Introduction to Transformers

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

LMH
Lady Margaret Hall

July 3, 2025

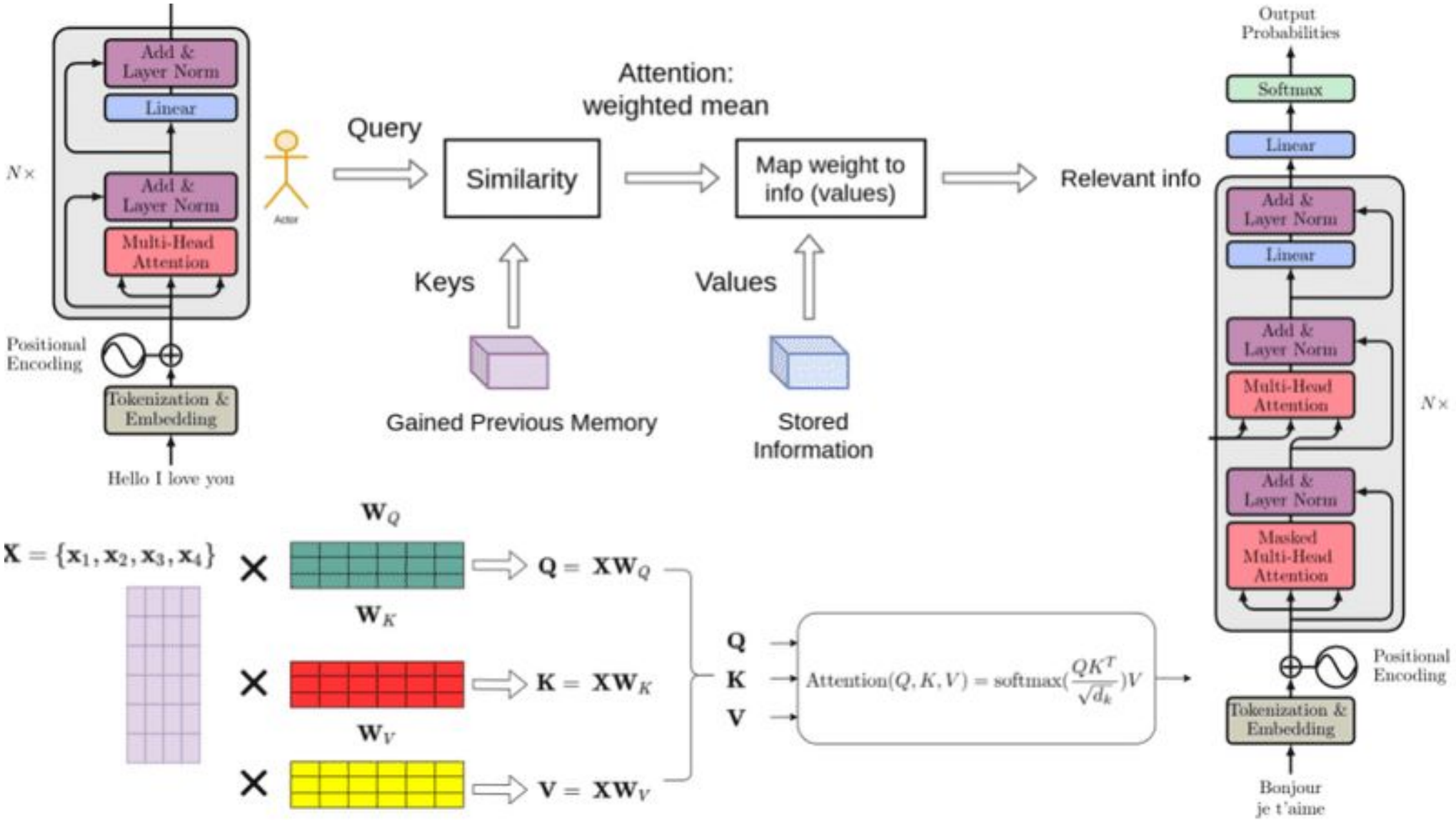

Add & Layer Norm
Linear
Add & Layer Norm
Multi-Head Attention
N×
Positional Encoding
Tokenization & Embedding
Hello I love you
Actor
Query
Attention: weighted mean
Similarity
Map weight to info (values)
Relevant info
Keys
Values
Gained Previous Memory
Stored Information
Output Probabilities
Softmax
Linear
Add & Layer Norm
Linear
Add & Layer Norm
Multi-Head Attention
Add & Layer Norm
Masked Multi-Head Attention
N×
Positional Encoding
Tokenization & Embedding
Bonjour je t'aime
$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$
$\mathbf{W}_Q$
$\mathbf{W}_K$
$\mathbf{W}_V$
$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$
$\mathbf{K} = \mathbf{X}\mathbf{W}_K$
$\mathbf{V} = \mathbf{X}\mathbf{W}_V$
$\mathbf{Q}$
$\mathbf{K}$
$\mathbf{V}$
$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

Table of Contents

Motivation

- **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

Learning Outcomes

After this session, you should be able to:

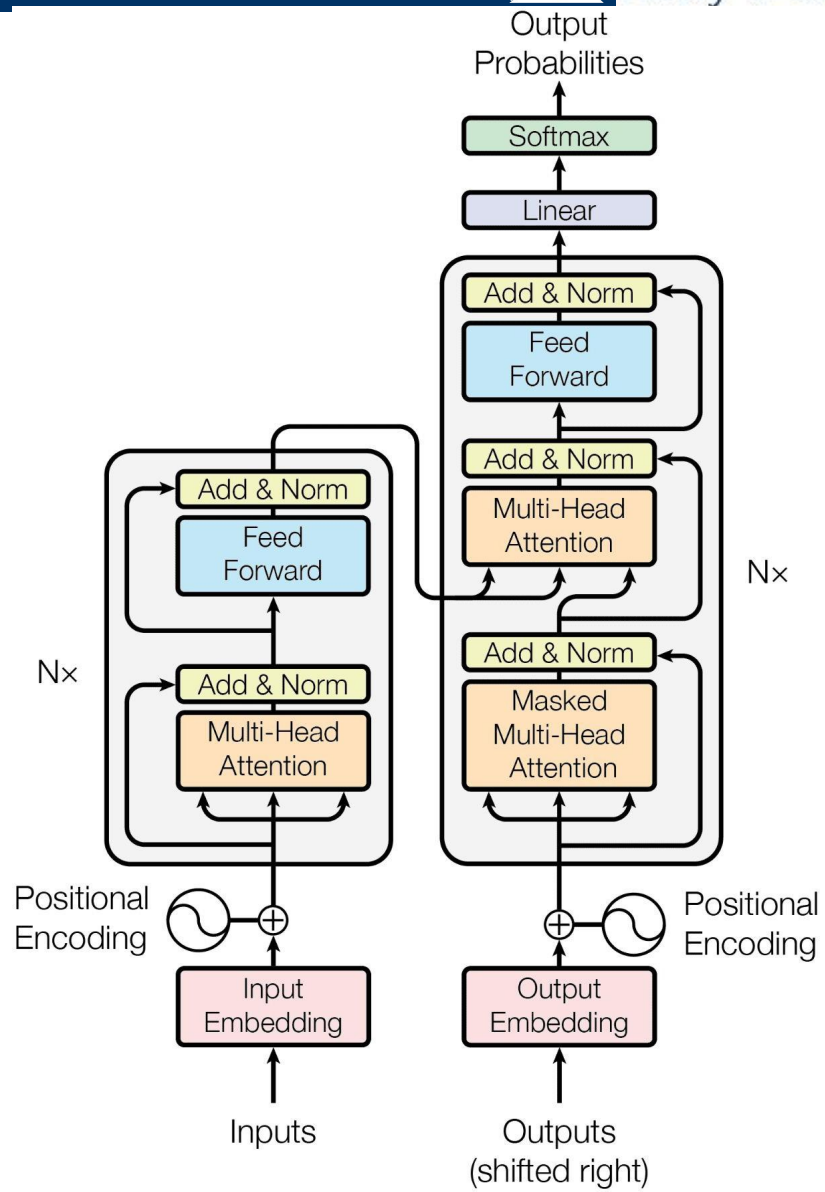
- Understand the self-attention mechanism
- Learn how multi-head attention works and why it helps
- Grasp the role of positional encoding
- Deconstruct transformer blocks and variants
- Compare Pre-LayerNorm vs. Post-LayerNorm
- Evaluate LayerNorm vs. RMSNorm
- Explore causal masking in decoder-only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

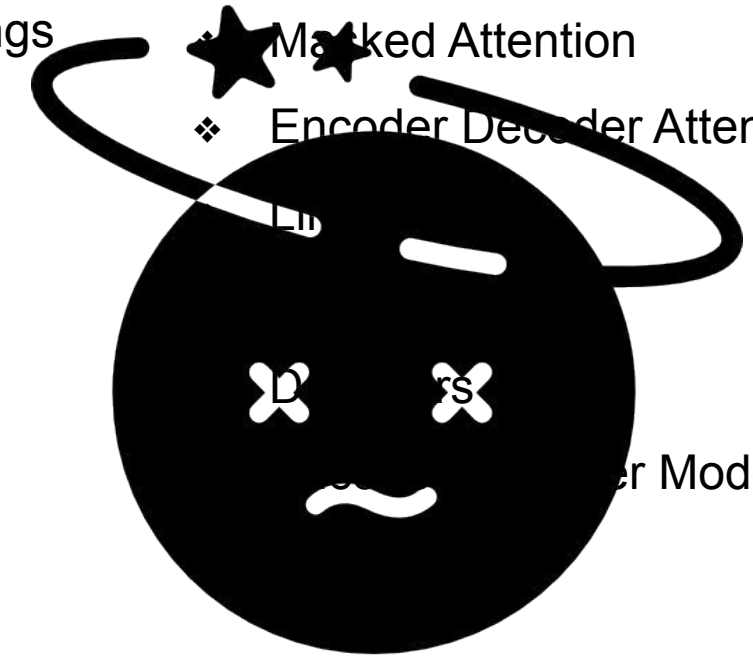
Transformers

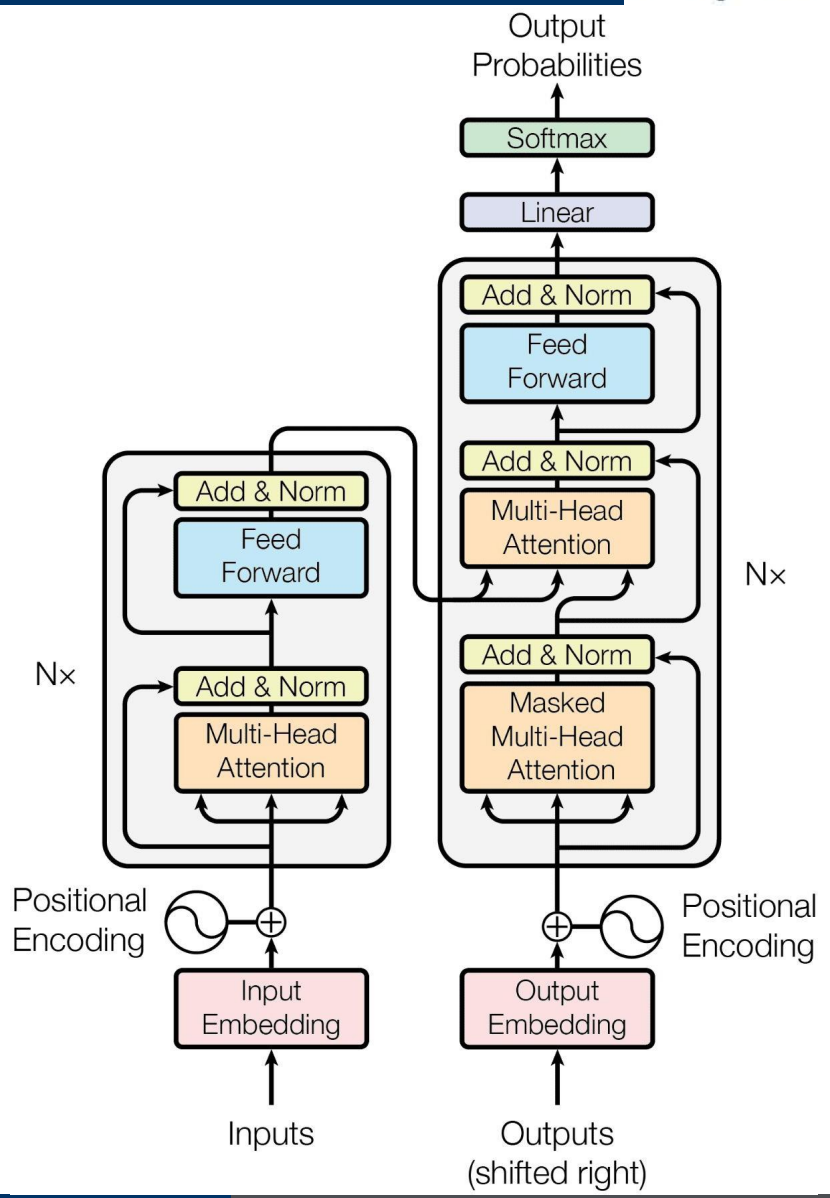
- Tokenization
- Input Embeddings
- Position Encodings
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders
- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models



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
- Li
- D rs
- er Models





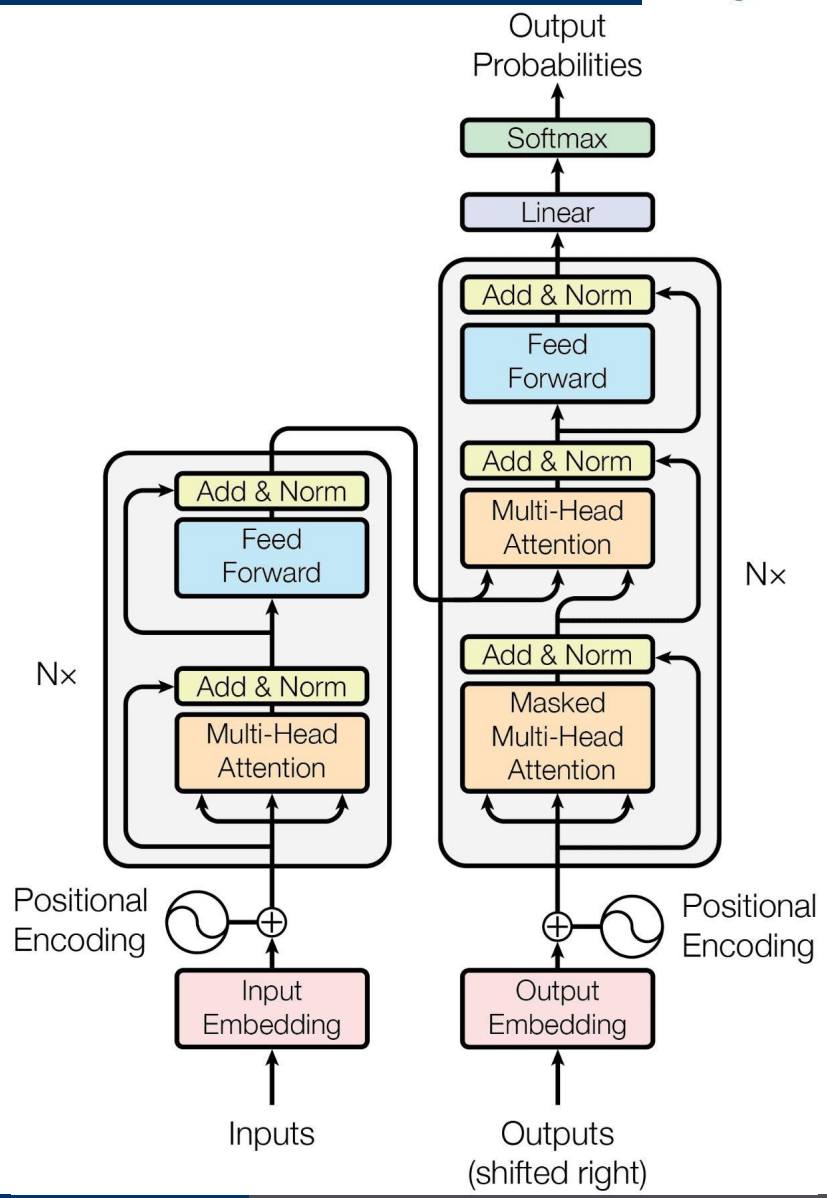
Machine Translation

Targets

Ich habe einen
Apfel gegessen

Inputs

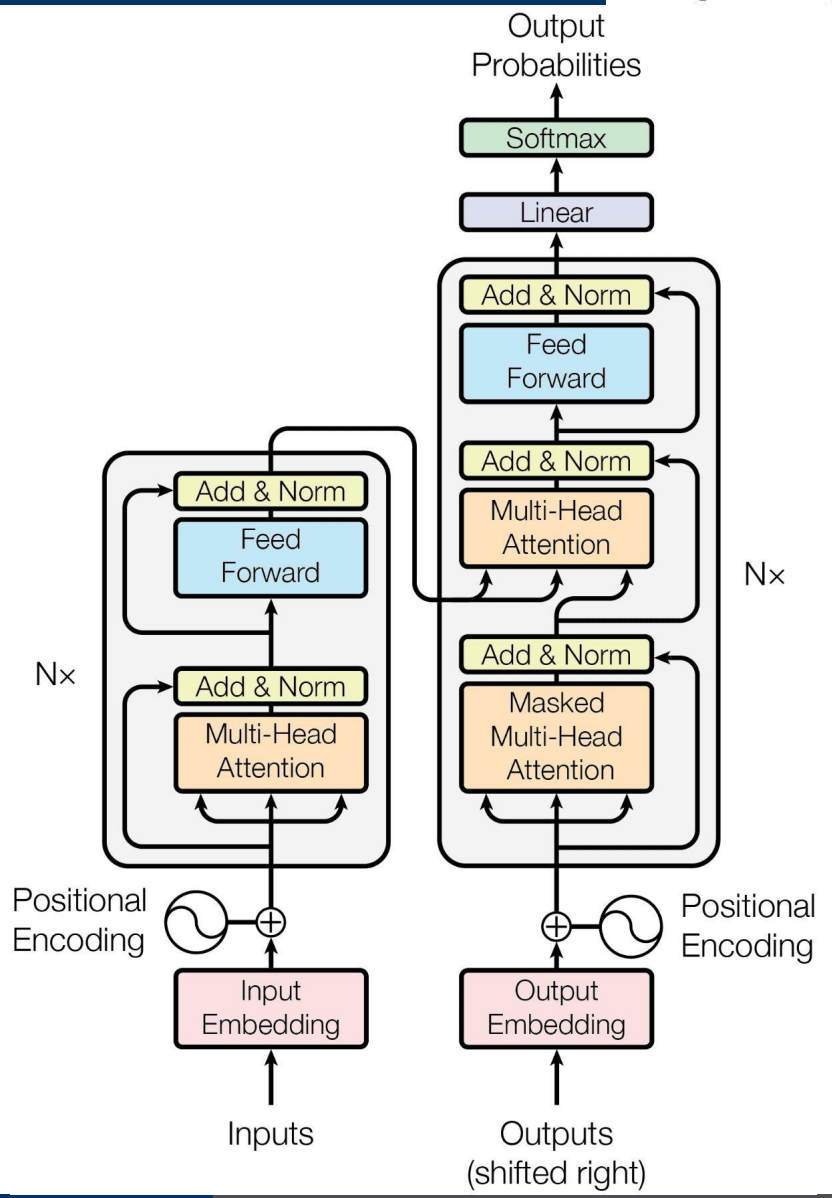
I ate an apple



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

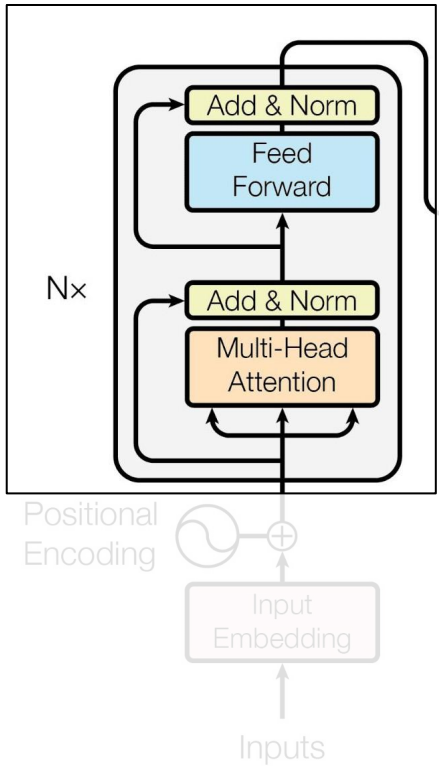
أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

- **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

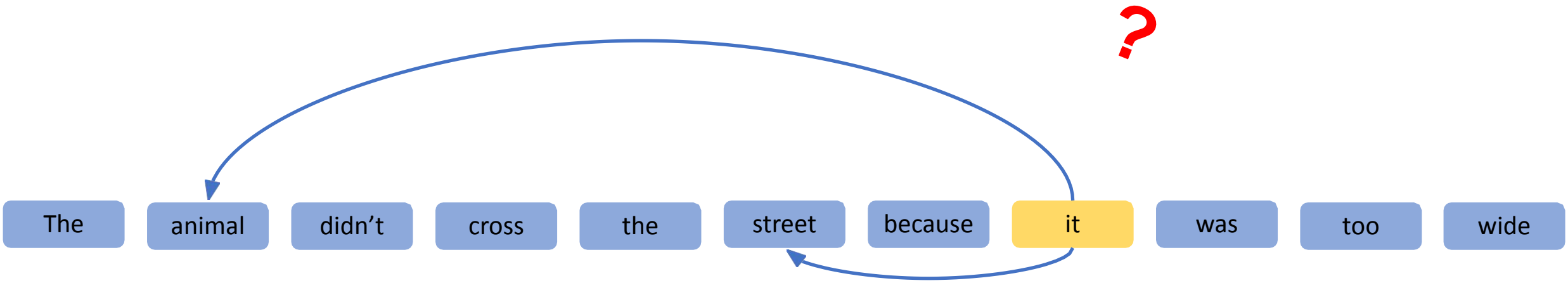
Self Attention

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



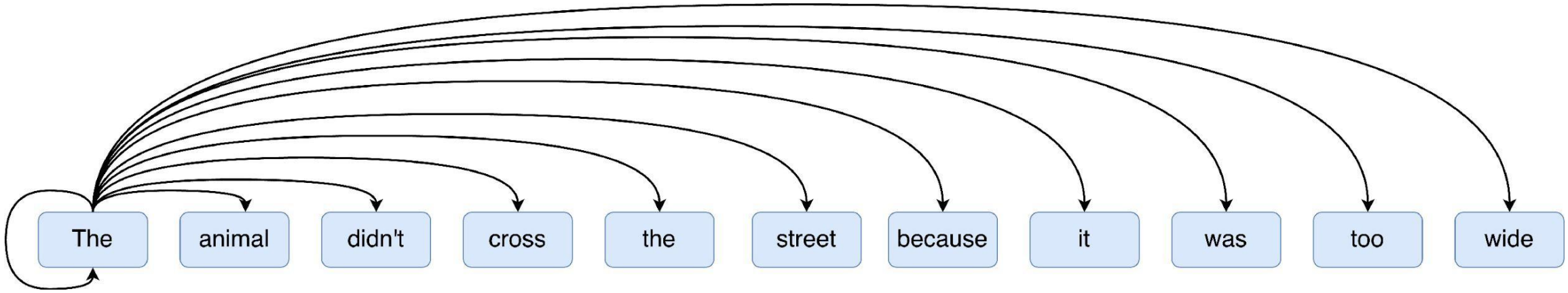
Self Attention

Self Attention

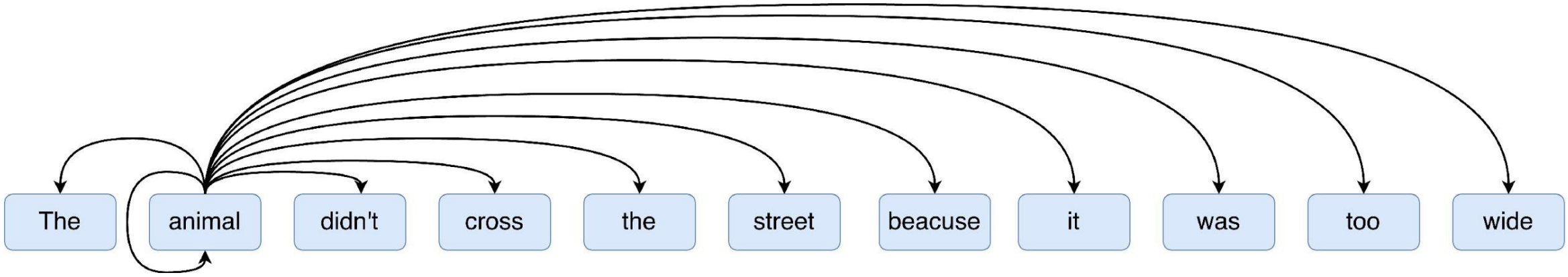


coreference resolution?

