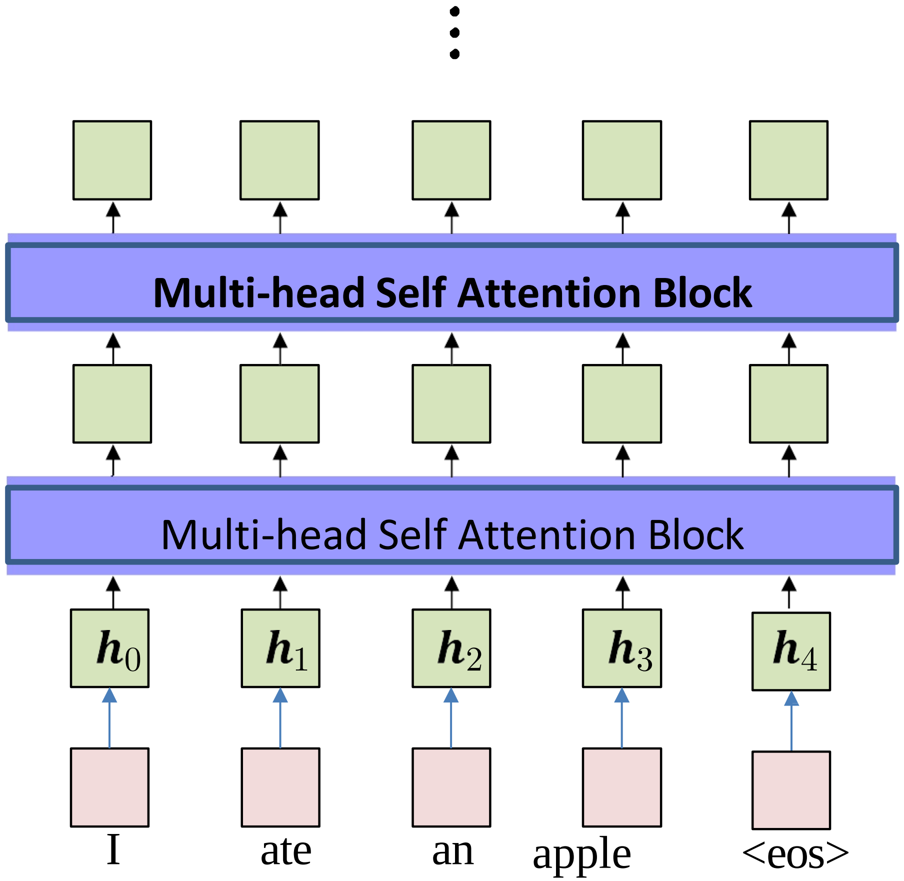
Transformer

CSE 849 Deep Learning
Spring 2025

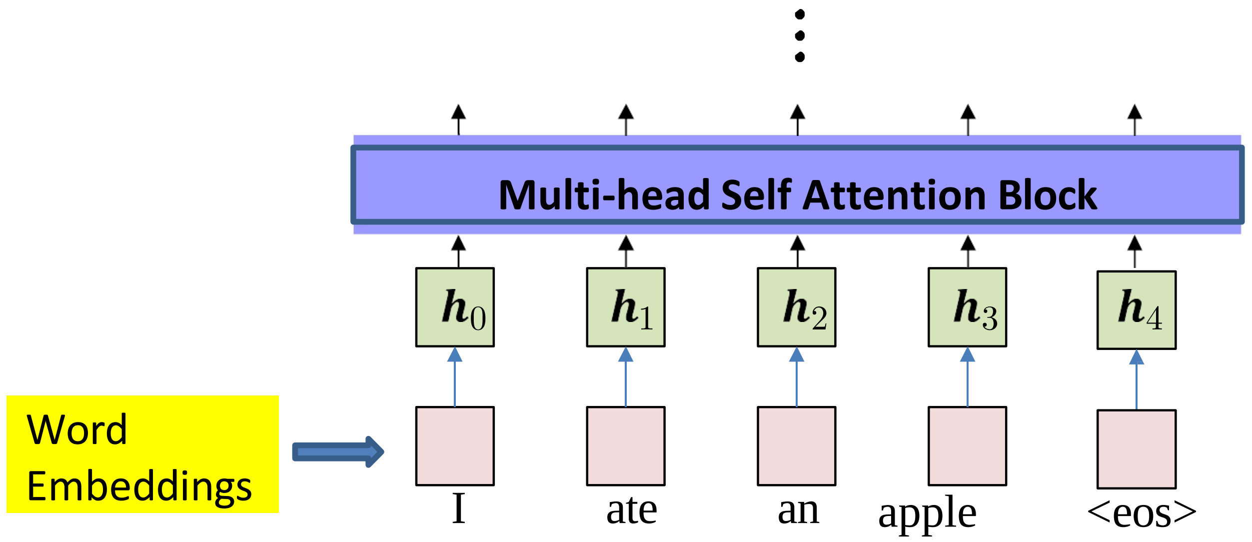
Zijun Cui

Partial slides are adapted from CMU 11-785 Introduction to Deep Learning by Bhiksha Raj

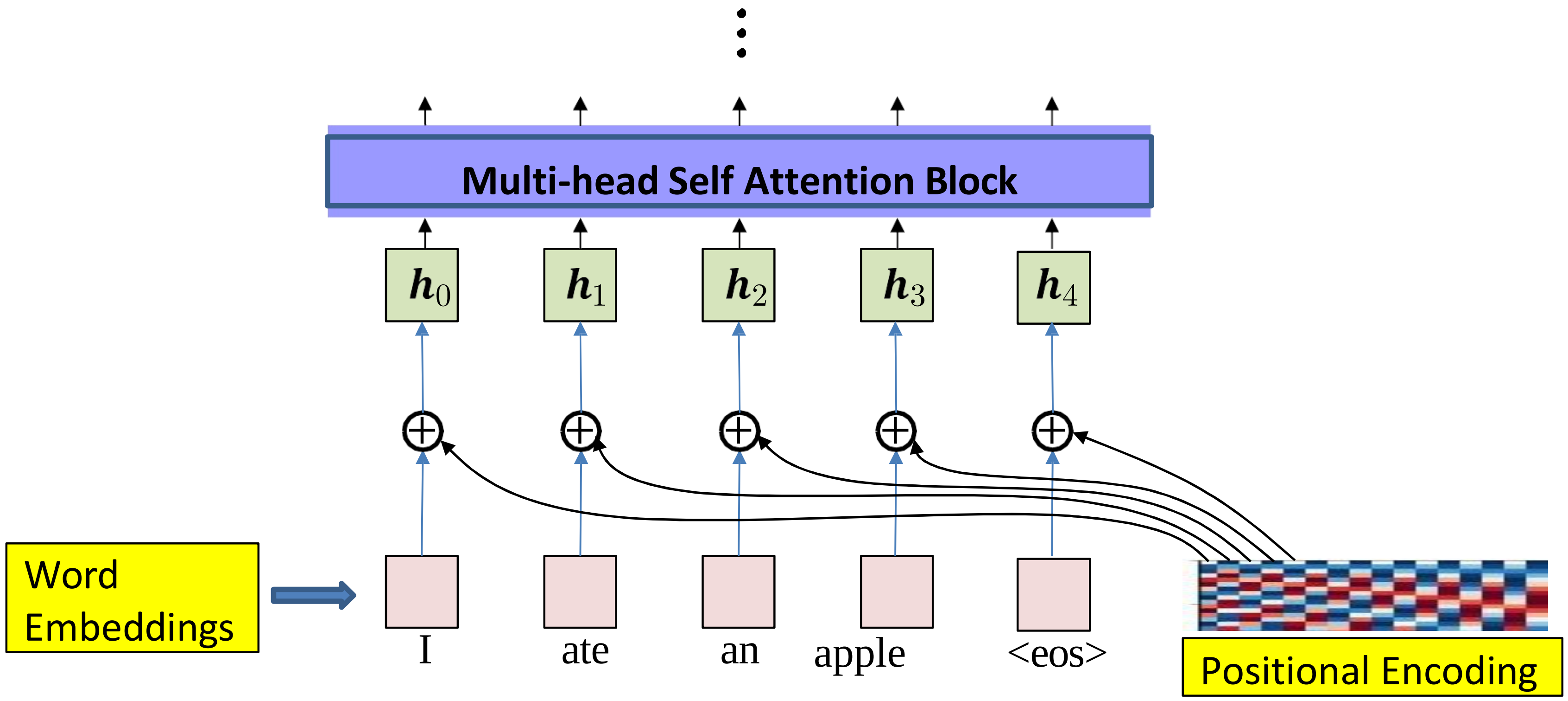
Continuing from the last lecture…



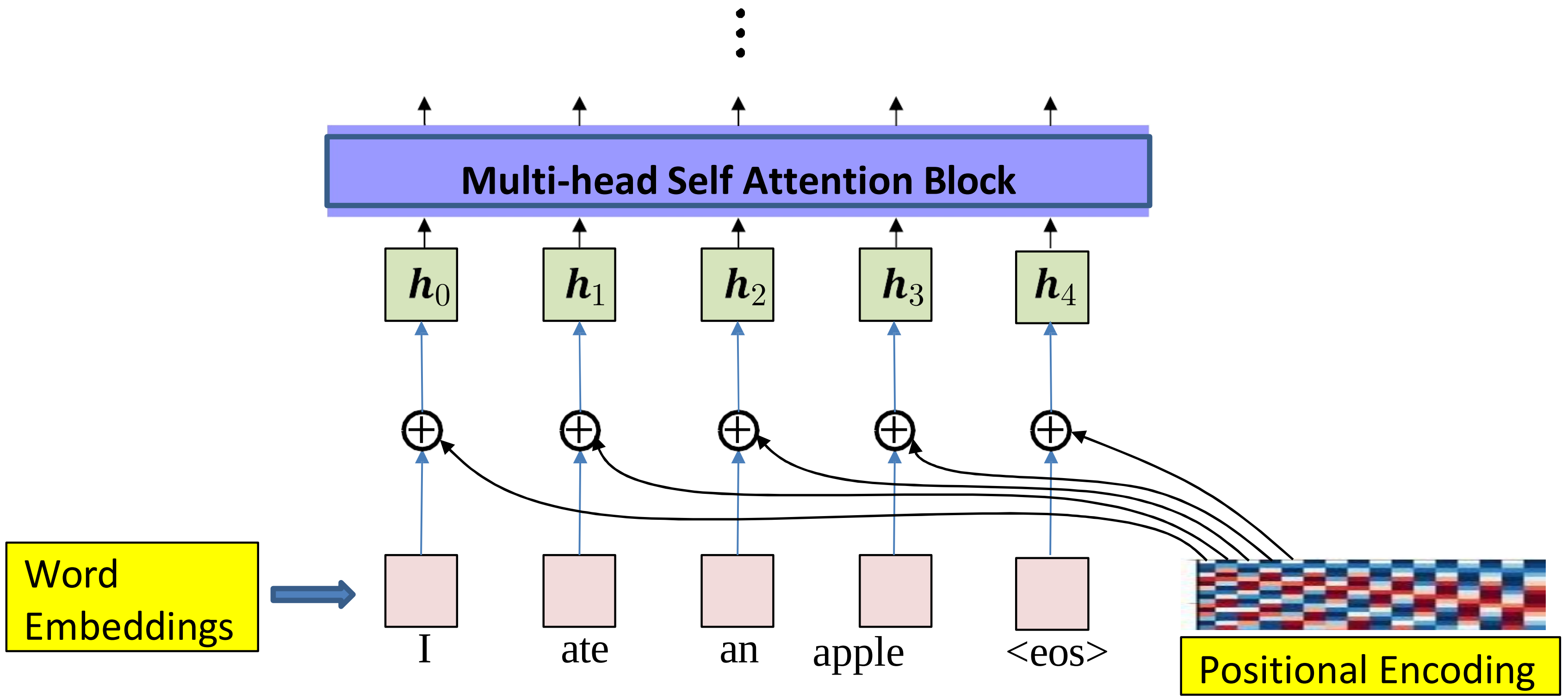
- The encoder in a sequence-to-sequence model can replace recurrence through a series of "multi-head self attention" blocks
- But this still ignores *relative position*
  - A context word one word away is different from one 10 words away
  - The attention framework does not take distance into consideration



- Note that the inputs are actually word *embeddings*



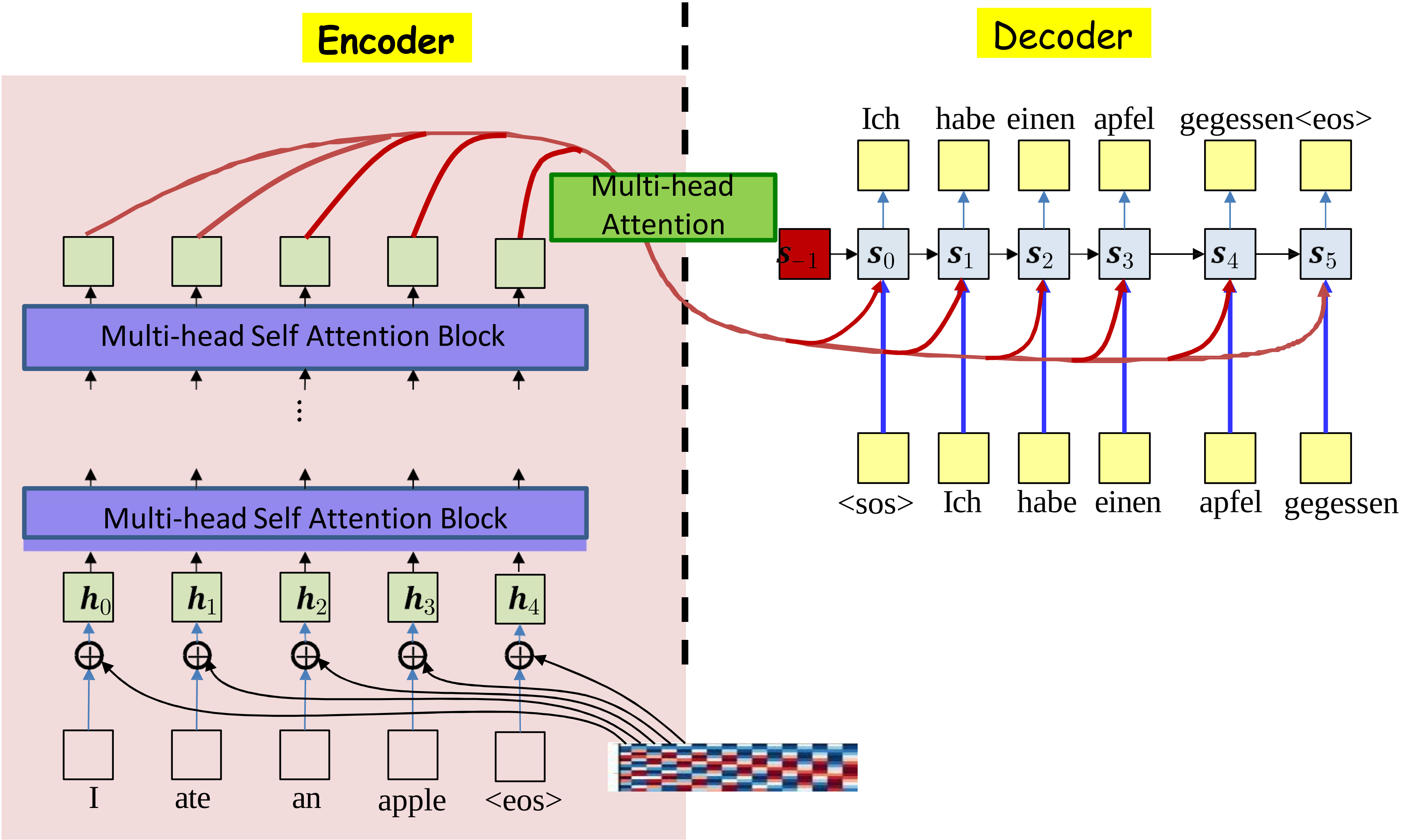
- We add a "positional" encoding to them to capture the relative distance from one another



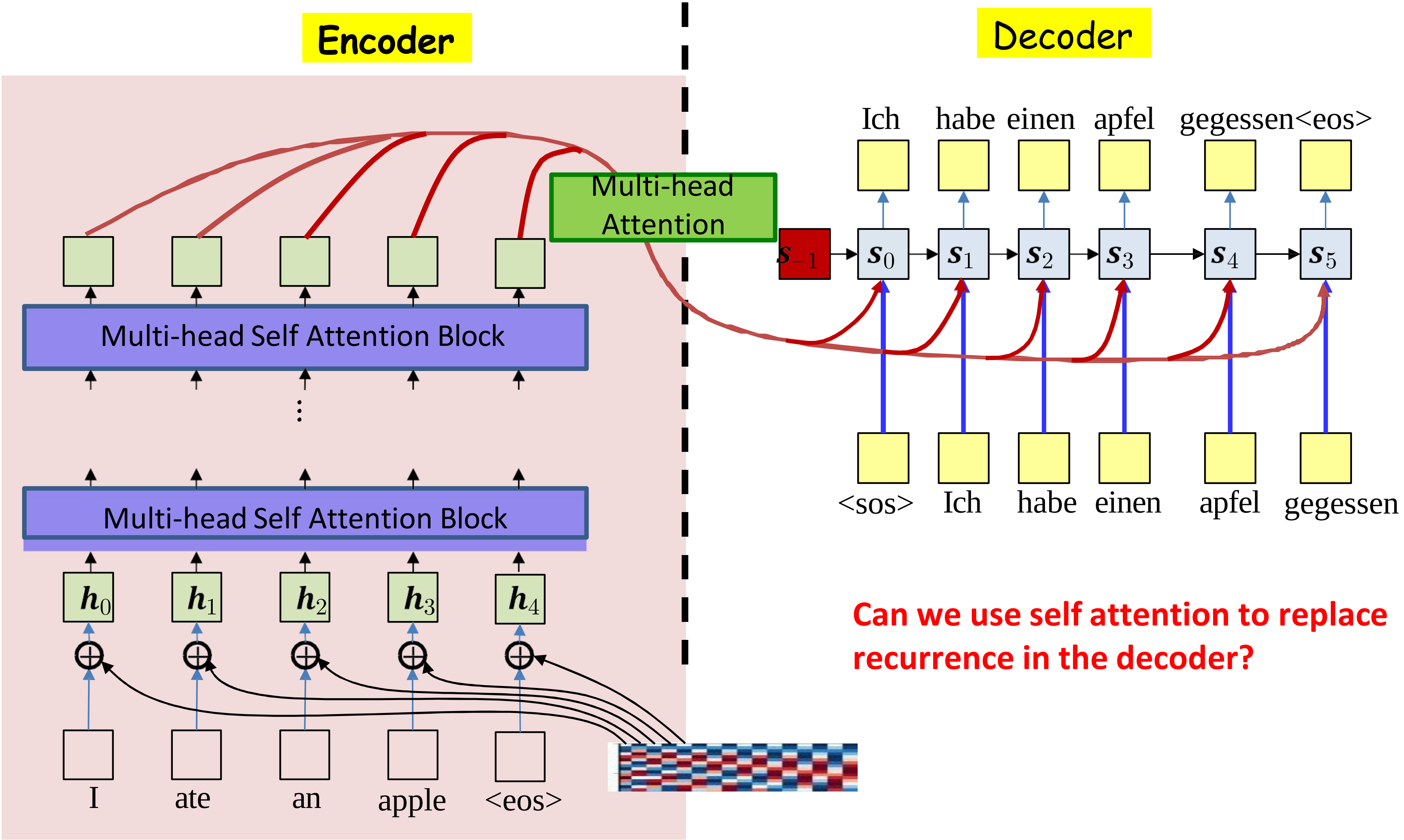
- **Positional Encoding:** A sequence of vectors $P_0, \ldots, P_N$, to encode position
  - Every vector is unique (and uniquely represents time)
  - Relationship between P_t and P_{t+c} only depends on the distance between them

$$P_{t+c} = M_c P_t$$

- The linear relationship between P_t and P_{t+c} enables the net to learn shift-invariant "gap" dependent relationships



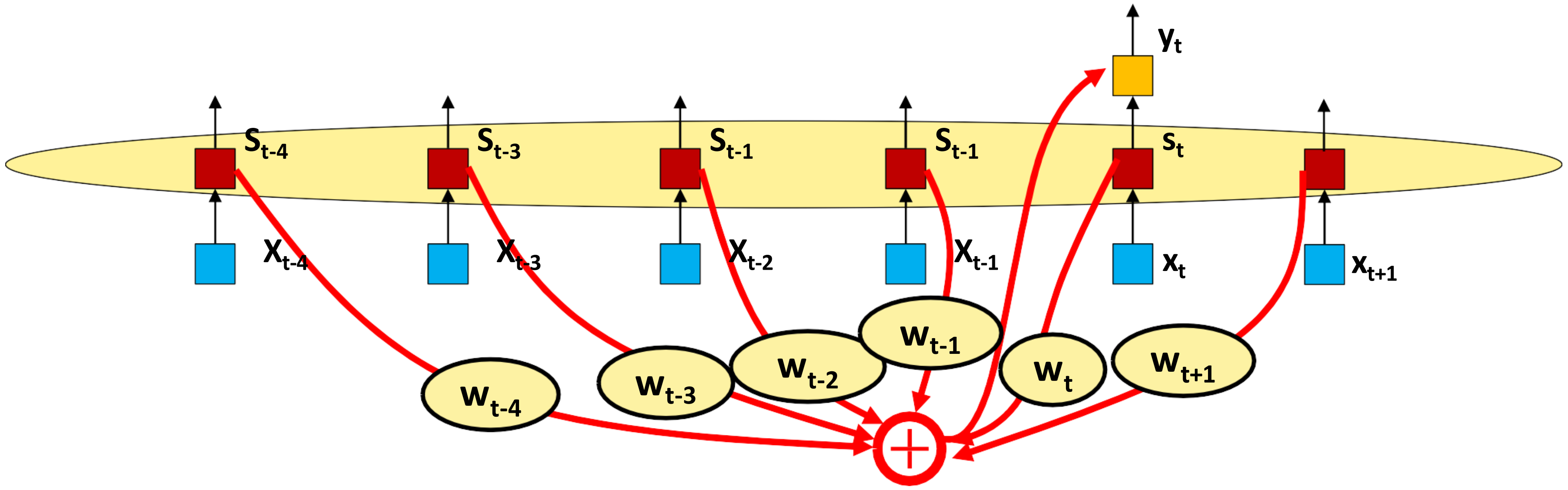
- The self-attending encoder



Can we use self attention to replace recurrence in the decoder?

- The self-attending encoder

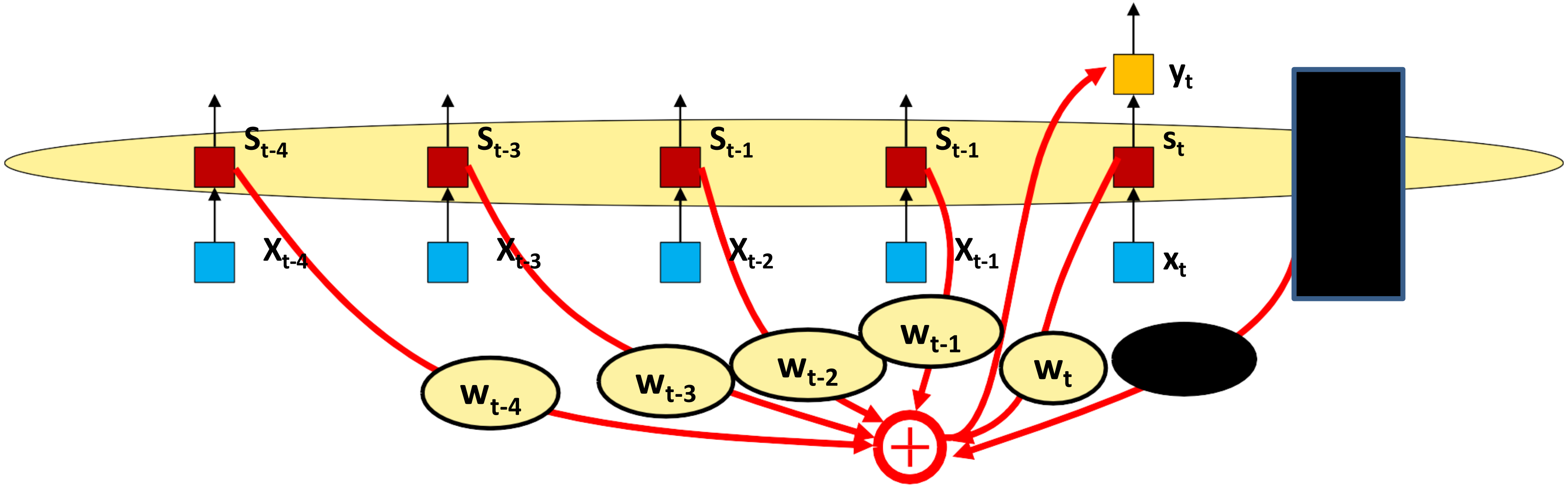
Self attention and masked self attention



- **Self attention in encoder:** Can use input embedding at time t+1 and further to compute output at time t, because all inputs are available

Self attention and masked self attention

- **Self attention in decoder:** Decoder is sequential
  - Each word is produced using the previous word as input
  - Only embeddings until time t are available to compute the output at time t
- The attention will have to be "masked", forcing attention weights for t+1 and later to 0



Masked self-attention block

$$q_i = W_q h_i$$

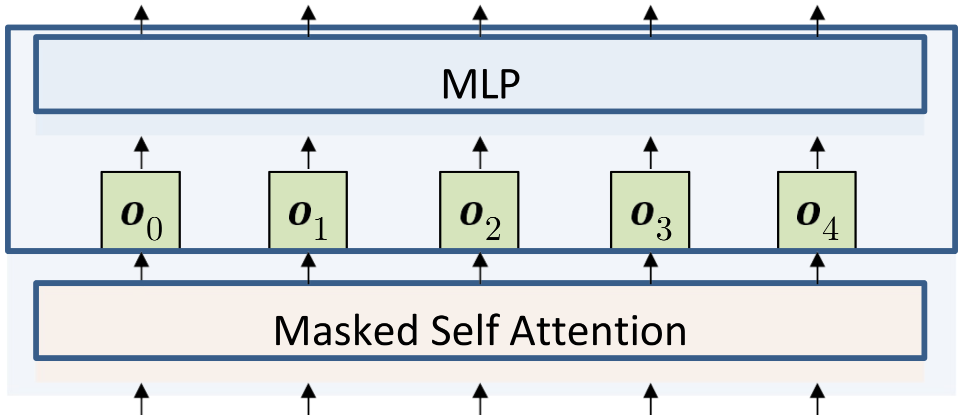
$$k_i = W_k h_i$$

$$v_i = W_v h_i$$

$$e_{ij} = q_i^T k_j$$

$$w_{i0}, \dots, w_{ii} = softmax(e_{i0}, \dots, e_{ii})$$

$$o_i = \sum_{j=0}^{i-1} w_{ij} v_j$$



- The "masked self attention ***block***" includes an MLP after the masked self attention
  - Like in the encoder

Masked self-attention block

$q_i = \boldsymbol{W}_q h_i$

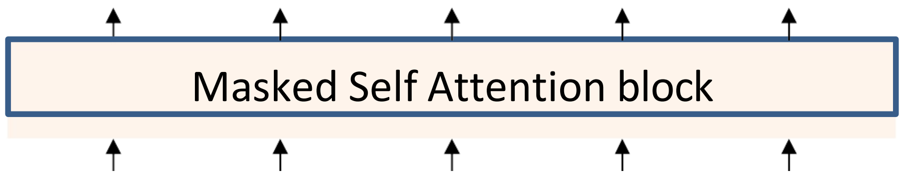
$k_i = \boldsymbol{W}_k h_i$

$v_i = \boldsymbol{W}_v h_i$

$e_{ij} = q_i^T k_j$

$w_{i0}, \dots, w_{ii} = softmax(e_{i0}, \dots, e_{ii})$

$$o_i = \sum_{j=0}^{i-1} w_{ij} v_j$$



- The "masked self attention ***block***" sequentially computes outputs begin to end
  - Sequential nature of decoding prevents outputs from being computed in parallel
  - Unlike in an encoder

