wbs Warwick Business School

Data Science & Generative AI

Dr Michael Mortenson

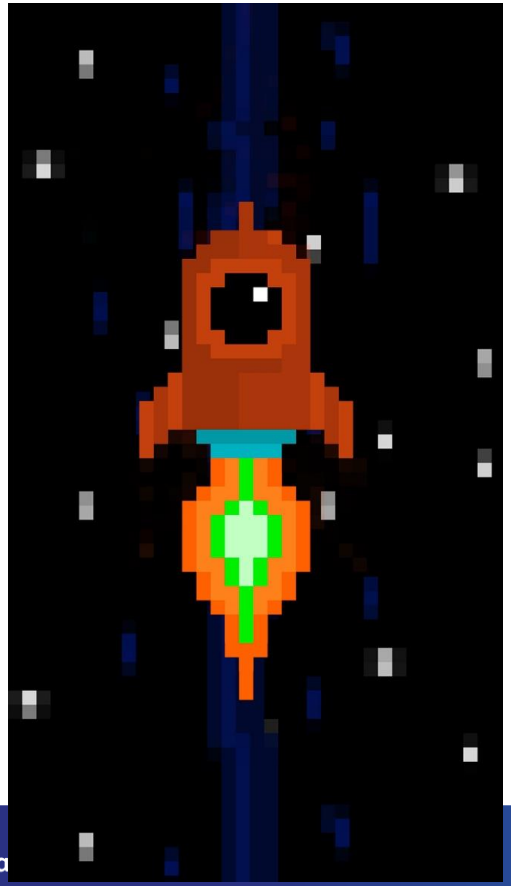
Associate Professor (Reader)
michael.mortenson@wbs.ac.uk

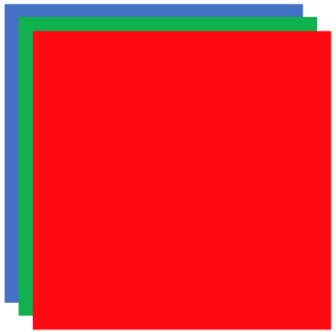
Session 8: Transformers – From T to G via P

1.1 The Layer Cake

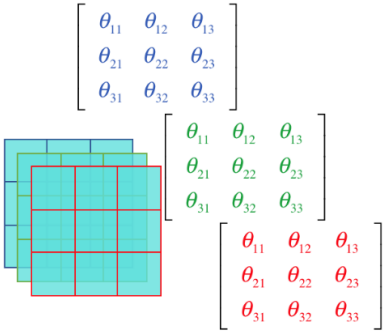
- Linear layers (based on linear regression)
- Normalisation layers (batch or layer … normalise between layers)
- Dropout layers (turn off paths to prevent overfitting)
- Convolution layers (learn filters to find shapes in the input)
- Pooling layers (reduce the size of the previous output)
- Recurrent layers (keep a memory vector, the hidden state, of previous inputs)
- Long Short Term Memory (LSTM) layers. A modification of recurrent layers to also add a “forget” mechanism.
- Self-attention layers.

1.2 Image Data





Color image



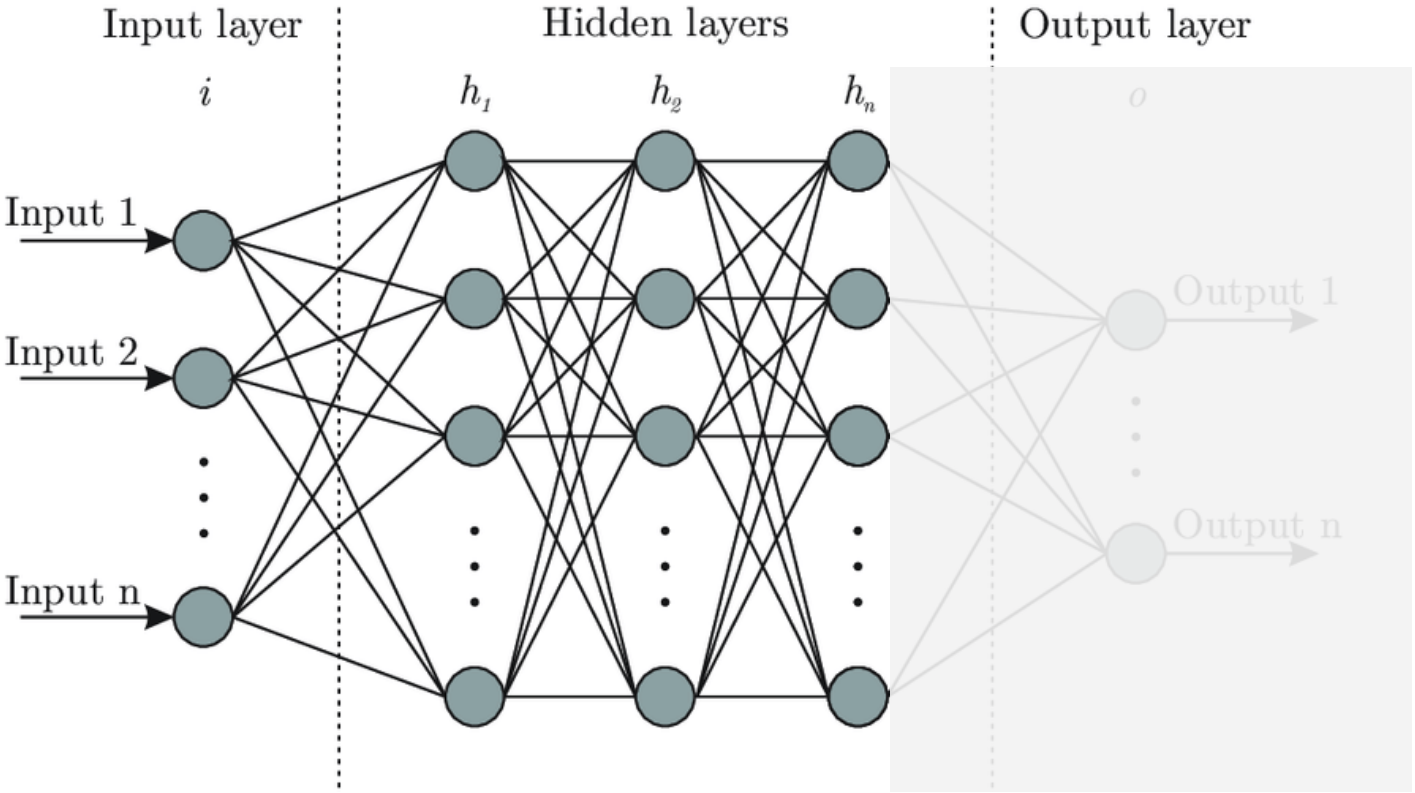
Parameterized filter

INPUT LAYER (RGB IMAGE)

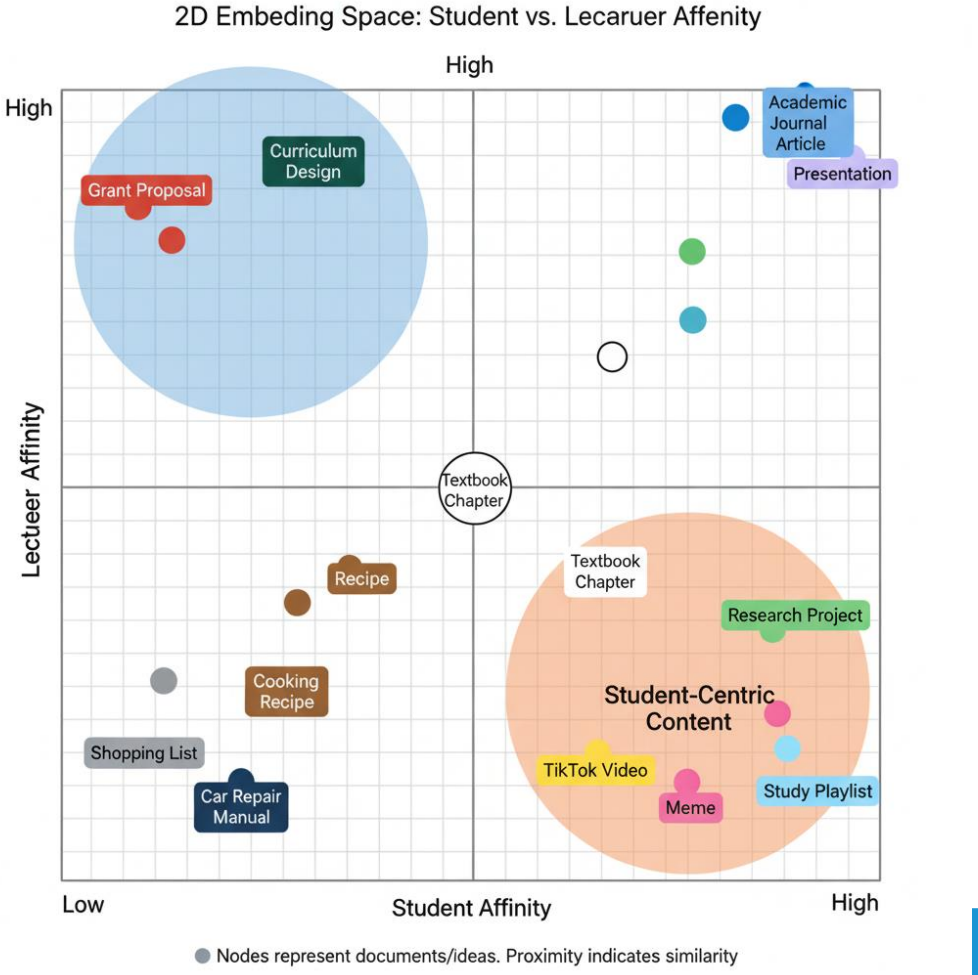
1. # of pixels width
2. # of pixels height
3. # of channels (i.e. colours)