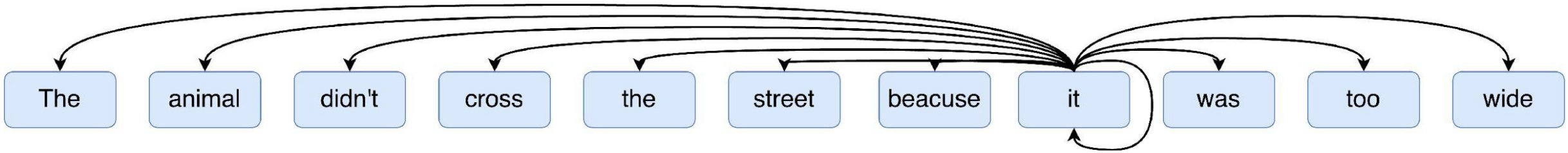
Self Attention



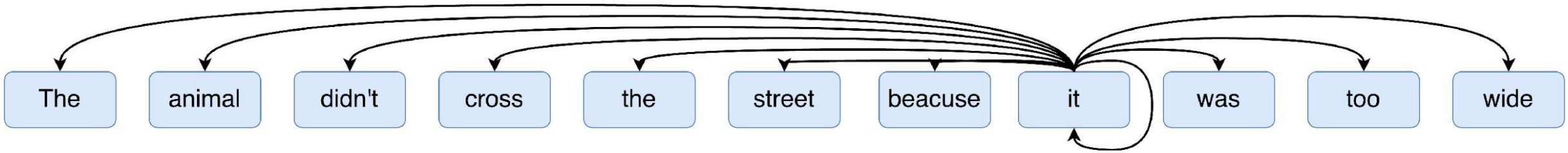
Self Attention



Self Attention



Self Attention

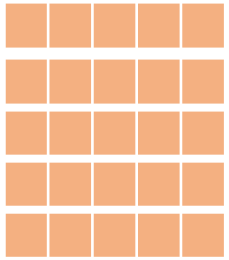


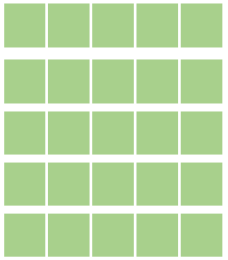
SELF

Query Inputs = Key Inputs = Value Inputs

Self Attention

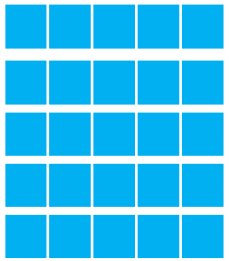
$R^{d_{model} \times d_{model}}$



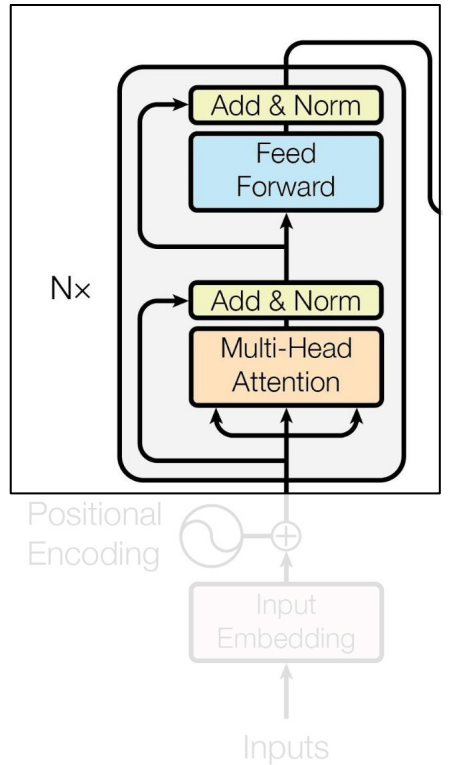


W_Q W_K W_v

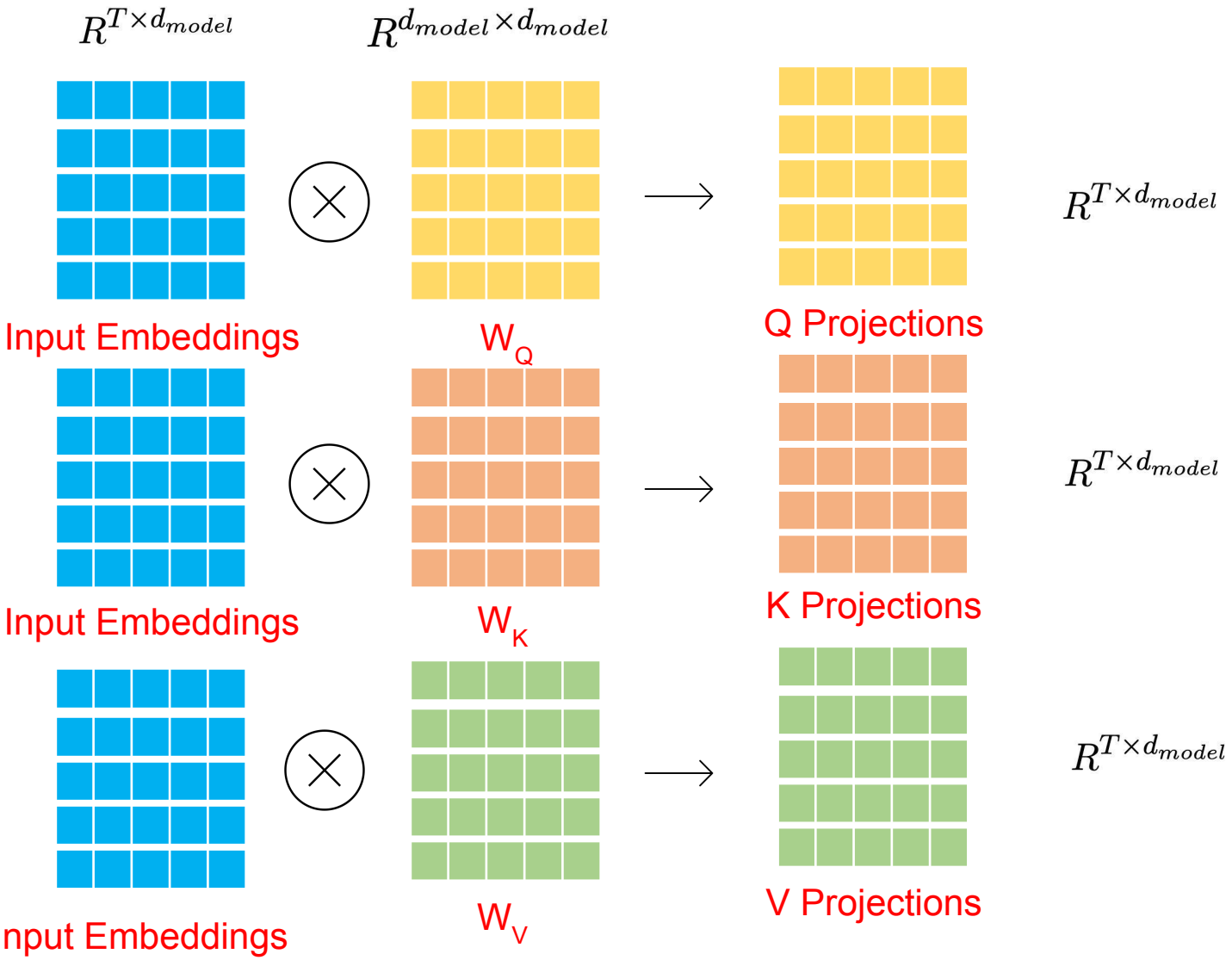
$R^{T \times d_{model}}$

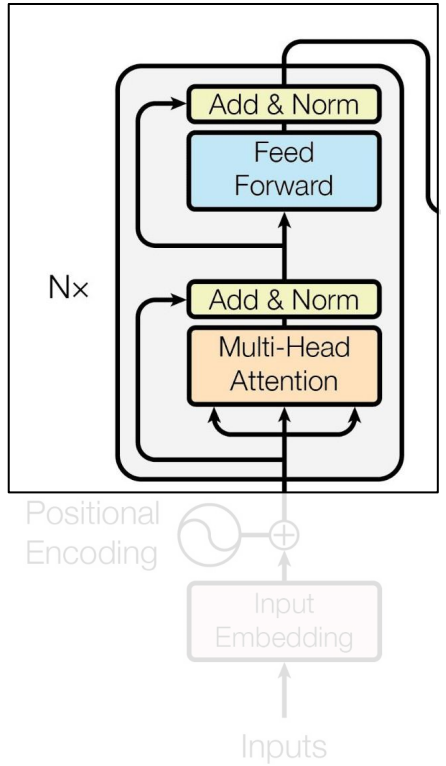


Input Embeddings



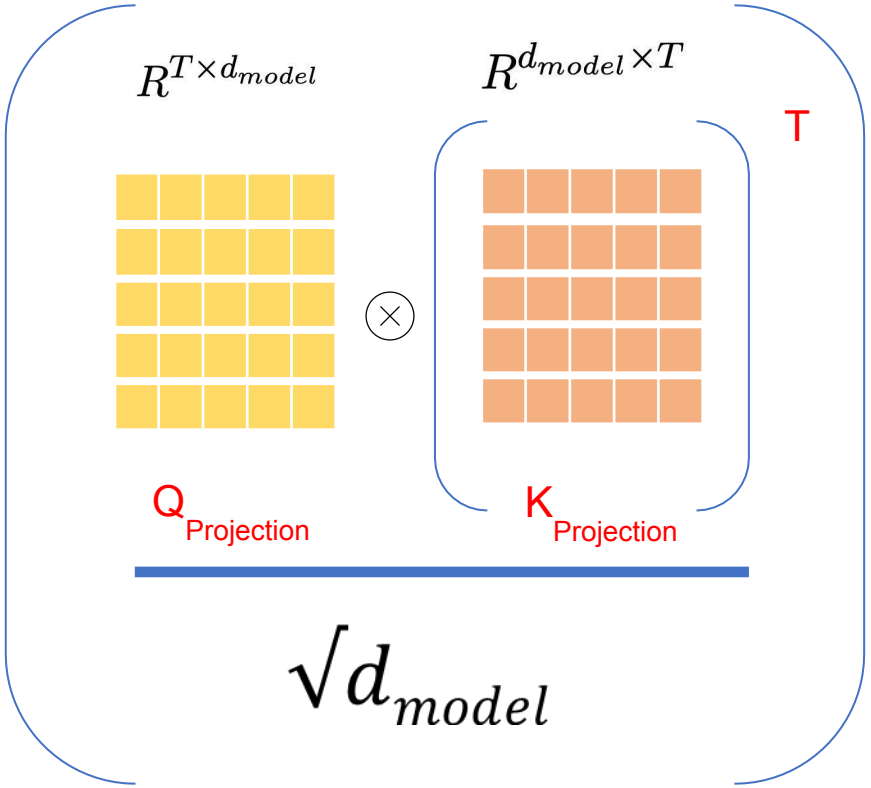
Self Attention



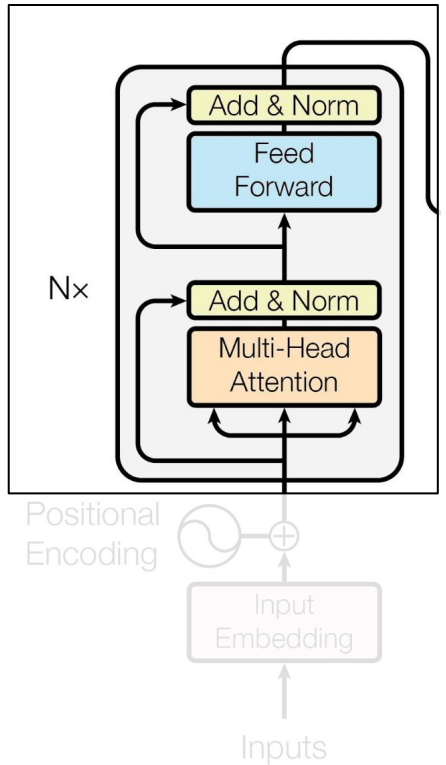


Self Attention

softmax



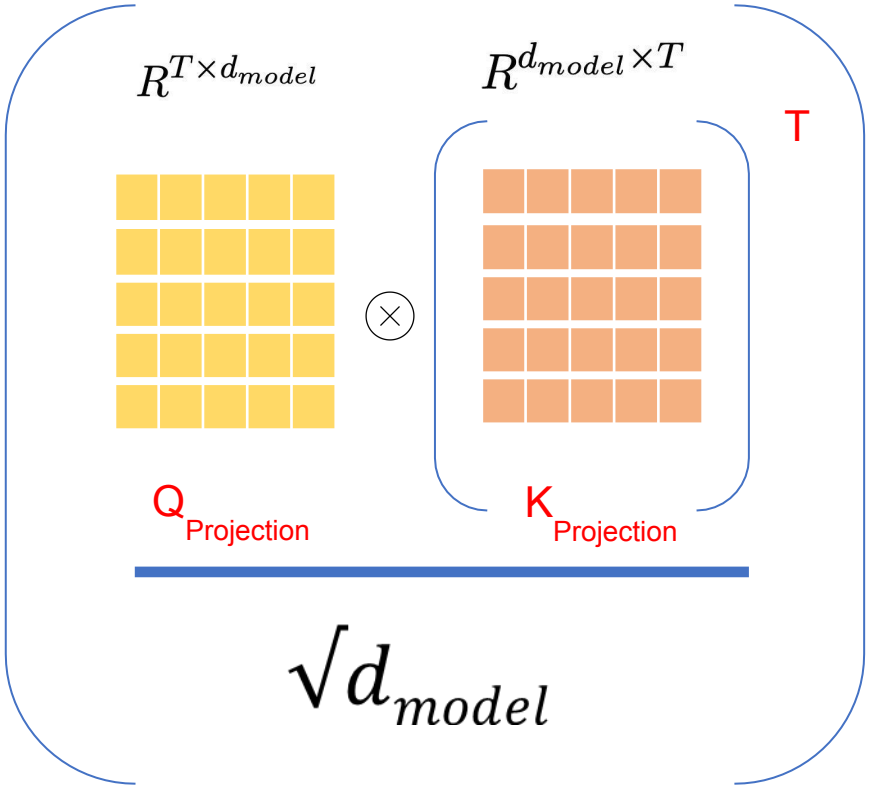
$R^{T \times T}$



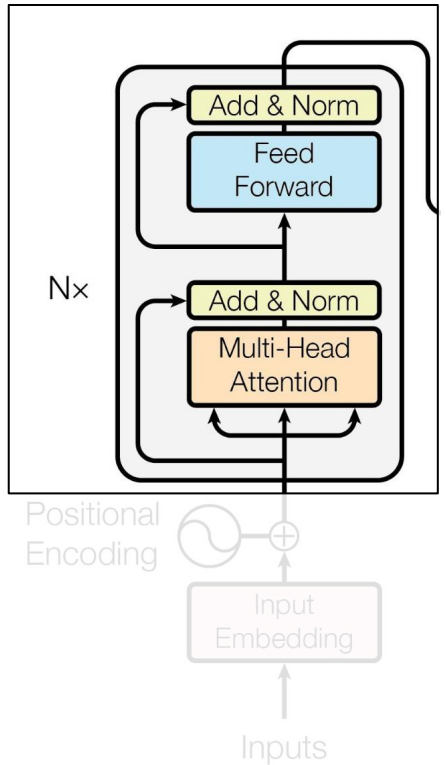
Self Attention

$O(T^2 \times d_{model})$

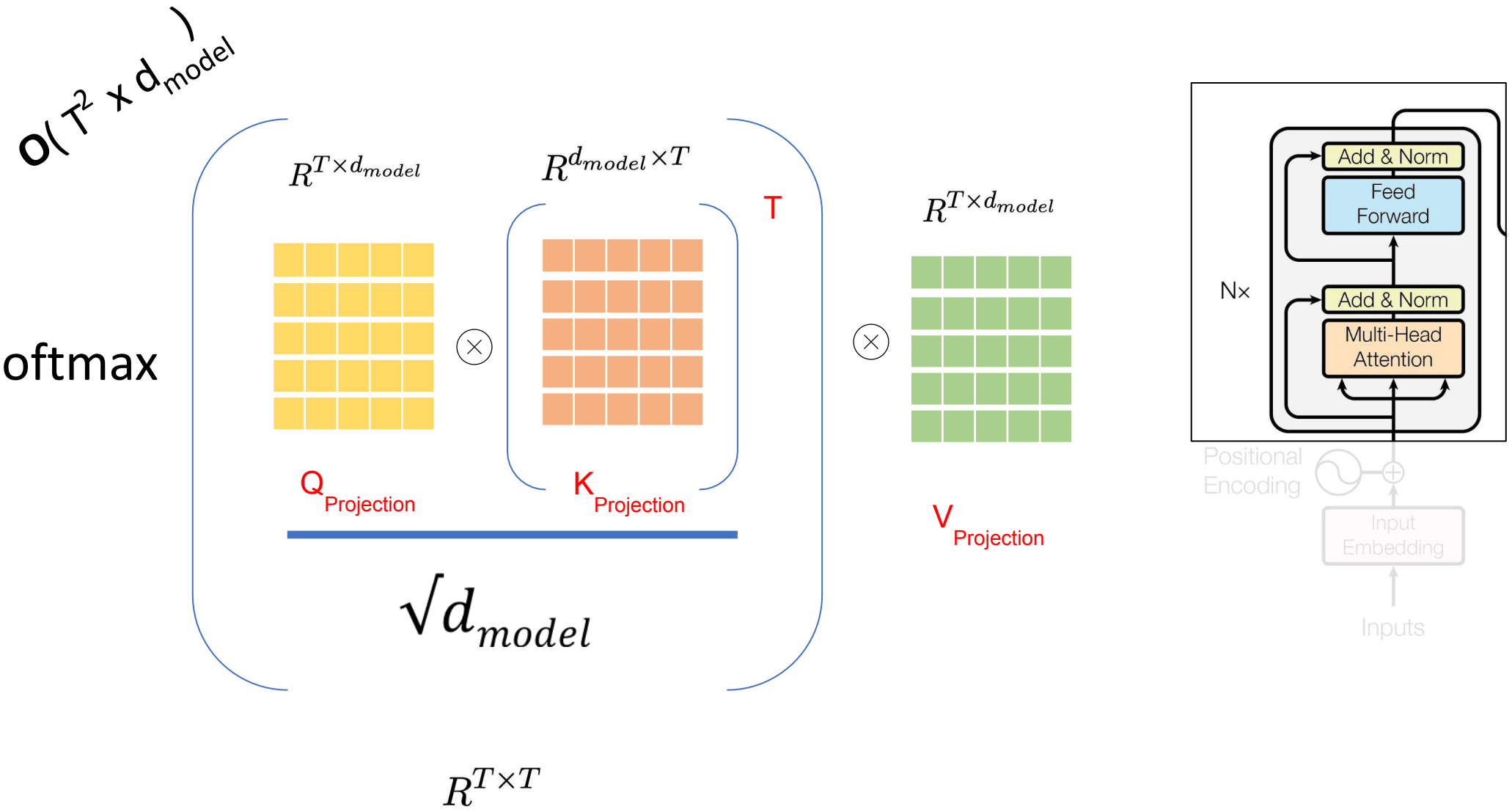
softmax



$R^{T \times T}$



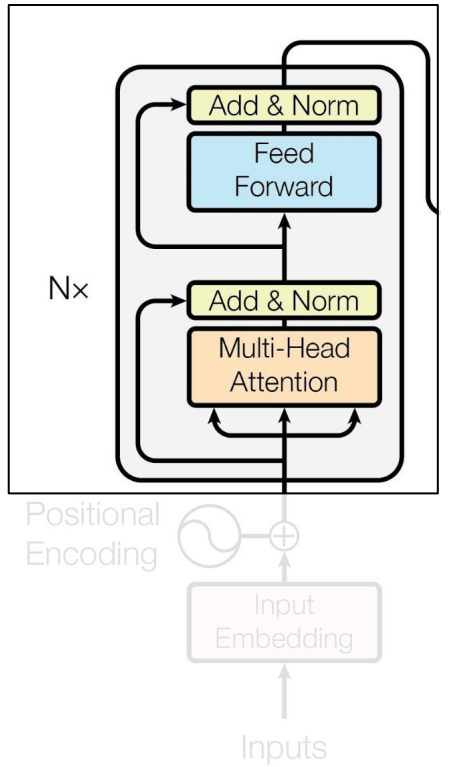
Self Attention



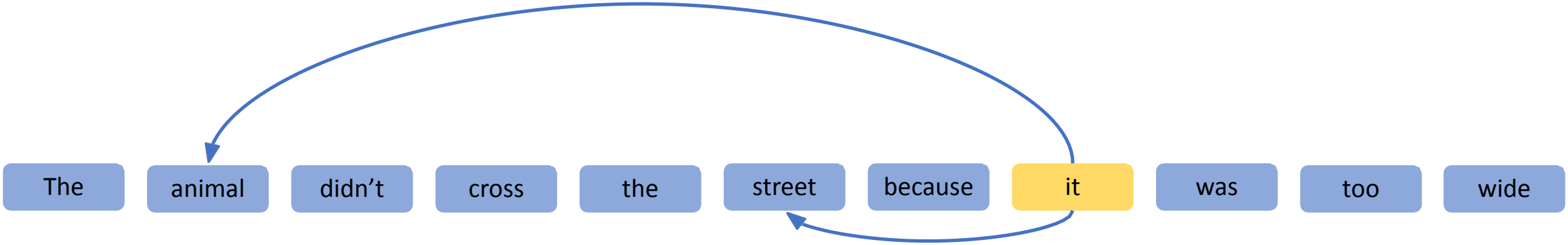
Self Attention

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

Attention: Z

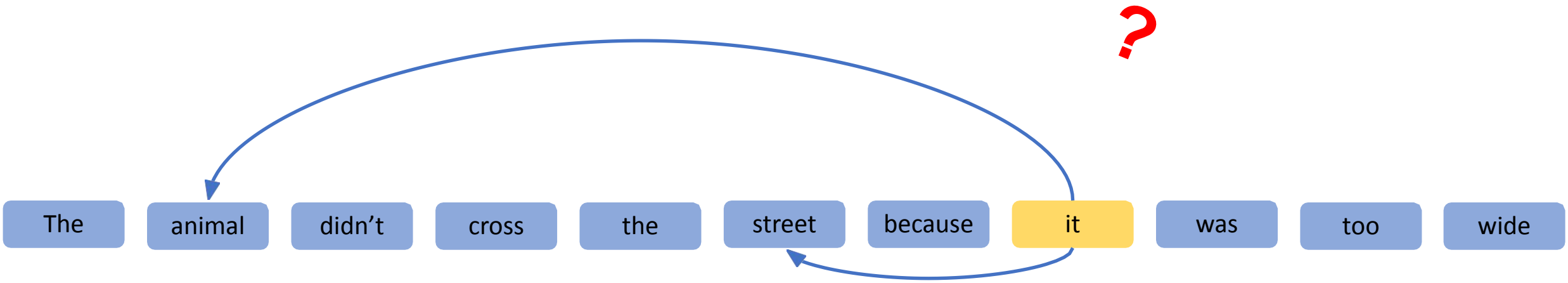


Self Attention



Coreference resolution ✓

Self Attention



Sentence boundaries ?

Coreference resolution ✓

Context ?

Semantic relationships ?

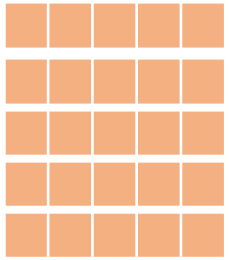
Part of Speech ?

Comparisons ?

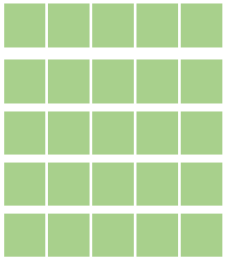
Self Attention

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

W_Q

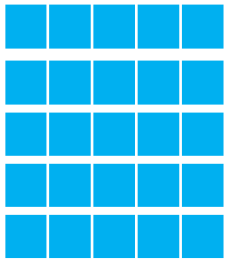


W_K

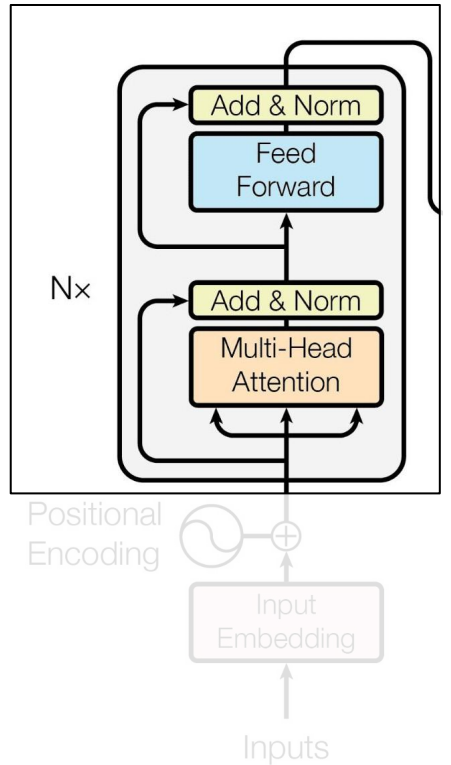


W_v

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



Input Embeddings



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:

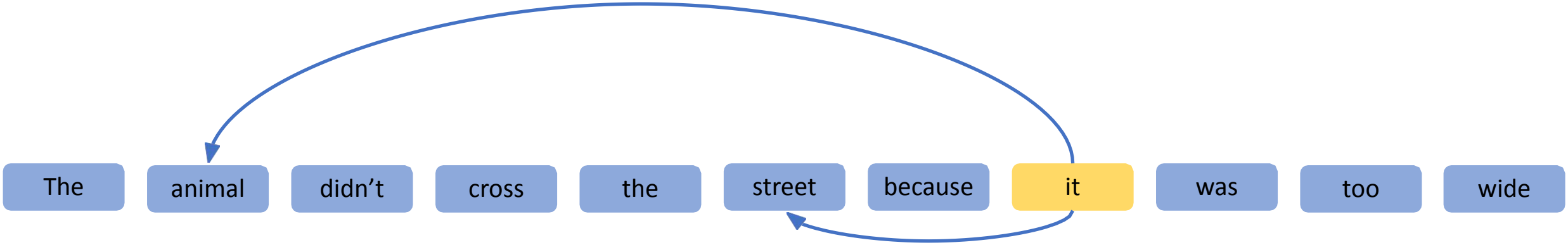
$$\text{MultiHead}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_h\right) W^O$$

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.

