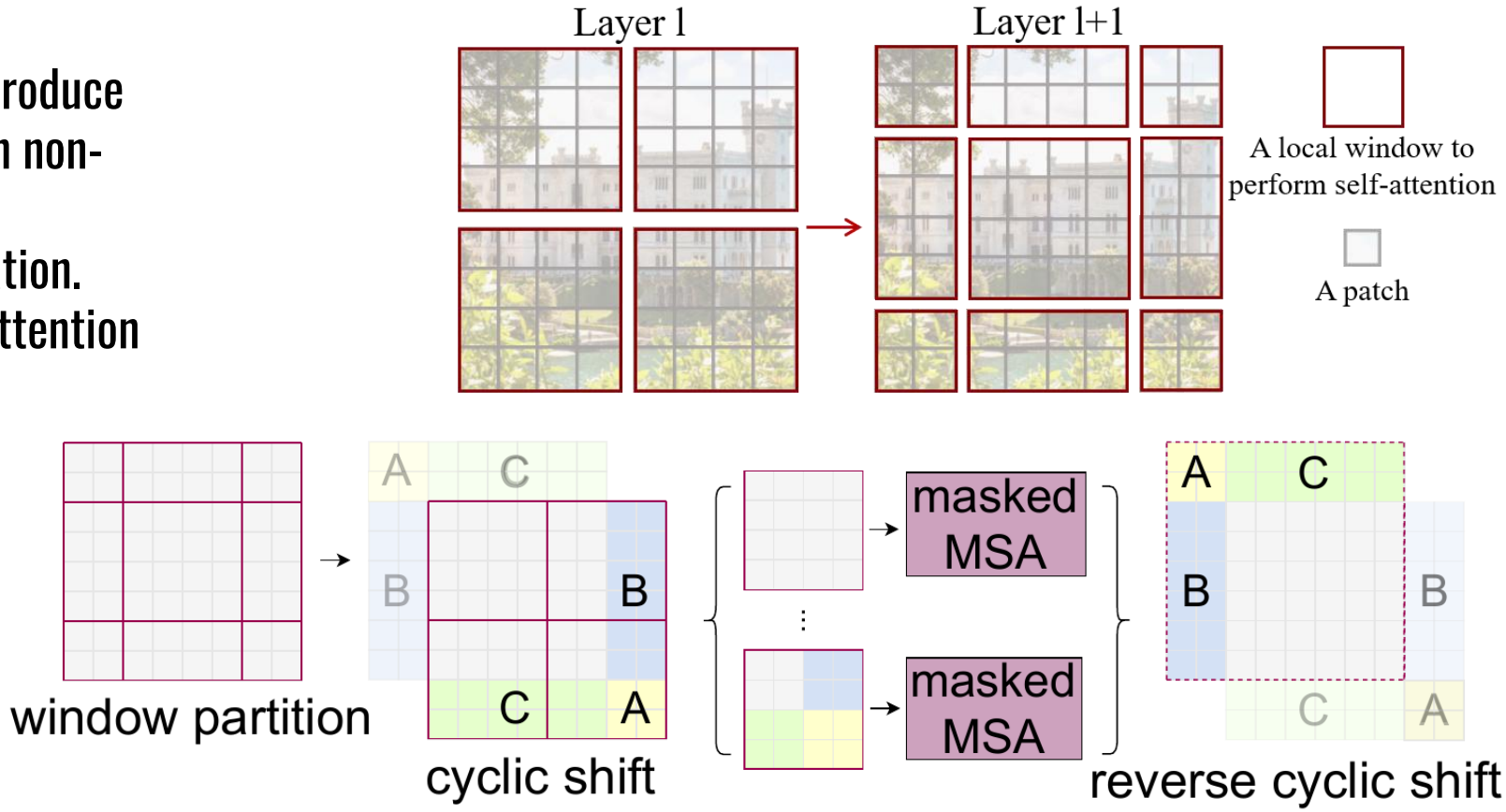
- The number of tokens reduced in the patch merging layer
  - Produce hierarchical representations
  - Similar to Feature Pyramid Networks



Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical vision Transformer using shifted windows. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2103.14030
Image Source: Figure 3. The architecture of a Swin Transformer - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2103.14030

Swin Transformer

- Shifted window partitioning to introduce cross-window connection between non-overlapping windows.
- Alternates between two configuration.
- Masking mechanism - limit self-attention non-adjacent windows



Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical vision Transformer using shifted windows. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2103.14030
Image Source: Figure 2. An illustration of the shifted window approach - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2103.14030
Image Source: Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2103.14030

Further Links + Resources

- https://theaisummer.com/attention/
- https://theaisummer.com/transformer/
- https://jalammar.github.io/illustrated-transformer/
- https://nlp.seas.harvard.edu/annotated-transformer/
- https://www.youtube.com/watch?v=bCz4OMemCcA
- Coursera Deep Learning Specialization
- Beam search: https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24
- BLEU score: https://cloud.google.com/translate/automl/docs/evaluate#bleu

Resources

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning
- https://developers.google.com/machine-learning/crash-course/embeddings

Papers

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2005.12872
2. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2010.11929
3. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical vision Transformer using shifted windows. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2103.14030
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *arXiv [cs.CL]*. arXiv. http://arxiv.org/abs/1706.03762
5. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F. E. H., Feng, J., & Yan, S. (2021). Tokens-to-Token ViT: Training vision transformers from scratch on ImageNet. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2101.11986

Image Sources

- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.
- Dancing Robot: Gif robot.git, Meks01, CC BY-SA 4.0, via Wikimedia Commons
- DNA RNA structure (full).png, Thomas Shafee, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- Signal-speech-martin-de.png, MartinThoma, CC0, via Wikimedia Commons
- Video frames of the Parallel Bars action category.png, Bikingdog, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- T-SNE visualisation of word embeddings generated using 19th century literature.png, Siobhán Grayson, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Translate.png, Impartial just, CC0, via Wikimedia Commons
- The entire Seq2Seq chatbot architecture — Hosameldeen et al. (2023), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://www.preprints.org/manuscript/202303.0070
- Seq2seq-training.svg, Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- Vanishing Gradient — Andreas Madsen, Visualizing memorization in RNNs, Distill (2019), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://distill.pub/2019/memorization-in-rnns/

Image Sources

- Attention between RNNs — Chris Olah & Shan Carter, Attention and Augmented Recurrent Neural Networks, Distill (2016), derived from Fig. 3 of Bahdanau et al. (2014), CC BY 2.0 <https://creativecommons.org/licenses/by/2.0/>, https://distill.pub/2016/augmented-rnns
- Transformer, attention block diagram.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- Attention Is All You Need - long-distance attention, encoder self-attention in layer 5 of 6.png, Google, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Multiheaded attention, block diagram.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- Process of a Single Attention Head in a Transformer Model.jpg, Shuang Zhang, Rui Fan, Yuti Liu, Shuang Chen, Qiao Liu, Wanwen Zeng, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- Transformer, full architecture.png, dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- BERT masked language modelling task.png, Daniel Voigt Godoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- BERT next sequence prediction task.png, Daniel Voigt Godoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- BERT embeddings 01.png, Daniel Voigt Godoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

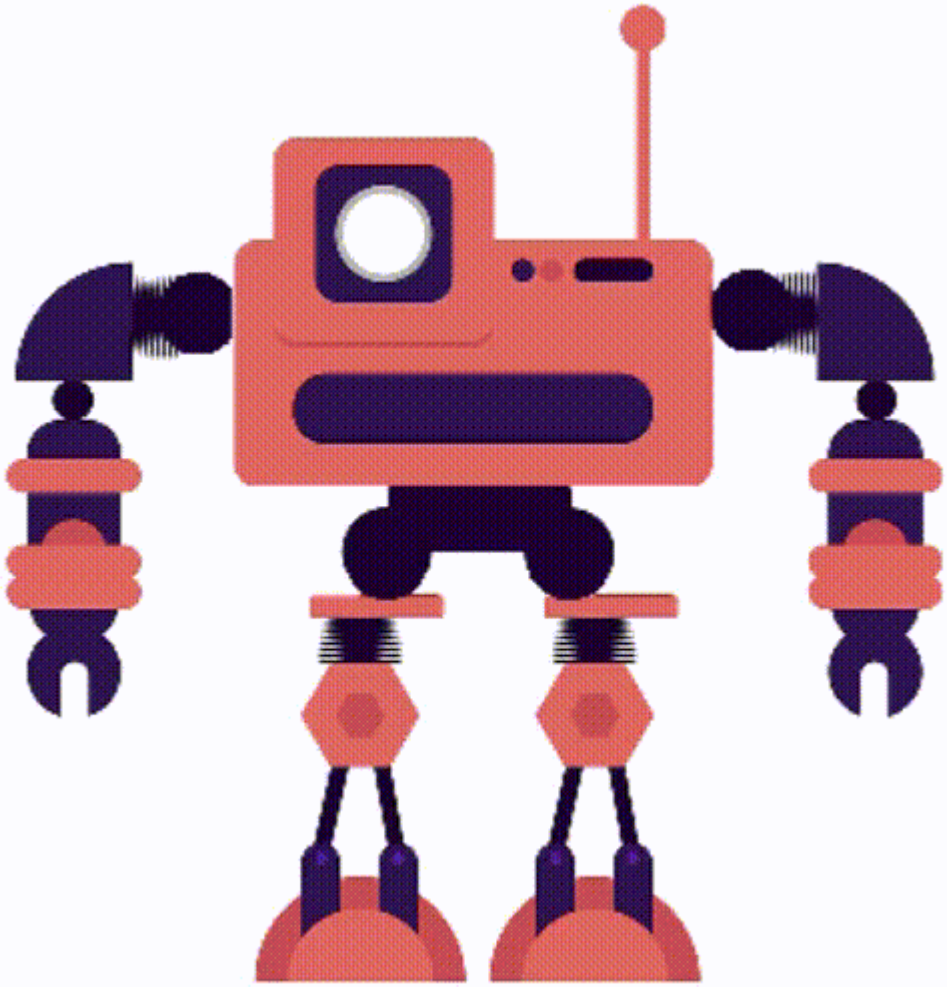
Image Sources

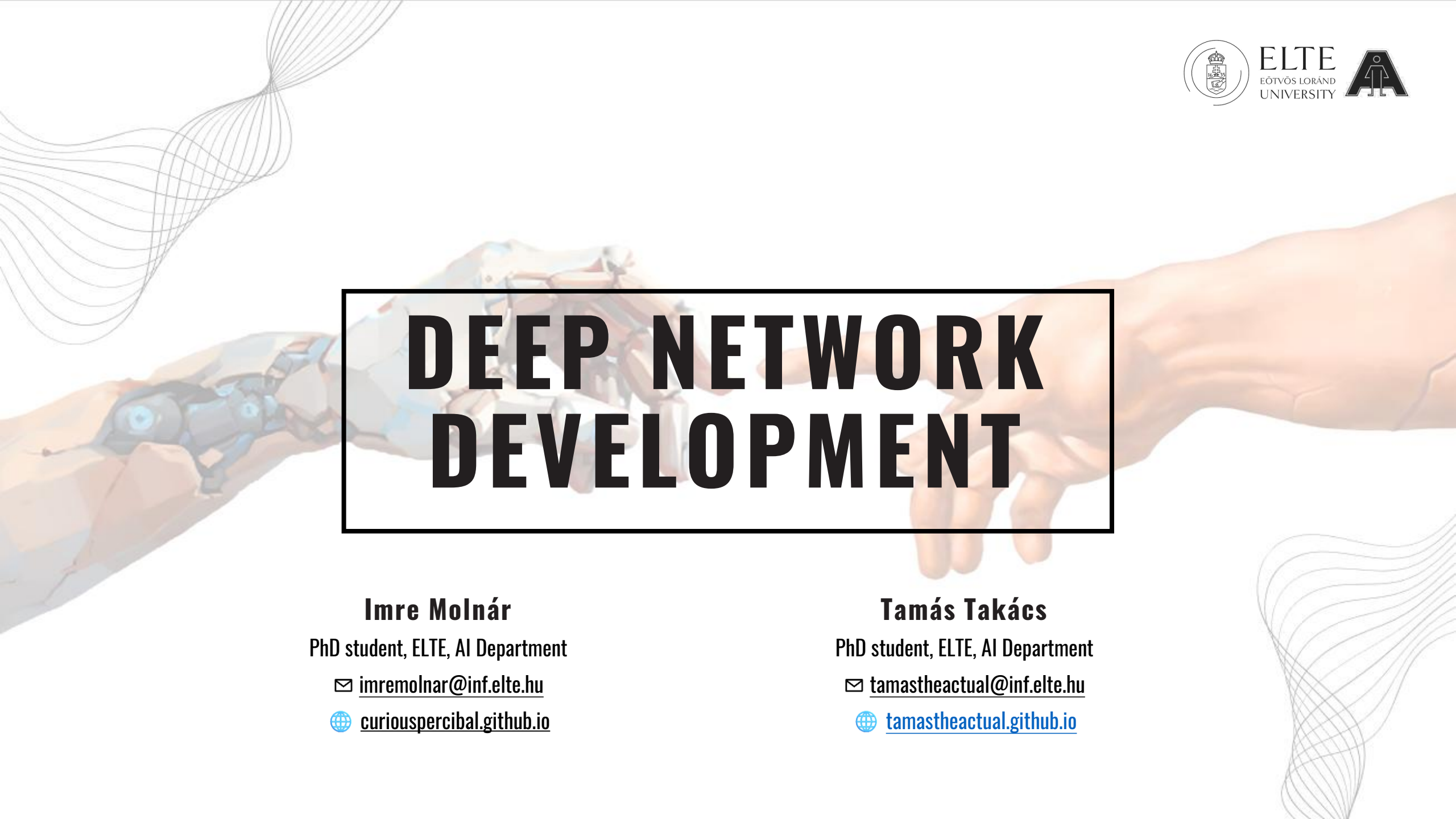
- Raccoon Family DSC 0037.jpg, Tracie Hall, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Horse Ride.jpg, Chris Feser, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons
- Fig. 1: DETR architecture - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 <https://creativecommons.org/publicdomain/zero/1.0/>, https://arxiv.org/abs/2005.12872
- Fig. 2: DETR architecture - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 <https://creativecommons.org/publicdomain/zero/1.0/>, https://arxiv.org/abs/2005.12872
- Fig. 3: Encoder self-attention visualization - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 <https://creativecommons.org/publicdomain/zero/1.0/>, https://arxiv.org/abs/2005.12872
- Fig. 6: Decoder attention visualization - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 <https://creativecommons.org/publicdomain/zero/1.0/>, https://arxiv.org/abs/2005.12872
- Fig. 8: Illustration of the panoptic head - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 <https://creativecommons.org/publicdomain/zero/1.0/>, https://arxiv.org/abs/2005.12872
- Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons
- A basket of fruits.jpg, Suyash Dwivedi, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Image Sources

- Cat and Dog Game.jpg, Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Aeroflot Airbus A330 Kustov.jpg, Sergey Kustov, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia Commons
- Raccoon in a garbage can in Prospect Park (72564).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CC0, via Wikimedia Commons
- Figure 1. Hierarchical feature map visualization - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://arxiv.org/abs/2103.14030
- Figure 2. An illustration of the shifted window approach - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://arxiv.org/abs/2103.14030
- Figure 3. The architecture of a Swin Transformer - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://arxiv.org/abs/2103.14030
- Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning - Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (2021), CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>, https://arxiv.org/abs/2103.14030

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 11

Deep Learning Tools For Computer Vision part 1

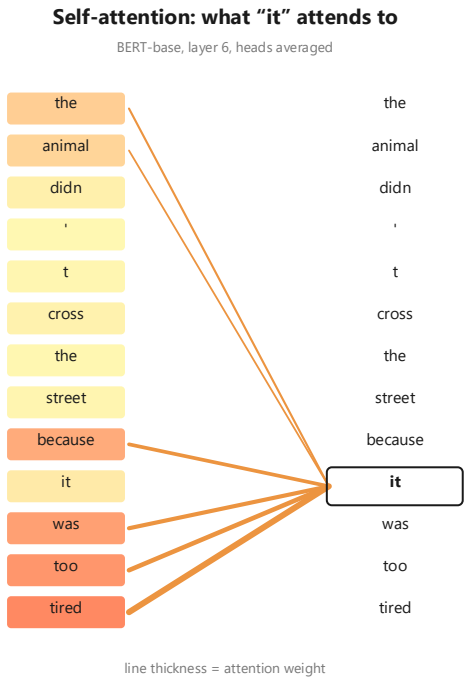
1 Annotation Tools

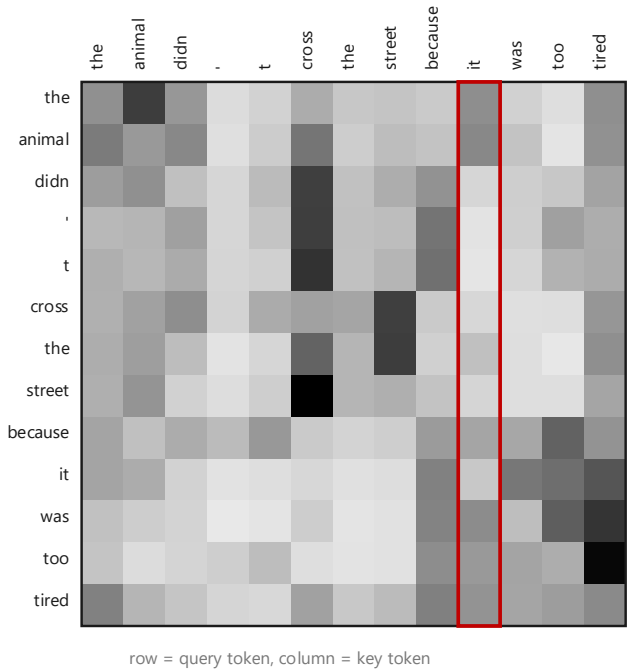
2 Depth Estimation

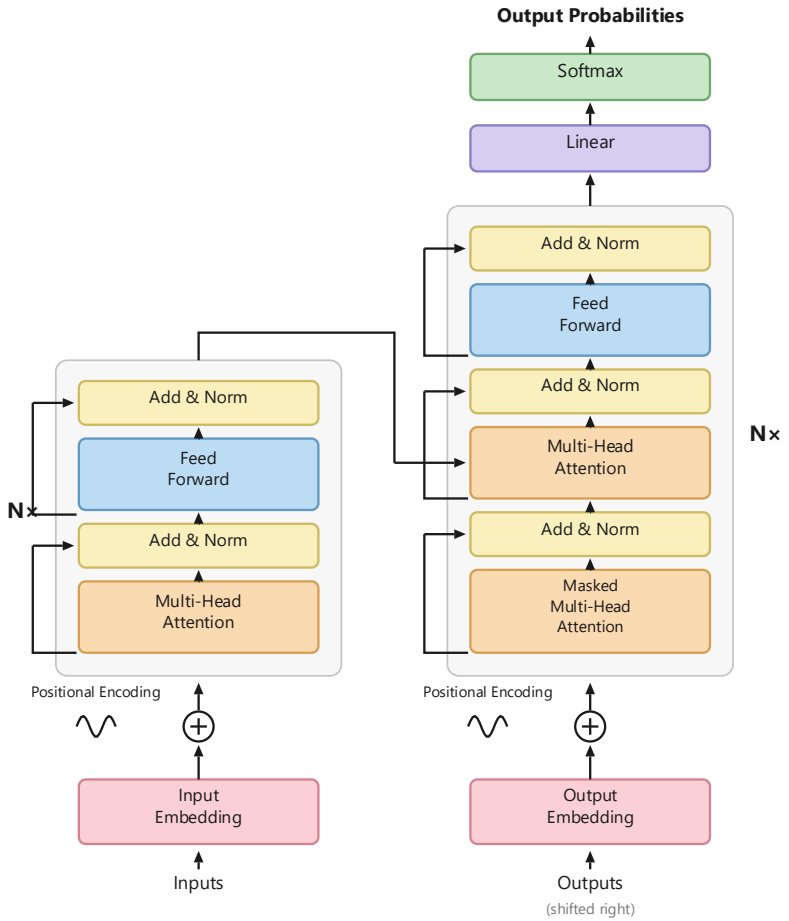
3 Optical Flow

Attention and Transformers

- Attention, Self-Attention, Multi-Head Attention
- Transformers

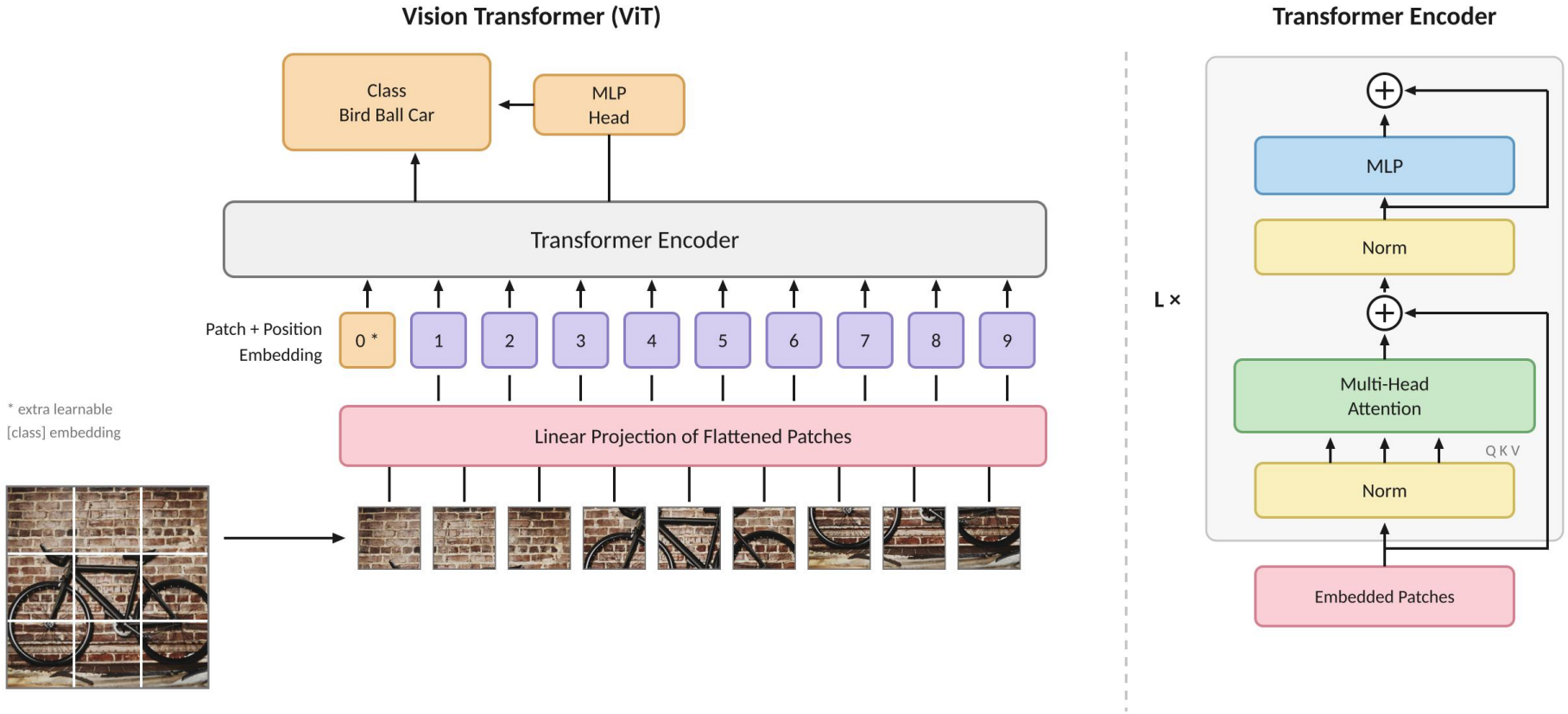






Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762

Vision Transformer

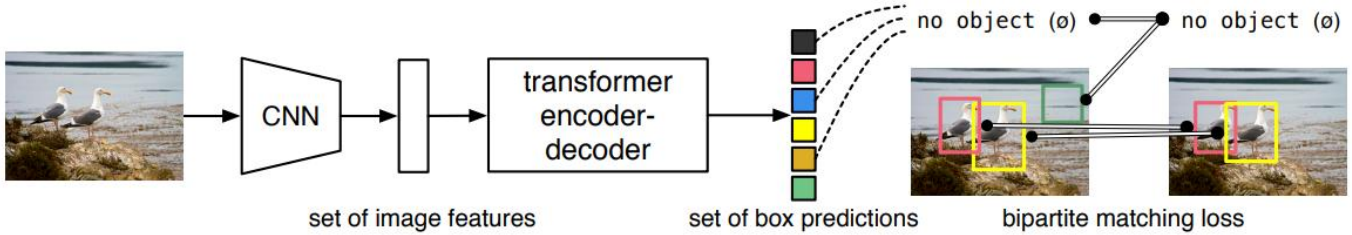


Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2010.11929
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

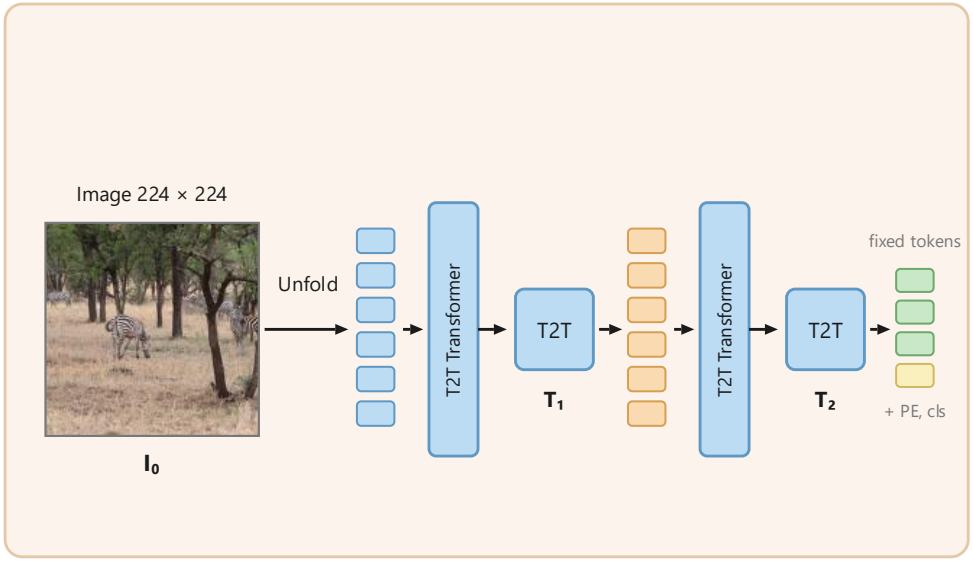
Token-to-Toke Vit

- Token-to-Token ViT
- SOTA Vision Transformers (DETR, SAM, etc.)

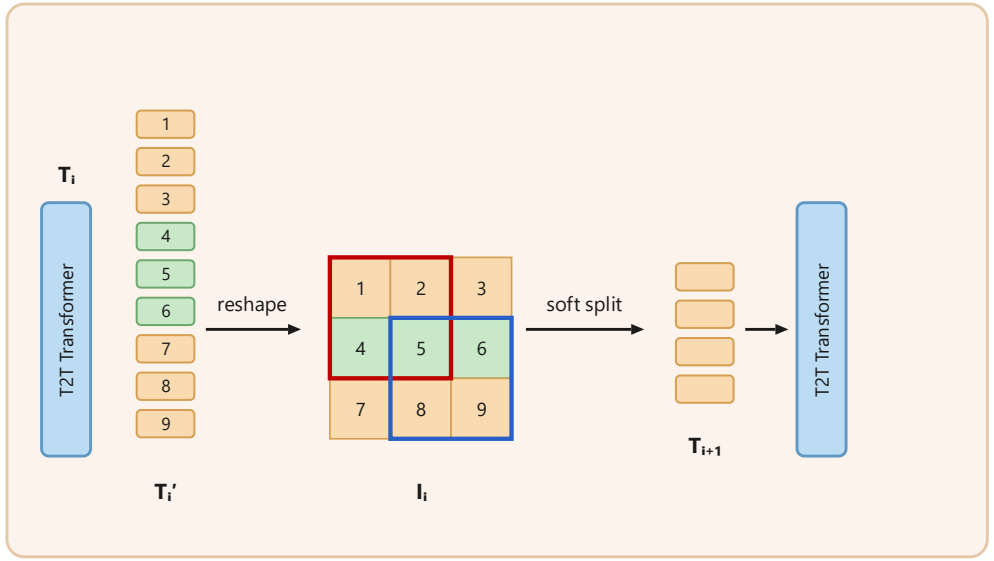
DETR: object detection as direct set prediction



Tokens-to-Token module

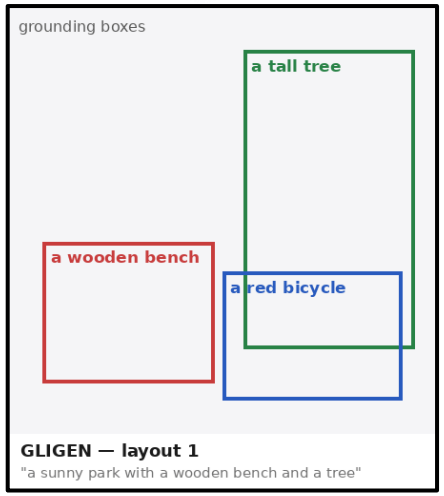


One T2T step

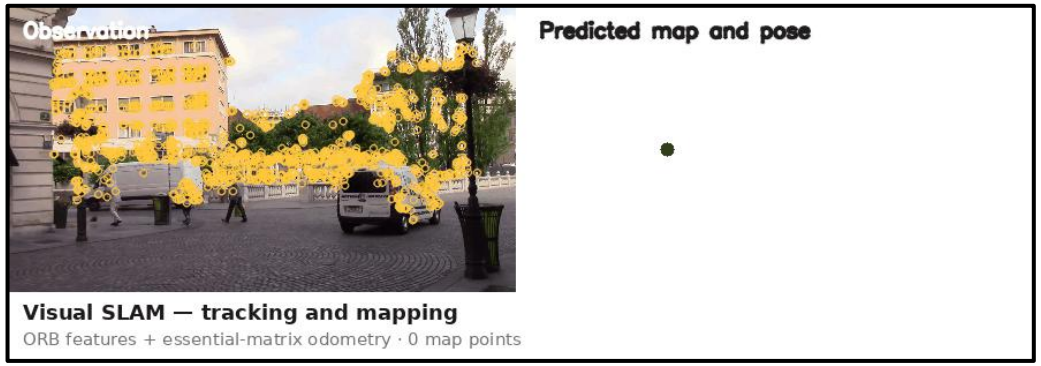


Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F. E. H., Feng, J., & Yan, S. (2021). Tokens-to-Token ViT: Training vision transformers from scratch on ImageNet. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2101.11986
Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2005.12872
Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment Anything. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2304.02643
Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CCO, via Wikimedia Commons
Image Source: Fig. 1: DETR architecture - Carion et al., End-to-End Object Detection with Transformers (2020), CCO 1.0 https://creativecommons.org/publicdomain/zero/1.0/, https://arxiv.org/abs/2005.12872

Tools Utilizing Deep Learning



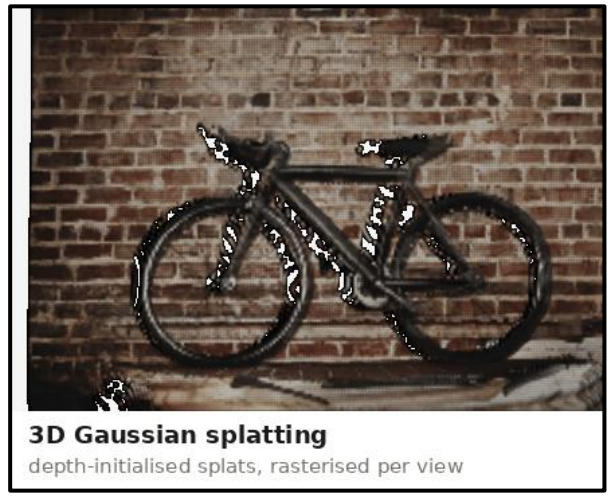
GLIGEN (Grounded Language-to-Image Generation)



Neural SLAM (Simultaneous Localization and Mapping)



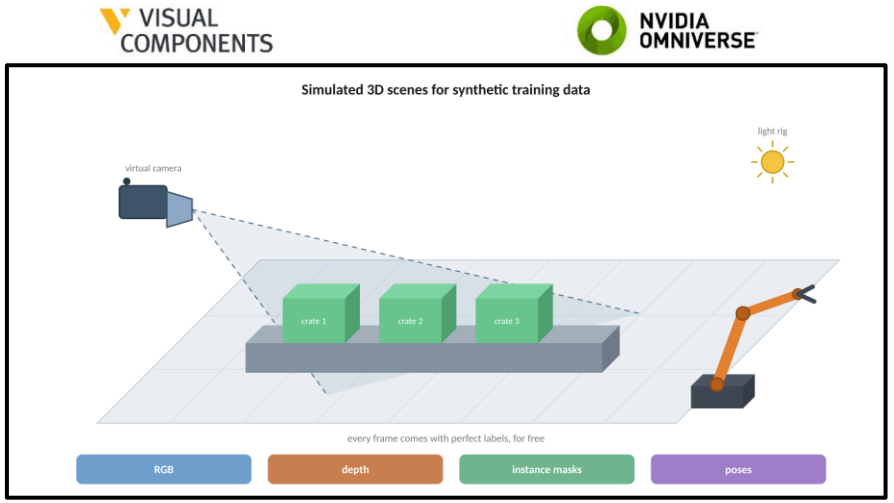
OpenAI Sora



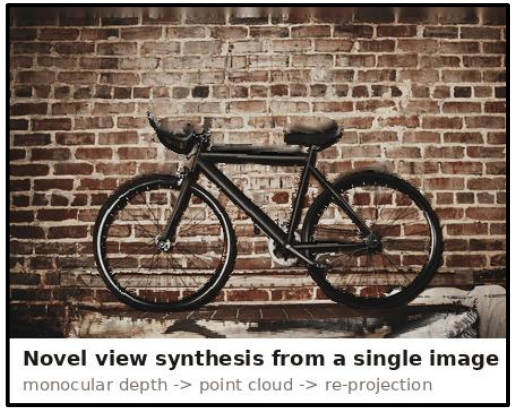
Gaussian Splatting

Video Source: Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

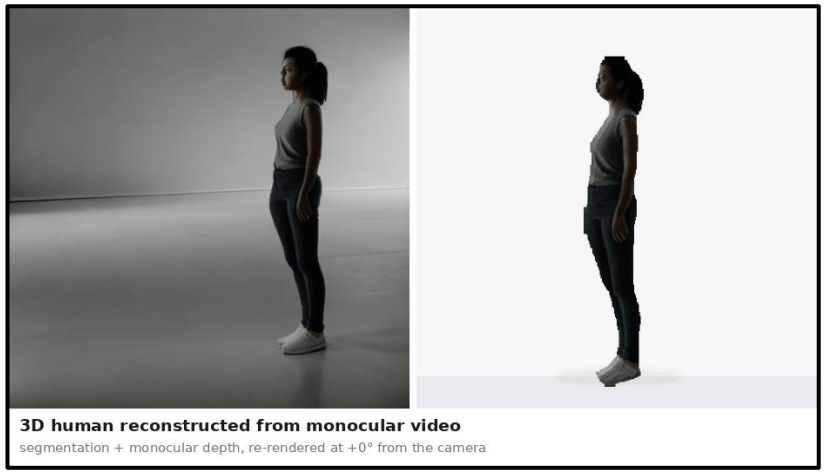
Tools Utilizing Deep Learning



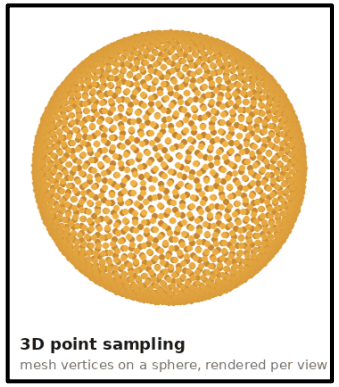
3D scene and object generation using NVIDIA's OMNIVERSE



NeRF (Neural Radiance Fields)



Vid2Avatar



Meta's PyTorch3D

Meta's PyTorch3D is a Python framework designed for 3D object processing and deep learning on 3D data.

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 11

Deep Learning Tools For Computer Vision part 1

1 Annotation Tools

2 Depth Estimation

3 Optical Flow

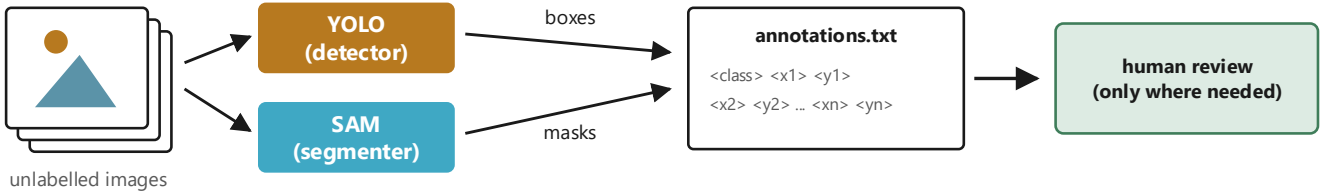
Automatic Annotation

- Automatic annotation refers to the process where a computer system automatically assigns metadata, such as bounding boxes or masks, to digital images without human intervention.

- Typically, these systems leverage pre-trained vision models like **YOLO**, **SAM**, or **Mask R-CNN**, sometimes using combinations or ensembles of these models to enhance predictions on new data.

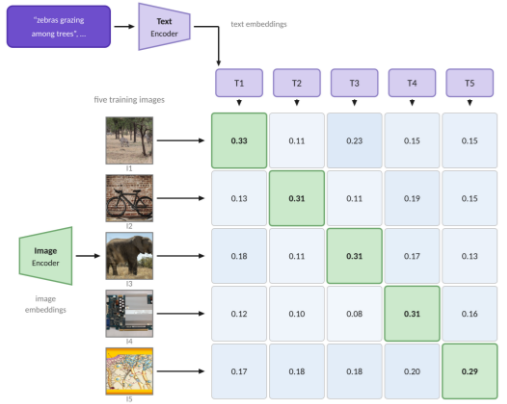
- Some solutions also offer adaptive options for detecting unusual objects. These options allow for real-time fine-tuning on a small subset of manually labeled data, enabling the model to perform automatic labeling on similar tasks afterward.

Simple model-assisted annotation pipeline



the model proposes, a person confirms - that is the whole idea

CLIP

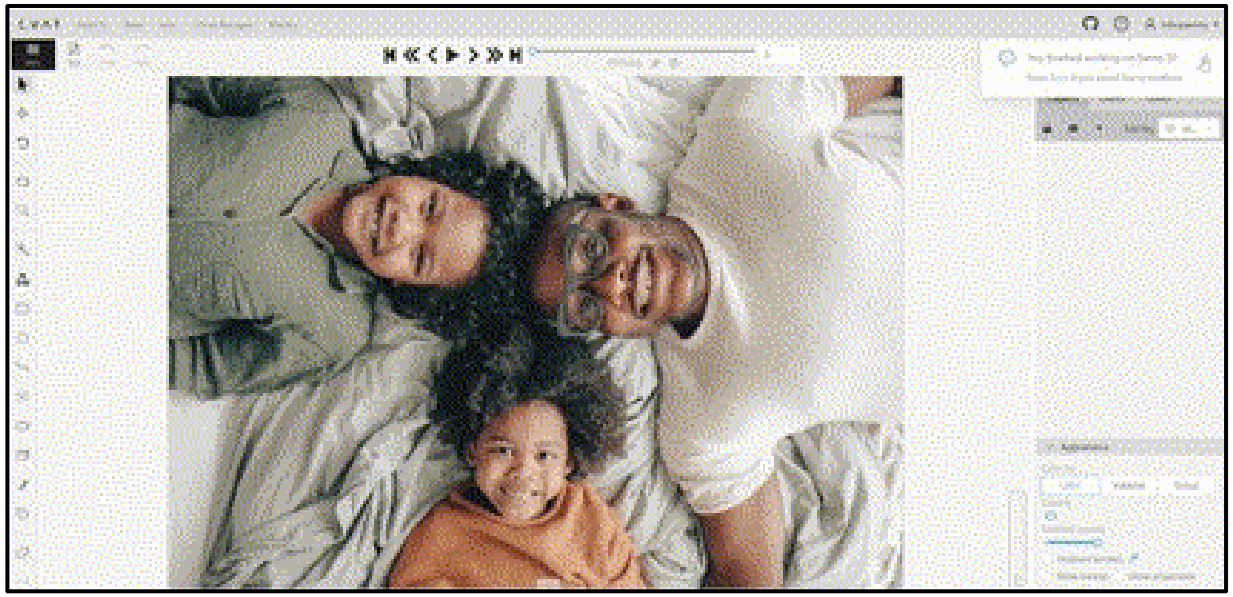


CLIP leverages contextual information about the specified task, making it effective as a model-assisted annotation tool for labeling specific object classes or shapes.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2103.00020

CVAT (Computer Vision Annotation Tool)

CVAT is an open-source, web-based model-assisted image and video annotation tool for labelling data for tasks of supervised machine learning: **object detection**, **image classification**, and **image segmentation**. It offers four basic types of annotation: boxes, polygons, polylines, and points.



Key features:

- Semi-automatic annotation
- Interpolation of shapes between keyframes
- Dashboard with a list of annotation projects and tasks
- CVAT supports PyTorch models through frameworks like Detectron2 and MMDetection for automatic annotation, allowing users to load and use models trained in PyTorch for annotation tasks.
- Exports the annotations in various formats (COCO, VOC, PASCAL, YOLO, KITTI, MOT)

https://github.com/cvat-ai/cvat
https://www.cvat.ai/post/an-introduction-to-automated-data-annotation-with-cvat-ai-cli
https://www.cvat.ai/

CVAT - Example

In this example, CVAT's SAM2 and YOLOv7 models are used in parallel to analyze the game screenshot. YOLOv7 is applied for object detection, specifically identifying a person, while SAM2 generates segmentation masks for various objects within the image.

Class: Person

Model: YOLOv7, SAM2



Image Source: Strand street pedestrian crossing 3.jpg, Mission for Inner City Cape Town, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons, Bounding boxes: YOLO11. Mask: FastSAM. Faces blurred.

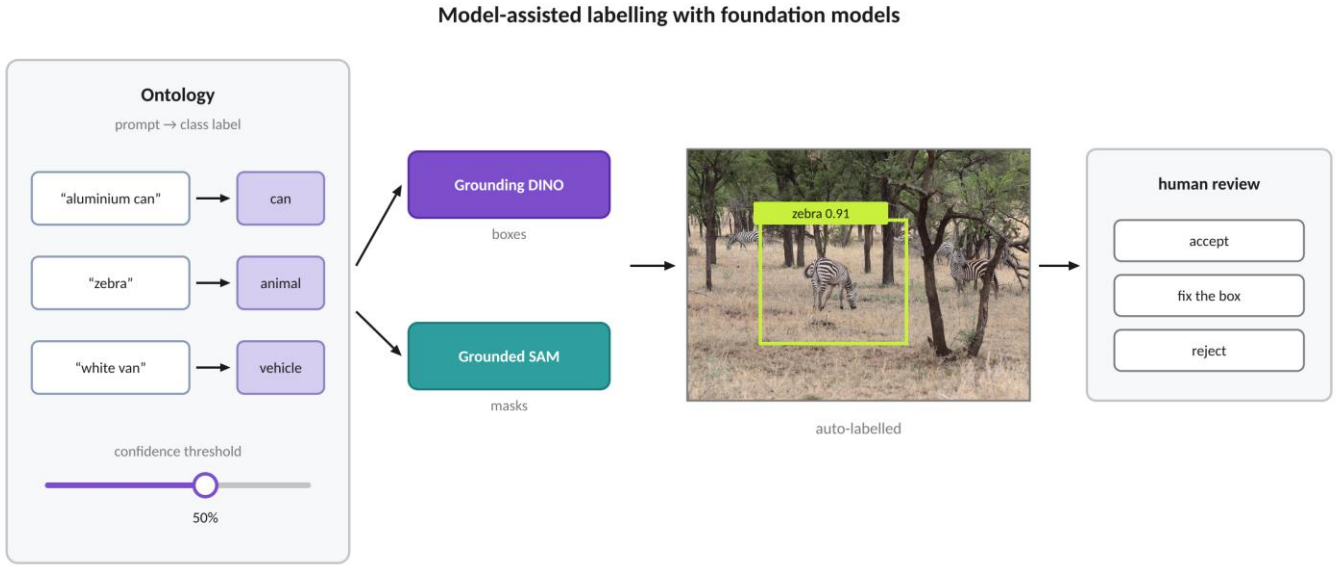
Roboflow/Autodistill

Model-Assisted annotation using foundation models:

- **Grounding DINO** for **Object Detection**
- **Grounded SAM** for **Segmentation**
- **Any other model** trained in Roboflow

Best suited **for labeling common objects**, such as vehicles, but may not perform well with specific or niche variants of objects.

An **ontology** (a structured representation of concepts and relationships) is used to map various prompts to object classes. The model takes a prompt and produces the labeling output with the specified class label.
https://docs.roboflow.com/annotate/automated-annotation-with-autodistill



Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., ... Others. (2023). Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv Preprint arXiv:2303. 05499.
Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., ... Zhang, L. (2024). Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2401.14159
Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CCO, via Wikimedia Commons

Make Sense

Make Sense is a free, open-source computer vision annotation tool.

- Licensed under a GPLv3 license.
- Does not require sing up or installation.
- Runs entirely in the browser, data never leaves
- Supports many formats: YOLO, VOC XML, VGG JSON etc.
- Annotations using boxes, lines, points and polygons.
- Integration with Roboflow as labelling assistant (private or public models)

https://www.makesense.ai/ (Piotr Skalski)
https://github.com/SkalskiP/make-sense

Scale AI

Scale AI is a paid, closed-source enterprise annotation tool.

- Suite for annotating data across domains of text, video, mapping, 3D and even audio.
- Comes with a built-in automatic evaluation tool that checks data annotation quality.
- Suggests areas in which you can improve your dataset.
- Supports many formats.
- Offers annotation teams and many auto-annotation tools as well.

https://scale.com/

Automatic Annotation – Other Tools

- Supervisely:
  - Can be used for LiDAR 3D sensor fusion data and DICOM volumes
  - Offers a community edition that is free for open-source projects by individuals
  - It provides a comprehensive manual annotation suite
- V7 (V7 Darwin):
  - primarily designed for larger teams and companies requiring annotation services
  - Offers a community license, though it is more limited compared to Supervisely, Roboflow, or CVAT
  - Supports multi-channel fluorescence and time-lapse imagery, as well as cell imaging
  - Provides voxel annotation tools for 3D and multi-planar views of medical images
- Datasaur:
  - For text datasets
  - Supports sentiment analysis, text classification, named entity recognition, and other related tasks

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 11

Deep Learning Tools For Computer Vision part 1

1 Annotation Tools

2 Depth Estimation

3 Optical Flow

Depth Estimation

Depth estimation (in a deep learning context) is the process of predicting the distance of objects from the camera using a neural network trained to infer depth information from visual input, such as RGB images or sequences.

- **Monocular** Depth Estimation:
  - Estimating depth from a single image
  - Utilizes spatial cues (monoscopic depth cues) and scene understanding from a single viewpoint
  - Also referred to as Single-image depth estimation or SIDE
- **Stereo** Depth Estimation:
  - Uses two images from slightly different viewpoints (2 camera setup)
  - Estimates depth by calculating the disparity between matching points in each image (stereopsis)
  - Usually done by epipolar geometry and block matching
  - Disparity can also be learned by vision models
- Depth (Structure) **from Motion**:
  - Multi-view scenario
  - Estimates depth by matching keypoints across multiple images and triangulating their positions in 3D space
  - Sensitive from motion blur

Depth Estimation

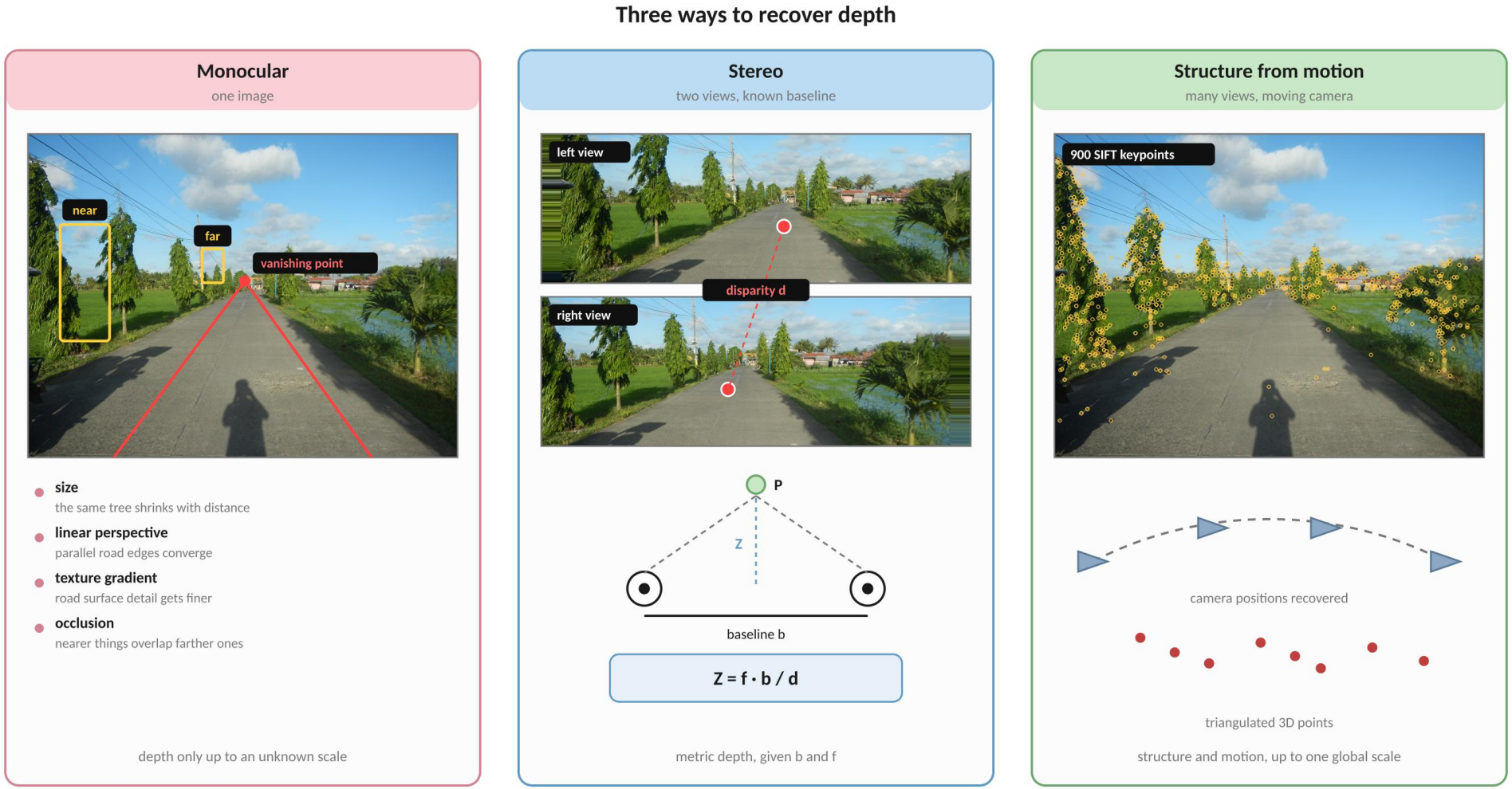


Image Source: https://commons.wikimedia.org/wiki/Category:Paddy_fields,_grasslands,_trees_and_irrigation_(Balante_Proper_%26_Balante_Boulevard,_Gapan_City), Judgefloro - Balante Boulevard, Gapan City, CCO, via Wikimedia Commons

Why Depth Estimation Matters

3D point sensors like LIDAR, ToF and depth cameras are often expensive, bulky, and power-hungry, making them impractical for many consumer devices.

In contrast, monocular or stereo depth estimation can be achieved with a single or pair of RGB cameras, which are affordable, compact, and already widely integrated into these devices, enabling depth perception without additional hardware costs.





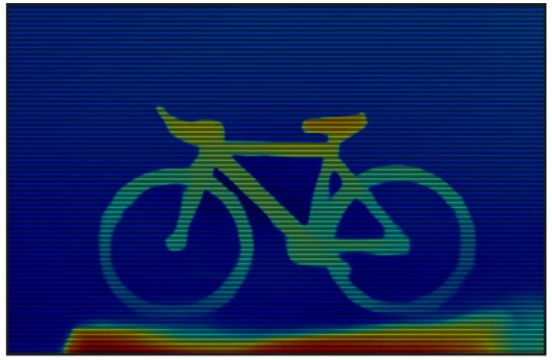




Stereo Camera



Structured Light Camera



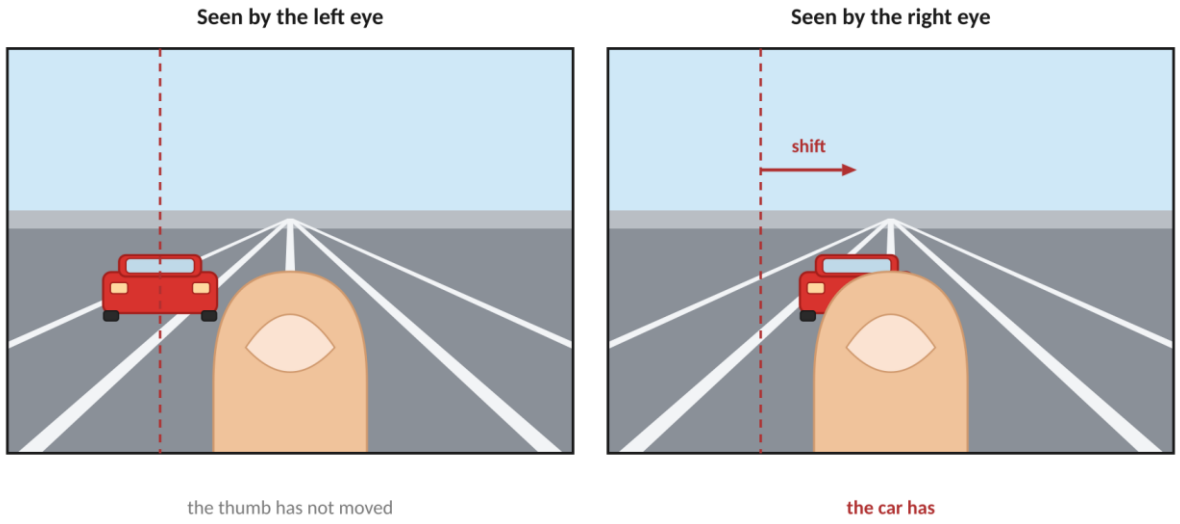
LIDAR

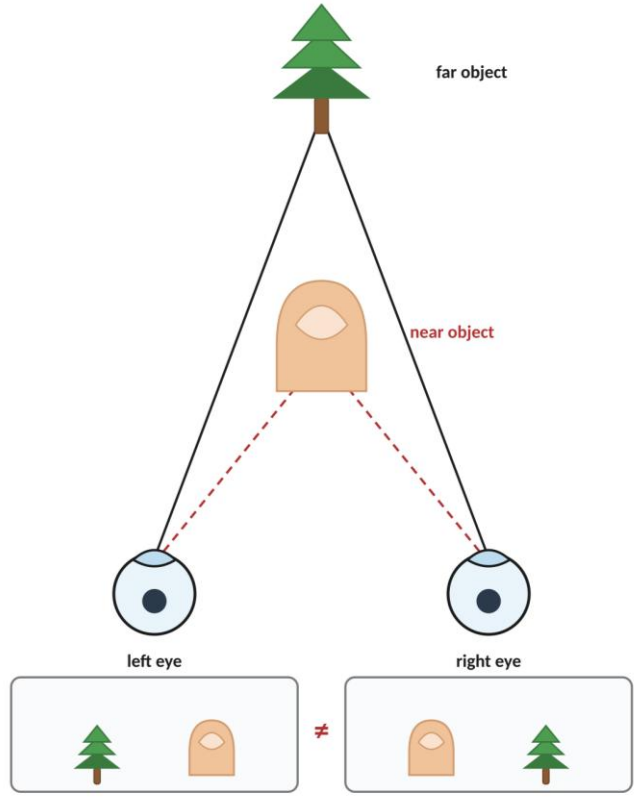
Image Source: **Sounds of Changes, Road traffic on Stritarjeva street** ,CC BY 3.0; HRowntree, CCO; Darius Soodmand, CCO,via Wikimedia Commons.

Stereo Vision

The human eye uses stereopsis, which is a crucial aspect of depth perception, relying on binocular disparity to create a 3D view of the world. It involves the brain fusing slightly different images from each eye, allowing us to judge distances and navigate our environment effectively.

Retinal disparity refers to the slight differences in the position of an object's image on the left and right retinas due to the horizontal separation of the eyes. Objects at different distances from the observer will have different amounts of retinal disparity, with closer objects having larger disparities than distant objects

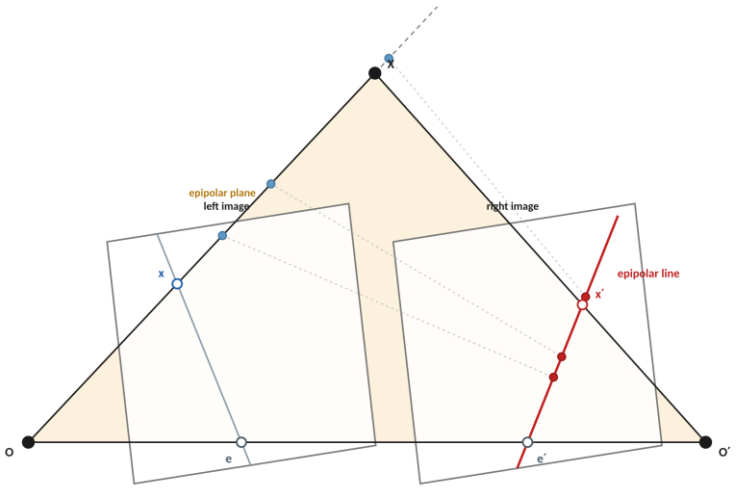


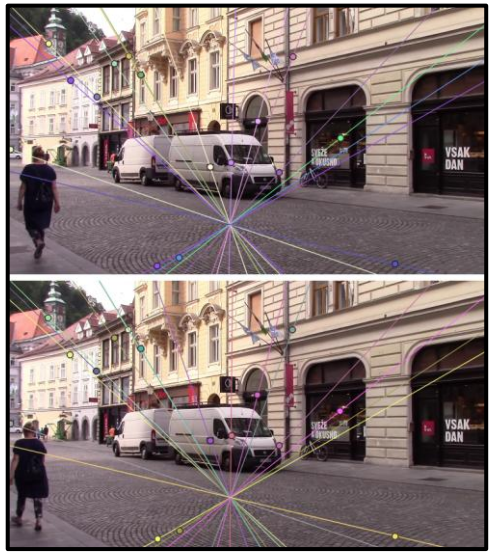


Stereo Depth Estimation

Stereo depth estimation is a technique that determines the depth of objects in a scene by analyzing the disparity between two images captured from slightly different viewpoints.

The earliest algorithms for stereo depth estimation, developed in the 1990s, achieved impressive results by leveraging geometric constraints to mathematically replicate the concept of stereopsis, enabling real-time processing.





Sounds of Changes - Road traffic on Stritarjeva street, CC BY 3.0, via Wikimedia Commons.

The core approach to depth estimation with stereo cameras relies on triangulation and stereo matching. Accurate results depend on proper calibration and rectification, which constrain the problem to a 2D plane called the epipolar plane, reducing the search to a line along the epipolar line.

Image Source: **Sounds of Changes, Road traffic on Stritarjeva street** ,CC BY 3.0; HRowntree, CCO; Darius Soodmand, CCO,via Wikimedia Commons.

Monocular Depth Estimation

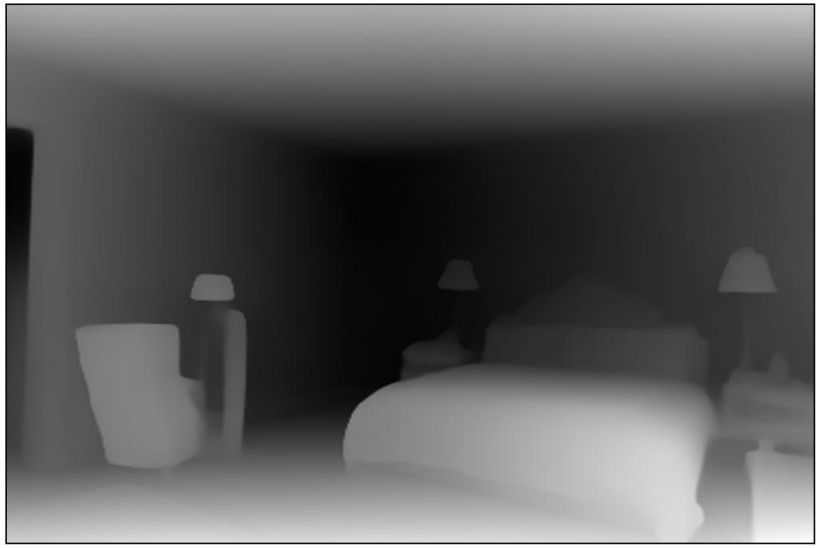
Depth computation from a single image is inherently ambiguous, as there are multiple ways to project the same 3D scene onto the 2D plane of an image (taking a photo of an object). To address this ambiguity, various cues such as object size, occlusion, and perspective must be considered.

Before the advent of deep learning and monocular depth estimation, special sensors or aligned stereo sensors were used to calculate depth by measuring discrepancies between stereo pairs. These solutions were mostly limited to lab conditions.