2095 x 3725 x 3

1.3 Text Data (Transformers Approach)



1.3 Text Data


2D Embeding Space: Student vs. Lecaruer Affenity
High
High
Lectueer Affinity
Curriculum Design
Grant Proposal
Academic Journal Article
Presentation
Textbook Chapter
Recipe
Textbook Chapter
Research Project
Cooking Recipe
Student-Centric Content
Shopping List
TikTok Video
Study Playlist
Car Repair Manual
Meme
Low
Student Affinity
High
Nodes represent documents/ideas. Proximity indicates similarity

1.3 Text Data (Transformers Approach)

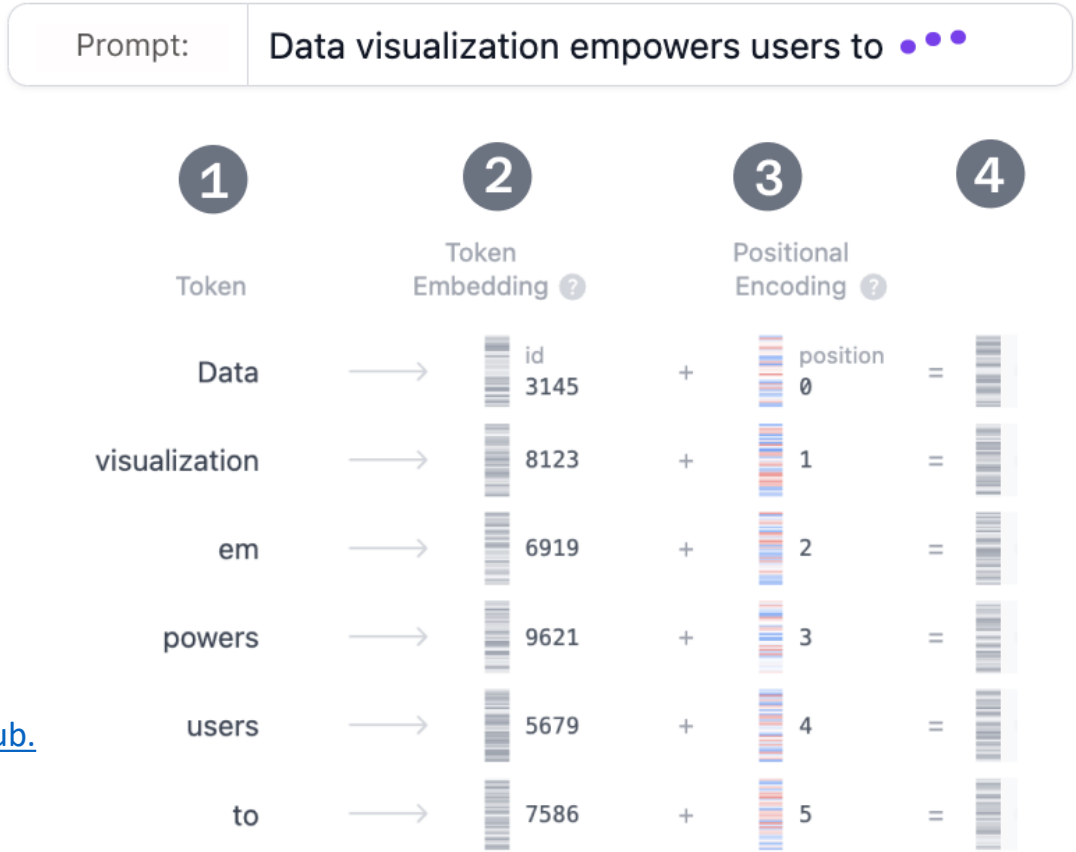


Image Credit: https://poloclub.github.io/transformer-explainer/

1.4 What's in a Name?

Generative

Pretrained

Transformers

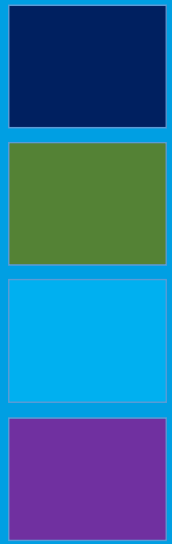
Session Aims

Introduction

Transformers

Pre-Training

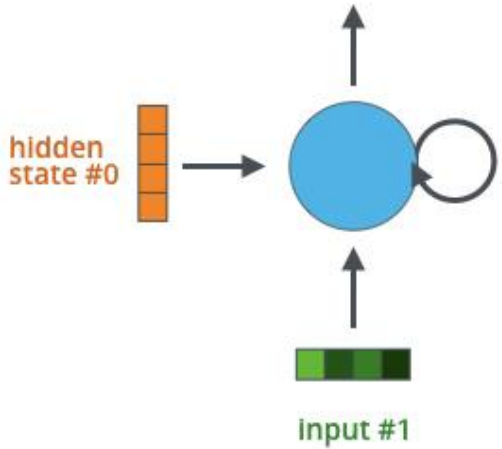
Generation



Recurrent Neural Network

Time step #1:

An RNN takes two input vectors:

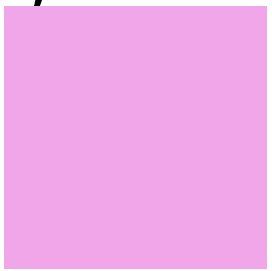


Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

2.1 On Memory and Motorbikes

2.2 Self-Attention Layers (by Metaphor)

the scary, blue monster hid under the child's pink bed



2.2 Self-Attention Layers (by Metaphor)

Query: *I am a noun looking for an adjective to add context to me*

the scary, blue monster hid under the child's pink bed

2.2 Self-Attention Layers (by Metaphor)

Key: *We are adjectives. We like to add context to nouns.*

the scary, blue monster hid under the child's pink bed

2.2 Self-Attention Layers (by Metaphor)

Semantically close

Positionally close

the scary, blue monster hid under the child's pink bed

Further away

2.2 Self-Attention Layers (by Metaphor)

Value: *How should I change the values of the embedding?*

the scary, blue monster hid under the child's pink bed

2.3 Attention by Convolution

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

Neilsen MA (2015). *Neural Networks and Deep Learning.* Determination Press.

2.4 Self-Attention Layers (by Matrices)

- Similarly to convolutional layers we learn to look for particular patterns. However, in convolution these are fixed filters we learn that we compare with the input. If the filter we learn matches the input (via dot product … multiplying them together), the result is an activation – the network has found something.
- In self-attention we learn a set of weights (like a filter), however in this case its three sets … *query* (W_Q), *key* (W_K) and *value* (W_V). Rather than using these directly to measure activation, instead we use them to compare different parts of the input.

2.4 Self-Attention Layers (by Matrices)

- Like a convolution layers, we scan through the input each time multiplying the word vector (semantic embedding and positional encoding - x_i) by the query weights (W_Q). This produces an output we call the *query projection* ($Q = XW_Q$) for the given input.
- For every other word (including the word itself) we also multiply those word vectors by the key weights (W_K), creating a set of *key projections* ($K = XW_K$).
- Rather than comparing filter with input, like we would for convolutions, we compare (via dot product similarity) the *query projection* (Q) with every other *key projection* (K). If they are similar, they produce a large number and represent an activation.