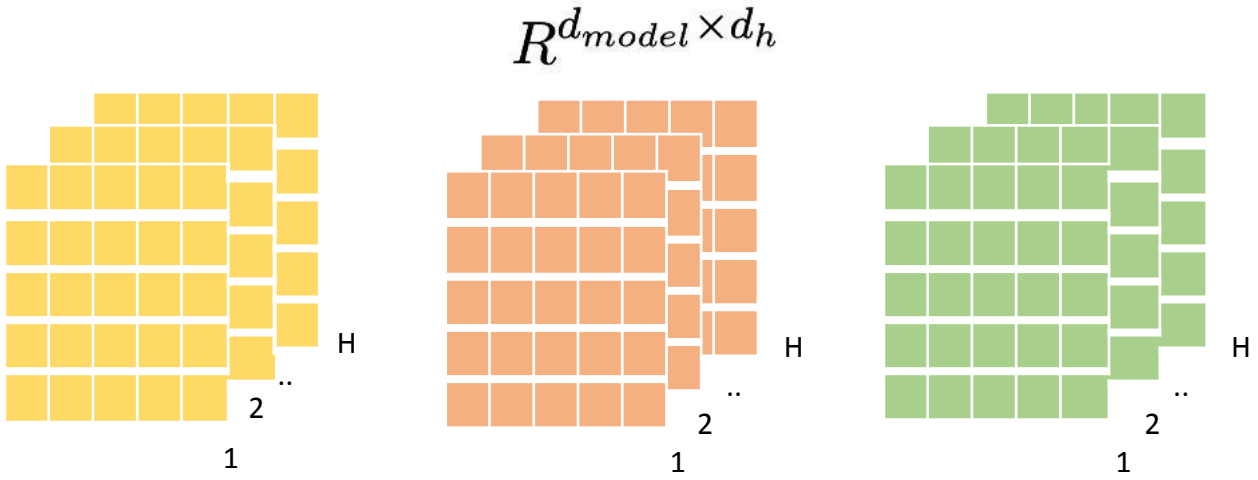
Multi-Head Attention



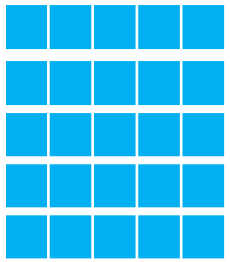
Sentence boundaries ✓ Coreference resolution ✓ Context ✓

Semantic relationships ✓ Part of speech ✓ Comparisons ✓

Multi-Head Attention



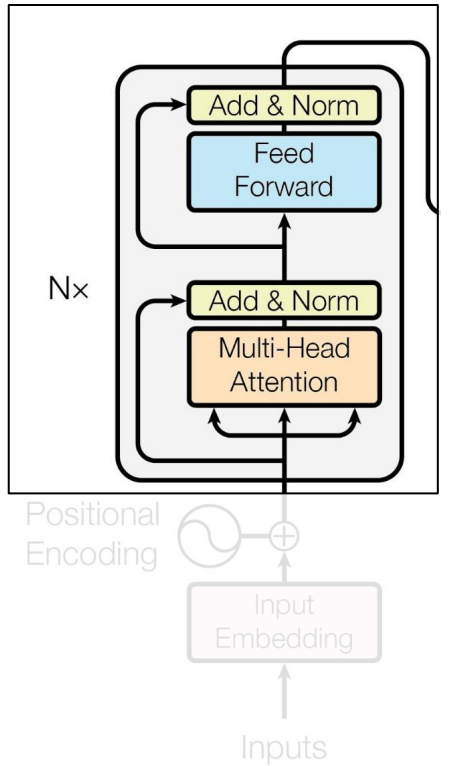
$W_{Q1,}$ $W_{Q2,\ \ldots}$ $W_{QH,}$ $W_{K1,}$ $W_{K2,\ \ldots}$ $W_{KH,}$ $W_{V1,}$ $W_{V2,\ \ldots}$ $W_{VH,}$



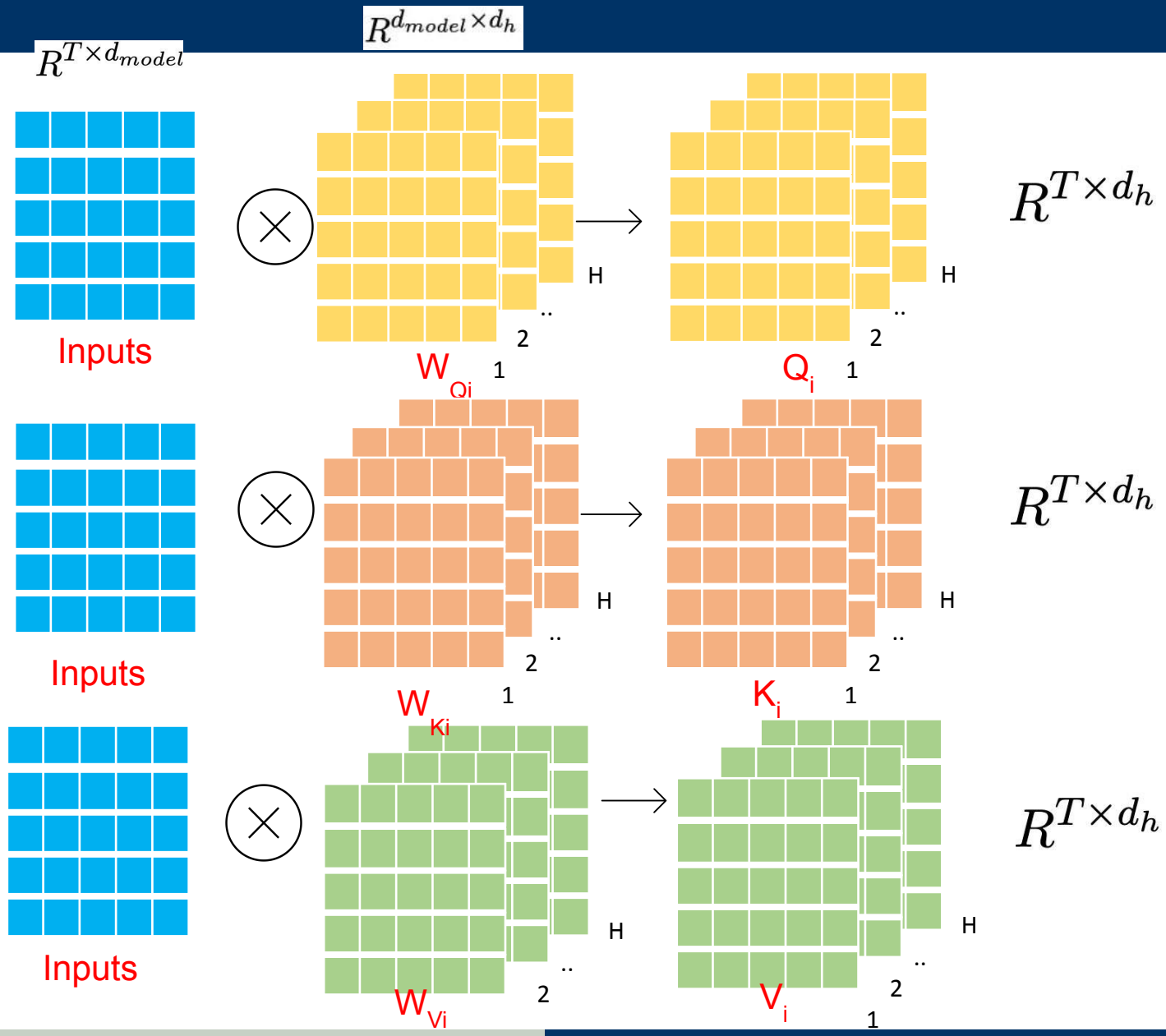
Input Embeddings

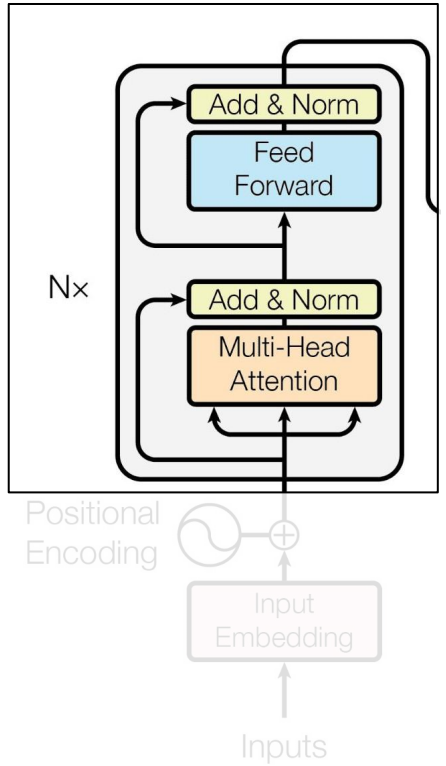
$R^{T \times d_{model}}$

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

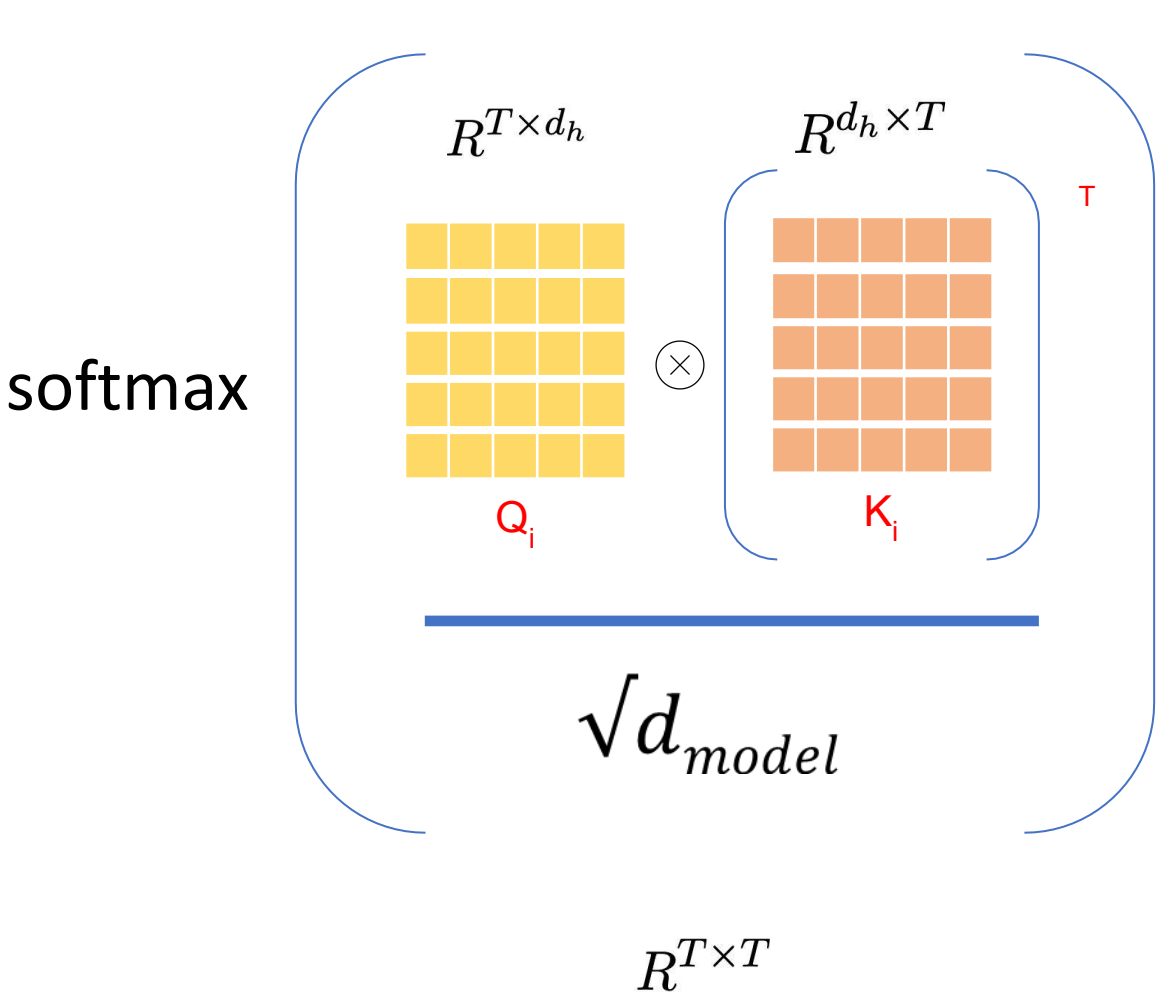


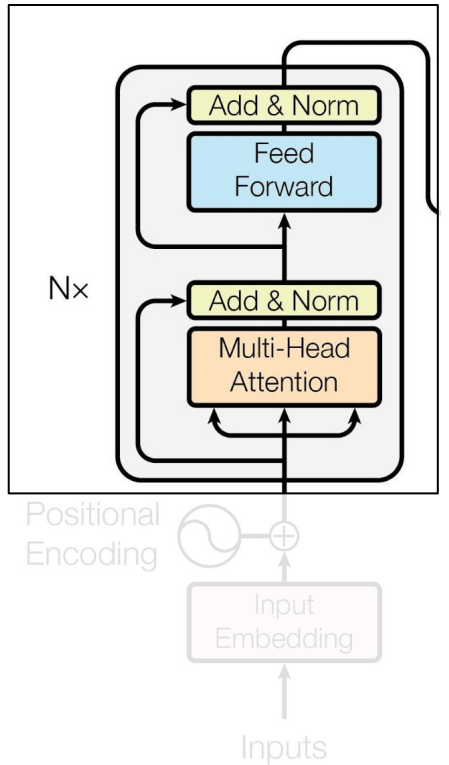
Multi-Head Attention





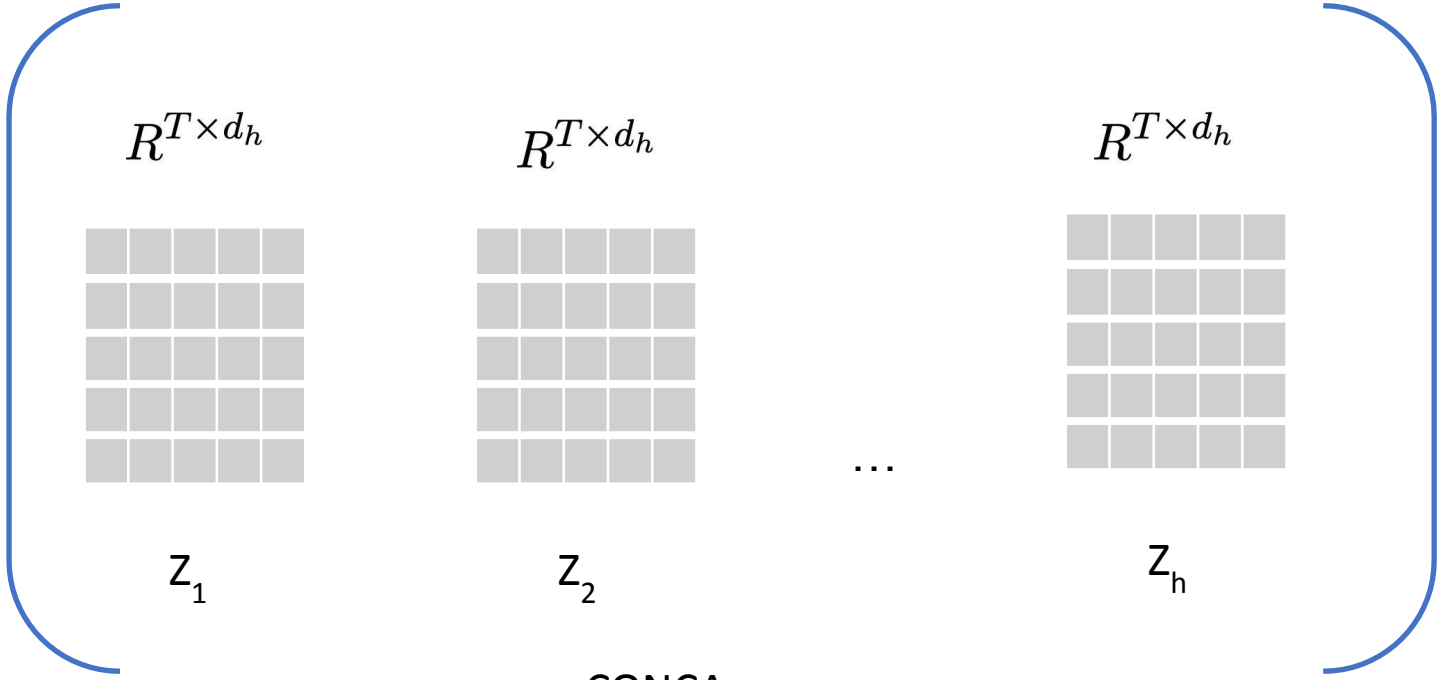
Multi-Head Attention

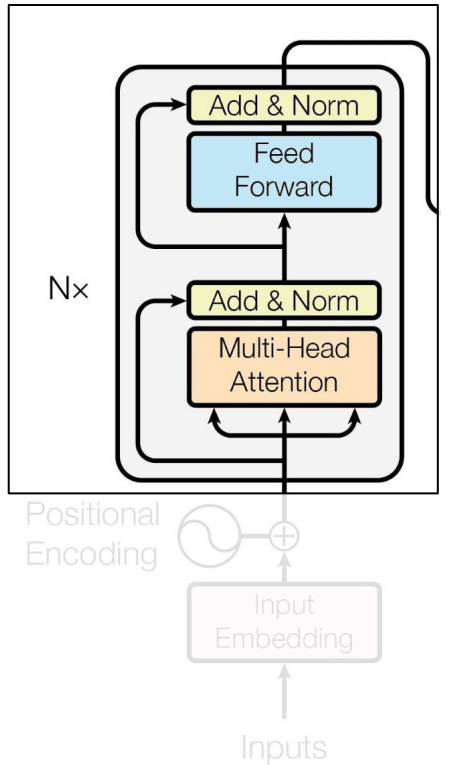




for all i ∈ [1, h]

Multi-Head Attention



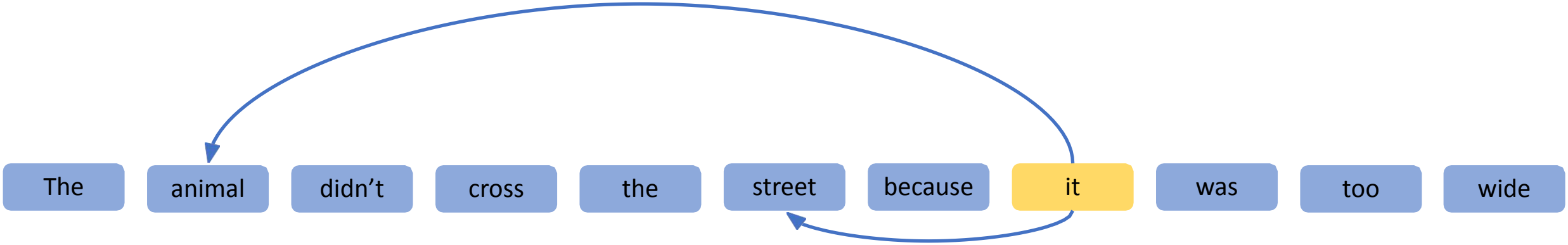


Multi Head Attention : Z

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

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

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.

Positional Encoding

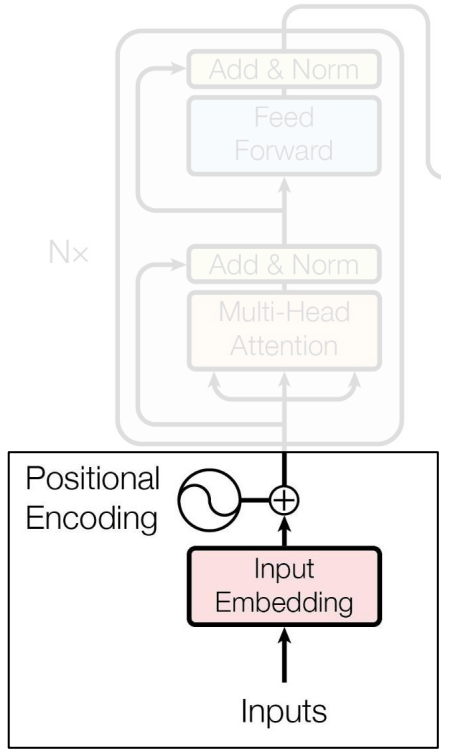
- Transformers lack sequence order awareness.
- **Positional encodings** are added to input embeddings to provide information about token positions.
- **Sinusoidal encoding:**

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

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

Position Encodings

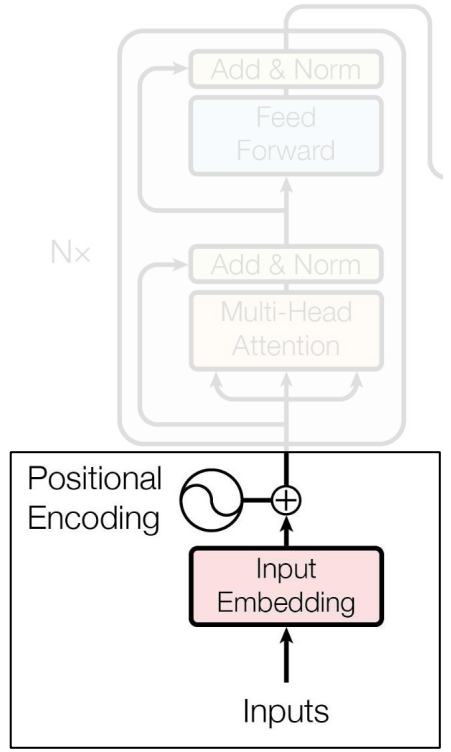
I | ate | an | apple | <eos>



Position Encodings

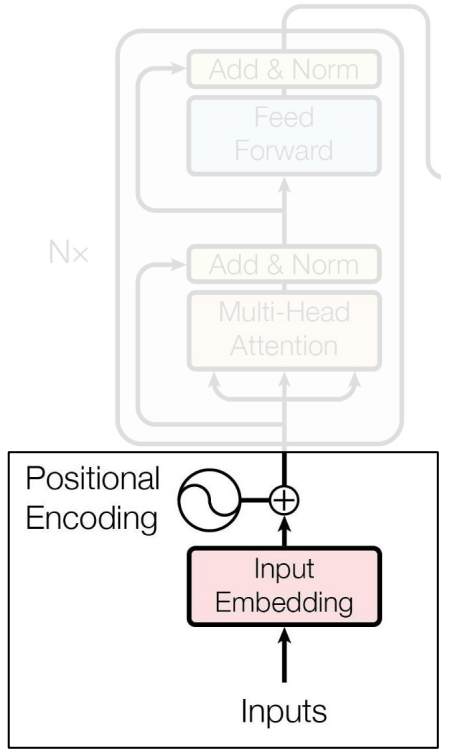
I	ate	an	apple	<eos>

apple	ate	an	I	<eos>



Position Encodings

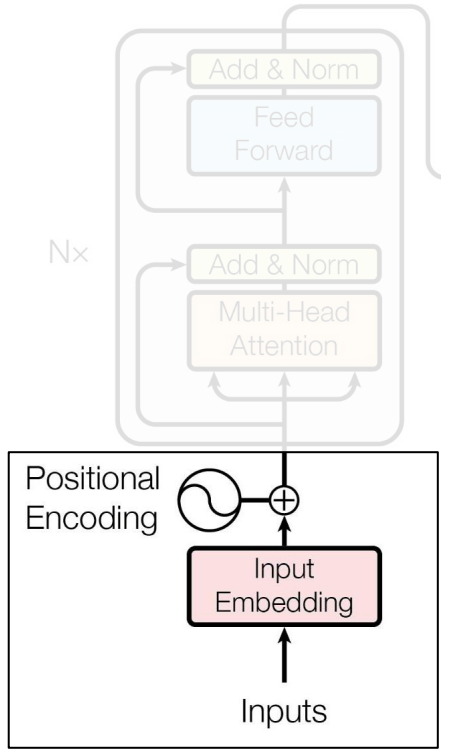
Requirements for Positional Encodings???



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

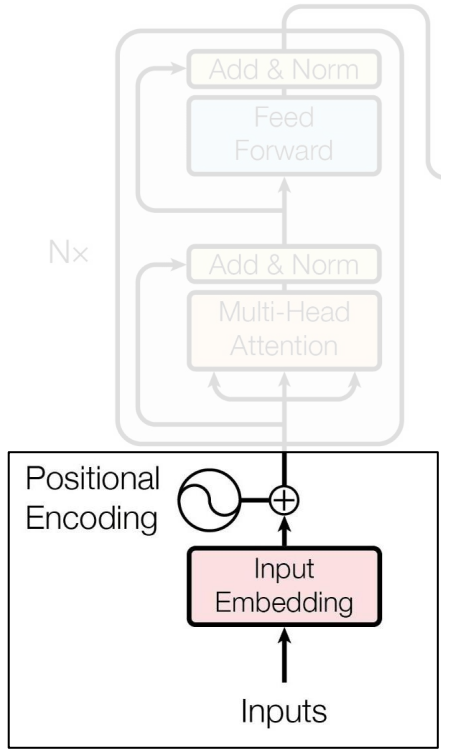
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic

Possible Candidates :

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

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

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



Position Encodings

Requirements for Positional Encodings

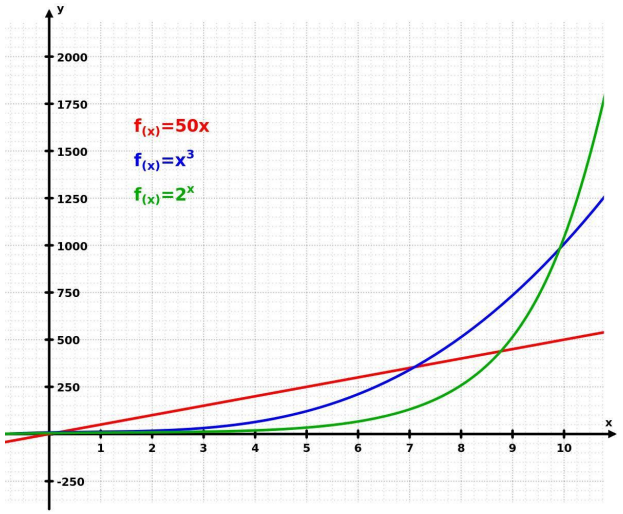
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic

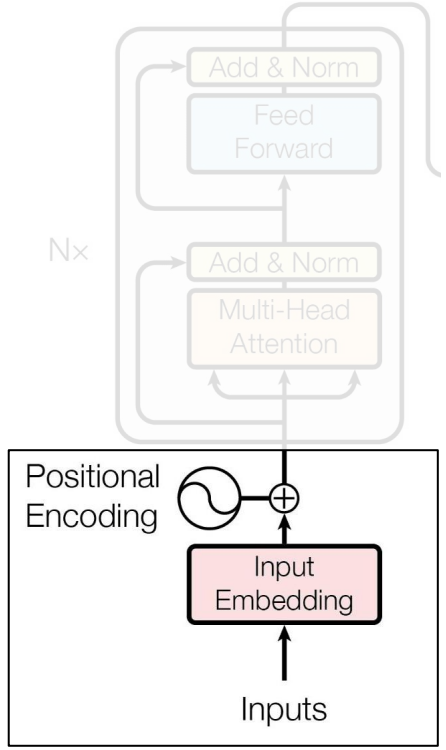
Possible Candidates :

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

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

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





Position Encodings

Requirements for Positional Encodings

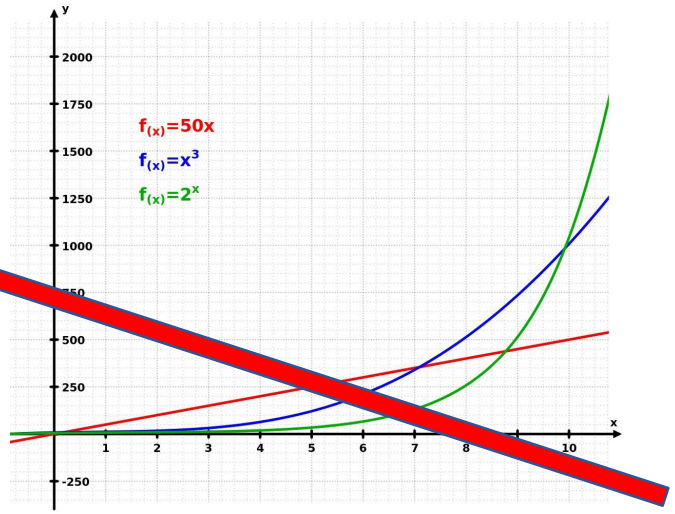
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic

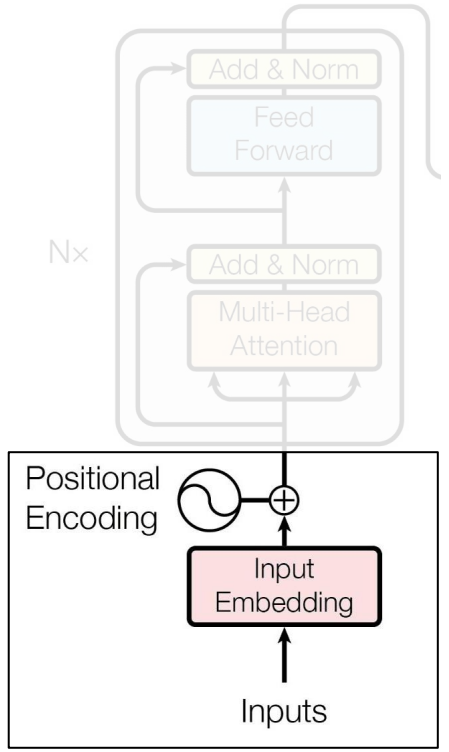
Possible Candidates :

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

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

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





Position Encodings

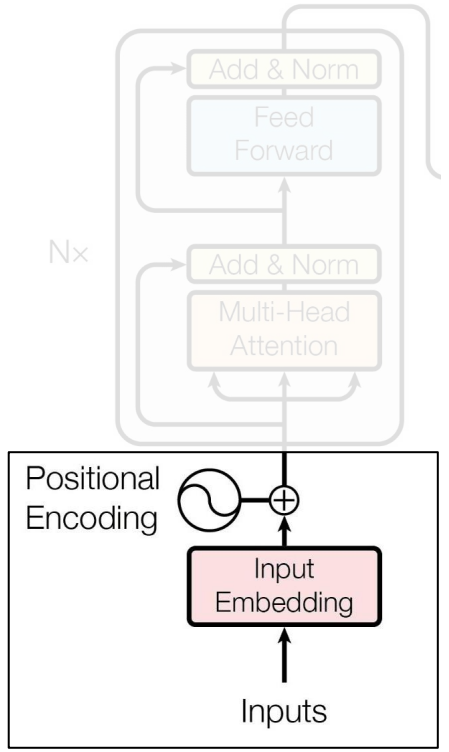
Requirements for Positional Encodings

- 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**

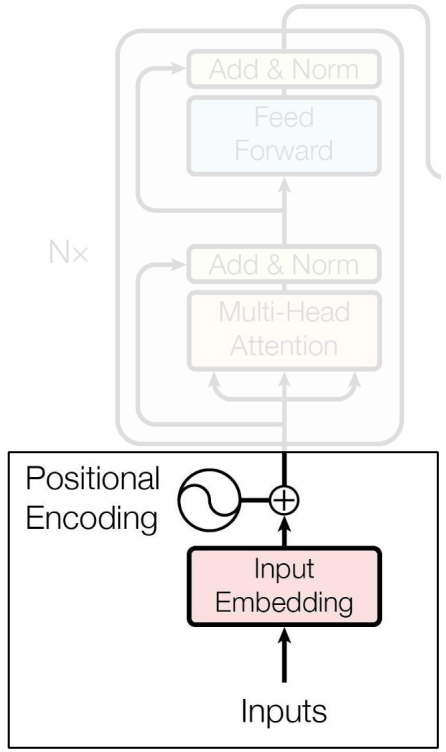
  Possible

  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

أكاديمية كاوست
KAUST ACADEMY

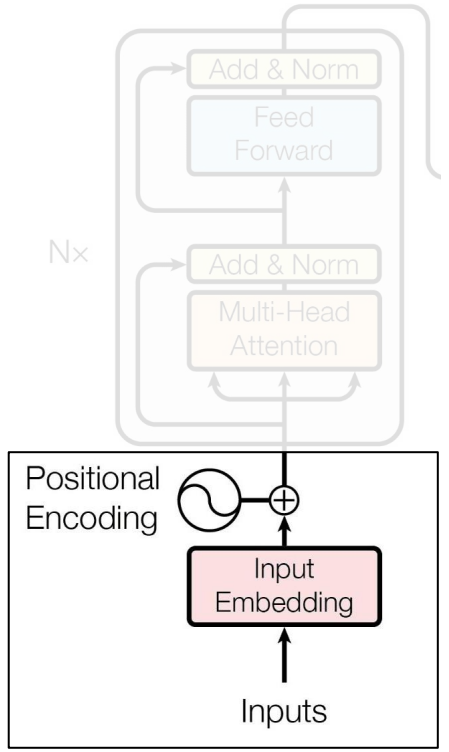
LMH
Lady Margaret Hall

RoFormer: Enhanced Transformer with Rotary Position Embedding

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

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	**89.5**	90.7	88.0	**87.0**	**86.4**	80.2/79.8



REF: Rotary Position Embeddings

Position Encoding

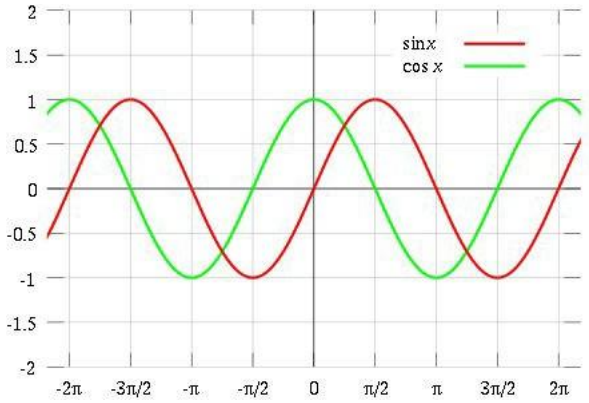
Requirements for Position Encodings

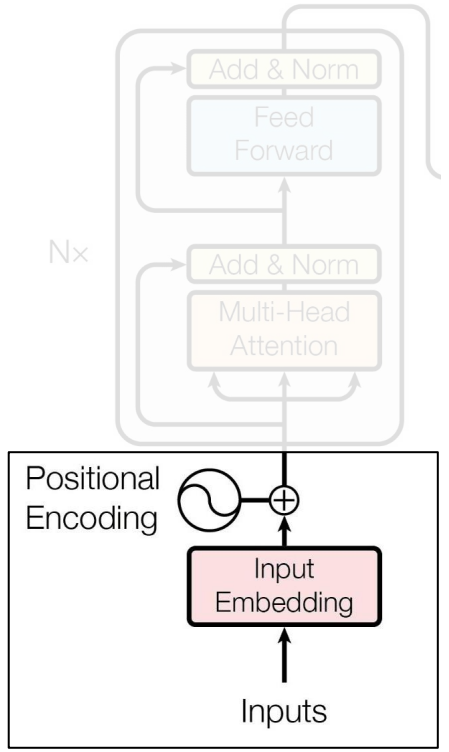
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position
- Bounded

Actual Candidates

*sine(**g(t)**)*

*cosine(**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**.

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

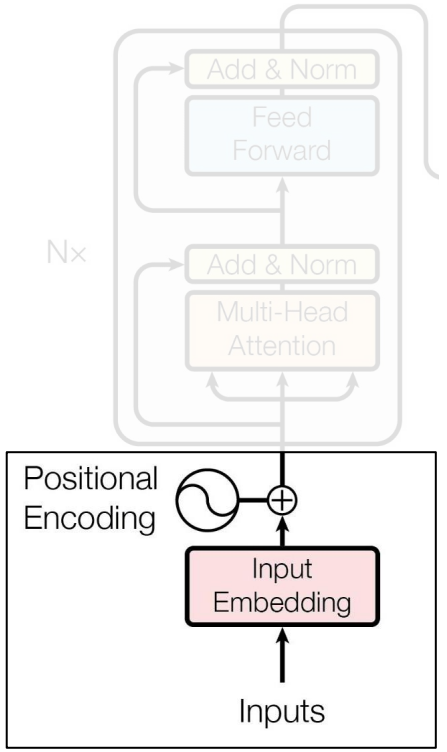
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

pos -> idx of the token in input sentence

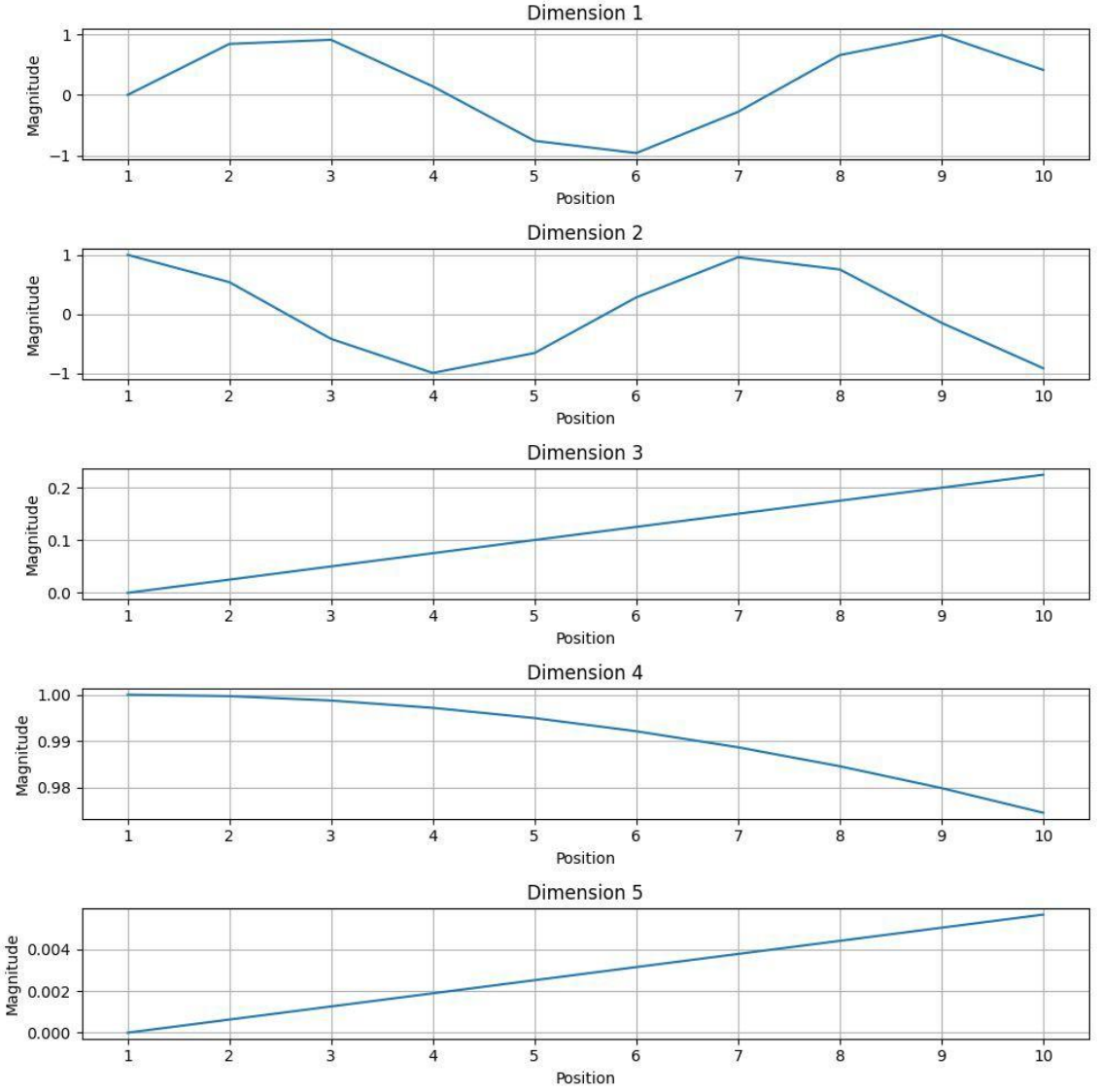
i -> i^{th} dimension out of d

d model -> embedding dimension of each token

Different calculations for odd and even embedding indices



Position Encoding

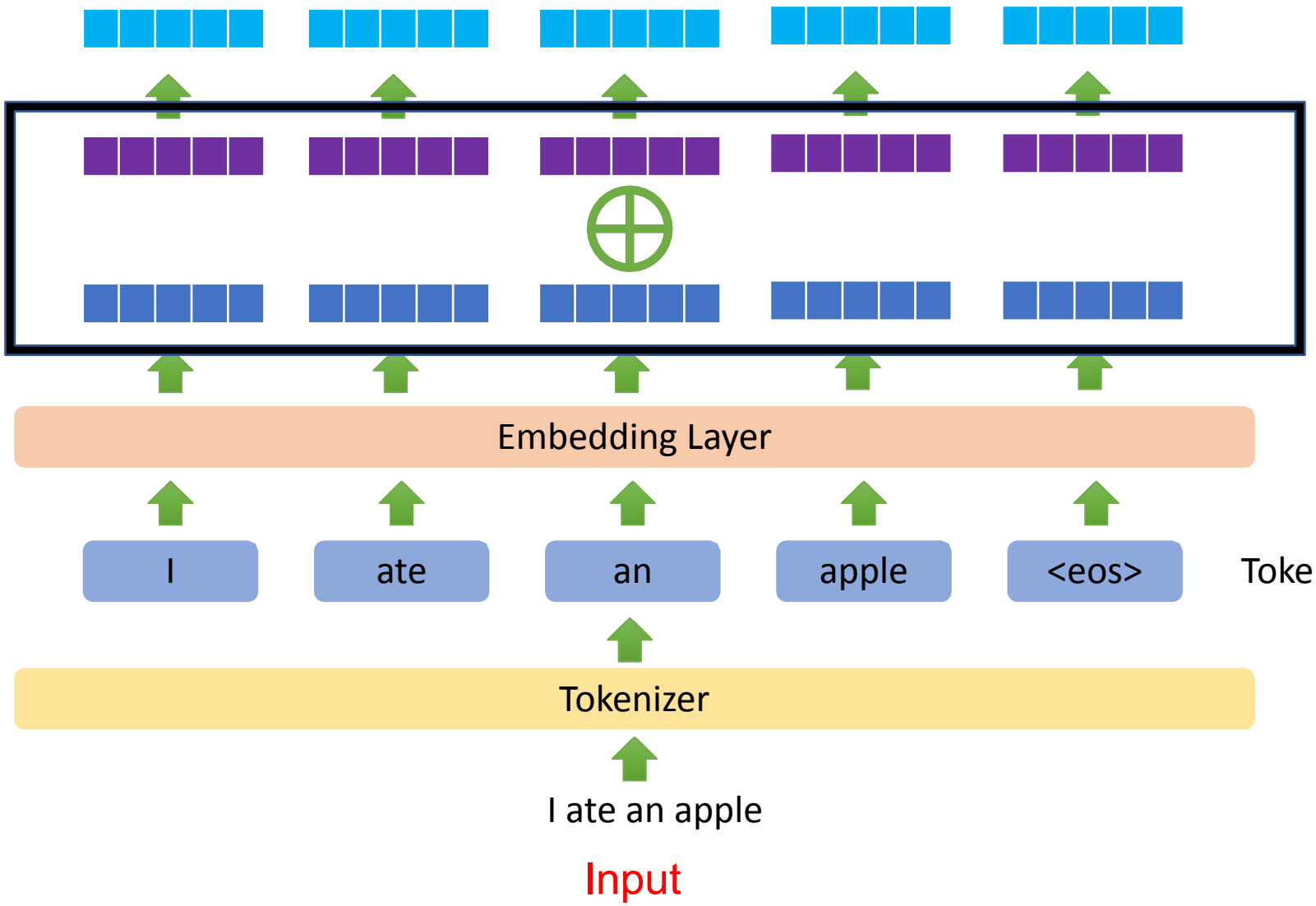


Positional Encoding:

	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

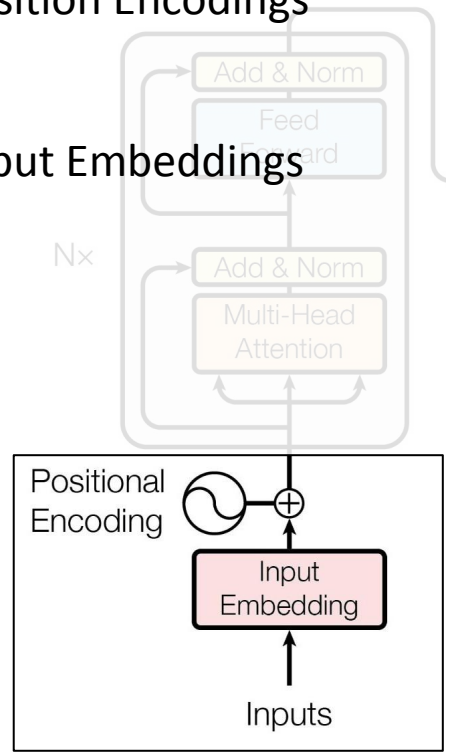
Position Encoding



Final Input Embeddings

Position Encodings

Input Embeddings



Alternatives to Positional Encoding

- **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.

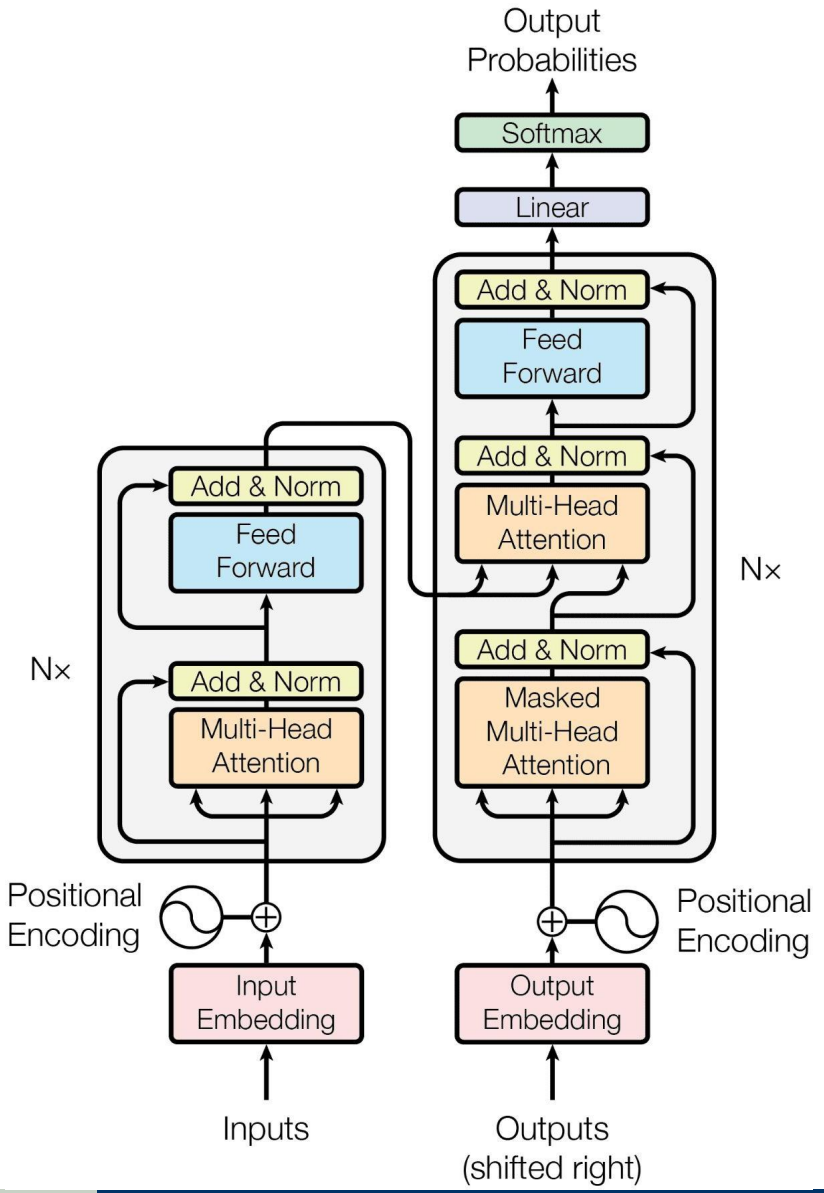
Transformer Block

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

Note

Residual connections and layer normalization are crucial for effective training.

Transformer Block

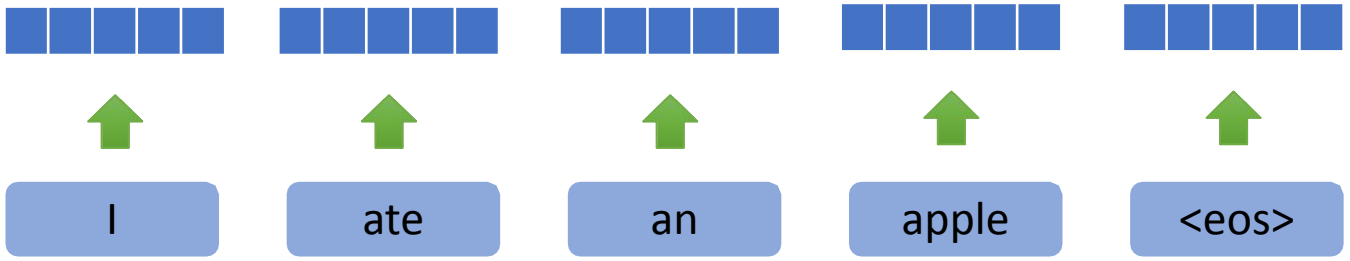


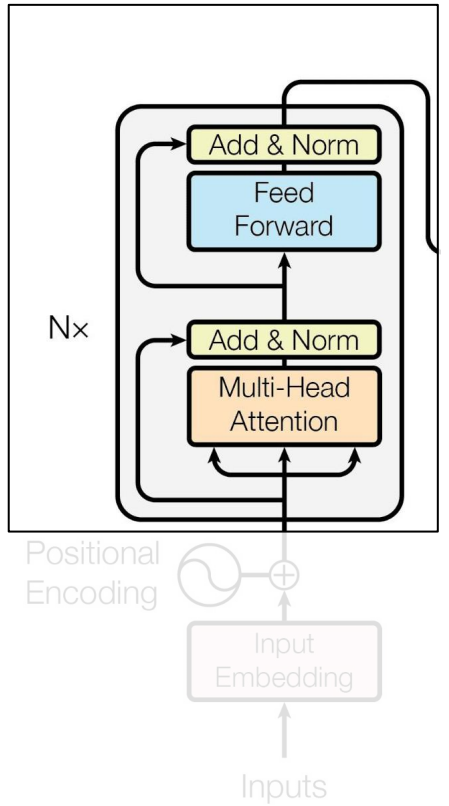
Encoder-Decoder Architecture

- **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.

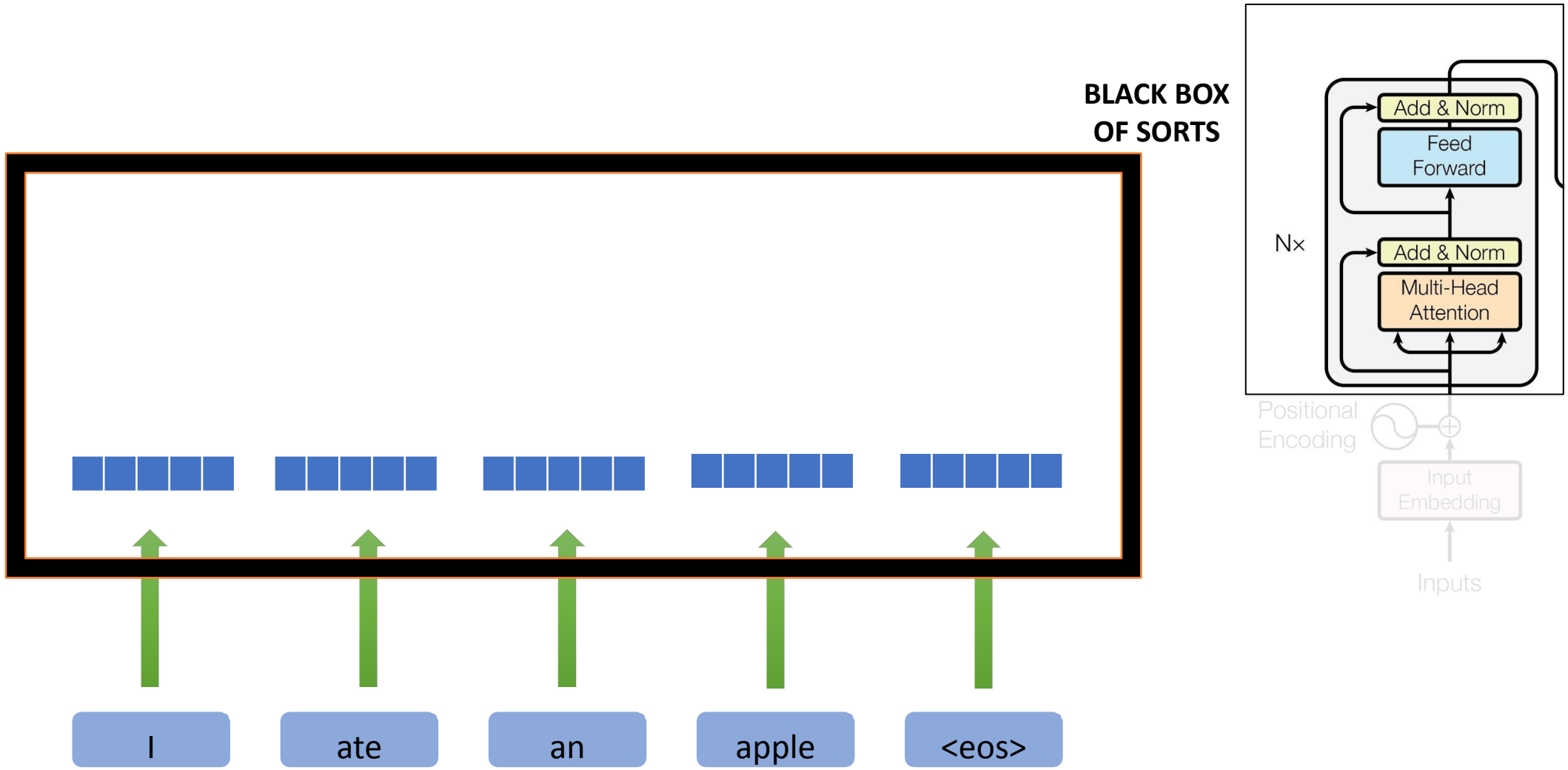
Encoder

WHERE IS THE CONTEXT ?