Masked multi-head self-attention block

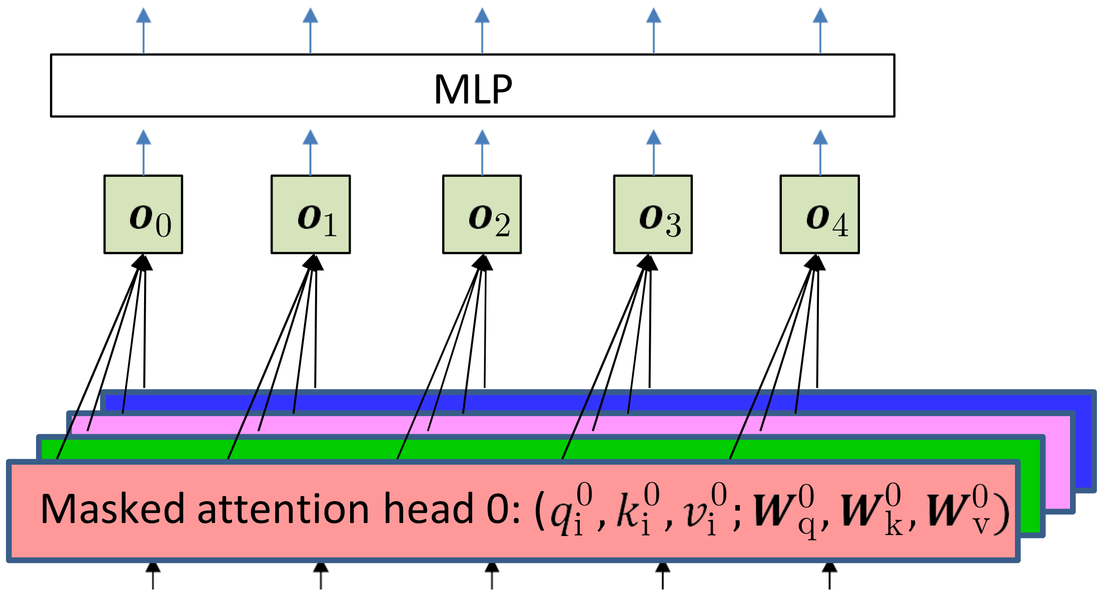
$$q_i^a = \boldsymbol{W}_q^a h_i$$

$$k_i^a = \boldsymbol{W}_k^a h_i$$

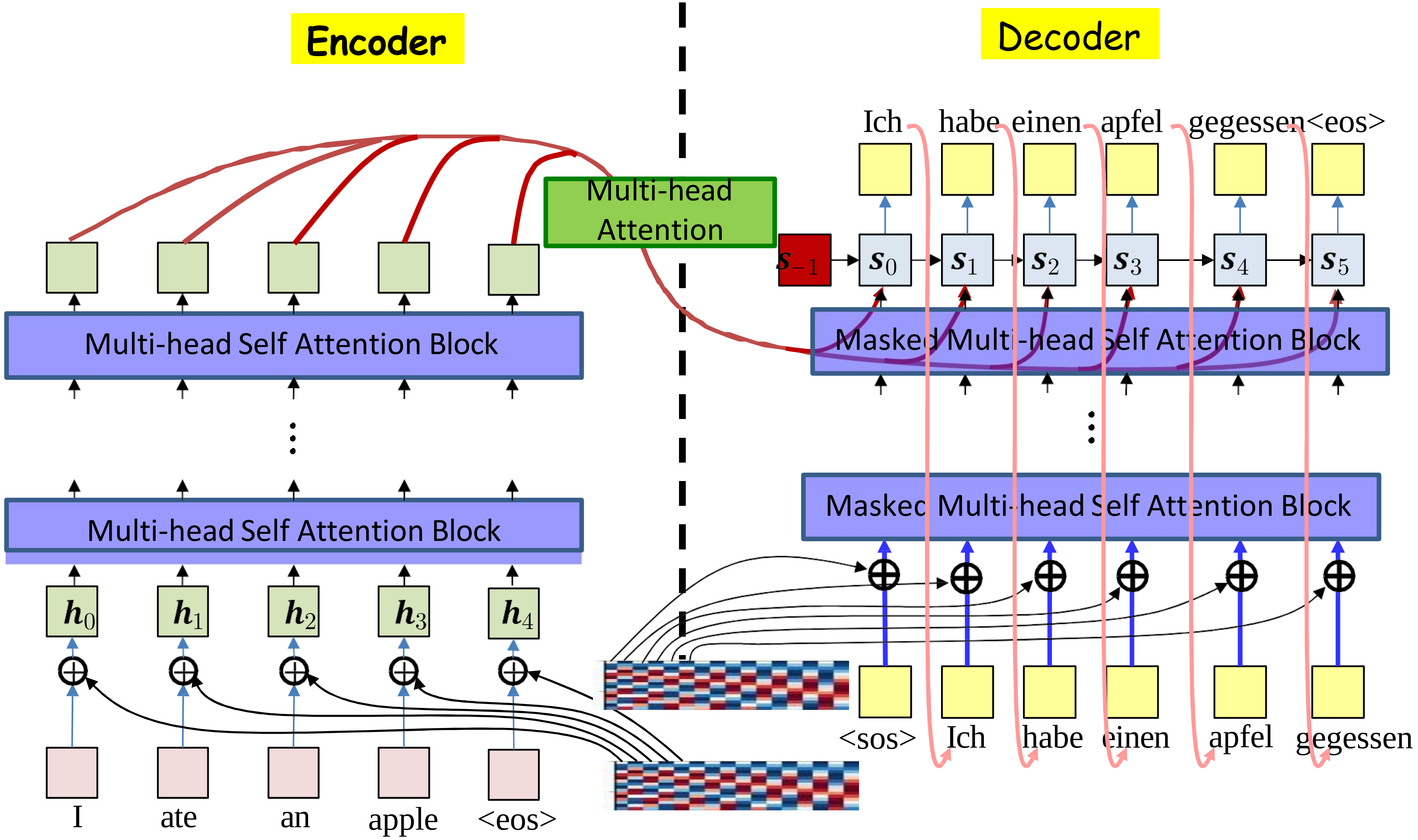
$$v_i^a = \boldsymbol{W}_v^a h_i$$

$$w_{ij}^a = attn(q_i^a, k_{0:i-1}^a)$$

$$o_i^a = \sum_j w_{ij}^a v_j^a$$

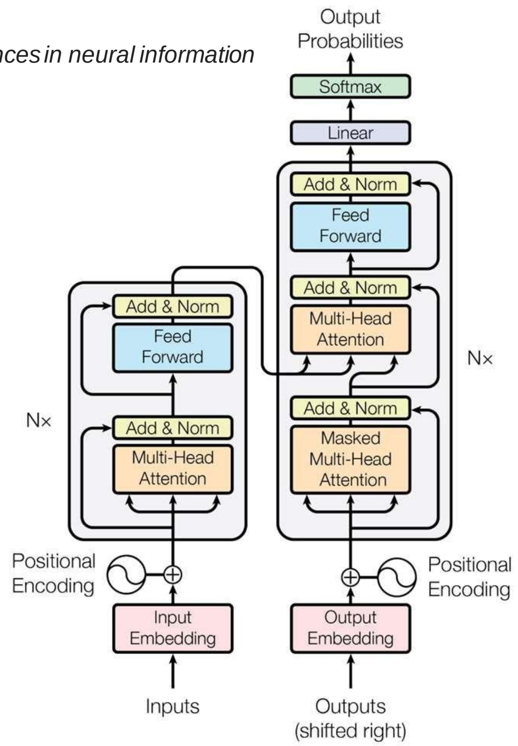


- The "masked ***multi-head*** self attention ***block***" includes multiple masked attention heads
  - Like in the encoder


Encoder
Decoder
Multi-head Attention
Multi-head Self Attention Block
Multi-head Self Attention Block
h_0 h_1 h_2 h_3 h_4
I ate an apple <eos>
Ich habe einen apfel gegessen<eos>
s_{-1} s_0 s_1 s_2 s_3 s_4 s_5
Masked Multi-head Self Attention Block
Masked Multi-head Self Attention Block
<sos> Ich habe einen apfel gegessen

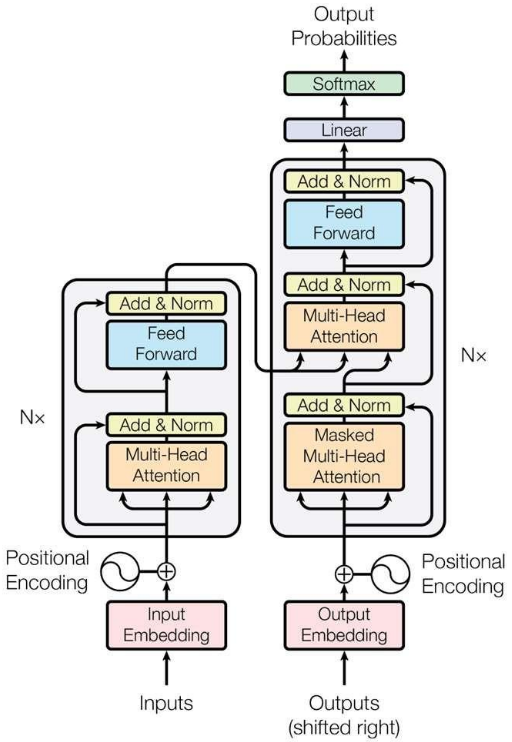
Transformer: Attention is all you need

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.



- Transformer: A sequence-to-sequence model that replaces recurrence with positional encoding and multi-head self attention
  - "Attention is all you need"

Transformer



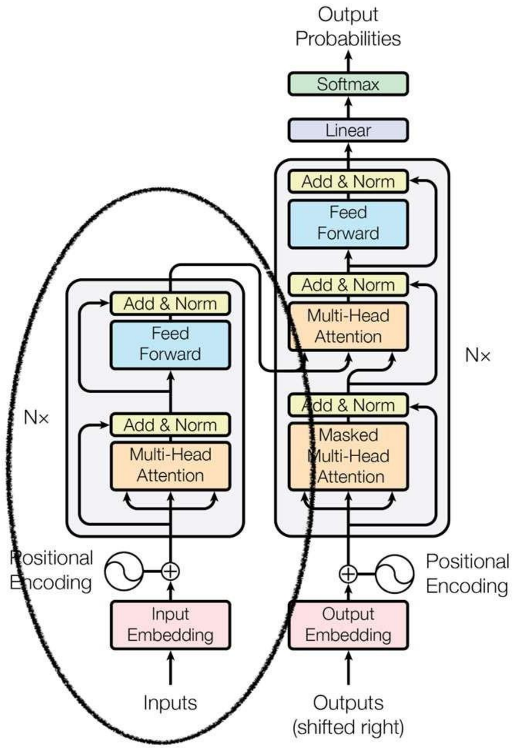
From "Attention is all you need"

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

- Transformer: tremendous decrease in model computation for similar performance as state-of-art translation models
- The last row in the table shows transformer performance
- The final two columns show computational cost.

BERT

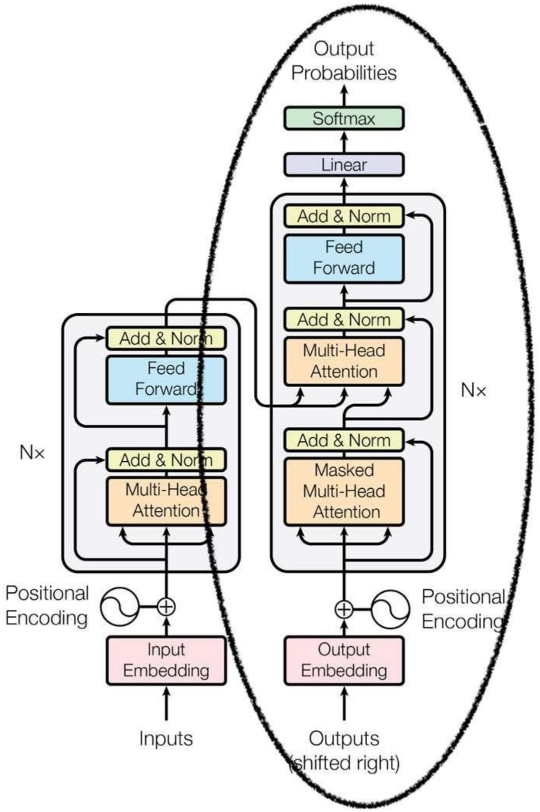


System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

- Bert: Only uses encoder of transformer to derive word and sentence embeddings
- Trained to "fill in the blanks"
- This is *representation learning*

GPT



Alec Radford et. al., Improving Language Understanding by Generative Pre-Training

Table 5: Analysis of various model ablations on different tasks. Avg. score is a unweighted average of all the results. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)	QNLI (acc)	RTE (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**	**88.1**	**56.0**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7	71.2	53.8
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1	86.9	54.4
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7	81.1	54.6

- GPT uses only the decoder of the transformer as an LM
  - “Transformer w/o aux LM”
- Large performance improvement in many tasks

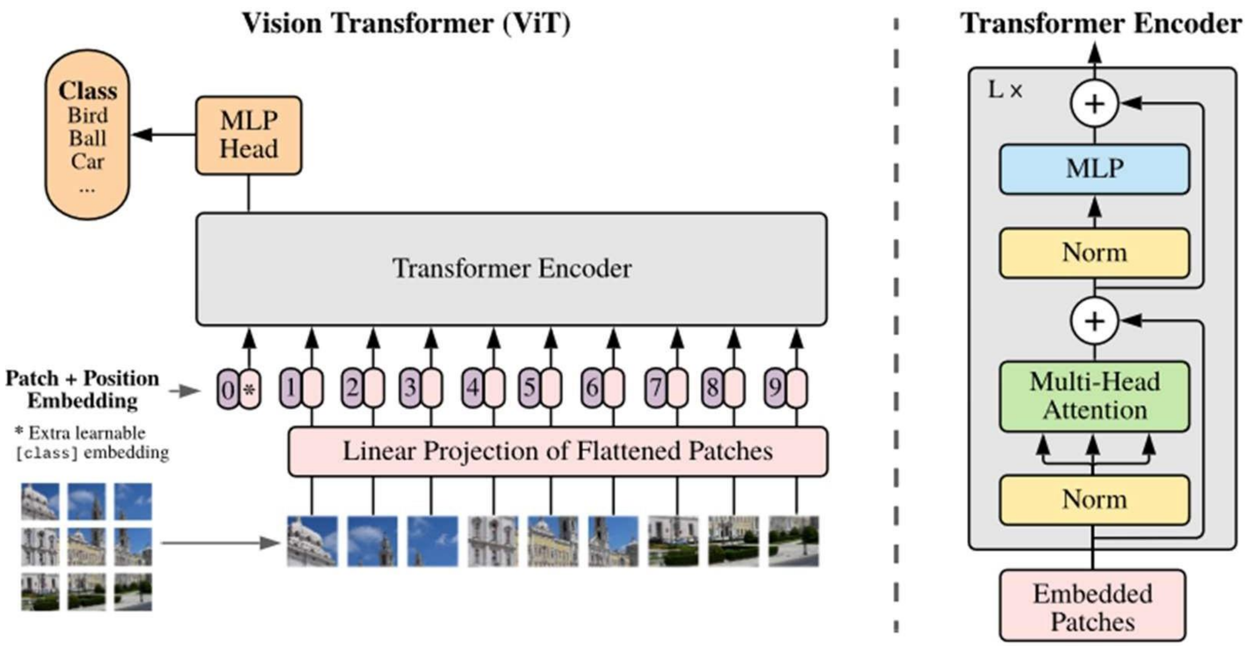
Attention is all you need

- Self-attention can effectively replace recurrence in sequence-to-sequence models
  - "Transformers"
  - Requires "positional encoding" to capture positional information

- Can also be used in regular sequence analysis settings as a substitute for recurrence

- Currently *the* state of the art in most sequence analysis/prediction… and even computer vison problems!

Vision Transformers

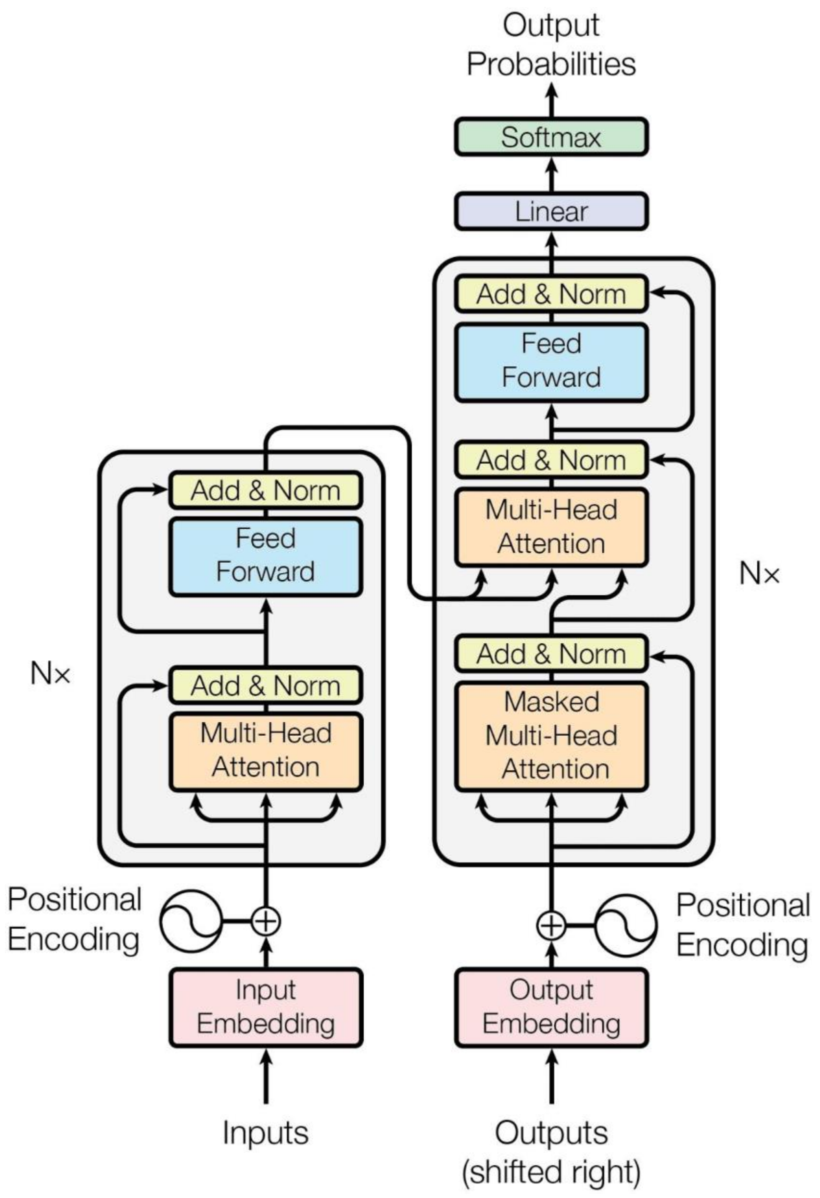


Dosovitskiy et al*, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, 2020

- Divide your image in patches with pos. encodings
- Apply Self-Attention!
- Sequential and image problems are similar when using transformers

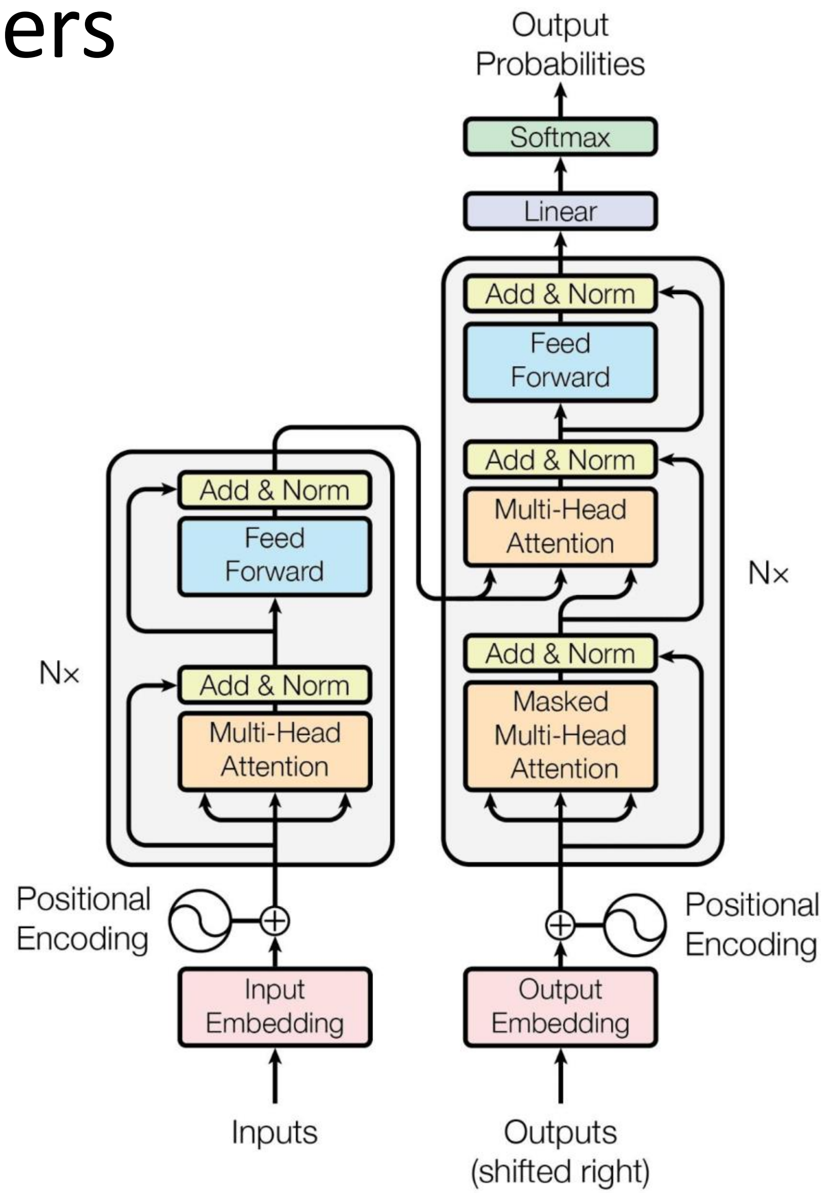
Table of contents

Transformer Architecture



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



Machine Translation

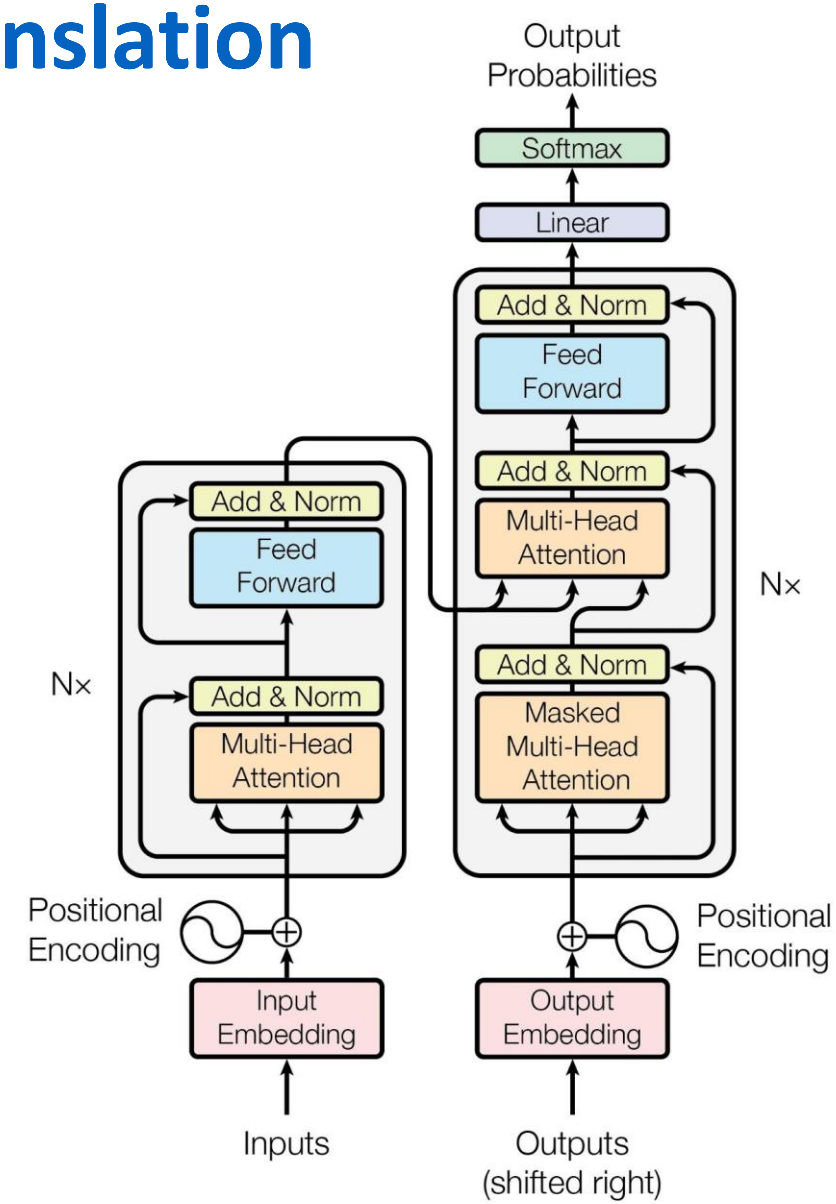
Targets

Ich habe einen Apfel gegessen

⬆

Inputs

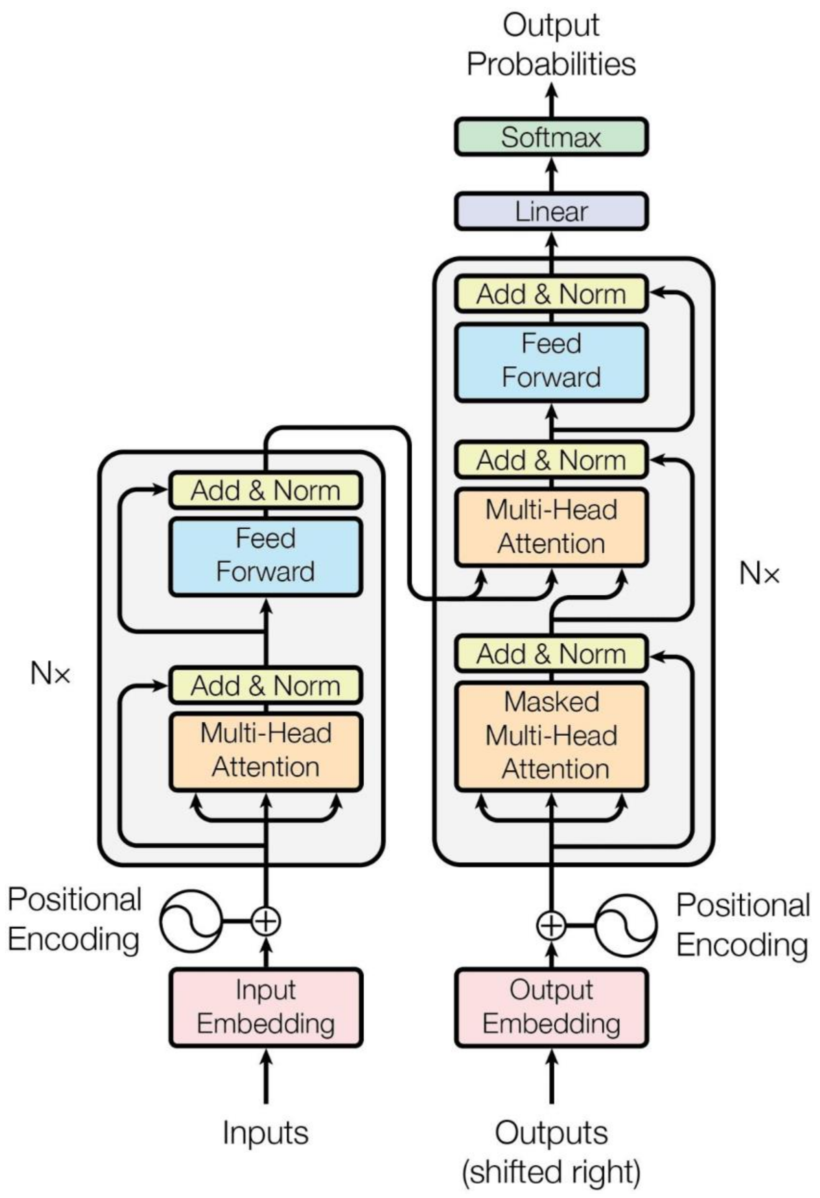
I ate an apple



Inputs

Processing Inputs

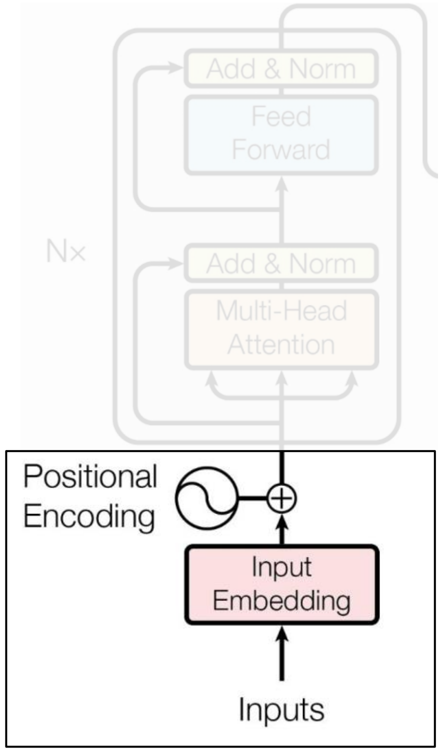
Inputs
I ate an apple



Tokenization

Tokenizer (split into individual words)

I ate an apple

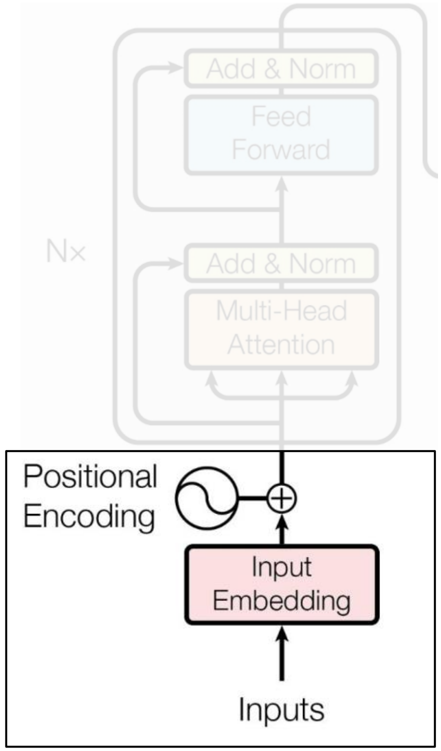


Tokenization

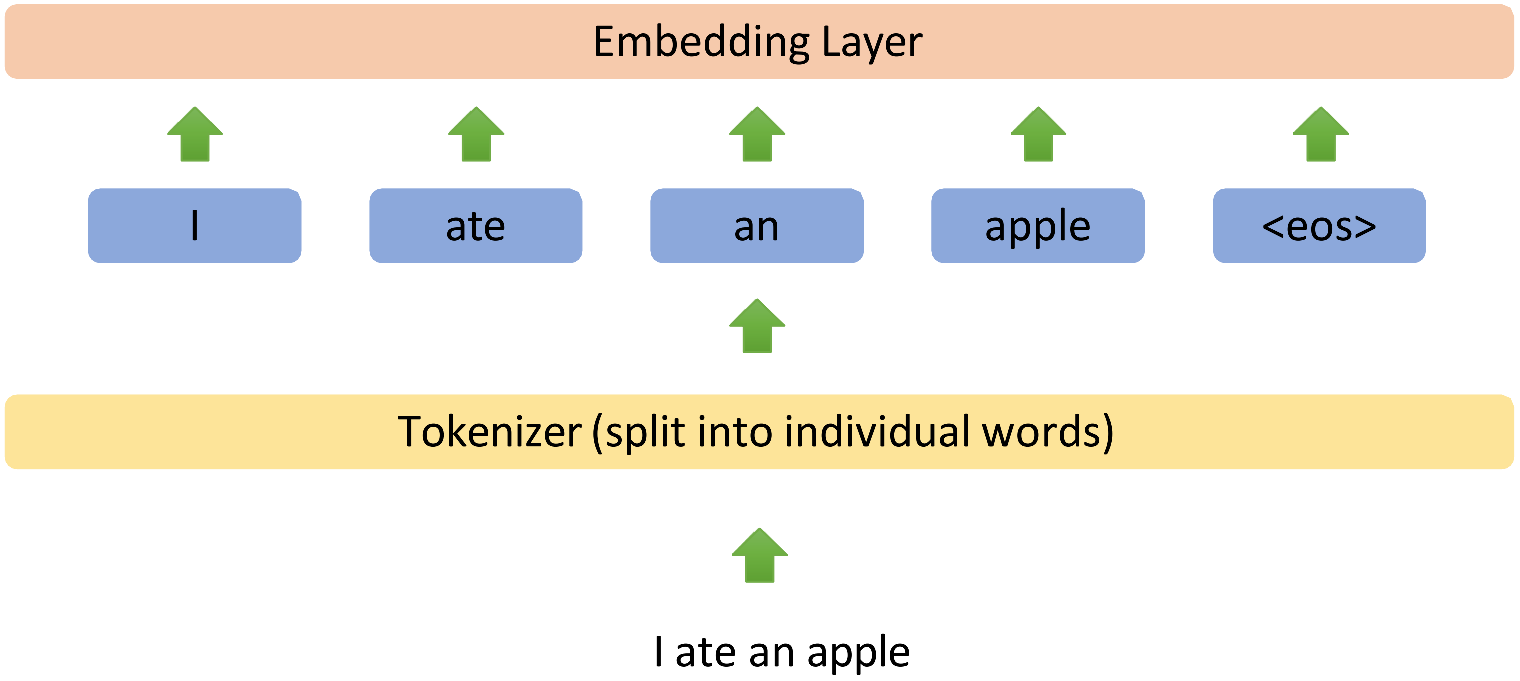
I | ate | an | apple | <eos>

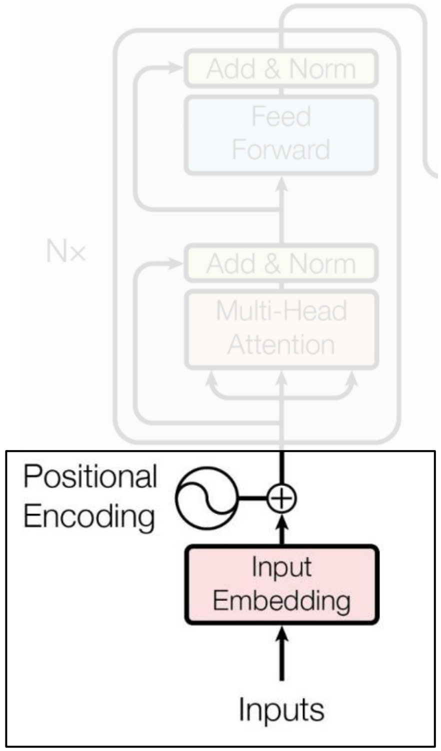
Tokenizer (split into individual words)

