

# Introduction to Transformers

Naeemullah Khan

[naeemullah.khan@kaust.edu.sa](mailto:naeemullah.khan@kaust.edu.sa)



جامعة الملك عبد الله  
للعلوم والتكنولوجيا  
King Abdullah University of  
Science and Technology



LMH  
Lady Margaret Hall

July 3, 2025

AI



CV

$$x \in \mathbb{R}^{nr \times nc}$$

$$\underline{\underline{y}} \in \{1, 2, \dots, c\}$$

NN, CNNs, ViT

NLP

$$x \in \mathbb{R}^d$$

$$y \in \mathbb{R}^{d'}$$

RNN, GRU, LSTM

$d$  is not fixed.

$d'$  is not fixed.

## RNNs

- ① infinite context ✓
- ② actual context window  
is  $\sim 10$  tokens X
- ③ linear complexity wrt  
context. ✓  $O(N)$
- ④ Not parallelizable
- ⑤ inference tree complexity  
is  $O(N)$

## Transformer

- ① fixed context X
- ② actual is full  
context window ✓
- ③ quadratic complexity X
- ④ parallelizable
- ⑤ inference tree complexity  
is  $O(N^2)$

①

Unsupervised

(self supervised learning)

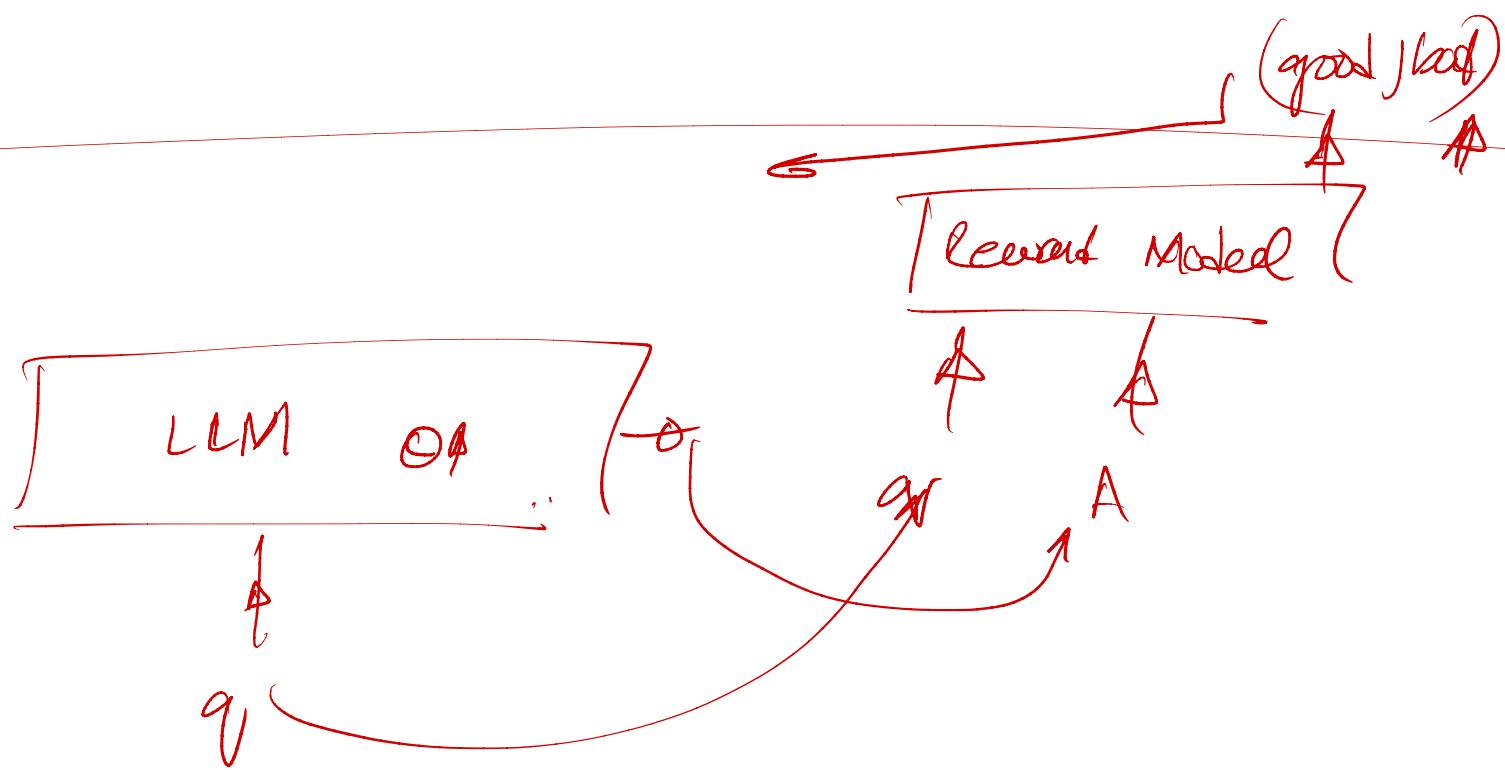


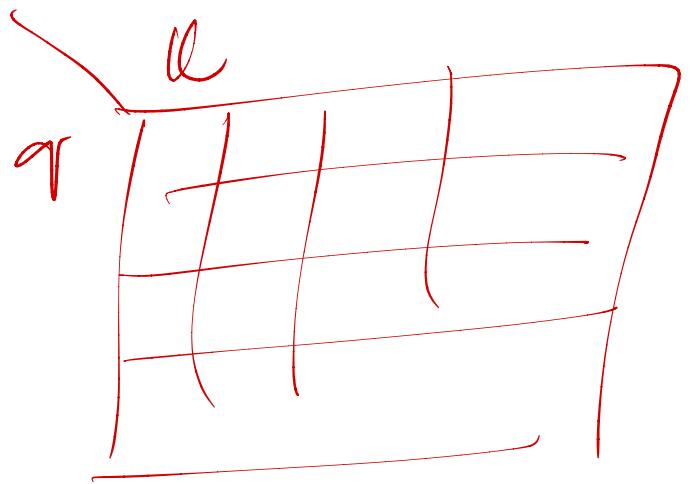
②

SFT

③

RLHP







# Language Modelling

$$\text{iid} \rightarrow P_\theta(x)$$

$$P(x) = \prod_{i=1}^N P_\theta(x_i)$$

$$= \prod_{i=1}^N \prod_{t=1}^{T_i} P_\theta(x_i^t | x_i^{t-1})$$

$$P_x(x) \sim$$

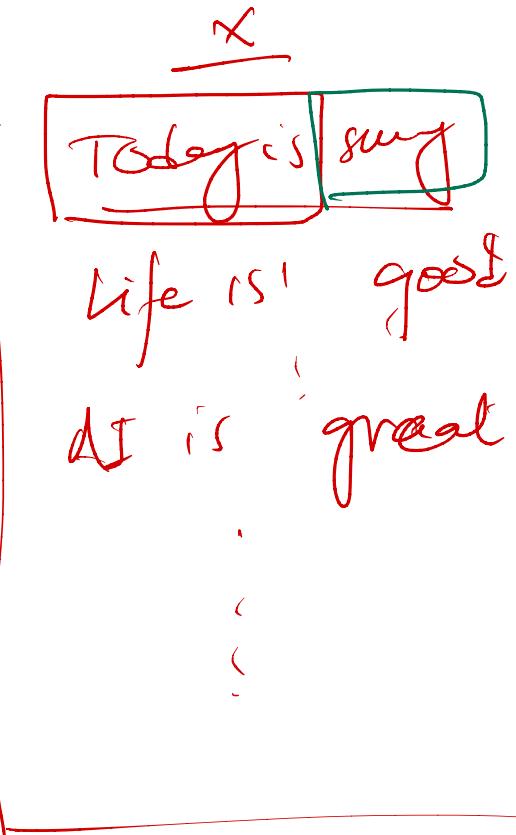
$$\sum_{i=1}^N \sum_{t=1}^{T_i}$$

corpus

$$\log P_\theta(x_i^t | x_i^{t-1})$$

$$P(x, y) = P_1(x) \cdot P_2(y)$$

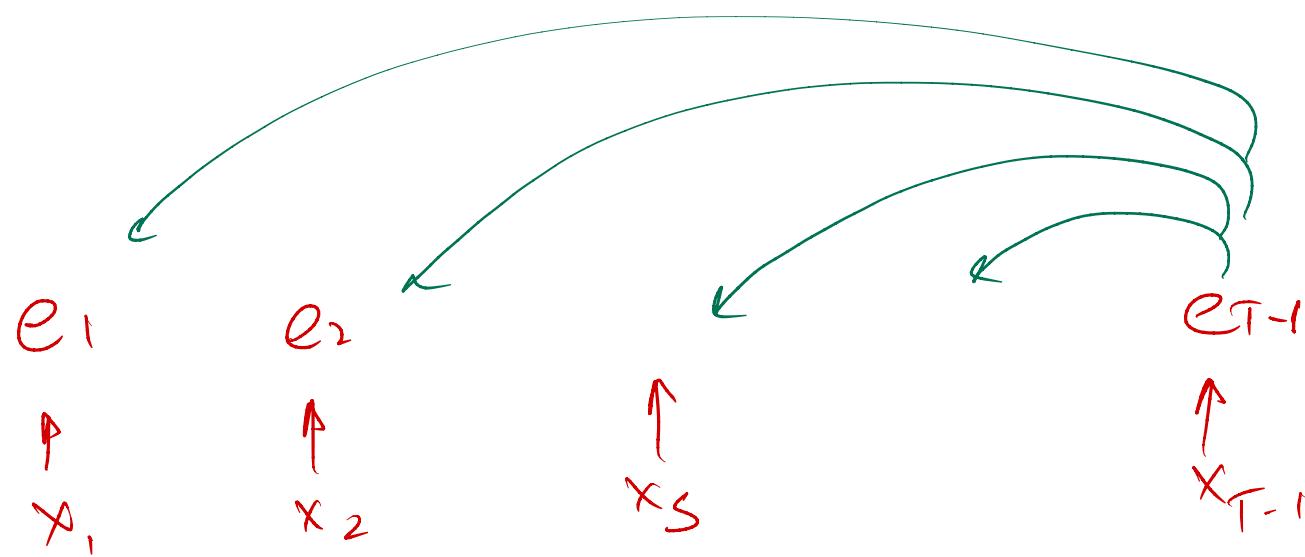
$$= P(x) \cdot P(y)$$



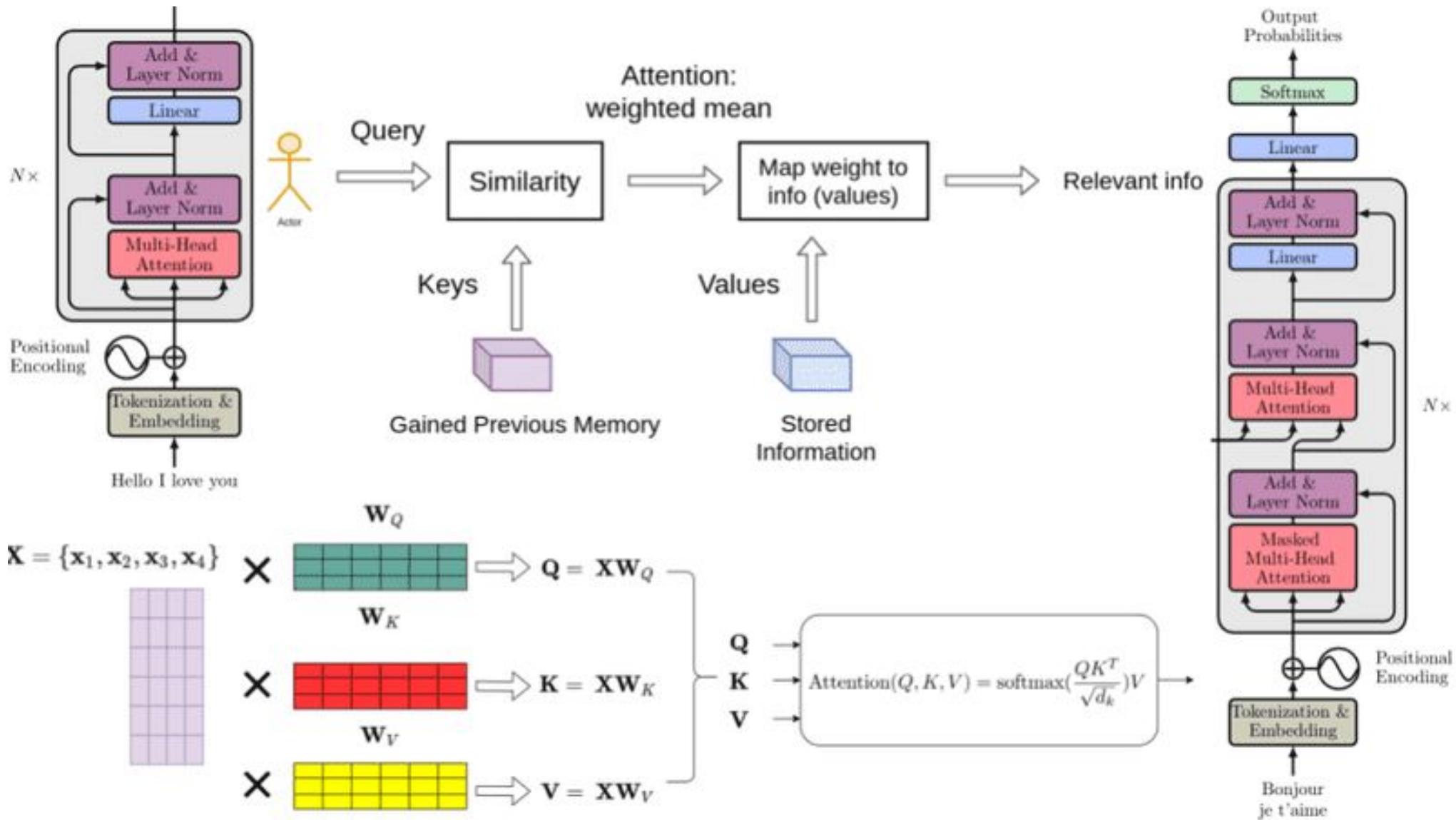
$$P(x_T | x_1, \dots, x_{T-1})$$

NN

$\sim e_{T-1}$



$$p(x_i) = \frac{p(x_i^1) \cdot p(x_i^2 | x_i^1) \cdot p(x_i^3 | x_i^1, x_i^2)}{\cdots p(x_i^T | x_i^1, \dots, x_i^{T-1})}$$



# Table of Contents

1. Motivation
2. Learning Outcomes
3. Introduction to Transformers
4. Self-Attention Mechanism
5. Multi-Head Attention
6. Positional Encoding
7. Transformer Block
8. Encoder-Decoder Architecture
9. Casual Masking
10. Pre-LayerNorm vs. Post-LayerNorm Architectures
11. Layer Normalization vs RMSNorm
12. Transformer Variants
13. Limitations of Transformers
14. Summary

GPT 3

Layers 96

# of heads 96

embedding size 12,228

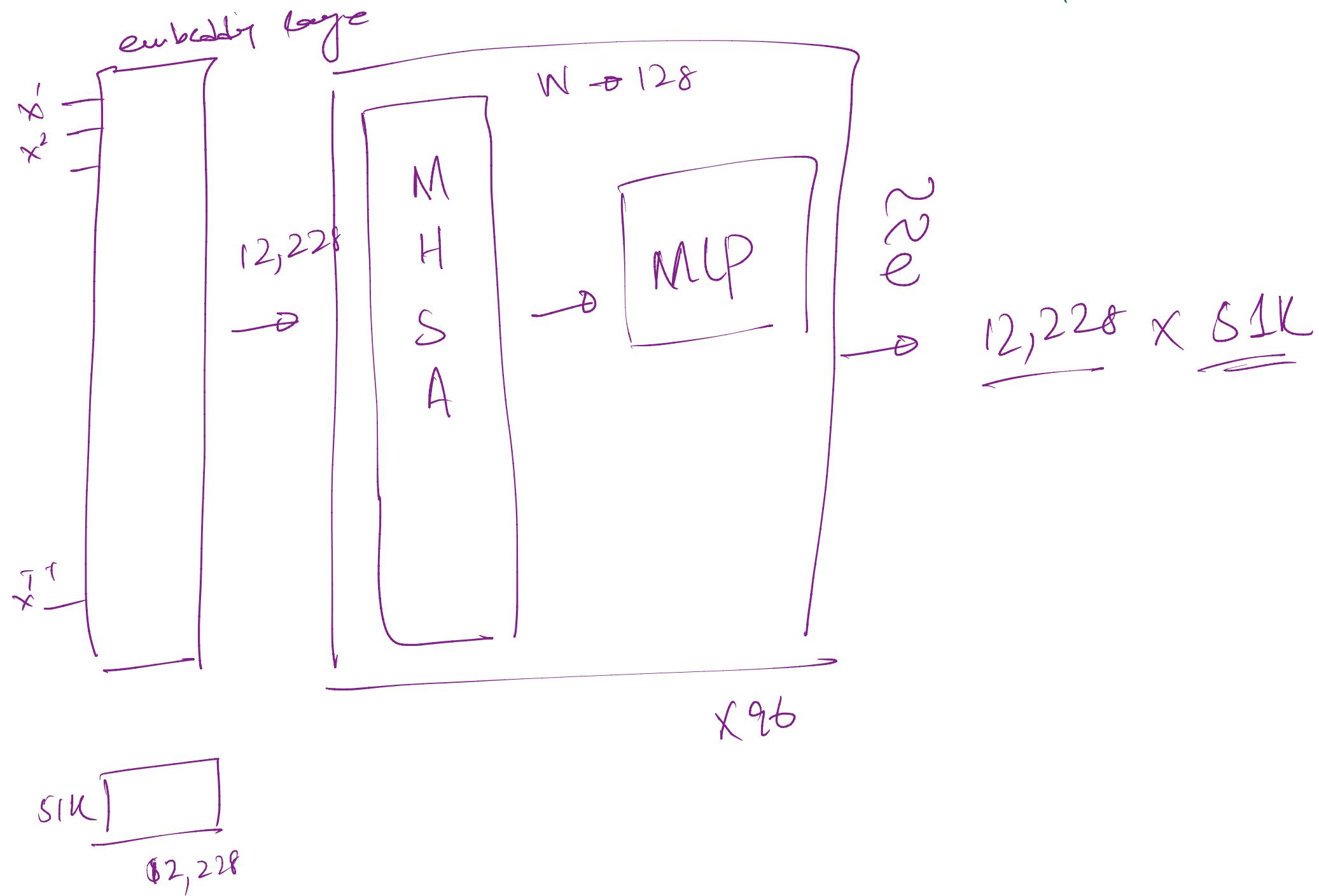
Vocab size ~50,257

# of token ~ 50K

size of embedding 12,228

# GPT3 architecture

$\sim 175$  billion parameters



is  
AI

easy

yes

,

AI

is

very

easy

(easy)

SPÍ 3



all

yes

AI

Embedding layer

$$81K \times 12,228 = 600 \text{ million}$$

$$12,228 \times 128 \times 4 \times 96 = 600 \text{ million}$$

$$\underline{\underline{96 \text{ MHS} \text{ layer}}} = 57 \text{ billion}$$

$$2(12,228 \times 4 \times 12,228) =$$

$$1.2 \text{ billion}$$

96 layer

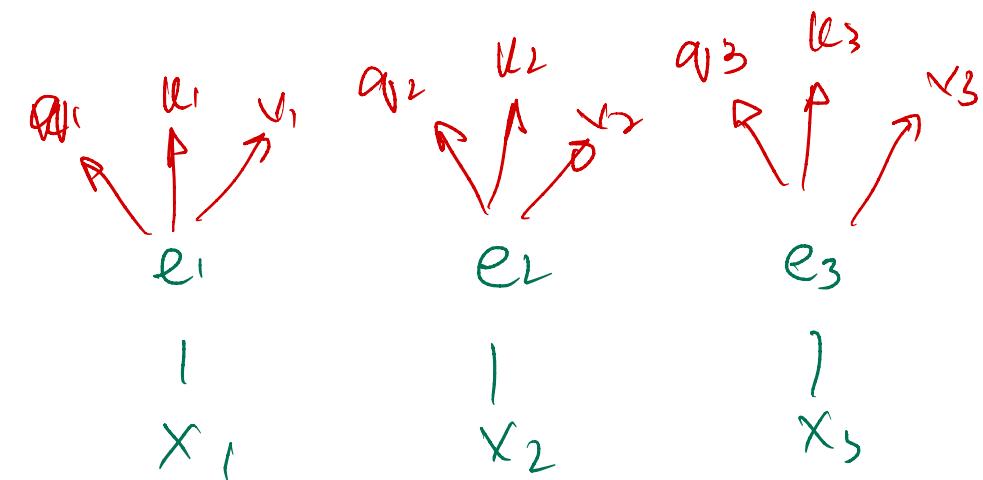
$$115 \text{ billion}$$

output layer  $12,228 \times 81K = 600$  million

# Multi-head Attention

$$\Delta e_{t+1} = \left( \sum_{i=1}^{H-1} \sum_{j=1}^{T-1} \frac{e^{\frac{(q_{k_{t+1}}^k, k_i^k)}{N\delta T}}}{\sum_j e^{\frac{(q_{k_{t+1}}^k, k_j^k)}{N\delta T}}} \right) x_i^k O_{t+1}$$

$\tilde{e}_{t+1} = e_{t+1} + \Delta e_{t+1}$



$$\begin{aligned}
 & \overset{2}{q_{T+1}} \quad \overset{2}{k_{T+1}} \quad \overset{2}{v_{T+1}} \quad \overset{2}{o_{T+1}} \\
 & \overset{1}{q_1} \quad \overset{1}{k_1} \quad \overset{1}{v_1} \quad \overset{1}{o_1} \\
 & \text{Wq} \quad \text{Wk} \quad \text{Wv} \quad \text{Wo} \\
 & \downarrow \\
 & x_{T-1}
 \end{aligned}$$

$$\tilde{e}_{t+1} = e_{t+1} + \Delta e_{t-1}$$

$4 \times 12,228$

$$\tilde{e}_t = \begin{bmatrix} 12,228 \\ \vdots \\ 0 \\ \text{NN} \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} \quad \leftarrow 12,228$$

- ▶ **Traditional RNNs/LSTMs suffer from:**
  - Sequential computation → slow training
  - Vanishing gradients → poor long-range dependencies
- ▶ **Need for a model that:**
  - Handles global context
  - Is parallelizable
  - Supports scalability

After this session, you should be able to:

- ▶ Understand the self-attention mechanism
- ▶ Learn how multi-head attention works and why it helps
- ▶ Grasp the role of positional encoding
- ▶ Deconstruct transformer blocks and variants
- ▶ Compare Pre-LayerNorm vs. Post-LayerNorm
- ▶ Evaluate LayerNorm vs. RMSNorm
- ▶ Explore causal masking in decoder-only transformers
- ▶ Understand tradeoffs and future directions



# Introduction to Transformers

## ► What is a Transformer?

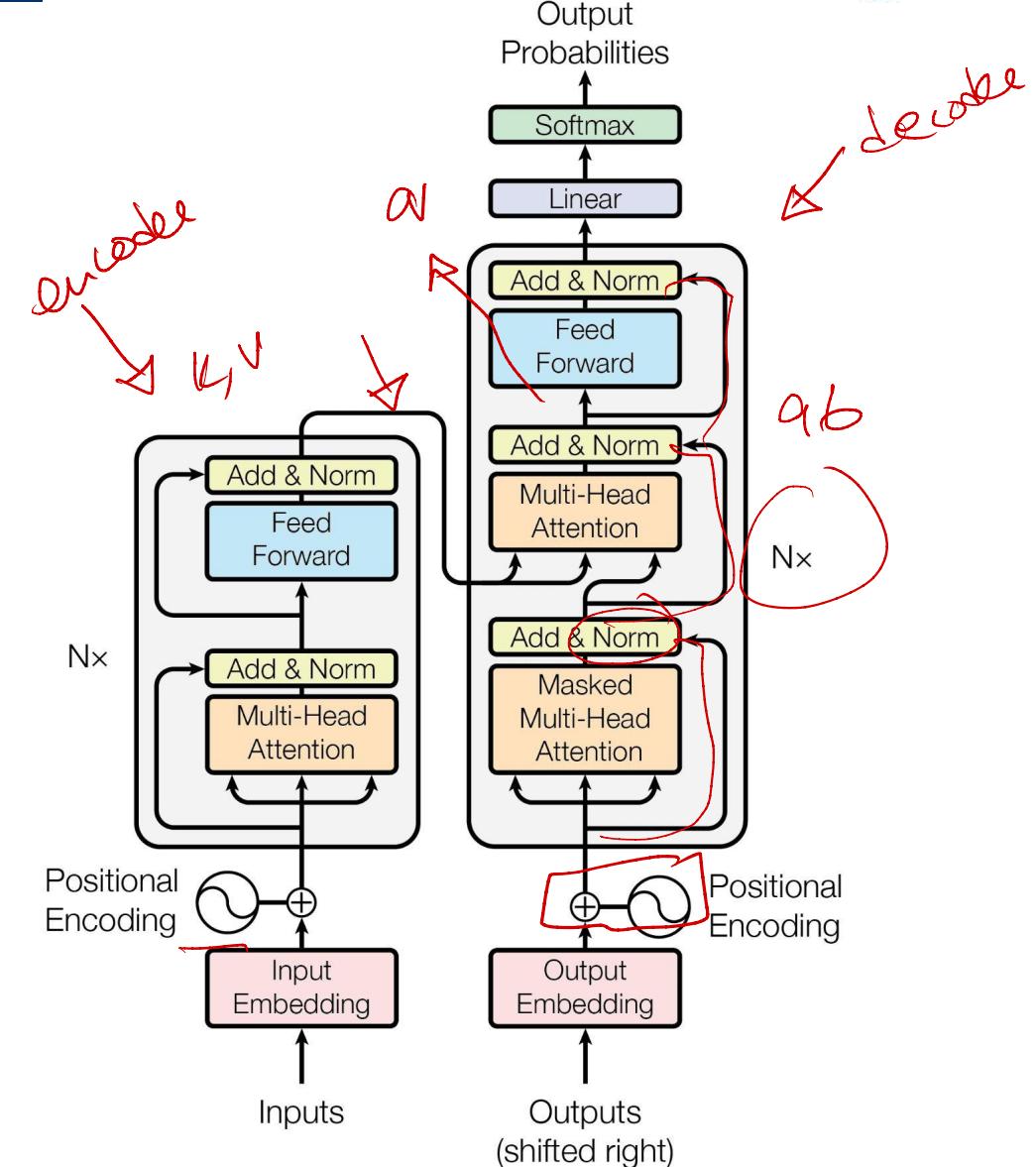
- Introduced by Vaswani et al. (2017) – “Attention is All You Need”

## ► Key features:

- No recurrence, all attention
- Encoder-decoder structure
- Enables parallel processing
- Scales well with data and compute

# Transformers

- ❖ Tokenization
- ❖ Input Embeddings
- ❖ Position Encodings
- ❖ Query, Key, & Value
- ❖ Attention
- ❖ Self Attention
- ❖ Multi-Head Attention
- ❖ Feed Forward
- ❖ Add & Norm
- ❖ Encoders
- ❖ Masked Attention
- ❖ Encoder Decoder Attention
- ❖ Linear
- ❖ Softmax
- ❖ Decoders
- ❖ Encoder-Decoder Models





BPE

a b ā c a d a a b b ā a b e s

a b e c a d a e bb ēb c d

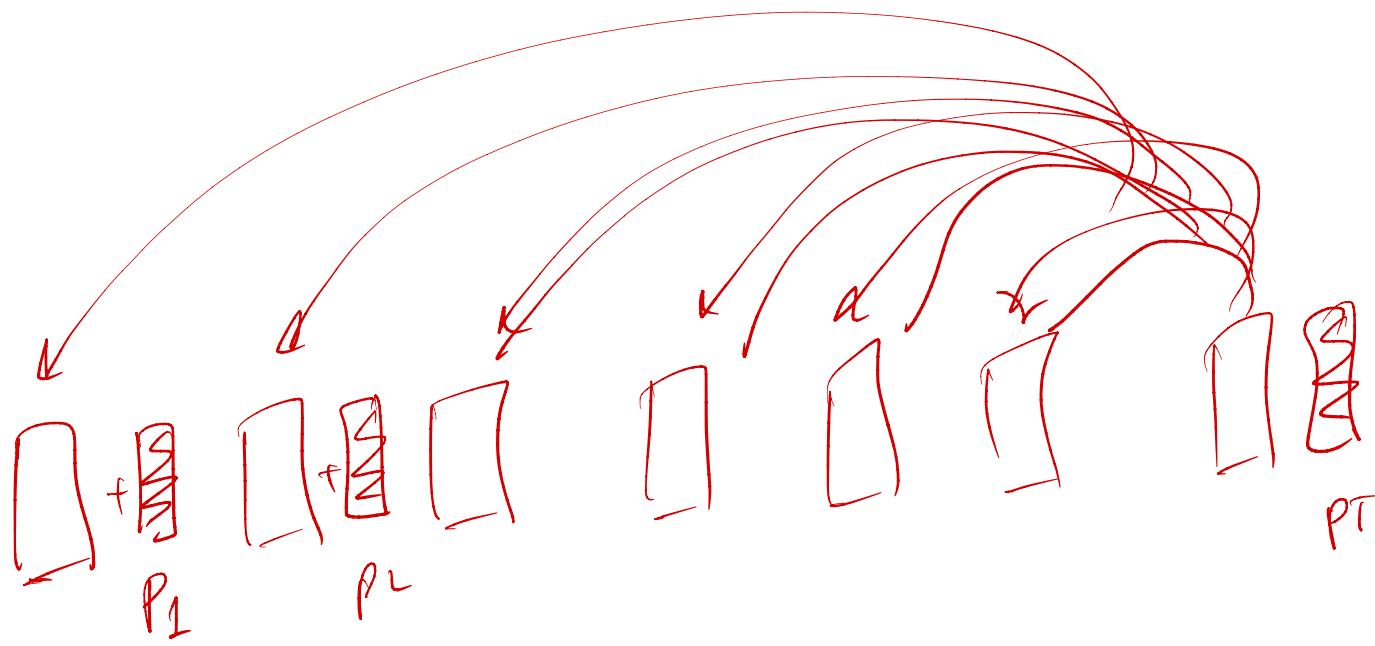
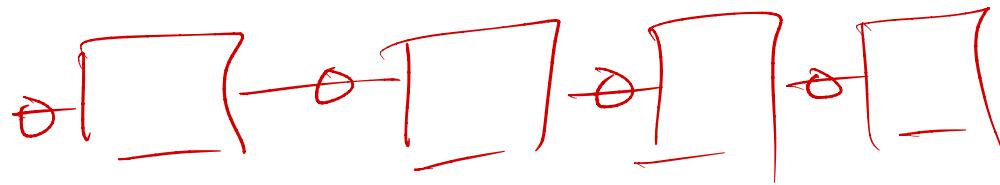
a b e c a d f bf c d

look

x ray

X ray

a, b, c, d, e, f  
—  
<sup>aa</sup><sup>ii</sup><sup>eb</sup>  
<sup>bb</sup><sup>ff</sup>



I am happy, not sad,

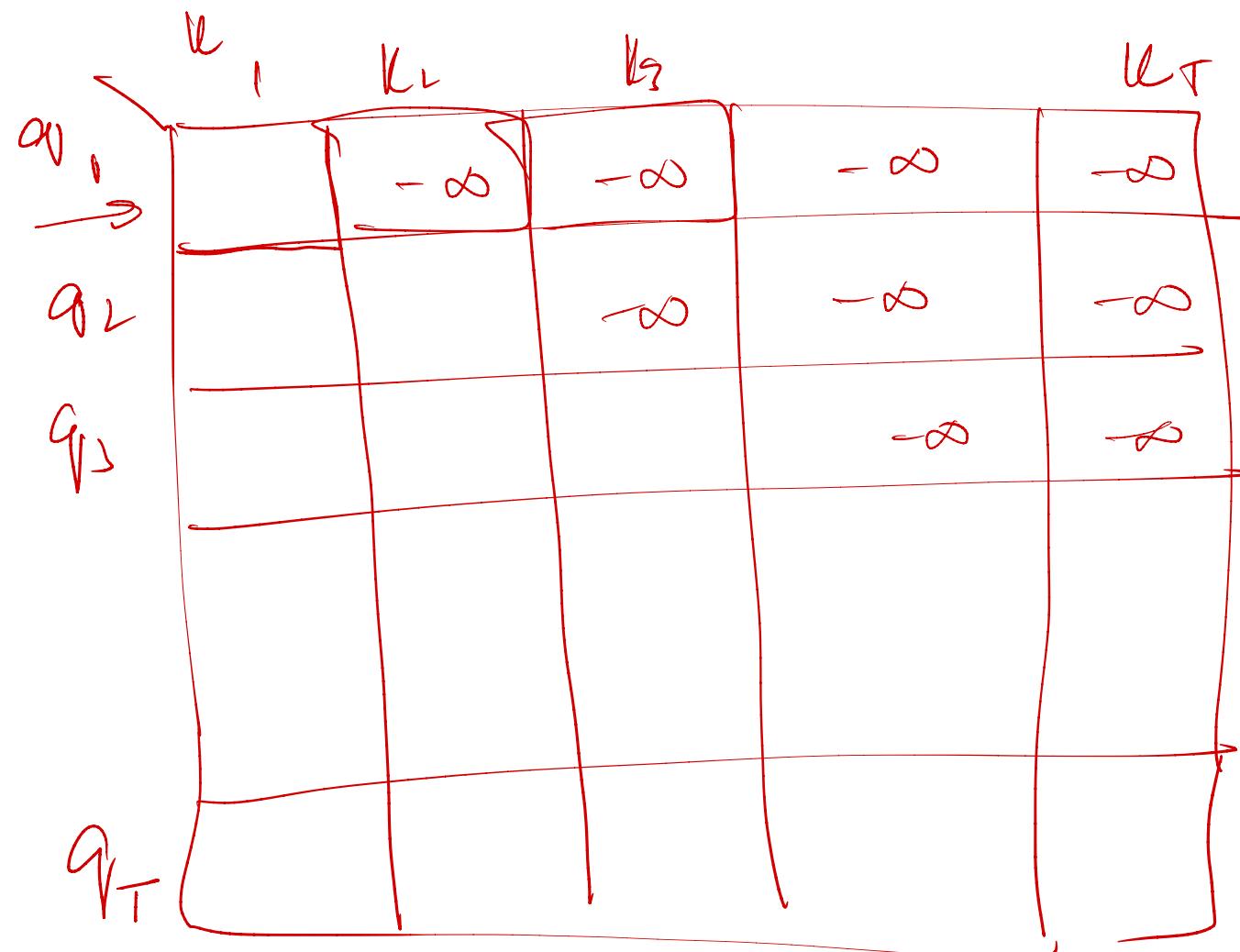
I am sad, not happy.