2.4 Self-Attention Layers (by Matrices)

- The calculations on the previous slide gives us an *attention score*, this is effectively a matrix of activations for every word (X) we have compared the query (Q) with. To these we apply the softmax algorithm. This exaggerates differences making big activations proportionally bigger, and reducing small activations towards 0.
- We also compute V, the *value projection* for every value in X ($V = XW_v$). By multiplying this with the *attention score*, we get the appropriate nudge to apply to the specific input word vector we have been evaluating (the output of our layer for that input).
- We would now move the scan on to the next input (x_{i+1}) and apply the same process again.

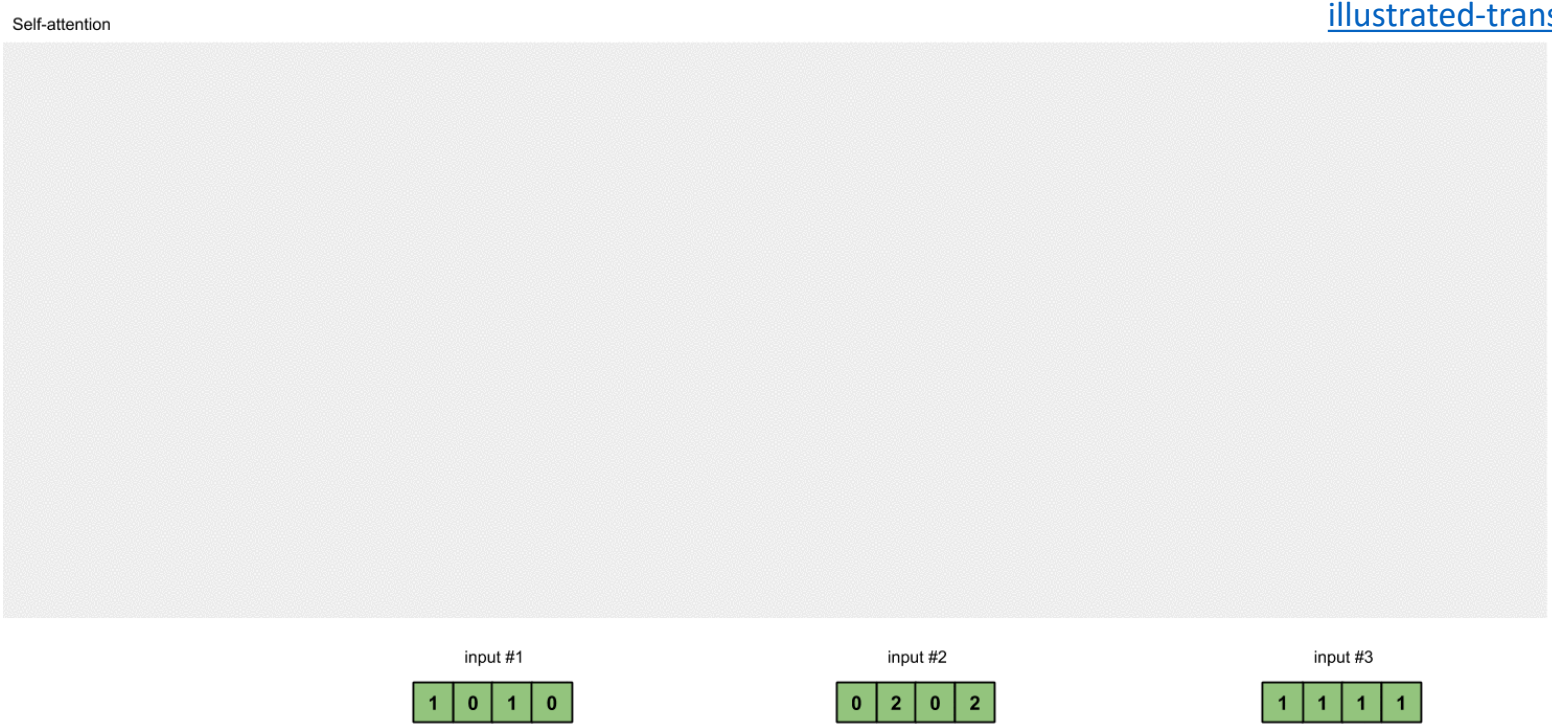
2.4 Self-Attention Layers (by Matrices)

- $Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

  where:
  - $Softmax$ changes the output to values that sum to 1. This also exaggerates differences so bigger activations look bigger.
  - QK^T is the query projection (Q) multiplied by the transpose of the key projection (K). The transpose just makes the matrix algebra work (as matrix algebra multiplies rows by columns).
  - $\sqrt{d_k}$ is the square root of the dimensionality of K - the *key projection* (XW_K). This just scales values so that the variance ≈ 1, and means we avoid different variances depending on the size of the input (X).

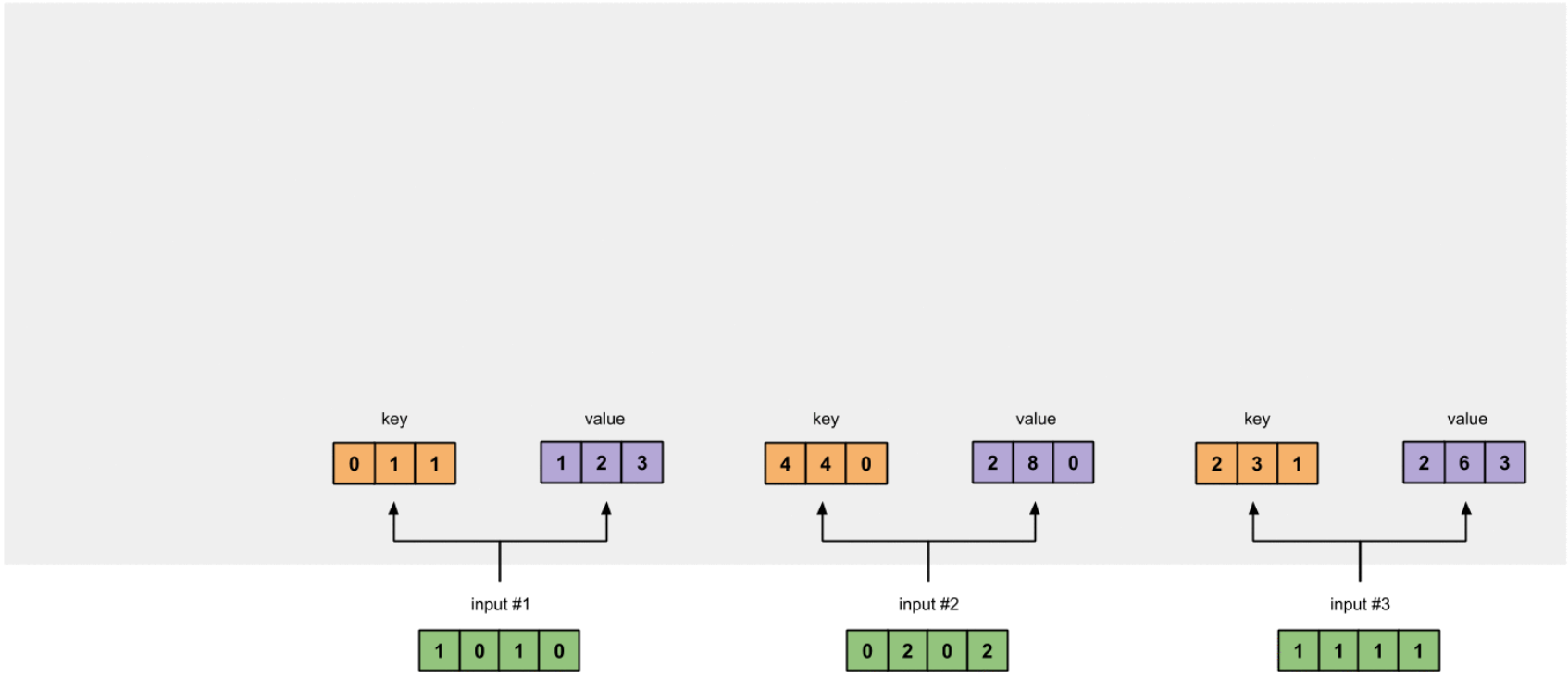
2.4 Self-Attention Layers (by Matrices)

