

Transformer: A Quick View

Spring 2026

Lecturer: Yuedong (Steven) Xu

Fudan University

ydxu@fudan.edu.cn

Disclaimer

Machine learning systems is a broad and rapidly evolving field. The course material has been developed using a broad spectrum of resources, including research papers, lecture slides, blogposts, research talks, tutorial videos, and other materials shared by the research community. We sincerely appreciate their efforts and assistance, and try our best to cite the sources of the materials used in this course.

Distributed LLM Training: Outline

- **Transformer: A Quick View**
- Data Parallelism
 - Parameter-Server
 - All-Reduce
 - Memory Optimization

Transformer Overview

- Transformer everywhere
 - Basic object in Transformer library: ***pipeline()*** which connects a model with its necessary preprocessing and postprocessing steps

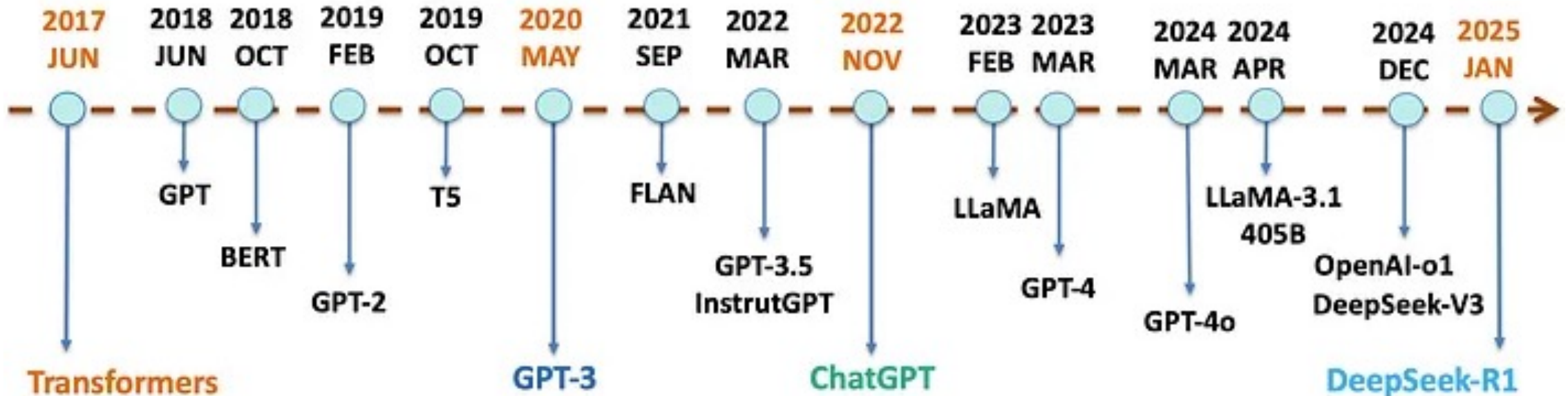
```
from transformers import pipeline
classifier = pipeline("sentiment-analysis")
classifier("I've been waiting for a machine learning system course my whole life.")

[{'label': 'POSITIVE', 'score': 0.9598047137260437}]
```

- Other pipelines
 - *text-generation, text-classification, summarization, translation, feature-extraction*
 - *image-to-text, image-classification, object-detection*
 - *automatic-speech-recognition, text-to-speech, audio-classification*