I ate an apple



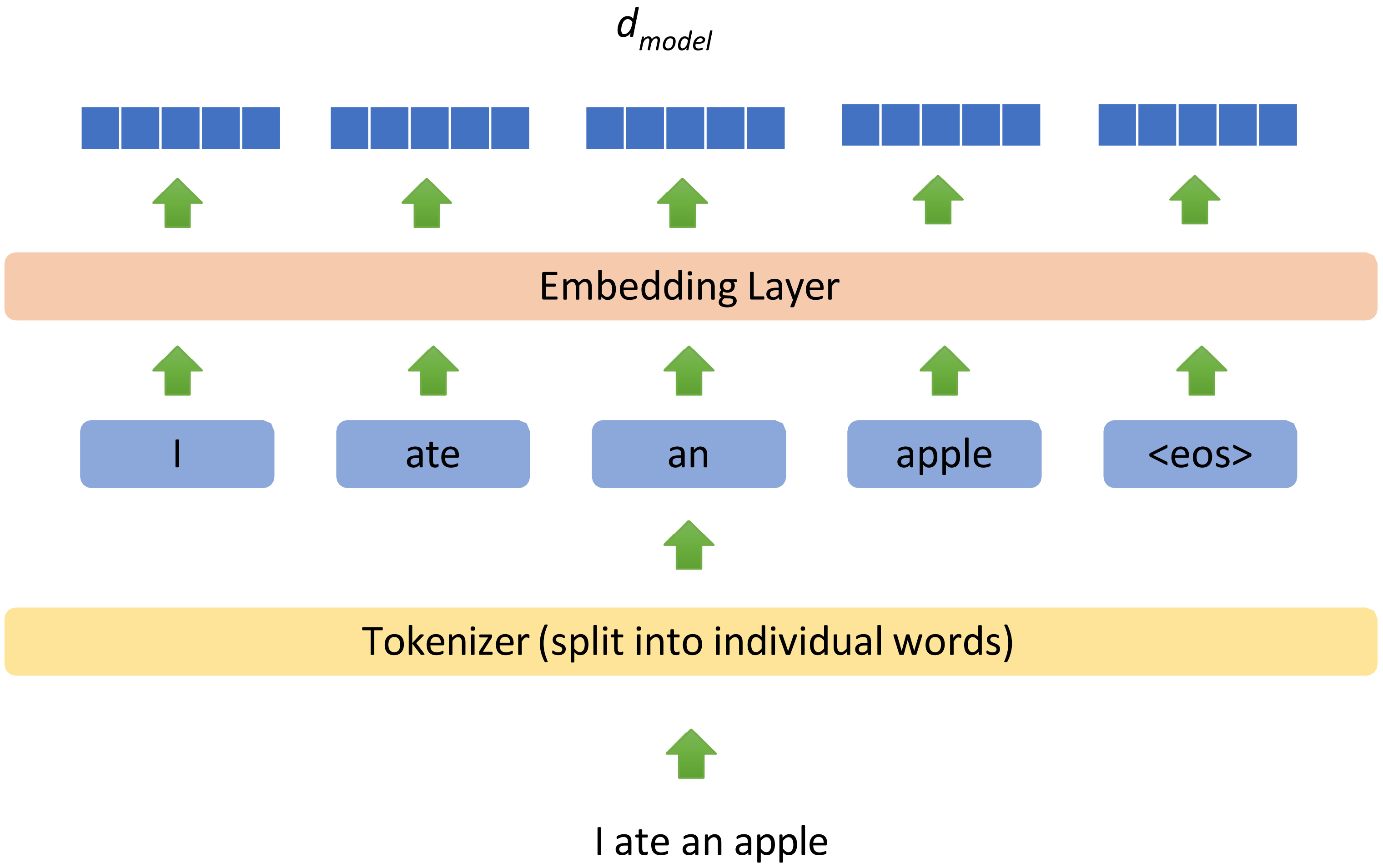
Input Embeddings

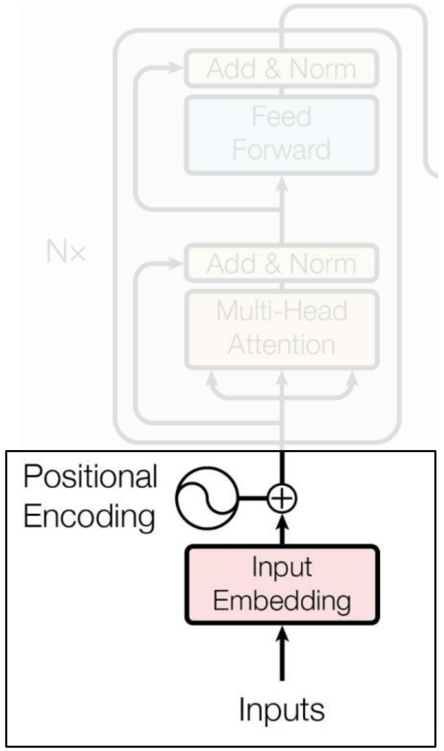




Generate Input Embeddings

Input Embeddings

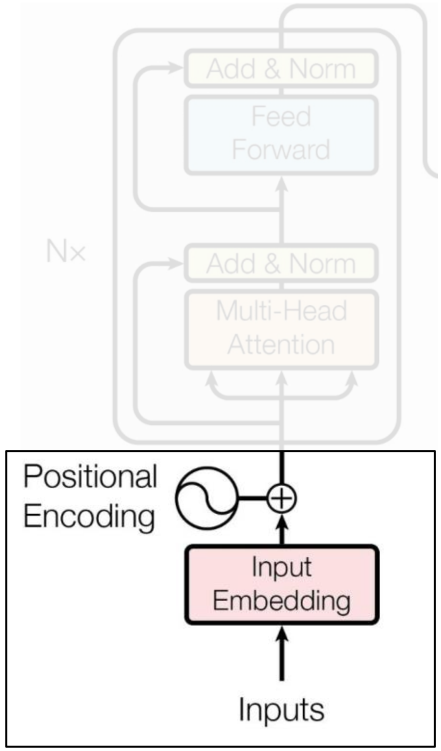




Generate Input Embeddings

Position Encodings

I | ate | an | apple | <eos>

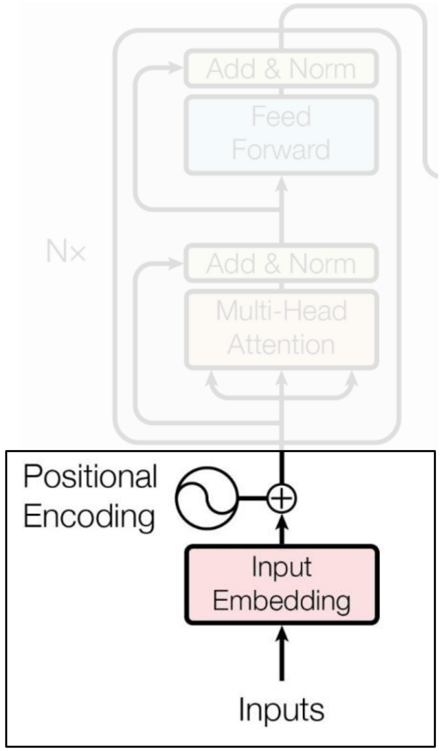


Requirements for Positional Encodings???

Position Encodings

Requirements for Positional Encodings

- Some representation of time
- Should be unique for each position – not cyclic



Position Encodings

Requirements for Positional Encodings

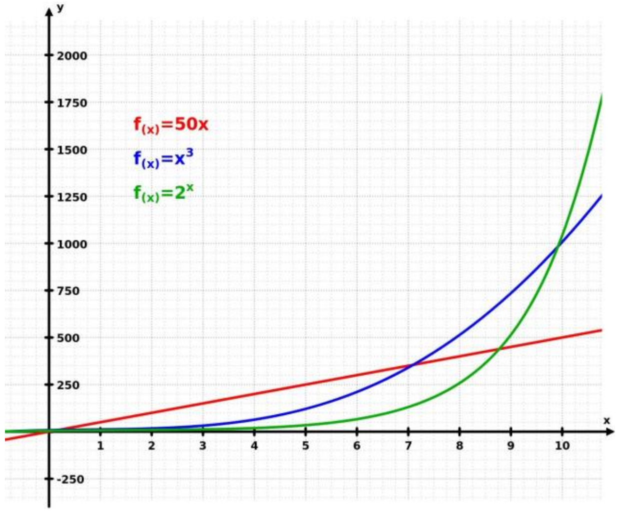
- Some representation of time
- Should be unique for each position – not cyclic

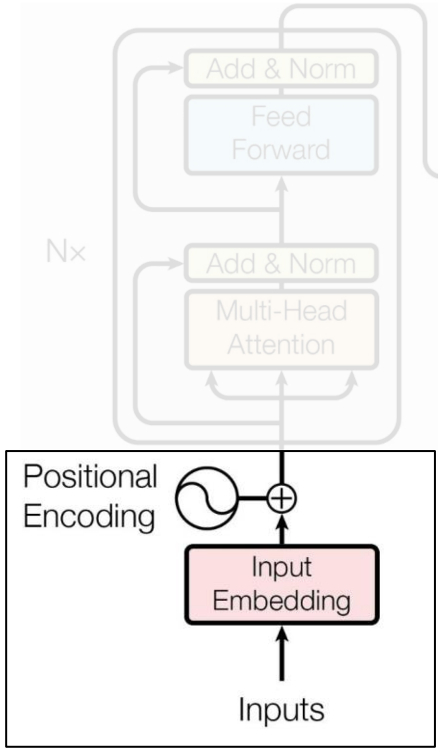
Possible Candidates :

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

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

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





Position Encodings

Requirements for Positional Encodings

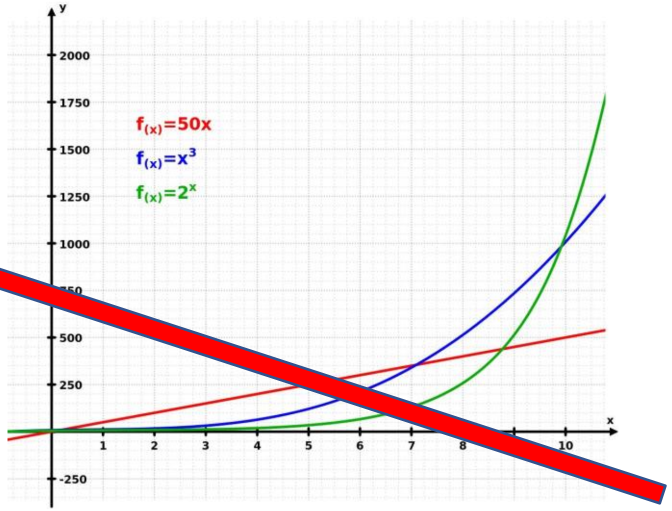
- Some representation of time
- Should be unique for each position – not cyclic

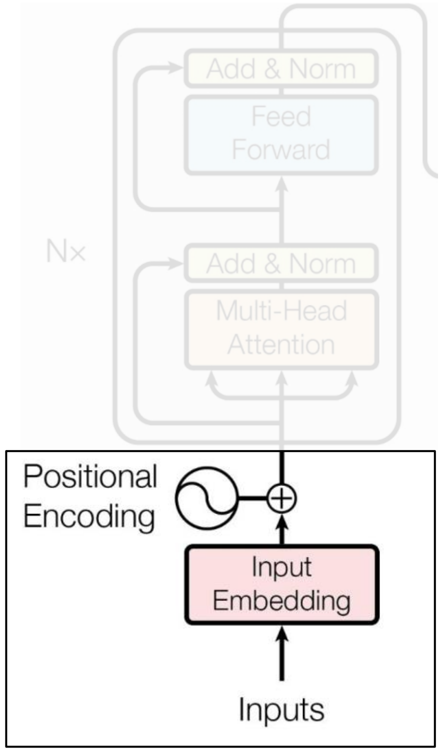
Possible Candidates :

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

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

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





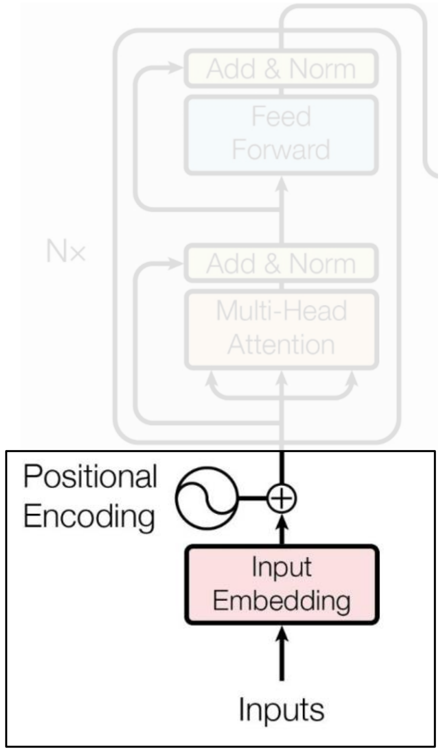
Position Encodings

Requirements for Positional Encodings

- Some representation of time
- 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

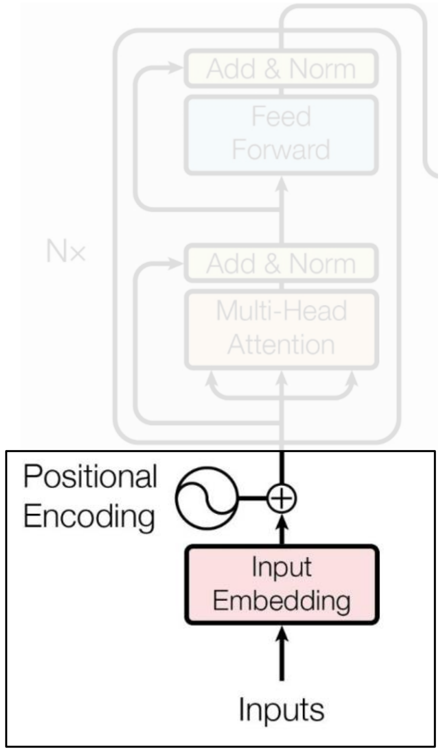
- Some representation of time
- Should be unique for each position – not cyclic
- **Bounded**

Possible Candidates

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

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**



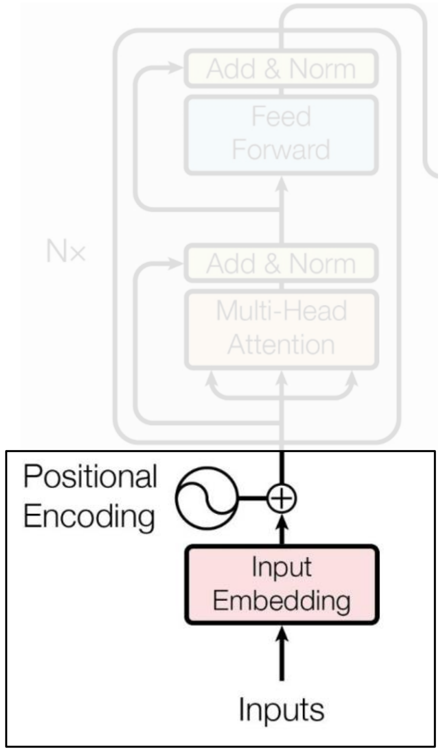
Rotary Position Embedding

ROFORMER: ENHANCED TRANSFORMER WITH ROTARY POSITION EMBEDDING

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

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

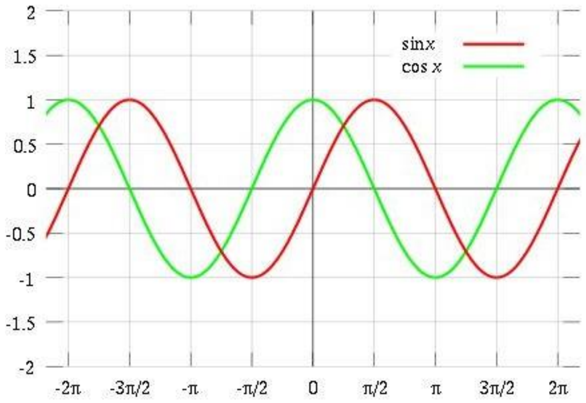
Requirements for Position Encodings

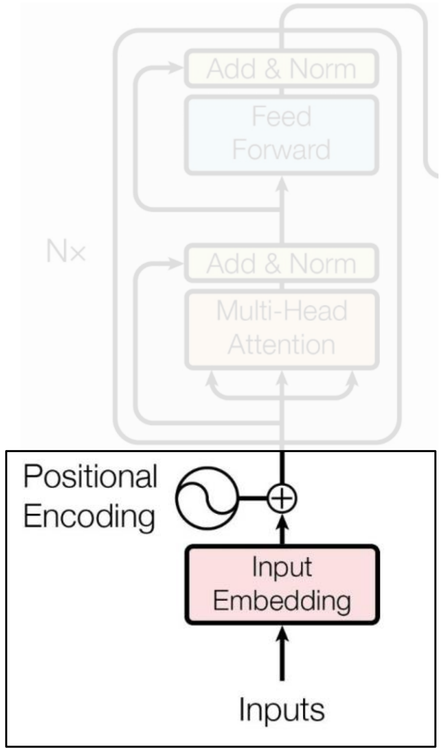
- Some representation of time
- Should be unique for each position
- Bounded

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**.

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

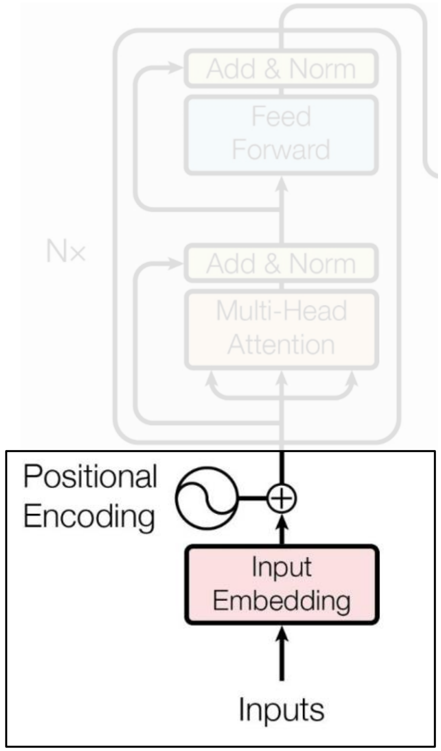
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

pos -> idx of the token in input sentence

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

d model -> embedding dimension of each token

Different calculations for odd and even embedding indices



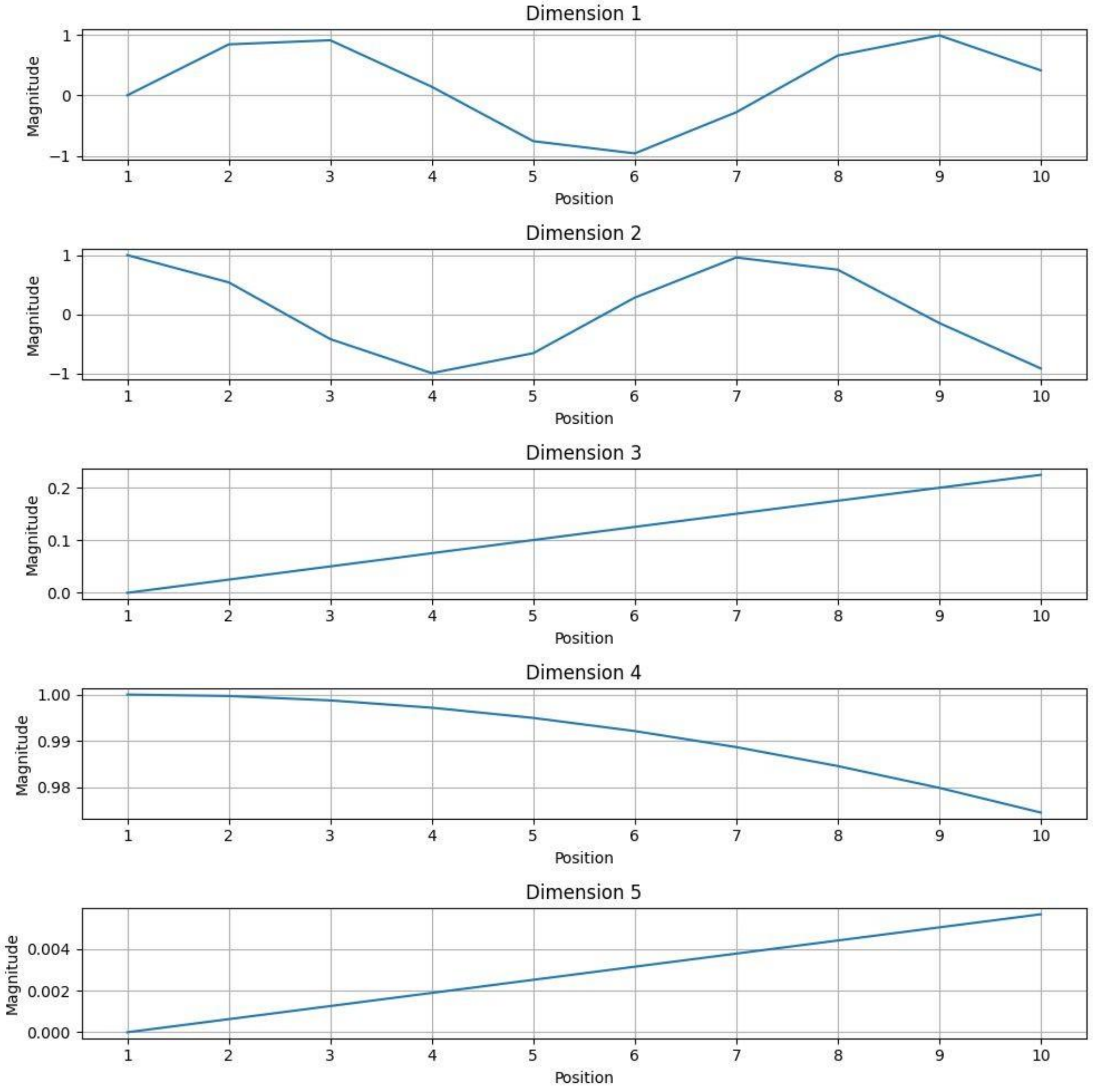
Position Encoding

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

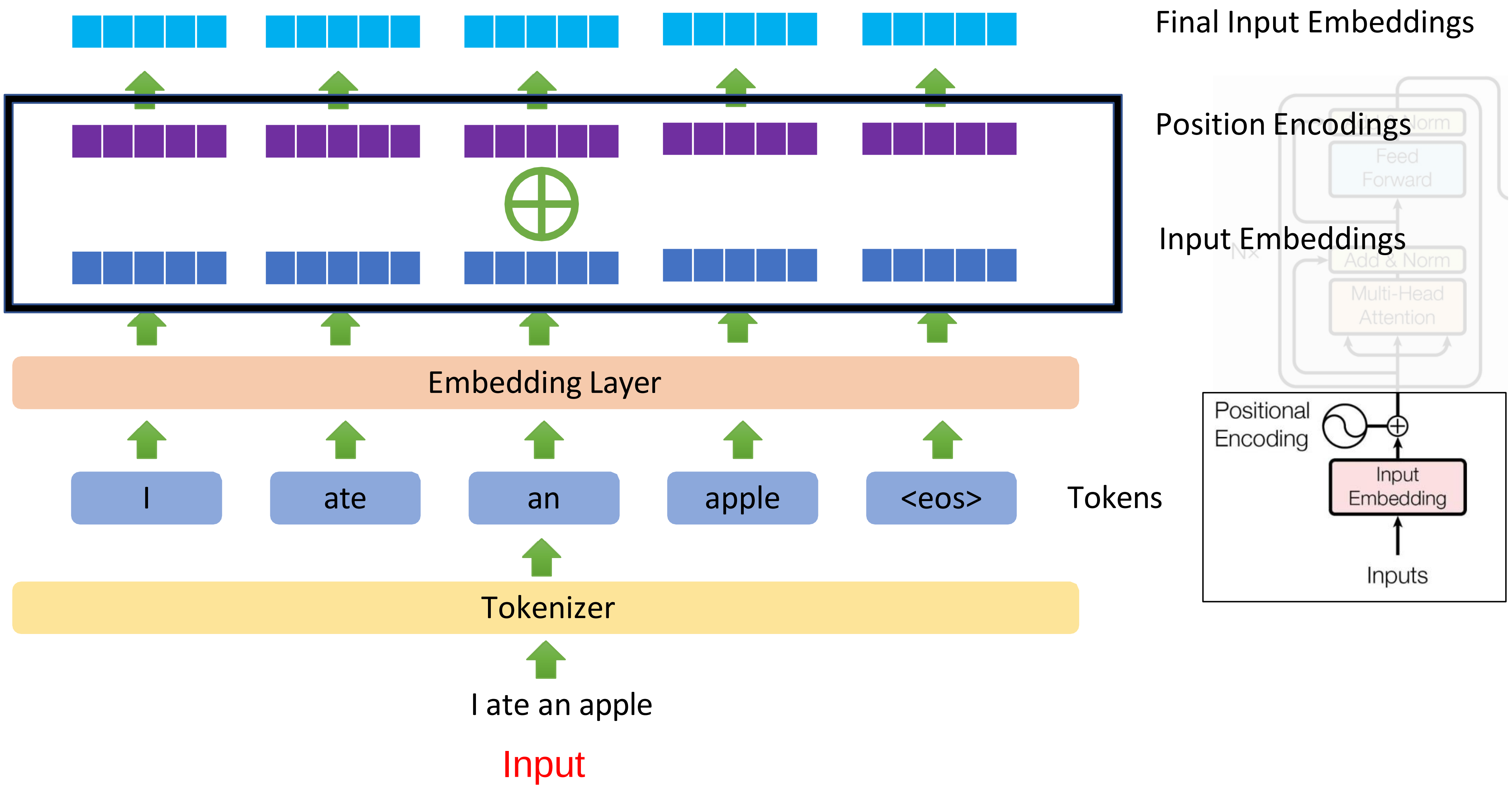
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

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

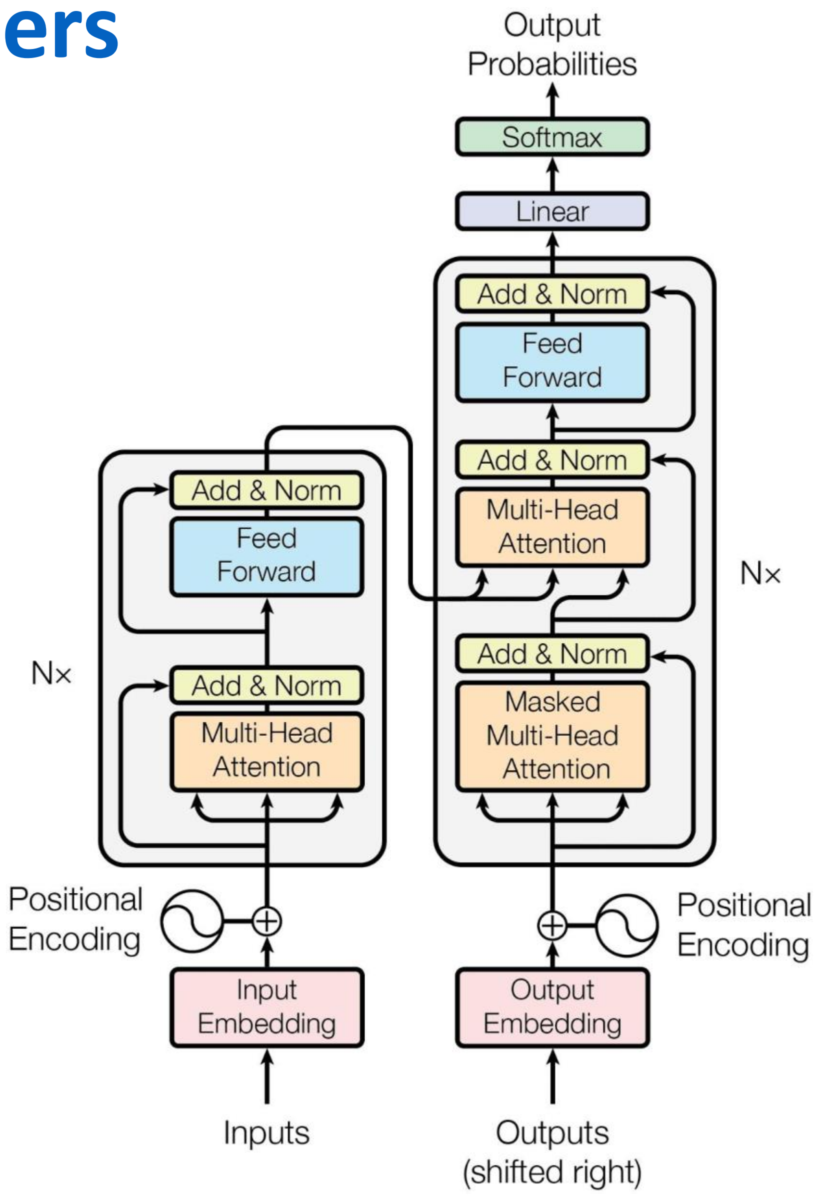


Position Encoding



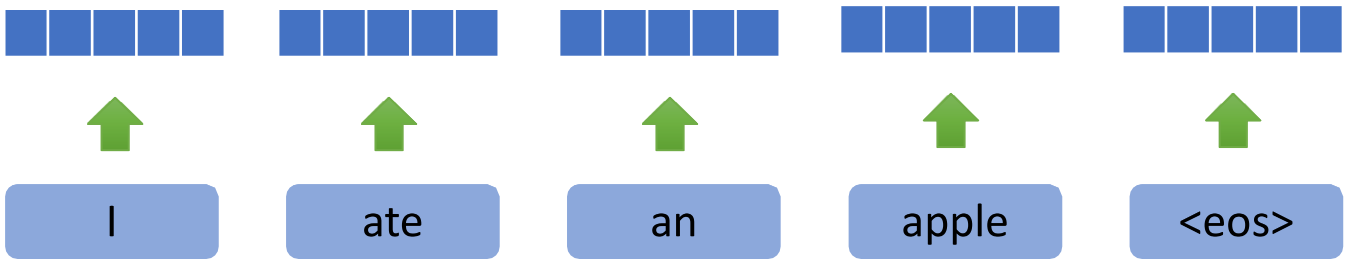
Transformers

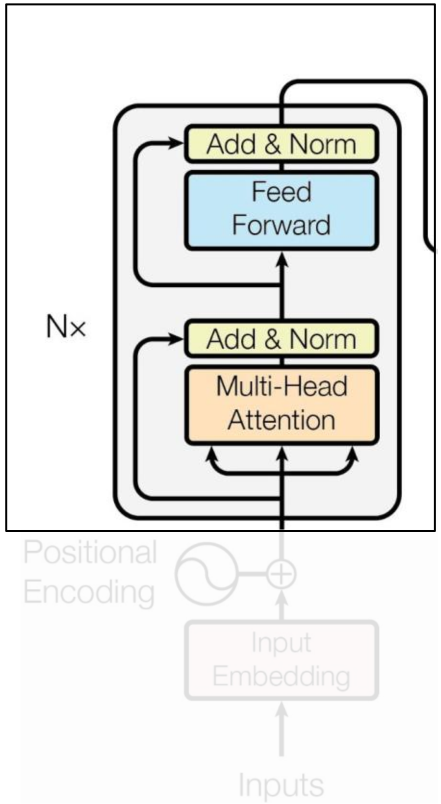
- ✔ **Tokenization**
- ✔ **Input Embeddings**
- ✔ **Position Encodings**
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders
- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models



Encoder

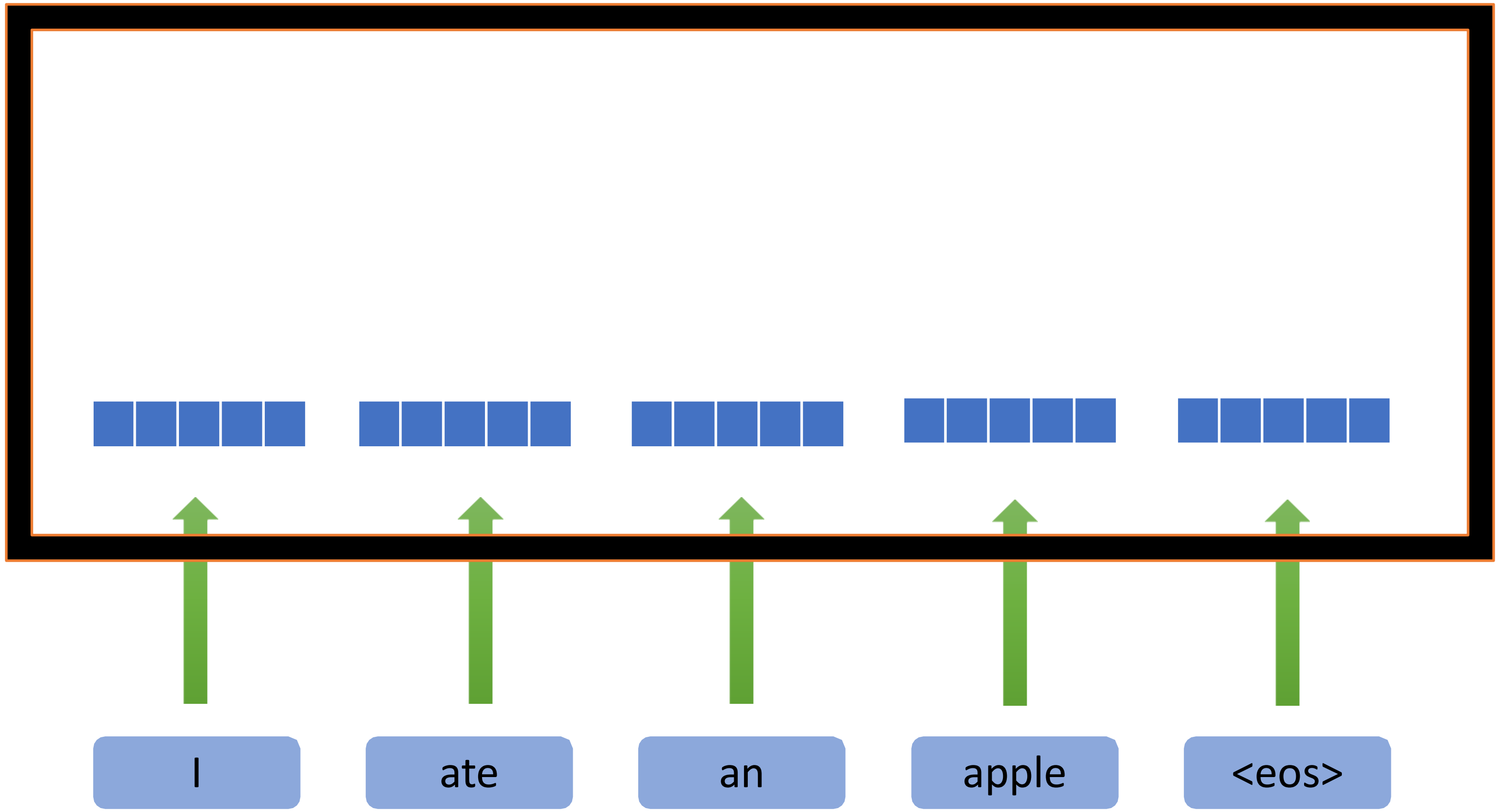
WHERE IS THE CONTEXT ?