today is a good day <end>

| | | | | |

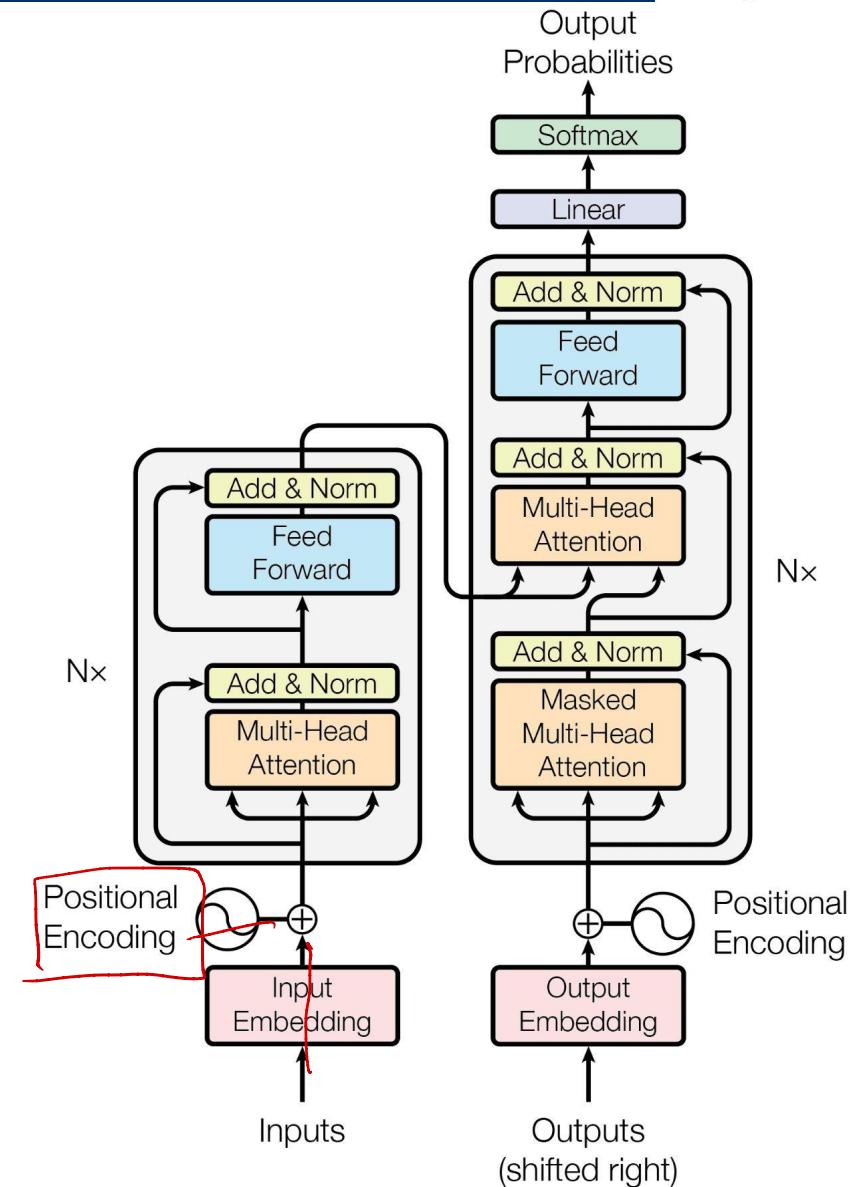
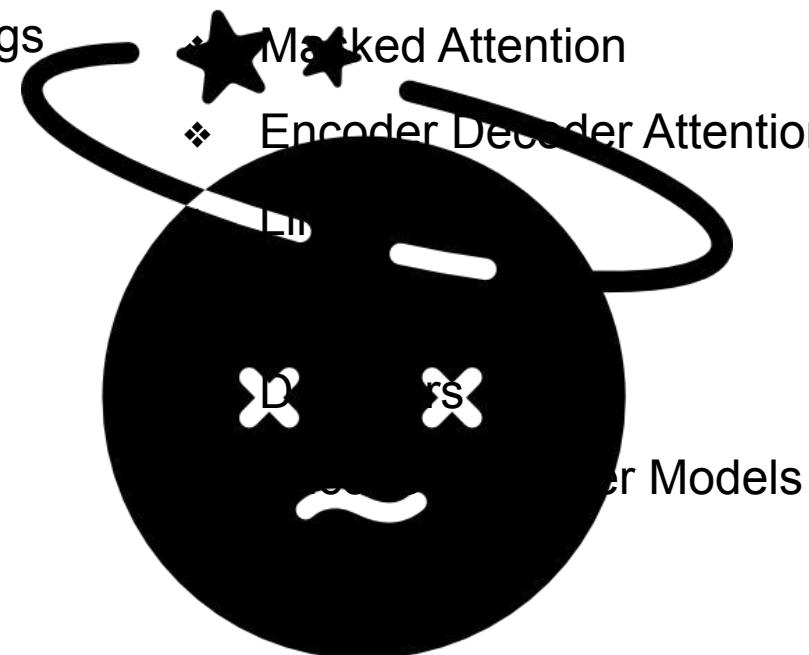
T B dog is a good day

(sos)



# Transformers

- ❖ Tokenization
- ❖ Input Embeddings
- ❖ Position Encodings
- ❖ Query, Key, & Value
- ❖ Attention
- ❖ Self Attention
- ❖ Multi-Head Attention
- ❖ Feed Forward
- ❖ Add & Norm
- ❖ Encoders



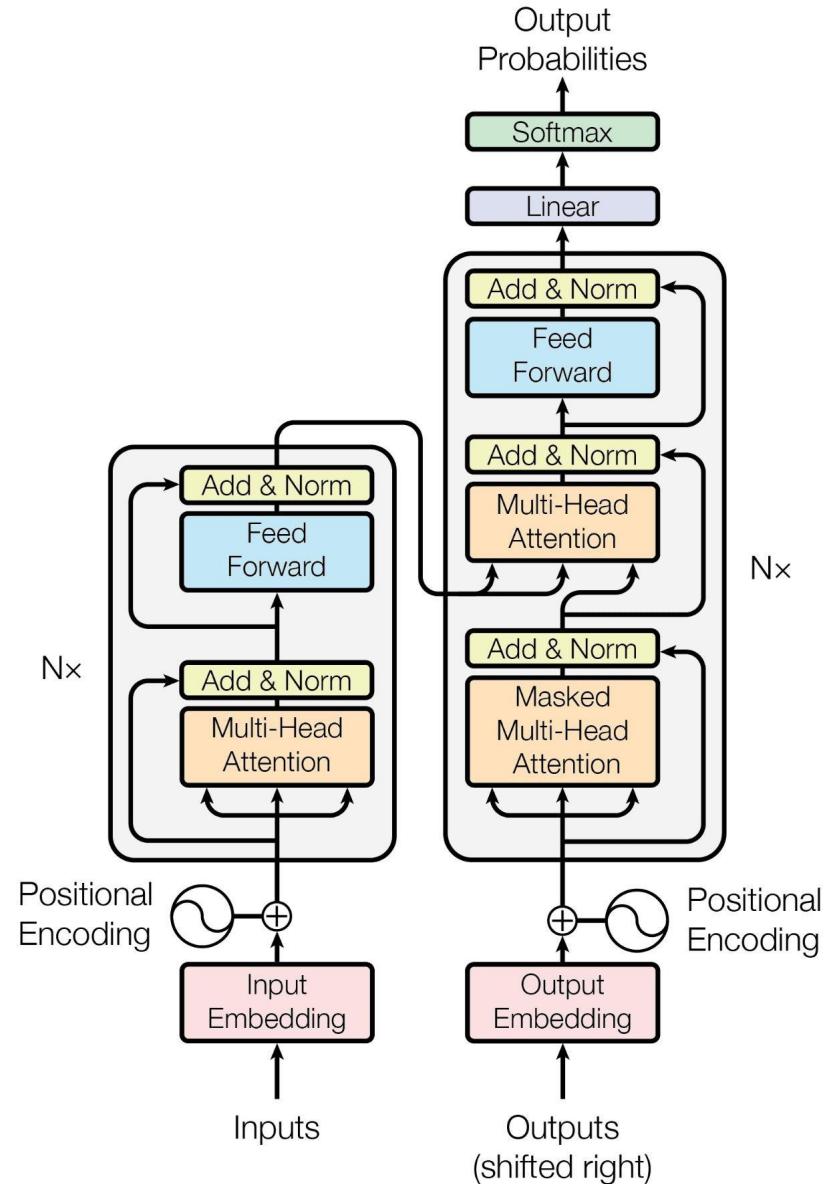
# Machine Translation

**Targets**

Ich habe einen  
Apfel gegessen

**Inputs**

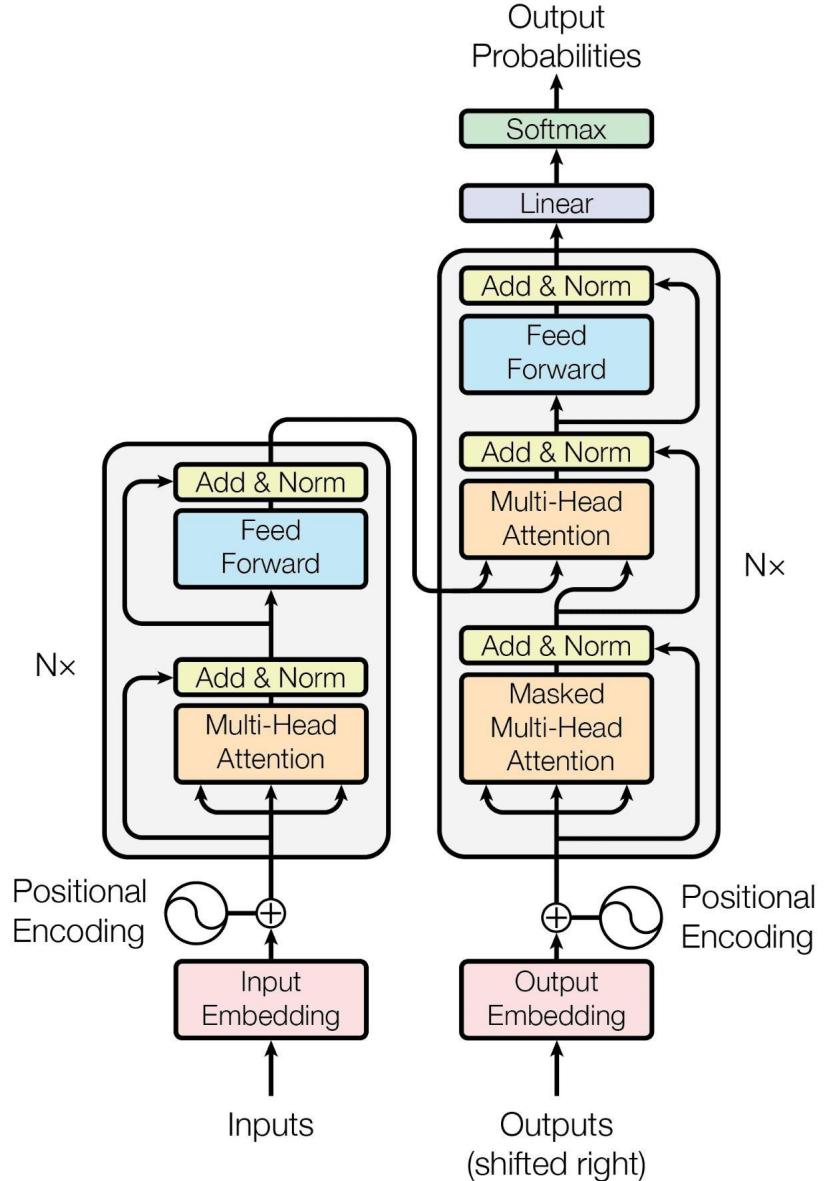
I ate an apple



# Inputs

## Processing Inputs

Inputs  
I ate an apple



# Self-Attention Mechanism

- ▶ Computes contextual representation of a word using all other words
- ▶ **Inputs:** Query (Q), Key (K), Value (V)
- ▶ **Output:** Weighted sum of values

## Attention formula:

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

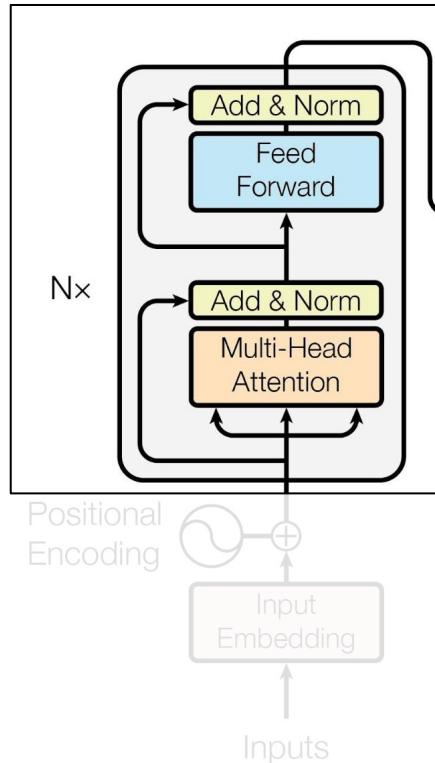
- ▶ Enables each word to attend to all others

# Self-Attention Intuition: The "Bank" Analogy

- ▶ **Query:** What we're looking for (e.g., the meaning of "bank" in context)
- ▶ **Keys:** Tags or features associated with each word/sentence (e.g., "river", "money")
- ▶ **Values:** Information linked to each key (e.g., context-specific meaning)
- ▶ Uses dot-product similarity between Query and Keys to assign attention weights
- ▶ Weighted sum of Values produces the contextual representation

# Self Attention

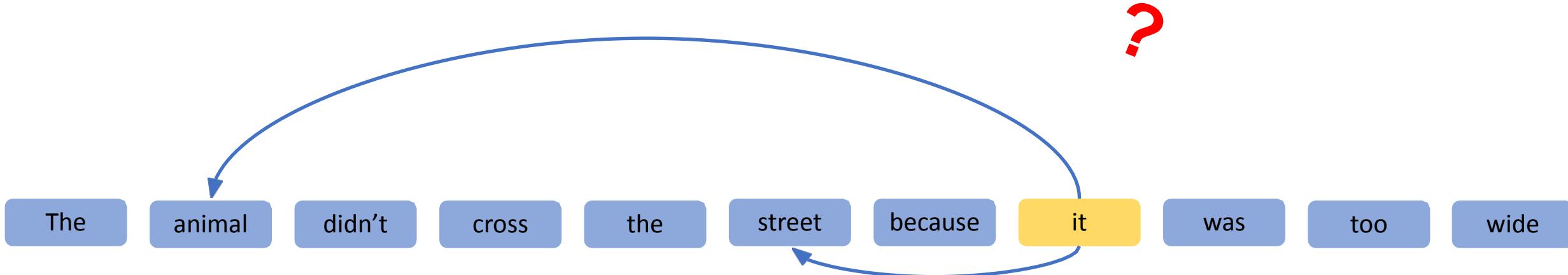
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



# Self Attention

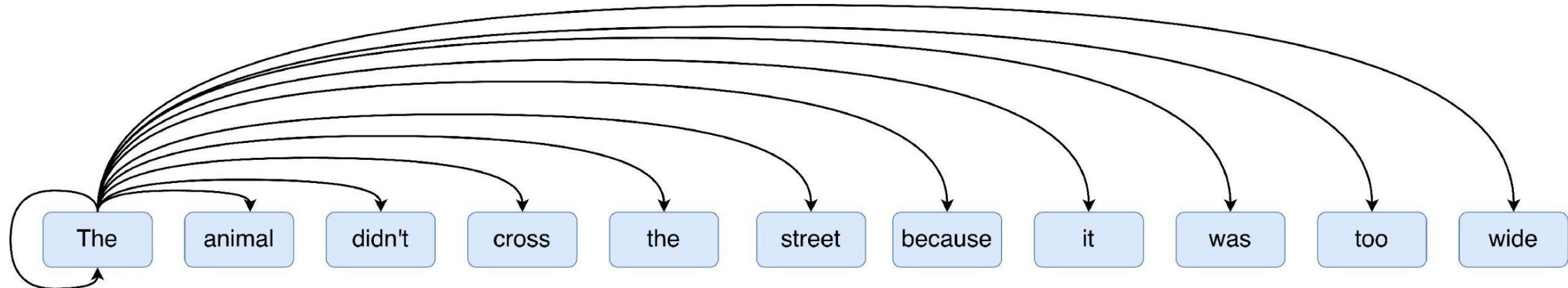
The animal didn't cross the street because it was too wide

# Self Attention

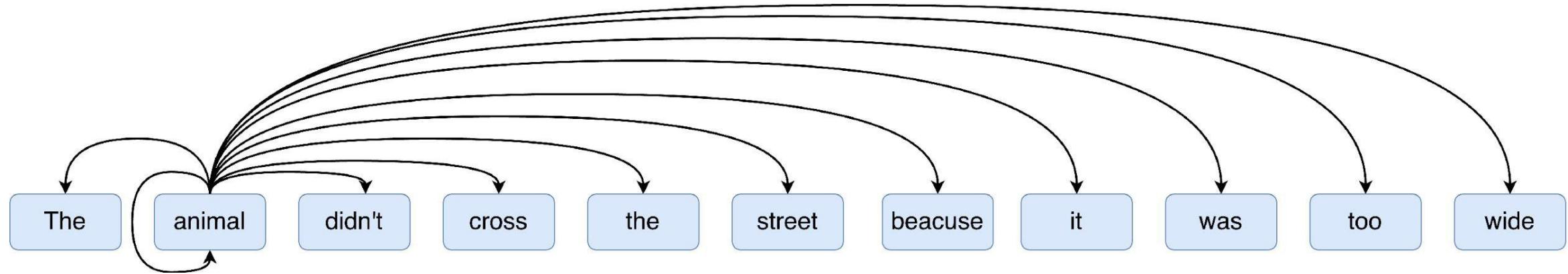


coreference resolution?

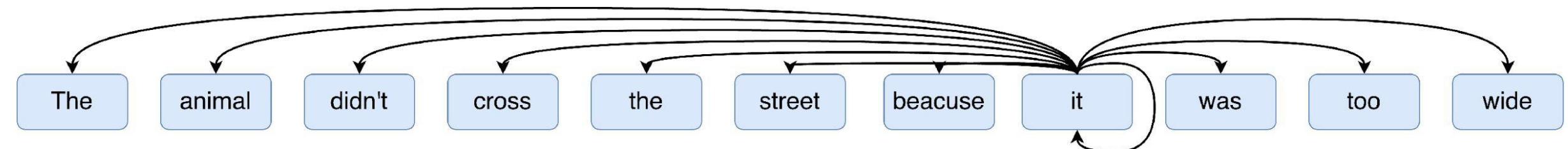
# Self Attention



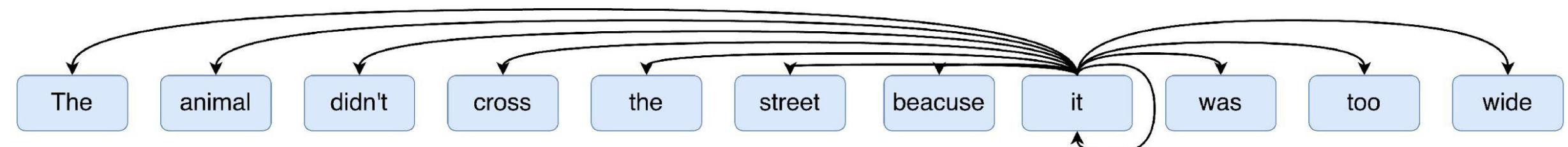
# Self Attention



# Self Attention



# Self Attention



**SELF**

Query Inputs

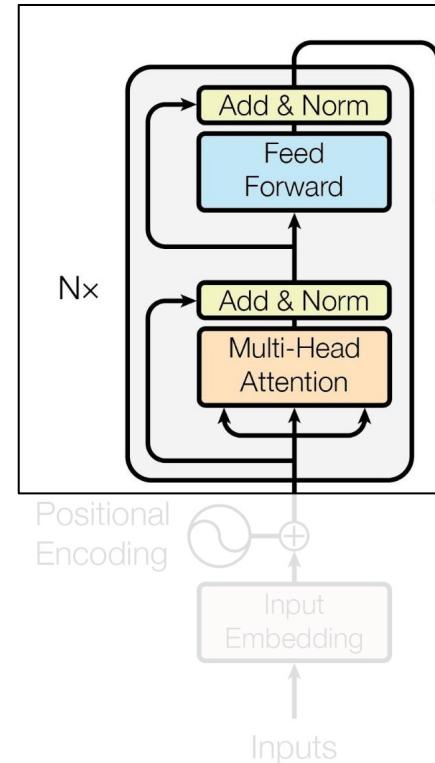
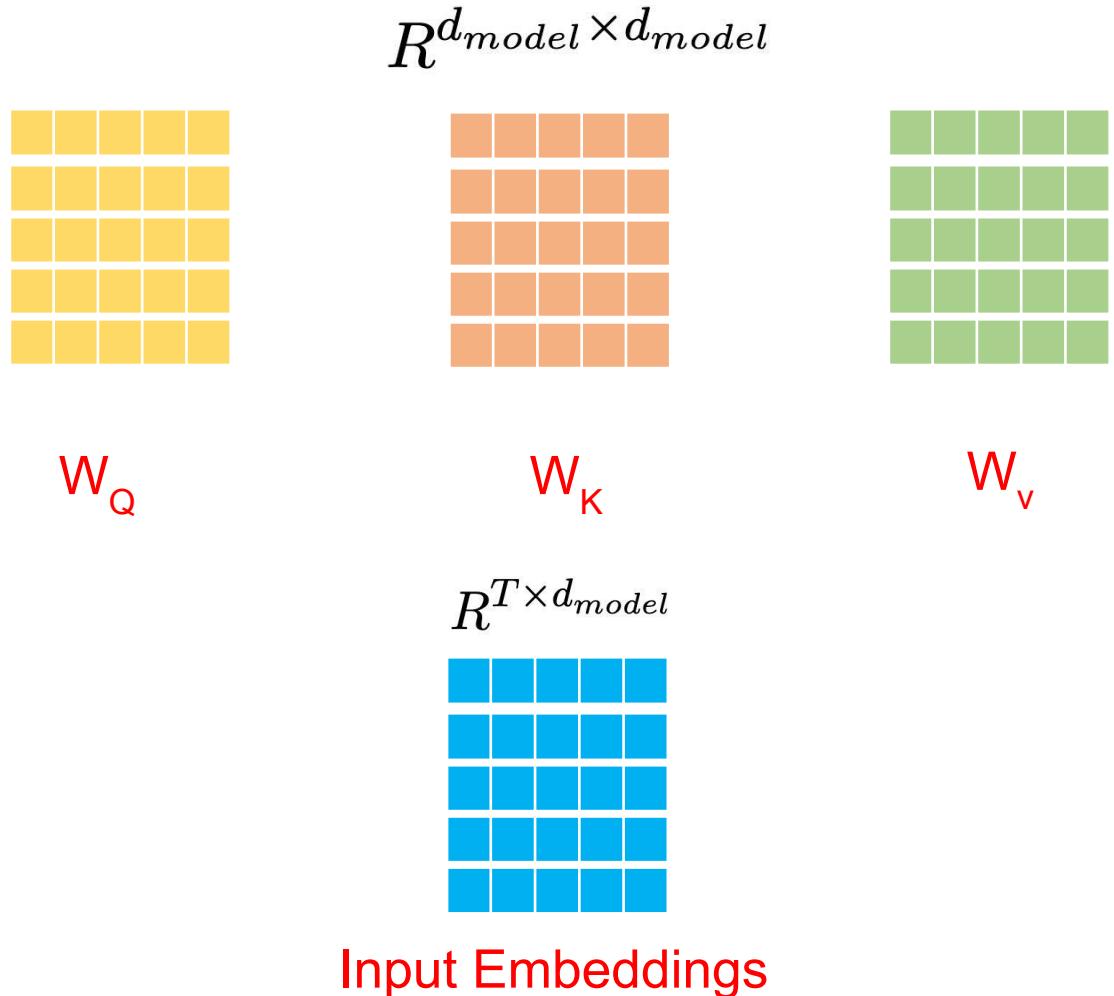
=

Key Inputs

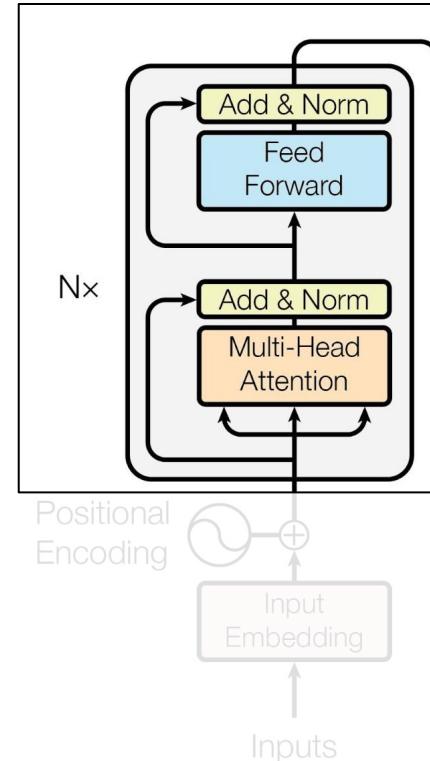
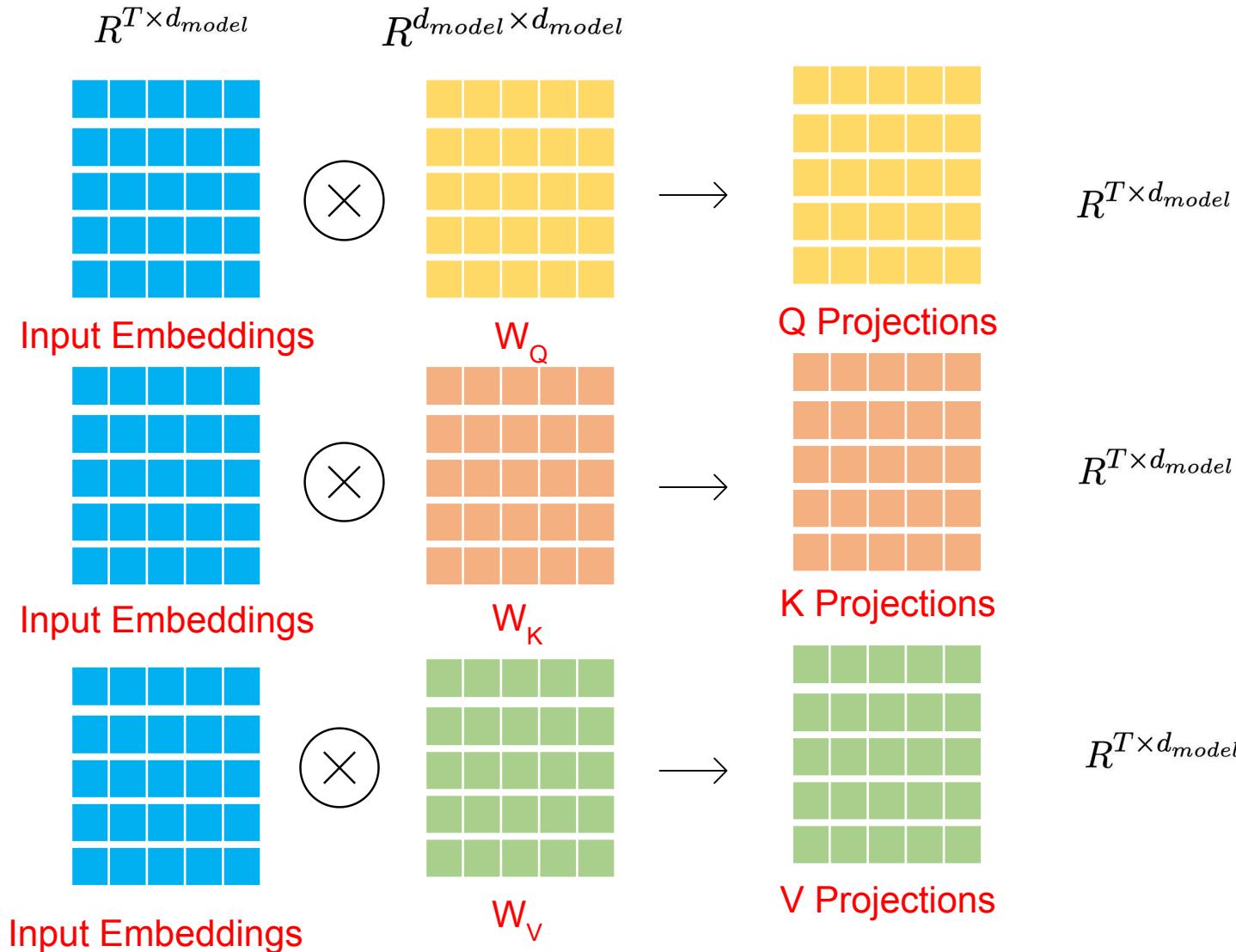
=