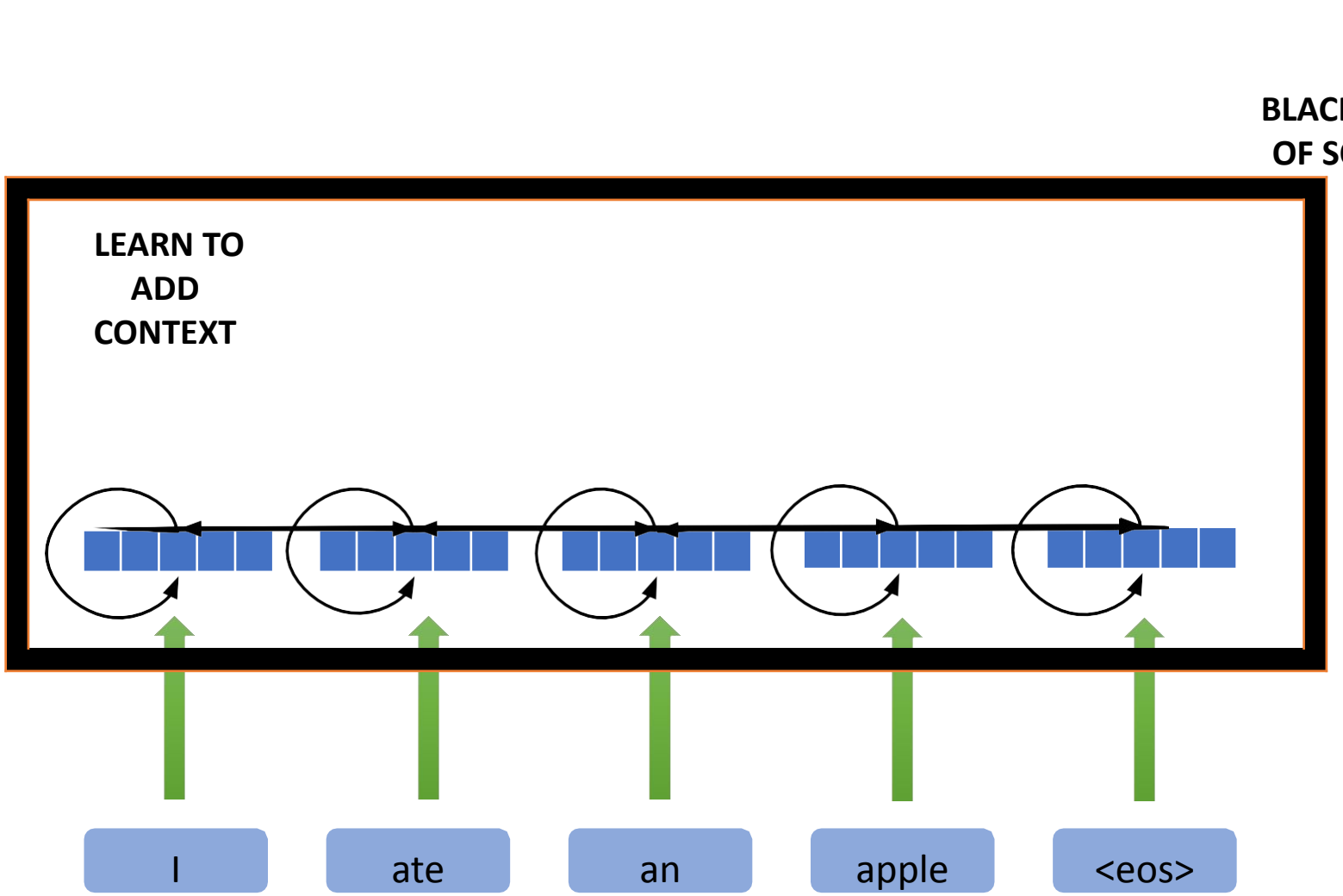


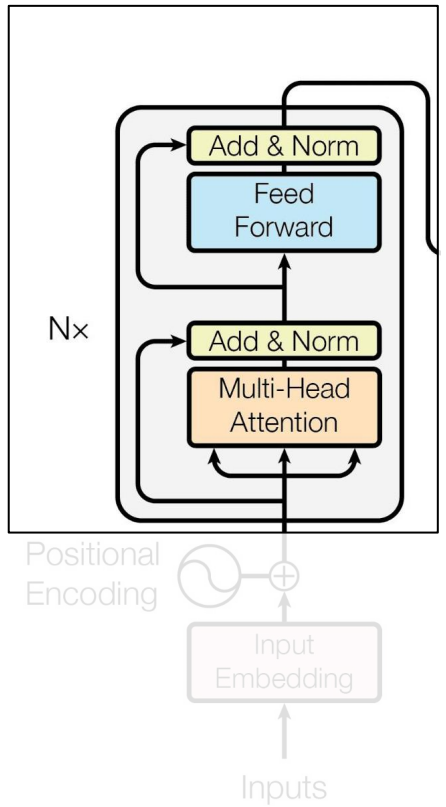
Encoder


BLACK BOX
OF SORTS
I
ate
an
apple
<eos>

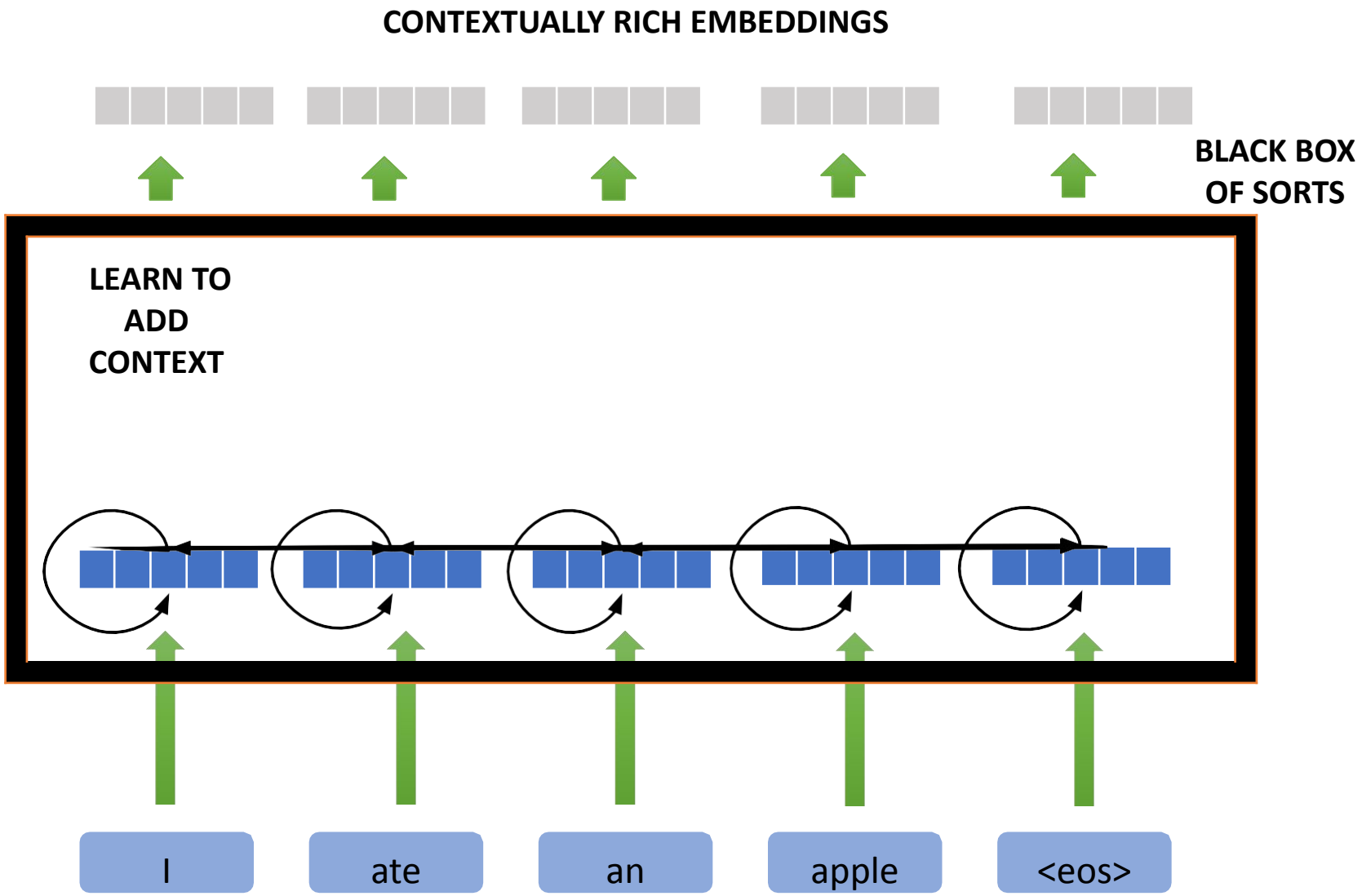
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs

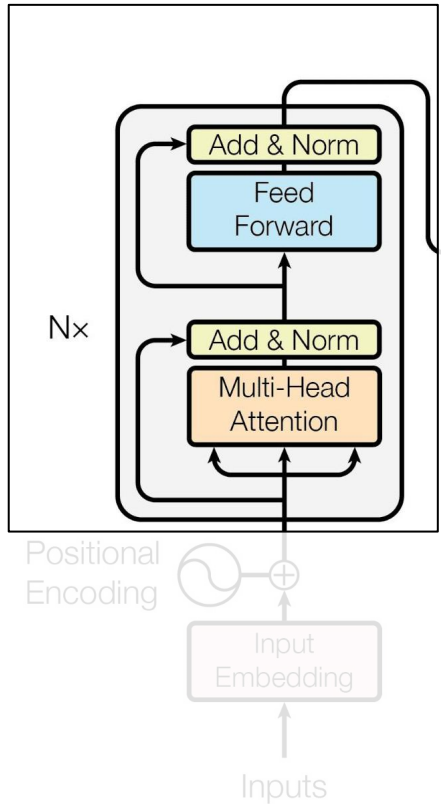
Encoder


BLACK BOX
OF SORTS
LEARN TO
ADD
CONTEXT
I
ate
an
apple
<eos>

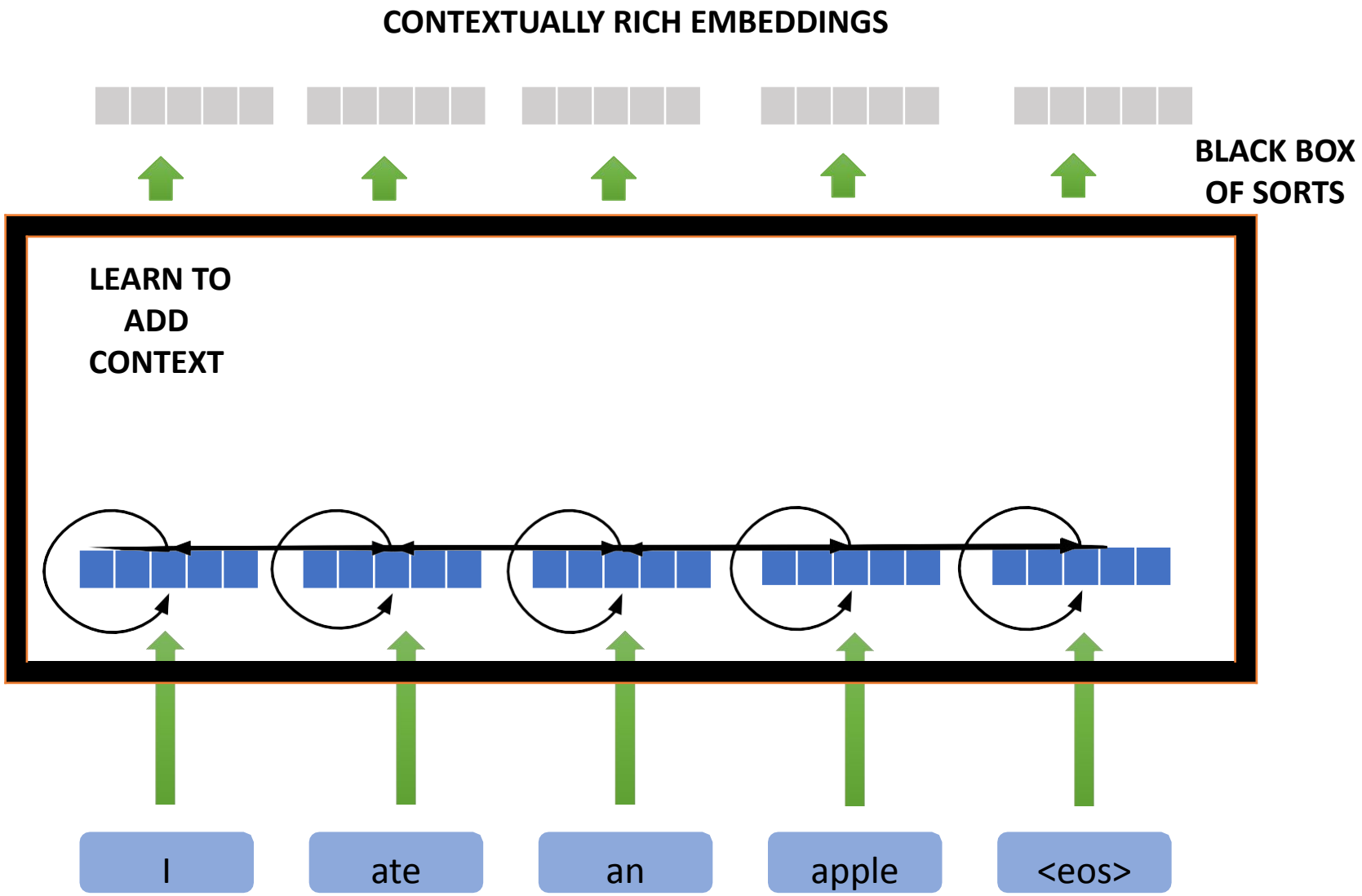

Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Inputs

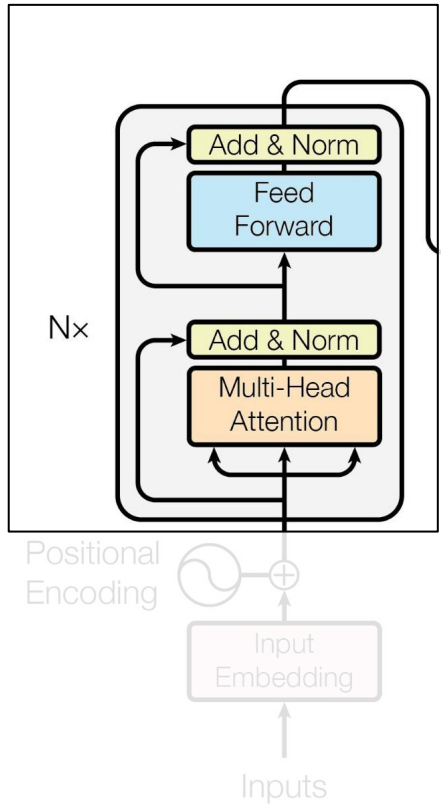
Encoder


CONTEXTUALLY RICH EMBEDDINGS
BLACK BOX OF SORTS
LEARN TO ADD CONTEXT
I
ate
an
apple
<eos>


Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

Encoder


CONTEXTUALLY RICH EMBEDDINGS
BLACK BOX OF SORTS
LEARN TO ADD CONTEXT
I
ate
an
apple
<eos>


Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

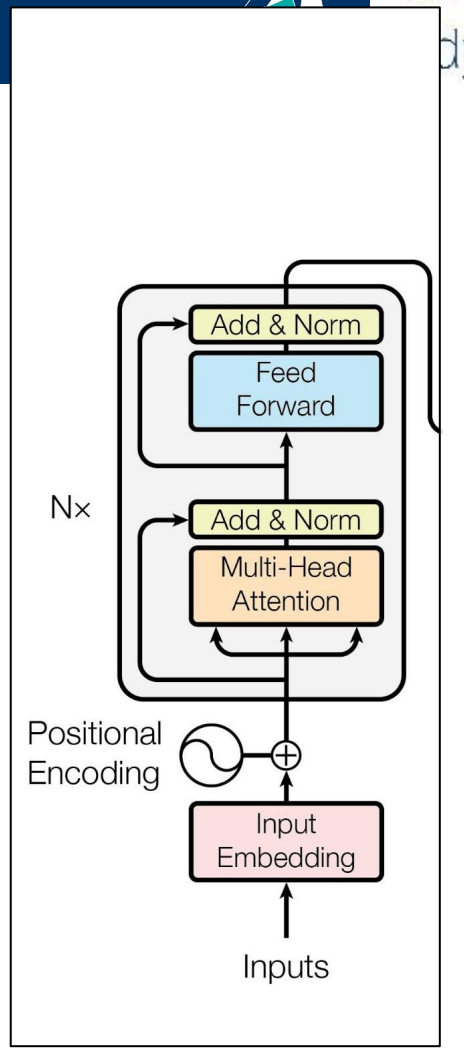
Encoder-Only Transformers

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

Encoders

Encoder

ENCODER



Encoders

Encoder

ENCODER

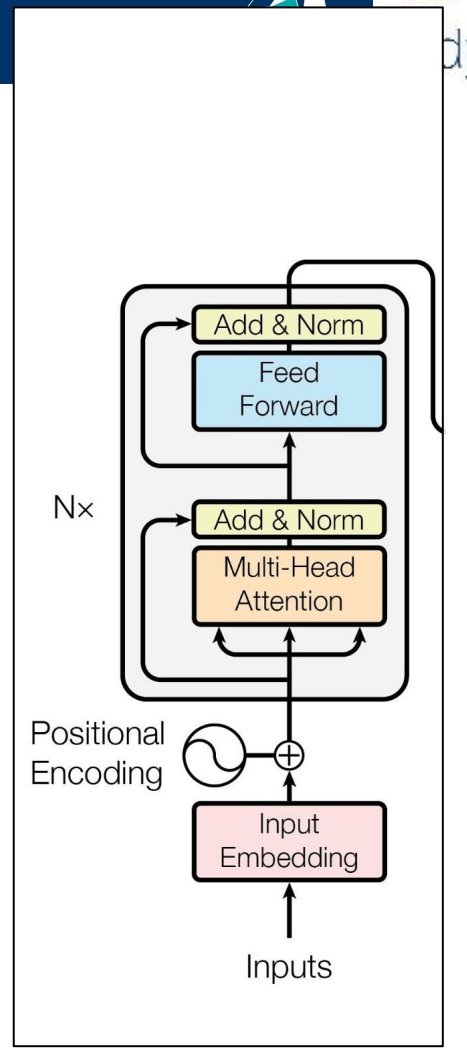
.
.
.

ENCODER

ENCODER

Input to Encoder$_{i+1}$

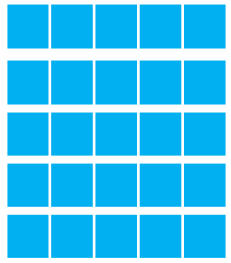
Output from Encoder$_{i}$



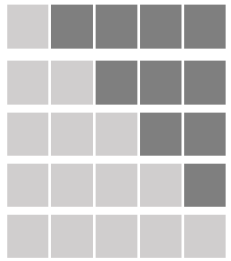
Encoder Decoder Attention

Encoder Decoder Attention ?

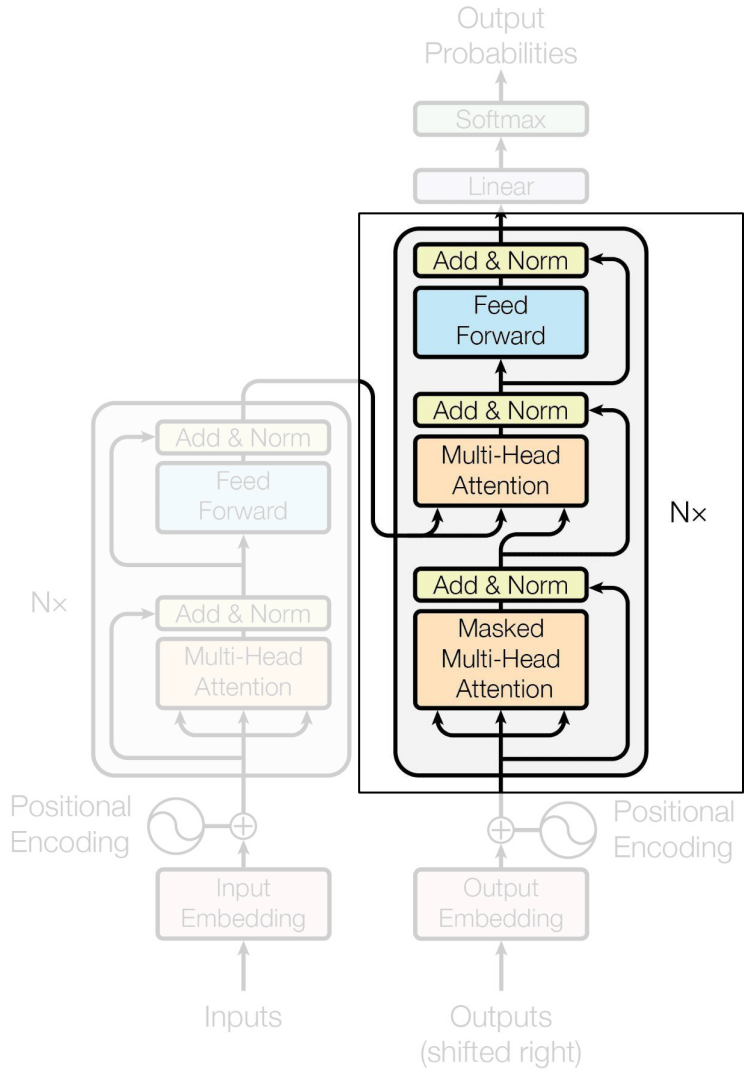
Add & Norm



Input



Norm(Z')

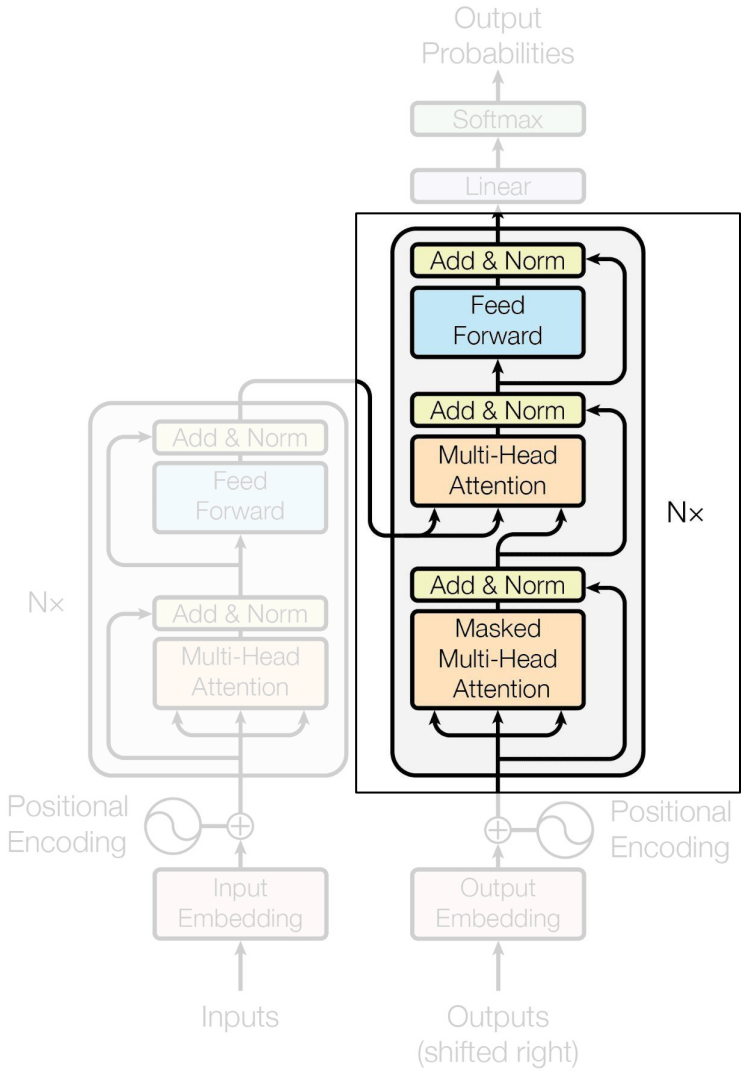


Encoder Decoder Attention

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

Encoder Decoder Attention ?