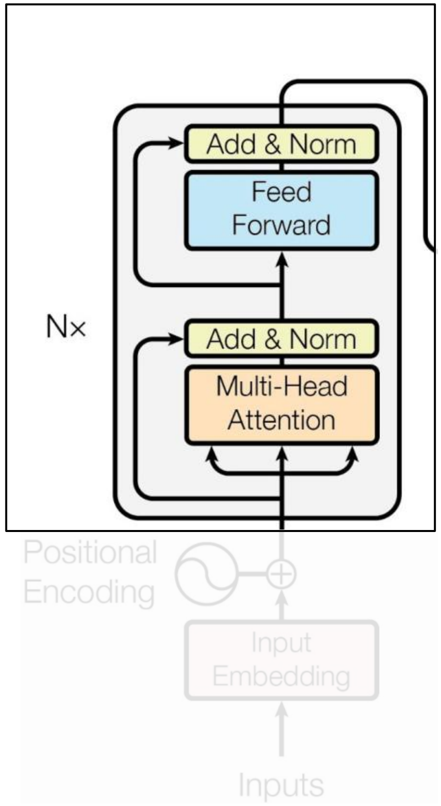




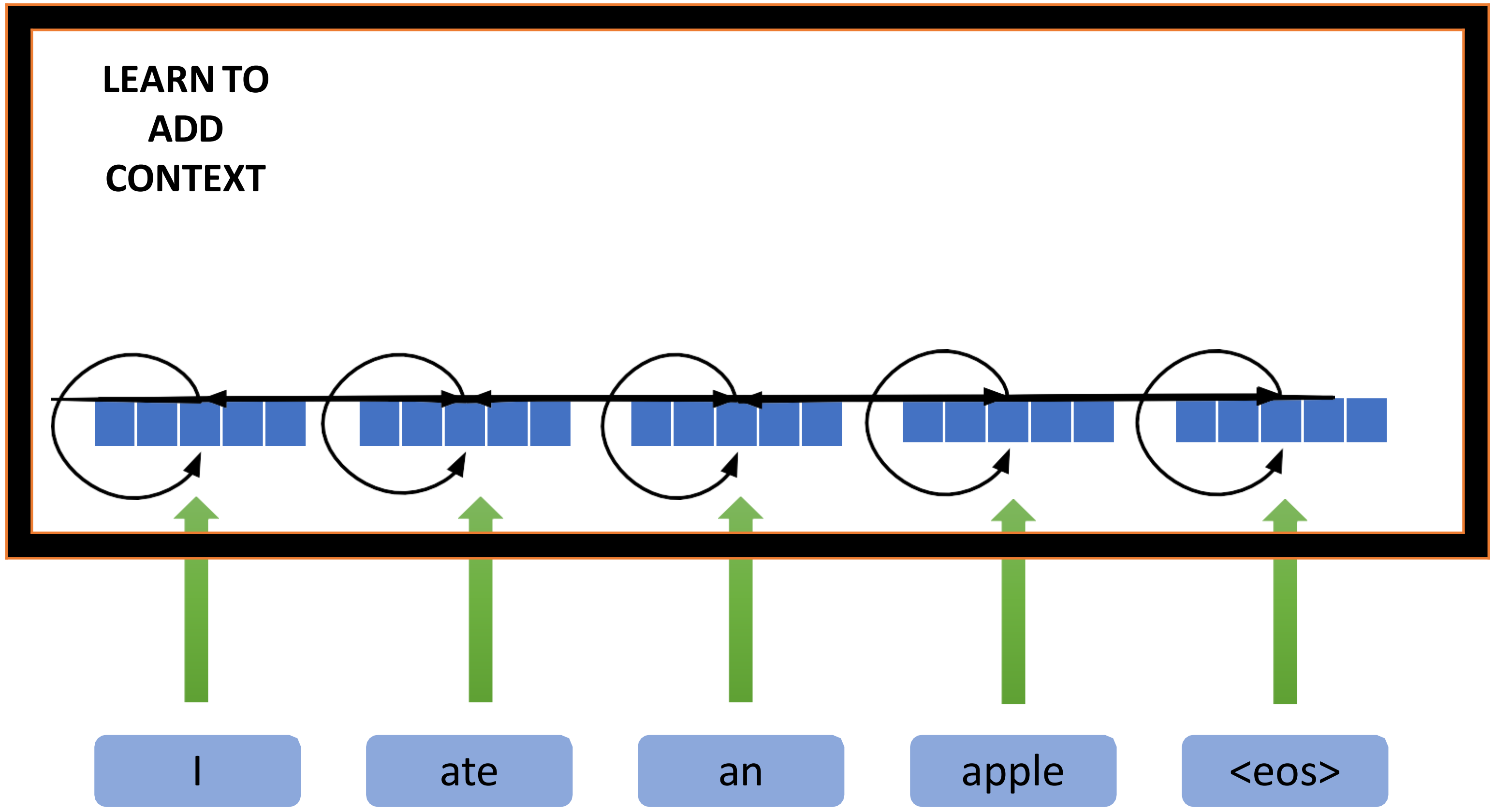
Encoder

BLACK BOX
OF SORTS

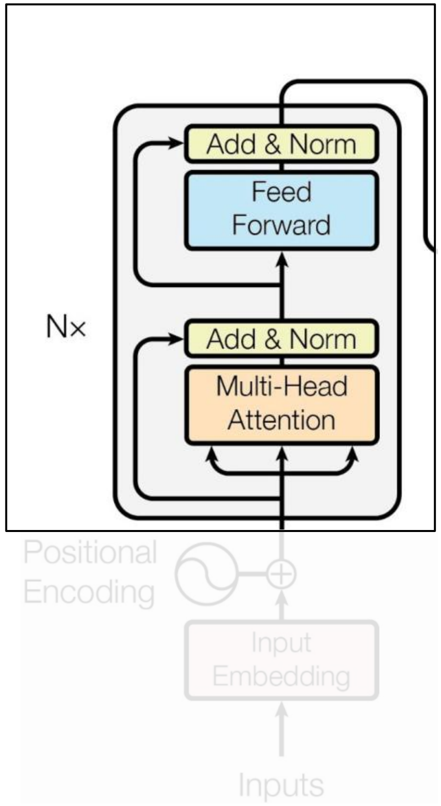




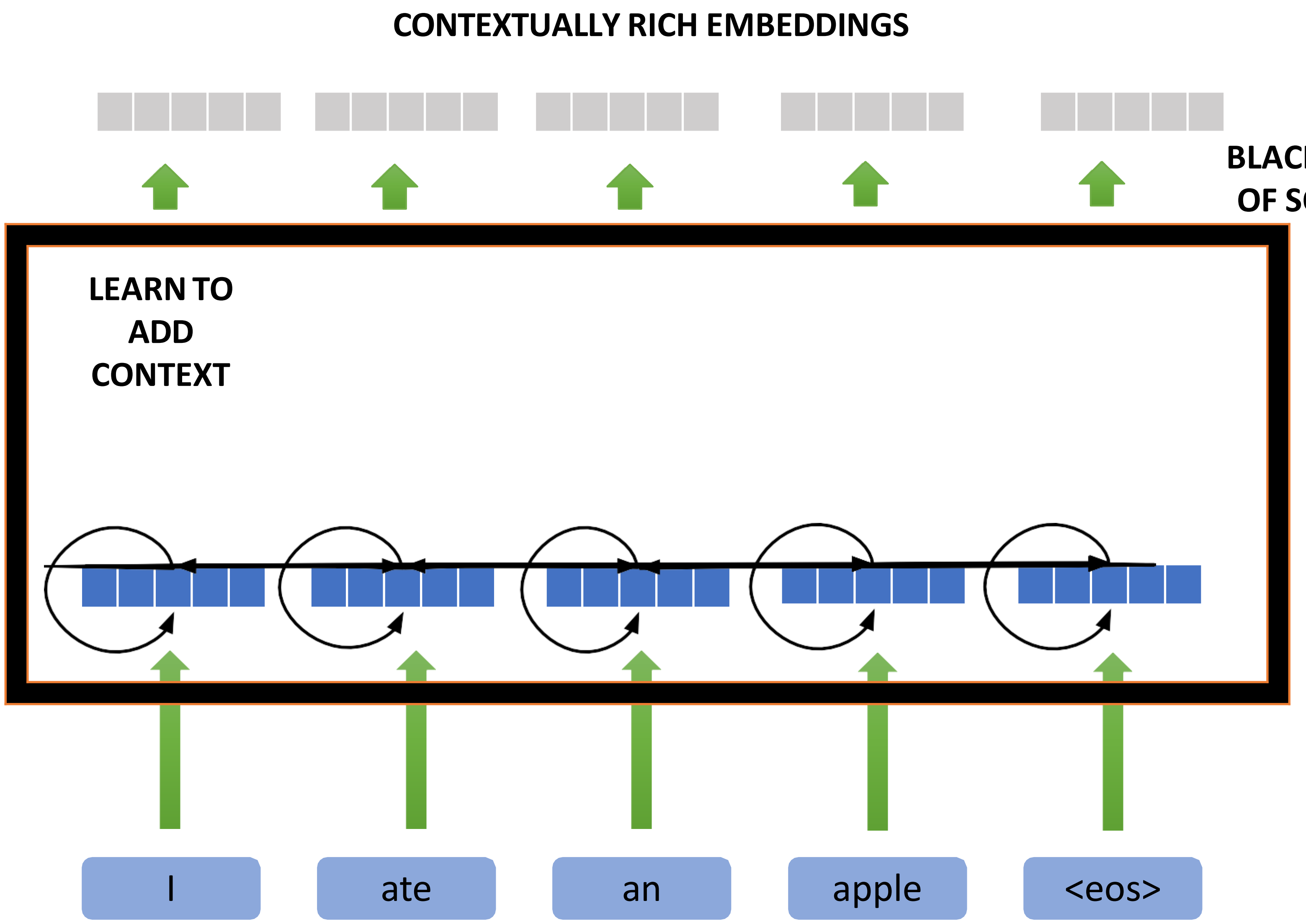
Encoder



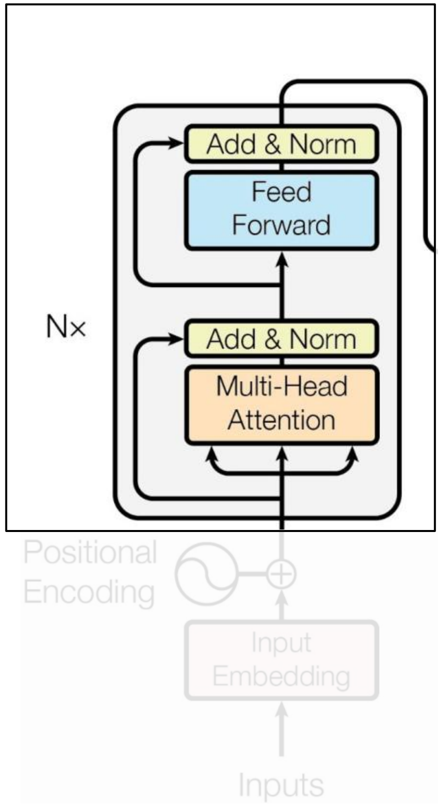
BLACK BOX
OF SORTS



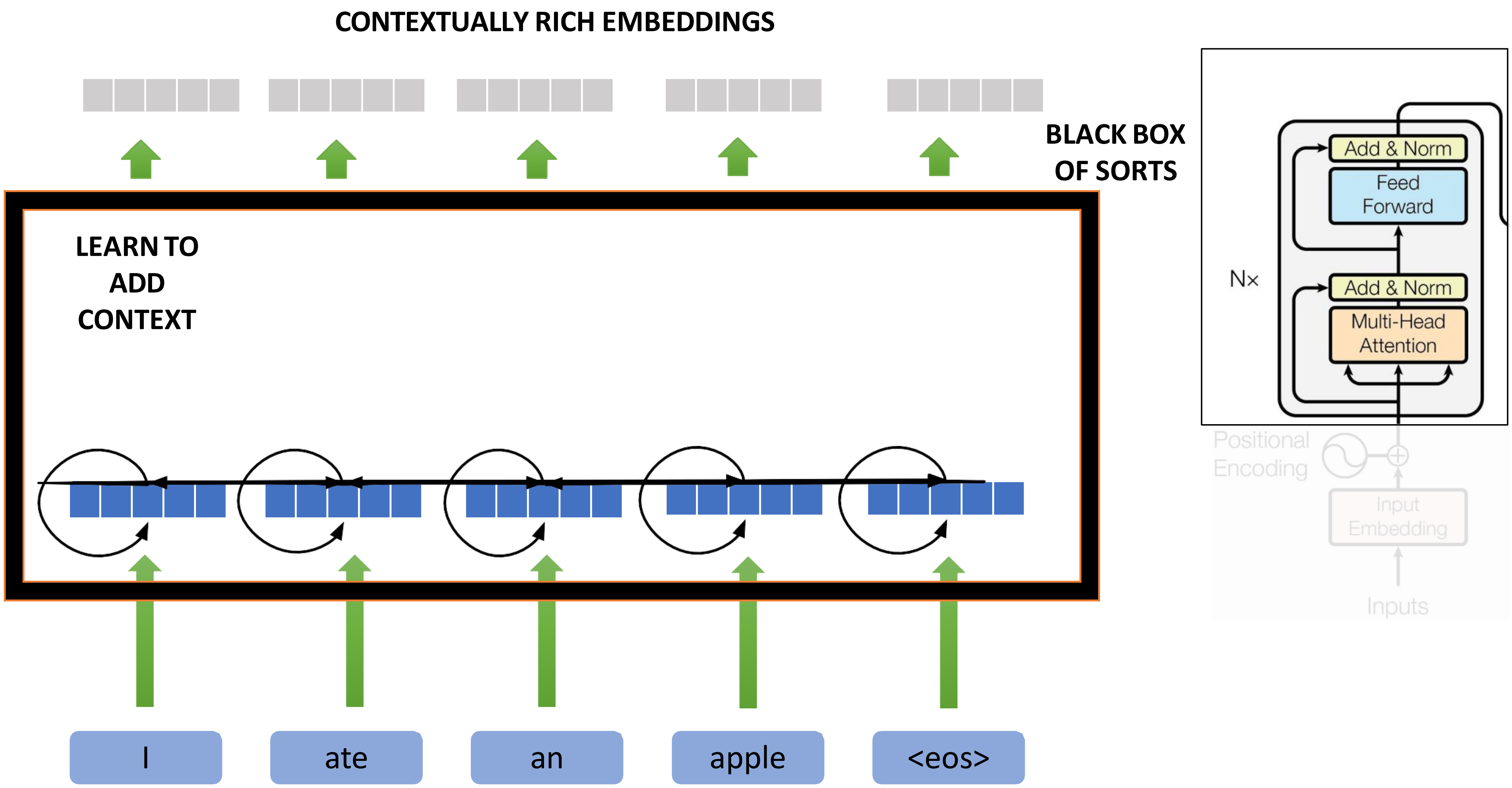
Encoder



BLACK BOX
OF SORTS



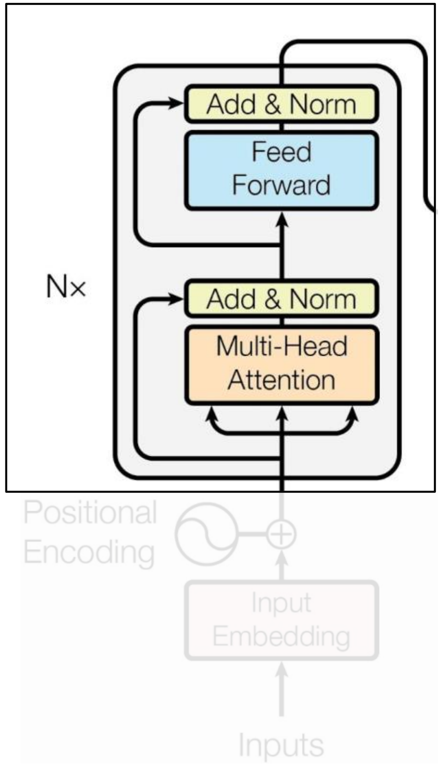
Encoder


CONTEXTUALLY RICH EMBEDDINGS
BLACK BOX OF SORTS
LEARN TO ADD CONTEXT
I
ate
an
apple
<eos>
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs

Attention

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

- Query
- Key
- Value



Query, Key & Value

{Query: "Order details of order_104"}

OR

{Query: "Order details of order_106"}

Database

{Key, Value store}

```
{"order_100": {"items":"a1", "delivery_date":"a2", ...}},
{"order_101": {"items":"b1", "delivery_date":"b2", ...}},
{"order_102": {"items":"c1", "delivery_date":"c2", ...}},
{"order_103": {"items":"d1", "delivery_date":"d2", ...}},
{"order_104": {"items":"e1", "delivery_date":"e2", ...}},
{"order_105": {"items":"f1", "delivery_date":"f2", ...}},
{"order_106": {"items":"g1", "delivery_date":"g2", ...}},
{"order_107": {"items":"h1", "delivery_date":"h2", ...}},
{"order_108": {"items":"i1", "delivery_date":"i2", ...}},
{"order_109": {"items":"j1", "delivery_date":"j2", ...}},
{"order_110": {"items":"k1", "delivery_date":"k2", ...}}
```

Query, Key & Value

{Query: "Order details of order_104"}

OR

{Query: "Order details of order_106"}

{Key, Value store}

```
{"order_100": {"items":"a1", "delivery_date":"a2", ...}},
{"order_101": {"items":"b1", "delivery_date":"b2", ...}},
{"order_102": {"items":"c1", "delivery_date":"c2", ...}},
{"order_103": {"items":"d1", "delivery_date":"d2", ...}},
{"order_104": {"items":"e1", "delivery_date":"e2", ...}},
{"order_105": {"items":"f1", "delivery_date":"f2", ...}},
{"order_106": {"items":"g1", "delivery_date":"g2", ...}},
{"order_107": {"items":"h1", "delivery_date":"h2", ...}},
{"order_108": {"items":"i1", "delivery_date":"i2", ...}},
{"order_109": {"items":"j1", "delivery_date":"j2", ...}},
{"order_110": {"items":"k1", "delivery_date":"k2", ...}}
```

Query, Key & Value

{Key, Value store}

{Query: “Order details of order_104”}

OR

{Query: “Order details of order_106”}

```
{"order_100": {"items":"a1", "delivery_date":"a2", ...}},
{"order_101": {"items":"b1", "delivery_date":"b2", ...}},
{"order_102": {"items":"c1", "delivery_date":"c2", ...}},
{"order_103": {"items":"d1", "delivery_date":"d2", ...}},
{"order_104": {"items":"e1", "delivery_date":"e2", ...}},
{"order_105": {"items":"f1", "delivery_date":"f2", ...}},
{"order_106": {"items":"g1", "delivery_date":"g2", ...}},
{"order_107": {"items":"h1", "delivery_date":"h2", ...}},
{"order_108": {"items":"i1", "delivery_date":"i2", ...}},
{"order_109": {"items":"j1", "delivery_date":"j2", ...}},
{"order_110": {"items":"k1", "delivery_date":"k2", ...}}
```

Query, Key & Value

{Query: “Order details of order_104”}

OR

{Query: “Order details of order_106”}

{Key, Value store}

```
{"order_100": {"items":"a1", "delivery_date":"a2", ...}},
{"order_101": {"items":"b1", "delivery_date":"b2", ...}},
{"order_102": {"items":"c1", "delivery_date":"c2", ...}},
{"order_103": {"items":"d1", "delivery_date":"d2", ...}},
{"order_104": {"items":"e1", "delivery_date":"e2", ...}},
{"order_105": {"items":"f1", "delivery_date":"f2", ...}},
{"order_106": {"items":"g1", "delivery_date":"g2", ...}},
{"order_107": {"items":"h1", "delivery_date":"h2", ...}},
{"order_108": {"items":"i1", "delivery_date":"i2", ...}},
{"order_109": {"items":"j1", "delivery_date":"j2", ...}},
{"order_110": {"items":"k1", "delivery_date":"k2", ...}}
```

Query, Key & Value

Done at the same time !!

{Query: “Order details of order_104”}

OR

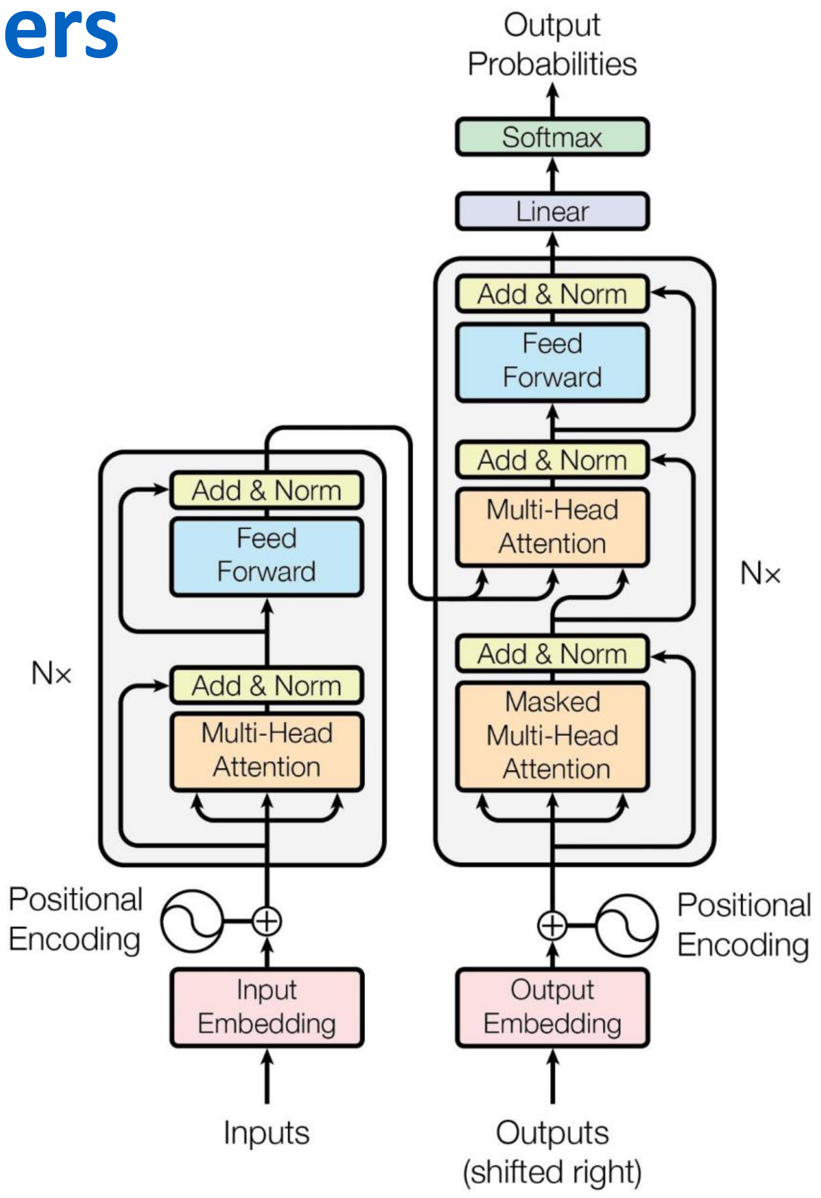
{Query: “Order details of order_106”}

{Key, Value store}

```
{"order_100": {"items":"a1", "delivery_date":"a2", ...}},
{"order_101": {"items":"b1", "delivery_date":"b2", ...}},
{"order_102": {"items":"c1", "delivery_date":"c2", ...}},
{"order_103": {"items":"d1", "delivery_date":"d2", ...}},
{"order_104": {"items":"e1", "delivery_date":"e2", ...}},
{"order_105": {"items":"f1", "delivery_date":"f2", ...}},
{"order_106": {"items":"g1", "delivery_date":"g2", ...}},
{"order_107": {"items":"h1", "delivery_date":"h2", ...}},
{"order_108": {"items":"i1", "delivery_date":"i2", ...}},
{"order_109": {"items":"j1", "delivery_date":"j2", ...}},
{"order_110": {"items":"k1", "delivery_date":"k2", ...}}
```

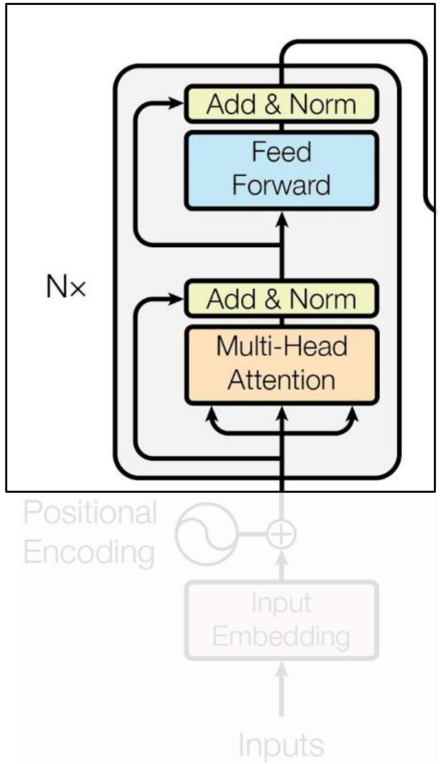
Transformers

- ✔ **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



Self Attention

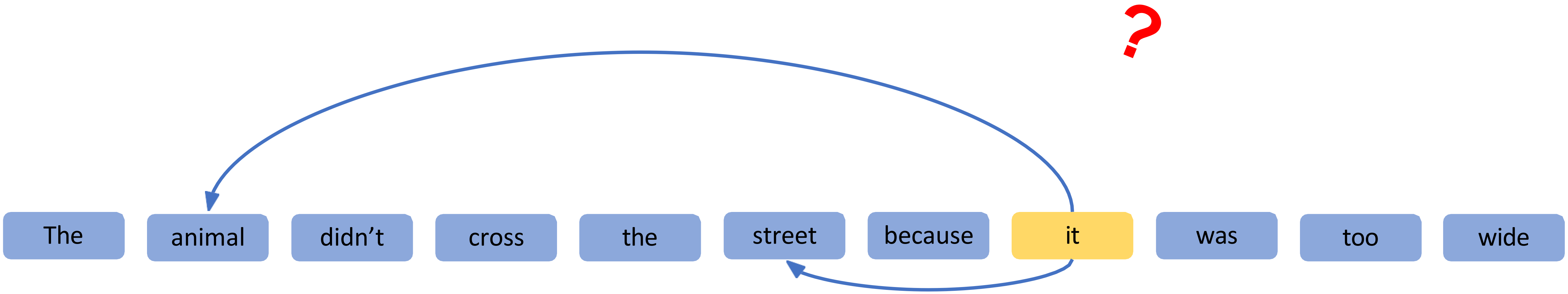
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Self Attention

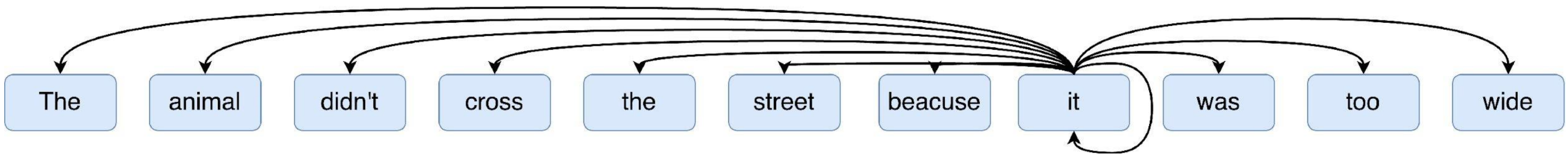
The animal didn’t cross the street because it was too wide

Self Attention



coreference resolution?