Value Inputs

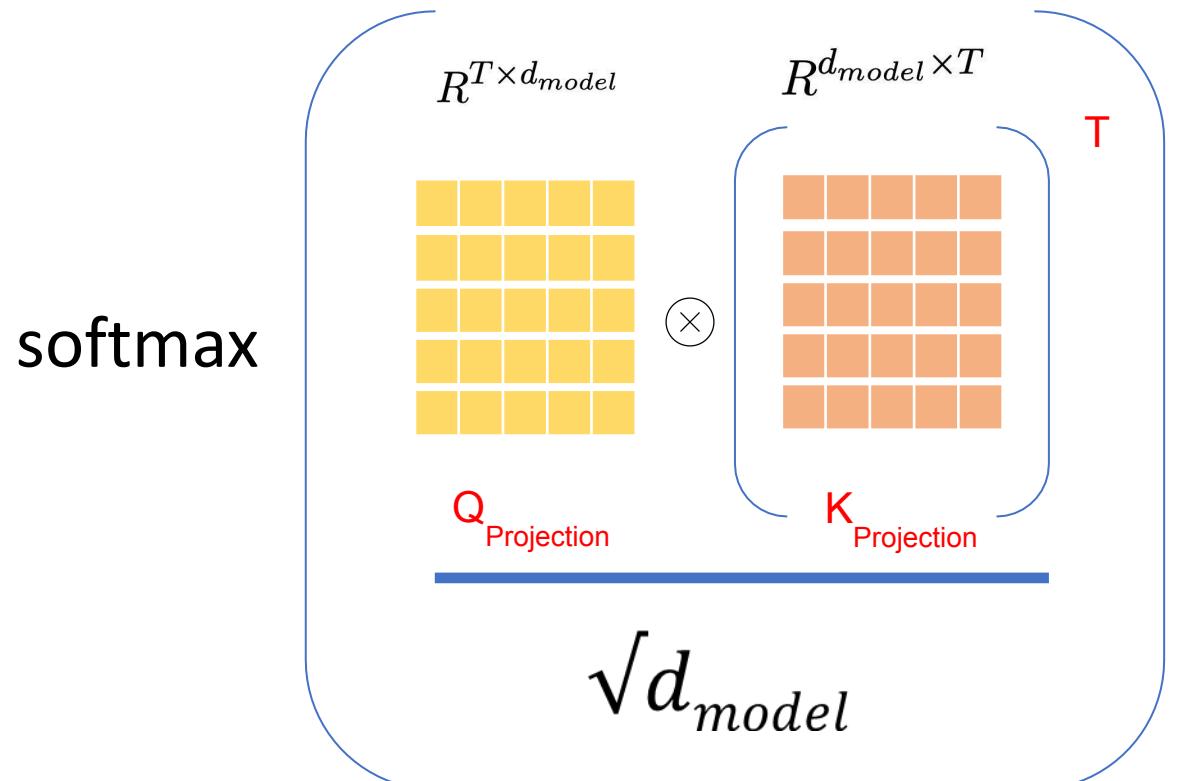
# Self Attention



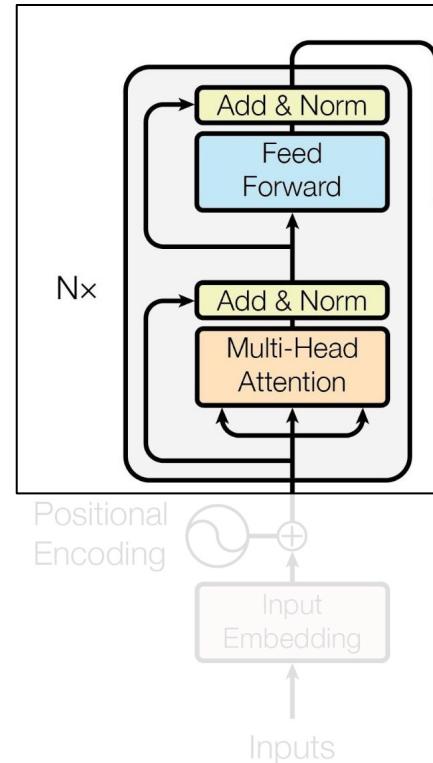
# Self Attention



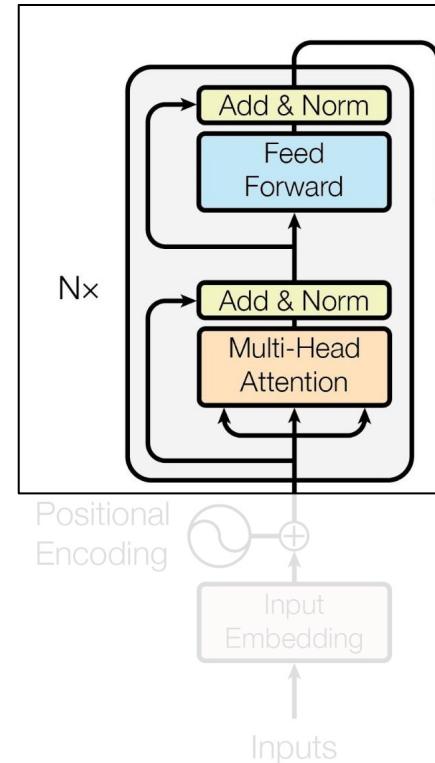
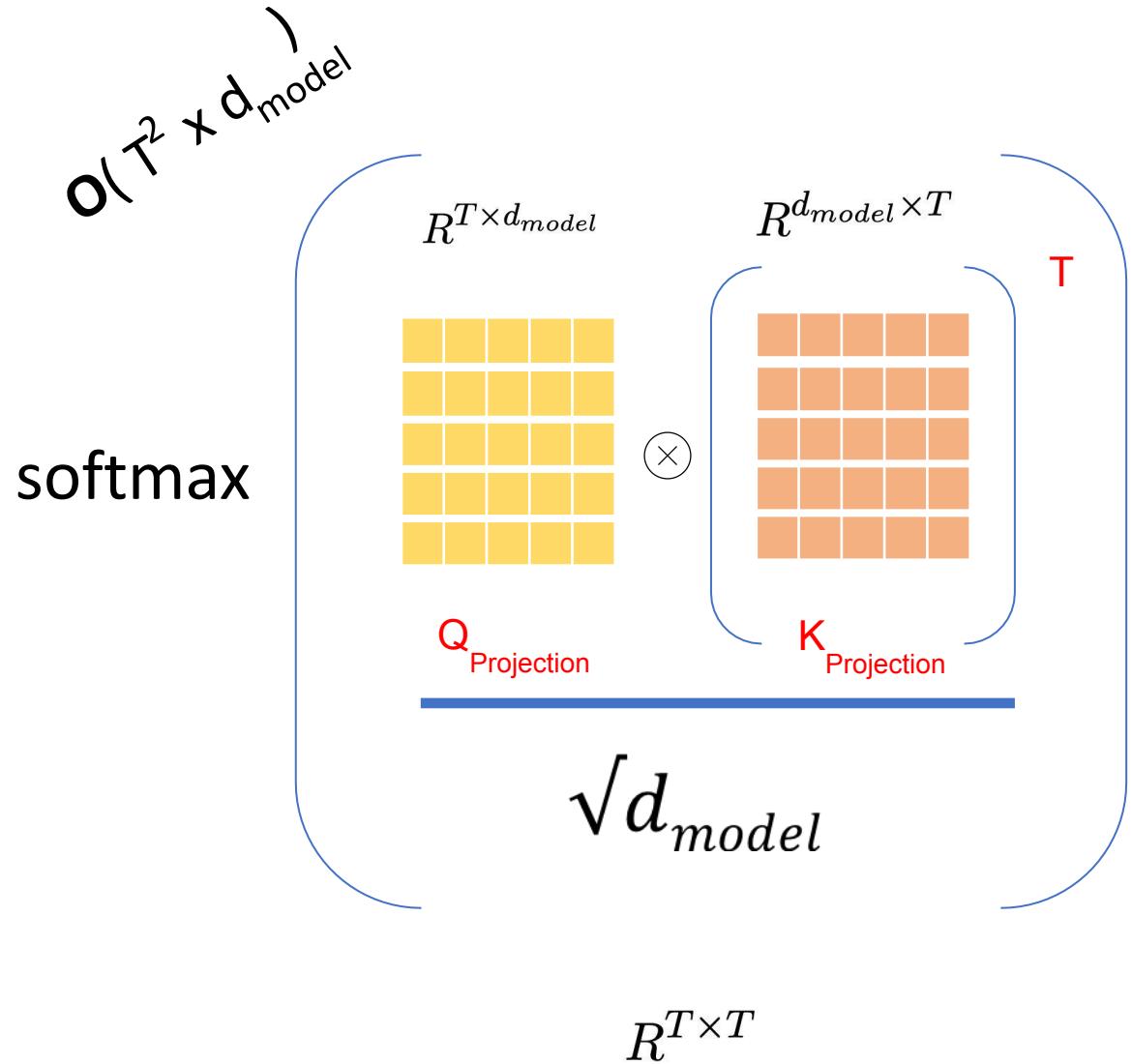
# Self Attention



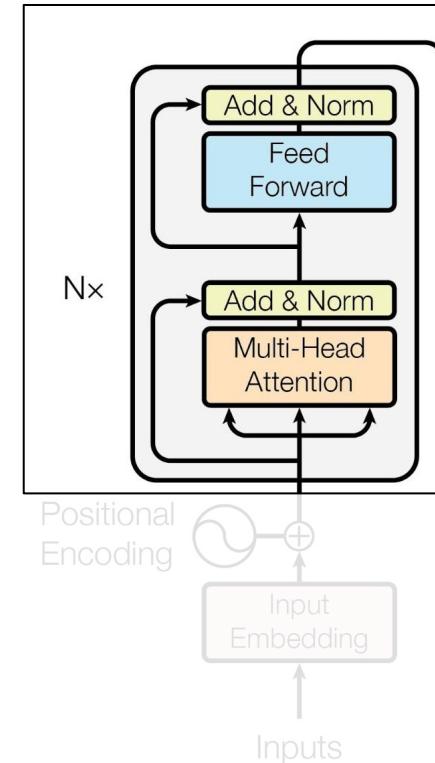
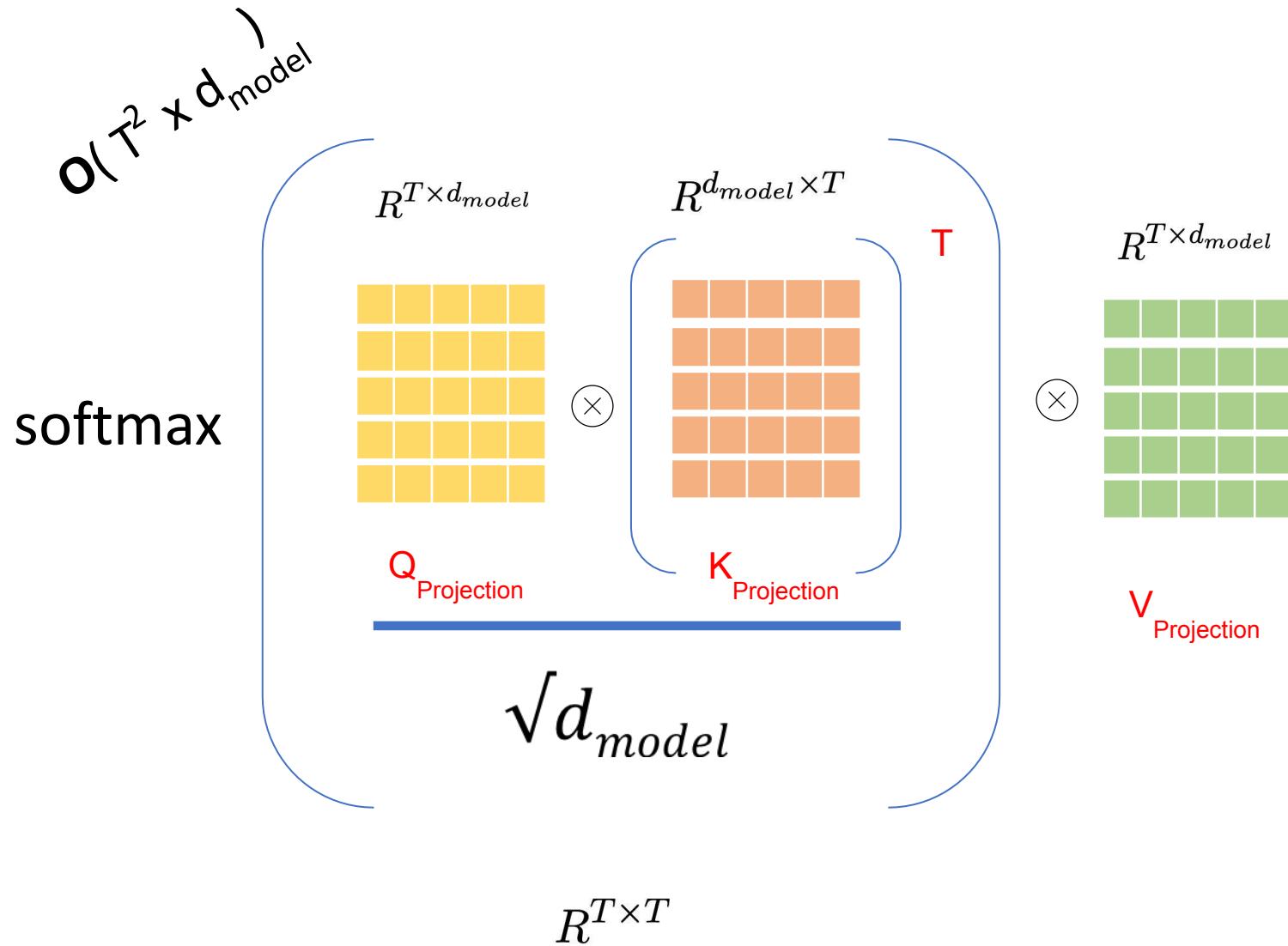
$$R^{T \times T}$$



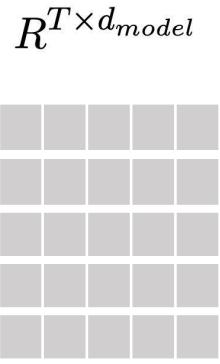
# Self Attention



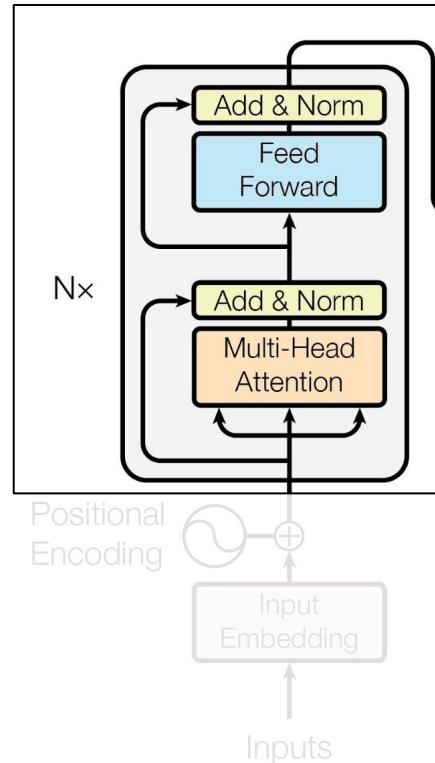
# Self Attention



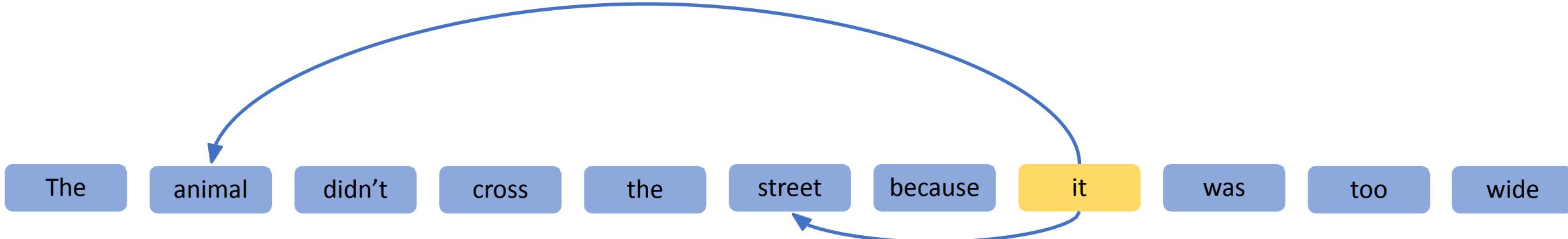
# Self Attention

$$R^{T \times d_{model}}$$


Attention: Z

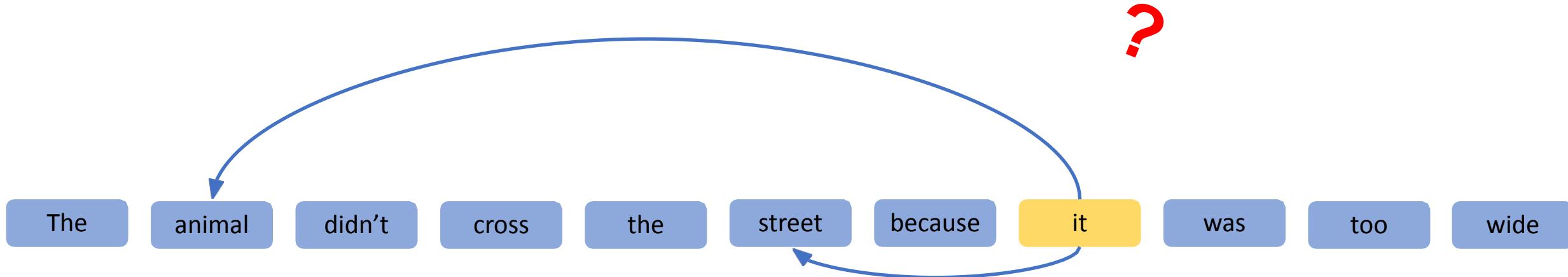


# Self Attention



Coreference resolution ✓

# Self Attention



Sentence boundaries ?

Coreference resolution ✓

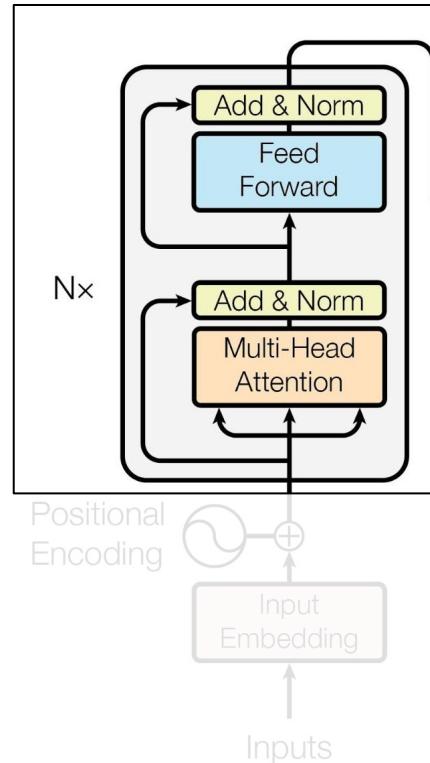
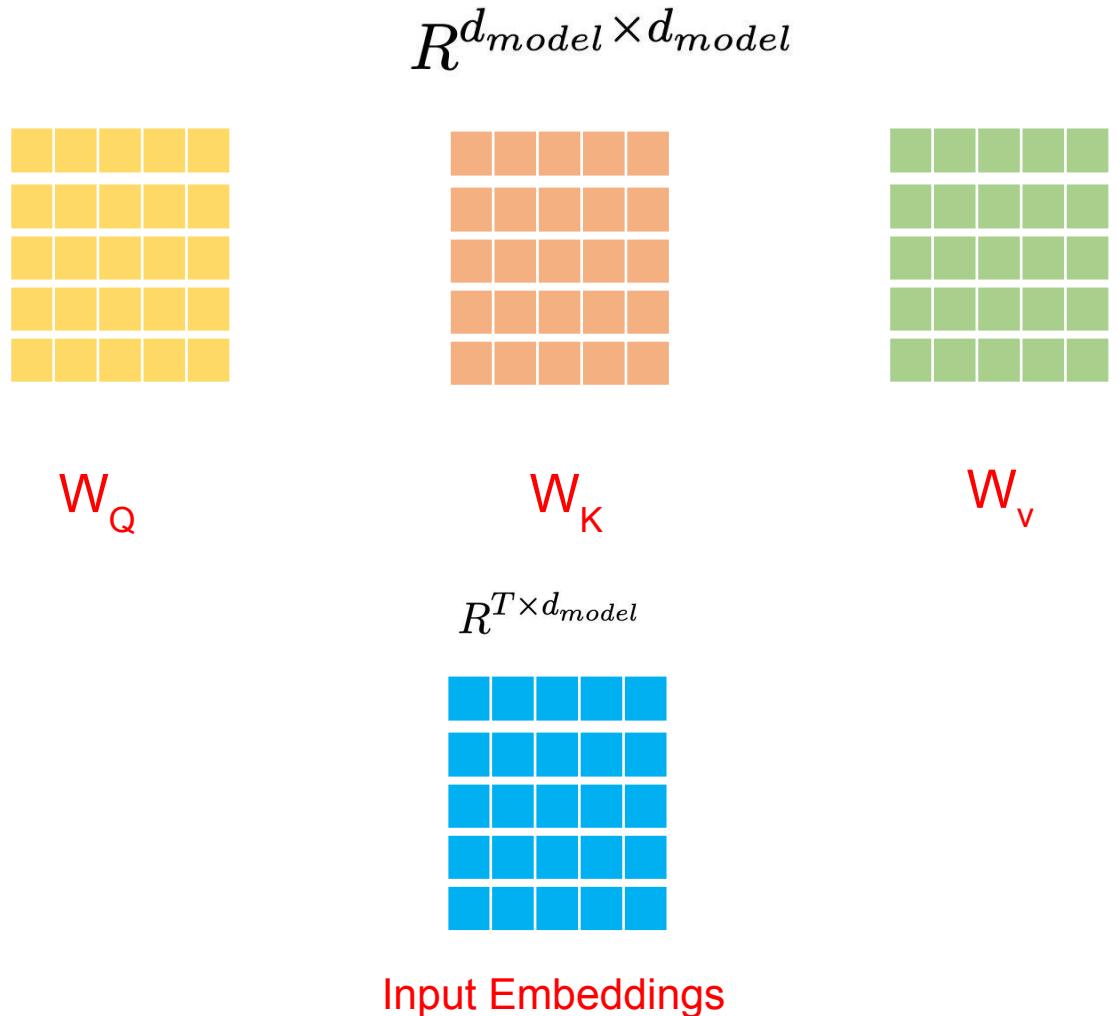
Context ?

Semantic relationships ?

Part of Speech ?

Comparisons ?

# Self Attention



# Multi-Head Attention

- ▶ Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.
- ▶ It consists of multiple attention heads, each with its own set of parameters.
- ▶ The outputs of all heads are concatenated and linearly transformed.

## Formula:

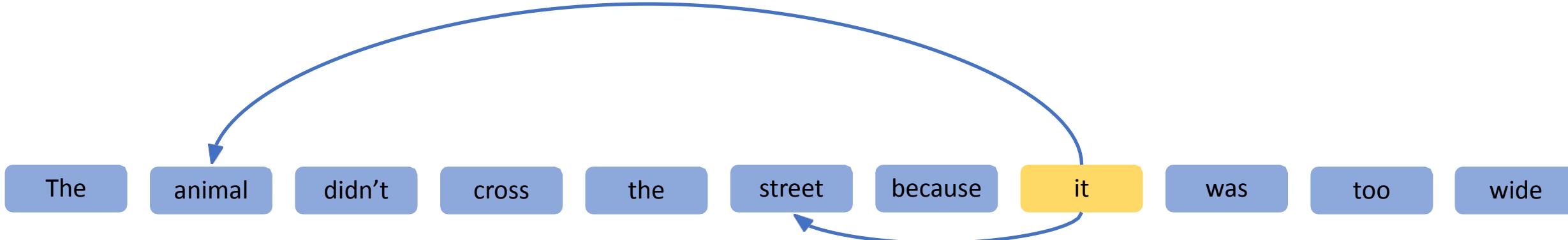
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

where each head is defined as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- ▶  $W_i^Q$ ,  $W_i^K$ , and  $W_i^V$  are learned projection matrices for each head.
- ▶  $W^O$  is the output projection matrix that combines the outputs of all heads.

# Multi-Head Attention



Sentence boundaries



Coreference resolution



Context



Semantic relationships



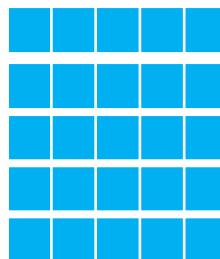
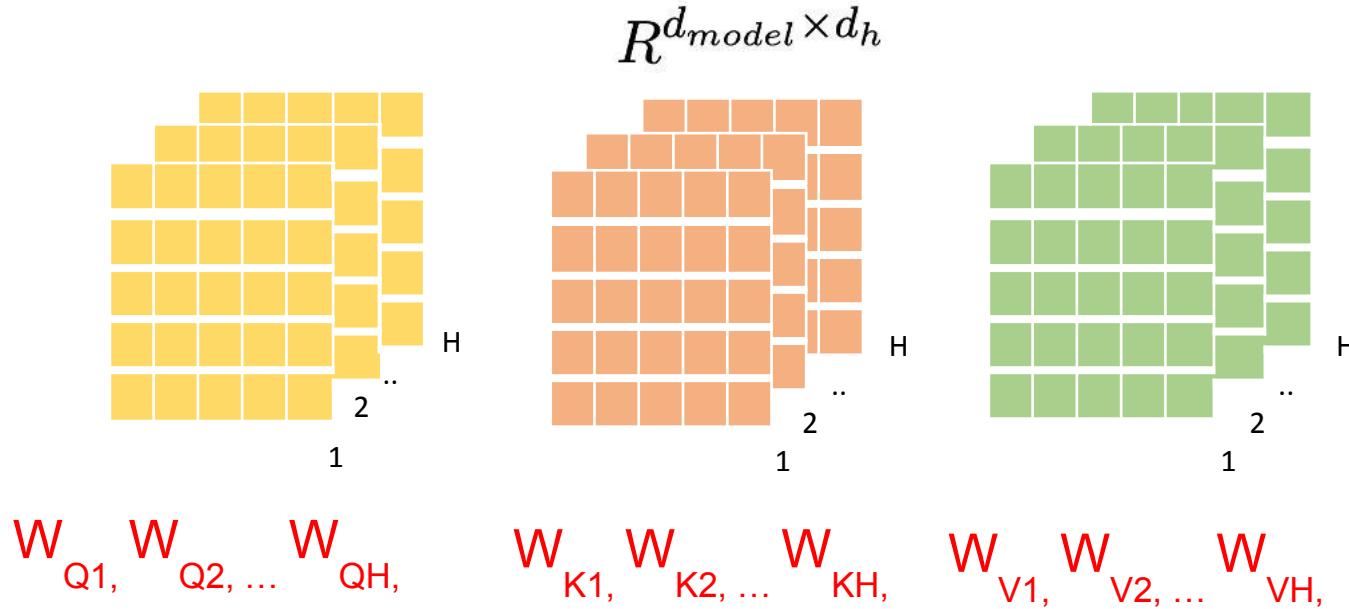
Part of speech



Comparisons



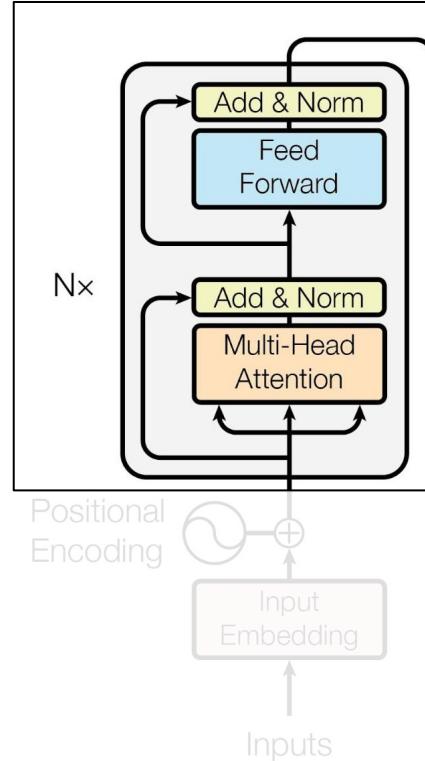
# Multi-Head Attention



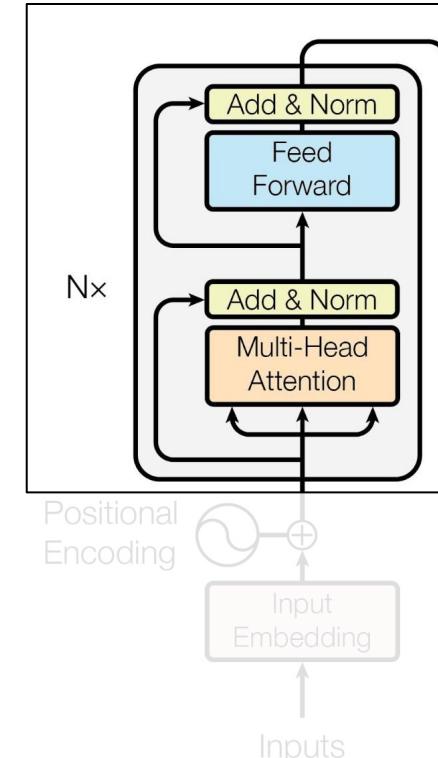
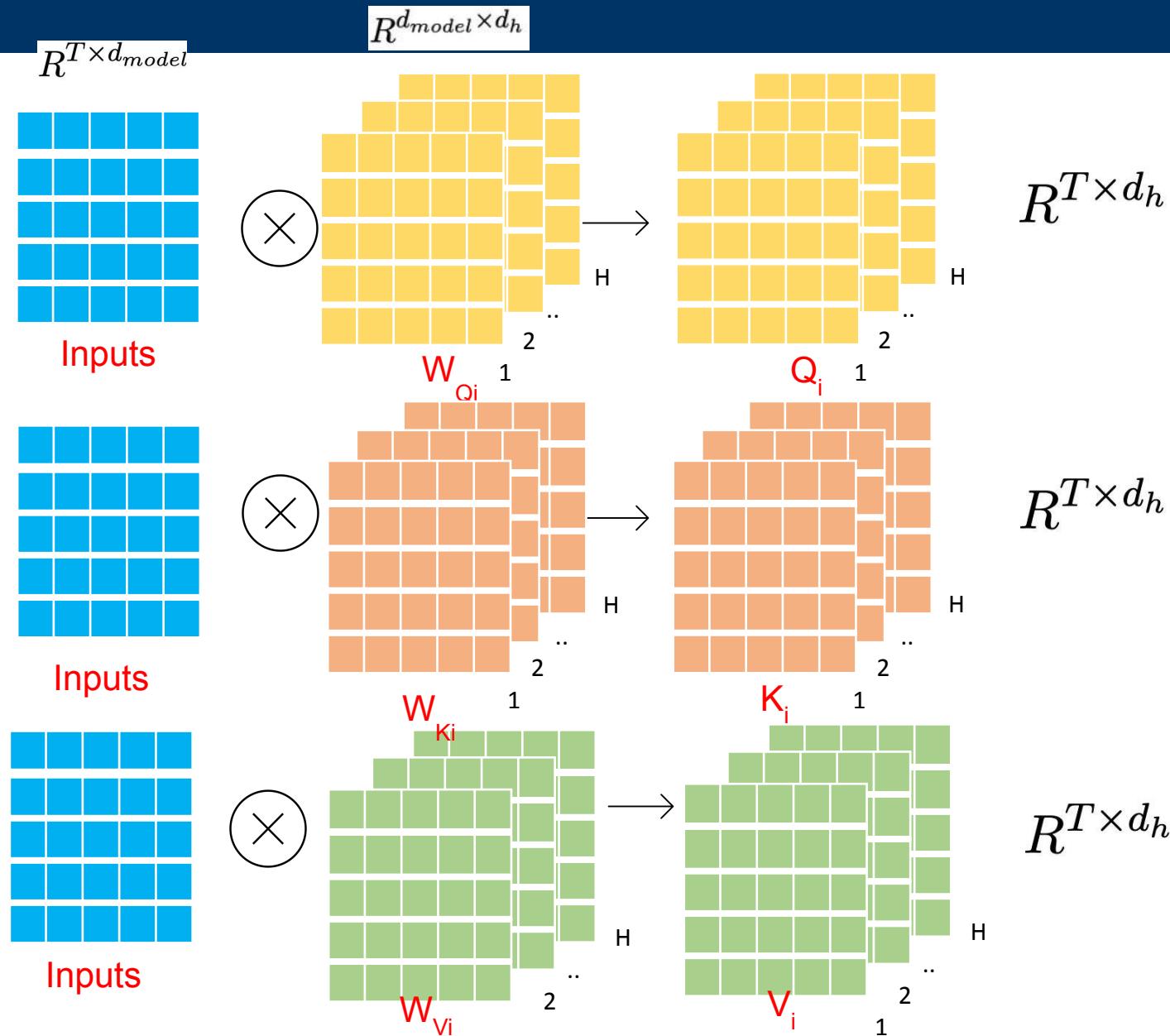
Input Embeddings

$$R^{T \times d_{model}}$$

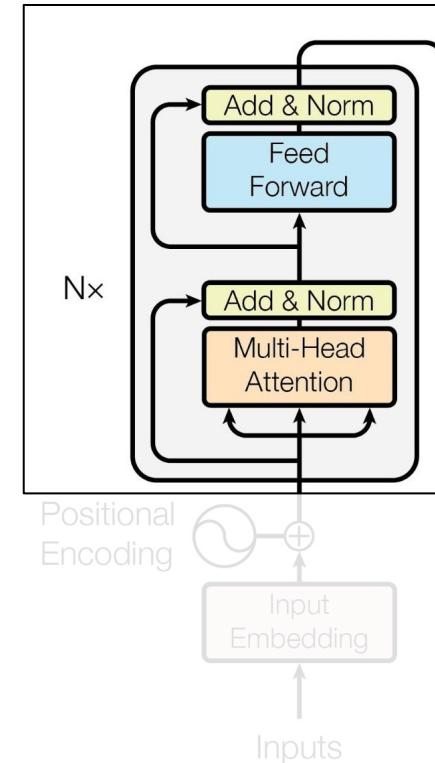
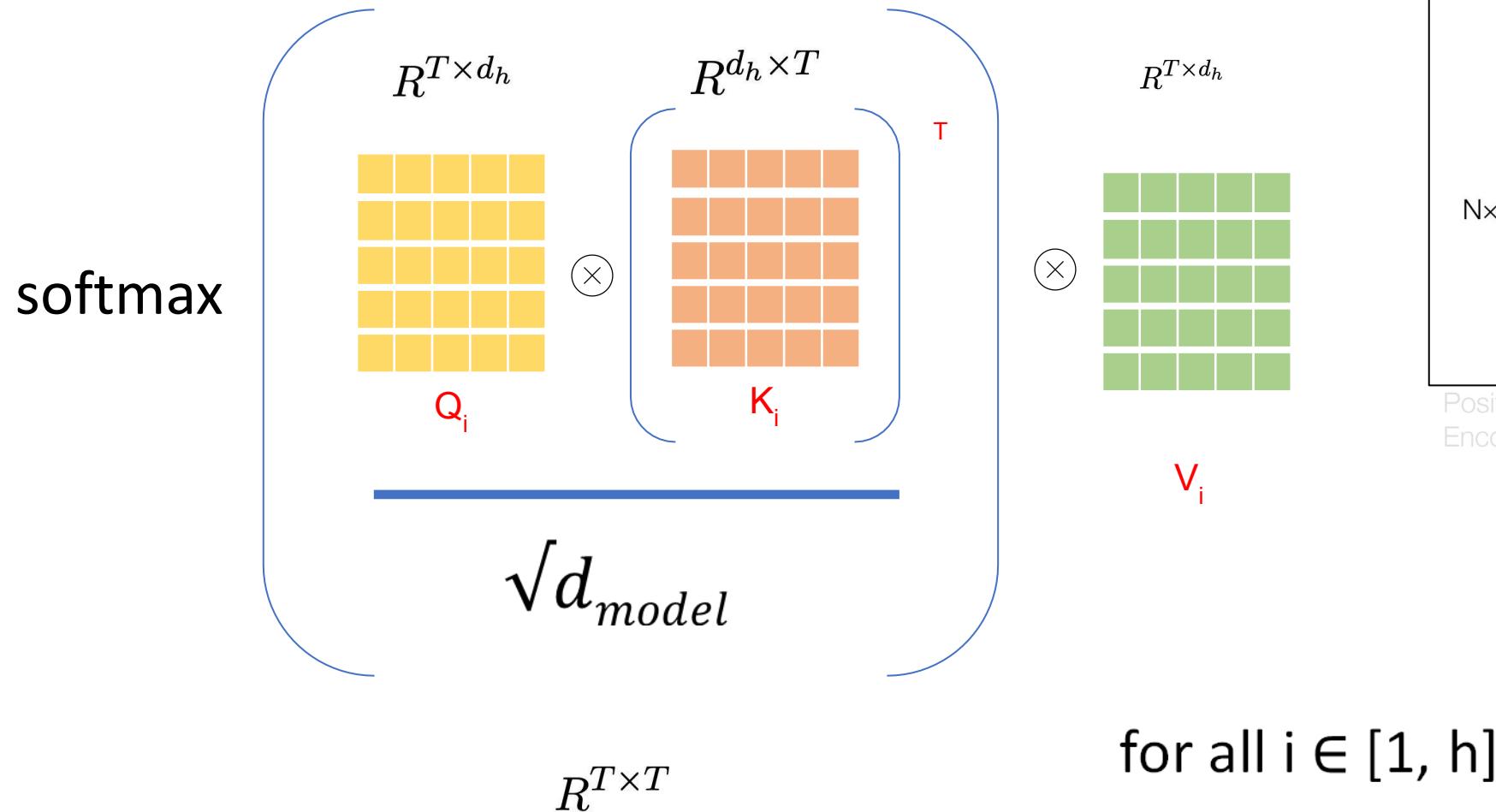
$$d_h = \frac{d_{model}}{h}$$