Image Credits: https://jalammar.github.io/illustrated-transformer/



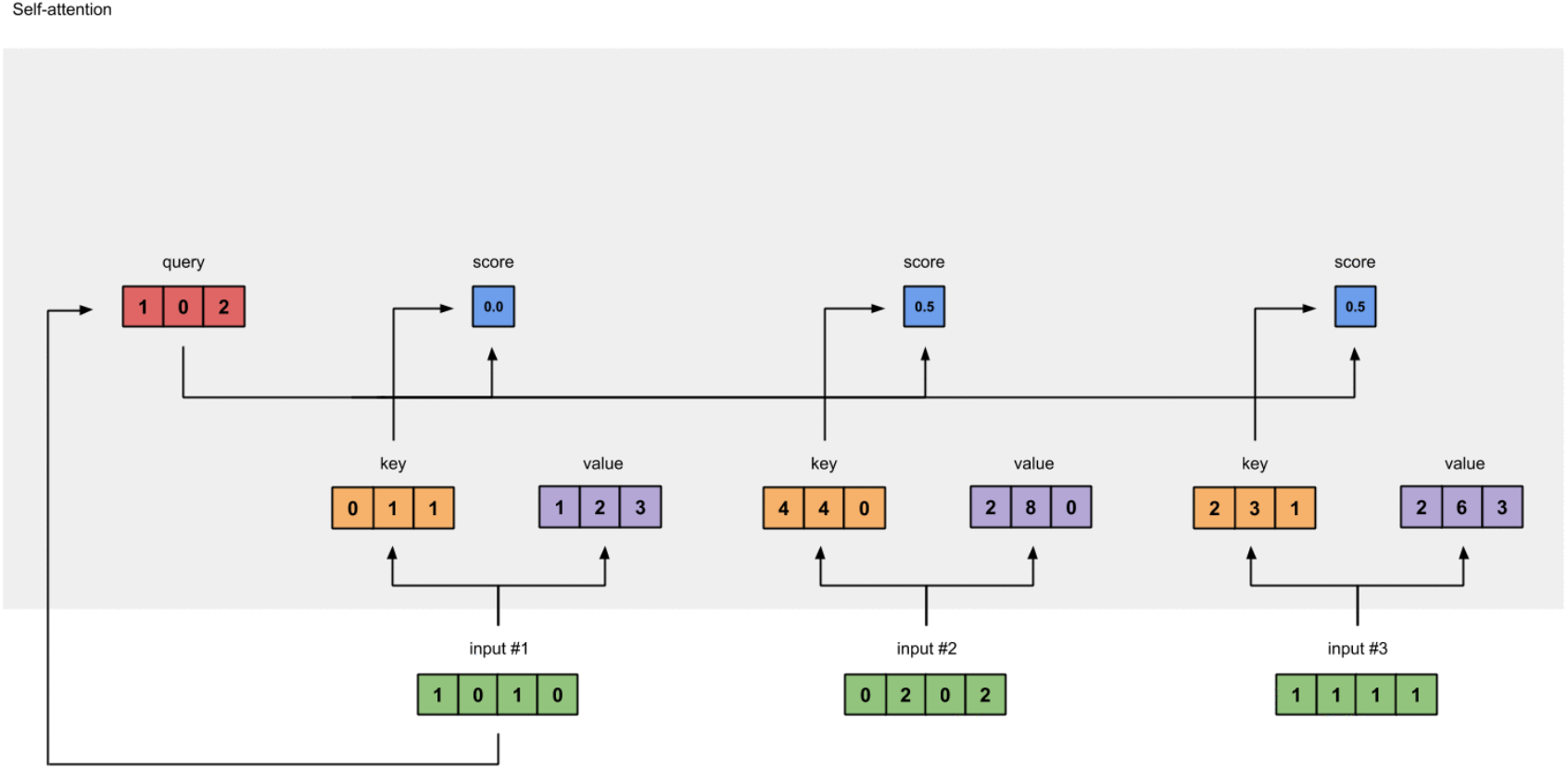
2.4 Self-Attention Layers (by Matrices)

Image Credits: https://jalammar.github.io/illustrated-transformer/



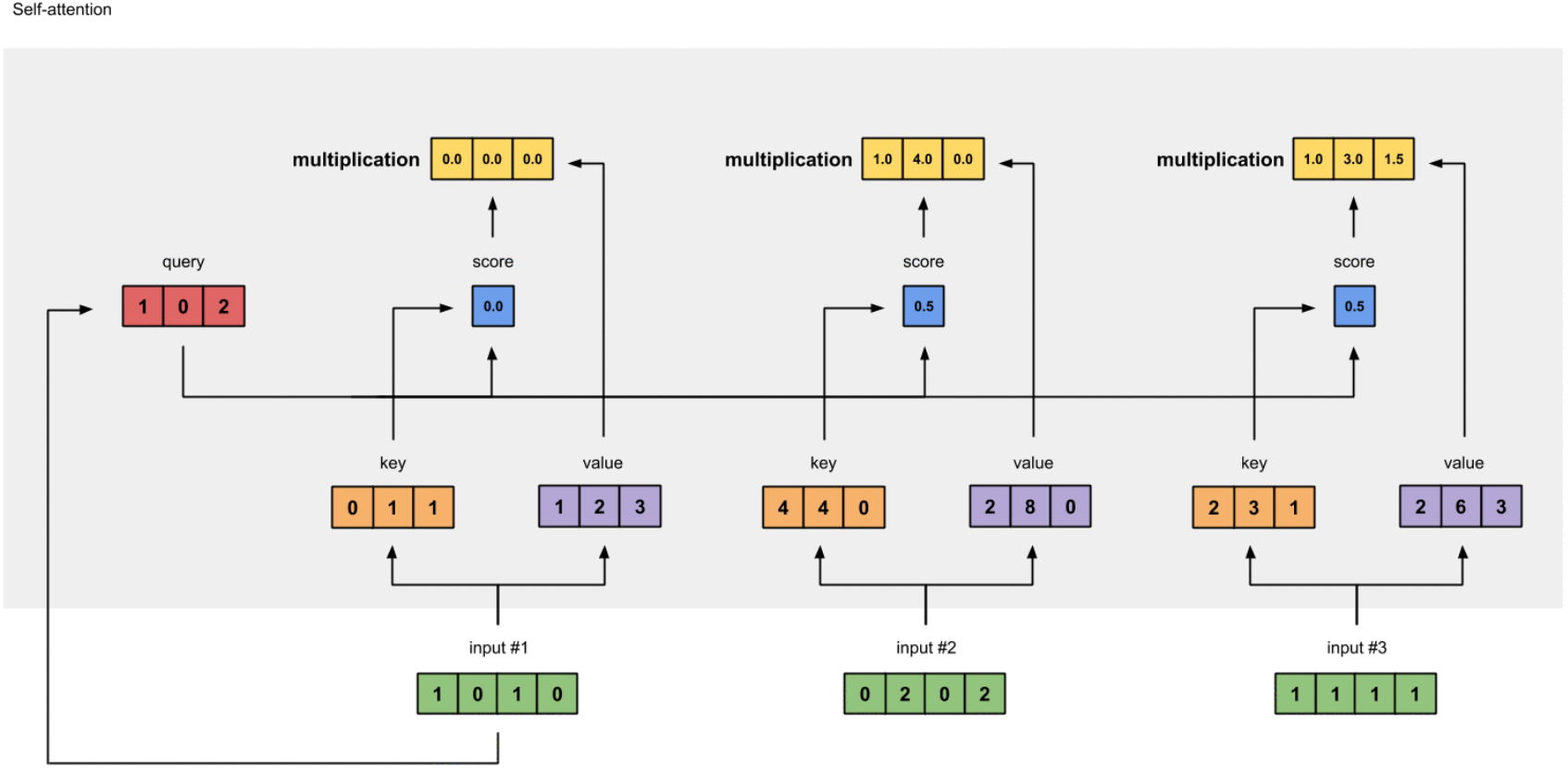
2.4 Self-Attention Layers (by Matrices)