Self Attention

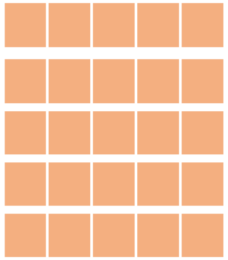


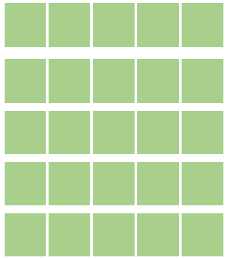
SELF

Query Inputs = Key Inputs = Value Inputs

Self Attention

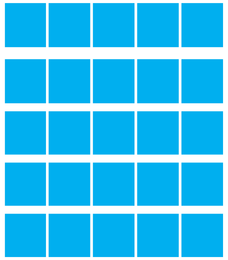
$R^{d_{model} \times d_{model}}$



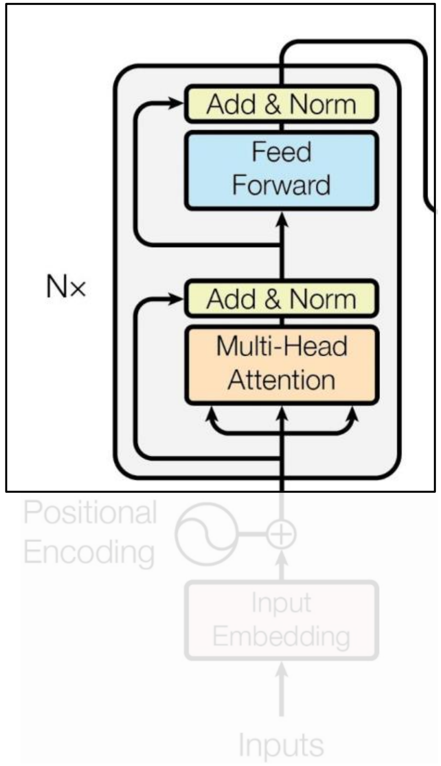


W_Q W_K W_V

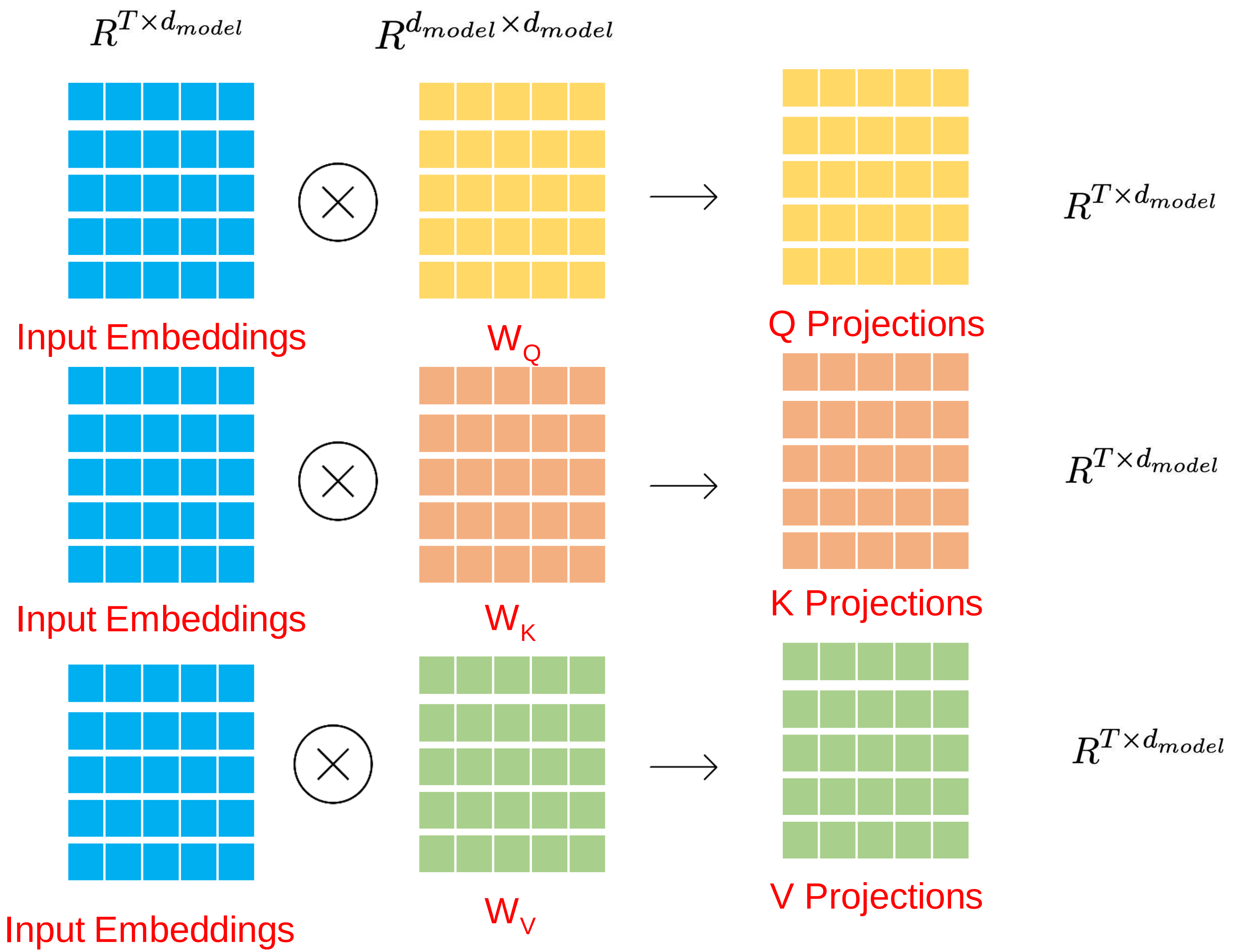
$R^{T \times d_{model}}$

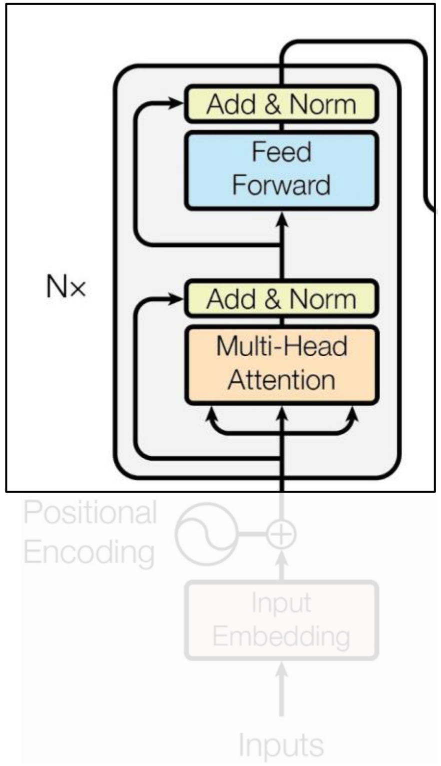


Input Embeddings



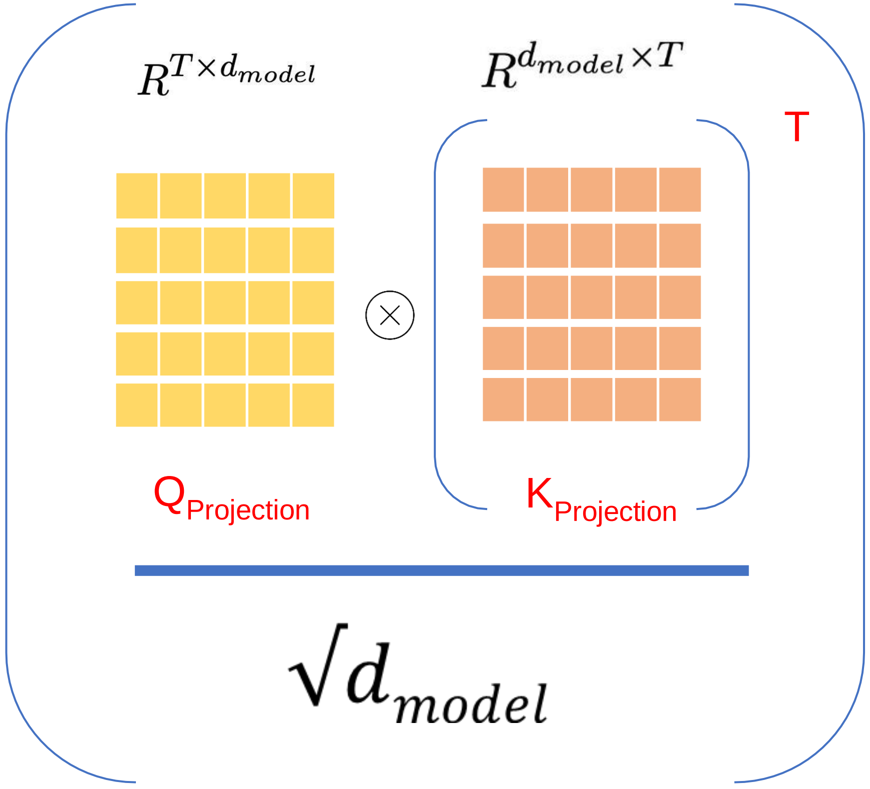
Self Attention



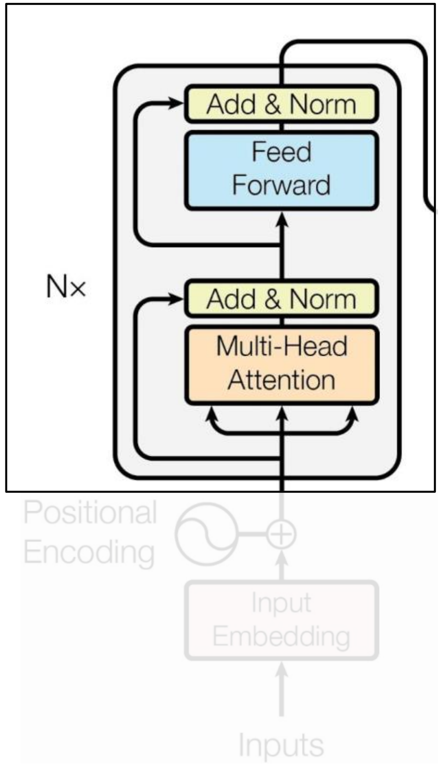


Self Attention

softmax



$R^{T\times T}$



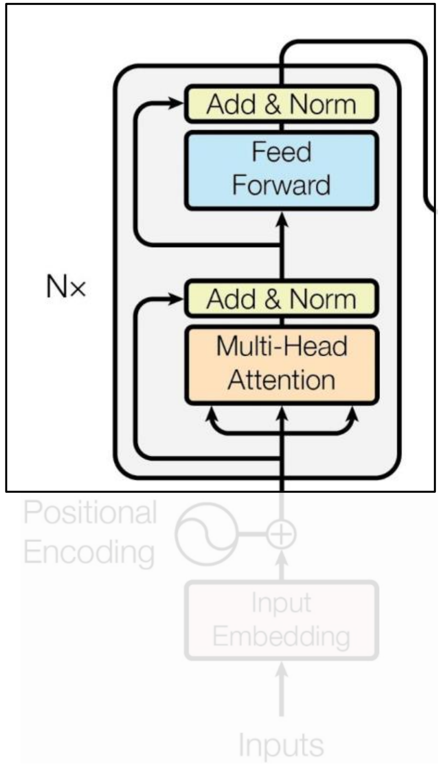
Self Attention

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

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

$$\text{softmax}\left(\frac{Q_{Projection} \otimes \left(K_{Projection}\right)^{T}}{\sqrt{d_{model}}}\right)$$

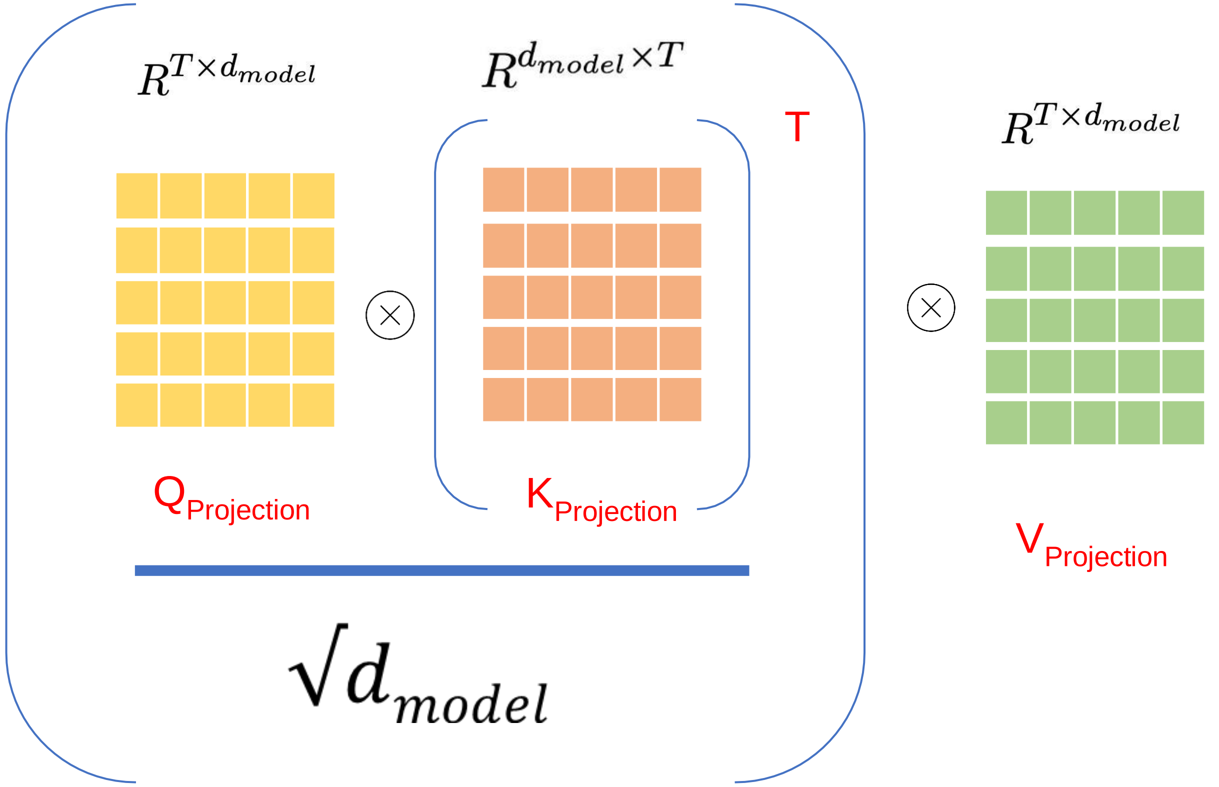
$R^{T\times T}$



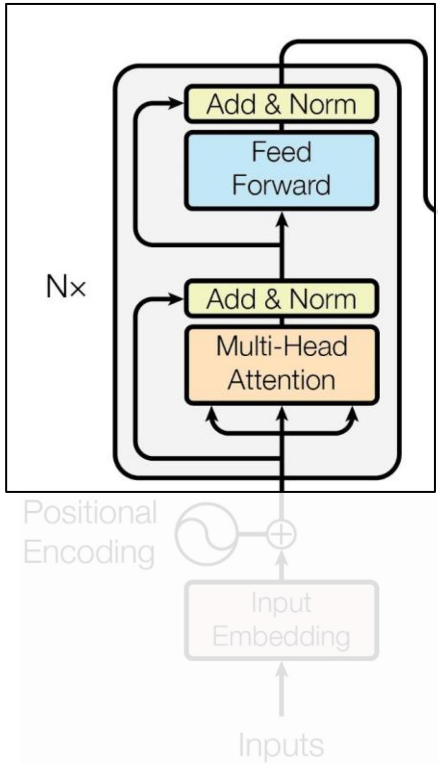
Self Attention

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

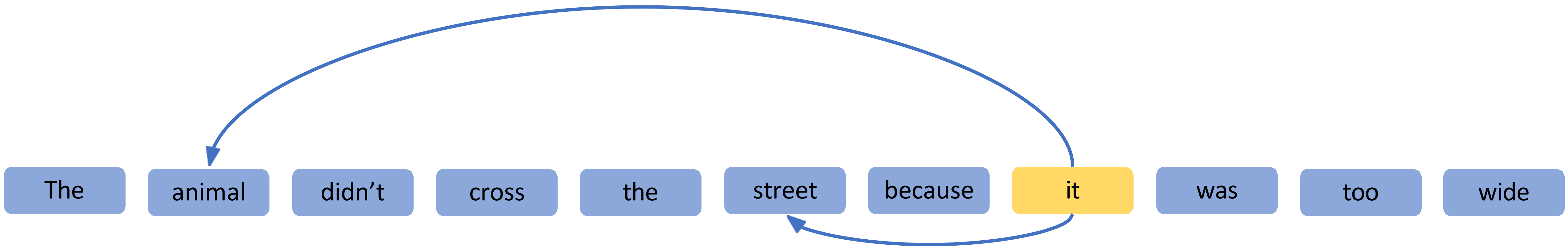
softmax



$R^{T\times T}$

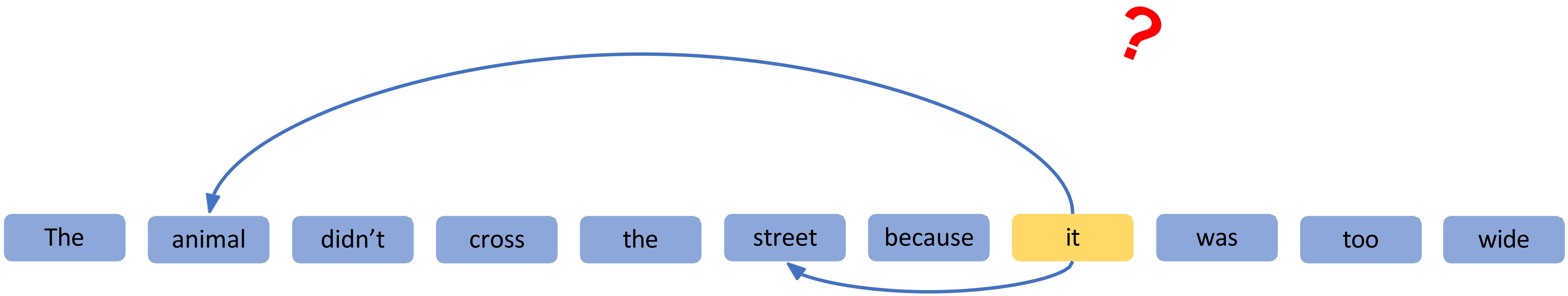


Self Attention



Coreference resolution

Self Attention



Sentence boundaries ?

Coreference resolution ✓

Context ?

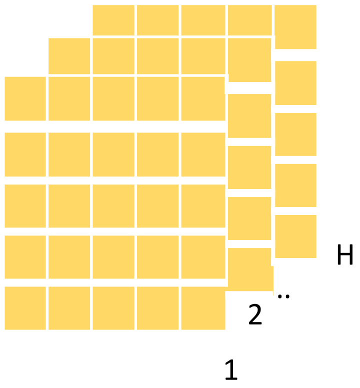
Semantic relationships ?

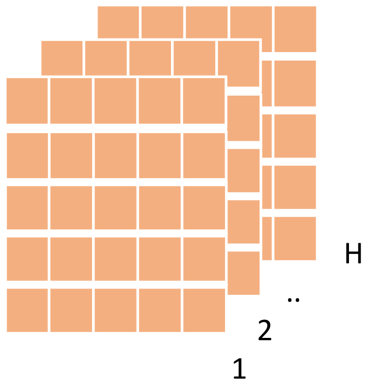
Part of Speech ?

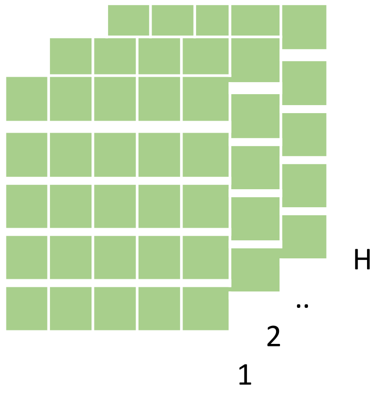
Comparisons ?

Multi-Head Attention

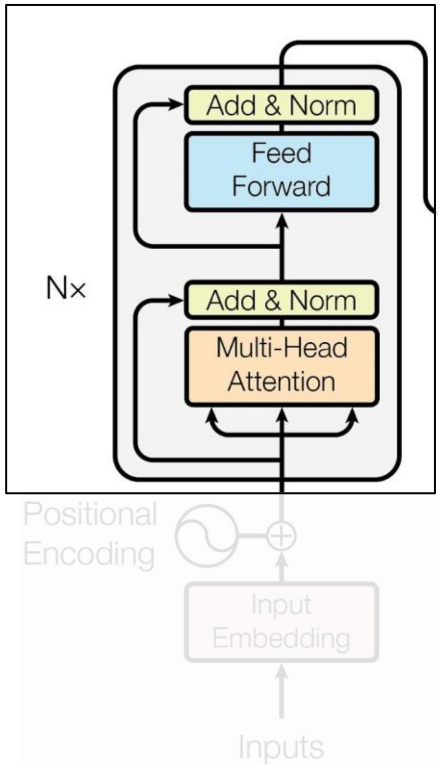
$R^{d_{model} \times d_h}$

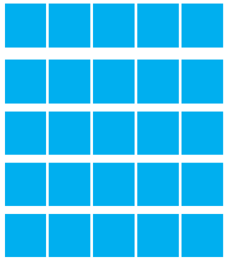






$W_{Q1,}$ $W_{Q2, \dots}$ $W_{QH,}$ $W_{K1,}$ $W_{K2, \dots}$ $W_{KH,}$ $W_{V1,}$ $W_{V2, \dots}$ $W_{VH,}$



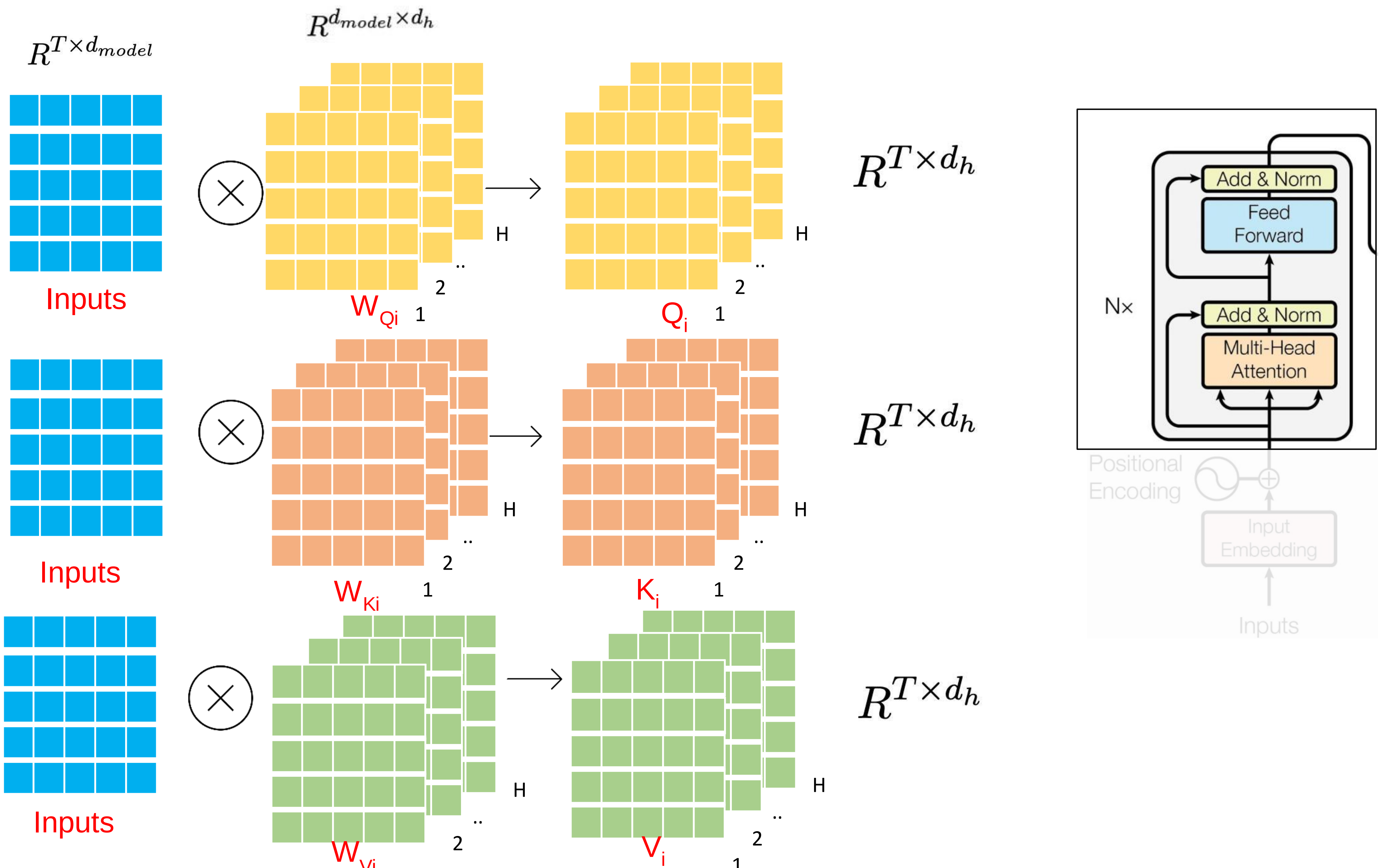


Input Embeddings

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

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

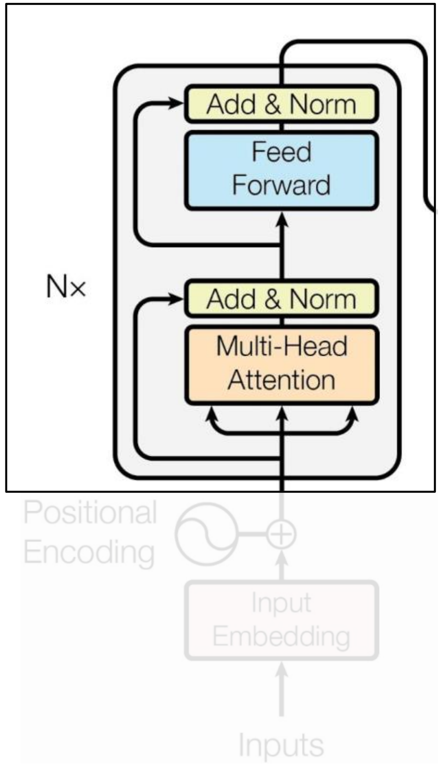
Multi-Head Attention



Multi-Head Attention

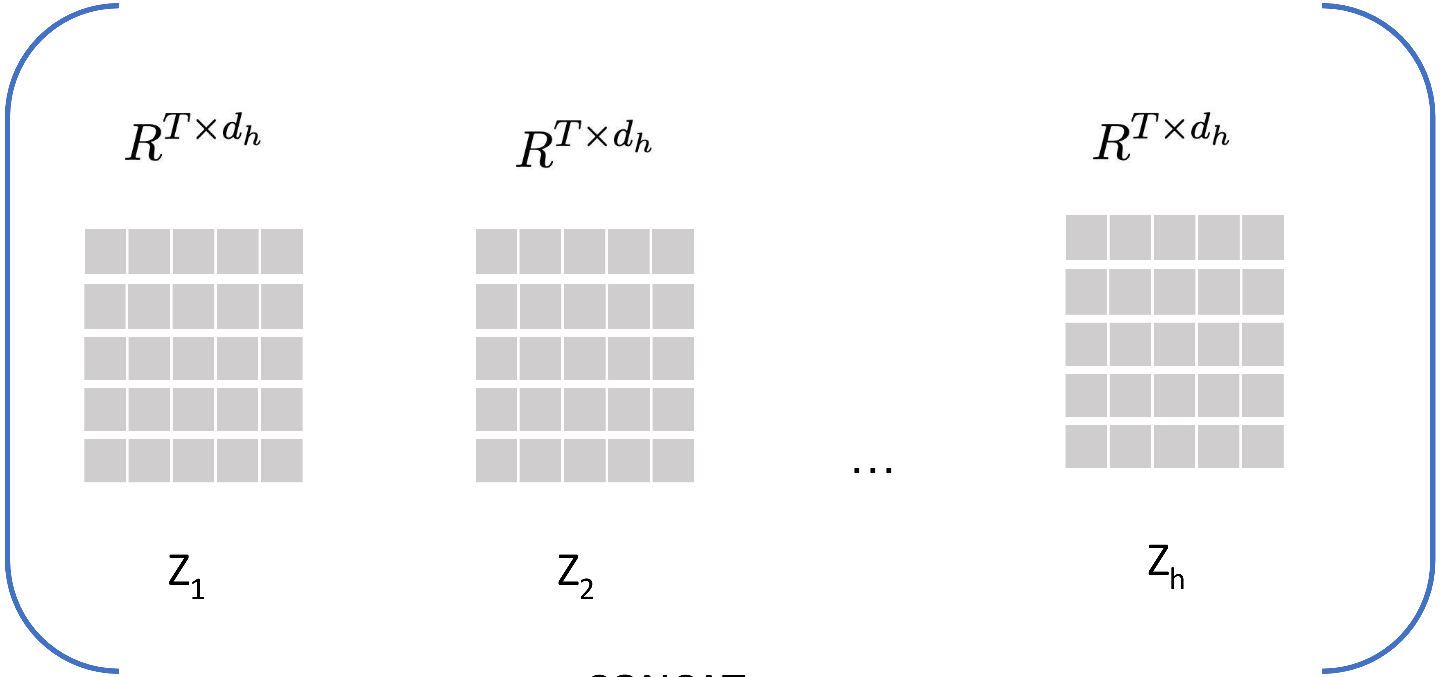
$$\text{softmax}\left(\frac{\overset{R^{T\times d_h}}{Q_i}\ \otimes\ \overset{R^{d_h\times T}}{(K_i)^{T}}}{\sqrt{d_{model}}}\right)\ \otimes\ \overset{R^{T\times d_h}}{V_i}$$

$R^{T\times T}$

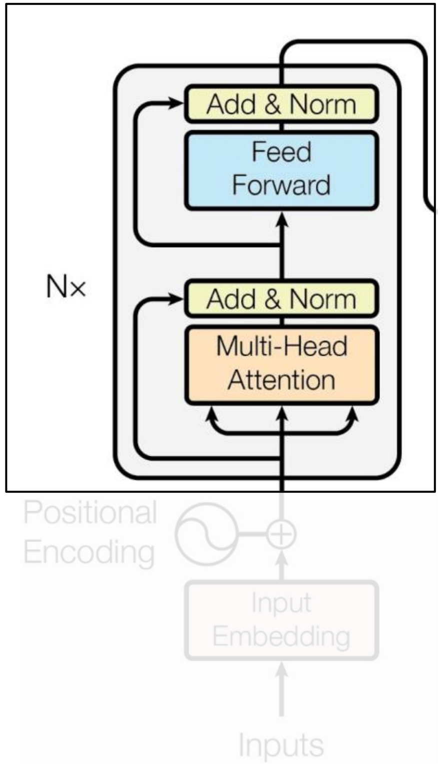


for all i ∈ [1, h]

