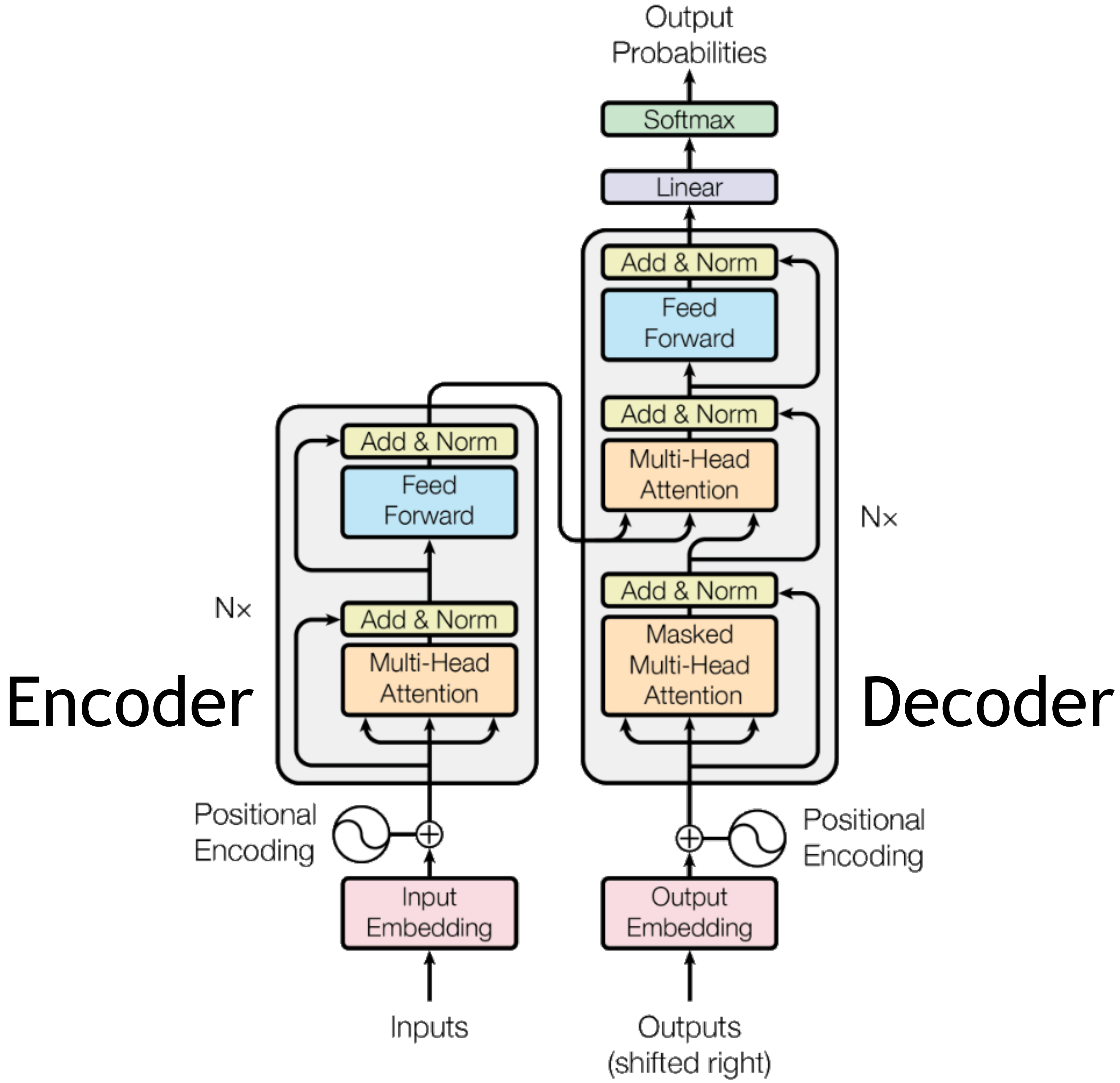
ANFÄNGERPRAKTIKUM
NEURAL NETWORKS FROM SCRATCH

TRANSFORMERS

Hendrik Borras, Franz Kevin Stehle
hendrik.borras@ziti.uni-heidelberg.de, kevin.stehle@ziti.uni-heidelberg.de
HAWAII Group, Institute of Computer Engineering
Heidelberg University

TRANSFORMERS - BASICS

- Introduced in infamous 2017 paper “Attention is All You Need”
- Based on the so-called *Self-Attention* mechanism
- Advantages of transformers over previous architectures (e.g. Recurrent Neural Networks, Long-Short-Term-Memory Networks):
  - Favorable scaling of memory/compute requirements -> good scalability
  - Input sequence processed as a whole instead of sequentially -> good parallelizability
  - Relatively good interpretability



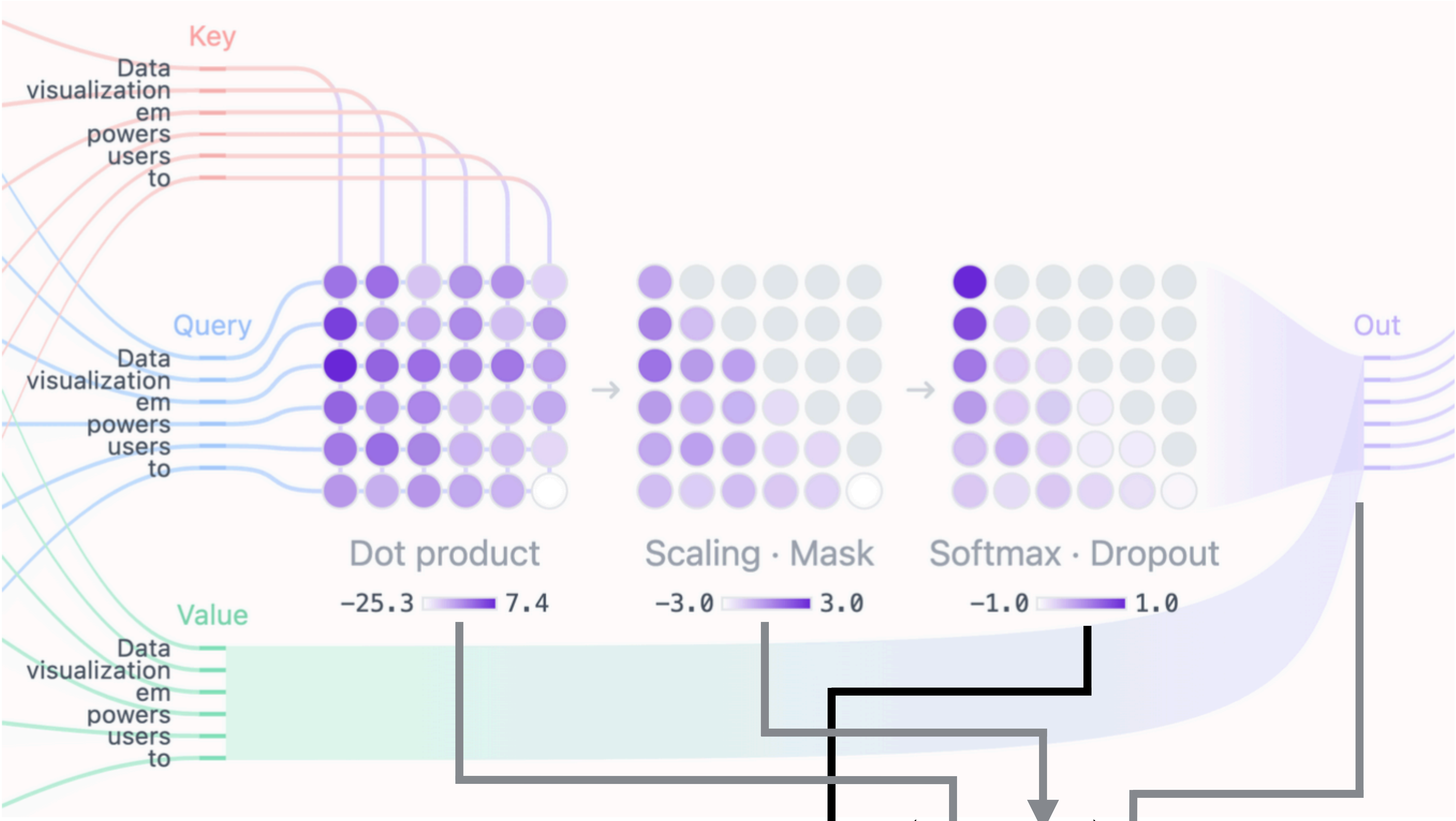
Ashish Vaswani et al.: Attention is All You Need, 2017. https://arxiv.org/abs/1706.03762

COMPUTATIONAL COSTS

Sequence length t, number of layers d, number of neurons per layer k

	Training complexity	Training memory	Test complexity	Test memory
RNN	tk^2d	tkd	tk^2d	kd
RNN + attention	t^2k^2d	t^2kd	t^2k^2d	tkd
Transformer	t^2kd	tkd	t^2kd	tkd