Encoder Decoder Attention

أكاديمية كاوست
KAUST ACADEMY

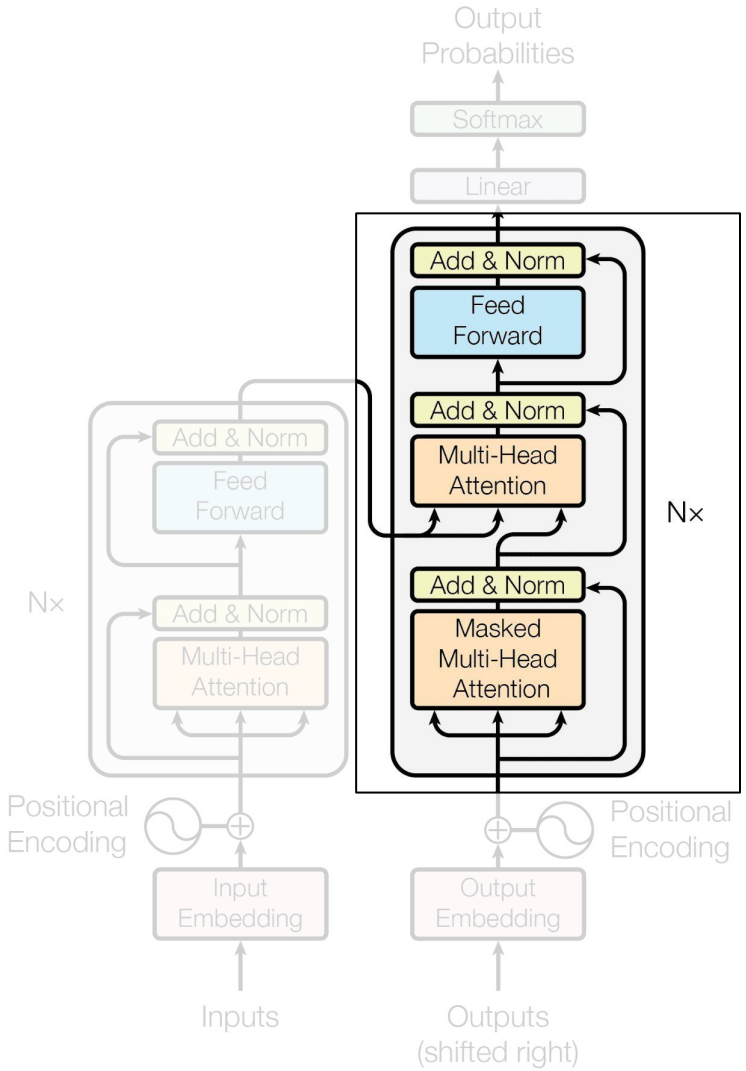
LMH
Lady Margaret Hall

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



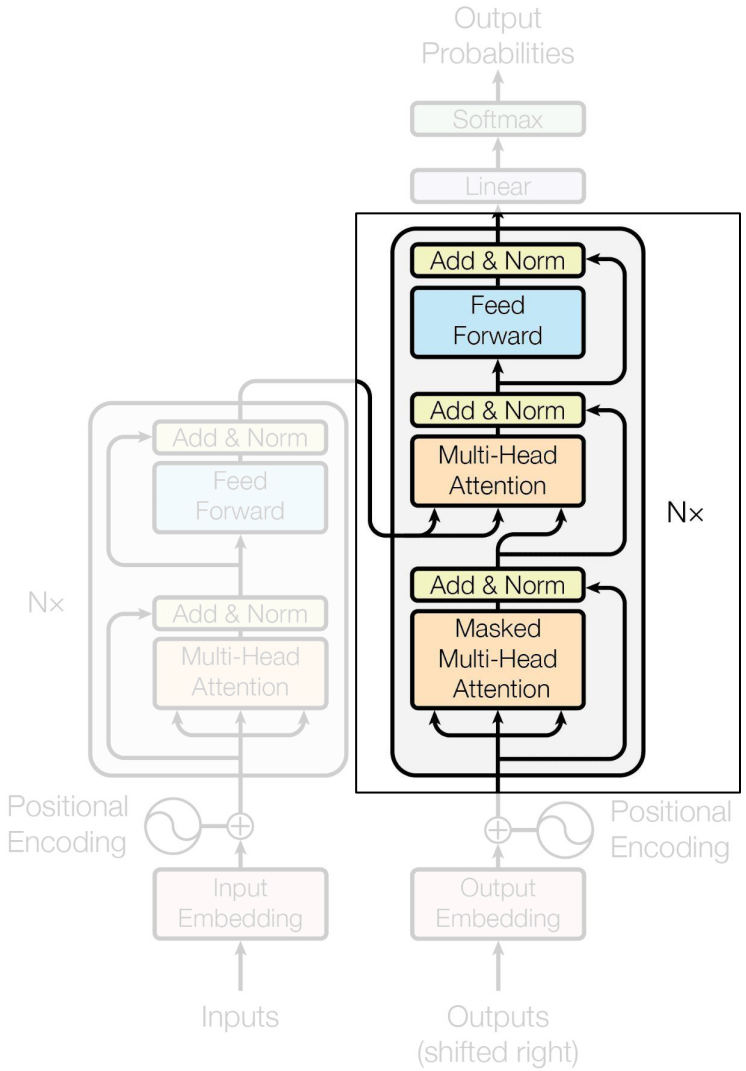
Encoder Decoder Attention

Encoder

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

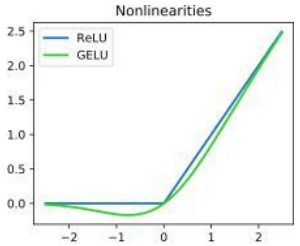
Decoder

Queries from **Decoder Inputs**



NOTE: Every decoder block receives the same FINAL encoder output

Encoder Decoder Attention



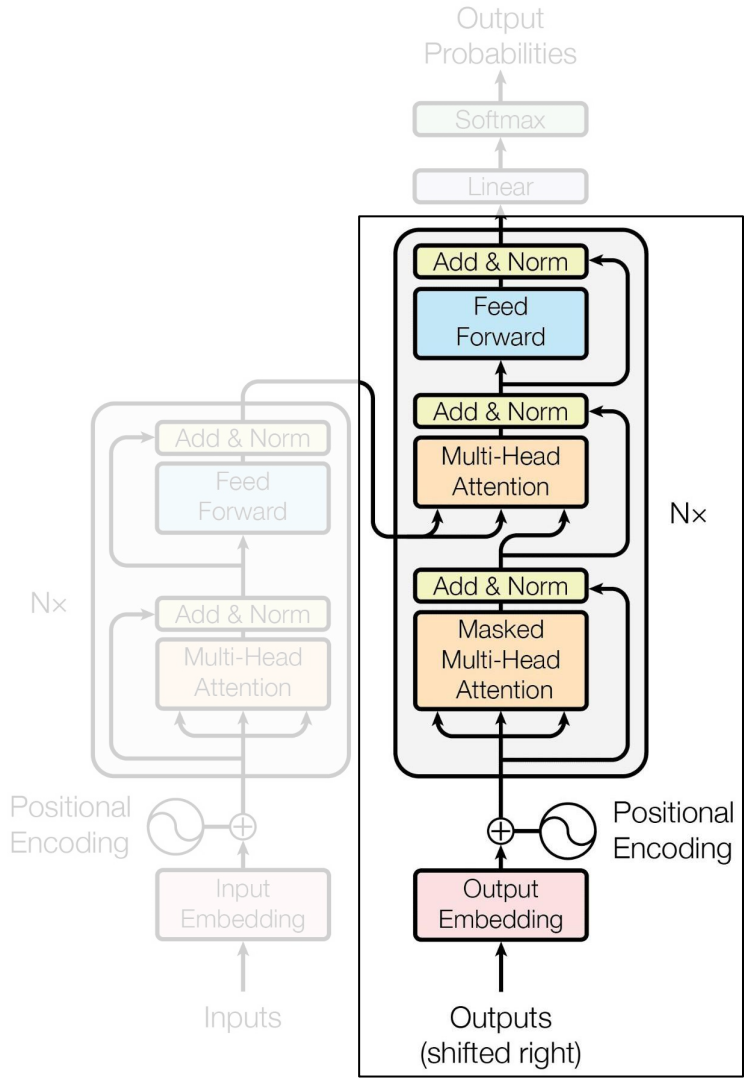
- Non Linearity
- Complex Relationships
- Learn from each other

Feed Forward

Residuals

Add n Norm Decoder Self Attn

Norm(Z’’)

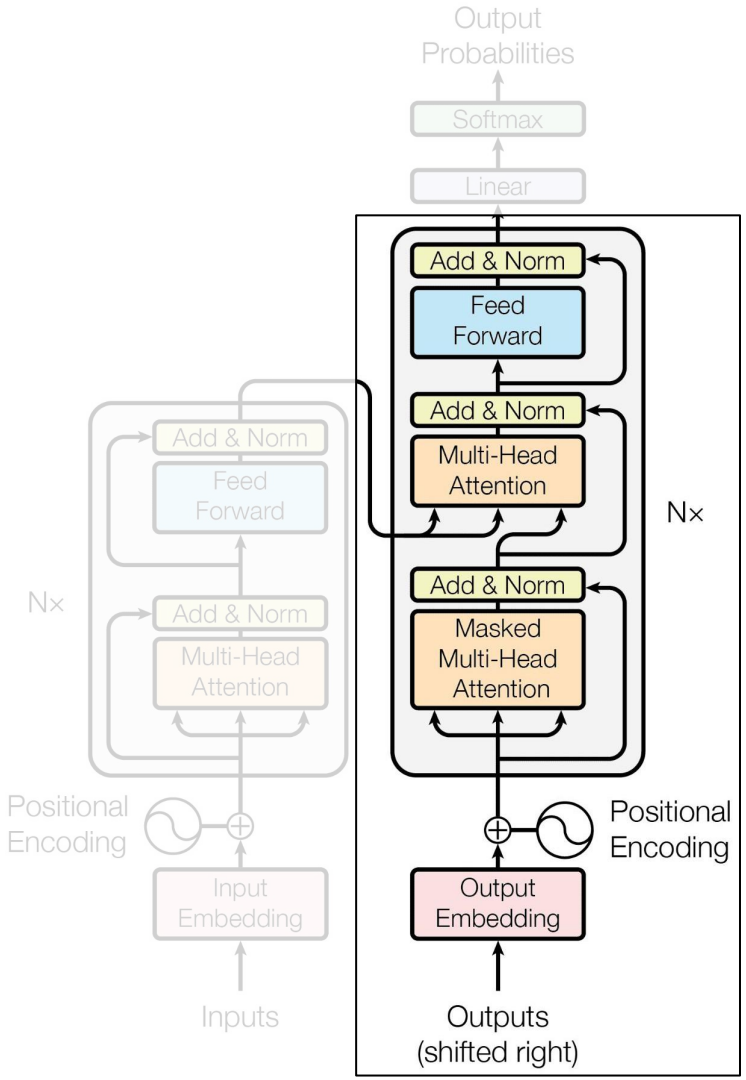


Decoder-Only Transformers

- **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

DECODER

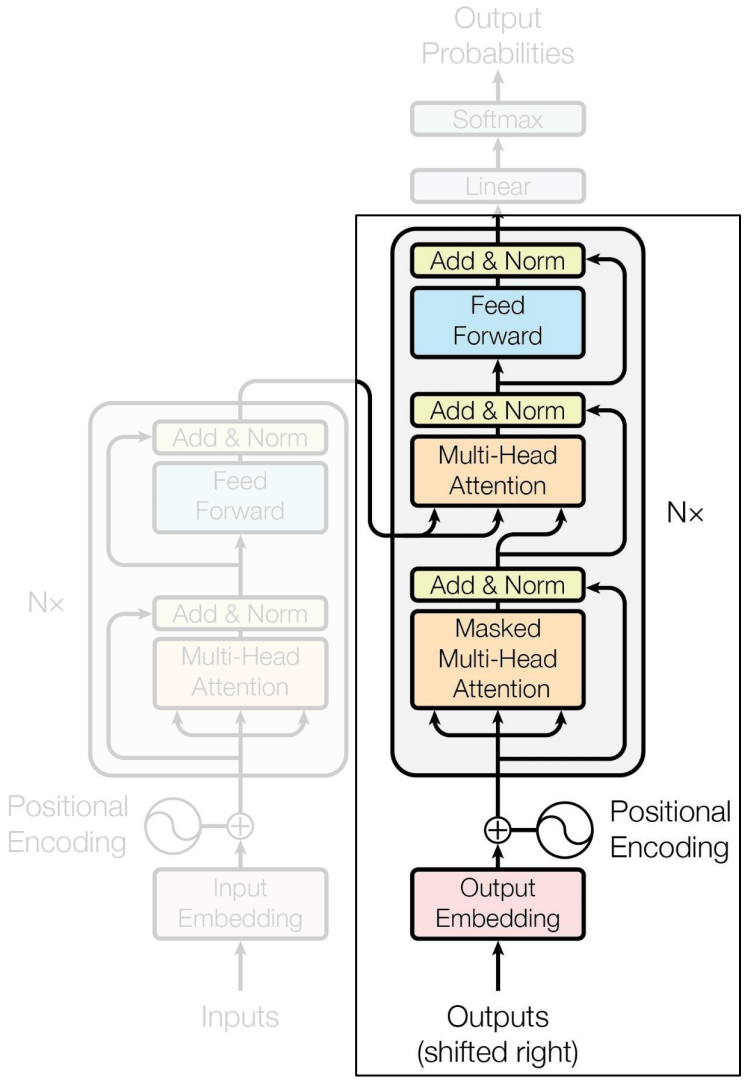
.
.
.

DECODER

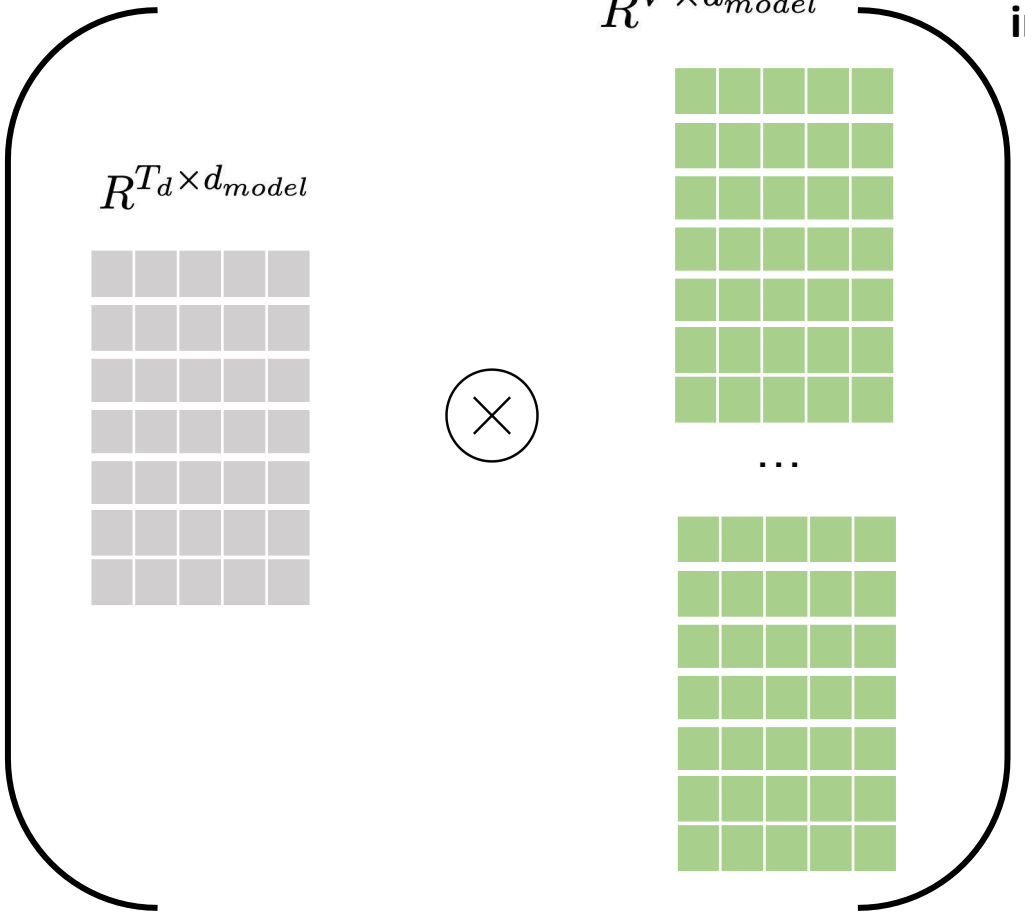
DECODER

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

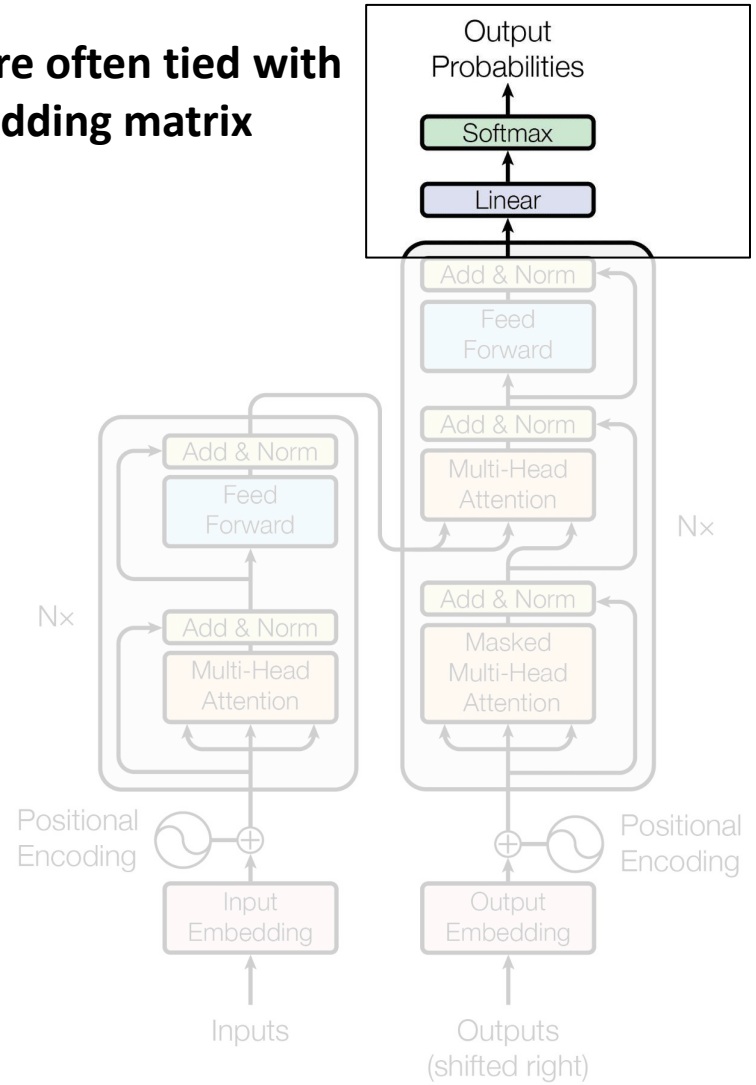
Decoder output



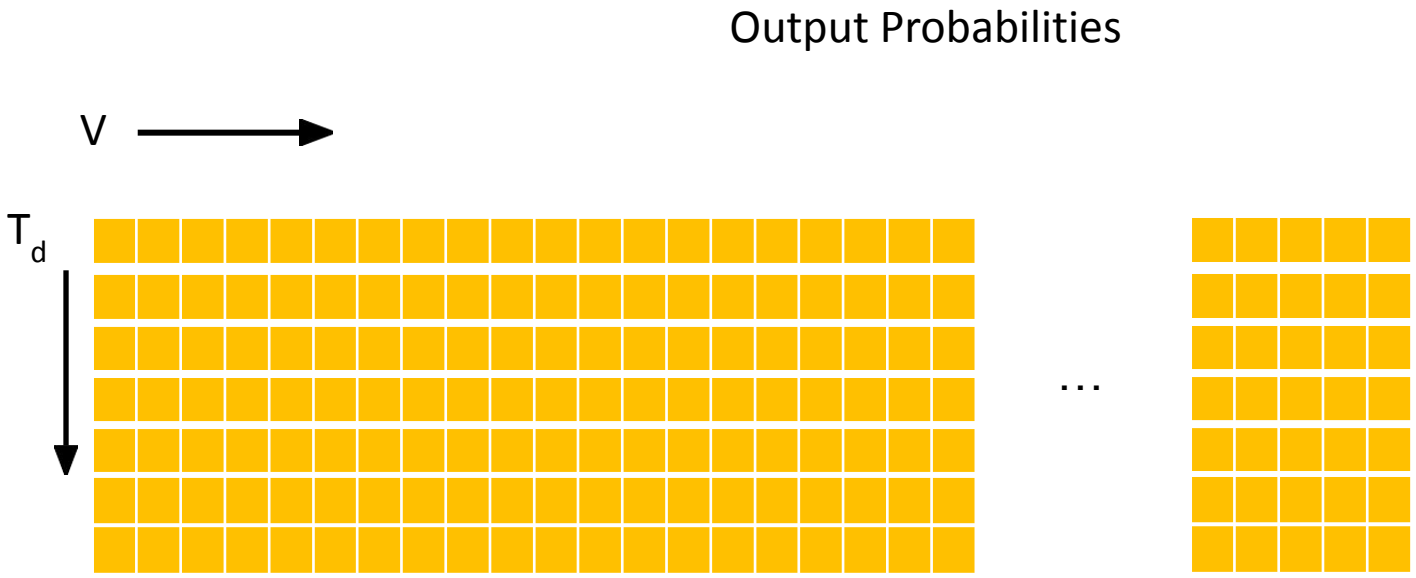
Linear



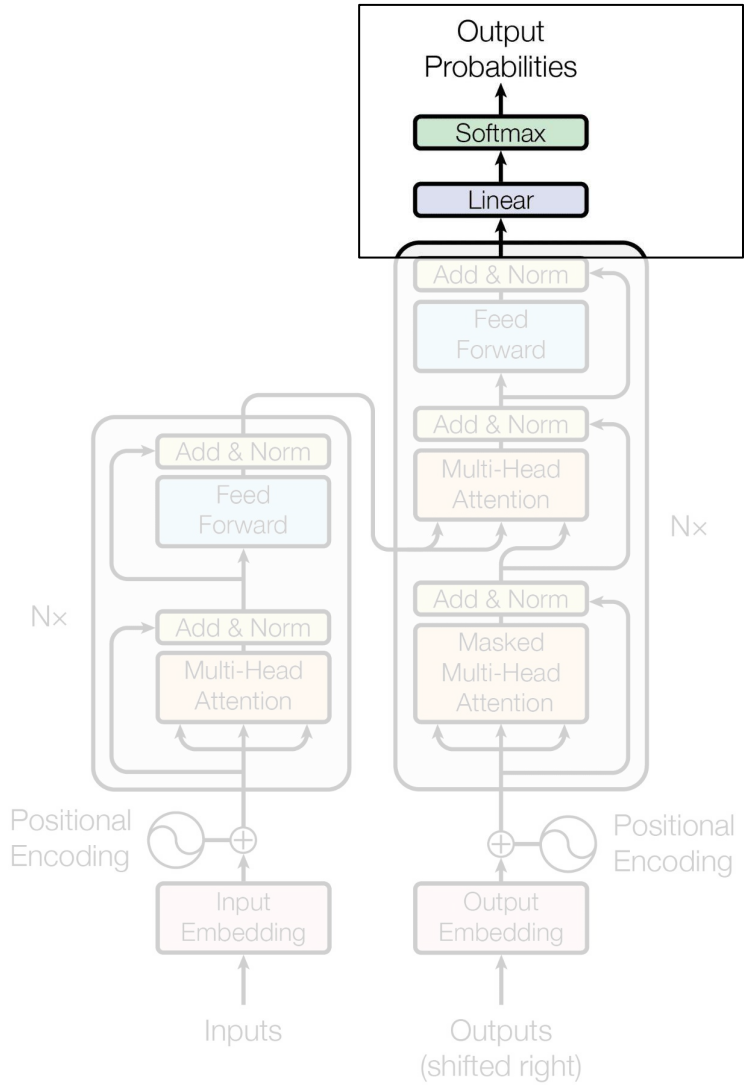
Linear weights are often tied with input embedding matrix