Input image



DPT depth estimate



The demo uses **DPT** (Dense-Prediction Transformer) showcasing its versatility in estimating depth on any 2D image that represents a projection of a 3D scene.

Image Source: EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons, Output: depth estimation mask

Why Monocular Depth Estimation?

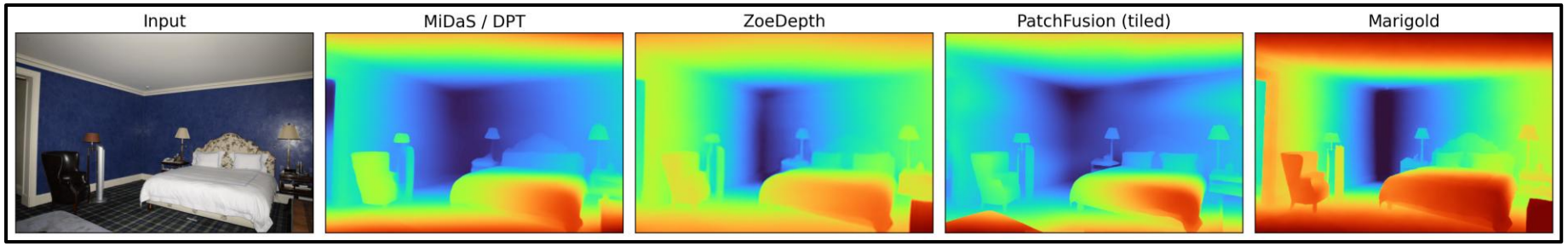
A single-camera setup is significantly cheaper than using LiDAR or stereo cameras (camera parameters are not required)

Monocular depth estimation can work with any standard camera in various settings, including mobile devices, drones, and surveillance

Monocular models typically allow for faster inference speeds compared to multi-camera setups

Monocular depth estimation models trained with diverse datasets tend to generalize well across different environments and lighting conditions

Vision Transformers have revolutionized monocular depth estimation with models such as **MiDAS v3.1**, **ZoeDepth**, **PatchFusion**, and **Marigold**.



Interior scene - public domain, via Wikimedia Commons. Depth maps: MiDaS/DPT, ZoeDepth, tiled DPT, Marigold.

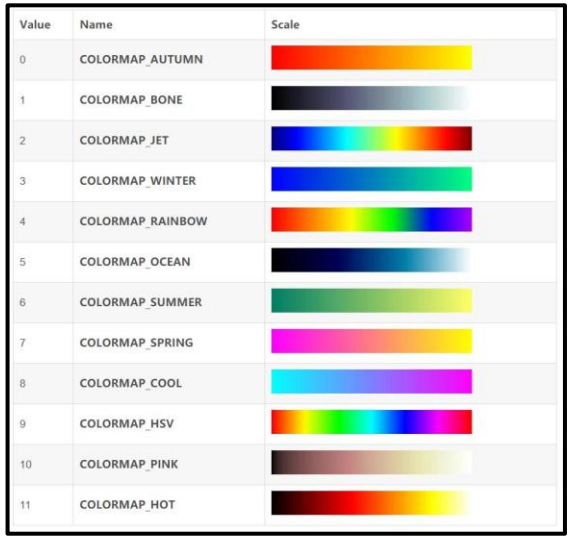


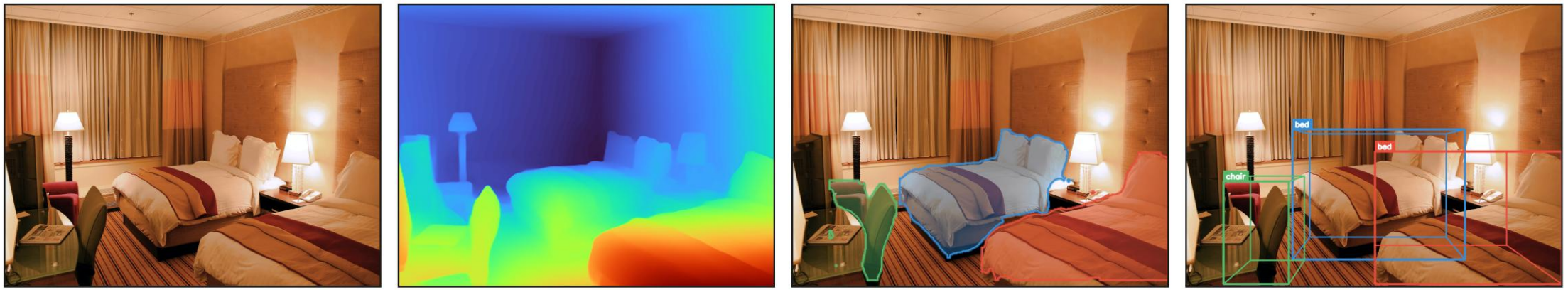
Image Source: EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons

Benchmark Datasets

SUN RGB-D: An RGB-D Scene Understanding Benchmark Suite. A dataset captured by four different sensors containing 10,000 RGB-D images. The whole dataset is densely annotated and includes 146,617 2D polygons and 58,657 3D bounding boxes with accurate object orientations, as well as a 3D room layout and category for scenes.

Contains images from the NYU depth v2, Berkeley B3DO, and SUN3D datasets.

One of the most popular datasets for depth estimation and segmentation tasks.



RGB | depth | 2D polygons | 3D boxes

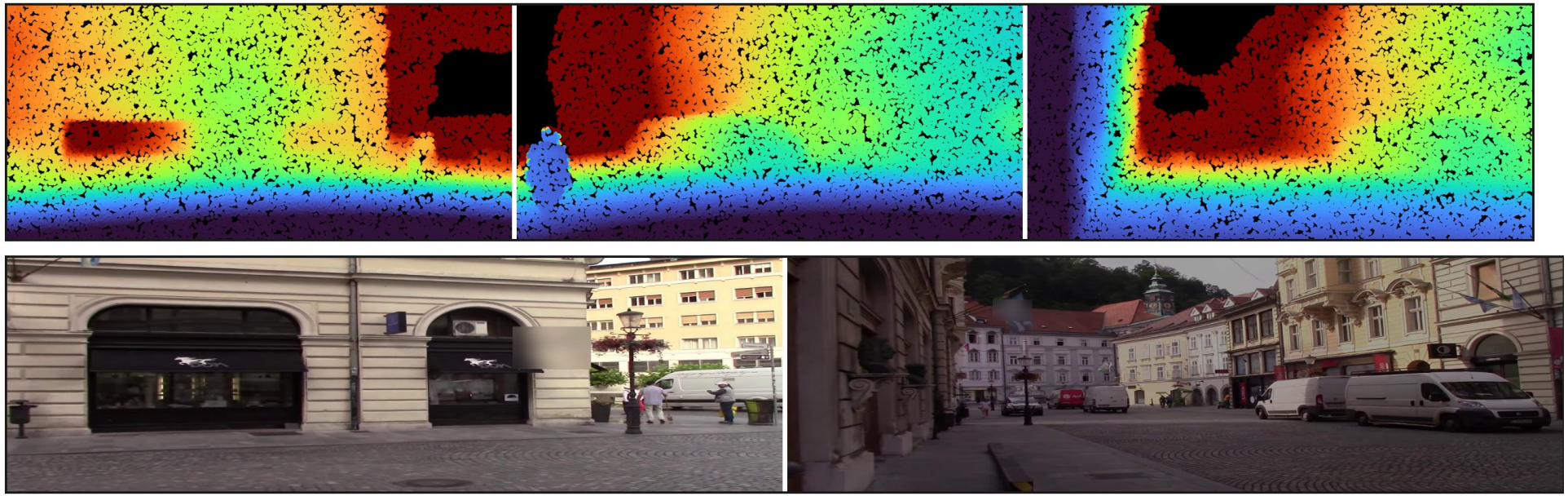
Song, S., Lichtenberg, S. P., & Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 567–576. doi:10.1109/CVPR.2015.7298655
Image Source: Hotel-room-renaissance-columbus-ohio.jpg, Derek Jensen (Tysto), Public domain, via Wikimedia Commons, . Depth, masks and 3D boxes are our own model output.

Benchmark Datasets

KITTI Vision Benchmark Suite: The **KITTI Vision Benchmark Suite** is a comprehensive real-world dataset for benchmarking computer vision tasks, including stereo depth estimation, optical flow, visual odometry, and 3D object detection.

Collected by driving around the mid-size city of Karlsruhe, rural areas, and highways, the dataset captures scenes with up to 15 cars and 30 pedestrians per image.

High-precision ground truth data is generated using a Velodyne laser scanner and a GPS localization system.



Video Source: Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, Point clouds are our own model output

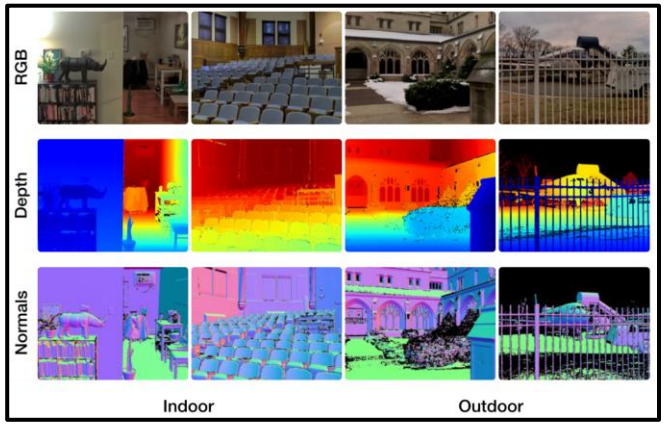
Benchmark Datasets

Many Others:

- Sintel (Animation)
- DDAD (Dense Depth for Autonomous Driving)
- ETH3D (Multi-view stereo benchmark)
- DIODE (Dense Indoor and Outdoor Depth)



Sintel



DIODE

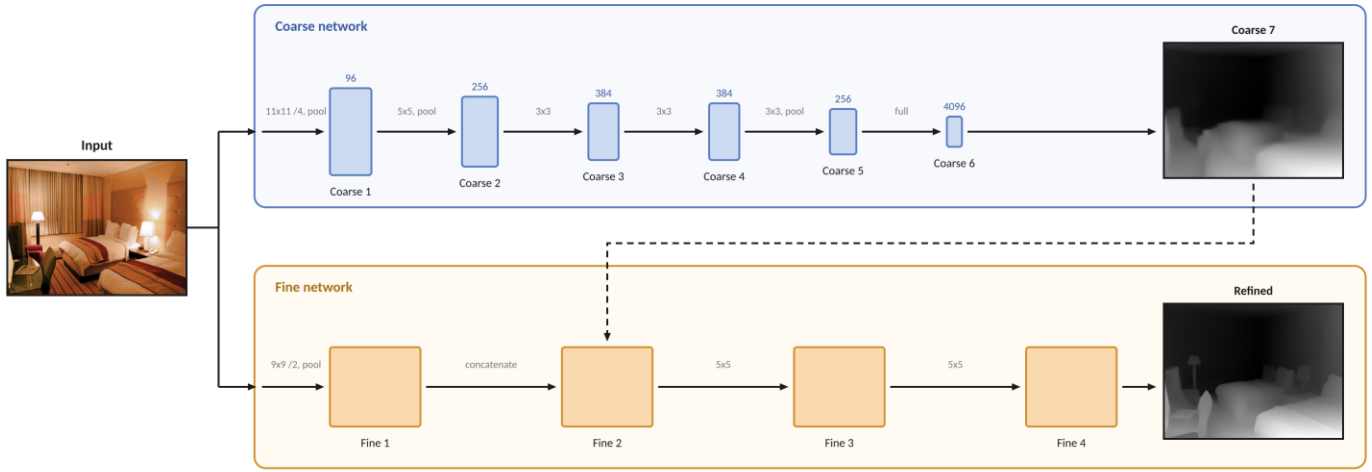
Image Source: DIODE — Vasiljevic et al., DIODE: A Dense Indoor and Outdoor DEpth Dataset (2019), MIT License, Copyright © 2021 diode-dataset, https://diode-dataset.org/
Image Source: Sintel — © Blender Foundation, sintel.org, CC BY 3.0 https://creativecommons.org/licenses/by/3.0/; optical-flow ground truth from the MPI Sintel dataset, Butler et al., ECCV 2012, https://sintel.is.tue.mpg.de/

Depth Estimation – Naïve Approach (2014)

The original paper that started the hype behind deep learning based single-image depth estimation was *"Depth Map Prediction from a Single Image using a Multi-Scale Deep Network"* introducing CNNs to estimate depth from RGB images without the need for multi-camera setup.

- A two-scale CNN architecture captures both **global** (coarse network) and **local** (fine network) **depth cues**
- The **end-to-end** approach relies solely on a CNN backbone
- The lower- and middle-layers use max-pooling to **aggregate information** from various parts of the image, reducing it to a smaller spatial dimension
- The final output, at a **quarter of the input resolution** *, is reshaped from the last fully connected layer

Derek Jensen (Tysto) - Hotel room, public domain, via Wikimedia Commons. Depth panels are our own model output.



Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1406.2283
Image Source: Hotel-room-renaissance-columbus-ohio.jpg, Derek Jensen (Tysto), Public domain, via Wikimedia Commons, . Depth, masks and 3D boxes are our own model output.

Naïve Approach - Loss

Due to the issues in different input and output sizes a pixel-wise loss function cannot be used. Traditional loss functions, like Mean Squared Error (MSE), calculate the error by comparing each pixel's predicted depth with its true depth value.

This means they heavily penalize any differences between the predicted depth and the exact ground truth depth but because there's only one viewpoint, **the model doesn't have enough information to determine the exact distances** (absolute depth) of objects.

Mean Error Squared

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Where Y_i is the target absolute depth value for a pixel, and $\hat{Y}_i$ is the predicted depth. The Mean Squared Error (MSE) is calculated for each pixel.

Naïve Approach - Loss

For example, if the true depth of object A is 5 meters and object B is 10 meters, the model might predict them as 2.5 meters and 5 meters, respectively. This prediction is **correct in a relative sense** (object A is still closer than object B) but is **off in an absolute sense** (distances are half of the actual values).

The scale-invariant loss focuses on the **relative differences** between predicted and true depth values across the image, reducing sensitivity to overall scale mismatches. It compares depth values in a way that penalizes inconsistencies in relative depth relationships rather than absolute depth values.

$$D(y, y^*) = \frac{1}{n^2} \sum_{i,j} \left((\log y_i - \log y_j) - (\log y_i^* - \log y_j^*) \right)^2$$

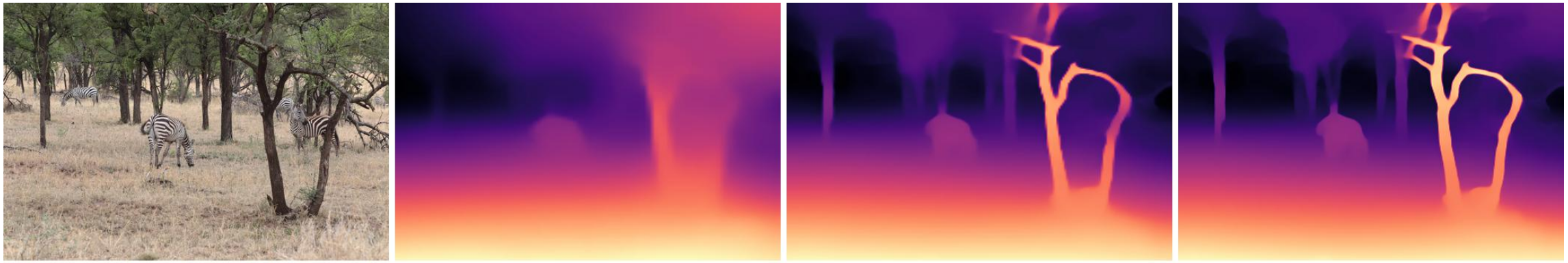
Where the terms $\log y_i$ and $\log y_i^*$ represent the true and predicted depth values for pixel i.

The pairwise differences capture **relative depth relationships** by comparing how the depth of one pixel relates to another.

The sum is taken over all pixel pairs, and the result is averaged (scales poorly with the size of the image).

Naïve Approach – Examples

Agness Abubakar Saidi - Zebras grazing among acacia trees, Serengeti National Park, CCO, via Wikimedia Commons



Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1406.2283
Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CCO, via Wikimedia Commons

Depth Estimation – Unsupervised Approach (2016-17)

The paper *"Unsupervised Monocular Depth Estimation with Left-Right Consistency"* proposes an unsupervised depth estimation approach using stereo image pairs.

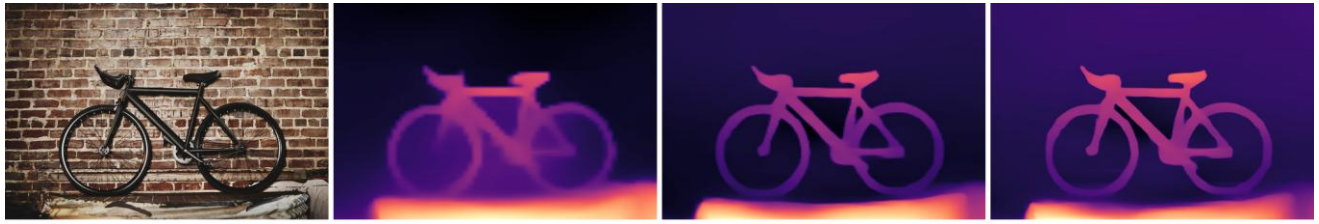
During training, the model predicts depth from a single image by leveraging **left-right consistency**. Although depth is predicted for each image individually, a consistency constraint ensures that the depth map for the left image aligns with the right image's view and vice versa, enforcing coherence without ground truth depth maps.

At test time, the model requires only one image to produce robust, generalized depth estimates.

No ground truth depth data are needed.

"Encoder"						
layer	**k**	**s**	**chns**	**in**	**out**	**input**
conv1	7	2	3/32	1	2	left
conv1b	7	1	32/32	2	2	conv1
conv2	5	2	32/64	2	4	conv1b
conv2b	5	1	64/64	4	4	conv2
conv3	3	2	64/128	4	8	conv2b
conv3b	3	1	128/128	8	8	conv3
conv4	3	2	128/256	8	16	conv3b
conv4b	3	1	256/256	16	16	conv4
conv5	3	2	256/512	16	32	conv4b
conv5b	3	1	512/512	32	32	conv5
conv6	3	2	512/512	32	64	conv5b
conv6b	3	1	512/512	64	64	conv6
conv7	3	2	512/512	64	128	conv6b
conv7b	3	1	512/512	128	128	conv7

"Decoder"						
layer	**k**	**s**	**chns**	**in**	**out**	**input**
upconv7	3	2	512/512	128	64	conv7b
iconv7	3	1	1024/512	64	64	upconv7+conv6b
upconv6	3	2	512/512	64	32	iconv7
iconv6	3	1	1024/512	32	32	upconv6+conv5b
upconv5	3	2	512/256	32	16	iconv6
iconv5	3	1	512/256	16	16	upconv5+conv4b
upconv4	3	2	256/128	16	8	iconv5
iconv4	3	1	128/128	8	8	upconv4+conv3b
disp4	3	1	128/2	8	8	iconv4
upconv3	3	2	128/64	8	4	iconv4
iconv3	3	1	130/64	4	4	upconv3+conv2b+disp4*
disp3	3	1	64/2	4	4	iconv3
upconv2	3	2	64/32	4	2	iconv3
iconv2	3	1	66/32	2	2	upconv2+conv1b+disp3*
disp2	3	1	32/2	2	2	iconv2
upconv1	3	2	32/16	2	1	iconv2
iconv1	3	1	18/16	1	1	upconv1+disp2*
disp1	3	1	16/2	1	1	iconv1

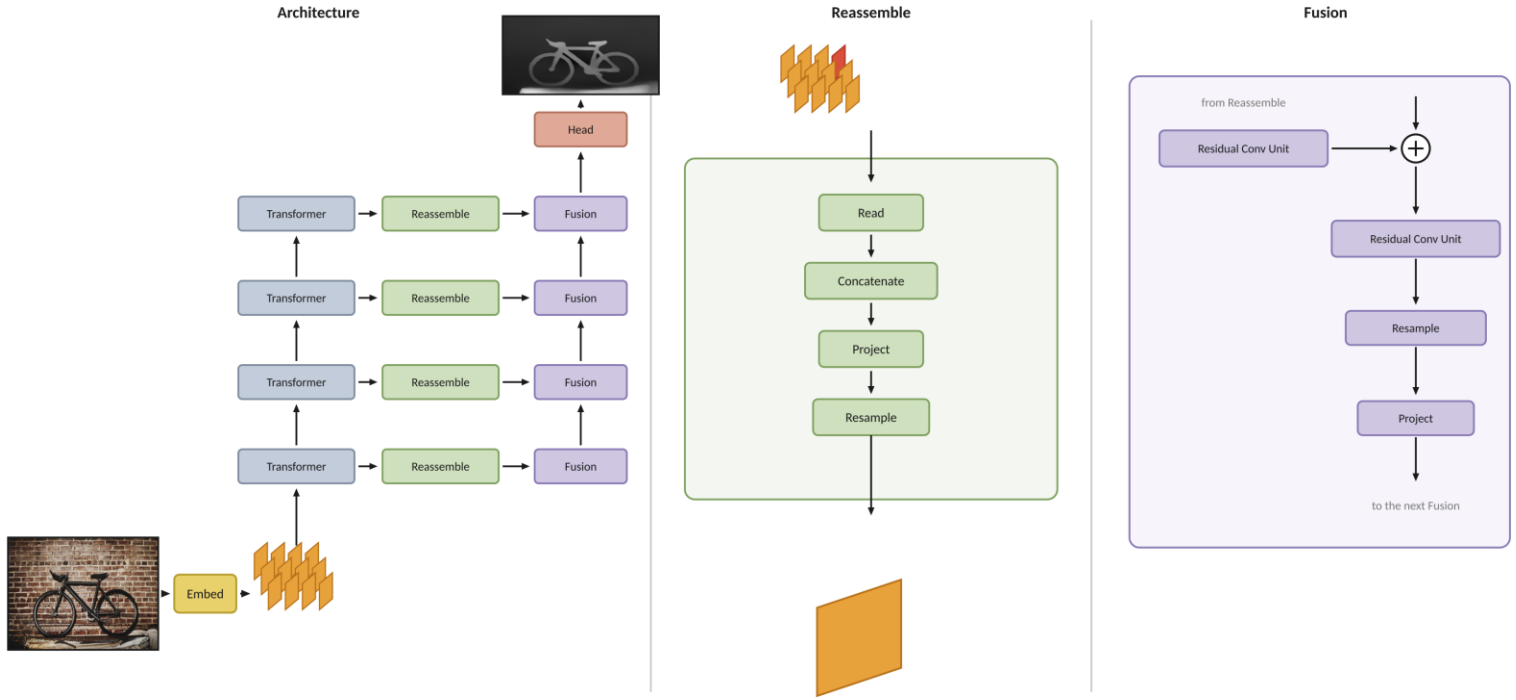


Godard, C., Aodha, O. M., & Brostow, G. J. (2017). Unsupervised Monocular Depth Estimation with Left-Right Consistency. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1609.03677
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Dense Prediction Transformer (DPT)

The Dense Prediction Transformer (DPT) was introduced in Vision Transformers for Dense Prediction (2021) from Intel Labs research. Estimatates relative depth.

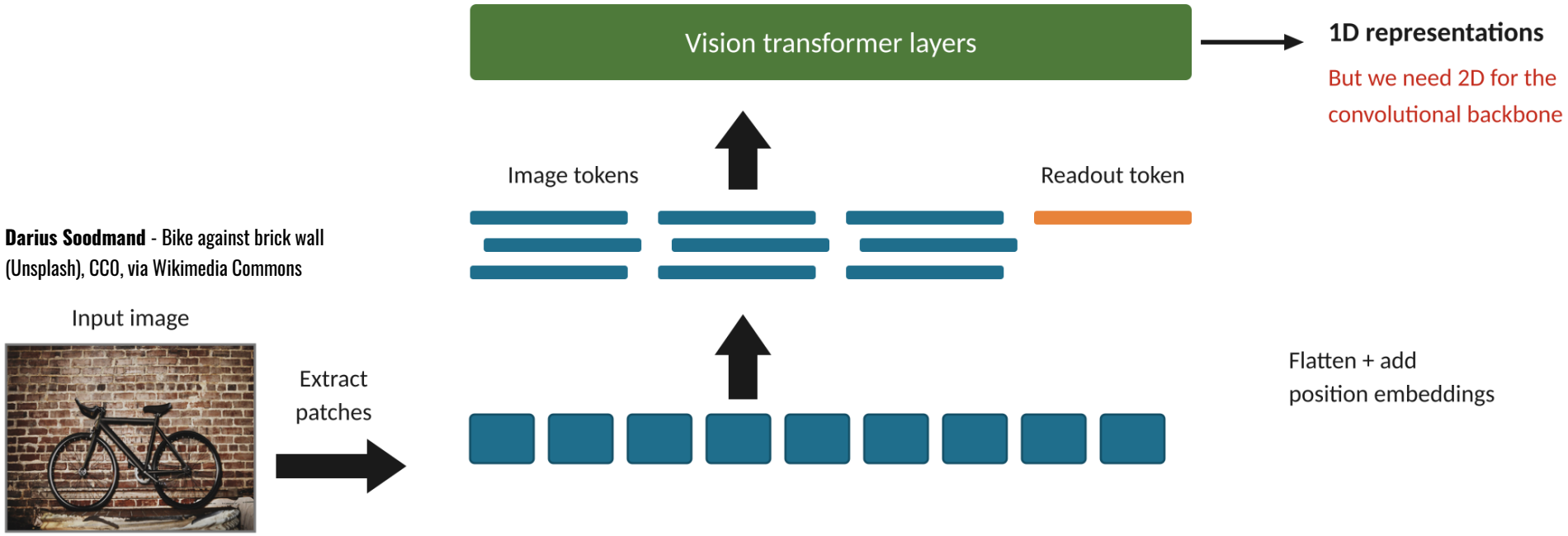
DPT has **an encoder-decoder structure** including transformer layers, while the overall architecture is **hybrid with convolutional layers.**



Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision Transformers for Dense Prediction. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2103.13413
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Dense Prediction Transformer (DPT)

1. Encoder is the ViT architecture from previous lecture. **CLS** token changed to **redout** token.



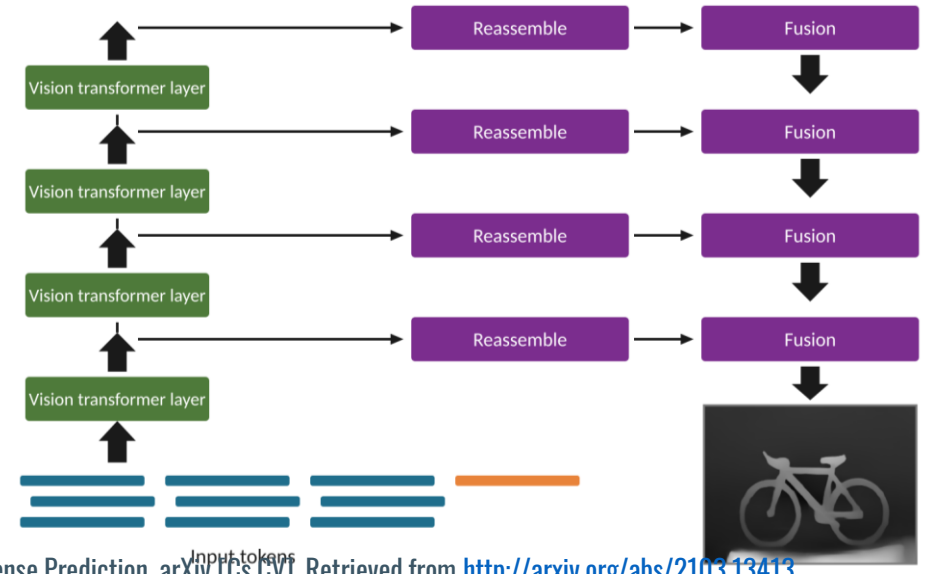
Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision Transformers for Dense Prediction. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2103.13413
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Dense Prediction Transformer (DPT)

2. To pass linear representations into the decoder, image-like representations are recovered using the reassemble operation.
   - During reassemble, image tokens are read, concatenated following the positions of the initial image patch, and then resampled.
   - Given D as the feature dimension of each of N image tokens, patch size p^2, then the spatial concatenation resulted in the representation shape with height and width as:

$$ND = \left(\frac{H}{p}\right)\left(\frac{W}{p}\right)D$$

   - The decoder uses RefineNet-based [5] feature fusion from reassembled image-like representations sampled at four different stages from four different resolutions



Darius Soodmand - Bike against brick wall (Unsplash), CC0, via Wikimedia Commons

Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision Transformers for Dense Prediction. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2103.13413
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Dense Prediction Transformer (DPT)



DPT - Hybrid



DPT - Large



Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision Transformers for Dense Prediction. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2103.13413
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Monocular Depth Estimation – MiDAS (2019-22)

MiDAS was another pioneering model that sparked interest in monocular depth estimation using CNNs. Although it is no longer considered state-of-the-art, newer models continue to build upon its backbone.

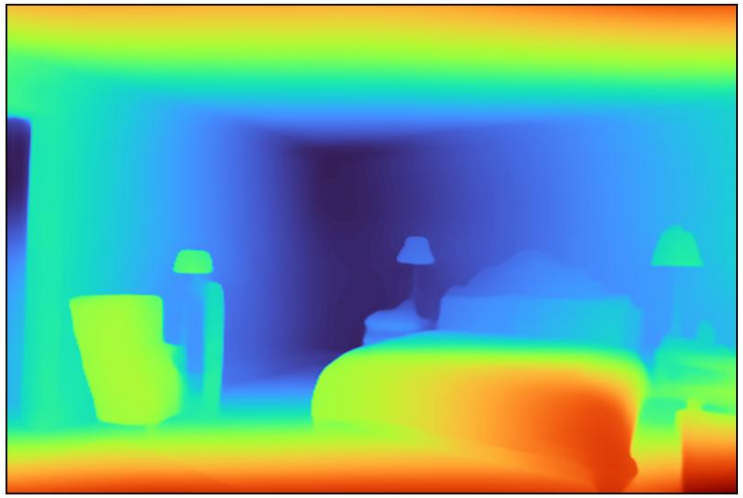
MiDAS has an important characteristic: the inferred depth is relative (relative depth estimation, **RDE**). In other words, it excels at estimating the relative position of pixels (e.g., one pixel being behind another), but it isn't designed to provide consistent, absolute depth measurements.

MiDAS v3.1

Original image



Depth map



3D point cloud using pixel extrusion



Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V. (2020). Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1907.01341
Image Source: EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons, Output: depth estimation mask

MDE – ZoeDepth (2023)

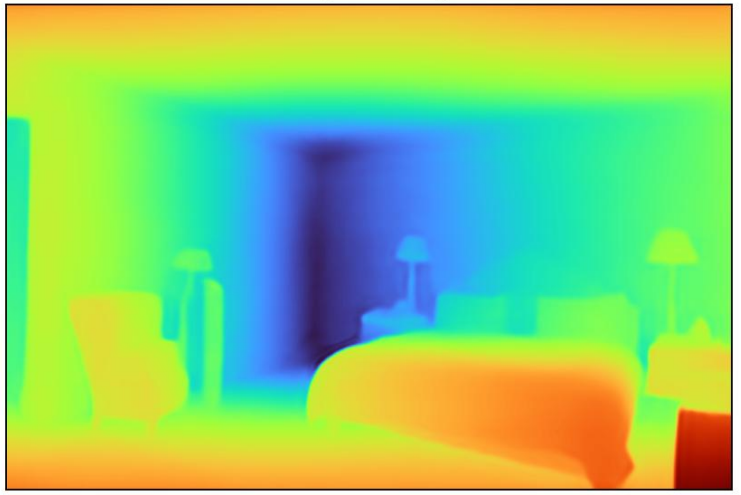
ZoeDepth was built on top of **MiDAS** as a clear improvement, designed to make inferences in metric units (metric depth estimation, **MDE**). The advancements over **MiDAS** become evident when projecting onto a 3D space and observing from novel viewpoints.

ZoeDepth

Original image



Depth map



3D point cloud using pixel extrusion



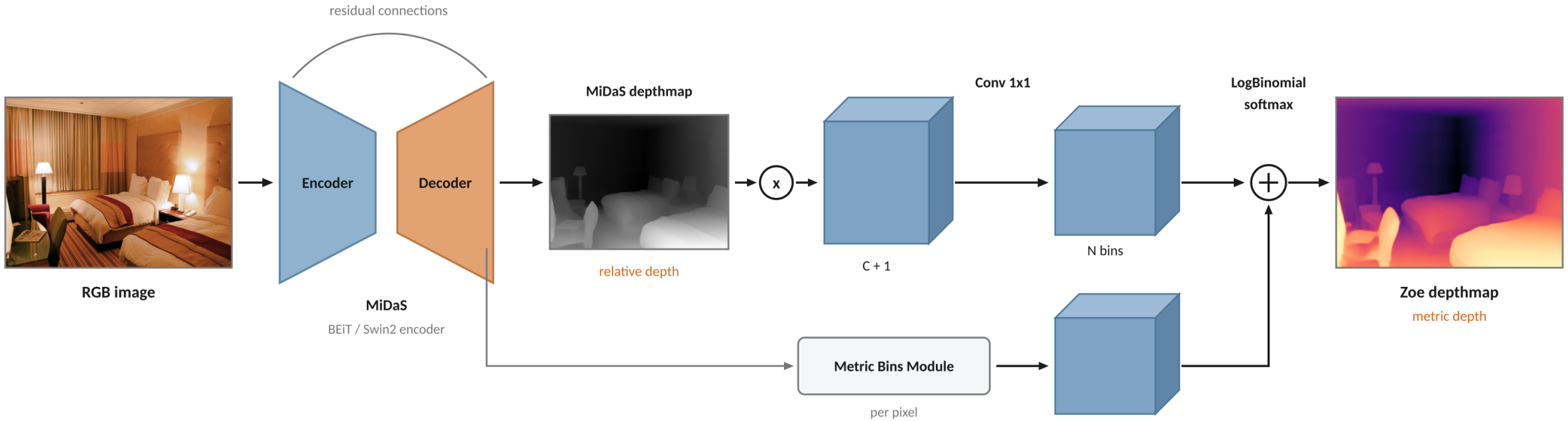
Bhat, S. F., Birkl, R., Wofk, D., Wonka, P., & Müller, M. (2023). ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2302.12288
Image Source: EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons, Output: depth estimation mask

MDE – ZoeDepth (2023)

ZoeDepth was built on top of **MiDAS** as a clear improvement, designed to make inferences in metric units. The advancements over **MiDAS** become evident when projecting onto a 3D space and observing from novel viewpoints.

ZoeDepth attempts to **combine the two tasks of relative depth estimation and metric depth estimation**.

Transition from CNNs to Vision Transformer encoders, such as BEiT and Swin2.



Bhat, S. F., Birkl, R., Wofk, D., Wonka, P., & Müller, M. (2023). ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2302.12288
Image Source: Hotel-room-renaissance-columbus-ohio.jpg, Derek Jensen (Tysto), Public domain, via Wikimedia Commons, . Depth, masks and 3D boxes are our own model output.

MDE – PatchFusion (2023-24)

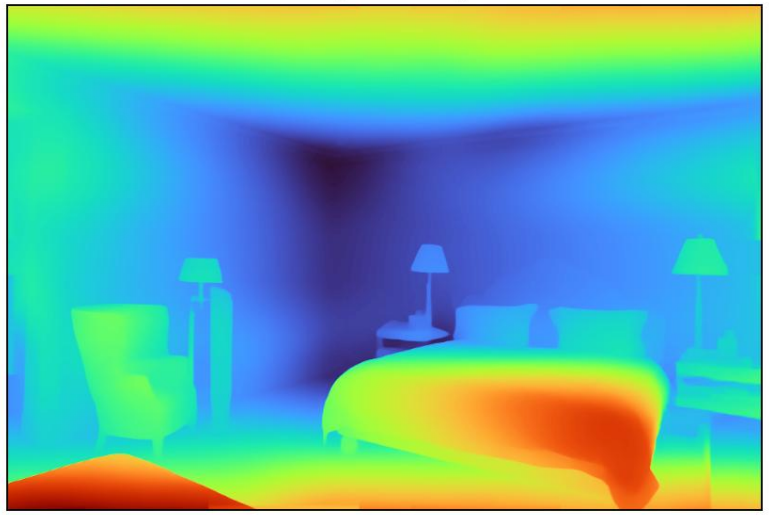
PatchFusion is built on top of ZoeDepth and its core principle of the algorithm is that instead of making a single estimation of an entire image, we iteratively create multiple estimations of different regions – patches – of the entire image.

These patches of estimated depth are “stitched” together.

Original image



Depth map



3D point cloud using pixel extrusion



Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C., & Schindler, K. (2024). Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2312.02145
Image Source: EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons, Output: depth estimation mask

MDE – Marigold (2023-24)

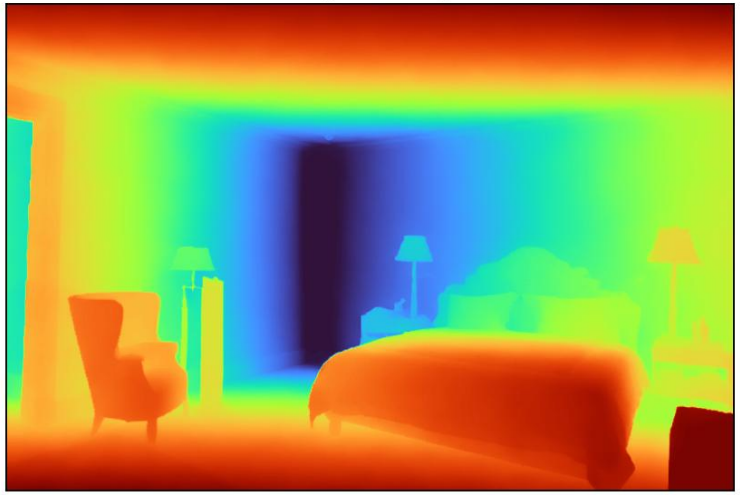
Marigold leverages the extensive visual knowledge embedded in modern generative models. Built on **Stable Diffusion** and fine-tuned with synthetic data, it achieves state-of-the-art monocular depth estimation with impressive zero-shot transfer capabilities on unseen data

- **Best Paper Award** - CVPR 2024
- **SOTA**

Original image



Depth map

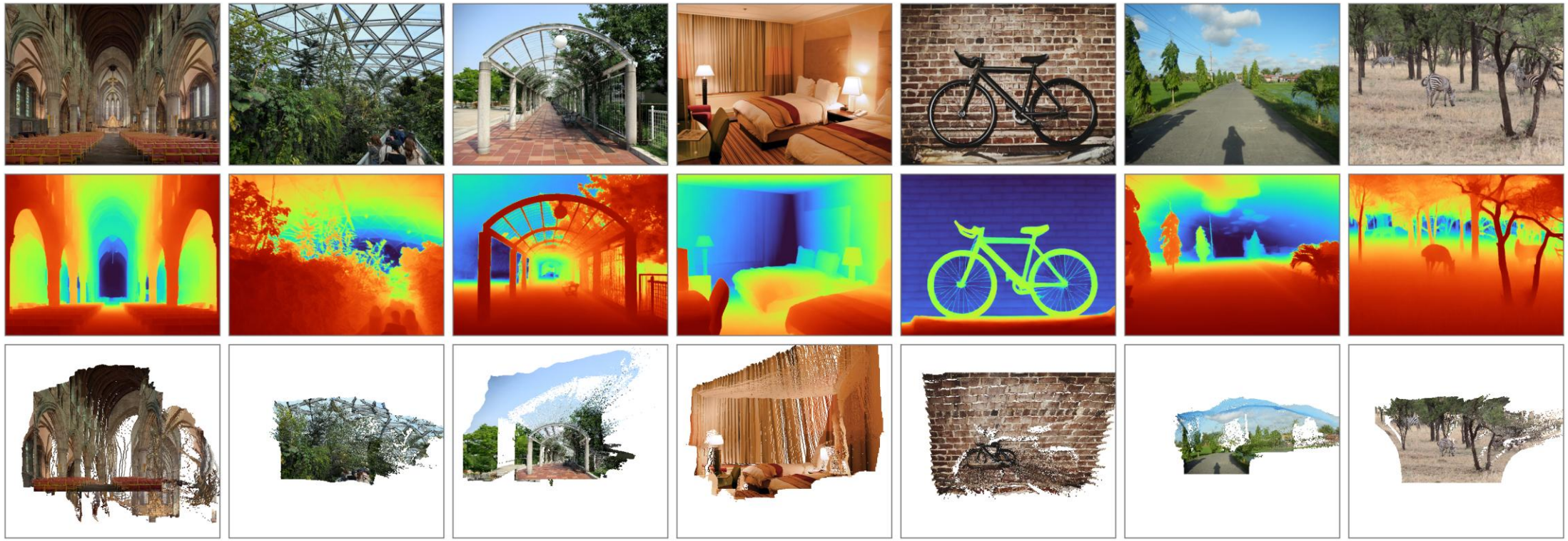


3D point cloud using pixel extrusion



Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C., & Schindler, K. (2024). Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2312.02145
Image Source: EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons, Output: depth estimation mask

MDE – Marigold (2023-24)



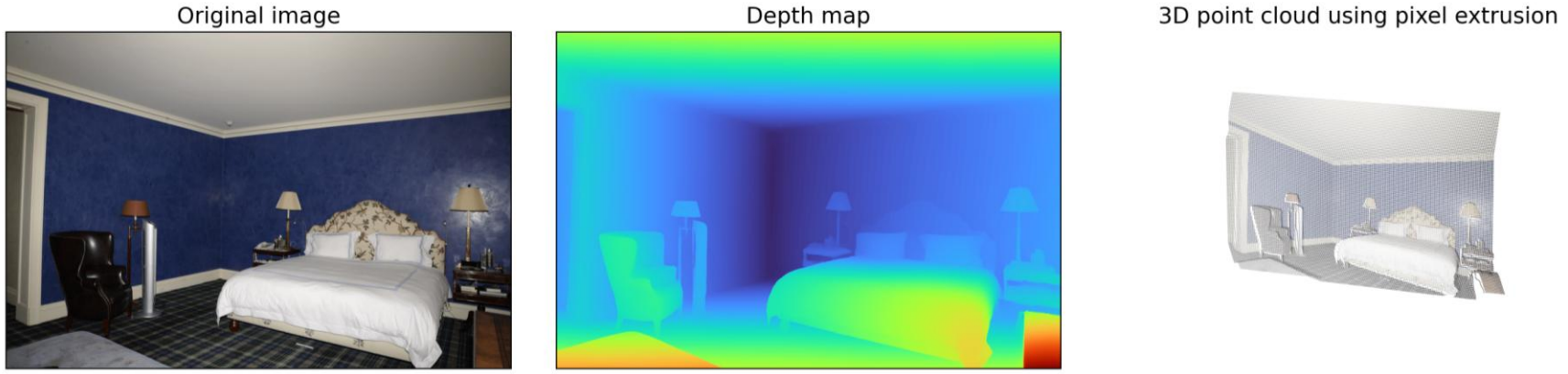
Marigold / Depth Anything V2 output - own model output. Photographs: Agness Abubakar Saidi (CC0); Darius Soodmand (CC0); Stephen Jacobs, CC BY 4.0; FBI (public domain), all via Wikimedia Commons.

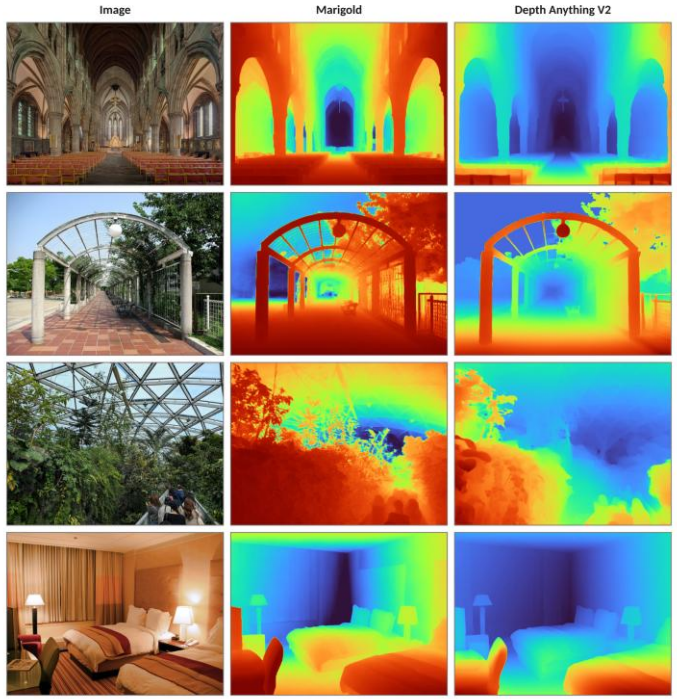
Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C., & Schindler, K. (2024). Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2312.02145, CC BY 4.0
Image Source: Marigold / Depth Anything V2 output - own model output. Photographs: Agness Abubakar Saidi (CC0); Darius Soodmand (CC0); Stephen Jacobs, CC BY 4.0; FBI (public domain), all via Wikimedia Commons.

MDE – Depth Anything v1-2 (2024-25)

A discriminative model utilizing a **Dense Prediction Transformer (DPT)** architecture with a DINOv2 backbone. It focuses on direct depth prediction from image features. The authors trained a reliable teacher model based on DINOv2-G purely on high-quality synthetic images, produced precise pseudo depths on real images and finally trained a student model on the pseudo-labeled real images.

- **CVPR 2024**
- **SOTA**





https://huggingface.co/spaces/depth-anything/Depth-Anything-V2

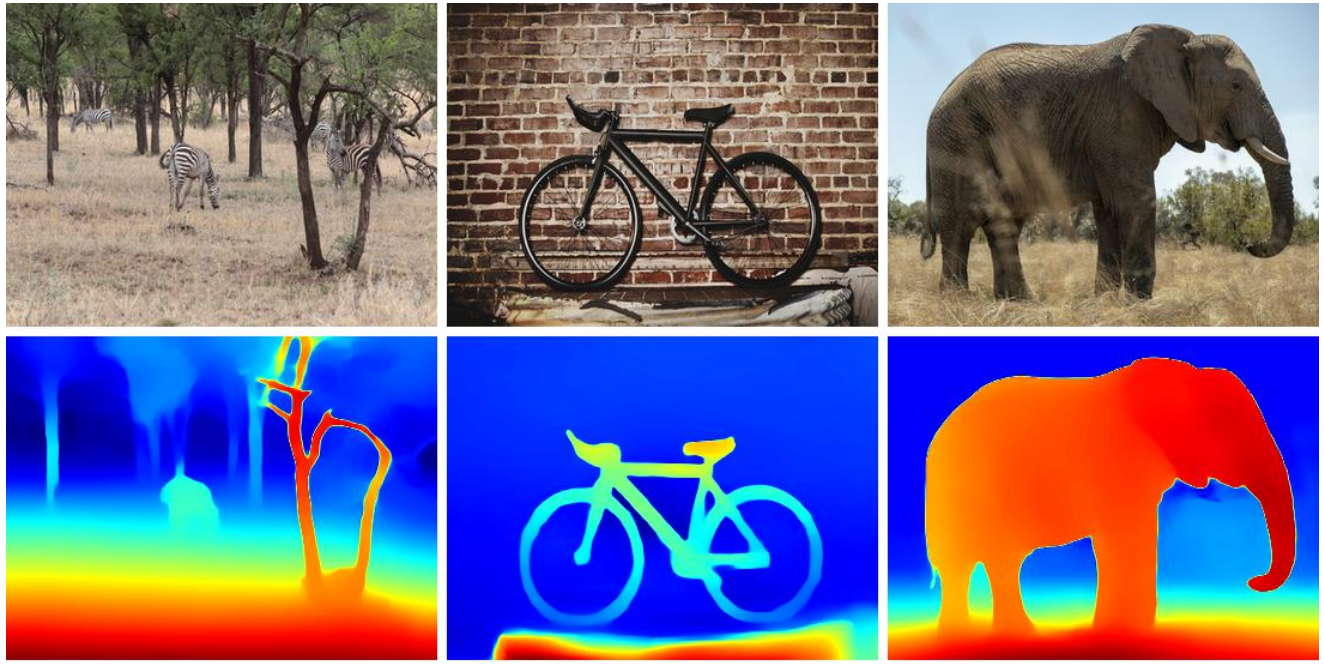
Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., & Zhao, H. (2024). Depth Anything V2. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2406.09414
Image Source: Marigold / Depth Anything V2 output - own model output. Photographs: Agness Abubakar Saidi (CC0); Darius Soodmand (CC0); Stephen Jacobs, CC BY 4.0; FBI (public domain), all via Wikimedia Commons.
Image Source: EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons, Output: depth estimation mask

MDE – Depth Pro (2025)

Depth Pro (Apple) foundation model for zero-shot metric monocular depth estimation. Synthesizes high-resolution depth maps with unparalleled sharpness and high-frequency details. The predictions are metric, with absolute scale, without relying on the availability of metadata such as camera intrinsics. Model is fast, producing a **2.25-megapixel depth map in 0.3 seconds on a standard GPU.**

Contains a training protocol that combines real and synthetic datasets to achieve high metric accuracy alongside fine boundary tracing

- **CVPR 2025**



https://github.com/apple/ml-depth-pro

Bochkovskii, A., Delaunoy, A., Germain, H., Santos, M., Zhou, Y., Richter, S. R., & Koltun, V. (2025). Depth Pro: Sharp Monocular Metric Depth in Less Than a Second. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2410.02073
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons
Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CC0, via Wikimedia Commons

Depth From Motion - Monocular Images (2023)

The paper *"Monocular 3D Object Detection with Depth from Motion"* presents a technique for estimating depth by leveraging motion in a sequence of monocular images (i.e., images from a single moving camera).

The model estimates depth by analyzing the apparent motion of objects between consecutive frames. When the camera moves, objects closer to the camera appear to move faster across the image than those farther away, providing cues about their relative depth.

Uses CNN backbone

https://github.com/Tai-Wang/Depth-from-Motion

Wang et al. (2022) - Monocular 3D Object Detection with Depth from Motion, arXiv:2207.12988, CC BY-NC-SA 4.0

Wang, T., Pang, J., & Lin, D. (2023). Monocular 3D Object Detection with Depth from Motion. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2207.12988

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 11

Deep Learning Tools For Computer Vision part 1

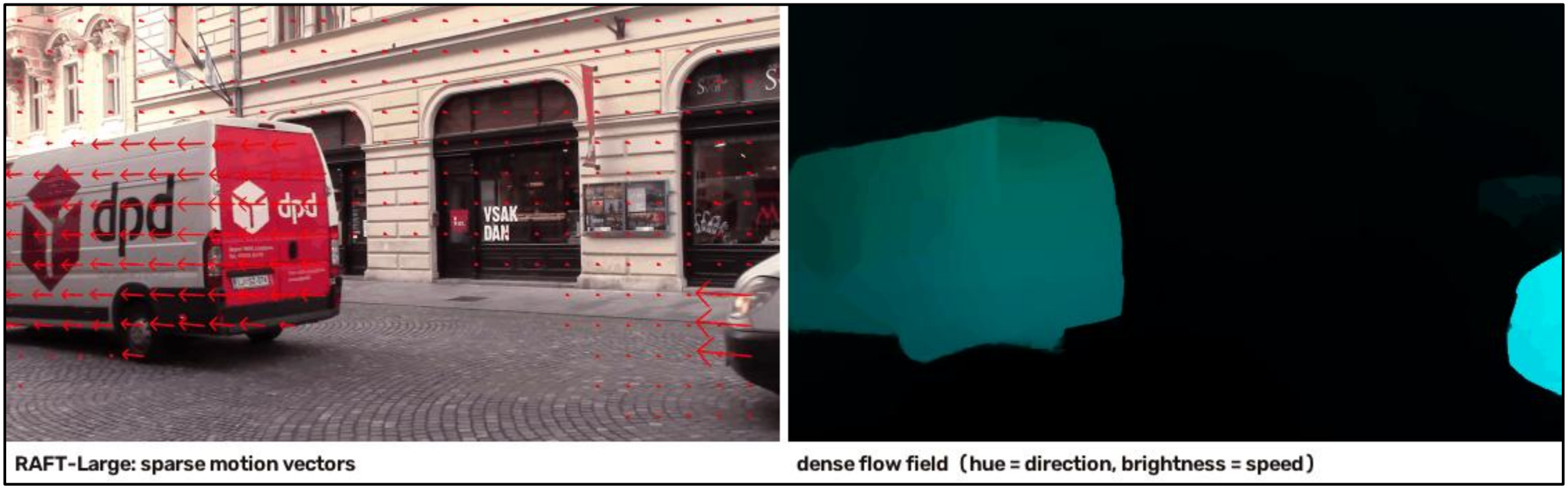
1 Annotation Tools

2 Depth Estimation

3 Optical Flow

Optical Flow

Optical flow is the motion of objects between consecutive frames of sequence, caused by the relative movement between the object and camera. Basically, it's about understanding how things are moving in an image.



Video Source: Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, Point clouds are our own model output

Optical Flow

The problem of optical flow may be expressed as:



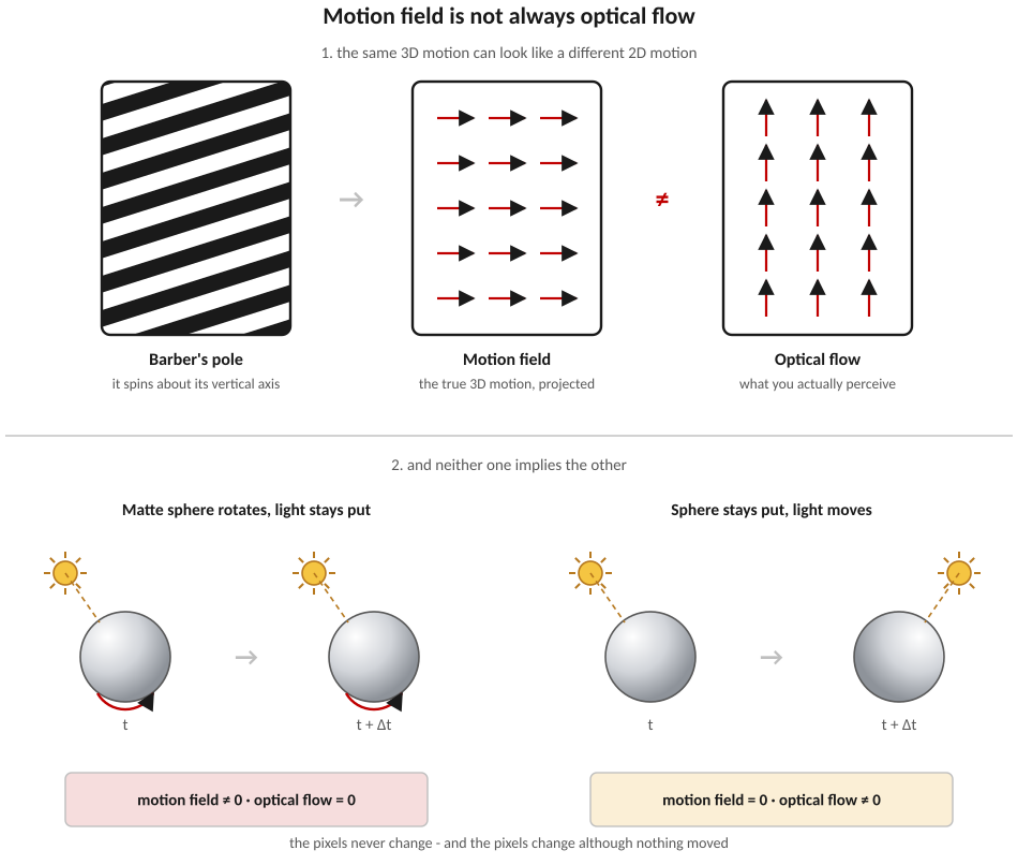
$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

brightness constancy: the patch keeps its intensity, only its position changes

Where between consecutive frames, we can express the image intensity (I) as a function of space (x, y) and time (t). In other words, if we take the first image $I(x, y, t)$ and move its pixels by (dx, dy) over time t, we obtain the new image $I(x + dx, y + dy, t + dt)$.

Optical Flow

Optical flow is the apparent motion of brightness patterns in the image. Generally, optical flow corresponds to the motion field, but not always. For example, the motion field and optical flow of a rotating barber's pole are different:



Optical flow is the **motion of each brightness pattern**. Conversely, the motion field represents the **motion of each pixel in the frame**.

In general, such cases are unusual, so assuming optical flow corresponds to the motion field is not unusual in literature.

A smooth sphere is rotating under constant illumination. Thus, the optical flow field is zero, but the motion field is not (no visible brightness change)

A fixed sphere is illuminated by a moving source and the shading of the image changes. Thus, the motion field is zero, but the optical flow field is not.

Optical Flow

Optical flow constraint equations describe the relationship between the motion of objects in a scene and the observed changes in image intensity over time. The most fundamental of these is the **Optical Flow Constraint Equation** (OFCE).

1. Brightness Constancy Assumption:

   This assumption states that the brightness (intensity) of a pixel remains constant over time as it moves across the image.

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

2. Optical Flow Constraint Equation:

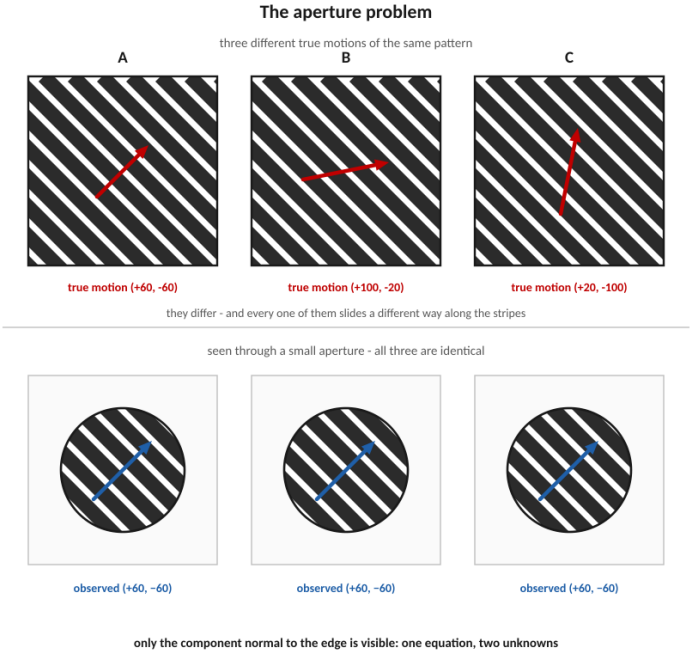
   By taking the first-order **Taylor expansion** of the brightness constancy assumption and assuming small movements, we get:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{dy}v + \frac{\partial I}{\partial t} = 0$$

   It provides **one constraint per pixel but has two unknowns** (u and v, horizontal and vertical motion), leading to what is known as the **aperture problem**. (one eq., two unknowns -> infinite solutions)

3. Displacement Size:

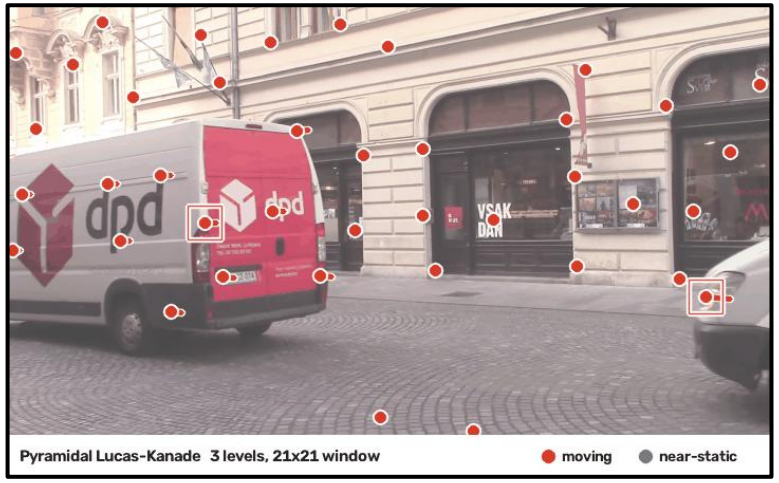
   Displacement $(\partial x, \partial y)$ and time step ∂t are small.