TRANSFORMERS - SELF-ATTENTION



$$Attention(Q, K, V) = softmax\left(QK^T \frac{1}{\sqrt{d_k}}\right)V$$

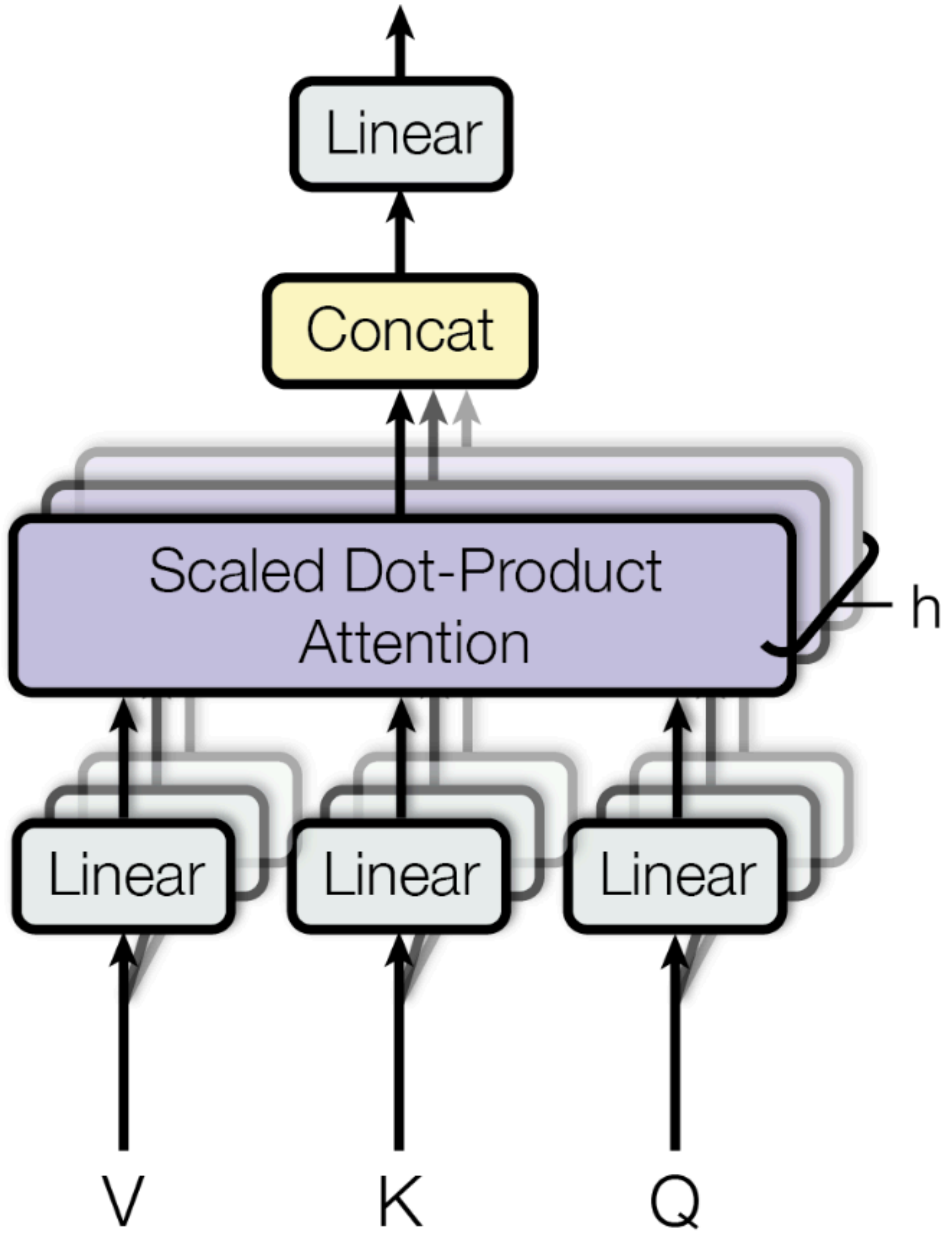
Aeree Cho et al.: Transformer Explainer: Interactive Learning of Text-Generative Models, 2024.

Paper: https://arxiv.org/abs/2408.04619 Demo: https://poloclub.github.io/transformer-explainer/

TRANSFORMERS - MULTI-HEAD-ATTENTION

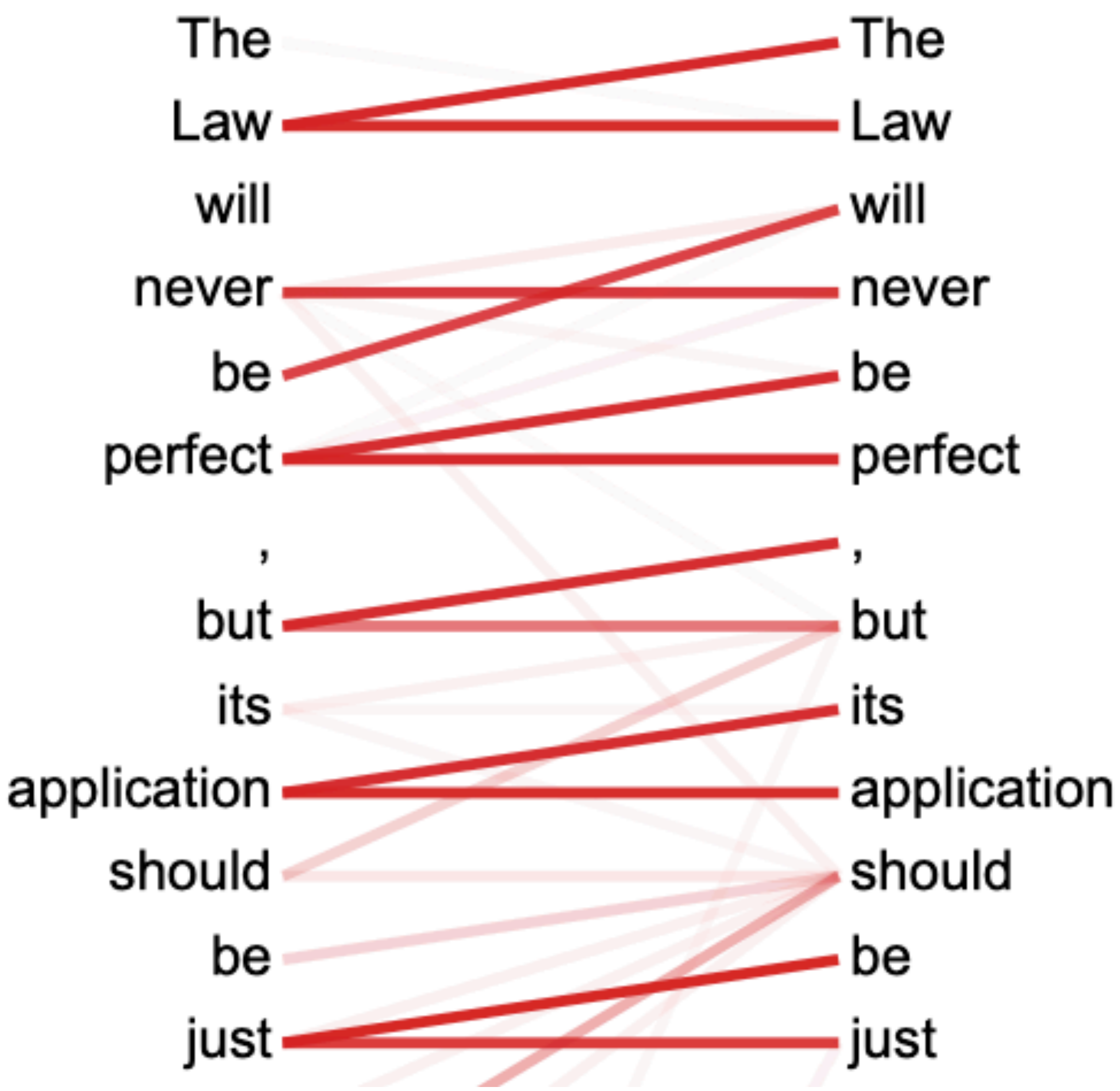
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHead(Q, K, V) = concat(head_0, \ldots, head_h)W^O$$

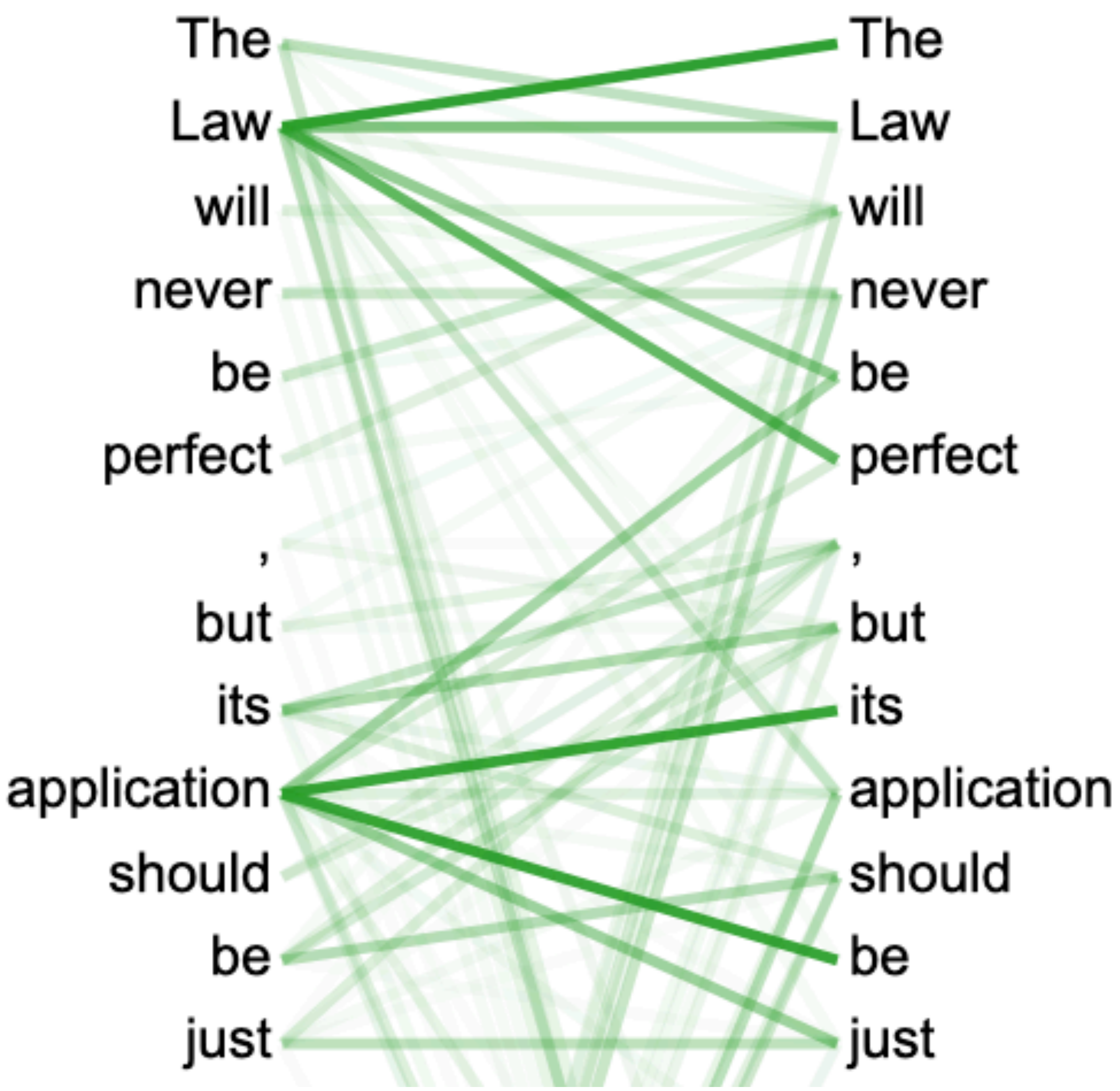


Ashish Vaswani et al.: Attention is All You Need, 2017. https://arxiv.org/abs/1706.03762