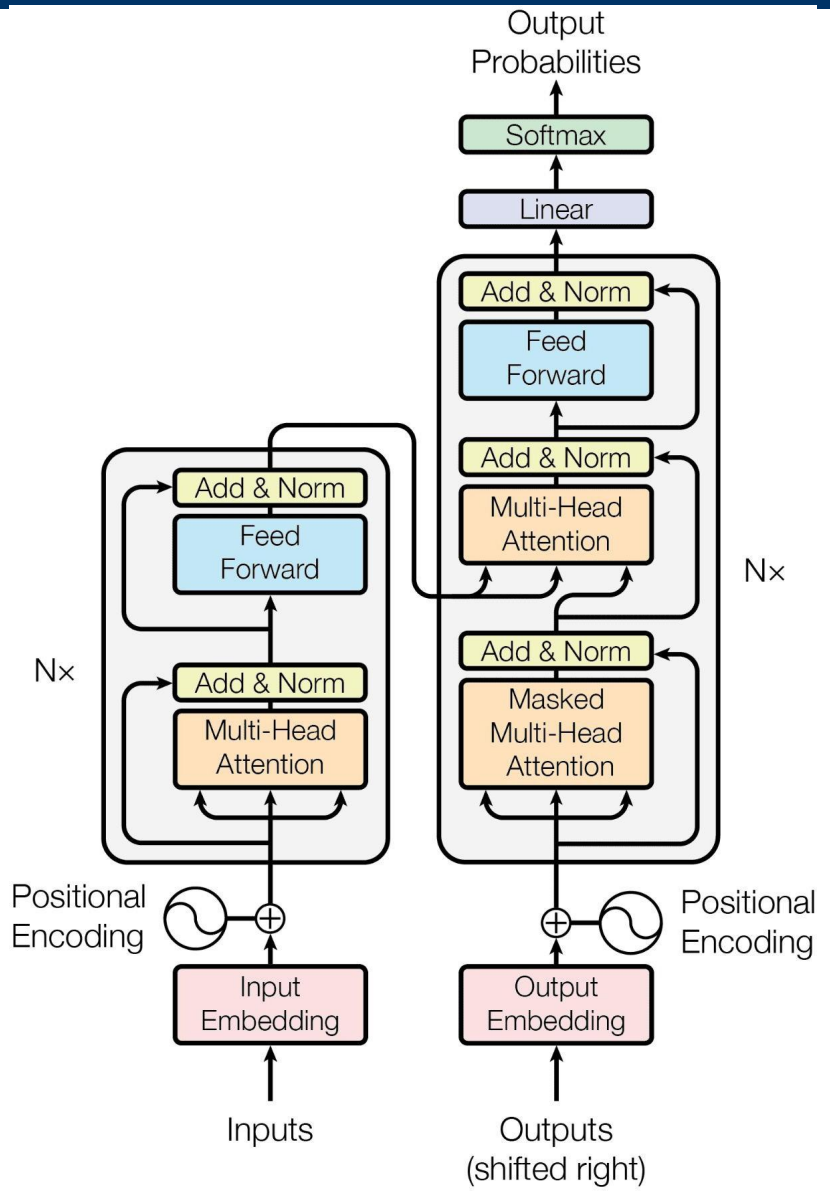
Softmax



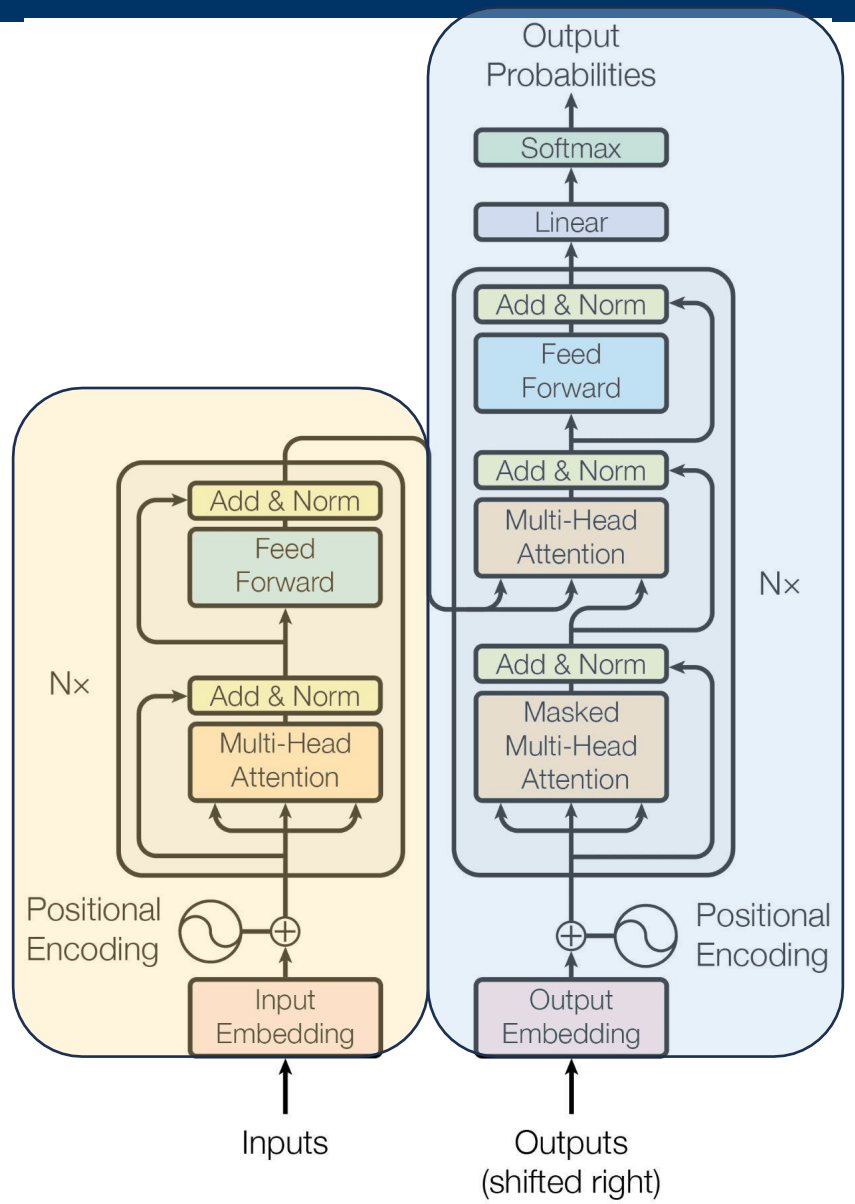
$$R^{T_d \times V}$$



Transformers



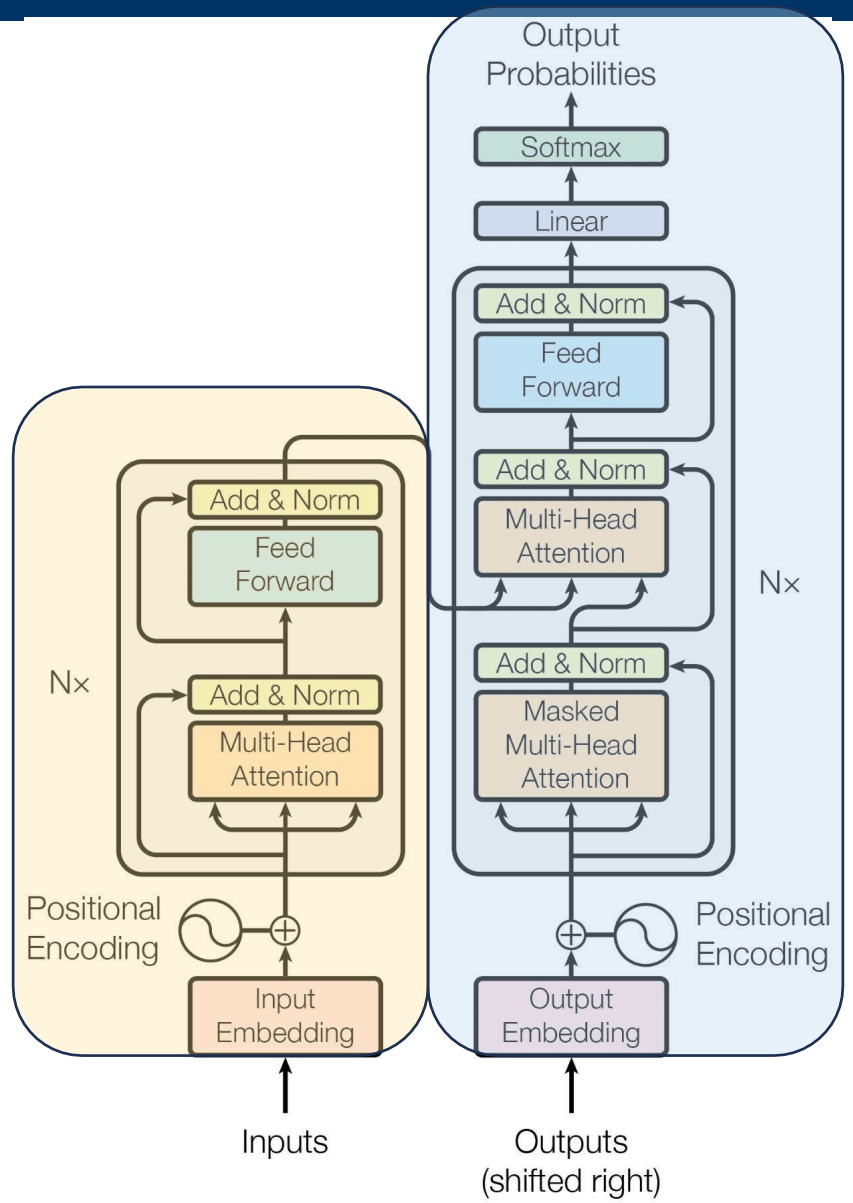
Transformers



Representation

Generation

Transformers



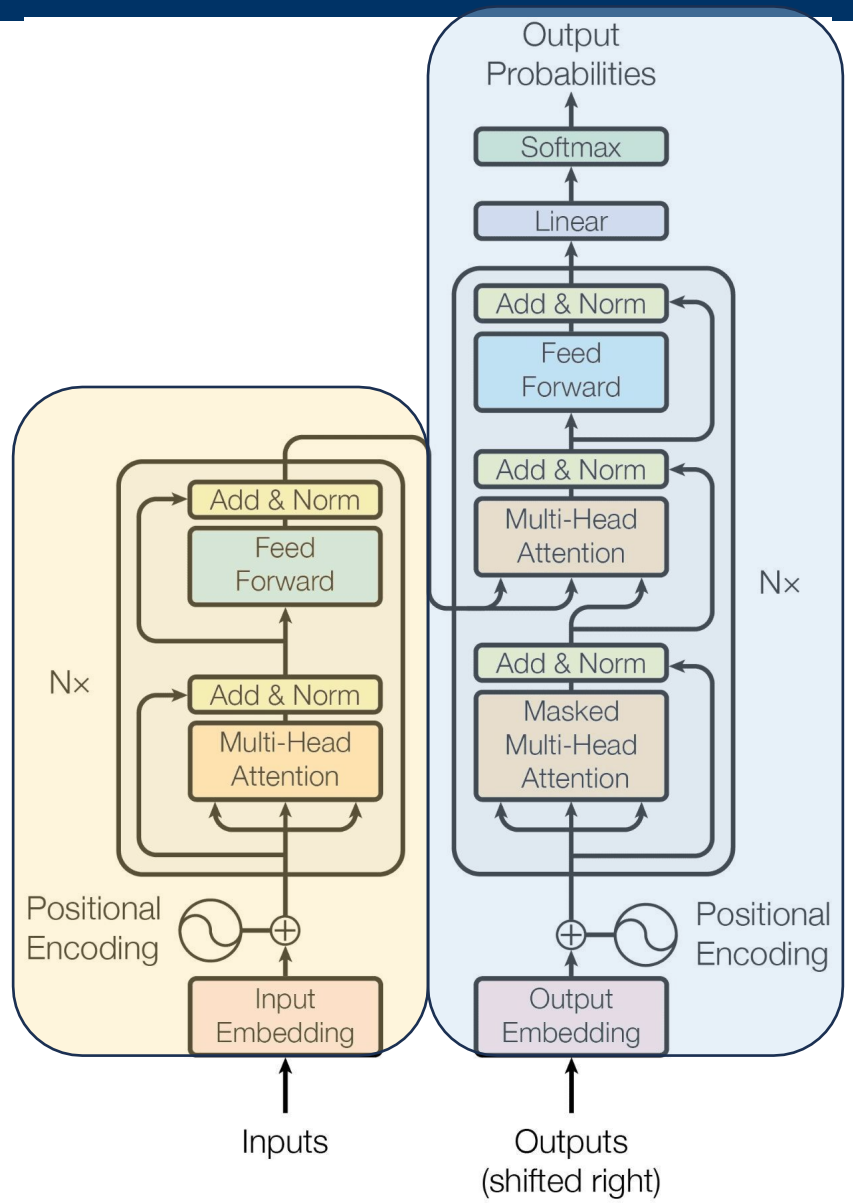
Input – input tokens
Output – hidden states

Representation

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

Generation

Transformers



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

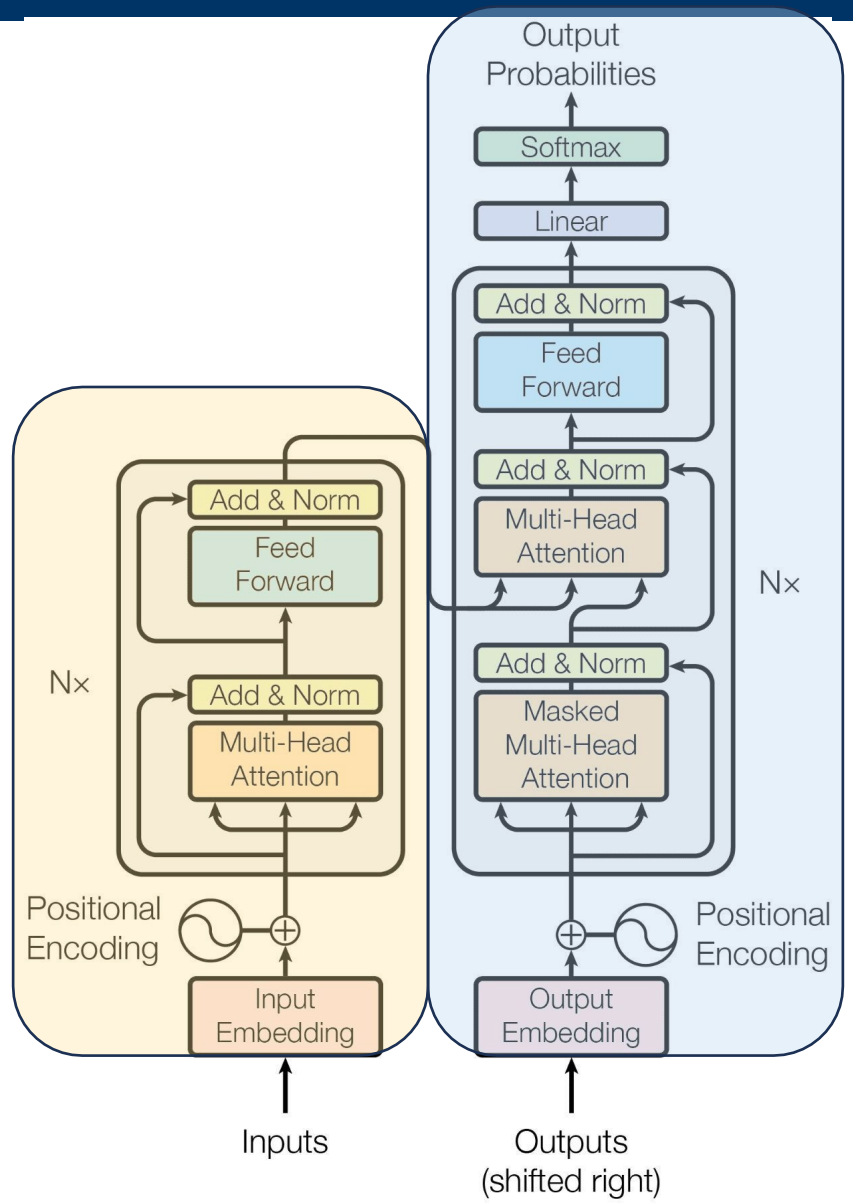
Transformers

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

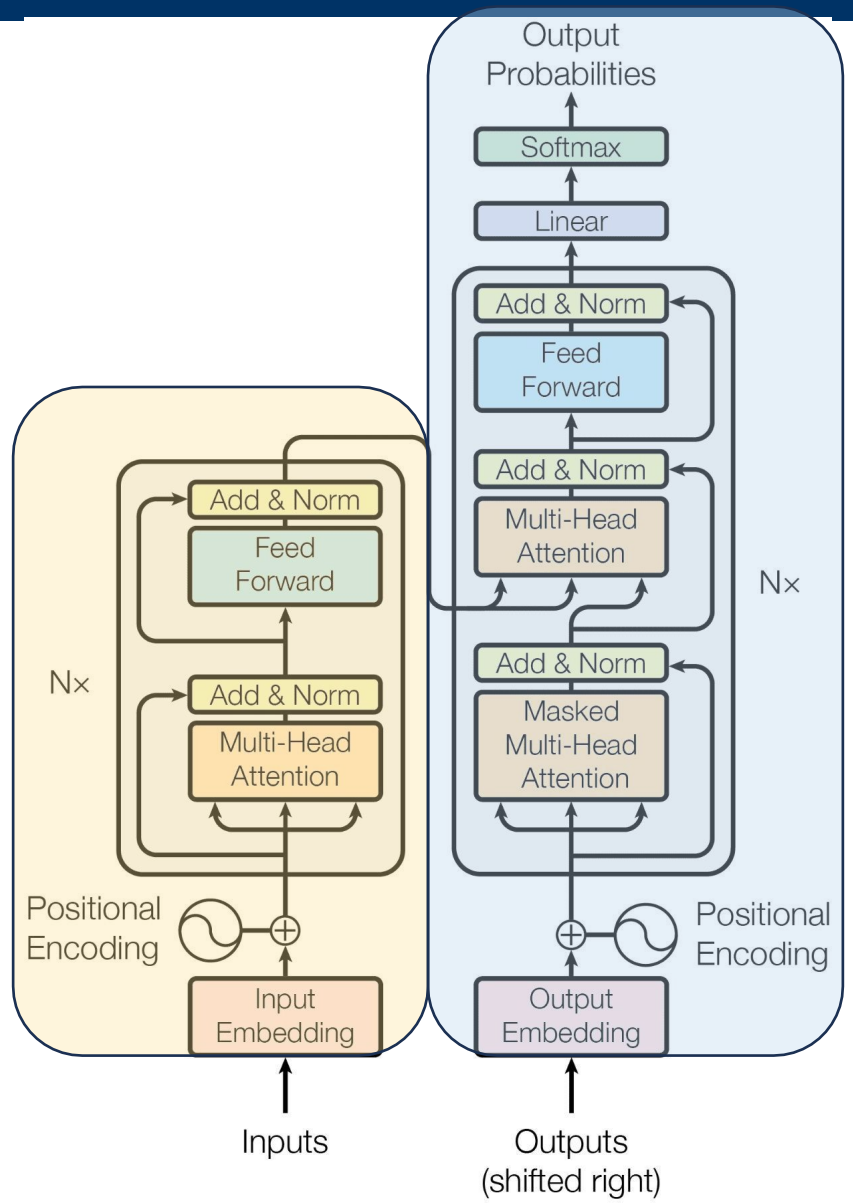
Input – input tokens
Output – hidden states

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

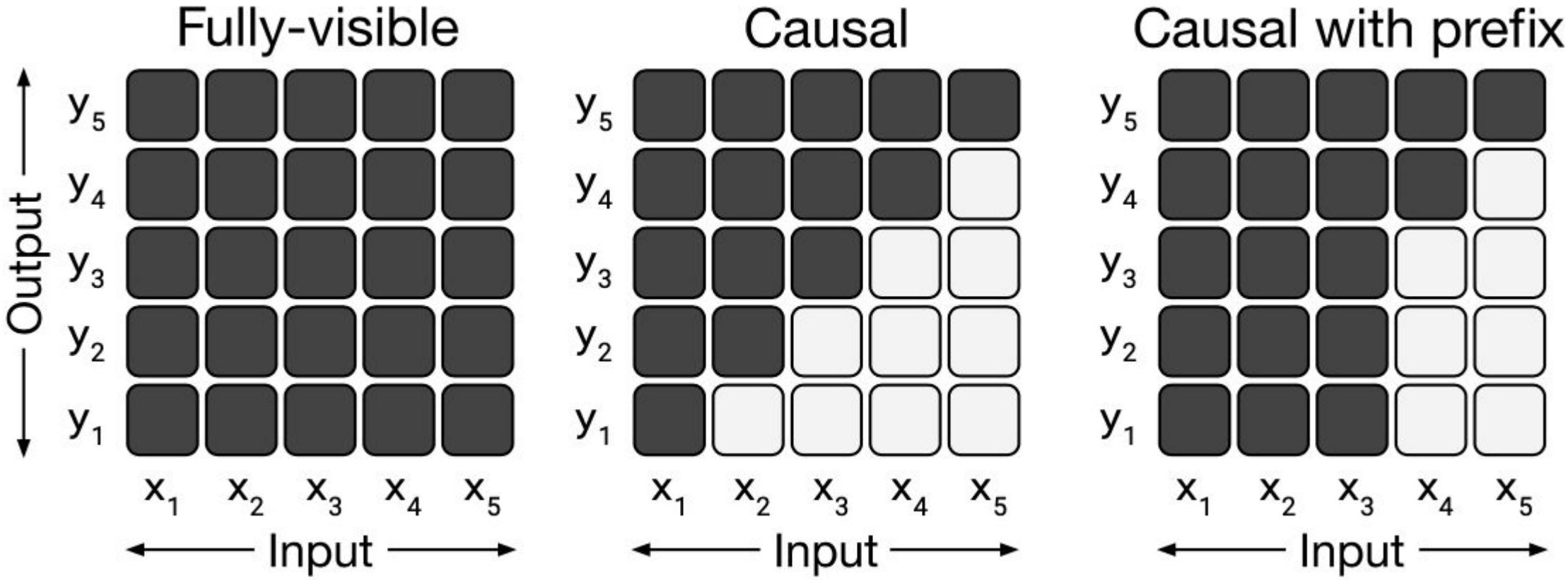
Can also be adapted to generate hidden states *by looking before token outputs*

Generation

Casual Masking

- **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



Casual Masking

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

Target:	S1b	EOT	S2b	EOT
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 corss-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

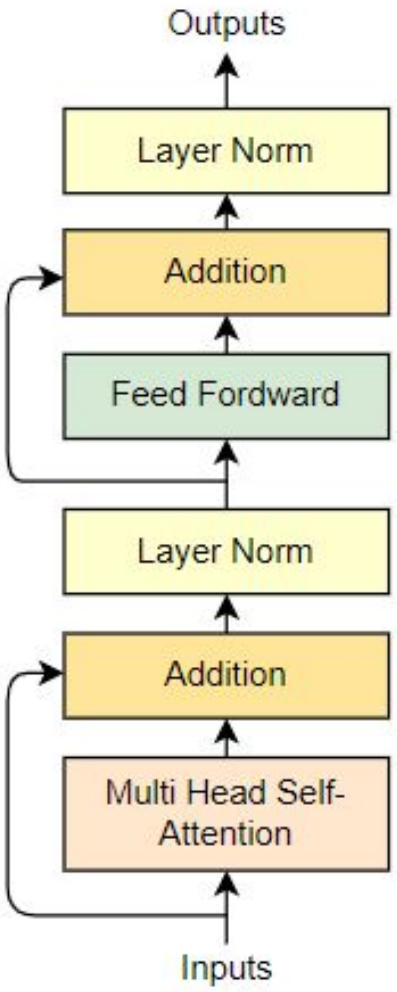
Table 1: Comparison of Pre-LayerNorm and Post-LayerNorm Architectures

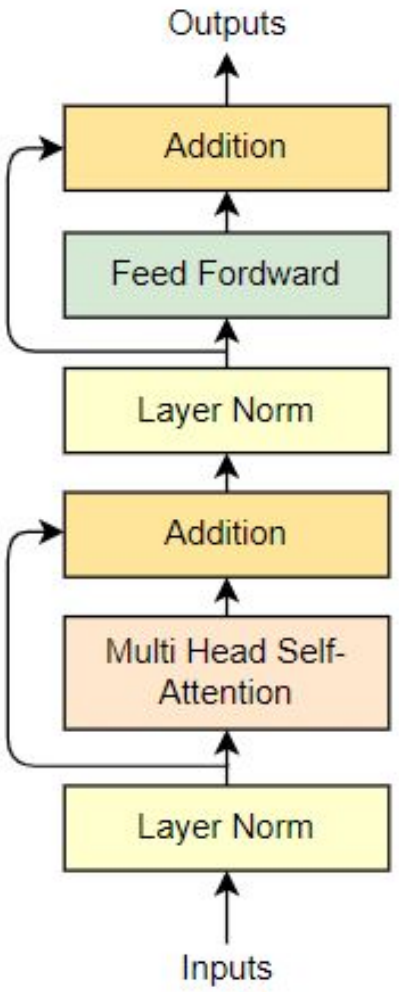
Pre-LN: LayerNorm applied *before* sublayers

Post-LN: LayerNorm applied *after* sublayers

Pre-LayerNorm vs. Post-LayerNorm

Post-LN

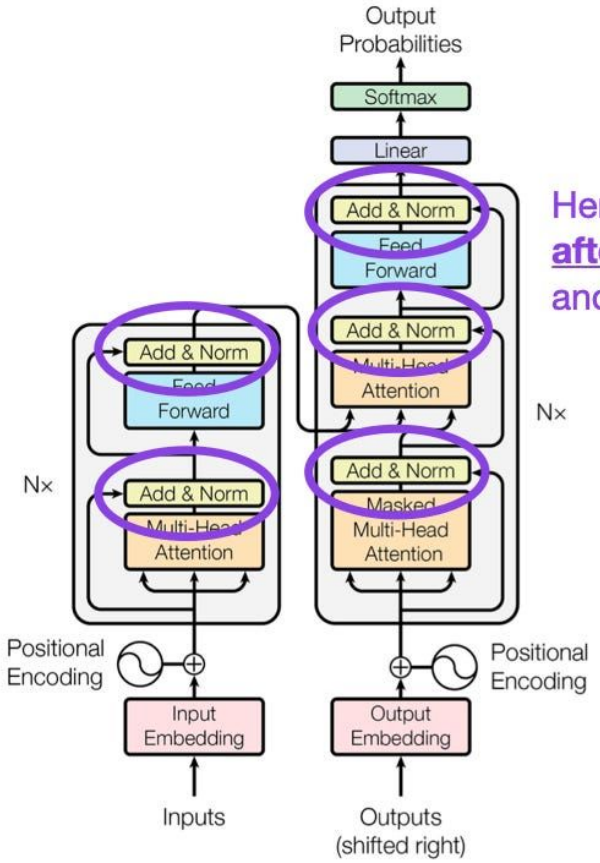




Pre-LN

Pre-LayerNorm vs. Post-LayerNorm

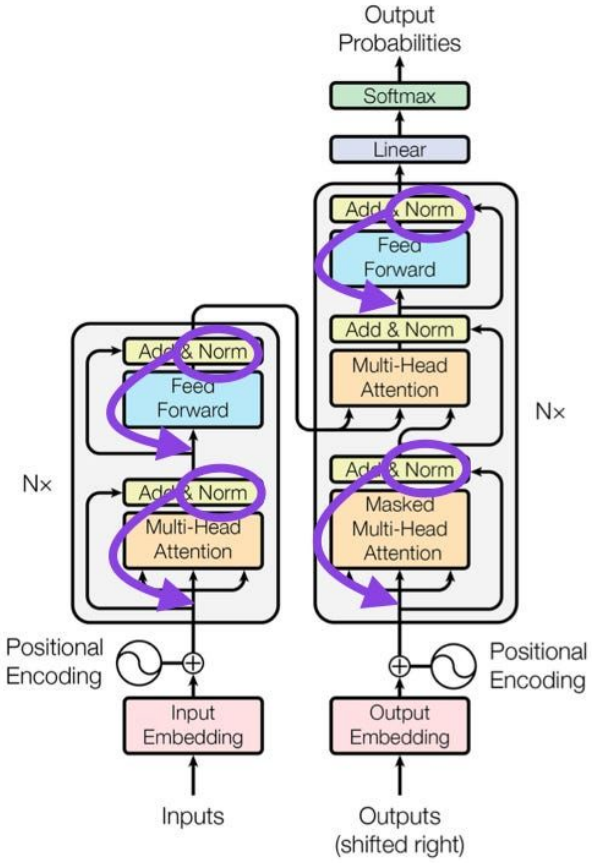
Post-LN Transformer

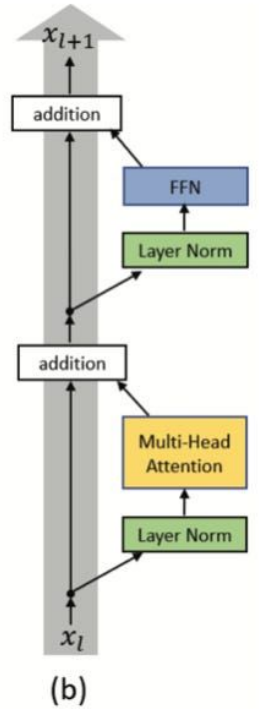


Here, LayerNorms are **after** attention and **after** fully connected layers

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

Pre-LN Transformer

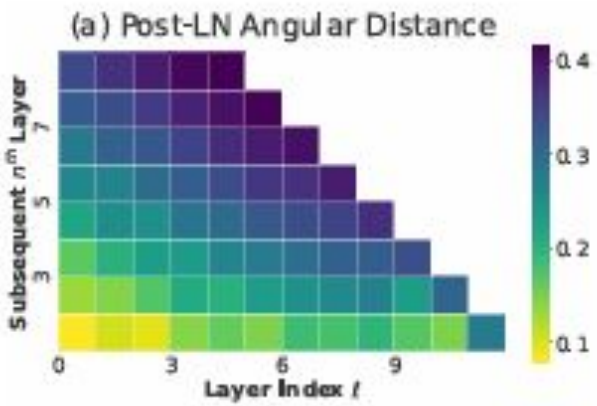


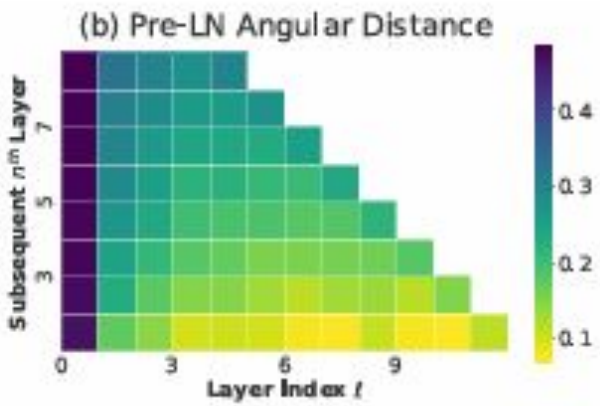


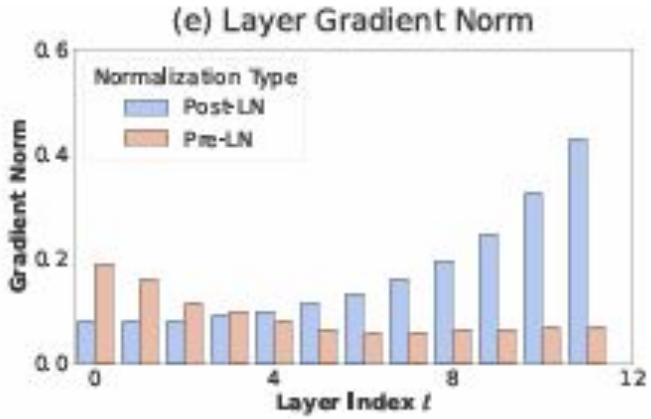
(b)

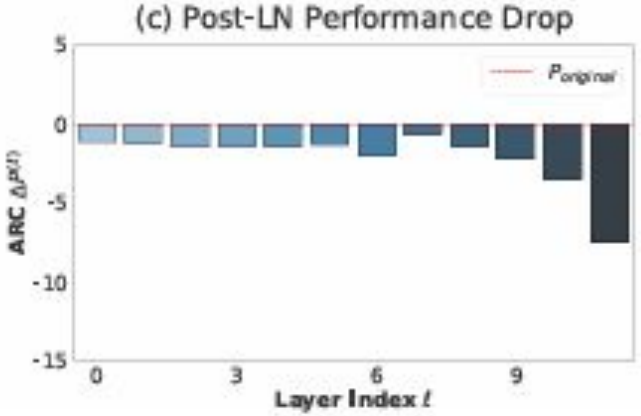
Places the layer normalization between the residual blocks: the expected gradients of the parameters near the output layer are large