Multi-Head Attention



CONCAT

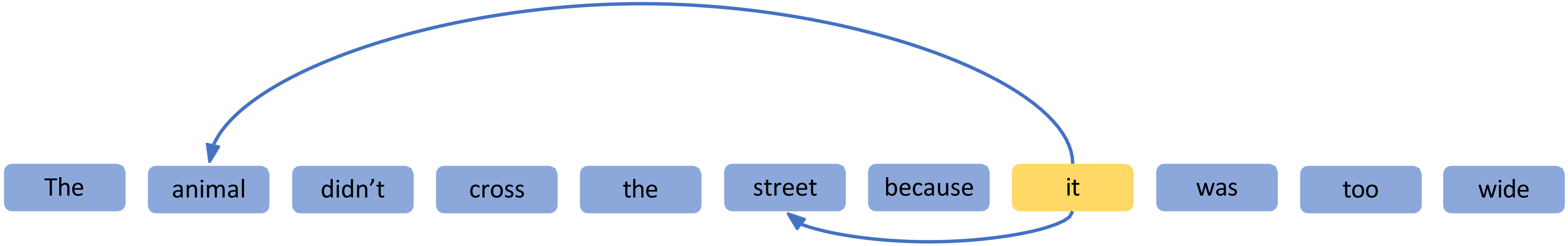


Multi Head Attention : Z

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

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

Multi-Head Attention



Sentence boundaries ✓

Coreference resolution ✓

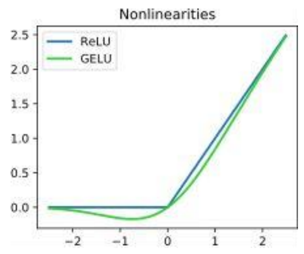
Context ✓

Semantic relationships ✓

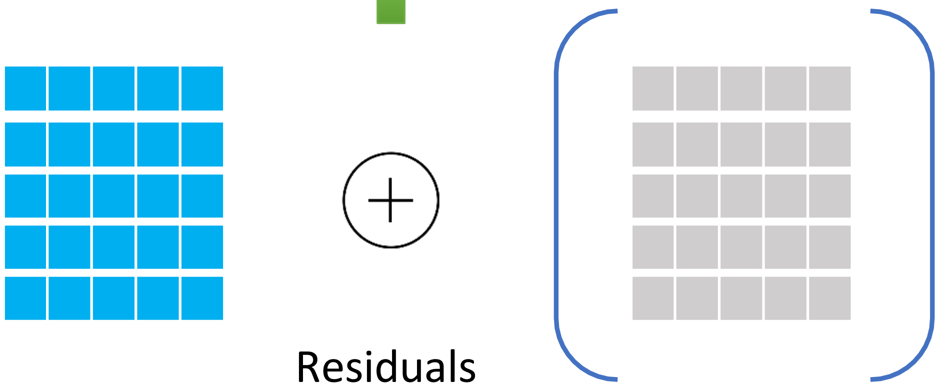
Part of speech ✓

Comparisons ✓

Feed Forward



Feed Forward

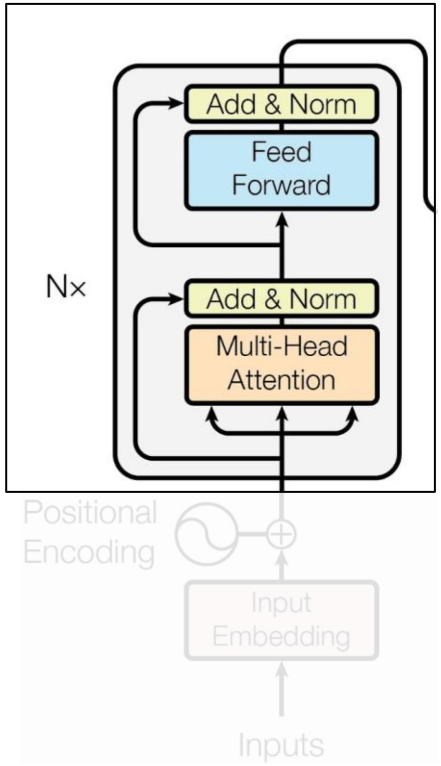


Input

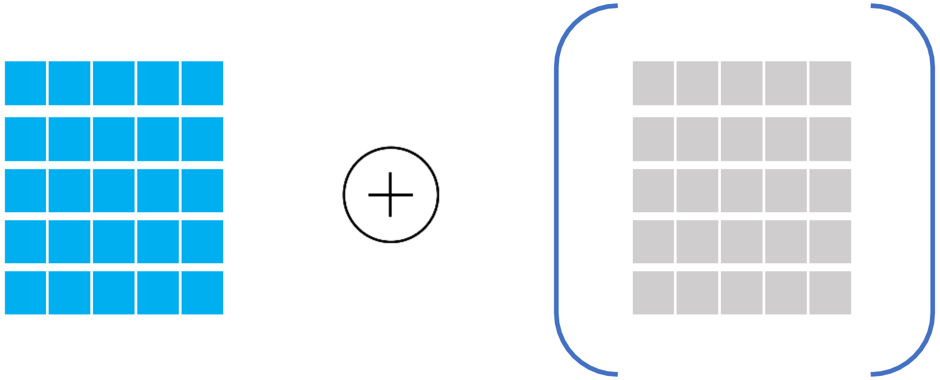
Norm(Z)

Feed Forward

- Non Linearity
- Complex Relationships
- Learn from each other

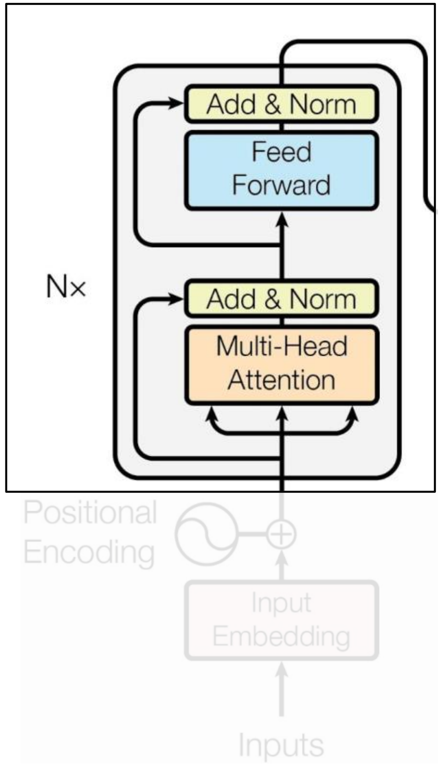


Add & Norm



Input

Norm(Z)



Normalization

Mean 0, Std dev 1

Stabilizes training

Regularization effect

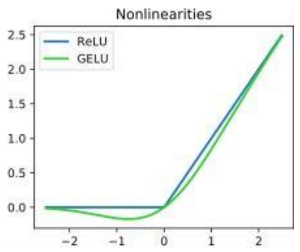
Add Residuals

Avoid vanishing gradients

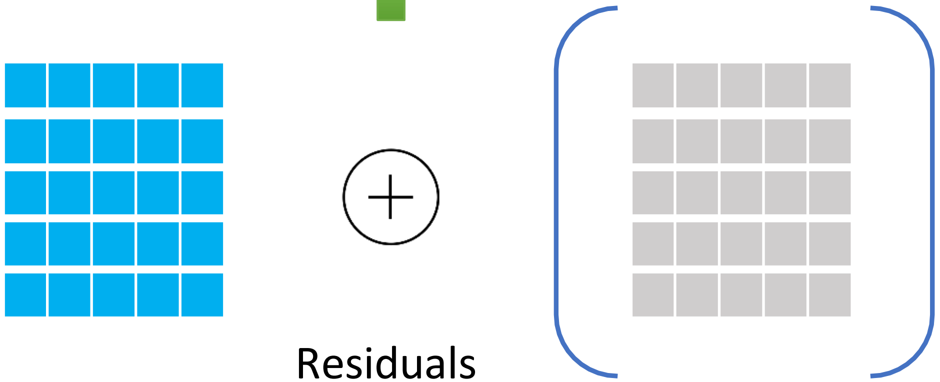
Train deeper networks

Add & Norm

Add & Norm



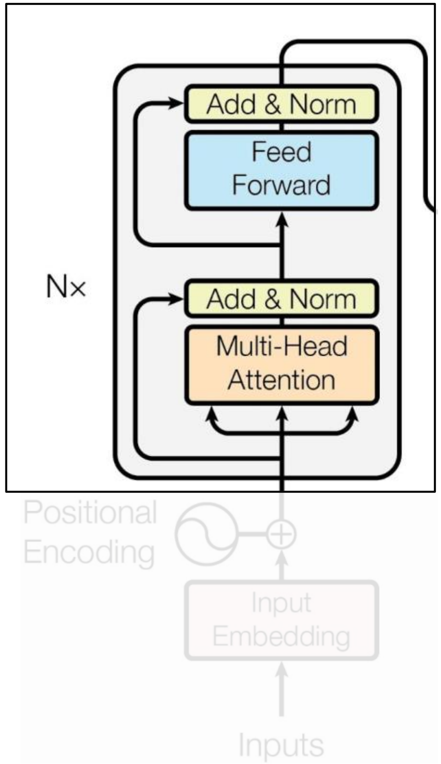
Feed Forward



Residuals

Input

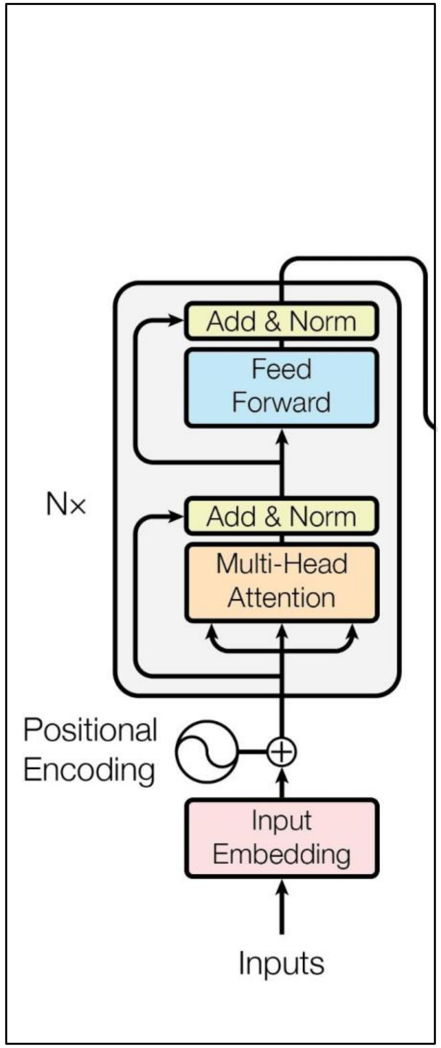
Norm(Z)



Encoders

Encoder

ENCODER



Encoders

Encoder

ENCODER

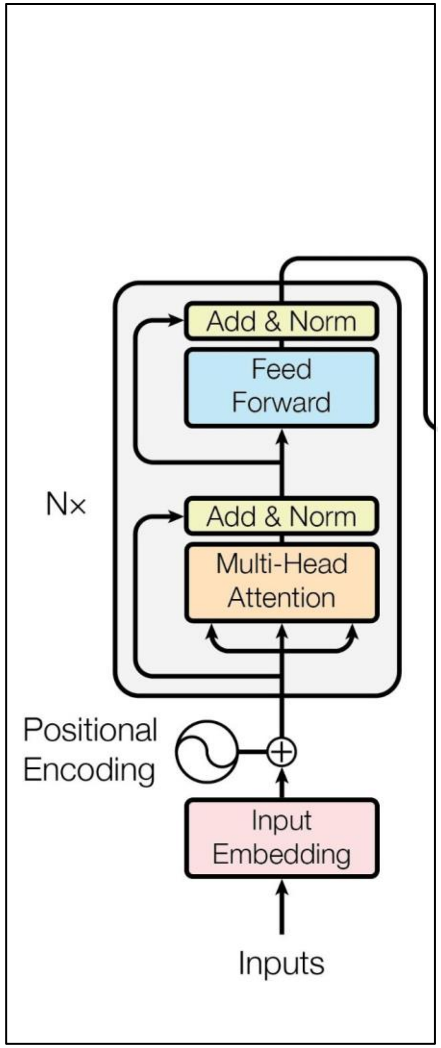
.
.
.

ENCODER

ENCODER

Input to Encoder_{i+1}

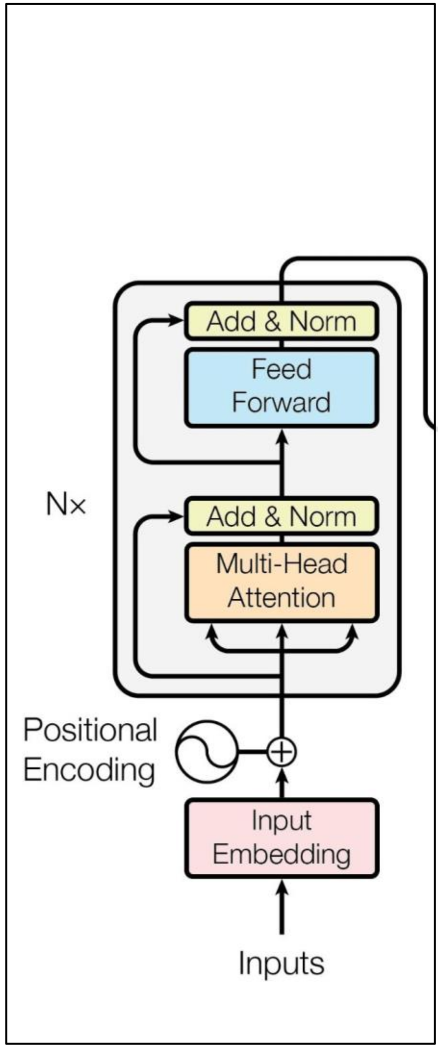
Output from Encoder_i



Transformers

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

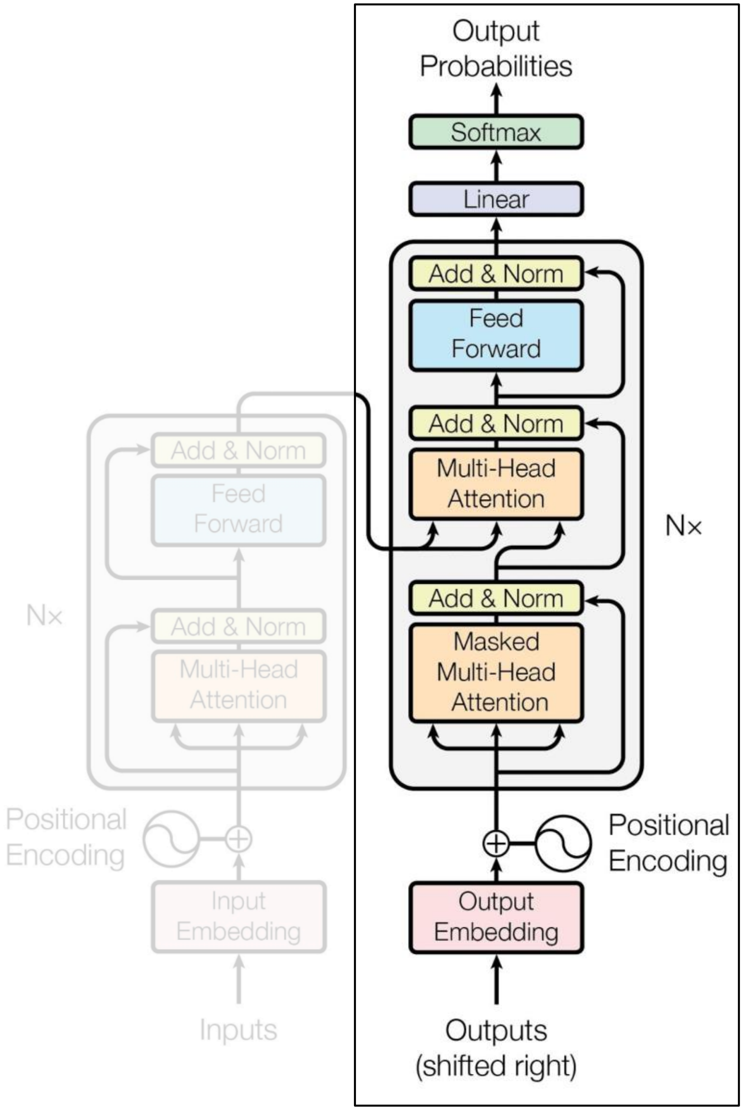
- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models



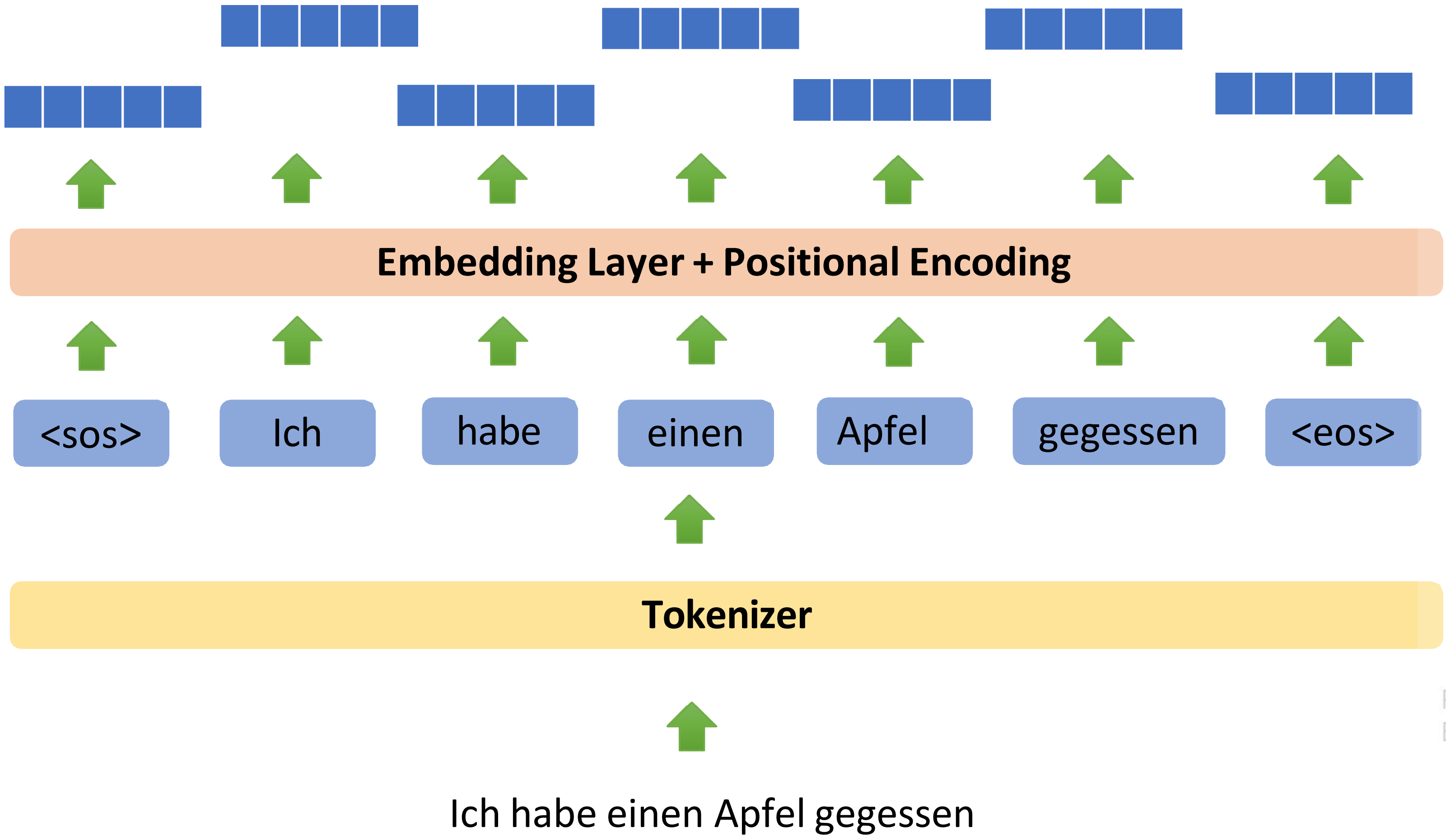
Targets

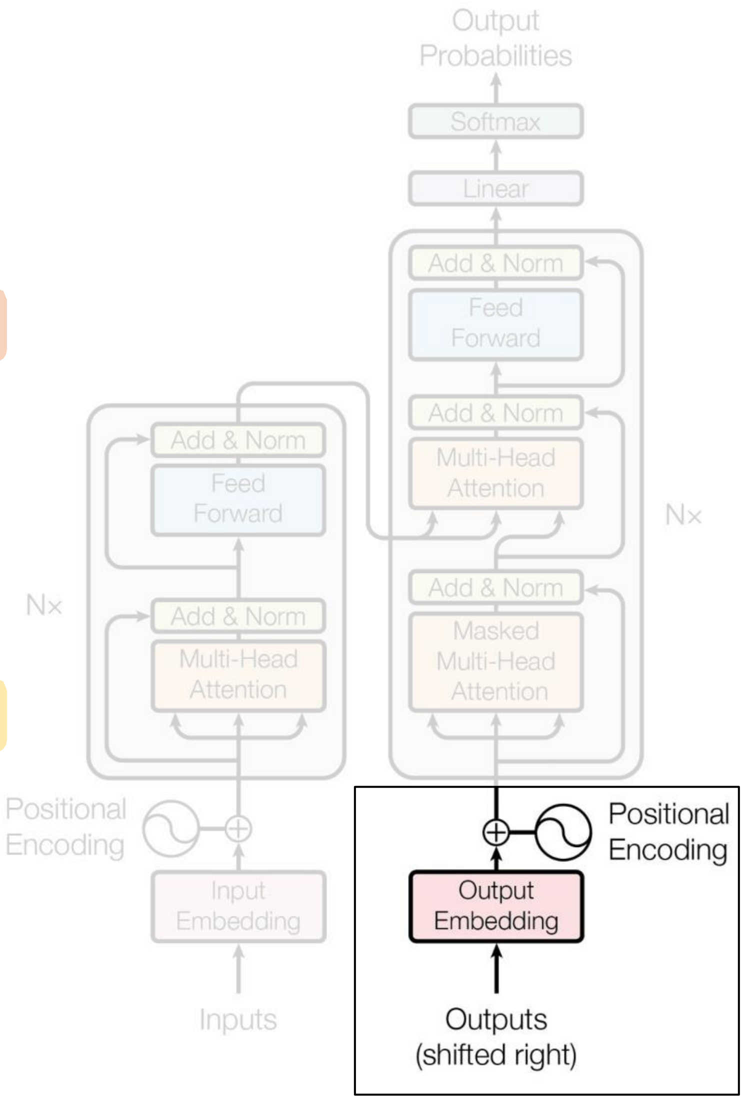
Targets

Ich habe einen Apfel gegessen



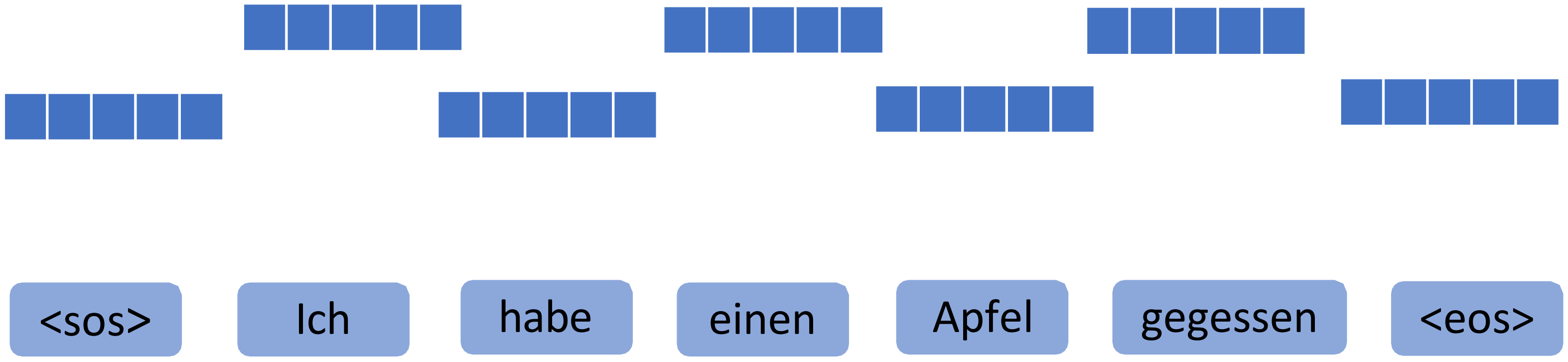
Targets

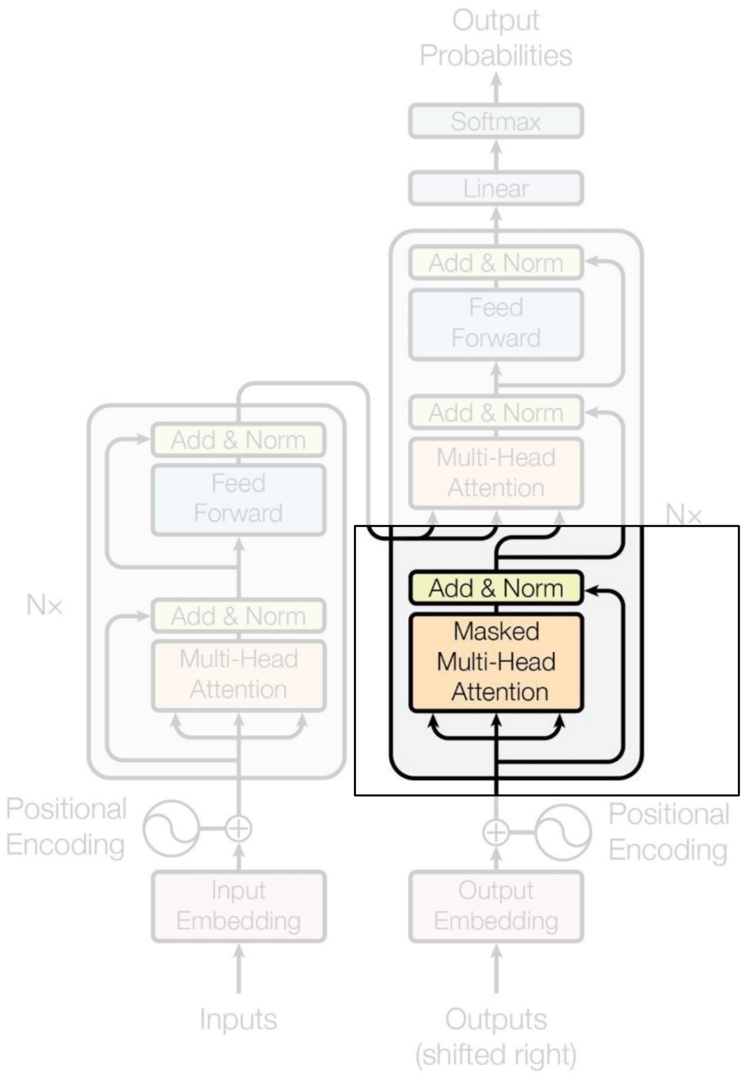




Generate Target Embeddings

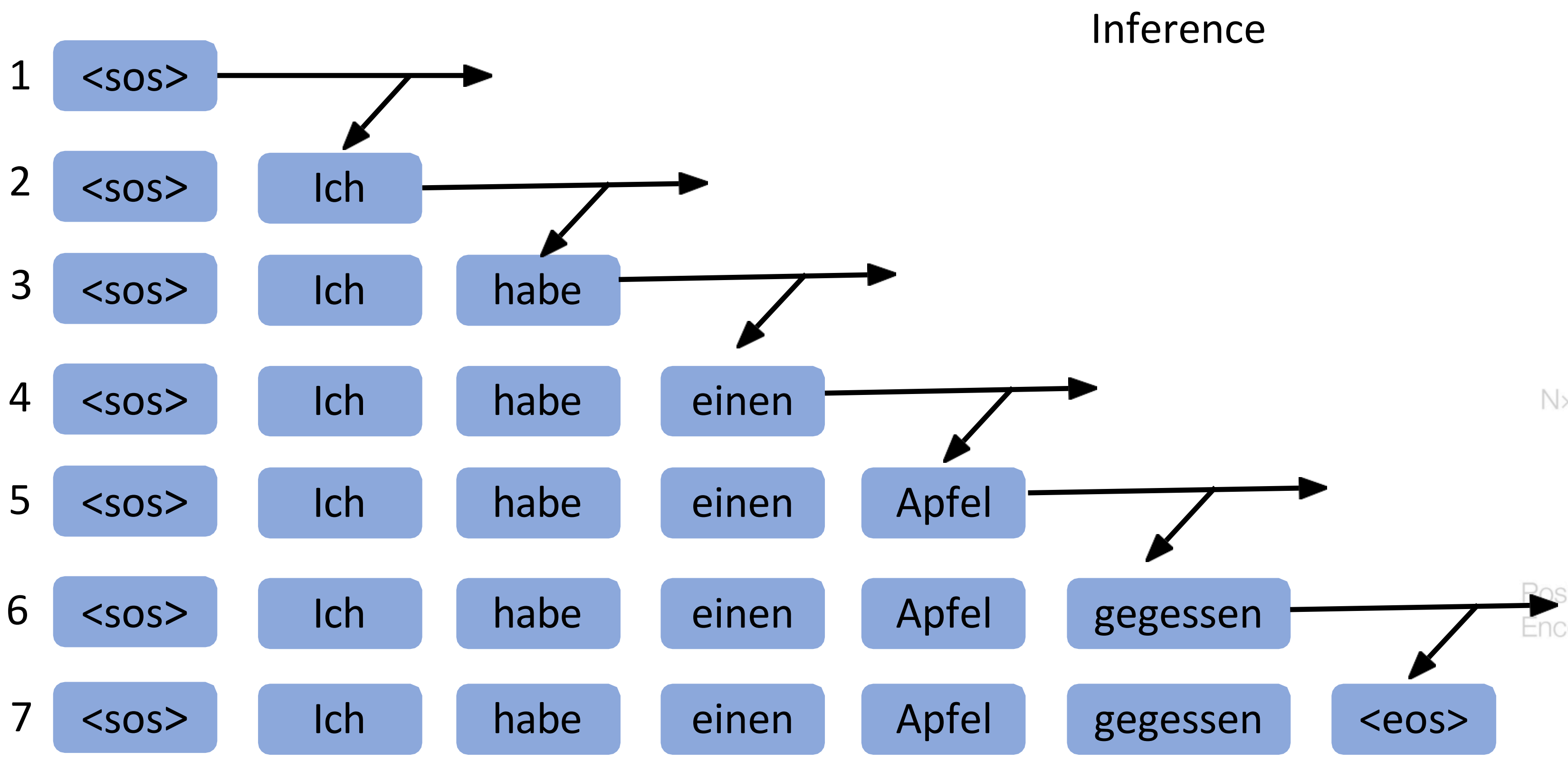
Masked Multi Head Attention

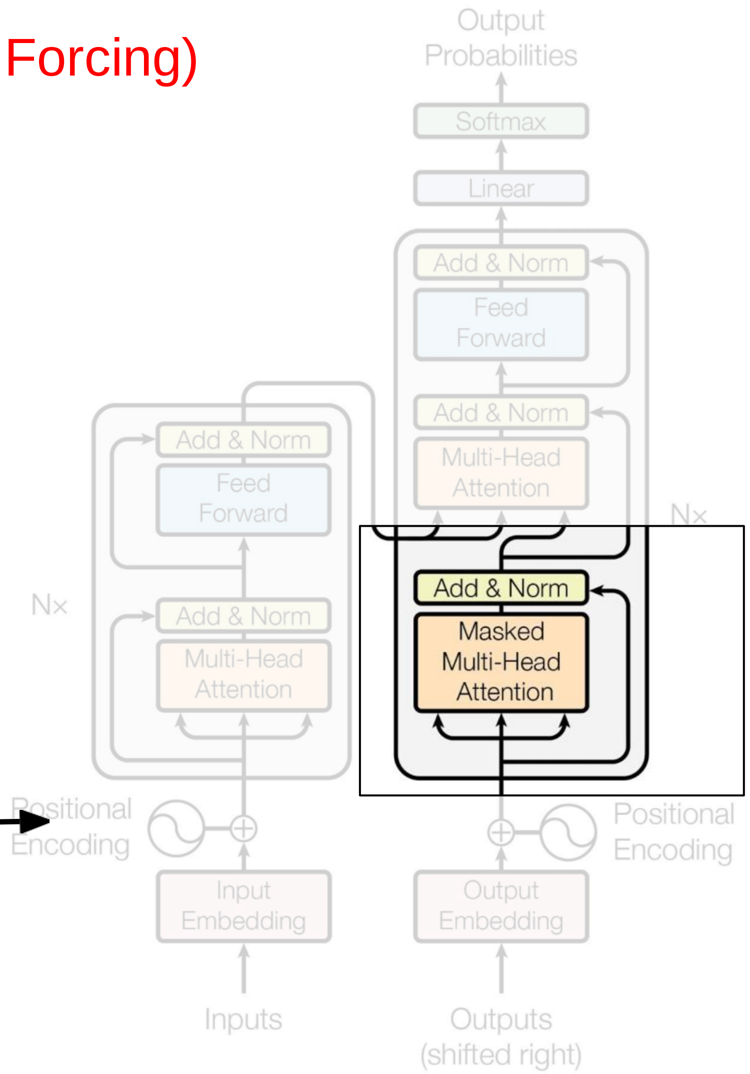




Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)