ATTENTION VISUALIZATION - TEXT DATA



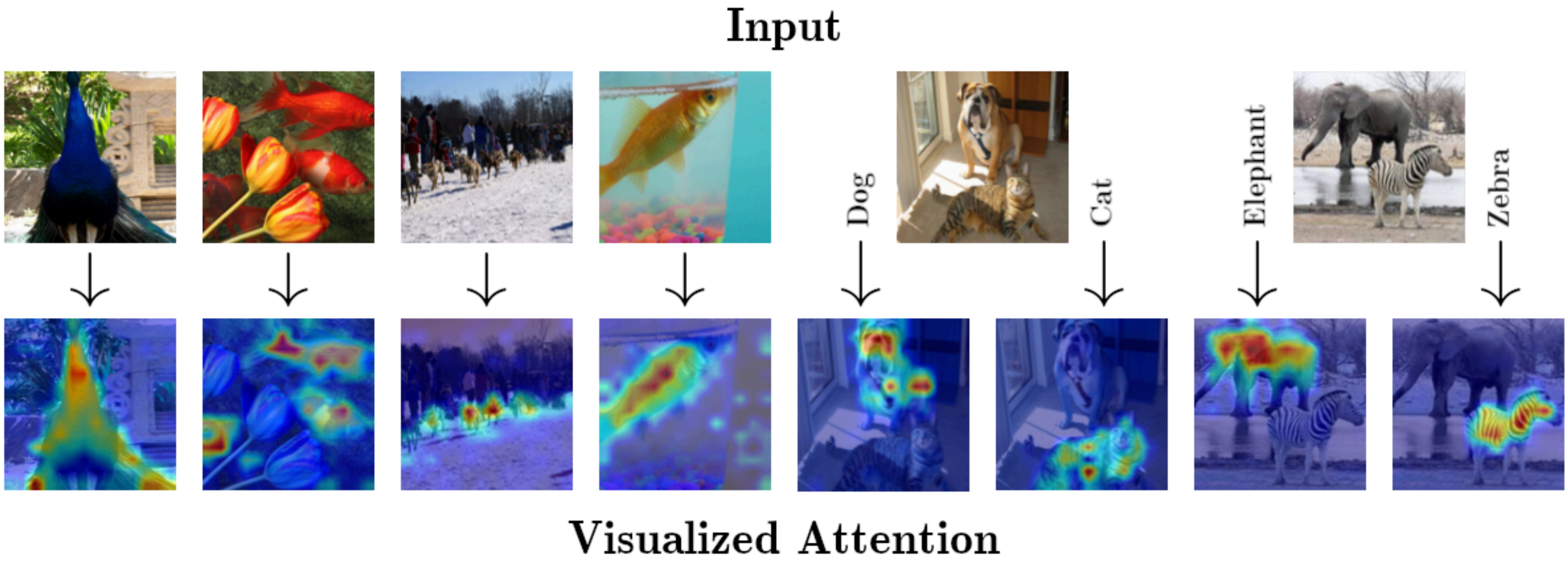
Attention Head 0



Attention Head 1

Ashish Vaswani et al.: Attention is All You Need, 2017. *https://arxiv.org/abs/1706.03762*

ATTENTION VISUALIZATION - IMAGE DATA



Hila Chefer, Shir Gur, Lior Wolf: Transformer Interpretability Beyond Attention Visualization, 2021. https://arxiv.org/abs/2012.09838v2

TRANSFORMER TAXONOMY

Encoder-Only

Auto-encoding models:
Attention layers can access the whole sentence

Example Tasks:
Classification, Question Answering

Example Models:
BERT family

Encoder-Decoder

sequence-to-sequence models:
Access patterns for encoder part as in encoder-only models, for decoder as in decoder-only models

Example Tasks:
Translation, Summarization

Example Models:
Original Transformer, BART, T5

Decoder-Only

Auto-regressive models:
At each position, attention layers can only access elements positioned before it in the sequence

Example Tasks:
Text Generation

Example Models:
GPT family

Shervin Minaee et al.: Large Language Models: A Survey, 2024. https://arxiv.org/abs/2402.06196

HOW DOES A MACHINE LEARNING MODEL PROCESS SENTENCES?

So far: Image data -> Fixed input size

Problem:

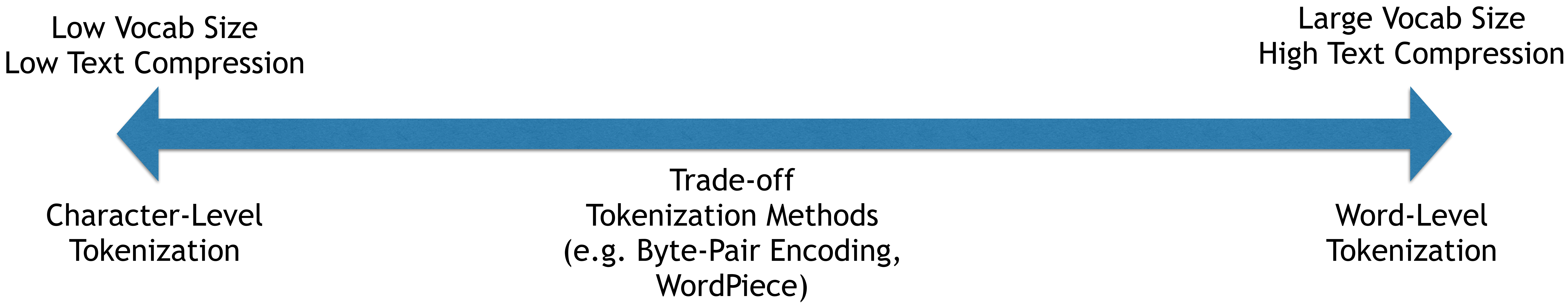
- How does a machine learning model handle sentences that vary immensely in length?
- How does one encode text in such a way that the model can make sense of it?

Solution:

1. Tokenization: Transformation of input text into numerical representations
2. Embedding: Project tokenized text into higher-dimensional embedding vectors so that the resulting vectors group sentence particles together, e.g. through co-occurrence or semantic closeness
3. Positional Encoding: Add position information to the embedding vectors

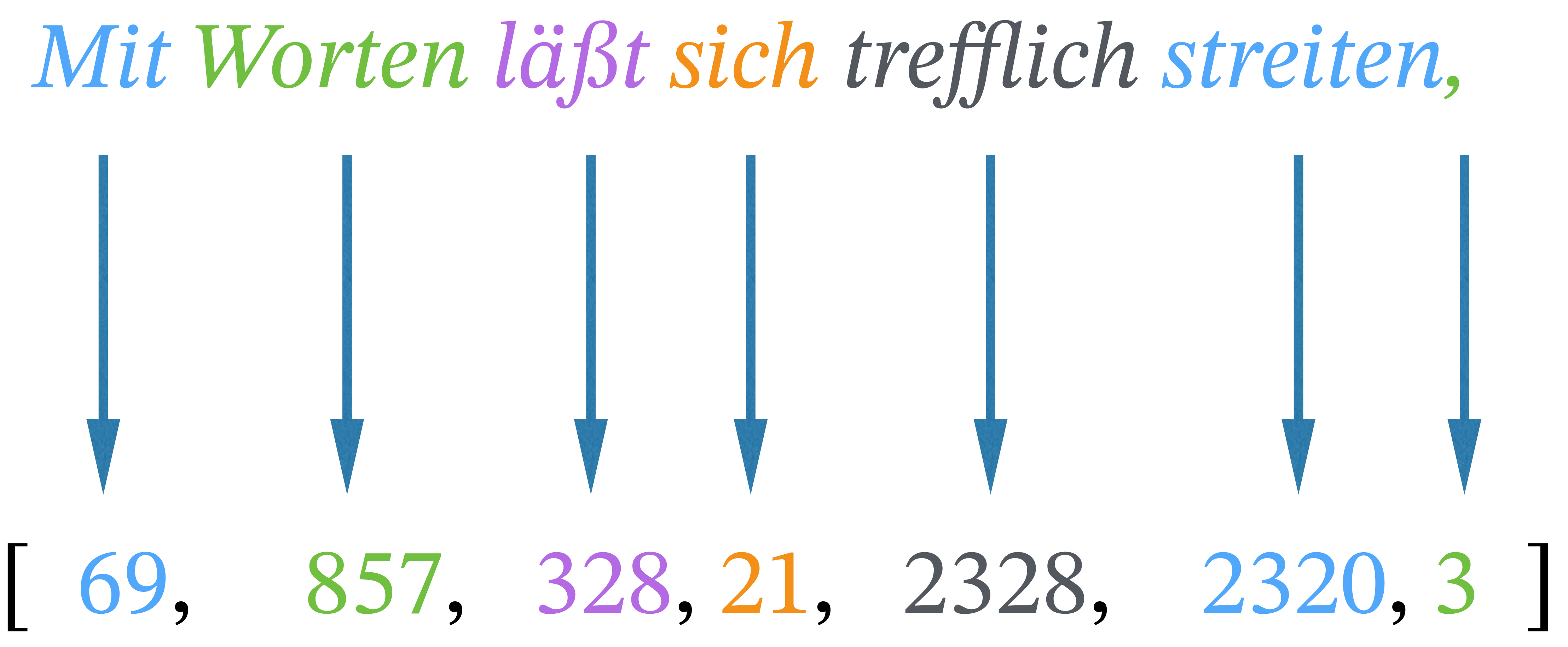
TOKENIZATION

- Representation of input text as tokens (most commonly integers) that form a fixed-size *vocabulary*
- Transformer output prediction represents likelihood of a given token following after the given input sequence
- Inherent trade-off between vocabulary size (determines model input/output dimensions) and number of tokens required to represent text (i.e., the compression ratio)