Optical Flow – Lucas-Kanade Solution

A classic algorithm for optical flow estimation that mitigates the aperture problem by analyzing a local neighborhood of pixels to infer motion. It is based on the **Optical Flow Constraint Equation** (OFCE)

1. Instead of estimating motion at each pixel independently, the Lucas-Kanade method considers a small neighborhood (e.g., a 5x5 or 7x7 window) around each pixel. By assuming that all pixels within this window have the same motion vector (same u and v), the method gets multiple equations from this local patch, helping to constrain the problem.
2. With multiple pixels in the neighborhood, the method builds a system of linear equations: $Av = b$, where A is a matrix of spatial gradients of all pixels in neighborhood, v is the optical flow vector we want to solve for and b contains the temporal gradients of each pixel.
3. If there are enough pixels in the neighborhood (more equations than unknowns), **the system can be solved using least squares to minimize the error.**



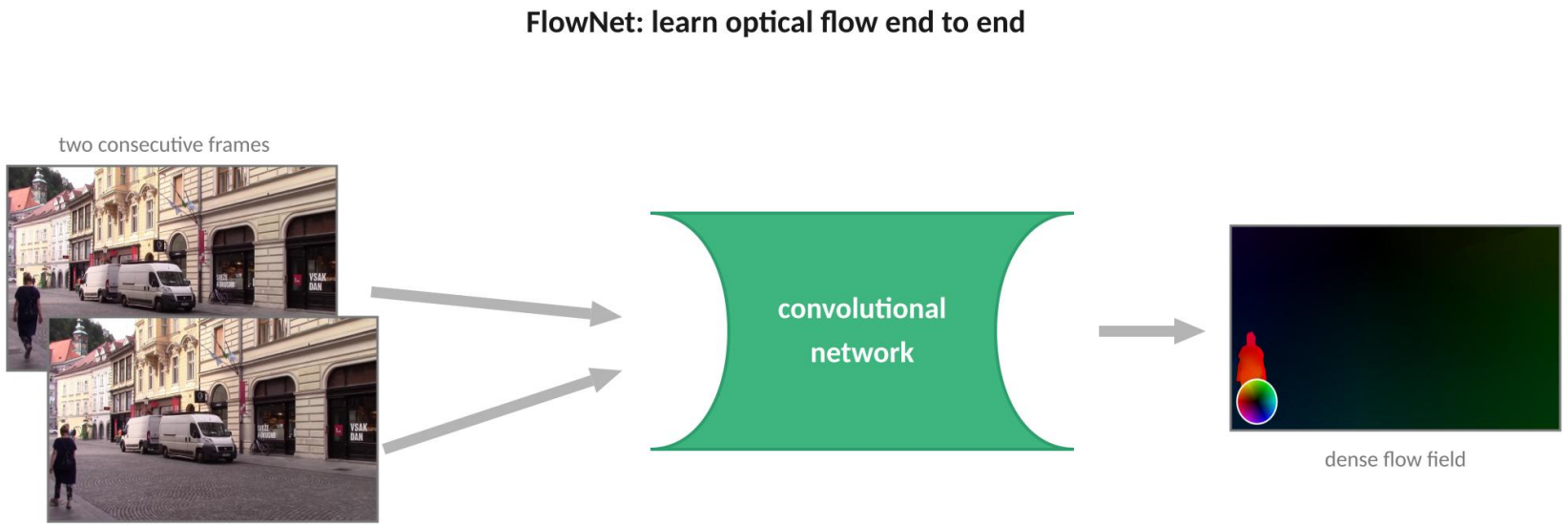
Sounds of Changes - Road traffic on Stritarjeva street, CC BY 3.0, via Wikimedia Commons.

Video Source: Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, Optical Flow outputs are our own outputs

Optical Flow – FlowNet (2015)

In the seminal paper *"FlowNet: Learning Optical Flow with Convolutional Networks"* , **FlowNet** introduces a CNN-based architecture designed to learn optical flow from synthetic data.

FlowNet has two main variants: **FlowNetS** and **FlowNetC**, each with unique approaches to processing image pairs.

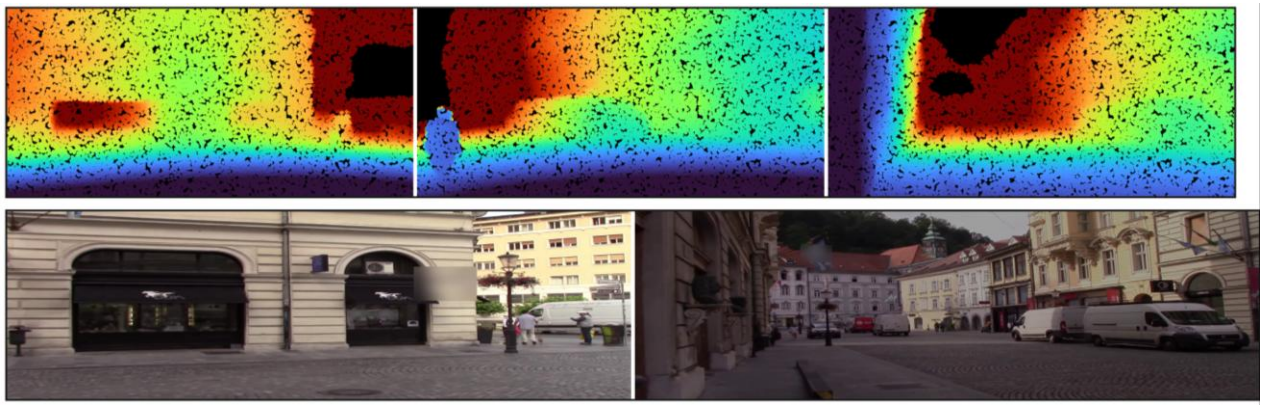


Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., ... Brox, T. (2015). FlowNet: Learning Optical Flow with Convolutional Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1504.06852
Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons

Benchmark Datasets



Sintel



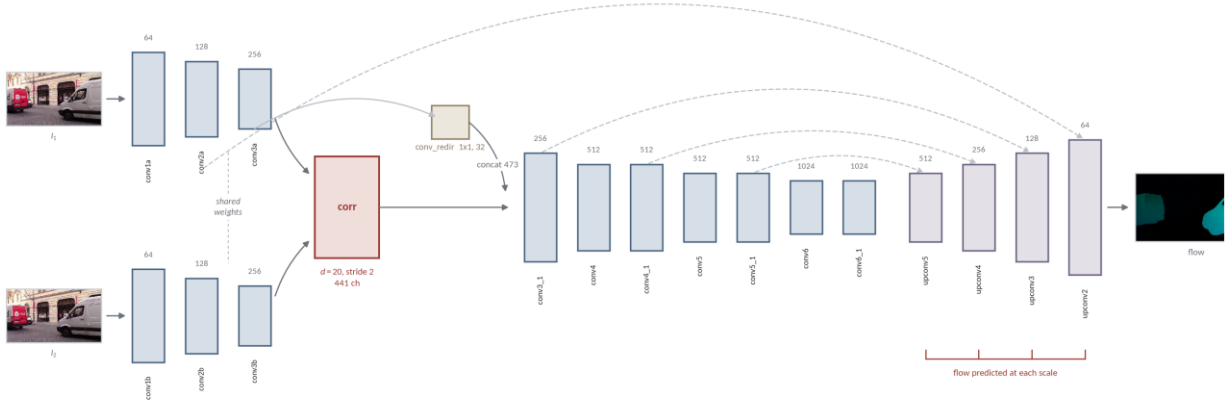
Kitti

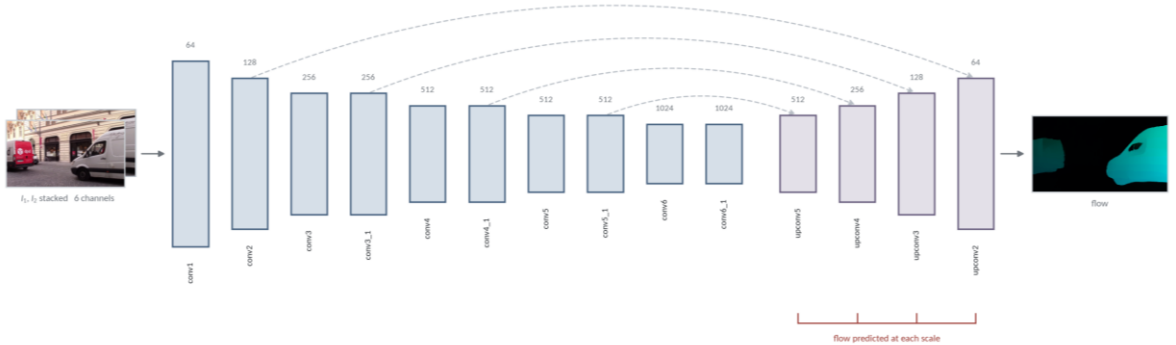
Sintel - © Blender Foundation, sintel.org, CC BY 3.0. Optical-flow ground truth from the MPI Sintel dataset (Butler et al., ECCV 2012).
Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, Point clouds are our own model output

Optical Flow – FlowNet (2015)

For **FlowNetS**, The two input images (frames) are concatenated along the channel dimension, creating a 6-channel input. The concatenated image pair is passed through several convolutional layers to extract hierarchical features. The decoder upsamples the features back to the original image resolution. At each upsampling stage, the network produces intermediate flow estimates, which are refined as features are upsampled.

FlowNetC adds a correlation layer to compute explicit motion information between the two images

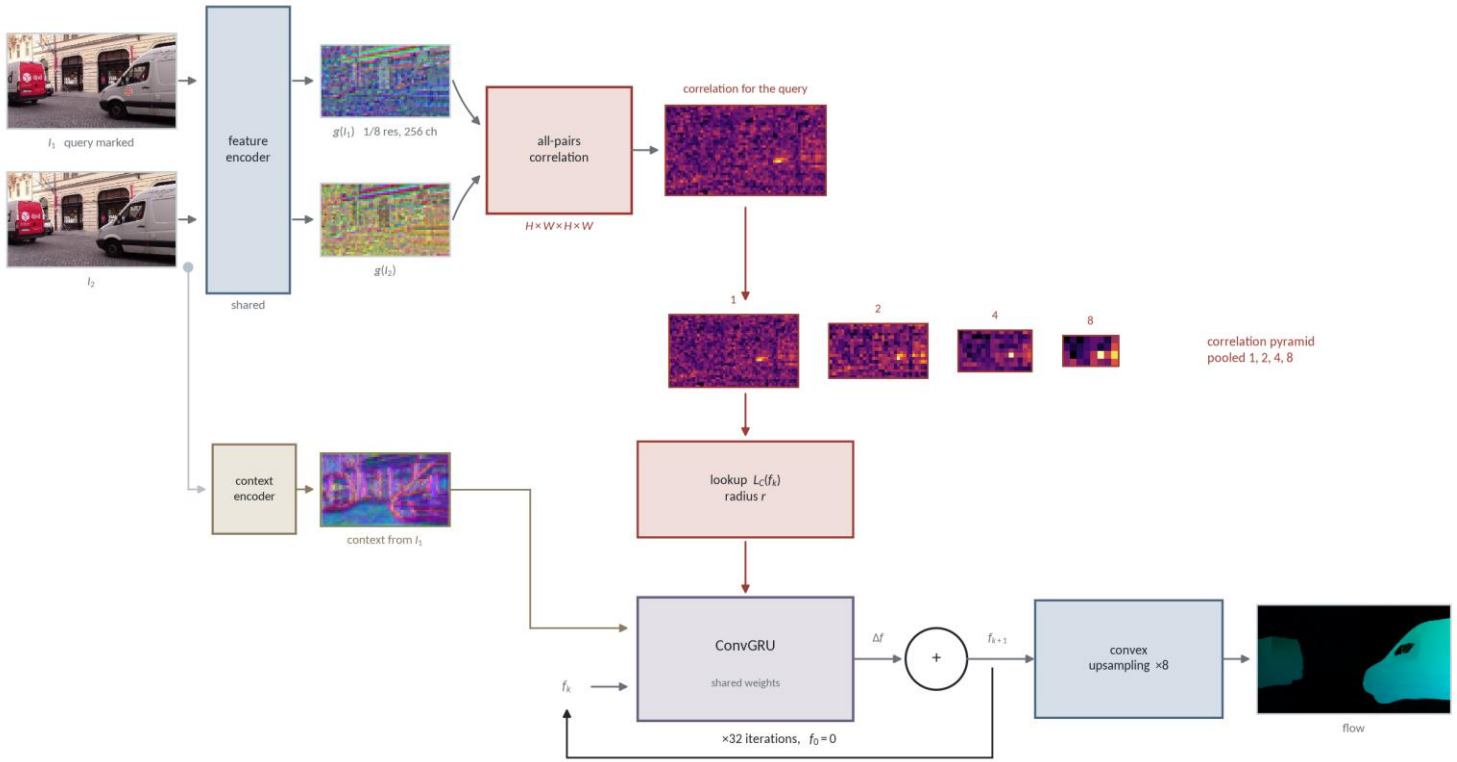




Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., ... Brox, T. (2015). FlowNet: Learning Optical Flow with Convolutional Networks. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1504.06852
Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons,

Optical Flow – RAFT (2020)

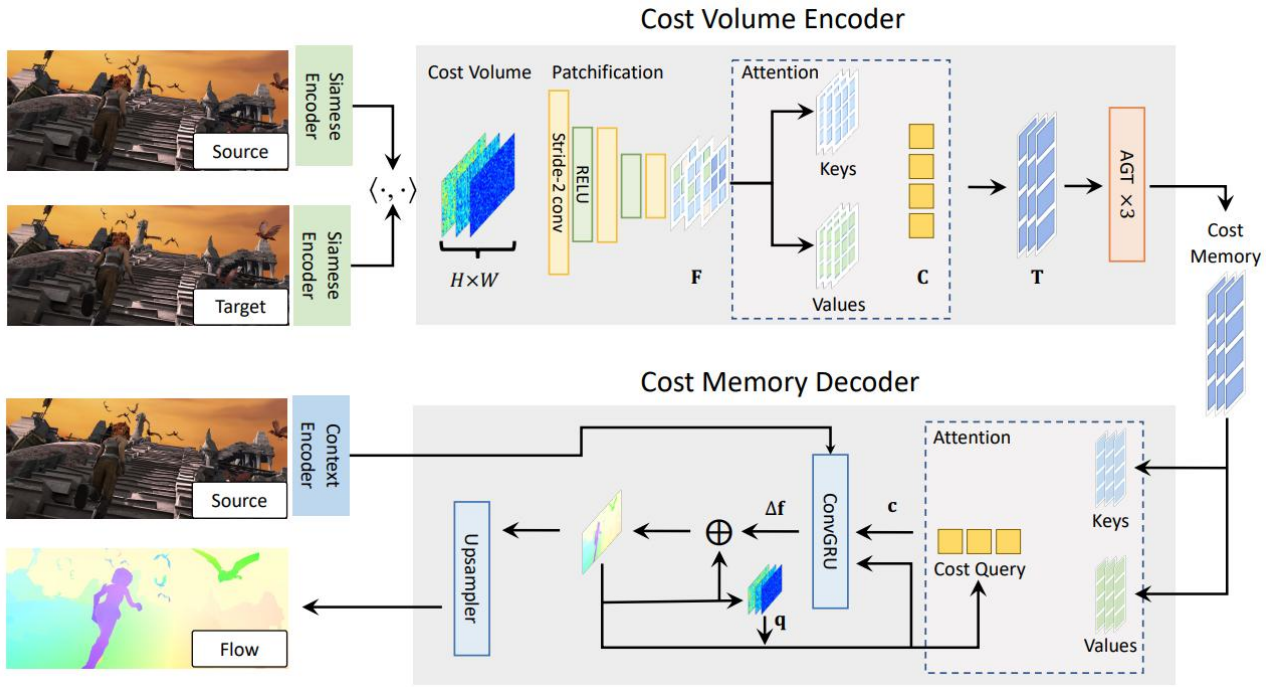
- **RAFT** combines CNN and RNN architectures and is still considered state-of-the-art.
- Like the previous architecture, it has two versions: **RAFT** and RAFT-S, with **RAFT-S** being a lightweight variant of the original RAFT model.
- **RAFT** consists of a feature encoder, a context encoder, a visual similarity module, and an iterative update module that utilizes RNNs.



Teed, Z., & Deng, J. (2020). RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2003.12039
Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, Point clouds are our own model output

Optical Flow – FlowFormer (++) (2022-23)

- **FlowFormer** introduces a transformer architecture into optical flow estimation and achieves state-of-the-art performance. The core component of **FlowFormer** is the transformer-based cost-volume encoder.
- Utilizes the success of **masked autoencoding (MAE) pretraining** in unleashing transformers' capacity of encoding visual representation
- **FlowFormer** tokenizes the 4D cost volume derived from image pairs and processes these tokens using alternate-group transformer layers to encode cost memory. A recurrent transformer decoder then refines the flow estimates using dynamic positional cost queries.



Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K. C., Qin, H., ... Li, H. (2022). FlowFormer: A Transformer Architecture for Optical Flow. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2203.16194
Image Source: Fig. 1: Architecture of FlowFormer — Huang et al., FlowFormer: A Transformer Architecture for Optical Flow (2022), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/ , https://arxiv.org/abs/2203.16194

Optical Flow – FlowFormer (++) (2022-23)



(a) Input (b) FlowFormer (Ours)

Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K. C., Qin, H., ... Li, H. (2022). FlowFormer: A Transformer Architecture for Optical Flow. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2203.16194
Image Source: Fig. 3: Qualitative comparison on the Sintel test set — Huang et al., FlowFormer: A Transformer Architecture for Optical Flow (2022), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/ , https://arxiv.org/abs/2203.16194 ; Sintel imagery © Blender Foundation, CC BY 3.0 https://creativecommons.org/licenses/by/3.0/

Optical Flow – VideoFlow (2023)

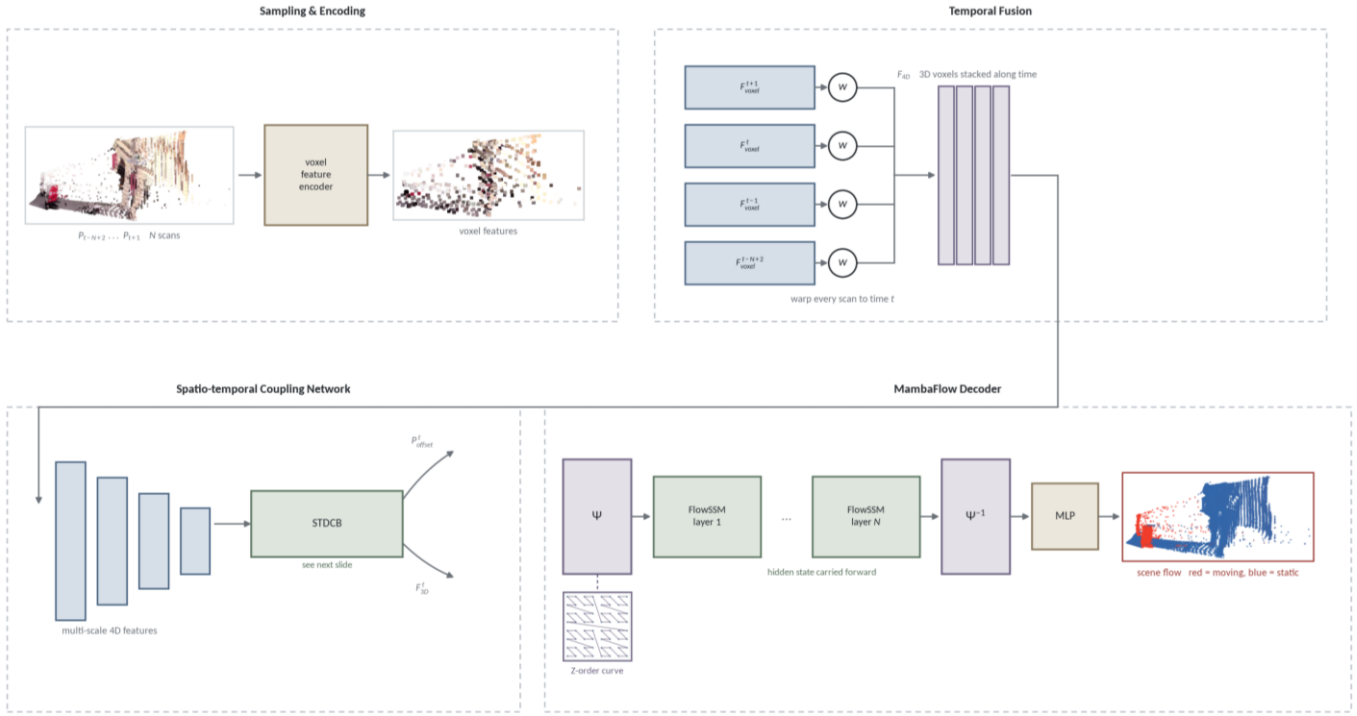


Sounds of Changes - Road traffic on Stritarjeva street, CC BY 3.0, via Wikimedia Commons.

Shi, X., Huang, Z., Bian, W., Li, D., Zhang, M., Cheung, K. C., ... Li, H. (2023). VideoFlow: Exploiting Temporal Cues for Multi-frame Optical Flow Estimation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2303.08340
Video Source: Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons,

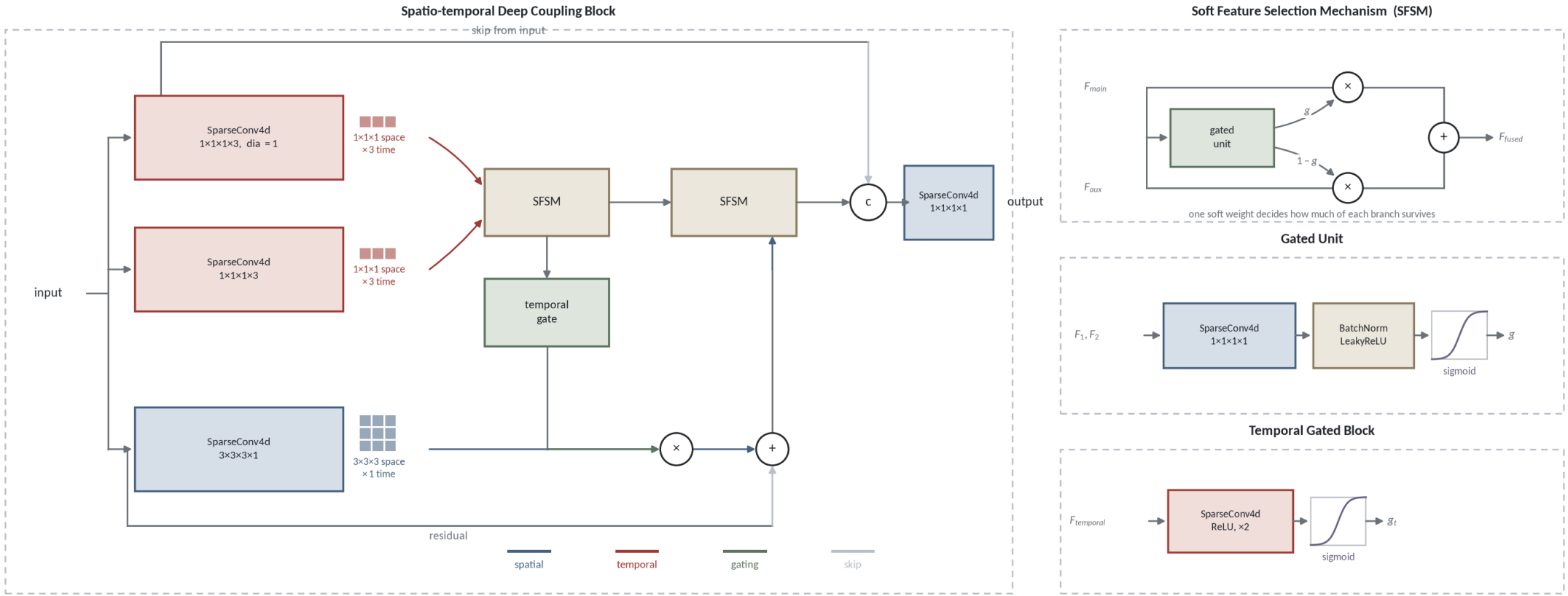
Optical Flow – MambaFlow (2025)

- **MambaFlow was designed to address the limitations of Transformer-based methods, particularly their high computational complexity**
- Utilizes the **Mamba** architecture, a state space model known for its linear computational complexity, **MambaFlow** achieves efficient and accurate motion estimation between consecutive video frames.
- Addresses occlusion challenges by effectively propagating flow information. Utilizes feature similarity to guide flow estimation in regions with missing data.



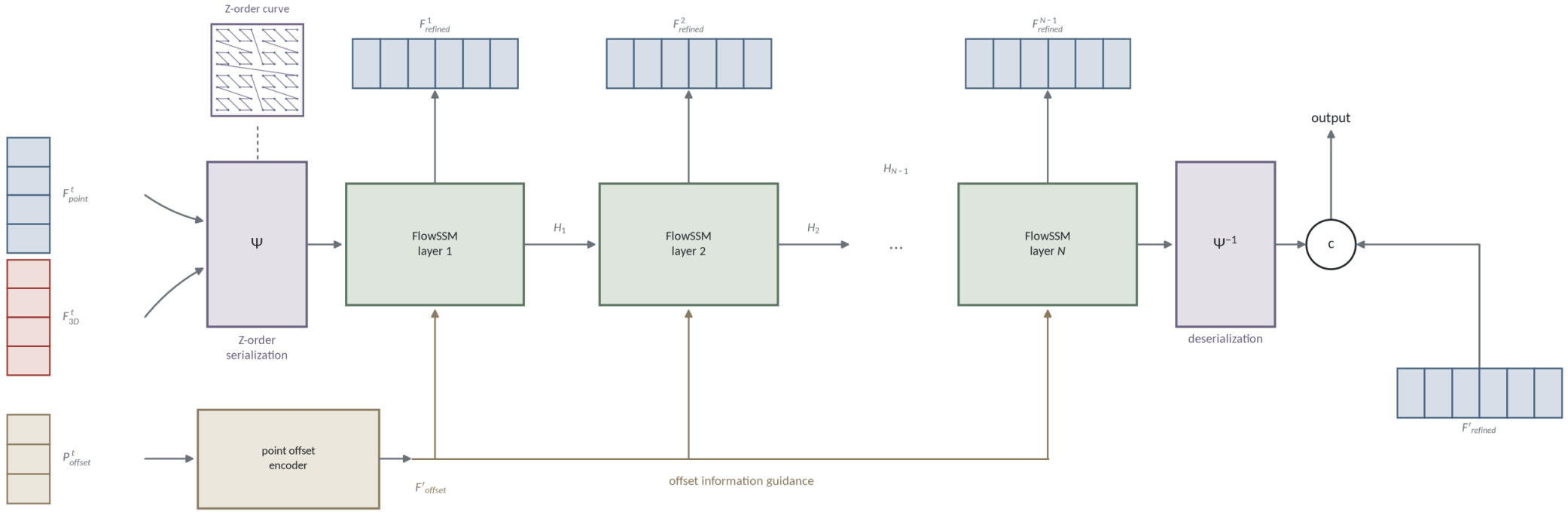
Luo, J., Cheng, J., Tang, X., Zhang, Q., Xue, B., & Fan, R. (2025). MambaFlow: A Novel and Flow-guided State Space Model for Scene Flow Estimation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2502.16907

Optical Flow – MambaFlow (2025)



Luo, J., Cheng, J., Tang, X., Zhang, Q., Xue, B., & Fan, R. (2025). MambaFlow: A Novel and Flow-guided State Space Model for Scene Flow Estimation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2502.16907

Optical Flow – MambaFlow (2025)



Luo, J., Cheng, J., Tang, X., Zhang, Q., Xue, B., & Fan, R. (2025). MambaFlow: A Novel and Flow-guided State Space Model for Scene Flow Estimation. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2502.16907

Summary

- Various **model-assisted annotation tools** can aid in the manual labeling of extensive datasets.
  - Tools like **CVAT** offer free services for individuals and open-source research, while automatic annotation tools can handle a wide range of data types, including images, **MRI**, and **DICOM** formats.

- In deep learning, **depth estimation** is the process of predicting the distance of objects from the camera using a neural network.
  - **Non-linear estimators** are important because they can replace expensive hardware with advanced software solutions that perform similarly.
  - **Stereo depth estimation** has long relied on epipolar geometry from stereo images, while **monocular** estimation offers a cheaper and more compact solution by leveraging the **power of CNNs and Transformers**.

- **Optical flow** refers to the motion of objects between consecutive frames in a sequence, resulting from relative movement between the object and the camera.
  - The **Lucas-Kanade** method provides a solution to the aperture problem (where there are more unknowns than constraints).
  - Neural networks, like **FlowNet**, **RAFT**, **FlowFormer** and **MambaFlow** can be used as approximators for flow vectors using **CNNs**.

Resources

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning
- https://developers.google.com/machine-learning/crash-course/embeddings

Papers

1. Bochkovskii, A., Delaunoy, A., Germain, H., Santos, M., Zhou, Y., Richter, S. R., & Koltun, V. (2024). Depth Pro: Sharp monocular metric depth in less than a second. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2410.02073
2. Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1406.2283
3. Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., van der Smagt, P., Cremers, D., & Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1504.06852
4. Godard, C., Mac Aodha, O., & Brostow, G. J. (2016). Unsupervised monocular depth estimation with left-right consistency. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1609.03677
5. Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K. C., Qin, H., Dai, J., & Li, H. (2022). FlowFormer: A Transformer Architecture for Optical Flow. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2203.16194
6. Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C., & Schindler, K. (2023). Repurposing diffusion-based image generators for monocular depth estimation. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2312.02145
7. Lin, G., Milan, A., Shen, C., & Reid, I. (2016). RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.1611.06612

Papers

8. Luo, J., Cheng, J., Tang, X., Zhang, Q., Xue, B., & Fan, R. (2025). MambaFlow: A novel and flow-guided state space model for scene flow estimation. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2502.16907
9. Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision Transformers for Dense Prediction. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2103.13413
10. Shi, X., Huang, Z., Bian, W., Li, D., Zhang, M., Cheung, K. C., See, S., Qin, H., Dai, J., & Li, H. (2023). VideoFlow: Exploiting temporal cues for multi-frame Optical Flow estimation. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2303.08340
11. Song, S., Lichtenberg, S. P., & Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 567–576. https://doi.org/10.1109/cvpr.2015.7298655
12. Teed, Z., & Deng, J. (2020). RAFT: Recurrent All-Pairs Field Transforms for optical flow. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2003.12039
13. Wang, T., Pang, J., & Lin, D. (2022). Monocular 3D object detection with depth from motion. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2207.12988
14. Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., & Zhao, H. (2024). Depth Anything V2. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2406.09414

Further Links + Resources

- The State of the Art of Depth Estimation from Single Images - https://medium.com/@patriciogv/the-state-of-the-art-of-depth-estimation-from-single-images-9e245d51a315
- How to Estimate Depth from a Single Image - https://towardsdatascience.com/how-to-estimate-depth-from-a-single-image-7f421d86b22d
- Monocular Depth Estimation using ZoeDepth : Our Experience - https://medium.com/@bhaskarbose1998/monocular-depth-estimation-using-zoedepth-our-experience-42fa5974cb59
- Epipolar Geometry - https://amroamroamro.github.io/mexopencv/opencv/epipolar_geometry_demo.html
- Optical Flow - https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html
- RAFT: Optical Flow estimation using Deep Learning - https://learnopencv.com/optical-flow-using-deep-learning-raft
- FlowFormer: A Transformer Architecture for Optical Flow - https://github.com/drinkingcoder/FlowFormer-Official
- Depth Pro: Sharp Monocular Metric Depth in Less Than a Second - https://github.com/apple/ml-depth-pro?tab=readme-ov-file
- Dept Anything V2 - https://github.com/DepthAnything/Depth-Anything-V2

Image Sources

- Unless otherwise noted, the figures were created for this course.
- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.
- Dancing Robot: Gif robot.git, Meks01, CC BY-SA 4.0, via Wikimedia Commons.
- Figures built on Share-Alike source images are themselves released under CC BY-SA 4.0.
- All depth maps, optical flow fields, feature maps, correlation volumes, point clouds, detections, segmentations and attention maps in this presentation are our own model output.

Image Sources

- Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons
- Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CC0, via Wikimedia Commons
- Fig. 1: DETR architecture - Carion et al., End-to-End Object Detection with Transformers (2020), CC0 1.0 https://creativecommons.org/publicdomain/zero/1.0/, https://arxiv.org/abs/2005.12872
- Strand street pedestrian crossing 3.jpg, Mission for Inner City Cape Town, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons
- https://commons.wikimedia.org/wiki/Category:Paddy_fields,_grasslands,_trees_and_irrigation_(Balante_Proper_%26_Balante_Boulevard,_Gapan_City), Judgefloro - Balante Boulevard, Gapan City, CC0, via Wikimedia Commons
- Sounds of Changes, Road traffic on Stritarjeva street, CC BY 3.0; HRowntree, CC0; Darius Soodmand, CC0, via Wikimedia Commons
- EFTA00000247 - Hotel room with deep blue walls a plaid carpet and a large bed with floral headboard Two lamps on nightstands flank the bed A leather armchair sits near a tall floor fan.jpg, Federal Bureau of Investigation, Public domain, via Wikimedia Commons
- Hotel-room-renaissance-columbus-ohio.jpg, Derek Jensen (Tysto), Public domain, via Wikimedia Commons
- DIODE — Vasiljevic et al., DIODE: A Dense Indoor and Outdoor DEpth Dataset (2019), MIT License, Copyright © 2021 diode-dataset, https://diode-dataset.org/
- Sintel — © Blender Foundation, sintel.org, CC BY 3.0 https://creativecommons.org/licenses/by/3.0/; optical-flow ground truth from the MPI Sintel dataset, Butler et al., ECCV 2012, https://sintel.is.tue.mpg.de/

Image Sources (cont. 2)

Image Sources (cont.)

- Marigold / Depth Anything V2 output - own model output. Photographs: Agness Abubakar Saidi (CC0); Darius Soodmand (CC0); Stephen Jacobs, CC BY 4.0; FBI (public domain), all via Wikimedia Commons
- Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons
- Fig. 1: Architecture of FlowFormer — Huang et al., FlowFormer: A Transformer Architecture for Optical Flow (2022), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2203.16194
- Fig. 3: Qualitative comparison on the Sintel test set — Huang et al., FlowFormer: A Transformer Architecture for Optical Flow (2022), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/, https://arxiv.org/abs/2203.16194; Sintel imagery © Blender Foundation, CC BY 3.0 https://creativecommons.org/licenses/by/3.0/

Video Sources

- Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons

Legal Notice

CVAT — Computer Vision Annotation Tool
Source: https://github.com/cvat-ai/cvat
License: MIT License

Copyright (C) 2018-2022 Intel Corporation
Copyright (C) 2022-2025 CVAT.ai Corporation

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Legal Notice

DIODE / diode-devkit
Source: https://github.com/diode-dataset/diode-devkit
License: MIT License

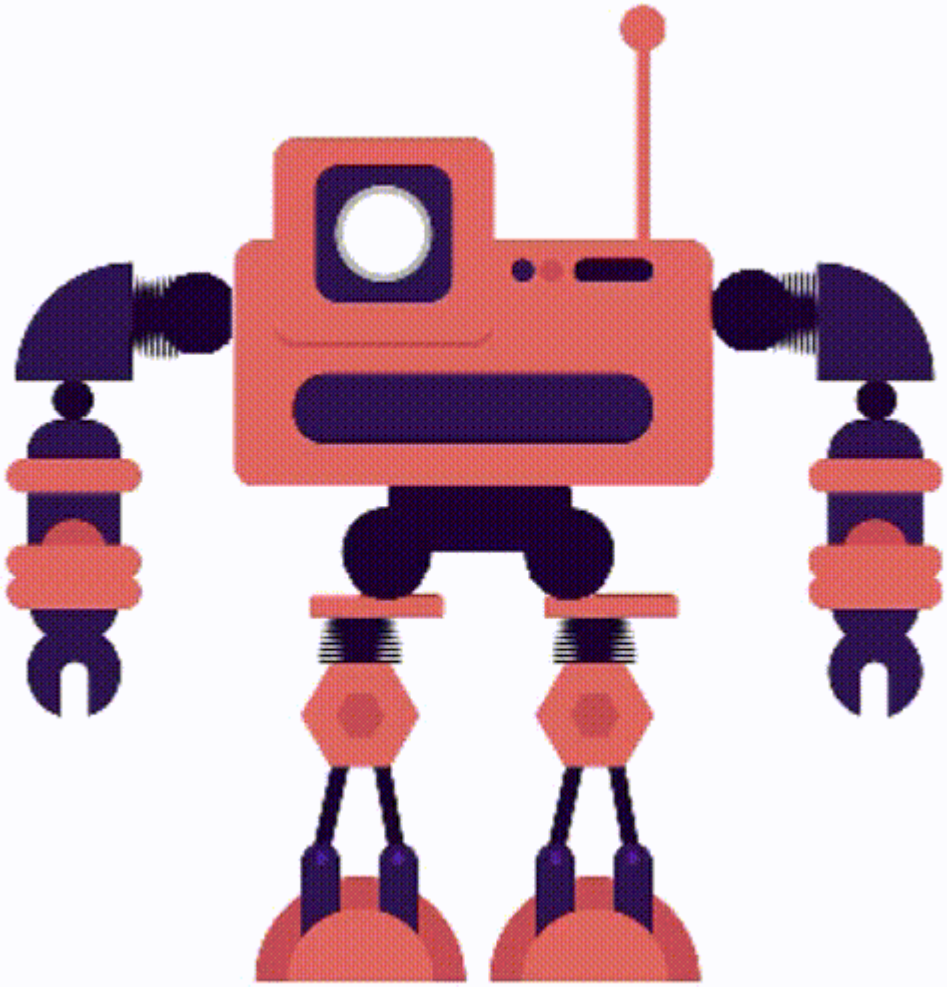
Copyright (c) 2021 diode-dataset

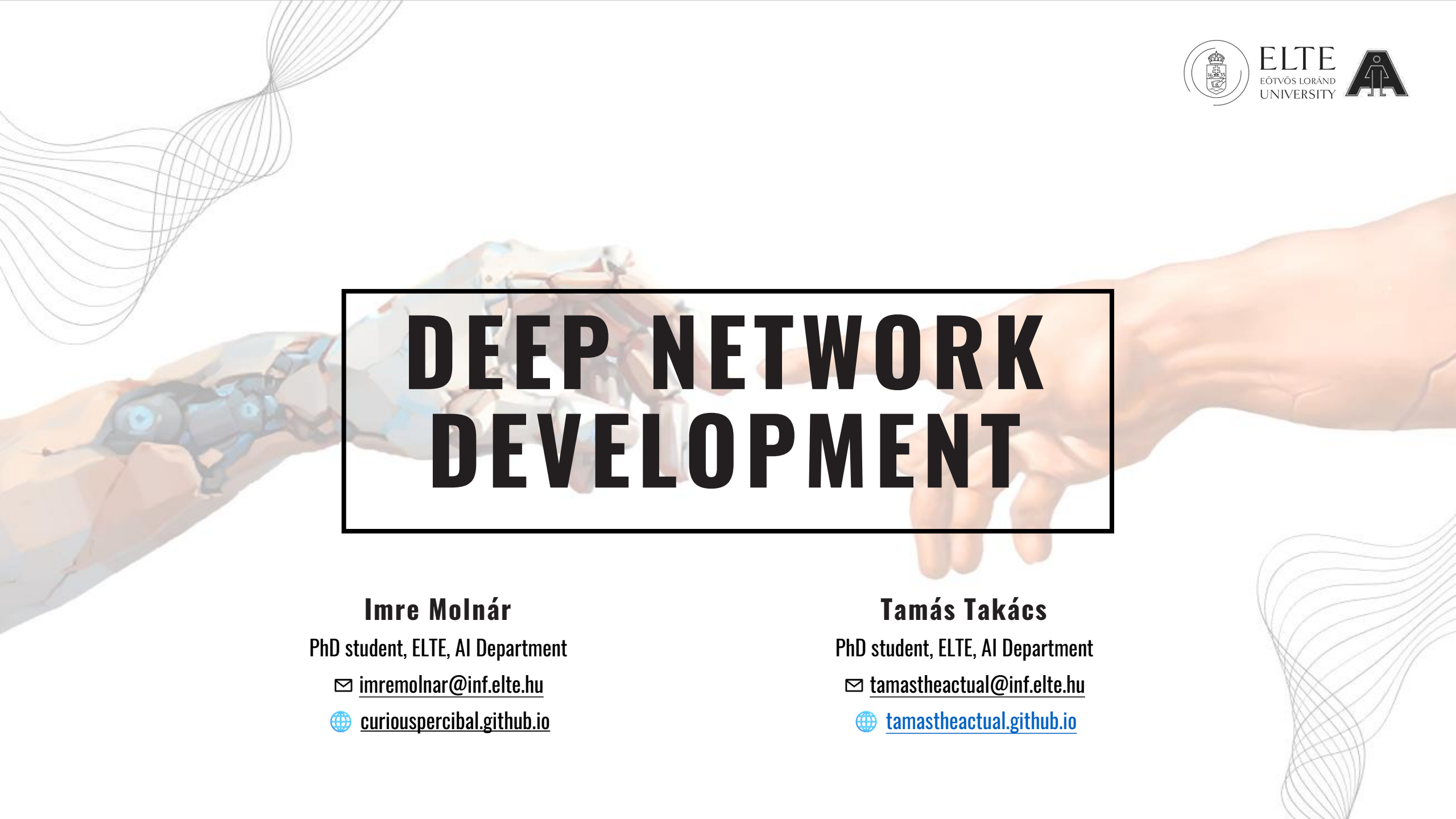
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

That’s all for today!





DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

Tamás Takács

PhD student, ELTE, AI Department

✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 12

Deep Learning Tools For Computer Vision

1 Image Inpainting

2 Generative Modeling

3 Neural Rendering

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 12

Deep Learning Tools For Computer Vision

1 Image Inpainting

2 Generative Modeling

3 Neural Rendering

Image Inpainting

Image inpainting is about filling in missing or occluded regions in digital images, aiming to restore plausible, realistic content. Its applications range from cultural relic restoration to virtual scene editing and film production.



Image Source: 06-Pont Notre-Dame sur le Lot à Mende (Alfred-Nicolas Normand, 1888).jpg, Alfred-Nicolas Normand, Public domain, via Wikimedia Commons, Restoration: own output (LaMa).
Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 https://creativecommons.org/licenses/by/3.0 , via Wikimedia Commons

Image Inpainting

Image inpainting is about filling in missing or occluded regions in digital images, aiming to restore plausible, realistic content. Its applications range from cultural relic restoration to virtual scene editing and film production.

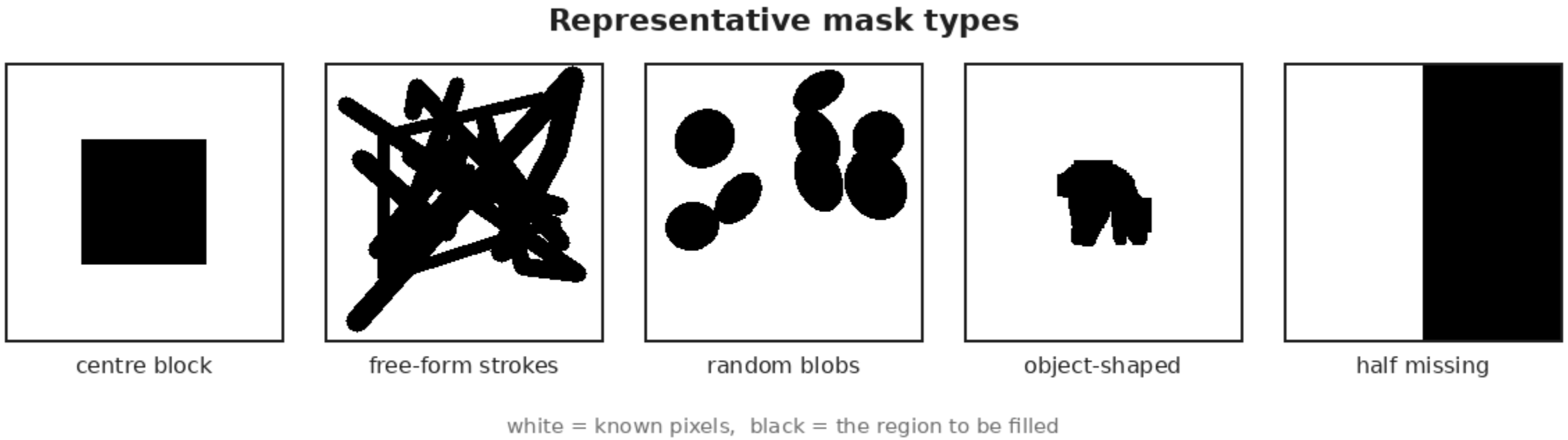


Image Inpainting

Adobe Photoshop image expanding

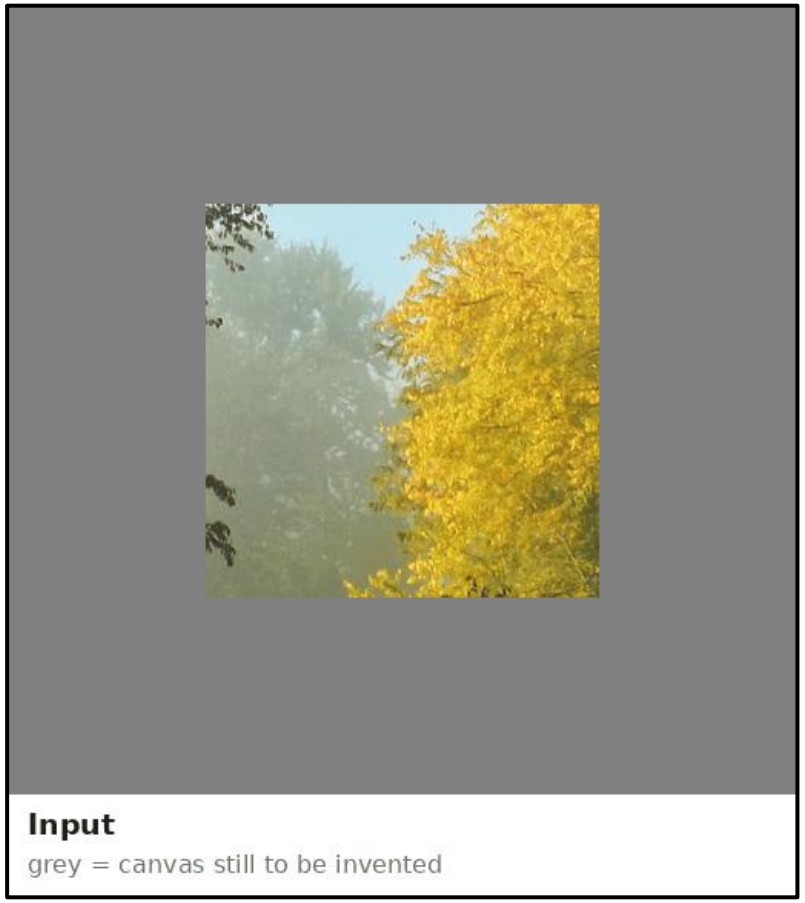
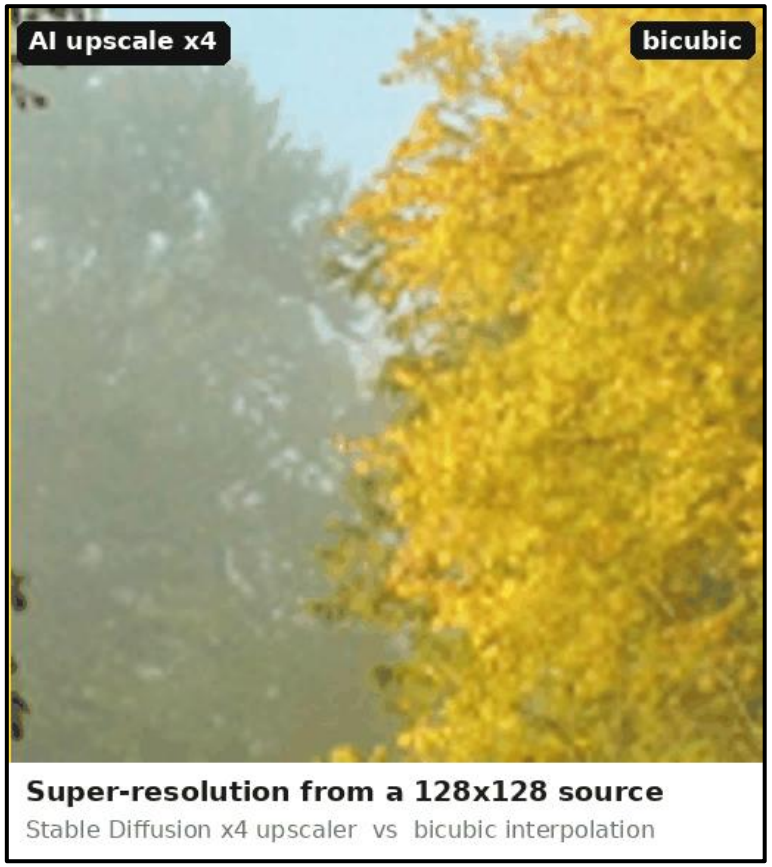


Image Inpainting

Topaz Video enhance AI (super resolution)



Super-resolution from a 128x128 source
Stable Diffusion x4 upscaler vs bicubic interpolation



Bicubic upscale
128 -> 512



AI upscale x4
128 -> 512

Image Inpainting

Object removal and Text Image editing

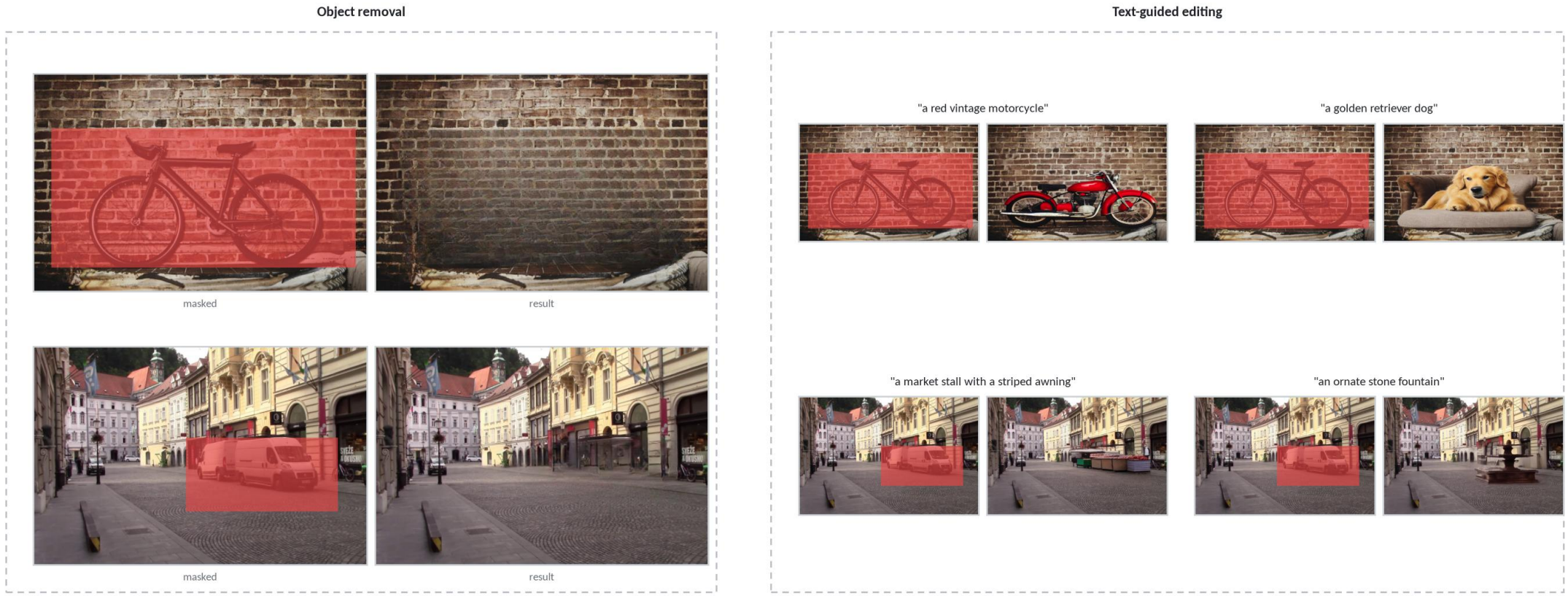


Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons
Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons,

Traditional Methods

Diffusion Based methods

Pros: Good at handling small gaps, extending smooth areas, and completing continuous structures like edges and curves.

Cons: Struggles with large texture areas, as it tends to blur the regions being filled, making it unsuitable for complex textures



(a) input with the region removed (b) diffusion-based fill

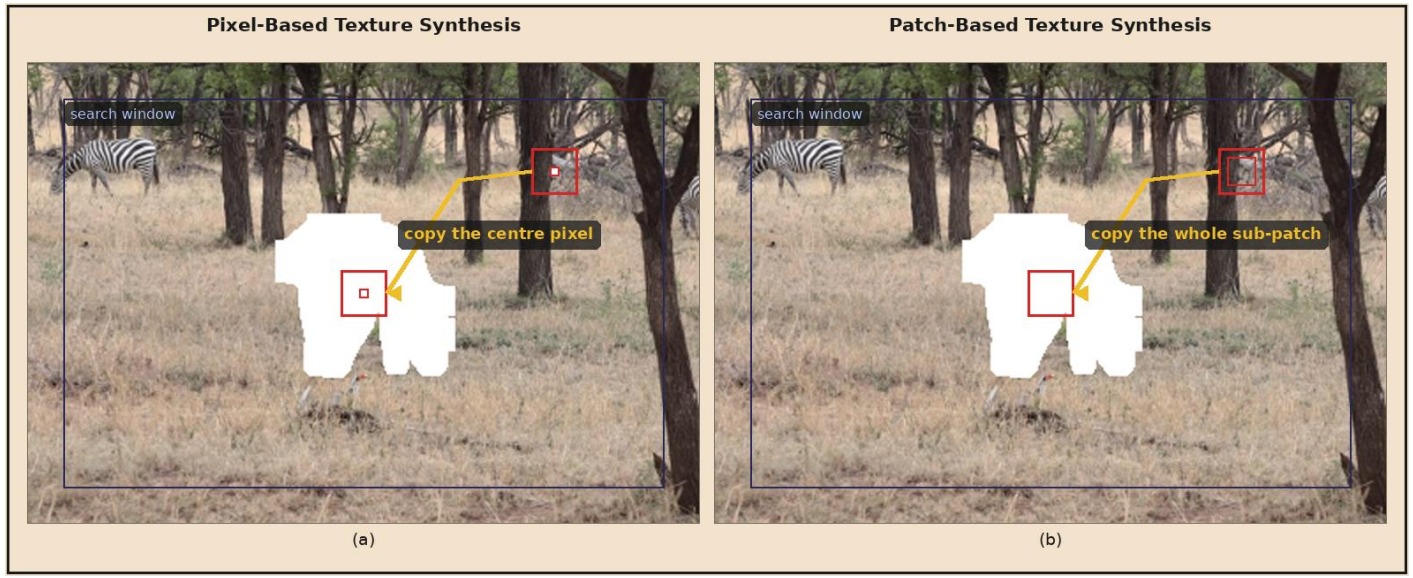
Guillemot, C., & Le Meur, O. (2014). Image inpainting : Overview and recent advances. IEEE Signal Processing Magazine, 31(1), 127–144. https://doi.org/10.1109/msp.2013.2273004
Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CC0, via Wikimedia Commons

Traditional Methods

Exemplar-Based Inpainting

Pros: Capable of filling large textured areas by copying similar patches from known regions. It leverages self-similarity to reproduce textures accurately.

Cons: Computationally intensive and may produce repetitive patterns or "texture garbage" when handling stochastic textures.



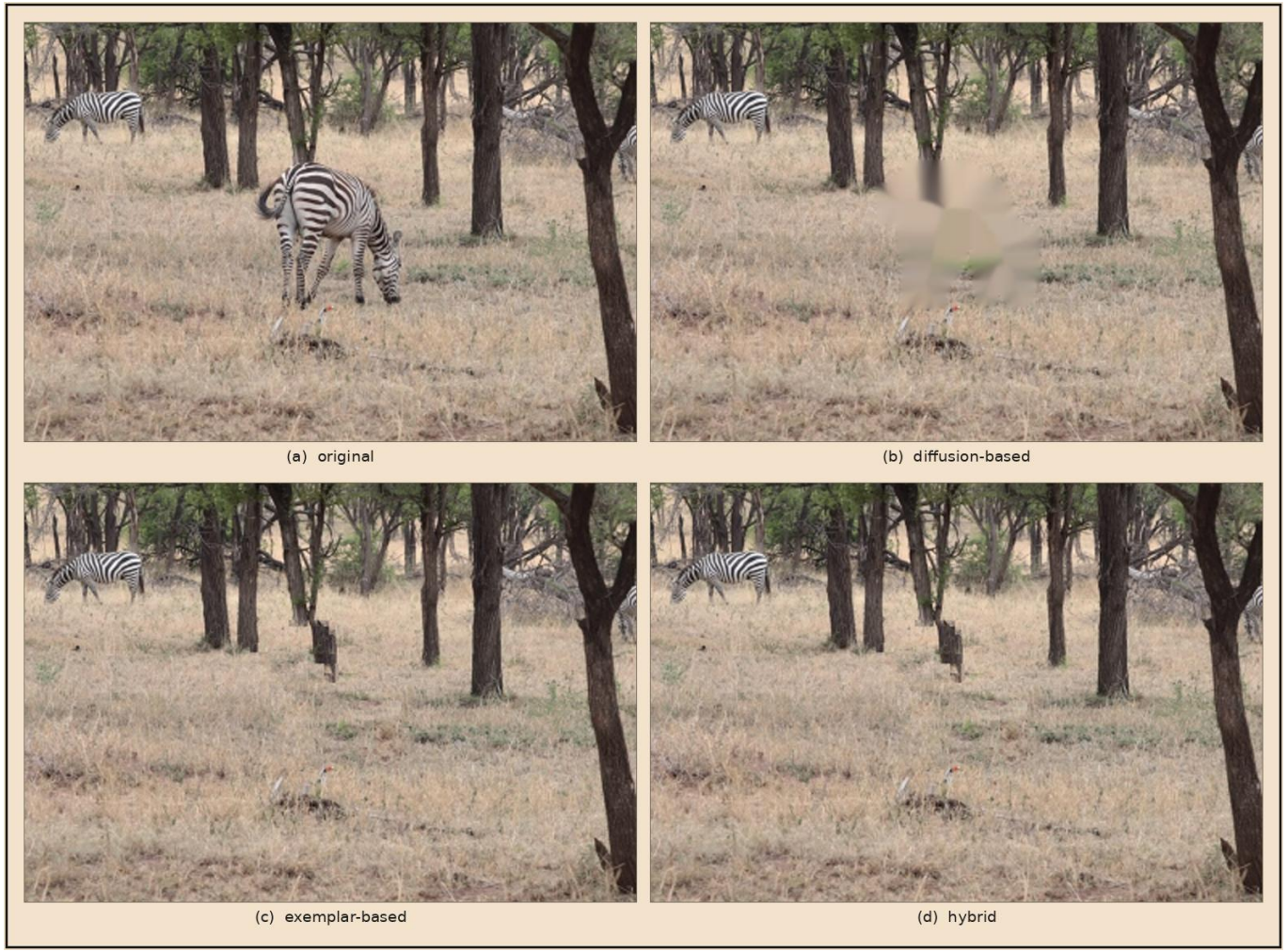
Guillemot, C., & Le Meur, O. (2014). Image inpainting : Overview and recent advances. IEEE Signal Processing Magazine, 31(1), 127–144. https://doi.org/10.1109/msp.2013.2273004
Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CCO, via Wikimedia Commons

Traditional Methods

Hybrid methods

Pros: Combine diffusion for structures and exemplar-based methods for textures, resulting in more visually consistent inpainted images. They offer better handling of complex textures and structures.

Cons: More complex and computationally intensive due to the combination of techniques



Guillemot, C., & Le Meur, O. (2014). Image inpainting : Overview and recent advances. IEEE Signal Processing Magazine, 31(1), 127–144. https://doi.org/10.1109/msp.2013.2273004
Image Source: Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CC0, via Wikimedia Commons

Deep Learning Based Approaches

Autoencoders:

Pros:
- Efficient feature extraction
- Simplicity and Stability
- Denoising Capabilition

Cons:
- Blurred output
- Lack of diversity
- Limited realms for large holes

Similarly GANs can also be used:
- Detail oriented results
- Diversity
- High Realms
- But more tricky to train
- Higher computational demand
- Artifacts and inconsistencies

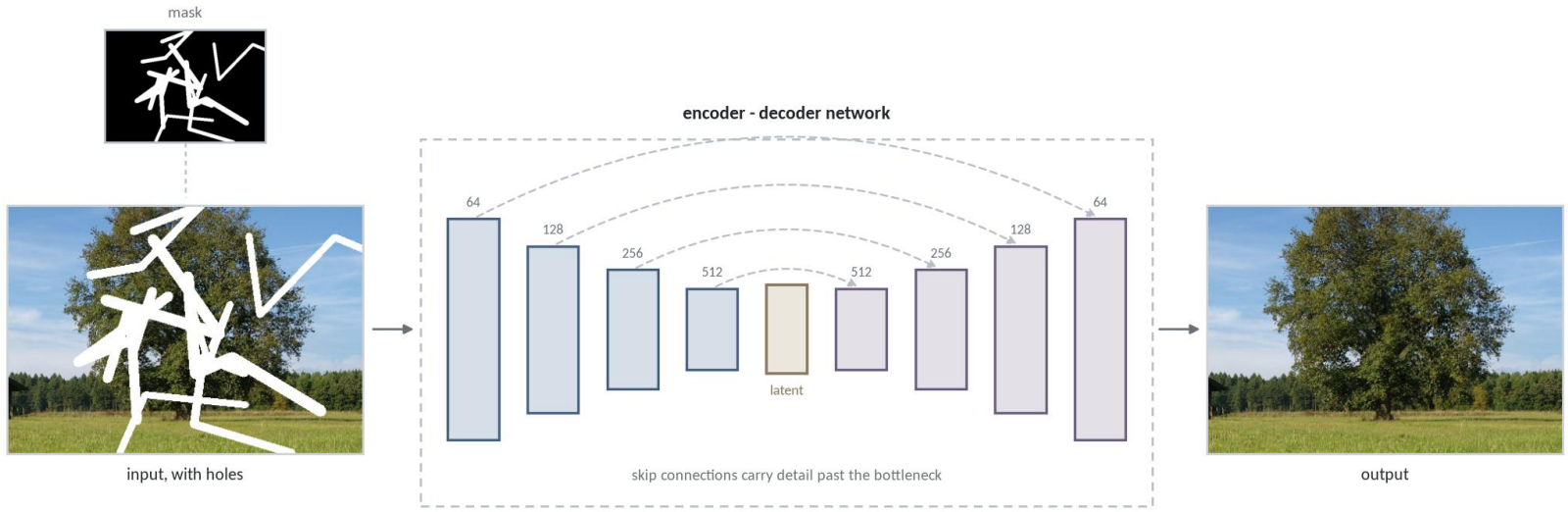


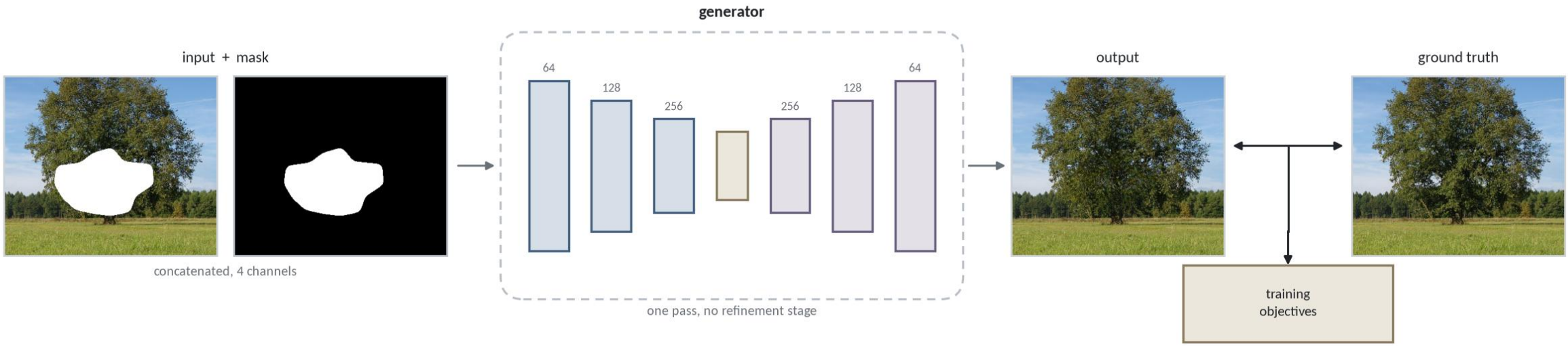
Image Source: LSG Westerwaelder Seenplatte Steinen.jpg, el_tommo, CCO, via Wikimedia Commons

Single-Shot

Uses one generator to directly produce the inpainted image.