Image Credits: https://jalammar.github.io/illustrated-transformer/



2.4 Self-Attention Layers (by Matrices)

Image Credits: https://jalammar.github.io/illustrated-transformer/



2.5 Multi-headed Self Attention

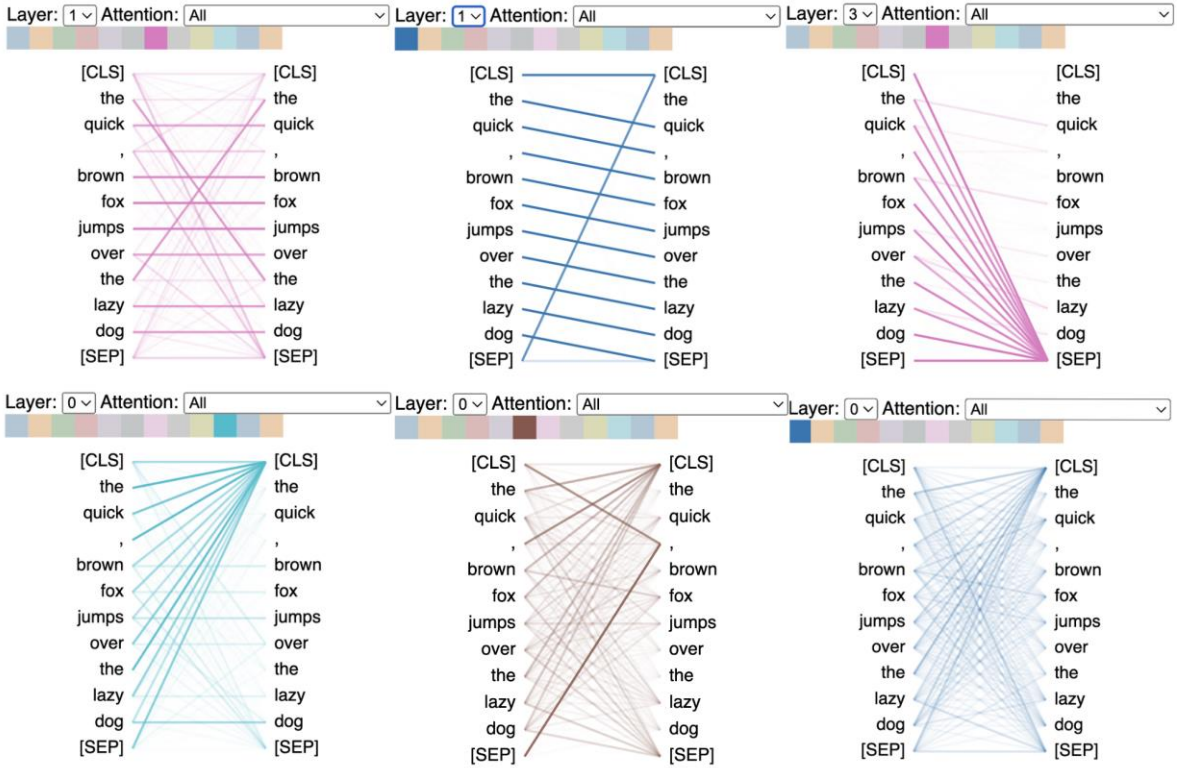


Image Credit: https://www.comet.com/site/blog/explainable-ai-for-transformers/

2.6 Full Transformer Architecture (DeepSeek)

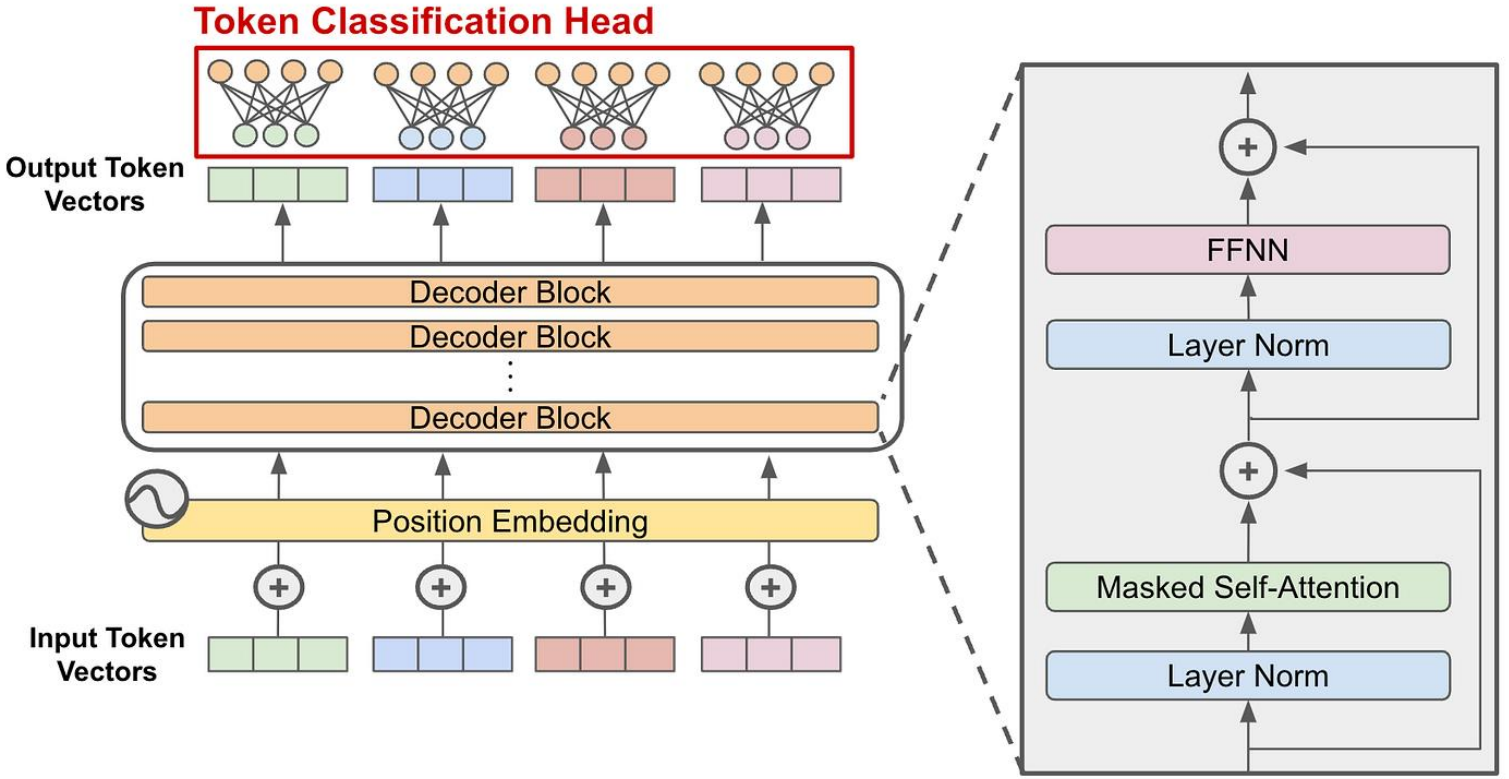


Image Credit: https://medium.com/@tanalpha-aditya/deepseek-coder-when-the-llm-meets-programming-better-than-gpt-3-5-054cf85e3493