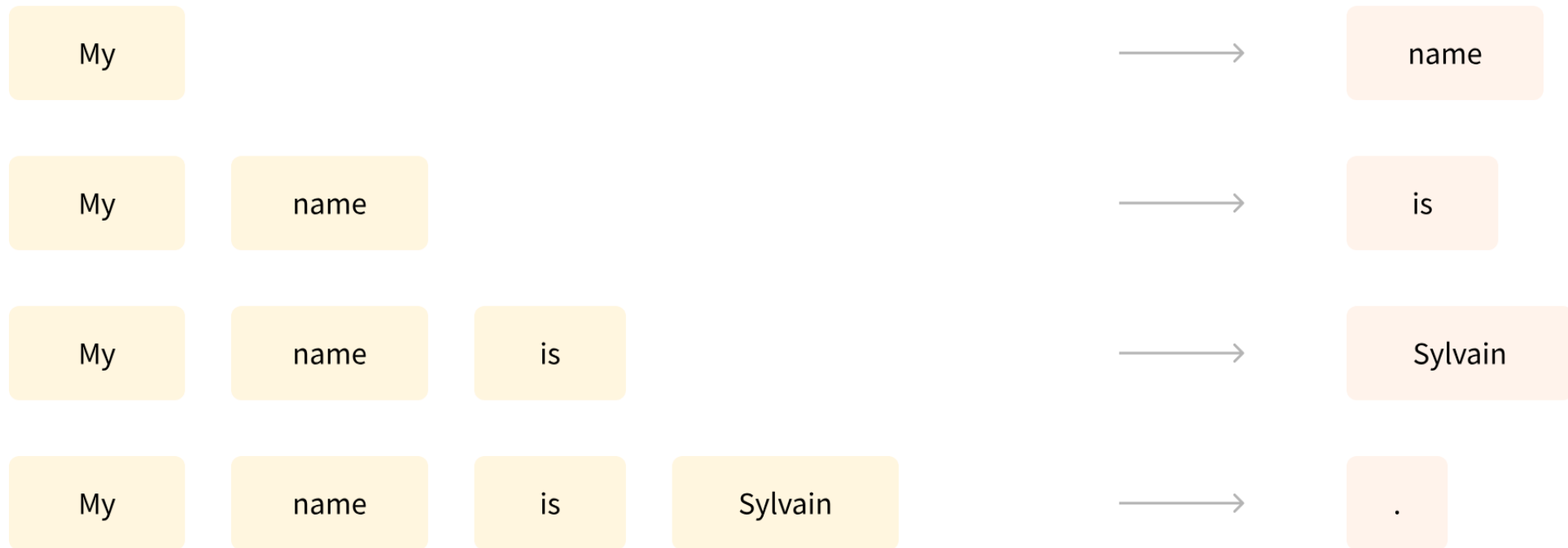
Transformer Overview

- A brief history of Transformer models



Transformer Overview

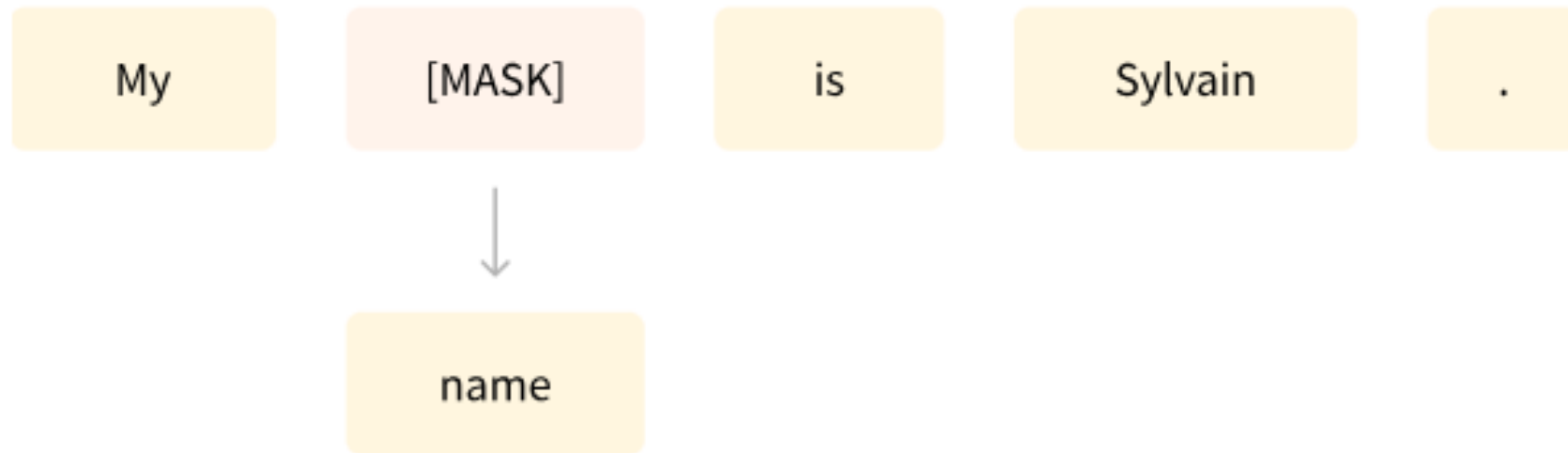
- Causal language model



Predicting the next word in a sentence having read a number of previous words

Transformer Overview

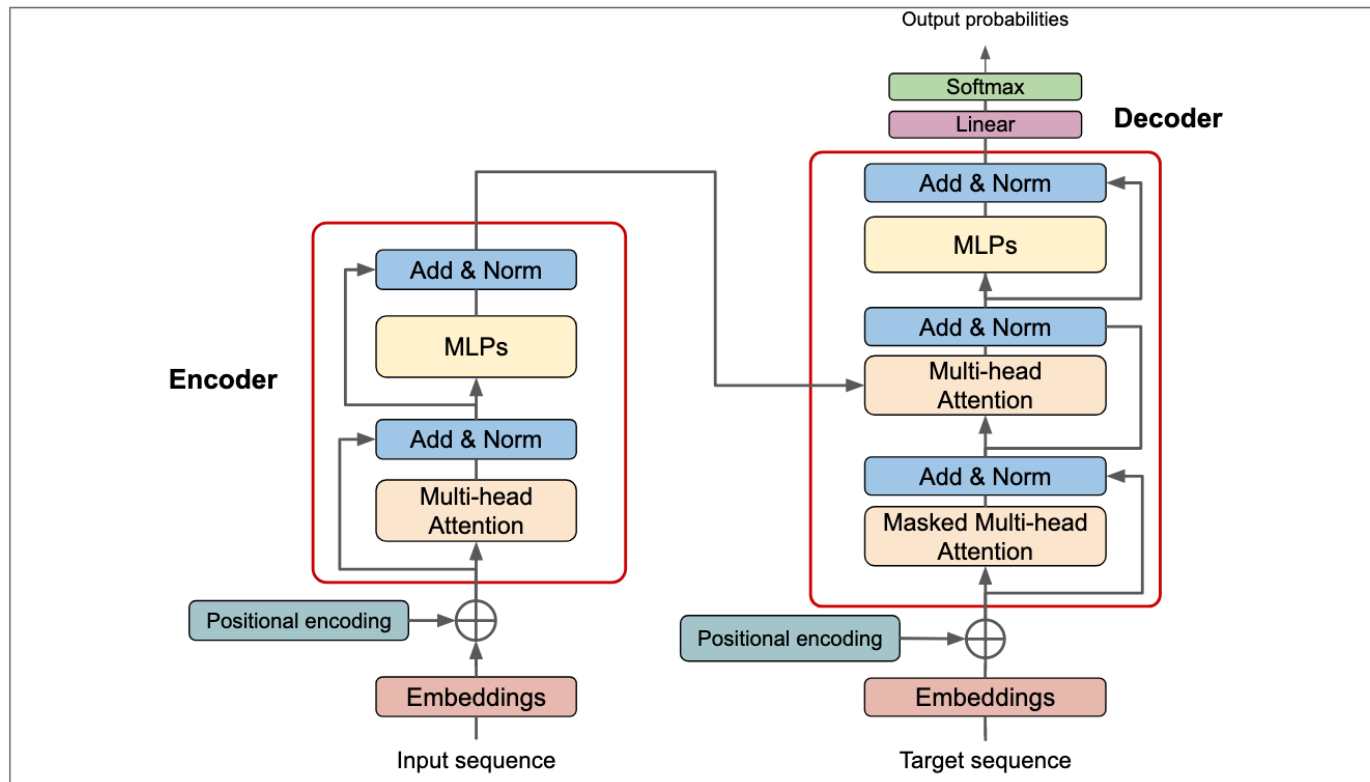
- Non-causal language model



Masked language modeling in which the model predicts a masked word in the sentence

Transformer Overview

- Transformer architecture



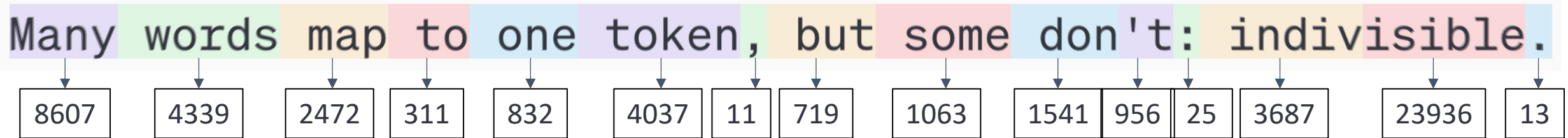
Encoder (left): receiving an input and building a representation of it (its features), i.e. acquire understanding from the input.

Decoder (right): utilizing the encoder's representation along with other inputs to generate a target sequence, i.e. generating outputs.

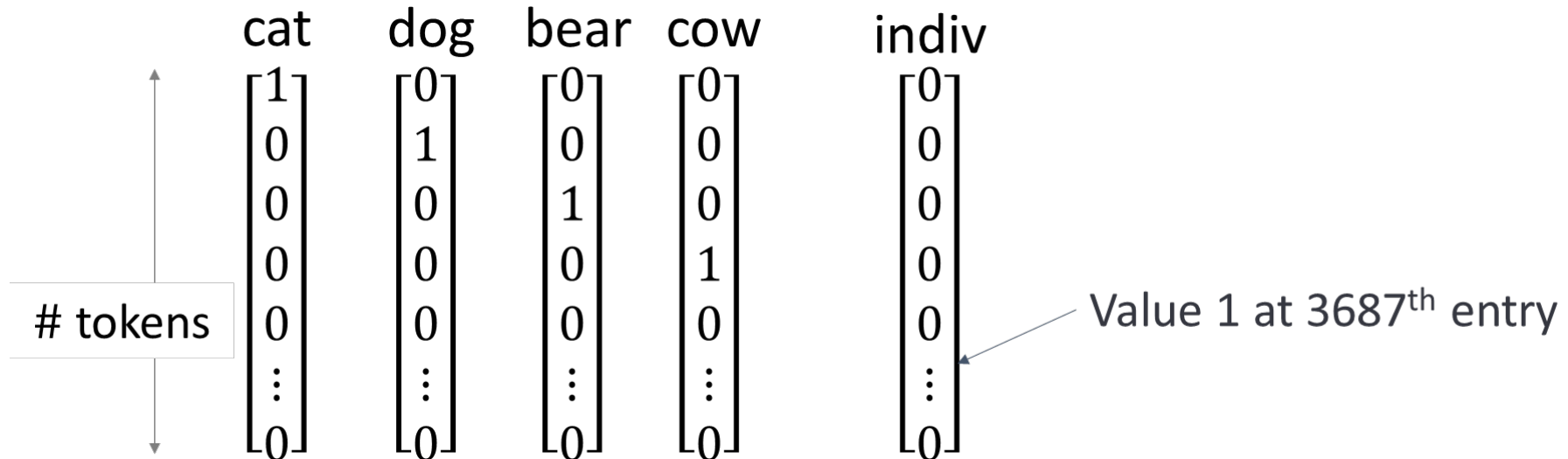
A minimalist description of Transformer *