Input: corrupted image and mask (concatenation)

Output: the completed image



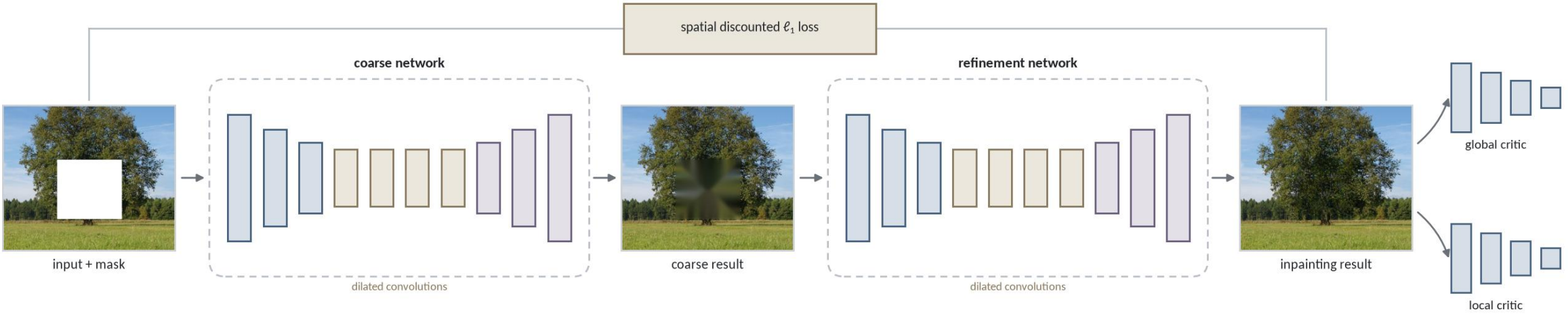
Quan, W., Chen, J., Liu, Y., Yan, D.-M., & Wonka, P. (2024). Deep learning-based image and video inpainting: A survey. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2401.03395
Image Source: LSG Westerwaelder Seenplatte Steinen.jpg, el_tommo, CC0, via Wikimedia Commons

Two-stage

Utilizes two generators

Coarse-to-fine:

- First creates a coarse filling on the missing area
- Second refines the filled area

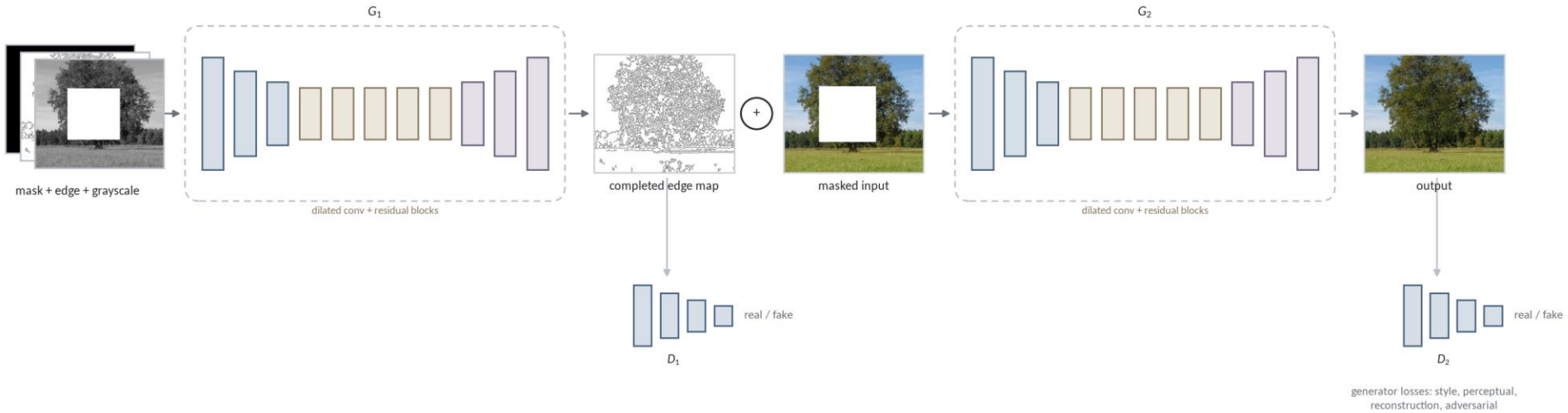


Quan, W., Chen, J., Liu, Y., Yan, D.-M., & Wonka, P. (2024). Deep learning-based image and video inpainting: A survey. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2401.03395
Image Source: LSG Westerwaelder Seenplatte Steinen.jpg, el_tommo, CC0, via Wikimedia Commons

Two-stage

Utilizes two generators

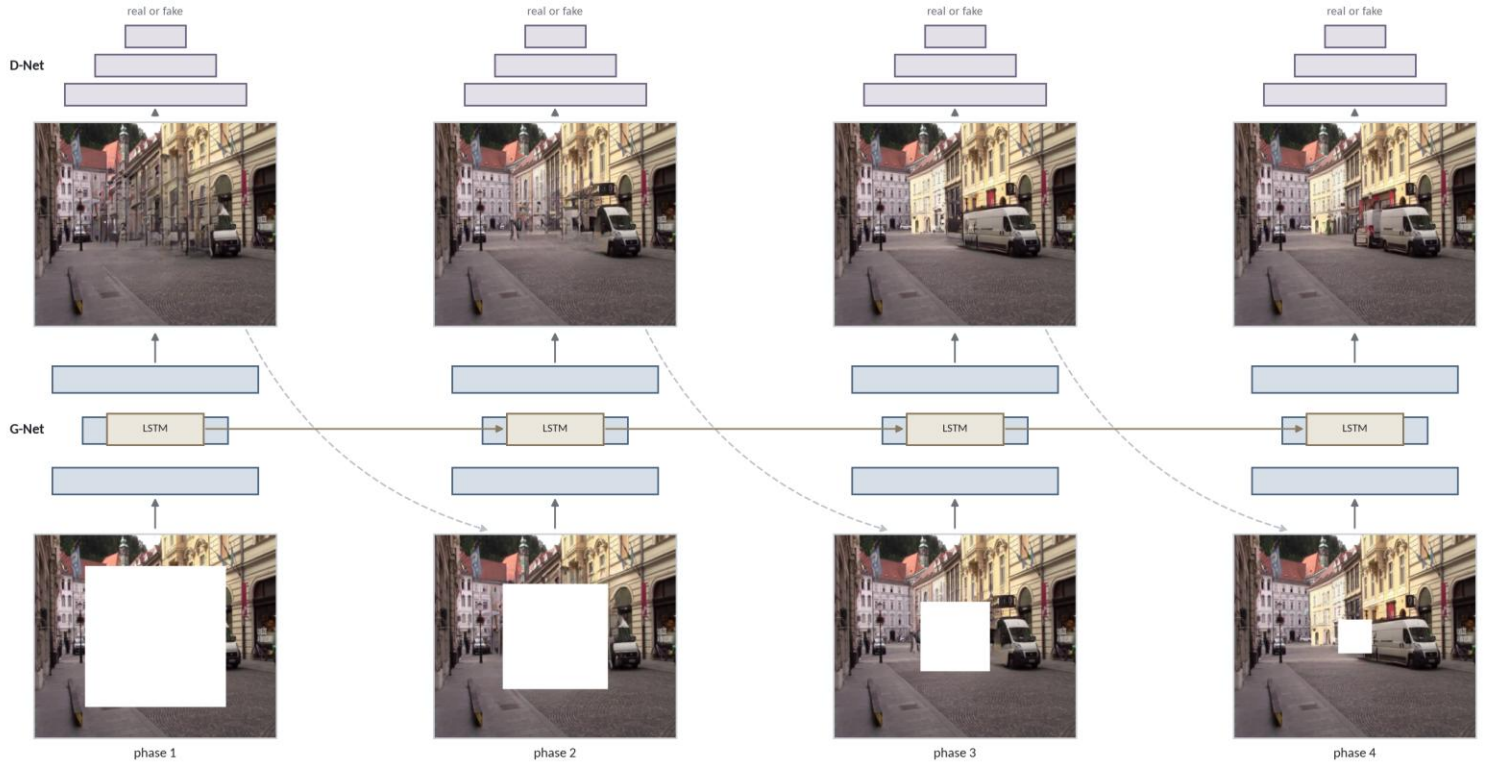
Structure-then-texture:
- First creates a structure map of the image
- Second predicts the complete image



Quan, W., Chen, J., Liu, Y., Yan, D.-M., & Wonka, P. (2024). Deep learning-based image and video inpainting: A survey. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2401.03395
Image Source: LSG Westerwaelder Seenplatte Steinen.jpg, el_tommo, CC0, via Wikimedia Commons

Progressive

Progressive: Iteratively fills the missing area from boundary to center, useful for handling larger missing areas



Quan, W., Chen, J., Liu, Y., Yan, D.-M., & Wonka, P. (2024). Deep learning-based image and video inpainting: A survey. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2401.03395

Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, Point clouds are our own model output

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 12

Deep Learning Tools For Computer Vision

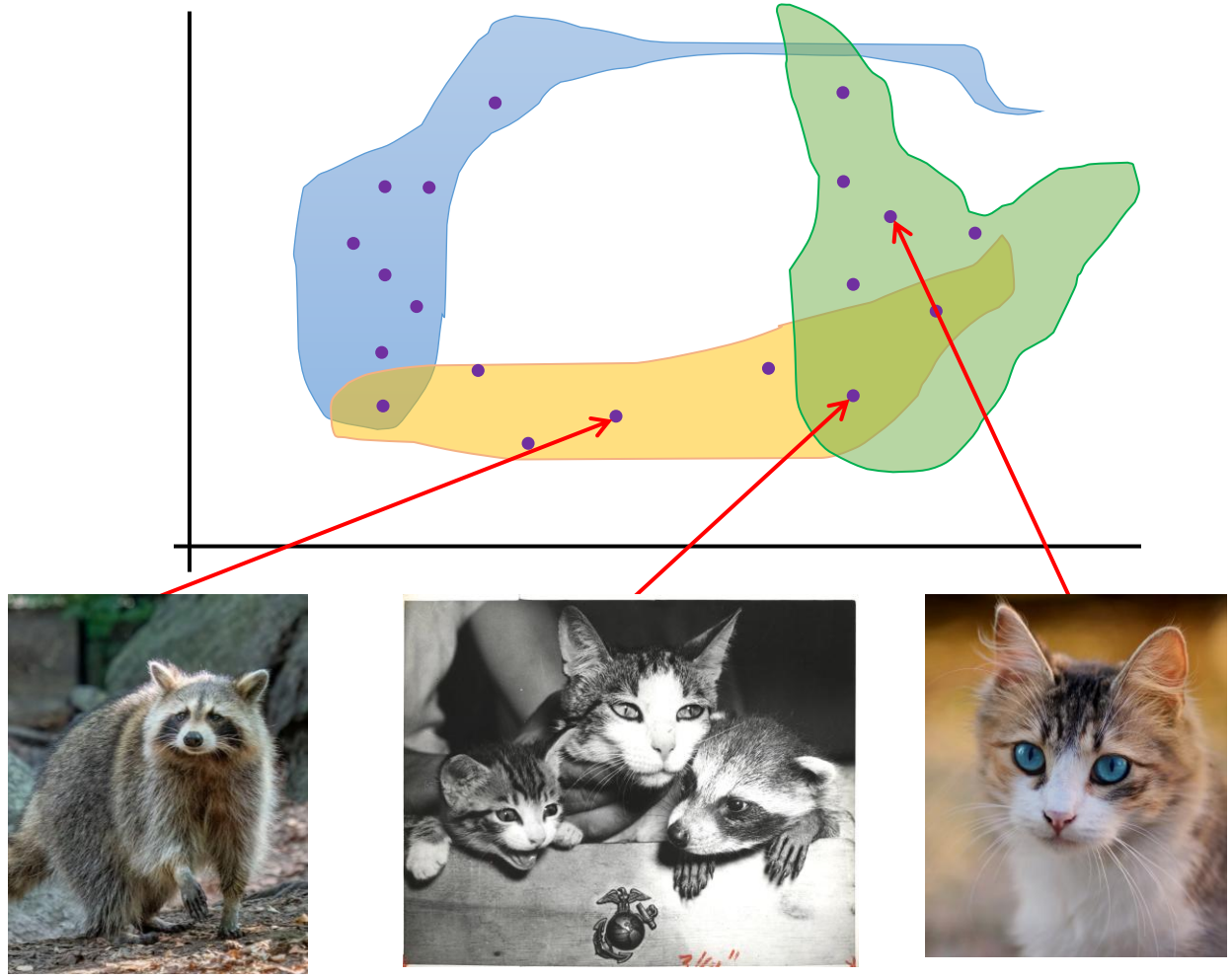
1 Image Inpainting

2 Generative Modeling

3 Neural Rendering

Deep Generative Modeling (DGM)

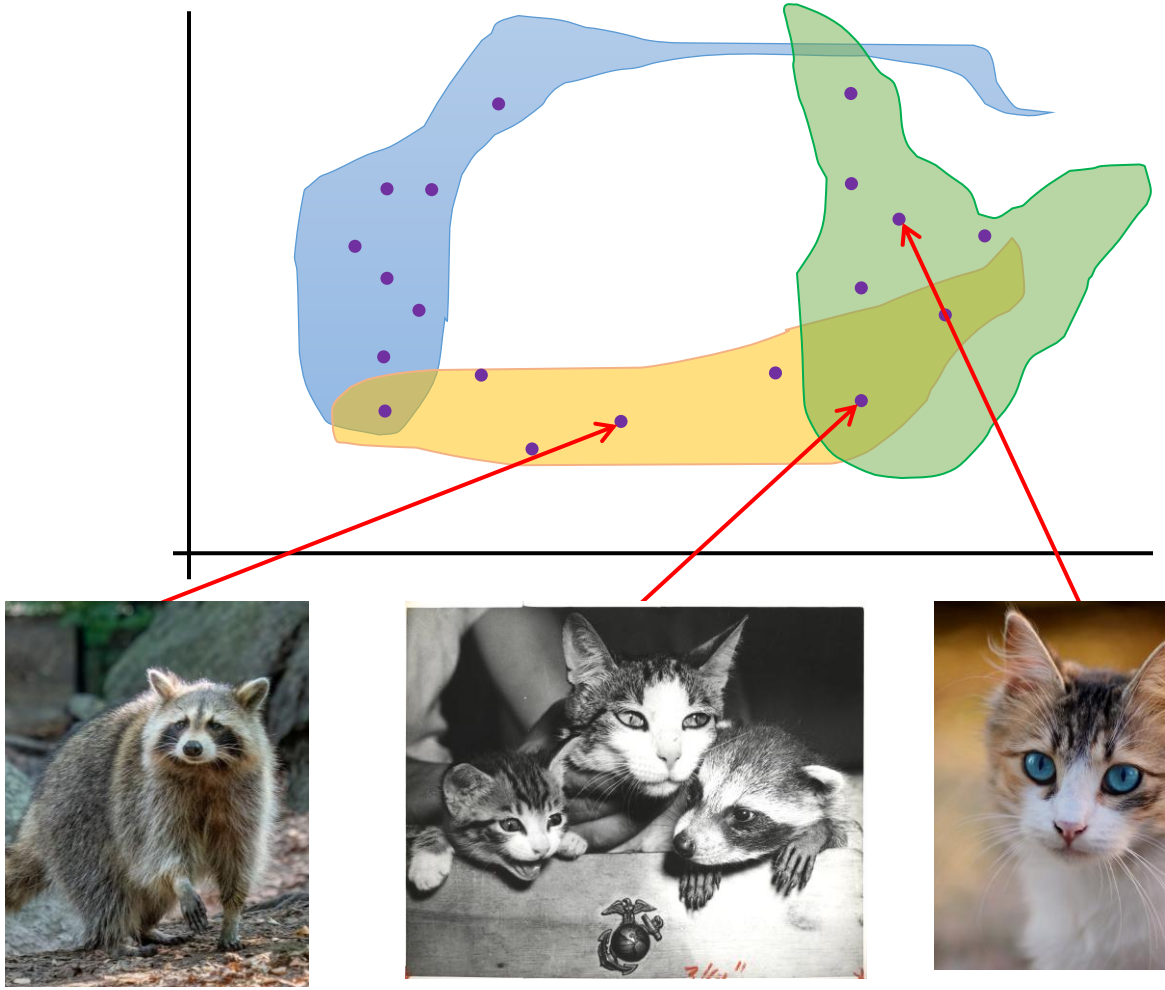
- The ambitious goal in DGM training is to learn an unknown probability distribution from a typically small number of independent and identically distributed samples.
- When trained successfully,
  - We can use the DGM to estimate the likelihood of a given sample
  - Create new samples that are similar to the samples from the unknown distribution (generator).



Ruthotto, L., & Haber, E. (2021). An introduction to deep generative modeling. In arXiv [cs.LG]. arXiv. https://doi.org/10.48550/arXiv.2103.05180
Image Source: Raccoon in Central Park (35264).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Marine Mascots, Date Unknown (5940362181).jpg, USMC Archives from Quantico, USA, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons
Image Source: Tabby cat with blue eyes-3336579.jpg, AdinaVoicu, CC0, via Wikimedia Commons

Deep Generative Modelling (DGM)

1. Uniquely identifying a probability distribution from a finite number of samples is impossible.
2. Training the generator requires a way to quantify its samples' similarity to those from the intractable distribution.
3. We assume, that we can approximate the intractable distribution by transforming a known and much simpler distribution (Gaussian) in a latent space of known dimension



Ruthotto, L., & Haber, E. (2021). An introduction to deep generative modeling. In arXiv [cs.LG]. arXiv. https://doi.org/10.48550/arXiv.2103.05180
Image Source: Raccoon in Central Park (35264).jpg, Rhododendrites, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
Image Source: Marine Mascots, Date Unknown (5940362181).jpg, USMC Archives from Quantico, USA, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons
Image Source: Tabby cat with blue eyes-3336579.jpg, AdinaVoicu, CC0, via Wikimedia Commons

Autoencoders

First, take a pre-trained autoencoder:



Image Source: John Thomas Rowntree Portrait Face.jpg, HRowntree, CC0, via Wikimedia Commons

Autoencoders

Split into encoder and decoder

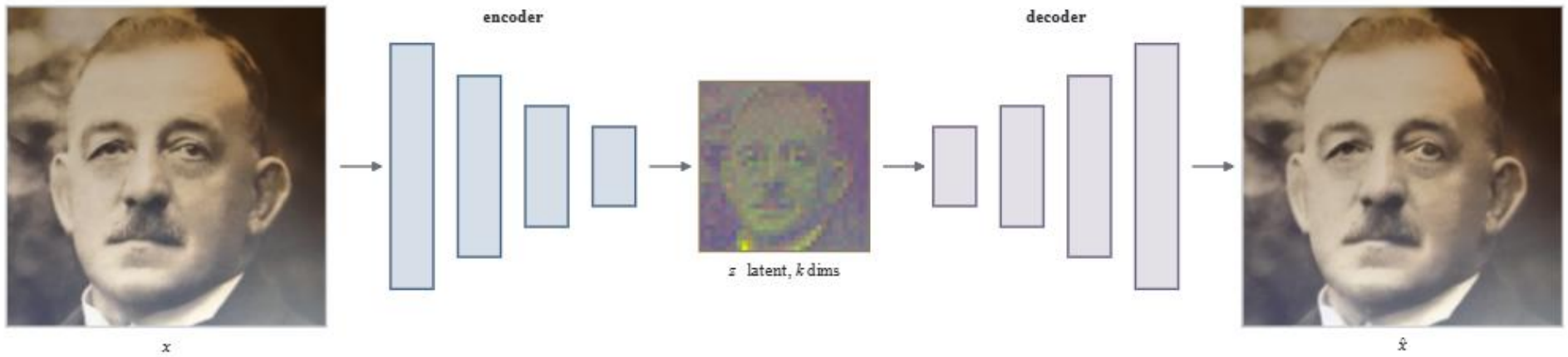


Image Source: John Thomas Rowntree Portrait Face.jpg, HRowntree, CCO, via Wikimedia Commons

Autoencoders

Then modify the latent space representation and feed it into the decoder

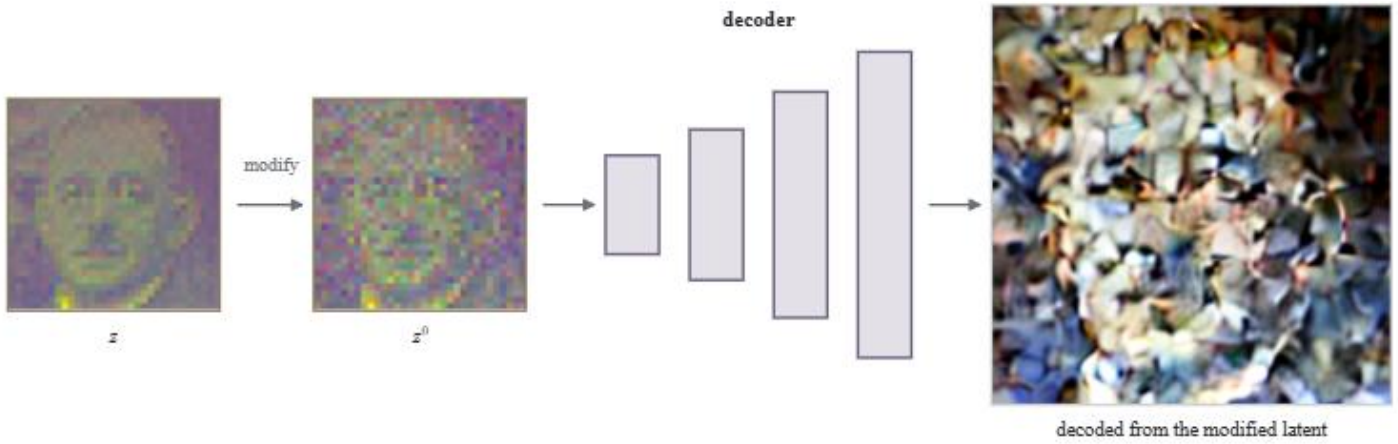


Image Source: John Thomas Rowntree Portrait Face.jpg, HRowntree, CCO, via Wikimedia Commons

Autoencoders

Similarly, try to blend two faces

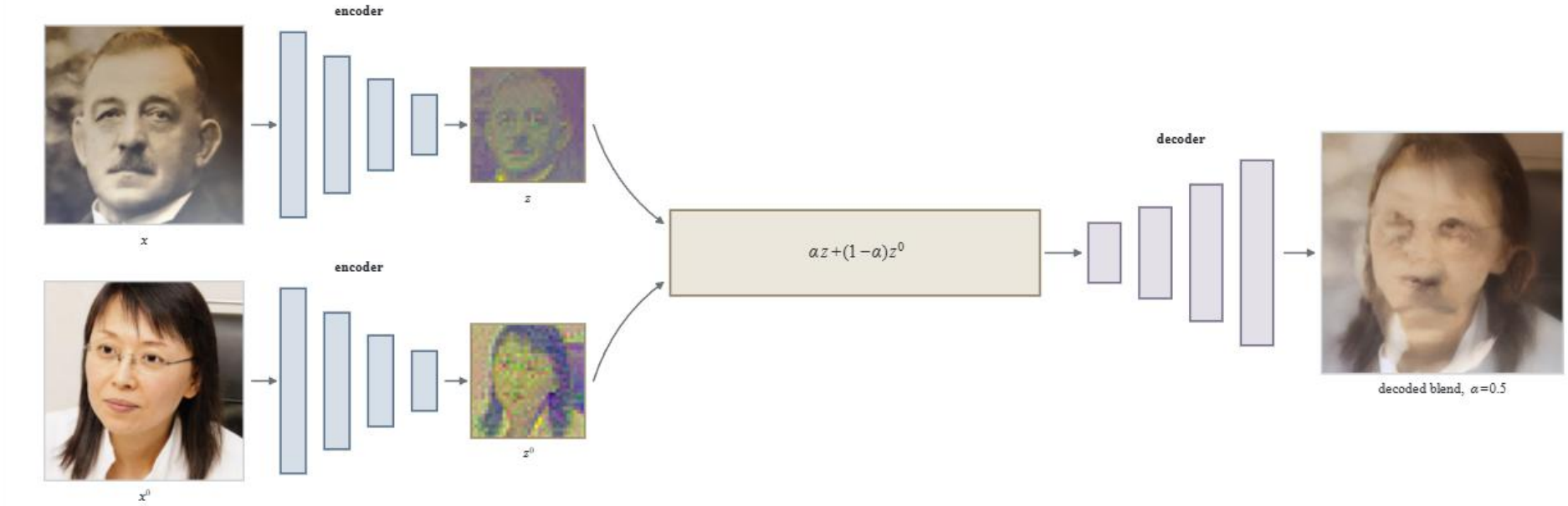
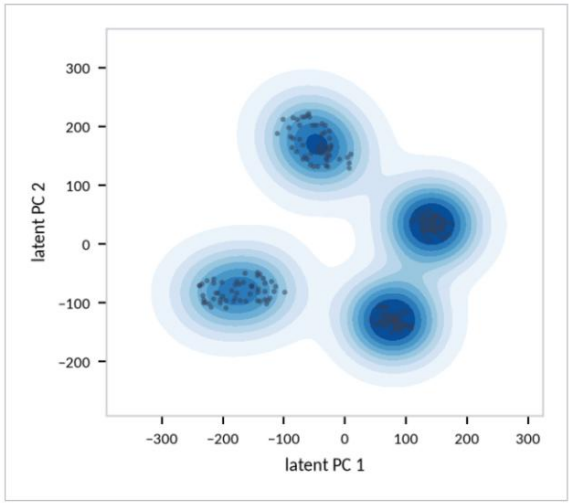


Image Source: John Thomas Rowntree Portrait Face.jpg, HRowntree, CCO, via Wikimedia Commons
Image Source: Chinese woman with glasses and white shirt.jpg, No machine-readable author provided. Aldousleung assumed (based on copyright claims)., Public domain, via Wikimedia Commons

Autoencoders

- The autoencoder didn't see the modified representation, so it struggles with reconstruction
- The blended representation might not be valid



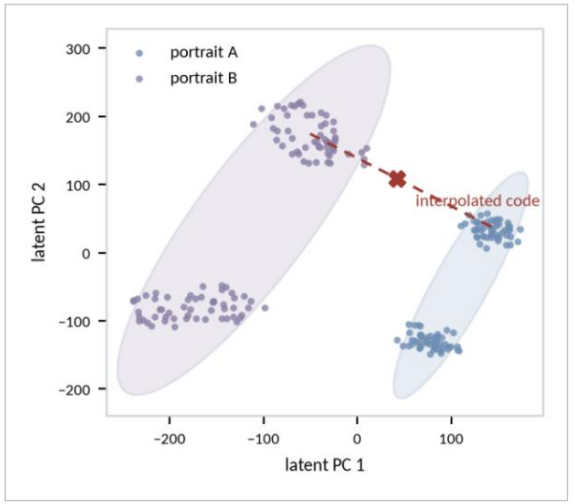




Image Source: John Thomas Rowntree Portrait Face.jpg, HRowntree, CCO, via Wikimedia Commons
Image Source: Chinese woman with glasses and white shirt.jpg, No machine-readable author provided. Aldousleung assumed (based on copyright claims)., Public domain, via Wikimedia Commons

Autoencoders

- For smaller dimensions we can use noising, but in higher dimensions we will likely generate invalid instances
- The distribution of the latent variable **z** that is producing the reconstructed data sample is generally intractable.



John Thomas Rowntree - HRowntree, CCO, via Wikimedia Commons



original

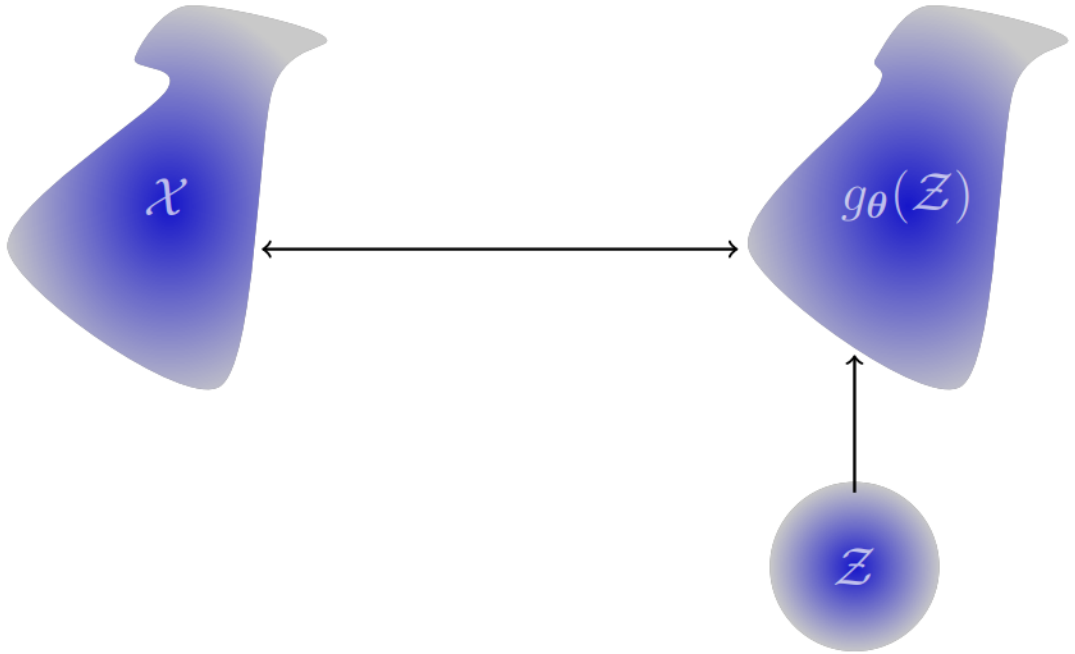


after adding Gaussian noise in pixel space

Image Source: John Thomas Rowntree Portrait Face.jpg, HRowntree, CCO, via Wikimedia Commons

Variational Autoencoders

- We approximate the distribution of the latent variable to be in a parameterized probability distribution (Normal distribution)
- The normal distribution is tractable – we can sample from the distribution and compute probabilities efficiently
- This network takes **x** as an input and generates the parameter of the parameters of the distribution – in case of Normal distribution the **mean** and **covariance**



Ruthotto & Haber (2021) - An Introduction to Deep Generative Modeling, arXiv:2103.05180, CC BY 4.0

Image Source: Fig. 1: A deep generative model — Ruthotto, L. & Haber, E., An Introduction to Deep Generative Modeling (2021), CC BY 4.0 https://creativecommons.org/licenses/by/4.0/ , https://arxiv.org/abs/2103.05180

Variational Autoencoders

Let's constrain the possible representations into a know distribution (Normal distribution)

- If we would know the distribution, the generation wouldn't be an issue
- The results are blurry because of the minimization of the MSE loss

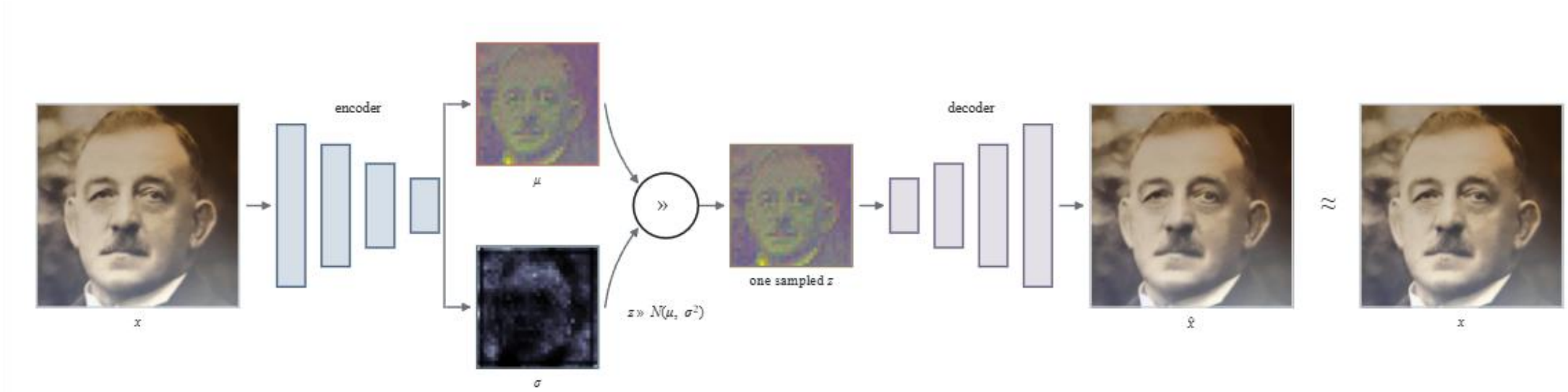


Image Source: John Thomas Rowntree Portrait Face.jpg, HRowntree, CCO, via Wikimedia Commons

Variational Autoencoders

https://xnought.github.io/vae-explainer/, - Donny Bertucci, MIT licence

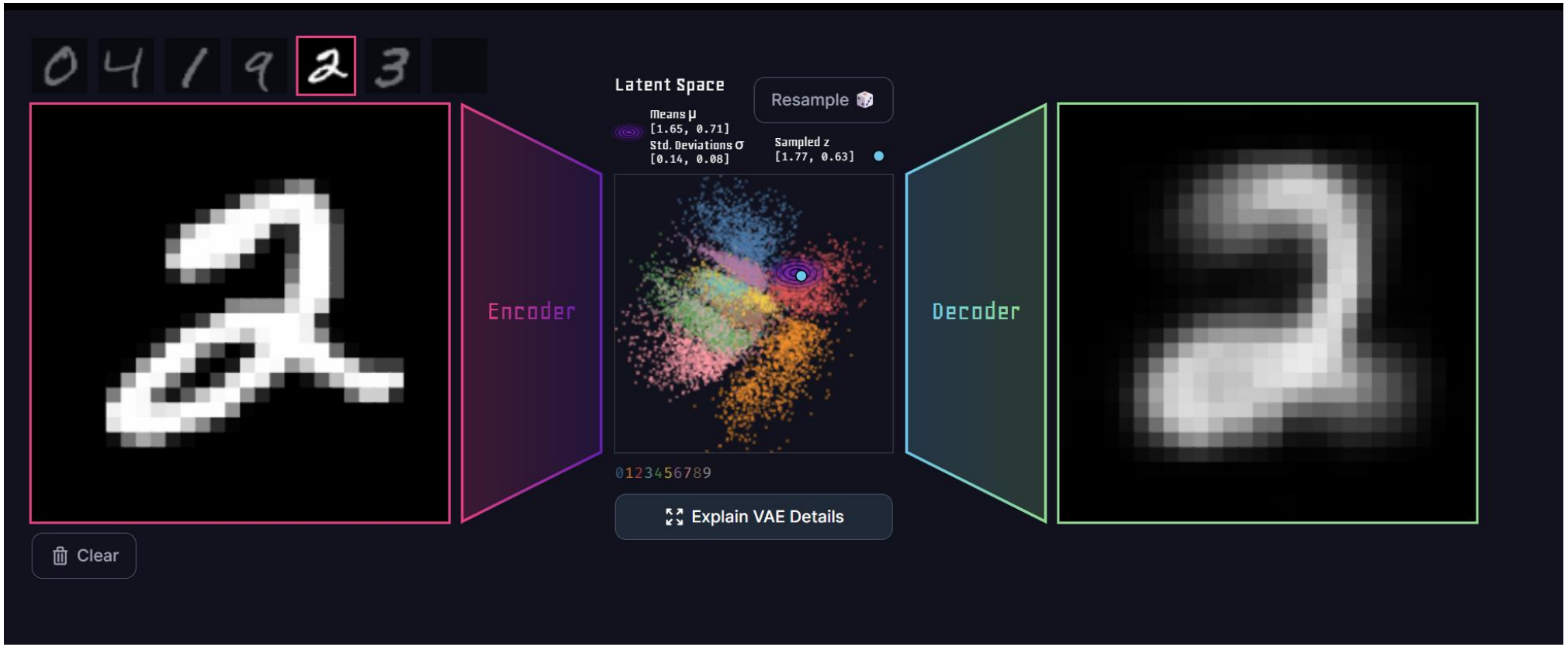
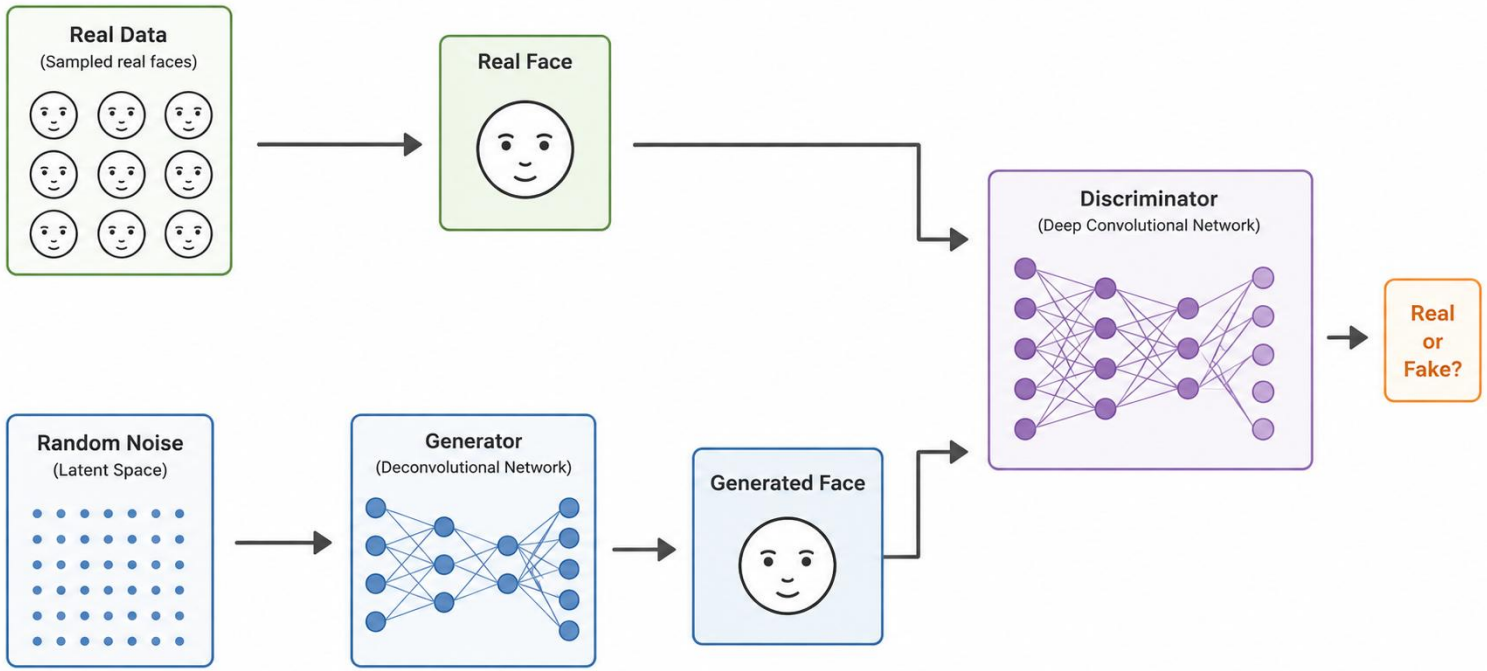


Image Source: Screenshot from **VAE Explainer** by Donald Bertucci and Alex Endert, based on the MIT-licensed xnought/vae-explainer repository. Source: https://github.com/xnought/vae-explainer.

Generative Adversarial Networks

- We train the network by minimizing a loss function that measures the distance between the generated and the sampled image.
- Compare the distribution in data space
- It does not try to infer the latent variable

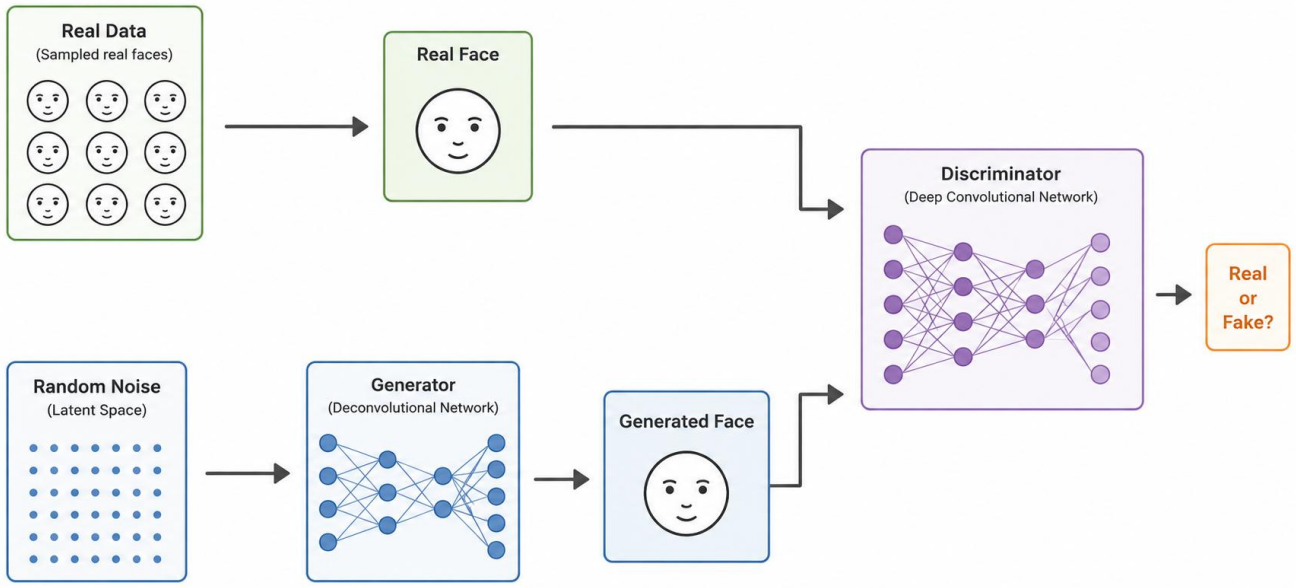


Generative Adversarial Networks

A Generator and a Discriminator are competing

- **Generator:** Generates images.
- **Discriminator**: Tries to predict if the input image is real or fake (binary classification).

- If the generator generates an image that the discriminator considers a valid image, we punish the discriminator
- If the discriminator predicts that the generated image is fake we punish the generator



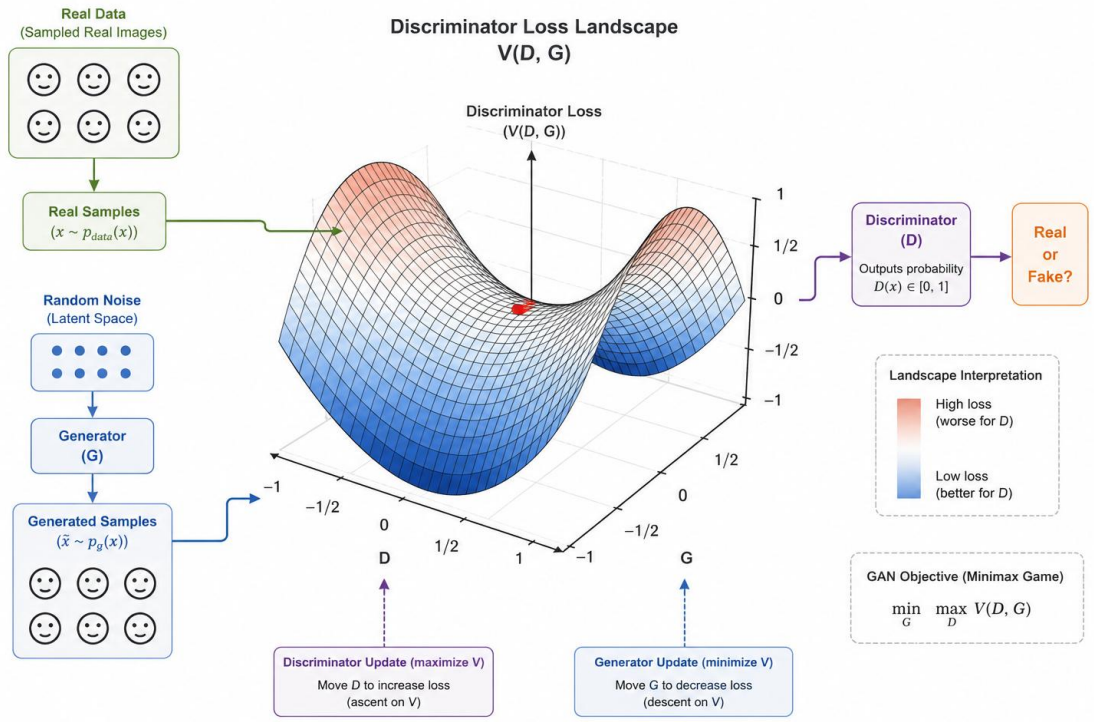
Generative Adversarial Networks

- Due to the binary classification the GAN is trained with the cross-entropy loss

$$J_{GAN}(\theta, \phi) = \mathbb{E}_{x \sim X}\left[\log\left(d_\phi(x)\right)\right] + \mathbb{E}_{z \sim Z}\left[\log\left(1 - d_\phi\left(g_\theta(z)\right)\right)\right]$$

- **d** – the discriminator with parameters ϕ
- **g** – the generator with parameters θ

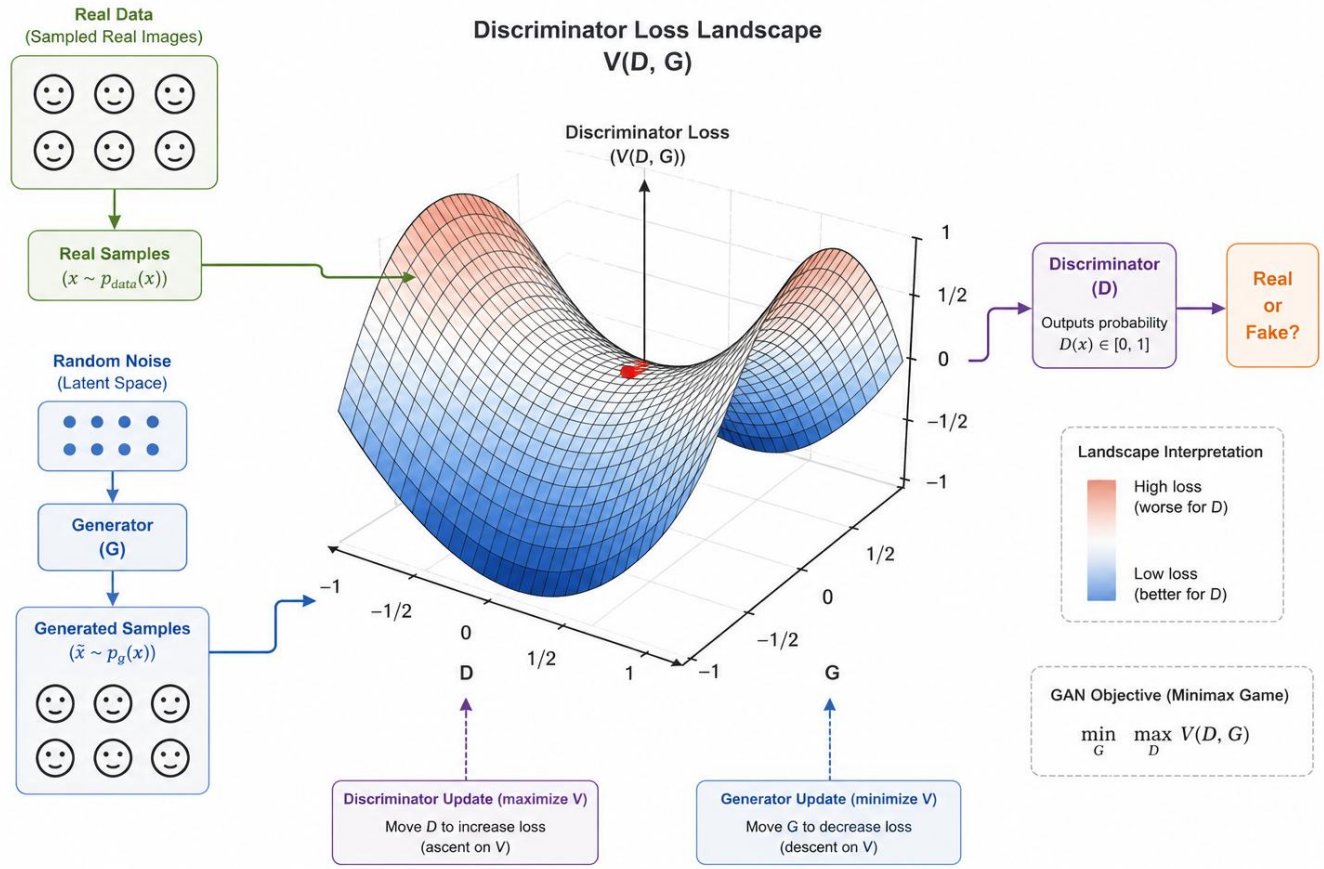
- The discriminator is trying to maximize the loss (it can effectively detect fakes from real images)
- The generator is trying to minimize the loss (it can generate deceiving images)
- Thus the generator and the discriminator play a zero-sum game



Generative Adversarial Networks

- Finding this saddle point is hard
- The loss does not tell exactly whether the generator or the discriminator is good

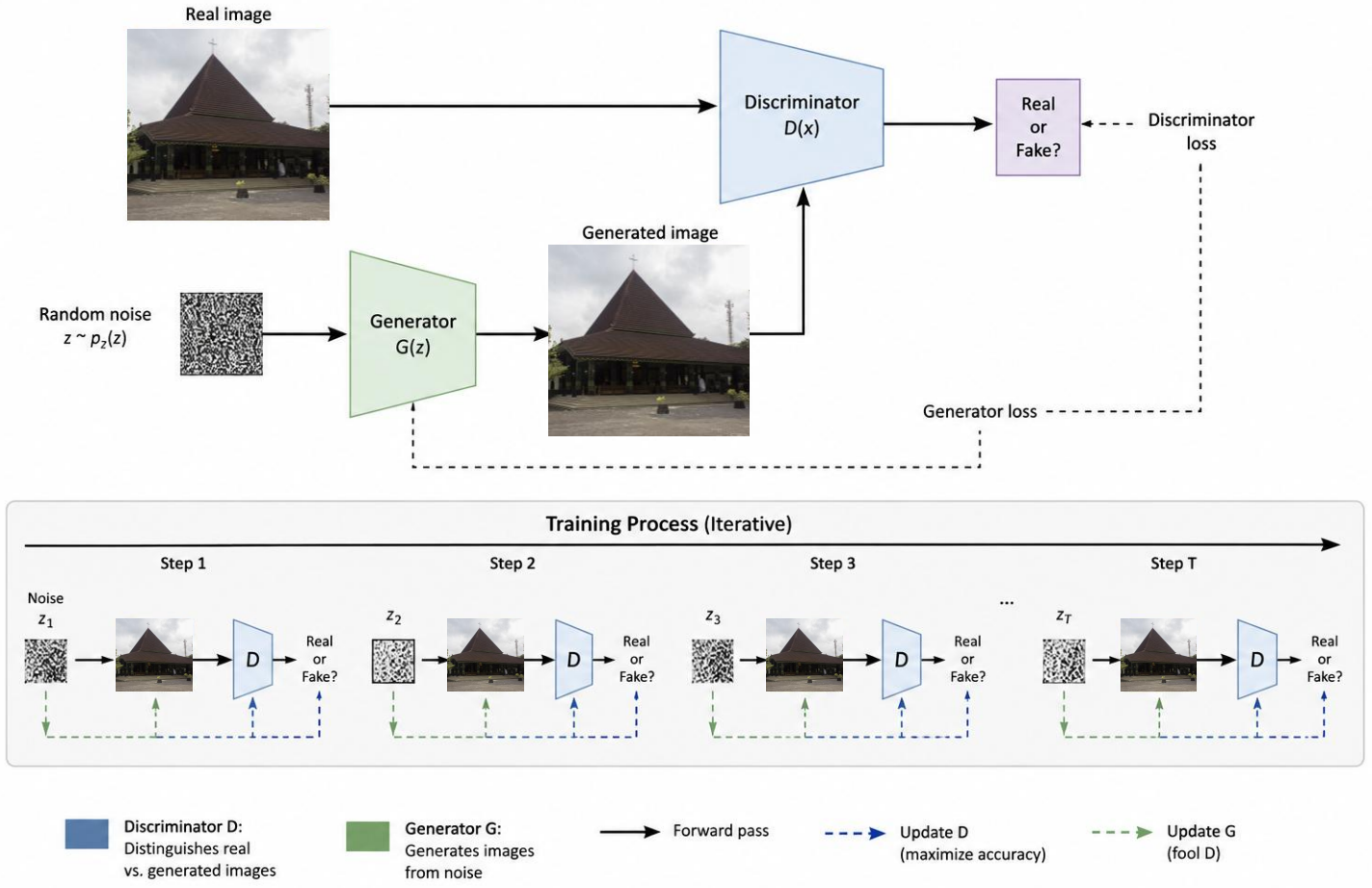
- Let's assume that we have an optimal generator – the discriminator can only random guess if the image is generated or sampled from the dataset
- If we have a slightly off generator, then the discriminator can greatly increase the objective



Stable Diffusion

The main idea is to train a denoising network.

- In the forward diffusion process we gradually add Gaussian noise to the image in each step
- The model estimates the added noise and subtracts it from the image (recreating the original image)
- In the reverse diffusion process we start from gaussian noise and the network gradually removes the noise (in multiple passes), thus generating an image



Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2112.10752
Image Source: Ganjuran Church, exterior 01.jpg, Crisco 1492, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

Deep Network Development

ELTE EÖTVÖS LORÁND UNIVERSITY

Lecture 12

Deep Learning Tools For Computer Vision

1 Image Inpainting

2 Generative Modeling

3 Neural Rendering

Neural Rendering

It combines the deep learning model with the physical knowledge of computer graphics, to obtain a controllable and realistic scene model, and realize the control of scene attributes such as lighting, camera parameters, posture and so on.

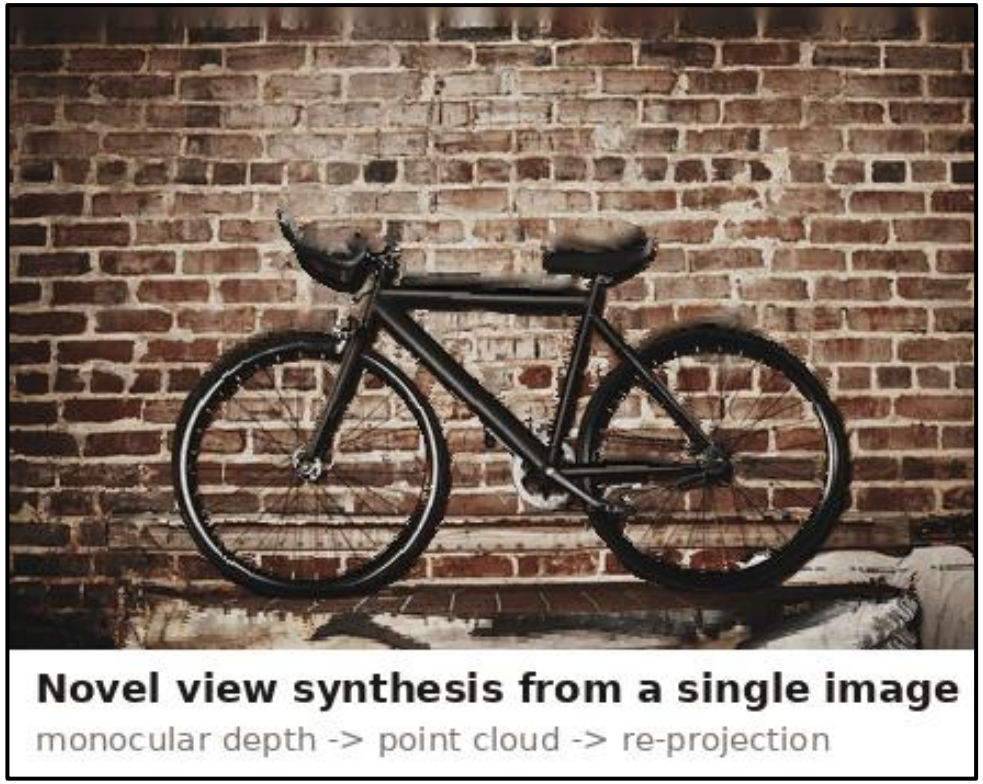


Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Differentiable rendering

- Optimization based on images
- Loss is typically the difference between a rendered image and a photo

- Most rendering algorithms are **not differentiable:**
  - Meshes have hard edges, giving abrupt discontinuities

Structure from Motion

- **Structure from Motion (SfM)** is a field within computer vision that seeks to reconstruct a three-dimensional structure of the environment from a sequence of two-dimensional images.
- Obtaining the geometry of 3D scenes from 2D images is a challenging task because the image formation process (3D world → 2D image captured) is generally not invertible and additional information is needed to solve the reconstruction problem.
- Given its image in two or more views, a 3D point can be reconstructed by triangulation.
- Structure from Motion usually have three steps:
    1. Feature extraction
    2. Feature matching
    3. Camera pose estimation

Structure from Motion – Feature Extraction

- First, we must find feature(point)s that can be matched between images.
- These features are usually **corners**
- After finding these points we create the **feature descriptors**
  - Around each point we take a 40x40 region.
  - Blur the region
  - Subsample it to create an 8x8 reduced region
  - Flatten to 64x1



Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, SIFT keypoints computed with OpenCV.