TOKENIZATION: WORD-LEVEL

- Process training set and assign a token to each unique word (or punctuation symbol) encountered
- High compression, high vocabulary size
- Requires special *unknown* token to represent words not encountered in the training set



TOKENIZATION: WORD-LEVEL

- Process training set and assign a token to each unique word (or punctuation symbol) encountered
- High compression, high vocabulary size
- Requires special *unknown* token to represent words not encountered in the training set

```
<|S|>
SCHÜLER.
Doch ein Begriff muß bei dem Worte sein.

MEPHISTOPHELES.
Schon gut! Nur muß man sich nicht allzu ängstlich quälen
Denn eben wo Begriffe fehlen,
Da stellt ein Wort zur rechten Zeit sich ein.
Mit Worten läßt sich trefflich streiten,
Mit Worten ein System bereiten,
An Worte läßt sich trefflich glauben,
Von einem Wort läßt sich kein Jota rauben.
<|E|>
```

TOKENIZATION: CHARACTER-LEVEL

- Process training set and assign a token to each unique *symbol* encountered
- Low compression, low vocabulary size
- Requires special *unknown* token to represent symbols not encountered in the training set

Mit Worten läßt sich trefflich streiten,

[36, 58, 69, 4, 46, 64, 67, 69, 54, 63, 4, 61, 81, 80, 69, 4, 68, 58, 52, 57, 4, 69, 67, 54, 55, 55, 61, 58, 52, 57, 4, 68, 69, 67, 54, 58, 69, 54, 63, 10]

TOKENIZATION: CHARACTER-LEVEL

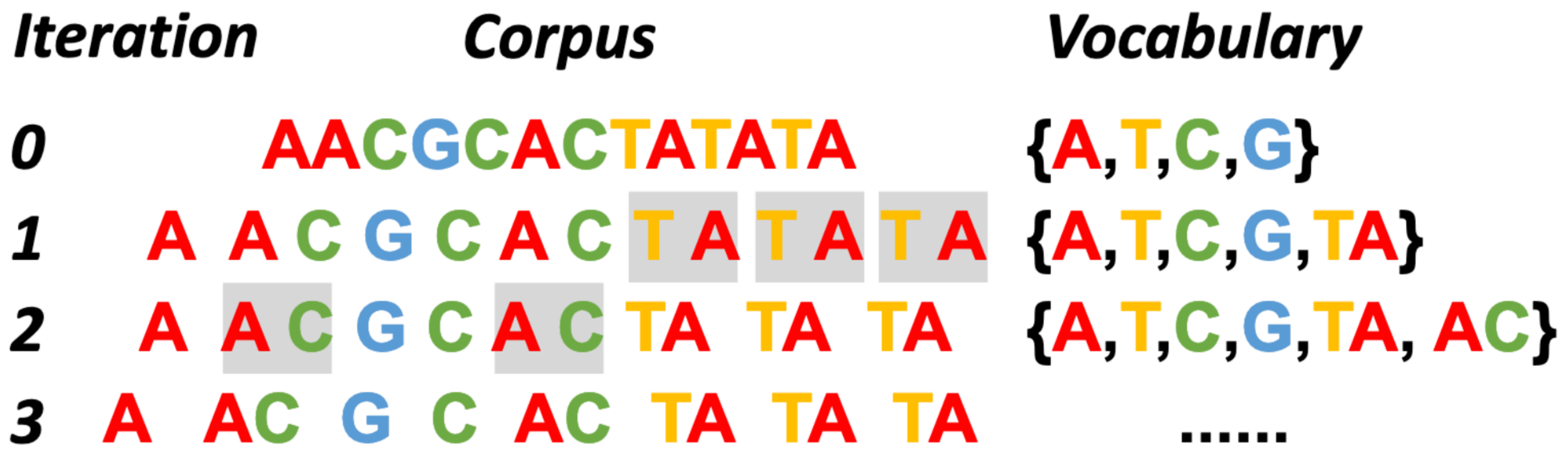
- Process training set and assign a token to each unique *symbol* encountered
- Low compression, low vocabulary size
- Requires special *unknown* token to represent symbols not encountered in the training set

```
<|S|>
SCHÜLER.
Doch ein Begriff muß bei dem Worte sein.

MEPHISTOPHELES.
Schon gut! Nur muß man sich nicht allzu ängstlich quälen
Denn eben wo Begriffe fehlen,
Da stellt ein Wort zur rechten Zeit sich ein.
Mit Worten läßt sich trefflich streiten,
Mit Worten ein System bereiten,
An Worte läßt sich trefflich glauben,
Von einem Wort läßt sich kein Jota rauben.
<|E|>
```

TOKENIZATION: BYTE-PAIR ENCODING

- Process training set and assign a token to each unique symbol. *Then:* Merge symbols commonly occurring together until a given vocabulary size is reached
- Variable trade-off between vocabulary size and compression



Zhihan Zhou et al.: DNABERT-2: Efficient Foundation Model and Benchmark for Multi-Species Genomes, 2024.
https://arxiv.org/abs/2306.15006v2

TOKENIZATION: BYTE-PAIR ENCODING

- Process training set and assign a token to each unique symbol. *Then:* Merge symbols commonly occurring together until a given vocabulary size is reached
- Variable trade-off between vocabulary size and compression

Mit Worten läßt sich trefflich streiten ,

[361, 2548, 1207, 178, 2610, 179, 252, 1471, 7]

TOKENIZATION: BYTE-PAIR ENCODING

- Process training set and assign a token to each unique symbol. *Then:* Merge symbols commonly occurring together until a given vocabulary size is reached
- Variable trade-off between vocabulary size and compression

```
<|S|>
SCHÜLER.
Doch ein Begriff muß bei dem Worte sein.

MEPHISTOPHELES.
Schon gut! Nur muß man sich nicht allzu ängstlich quälen
Denn eben wo Begriffe fehlen,
Da stellt ein Wort zur rechten Zeit sich ein.
Mit Worten läßt sich trefflich streiten,
Mit Worten ein System bereiten,
An Worte läßt sich trefflich glauben,
Von einem Wort läßt sich kein Jota rauben.
<|E|>
```

EMBEDDING

-Projection of tokens into higher-dimensional space

-Intends to capture relationships between tokens so that related tokens are close together in the embedding space

-Usually trained alongside the model and thus "hidden"

Vocabulary size: V

Token: $x \in \{0,1,\ldots,V-1\}$

Embedding matrix: $E \in \mathbb{R}^{V \times n_{embd}}$

Embedding vector: $f(x) \in \mathbb{R}^{n_{embd}}$

Embedding function:

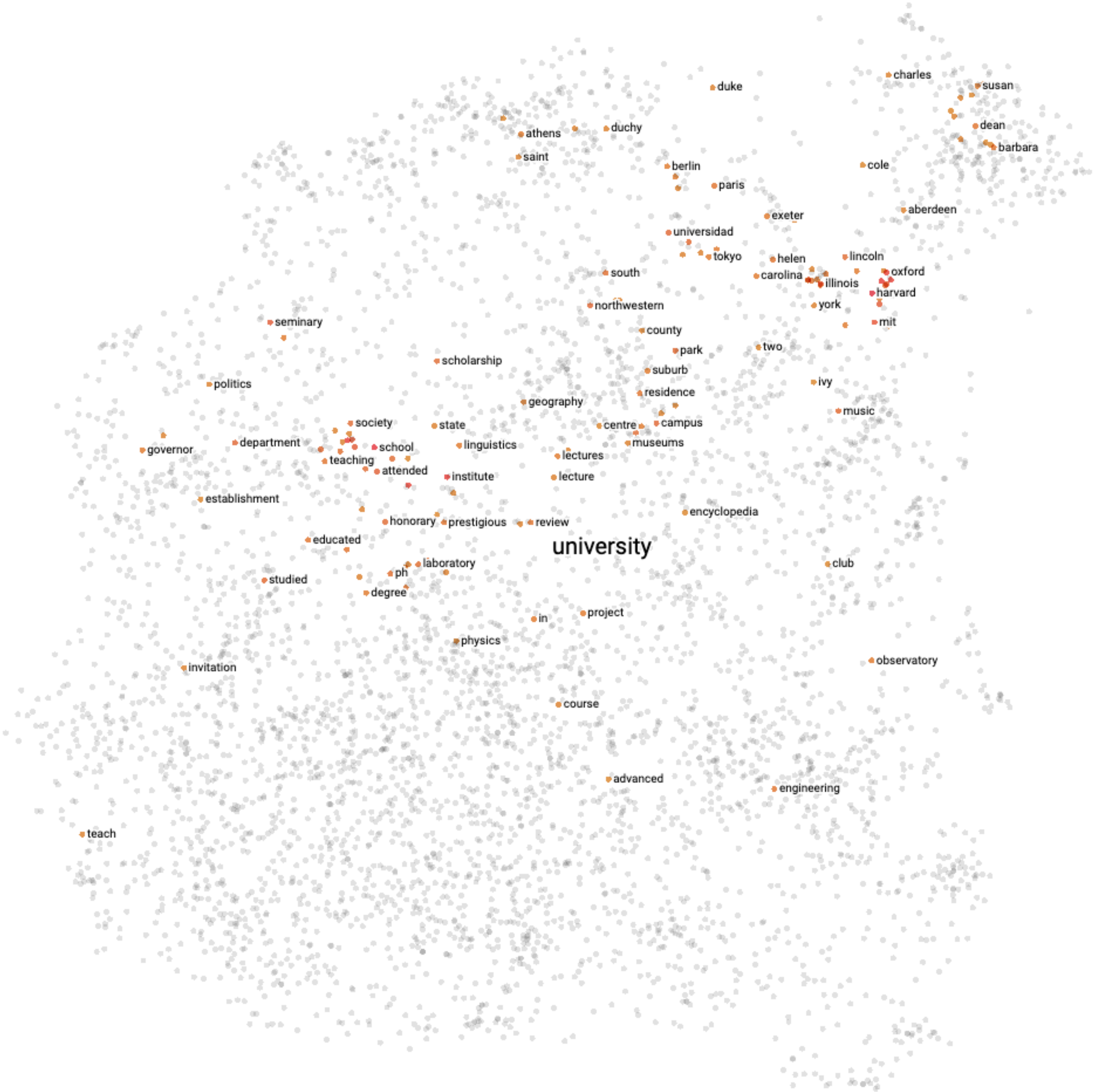
$$f : \{0,1,\ldots,V-1\} \to \mathbb{R}^{n_{embd}}$$

Batched dimensions:

Tokens: $X \in \mathbb{Z}^{B \times len_{seq}}$

Embeddings: $f(X) \in \mathbb{R}^{B \times len_{seq} \times n_{embd}}$

EMBEDDING - EXAMPLE



Nearest points in the original space:

college	0.235
harvard	0.278
school	0.294
universities	0.301
institute	0.304
cambridge	0.311
graduate	0.315
oxford	0.326
yale	0.332
stanford	0.332
professor	0.335
columbia	0.365
students	0.373
berkeley	0.374
colleges	0.375
princeton	0.376
mit	0.380
faculty	0.395
undergraduate	0.395
seminary	0.395
illinois	0.398
education	0.399
academic	0.401
chicago	0.402
academy	0.404
press	0.408
california	0.409
attended	0.412
cornell	0.414
student	0.415
arts	0.418

https://projector.tensorflow.org/

POSITIONAL ENCODING

As transformers process every element in the input sequence simultaneously, they have no inherent sense of position. [1]

Positional encodings are thus added to the embedded data to add positional information.

Example:

Original transformer positional encoding (sinusoid) [1]:

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

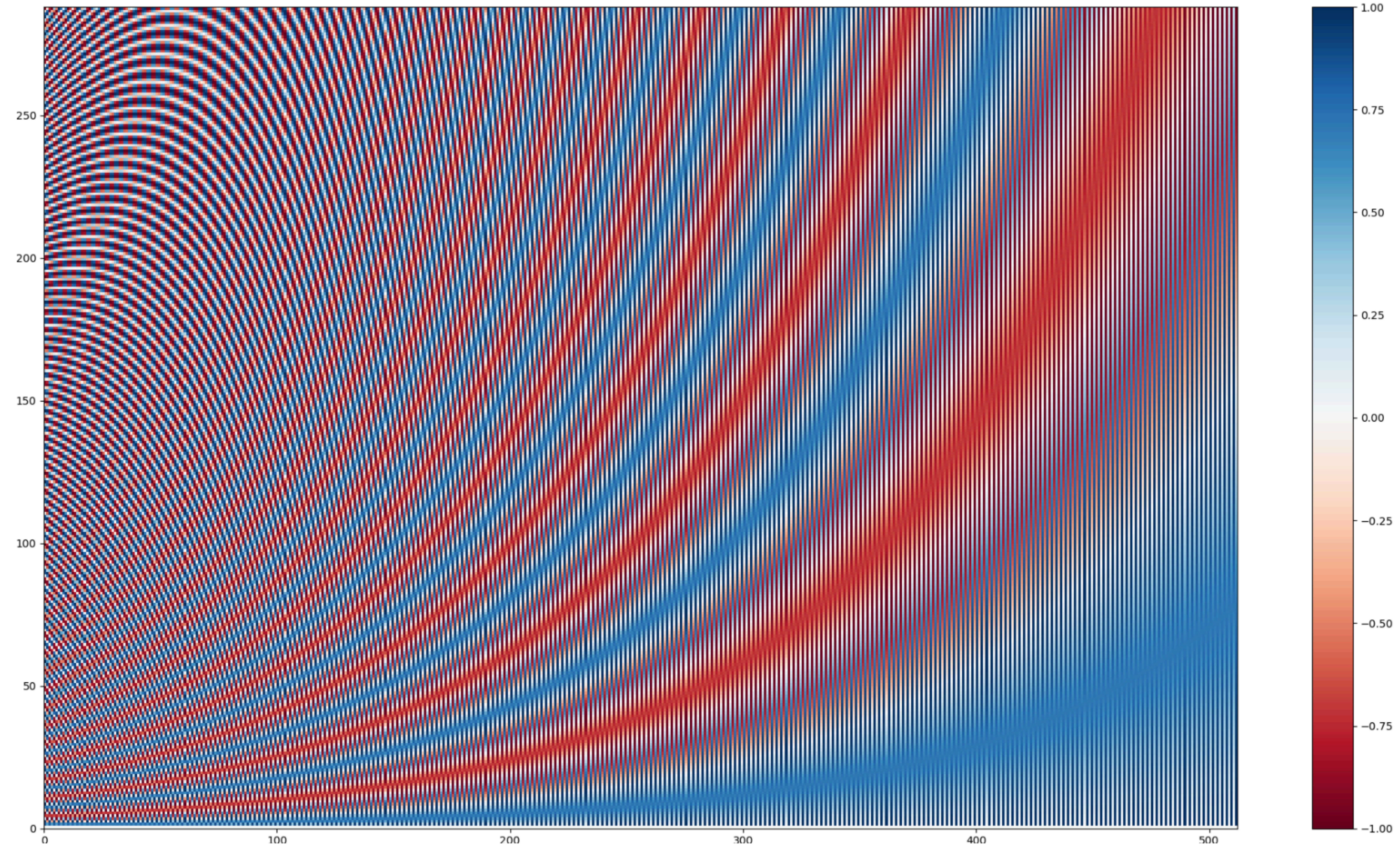
Application to the embeddings [2]:

$$z_i = WE(x_i) + PE(i)$$

[1] Ashish Vaswani et al.: Attention is All You Need, 2017. https://arxiv.org/abs/1706.03762

[2] Yu-An Wang, Yun-Nung Cheng: What Do Position Embeddings Learn? An Empirical Study of Pre-Trained Language Model Positional Encoding, 2020. https://arxiv.org/abs/2010.04903