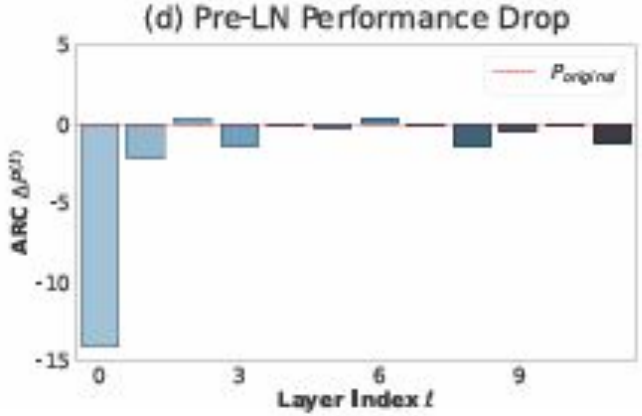
Pre-LayerNorm vs. Post-LayerNorm

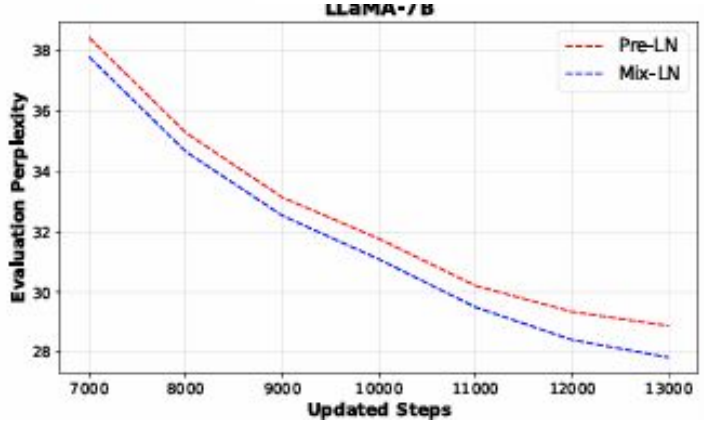












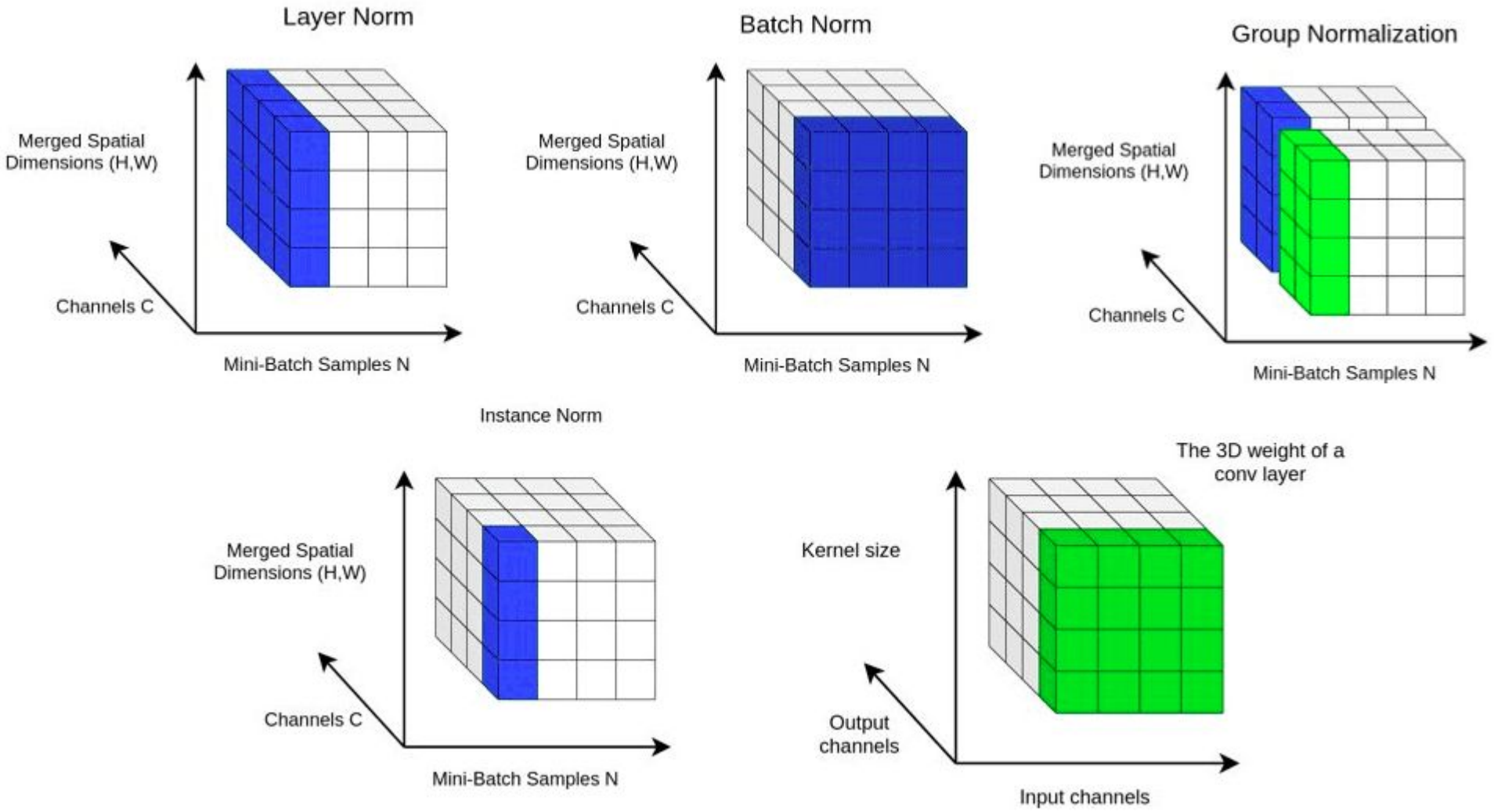
Layer Normalization vs RMSNorm

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

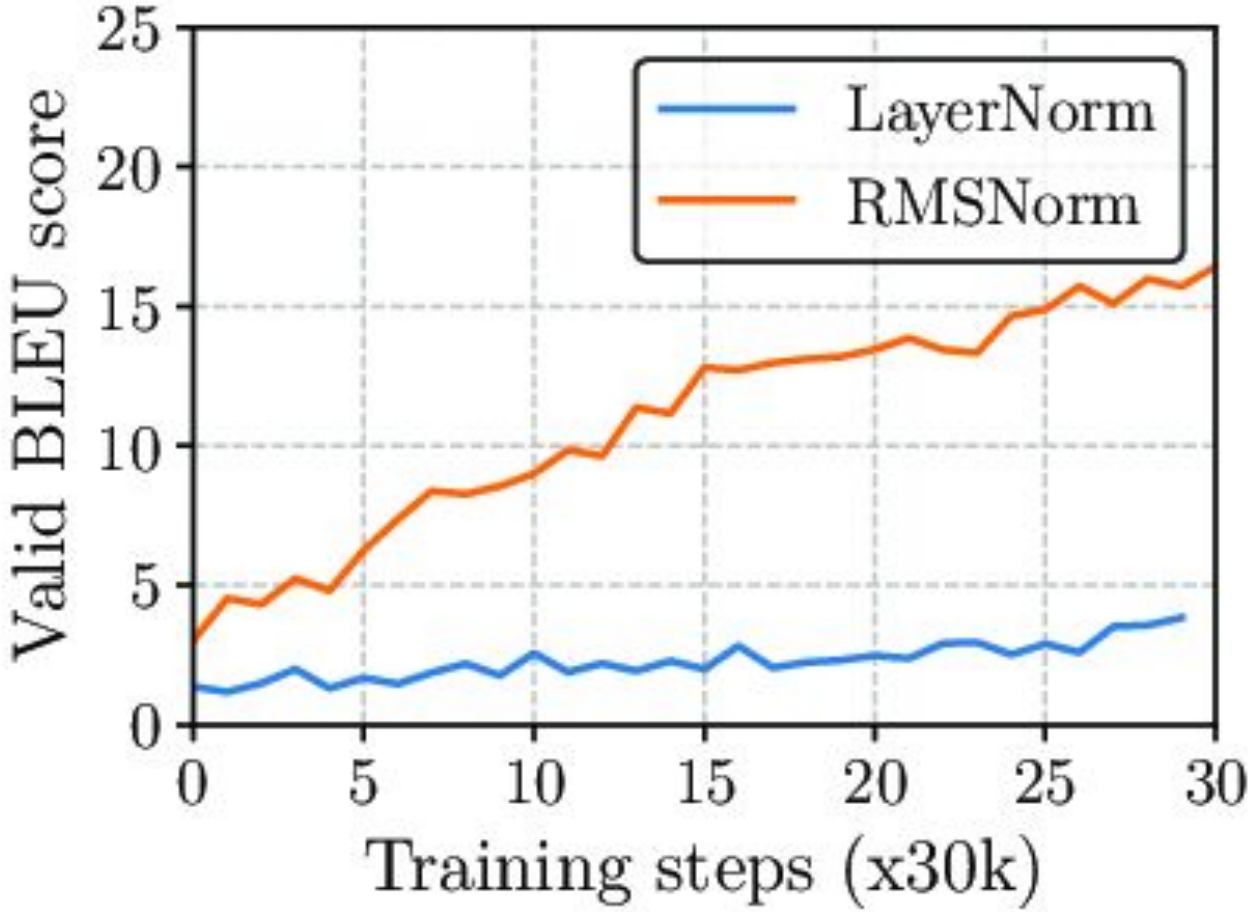
RMSNorm: Removes the mean and scales by the root mean square (RMS) of the features.

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

Layer Normalization vs RMSNorm



Layer Normalization vs RMSNorm



Transformer Variants

- **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:

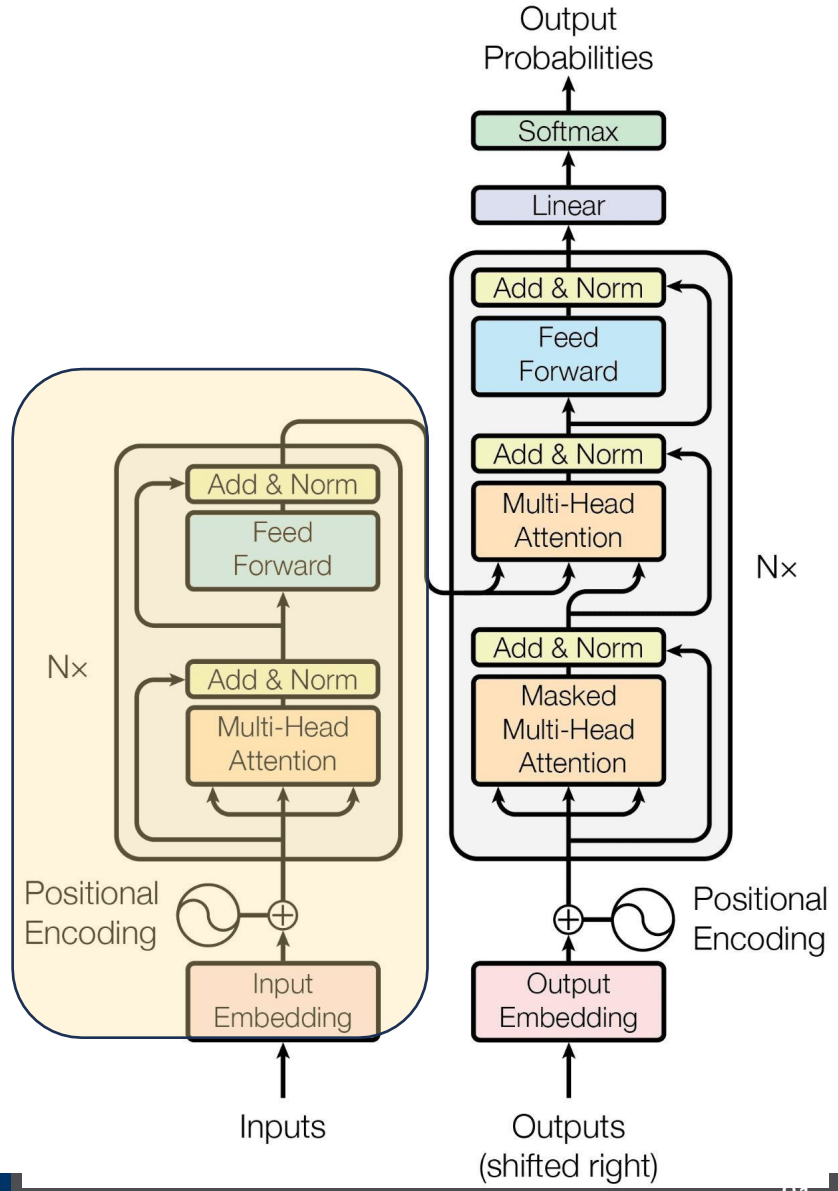
- English Wikipedia - 2,500 million words
- Book Corpus - 800 million words

BERT Pre-Training Tasks:

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

BERT Pre-Training Results:

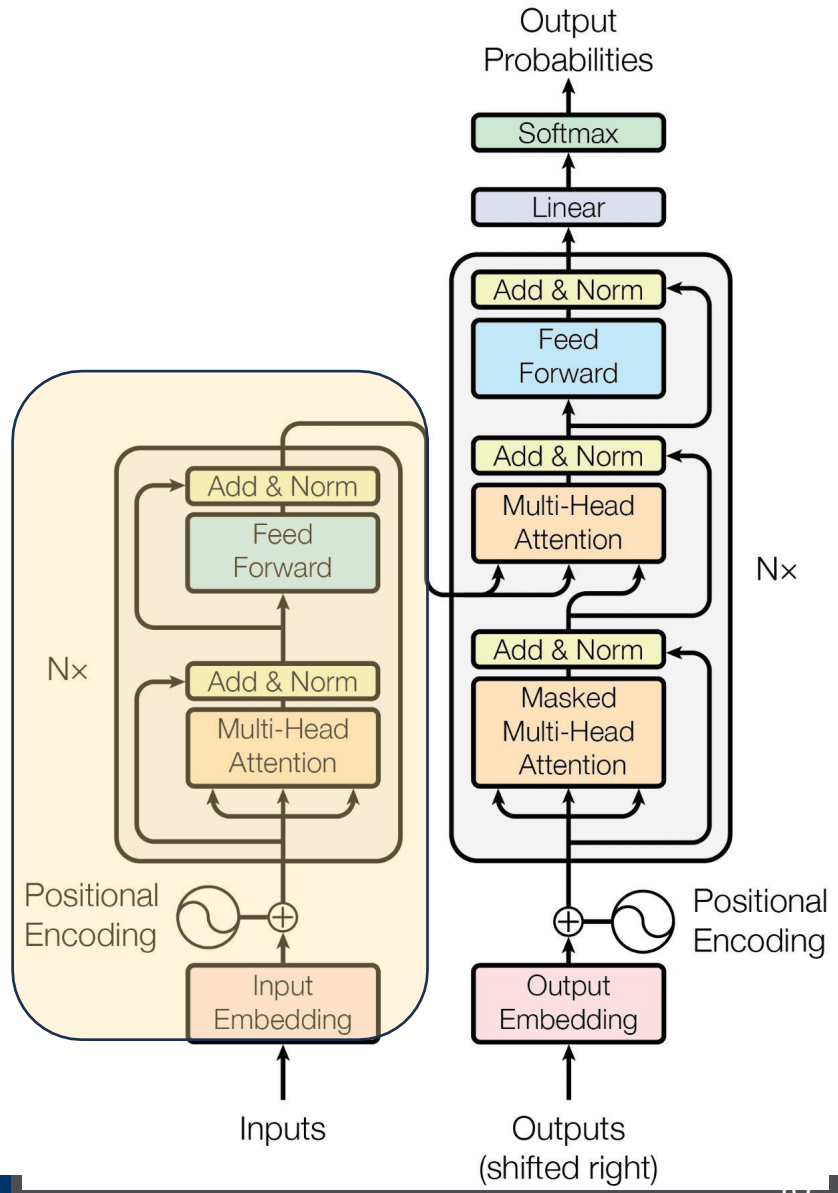
- BERT-Base – 110M Params
- BERT-Large – 340M Params



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:

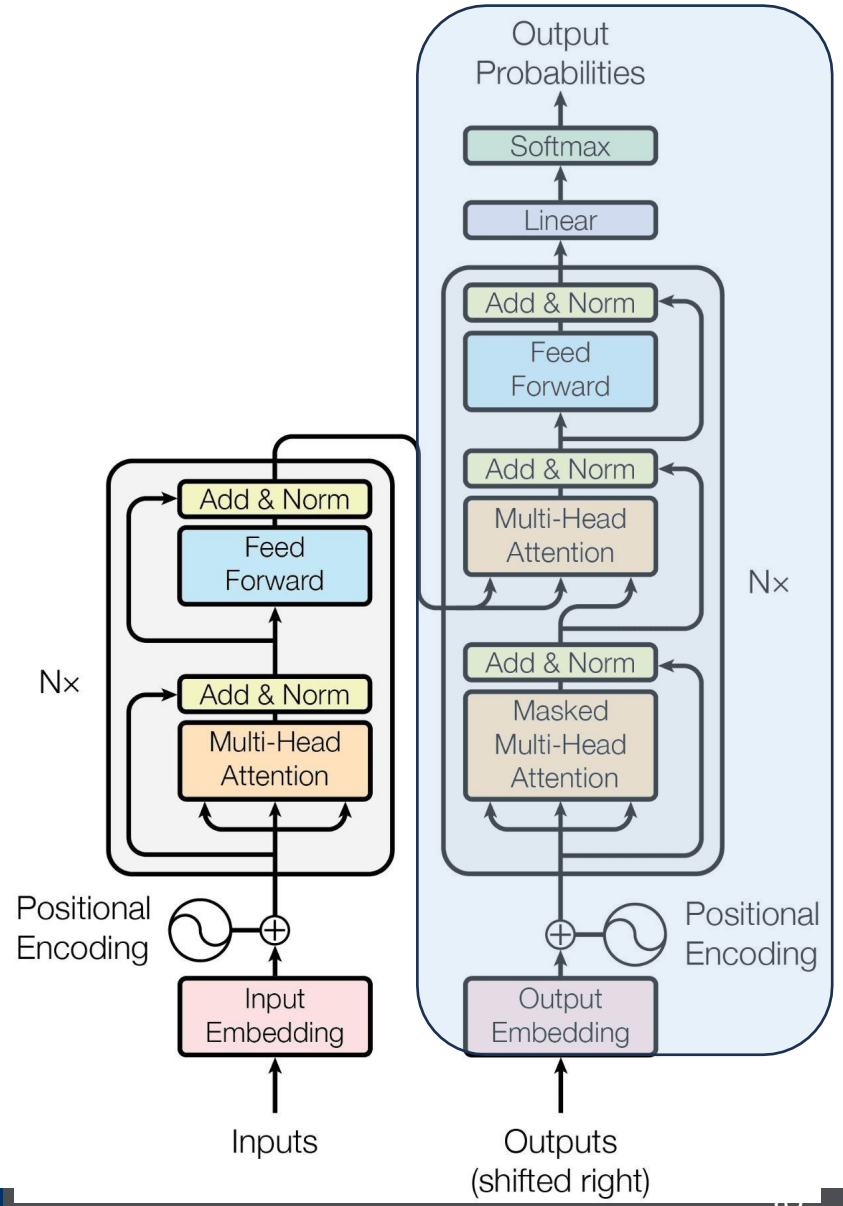
- Similarly, BooksCorpus and English Wikipedia

GPT Pre-Training Tasks:

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

GPT Pre-Training Results:

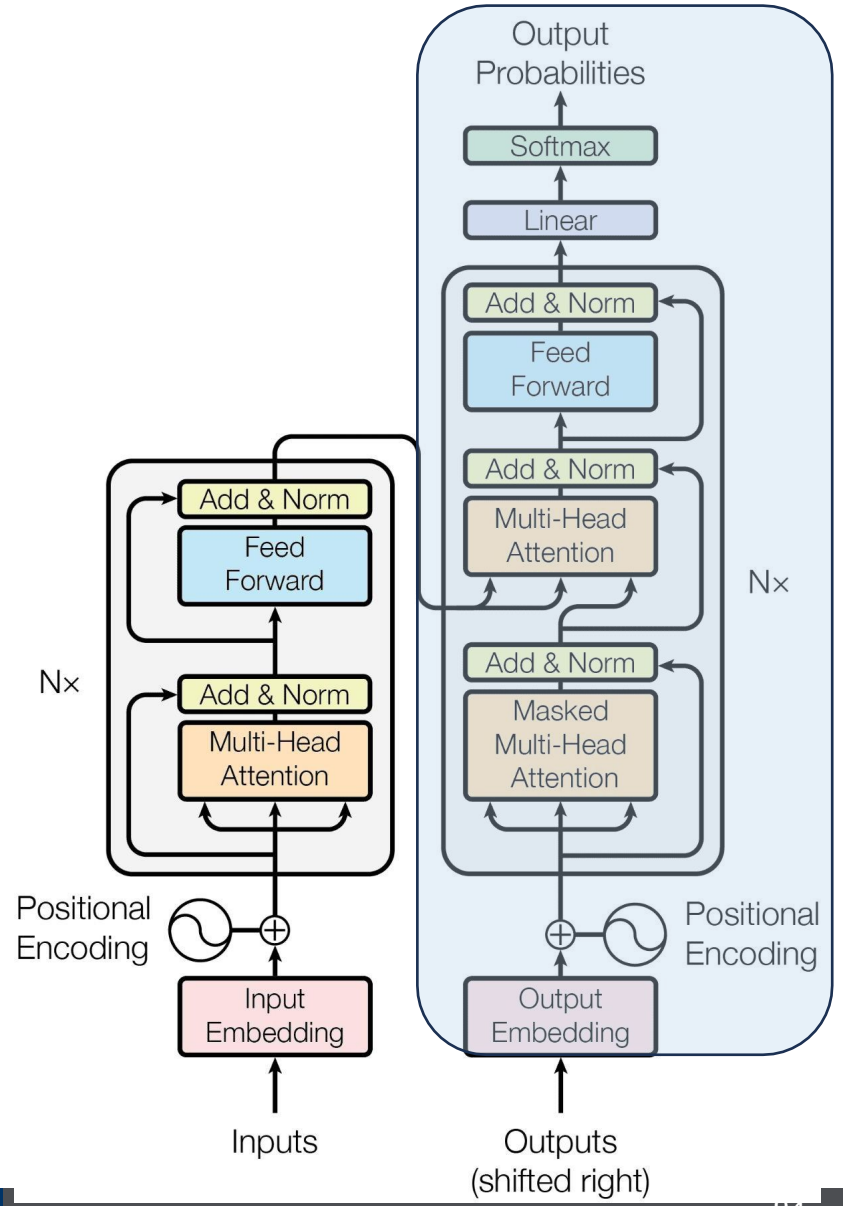
- GPT – 117M Params
  - Similarly competitive on GLUE and SQuAD



GPT – Generative 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!!!**



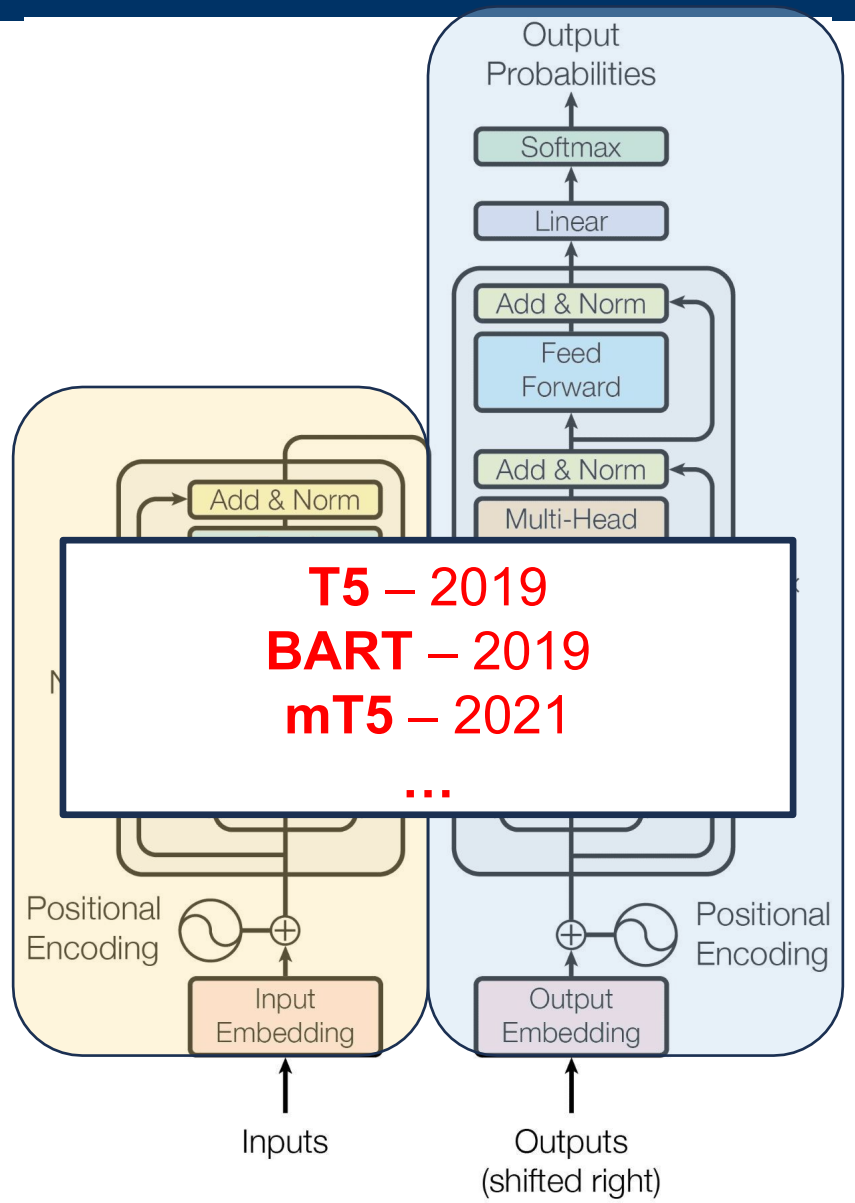
The LLM Era – Paradigm Shift in Machine Learning

أكاديمية كاوست
KAUST ACADEMY

LMH
Lady Margaret Hall

BERT – 2018
DistilBERT – 2019
RoBERTa – 2019
ALBERT – 2019
ELECTRA – 2020
DeBERTa – 2020
…

Representation



T5 – 2019
BART – 2019
mT5 – 2021
…

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

Future Directions

- 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.

References

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.

References

[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

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