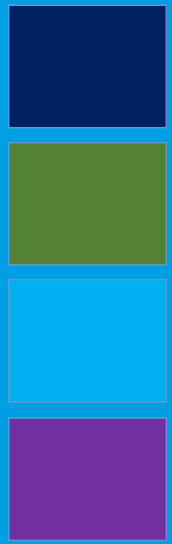
Session Aims

Introduction

Transformers

Pre-Training

Generation



3.1 Pre-training

570Gb

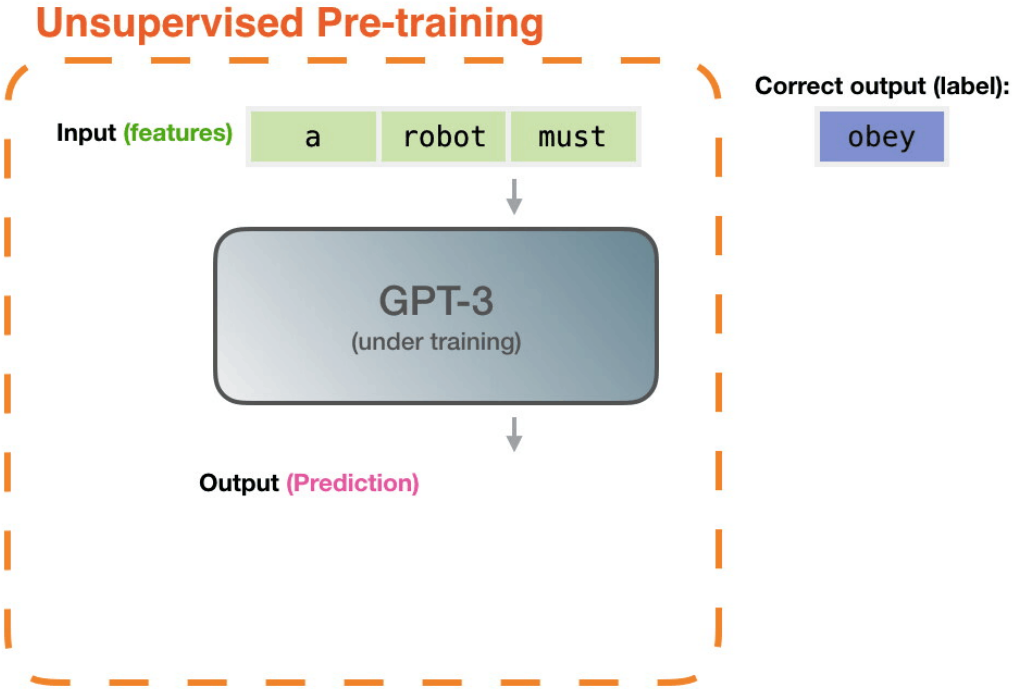
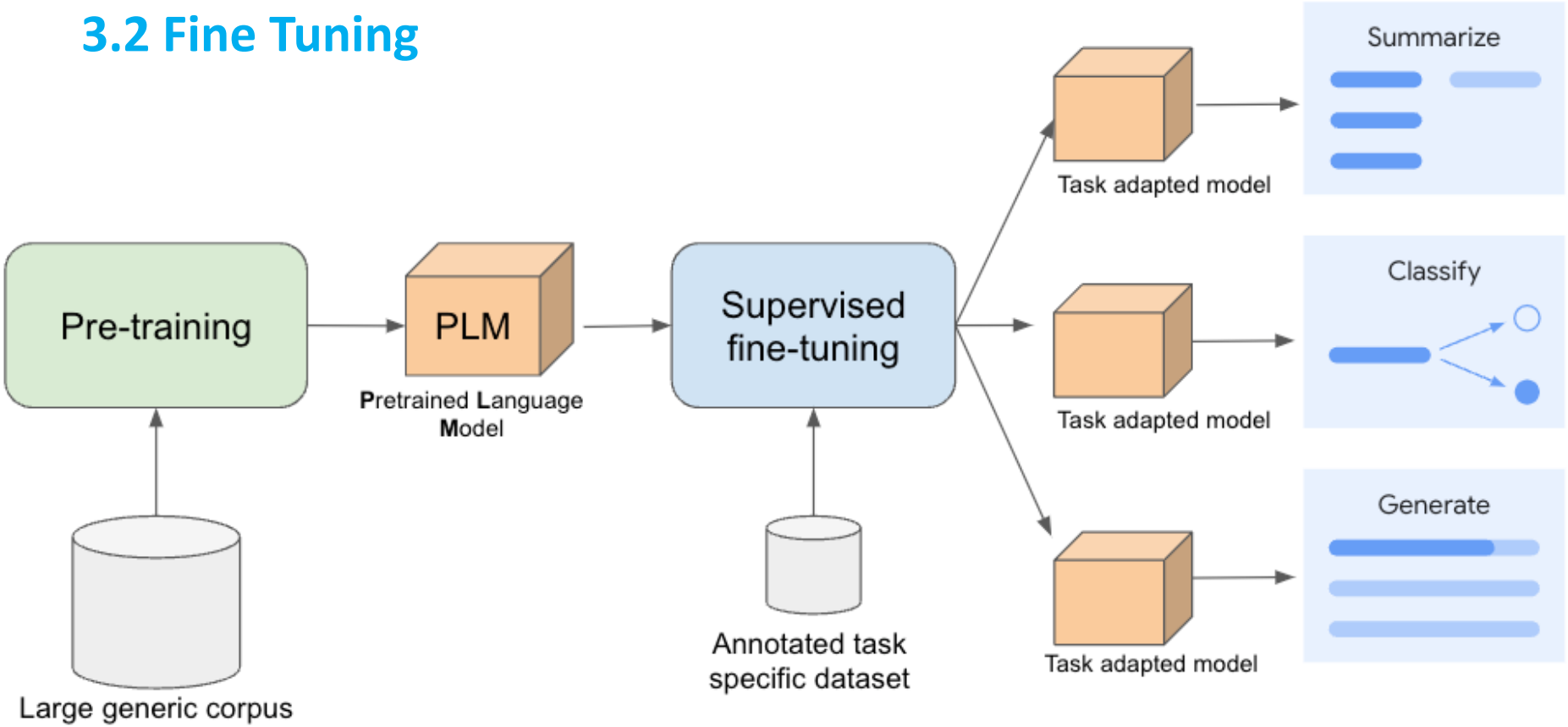


Image Credit: https://jalammar.github.io/illustrated-transformer/

3.2 Fine Tuning



3.2 Fine Tuning

- After this pre-training, where it learns a general skillset in language (and/or other modalities such as images), it is given a much smaller, more specific round of training called *fine tuning*.
- Fine tuning is a second training process, using different data, (typically) a different learning objective (loss function) and a low learning rate (η). You have three years learning your discipline at Uni, your new employer will give you two weeks further training to learn the specifics of how the company works in this space.
- Effectively we nudge the the model to go beyond just word prediction to perform specialist tasks. I.e. for ChatGPT this fine tuning means the model goes from being able to write coherent sentences, to coherent sentences people actually want to read!

ChatGPT 4o

3.3 Reinforcement Learning with Human Feedback (RLHF)

A

Which response do you prefer?

Your choice will help make ChatGPT better.

Response 1

Correct.

Explanation: The main categories of Microsoft 365 plans are Home, Business, and Enterprise.

Next Question:

Question 4: Which Microsoft 365 Enterprise plan includes advanced security and compliance features?

- A. E1
- B. E2
- C. E3
- D. E5

What's your answer?

Response 2

Correct.

Explanation: The main categories of Microsoft 365 plans are Home, Business, and Enterprise.

Next Question:

Question 4: Which Microsoft 365 Enterprise plan includes advanced security and compliance features?

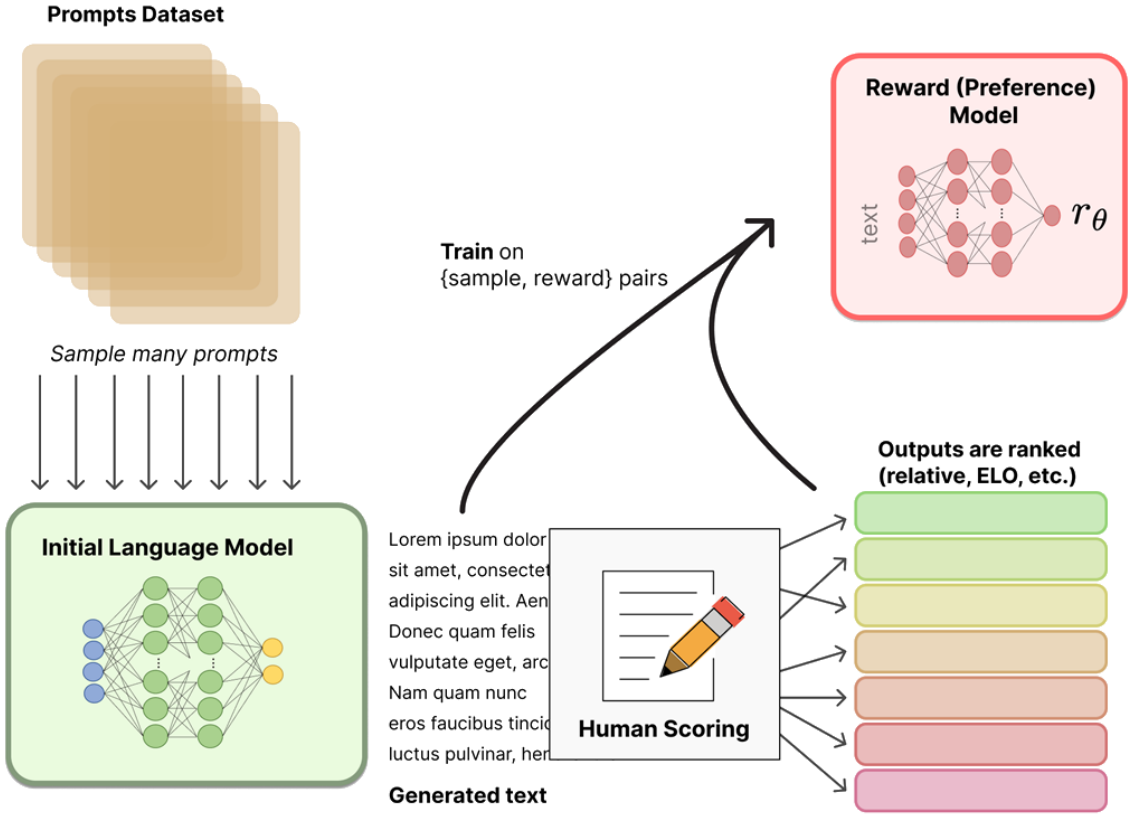
- A. E1
- B. E2
- C. E3
- D. E5

What's your answer?