Transformer Overview

- Sequence to tokens



One-hot encoding



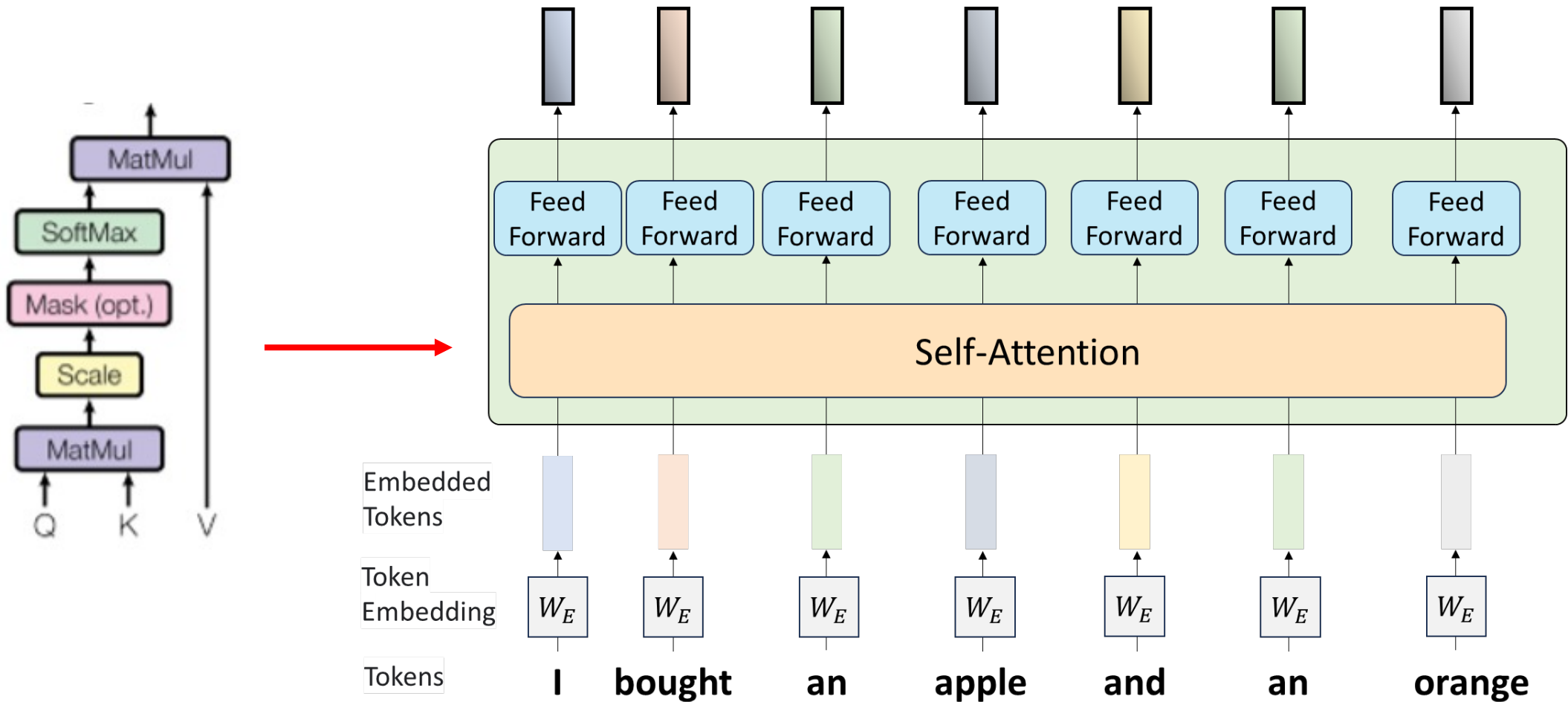
Transformer Overview

- Token embedding
 - Compressing a high dimensional vector into a lower one
 - Multiplying one-hot encoded vector by embedding matrix (index lookup)

$$\begin{array}{c} \begin{array}{c} \uparrow \\ d \\ \downarrow \end{array} \begin{bmatrix} 0.5 \\ 2.7 \\ 1.2 \\ \vdots \\ 0.2 \end{bmatrix} \\ \text{Embedded token} \end{array} = \begin{array}{c} \begin{array}{c} \uparrow \\ d \\ \downarrow \end{array} \begin{bmatrix} \begin{array}{c} \leftarrow \text{total \# tokens} \rightarrow \\ \mathbf{W} \\ \mathbf{E} \end{array} \\ \text{Embedding Matrix} \end{bmatrix} \begin{bmatrix} \text{dog} \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{array}$$