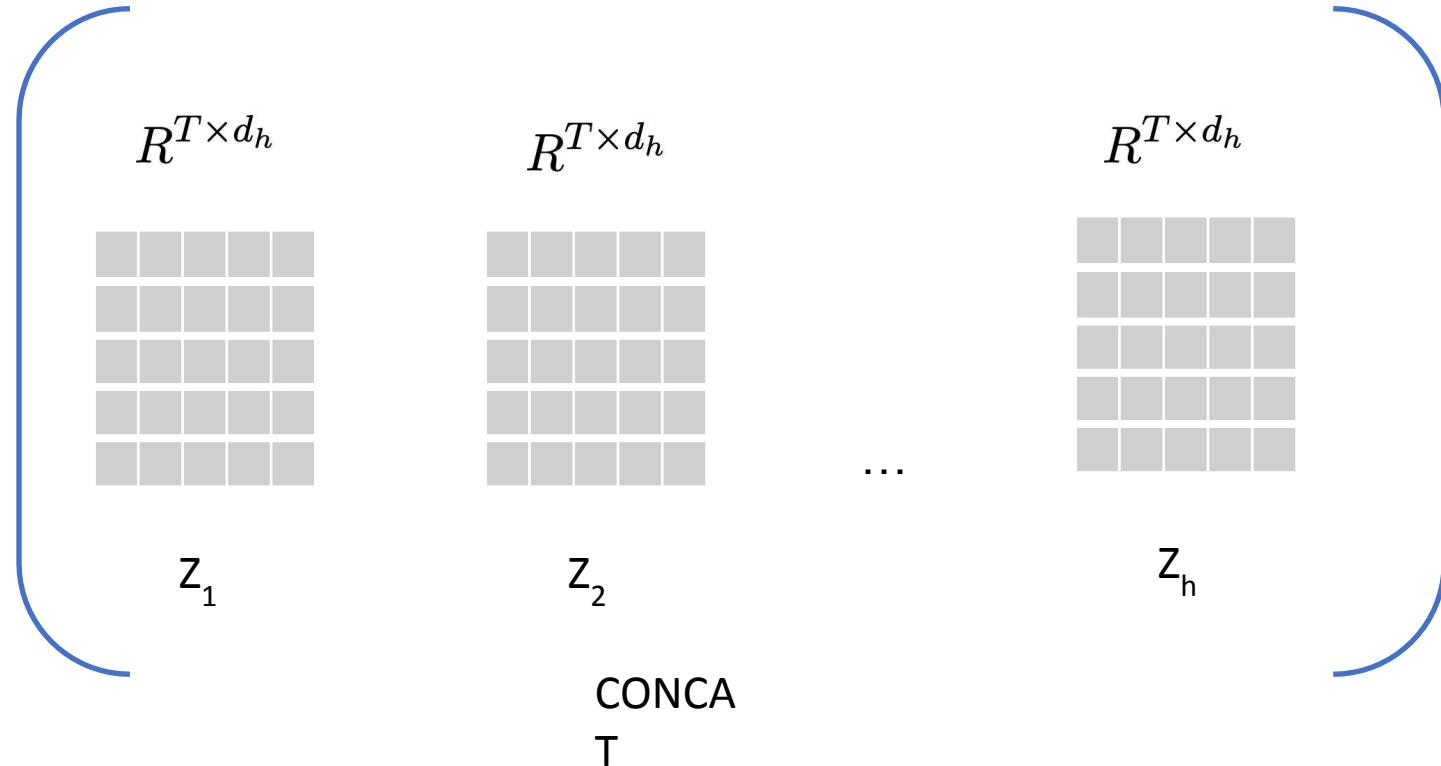
# Multi-Head Attention



# Multi-Head Attention



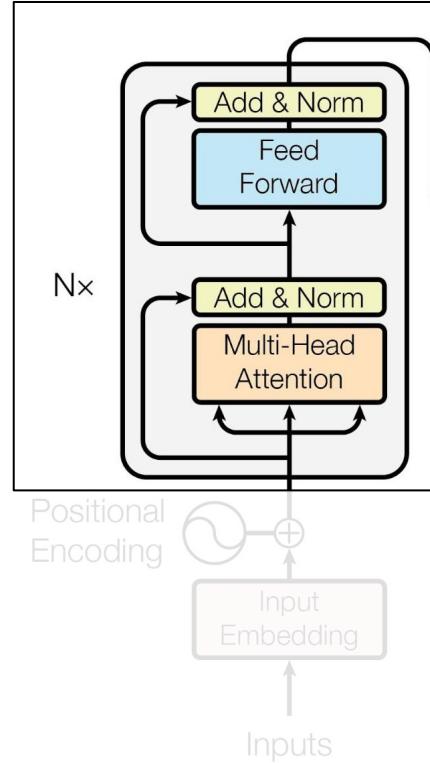
# Multi-Head Attention



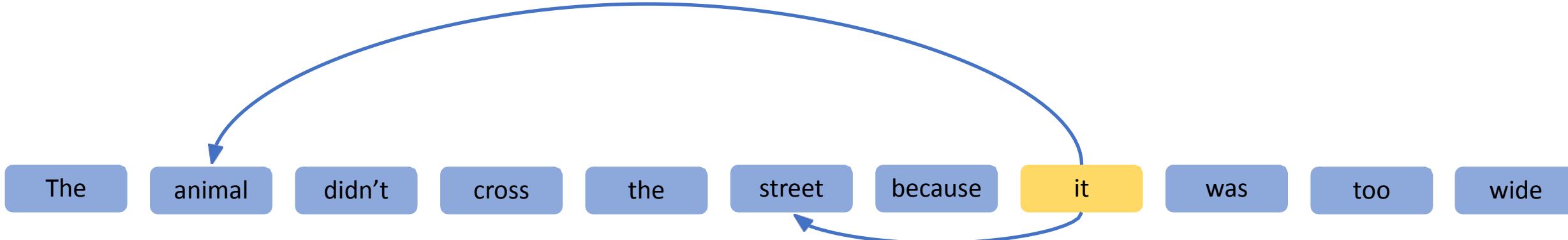
Multi Head Attention : Z

$$d_h = \frac{d_{model}}{h}$$

$$R^{T \times d_{model}}$$



# Multi-Head Attention



Sentence boundaries



Coreference resolution



Context



Semantic relationships



Part of speech



Comparisons



# Benefits of Multi-Head Attention

- ▶ Enables the model to jointly attend to information from different representation subspaces at different positions.
- ▶ Captures various linguistic and syntactic patterns by using multiple attention heads.
- ▶ Improves the model's ability to represent complex relationships in the data.
- ▶ Provides richer and more diverse feature representations.

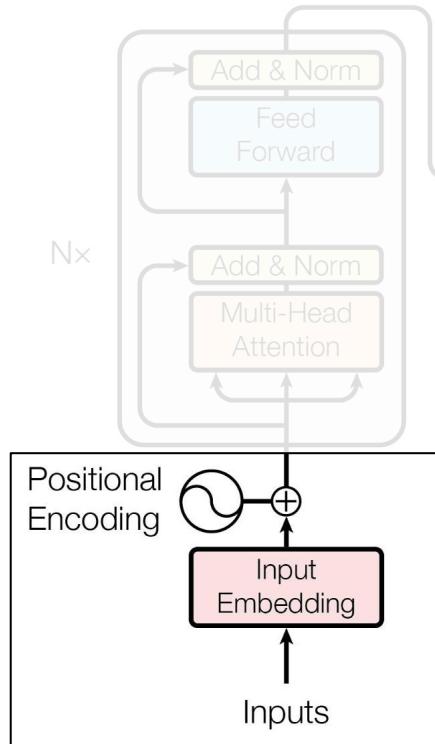
- ▶ Transformers lack sequence order awareness.
- ▶ **Positional encodings** are added to input embeddings to provide information about token positions.
- ▶ **Sinusoidal encoding:**

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

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

# Position Encodings

I      ate      an      apple      <eos>

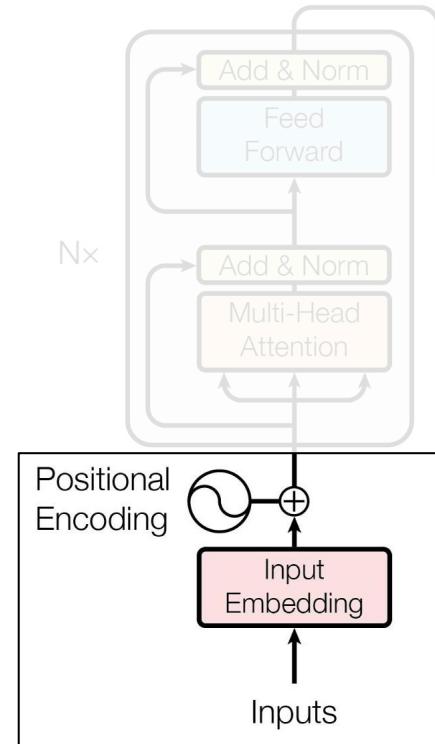


# Position Encodings

I ate an apple <eos>

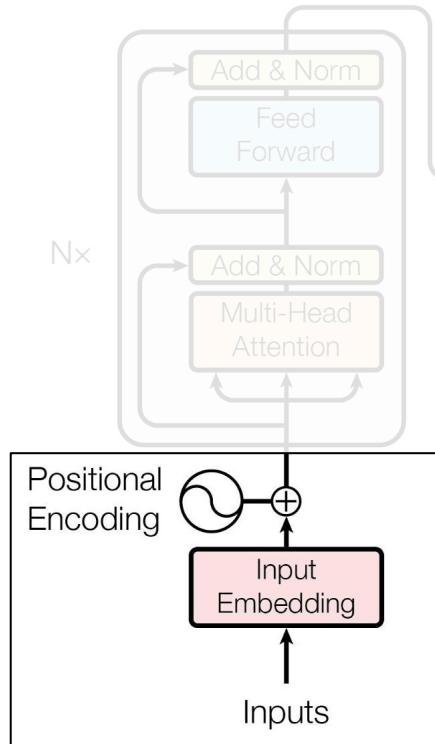


apple ate an I <eos>



# Position Encodings

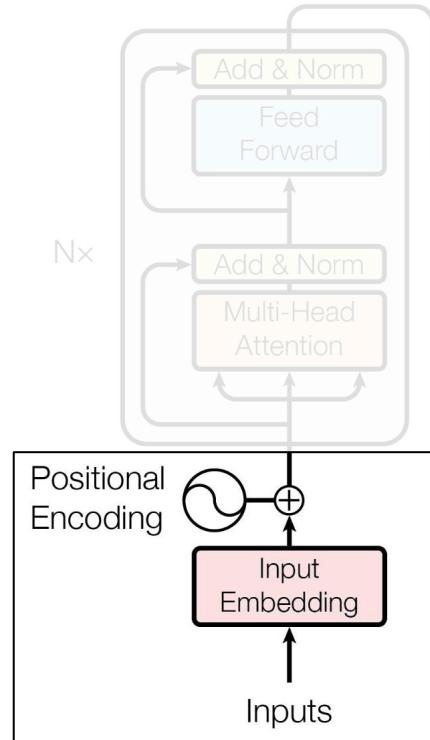
**Requirements for Positional Encodings???**



# Position Encodings

## Requirements for Positional Encodings

- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic



# Position Encodings

## Requirements for Positional Encodings

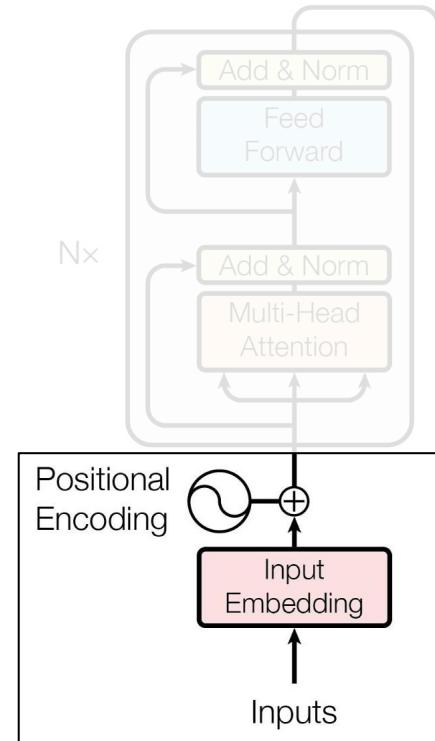
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic

Possible Candidates :

$$P_{t+1} = P_t + \Delta c$$

$$P_{t+1} = e^{P_t \Delta c}$$

$$P_{t+1} = P_t e^{t \Delta c}$$



# Position Encodings

## Requirements for Positional Encodings

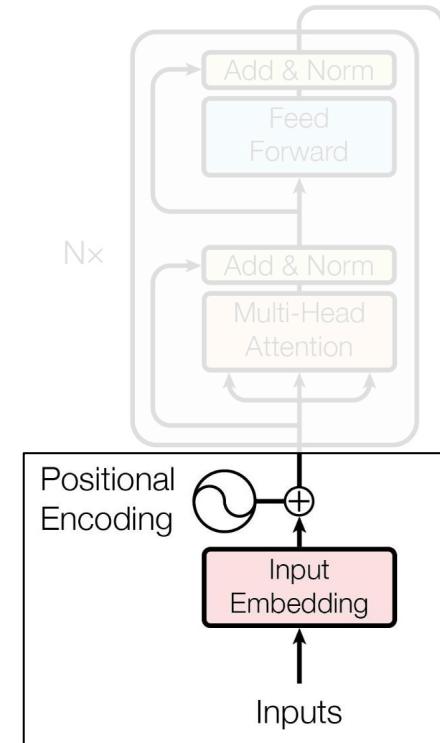
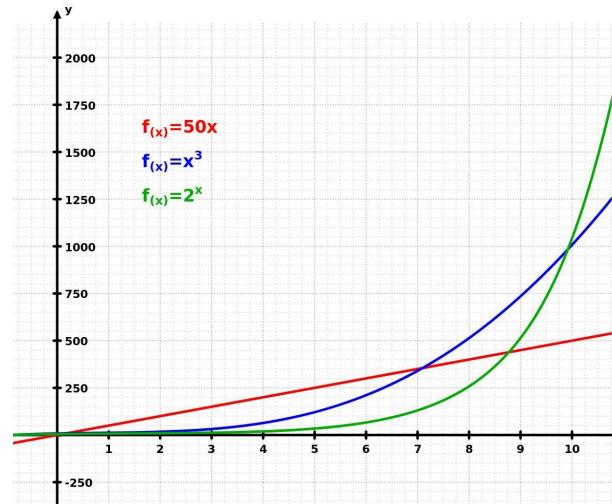
- Some representation of time? (like seq2seq?)
- Should be unique for each position – not cyclic

Possible Candidates :

$$P_{t+1} = P_t + \Delta c$$

$$P_{t+1} = e^{P_t \Delta c}$$

$$P_{t+1} = P_t e^{t \Delta c}$$



# Position Encodings

## Requirements for Positional Encodings

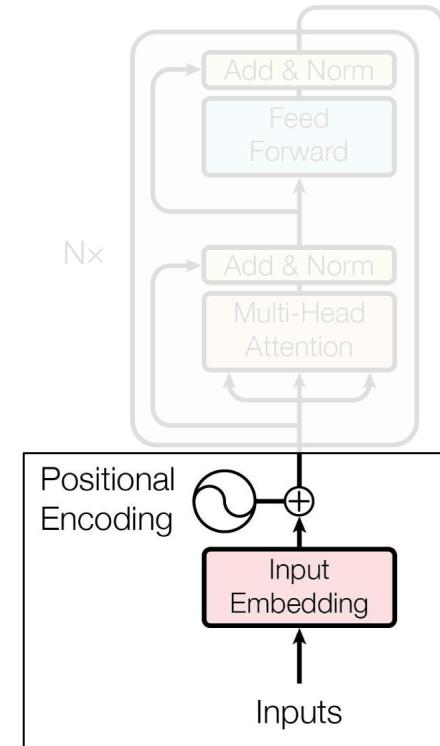
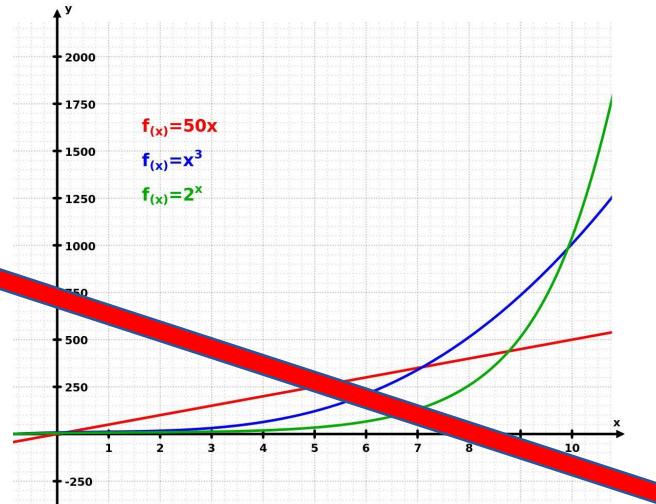
- Some representation of time? (like seq2seq?)
- Should be unique for each position – not cyclic

Possible Candidates :

$$P_{t+1} = P_t + \Delta c$$

$$P_{t+1} = P_t \cdot c$$

$$P_{t+1} = P_t^{t\Delta c}$$



# Position Encodings

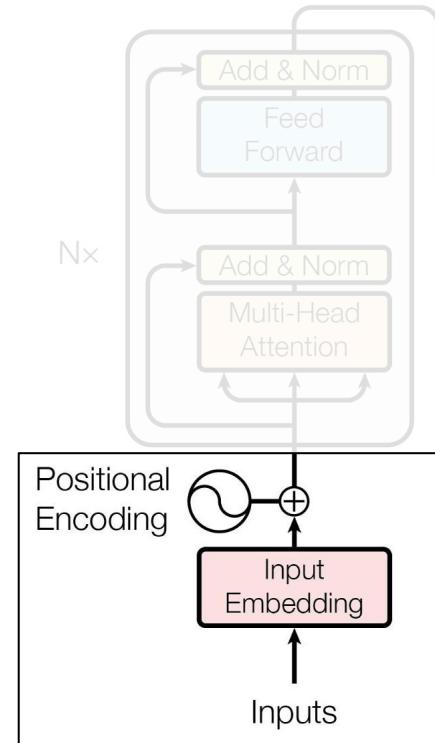
## Requirements for Positional Encodings

- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic
- **Bounded**

**Possible**

**Candidates**

$$P(t + t') = M^{t'} \times P(t)$$



# Position Encodings

## Requirements for Positional Encodings

- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic
- **Bounded**

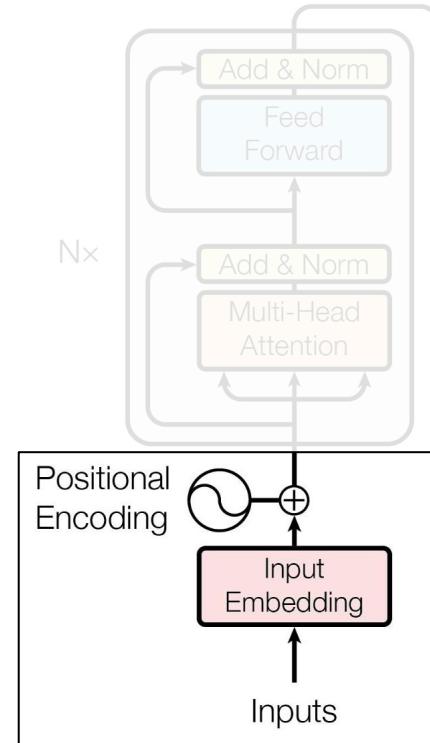
**Possible**

**Candidates**

$$P(t + t') = M^{t'} \times P(t)$$

**M?**

1. **Should be a unitary matrix**
2. **Magnitudes of eigen value should be 1 -> norm preserving**
3. **The matrix can be learnt**
4. **Produces unique rotated embeddings each time**



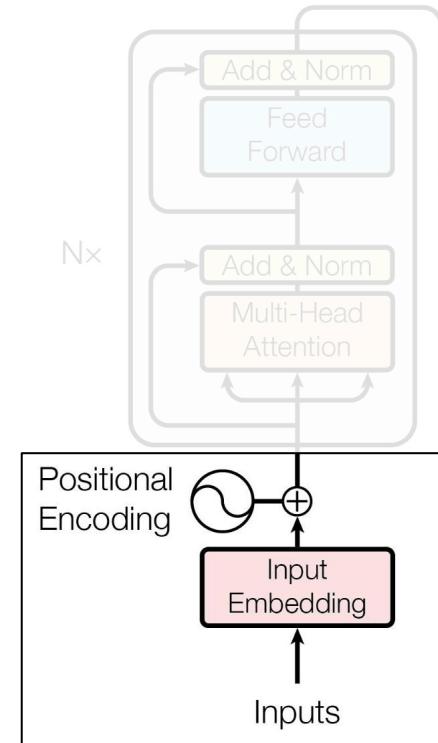
# Rotary Position Embedding

## RoFormer: ENHANCED TRANSFORMER WITH ROTARY POSITION EMBEDDING

$$f_{\{q,k\}}(\mathbf{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} \mathbf{x}_m^{(1)} \\ \mathbf{x}_m^{(2)} \end{pmatrix}$$

Table 2: Comparing RoFormer and BERT by fine tuning on downstream GLEU tasks.

Model	MRPC	SST-2	QNLI	STS-B	QQP	MNLI(m/mm)
BERTDevlin et al. [2019]	88.9	93.5	90.5	85.8	71.2	84.6/83.4
RoFormer	<b>89.5</b>	90.7	88.0	<b>87.0</b>	<b>86.4</b>	80.2/79.8



[REF: Rotary Position Embeddings](#) 

# Position Encoding

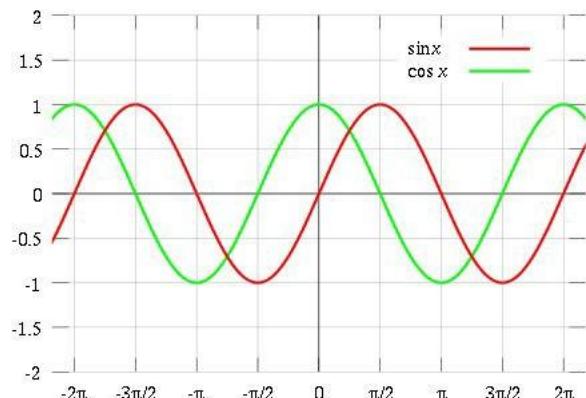
## Requirements for Position Encodings

- Some representation of time? (like seq2seq?)
- Should be unique for each position
- Bounded

## Actual Candidates

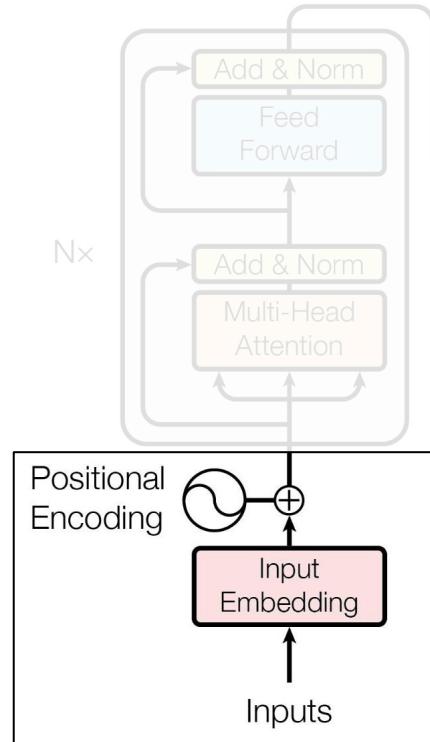
$\sin(g(t))$

$\cos(g(t))$



## Requirements for $g(t)$

- Must have same dimensions as input embeddings
- Must produce overall unique encodings



# Position Encoding

For each position, an embedded input is moved the same distance but at a different angle. **Inputs that are close to each other in the sequence have similar perturbations, but inputs that are far apart are perturbed in different directions.**

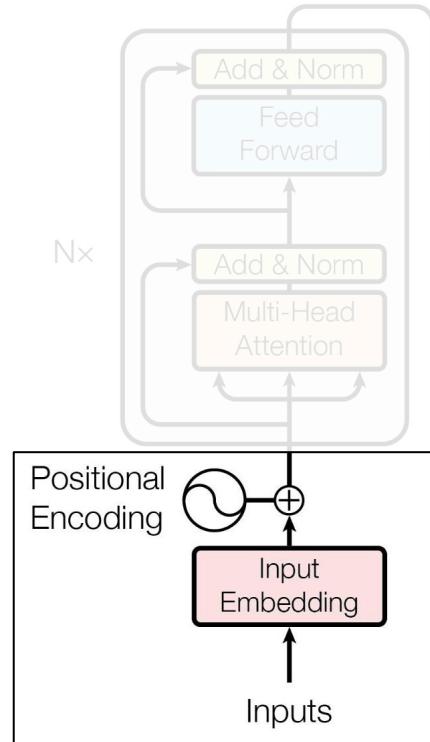
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

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

**pos** -> idx of the token in input sentence

**i** -> i<sup>th</sup> dimension out of d

**d model** -> embedding dimension of each token  
 Different calculations for odd and even embedding indices



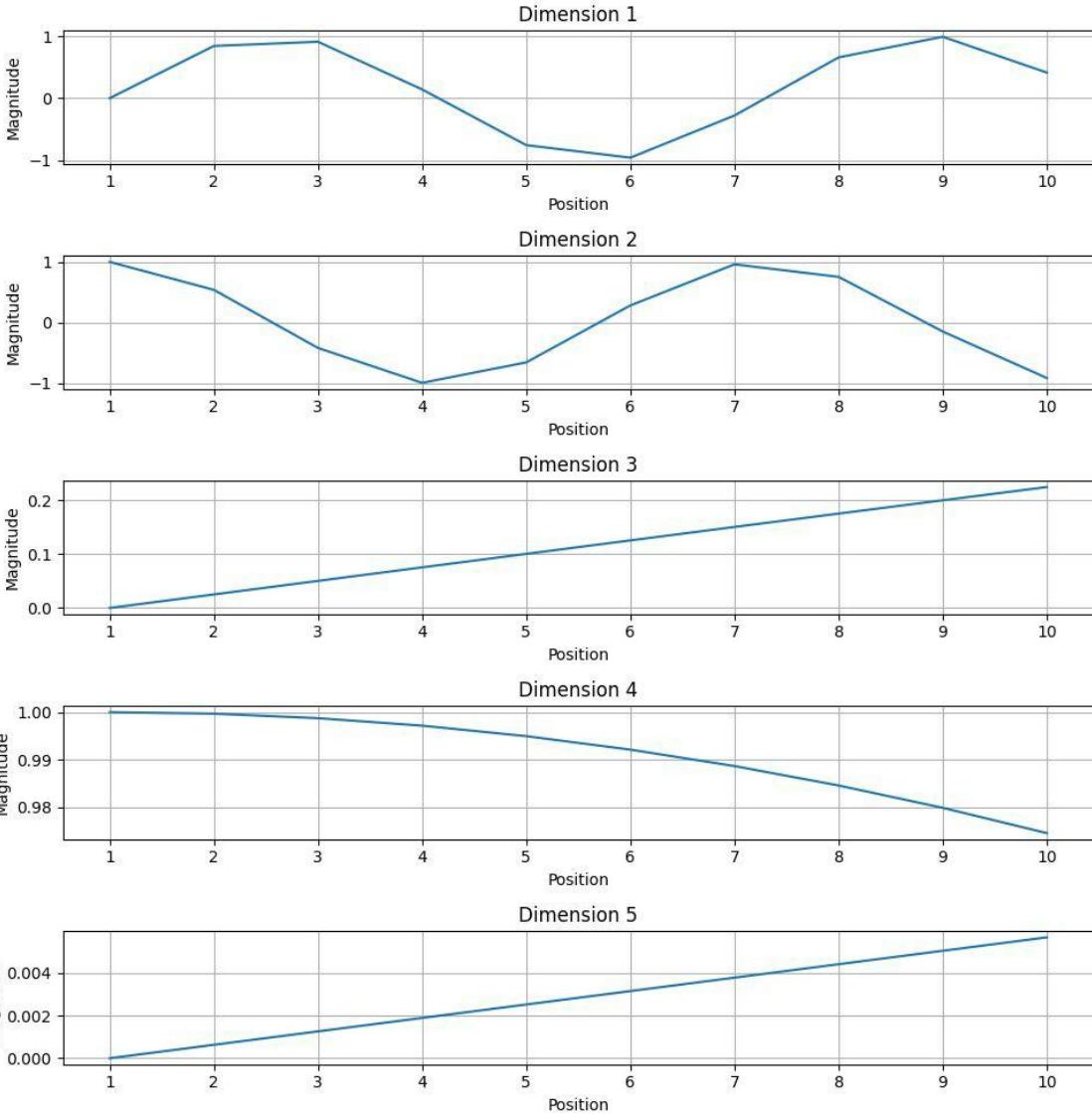
# Position Encoding



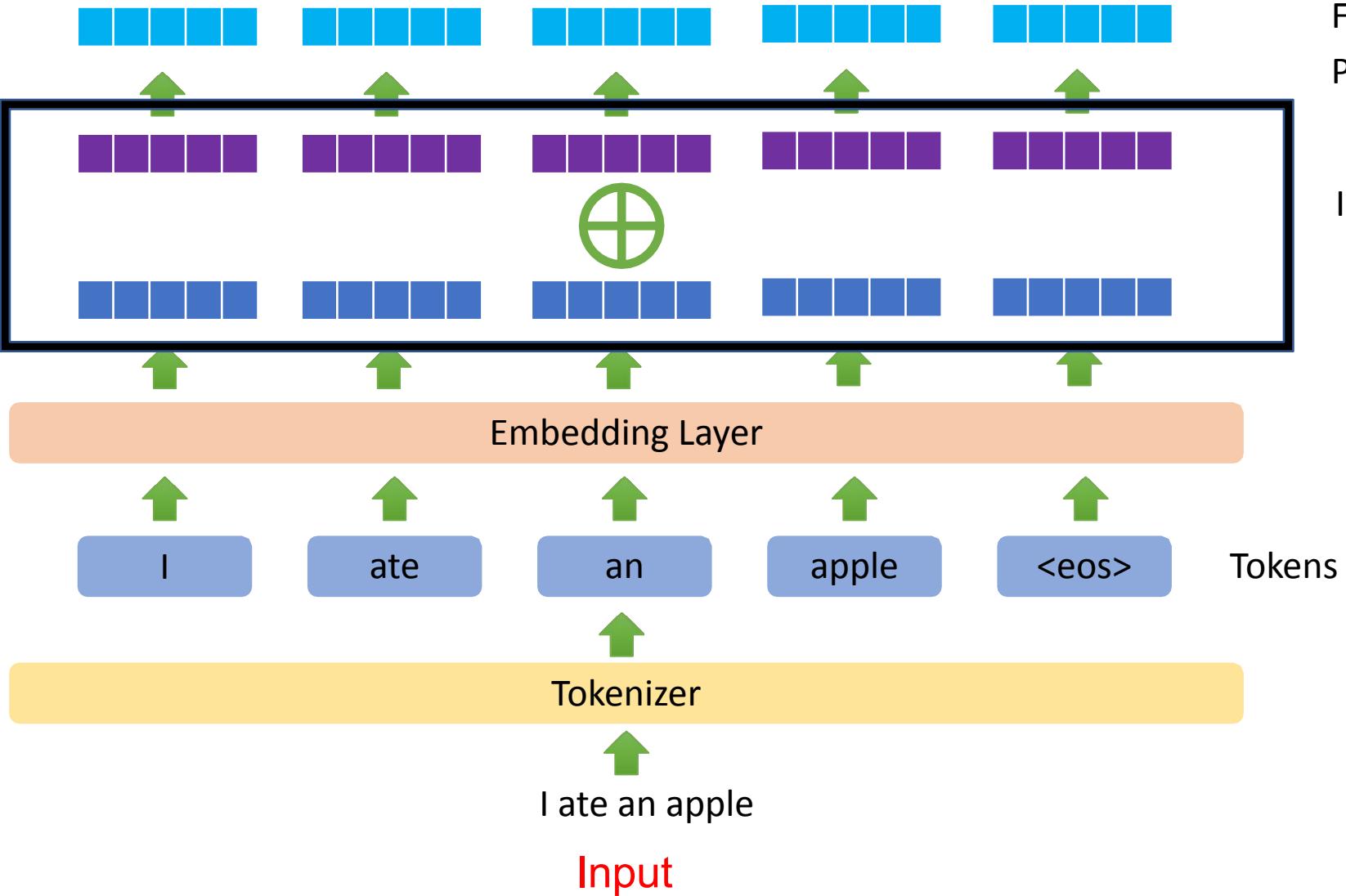
**Positional Encoding:**

	0	1	2	3	4
Dim 1	0.000	0.841	0.909	0.141	-0.757
Dim 2	1.000	0.540	-0.416	-0.990	-0.654
Dim 3	0.000	0.025	0.050	0.075	0.100
Dim 4	1.000	1.000	0.999	0.997	0.995
Dim 5	0.000	0.001	0.001	0.002	0.003

**Positional Encoding**

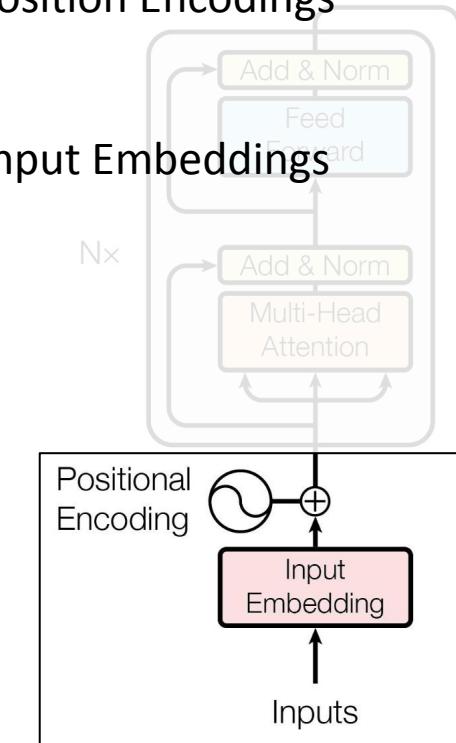


# Position Encoding



Final Input Embeddings

Position Encodings



## ► **Learnable position embeddings:**

- Position embeddings are parameters learned during training.
- Allow the model to adapt position representations to the task.