Structure from Motion – Feature Matching

- We sample two images from the set
  - Select one point from the first image
  - Match the descriptor of the selected points against the points in the second image
  - Take the ratio of best match (lowest distance) to the second best match (second lowest distance) and if this is **below some ratio keep** the matched pair or **reject** it
  - Repeat for all the points in the first image
- You will be left with only the confident feature correspondences
- These points will be used to estimate the transformation between the 2 images

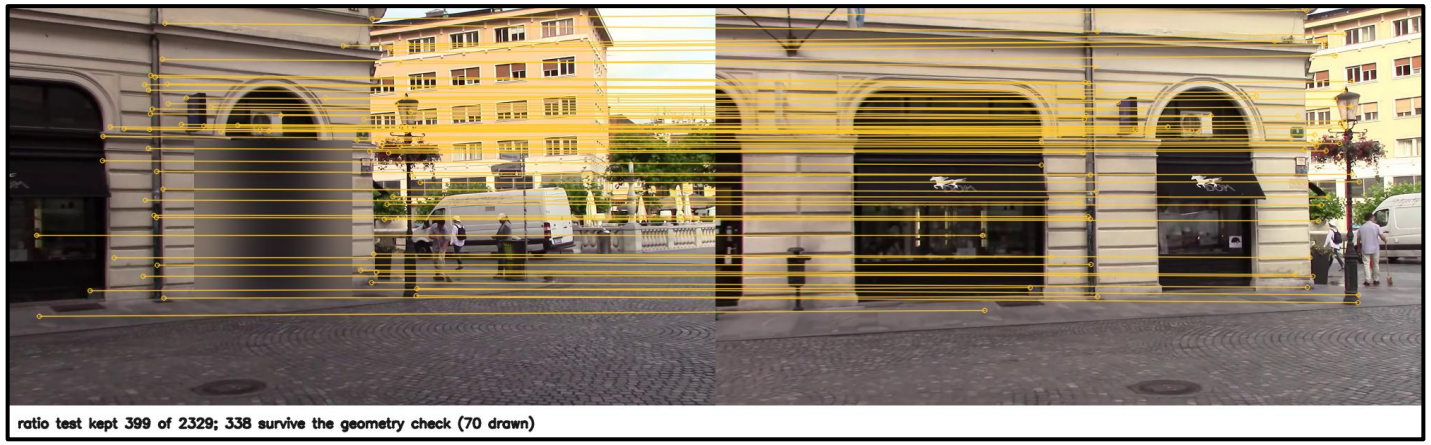
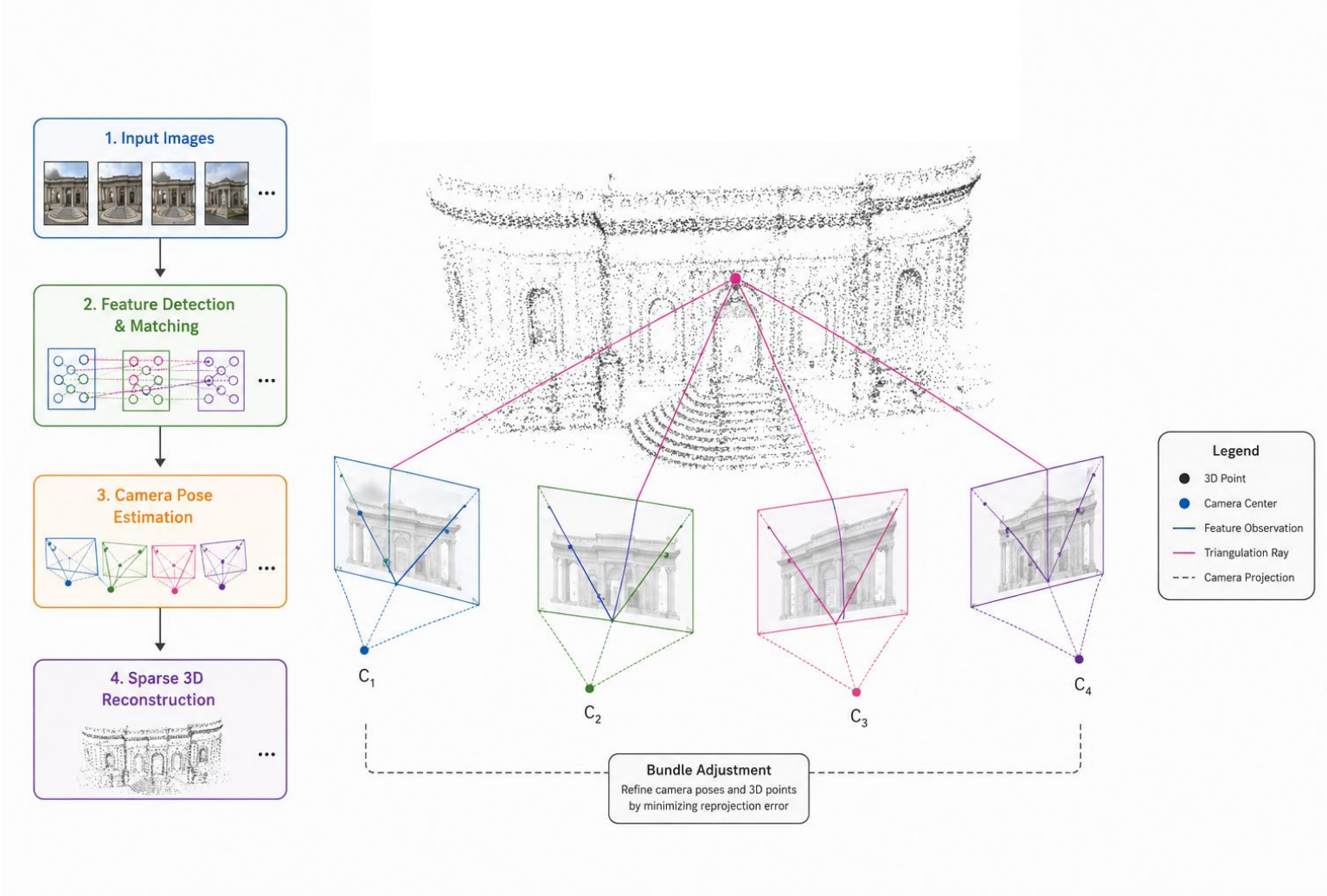


Image Source: Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons, SIFT keypoints computed with OpenCV.

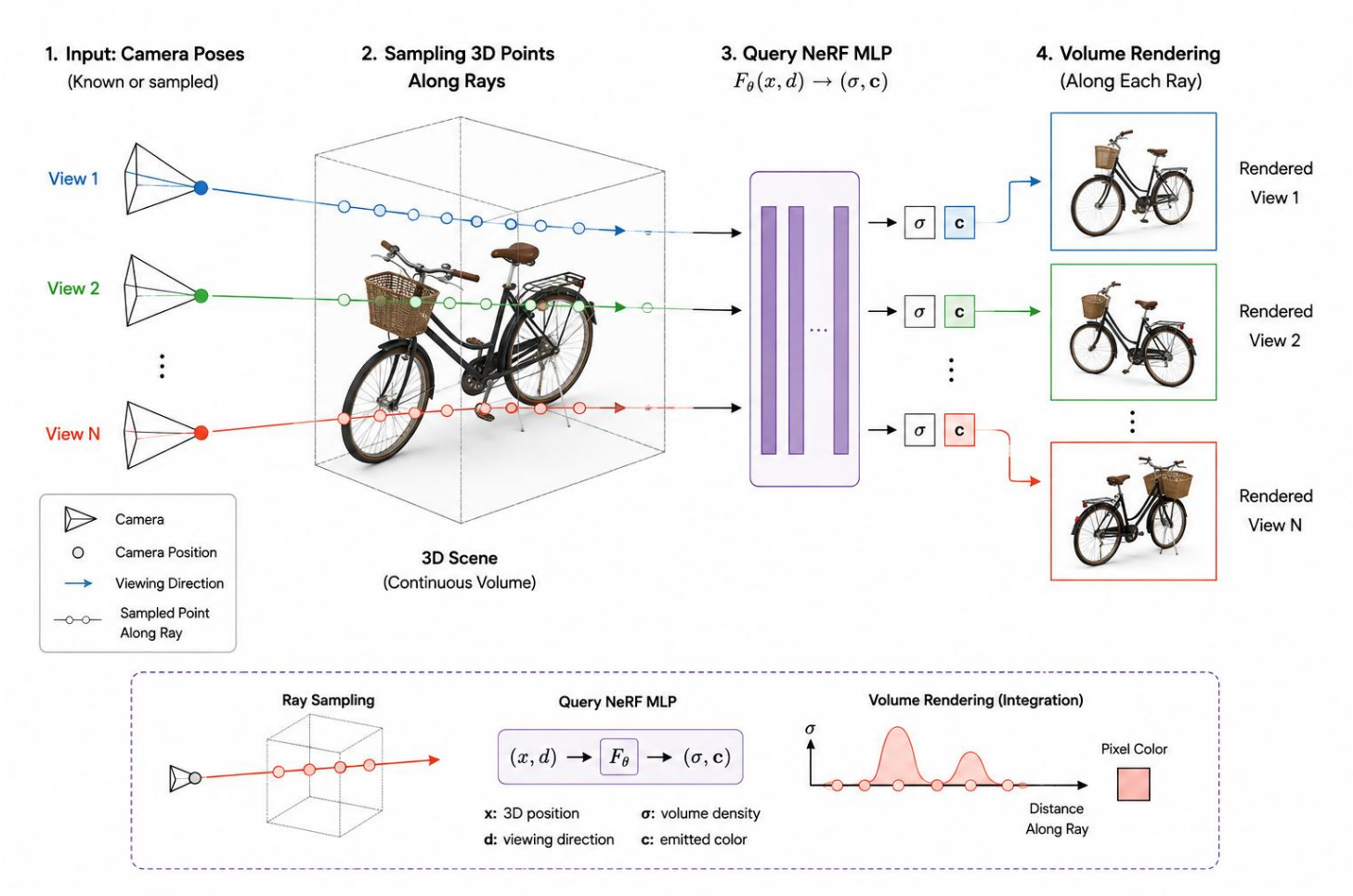
Structure from Motion

- Based on the matched features the camera positions can be calculated
- **Note**: These positions are only the relative camera positions



Neural Radiance Fields (NeRF)

- Used to represent a continuous scene with a neural network
- Treats the scene as a continuous function
- Uses differentiable rendering (volumetric rendering) to recreate the scene
- The differentiable rendering allows us to train the network with backpropagation



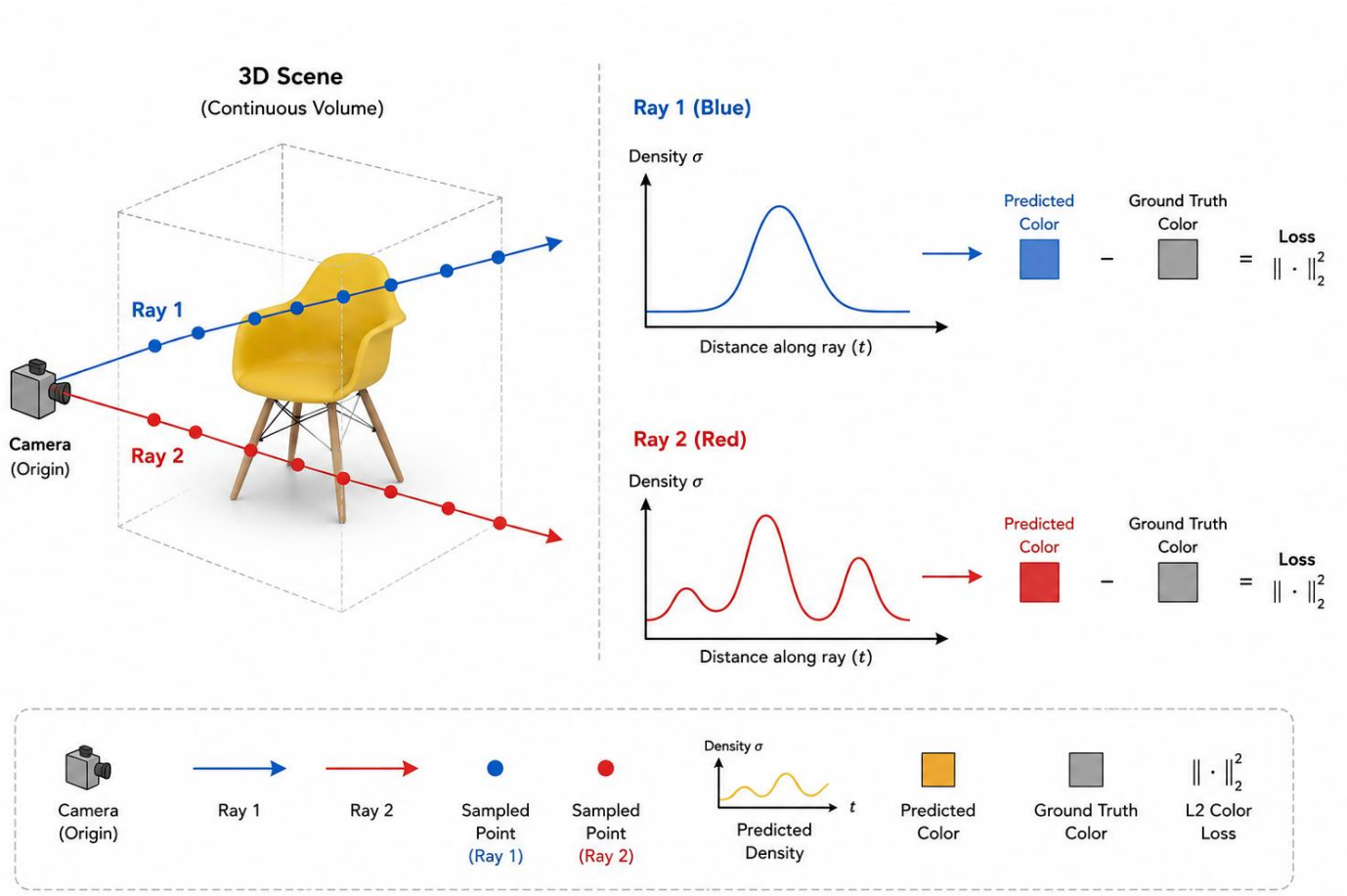
Neural Radiance Fields (NeRF)

NeRF represents a scene using a fully-connected (non-convolutional) deep network

Input: 5D coordinate

- spatial location (x, y, z)
- viewing direction (θ, φ)

Output: the volume density and view-dependent emitted radiance at that spatial location.

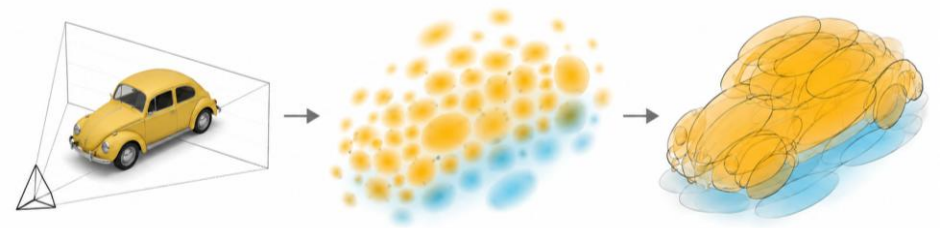


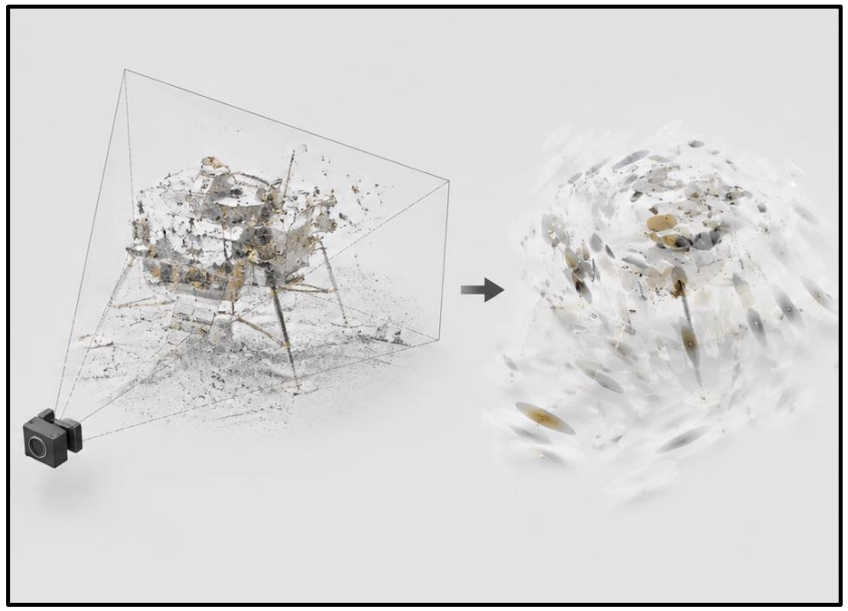
Gaussian Splatting (GS)

3D Gaussian Splatting is a rasterization technique described in [3D Gaussian Splatting for Real-Time Radiance Field Rendering] that allows real-time rendering of photorealistic scenes learned from small samples of images.

Instead of triangles let's use gaussian splats to render a 3D scene.

Each splat can be described as:
- **Position**: where it's located (XYZ)
- **Covariance**: how it's stretched/scaled (3x3 matrix)
- **Color**: what color it is (RGB)
- **Alpha**: how transparent it is (α)



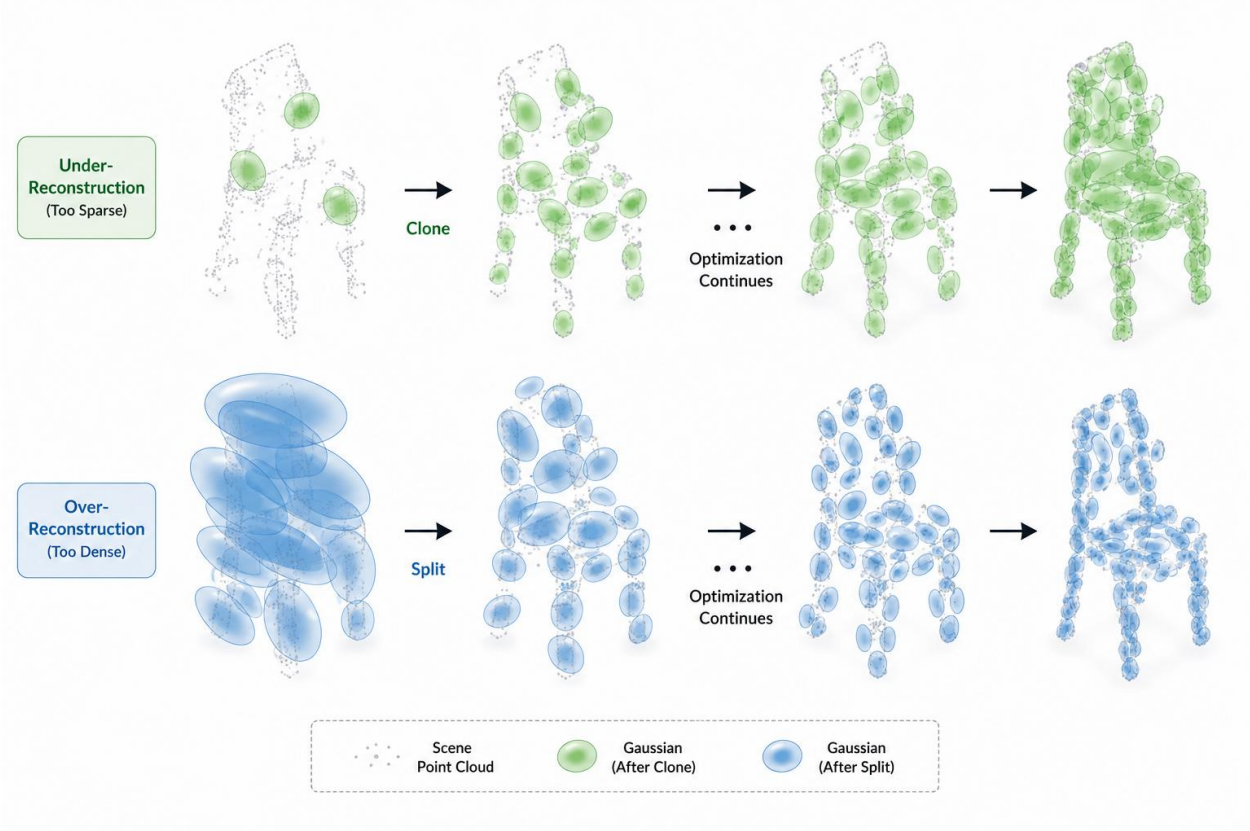


https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Gaussian Splatting (GS)

The training procedure uses Stochastic Gradient Descent, similar to a neural network, but without the layers. The training steps are:

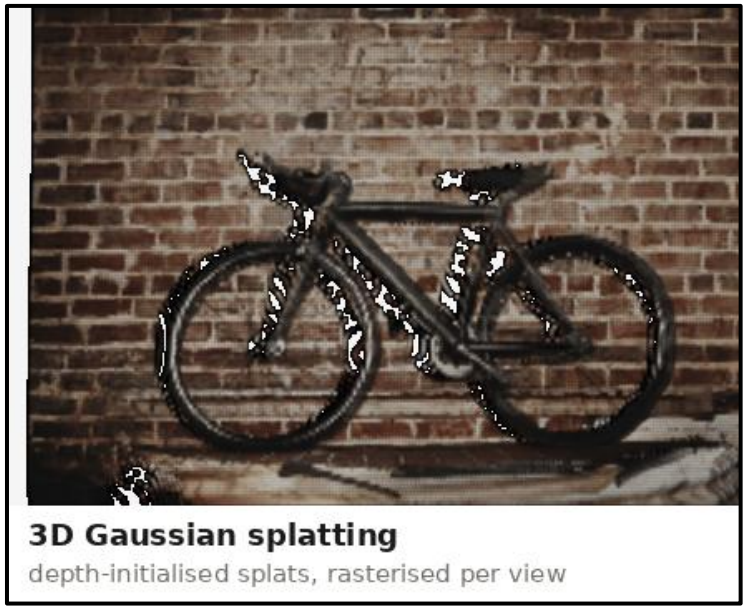
1. Rasterize the gaussians to an image
2. Calculate the loss based on the difference between the rasterized image and ground truth image
3. Adjust the gaussian parameters according to the loss
4. Apply automated densification and pruning:
    - If the gradient is large for a given gaussian (i.e. it's too wrong), split/clone it
    - If the gaussian is small, clone it
    - If the gaussian is large, split it
    - If the alpha of a gaussian gets too low, remove it



https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Gaussian Splatting (GS)





Kerbl, B., Kopanas, G., Leimkuehler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance Field rendering. ACM Transactions on Graphics, 42(4), 1–14. https://doi.org/10.1145/3592433
Image Source: Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Gaussian Splatting (GS)

[3D Gaussian Splatting for Real-Time Radiance Field Rendering]

https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Summary

- In **Deep Generative Modeling** the goal is to learn an unknown probability distribution, then create new samples that are similar to the samples from the unknown distribution.
- **Image Inpainting** is about filling in missing or occluded regions in digital images
- **Neural Rendering** is a technique that uses neural networks to generate, enhance or manipulate visual content in a realistic way.

Resources

Papers:

1. Guillemot, C., & Le Meur, O. (2014). Image inpainting : Overview and recent advances. *IEEE Signal Processing Magazine*, *31*(1), 127–144. https://doi.org/10.1109/msp.2013.2273004
2. Kerbl, B., Kopanas, G., Leimkuehler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance Field rendering. *ACM Transactions on Graphics*, *42*(4), 1–14. https://doi.org/10.1145/3592433
3. Quan, W., Chen, J., Liu, Y., Yan, D.-M., & Wonka, P. (2024). Deep learning-based image and video inpainting: A survey. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2401.03395
4. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. In *arXiv [cs.CV]*. arXiv. https://doi.org/10.48550/arXiv.2112.10752
5. Ruthotto, L., & Haber, E. (2021). An introduction to deep generative modeling. In *arXiv [cs.LG]*. arXiv. https://doi.org/10.48550/arXiv.2103.05180
6. Structure from Motion. (n.d.). Retrieved November 17, 2025, from https://cmsc426.github.io/sfm/

Resources

Books:

- Goodfellow, I., Bengio, Y., & Courville, A. (n.d.). *Deep learning*. https://www.deeplearningbook.org/
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. https://d2l.ai/

Courses:

- Deep Learning Specialization by Andrew NG - https://www.coursera.org/specializations/deep-learning
- https://developers.google.com/machine-learning/crash-course/embeddings

Further Links + Resources

- Deep Learning-based Image and Video Inpainting: A Survey - https://arxiv.org/abs/2401.03395
- An Introduction to Deep Generative Modeling - https://arxiv.org/abs/2103.05180
- Advances in Neural Rendering - https://arxiv.org/abs/2111.05849
- 3D Gaussian Splatting for Real-Time Radiance Field Rendering - https://arxiv.org/abs/2308.04079
- Introduction to 3D Gaussian Splatting - https://huggingface.co/blog/gaussian-splatting

Image Sources

- Unless otherwise noted, the figures were created for this course.
- Some icons/images used in this presentation are from Microsoft 365 Stock Images / Icons and are used under the applicable Microsoft 365 license terms.
- Dancing Robot: Gif robot.git, Meks01, CC BY-SA 4.0, via Wikimedia Commons.
- Figures built on Share-Alike source images are themselves released under CC BY-SA 4.0.
- All depth maps, optical flow fields, feature maps, correlation volumes, point clouds, detections, segmentations and attention maps in this presentation are our own model output.
- 06-Pont Notre-Dame sur le Lot à Mende (Alfred-Nicolas Normand, 1888).jpg, Alfred-Nicolas Normand, Public domain, via Wikimedia Commons, Restoration: own output (LaMa).
- Frame from - Road traffic on Stritarjeva street.webm, Sounds of Changes, CC BY 3.0, via Wikimedia Commons.
- Bike against brick wall (Unsplash).jpg, Darius Soodmand dsoodmand, CC0, via Wikimedia Commons.
- Zebras grazing among acacia trees, Serengeti National Park.jpg, Agness Abubakar Saidi, CC0, via Wikimedia Commons.
- LSG Westerwaelder Seenplatte Steinen.jpg, el_tommo, CC0, via Wikimedia Commons.
- Raccoon in Central Park (35264).jpg, Rhododendrites, CC BY-SA 4.0, via Wikimedia Commons.
- Marine Mascots, Date Unknown (5940362181).jpg, USMC Archives from Quantico, USA, CC BY 2.0, via Wikimedia Commons.

Image Sources (cont.)

- Tabby cat with blue eyes-3336579.jpg, AdinaVoicu, CC0, via Wikimedia Commons.
- John Thomas Rowntree Portrait Face.jpg, HRowntree, CC0, via Wikimedia Commons.
- Chinese woman with glasses and white shirt.jpg, author unknown (Aldousleung assumed), Public domain, via Wikimedia Commons.
- Fig. 1: A deep generative model — Ruthotto, L. & Haber, E., An Introduction to Deep Generative Modeling (2021), CC BY 4.0, arxiv.org/abs/2103.05180.
- Screenshot from VAE Explainer by Donald Bertucci and Alex Endert, based on the MIT-licensed xnought/vae-explainer repository, github.com/xnought/vae-explainer.
- Ganjuran Church, exterior 01.jpg, Crisco 1492, CC BY-SA 3.0, via Wikimedia Commons.

Legal Notice

vae-explainer — Donny Bertucci
Source: https://github.com/xnought/vae-explainer
License: MIT License

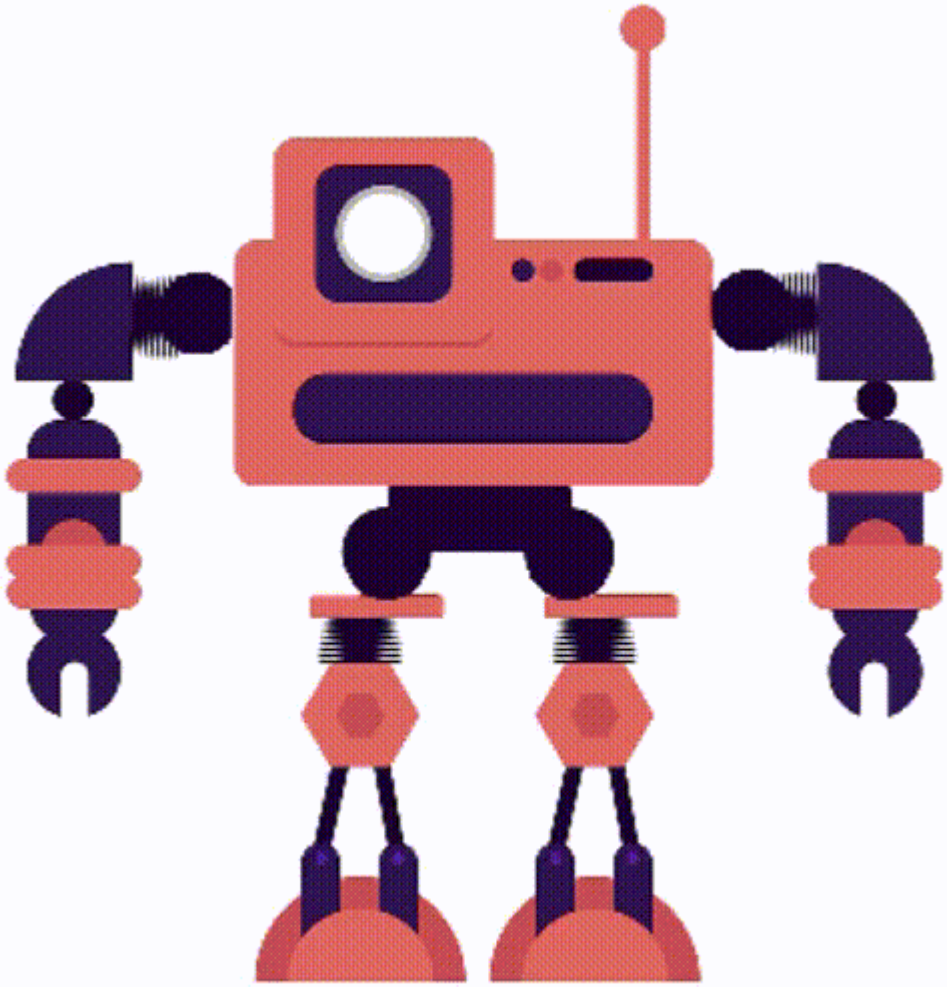
Copyright (c) 2024 Donny Bertucci

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

That’s all for today!



Part II.
Quizzes

1 Deep Network Development - Quiz 1 Question Examples

1.1 Cheat Sheet

Activation functions:

$$\begin{aligned} \text{ReLU}(x) &= \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x <= 0 \end{cases} \\ \text{LeakyReLU}(x) &= \begin{cases} x, & \text{if } x > 0 \\ \gamma x, & \text{if } x <= 0, \text{ where } \gamma \text{ is a negative slope} \end{cases} \end{aligned} \tag{1}$$

Update Rules:

Gradient Descent:

$$\theta_t = \theta_{t-1} - \alpha \nabla J(\theta_{t-1}) \quad \text{where } \alpha \text{ denotes the } \textbf{learning rate} \tag{2}$$

Stochastic Gradient Descent with Momentum:

1. From lecture slides (Andrew Ng - Deep Learning Specialization)

$$\begin{aligned} m_t &= \beta m_{t-1} + (1-\beta)\nabla J(\theta_{t-1}), \quad \text{where } \beta \text{ denotes the } \textbf{momentum } (m_0 = \vec{0}) \\ \theta_t &= \theta_{t-1} - \alpha \cdot m_t, \qquad\qquad \text{where } \alpha \text{ denotes the } \textbf{learning rate} \end{aligned} \tag{3}$$

2. From Pytorch documentation (Dive into Deep Learning book)

$$\begin{aligned} b_t &= \beta b_{t-1} + \nabla J(\theta_{t-1}), \quad \text{where } \beta \text{ denotes the } \textbf{momentum } (b_0 = \vec{0}) \\ \theta_t &= \theta_t - \alpha \cdot b_t, \qquad\qquad \text{where } \alpha \text{ denotes the } \textbf{learning rate} \end{aligned} \tag{4}$$

Convolution Calculations:

Calculating the output shapes:

These are the same methods available at:

https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor \tag{5}$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor \tag{6}$$

1.2 Backpropagation

1.2.1 Example 1. (From lecture 2)

Consider the following model and loss function:

$$\hat{y} = \text{ReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \;\; y = 1, \qquad \theta_0 = 5, \;\; \theta_1 = -3, \qquad \alpha = 0.1$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Gradient Descent**.
4. Did the model improve?

1.2.2 Example 2. (From lecture 2)

Consider the following model and loss function:

$$\hat{y} = \text{ReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \;\; y = 1, \qquad \theta_0 = 5, \;\; \theta_1 = -1, \qquad \alpha = 0.1$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Gradient Descent**.
4. Did the model improve?

1.2.3 Example 3.

Consider the following model and loss function:

$$\hat{y} = \text{LeakyReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \;\; y = 1, \qquad \theta_0 = 5, \;\; \theta_1 = -3, \qquad \alpha = 0.1, \;\; \beta = 0, \;\; \gamma = 0.5$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Stochastic Gradient Descent** with **Momentum**.
4. Did the model improve?

1.2.4 Example 4.

Consider the following model and loss function:

$$\hat{y} = \text{LeakyReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \quad y = 1, \qquad \theta_0 = 5, \quad \theta_1 = -1, \qquad \alpha = 0.2, \quad \beta = 0.1, \quad \gamma = 0.7, \quad m_{0,\theta_0} = 2.5, \quad m_{0,\theta_1} = 7.6$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Stochastic Gradient Descent** with **Momentum**.
4. Did the model improve?

1.3 Convolution

1.4 Example 5.

Given the following code snippet calculate the output shape in (C, H, W) format!

```
m = nn.Conv2d(in_channels=12, out_channels=60, kernel_size=11, padding='valid', stride=4)
input = torch.randn(12, 567, 834)
m(input).shape

# Solution: torch.Size([60, 140, 206])
```

1.5 Example 6.

Given the following code snippet calculate the output shape in (C, H, W) format!

```
m = nn.Conv2d(in_channels=59, out_channels=3, kernel_size=3, padding='valid', stride=2)
input = torch.randn(59, 873, 304)
m(input).shape

# Solution: torch.Size([3, 436, 151])
```

1.6 Example 7.

Given the following code snippet calculate the output shape in (C, H, W) format! Calculate the value of the padding?

```
m = nn.Conv2d(in_channels=7, out_channels=60, kernel_size=15, padding='same')
i = torch.randn(7, 512, 512)
m(i).shape

# Solution: torch.Size([60, 512, 512]), padding=(7, 7)
```

1.7 Solutions

1.7.1 Solution 1.

Consider the following model and loss function:

$$\hat{y} = \text{ReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \quad y = 1, \qquad \theta_0 = 5, \quad \theta_1 = -3, \qquad \alpha = 0.1$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Gradient Descent**.
4. Did the model improve?

---

1. Calculate the predicted value:

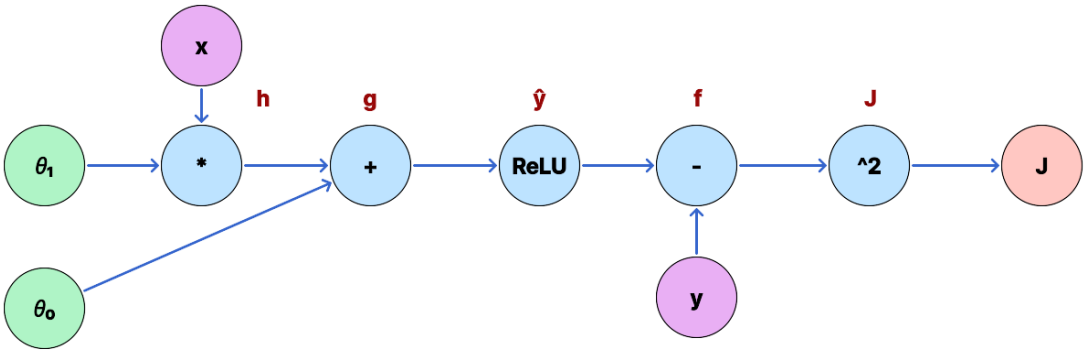
$$\begin{aligned} \hat{y} &= \text{ReLU}(5 + 2 \cdot (-3)) \\ &= \text{ReLU}(-1) \\ &= 0 \end{aligned}$$

2. Calculate the loss:

$$\begin{aligned} J(\theta) &= (\hat{y} - y)^2 \\ &= (0 - 1)^2 \\ &= (-1)^2 \\ &= 1 \end{aligned}$$

3. Update the model parameters using **Gradient Descent**:

3.1 Draw the **Directed Acyclic Graph** of **operations** and name the nodes:



3.2 Express the names:

$$
\begin{aligned}
J &= f^2 \\
f &= \hat{y} - y \\
\hat{y} &= \mathrm{ReLU}(g) \\
g &= h + \theta_0 \\
h &= \theta_1 \cdot x
\end{aligned}
$$

3.3 Symbolic differentiation:

$$
\begin{aligned}
J = f^2 \qquad & \frac{\partial J}{\partial f} = 2f \\
f = \hat{y} - y \qquad & \frac{\partial f}{\partial \hat{y}} = 1 \\
\hat{y} = \mathrm{ReLU}(g) \qquad & \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & g \le 0 \\ 1, & g > 0 \end{cases} \\
g = h + \theta_0 \qquad & \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1 \\
h = \theta_1 \cdot x \qquad & \frac{\partial h}{\partial x} = \theta_1, \qquad \frac{\partial h}{\partial \theta_1} = x
\end{aligned}
$$

$$
\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} \quad = ?
$$

$$
\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} \quad = ?
$$

3.4 Substitution - forward pass - backward pass:

$$J = f^2 \quad = 1 \qquad \frac{\partial J}{\partial f} = 2f \quad = -2$$

$$f = \hat{y} - y \quad = -1 \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = \text{ReLU}(g) \quad = 0 \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & g \leq 0 \\ 1, & g > 0 \end{cases} \quad = 0$$

$$g = h + \theta_0 \quad = -1 \qquad \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 \cdot x \quad = -6 \qquad \frac{\partial h}{\partial x} = \theta_1 \quad = -3, \qquad \frac{\partial h}{\partial \theta_1} = x \quad = 2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} \qquad = (-2) \cdot 1 \cdot 0 \cdot 1 = 0$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} \quad = (-2) \cdot 1 \cdot 0 \cdot 1 \cdot 2 = 0$$

3.5 Update weights

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = 5 - 0.1 \cdot 0 = 5$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = -3 - 0.1 \cdot 0 = -3$$

Our model did not improve!

1.7.2 Solution 2.

Consider the following model and loss function:

$$\hat{y} = \text{ReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \;\; y = 1, \qquad \theta_0 = 5, \;\; \theta_1 = -1, \qquad \alpha = 0.1$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Gradient Descent**.
4. Did the model improve?

---

1. Calculate the predicted value:

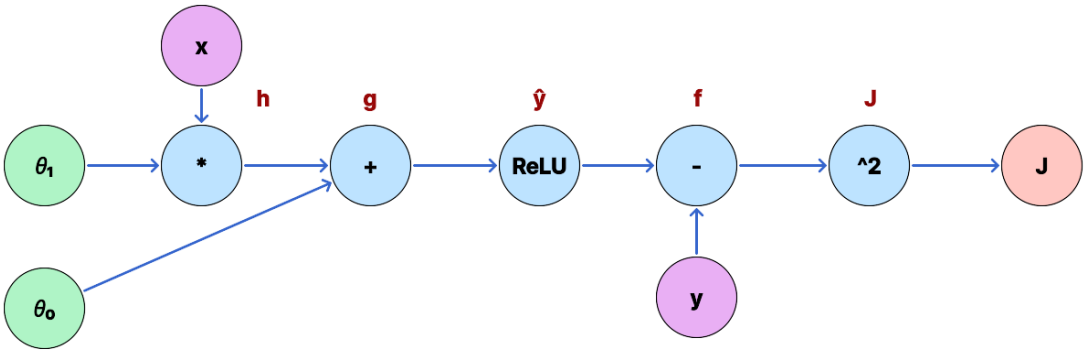
$$\begin{aligned} \hat{y} &= \text{ReLU}(5 + 2 \cdot (-1)) \\ &= \text{ReLU}(3) \\ &= 3 \end{aligned}$$

2. Calculate the loss:

$$\begin{aligned} J(\theta) &= (\hat{y} - y)^2 \\ &= (3 - 1)^2 \\ &= 2^2 \\ &= 4 \end{aligned}$$

3. Update the model parameters using **Gradient Descent**:

3.1 Draw the **Directed Acyclic Graph** of **operations** and name the nodes:



3.2 Express the names:

$$
\begin{aligned}
J &= f^2 \\
f &= \hat{y} - y \\
\hat{y} &= \mathrm{ReLU}(g) \\
g &= h + \theta_0 \\
h &= \theta_1 \cdot x
\end{aligned}
$$

3.3 Symbolic differentiation:

$$
\begin{aligned}
J = f^2 \qquad & \frac{\partial J}{\partial f} = 2f \\
f = \hat{y} - y \qquad & \frac{\partial f}{\partial \hat{y}} = 1 \\
\hat{y} = \mathrm{ReLU}(g) \qquad & \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & g \le 0 \\ 1, & g > 0 \end{cases} \\
g = h + \theta_0 \qquad & \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1 \\
h = \theta_1 \cdot x \qquad & \frac{\partial h}{\partial x} = \theta_1, \qquad \frac{\partial h}{\partial \theta_1} = x
\end{aligned}
$$

$$
\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} \quad = ?
$$

$$
\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} \quad = ?
$$

3.4 Substitution - forward pass - backward pass:

$$J = f^2 \quad = 4 \qquad \frac{\partial J}{\partial f} = 2f \quad = 4$$

$$f = \hat{y} - y \quad = 2 \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = \text{ReLU}(g) \quad = 3 \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & g \leq 0 \\ 1, & g > 0 \end{cases} \quad = 1$$

$$g = h + \theta_0 \quad = 3 \qquad \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 \cdot x \quad = -2 \qquad \frac{\partial h}{\partial x} = \theta_1 \quad = -1, \qquad \frac{\partial h}{\partial \theta_1} = x \quad = 2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} \qquad = 4 \cdot 1 \cdot 1 \cdot 1 = 4$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} \quad = 4 \cdot 1 \cdot 1 \cdot 1 \cdot 2 = 8$$

3.5 Update weights

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = 5 - 0.1 \cdot 4 = 4.6$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = -1 - 0.1 \cdot 8 = -1.8$$

Our model improved!

1.7.3 Solution 3.

Consider the following model and loss function:

$$\hat{y} = \text{LeakyReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \;\; y = 1, \qquad \theta_0 = 5, \;\; \theta_1 = -3, \qquad \alpha = 0.1, \;\; \beta = 0, \;\; \gamma = 0.5$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Stochastic Gradient Descent** with **Momentum**.
4. Did the model improve?

---

1. Calculate the predicted value:

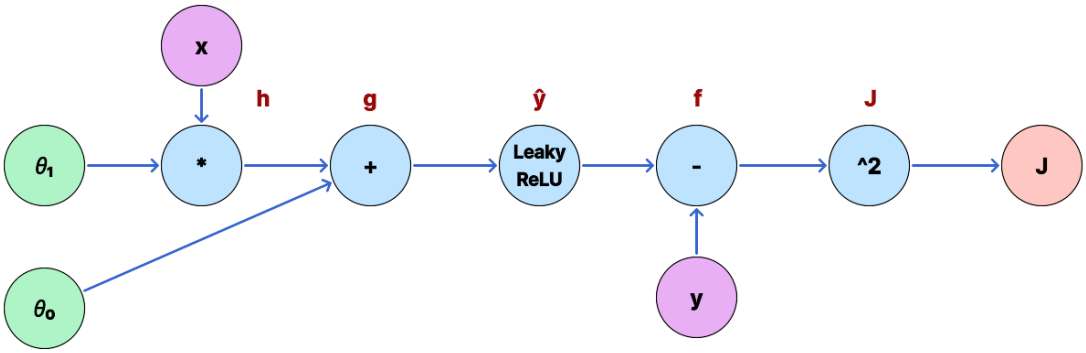
$$\begin{aligned} \hat{y} &= \text{LeakyReLU}(5 + 2 \cdot (-3)) \\ &= \text{LeakyReLU}(-1) \\ &= -0.5 \end{aligned}$$

2. Calculate the loss:

$$\begin{aligned} J(\theta) &= (\hat{y} - y)^2 \\ &= (-0.5 - 1)^2 \\ &= (-1.5)^2 \\ &= 2.25 \end{aligned}$$

3. Update the model parameters using **Stochastic Gradient Descent** with **Momentum**:

3.1 Draw the **Directed Acyclic Graph** of **operations** and name the nodes:



3.2 Express the names:

$$
\begin{aligned}
J &= f^2 \\
f &= \hat{y} - y \\
\hat{y} &= \text{LeakyReLU}(g) \\
g &= h + \theta_0 \\
h &= \theta_1 \cdot x
\end{aligned}
$$

3.3 Symbolic differentiation:

$$
\begin{aligned}
J &= f^2 && \frac{\partial J}{\partial f} = 2f \\
f &= \hat{y} - y && \frac{\partial f}{\partial \hat{y}} = 1 \\
\hat{y} &= \text{LeakyReLU}(g) && \frac{\partial \hat{y}}{\partial g} = \begin{cases} \gamma, & g \le 0 \\ 1, & g > 0 \end{cases} \\
g &= h + \theta_0 && \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1 \\
h &= \theta_1 \cdot x && \frac{\partial h}{\partial x} = \theta_1, \qquad \frac{\partial h}{\partial \theta_1} = x
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial J}{\partial \theta_0} &= \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} && = ? \\
\frac{\partial J}{\partial \theta_1} &= \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} && = ?
\end{aligned}
$$

3.4 Substitution - forward pass - backward pass:

$$J = f^2 \quad = 2.25 \qquad \frac{\partial J}{\partial f} = 2f \quad = -3$$

$$f = \hat{y} - y \quad = -1.5 \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = \text{LeakyReLU}(g) \quad = -0.5 \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} \gamma, & g \leq 0 \\ 1, & g > 0 \end{cases} \quad = 0.5$$

$$g = h + \theta_0 \quad = -1 \qquad \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 \cdot x \quad = -6 \qquad \frac{\partial h}{\partial x} = \theta_1 \quad = -3, \qquad \frac{\partial h}{\partial \theta_1} = x \quad = 2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} \qquad = (-3) \cdot 1 \cdot 0.5 \cdot 1 = -1.5$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} \quad = (-2) \cdot 1 \cdot 0.5 \cdot 1 \cdot 2 = -3$$

3.5 Update weights

$$m_{t,\theta_0} = \beta \cdot m_{t-1,\theta_0} + (1-\beta) \cdot \frac{\partial J}{\partial \theta_0} = 0 \cdot 0 + (1-0) \cdot (-1.5) = -1.5$$

$$\theta_0 = \theta_0 - \alpha \cdot m_{t,\theta_0} = 5 - 0.1 \cdot (-1.5) = 4.85$$

$$m_{t,\theta_1} = \beta \cdot m_{t-1,\theta_1} + (1-\beta) \cdot \frac{\partial J}{\partial \theta_1} = 0 \cdot 0 + (1-0) \cdot (-3) = -3$$

$$\theta_1 = \theta_1 - \alpha \cdot m_{t,\theta_1} = -3 - 0.1 \cdot (-3) = -2.7$$

Our model improved!

1.7.4 Solution 4.

Consider the following model and loss function:

$$\hat{y} = \text{LeakyReLU}(\theta_0 + x\theta_1), \qquad J(\theta) = (\hat{y} - y)^2$$

Initial values:

$$x = 2, \;\; y = 1, \qquad \theta_0 = 5, \;\; \theta_1 = -1, \qquad \alpha = 0.2, \;\; \beta = 0.1, \;\; \gamma = 0.7, \;\; m_{0,\theta_0} = 2.5, \;\; m_{0,\theta_1} = 7.6$$

Your task will be the following:

1. calculate the predicted value,
2. calculate the loss,
3. and update the model parameters using **Stochastic Gradient Descent** with **Momentum**.
4. Did the model improve?

---

1. Calculate the predicted value:

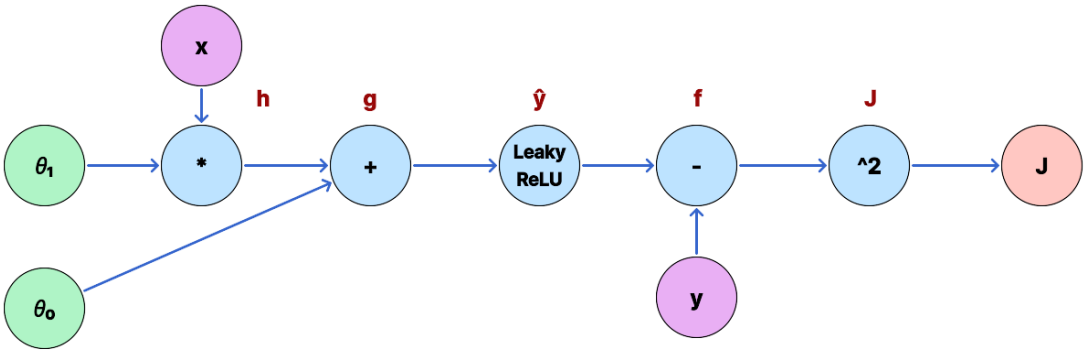
$$\begin{aligned}\hat{y} &= \text{LeakyReLU}(5 + 2 \cdot (-1)) \\ &= \text{LeakyReLU}(3) \\ &= 3\end{aligned}$$

2. Calculate the loss:

$$\begin{aligned}J(\theta) &= (\hat{y} - y)^2 \\ &= (3 - 1)^2 \\ &= 2^2 \\ &= 4\end{aligned}$$

3. Update the model parameters using **Stochastic Gradient Descent** with **Momentum**:

3.1 Draw the **Directed Acyclic Graph** of **operations** and name the nodes:



3.2 Express the names:

$$
\begin{aligned}
J &= f^2 \\
f &= \hat{y} - y \\
\hat{y} &= \text{LeakyReLU}(g) \\
g &= h + \theta_0 \\
h &= \theta_1 \cdot x
\end{aligned}
$$

3.3 Symbolic differentiation:

$$
\begin{aligned}
J = f^2 \qquad & \frac{\partial J}{\partial f} = 2f \\
f = \hat{y} - y \qquad & \frac{\partial f}{\partial \hat{y}} = 1 \\
\hat{y} = \text{LeakyReLU}(g) \qquad & \frac{\partial \hat{y}}{\partial g} = \begin{cases} \gamma, & g \le 0 \\ 1, & g > 0 \end{cases} \\
g = h + \theta_0 \qquad & \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1 \\
h = \theta_1 \cdot x \qquad & \frac{\partial h}{\partial x} = \theta_1, \qquad \frac{\partial h}{\partial \theta_1} = x
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial J}{\partial \theta_0} &= \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} && = ? \\
\frac{\partial J}{\partial \theta_1} &= \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} && = ?
\end{aligned}
$$

3.4 Substitution - forward pass - backward pass:

$$J = f^2 \quad = 4 \qquad \frac{\partial J}{\partial f} = 2f \quad = 4$$

$$f = \hat{y} - y \quad = 2 \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = \text{LeakyReLU}(g) \quad = 3 \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} \gamma, & g \leq 0 \\ 1, & g > 0 \end{cases} \quad = 1$$

$$g = h + \theta_0 \quad = 3 \qquad \frac{\partial g}{\partial h} = 1, \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 \cdot x \quad = -2 \qquad \frac{\partial h}{\partial x} = \theta_1 \quad = -1, \qquad \frac{\partial h}{\partial \theta_1} = x \quad = 2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} \qquad = 4 \cdot 1 \cdot 1 \cdot 1 = 4$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} \qquad = 4 \cdot 1 \cdot 1 \cdot 1 \cdot 2 = 8$$

3.5 Update weights

$$m_{t,\theta_0} = \beta \cdot m_{t-1,\theta_0} + (1-\beta) \cdot \frac{\partial J}{\partial \theta_0} = 2.5 \cdot 0.1 + (1 - 0.1) \cdot 4 = 3.85$$

$$\theta_0 = \theta_0 - \alpha \cdot m_{t,\theta_0} = 5 - 0.2 \cdot 3.85 = 4.23$$

$$m_{t,\theta_1} = \beta \cdot m_{t-1,\theta_1} + (1-\beta) \cdot \frac{\partial J}{\partial \theta_1} = 7.6 \cdot 0.1 + (1 - 0.1) \cdot 8 = 7.96$$

$$\theta_1 = \theta_1 - \alpha \cdot m_{t,\theta_1} = -1 - 0.2 \cdot 7.96 = -2.59$$

Our model improved!

1.7.5 Solution 5.

Given the following code snippet calculate the output shape in (C, H, W) format!

```
m = nn.Conv2d(in_channels=12, out_channels=60, kernel_size=11, padding='valid', stride=4)
input = torch.randn(12, 567, 834)
m(input).shape

# Solution: torch.Size([60, 140, 206])
```

$$
\begin{aligned}
H_{out} &= \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor \\
&= \left\lfloor \frac{567 + 2 \times 0 - 1 \times (11 - 1) - 1}{4} + 1 \right\rfloor \\
&= 140 \\
W_{out} &= \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor \\
&= \left\lfloor \frac{834 + 2 \times 0 - 1 \times (11 - 1) - 1}{4} + 1 \right\rfloor \\
&= 206
\end{aligned}
$$

So the output shape is: **(60, 140, 206)**

1.7.6 Solution 6.

Given the following code snippet calculate the output shape in (C, H, W) format!

```
m = nn.Conv2d(in_channels=59, out_channels=3, kernel_size=3, padding='valid', stride=2)
input = torch.randn(59, 873, 304)
m(input).shape

# Solution: torch.Size([3, 436, 151])
```

$$
\begin{aligned}
H_{out} &= \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor \\
&= \left\lfloor \frac{873 + 2 \times 0 - 1 \times (3 - 1) - 1}{2} + 1 \right\rfloor \\
&= 436
\end{aligned}
$$

$$
\begin{aligned}
W_{out} &= \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor \\
&= \left\lfloor \frac{304 + 2 \times 0 - 1 \times (3 - 1) - 1}{2} + 1 \right\rfloor \\
&= 151
\end{aligned}
$$

So the output shape is: **(3, 436, 151)**

1.7.7 Solution 7.

Given the following code snippet calculate the output shape in (C, H, W) format! Calculate the value of the padding?

```
m = nn.Conv2d(in_channels=7, out_channels=60, kernel_size=15, padding='same')
i = torch.randn(7, 512, 512)
m(i).shape

# Solution: torch.Size([60, 512, 512]), padding=(7, 7)
```

Because the padding is **same** the output **width** and **height** and the input **width** and **height** will be the same. The value of the channel is base on **out_channels**. Thus to calculate the value:

$$
\begin{aligned}
512 &= \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor \\
512 &= \left\lfloor \frac{512 + 2 \times \text{padding}[0] - 1 \times (15 - 1) - 1}{1} + 1 \right\rfloor \\
512 &= \left\lfloor \frac{497 + 2 \times \text{padding}[0]}{1} + 1 \right\rfloor \\
512 &= \left\lfloor 498 + 2 \times \text{padding}[0] \right\rfloor \\
14 &= \left\lfloor 2 \times \text{padding}[0] \right\rfloor \\
7 &= \text{padding}[0]
\end{aligned}
$$

$$
\begin{aligned}
512 &= \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor \\
512 &= \left\lfloor \frac{512 + 2 \times \text{padding}[1] - 1 \times (15 - 1) - 1}{1} + 1 \right\rfloor \\
512 &= \left\lfloor \frac{497 + 2 \times \text{padding}[1]}{1} + 1 \right\rfloor \\
512 &= \left\lfloor 498 + 2 \times \text{padding}[1] \right\rfloor \\
14 &= \left\lfloor 2 \times \text{padding}[1] \right\rfloor \\
7 &= \text{padding}[1]
\end{aligned}
$$

So the output shape is: **(3, 512, 512)**. The value of the padding is **(7, 7)**. This is equivalent to the following parameterization of **Conv2d**:

```
m = nn.Conv2d(in_channels=7, out_channels=60, kernel_size=15, padding=(7, 7), stride=1)
i = torch.randn(7, 512, 512)
m(i).shape
```

1 Deep Network Development - Quiz 2 Question Examples

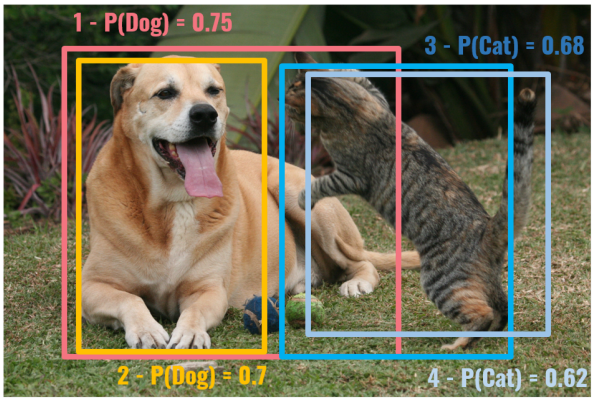
1.1 Cheat Sheet

Non-Maximum Suppression algorithm:

Non-Maximum Suppression (NMS) removes duplicate detections:

1. Select the bounding box with the highest confidence score.
2. Remove other boxes that overlap it above an IoU threshold (i.e. 0.5).
3. Repeat with the remaining boxes.

The result keeps the strongest box for each detected object.



Cat and Dog Game.jpg https://commons.wikimedia.org/wiki/File:Cat_and_Dog_Game.jpg Faypearse, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

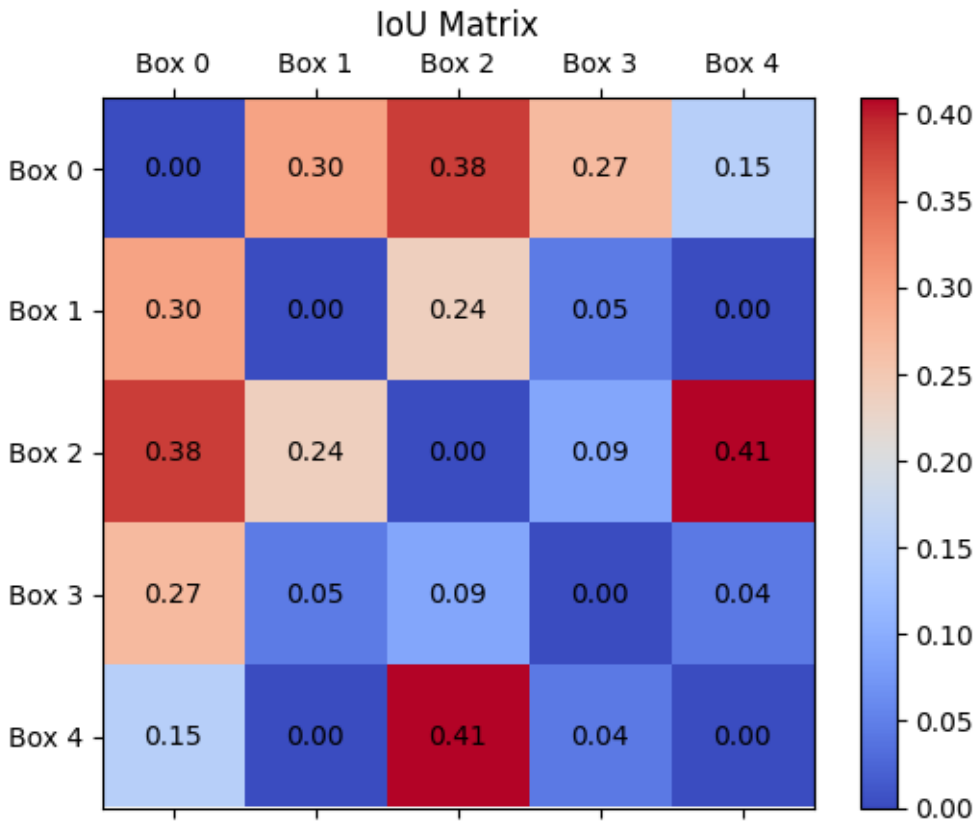
1.2 Examples

1.2.1 Example 1.

Given the following **5 bounding boxes**, **IoU between all the bounding boxes**, and **IoU threshold 0.2** select the bounding boxes that remain after applying **Non-Max Suppression**:

The bounding box scores:

- **Box 0** score: **0.7**
- **Box 1** score: **0.55**
- **Box 2** score: **0.86**
- **Box 3** score: **0.93**
- **Box 4** score: **0.62**



1.2.2 Solution 1.

1. Sort boxes based on their scores: **[Box 3, Box 2, Box 0, Box 4, Box 1]**
2. Select the highest scoring box: **Box 3**
3. Eliminate lower scoring boxes with **IoU > threshold** (in this example it is **0.2**). The **IoU** for each lower-scoring boxes compared to **Box 3**:

	Box 0	**Box 1**	**Box 2**	**Box 3**	**Box 4**
Box 3	0.27	0.05	0.09	-	0.04

Based on these we eliminate **Box 0**, because its IoU is above the 0.2 threshold

4. Now our list looks like this: **[Box 3, Box 2, ~~Box 0~~, Box 4, Box 1]**
5. Select next highest scoring box: **Box 2**
6. Eliminate lower scoring boxes with **IoU > threshold** (in this example it is **0.2**). The **IoU** for each lower-scoring boxes compared to **Box 2**:

	Box 0	**Box 1**	**Box 2**	**Box 3**	**Box 4**
Box 2	0.38	0.24	-	0.09	0.41

Based on these we eliminate **Box 1** and **Box 4**, because its IoU is above the 0.2 threshold

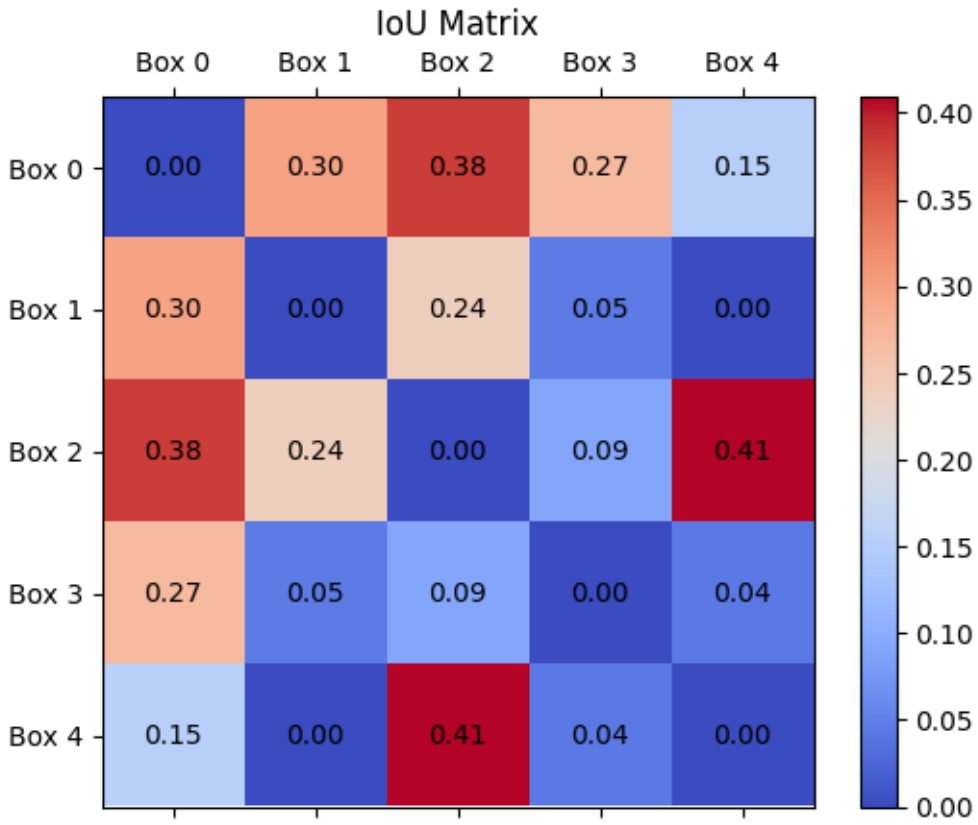
7. Now our list looks like this: **[Box 3, Box 2, ~~Box 0~~, ~~Box 4~~, ~~Box 1~~]**
8. Since there are no more boxes left the final answer is that **Box 3** and **Box 2** remain after applying Non-Maximum Suppression.