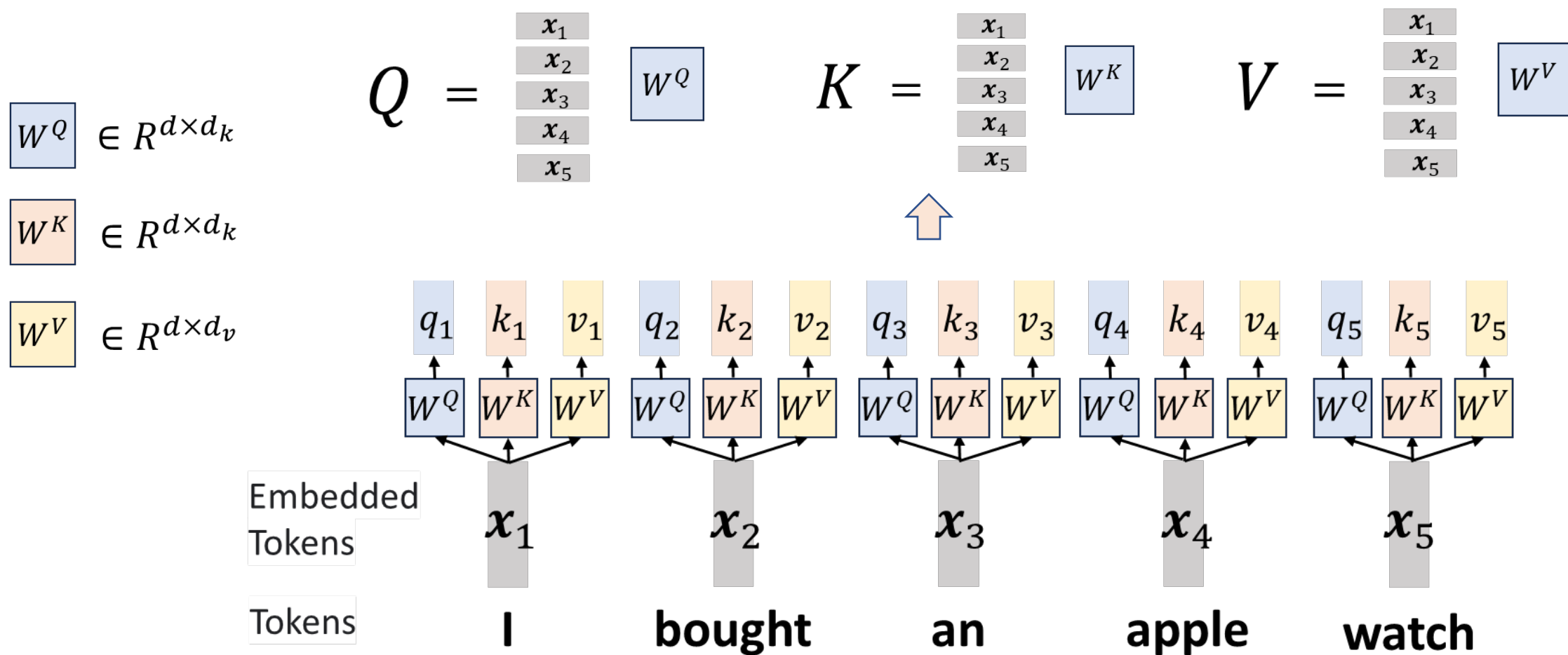
Transformer Overview

- Attention computing



Transformer Overview

- Attention computing

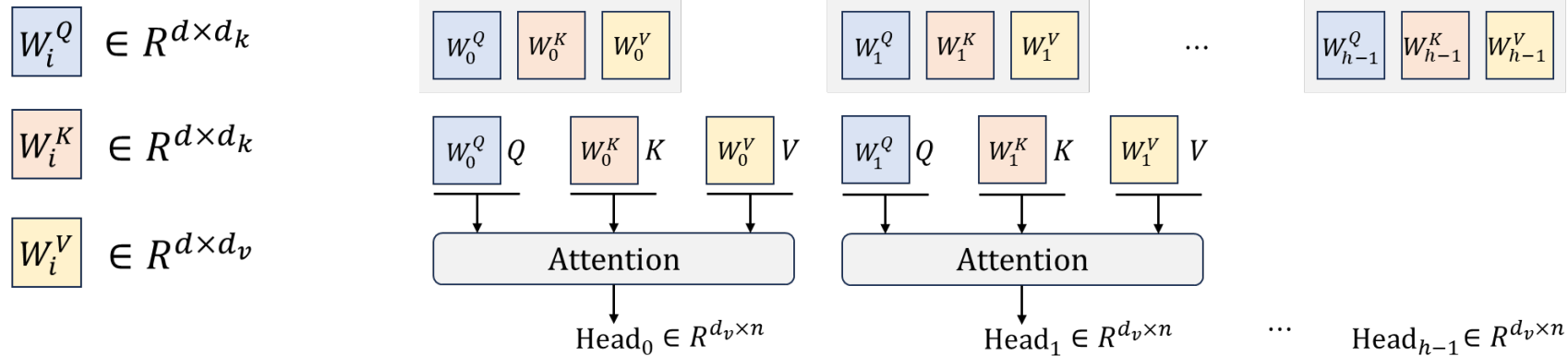


Transformer Overview

- Attention computing
 - Single head ($d_k = d$)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{K^\top Q}{\sqrt{d_k}}\right) V$$

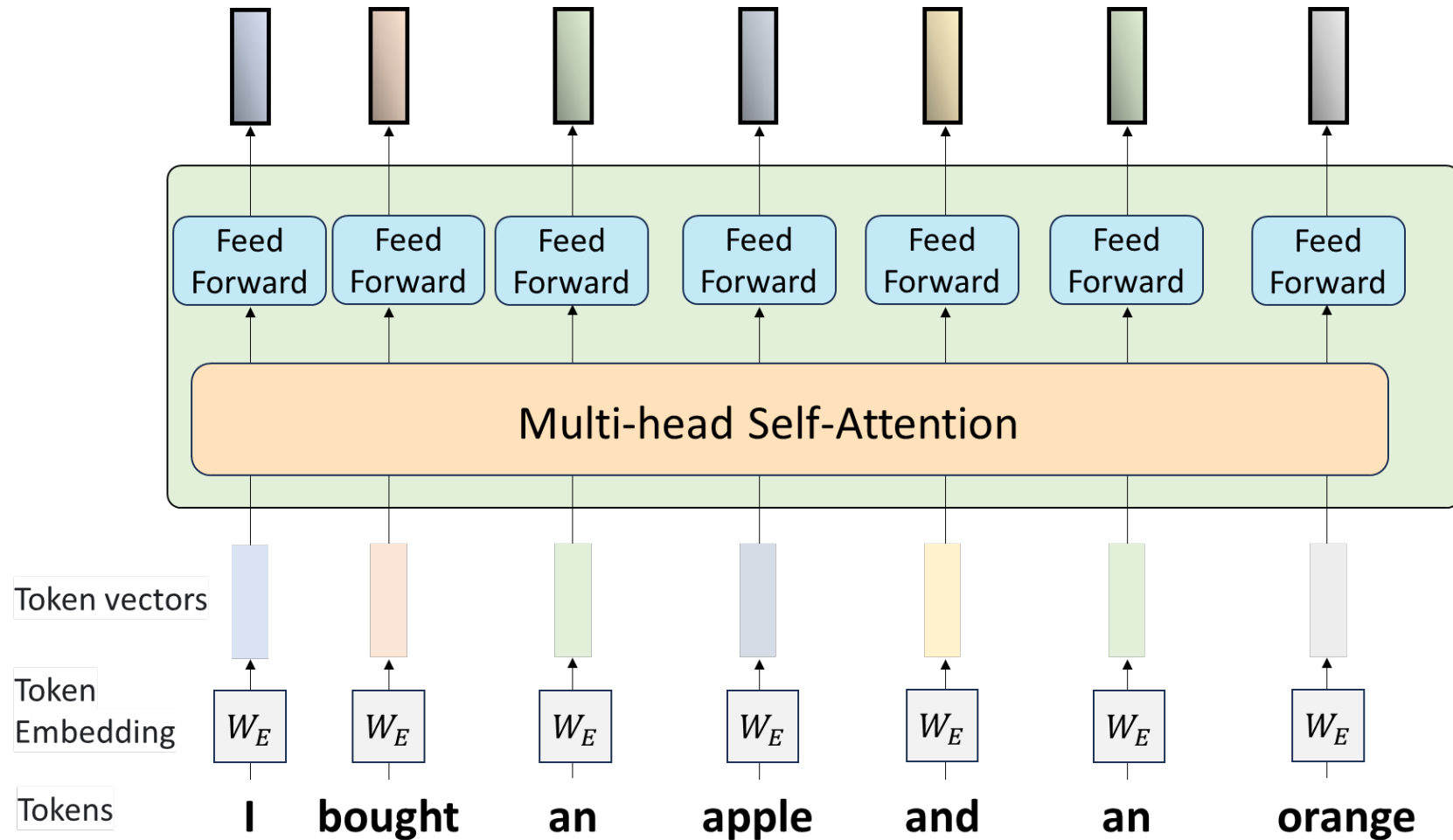
- Multi-head ($d_k = d/h$)



$$\text{MultiHeadedAttention}(Q, K, V) = W^O \begin{bmatrix} \text{Head}_0 \\ \text{Head}_1 \\ \vdots \\ \text{Head}_{h-1} \end{bmatrix}$$

Transformer Overview

- FFN computing

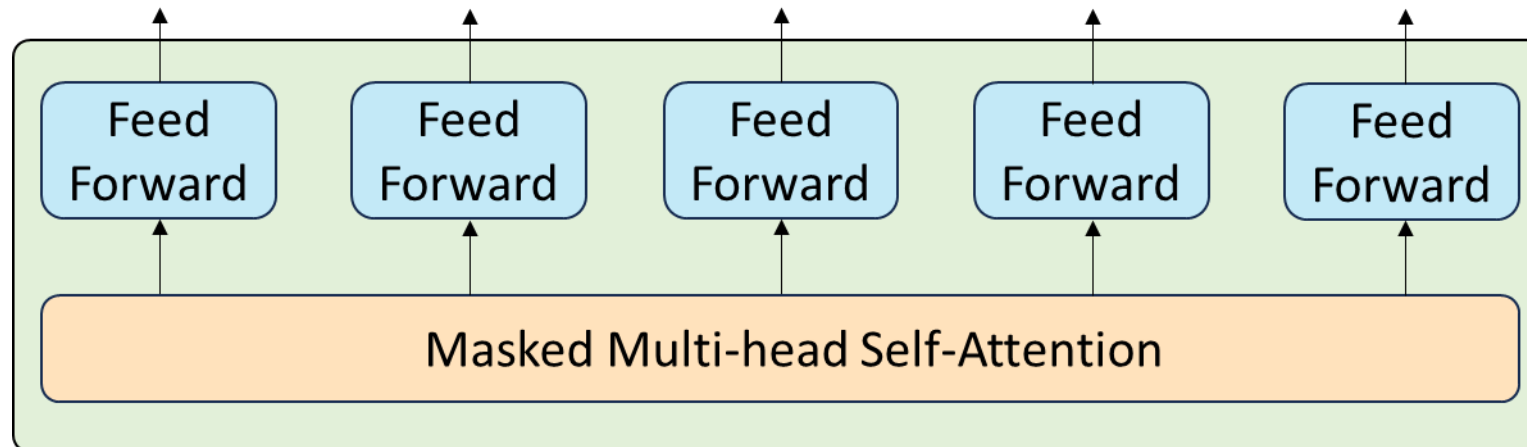
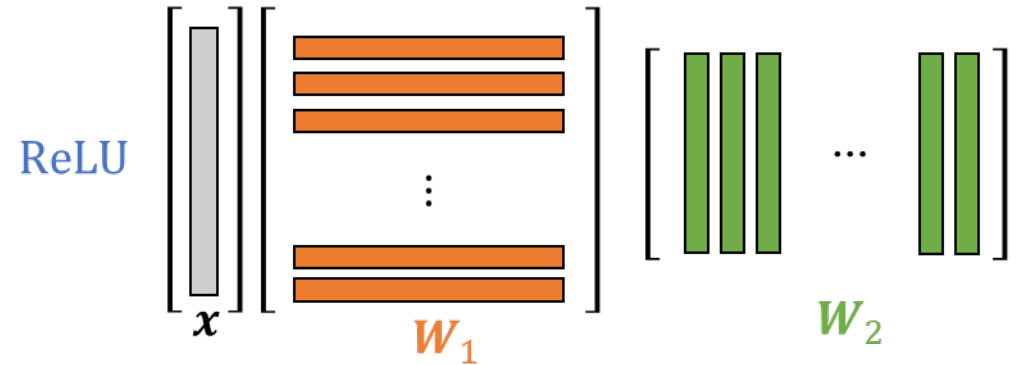