POSITIONAL ENCODING

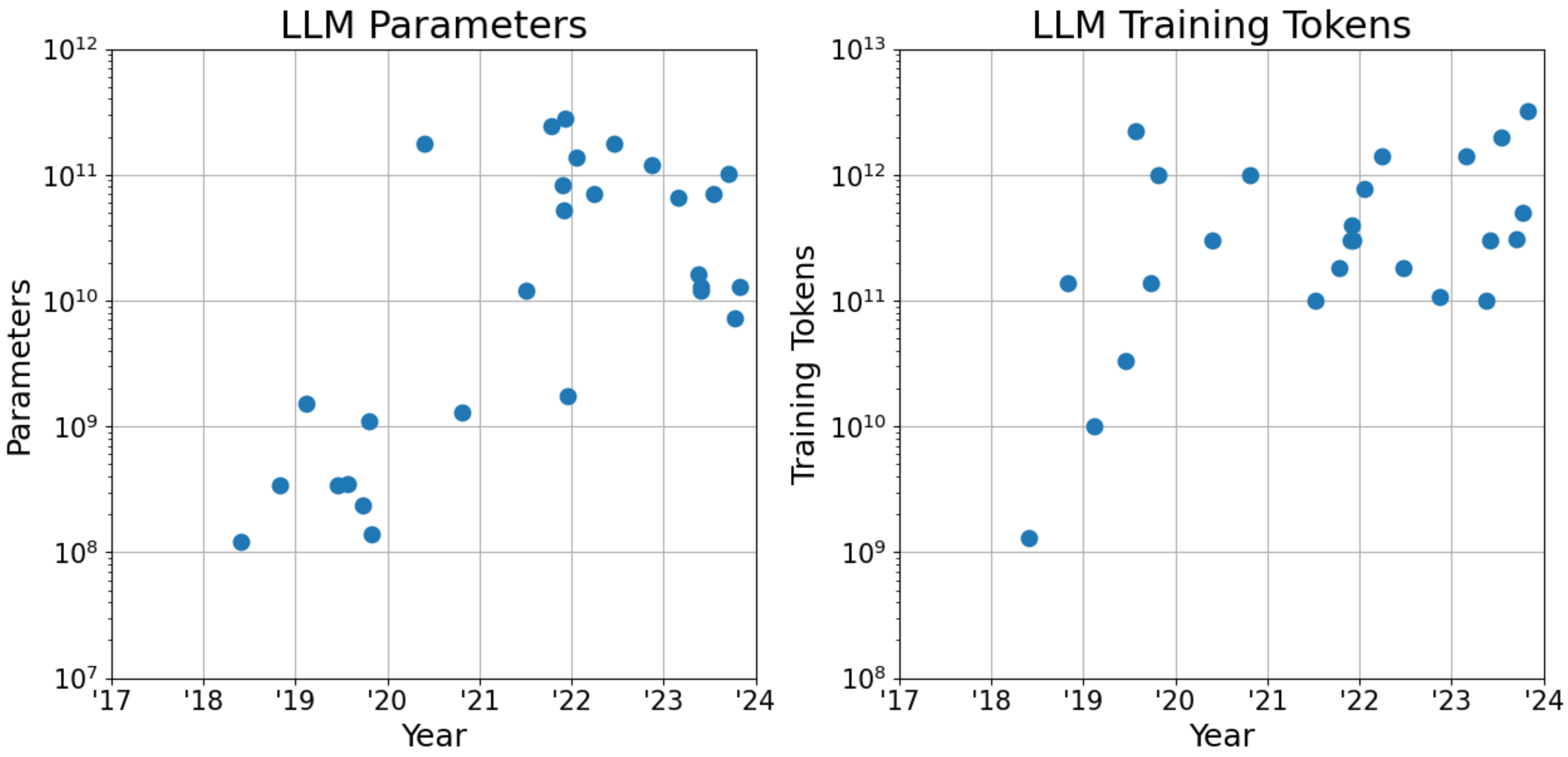


$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

Bin Yang, Tinghuai Ma, and Xuejian Huang: ATFSAD: Enhancing Long Sequence Time-Series Forecasting on Air Temperature Prediction. IEEE Access, vol. 11, pp. 92080-92091, 2023, doi: 10.1109/ACCESS.2023.3308693.

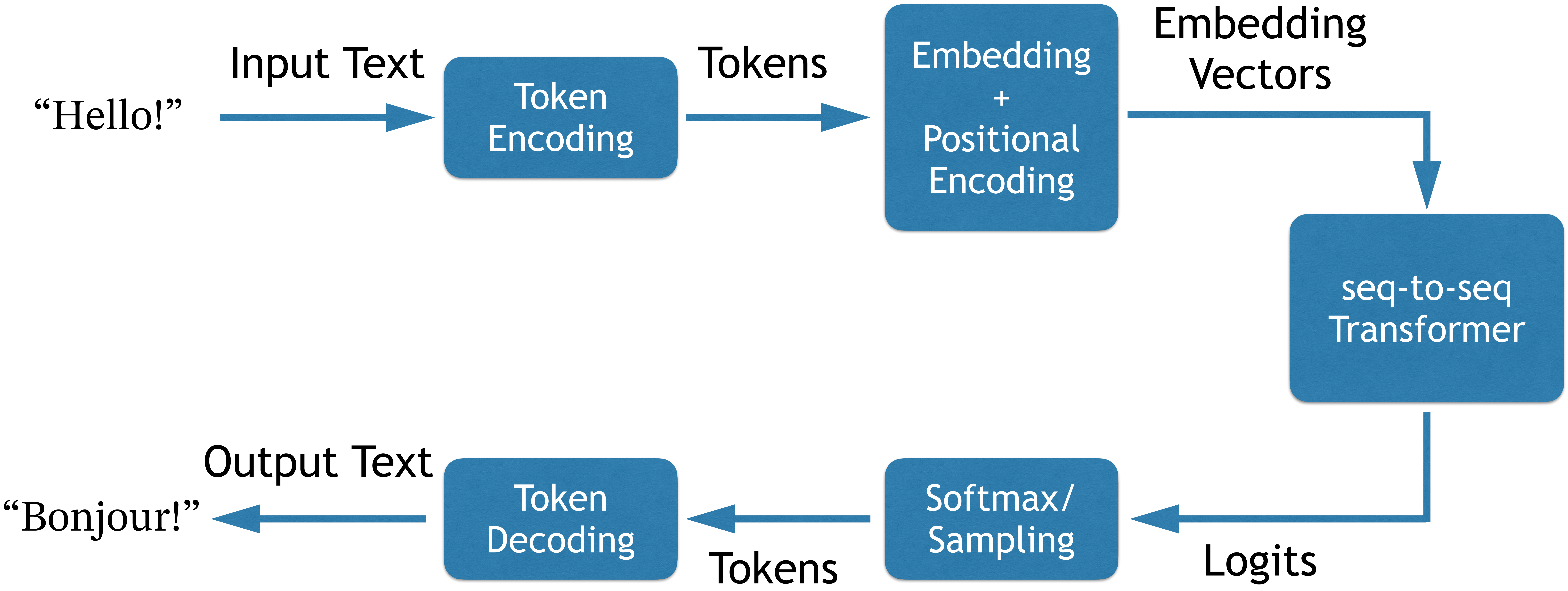
THE GROWING SIZE AND COST OF STATE-OF-THE-ART LLMS



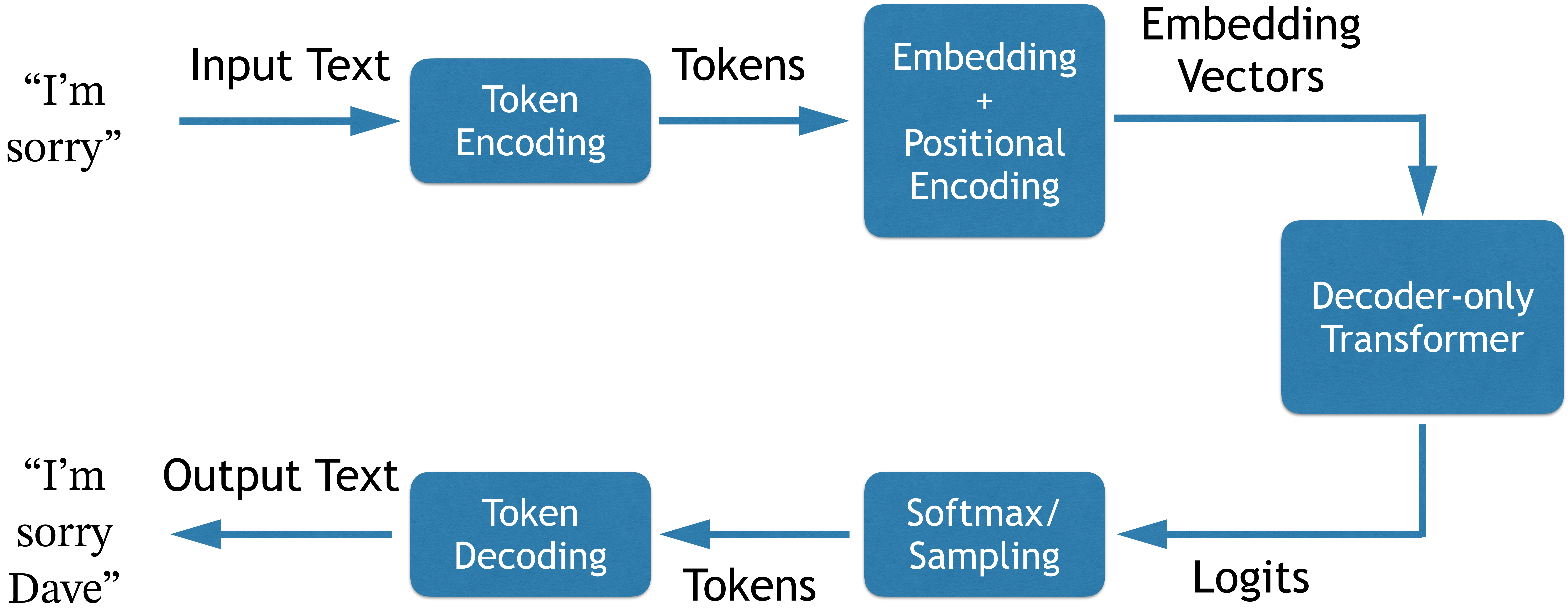
Development of LLM parameter counts and training data tokens for a selection of well-known models.

Kevin Franz Stehle: How much "Brain Damage" can an LLM Tolerate?, 2024.
https://csg.ziti.uni-heidelberg.de/blog/llm-brain-damage/