1.2.3 Example 2.

Given the following **5 bounding boxes**, **IoU between all the bounding boxes**, and **IoU threshold 0.2** select the bounding boxes that remain after applying **Non-Max Suppression**:

The bounding box scores:

- **Box 0** score: **0.94**
- **Box 1** score: **0.58**
- **Box 2** score: **0.88**
- **Box 3** score: **0.69**
- **Box 4** score: **0.95**



1. Sort boxes based on their scores: **[Box 4, Box 0, Box 2, Box 3, Box 1]**
2. Select the highest scoring box: **Box 4**
3. Eliminate lower scoring boxes with **IoU > threshold** (in this example it is **0.2**). The **IoU** for each lower-scoring boxes compared to **Box 4**:

	Box 0	**Box 1**	**Box 2**	**Box 3**	**Box 4**
Box 4	0.15	0.00	0.41	0.04	-

Based on these we eliminate **Box 2**, because their IoU is above the 0.2 threshold

4. Now our list looks like this: **[Box 4, Box 0, ~~Box 2~~, Box 3, Box 1]**
5. Select next highest scoring box: **Box 0**
6. Eliminate lower scoring boxes with **IoU > threshold** (in this example it is **0.2**). The **IoU** for each lower-scoring boxes compared to **Box 0**:

	Box 0	**Box 1**	**Box 2**	**Box 3**	**Box 4**
Box 0	-	0.3	0.38	0.27	0.15

Based on these we eliminate **Box 1** and **Box 3**, because their IoU is above the 0.2 threshold

7. Now our list looks like this: **[Box 4, Box 0, ~~Box 2~~, ~~Box 3~~, ~~Box 1~~]**,
8. Since there are no more boxes left the final answer is that **Box 4** and **Box 0** remain after applying Non-Maximum Suppression.

Part III.
Midterms

Name: _____

Neptun: _____

Deep Network Development
Midterm 1/A
2025 October 21

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

Tasks

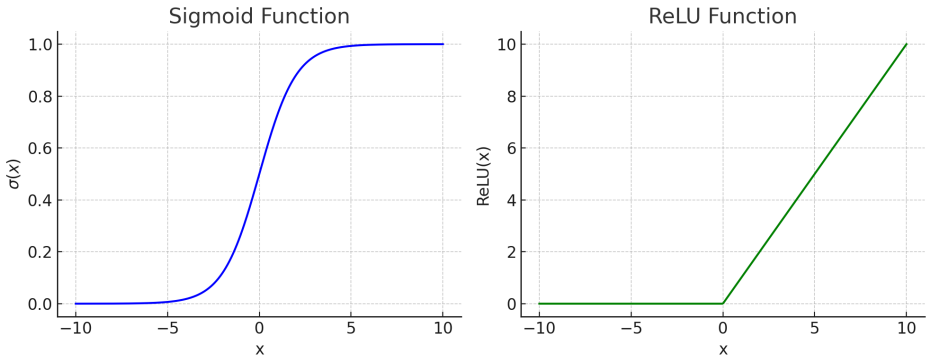
1. (10 points) Activation functions play a central role in enabling deep neural networks to learn complex representations.
   (a) (3 points) Consider a multi-layer perceptron (MLP) composed of linear layers. What would happen if all activation functions were removed from the network? What type of hypothesis function would the resulting model represent?
   (b) (4 points) Draw the shape of the activation functions listed below. Indicate clearly their behavior on negative, zero, and positive inputs.
   - $\sigma(x) = \dfrac{1}{1 + e^{-x}}$
   - $\text{ReLU}(x) = \max(0, x)$

   (c) (3 points) Briefly describe the role and benefits of sigmoid and ReLU in modern deep neural networks. In what contexts are they typically used, and why?

Solution:

(a): If all activation functions are removed from a multi-layer perceptron (MLP), the network becomes a series of linear transformations. Since a composition of linear functions is itself linear, the resulting model is equivalent to a single linear transformation of the input. Therefore, the hypothesis function becomes a linear model.

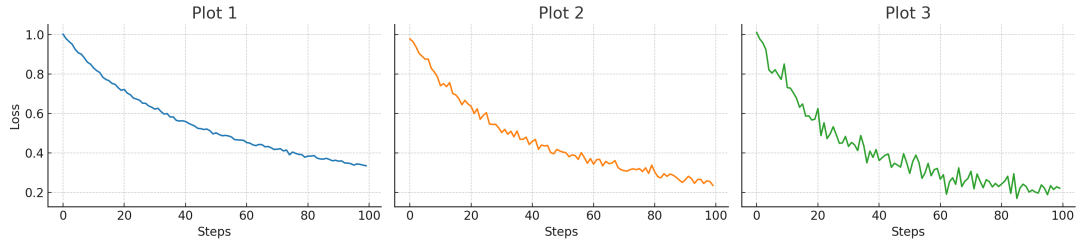
(b): Students are expected to draw:



(c): Sigmoid: Used primarily in binary classification tasks and as the output activation for logistic regression or binary classification heads in neural networks. It compresses input values into the $(0, 1)$ range, making it interpretable as a probability.

ReLU: The most commonly used activation function in modern deep learning. It is computationally efficient and helps mitigate the vanishing gradient problem by having a constant gradient of 1 for positive values. ReLU enables sparse activation and works well in hidden layers of deep networks.

2. (10 points) Stochastic Gradient Descent is a widely used optimization technique in deep learning, particularly effective when the loss function is differentiable.
   (a) (2 points) What does the term *stochastic* refer to in the context of gradient descent? How does it differ from batch gradient descent?
   (b) (3 points) What is the goal of gradient descent? List and briefly describe two important hyperparameters that affect its behavior.
   (c) (5 points) The three training curves shown below visualize the behavior of the loss over training steps under different batch sizes. Assume the same model, optimizer type, and learning rate were used.



Match each plot to the correct batch size.

Solution:

(a): The term stochastic in stochastic gradient descent refers to the use of random subsets (often of size 1) of the training data to compute each gradient update. Unlike batch gradient descent, which computes gradients using the full dataset at each step, SGD introduces randomness and more frequent updates, allowing it to escape local minima and saddle points more effectively, albeit with noisier convergence.

(b): The goal of gradient descent is to minimize the loss function by updating the model's parameters in the direction of the negative gradient. Two important hyperparameters are:

- **Learning rate:** Controls the step size during parameter updates. A value too large may cause divergence, while a value too small may slow convergence.
- **Batch size:** Determines how many samples are used to compute the gradient. Smaller batches introduce noise but allow more frequent updates, while larger batches reduce variance but require more memory.

(c): Matching and justification:

- **Plot 1 → B (Small Batch Size):** Shows highly noisy updates with significant fluctuation in loss, characteristic of stochastic behavior with low batch sizes.
- **Plot 2 → C (Medium Batch Size):** Displays moderately noisy updates with a balance between stability and variability.
- **Plot 3 → A (Large Batch Size):** Demonstrates smooth and stable convergence, typical of full-batch training where gradients are averaged across many samples.

3. (10 points) Convolutional Neural Networks (CNNs)
   (a) (3 points) Define translation invariance in the context of image classification. Is the convolution operation inherently translation invariant?
   (b) (3 points) What is translation equivariance? Is the convolution operation translation equivariant?
   (c) (2 points) Suppose a CNN perfectly classifies a digit at the center of a 28×28 image. What might happen if you shift the digit 4 pixels to the left? Explain using the concepts of equivariance and invariance.
   (d) (2 points) Which of the following statements correctly distinguishes equivariance from invariance in the context of CNNs? Circle the correct answers.
      1. Equivariance implies that applying a transformation to the input results in a corresponding transformation of the output.
      2. Invariance implies that any transformation applied to the input yields an identical output.
      3. Equivariance ensures the output is completely unchanged when the input is transformed.
      4. Invariance means the model explicitly learns how to counteract transformations through convolutional filters.

Solution:

(a): Translation invariance means that when the input is shifted, the output remains unchanged. The convolution operation itself is **not** inherently translation invariant. Instead, convolution is translation equivariant, which means the feature maps shift in the same way as the input. True invariance requires additional mechanisms (e.g., pooling, data augmentation, or positional encoding).

(b): Translation equivariance means that a shift in the input causes an equivalent shift in the output. Convolutions satisfy this property, making them well-suited for detecting local patterns consistently across spatial locations. This property helps the CNN detect features like edges or corners in various parts of the image.

(c): If a CNN is trained on digits centered in the image, shifting a digit to the left may reduce accuracy. This is because although convolution layers are translation equivariant, the network as a whole (especially with fully connected layers at the end) may not be invariant. Without appropriate translation-invariant mechanisms (e.g., global pooling), classification performance may degrade when input location changes.

(d): Correct answers: (1, 2)

4. (10 points) Convolution Operation
   (a) (3 points) Explain the convolution operation and how it is implemented within convolutional layers in neural networks.
   (b) (3 points) Identify and describe the key components and hyperparameters of a convolutional layer.
   (c) (4 points) Given the input matrix and convolutional kernel below, compute the resulting feature map using a 2D convolution with $s = 2$ and $p = 0$. Show all steps clearly.

$$\text{Input: } \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \\ 2 & 3 & 0 & 1 \\ -2 & 0 & 1 & 2 \\ -1 & 3 & 0 & 1 \end{bmatrix} \qquad \text{Kernel: } \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ -1 & 0 \end{bmatrix}$$

Solution:

(a): Convolution is a mathematical operation that combines two functions to produce a third function expressing how one shape is modified by the other. In image processing, a small matrix called a kernel or filter slides over the input image to compute a weighted sum, producing a feature map.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

where I is the input image, K is the kernel, and S is the resulting feature map.

In Convolutional Neural Networks (CNNs), convolutional layers apply multiple learnable filters to the input. Each filter scans across the spatial dimensions, computing dot products with patches of the input and producing a separate feature map. These feature maps are stacked to form the output.

(b): Key components and hyperparameters:

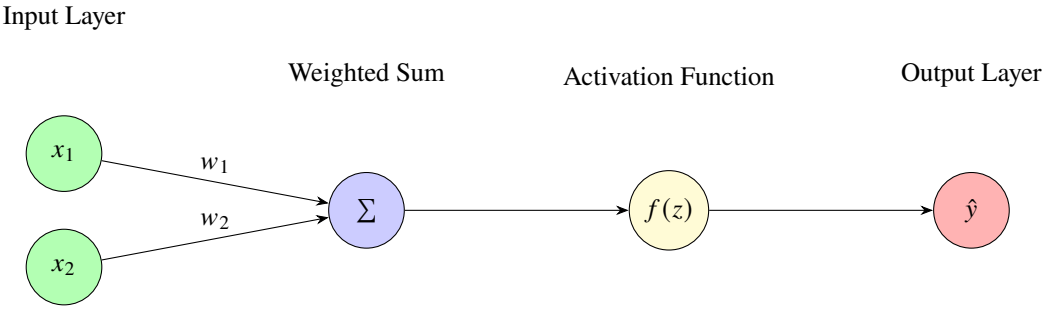
- **Kernel size**: Dimensions of the filter (e.g., 3×3, 5×5).
- **Stride**: The number of pixels the filter moves across (e.g., stride 1).
- **Padding**: Zeros added around the input to control output size (e.g., 'same', 'valid').
- **Number of filters**: Determines the number of output channels.

(c): Resulting feature map with a 3×2 kernel applied to a 6×4 input (stride 2, no padding):

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \\ 2 & 3 & 0 & 1 \\ -2 & 0 & 1 & 2 \\ -1 & 3 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ -1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 9 & 20 \\ 25 & 8 \end{bmatrix}$$

5. (10 points) Backpropagation

The following diagram shows a simple single-layer neural network with two inputs and one output:



You are given the following:

- Inputs: $x_1 = 2.0, x_2 = -1.0$
- Weights: $w_1 = -0.4, w_2 = 0.5$
- Bias: $b = 0$
- Activation function: Identity ($f(z) = z$)
- Target output: $y = 2.0$
- Learning rate: $\eta = 0.1$

Compute one weight update according to the following steps:

(a) (2 points) Perform the forward pass and compute the prediction $\hat{y}$.

(b) (3 points) Compute the loss using Mean Squared Error: $L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

(c) (3 points) Compute the gradient of the loss with respect to w_1 and w_2.

(d) (2 points) Update w_1 and w_2 using gradient descent.

Solution: (a): Forward Pass

We compute the linear output:

$$z = x_1 w_1 + x_2 w_2 + b = 2.0 \cdot (-0.4) + (-1.0) \cdot 0.5 + 0 = -0.8 - 0.5 = -1.3$$

$$\hat{y} = f(z) = z = -1.3$$

(b): Compute Loss

Using MSE:

$$L = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(2.0 - (-1.3))^2 = \frac{1}{2}(3.3)^2 = \frac{1}{2} \cdot 10.89 = 5.445$$

(c): Compute Gradients

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = -1.3 - 2.0 = -3.3$$

$$\frac{\partial L}{\partial w_1} = -3.3 \cdot x_1 = -3.3 \cdot 2.0 = -6.6$$

$$\frac{\partial L}{\partial w_2} = -3.3 \cdot x_2 = -3.3 \cdot (-1.0) = 3.3$$

(d): Weight Update

$$w_1 \leftarrow w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = -0.4 - 0.1 \cdot (-6.6) = -0.4 + 0.66 = \boxed{0.26}$$

$$w_2 \leftarrow w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = 0.5 - 0.1 \cdot 3.3 = 0.5 - 0.33 = \boxed{0.17}$$

Final Answer:

$$w_1 = 0.26, \quad w_2 = 0.17$$

Name: _____

Neptun: _____

Deep Network Development
Midterm 1/B
2025 October 21

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

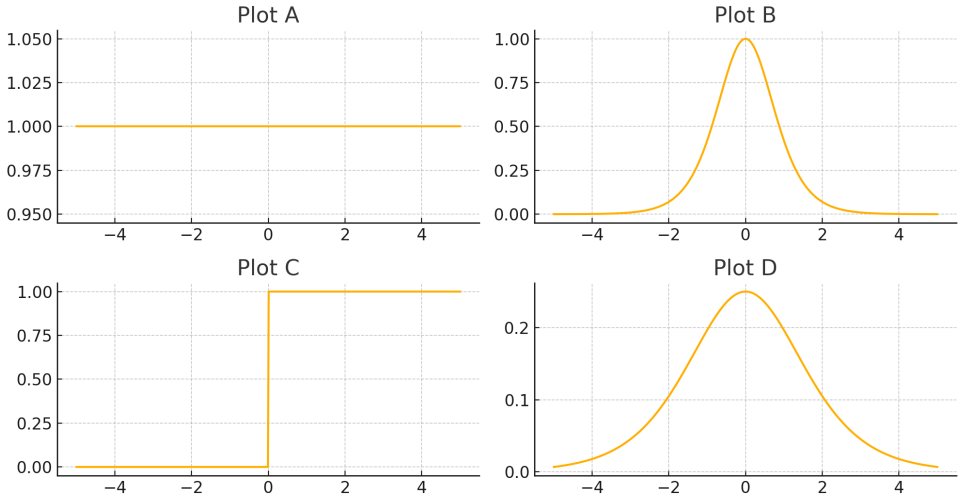
Exam Tasks

1. (10 points) Linear Regression/Logistic Regression/Multi-layer Perceptron
   (a) (2 points) Write the general formula for a linear regression model. Clearly explain the role of each component in terms of inputs, parameters, and outputs (vectorized form).
   (b) (2 points) **Although linear regression is designed for regression problems, it can be applied to binary classification problems (without modifying its hypothesis function). Describe how this can be done, and highlight potential drawbacks of using it in classification tasks.**
   (c) (3 points) Provide the formula for logistic regression. Explain how applying the sigmoid function addresses the limitations of linear regression.
   (d) (3 points) **Consider a binary classification dataset where the two classes are not linearly separable. Sketch or plot the behavior of a logistic regression decision boundary versus that of a Multi-Layer Perceptron. What differences emerge?**

2. (10 points) Activation functions play a key role in introducing non-linearity into neural networks.
   (a) (2 points) Why is it important to know the derivative of an activation function when training neural networks?
   (b) (8 points) Below you are given the equations of four activation functions and four unlabeled plots (A, B, C, D) showing their respective derivatives.
   Match each function to the correct plot of its derivative. Explain your choices.

   Activation Functions:

   (1) $f(x) = \dfrac{1}{1+e^{-x}}$ (Sigmoid)
   (2) $f(x) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ (Tanh)
   (3) $f(x) = \max(0, x)$ (ReLU)
   (4) $f(x) = x$ (Linear / Identity)



Write your answers clearly, e.g., (1) → A, (2) → D, ...

3. (10 points) Regularization is a fundamental concept in deep learning that aims to improve a model's ability to generalize to unseen data.
   (a) (2 points) What is weight regularization, and to which component of the training objective is it applied? Briefly explain its purpose.
   (b) (5 points) Match each regularization technique below to its corresponding formulation or description.
   Match the Regularization Techniques to their Descriptions:

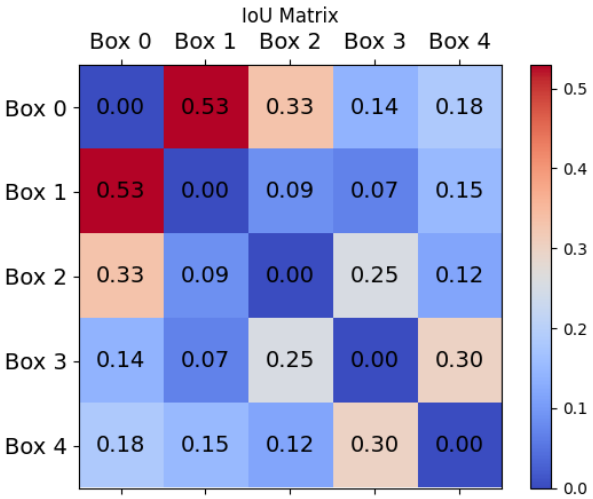
Technique	**Description**
(1) L1 Regularization	(A) Terminates training when validation loss stops improving
(2) L2 Regularization	(B) Constrains weight magnitudes to remain within a fixed range
(3) Dropout	(C) Applies a random binary mask to layer outputs
(4) Early Stopping	(D) Adds $\lambda \sum_i \lvert w_i \rvert$ to the loss
(5) Weight Clipping	(E) Adds $\lambda \sum_i w_i^2$ to the loss

   Briefly justify one of your choices.
   (c) (3 points) Among the listed regularization methods, which are active only during training and which affect the model's structure or behavior at inference time as well? Provide examples and justify your classification.

4. (10 points) Non-Maximum Suppression (NMS)
   (a) (3 points) Briefly explain how the Intersection over Union is computed between two bounding boxes. Use a sketch or describe the formula.
   (b) (7 points) The figure at right shows five predicted bounding boxes, their confidence scores, and a table of pairwise IoU values. Given an IoU threshold of 0.2, apply Non-Maximum Suppression to determine which bounding boxes should be retained. State the steps taken.

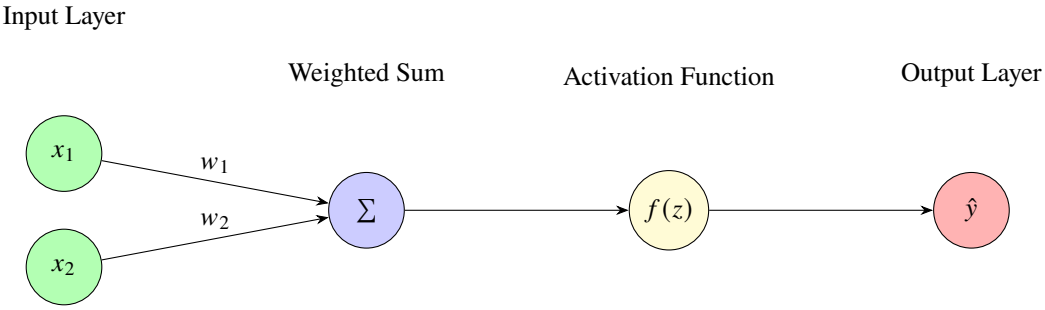
Confidence Scores

Box 0	**Box 1**	**Box 2**	**Box 3**	**Box 4**
0.81	0.54	0.76	0.82	0.96



5. (10 points) Backpropagation

The following diagram shows a simple single-layer neural network with two inputs and one output:



You are given the following:

- Inputs: $x_1 = 1.0, x_2 = 2.0$
- Weights: $w_1 = 0.1, w_2 = -0.2$
- Bias: $b = 0$
- Activation function: Identity ($f(z) = z$)
- Target output: $y = 1.0$
- Learning rate: $\eta = 0.1$

Compute one weight update according to the following steps:

(a) (2 points) Perform the forward pass and compute the prediction $\hat{y}$.

(b) (3 points) Compute the loss using Mean Squared Error: $L = \frac{1}{2}(y - \hat{y})^2$

(c) (3 points) Compute the gradient of the loss with respect to w_1 and w_2.

(d) (2 points) Update w_1 and w_2 using gradient descent.

Name: _____

Neptun: _____

Deep Network Development
Midterm 1 Retake
2025 October 21

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

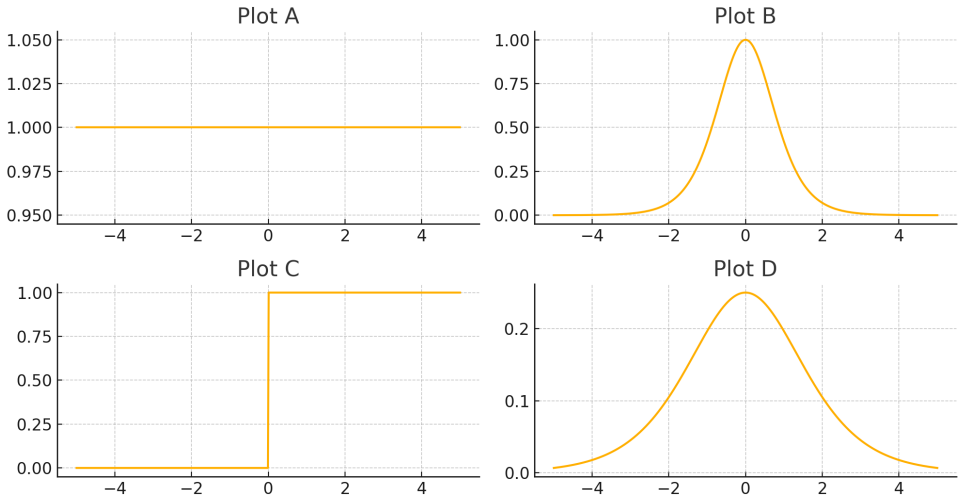
Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

Tasks

1. (10 points) Activation functions play a key role in introducing non-linearity into neural networks.
   (a) (2 points) Why is it important to know the derivative of an activation function when training neural networks?
   (b) (8 points) Below you are given the equations of four activation functions and four unlabeled plots (A, B, C, D) showing their respective derivatives.
   Match each function to the correct plot of its derivative. Explain your choices.

   Activation Functions:

   $$(1) \quad f(x) = \frac{1}{1 + e^{-x}} \qquad \text{(Sigmoid)}$$
   $$(2) \quad f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad \text{(Tanh)}$$
   $$(3) \quad f(x) = \max(0, x) \qquad \text{(ReLU)}$$
   $$(4) \quad f(x) = x \qquad \text{(Linear / Identity)}$$



   Write your answers clearly, e.g., (1) → A, (2) → D, ...

Solution:

(a): Knowing the derivative of an activation function is essential key for backpropagation, as gradients are propagated through each layer during training. The shape and properties of these derivatives directly affect gradient flow and convergence.

(b): Matching each derivative plot to its corresponding activation function:

$$\text{Plot A} \rightarrow (4) \quad f(x) = x \quad \text{(derivative is constant 1)}$$
$$\text{Plot B} \rightarrow (2) \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$\text{Plot C} \rightarrow (3) \quad \text{ReLU}(x) = \max(0, x)$$
$$\text{Plot D} \rightarrow (1) \quad \sigma(x) = \frac{1}{1 + e^{-x}}$$

2. (10 points) Regularization is a fundamental concept in deep learning that aims to improve a model's ability to generalize to unseen data.
   (a) (2 points) What is weight regularization, and to which component of the training objective is it applied? Briefly explain its purpose.
   (b) (5 points) Match each regularization technique below to its corresponding formulation or description.
   Match the Regularization Techniques to their Descriptions:

Technique	**Description**
(1) L1 Regularization	(A) Terminates training when validation loss stops improving
(2) L2 Regularization	(B) Constrains weight magnitudes to remain within a fixed range
(3) Dropout	(C) Applies a random binary mask to layer outputs
(4) Early Stopping	(D) Adds $\lambda \sum_i \lvert w_i \rvert$ to the loss
(5) Weight Clipping	(E) Adds $\lambda \sum_i w_i^2$ to the loss

   Briefly justify one of your choices.
   (c) (3 points) Among the listed regularization methods, which are active only during training and which affect the model's structure or behavior at inference time as well? Provide examples and justify your classification.

Solution:

(a): Weight regularization refers to the addition of a penalty term to the loss function that discourages large weight values. It is applied directly to the model's loss function during training. The purpose is to reduce model complexity, help prevent overfitting, and improve generalization by pushing weights toward smaller magnitudes.

(b): Matching Answers:

Regularization Technique	**Matches Description**
(1) L1 Regularization	(D) Adds $\lambda \sum_i \lvert w_i \rvert$ to the loss
(2) L2 Regularization	(E) Adds $\lambda \sum_i w_i^2$ to the loss
(3) Dropout	(C) Applies a random binary mask to layer outputs
(4) Early Stopping	(A) Terminates training when validation loss stops improving
(5) Weight Clipping	(B) Constrains weight magnitudes to remain within a fixed range

(c): Training-only techniques:

- **Dropout**: only active during training; units are randomly deactivated to simulate ensembling.
- **Early Stopping**: stops training based on validation performance but does not modify the final model architecture.
- **Weight Clipping**: constraints applied during training updates.
- **L1 and L2 Regularization**: influence the final learned weights that remain in the model during inference.

3. (10 points) Convolutional Neural Networks (CNNs)
   (a) (3 points) Define translation invariance in the context of image classification. Is the convolution operation inherently translation invariant?
   (b) (3 points) What is translation equivariance? Is the convolution operation translation equivariant?
   (c) (2 points) Suppose a CNN perfectly classifies a digit at the center of a 28×28 image. What might happen if you shift the digit 4 pixels to the left? Explain using the concepts of equivariance and invariance.
   (d) (2 points) Which of the following statements correctly distinguishes equivariance from invariance in the context of CNNs? Circle the correct answers.
      1. Equivariance implies that applying a transformation to the input results in a corresponding transformation of the output.
      2. Invariance implies that any transformation applied to the input yields an identical output.
      3. Equivariance ensures the output is completely unchanged when the input is transformed.
      4. Invariance means the model explicitly learns how to counteract transformations through convolutional filters.

Solution:

(a): Translation invariance means that when the input is shifted, the output remains unchanged. The convolution operation itself is **not** inherently translation invariant. Instead, convolution is translation equivariant, which means the feature maps shift in the same way as the input. True invariance requires additional mechanisms (e.g., pooling, data augmentation, or positional encoding).

(b): Translation equivariance means that a shift in the input causes an equivalent shift in the output. Convolutions satisfy this property, making them well-suited for detecting local patterns consistently across spatial locations. This property helps the CNN detect features like edges or corners in various parts of the image.

(c): If a CNN is trained on digits centered in the image, shifting a digit to the left may reduce accuracy. This is because although convolution layers are translation equivariant, the network as a whole (especially with fully connected layers at the end) may not be invariant. Without appropriate translation-invariant mechanisms (e.g., global pooling), classification performance may degrade when input location changes.

(d): Correct answers: (1, 2)

4. (10 points) Convolution Operation
   (a) (3 points) Explain the convolution operation and how it is implemented within convolutional layers in neural networks.
   (b) (3 points) Identify and describe the key components and hyperparameters of a convolutional layer.
   (c) (4 points) Given the input matrix and convolutional kernel below, compute the resulting feature map using a 2D convolution with $s = 2$ and $p = 0$. Show all steps clearly.

$$\text{Input:} \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \\ 2 & 3 & 0 & 1 \\ -2 & 0 & 1 & 2 \\ -1 & 3 & 0 & 1 \end{bmatrix} \qquad \text{Kernel:} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ -1 & 0 \end{bmatrix}$$

Solution:

(a): Convolution is a mathematical operation that combines two functions to produce a third function expressing how one shape is modified by the other. In image processing, a small matrix called a kernel or filter slides over the input image to compute a weighted sum, producing a feature map.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

where I is the input image, K is the kernel, and S is the resulting feature map.

In Convolutional Neural Networks (CNNs), convolutional layers apply multiple learnable filters to the input. Each filter scans across the spatial dimensions, computing dot products with patches of the input and producing a separate feature map. These feature maps are stacked to form the output.

(b): Key components and hyperparameters:

- **Kernel size**: Dimensions of the filter (e.g., 3×3, 5×5).
- **Stride**: The number of pixels the filter moves across (e.g., stride 1).
- **Padding**: Zeros added around the input to control output size (e.g., 'same', 'valid').
- **Number of filters**: Determines the number of output channels.

(c): Resulting feature map with a 3×2 kernel applied to a 6×4 input (stride 2, no padding):

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \\ 2 & 3 & 0 & 1 \\ -2 & 0 & 1 & 2 \\ -1 & 3 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ -1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 9 & 20 \\ 25 & 8 \end{bmatrix}$$

5. (10 points) Backpropagation

The following diagram shows a simple single-layer neural network with two inputs and one output:

You are given the following scalar regression model and hyperparameters:

$$\hat{y} = \text{LeakyReLU}(\theta_0 + \theta_1 x), \qquad \text{LeakyReLU}(z) = \begin{cases} z, & z \geq 0 \\ \text{negative\_slope} \cdot z, & z < 0 \end{cases}$$

$$J(\theta) = (\hat{y} - y)^2$$

- Input: $x = 2$
- Target output: $y = 4$
- Parameters: $\theta_1 = 2$ (slope), $\theta_0 = 4$ (intercept)
- Activation function: LeakyReLU with negative_slope = 0.5
- Loss function: Squared Error (SE), $J(\theta) = (\hat{y} - y)^2$

- Learning rate: learning_rate = 0.1

Compute one parameter update according to the following steps:

(a) (2 points) Perform the forward pass and compute the prediction $\hat{y}$.

(b) (3 points) Compute the loss using Squared Error: $J(\theta) = (\hat{y} - y)^2$.

(c) (3 points) Compute the gradient of the loss with respect to θ_0 and θ_1.

(d) (2 points) Update θ_0 and θ_1 using Gradient Descent.

Solution:

(a): Forward Pass

First compute the linear combination:

$$z = \theta_0 + \theta_1 x = 4 + 2 \cdot 2 = 4 + 4 = 8$$

Since $z > 0$, the LeakyReLU activation returns z:

$$\hat{y} = \text{LeakyReLU}(z) = z = 8$$

(b): Compute Loss

Using the Squared Error (SE):

$$J(\theta) = (\hat{y} - y)^2 = (8 - 4)^2 = 4^2 = 16$$

(c): Compute Gradients

We apply the chain rule. First,

$$\frac{\partial J}{\partial \hat{y}} = 2(\hat{y} - y) = 2(8 - 4) = 8$$

For $z > 0$, the derivative of LeakyReLU is 1, so

$$\frac{\partial \hat{y}}{\partial z} = 1 \quad \Rightarrow \quad \frac{\partial J}{\partial z} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} = 8 \cdot 1 = 8$$

Now use $z = \theta_0 + \theta_1 x$:

$$\frac{\partial z}{\partial \theta_0} = 1 \quad \Rightarrow \quad \frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial z} \cdot \frac{\partial z}{\partial \theta_0} = 8 \cdot 1 = 8$$

$$\frac{\partial z}{\partial \theta_1} = x = 2 \quad \Rightarrow \quad \frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial z} \cdot \frac{\partial z}{\partial \theta_1} = 8 \cdot 2 = 16$$

(d): Parameter Update (SGD)

With learning rate $\eta = 0.1$:

$$\theta_0 \leftarrow \theta_0 - \eta \cdot \frac{\partial J}{\partial \theta_0} = 4 - 0.1 \cdot 8 = 4 - 0.8 = \boxed{3.2}$$

$$\theta_1 \leftarrow \theta_1 - \eta \cdot \frac{\partial J}{\partial \theta_1} = 2 - 0.1 \cdot 16 = 2 - 1.6 = \boxed{0.4}$$

Final Answer:

$$\theta_0 = 3.2, \quad \theta_1 = 0.4$$

Name: _____

Neptun: _____

Deep Network Development
Midterm Sample
2025 October 21

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

Exam Tasks

1. (10 points) A common anomaly during training is overfitting and underfitting.
   (a) (2 points) How can we identify the two cases?
   (b) (2 points) How do two models (1) overfitted, (2) underfitted perform compared to an ideal model?
   (c) (6 points) List all techniques you learned that can solve this issue.

Solution:

(a): Both are diagnosed by comparing training and validation loss curves.

- **Overfitting:** training loss continues to decrease while validation loss stagnates or increases. The generalization gap (validation minus training) grows over epochs.
- **Underfitting:** both training and validation losses remain high and plateau early. The model cannot even fit the training data, indicating insufficient capacity or optimization.

(b): Relative to an ideal (well-fit) model:

- An **overfitted** model achieves very low training error but high validation/test error — it has memorized training noise and does not generalize.
- An **underfitted** model has high error on both training and validation sets — it is too simple (or undertrained) to capture the underlying pattern.

(c): Techniques that address overfitting/underfitting include:

- **Data augmentation** (flips, crops, noise) — enlarges the effective training set.
- **Regularization**: L1/L2 weight decay, dropout, early stopping.
- **Batch normalization** — stabilizes training and acts as mild regularizer.
- **Model capacity adjustment** — reduce parameters (for overfitting) or increase depth/width (for underfitting).
- **More training data** or **transfer learning / pretraining**.
- **Learning-rate schedule tuning** and longer training for underfitting.
- **Cross-validation** — better model selection across hyperparameters.

2. (10 points) Loss functions are critical in training deep neural networks, guiding the optimization process by quantifying prediction errors.

   (a) (2 points) What is the most important mathematical property a loss function must satisfy in order to be usable in neural network training? Briefly explain why this property is necessary.

   (b) (8 points) Below are several commonly used loss functions along with a list of task categories. Match each loss function to one or more appropriate tasks. Then, justify your choices based on the purpose of the loss.

   Loss Functions:

Loss Name	**Equation**
Mean Squared Error (MSE)	$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Huber Loss	$\mathcal{L}_\delta = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } \lvert y-\hat{y}\rvert \le \delta \\ \delta(\lvert y-\hat{y}\rvert - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$
Cross Entropy (CE)	$\mathcal{L}_{\text{CE}} = -\sum_i y_i \log(\hat{y}_i)$
Binary Cross Entropy (BCE)	$\mathcal{L}_{\text{BCE}} = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$
Dice Loss	$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\sum_i y_i \hat{y}_i}{\sum_i y_i + \sum_i \hat{y}_i}$

   Available Task Categories:

   (A) Binary Classification (B) Multi-class Classification (C) Semantic Segmentation
   (D) Object Detection (E) Regression (F) Neural Machine Translation (G) Sentiment Analysis

   Write your matches clearly, e.g., (1) → E, (3) → B, F, G, etc. Provide 1–2 sentence justifications per match.

Solution:

(a): The loss function must be **differentiable** (at least almost everywhere) with respect to the model parameters. Neural networks are trained by gradient descent on $\nabla_\theta \mathcal{L}$, so the gradient must exist and be well-defined for backpropagation to flow signal from the output to the parameters. A loss with no gradient (e.g., 0/1 classification error) gives no learning signal.

(b): Matches and justifications:

- **MSE** → (E) *Regression*. Penalizes the squared deviation between continuous predictions and continuous ground-truth values; the quadratic shape gives larger errors disproportionately more influence.
- **Huber Loss** → (E) *Regression*, especially with outliers. It is quadratic for small residuals and linear for large residuals, so it retains MSE's smoothness near zero while being robust to outliers (bounded gradient like MAE).
- **Cross Entropy (CE)** → (B) *Multi-class Classification*, (F) *Neural Machine Translation*, (G) *Sentiment Analysis* (when more than two classes). Measures the divergence between the predicted softmax distribution and the one-hot target, producing a well-defined gradient on the logits.
- **Binary Cross Entropy (BCE)** → (A) *Binary Classification*, (G) *Sentiment Analysis* (binary), and pixel-wise (C) *Semantic Segmentation* on two classes. Uses a sigmoid output per element and penalizes each independently.
- **Dice Loss** → (C) *Semantic Segmentation*, (D) *Object Detection* (mask head). Optimizes overlap between predicted and ground-truth masks; it is robust to class imbalance (e.g., small foreground regions) where BCE alone would be dominated by background pixels.

3. (10 points) Pooling operations are essential components in convolutional neural networks, used to reduce spatial dimensions while preserving important features.
   (a) (3 points) What is max pooling and how does it differ from average pooling? Why is it used in convolutional neural networks?
   (b) (3 points) List the key hyperparameters of a pooling layer. Explain how they influence the size and behavior of the output feature map.
   (c) (4 points) Given the input matrix below, perform 2D max pooling with $k = 2\times2$, $s = 2$, and $p = 0$. Show your sliding window positions and intermediate steps clearly.

$$\text{Input: } \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 6 & 1 & 2 \\ 0 & 1 & 3 & 1 \\ 4 & 2 & 0 & 6 \end{bmatrix}$$

Solution:

(a): Max pooling outputs the *maximum* activation in each local window; average pooling outputs the *arithmetic mean*. Max pooling emphasizes the strongest feature response and is more robust to small translations, which is why it is the dominant choice in classification CNNs: it retains the most salient feature while discarding weaker evidence. Average pooling is smoother and used when global context matters (e.g., global average pooling before a classifier head).

(b): The key hyperparameters are:

- **Kernel size** k — the width of the pooling window. Larger k aggregates over more pixels, reducing spatial resolution more aggressively.
- **Stride** s — step between successive windows. Larger s produces a smaller output feature map.
- **Padding** p — extra zeros around the input. It controls whether border pixels are fully covered by the pooling windows.

The output side length is $\lfloor (H + 2p - k)/s \rfloor + 1$, so ($k = s = 2$, $p = 0$) halves each spatial dimension.

(c): With $k = 2 \times 2$, $s = 2$, $p = 0$ on the 4×4 input, there are four 2×2 windows:

$$\begin{bmatrix} 1 & 3 \\ 5 & 6 \end{bmatrix}, \quad \begin{bmatrix} 2 & 4 \\ 1 & 2 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 4 & 2 \end{bmatrix}, \quad \begin{bmatrix} 3 & 1 \\ 0 & 6 \end{bmatrix}.$$

Maxima are $6, 4, 4, 6$ respectively, giving the output

$$\begin{bmatrix} 6 & 4 \\ 4 & 6 \end{bmatrix}.$$

4. (10 points) Dropout and Batch Normalization are common techniques in modern deep learning models.
   (a) (3 points) Why are Dropout and Batch Normalization used in modern deep learning? Briefly explain their respective roles in training.
   (b) (2 points) List the key hyperparameters for both Dropout and Batch Normalization layers. Clearly separate the two.
   (c) (5 points) Describe in a few sentences:
      - What Dropout does during training and inference.
      - What Batch Normalization does and how it affects the forward pass.

Solution:

(a): Both are regularization/stabilization techniques.

- **Dropout** is a regularizer that prevents co-adaptation of neurons by randomly zeroing activations during training. It forces the network to distribute representations and reduces overfitting.
- **Batch Normalization** normalizes each feature across a mini-batch to zero mean and unit variance, then rescales by learnable parameters γ, β. It stabilizes the distribution of activations, enables higher learning rates, and provides a mild regularizing effect.

(b): Key hyperparameters:

- **Dropout**: the drop probability p (typically 0.1–0.5).
- **Batch Normalization**: momentum for running statistics (typically 0.9–0.99), the numerical-stability constant ϵ (e.g. 10^{-5}), and the learnable affine parameters γ, β (on by default).

(c):

- **Dropout — training:** every forward pass, each activation is kept with probability $1 - p$ and set to zero otherwise; kept activations are scaled by $1/(1-p)$ (inverted dropout) so expected magnitude is preserved.
- **Dropout — inference:** dropout is disabled; all units are active and no scaling is applied.
- **Batch Normalization — forward pass:** for a mini-batch, compute per-channel mean μ_B and variance σ_B^2, normalize as $\hat{x} = (x - \mu_B)/\sqrt{\sigma_B^2 + \epsilon}$, then output $y = \gamma\hat{x} + \beta$. During inference the batch statistics are replaced by running estimates accumulated during training.

5. (10 points) Backpropagation is an algorithm used to update the weights of a neural network by computing gradients of the loss function with respect to each parameter.

   Given the following case where the input $x = 2$, the ground truth is $y = 4$, $\theta_1 = 2$ and $\theta_0 = 4$.

   Compute one weight update according to the following steps:

   (a) (1 point) Standardize input values using $\mu = 1$, $\sigma = 1$.

   (b) (2 points) Calculate the model prediction if the model is defined as: $\hat{y} = \text{LeakyReLU}(\theta_0 + \theta_1 x)$, where the negative slope $\beta = 0.5$.

   (c) (3 points) Calculate the loss if the loss function is the **Squared Error (SE)**.

   (d) (4 points) Update the parameters of the model using Gradient Descent with a learning rate of $\eta = 0.1$.

Solution:

(a) Standardize the input.

$$x' = \frac{x - \mu}{\sigma} = \frac{2 - 1}{1} = 1.$$

(b) Forward pass (prediction). The pre-activation is

$$z = \theta_0 + \theta_1 x' = 4 + 2 \cdot 1 = 6.$$

Since $z > 0$, $\text{LeakyReLU}(z) = z$, so

$$\hat{y} = \text{LeakyReLU}(6) = 6.$$

(c) Squared-error loss.

$$\mathcal{L} = (y - \hat{y})^2 = (4 - 6)^2 = 4.$$

(d) Gradients and update. Since $z > 0$, $\frac{\partial \hat{y}}{\partial z} = 1$, and

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -2(y - \hat{y}) = -2(4 - 6) = 4.$$

With $\frac{\partial z}{\partial \theta_0} = 1$ and $\frac{\partial z}{\partial \theta_1} = x' = 1$,

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = 4 \cdot 1 \cdot 1 = 4, \qquad \frac{\partial \mathcal{L}}{\partial \theta_1} = 4 \cdot 1 \cdot 1 = 4.$$

Gradient-descent update with $\eta = 0.1$:

$$\theta_0 \leftarrow 4 - 0.1 \cdot 4 = 3.6, \qquad \theta_1 \leftarrow 2 - 0.1 \cdot 4 = 1.6.$$

Name: _____

Neptun: _____

Deep Network Development
Midterm 2/A
2025 November 25

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

Tasks

1. (10 points) **LSTM**
   (a) (5 points) Explain how LSTMs improve on Vanilla RNNs. Clearly describe the function of each component of the LSTM gates and states. Provide a figure or write the equations.
   (b) (3 points) Describe three ways an RNN can connect its inputs and outputs, based on how many inputs it processes over time and how many outputs it produces. Then give three real-world examples, explaining for each what the input and output are, which of the three arrangements it follows, and why that arrangement fits the task.
   (c) (2 points) Explain how the LSTM handles long-term dependencies. Why is it more effective at this than a Vanilla RNN?

2. (10 points) **Transformers in Computer Vision**
   Transformers were originally developed for sequential NLP tasks, yet they can be adapted for image understanding as well.
   (a) (4 points) Explain how the Transformer architecture can be applied to images. Your explanation should follow the Vision Transformer (ViT) pipeline.
   (b) (3 points) Compared to CNNs, what inductive bias is missing in the vanilla ViT architecture? Explain why this makes ViT only perform well when trained on very large datasets.
   (c) (3 points) Describe at least two architectural techniques that counteract this missing inductive bias (e.g. Token-to-Token ViT, Swin Transformer). Explain how each method restores functionality that CNNs naturally provide.

3. (10 points) **NLP Tasks and Transformer Models**
   (a) (3 points) List at least two NLP tasks that were introduced in the lecture. For each task, describe the dataset inputs and outputs.
   (b) (3 points) Explain how GPT or BERT can be used to solve the tasks you listed.
   (c) (3 points) Describe the main differences between GPT and BERT.
   (d) (1 point) Explain how GPT generates text.

4. (10 points) **Generative Modelling and Variational Autoencoders**
   (a) (3 points) What is the goal of Generative Modelling according to the lecture? Explain why learning an unknown probability distribution is difficult.
   (b) (3 points) How can we use generative modelling for image generation? **HINT**: Use the autoencoder examples shown in the lecture to describe the idea.
   (c) (4 points) How does a Variational Autoencoder (VAE) address the problems of a standard autoencoder in generative modelling? Explain the role of the latent variable distribution.

5. (10 points) **BLEU score calculation**

Consider the following candidate translation and reference translations. Compare the words exactly as written (no lemmatization or tokenization other than splitting on whitespace).

- **Candidate:** *She enjoys playing tennis with her friends*
- **Reference 1:** *She loves to play tennis with her friends*
- **Reference 2:** *She enjoys to play tennis with her friends*

(a) (4 points) List the 1-, 2-, 3-, and 4-grams of the candidate sentence and compute the **clipped precision** p_1, p_2, p_3, p_4 with respect to the two references.

(b) (3 points) Using your clipped precisions, compute the **Geometric Average Precision (GAP(N))** score and compute the **Brevity Penalty** as defined in the lecture.

(c) (3 points) Compute the final **BLEU-1, BLEU-2, BLEU-3, BLEU-4 score**: **NOTE:** During the calculation, use the values rounded to three decimal places.

Name: _____

Neptun: _____

Deep Network Development
Midterm 2/B
2025 November 25

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

Tasks

1. (10 points) **Transformer Encoder and Attention Mechanisms**
   (a) (4 points) Draw the Transformer encoder block and explain the function of each of its components, including the embedding process.
   (b) (3 points) Explain the scaled dot-product attention mechanism. Describe the intuition behind attention, the roles of queries, keys, and values, and write the corresponding formula.
   (c) (3 points) Explain self-attention and multi-head attention. Describe the difference between them and explain why the Transformer uses multiple attention heads.

2. (10 points) **Encoder–Decoder Context and Attention**
   (a) (3 points) In an RNN-based encoder–decoder model for machine translation, what is the context passed from the encoder to the decoder?
   (b) (3 points) What limitation or problem arises from using this original context representation?
   (c) (4 points) How can attention improve this architecture? Explain the additive attention mechanism using either the relevant equations or a figure.

3. (10 points) **Structure from Motion**
   (a) (3 points) What is the purpose of Structure from Motion (SfM) in computer vision?
   (b) (4 points) Explain how two images can be stitched or aligned using the steps of Structure from Motion. Describe the main steps involved in extracting and matching features, and how these allow the relative transformation between the images to be estimated.
   (c) (3 points) Why is it useful to know the camera position for each image? Give an example related to neural rendering or 3D Gaussian splatting generative methods.

4. (10 points) **BLEU score calculation**

   Calculate the BLEU-4 score for the following candidate sentence using the reference translations. Compare words exactly as written (no lemmatization or extra tokenization, just split on whitespace).

   - **Candidate:** *I need to finish my homework*
   - **Reference 1:** *I really need to finish my homework today*
   - **Reference 2:** *I still need to finish my homework tonight*

   (a) (4 points) List the 1-, 2-, 3-, and 4-grams of the candidate sentence and compute the **clipped precision** p_1, p_2, p_3, p_4 with respect to the two references.
   (b) (3 points) Using your clipped precisions, compute the **Geometric Average Precision (GAP(N))** score and the **Brevity Penalty** as defined in the lecture.
   (c) (3 points) Compute the final **BLEU-1, BLEU-2, BLEU-3, BLEU-4 scores**. **NOTE:** During the calculation, use the values rounded to three decimal places.

5. (10 points) **YOLO and DETR**

YOLO and DETR represent two fundamentally different approaches to object detection. While both are state-of-the-art detectors, they differ significantly in design and philosophy.

(a) (3 points) Is YOLO considered a single-stage or two-stage detector? Justify your answer based on its architectural design and inference behavior.

(b) (3 points) Compare the architectural pipeline of YOLO with that of DETR. Focus your answer on how predictions are generated.

(c) (4 points) DETR introduces the concept of ”set prediction” using a transformer and bipartite matching.

- Explain what ”set prediction” means in this context.
- Why does this make traditional components like Non-Maximum Suppression (NMS) unnecessary?
- What are the trade-offs of this approach compared to grid-based detectors like YOLO?

Name: _____

Neptun: _____

Deep Network Development
Midterm 2 Retake
2025 November 25

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

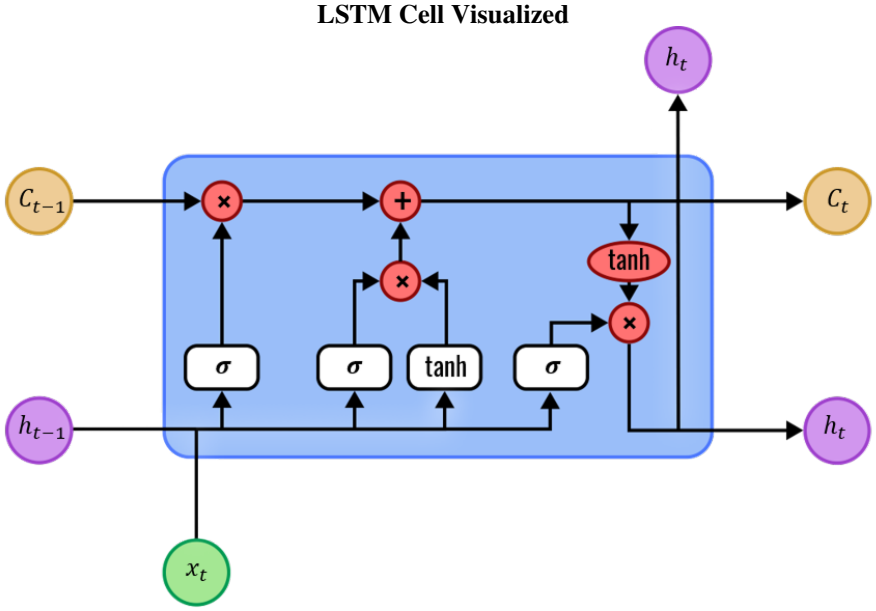
Tasks

1. (10 points) **LSTM**
   (a) (5 points) Explain how LSTMs improve on Vanilla RNNs. Clearly describe the function of each component of the LSTM gates and states. Provide a figure or write the equations.
   (b) (3 points) Describe three ways an RNN can connect its inputs and outputs, based on how many inputs it processes over time and how many outputs it produces. Then give three real-world examples, explaining for each what the input and output are, which of the three arrangements it follows, and why that arrangement fits the task.
   (c) (2 points) Explain how the LSTM handles long-term dependencies. Why is it more effective at this than a Vanilla RNN?

Solution:

(a): LSTM vs. Vanilla RNN, structure, and gates

LSTM Cell Visualized



Vanilla RNNs suffer from vanishing gradients and cannot retain information over long sequences. The LSTM solves this problem using gates and a memory cell to regulate information flow. Below are the equations shown in the lecture, each named and explained.

1. Forget Gate Equation

$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t + b_f)$$

Purpose: The forget gate decides how much of the previous cell state C_{t-1} should be kept. Values close to 1 mean "keep this information," while values close to 0 mean "forget it." This allows the LSTM to drop irrelevant older memory.

2. Input (Ignore) Gate Equation

$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t + b_i)$$

Purpose: Controls which parts of the new candidate memory should be added to the cell state. It regulates how much new information enters the long-term memory.

3. Candidate Cell State Equation

$$\tilde{C}_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$

Purpose: Creates a vector of new candidate values that could be written into the memory. This is the "proposed update" to the cell state, shaped by a tanh nonlinearity.

4. Cell State Update Equation

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

Purpose: Combines old memory (scaled by the forget gate) and new information (scaled by the input gate). The cell state uses elementwise addition, allowing gradients to flow easily—this is the key reason LSTMs avoid vanishing gradients.

5. Output Gate Equation

$$o_t = \sigma(W_{ho} h_{t-1} + W_{xo} x_t + b_o)$$

Purpose: Determines which parts of the updated cell state will influence the hidden state (the LSTM's output). Acts as a "filter" between memory and output.

6. Hidden State Equation

$$h_t = o_t \cdot \tanh(C_t)$$

Purpose: Produces the hidden output of the LSTM at time t. The output gate controls which parts of the memory become visible to the next layer or next time step.

(b) Examples and configurations

- **Sentiment Analysis (Many-to-One):** Input is a sequence of words, output is a single sentiment label. A many-to-one LSTM processes the entire sequence and outputs one classification.
- **Machine Translation (Many-to-Many):** Encoder LSTM reads the full input sentence, decoder LSTM produces the target-language sentence. This corresponds to a many-to-many seq2seq structure.
- **Image Captioning (One-to-Many):** An image embedding is input to the LSTM, which then generates a sequence of words. This is a one-to-many configuration.

(c) Long-term dependencies

LSTMs handle long-term dependencies by using a cell state that is updated using elementwise addition, allowing gradients to flow without vanishing. The forget, input, and output gates determine which information is stored, discarded, or exposed. As a result, LSTMs avoid the vanishing gradient problem typical of Vanilla RNNs and can preserve information across many time steps.

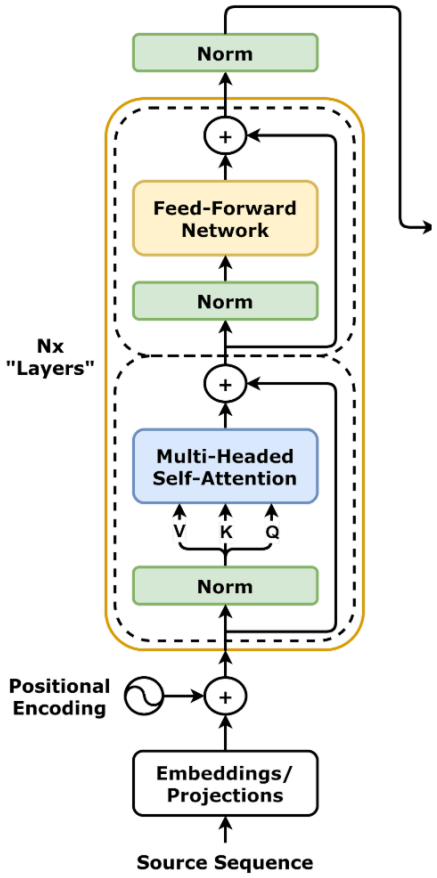
2. (10 points) **Transformer Encoder and Attention Mechanisms**

   (a) (4 points) Draw the Transformer encoder block and explain the function of each of its components, including the embedding process.

(b) (3 points) Explain the scaled dot-product attention mechanism. Describe the intuition behind attention, the roles of queries, keys, and values, and write the corresponding formula.

(c) (3 points) Explain self-attention and multi-head attention. Describe the difference between them and explain why the Transformer uses multiple attention heads.

Solution:



Transformer, full architecture.png
https://commons.wikimedia.org/wiki/File:Transformer,_full_architecture.png
dvgodoy, CC BY 4.0
<https://creativecommons.org/licenses/by/4.0>, via
Wikimedia Commons

Transformer Encoder visualized

A Transformer encoder block (as shown in the lecture) consists of:

1. **Word Embeddings:** Each input token is mapped to a continuous vector representation.
2. **Positional Encoding:** Since the Transformer processes all tokens in parallel, it has no inherent notion of order. Sinusoidal positional encodings are added to embeddings so that the model knows the position of each token.
3. **Multi-Head Self-Attention:** Computes relationships between every pair of input tokens. Each token attends to all others in the sequence to capture contextual dependencies.
4. **Add & Norm:** A residual connection adds the original input of the layer to the attention output, followed by Layer Normalization.

5. **Feed-Forward Network (FFN):** A fully connected network applied independently to each token. It increases model capacity by transforming each token representation in a nonlinear way.
6. **Add & Norm:** Another residual connection and normalization applied after the FFN.

The encoder is built by stacking several identical blocks (6 in the original Transformer).

(b) Scaled Dot-Product Attention

Intuition:

- The **Query** represents what information a token is searching for.
- The **Key** represents what information each token contains.
- The **Value** is the actual content that will be aggregated based on attention weights.

The similarity between a Query and each Key determines how much attention one token should pay to another.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V.$$

- The dot product $QK^\top$ measures similarity between tokens.
- Dividing by $\sqrt{d_k}$ stabilizes training by preventing large values.
- Softmax converts similarity scores into a probability distribution (attention weights).
- Weighted sum with V produces the final context-aware representation.

(c) Self-Attention vs. Multi-Head Attention

Self-Attention: Each token in the sequence attends to all other tokens *within the same sequence*. It creates context-aware token representations by computing attention scores between every pair of input tokens.

$$\text{SelfAttention}(X) = \text{Attention}(XW^Q, XW^K, XW^V).$$

Multi-Head Attention: Instead of computing attention once, the Transformer computes it multiple times in parallel using different learned projections:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

The outputs are concatenated and linearly projected:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W^O.$$

Reason (as explained in the lecture):

- Each head learns to focus on different types of relationships.
- The model can look at the sequence from multiple perspectives.
- It improves the ability to capture complex dependencies.

Thus, multi-head attention is a richer and more expressive version of self-attention.

3. (10 points) **Transformers in Computer Vision**

Transformers were originally developed for sequential NLP tasks, yet they can be adapted for image under-

standing as well.

(a) (4 points) Explain how the Transformer architecture can be applied to images. Your explanation should follow the Vision Transformer (ViT) pipeline.

(b) (3 points) Compared to CNNs, what inductive bias is missing in the vanilla ViT architecture? Explain why this makes ViT only perform well when trained on very large datasets.

(c) (3 points) Describe at least two architectural techniques that counteract this missing inductive bias (e.g. Token-to-Token ViT, Swin Transformer). Explain how each method restores functionality that CNNs naturally provide.

Solution:

(a) Applying Transformers to Images

In the Vision Transformer (ViT), an image is processed using the following steps:

- **Patch Extraction:** The input image is divided into fixed-size patches (e.g. 16×16 pixels). Each patch is flattened and projected linearly into a vector, forming a *patch token*. This converts the 2D image into a 1D sequence of tokens similar to word embeddings.
- **Class Token:** A special learnable "class token" is prepended to the token sequence. Its output after the transformer encoders will represent the entire image for classification.
- **Positional Embeddings:** Since transformers do not have a built-in notion of order, positional embeddings are added to each patch token so that the model knows the spatial location of each patch.
- **Transformer Encoder Blocks:** The sequence is processed by stacked encoder layers consisting of: multi-head self-attention, residual connections, layer normalization, and feed-forward layers.
- **Classification Head:** The final vector corresponding to the class token is passed to a linear classifier.

This allows the transformer to operate on image data similarly to sequences in NLP, but without using convolutions.

(b) Missing inductive bias in Vanilla ViT

Compared to CNNs, the vanilla ViT lacks the following image-specific inductive biases:

- **Locality:** CNNs assume nearby pixels are related, allowing them to detect edges and textures. ViT treats all tokens equally and does not automatically focus on local neighborhoods.
- **Translation Equivariance / Weight Sharing:** CNN filters are reused across the entire image. ViT uses fully global attention without local weight sharing.
- **2D Spatial Structure and Hierarchy:** CNNs maintain the 2D grid and learn hierarchical features (edges $\rightarrow$ textures $\rightarrow$ objects). ViT flattens patches into a sequence, losing this structure.

Because these biases are not built into ViT, it must *learn* them from data. This requires extremely large datasets; on medium-sized datasets (e.g. ImageNet-1k), vanilla ViT performs worse than CNNs.

(c) Counteracting the Missing Inductive Bias

Two techniques discussed in the lecture restore locality and hierarchy:

- **Token-to-Token ViT (T2T-ViT):** Introduces a token-to-token transformation that progressively aggregates neighboring patches. This recreates local relationships and restores part of the hierarchical structure lost in vanilla ViT. It captures finer details in early layers and improves performance on medium-sized datasets.
- **Swin Transformer:** Uses local *window-based* self-attention instead of full global attention. "Shifted windows" allow information exchange across windows. Swin also uses patch merging to build a hierarchical feature pyramid similar to CNNs. This reintroduces locality, scalability, and multi-resolution features.

Both methods compensate for ViT's lack of inductive biases by adding locality or hierarchy back into the architecture, improving performance and data efficiency.

4. (10 points) **Structure from Motion**
   (a) (3 points) What is the purpose of Structure from Motion (SfM) in computer vision?
   (b) (4 points) Explain how two images can be stitched or aligned using the steps of Structure from Motion. Describe the main steps involved in extracting and matching features, and how these allow the relative transformation between the images to be estimated.
   (c) (3 points) Why is it useful to know the camera position for each image? Give an example related to neural rendering or 3D Gaussian splatting generative methods.

Solution:

(a) Purpose of Structure from Motion

Structure from Motion (SfM) aims to recover the 3D structure of a scene from multiple 2D images. It jointly estimates:

- the 3D geometry of the scene, and
- the relative camera poses (positions and orientations) for each input image.

(b) Stitching or aligning two images using SfM

As described in the lecture, SfM aligns two images using:

1. **Feature extraction:** Detect corner-like features in both images and compute descriptors for patches around them (e.g., $40 \times 40 \rightarrow$ blurred $\rightarrow 8 \times 8 \rightarrow$ flattened).
2. **Feature matching:** For each feature in image 1, compare its descriptor to all descriptors in image 2. Keep only confident matches using the ratio between the best and second-best match distances.
3. **Camera transform estimation:** Using the matched feature correspondences, compute the relative rotation and translation between the two camera views. This allows the two images to be placed in a common coordinate system.