* LayerNorm and residual omitted; Slides from Prof. Kai-Bin Huang

Transformer Overview

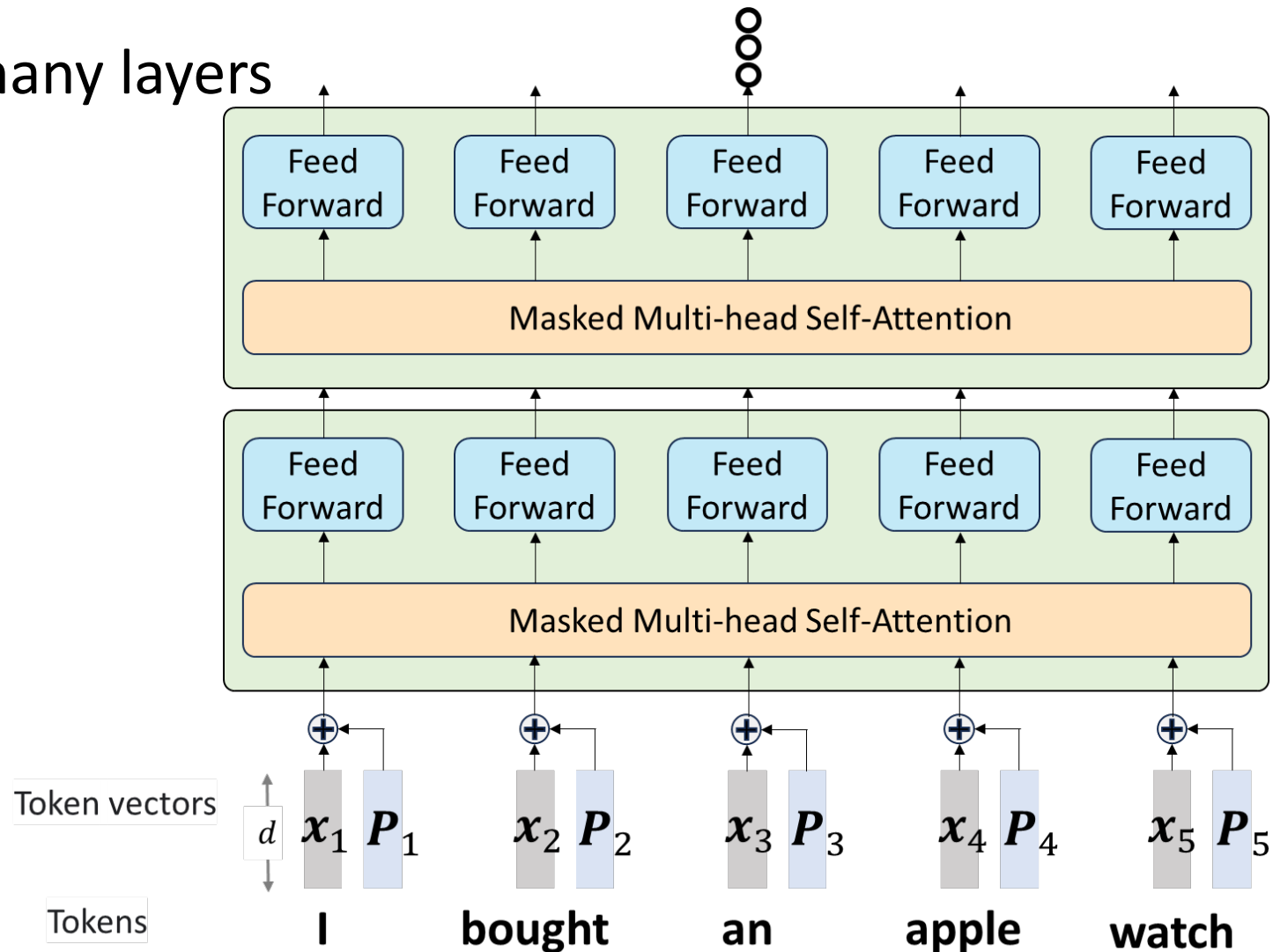
- FFN computing

$$FFN(\mathbf{x}) = \text{ReLU}(\mathbf{x}W_1 + \mathbf{b}_1)W_2 + \mathbf{b}_2$$



Transformer Overview

- Stacking many layers



* LayerNorm and residual omitted; Slides from Prof. Kai-Bin Huang

Transformer Overview

- Encoder versus decoder

Encoder:
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

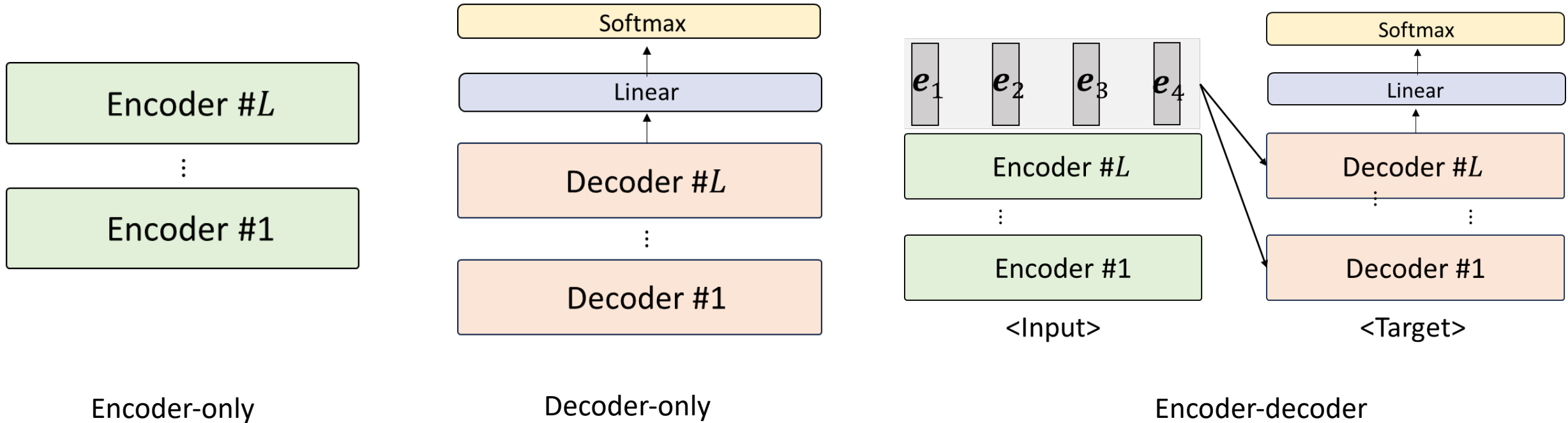
Decoder:
$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V$$