## ► **Rotary Position Embeddings (RoPE):**

- Encodes positions by rotating query and key vectors in multi-head attention.
- Enables better extrapolation to longer sequences.

## ► **ALiBi (Attention with Linear Biases):**

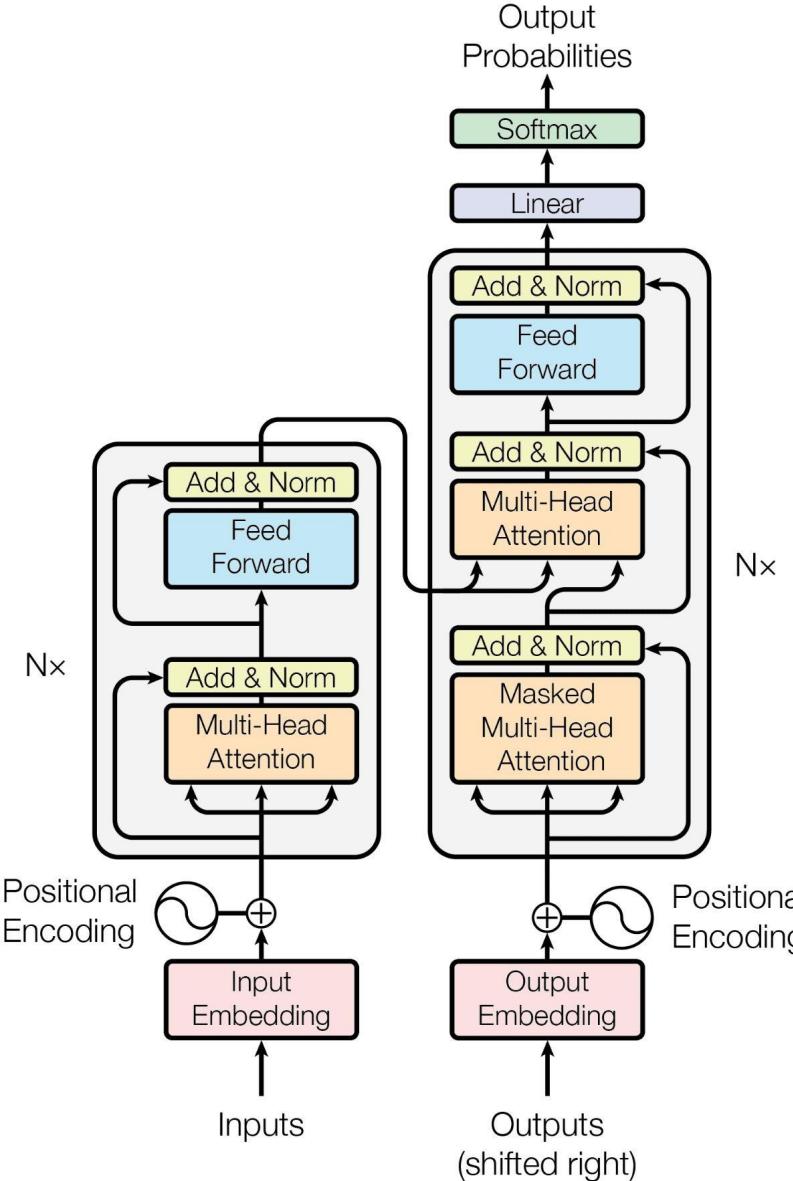
- Adds linear biases to attention scores based on distance between tokens.
- Encourages attention to nearby tokens without explicit position embeddings.

- ▶ **Multi-head Attention**
- ▶ **Add & Norm**
- ▶ **Feed-forward Layer**
- ▶ **Add & Norm**

## Note

Residual connections and layer normalization are crucial for effective training.

# Transformer Block

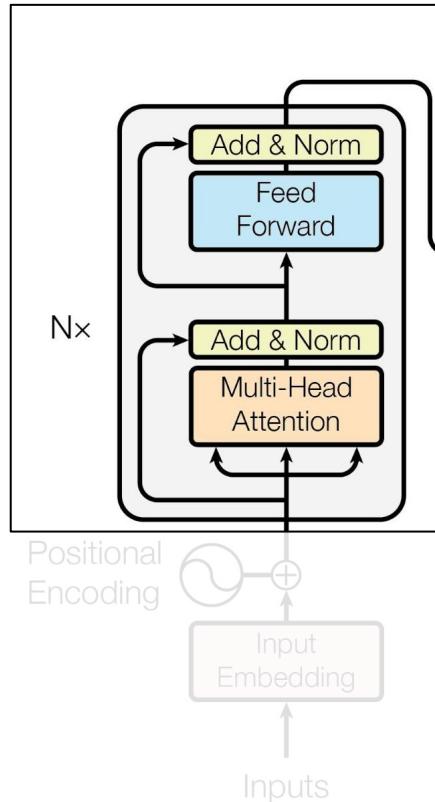
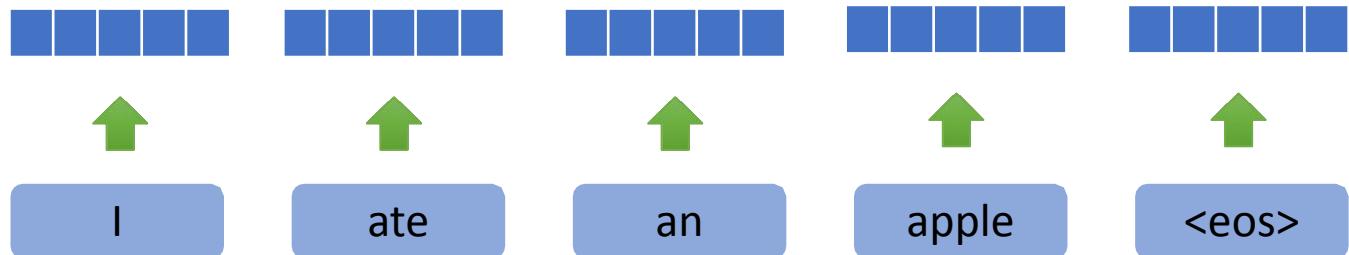


# Encoder-Decoder Architecture

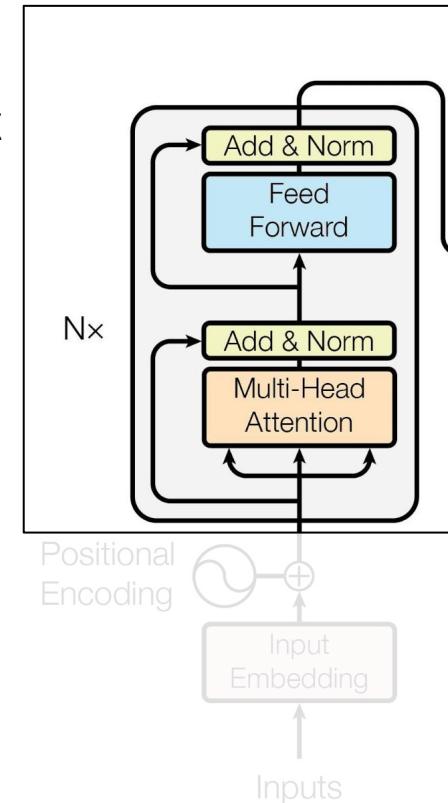
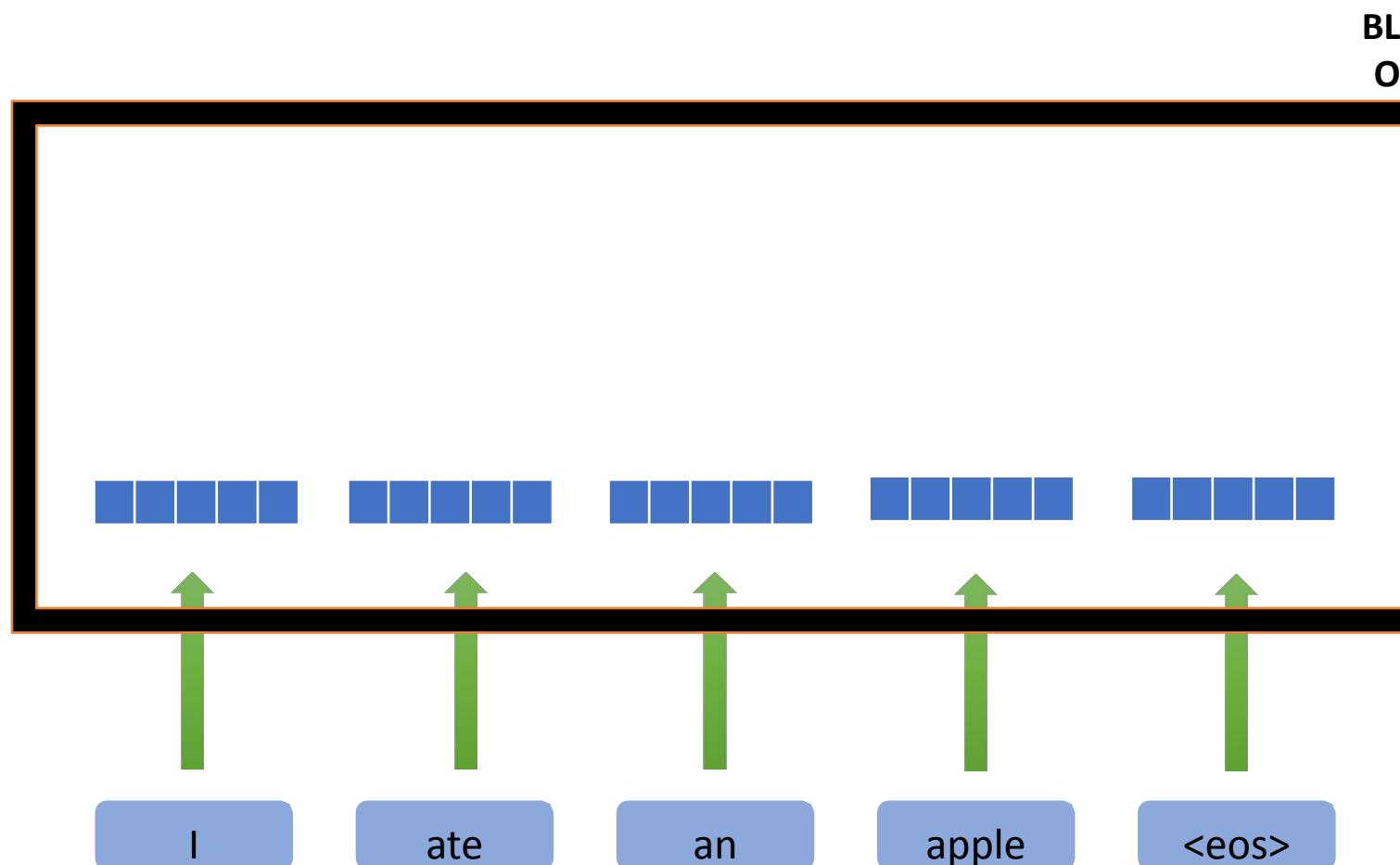
- ▶ **Encoder:** Stack of self-attention and feed-forward (FF) layers.
- ▶ **Decoder:** Adds masked self-attention and encoder-decoder attention on top of FF layers.
- ▶ **Applications:** Widely used in tasks like translation, summarization, and more.

# Encoder

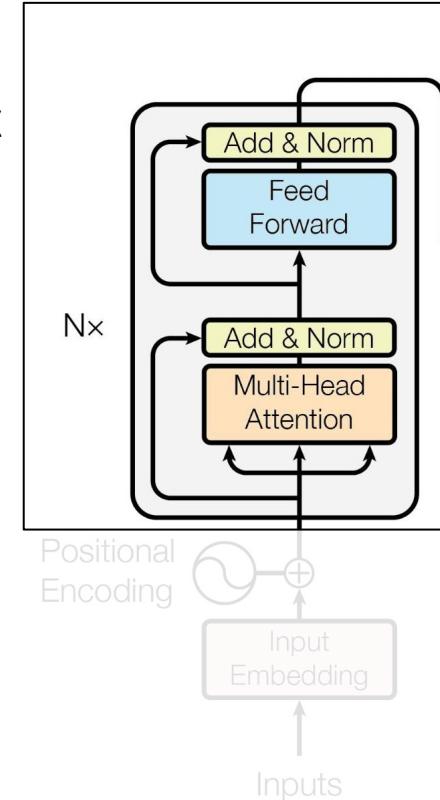
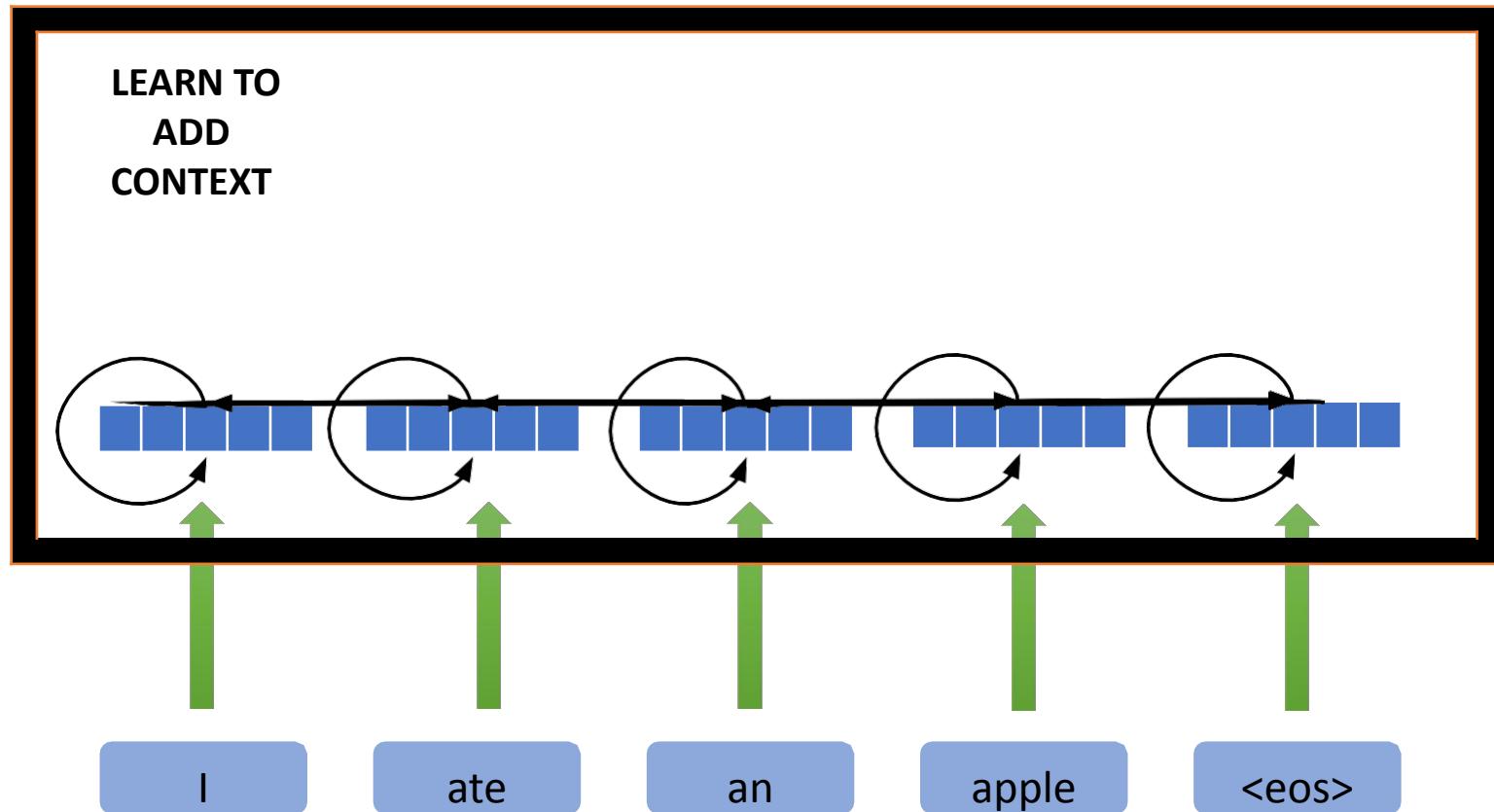
WHERE IS THE  
CONTEXT?



# Encoder

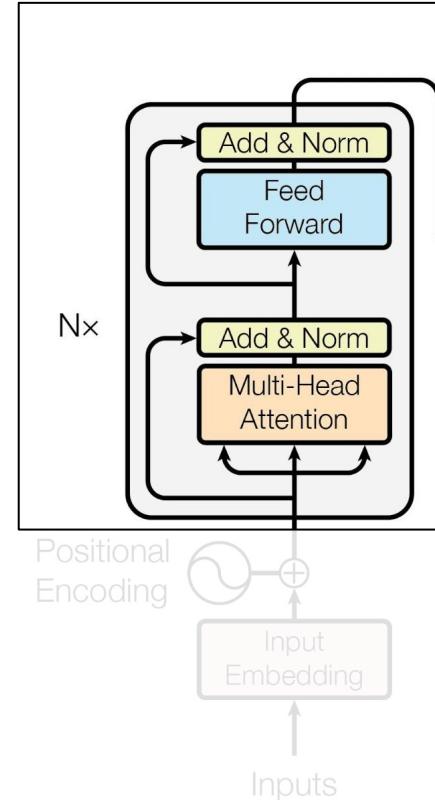
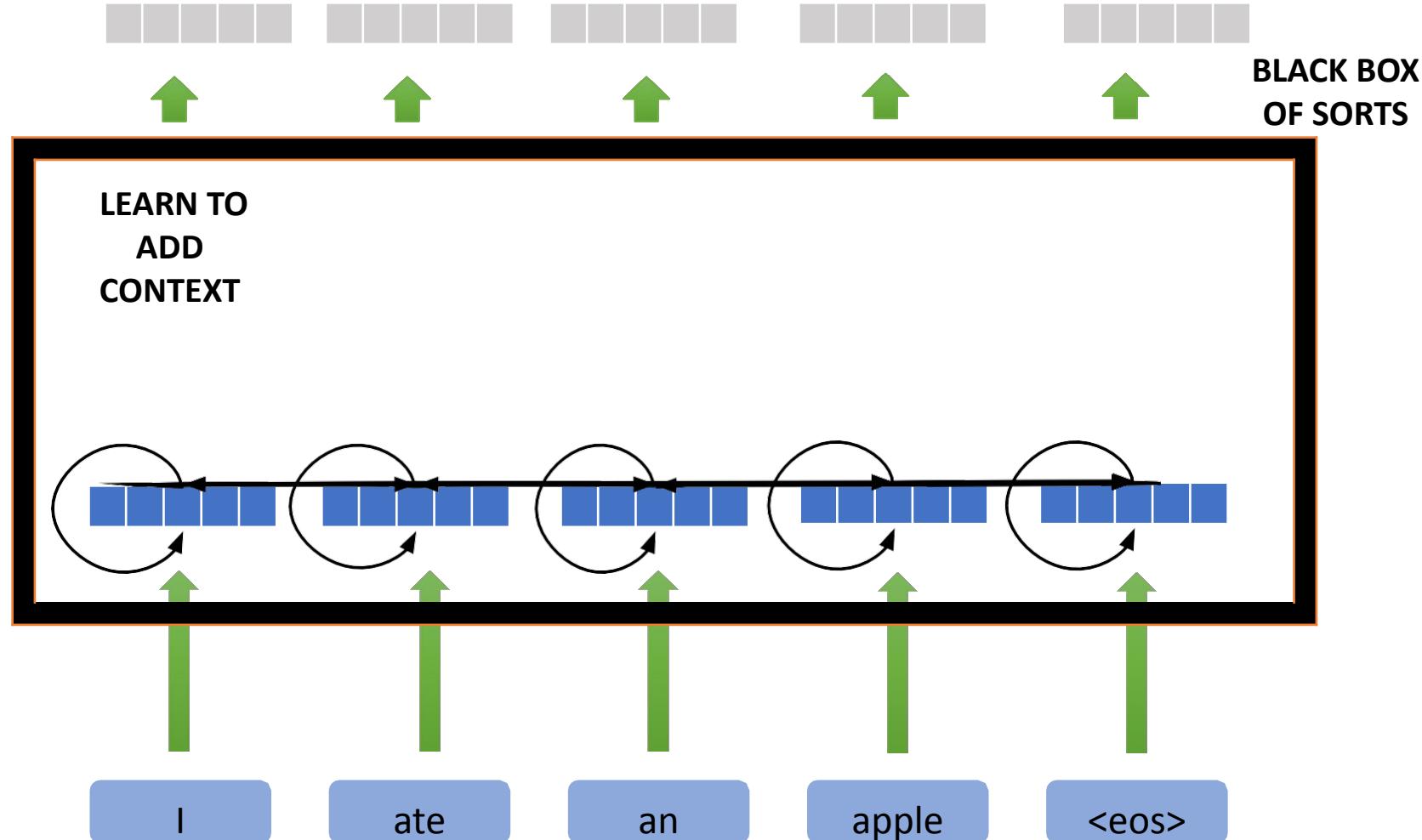


# Encoder



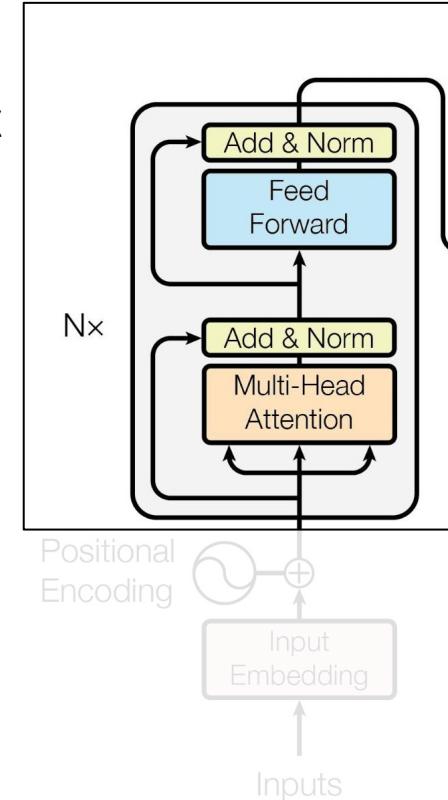
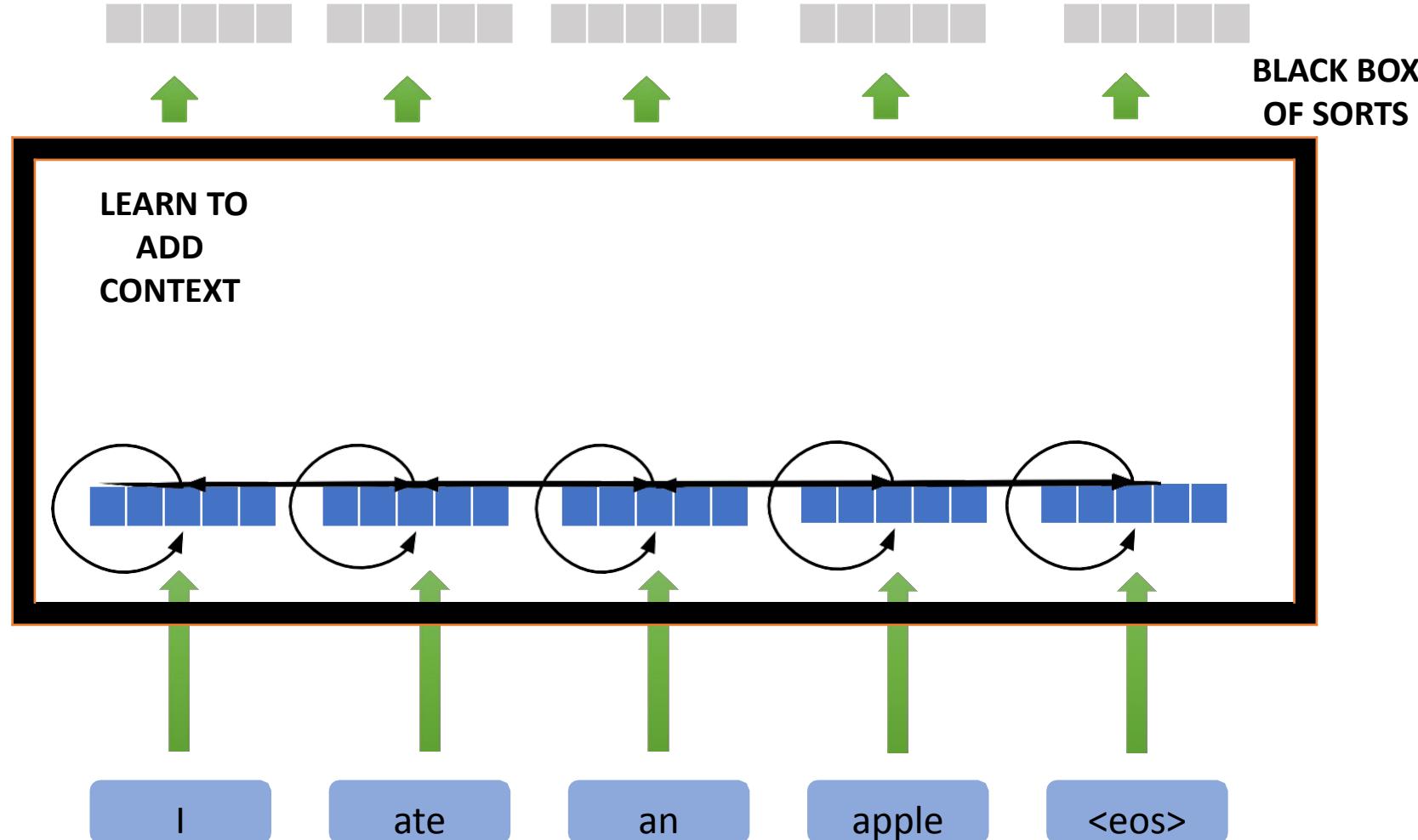
# Encoder

## CONTEXTUALLY RICH EMBEDDINGS



# Encoder

## CONTEXTUALLY RICH EMBEDDINGS



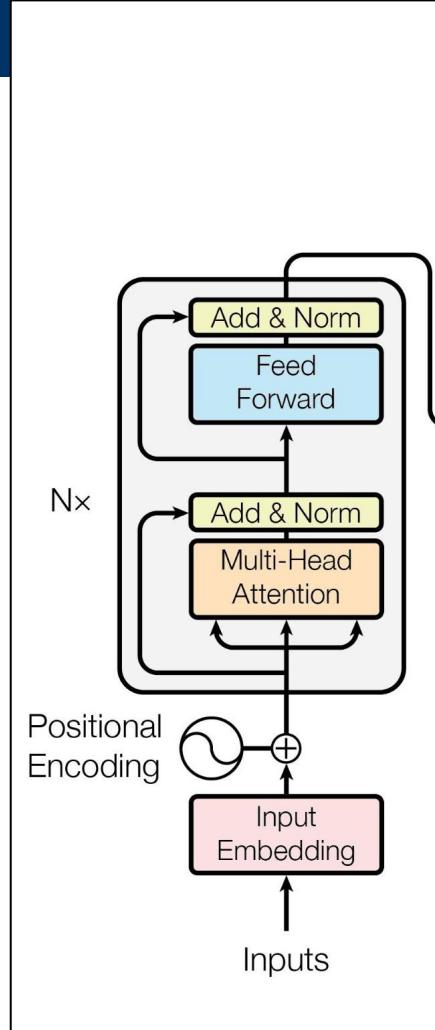
# Encoder-Only Transformers

- ▶ **Example:** BERT
- ▶ Only the encoder stack is used.
- ▶ **Applications:** Classification, question answering, embeddings.

# Encoders

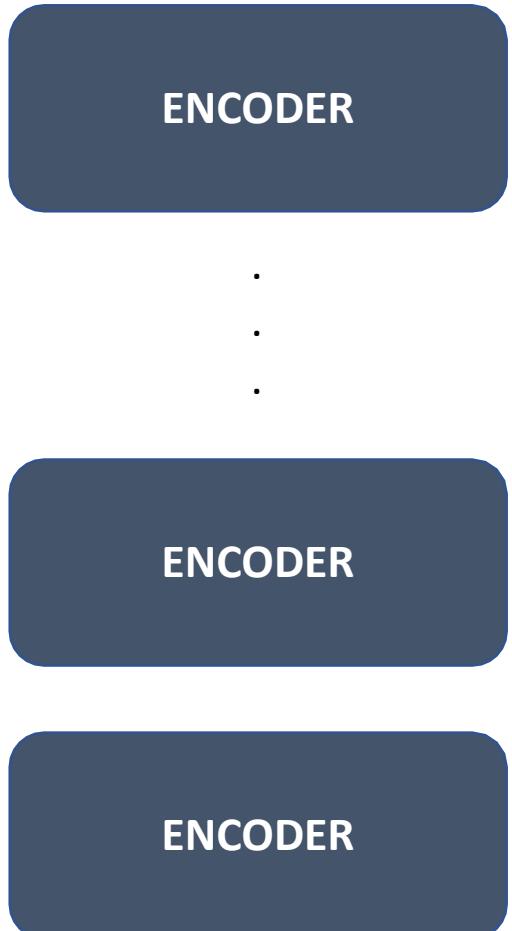
## Encoder

ENCODER

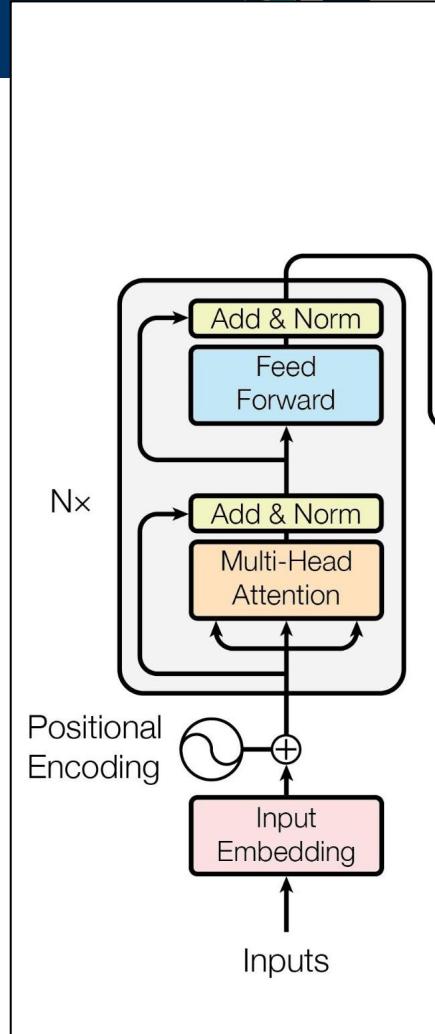


# Encoders

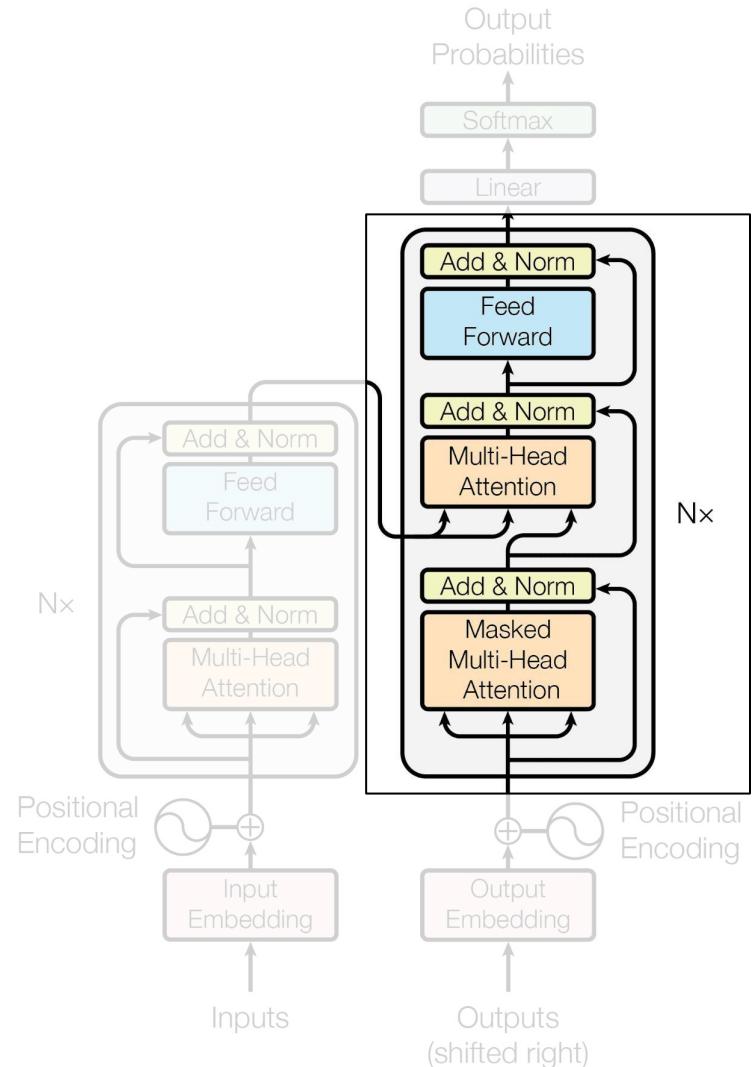
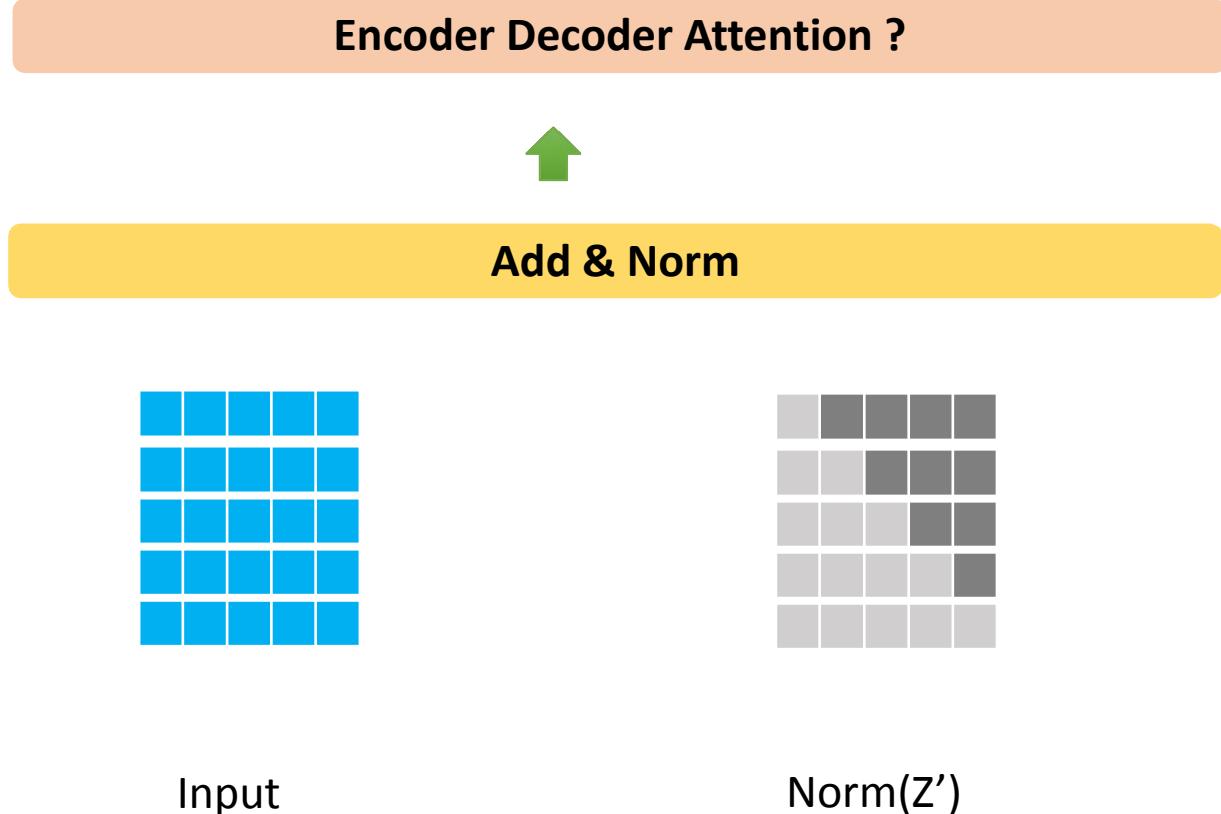
## Encoder