Once the relative pose is known, the images can be aligned, stitched, or used for further 3D reconstruction through triangulation.

(c) Why camera positions are useful (NeRF / 3DGS example)

Knowing the camera pose for each image is essential because it tells the model *from where* each photo was taken. This is required for reconstructing a consistent 3D scene.

In neural rendering methods such as:

- **NeRF (Neural Radiance Fields)**, or
- **3D Gaussian Splatting (3DGS)**,

the model uses camera positions to project 3D points into each image and compare the rendered view with the actual photograph.

Example: In NeRF, each training step renders a ray from the camera through the scene. Without knowing the exact camera pose for each training image, the model cannot correctly align rays, and the resulting 3D reconstruction would collapse or blur. Thus, SfM-derived camera poses are often used as input for NeRF or 3DGS pipelines.

5. (10 points) **BLEU score calculation**

Calculate the BLEU-4 score for the following candidate sentence using the reference translations. Compare words exactly as written (no lemmatization or extra tokenization, just split on whitespace).

- **Candidate:** *The project was completed successfully last week*
- **Reference 1:** *The entire project was successfully completed last week after months of hard work*
- **Reference 2:** *The project was finally completed last week and the team celebrated the achievement*

(a) (4 points) Compute the **clipped precision** p_1, p_2, p_3, p_4 with respect to the two references.

(b) (3 points) Using your clipped precisions, compute the **Geometric Average Precision (GAP(N))** score and the **Brevity Penalty** as defined in the lecture.

(c) (3 points) Compute the final **BLEU-1, BLEU-2, BLEU-3, BLEU-4 scores**. **NOTE:** During the calculation, use the values rounded to three decimal places.

Solution:

(a) Clipped precisions

Tokenization (split on whitespace):

Candidate (length $c = 7$):

The, project, was, completed, successfully, last, week

Reference 1 (length 13):

The, entire, project, was, successfully, completed, last, week, after, months, of, hard, work

Reference 2 (length 13):

The, project, was, finally, completed, last, week, and, the, team, celebrated, the, achievement

Both references have length 13, so the effective reference length is:

$$r = 13.$$

Unigrams (1-grams) of the candidate

{The, project, was, completed, successfully, last, week}

All of these occur in at least one of the references, so clipped count = 7, total candidate unigrams = 7:

$$p_1 = \frac{7}{7} = 1.000.$$

Bigrams (2-grams) of the candidate

Overlapping bigrams (candidate vs any reference):

{The project, project was, last week}

So clipped count = 3, total candidate bigrams = 6:

$$p_2 = \frac{3}{6} = 0.500.$$

Trigrams (3-grams) of the candidate

Overlapping trigrams:

{The project was}

So clipped count = 1, total candidate trigrams = 5:

$$p_3 = \frac{1}{5} = 0.200.$$

4-grams of the candidate

There are no overlapping 4-grams between the candidate and the references, so clipped count = 0, total candidate 4-grams = 4:

$$p_4 = \frac{0}{4} = 0.000.$$

Summary:

$$p_1 = 1.000, \quad p_2 = 0.500, \quad p_3 = 0.200, \quad p_4 = 0.000.$$

(b) GAP(N) and brevity penalty

Since $c = 7$ and $r = 13$, we have $c < r$, so:

$$BP = e^{1-r/c} = e^{1-13/7} = e^{-6/7} \approx 0.424.$$

We use the lecture definition:

$$\mathrm{GAP}(N) = \prod_{n=1}^{N} p_n^{1/N}.$$

BLEU-1

$$\mathrm{GAP}(1) = p_1 = 1.000.$$

BLEU-2

$$p_1^{1/2} = 1.000^{1/2} = 1.000, \quad p_2^{1/2} = 0.500^{1/2} \approx 0.707,$$
$$\mathrm{GAP}(2) \approx 1.000 \cdot 0.707 = 0.707.$$

BLEU-3

$$p_1^{1/3} = 1.000^{1/3} = 1.000, \quad p_2^{1/3} = 0.500^{1/3} \approx 0.794, \quad p_3^{1/3} = 0.200^{1/3} \approx 0.585,$$
$$\mathrm{GAP}(3) \approx 1.000 \cdot 0.794 \cdot 0.585 \approx 0.464.$$

BLEU-4

$$p_1^{1/4} = 1.000^{1/4} = 1.000, \quad p_2^{1/4} = 0.500^{1/4} \approx 0.841, \quad p_3^{1/4} = 0.200^{1/4} \approx 0.669, \quad p_4^{1/4} = 0.000^{1/4} = 0.000,$$
$$\mathrm{GAP}(4) \approx 1.000 \cdot 0.841 \cdot 0.669 \cdot 0.000 \approx 0.000.$$

(c) BLEU scores

$$\text{BLEU-1} = BP \cdot \mathrm{GAP}(1) \approx 0.424 \cdot 1.000 \approx 0.424,$$
$$\text{BLEU-2} \approx 0.424 \cdot 0.707 \approx 0.300,$$
$$\text{BLEU-3} \approx 0.424 \cdot 0.464 \approx 0.197,$$
$$\text{BLEU-4} \approx 0.424 \cdot 0.000 \approx 0.000.$$

So, approximately:

$$\boxed{\text{BLEU-1} \approx 0.424, \quad \text{BLEU-2} \approx 0.300, \quad \text{BLEU-3} \approx 0.197, \quad \text{BLEU-4} \approx 0.000.}$$

Name: _____

Neptun: _____

Deep Network Development
Midterm 2/Sample
2025 November 25

You have exactly **90 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **20 points** is required to achieve a passing grade. The grades are calculated as follows.

Grade	Final Score Range
5	> 42
4	$38 - 42$
3	$33 - 37$
2	$20 - 32$
Fail (0)	< 20

Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

Tasks

1. (10 points) **Recurrent Neural Networks**
   (a) (3 points) Explain how a Recurrent Neural Network (RNN) processes a sequence. Describe how information flows through time in an RNN.
   (b) (2 points) How does using an RNN differ from applying a standard multilayer perceptron (MLP) to NLP tasks? What advantages do RNNs provide over MLPs when dealing with sequential data?
   (c) (3 points) Describe the components of a vanilla RNN. Explain how the hidden state is updated at each timestep, using either a diagram or the relevant equations.
   (d) (2 points) What major issue or limitation is associated with vanilla RNNs? Explain why this issue arises.

Solution:

(a): An RNN processes a sequence one token at a time, maintaining a hidden state h_t that summarizes everything seen so far. At each timestep t, the cell receives the current input x_t and the previous hidden state h_{t-1}, and produces a new hidden state h_t (and optionally an output y_t). Information flows through time via this hidden state, which is updated recurrently — the same weights are reused at every step.

(b): An MLP takes a fixed-size input and has no concept of order: it treats words independently, losing sequential context. An RNN (i) handles *variable-length* sequences, (ii) *shares parameters* across timesteps (far fewer parameters than an MLP over a long context), and (iii) carries *temporal context* through the hidden state. This makes RNNs natural for NLP tasks where meaning depends on word order.

(c): A vanilla RNN cell has three learnable weight matrices (W_{xh}, W_{hh}, W_{hy}) and two biases (b_h, b_y). The hidden-state update and output are

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \qquad y_t = W_{hy}h_t + b_y.$$

The same W_{xh}, W_{hh}, W_{hy} are applied at every timestep.

(d): The main limitation is the **vanishing (and exploding) gradient** problem during backpropagation through time. Because the same weight matrix W_{hh} is multiplied repeatedly across timesteps, gradients either shrink exponentially (if its spectral radius < 1) or blow up (if > 1). This makes vanilla RNNs unable to learn long-range dependencies, motivating the introduction of gated architectures like LSTM and GRU.

2. (10 points) **Image Inpainting and Denoising**
   (a) (3 points) Describe the main categories of traditional methods used for image inpainting and image denoising. What limitations do these approaches have?
   (b) (4 points) Explain how deep-learning-based methods can be used to fill large missing or occluded regions in an image. Describe the general idea behind these approaches.

(c) (3 points) Several deep-learning-based inpainting models operate in multiple stages. What multi-stage strategies were discussed in the lecture, and how do they conceptually differ?

Solution:

(a): Traditional approaches fall into two broad families:

- **Diffusion / PDE-based methods** (e.g. Navier–Stokes-inspired inpainting, total-variation denoising) — propagate pixel information from the surrounding region into the hole by solving a partial differential equation that enforces smoothness.
- **Patch / exemplar-based methods** (e.g. PatchMatch, non-local means) — copy and blend patches from other parts of the same image based on similarity.

Both work well for small or thin holes with repetitive texture, but they fail for large missing regions or when the missing content is semantic (e.g., reconstructing a face, restoring an occluded object). They have no understanding of object structure and cannot invent plausible content that is not already present elsewhere in the image.

(b): Deep learning approaches are conditional generative models: a network takes the masked image plus the mask as input and outputs a completed image. Early variants use an encoder–decoder (context encoder) with L1 or perceptual losses; modern versions use GANs or diffusion models, trained with masks randomly generated during training so the network learns strong priors over natural image statistics. Because these models learn a data-driven prior, they can hallucinate *new* content consistent with the surroundings (faces, textures, objects) rather than merely extrapolating nearby pixels. Skip connections, partial/gated convolutions, and attention help condition the output on the known region while preserving global coherence.

(c): Multi-stage strategies split the problem into easier sub-problems, completed in sequence:

- **Coarse-to-fine** — a first network produces a low-resolution or blurry completion, and a second refines it, often with contextual attention that copies coherent features from the known region.
- **Structure-first then texture** — stage one predicts edges/semantic layout inside the hole; stage two fills in textures conditioned on that structure.
- **Latent-diffusion-based** — encode to a latent space, denoise only inside the masked latent region, then decode. Iterative denoising itself can be viewed as many-stage refinement.

Conceptually the stages differ in *what* they predict first: spatial structure, low-frequency content, or semantic layout — each making the subsequent stage's job easier.

3. (10 points) **Attention Mechanisms**

(a) (3 points) What is the main purpose of the attention mechanism in neural network models? Explain why attention is useful when working with sequences.

(b) (3 points) What is the key difference between the attention used in RNN-based seq2seq architectures and the attention used in Transformers? Describe how these two settings apply attention differently.

(c) (2 points) Why do Transformers not require recurrence or convolution when processing sequences? Explain how attention enables this.

(d) (2 points) In the context of multi-head attention, why is it beneficial to use multiple attention heads instead of a single one? Describe the motivation at a high level.

Solution:

(a): Attention lets each element of a sequence selectively aggregate information from other elements that are relevant to it. Instead of compressing the whole input into a single fixed-size vector (as a plain encoder

RNN does), attention computes a weighted sum of value vectors where the weights come from learned query/key compatibilities. This preserves long-range context and avoids the information bottleneck of sequential hidden states.

(b): In RNN-based seq2seq, attention is used only between the decoder and encoder: at each decoder step, the decoder hidden state acts as a query over the encoder's per-step outputs (keys/values) and pulls relevant source information. In Transformers attention is ubiquitous and parallel: *self-attention* is applied within the encoder (tokens attend to all other source tokens), within the decoder (causal self-attention over target tokens), and between decoder and encoder (cross-attention). Transformer attention is multi-headed and scaled-dot-product; RNN attention is typically single-head additive (Bahdanau) or dot-product (Luong) on a per-decoder-step basis.

(c): Self-attention directly connects every pair of tokens in one step, so it captures arbitrary long-range dependencies without any recurrence. Together with positional encodings (to restore order information) and stacked feed-forward layers, attention alone already mixes information across the full sequence. Parallelism over sequence positions is the other major benefit — no sequential dependence makes training dramatically faster than RNNs.

(d): Multiple heads let the model attend to different types of relationships *simultaneously* (e.g. one head tracks syntactic subject–verb agreement, another tracks coreference). Each head operates in its own projected subspace, then outputs are concatenated and linearly projected. This yields a richer, more expressive representation than a single attention function over a single subspace.

4. (10 points) **Generative Models: GANs and Diffusion**
   (a) (2 points) What are the main components of a Generative Adversarial Network (GAN) architecture?
   (b) (3 points) What makes GANs challenging to train? Describe the main difficulties that arise during GAN optimization.
   (c) (3 points) Compared to GANs, how is a diffusion-based model such as Stable Diffusion trained? Explain the key conceptual differences between the two types of generative models.
   (d) (2 points) How can a Stable Diffusion model be used to generate new images? Describe the process at a high level.

Solution:

(a): A GAN consists of two networks trained jointly:

- a **generator** $G(z)$ that maps a latent noise vector z to a synthetic sample, and
- a **discriminator** $D(x)$ that classifies samples as real (from the training distribution) or fake (produced by G).

The two are trained adversarially: D learns to distinguish real from fake, and G learns to fool D.

(b): GAN training is notoriously unstable because:

- the min–max objective has no guaranteed convergence and oscillates;
- if D overpowers G, the gradient signal through D vanishes and G stops improving;
- **mode collapse** — G learns to produce only a few high-confidence outputs rather than covering the full data distribution;
- the loss value does not correlate with sample quality, so there is no easy stopping criterion.

(c): Diffusion models like Stable Diffusion are trained with a simple regression objective: given a clean image x_0, add scheduled Gaussian noise to produce x_t at timestep t, and train a denoising network to predict

the noise (or equivalently x_0). The training loss is a per-timestep MSE. The key conceptual differences to GANs are: (i) a single network rather than an adversarial pair, so training is stable and does not suffer from mode collapse; (ii) the generative process is iterative — many small denoising steps rather than one forward pass; (iii) the loss is a well-behaved likelihood-related quantity instead of a min–max game.

(d): To sample, start from pure Gaussian noise x_T and iteratively apply the trained denoiser to predict a slightly less-noisy x_{t-1}, repeating for T steps (or fewer with a fast sampler like DDIM) until x_0 emerges. For *text-to-image*, a text encoder (e.g. CLIP) conditions the denoiser via cross-attention so the final x_0 matches the prompt. Classifier-free guidance mixes conditional and unconditional predictions to strengthen prompt adherence.

5. (10 points) **BLEU Score Calculation**

Calculate the BLEU-4 score for the following candidate sentence using the reference translations. Use clipped precision and the brevity penalty. Compare the words exactly as written (no lemmatization, stemming, or additional tokenization).

- **Candidate:** *The dog barks loudly when it sees strangers*
- **Reference 1:** *The dog barks when it sees people*
- **Reference 2:** *The dog growl loudly when it sees strangers*

(a) (4 points) List the 1-, 2-, 3-, and 4-grams of the candidate sentence and compute the **clipped precision** p_1, p_2, p_3, p_4 with respect to the two references.

(b) (3 points) Using your clipped precisions, compute the **Geometric Average Precision (GAP(N))** score and the **Brevity Penalty** as defined in the lecture.

(c) (3 points) Compute the final **BLEU-1, BLEU-2, BLEU-3, BLEU-4 scores**. **NOTE:** During the calculation, use the values rounded to three decimal places.

Solution:

Tokenization (whitespace split).

- **Candidate** ($c = 8$): The, dog, barks, loudly, when, it, sees, strangers
- **Reference 1** (length 7): The, dog, barks, when, it, sees, people
- **Reference 2** (length 8): The, dog, growl, loudly, when, it, sees, strangers

The effective reference length is the closest-length reference to the candidate, so $r = 8$.

(a) Clipped precisions.

Unigrams (8 total in candidate). Every candidate unigram occurs in at least one reference, so clipped count = 8:

$$p_1 = \frac{8}{8} = 1.000.$$

Bigrams (7 total). Matches in references: (The, dog), (dog, barks), (loudly, when), (when, it), (it, sees), (sees, strangers) = 6. Miss: (barks, loudly).

$$p_2 = \frac{6}{7} \approx 0.857.$$

Trigrams (6 total). Matches: (The, dog, barks), (loudly, when, it), (when, it, sees), (it, sees, strangers) = 4. Misses: (dog, barks, loudly), (barks, loudly, when).

$$p_3 = \frac{4}{6} \approx 0.667.$$

4-grams (5 total). Matches: (loudly, when, it, sees), (when, it, sees, strangers) = 2.

$$p_4 = \frac{2}{5} = 0.400.$$

(b) GAP(N) and brevity penalty. Since $c = r = 8$, $c \geq r$, so

$$BP = 1.000.$$

Geometric average of precisions:

$$\text{GAP}(N) = \left(\prod_{n=1}^{N} p_n \right)^{1/N}.$$

(c) BLEU scores (with rounding to three decimals at each step):

$$\text{BLEU-1} = BP \cdot \text{GAP}(1) = 1.000 \cdot 1.000 = 1.000,$$
$$\text{BLEU-2} = 1.000 \cdot (1.000 \cdot 0.857)^{1/2} = (0.857)^{1/2} \approx 0.926,$$
$$\text{BLEU-3} = 1.000 \cdot (1.000 \cdot 0.857 \cdot 0.667)^{1/3} = (0.572)^{1/3} \approx 0.830,$$
$$\text{BLEU-4} = 1.000 \cdot (1.000 \cdot 0.857 \cdot 0.667 \cdot 0.400)^{1/4} = (0.229)^{1/4} \approx 0.692.$$

Summary:

BLEU-1 ≈ 1.000,	BLEU-2 ≈ 0.926,	BLEU-3 ≈ 0.830,	BLEU-4 ≈ 0.692

Part IV.
Final Exams

Name: _____

Neptun: _____

Deep Network Development
Exam 1
May 23, 2025

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

We wish you the best of luck.

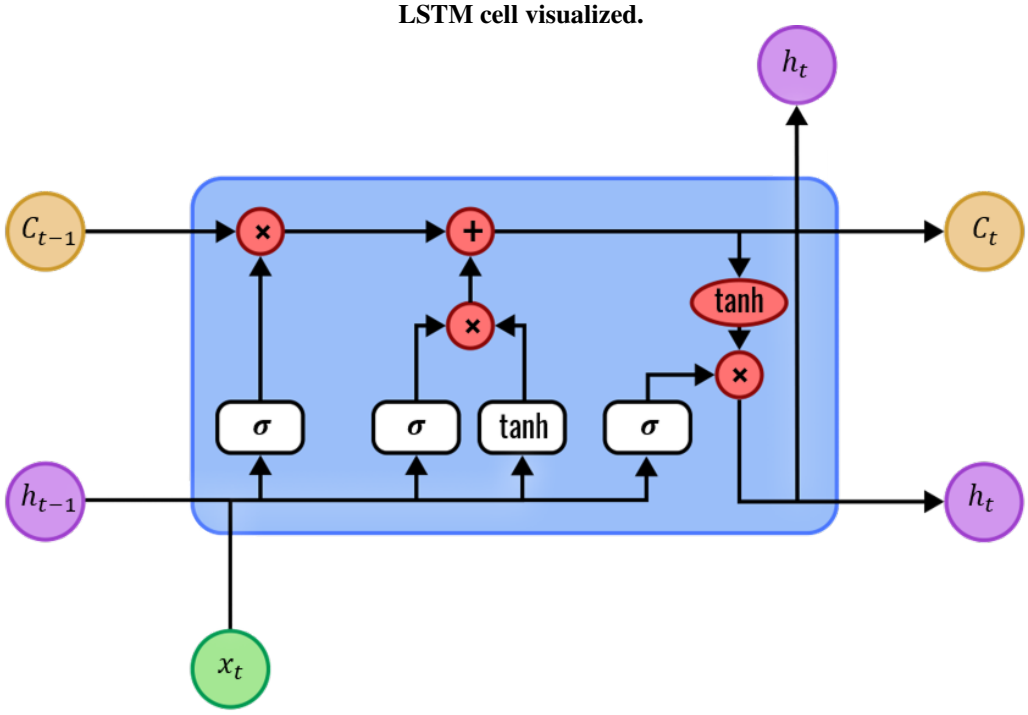
Exam Tasks

1. (10 points) LSTMs are a type of recurrent neural network designed to effectively capture long-term dependencies in sequential data:
   (a) (3 points) Illustrate the architecture of a basic LSTM cell (simple figure).
   (b) (4 points) Label and briefly describe the function of each core component.
   (c) (3 points) Explain how the LSTM architecture mitigates the vanishing gradient problem and why it is considered an improvement over vanilla RNNs.

Solution:

(a):

LSTM cell visualized.



(b): A basic LSTM cell includes the following components:

- **Cell state** (C_t): A highway for information flow across time steps, enabling long-term memory retention.
- **Forget gate** (f_t): Determines which parts of the previous cell state should be discarded.

- **Input gate** (i_t): Controls what new information will be added to the cell state.
- **Output gate** (o_t): Regulates which part of the cell state is exposed as output (h_t).

(c): In vanilla RNNs, the hidden state updates involve repeated multiplication, causing gradients to vanish or explode during backpropagation. LSTMs alleviate this by updating the cell state additively.

2. (10 points) Convolution is a mathematical operation that slides a filter over input data to extract local features.
   (a) (3 points) Explain the convolution operation and how it is implemented within convolutional layers in neural networks.
   (b) (3 points) Identify and describe the key components and hyperparameters of a convolutional layer.
   (c) (4 points) Given the input matrix and convolutional kernel below, compute the resulting feature map using a 2D convolution with $s = 2$ and $p = 0$. Show all steps clearly.

$$\text{Input: } \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \\ 2 & 3 & 0 & 1 \\ -2 & 0 & 1 & 2 \\ -1 & 3 & 0 & 1 \end{bmatrix} \qquad \text{Kernel: } \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ -1 & 0 \end{bmatrix}$$

Solution:

(a): Convolution is a mathematical operation that combines two functions to produce a third function expressing how one shape is modified by the other. In image processing, a small matrix called a kernel or filter slides over the input image to compute a weighted sum, producing a feature map.

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) \cdot K(m, n)$$

where I is the input image, K is the kernel, and S is the resulting feature map.

In Convolutional Neural Networks (CNNs), convolutional layers apply multiple learnable filters to the input. Each filter scans across the spatial dimensions, computing dot products with patches of the input and producing a separate feature map. These feature maps are stacked to form the output.

(b): Key components and hyperparameters:

- **Kernel size**: Dimensions of the filter (e.g., 3×3, 5×5).
- **Stride**: The number of pixels the filter moves across (e.g., stride 1).
- **Padding**: Zeros added around the input to control output size (e.g., 'same', 'valid').
- **Number of filters**: Determines the number of output channels.

(c): Resulting feature map with a 3×2 kernel applied to a 6×4 input (stride 2, no padding):

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \\ 2 & 3 & 0 & 1 \\ -2 & 0 & 1 & 2 \\ -1 & 3 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ -1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 9 & 20 \\ 25 & 8 \end{bmatrix}$$

3. (10 points) Cross-Entropy is a loss function in deep learning used to measure the difference between predicted probabilities and actual class labels.

   (a) (2 points) Complete the binary cross-entropy loss function below by filling in the missing term:

$$\mathcal{L}_{\text{BCE}} = -\left[y \log(\hat{y}) + \underline{\qquad\qquad\qquad\qquad}\right]$$

   (b) (5 points) Then, draw a simple plot of the binary cross-entropy loss for both possible target values:

   - when $y = 1$
   - when $y = 0$

   Show how the loss changes as the predicted probability $\hat{y}$ varies between 0 and 1. Mark key points (e.g., $\hat{y} = 0, 0.5, 1$) on your plots.

   (c) (3 points) BCE loss is often paired with the sigmoid activation function in binary classification tasks.

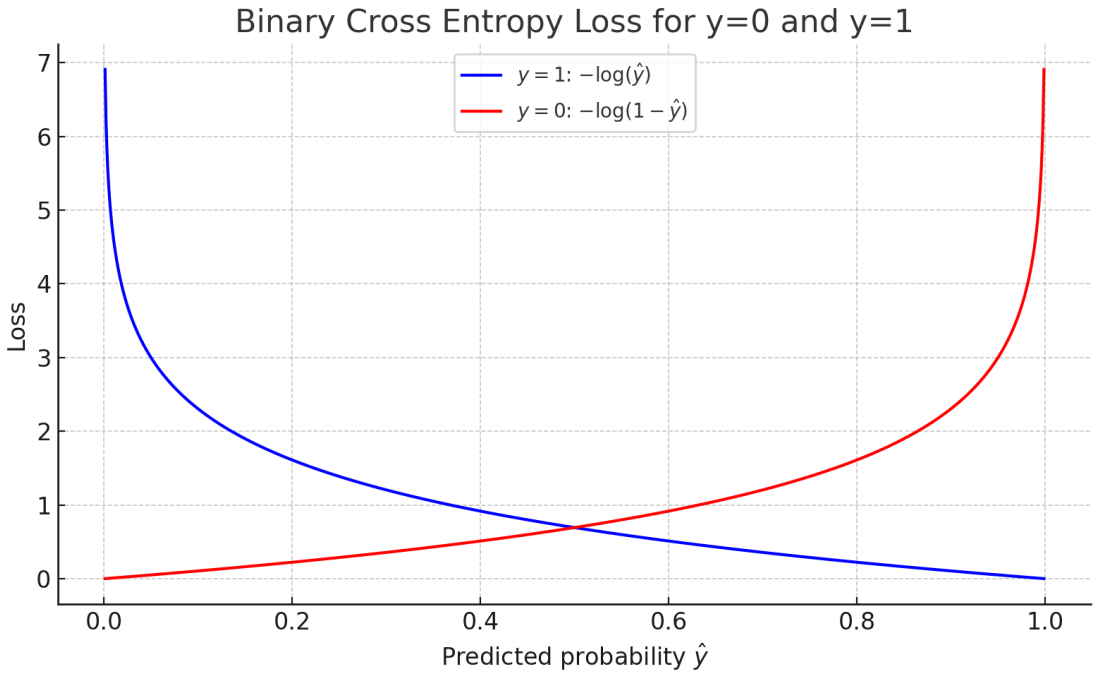
   - Why is this pairing important?
   - What would go wrong if we removed the sigmoid activation?

Solution:

(a): The complete equation for the BCE loss is:

$$\mathcal{L}_{\text{BCE}} = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

BCE loss visualized.



(b): Binary Cross Entropy (BCE) loss assumes that the predicted value $\hat{y}$ is a probability in the range $(0, 1)$, representing the model's confidence that the true label is 1.

(c): The sigmoid activation function maps raw model outputs (logits) $z \in (-\infty, \infty)$ to the range $(0, 1)$ using:

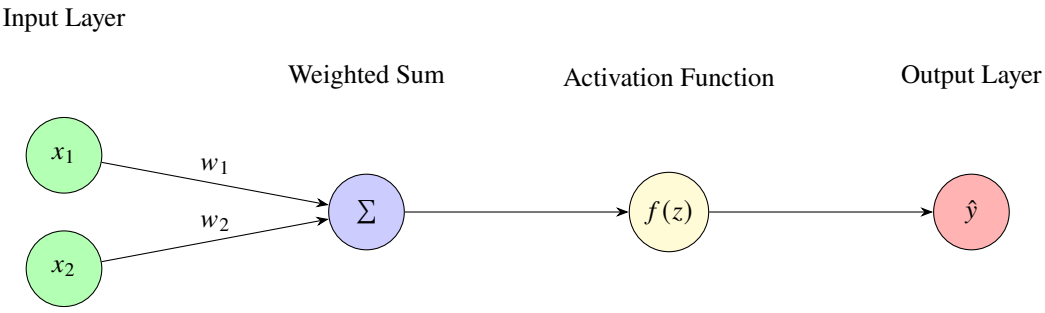
$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

This ensures that $\hat{y}$ is a valid probability, which is required by the BCE formula:

$$\mathcal{L}_{\text{BCE}} = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

If the sigmoid is omitted and raw logits are passed directly to the BCE loss, $\hat{y}$ may fall outside the $(0, 1)$ range.

4. (10 points) Backpropagation is an algorithm used to update the weights of a neural network by computing gradients of the loss function with respect to each parameter. The following diagram shows a simple single-layer neural network with two inputs and one output:



You are given the following:

- Inputs: $x_1 = 1.0, x_2 = 2.0$
- Weights: $w_1 = 0.1, w_2 = -0.2$
- Bias: $b = 0$
- Activation function: Identity ($f(z) = z$)
- Target output: $y = 1.0$
- Learning rate: $\eta = 0.1$

Compute one weight update according to the following steps:

(a) (2 points) Perform the forward pass and compute the prediction $\hat{y}$.

(b) (3 points) Compute the loss using Mean Squared Error: $L = \frac{1}{2}(y - \hat{y})^2$

(c) (3 points) Compute the gradient of the loss with respect to w_1 and w_2.

(d) (2 points) Update w_1 and w_2 using gradient descent.

Solution:

(a): Forward Pass

We compute the linear output:

$$z = x_1 w_1 + x_2 w_2 + b = 1.0 \cdot 0.1 + 2.0 \cdot (-0.2) + 0 = 0.1 - 0.4 = -0.3$$

Since the activation function is the identity:

$$\hat{y} = f(z) = z = -0.3$$

(b): Compute Loss

Using Mean Squared Error (MSE):

$$L = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(1.0 - (-0.3))^2 = \frac{1}{2}(1.3)^2 = \frac{1}{2} \cdot 1.69 = 0.845$$

(c): Compute Gradients

The gradient of the loss with respect to the prediction:

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = -0.3 - 1.0 = -1.3$$

Since $\hat{y} = z$, and $z = x_1 w_1 + x_2 w_2 + b$, the partial derivatives of the loss with respect to the weights are:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1} = -1.3 \cdot x_1 = -1.3 \cdot 1.0 = -1.3$$

$$\frac{\partial L}{\partial w_2} = -1.3 \cdot x_2 = -1.3 \cdot 2.0 = -2.6$$

(c): Weight Update

Using the learning rate $\eta = 0.1$:

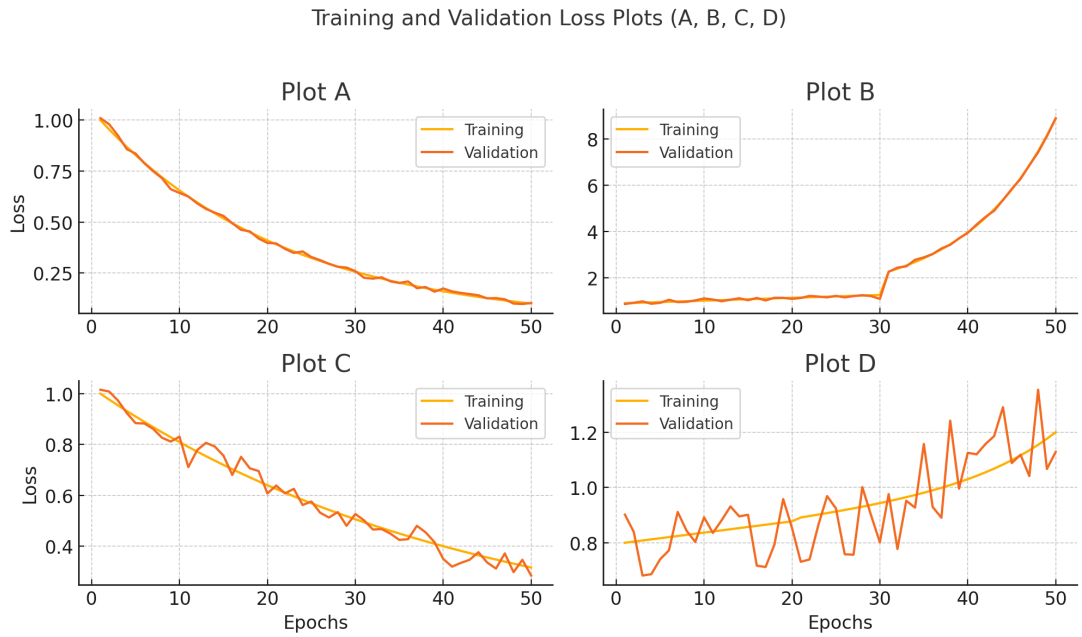
$$w_1 \leftarrow w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = 0.1 - 0.1 \cdot (-1.3) = 0.1 + 0.13 = \boxed{0.23}$$

$$w_2 \leftarrow w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = -0.2 - 0.1 \cdot (-2.6) = -0.2 + 0.26 = \boxed{0.06}$$

Final Answer:

$$w_1 = 0.23, \quad w_2 = 0.06$$

5. (10 points) Gradient explosion is a phenomenon in deep learning.
   (a) (2 points) What is gradient explosion in deep neural networks?
   (b) (3 points) Describe the phenomenon, how it occurs during training, and list common causes that can lead to it.
   (c) (5 points) Below you are given four plots (A, B, C, D), each showing training and validation loss over epochs for different models.



- Identify the plots that exhibit signs of gradient explosion.
- Justify your choices based on the behavior of the loss curves.

Solution:

(a): Gradient explosion is a phenomenon in deep neural networks where the magnitude of the gradients becomes excessively large during backpropagation. This leads to unstable weight updates, causing the loss to increase sharply or diverge entirely, sometimes resulting in NaNs.

(b):

- High learning rate
- Poor weight initialization (e.g., large initial values)
- Lack of normalization (e.g., missing batch normalization)
- Deep networks with repeated multiplication of weights during backpropagation
- Absence of gradient clipping in recurrent or sequential models

(c):

- **Plot A:** Training and validation losses decrease smoothly. No signs of instability. This is a healthy training curve.
- **Plot B:** Around epoch 30, the training and validation losses both increase rapidly. This is a clear sign of gradient explosion, possibly due to a high learning rate or lack of regularization.
- **Plot C:** Both curves show a gradual decrease with some noise. This is normal behavior. No gradient explosion is observed.
- **Plot D:** The training loss increases slowly, and the validation loss becomes highly erratic and unstable. This could indicate a late-stage gradient explosion or numerical instability.

Name: _____

Neptun: _____

Deep Network Development
Exam 2
May 23, 2025

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

We wish you the best of luck.

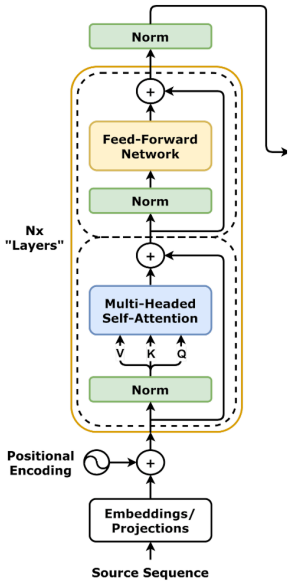
Exam Tasks

1. (10 points) The Transformer architecture revolutionized the processing of sequential data and the design of modern AI models. The original model introduced in [1] consists of an encoder–decoder structure.
   (a) (3 points) Draw a simplified diagram of the Transformer encoder architecture.
   (b) (5 points) Label the main components in your diagram and briefly describe the role of each one.
   (c) (2 points) What is the typical output of a Transformer encoder, and how is this output used within the decoder during inference or training?

Solution:

(a):

The encoder architecture visualized.



Transformer, full architecture.png
https://commons.wikimedia.org/wiki/File:Transformer,_full_architecture.png
dvgodoy, CC BY 4.0
<https://creativecommons.org/licenses/by/4.0>, via
Wikimedia Commons

(b): The Transformer encoder consists of the following key components:

- **Input Embedding:** Transforms discrete input tokens into continuous vector representations suitable for processing.
- **Positional Encoding:** Adds information about token positions to the embeddings, since the architecture itself has no recurrence or convolution.

- **Multi-Head Self-Attention:** Allows the model to attend to different positions of the input sequence simultaneously, capturing contextual relationships.
- **Feedforward Neural Network (FFN):** A two-layer fully connected network applied independently to each position, enabling non-linear transformations.
- **Residual Connections and Layer Normalization:** Ensure stable training and allow gradient flow by adding skip connections and normalizing outputs between layers.

(c): The output of the encoder is a sequence of context-aware vector representations (one per input token). Each vector captures both the meaning of the token and its relation to others in the sequence. These encoder outputs are used as keys and values in the decoder's encoder–decoder attention block, allowing the decoder to condition its predictions on the entire encoded input sequence.

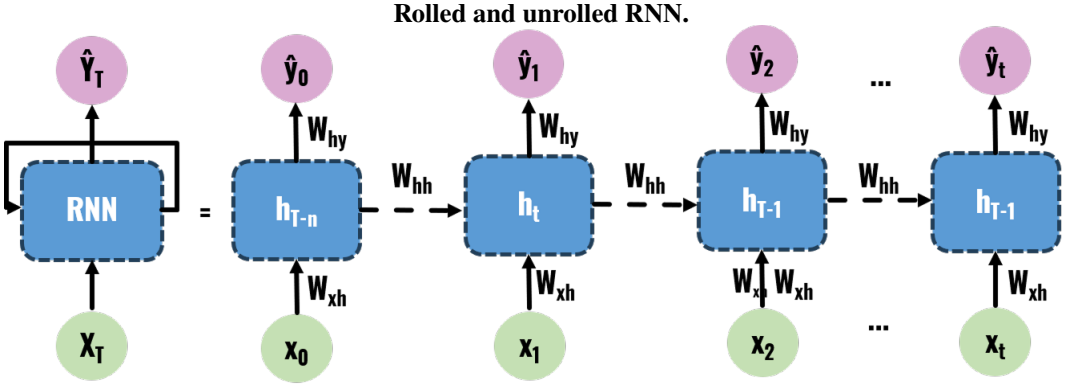
2. (10 points) Recurrent Neural Networks are designed to process sequential data by maintaining a hidden state across time steps. This question focuses on the basic (vanilla) RNN architecture.
   (a) (3 points) Illustrate both the rolled and unrolled versions of a vanilla RNN. Clearly indicate the flow of data and recurrence through time.
   (b) (6 points) Write down the mathematical equations for computing:
   - the hidden state h_t at time step t, and
   - the output y_t at time step t.

   Briefly explain each term and its role in the computation.
   (c) (1 point) Why are vanilla RNNs often inadequate for gradient-based learning, especially over long sequences?

Solution:

(a):

Rolled and unrolled RNN.



(b): The core equations for a vanilla RNN at time step t are:

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h)$$

$$y_t = W_y h_t + b_y$$

Where:

- x_t is the input at time step t
- h_t is the hidden state at time step t

- W_x is the weight matrix for the input
- W_h is the recurrent weight matrix for the hidden state
- b_h is the bias for the hidden layer
- tanh is the activation function (often tanh or ReLU)
- y_t is the output at time t
- W_y and b_y are the output weight matrix and bias

The hidden state h_t carries information from previous time steps and allows the network to maintain memory across the sequence. The output y_t is computed from this hidden state.

(c): Vanilla RNNs struggle with learning long-term dependencies due to the vanishing gradient problem. During backpropagation through time (BPTT), repeated multiplication of gradients through time steps can cause them to shrink exponentially, making it difficult to update earlier layers effectively. This limits the network's ability to learn patterns across long sequences.

3. (10 points) BLEU is a widely used metric in natural language processing for evaluating the quality of generated text.
   (a) (3 points) Briefly define the BLEU score. Can it be used as a loss function for training neural networks? Justify your answer with a short explanation.
   (b) (7 points) Calculate the BLEU-4 score for the predicted sentence using the two references below. Use Clipped Precision and apply the Brevity Penalty if applicable. No lemmatization, stemming, or token normalization should be performed—evaluate on raw tokens as shown.
      - **Predicted:** "He went to the store to buy some groceries"
      - **Reference 1:** "He went to the shop to get groceries"
      - **Reference 2:** "He went to the store for some food"

      Hint: Show clipped precision for $n = 1$ through $n = 4$, then compute geometric mean and brevity penalty.

Solution:

(a): The BLEU score is an automatic evaluation metric for machine-generated text, such as translations or summaries. It calculates the overlap between predicted and reference sequences using modified (clipped) n-gram precision and penalizes short outputs using a brevity penalty.

BLEU is not a loss function. It is non-differentiable and computed only after inference, making it unsuitable for gradient-based optimization during training.

(b): *Predicted:* "He went to the store to buy some groceries" (8 tokens) *Reference 1:* "He went to the shop to get groceries" (8 tokens) *Reference 2:* "He went to the store for some food" (8 tokens)

1-grams: Predicted: {He, went, to, the, store, to, buy, some, groceries} Match counts (clipped): He (1), went (1), to (1), the (1), store (1), some (1), groceries (1) → 7 matches Precision: $p_1 = \frac{7}{9}$

2-grams: Matched: (He went), (went to), (to the), (the store), (some groceries) → 5 matches Total: 8 predicted 2-grams $p_2 = \frac{5}{8}$

3-grams: Matched: (He went to), (went to the), (to the store) → 3 matches Total: 7 predicted 3-grams $p_3 = \frac{3}{7}$

4-grams: Matched: (He went to the), (went to the store) → 2 matches Total: 6 predicted 4-grams $p_4 = \frac{2}{6} = \frac{1}{3}$

Geometric mean of modified precisions:

$$P = \exp\left(\frac{1}{4}\sum_{n=1}^{4}\log p_n\right) = \exp\left(\frac{1}{4}(\log\frac{7}{9} + \log\frac{5}{8} + \log\frac{3}{7} + \log\frac{1}{3})\right) \approx \exp(-0.6931) = 0.5$$

Brevity Penalty (BP):

Predicted length = 9, Closest reference length = 8 Since predicted length > reference, $BP = 1$

Final BLEU score:

$$\text{BLEU-4} = BP \cdot P = 1 \cdot 0.5 = \boxed{0.50}$$

4. (10 points) Dropout and Batch Normalization are common techniques in modern deep learning models.
   (a) (3 points) Why are Dropout and Batch Normalization used in modern deep learning? Briefly explain their respective roles in training.
   (b) (2 points) List the key hyperparameters for both Dropout and Batch Normalization layers. Clearly separate the two.
   (c) (5 points) Describe in a few sentences:
      - What Dropout does during training and inference.
      - What Batch Normalization does and how it affects the forward pass.

Solution:

(a): Dropout is used as a regularization technique to reduce overfitting. It prevents complex co-adaptations of neurons by randomly dropping units (setting them to zero) during training, forcing the network to learn more robust features.

Batch Normalization is used to stabilize and accelerate training by normalizing layer inputs. It reduces internal covariate shift and helps gradients flow more smoothly, allowing the use of higher learning rates.

(b): Dropout Hyperparameters:

- `p` or `dropout rate`: The probability of dropping a unit (e.g., 0.5).

BatchNorm Hyperparameters:

- `momentum`: Used in updating running mean/variance.
- `epsilon` (ε): A small value added to variance for numerical stability.
- `affine`: Whether the layer has learnable scale (γ) and shift (β) parameters.

(c): Dropout: During training, dropout randomly sets a fraction of the input units to zero (based on the dropout rate). During inference, no units are dropped, but the outputs are scaled accordingly to maintain expected activation magnitudes.

BatchNorm: For each mini-batch, it normalizes the input features to have zero mean and unit variance, followed by an optional learnable affine transformation (scaling and shifting). This improves gradient flow and speeds up convergence during training.

5. (10 points) Activation functions play a key role in introducing non-linearity into neural networks.

   (a) (2 points) Why is it important to know the derivative of an activation function when training neural networks?

   (b) (8 points) Below you are given the equations of four activation functions and four unlabeled plots (A, B, C, D) showing their respective derivatives.

   Match each function to the correct plot of its derivative. Explain your choices.

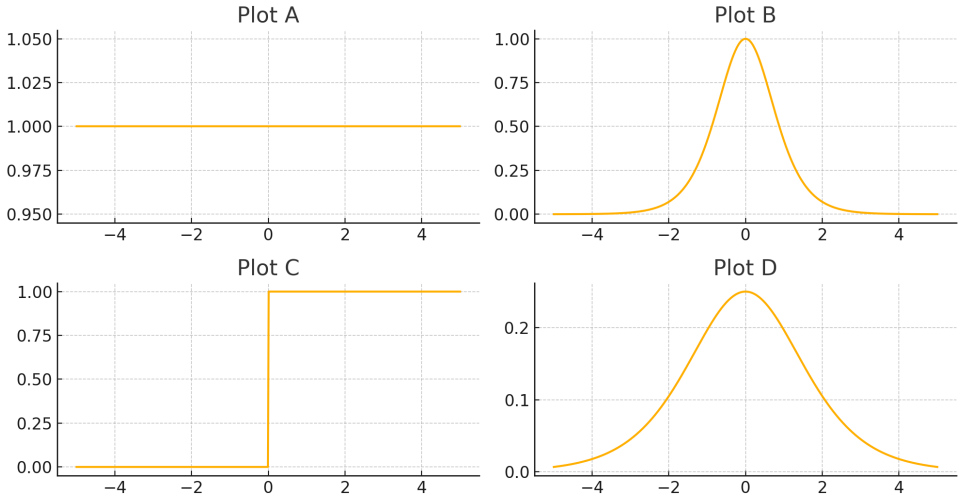
   Activation Functions:

$$(1) \quad f(x) = \frac{1}{1+e^{-x}} \qquad \text{(Sigmoid)}$$

$$(2) \quad f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad \text{(Tanh)}$$

$$(3) \quad f(x) = \max(0,x) \qquad \text{(ReLU)}$$

$$(4) \quad f(x) = x \qquad \text{(Linear / Identity)}$$



   Write your answers clearly, e.g., (1) → A, (2) → D, ...

Solution:

(a): Knowing the derivative of an activation function is essential key for backpropagation, as gradients are propagated through each layer during training. The shape and properties of these derivatives directly affect gradient flow and convergence.

(b): Matching each derivative plot to its corresponding activation function:

$$\text{Plot A} \rightarrow (4) \quad f(x) = x \quad \text{(derivative is constant 1)}$$

$$\text{Plot B} \rightarrow (2) \quad \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

$$\text{Plot C} \rightarrow (3) \quad \text{ReLU}(x) = \max(0,x)$$

$$\text{Plot D} \rightarrow (1) \quad \sigma(x) = \frac{1}{1+e^{-x}}$$

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

Name: _____

Neptun: _____

Deep Network Development
Exam 3
August 18, 2026

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

We wish you the best of luck.

Exam Tasks

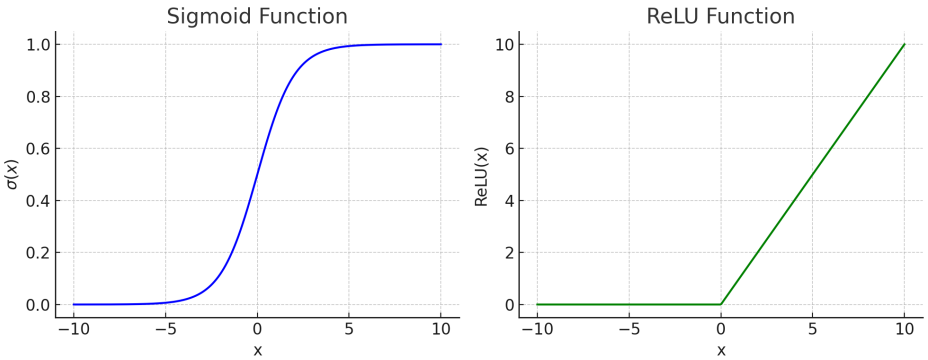
1. (10 points) Activation functions play a central role in enabling deep neural networks to learn complex representations.
   (a) (3 points) Consider a multi-layer perceptron (MLP) composed of linear layers. What would happen if all activation functions were removed from the network? What type of hypothesis function would the resulting model represent?
   (b) (4 points) Draw the shape of the activation functions listed below. Indicate clearly their behavior on negative, zero, and positive inputs.
   - $\sigma(x) = \dfrac{1}{1 + e^{-x}}$
   - $\mathrm{ReLU}(x) = \max(0, x)$

   (c) (3 points) Briefly describe the role and benefits of sigmoid and ReLU in modern deep neural networks. In what contexts are they typically used, and why?

Solution:

(a): If all activation functions are removed from a multi-layer perceptron (MLP), the network becomes a series of linear transformations. Since a composition of linear functions is itself linear, the resulting model is equivalent to a single linear transformation of the input. Therefore, the hypothesis function becomes a linear model.

(b): Students are expected to draw:



(c): Sigmoid: Used primarily in binary classification tasks and as the output activation for logistic regression or binary classification heads in neural networks. It compresses input values into the $(0, 1)$ range, making it interpretable as a probability.

ReLU: The most commonly used activation function in modern deep learning. It is computationally efficient and helps mitigate the vanishing gradient problem by having a constant gradient of 1 for positive values. ReLU enables sparse activation and works well in hidden layers of deep networks.

2. (10 points) Optical flow is a fundamental concept in computer vision, used to estimate motion between frames in a video sequence.
   (a) (3 points) Define optical flow. How does it differ from general motion detection or motion flow?
   (b) (2 points) Illustrate the difference between optical flow and motion flow using a simple known object. Annotated sketches are sufficient.
   (c) (3 points) State the optical flow constraint equation and briefly explain the assumptions it makes about brightness and motion.
   (d) (2 points) List two practical applications of optical flow in computer vision and briefly describe the role it plays in each.

Solution:

(a): Optical flow refers to the apparent motion of brightness patterns between consecutive frames in a video. It estimates the pixel-wise displacement vector field, indicating how each point in the image moves over time.

Motion flow (or motion detection), on the other hand, refers to the physical motion of objects in the 3D world. While optical flow estimates motion based on 2D image intensity changes, motion flow may involve depth, occlusion, and real-world object dynamics, which optical flow cannot directly capture.

(b): Illustration of optical and motion flow:

(c): The optical flow constraint equation is:

$$I_x u + I_y v + I_t = 0$$

Where:

- I_x, I_y, I_t are the partial derivatives of the image intensity with respect to x, y, and time t
- u, v are the horizontal and vertical components of the flow vector

This equation assumes:

- **Brightness constancy:** The intensity of a moving point remains constant over time.
- **Small motion:** The motion between consecutive frames is small enough for a Taylor approximation to be valid.

(d): Two example applications:

- **Autonomous Driving:** Optical flow is used to detect and track moving objects to avoid collisions.
- **Video Stabilization:** Flow fields help estimate and compensate for camera shake by aligning frames based on motion vectors.

3. (10 points) Loss functions are critical in training deep neural networks, guiding the optimization process by quantifying prediction errors.
   (a) (2 points) What is the most important mathematical property a loss function must satisfy in order to be usable in neural network training? Briefly explain why this property is necessary.
   (b) (8 points) Below are several commonly used loss functions along with a list of task categories. Match each loss function to one or more appropriate tasks. Then, justify your choices based on the purpose of the loss.

   Loss Functions:

Loss Name	Equation
Mean Squared Error (MSE)	$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Huber Loss	$\mathcal{L}_\delta = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } \lvert y-\hat{y}\rvert \le \delta \\ \delta(\lvert y-\hat{y}\rvert - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$
Cross Entropy (CE)	$\mathcal{L}_{\text{CE}} = -\sum_i y_i \log(\hat{y}_i)$
Binary Cross Entropy (BCE)	$\mathcal{L}_{\text{BCE}} = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$
Dice Loss	$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\sum_i y_i \hat{y}_i}{\sum_i y_i + \sum_i \hat{y}_i}$

Available Task Categories:
(A) Binary Classification (B) Multi-class Classification (C) Semantic Segmentation
(D) Object Detection (E) Regression (F) Neural Machine Translation (G) Sentiment Analysis

Write your matches clearly, e.g., (1) → E, (3) → B, F, G, etc. Provide 1–2 sentence justifications per match.

Solution:

(a): The most important property of a loss function in neural network training is that it must be differentiable. This is essential because training relies on gradient-based optimization methods (e.g., backpropagation) that require computing derivatives of the loss with respect to model parameters.

(b): Answers:

- Mean Squared Error (MSE) → E (Regression)
  MSE measures the average squared difference between predicted and true continuous values. It is used in tasks where the output is numeric, such as age prediction or temperature forecasting.
- Huber Loss → E (Regression), D (Detection)
  Huber Loss is robust to outliers by combining L1 and L2 behaviors. It is ideal for regression tasks with noisy data and is commonly used in bounding box regression in object detection.
- Cross Entropy Loss → B (Multi-class Classification), F (Translation), G (Sentiment Analysis)
  CE loss is used when outputs are categorical and mutually exclusive (e.g., image classification). It is also standard in NLP tasks like translation and sentiment analysis where the target is a sequence of categorical predictions.
- Binary Cross Entropy (BCE) → A (Binary Classification), D (Detection)
  BCE is suitable for binary classification tasks and multi-label setups, such as detecting multiple objects where each output is independent. It's also used for predicting presence/absence of object classes in detection heads.
- Dice Loss → C (Semantic Segmentation)
  Dice Loss measures overlap between predicted and ground truth masks, making it highly suitable for class-imbalanced tasks like segmentation. It focuses on spatial agreement and is differentiable, making it ideal for medical image segmentation, etc.

4. (10 points) You have recently started as an artificial intelligence intern at a media company. Your first task involves processing a collection of video files in which objects are falling from extreme heights. Your objective is to design a pipeline that estimates the number of falling objects over time in each video.
   (a) (5 points) You are given the following two options for solving the task. For each, describe how the pipeline works and justify whether or not it is suitable for counting falling objects:

1. Convert the video into a sequence of frames → Apply a segmentation model to detect objects → Count an object only if its segmentation mask moves between frames.
2. Apply a neural optical flow estimation method to compute motion vectors → Use motion magnitude or clustering of motion vectors to infer object presence and movement.
3. Convert video into grayscale frames → Calculate histogram differences between frames → Count objects when histogram difference exceeds a threshold.
4. Apply a classification network on each frame to predict "falling" vs "not falling" → Sum predictions to get object count.

(b) (2 points) What is the typical output of an optical flow estimation network? Describe its format and how it relates to the input video frames.

(c) (3 points) Optical flow networks are typically trained on which kind of dataset task categories? Choose from the following and justify your answer:

- (A) Image Classification
- (B) Object Detection
- (C) Segmentation
- (D) Motion Estimation / Correspondence

Solution:

(a): Answers:

1. **Valid.** This is a solid approach: segmenting objects and checking whether their masks change position between frames helps infer which instances are moving. However, this may miss very slow or partially occluded objects unless temporal continuity is handled carefully.
2. **Valid.** Optical flow captures pixel-level motion vectors, which can be clustered to identify independent moving objects. This is particularly effective when objects have no strong semantic boundaries but are clearly moving.
3. **Not valid.** Histogram difference is a weak signal for object movement. It may detect changes in lighting or camera shifts but lacks spatial resolution to distinguish individual objects, let alone count them.
4. **Not valid.** Classification models predict global labels for a frame. Even if trained to recognize "falling" scenes, they cannot distinguish the number of objects or track them temporally.

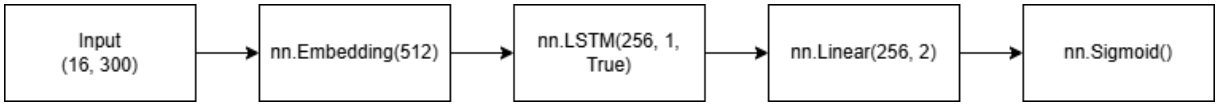
(b): The output of an optical flow estimator is a dense 2D motion field: a tensor of shape $(H \times W \times 2)$, where each pixel has an (x, y) displacement vector indicating its motion between two consecutive frames.

(c): The correct answer is: **(D) Motion Estimation / Correspondence**.

Optical flow networks are trained on synthetic or real datasets where pixel-level correspondences are known across consecutive frames. They require fine-grained motion ground truth and are not trained on traditional image classification, detection, or segmentation datasets.

5. (10 points) You are given a neural network designed for sentiment analysis on English text reviews. The preprocessing pipeline includes static tokenization using a fixed vocabulary of size 14,832. Each tokenized review is padded or truncated to a maximum length of 300 tokens. Batch size is set to 16.

   The network architecture is as follows:



(a) (3 points) Explain the role of the embedding layer in this architecture. What is the size of the embedding matrix in terms of its shape?

(b) (3 points) Is it reasonable for the LSTM's hidden dimension to be smaller than the embedding size? Justify your answer. Also, describe the shape of the LSTM layer's output (assume batch size is 16 and the LSTM layer has $batch_first$ set to $True$).

(c) (4 points) Determine the shape of the weight matrix in the final linear projection layer. What does this layer do in terms of classification? Comment on whether a sigmoid activation is the appropriate choice here.

Solution:

(a): The embedding layer maps discrete word indices (integers from the vocabulary) into continuous dense vectors of fixed size. This allows the model to capture semantic relationships between words. Since the vocabulary size is 14,832 and the embedding dimension is 512, the shape of the embedding matrix is:

$$\text{Embedding matrix shape: } (14832 \times 512)$$

(b): Yes, it is reasonable for the LSTM's hidden size (256) to be smaller than the embedding size (512). The embedding space is optimized for representing lexical semantics, while the LSTM hidden state captures sequence-level dependencies, which may not require the full dimensionality.

The output of a unidirectional LSTM layer processing a sequence of length 300 with hidden size 256 is:

$$\text{Output shape: } (B \times T \times H) = (16 \times 300 \times 256)$$

(c): The final linear layer maps the LSTM's output into a 2D output space for binary classification. Assuming we use the final hidden state, the input to the linear layer is a vector of size 256, and the output is of size 2. Therefore, the weight matrix has shape:

$$\text{Weight matrix shape: } (2 \times 256)$$

Since the model uses a sigmoid activation over sentiments, it may be more appropriate to use single output with sigmoid for binary classification (0/1).

Name: _____

Neptun: _____

Deep Network Development
Exam 4
August 18, 2026

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

We wish you the best of luck.

Exam Tasks

1. (10 points) You are training a PyTorch neural network for an image classification task with 10 output classes. The model outputs raw logits. You have already defined your model, optimizer, and loss function (`criterion`), and placed the model on the appropriate device (`device`). The dataset and data loaders (`train_loader`, `valid_loader`) are fully implemented.

   Below is a partially completed training and evaluation loop. Fill in the 10 missing lines (marked with `# (1)` to `# (10)`) to complete a working PyTorch training setup. Assume all standard PyTorch conventions and imports are available.

```
epochs = 100
for epoch in range(epochs):
    model.train()
    for batch in train_loader:
        inputs, labels = batch
        # (1)
        # (2)
        # (3)
        outputs = model(inputs)
        # (4)
        loss.backward()
        # (5)

    # (6)
    correct = 0
    total = 0
    # (7):
        for batch in valid_loader:
            inputs, labels = batch
            # (8)
            labels = labels.to(device)
            outputs = model(inputs)
            # (9)
            total += labels.size(0)
            correct += # (10)
    print(f"Epoch {epoch+1}, Accuracy: {100 * correct / total:.2f}%")
```

Solution:

```
epochs = 100
for epoch in range(epochs):
    model.train()
    for batch in train_loader:
        inputs, labels = batch
        inputs = inputs.to(device)
        labels = labels.to(device)
```

```
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for batch in valid_loader:
            inputs, labels = batch
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    print(f"Epoch {epoch+1}, Accuracy: {100 * correct / total:.2f}%")
```

2. (10 points) Depth estimation has become increasingly important in computer vision, especially in the context of monocular depth estimation.
   (a) (3 points) What is the depth estimation task? Describe its expected input and output.
   (b) (2 points) Why are most deep learning models built around monocular depth estimation? Why is stereo depth estimation less commonly used in this context?
   (c) (2 points) List two state-of-the-art models for monocular depth estimation.
   (d) (3 points) Define relative depth estimation and absolute (metric) depth estimation. What are the key differences between the two in terms of learning objectives and application scenarios?

Solution:

(a): The task of depth estimation involves predicting the distance from the camera to each pixel in an image. The input is typically a single RGB image (in monocular depth estimation), and the output is a dense depth map of the same spatial resolution, where each pixel represents the estimated depth value.

(b): Monocular depth estimation has gained popularity because it requires only a single image, making it suitable for a wide range of applications where stereo or multi-view setups are not feasible (e.g., mobile phones, autonomous drones). In contrast, stereo depth estimation relies on precise calibration and alignment of two cameras, and its performance degrades in low-texture or occluded regions. Monocular methods can also leverage large-scale datasets without depth labels via self-supervised learning techniques.

(c): Two SOTA monocular depth estimation models can be:

- **MiDAS**
- **ZoeDepth**
- **PatchFusion**
- **Marigold**
- **Depth Anything v1-2**
- **Depth Pro**

(d): Relative depth estimation aims to infer the relative ordering or ranking of depths between regions or objects in a scene (e.g., which object is closer), without predicting actual metric distances.

Absolute (metric) depth estimation, on the other hand, seeks to predict real-world distances (e.g., in meters) for each pixel.

The key difference lies in scale sensitivity: relative methods are often trained without ground-truth scale and are more robust to dataset variation, while absolute methods require precise ground-truth depth annotations and are suited for applications like robotics and 3D reconstruction.

3. (10 points) You are developing an object detection model for an automated production line system, where the task is to detect and count items as they pass beneath a camera. During training, you observe that the input images captured from the real-time camera feed contain significant random visual noise due to poor lighting and vibrations in the machinery.
   (a) (2 points) How does the presence of random noise in training images typically affect the training progress and convergence of the object detection model?
   (b) (3 points) As a machine learning engineer, you suggest a solution that involves augmenting the training set to match real-world conditions. Which of the following data augmentation strategies is most appropriate for this scenario?
      A. Apply heavy blur and color shifting to all training images.
      B. Randomly crop the training images to 50% of their original size.
      C. Add Gaussian noise to the training images during preprocessing.
      D. Reduce image resolution to simulate low-light sensors.
   (c) (2 points) Explain why the selected augmentation technique (from the previous question) can improve the model's generalization in noisy real-world conditions.
   (d) (3 points) You decide to perform ablation experiments to evaluate the model's robustness. Which of the following outcomes would most strongly indicate that the model is overfitting to clean data?
      A. The model performs well on validation data with noise but poorly on clean test data.
      B. The model achieves high accuracy on clean validation data but fails on noisy real-world images.
      C. The model's loss fluctuates during training, but both training and validation performance improve.
      D. The model shows a consistent drop in precision and recall across all datasets.

Solution:

(a): The presence of random noise in the training images can negatively affect the model's ability to converge. The model may learn spurious patterns or fail to detect consistent features, leading to poor generalization and slower or unstable training. It may also overfit to noisy artifacts if not properly regularized or augmented.