$M =$

0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	0	$-\infty$	$-\infty$	$-\infty$
0	0	0	$-\infty$	$-\infty$
0	0	0	0	$-\infty$
0	0	0	0	0

Transformer Overview

- Encoder-only, decoder-only and encoder-decoder



Transformer Overview

- Output
 - Linear projection

$$\text{Logits} = x \cdot W^T + b$$

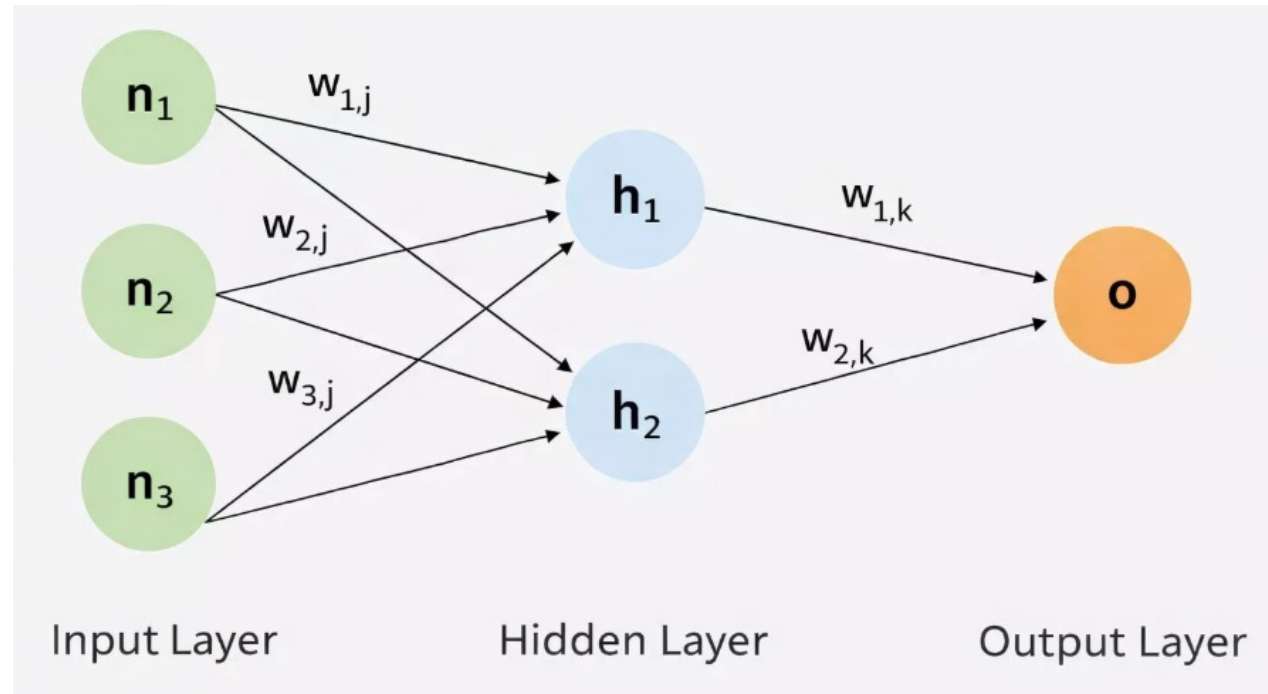
- Softmax to probability distribution

$$P(w_i) = \frac{e^{z_i}}{\sum_{j=1}^V e^{z_j}}$$

- Decoding/Sampling to predict a word

DNN Computation

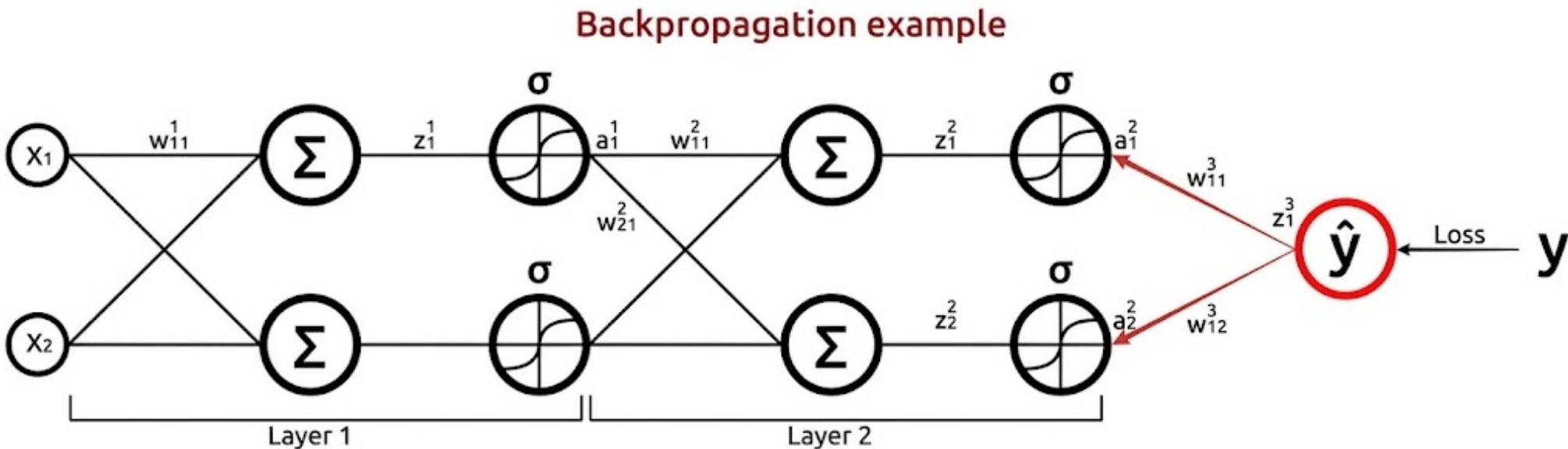
- Forward propagation



Forward propagation

DNN Computation

- Backward propagation



$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial z_1^3} \left[\frac{\partial z_1^3}{\partial a_1^2} \frac{\partial a_1^2}{\partial z_1^2} \frac{\partial z_1^2}{\partial a_1^1} + \frac{\partial z_1^3}{\partial a_2^2} \frac{\partial a_2^2}{\partial z_2^2} \frac{\partial z_2^2}{\partial a_1^1} \right] \frac{\partial a_1^1}{\partial z_1^1} \frac{\partial z_1^1}{\partial w_{11}^1}$$

Distributed LLM Training: Outline

- What to memorize
 - Computation flow instead of functionalities of different modules
 - Computational cost and memory consumption
- What to know beforehand
 - Block matrix multiplication
 - Some basic software, hardware and networking knowledge

Why do we need “distributed training”?