Input to Encoder<sub>i+1</sub>  
↑  
Output from Encoder<sub>i</sub>

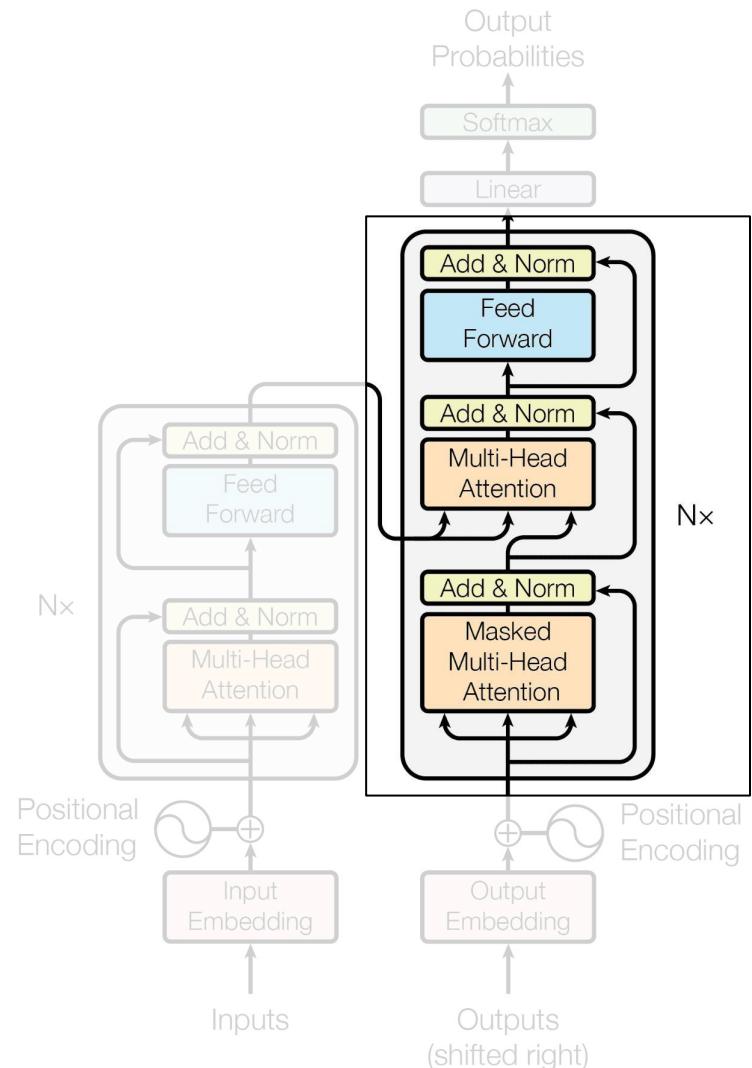


# Encoder Decoder Attention



# Encoder Decoder Attention

**Encoder Decoder Attention ?**



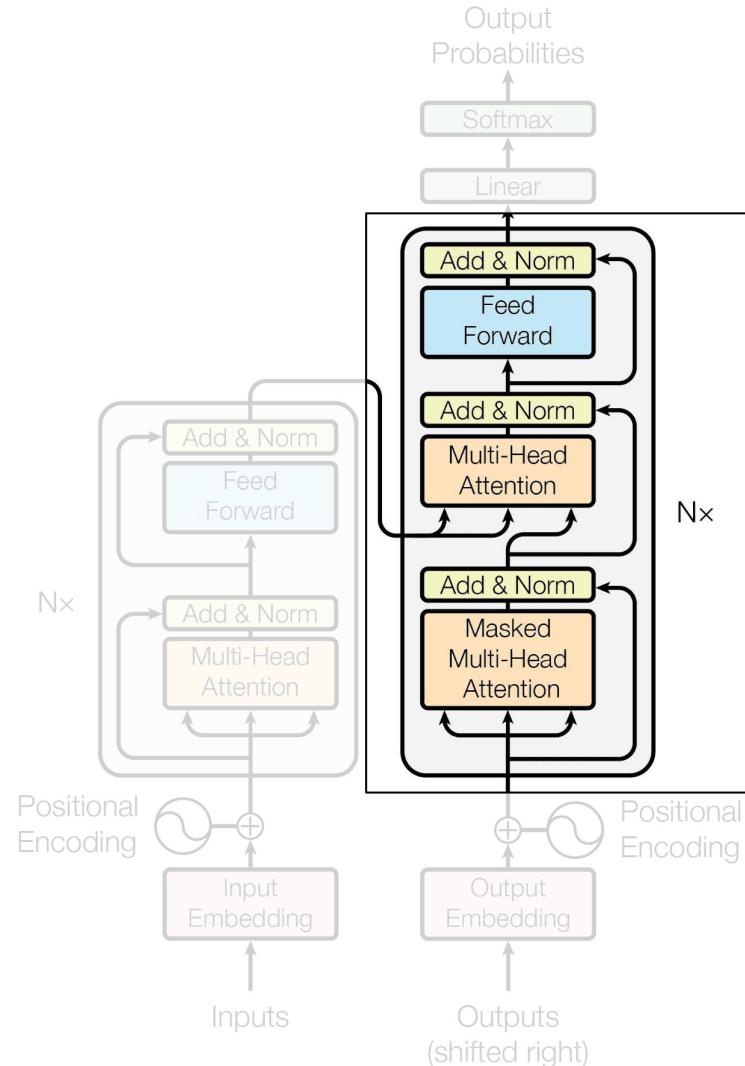
# Encoder Decoder Attention

## Encoder Self Attention

1. Queries from Encoder Inputs
2. Keys from Encoder Inputs
3. Values from Encoder Inputs

## Decoder Masked Self Attention

1. Queries from Decoder Inputs
2. Keys from Decoder Inputs
3. Values from Decoder Inputs



# Encoder Decoder Attention

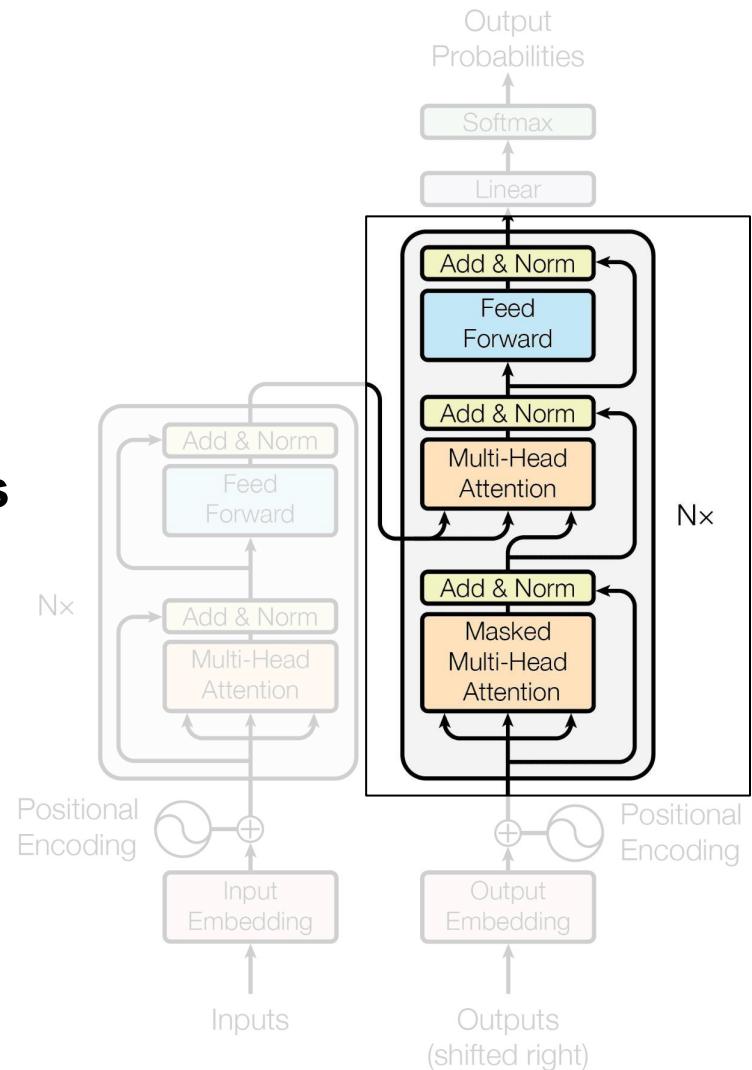
## Encoder

Keys from **Encoder Outputs**  
Values from **Encoder Outputs**

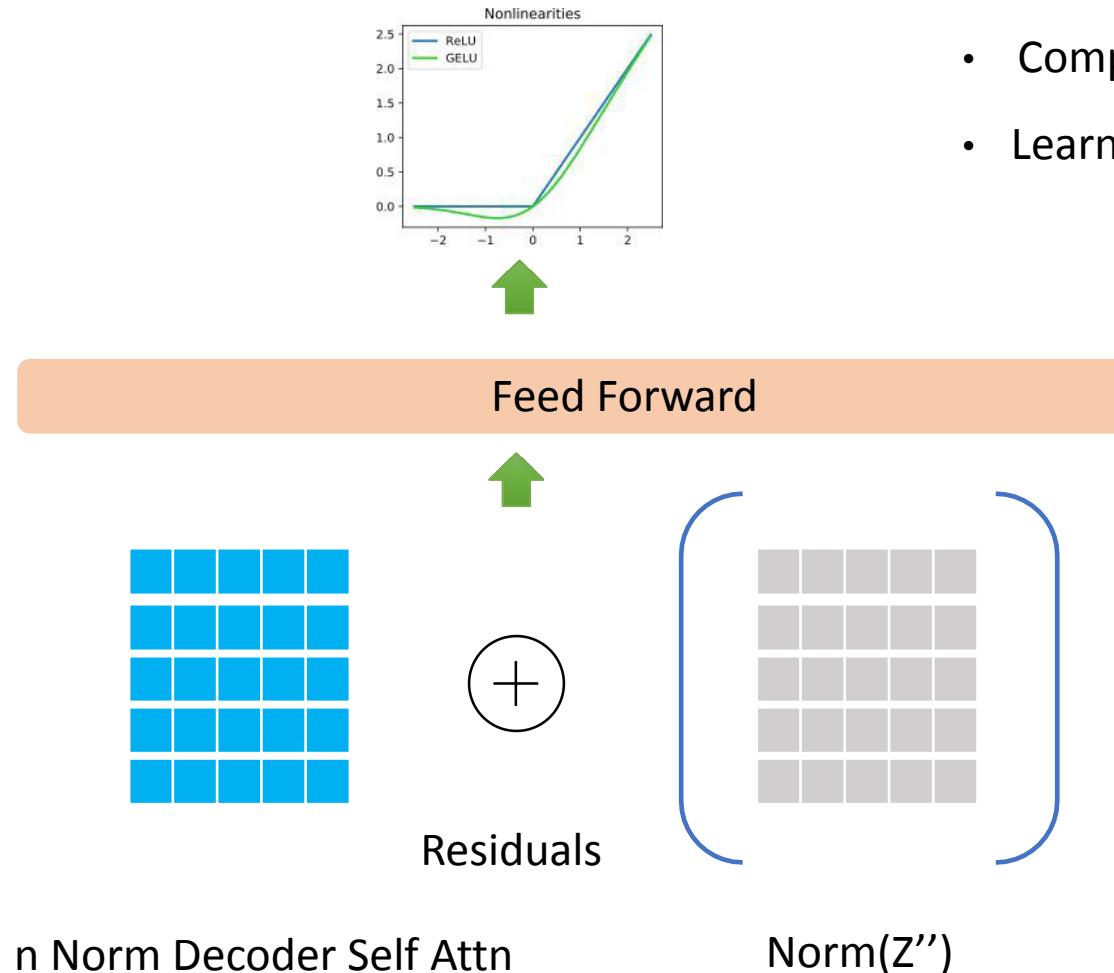
NOTE: Every decoder block receives the same FINAL encoder output

## Decoder

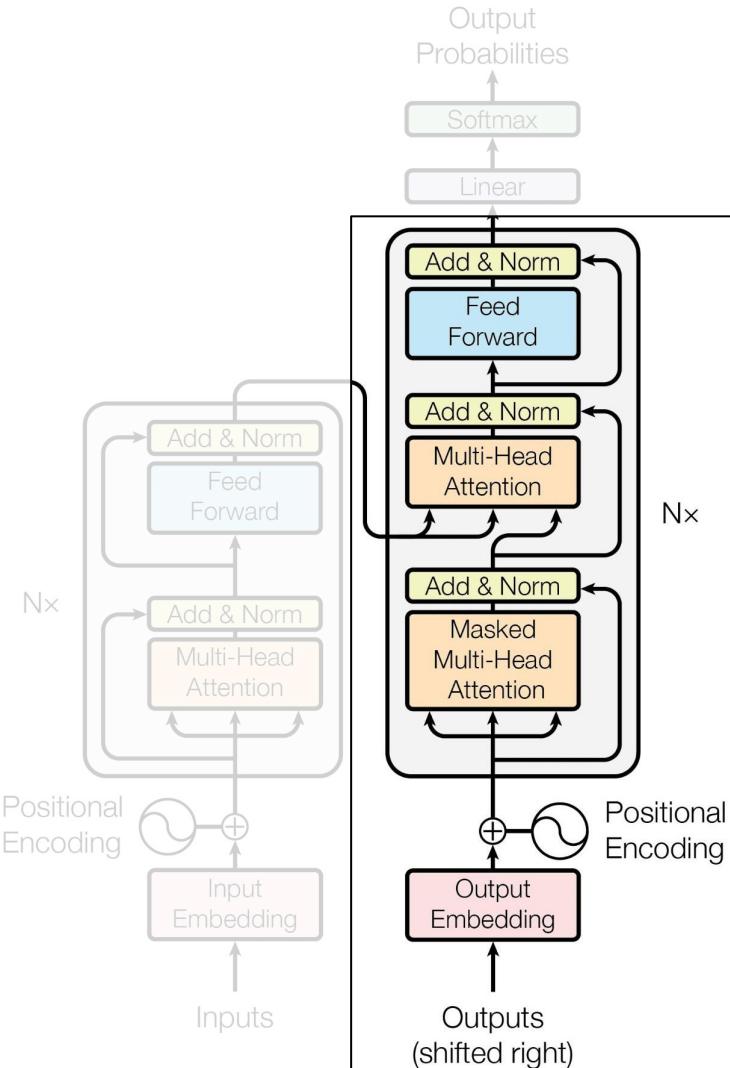
Queries from **Decoder Inputs**



# Encoder Decoder Attention



- Non Linearity
- Complex Relationships
- Learn from each other

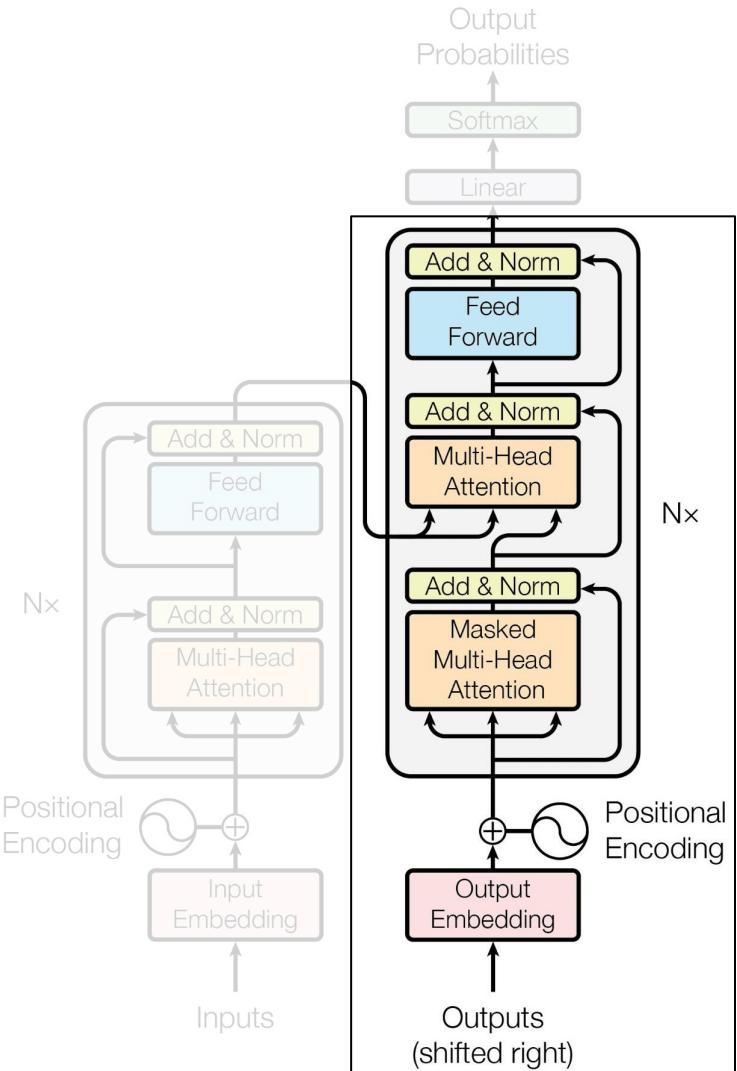


# Decoder-Only Transformers

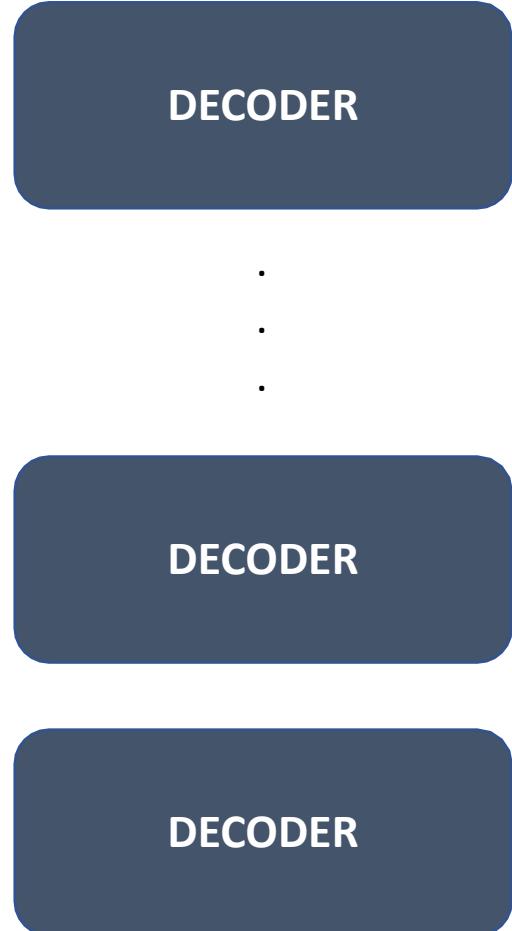
- ▶ **Example:** GPT, LLaMA
- ▶ Uses only the decoder block.
- ▶ **Masked self-attention (causal):** Each token can only attend to previous tokens.
- ▶ **Applications:** Language modeling, text generation.

# Decoder

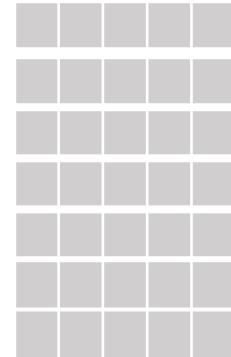
**DECODER**



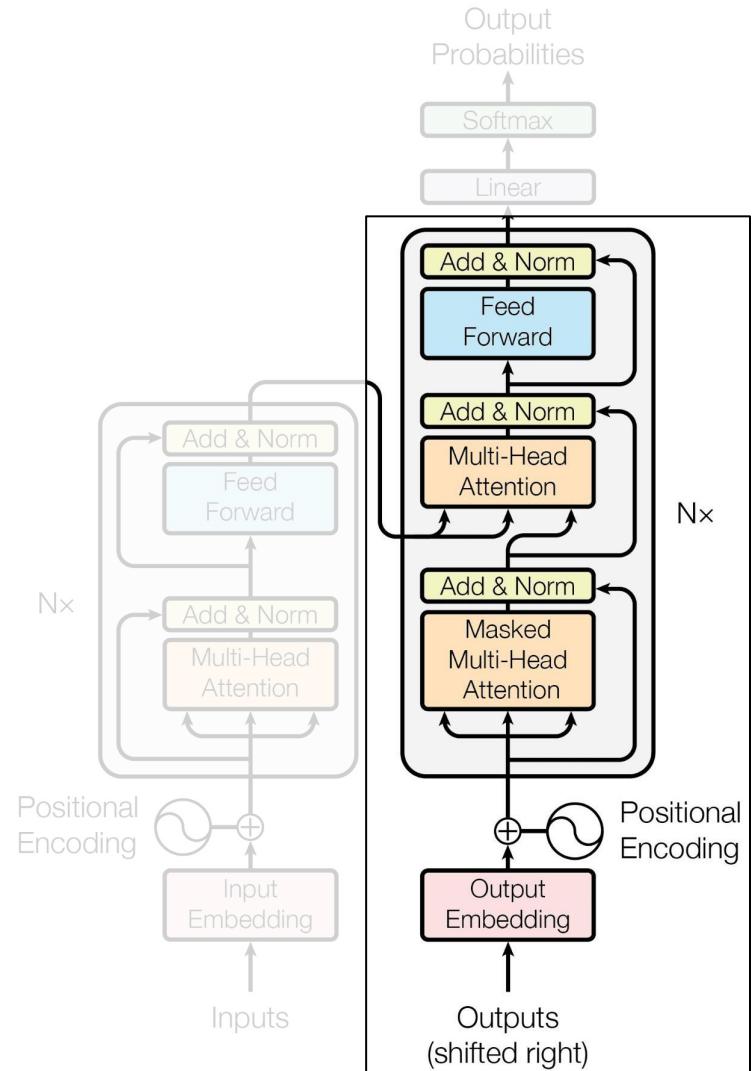
# Decoder

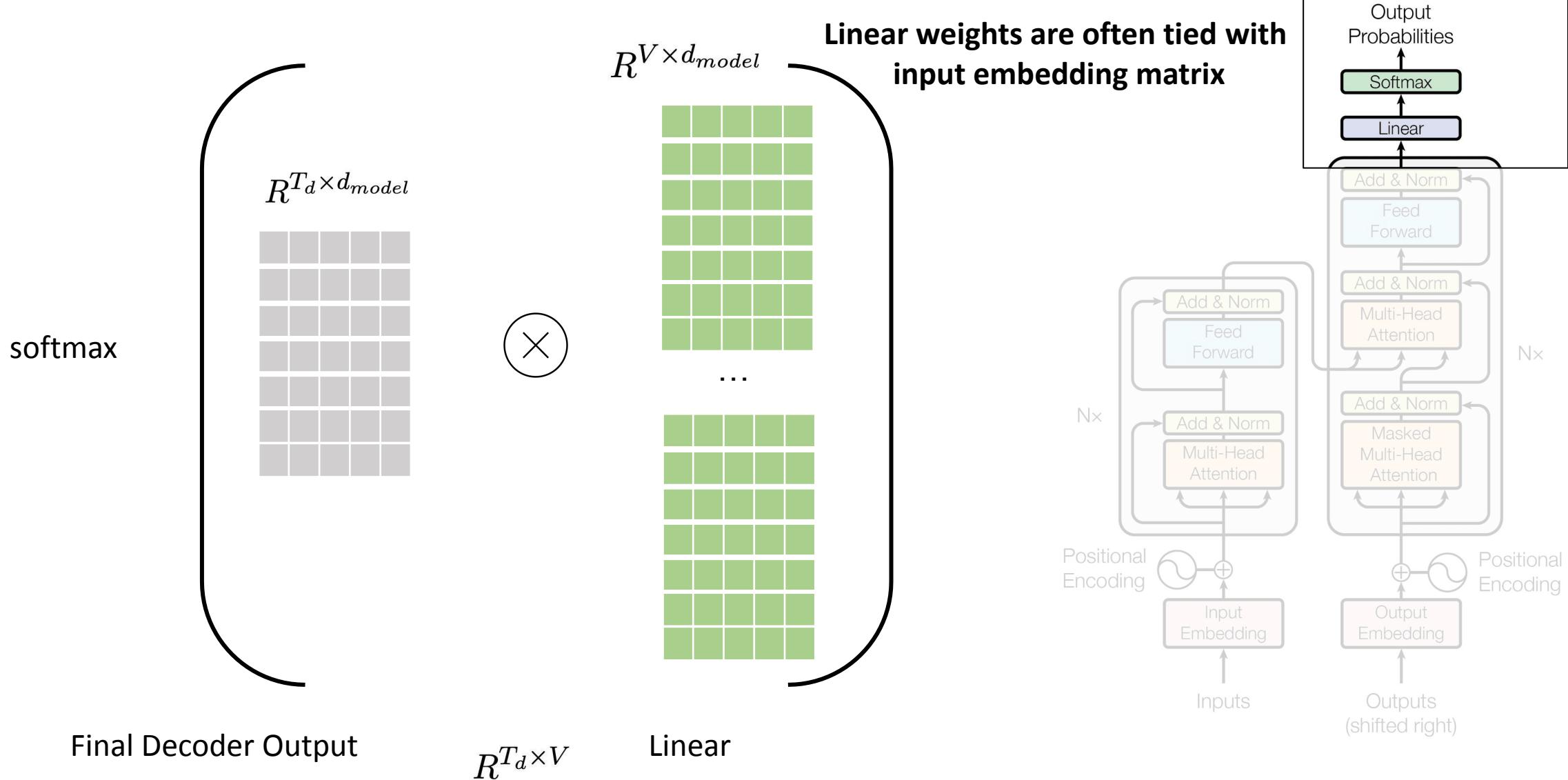


$$R^{T_d \times d_{model}}$$

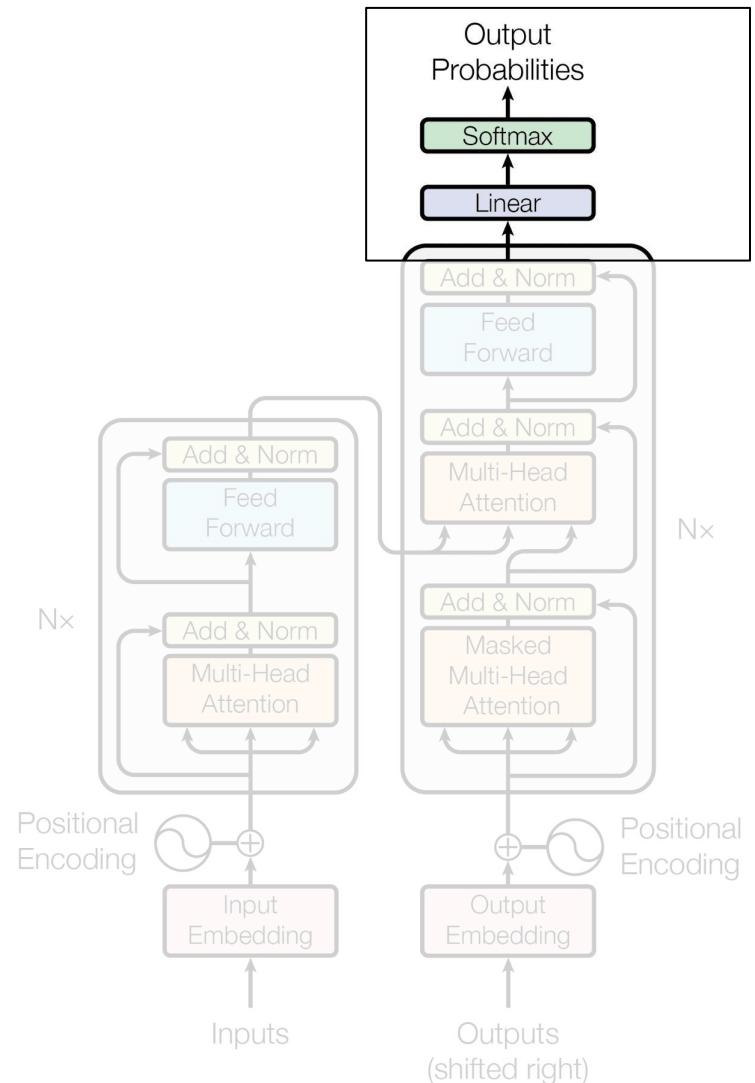
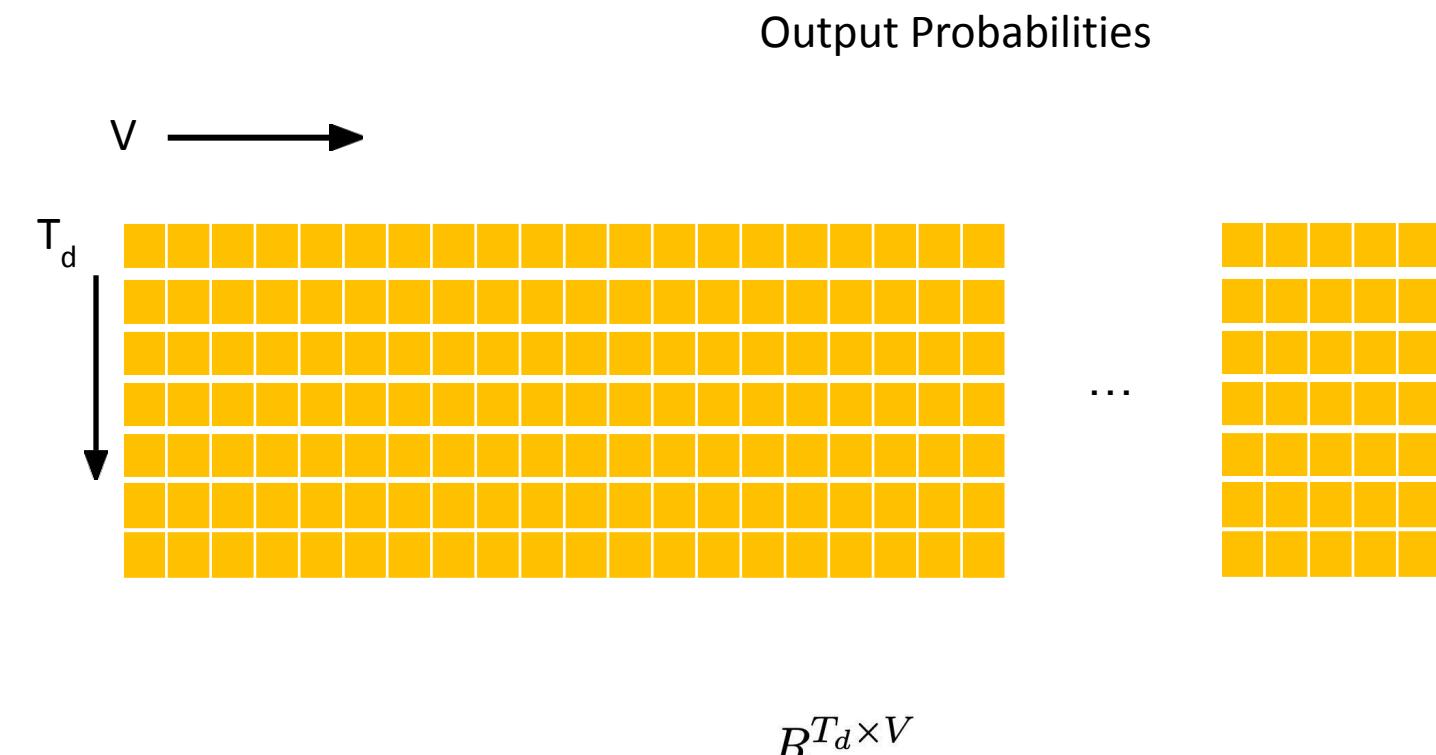