PUTTING IT ALL TOGETHER



PUTTING IT ALL TOGETHER



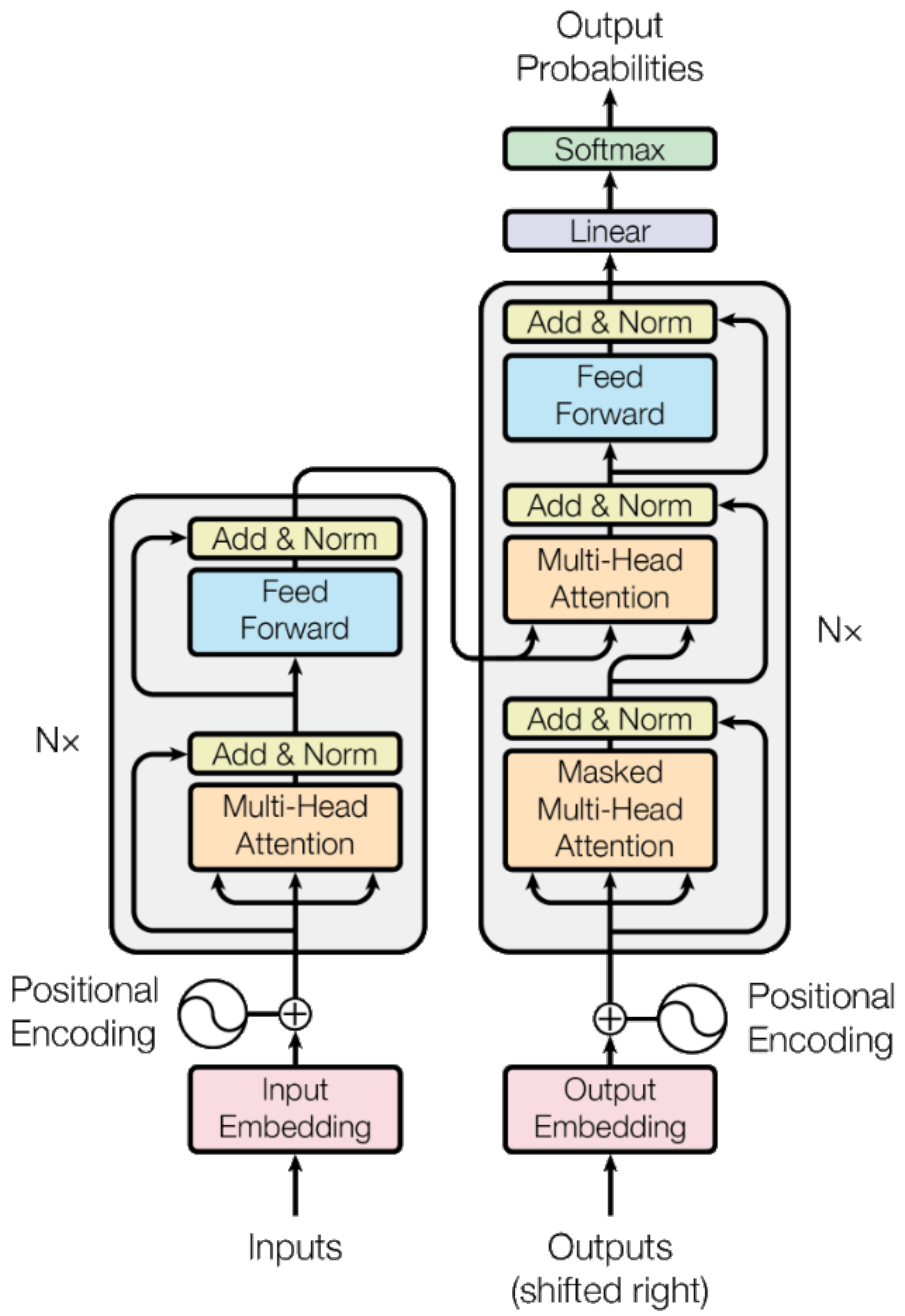
WRAPPING UP

SUMMARY

-Transformers are powerful text predictors

- Much more efficient than previous methods
- Based on self-attention and MLPs
- Are slowly also taking over other domains

-Type of tokenization has significant impact on performance



5 MIN BREAK

Then Exercises

NNs on GPU by Group 1 (David, Jakob, Robin)

HEICO ENTRY IS NOW ONLINE

-Please register at your earliest convenience

-Direct link: https://heico.uni-heidelberg.de/heiCO/ee/ui/ca2/app/desktop/#/slc.tm.cp/student/courses/367254?$scrollTo=toc_overview

THIS WEEKS EXERCISE

NNs on GPU by Group 1 (David, Jakob, Robin)

EXERCISE 2

-For small problems the CPU is faster

Communication overhead for GPU

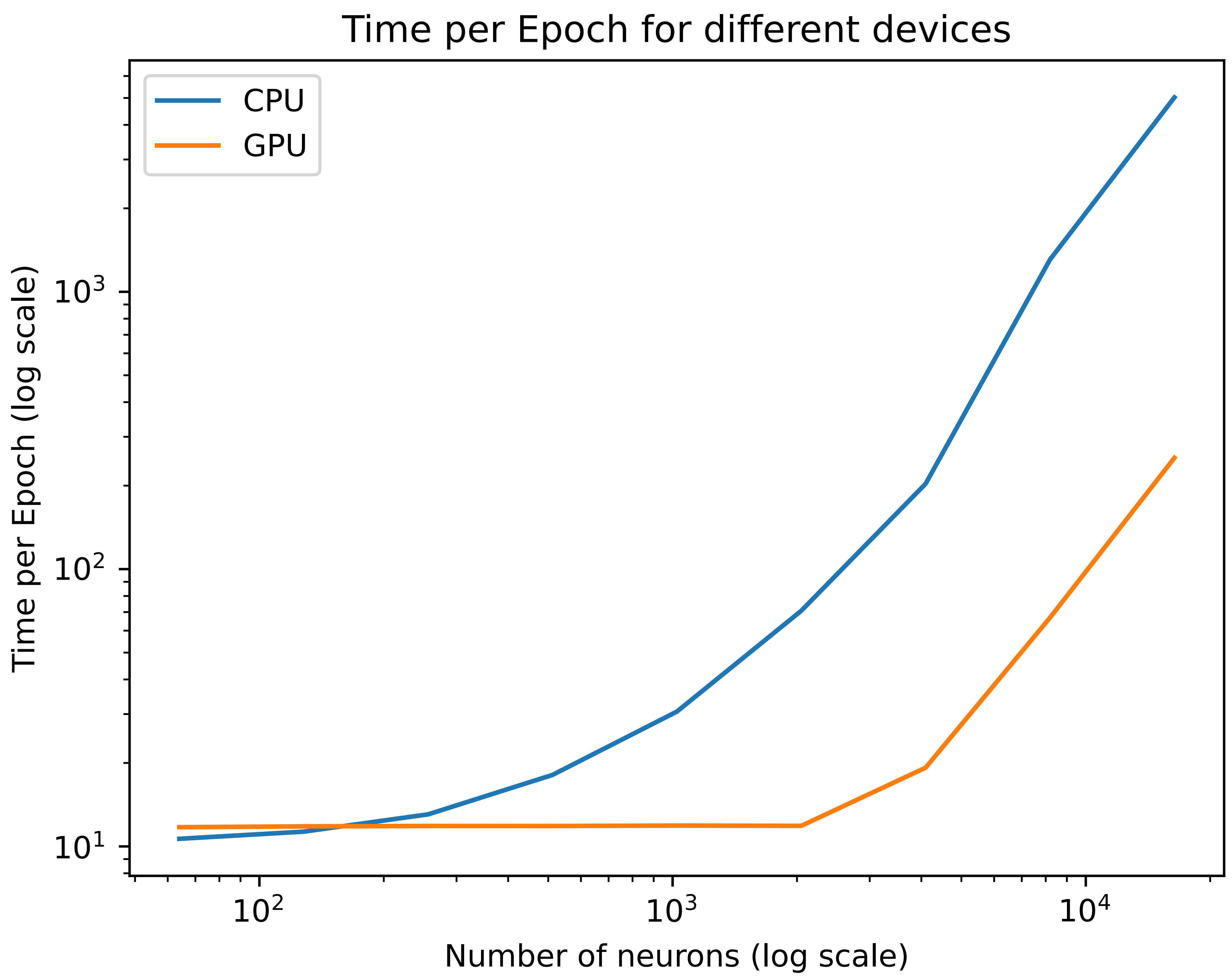
-Maximum speedup heavily depends on system configuration

-Approximate Expected speedup:

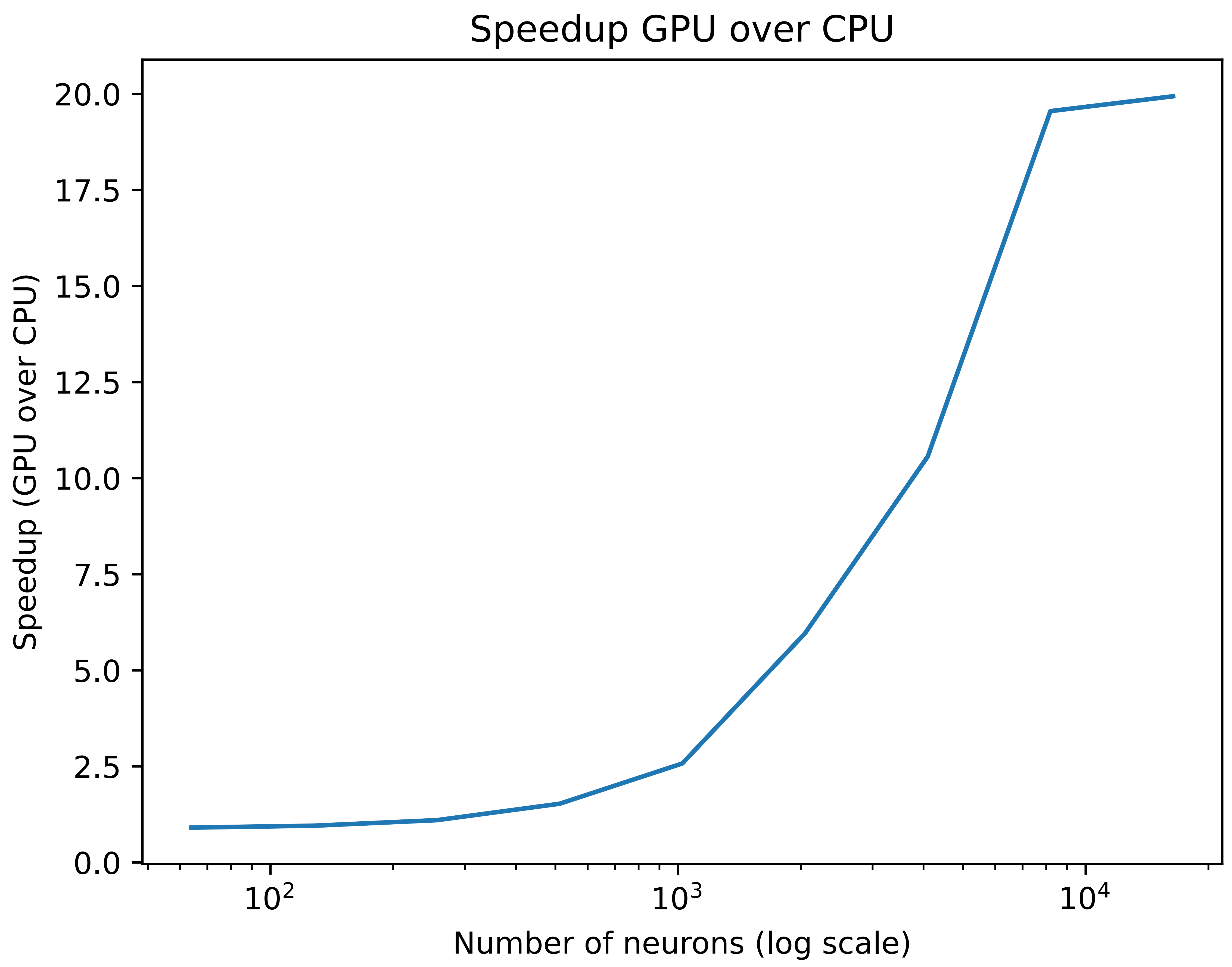
Lecture system (Brook): 20x

Research system (Rivulet): 50x

EXERCISE 2

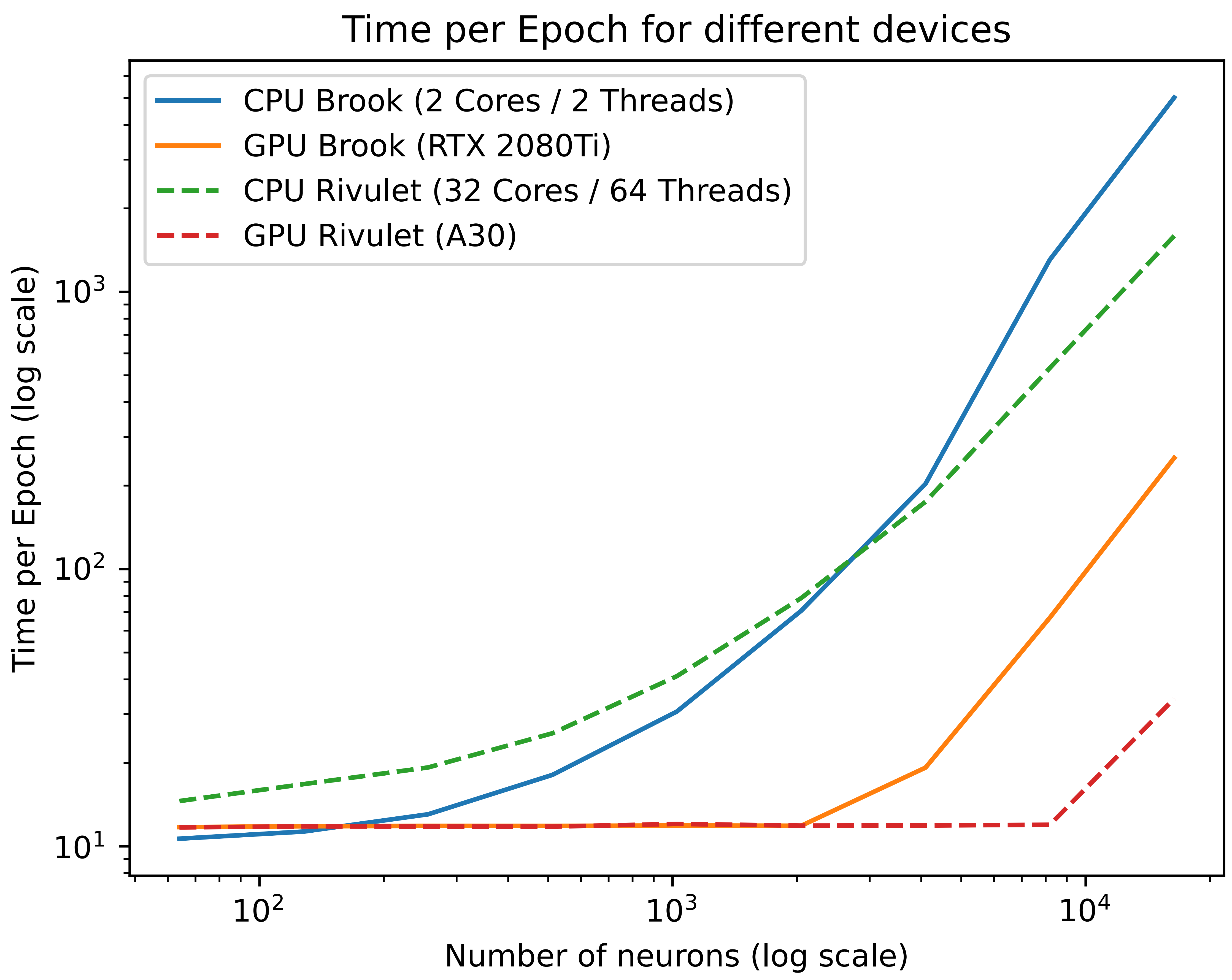


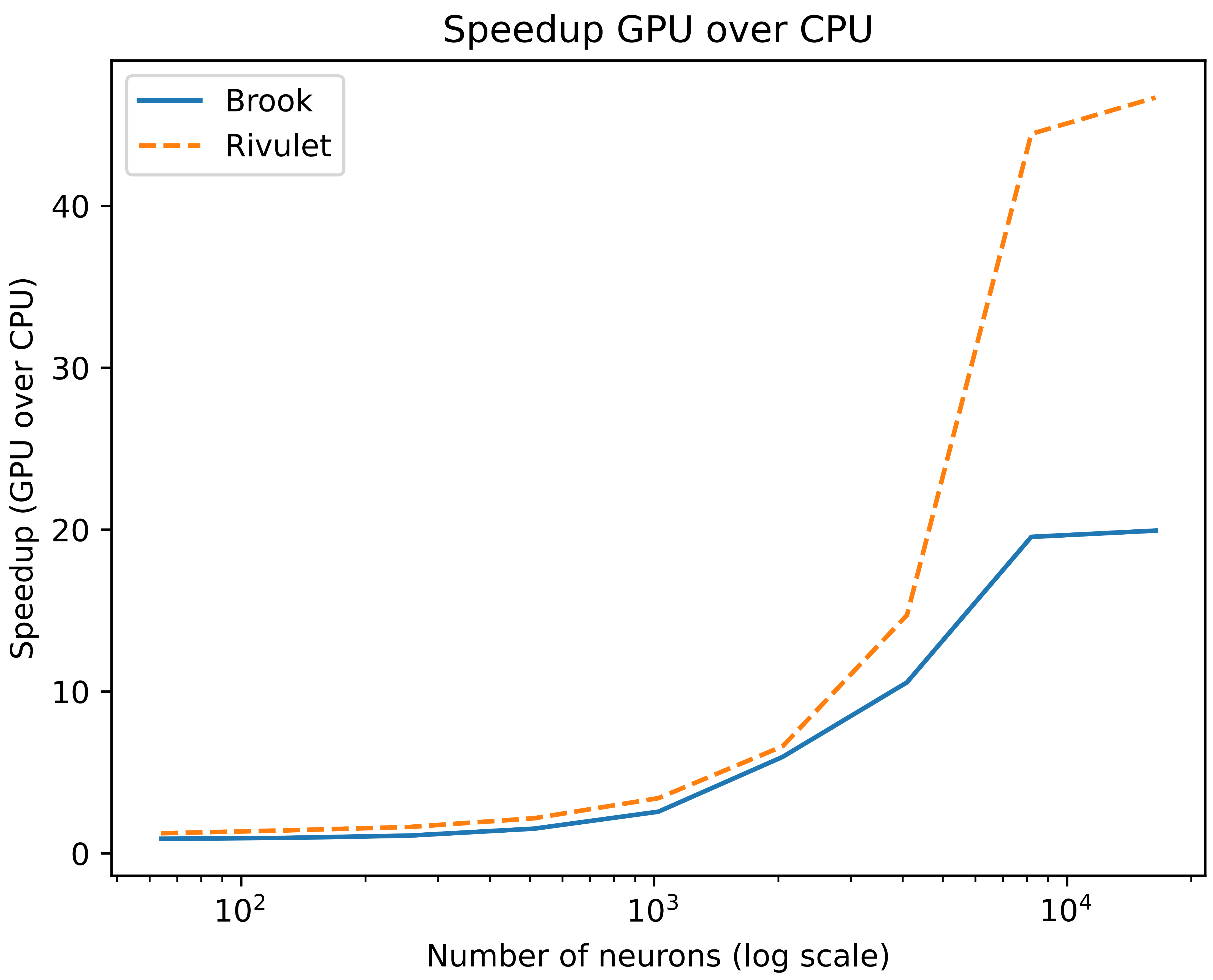
Time per Epoch for different devices
CPU
GPU
Time per Epoch (log scale)
10^3
10^2
10^1
10^2
10^3
10^4
Number of neurons (log scale)



Speedup GPU over CPU
Speedup (GPU over CPU)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
10^2
10^3
10^4
Number of neurons (log scale)

EXERCISE 2


Time per Epoch for different devices
CPU Brook (2 Cores / 2 Threads)
GPU Brook (RTX 2080Ti)
CPU Rivulet (32 Cores / 64 Threads)
GPU Rivulet (A30)
Time per Epoch (log scale)
10^3
10^2
10^1
10^2
10^3
10^4
Number of neurons (log scale)


Speedup GPU over CPU
Brook
Rivulet
Speedup (GPU over CPU)
40
30
20
10
0
10^2
10^3
10^4
Number of neurons (log scale)

NEXT WEEKS EXERCISE

NEXT WEEKS EXERCISE

- Transformer Paper reading
    - Attention is all you need
- Choice of one
    - An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
    - Tokenizer Choice For LLM Training: Negligible or Crucial?
- **Submission deadline: Tuesday 09:00 am**

- https://csg.ziti.uni-heidelberg.de/teaching/ap_nn_from_scratch_materials/