Quantitative Analysis

GPT-175B MODELS: PARAMETERS

EMBEDDING LAYER



• n_{vocab} : size of vocabulary



• n_{model} : model dimension

$$n_{embed} = n_{vocab} \times n_{model}$$

MULTI-HEAD ATTENTION

→ • d_k : dimension of head Q and K

→ • d_v : dimension of head V



• n_{heads} : number of heads



• n_{layers} : number of Transformer layers



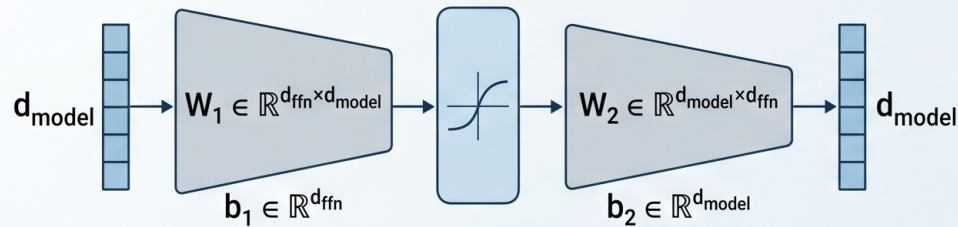
• d_{head} : typical head dimension
(typically equal to d_k and d_v)

$$W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{model} \times d_{head}},$$
$$\text{and } W^O \in \mathbb{R}^{d_{model} \times d_{head}}$$

Quantitative Analysis

GPT Models: Parameter Counting

Multi-Layer Perceptron (MLP) Parameter Breakdown



d_{ffn} : Hidden Dimension of Fully Connected Layer

Parameters: $W_1 \in \mathbb{R}^{d_{\text{ffn}} \times d_{\text{model}}}$, $b_1 \in \mathbb{R}^{d_{\text{ffn}}}$, $b_2 \in \mathbb{R}^{d_{\text{model}}}$

$$n_{\text{MLP}} = 2d_{\text{ffn}} \times d_{\text{model}} + d_{\text{ffn}} + d_{\text{model}}$$

Simplified Total Parameter Count (Excluding Biases)

$$n_{\text{total}} = n_{\text{vocab}} d_{\text{model}} + n_{\text{layers}} (4d_{\text{model}}^2 + 2d_{\text{ffn}} d_{\text{model}})$$

*This formula counts key parameter matrices and input token embeddings, excluding all biases and positional embeddings.

Simplified Total Parameter Count (Excluding Biases)



(Variable d_{model} is used for consistency)

$$n_{\text{total}} = n_{\text{vocab}} d_{\text{model}} + n_{\text{layers}} (4d_{\text{model}}^2 + 2d_{\text{ffn}} d_{\text{model}})$$

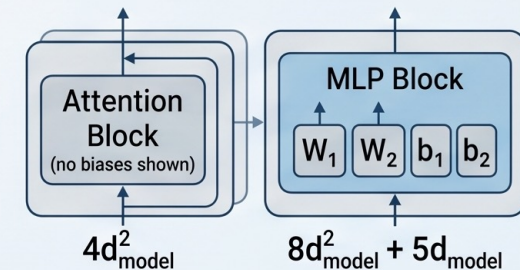
*This formula counts key parameter matrices and input token embeddings, excluding all biases and positional embeddings.

Parameter Estimation for Standard GPT Models

- $d_{\text{model}} = d_{\text{head}} \times n_{\text{heads}}$

- $d_{\text{ffn}} = 4 d_{\text{model}}$

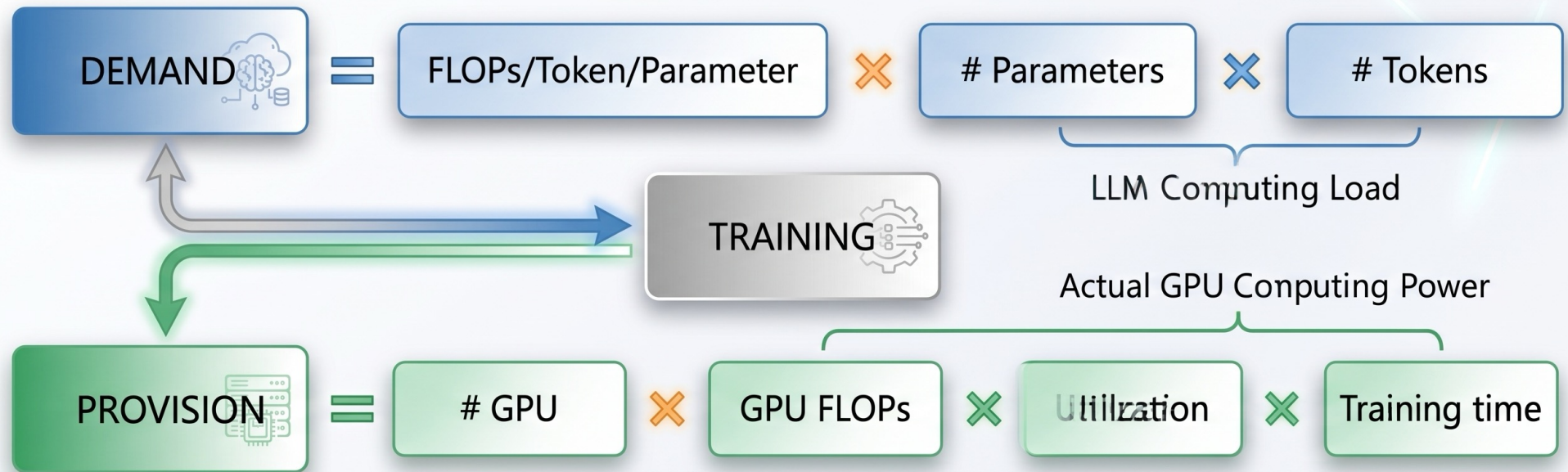
Note: Today's largest models often use different ratios.



$$n_{\text{total}} = n_{\text{vocab}} d_{\text{model}} + n_{\text{layers}} (12d_{\text{model}}^2 + 5d_{\text{model}})$$

Quantitative Analysis

How many GPUs do we need?



$$6 \times 175B \times 300B / 160TFLOPs / 3600 * 24 * 30 = 800 \text{ GPUs Needed}$$

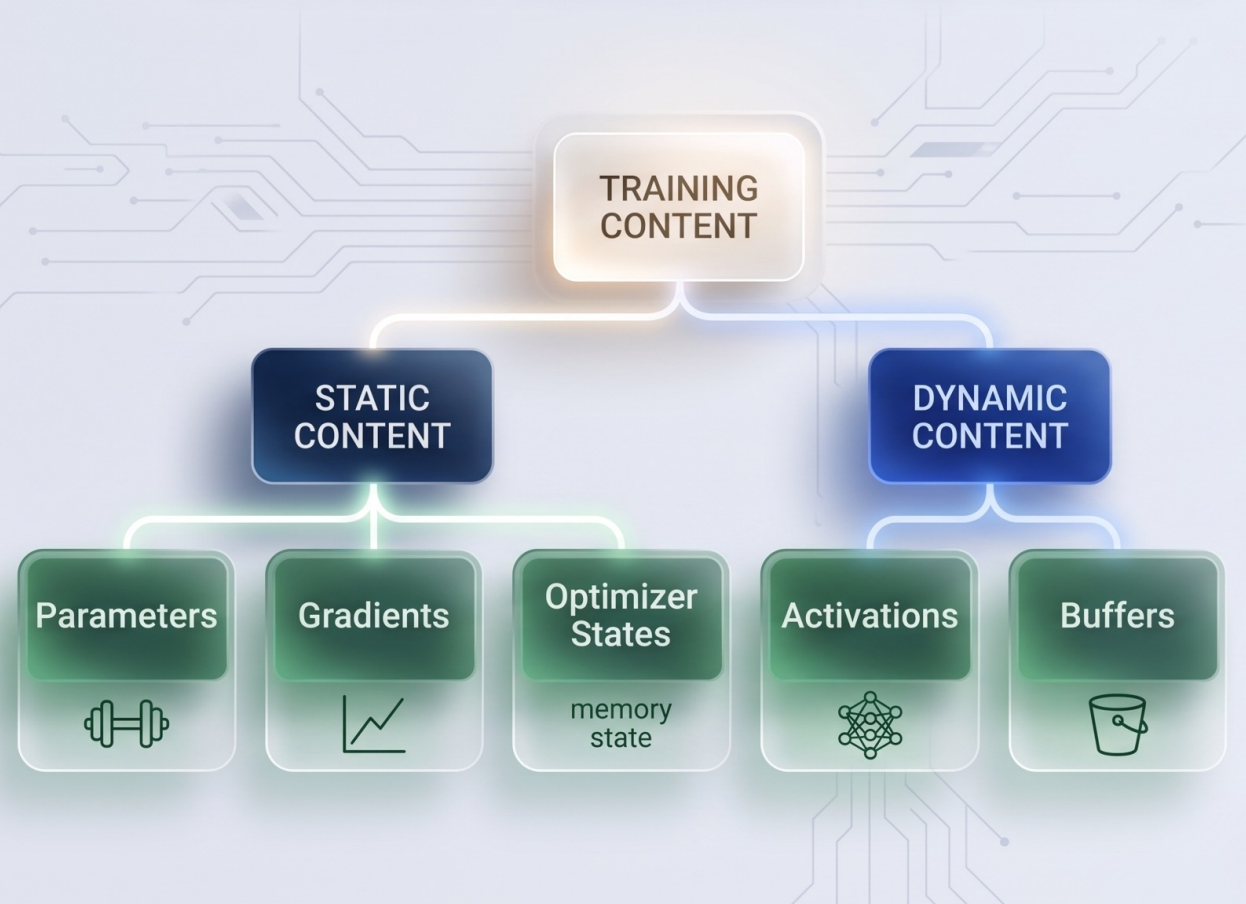
Parameters # Tokens TFLOPs/GPU (Peak) Time (Seconds for 30 Days) Total GPUs Required