Message ChatGPT

ChatGPT can make mistakes. Check important info.

3.3 Reinforcement Learning with Human Feedback (RLHF)



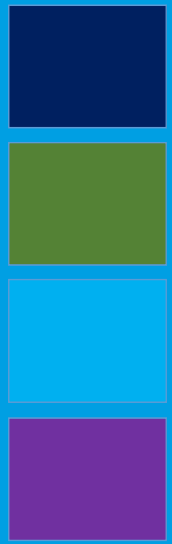
Session Aims

Introduction

Transformers

Pre-Training

Generation



4.1 Transformer Generation

- In the output layer for generative models we want to produce a word. This is equivalent to a multi-classification problem with one output neuron for each word in the vocabulary (including when to stop).
- If the previous input is “please complete the module ___”, the probability that the next word should be “evaluation” goes up.
- We can then select, according to this distribution, a highly probable word to generate next.

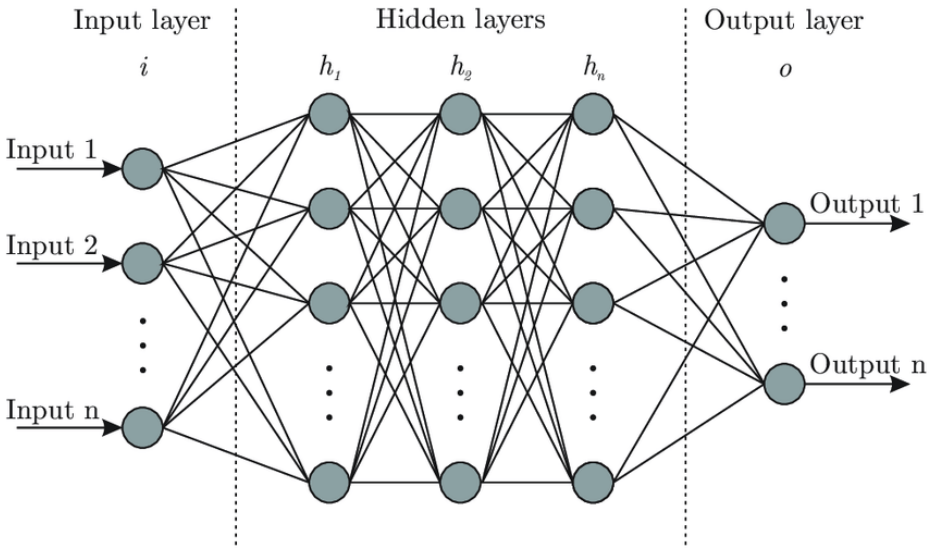


Image Credits:
https://jalammar.github.io/illustrated-transformer/

Output of the last decoder block's linear layer. A score for every word in the vocab.

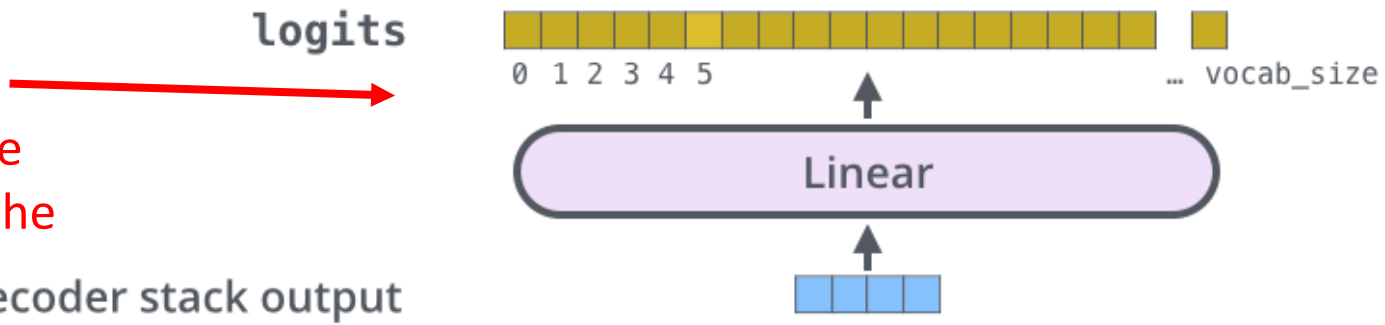


Image Credits:
https://jalammar.github.io/illustrated-transformer/

Convert to class probabilities.

Output of the last decoder block's linear layer. A score for every word in the vocab.