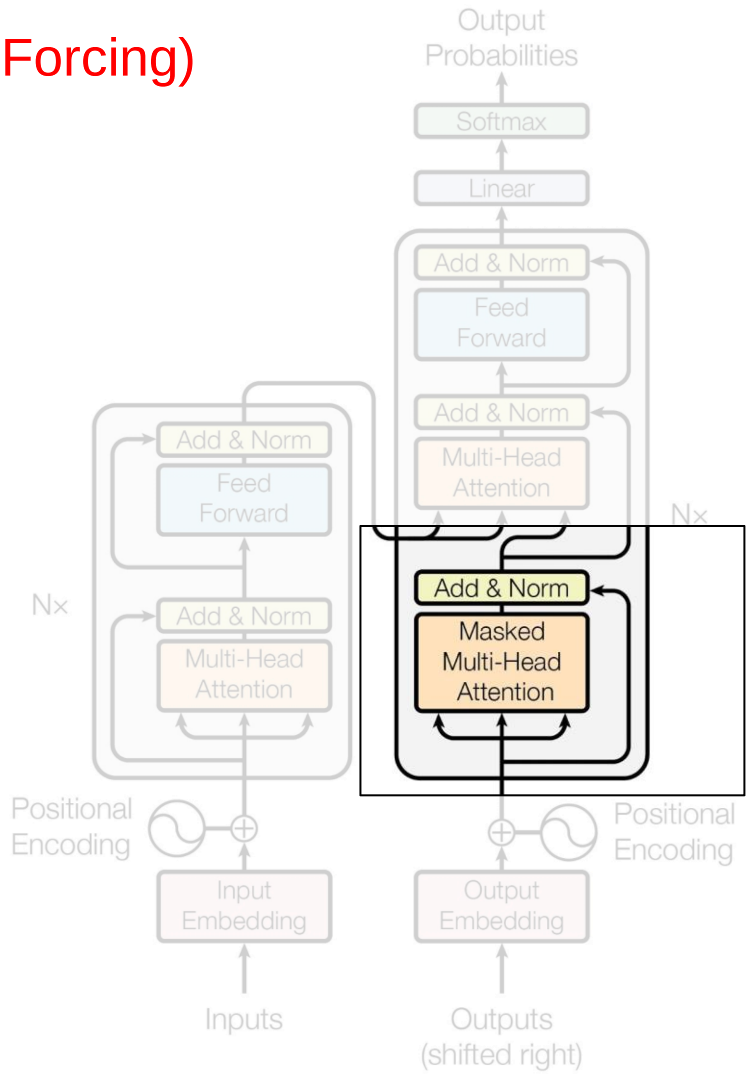


Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

Training

<sos> Ich habe einen Apfel gegessen <eos>



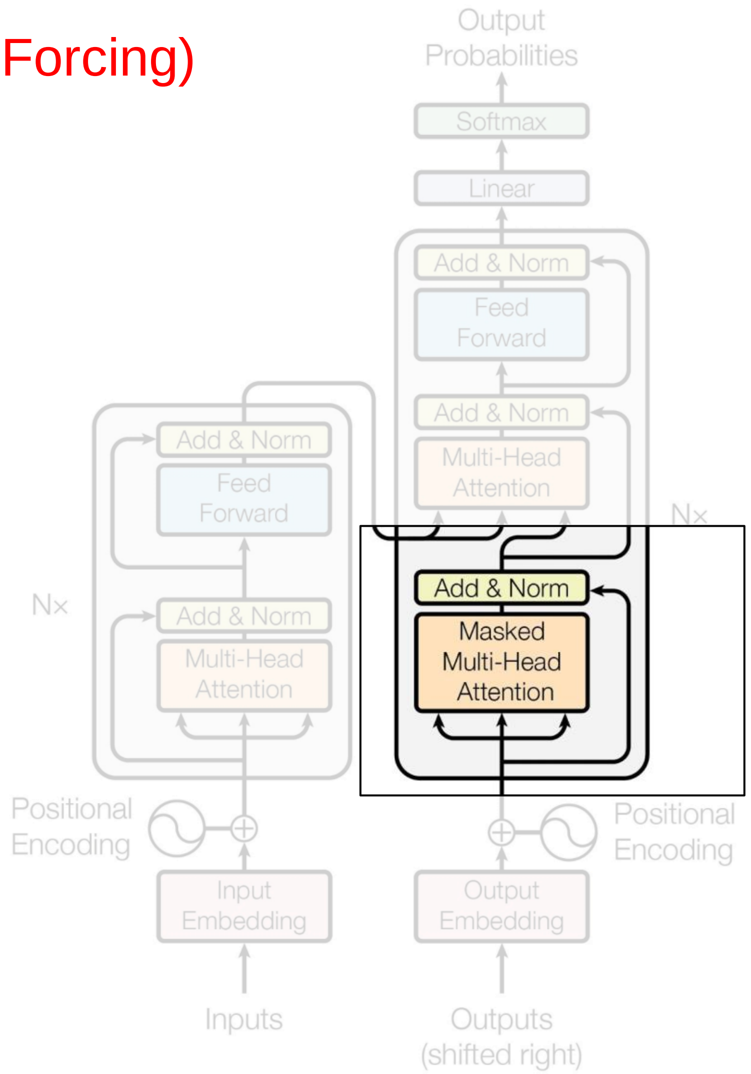
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

Training

<sos> | Ich | habe | einen | Apfel | gegessen | <eos>

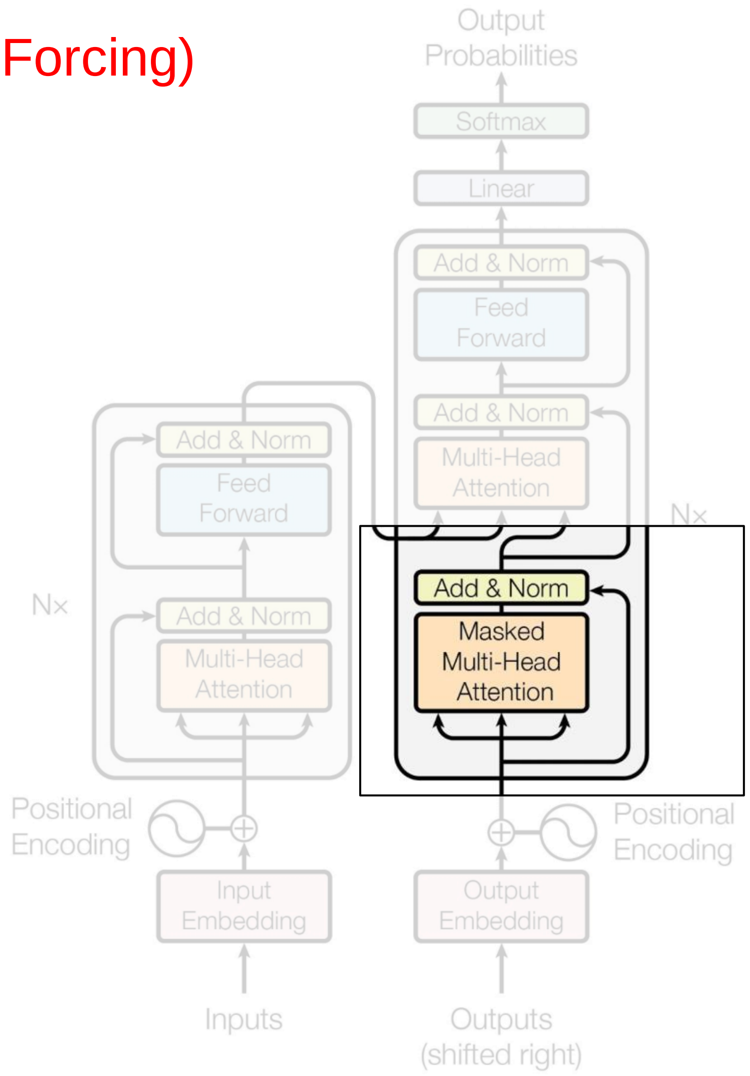
Outputs at time T should only pay attention to outputs until time T-1



Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

1	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
2	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
3	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
4	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
5	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
6	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
7	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>



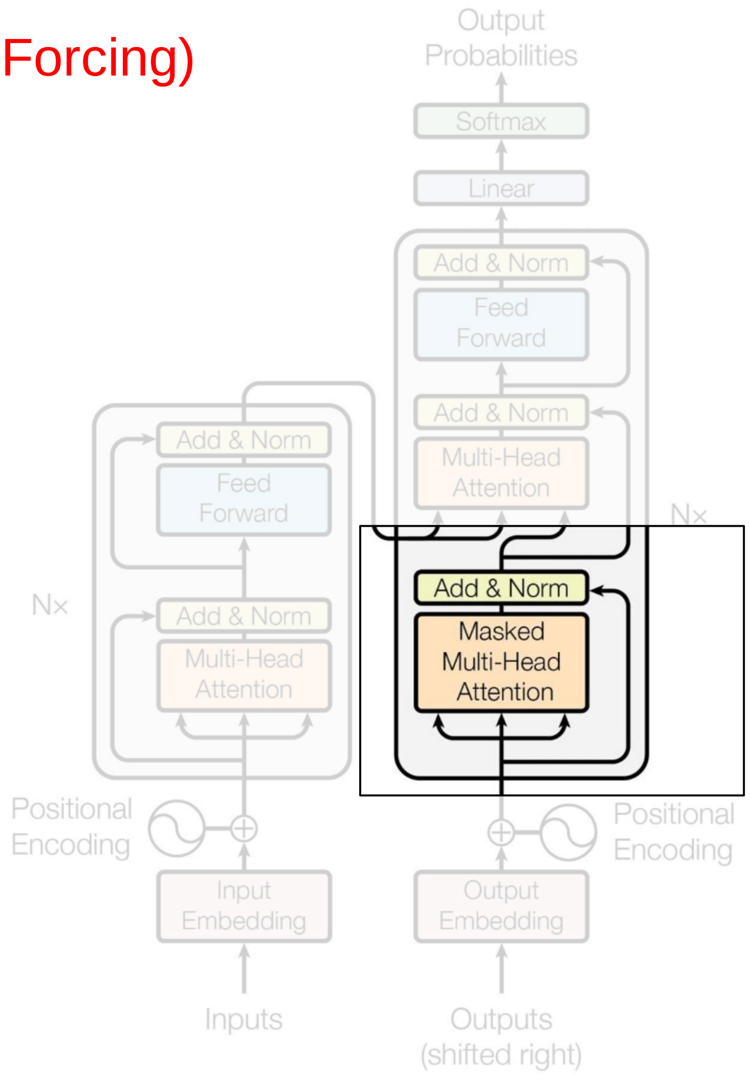
Mask the available attention values ?

Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

1	<sos>	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	<sos>	Ich	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
3	<sos>	Ich	habe	$-\infty$	$-\infty$	$-\infty$	$-\infty$
4	<sos>	Ich	habe	einen	$-\infty$	$-\infty$	$-\infty$
5	<sos>	Ich	habe	einen	Apfel	$-\infty$	$-\infty$
6	<sos>	Ich	habe	einen	Apfel	gegessen	$-\infty$
7	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>

Softmax $-\infty$ -> 0

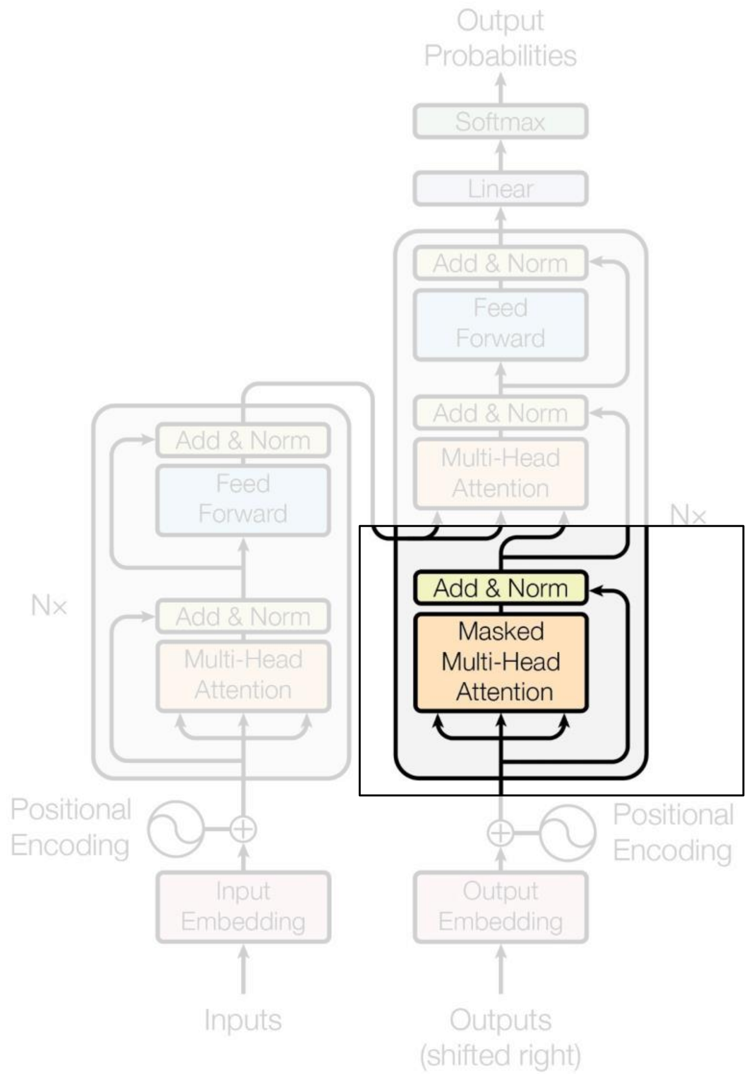


Masked Multi Head Attention

Masked Multi Head Attention

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

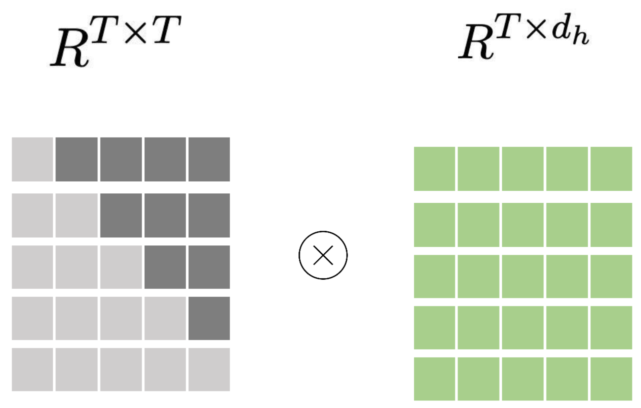
QK^T $\quad$ Attention Mask: M $\quad$ Masked Attention



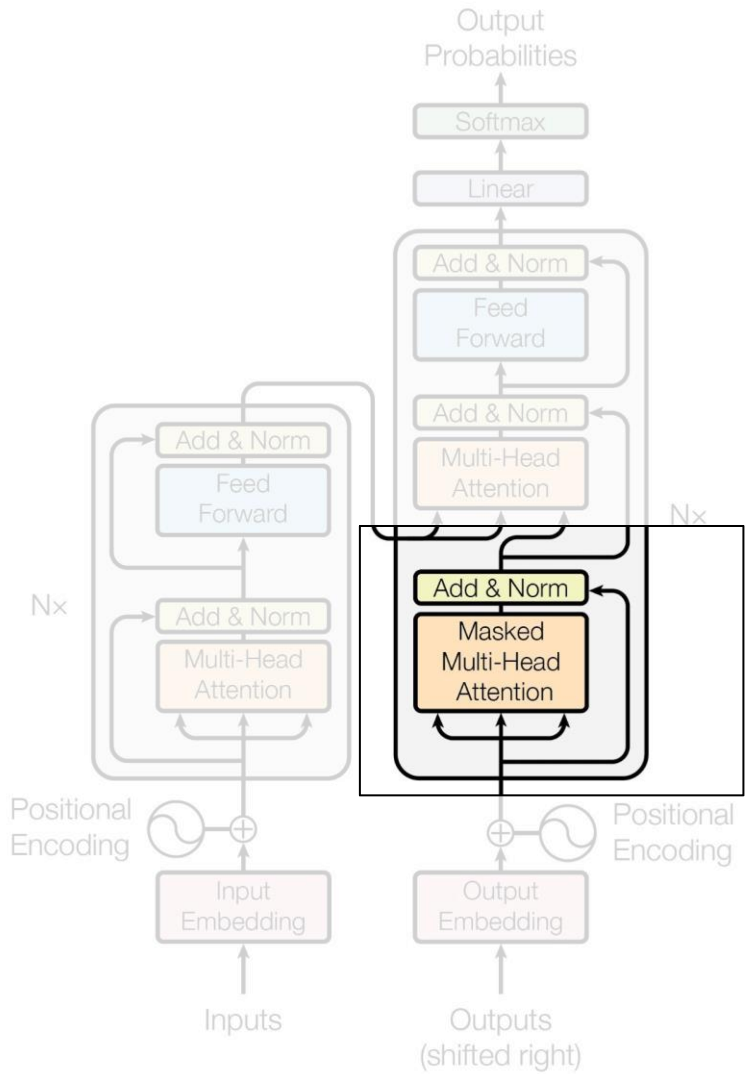
Masked Multi Head Attention : Z’

Masked Multi Head Attention

Masked Multi Head Attention



Masked Attention Values

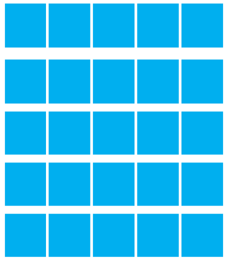


Masked Multi Head Attention : Z’

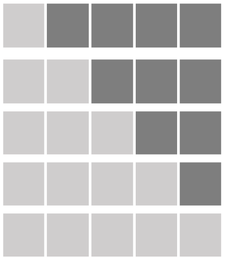
Encoder Decoder Attention

Encoder Decoder Attention ?

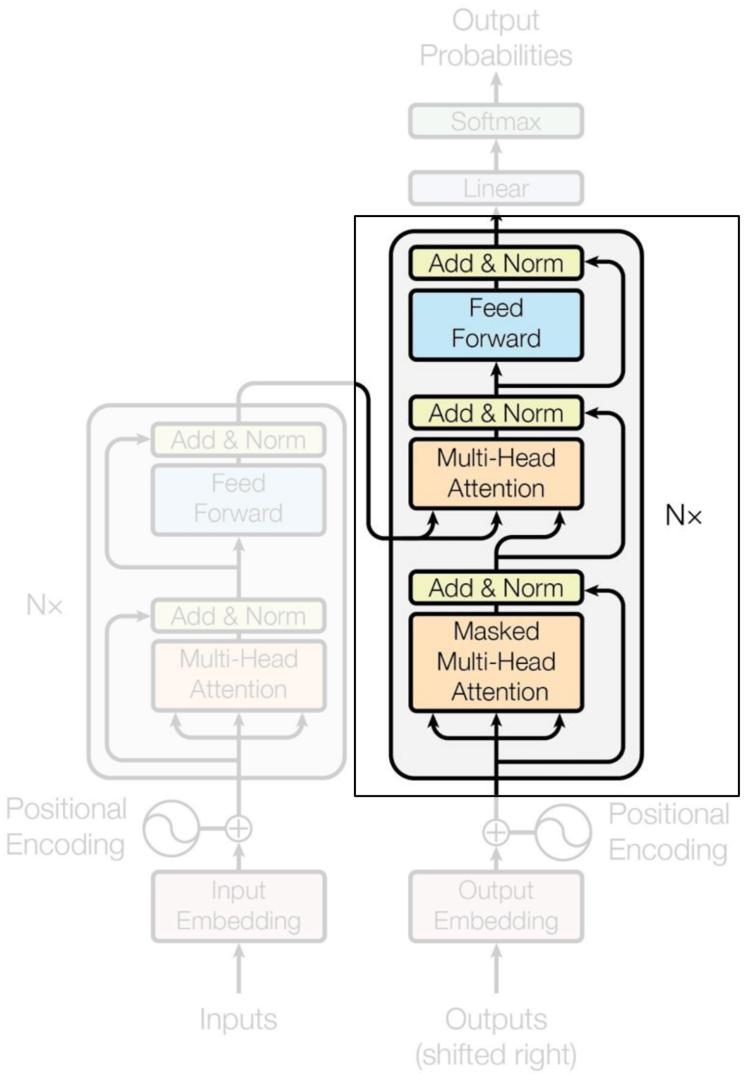
Add & Norm



Input



Norm(Z')



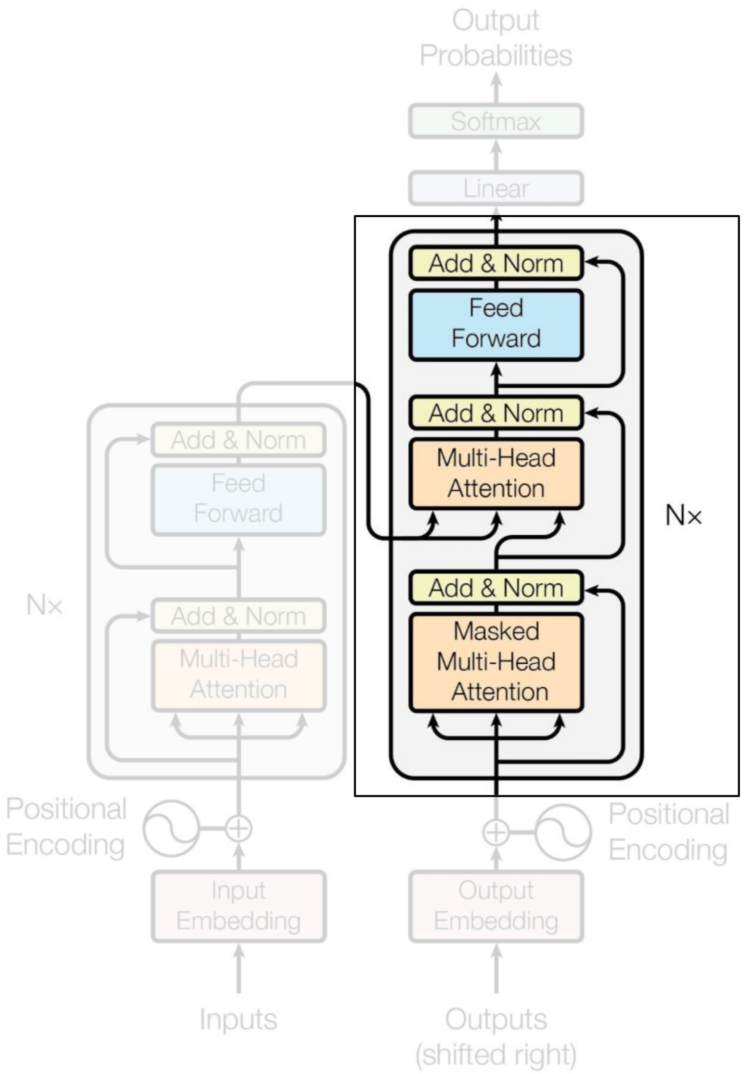
Encoder Decoder Attention

Encoder

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

Decoder

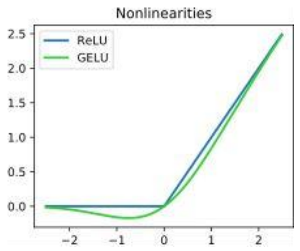
Queries from **Decoder Inputs**

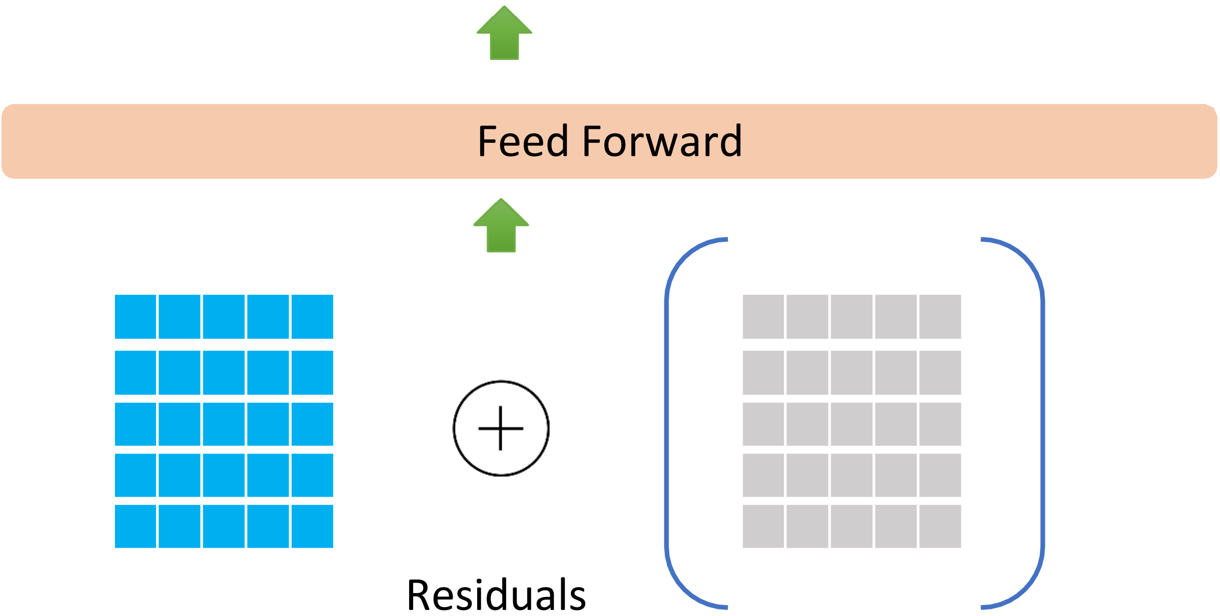


NOTE: Every decoder block receives the same FINAL encoder output

Encoder Decoder Attention

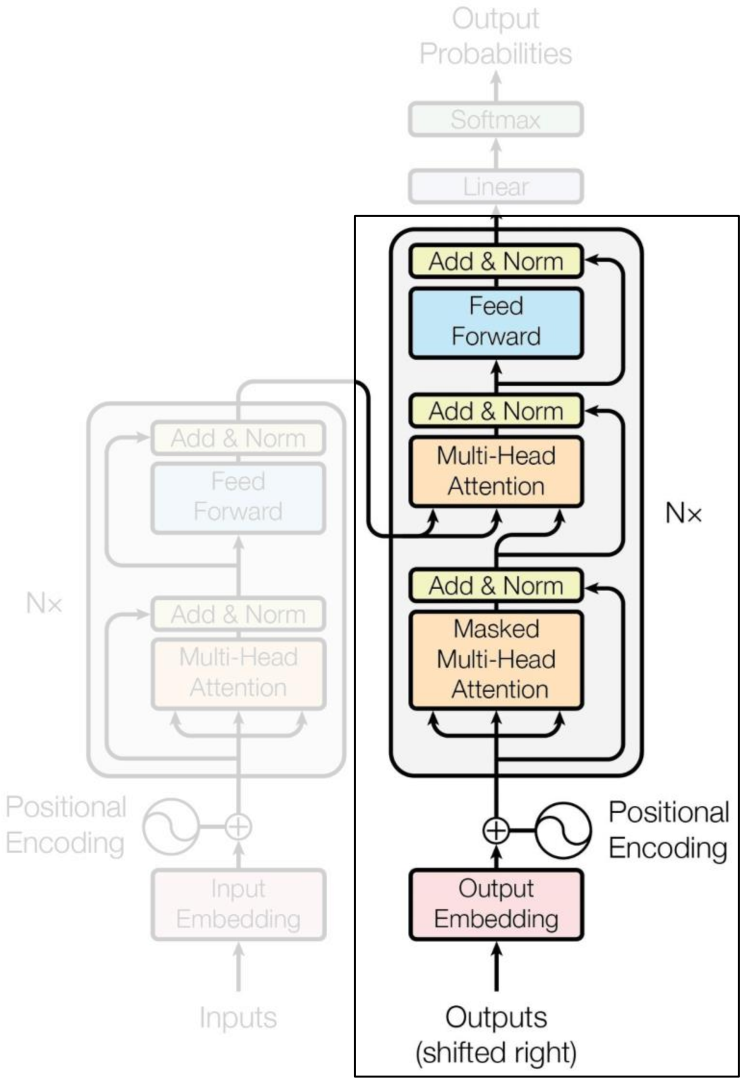
- Non Linearity
- Complex Relationships
- Learn from each other



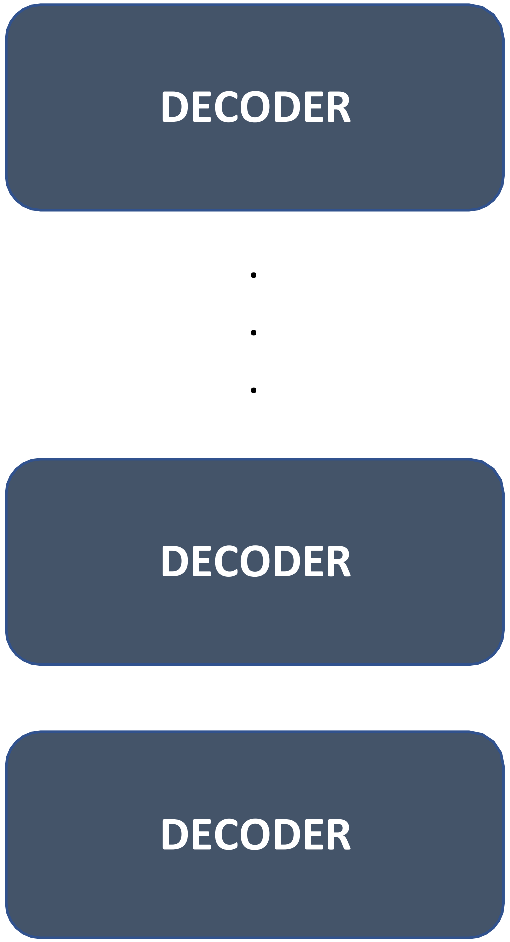


Add n Norm Decoder Self Attn

Norm(Z'')