Decoder output

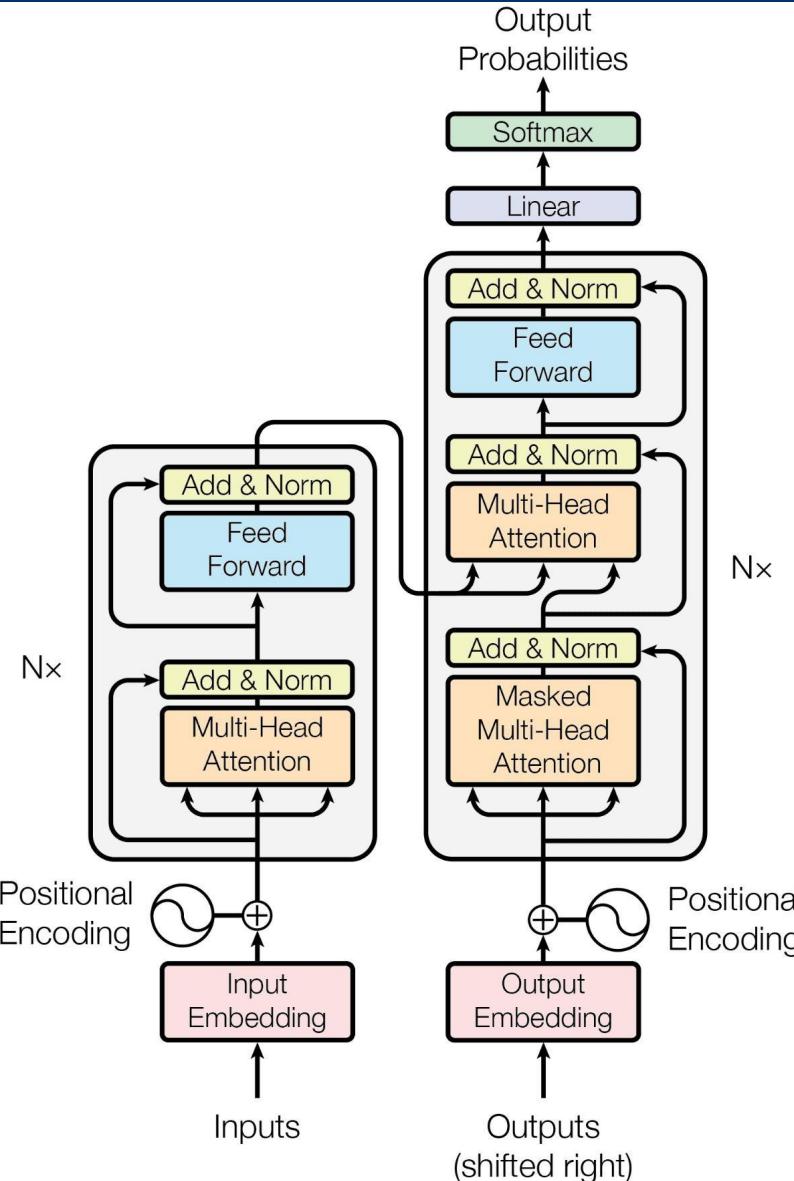




# Softmax



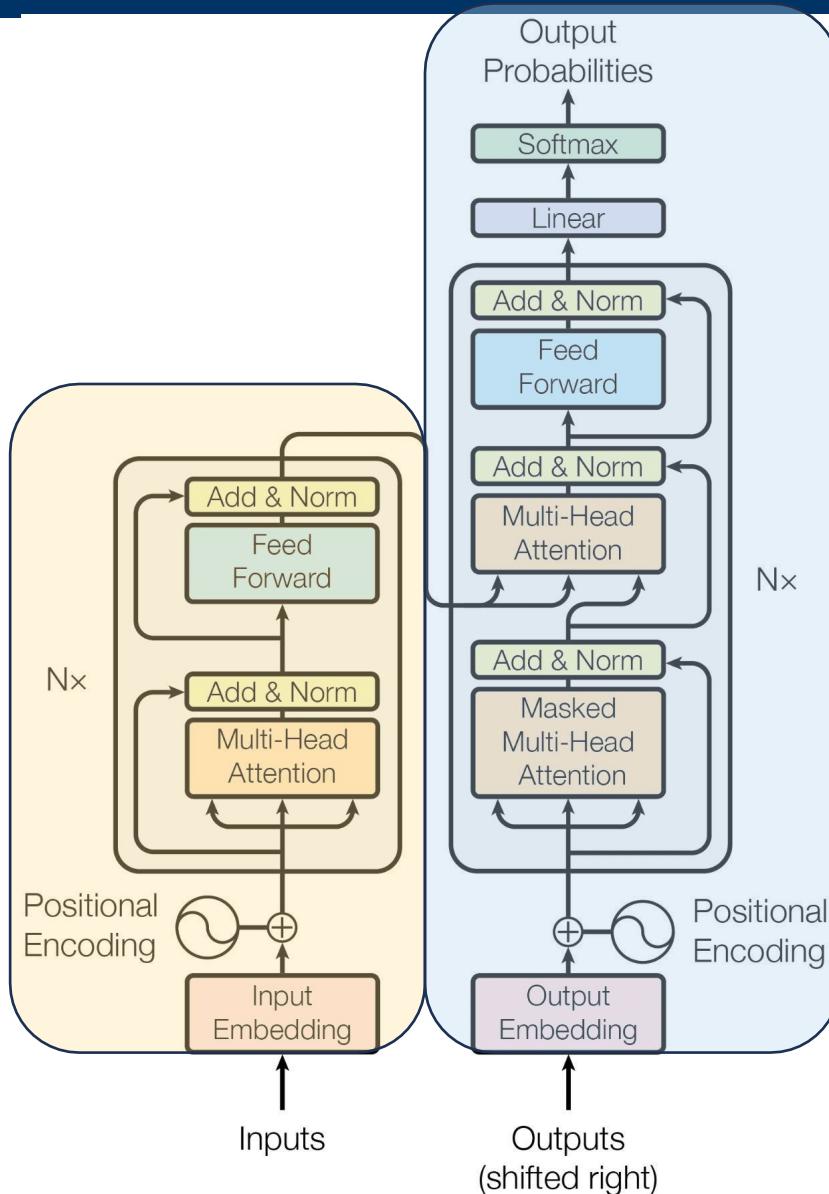
# Transformers



# Transformers



**Representation**

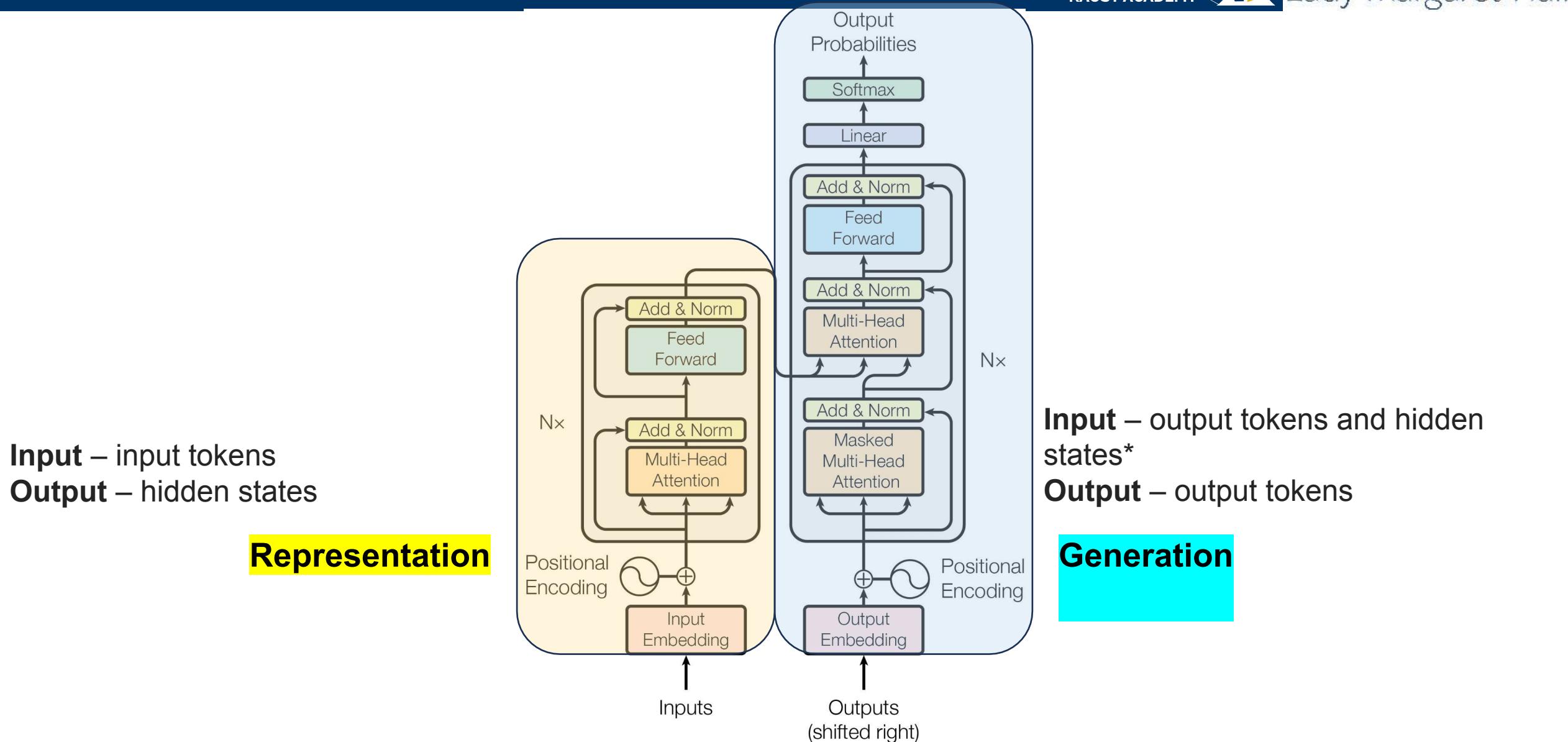


**Generation**

# Transformers



أكاديمية كاوست  
KAUST ACADEMY



# Transformers

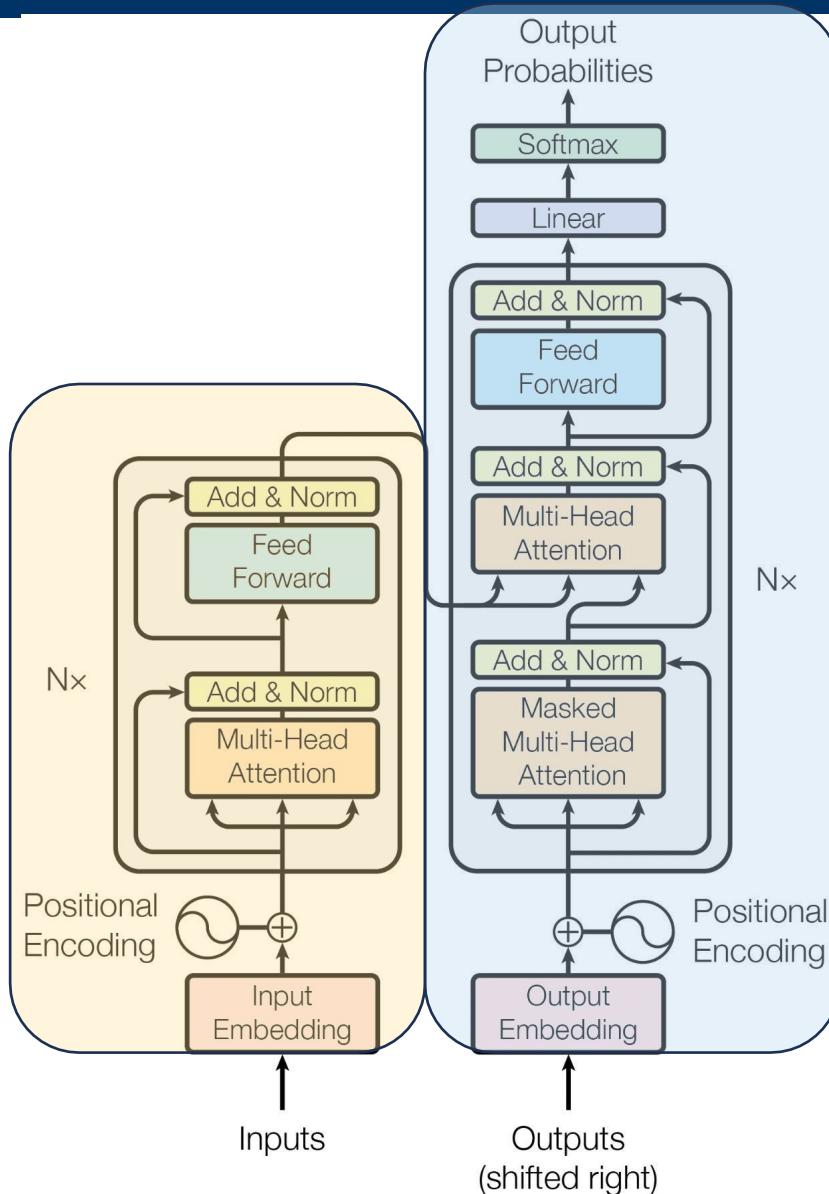


أكاديمية كاوفست  
KAUST ACADEMY

**Input** – input tokens  
**Output** – hidden states

**Model can see all timesteps**

**Representation**



**Input** – output tokens and hidden states\*

**Output** – output tokens

**Model can only see previous timesteps**

**Generation**

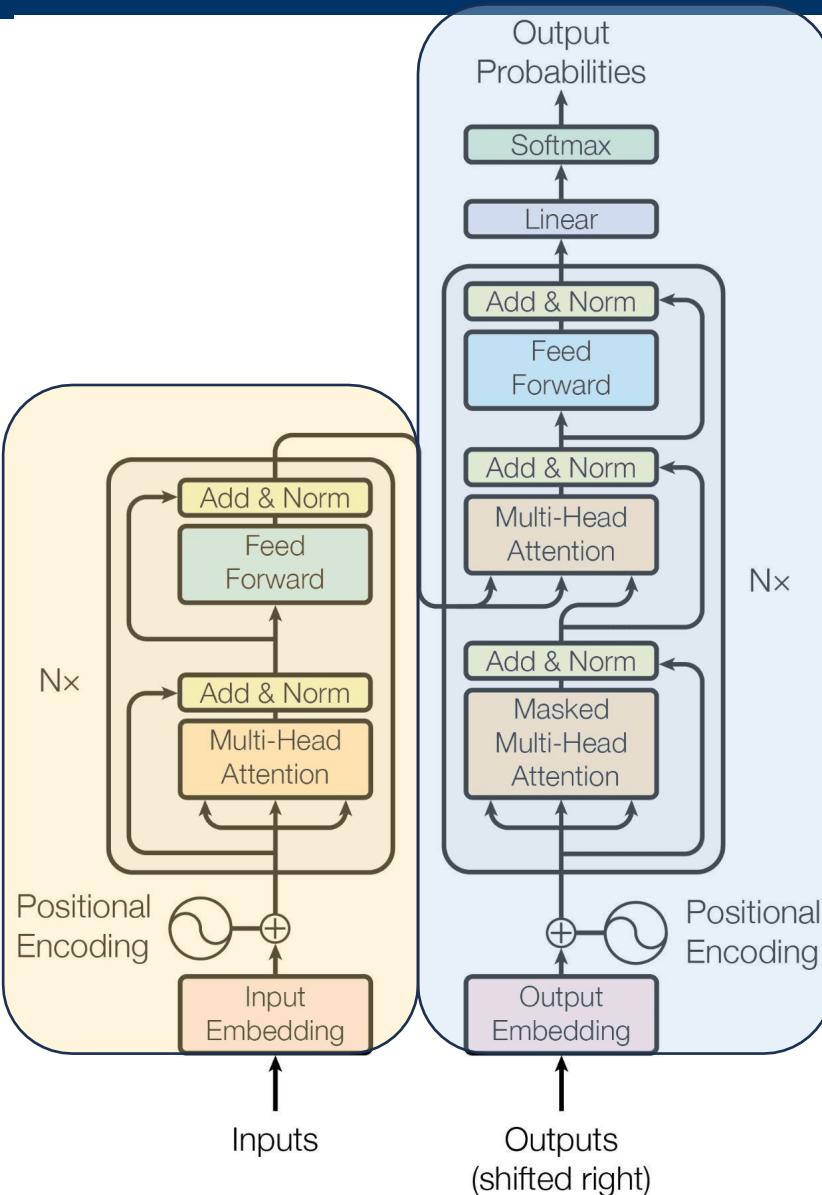
# Transformers

**Input** – input tokens  
**Output** – hidden states

**Model can see all timesteps**

**Does not usually output tokens, so no inherent auto-regressivity**

## Representation



**Input** – output tokens and hidden states\*

**Output** – output tokens

**Model can only see previous timesteps**

**Model is auto-regressive with previous timesteps' outputs**

## Generation

# Transformers

**Input** – input tokens

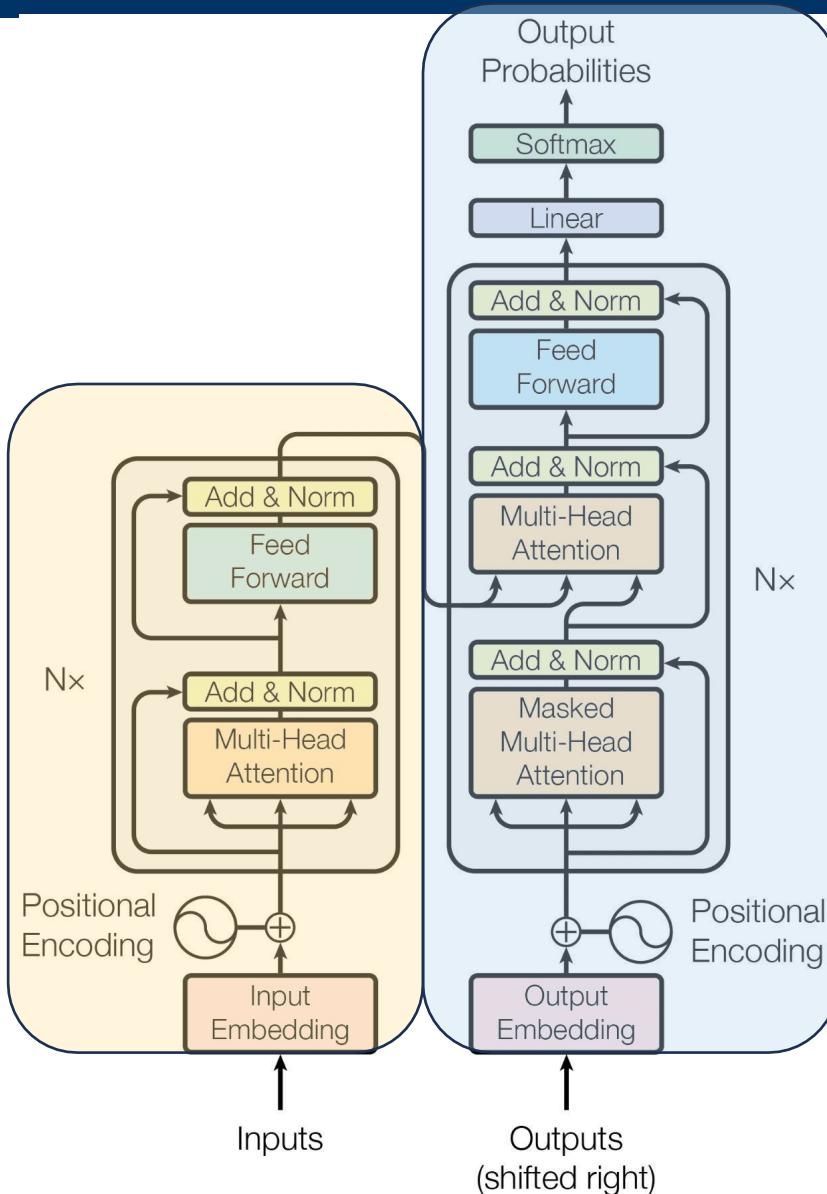
**Output** – hidden states

**Model can see all timesteps**

**Does not usually output tokens, so no inherent auto-regressivity**

**Can also be adapted to generate tokens by appending a module that maps hidden state dimensionality to vocab size**

**Representation**



**Input** – output tokens and hidden states\*

**Output** – output tokens

**Model can only see previous timesteps**

**Model is auto-regressive with previous timesteps' outputs**

**Can also be adapted to generate hidden states by looking before token outputs**

**Generation**

- ▶ **Purpose:** Prevents the model from attending to future tokens in the sequence.
- ▶ **Implementation:** Uses a triangular mask in the self-attention mechanism.
- ▶ **Effect:** Ensures that predictions for a token only depend on previous tokens, maintaining the autoregressive property.

# Casual Masking

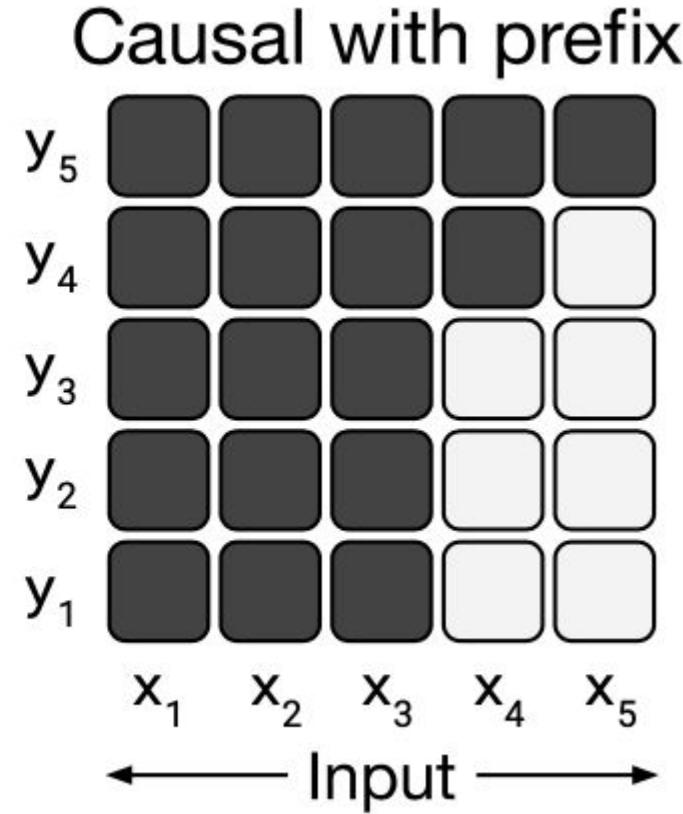
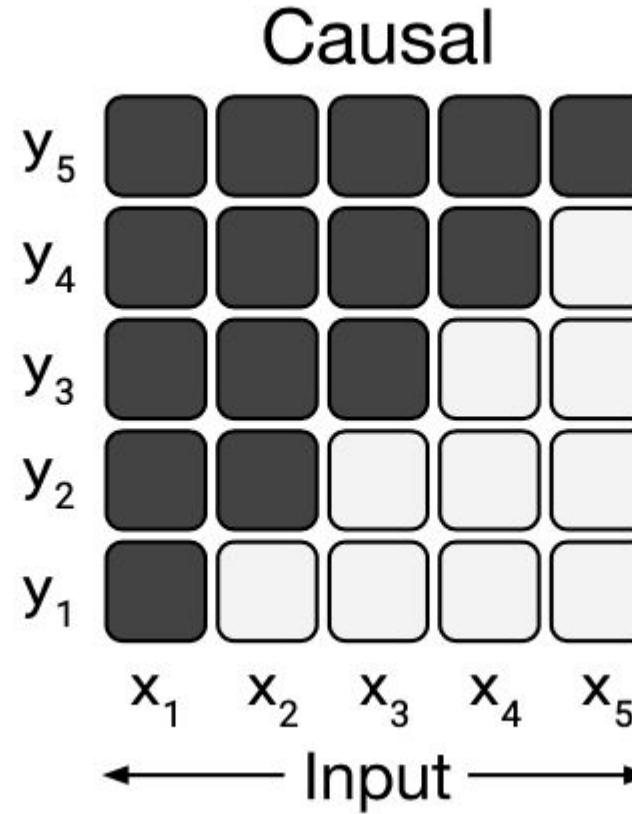
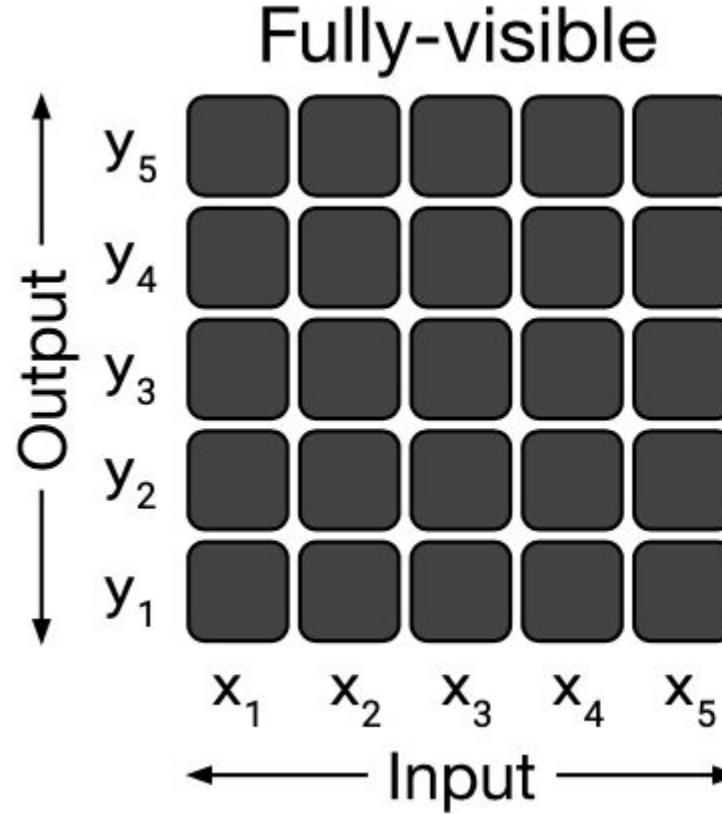


أكاديمية كاوهست  
KAUST ACADEMY



LMH

Lady Margaret Hall



# Casual Masking

Target: S1b EOT S2a S2b EOT      Target: S1b EOT S2b EOT  
Model input: S1a S1b EOT S2a S2b      Model input: S1a S1b S2a S2b

Possible attention masks:

	S1	S1	EOT	S2	S2
S1	1	0	0	0	0
S1	1	1	0	0	0
EOT	1	1	1	0	0
S2	1	1	1	1	0
S2	1	1	1	1	1

	S1	S1	S2	S2
S1	1	0	0	0
S1	1	1	0	0
S2	0	0	1	0
S2	0	0	1	1

Left: Causal mask when S1 and S2 can attend to each other

Right: Preventing attention from cross-attending between S1 and S2

# Pre-LayerNorm vs. Post-LayerNorm

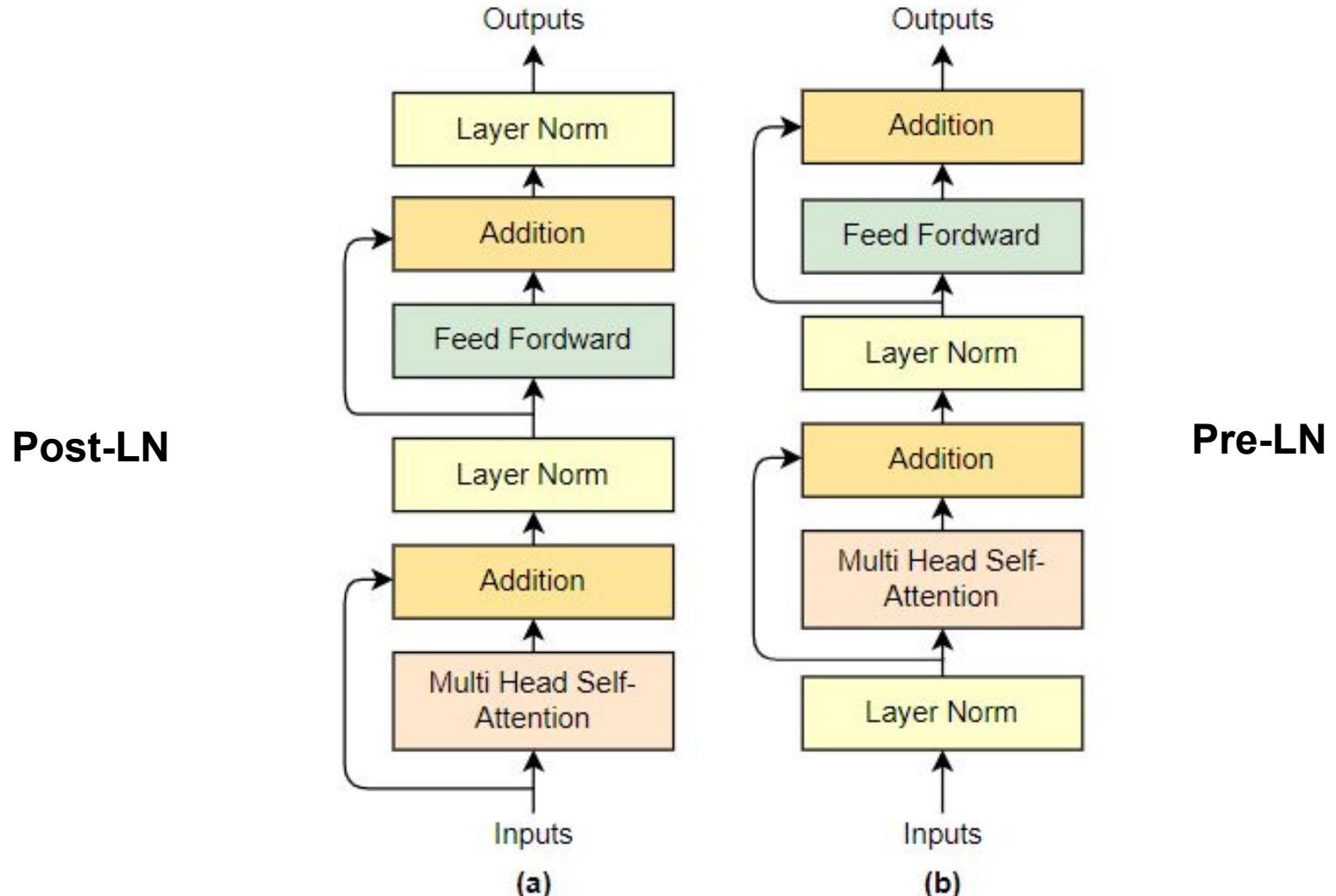
Feature	Pre-LN	Post-LN
Stability	✓ More stable	✗ Less stable
Gradient flow	✓ Better	✗ Can explode/vanish
Used in	GPT-3, T5	Original Transformer

Table 1: Comparison of Pre-LayerNorm and Post-LayerNorm Architectures

**Pre-LN:** LayerNorm applied *before* sublayers

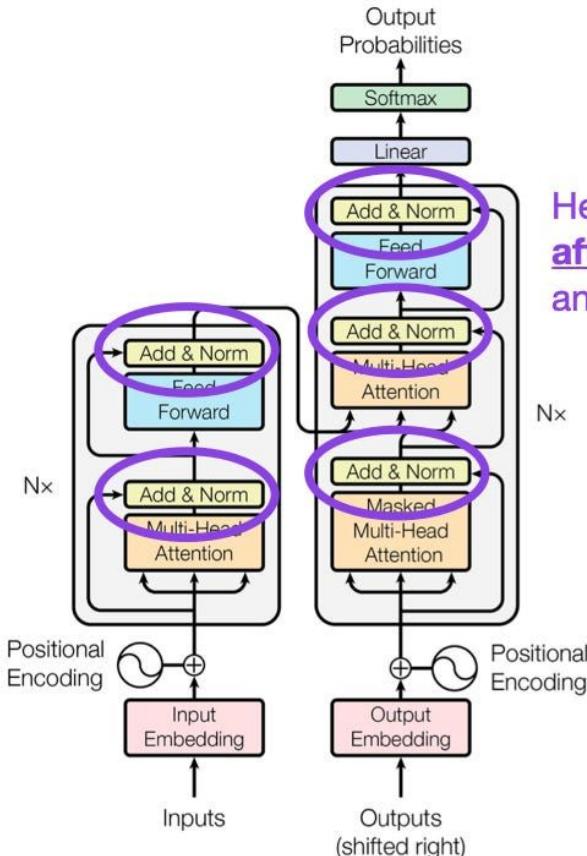
**Post-LN:** LayerNorm applied *after* sublayers