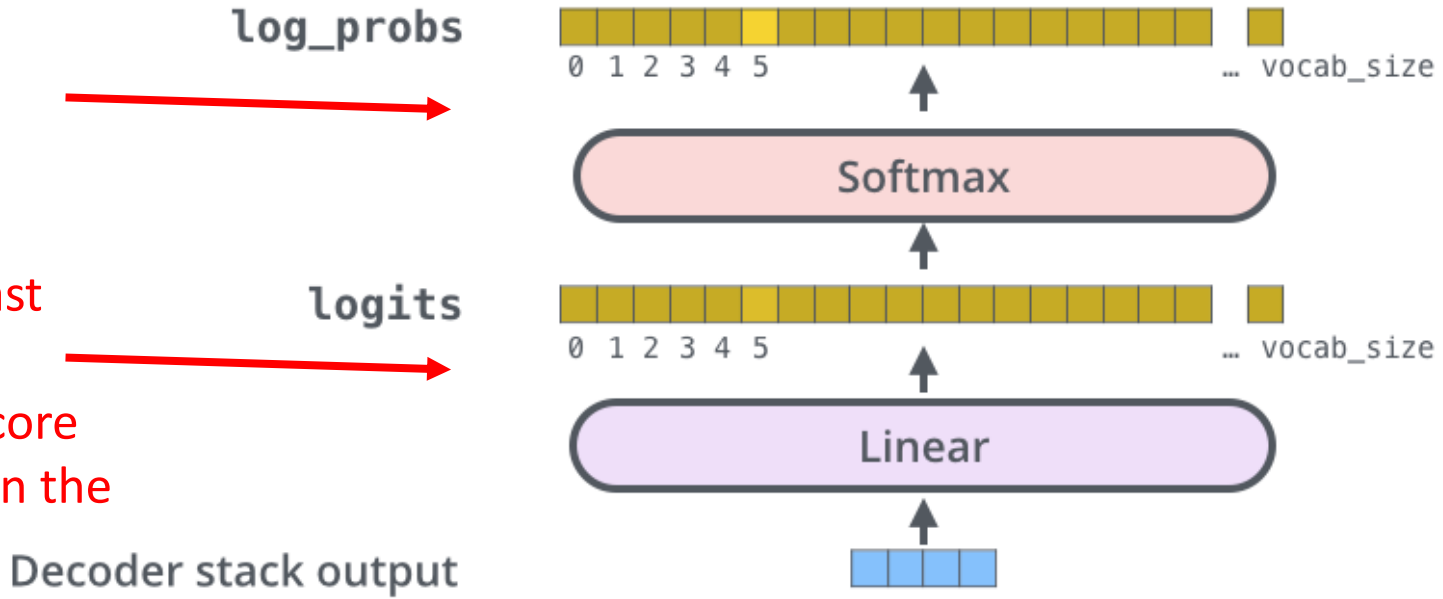
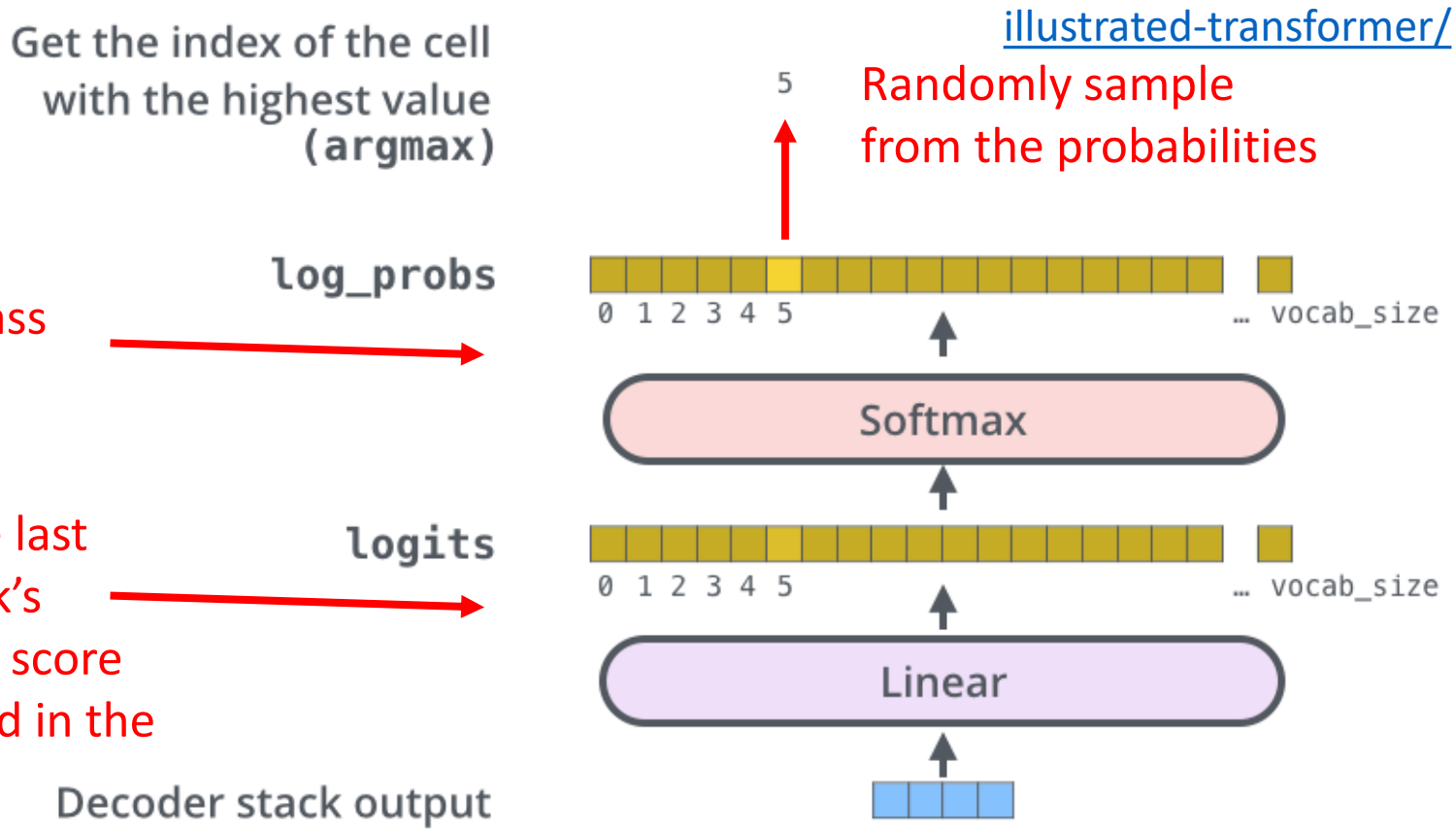


Image Credits:
https://jalammar.github.io/illustrated-transformer/



Which word in our vocabulary is associated with this index?

Get the index of the cell with the highest value (argmax)

log_probs

Convert to class probabilities.

Output of the last decoder block's linear layer. A score for every word in the vocab.

logits

Decoder stack output

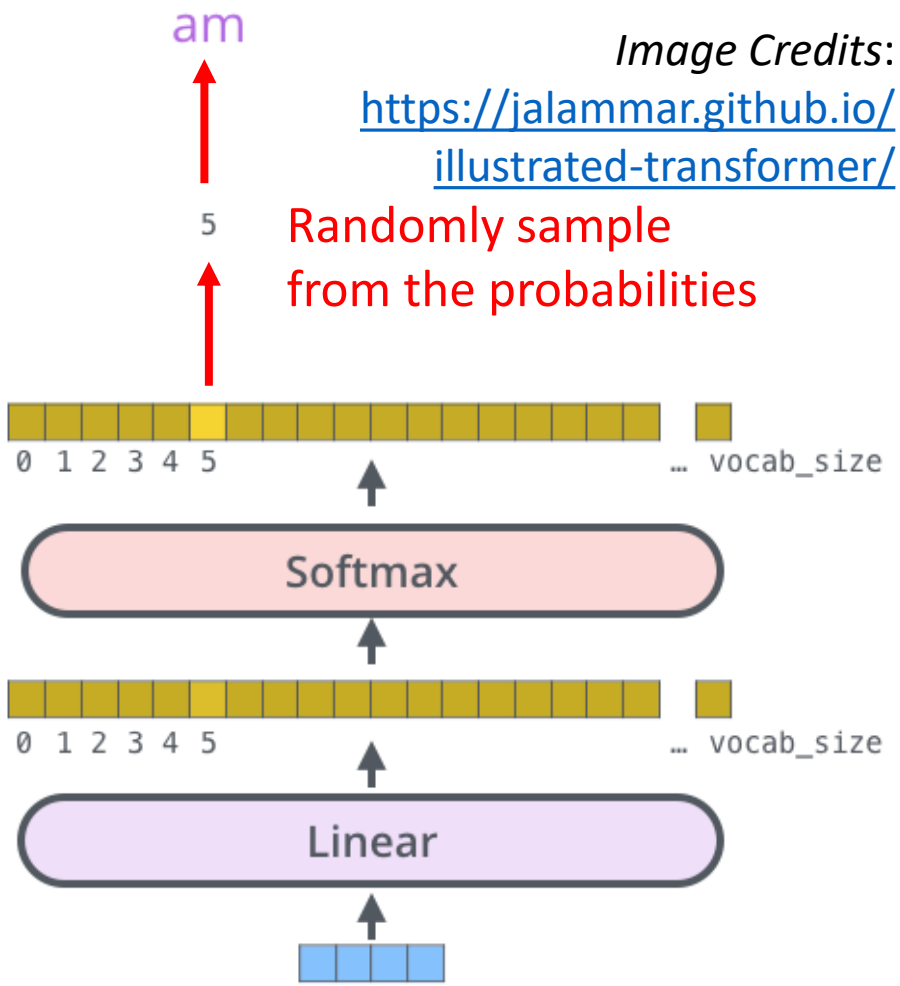


Image Credits: https://jalammar.github.io/illustrated-transformer/

Randomly sample from the probabilities

4.2 Output Generation

- Every time the AI generates a word it is paying attention to the words you typed in (the *prompt*) and also **all** of the words it has already generated.
- If your prompt is not perfect then the output may not be perfect! If parts of the previous output are not perfect, it may well pay attention to these too and continually produce imperfect output. This means AI tools can get stuck in a loop where they keep making the same mistakes.
- The same process can be used to generate pixels (image generation) and ultimately any other form of output (videos, audio files, etc.). We'll discuss best practices next week!

Further Reading

- 3Blue1Brown (2024). *Transformers (how LLMs work) explained visually | DL5*. https://www.youtube.com/watch?v=wjZofJX0v4M.
- Alammar J and Grootendorst M (2024). *Hands-On Large Language Models*. Sebastopol, CA: O'Reily.
- Alammar J (n.d.). *The Illustrated Transformer.* https://jalammar.github.io/illustrated-transformer/.
- **Goodfellow I, Bengio Y and Courville A (2016). *Deep Learning*. The MIT Press: Cambridge, MA.**
- Karpathy A (2023). *Intro to Large Language Models*. https://www.youtube.com/watch?v=zjkBMFhNj_g.
- Pajankar A and Joshi A (2022). *Hands-on machine learning with Python: implement neural network solutions with Scikit-learn and PyTorch.* Apress: Berkeley, CA.
- Vaswani A *et al.* (2017). Attention is all you need. In: *31st International Conference on Neural Information Processing Systems (NIPS '17)*. Red Hook, NY, USA, 6000–6010.