(b): The correct choice is:

- **(C.)** Add Gaussian noise to the training images during preprocessing.

(c): Adding Gaussian noise to the training images closely simulates the type of noise observed in real-world conditions (e.g., sensor noise, environmental interference), allowing the model to learn robust features that generalize despite noise.

(d): The correct answer is:

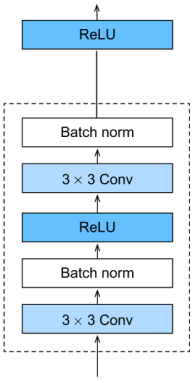
- **(B.)** The model achieves high accuracy on clean validation data but fails on noisy real-world images.

This suggests the model has learned to rely on features that exist only in clean images and cannot generalize to noisy inputs, indicating overfitting to the clean training distribution.

4. (10 points) The following diagrams illustrate simplified versions of well-known neural network architectures, each missing one core structural component.

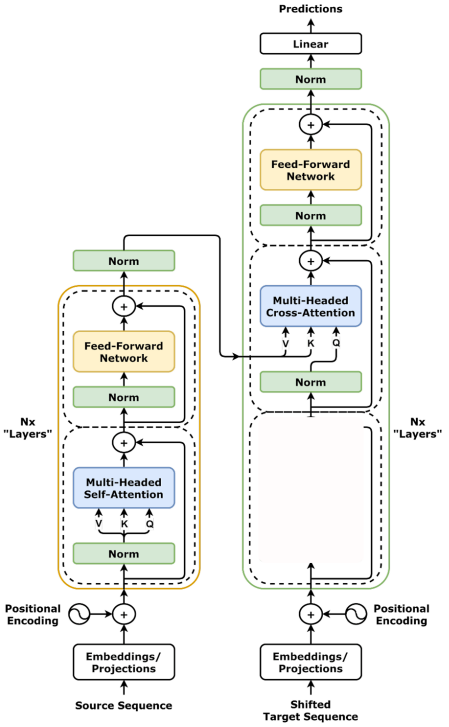
   Your task is to identify the missing component and explain its original purpose in the architecture, as well as what would likely go wrong or degrade in the network's performance without it.

(a) (2 points) ResNet Block



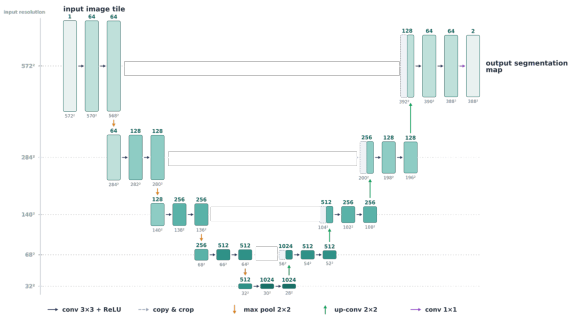
ResNet block.svg
https://commons.wikimedia.org/wiki/File:ResNet_block.svg
Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J., CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

(b) (4 points) Transformer Architecture

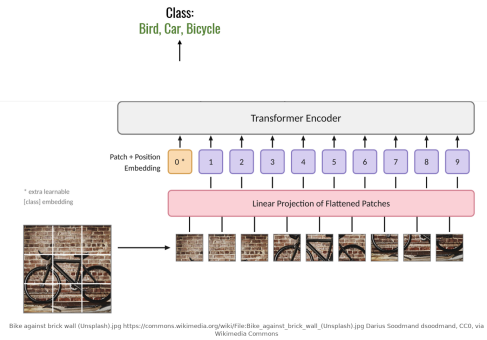


Transformer, full architecture.png https://commons.wikimedia.org/wiki/File:Transformer,_full_architecture.png dvgodoy, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons

(c) (2 points) U-Net



(d) (2 points) Vision Transformer



Bike against brick wall (Unsplash).jpg https://commons.wikimedia.org/wiki/File:Bike_against_brick_wall_(Unsplash).jpg Darius Soodmand dsoodmand, CC0, via Wikimedia Commons

Solution:

(a): Residual connections in ResNet allow gradients to flow more easily through deep networks by enabling identity mappings. Removing them leads to vanishing gradients, poor convergence, and limited depth scalability.

(b): Masked multi-head self-attention is essential in the decoder for preserving autoregressive behavior—without it, the model can *see* future tokens during training. The LayerNorm and residual addition operations stabilize and regulate learning; their absence leads to training instability and degraded performance.

(c): Skip connections in U-Net carry fine-grained spatial details from the encoder to the decoder, helping

restore resolution. Without them, the decoder operates with overly abstracted features, resulting in blurry or imprecise segmentation outputs.

(d): The MLP head processes the class token after the transformer encoder and produces class logits. Without it, the model cannot map the learned representation to a meaningful prediction space, effectively disabling classification.

5. (10 points) Attention mechanisms are a central component in modern neural architectures, allowing models to dynamically focus on relevant parts of the input. In this task, you will analyze and compute dot-product attention using small matrices.
   (a) (2 points) What is dot-product attention? Write down the formula and briefly describe its role in deep learning models.
   (b) (2 points) What are the roles of the matrices Q, K, and V in the attention mechanism? What does each one represent during the computation?
   (c) (6 points) Given the following matrices:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad V = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

   Compute the dot-product attention using the formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

   where d_k is the dimensionality of the key vectors. Show all intermediate steps: matrix multiplications, scaling, softmax computation, and the final output.

Solution:

(a): Dot-product attention computes a weighted sum of values based on the similarity between queries and keys. The formula is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

This mechanism allows the model to focus on relevant parts of the input when computing outputs.

(b): Answer:

- Q (Query): the current element looking for relevant context.
- K (Key): the elements being compared to the query.
- V (Value): the information retrieved/aggregated based on attention scores.

(c): Given:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad K = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Step 1: Compute $QK^\top$

$$K^\top = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \quad \Rightarrow \quad QK^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Step 2: Scale by $\sqrt{d_k}$, where $d_k = 2 \Rightarrow \sqrt{2} \approx 1.414$

$$\frac{QK^\top}{\sqrt{2}} \approx \begin{bmatrix} 0 & 0 \\ 0.71 & 0.71 \end{bmatrix}$$

Step 3: Apply softmax row-wise:

- Row 1: softmax of $[0, 0] \rightarrow$ equal values $\rightarrow [0.5, 0.5]$ - Row 2: softmax of $[0.71, 0.71] \rightarrow$ equal values $\rightarrow [0.5, 0.5]$

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{2}}\right) = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Step 4: Multiply by V:

$$\begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Final Output:

$$\text{Attention}(Q, K, V) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Name: _____

Neptun: _____

Deep Network Development
Exam 5
June 17, 2025

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

We wish you the best of luck.

Exam Tasks

1. (10 points) Activation functions introduce non-linearity into neural networks, allowing them to model complex patterns and decision boundaries. Consider a convolutional neural network designed for image classification, where each convolutional block ends with a ReLU activation function.
   (a) (3 points) What would happen if all the ReLU activations in the network were replaced with Sigmoid activations? Describe the effect on gradient flow, training stability, and performance.
   (b) (3 points) What would be the consequence of removing all activation functions entirely? How would this affect the representational power of the model?
   (c) (4 points) Consider the following network: a CNN backbone with ReLU activations, followed by a fully connected layer, and a final softmax output layer. During training, you observe that the gradients are vanishing rapidly and the model fails to learn. You suspect the activation function might be involved. Which configuration(s) of activation functions could be contributing to vanishing gradients despite using ReLU, and what architectural remedies might help resolve this?

   Activation functions play a central role in enabling deep neural networks to learn complex representations.

Solution:

(a): Replacing ReLU with Sigmoid introduces several issues. Sigmoid outputs are bounded in the range $(0, 1)$, which causes two main problems:

- The gradients saturate for large positive or negative inputs, leading to the vanishing gradient problem.
- It slows down convergence, especially in deep networks like CNNs.

As a result, training becomes unstable or extremely slow, and the network struggles to learn expressive features from images.

(b): Removing all activation functions makes the entire network a composition of linear operations. Mathematically, a stack of linear transformations collapses into a single linear transformation, meaning the network loses its ability to model non-linear relationships in the data. The result is that the CNN becomes no more powerful than a single linear layer, severely limiting its performance.

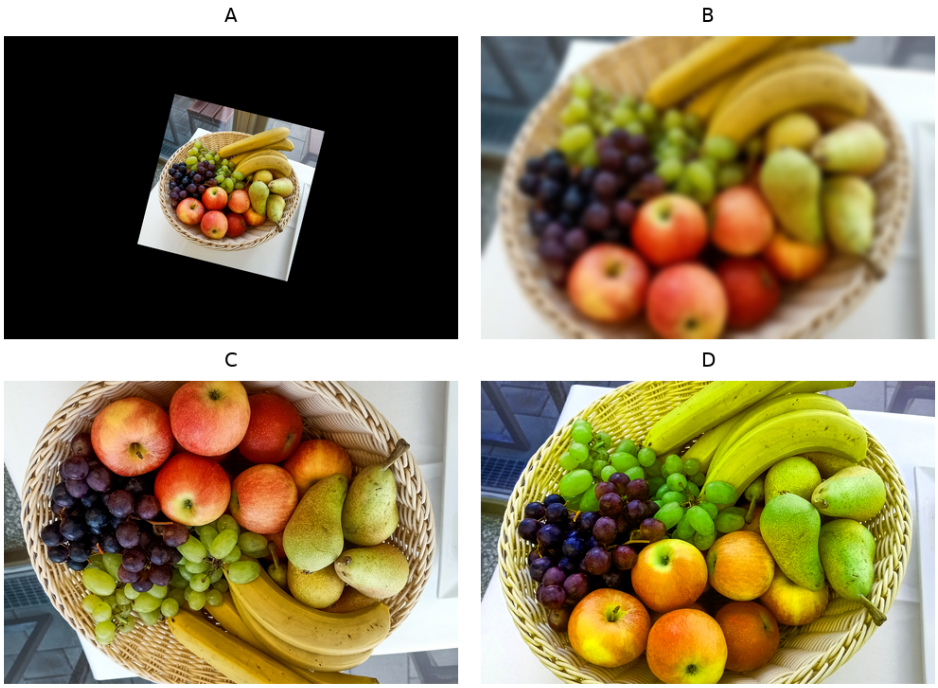
(c): Even though ReLU is used in the convolutional blocks, vanishing gradients can still occur due to:

- Improper weight initialization, which can lead to dead neurons (ReLU outputs zero permanently).
- Overuse of activations like softmax or tanh in intermediate layers, which squash values and reduce gradient magnitudes.
- Deep networks with many layers of ReLU followed by BatchNorm in the wrong order (e.g., activation before normalization).

Remedies:

- Use He initialization with ReLU to prevent dead activations.
- Add BatchNorm layers after convolution and before ReLU.
- Consider LeakyReLU or GELU to maintain some gradient in negative regimes.
- Verify that ReLU is not mistakenly placed after the output layer if softmax is already used there.

2. (10 points) Data augmentation is a widely used technique in deep learning to artificially expand training data and improve generalization.
   (a) (3 points) What do augmentation techniques aim to simulate in the training process? Why do they help models generalize better?
   (b) (4 points) Match the four images below to the augmentation applied.



Suyash Dwivedi, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
https://commons.wikimedia.org/wiki/File:A_basket_of_fruits.jpg A basket of fruits.jpg

Augmentation choices:

1. Gaussian Blur
2. Random Cutout
3. Color Jitter
4. Rotation
5. Horizontal Flip
6. Perspective Transform
7. Random Affine Transformation
8. Grayscale Conversion
9. Vertical Flip

Write your matchings clearly, e.g., B $\rightarrow$ 1, etc.

   (c) (3 points) Which of the augmentation methods listed above would likely not be suitable for an optical character recognition task? Justify your answer with reasoning specific to OCR sensitivity.

Solution:

(a): Data augmentation techniques simulate the variability found in real-world data, such as shifts in viewpoint, lighting changes, or occlusions, to make the training dataset more diverse. These methods help models generalize better by reducing overfitting and encouraging robustness to irrelevant transformations. For instance, a flipped or blurred image should still represent the same class, and training the model to recognize such variants leads to better performance on unseen data.

(b):

- A $\rightarrow$ 7 (Random Affine Transformation)
- B $\rightarrow$ 1 (Gaussian Blur)
- C $\rightarrow$ 9 (Vertical Flip)
- D $\rightarrow$ 3 (Color Jitter)

(c): The following augmentation methods may not be suitable for optical character recognition (OCR):

- **Random Cutout (2)**: This may obscure critical parts of characters, breaking the legibility and structure of the text.
- **Perspective Transform (6)**: May cause unnatural warping, making letters slanted or compressed, reducing readability.
- **Horizontal Flip (5)**: This can reverse the character sequence, especially problematic for left-to-right languages like English.
- **Vertical Flip (9)**: Flips text upside down, which is almost never valid in real-world OCR scenarios.
- **Gaussian Blur (1)**: Can obscure character edges and fine details, which are essential for recognizing text.

3. (10 points) The patching operation is a crucial component of the Vision Transformer architecture for image understanding.
   (a) (2 points) What is the purpose of the patching operation in Vision Transformers? How does it differ from convolutional feature extraction in CNNs?
   (b) (2 points) Besides fixed-size patch embedding, name at least one alternative image tokenization method that can be used in ViT-like models.
   (c) (3 points) You are working with a dataset of 256×256 pixel RGB images. You apply a patching operation that splits each image into non-overlapping 16×16 patches. How many patch tokens are generated per image?
   (d) (3 points) In the original Vision Transformer architecture [1], what is the role of the `[CLS]` token?

Solution:

(a): The patching operation in Vision Transformers divides an image into smaller non-overlapping patches. Each patch is flattened and linearly projected into a vector to serve as a token, similar to word tokens in NLP. Unlike CNNs, which extract features using convolutional filters hierarchically, ViTs treat patches as unordered tokens and rely entirely on attention mechanisms to learn spatial relationships.

(b): Alternative image tokenization methods include:

- **Shifted Patch Tokenization (SPT)**

- **Convolutional Tokenization**
- **Superpixel Tokenization**

(c): Each 256×256 image is divided into patches of size 16×16. The number of patches along one dimension is:

$$\frac{256}{16} = 16$$

Thus, the total number of image tokens per image is:

$$16 \times 16 = \boxed{256 \text{ tokens}}$$

(d): The `[CLS]` token in ViT is a special learnable embedding prepended to the sequence of patch tokens. It serves as a summary representation of the entire image. After passing through the transformer layers, the final state of the `[CLS]` token is used for classification tasks. During training, gradients are propagated through the `[CLS]` token, allowing it to learn how to aggregate information from all patch tokens via self-attention.

4. (10 points) The learning rate is a critical hyperparameter that influences the convergence behavior of deep neural networks during training.
   (a) (2 points) Is the learning rate a bounded value? If so, what are the typical bounds and why are they important?
   (b) (3 points) Explain what could happen to a neural network's training dynamics if:
      - the learning rate is set too high
      - the learning rate is set too low
   (c) (5 points) Many training schedules adjust the learning rate dynamically using a decay function. Suppose the learning rate is defined as:

$$\eta_t = \eta_0 \cdot \gamma^t$$

   where:
      - η_t is the learning rate at time step t
      - η_0 is the initial learning rate
      - γ is the decay factor, with $0 < \gamma < 1$

   If the learning rate at time step $t = 5$ is $\eta_5 = 0.001$ and the decay factor is $\gamma = 0.8$, calculate the original learning rate η_0 used at the beginning of training.

Solution:

(a): Yes, learning rates are typically bounded (technically not). The learning rate η must be greater than 0 and is usually less than 1. Values like 0.1, 0.01, or 0.001 are common in practice. If it is too large, the model may diverge; if it is too small, training can stagnate or converge extremely slowly.

(b): Answers:

- **Too high:** The model's updates may overshoot the optimal point, leading to divergence or oscillations in the loss function.
- **Too low:** The model will take excessively small steps, resulting in very slow convergence and possibly getting stuck in suboptimal regions.

(c): We are given:

$$\eta_5 = 0.001, \quad \gamma = 0.8, \quad t = 5$$

Using the exponential decay formula:

$$\eta_t = \eta_0 \cdot \gamma^t$$

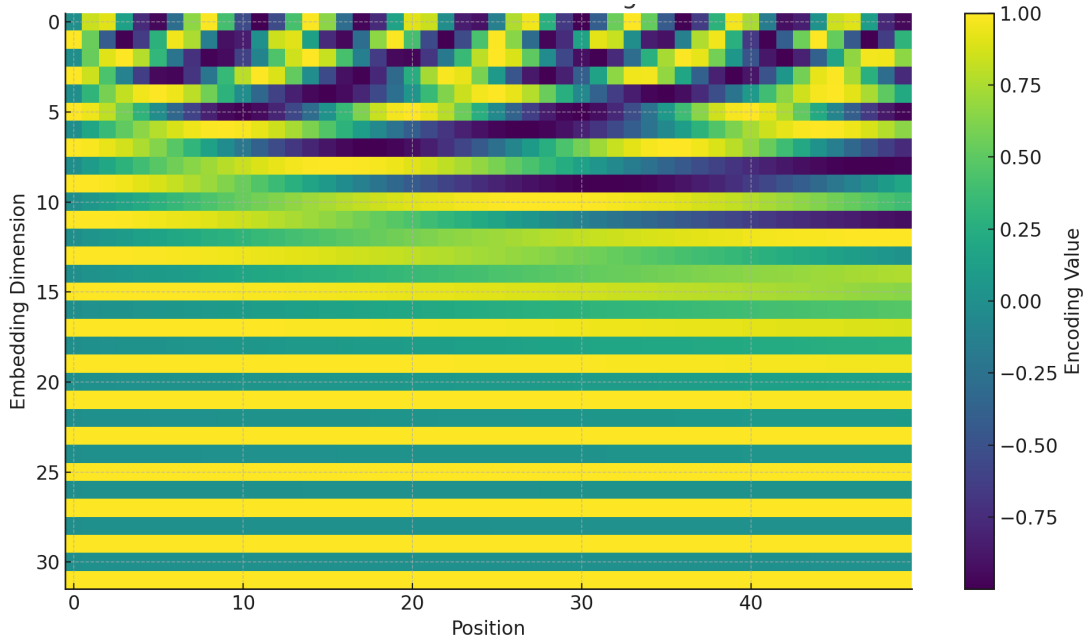
Substitute in the known values:

$$0.001 = \eta_0 \cdot 0.8^5 \Rightarrow \eta_0 = \frac{0.001}{0.8^5} = \frac{0.001}{0.32768} \approx \boxed{0.00305}$$

5. (10 points) Positional encoding is a crucial component of transformer-based architectures.
   (a) (2 points) What are positional encodings used for in transformer models? Why are they necessary?
   (b) (3 points) What is sinusoidal positional encoding? Below is the incomplete formulation for generating positional encoding vectors of dimension d:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \quad PE_{(pos,2i+1)} = \underline{\qquad\qquad\qquad\qquad}$$

   Fill in the missing part of the equation above, and briefly explain why sinusoidal encodings can generalize to sequences longer than those seen during training.
   (c) (2 points) List at least one other positional encoding technique used in modern transformer variants.
   (d) (3 points) Examine the plot of a positional encoding matrix shown below:



   What kind of encoding is shown in this plot? Justify your answer by interpreting the patterns you observe.

Solution:

(a): Positional encodings are used in transformer architectures to inject information about the position (or order) of tokens in a sequence. Since the transformer's self-attention mechanism is inherently permutation-invariant, it lacks any notion of sequence order. Positional encodings resolve this by providing a unique, learnable or deterministic vector for each position, enabling the model to capture positional relationships.

(b): Sinusoidal positional encoding is a deterministic method introduced in the original Transformer paper. It uses sine and cosine functions of varying frequencies to generate unique position vectors. The formulation is:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

(c): Other positional encoding techniques include:

- **Learnable Position Embeddings:** Each position is associated with a learnable embedding vector, similar to token embeddings.
- **Rotary Positional Embedding (RoPE):** Encodes position by rotating the query and key vectors in a complex-valued space.
- **ALiBi (Attention with Linear Biases):** Adds a bias term to the attention logits based on position distance instead of encoding position into the input.

(d): The provided plot clearly shows sinusoidal positional encoding, as evidenced by the smooth, wave-like curves of different frequencies across dimensions. In this encoding:

- Even-numbered dimensions show sinusoidal curves (sine),
- Odd-numbered dimensions alternate with cosine curves,
- Lower dimensions capture slower variations, and higher dimensions capture faster fluctuations,

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Name: _____

Neptun: _____

Deep Network Development
Exam 6
June 17, 2025

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

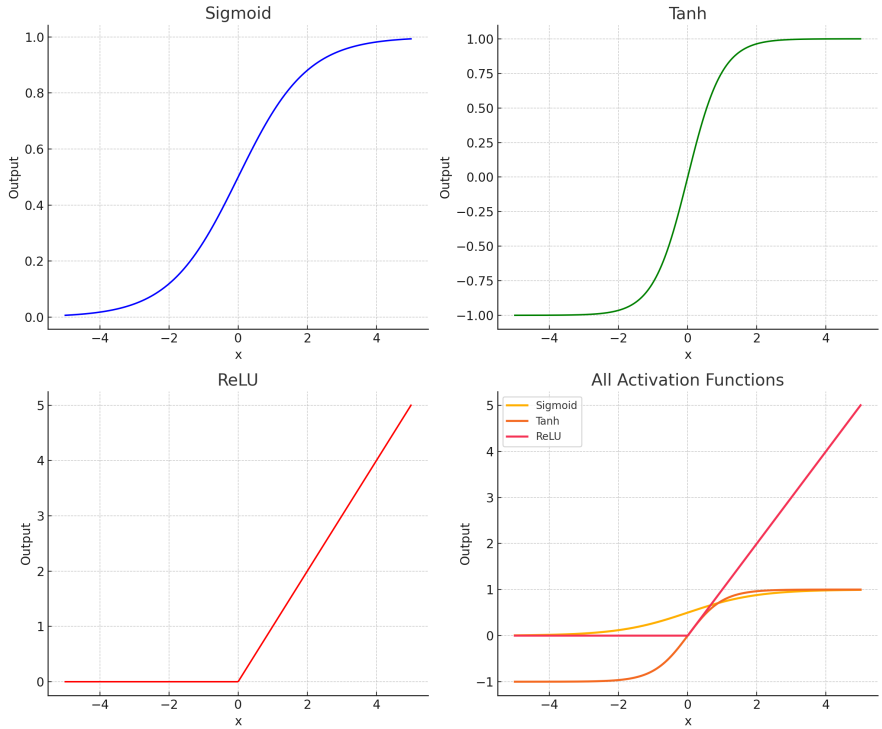
We wish you the best of luck.

Exam Tasks

1. (10 points) Activation functions introduce non-linearity into neural networks, enabling them to model complex patterns and learn non-trivial decision boundaries.
   (a) (4 points) Given the following activation functions, sketch a 2D plot for each of them (hand-drawn). Use the range $x \in [-5, 5]$:
      - Sigmoid
      - Tanh
      - ReLU
   (b) (3 points) Below each plot, write the mathematical formula of the corresponding activation function.
   (c) (3 points) Analyze and compare the output behavior of each activation function as the input x becomes very large (positive) or very small (negative). Which functions saturate? Which ones maintain gradient flow?

Solution:

(a): Plots:



(b): Formulas:

- **Sigmoid:**
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
- **Tanh:**
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
- **ReLU:**
$$\text{ReLU}(x) = \max(0, x)$$

(c): Answers

- **Sigmoid:** As $x \to +\infty$, $\sigma(x) \to 1$; as $x \to -\infty$, $\sigma(x) \to 0$. The gradient vanishes in both extremes, causing the "vanishing gradient" problem.
- **Tanh:** As $x \to +\infty$, $\tanh(x) \to 1$; as $x \to -\infty$, $\tanh(x) \to -1$. Like sigmoid, it saturates at both ends, but it is zero-centered, which helps optimization.
- **ReLU:** As $x \to +\infty$, $\text{ReLU}(x) \to x$ (no saturation); for $x < 0$, the gradient is zero. This avoids saturation for positive values and helps gradient flow, though it can cause "dead neurons" if too many activations are in the negative range.

2. (10 points) YOLO and DETR represent two fundamentally different approaches to object detection. While both are state-of-the-art detectors, they differ significantly in design and philosophy.
   (a) (3 points) Is YOLO considered a single-stage or two-stage detector? Justify your answer based on its architectural design and inference behavior.
   (b) (3 points) Compare the architectural pipeline of YOLO with that of DETR. Focus your answer on how predictions are generated.
   (c) (4 points) DETR introduces the concept of "set prediction" using a transformer and bipartite matching.
   - Explain what "set prediction" means in this context.
   - Why does this make traditional components like Non-Maximum Suppression (NMS) unnecessary?
   - What are the trade-offs of this approach compared to grid-based detectors like YOLO?

Solution:

(a): YOLO is a single-stage object detector. This means it performs object localization and classification in a single forward pass through the network, without a separate region proposal step. It divides the image into a grid and directly predicts bounding boxes and class probabilities from image features, making it very fast and efficient at inference time.

(b): The architectural differences between YOLO and DETR are significant:

- **YOLO:** Uses a convolutional backbone to extract features, followed by fully connected or convolutional layers that predict bounding box coordinates, confidence scores, and class probabilities over a grid. It relies heavily on post-processing like Non-Maximum Suppression (NMS) to remove overlapping boxes.
- **DETR:** Uses a CNN backbone for feature extraction, followed by a transformer encoder-decoder structure. It predicts a fixed-size set of bounding boxes using learned object queries and matches predictions to ground truth using Hungarian matching. DETR does not rely on NMS.

(c): Answers

- **Set prediction:** In DETR, object detection is formulated as a set prediction problem where the model outputs a fixed-size set of predictions, and Hungarian matching is used to assign predictions to ground truth boxes optimally. This avoids duplicate detections.
- **No NMS:** Because of the one-to-one matching between predictions and ground truth, each object is predicted only once, eliminating the need for Non-Maximum Suppression.
- **Trade-offs:** DETR offers a cleaner end-to-end pipeline with fewer heuristic post-processing steps. However, it typically requires more training data and longer convergence times. YOLO is more lightweight and converges faster but relies on handcrafted components like anchor boxes and NMS.

3. (10 points) Regularization is a fundamental concept in deep learning that aims to improve a model's ability to generalize to unseen data.
   (a) (2 points) What is weight regularization, and to which component of the training objective is it applied? Briefly explain its purpose and effect on model training.
   (b) (5 points) Match each regularization technique below to its corresponding formulation or description. Focus on the core idea behind each method.
   Match the Regularization Techniques to their Descriptions:

Technique	**Description**
(1) L1 Regularization	(A) Terminates training when validation loss stops improving
(2) L2 Regularization	(B) Constrains weight magnitudes to remain within a fixed range
(3) Dropout	(C) Applies a random binary mask to layer outputs
(4) Early Stopping	(D) Adds $\lambda \sum_i \lvert w_i \rvert$ to the loss
(5) Weight Clipping	(E) Adds $\lambda \sum_i w_i^2$ to the loss

   Write your matchings clearly (e.g., $1 \rightarrow$ D, $2 \rightarrow$ A, etc.). Briefly justify one of your choices.
   (c) (3 points) Among the listed regularization methods, which are active only during training and which affect the model's structure or behavior at inference time as well? Provide examples and justify your classification.

Solution:

(a): Weight regularization refers to the addition of a penalty term to the loss function that discourages large weight values. It is applied directly to the model's loss function during training. The purpose is to reduce model complexity, help prevent overfitting, and improve generalization by pushing weights toward smaller magnitudes.

(b): Matching Answers:

Regularization Technique	**Matches Description**
(1) L1 Regularization	(D) Adds $\lambda \sum_i \lvert w_i \rvert$ to the loss
(2) L2 Regularization	(E) Adds $\lambda \sum_i w_i^2$ to the loss
(3) Dropout	(C) Applies a random binary mask to layer outputs
(4) Early Stopping	(A) Terminates training when validation loss stops improving
(5) Weight Clipping	(B) Constrains weight magnitudes to remain within a fixed range

(c): Training-only techniques:

- **Dropout**: only active during training; units are randomly deactivated to simulate ensembling.
- **Early Stopping**: stops training based on validation performance but does not modify the final model architecture.
- **Weight Clipping**: constraints applied during training updates.
- **L1 and L2 Regularization**: influence the final learned weights that remain in the model during inference.

4. (10 points) Linear regression is a foundational model in machine learning, often introduced as the simplest method for fitting a line (or hyperplane) to data. Logistic regression extends this idea to classification tasks, but their behavior differs in important ways.
   (a) (2 points) Write the general formula for a linear regression model. Clearly explain the role of each component in terms of inputs, parameters, and outputs (vectorized form).
   (b) (2 points) Although linear regression is designed for regression problems, it can be adapted for binary classification. Describe how this can be done, and highlight a potential drawback of using linear regression in classification tasks.
   (c) (3 points) Provide the formula for logistic regression. Explain how applying the sigmoid function addresses the limitations of linear regression in classification, particularly in handling non-linearly separable data.
   (d) (3 points) Consider a binary classification dataset where the two classes are not linearly separable. Sketch or plot the behavior of a linear regression decision boundary versus that of logistic regression. What differences emerge, and why does one perform better than the other?

Solution:

(a): The general formula for linear regression is:

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{w}^\top \mathbf{x} + b$$

Where:

- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the input feature vector.
- $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ is the weight vector.
- b (or w_0) is the bias or intercept term.
- $\hat{y}$ is the predicted continuous output.

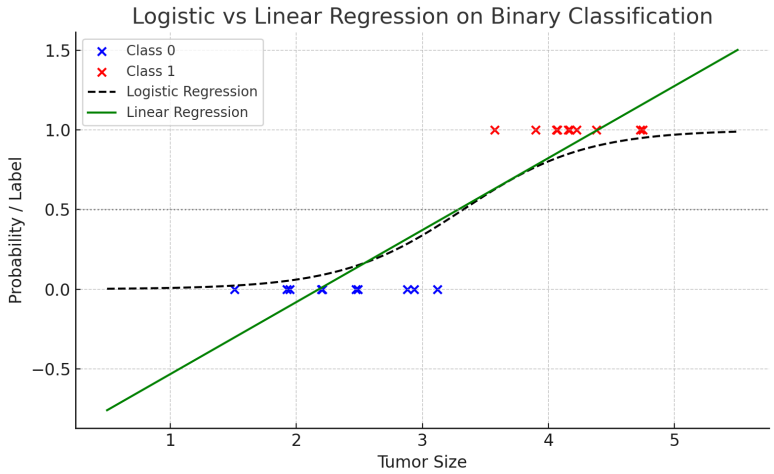
(b): Linear regression can be used for binary classification by thresholding the output. For example, if $\hat{y} > 0.5$, predict class 1; otherwise, predict class 0. However, this approach has a major limitation: predictions are not constrained to $[0, 1]$, so the model may output values that are not interpretable as probabilities. It is also not well-suited for data where the classes are not linearly separable.

(c): Logistic regression models the probability of class membership using the sigmoid function applied to a linear combination of inputs:

$$\hat{p} = \sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}$$

This ensures that $\hat{p} \in (0, 1)$, making it suitable for probability interpretation. The sigmoid "squashes" large positive and negative values, introducing non-linearity and better handling uncertainty near decision boundaries.

(d): In datasets where classes overlap (e.g., two Gaussian blobs with noise), linear regression may produce a boundary that misclassifies many points or produces extreme output values. Logistic regression, by modeling probability and learning a probabilistic boundary, performs better in these cases. Below is an illustrative comparison:



5. (10 points) Weight initialization plays a crucial role in the successful training of deep neural networks by influencing gradient flow and convergence behavior.
   (a) (2 points) What is weight initialization, and why is it considered a critical step before training a neural network?
   (b) (2 points) Consider initializing all weights in a neural network to a constant zero value. Explain what happens during forward and backward propagation and why this is problematic.
   (c) (2 points) What does it mean to initialize weights using a standard normal distribution? Why is this often insufficient for deep networks without additional scaling?
   (d) (4 points) The formula below corresponds to a widely used weight initialization technique known as He Initialization [**?**] (also called Kaiming Initialization):

$$W \sim \mathcal{N}\left(0, \frac{2}{\texttt{fan\_in}}\right)$$

   Here, `fan_in` refers to the number of input connections (i.e., incoming units) to a neuron in a given layer. This initialization is particularly effective when used with ReLU activations. Answer the following:

   - Why does this variance help stabilize the training process with ReLU activations?
   - What might go wrong if the variance were significantly higher or lower than $\frac{2}{\text{fan\_in}}$?

Solution:

(a): Weight initialization refers to the process of setting the initial values of the model's weights before training begins. It is crucial because poor initialization can lead to vanishing/exploding gradients, slow convergence, or convergence to poor local minima.

(b): If all weights are initialized to zero, every neuron in a given layer will receive the same gradient during backpropagation, leading them to update identically. This symmetry prevents the network from learning diverse representations, essentially making each neuron redundant and defeating the purpose of having multiple units.

(c): Initializing weights using a standard normal distribution $\mathcal{N}(0, 1)$ means each weight is drawn from a Gaussian distribution with mean 0 and variance 1. While this introduces randomness and breaks symmetry, it does not account for the number of inputs to a neuron. In deeper networks, this can cause the variance of activations to explode or vanish as they propagate through layers, harming learning stability.

(d): He Initialization (or Kaiming Initialization) is designed to preserve the variance of activations throughout layers when using ReLU. Since ReLU zeroes out half of the input values (negative ones), we compensate by scaling the variance of the weights by $\frac{2}{\text{fan\_in}}$ to maintain the signal flow.

If the variance is set **too high**, activations can explode across layers, leading to gradient explosion and numerical instability. If the variance is **too low**, the activations diminish layer by layer, leading to vanishing gradients and poor learning.

Name: _____

Neptun: _____

Deep Network Development
Exam 8
August 18, 2026

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

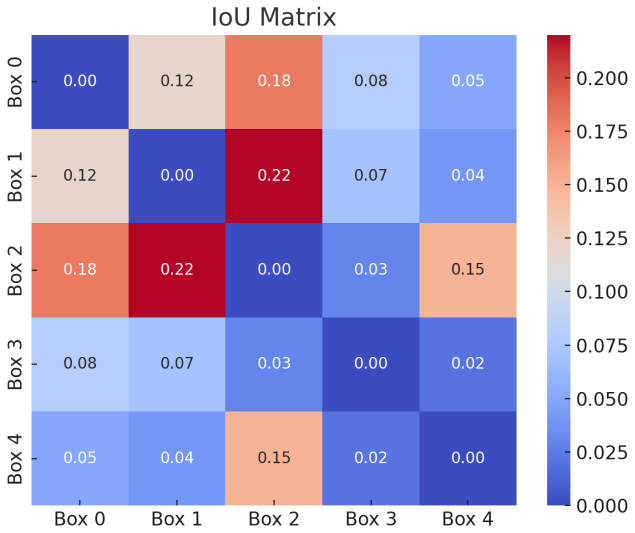
This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

We wish you the best of luck.

Exam Tasks

1. (10 points) Non-Maximum Suppression (NMS) is a crucial post-processing step in object detection to eliminate redundant bounding boxes.
   (a) (3 points) Briefly explain how the Intersection over Union is computed between two bounding boxes. Use a sketch or describe the formula if needed.
   (b) (7 points) The figure below shows five predicted bounding boxes, along with their associated confidence scores and a table of pairwise IoU values. Given an IoU threshold of 0.2, apply Non-Maximum Suppression to determine which bounding boxes should be retained. Clearly state the steps taken and justify your final selection.

Box 0	**Box 1**	**Box 2**	**Box 3**	**Box 4**
0.65	0.72	0.81	0.91	0.56



Solution:

(a): The Intersection over Union (IoU) is a metric used to evaluate the overlap between two bounding boxes. It is defined as:

$$\text{IoU}(A, B) = \frac{\text{Area of } A \cap B}{\text{Area of } A \cup B}$$

This value ranges from 0 (no overlap) to 1 (perfect overlap). IoU is commonly used to determine whether two detections refer to the same object.

(b): Answer:

1. Sort boxes by descending score: [Box 3, Box 2, Box 1, Box 0, Box 4]
2. Select Box 3 (0.91). Compare IoUs with:
   - Box 0: 0.32 → discard (IoU > 0.2)
   - Box 1: 0.10 → keep
   - Box 2: 0.15 → keep
   - Box 4: 0.05 → keep

   Remaining: [Box 3, Box 2, Box 1, Box 4]
3. Select Box 2 (0.81). Compare with:
   - Box 1: 0.35 → discard
   - Box 4: 0.18 → keep

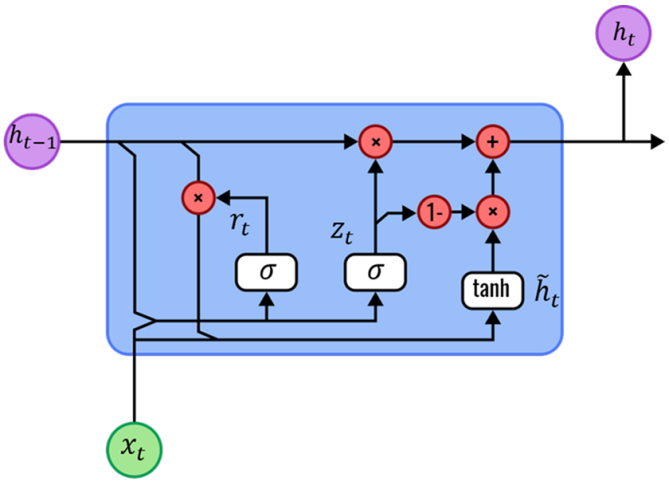
   Remaining: [Box 3, Box 2, Box 4]

Final retained boxes: Box 3, Box 2, Box 4

2. (10 points) GRUs are a simplified type of recurrent neural network architecture that aim to effectively capture long-term dependencies in sequential data while using fewer parameters than LSTMs.
   (a) (3 points) Illustrate the architecture of a basic GRU cell (simple diagram).
   (b) (4 points) Label and briefly describe the function of each core component in the GRU cell.
   (c) (3 points) Explain how the GRU architecture addresses the vanishing gradient problem and in what ways it improves upon vanilla RNNs.

Solution:

(a): GRU cell visualized:



(b): A basic GRU cell includes the following components:

- **Update gate** (z_t): Determines how much of the previous hidden state should be carried over to the current step.

- **Reset gate** (r_t): Controls how much of the previous hidden state should be forgotten.
- **Candidate hidden state** ($\tilde{h}_t$): A new candidate for the hidden state, computed using the reset gate.
- **Final hidden state** (h_t): A linear interpolation between the previous hidden state and the candidate state, guided by the update gate.

(c): Vanilla RNNs suffer from vanishing gradients due to repeated multiplications in deep time sequences. GRUs mitigate this by using gating mechanisms that control information flow, allowing gradients to propagate more effectively across long sequences. Unlike LSTMs, GRUs merge the cell state and hidden state into one and use fewer gates, making them computationally efficient while still effective at preserving memory over time.

3. (10 points) Pooling operations are essential components in convolutional neural networks, used to reduce spatial dimensions while preserving important features.
   (a) (3 points) What is max pooling and how does it differ from average pooling? Why is it used in convolutional neural networks?
   (b) (3 points) List the key hyperparameters of a pooling layer. Explain how they influence the size and behavior of the output feature map.
   (c) (4 points) Given the input matrix below, perform 2D max pooling with $k = 2\times2$, $s = 2$, and $p = 0$. Show your sliding window positions and intermediate steps clearly.

$$\text{Input: } \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 6 & 1 & 2 \\ 0 & 1 & 3 & 1 \\ 4 & 2 & 0 & 6 \end{bmatrix}$$

Solution:

(a): Max pooling is a downsampling operation that selects the maximum value within each subregion of the input. Average pooling, in contrast, computes the average of values within the patch. Max pooling helps preserve prominent features while reducing spatial dimensions.

(b): Key hyperparameters of pooling layers:

- **Kernel size**: The size of the patch the pooling operation examines (e.g., 2×2).
- **Stride**: How many steps the kernel moves across the input matrix. A larger stride results in greater spatial reduction.
- **Padding**: Optionally adds borders to the input. With no padding, pooling is only done on full patches.

(c): Apply max pooling with 2×2 kernel and stride 2:

$$\text{Input: } \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 6 & 1 & 2 \\ 0 & 1 & 3 & 1 \\ 4 & 2 & 0 & 6 \end{bmatrix} \quad \Rightarrow \quad \text{Max values from each } 2 \times 2 \text{ region:}$$

- Region 1: $\max(1, 3, 5, 6) = \boxed{6}$
- Region 2: $\max(2, 4, 1, 2) = \boxed{4}$

- Region 3: $\max(0, 1, 4, 2) = \boxed{4}$
- Region 4: $\max(3, 1, 0, 6) = \boxed{6}$

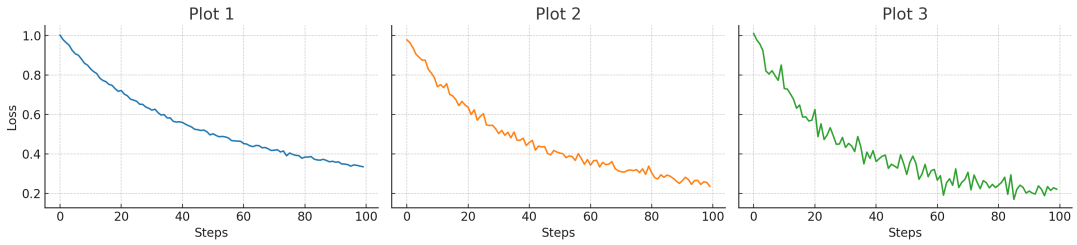
Final output feature map: $\begin{bmatrix} 6 & 4 \\ 4 & 6 \end{bmatrix}$

4. (10 points) Stochastic Gradient Descent is a widely used optimization technique in deep learning, particularly effective when the loss function is differentiable.

   (a) (2 points) What does the term *stochastic* refer to in the context of gradient descent? How does it differ from batch gradient descent?

   (b) (3 points) What is the goal of gradient descent? List and briefly describe two important hyperparameters that affect its behavior.

   (c) (5 points) The three training curves shown below visualize the behavior of the loss over training steps under different batch sizes. Assume the same model, optimizer type, and learning rate were used.



(A) Large Batch (B) Small Batch (C) Medium Batch

Match each plot to the correct batch size.

Write your matchings clearly (e.g., Plot 1 → A, etc.). Justify your answer by describing what characteristics in the curves helped you identify the batch size used.

Solution:

(a): The term stochastic in stochastic gradient descent refers to the use of random subsets (often of size 1) of the training data to compute each gradient update. Unlike batch gradient descent, which computes gradients using the full dataset at each step, SGD introduces randomness and more frequent updates, allowing it to escape local minima and saddle points more effectively, albeit with noisier convergence.

(b): The goal of gradient descent is to minimize the loss function by updating the model's parameters in the direction of the negative gradient. Two important hyperparameters are:

- **Learning rate:** Controls the step size during parameter updates. A value too large may cause divergence, while a value too small may slow convergence.
- **Batch size:** Determines how many samples are used to compute the gradient. Smaller batches introduce noise but allow more frequent updates, while larger batches reduce variance but require more memory.

(c): Matching and justification:

- **Plot 1 → B (Small Batch Size)**: Shows highly noisy updates with significant fluctuation in loss, characteristic of stochastic behavior with low batch sizes.
- **Plot 2 → C (Medium Batch Size)**: Displays moderately noisy updates with a balance between stability and variability.
- **Plot 3 → A (Large Batch Size)**: Demonstrates smooth and stable convergence, typical of full-batch training where gradients are averaged across many samples.

5. (10 points) Convolutional Neural Networks (CNNs) utilize the convolution operation to process visual input efficiently. This operation exhibits important structural properties that influence the behavior of CNNs.
   (a) (3 points) Define translation invariance in the context of image classification. Is the convolution operation inherently translation invariant? Justify your answer.
   (b) (3 points) What is translation equivariance? Is the convolution operation translation equivariant? Provide a brief explanation.
   (c) (2 points) Suppose a CNN perfectly classifies a digit at the center of a 28×28 image. What might happen if you shift the digit 4 pixels to the left? Explain using the concepts of equivariance and invariance.
   (d) (2 points) Which of the following statements correctly distinguishes equivariance from invariance in the context of CNNs? Circle the correct answers.
      1. Equivariance implies that applying a transformation to the input results in a corresponding transformation of the output.
      2. Invariance implies that any transformation applied to the input yields an identical output.
      3. Equivariance ensures the output is completely unchanged when the input is transformed.
      4. Invariance means the model explicitly learns how to counteract transformations through convolutional filters.

Solution:

(a): Translation invariance means that when the input is shifted, the output remains unchanged. The convolution operation itself is **not** inherently translation invariant. Instead, convolution is translation equivariant, which means the feature maps shift in the same way as the input. True invariance requires additional mechanisms (e.g., pooling, data augmentation, or positional encoding).

(b): Translation equivariance means that a shift in the input causes an equivalent shift in the output. Convolutions satisfy this property, making them well-suited for detecting local patterns consistently across spatial locations. This property helps the CNN detect features like edges or corners in various parts of the image.

(c): If a CNN is trained on digits centered in the image, shifting a digit to the left may reduce accuracy. This is because although convolution layers are translation equivariant, the network as a whole (especially with fully connected layers at the end) may not be invariant. Without appropriate translation-invariant mechanisms (e.g., global pooling), classification performance may degrade when input location changes.

(d): Correct answers: **(1, 2)**

Name: _____

Neptun: _____

Deep Network Development
Exam 7
August 18, 2026

You have exactly **60 minutes** to complete this written examination. The exam consists of **5 questions**, each carrying **10 points**, for a total of **50 points**. A minimum of **25 points** is required to achieve a passing grade.

This is a **pass/fail assessment**. Answers must be clear, concise, and directly address the question. Irrelevant or excessively verbose responses may result in point deductions.

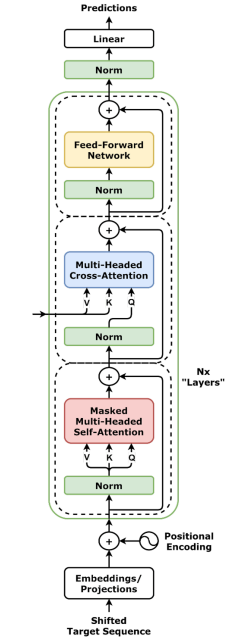
We wish you the best of luck.

Exam Tasks

1. (10 points) The Transformer architecture revolutionized the processing of sequential data and the design of modern AI models. The original model introduced in [1] consists of an encoder–decoder structure.
   (a) (3 points) Draw a simplified diagram of the Transformer decoder architecture.
   (b) (5 points) Label the key components of the decoder and briefly describe their roles.
   (c) (2 points) What is the output of the Transformer decoder, and how is it typically used during training and inference?

Solution:

(a): The decoder architecture visualized:



Transformer, full architecture.png
https://commons.wikimedia.org/wiki/File:Transformer,_full_architecture.png
dvgodoy, CC BY 4.0
<https://creativecommons.org/licenses/by/4.0>, via
Wikimedia Commons

(b): The Transformer decoder includes the following major components:

- **Input Embedding:** Converts target sequence tokens into dense vector representations.
- **Positional Encoding:** Injects information about token positions into the embeddings.
- **Masked Multi-Head Self-Attention:** Allows the decoder to attend only to previous positions in the output sequence (causal masking), enabling autoregressive generation.

- **Encoder–Decoder Attention:** Attends to the encoder's outputs, allowing the decoder to incorporate information from the input sequence.
- **Feedforward Neural Network (FFN):** A fully connected non-linear layer applied to each position.
- **Residual Connections and Layer Normalization:** Stabilize training and allow gradients to flow effectively through the network.

(c): The output of the Transformer decoder is a sequence of hidden vectors corresponding to the target sequence tokens. These are passed through a final linear projection layer followed by a softmax to produce a probability distribution over the vocabulary. During training, the decoder learns to predict the next token in the sequence; during inference, it autoregressively generates tokens one by one using its own past outputs and the encoder's representations.

2. (10 points) You are working as a junior AI engineer at an agricultural tech company. Your first task is to analyze drone footage taken above vineyards to estimate the number of birds landing in the field. However, the footage is unstable due to wind and drone vibrations, and birds are small, fast, and often land in groups.

   (a) (5 points) You are given the following possible solutions. For each, briefly describe how the pipeline works and whether you think it is a good solution for counting the birds. Justify your reasoning.

   1. Stabilize the video using frame alignment → Apply object detection on each frame → Count each new detection as a bird.
   2. Apply object detection on the raw shaky video → Count the number of detections across all frames.
   3. Use background subtraction to detect movement → Count the number of new moving blobs as birds.
   4. Convert video to frames → Apply image classification to each frame to predict whether "bird is present" or "no bird" → Count frames where a bird is predicted.

   (b) (2 points) Why is video stabilization helpful in computer vision pipelines that involve motion or detection?

   (c) (3 points) You decide to use an object detection model to identify birds in each frame of the video. However, you observe many false detections and inconsistent predictions. Which of the following is the most likely cause of this instability?

   - (A) The object detector was trained only on ground-level bird images, not aerial views.
   - (B) The model is using too small a batch size during training.
   - (C) The input resolution is very high, making detections too slow.
   - (D) The detection threshold is set too high, leading to fewer detections.

Solution:

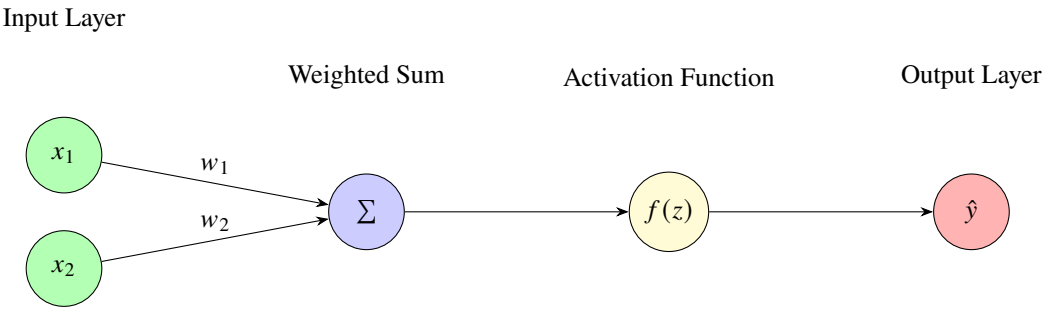
(a): Answers

1. **Valid.** Stabilizing the video helps eliminate camera-induced motion, making detection more accurate. Counting new detections per frame works if objects are not repeatedly detected or miscounted.
2. **Not ideal.** Without stabilization, the camera shake may cause false detections or repeated counting of the same birds across frames.
3. **Unreliable.** Background subtraction assumes a fixed background. Drone footage is not static, so any movement in the background may be mistakenly counted as a bird.
4. **Weak.** Classification detects the presence of a bird but cannot count how many. It gives no spatial or object-level information, so this method is not suited for precise counting.

(b): Stabilization improves detection and motion analysis by removing artificial movements caused by the camera itself. This helps focus only on the motion of actual objects (e.g., birds), leading to more reliable detection.

(c): The correct answer is **(A)**. If the object detection model was trained on images of birds taken from the ground, it may struggle to detect birds in aerial footage due to changes in scale, angle, and background. This domain shift can lead to false positives or missed detections. Other options are either unrelated (B, C) or would reduce detections rather than cause instability (D).

3. (10 points) Backpropagation is an algorithm used to update the weights of a neural network by computing gradients of the loss function with respect to each parameter. The following diagram shows a simple single-layer neural network with two inputs and one output:



You are given the following:

- Inputs: $x_1 = 2.0, x_2 = -1.0$
- Weights: $w_1 = -0.4$, $w_2 = 0.5$
- Bias: $b = 0$
- Activation function: Identity ($f(z) = z$)
- Target output: $y = 2.0$
- Learning rate: $\eta = 0.1$

Compute one weight update according to the following steps:

(a) (2 points) Perform the forward pass and compute the prediction $\hat{y}$.

(b) (3 points) Compute the loss using Mean Squared Error: $L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

(c) (3 points) Compute the gradient of the loss with respect to w_1 and w_2.

(d) (2 points) Update w_1 and w_2 using gradient descent.

Solution:

(a): Forward Pass

We compute the linear output:

$$z = x_1 w_1 + x_2 w_2 + b = 2.0 \cdot (-0.4) + (-1.0) \cdot 0.5 + 0 = -0.8 - 0.5 = -1.3$$

$$\hat{y} = f(z) = z = -1.3$$

(b): Compute Loss

Using MSE:

$$L = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(2.0 - (-1.3))^2 = \frac{1}{2}(3.3)^2 = \frac{1}{2} \cdot 10.89 = 5.445$$

(c): Compute Gradients

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = -1.3 - 2.0 = -3.3$$

$$\frac{\partial L}{\partial w_1} = -3.3 \cdot x_1 = -3.3 \cdot 2.0 = -6.6$$

$$\frac{\partial L}{\partial w_2} = -3.3 \cdot x_2 = -3.3 \cdot (-1.0) = 3.3$$

(d): Weight Update

$$w_1 \leftarrow w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = -0.4 - 0.1 \cdot (-6.6) = -0.4 + 0.66 = \boxed{0.26}$$

$$w_2 \leftarrow w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = 0.5 - 0.1 \cdot 3.3 = 0.5 - 0.33 = \boxed{0.17}$$

Final Answer:

$$w_1 = 0.26, \quad w_2 = 0.17$$

4. (10 points) You are training a PyTorch neural network for an image classification task with 10 output classes. The model outputs raw logits. You have already defined your model, optimizer, and loss function (`criterion`), and placed the model on the appropriate device (`device`). The dataset and data loaders (`train_loader`, `valid_loader`) are fully implemented.

   Below is a partially completed training and evaluation loop. Fill in the 10 missing lines (marked with `# (1)` to `# (10)`) to complete a working PyTorch training setup. Assume all standard PyTorch conventions and imports are available.

```
epochs = 100
# (1)
for epoch in range(epochs):
    # (2)
    # (3)
    for batch in train_loader:
        inputs, labels = batch
        # (4)
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        # (5)
        # (6)

    # (7)
    correct = 0
    total = 0
    with torch.no_grad():
        for batch in valid_loader:
            # (8)
            labels = labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)
```

```
            total += labels.size(0)
            # (9)
    # (10)
```

Solution:

```
epochs = 100
for epoch in range(epochs):
    model.train()
    optimizer.zero_grad()
    for batch in train_loader:
        inputs, labels = batch
        optimizer.zero_grad()
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for batch in valid_loader:
            inputs, labels = batch
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    print(f"Epoch {epoch+1}, Accuracy: {100 * correct / total:.2f}%")
```

5. (10 points) You are given a trained feedforward neural network with the following architecture:

   - Input layer with 2 features
   - Two hidden layers with 100 neurons each
   - Output layer with 1 neuron
   - All hidden layers use the ReLU activation function
   - The final layer uses the Sigmoid activation

   The model was trained on a 2D binary classification task with classes distributed in a ring shape (like a donut), where class 0 occupies the center, and class 1 surrounds it in a circular region.

   (a) (3 points) You observe that the model only learns to predict class 1 everywhere, with output probabilities close to 1 regardless of input. What could be the most likely reason? Provide a hypothesis based on model architecture or initialization.

   (b) (3 points) Explain why ReLU activation functions may lead to such behavior in deeper networks, especially in low-dimensional input settings.

   (c) (4 points) Suggest and justify one architectural or training changes that could help this model learn the circular decision boundary properly.

Solution:

(a): The most likely cause is dead neurons due to ReLU. With low-dimensional input and poor weight initialization, it's possible that the ReLU layers output only zeros for a large portion of the training data, especially early in training. This prevents gradient flow and can lead to a degenerate model predicting only one class (in this case, class 1 due to Sigmoid bias).

(b): ReLU outputs zero for all negative inputs. In deep networks, if the activations at any layer fall entirely into the negative range, the outputs become zero. If this happens during initialization or early updates, gradients stop flowing back through those units. With 2D inputs and wide hidden layers, most neurons may die early and never recover, especially if all input directions are compressed into zero-activation paths.

(c): Two improvements:

- **Use Leaky ReLU or GELU instead of ReLU:** These activations allow a small gradient to flow even for negative inputs, reducing the risk of dead neurons.
- **Add a batch normalization layer before each ReLU:** This stabilizes activation distributions and helps prevent them from collapsing into the negative regime, improving gradient flow.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.0473

Bochkovskii, A., Delaunoy, A., Germain, H., Santos, M., Zhou, Y., Richter, S. R., & Koltun, V. (2024). Depth Pro: Sharp monocular metric depth in less than a second. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2410.02073

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2005.12872

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1409.1259

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2010.11929

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1406.2283

Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., van der Smagt, P., Cremers, D., & Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1504.06852

Fukushima, K. (1980). Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4), 193–202. https://doi.org/10.1007/bf00344251

Girshick, R. (2015). Fast R-CNN. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1504.08083

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 580–587. https://doi.org/10.1109/cvpr.2014.81

Godard, C., Mac Aodha, O., & Brostow, G. J. (2016). Unsupervised monocular depth estimation with left-right consistency. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1609.03677

Guillemot, C., & Le Meur, O. (2014). Image inpainting: Overview and recent advances. IEEE Signal Processing Magazine, 31(1), 127–144. https://doi.org/10.1109/msp.2013.2273004

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. https://doi.org/10.1109/cvpr.2016.90

References

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1704.04861

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K. C., Qin, H., Dai, J., & Li, H. (2022). FlowFormer: A Transformer Architecture for Optical Flow. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2203.16194

Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C., & Schindler, K. (2023). Repurposing diffusion-based image generators for monocular depth estimation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2312.02145

Kerbl, B., Kopanas, G., Leimkuehler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics, 42(4), 1–14. https://doi.org/10.1145/3592433

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Weinberger (Eds.), Advances in Neural Information Processing Systems (Vol. 25). Curran Associates, Inc.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

Lin, G., Milan, A., Shen, C., & Reid, I. (2016). RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1611.06612

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (Vol. 9905, pp. 21–37). Springer International Publishing. https://doi.org/10.1007/978-3-319-46448-0_2

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical vision Transformer using shifted windows. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2103.14030

Long, J., Shelhamer, E., & Darrell, T. (2014). Fully convolutional networks for semantic segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1411.4038

Luo, J., Cheng, J., Tang, X., Zhang, Q., Xue, B., & Fan, R. (2025). MambaFlow: A novel and flow-guided state space model for scene flow estimation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2502.16907

References

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.1508.04025

Mitsuhashi, N., Fujieda, K., Tamura, T., Kawamoto, S., Takagi, T., & Okubo, K. (2009). BodyParts3D: 3D structure database for anatomical concepts. Nucleic Acids Research, 37(Database issue), D782–D785. https://doi.org/10.1093/nar/gkn613

Morris, M. R., Sohl-Dickstein, J., Fiedel, N., Warkentin, T., Dafoe, A., Faust, A., Farabet, C., & Legg, S. (2024). Position: Levels of AGI for operationalizing progress on the path to AGI. Proceedings of the 41st International Conference on Machine Learning, PMLR 235:36308–36321. https://proceedings.mlr.press/v235/morris24b.html

Quan, W., Chen, J., Liu, Y., Yan, D.-M., & Wonka, P. (2024). Deep learning-based image and video inpainting: A survey. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2401.03395

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. In arXiv [cs.CV]. arXiv. http://arxiv.org/abs/2204.06125

Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision Transformers for dense prediction. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2103.13413

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., & Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2408.00714

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1506.01497

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2112.10752

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.1505.04597

Ruthotto, L., & Haber, E. (2021). An introduction to deep generative modeling. In arXiv [cs.LG]. arXiv. https://doi.org/10.48550/arXiv.2103.05180

Shi, X., Huang, Z., Bian, W., Li, D., Zhang, M., Cheung, K. C., See, S., Qin, H., Dai, J., & Li, H. (2023). VideoFlow: Exploiting temporal cues for multi-frame optical flow estimation. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2303.08340

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations, ICLR 2015. https://arxiv.org/abs/1409.1556

References

Song, S., Lichtenberg, S. P., & Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 567–576. https://doi.org/10.1109/cvpr.2015.7298655

Structure from Motion. (n.d.). Retrieved November 17, 2025, from https://cmsc426.github.io/sfm/

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1–9. https://doi.org/10.1109/cvpr.2015.7298594

Teed, Z., & Deng, J. (2020). RAFT: Recurrent all-pairs field transforms for optical flow. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2003.12039

Tu, T., Azizi, S., Driess, D., Schaekermann, M., Amin, M., Chang, P.-C., Carroll, A., Lau, C., Tanno, R., Ktena, I., Mustafa, B., Chowdhery, A., Liu, Y., Kornblith, S., Fleet, D., Mansfield, P., Prakash, S., Wong, R., Virmani, S., … Natarajan, V. (2023). Towards generalist biomedical AI. In arXiv [cs.CL]. arXiv. https://doi.org/10.48550/arXiv.2307.14334

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762

Vennerød, C. B., Kjærran, A., & Bugge, E. S. (2021). Long short-term memory RNN. In arXiv [cs.LG]. arXiv. https://doi.org/10.48550/arXiv.2105.06756

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). YOLOv10: Real-time end-to-end object detection. In arXiv [cs.CV]. arXiv. http://arxiv.org/abs/2405.14458

Wang, T., Pang, J., & Lin, D. (2022). Monocular 3D object detection with depth from motion. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2207.12988

Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., & Zhao, H. (2024). Depth Anything V2. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2406.09414

Yeh, F. C., Panesar, S., Fernandes, D., Meola, A., Yoshino, M., Fernandez-Miranda, J. C., … & Verstynen, T. (2018). Population-averaged atlas of the macroscale human structural connectome and its network topology. NeuroImage, 178, 57–68.

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F. E. H., Feng, J., & Yan, S. (2021). Tokens-to-Token ViT: Training vision transformers from scratch on ImageNet. In arXiv [cs.CV]. arXiv. https://doi.org/10.48550/arXiv.2101.11986