# Pre-LayerNorm vs. Post-LayerNorm



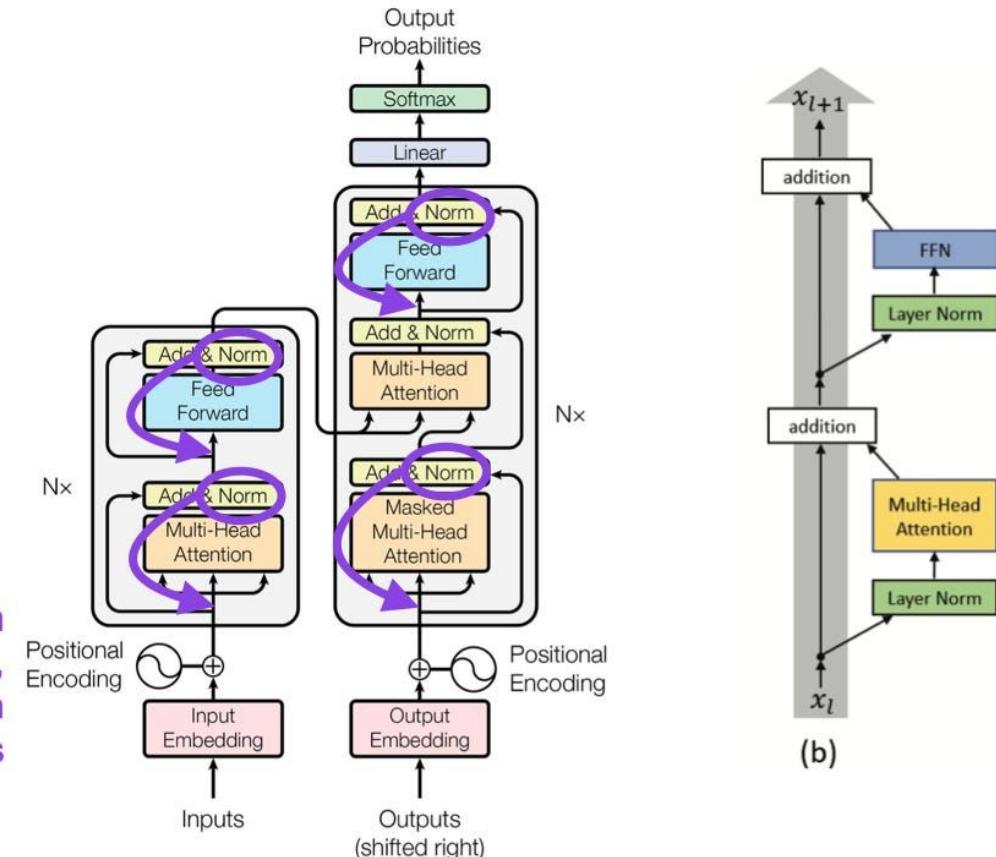
# Pre-LayerNorm vs. Post-LayerNorm

Post-LN Transformer



Here, LayerNorms are after attention and after fully connected layers

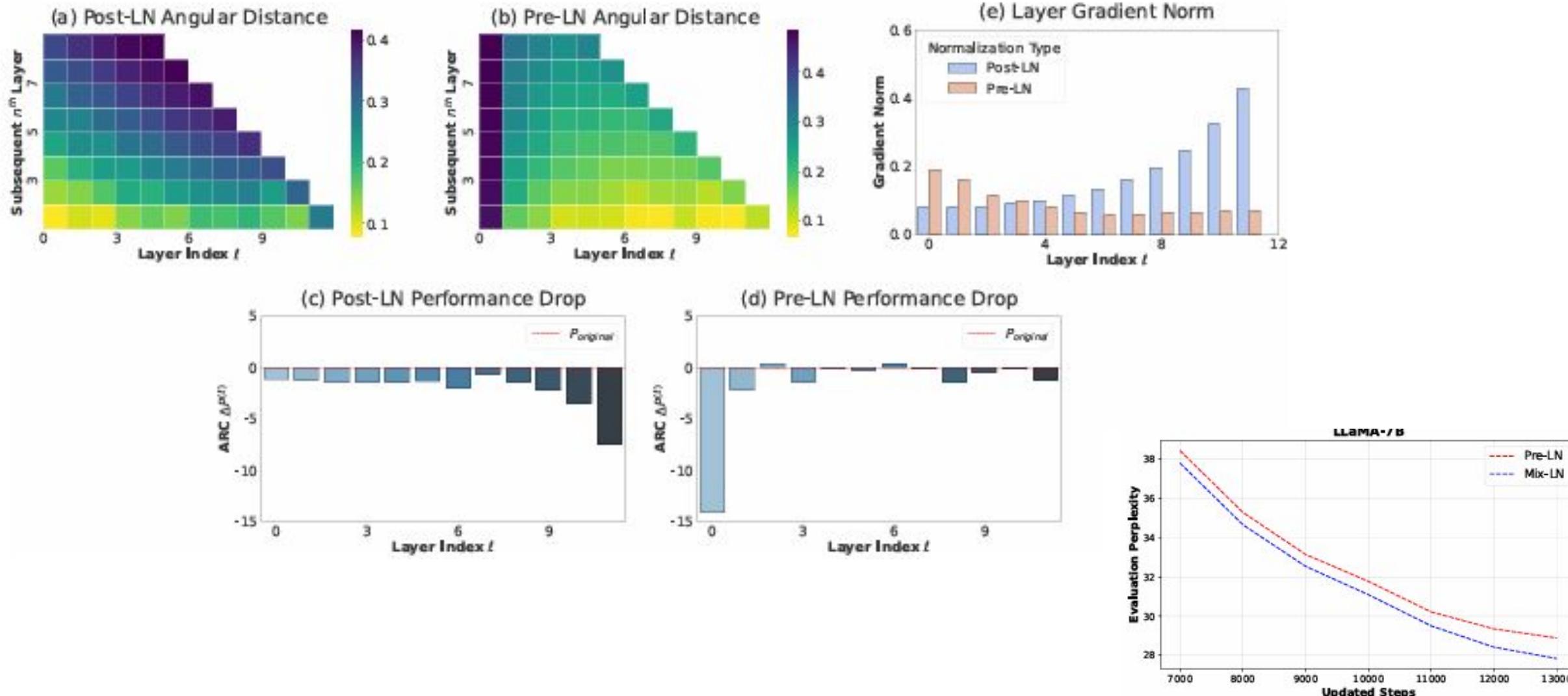
Pre-LN Transformer



Better gradients if LayerNorm is placed inside residual connection, before the attention and fully connected layers

Places the layer normalization between the residual blocks: the expected gradients of the parameters near the output layer are large

# Pre-LayerNorm vs. Post-LayerNorm



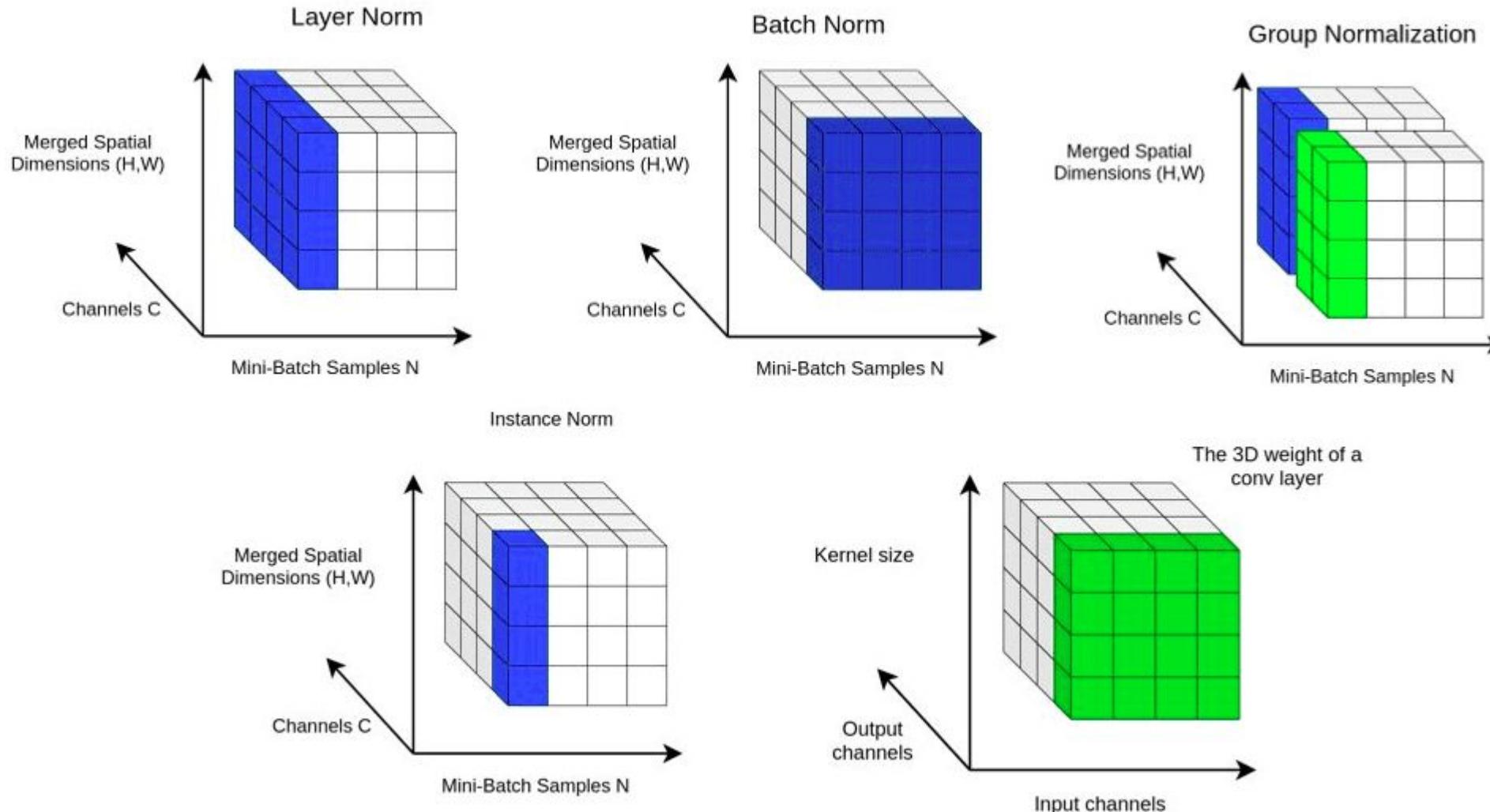
# Layer Normalization vs RMSNorm

**LayerNorm:** Normalizes across the feature dimension by subtracting the mean and dividing by the standard deviation.

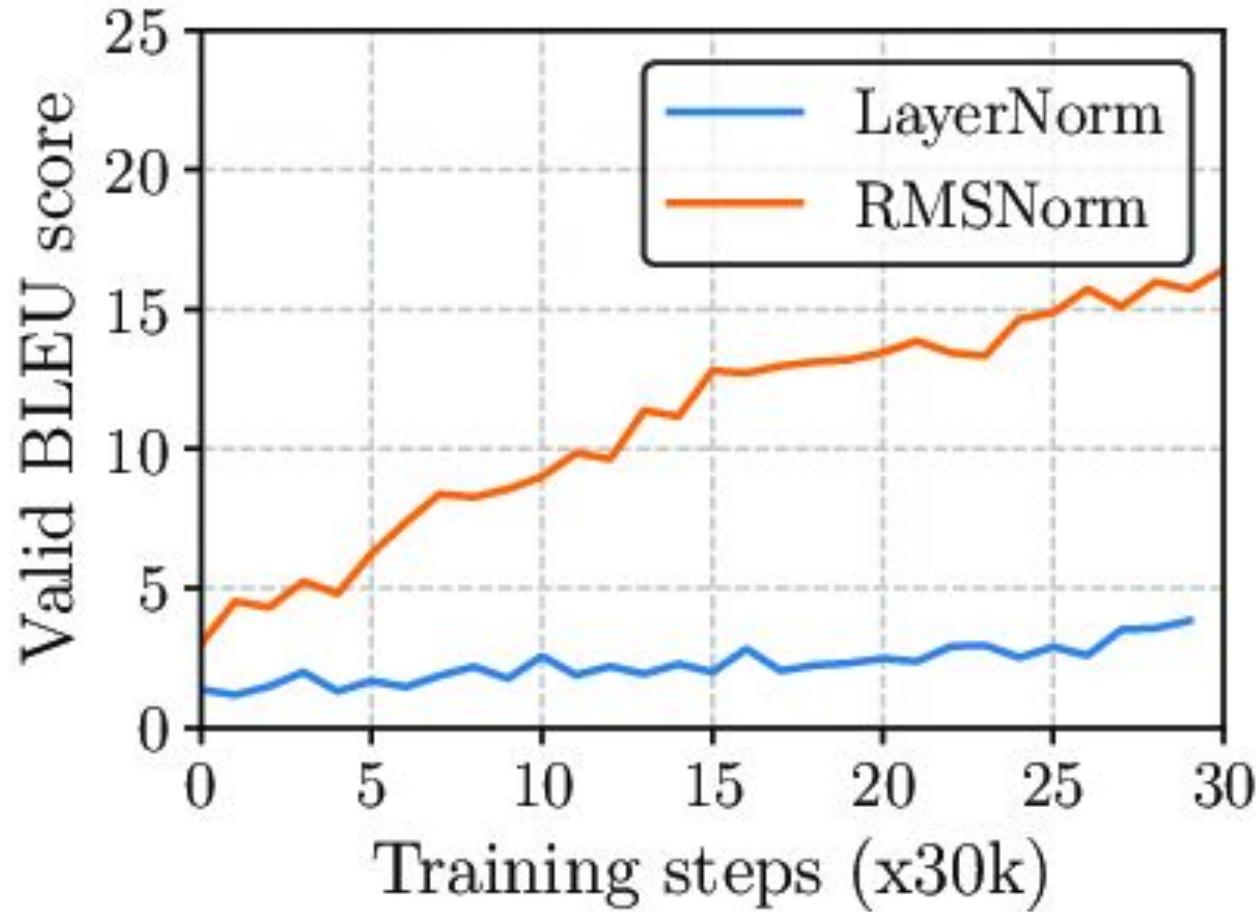
**RMSNorm:** Removes the mean and scales by the root mean square (RMS) of the features.

- ▶ Simpler
- ▶ Faster
- ▶ Slightly better in large-scale settings **zhou2022rmsnorm**

# Layer Normalization vs RMSNorm



# Layer Normalization vs RMSNorm



- ▶ **Encoder-only:** BERT, RoBERTa
- ▶ **Decoder-only:** GPT, LLaMA, Falcon
- ▶ **Encoder-Decoder:** T5, BART, mT5

## Modifications:

- ▶ **Long-range:** Longformer, BigBird
- ▶ **Efficient:** Linformer, Performer
- ▶ **Sparse:** Sparse Transformer, Reformer

# BERT - Bidirectional Encoder

## BERT Pre-Training Corpus:

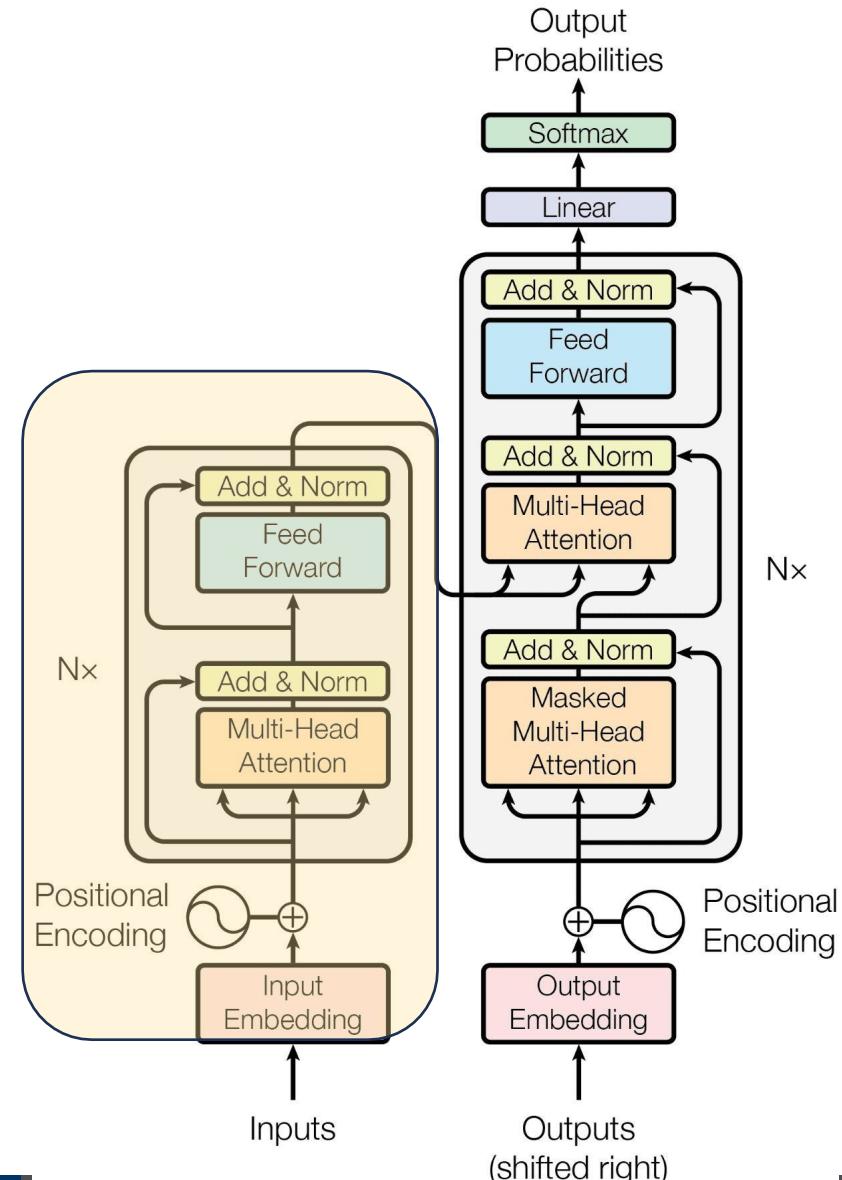
- English Wikipedia - 2,500 million words
- Book Corpus - 800 million words

## BERT Pre-Training Tasks:

- MLM (Masked Language Modeling)
- NSP (Next Sentence Prediction)

## BERT Pre-Training Results:

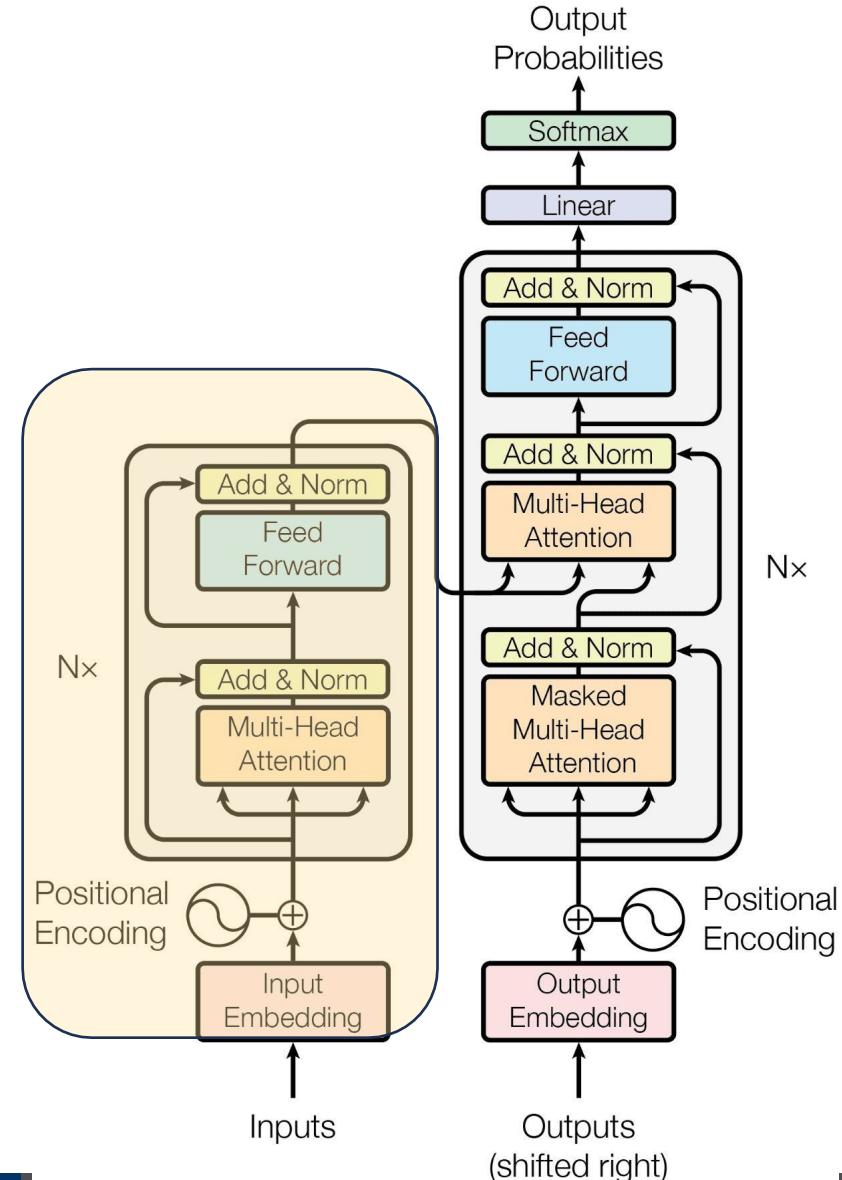
- BERT-Base – 110M Params
- BERT-Large – 340M Params



# BERT - Bidirectional Encoder

## What is our takeaway from BERT?

- Pre-training tasks can be invented flexibly...
  - Effective representations can be derived from a flexible regime of pre-training tasks.
- Different NLP tasks seem to be highly transferable with each other...
  - As long as we have effective representations, that seems to form a general model which can serve as the backbone for many specialized models.
- And scaling works!!!
  - 340M was considered large in 2018



# GPT – Generative Pretrained Transformer

## GPT Pre-Training Corpus:

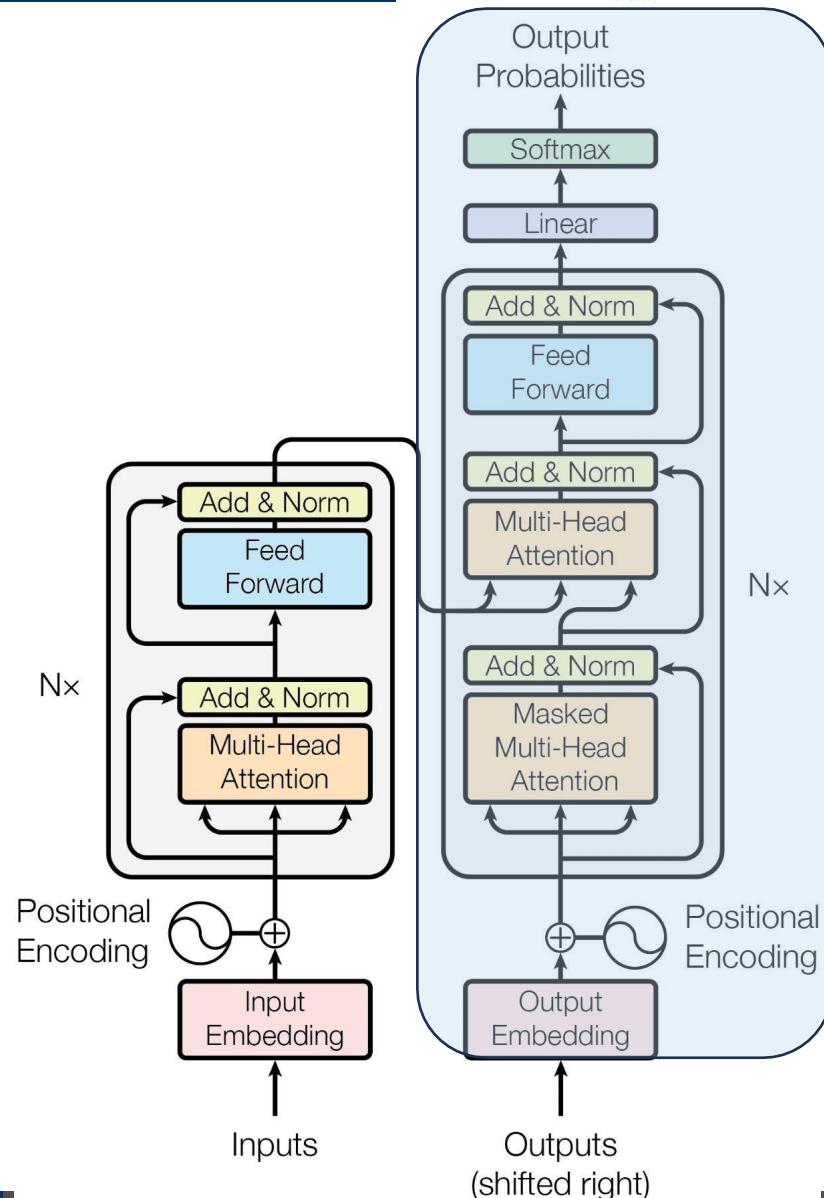
- Similarly, BooksCorpus and English Wikipedia

## GPT Pre-Training Tasks:

- Predict the next token, given the previous tokens
  - More learning signals than MLM

## GPT Pre-Training Results:

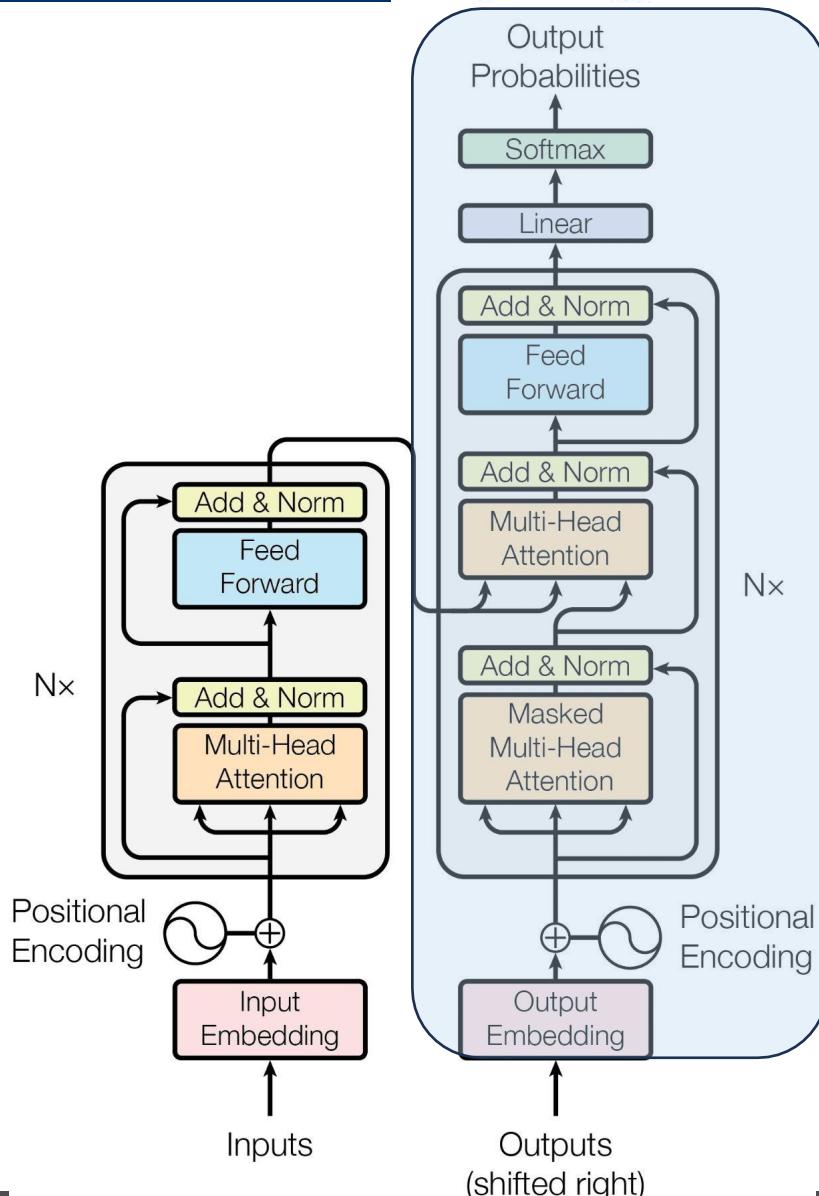
- GPT – 117M Params
  - Similarly competitive on GLUE and SQuAD



# GPT - Generativ Pretrained Transformer

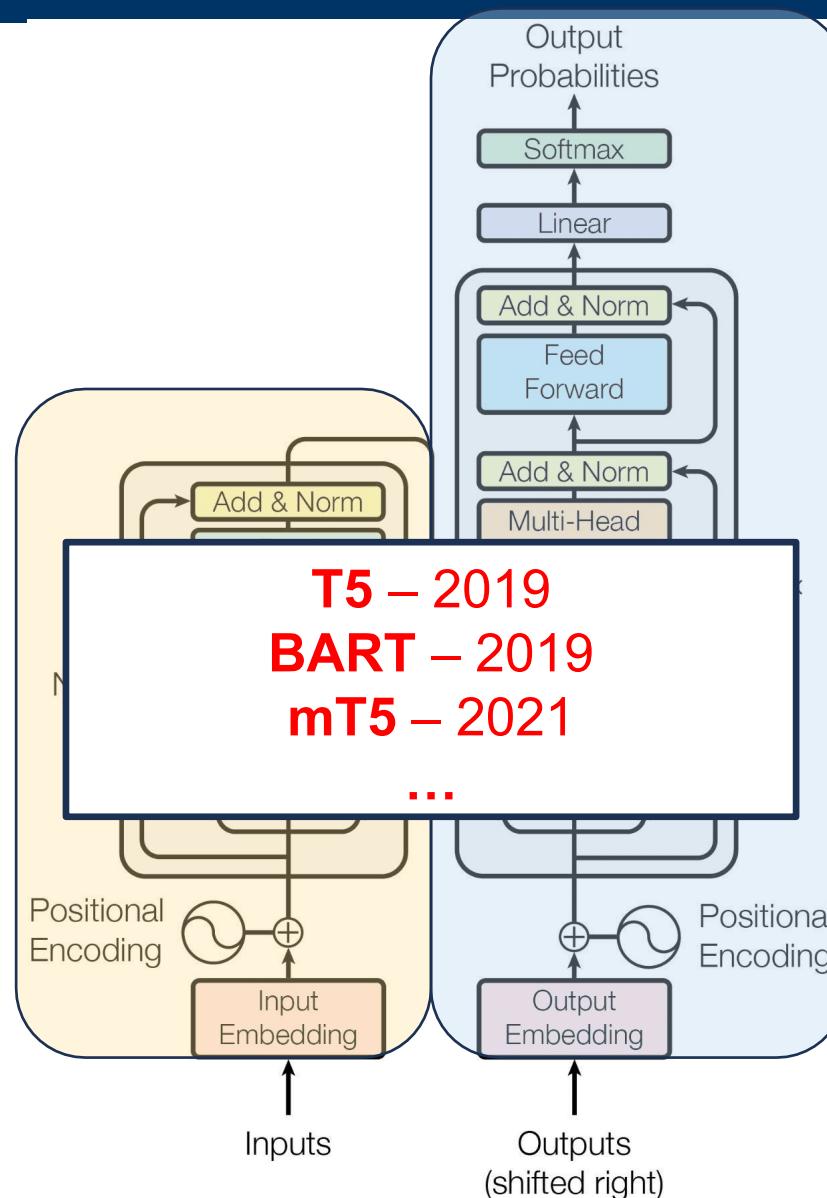
What is our takeaway from GPT?

- **The Effectiveness of Self-Supervised Learning**
  - Specifically, the model seems to be able to learn from generating the language *itself*, rather than from any specific task we might cook up.
- **Language Model as a Knowledge Base**
  - Specifically, a generatively pretrained model seems to have a decent zero-shot performance on a range of NLP tasks.
- **And scaling works!!!**



# The LLM Era – Paradigm Shift in Machine Learning

**BERT** – 2018  
**DistilBERT** – 2019  
**RoBERTa** – 2019  
**ALBERT** – 2019  
**ELECTRA** – 2020  
**DeBERTa** – 2020  
 ...  
**Representation**



**GPT** – 2018  
**GPT-2** – 2019  
**GPT-3** – 2020  
**GPT-Neo** – 2021  
**GPT-3.5 (ChatGPT)** – 2022  
**LLaMA** – 2023  
**GPT-4** – 2023  
 ...

**Generation**

# Limitations of Transformers

- ▶ High compute and memory requirements
- ▶ Poor extrapolation capabilities
- ▶ Limitations due to hard-coded positional encoding
- ▶ Quadratic attention cost ( $O(n^2)$ )
- ▶ Lack of reasoning and grounding

- ▶ Linear and sparse attention mechanisms
- ▶ Memory-efficient transformers (e.g., FlashAttention)
- ▶ Integration with retrieval, reasoning, and tool use
- ▶ Multimodal transformers (e.g., Flamingo, GPT-4V)
- ▶ Biologically inspired architectures (e.g., RWKV, State Space Models)

# Summary

- ▶ Transformers revolutionized deep learning by removing recurrence and leveraging attention.
- ▶ Self-attention allows contextual understanding.
- ▶ Multi-head and positional encodings add flexibility.
- ▶ Variants serve different tasks (BERT, GPT, T5).
- ▶ Ongoing research improves efficiency, grounding, and capabilities.

These slides have been adapted from

- Bhiksha Raj & Rita Singh, 11-785 Introduction to Deep Learning, CMU

# References

- [1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. In *NeurIPS*.
- [2] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers. In *NAACL*.
- [3] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners (GPT-2). OpenAI.
- [4] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5). *JMLR*.
- [5] Zhou, H., Zhang, S., Ma, S., & Sun, J. (2022). RMSNorm: Root Mean Square Layer Normalization. arXiv preprint arXiv:1910.07467.

- [6] Dao, T., Fu, T., Ermon, S., Rudra, A., & Ré, C. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *NeurIPS*.
- [7] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. arXiv preprint arXiv:2004.05150.
- [8] Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2020). Efficient Transformers: A Survey. arXiv preprint arXiv:2009.06732.

## Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

[p.aparajeya@aisimply.uk](mailto:p.aparajeya@aisimply.uk)

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.