《Language Models are Few-Shot Learners》

《Efficient large-scale language model training on GPU clusters using megatron-LM》

Quantitative Analysis


- GPU HBM Content During Training




- An example of GPT-3 175B


- OPTIMIZER STATES**

 32-bit Parameter	700 GB
 Adam Moment	700 GB
 Adam Variance	700 GB

 16-bit Parameter
350 GB

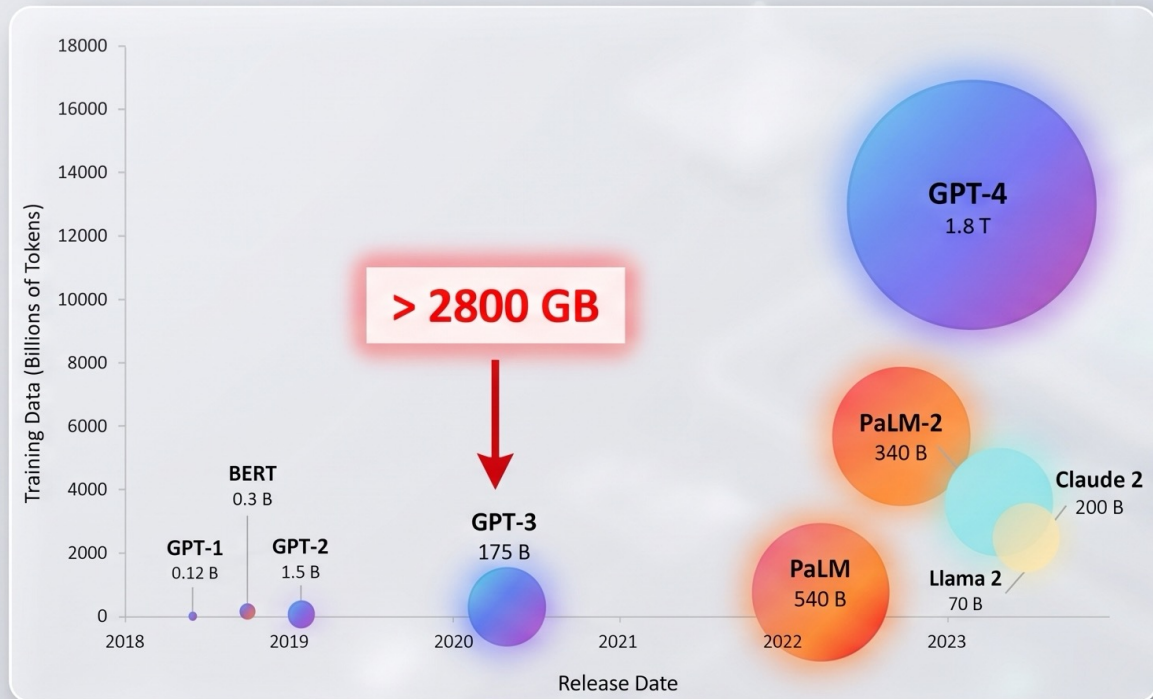
 16-bit Gradient
350 GB

 Activations
(depending on batch size)

 Buffer and
Fragmentation

Quantitative Analysis

- **GPT-175B models: storage**



Models become larger and larger

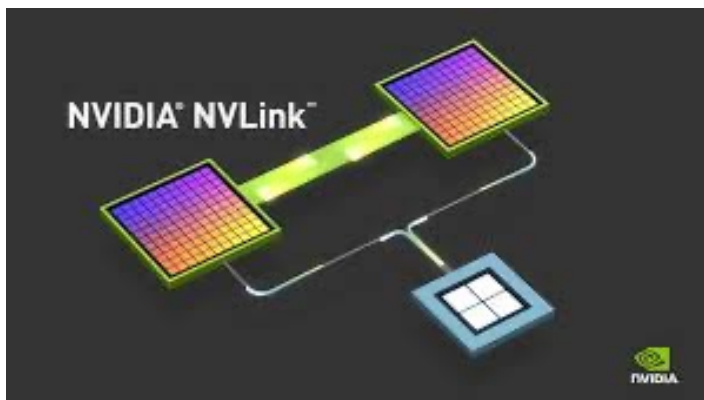
GPU	Memory	Type
Tesla P100	16GB	HBM2
Tesla V100	32GB	HBM2
Tesla A100	40/80GB	HBM2E
Tesla A800	80GB	HBM2E
Tesla H100	80GB	HBM3
Tesla H800	80GB	HBM3

GPU HBM size remains small

Distributed Training

- Multiple GPUs compute collaboratively
 - Data Parallel, Pipeline Parallel, Tensor Parallel, Expert Parallel, Sequence Parallel, Context Parallel

- GPU interconnection



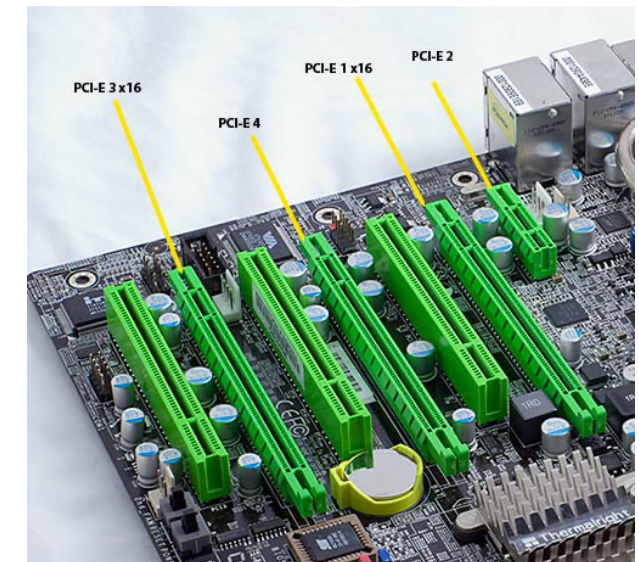
NVLink 5.0 (e.g. 1.8TB/s)



InfiniBand NDR 400Gb/s Single-port PCIe 5.0 x16



ConnectX-7 400GbE Single-port OSFP, PCIe 5.0 x16



PCIe 5.0 (e.g. 128GB/s, 16lanes)

RDMA (RoCE/InfiniBand, e.g. 400Gbps)

Distributed Training

- Importance of Communication

- Traffic volume

- 16-bit gradient of GPT3-175B: 350GB needs to be transmitted in every iteration
 - Even more data communication if model is partitioned

- Imbalanced technology upgrading

- Ampere A100 (FP16 312TFLOPS, released in 2020) → Blackwell B300 (FP8 72PFLOPS, 2025) → Rubin (2026)
 - NVLink 3.0 (600GB/s, 2020) → NVLink 4.0 (900GB/s, 2022) → NVLink 5.0 (1.8TB/s, 2025) → NVLink 6.0 (3.6TB/s, 2026)

Thanks!