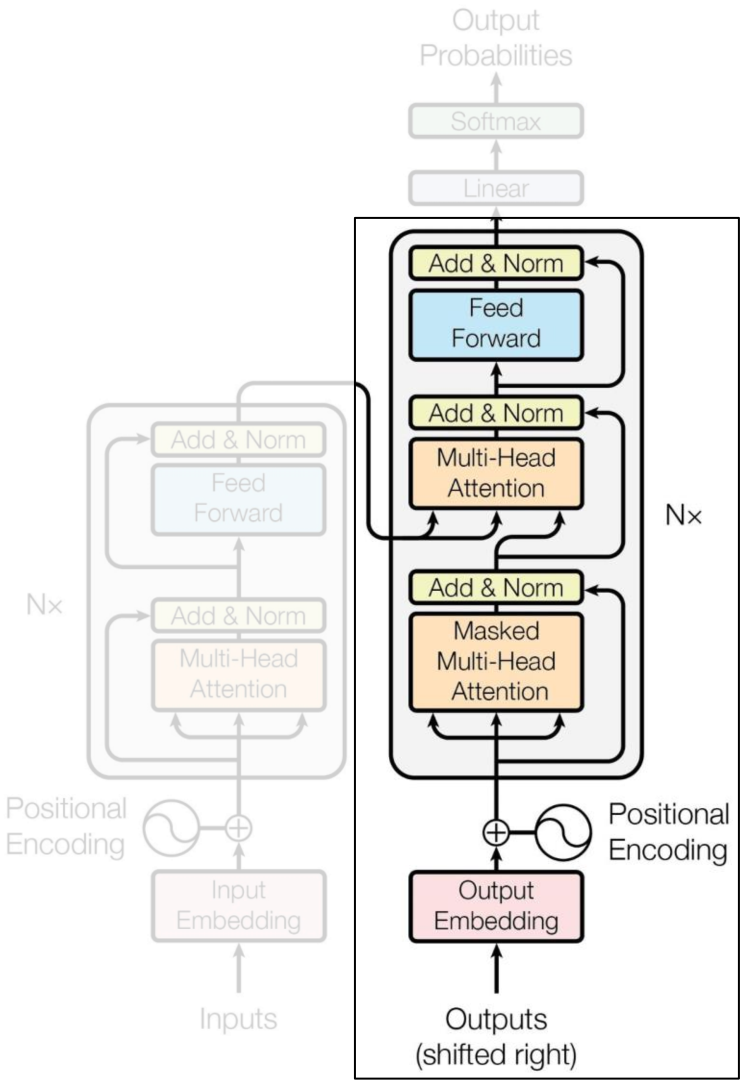


Decoder



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

Decoder output

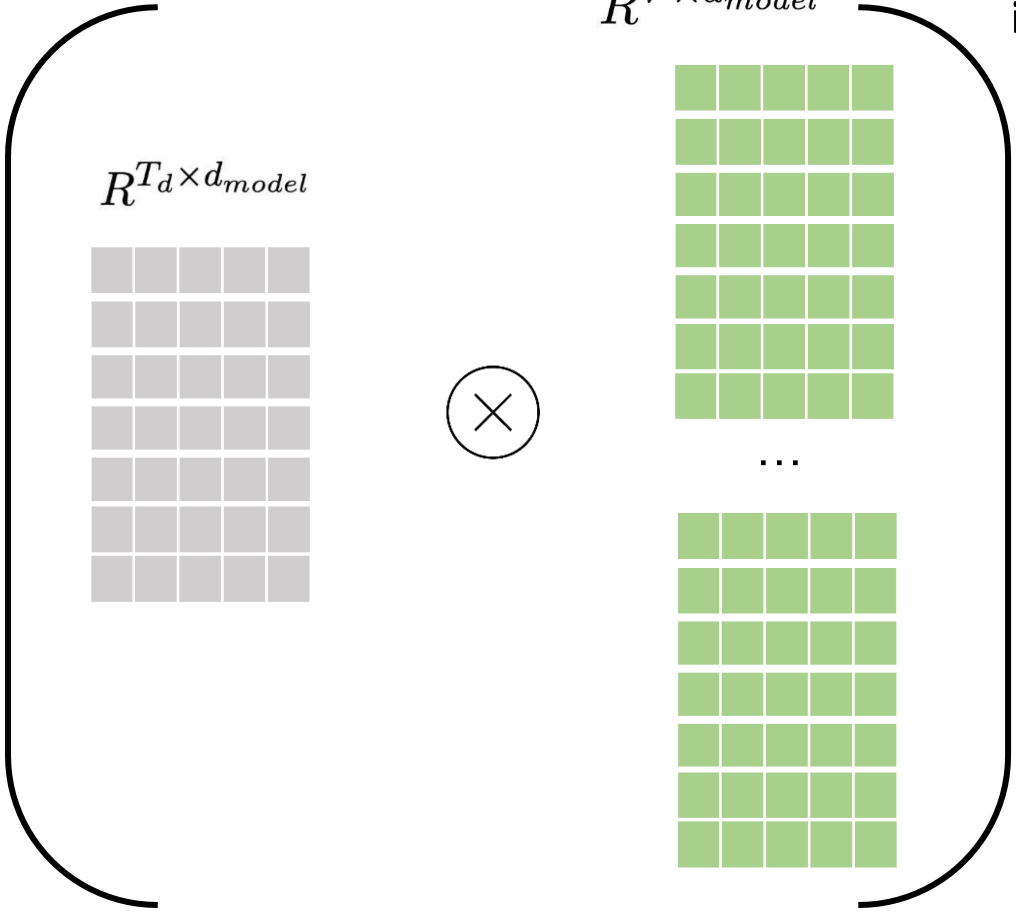


Linear

Linear weights are often tied with input embedding matrix

softmax

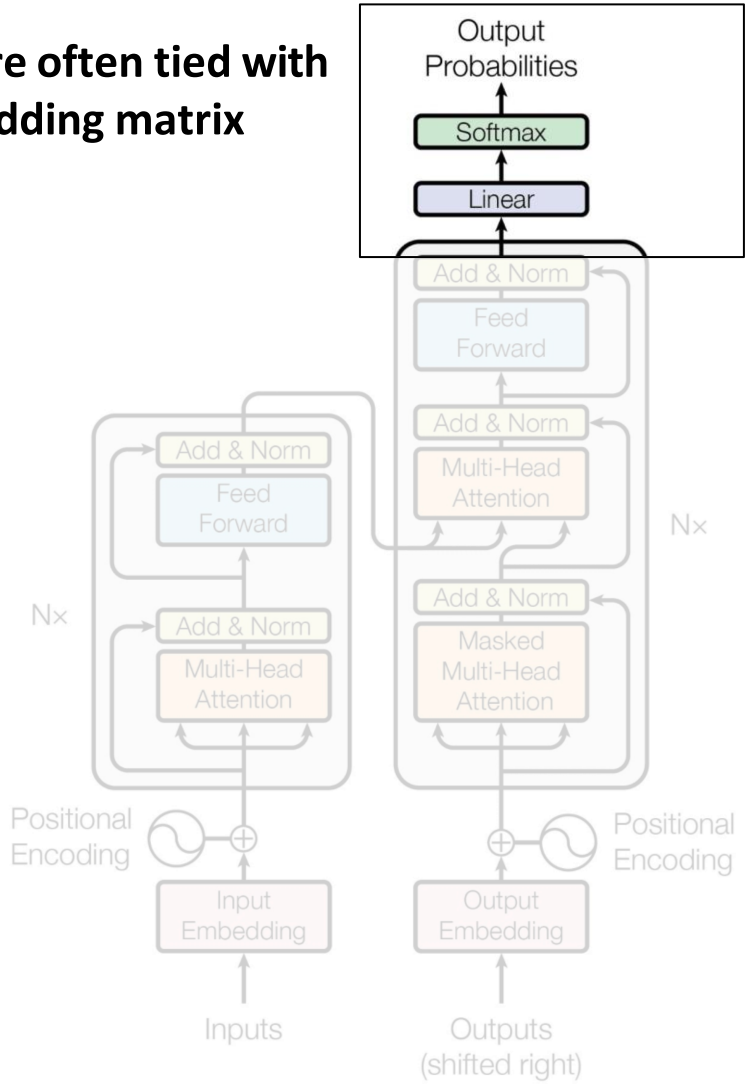
$R^{T_d \times d_{model}}$ $\otimes$ $R^{V \times d_{model}}$



Final Decoder Output

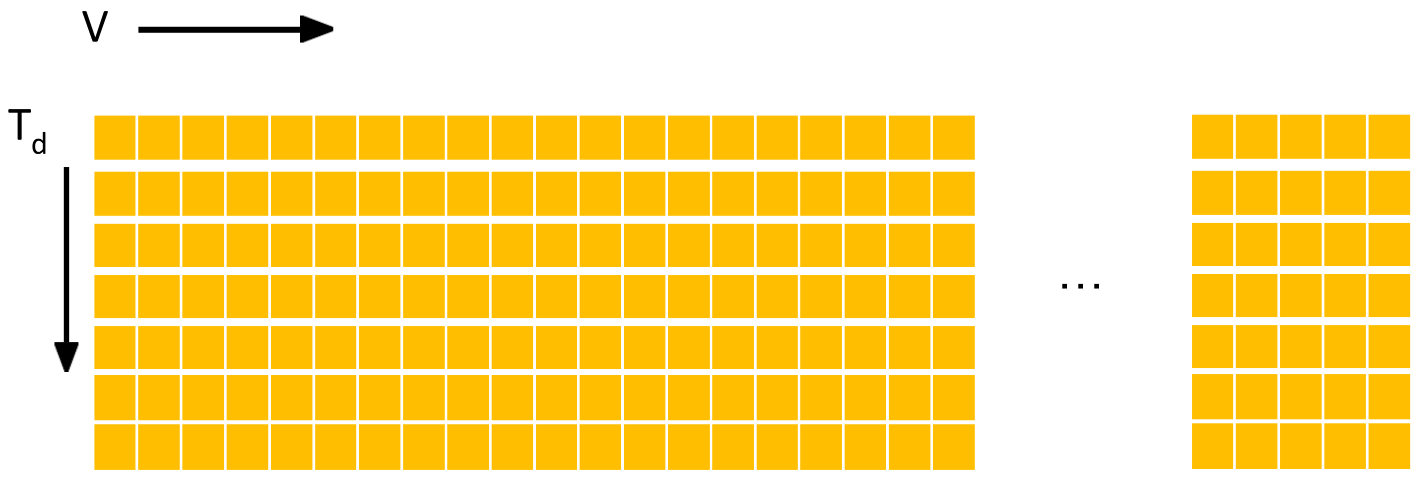
$R^{T_d \times V}$

Linear

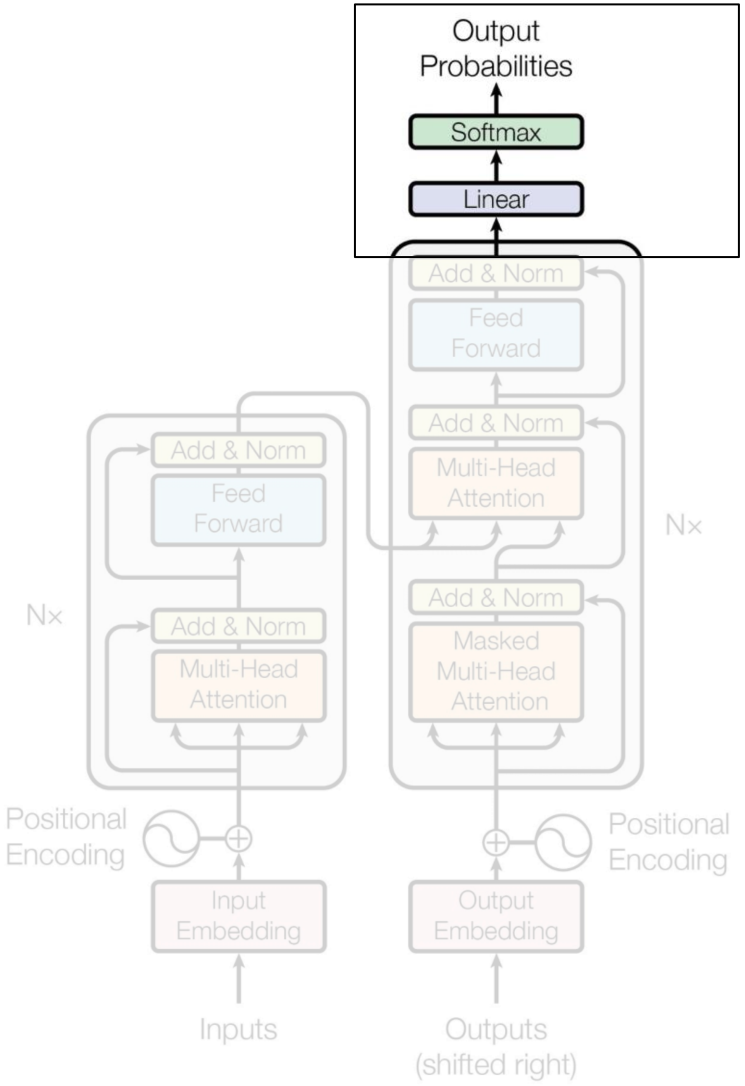


Softmax

Output Probabilities

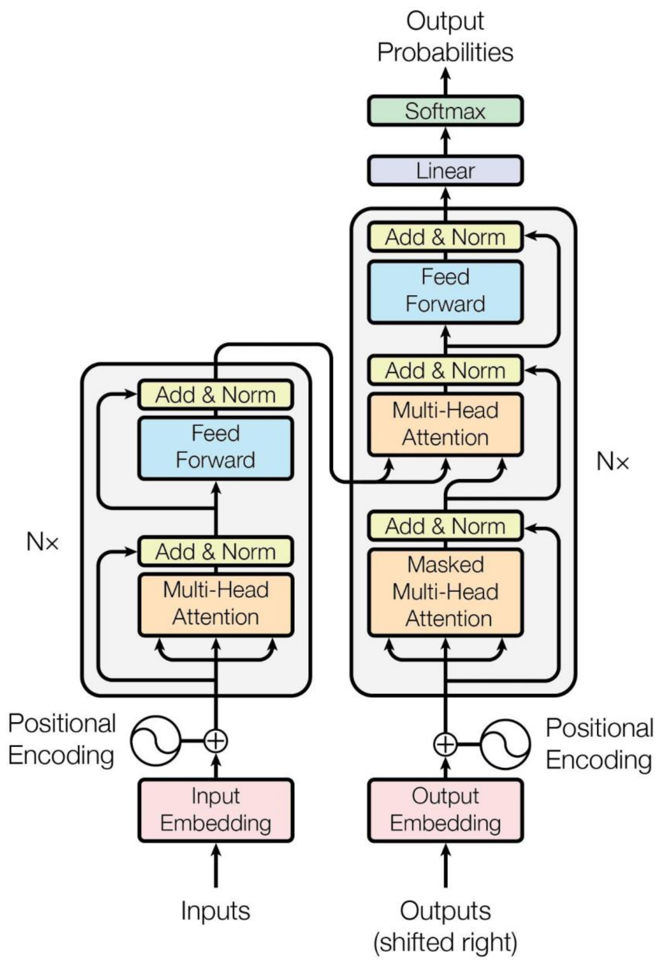


$$R^{T_d \times V}$$

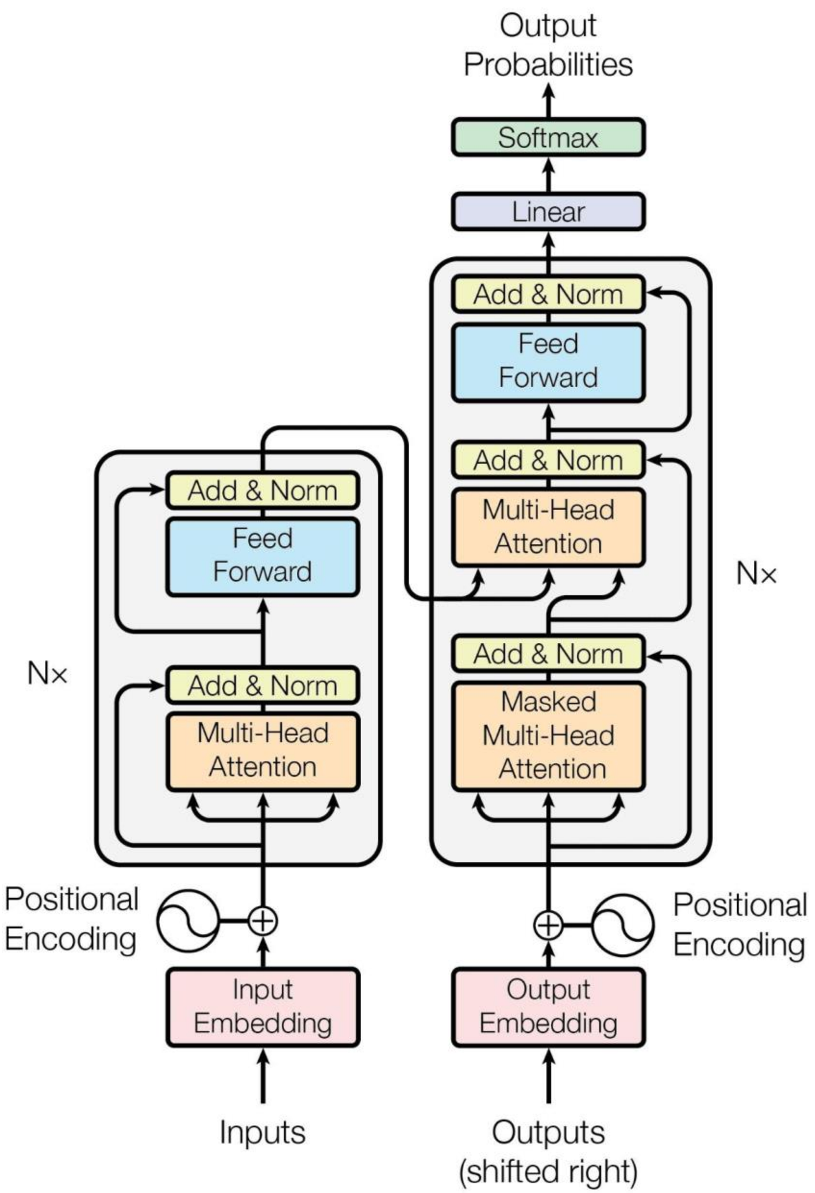


Transformers

- ✔ 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

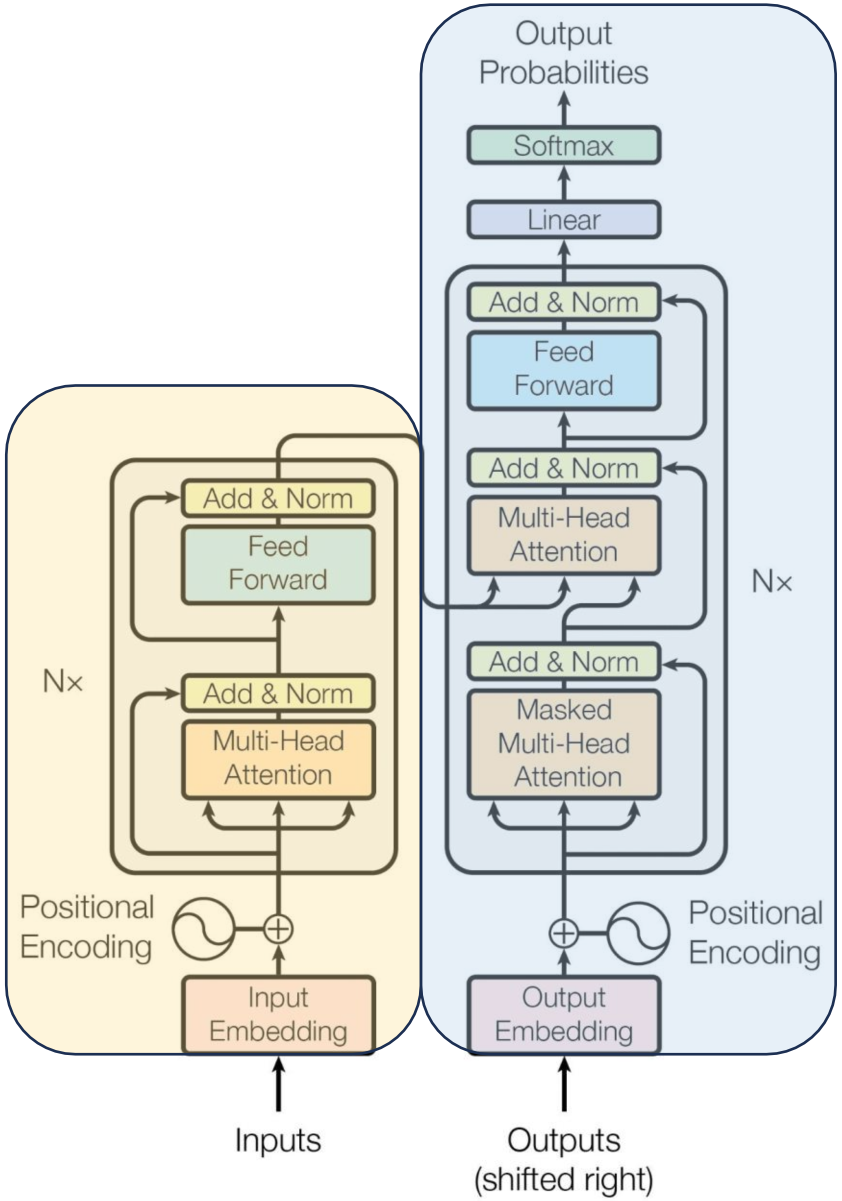


Transformers



Transformers

Representation

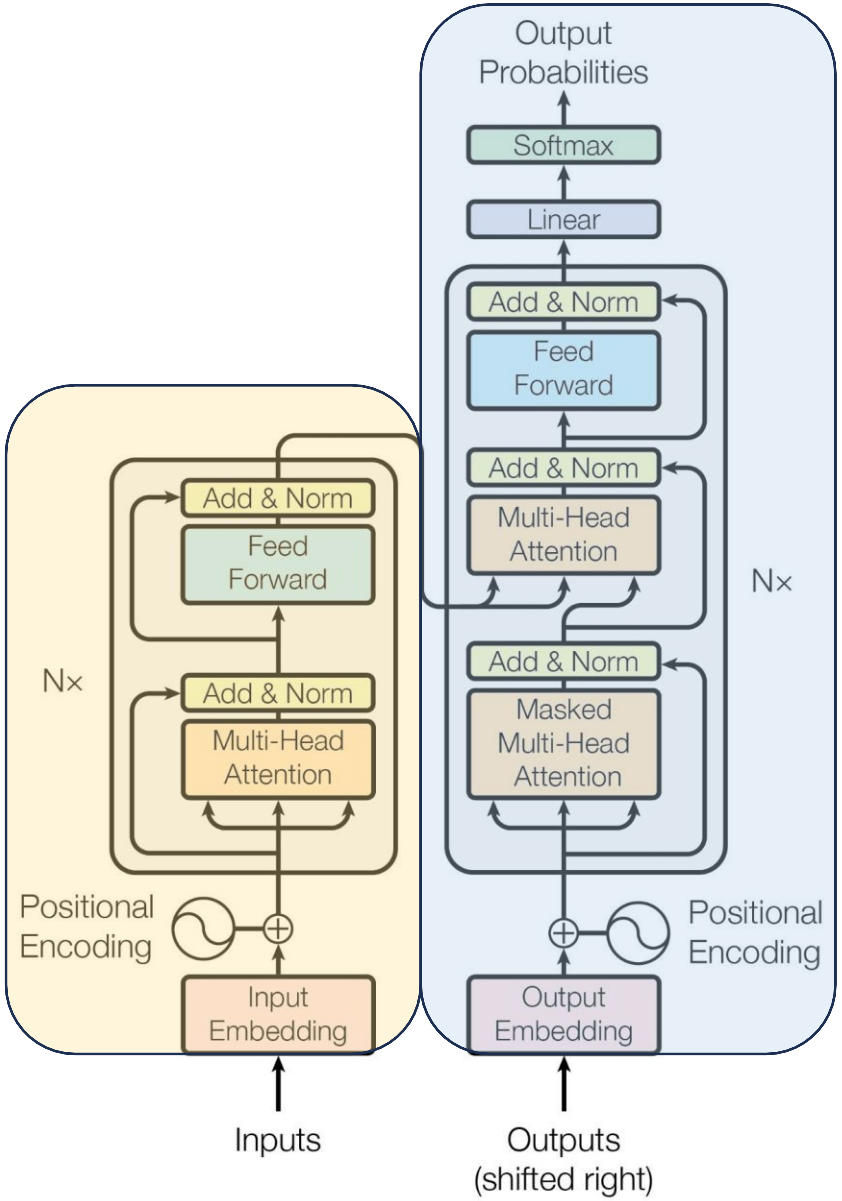


Generation

Transformers

Input – input tokens
Output – hidden states

Representation



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

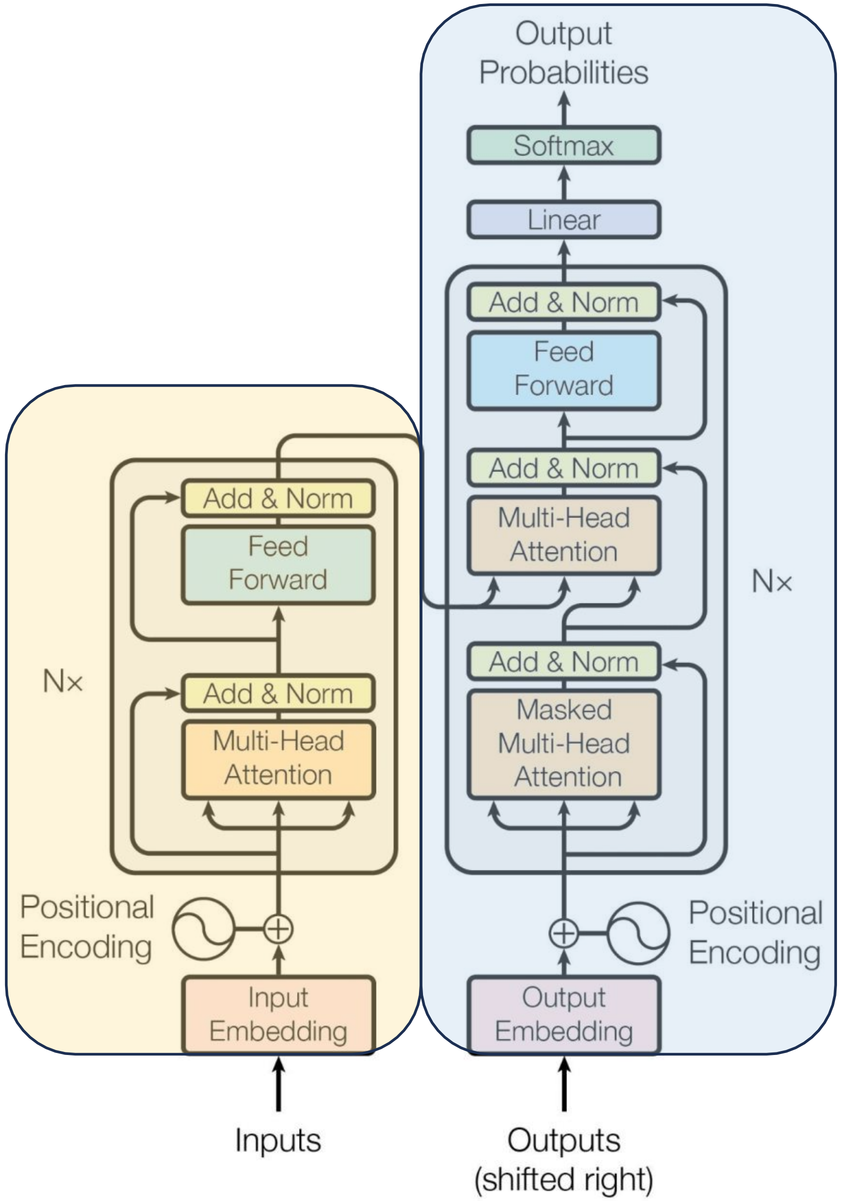
Generation

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

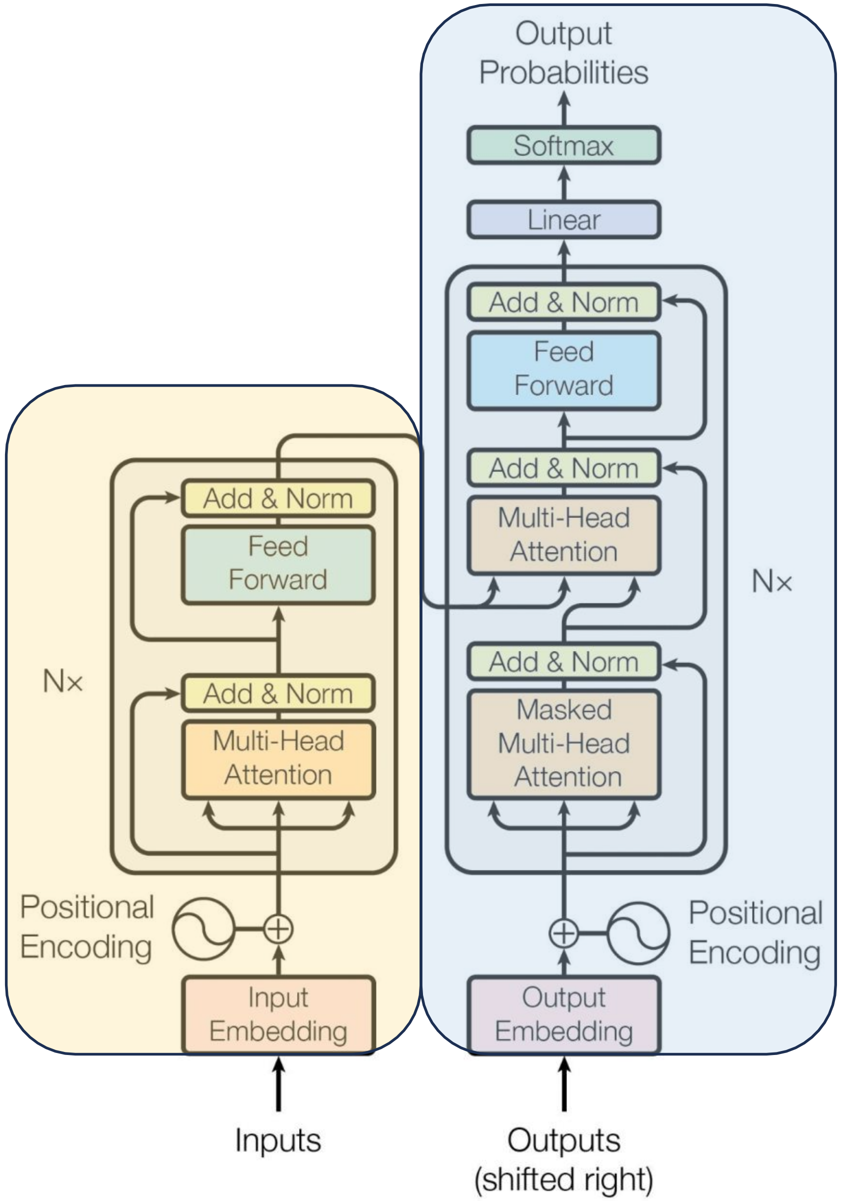
Transformers

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

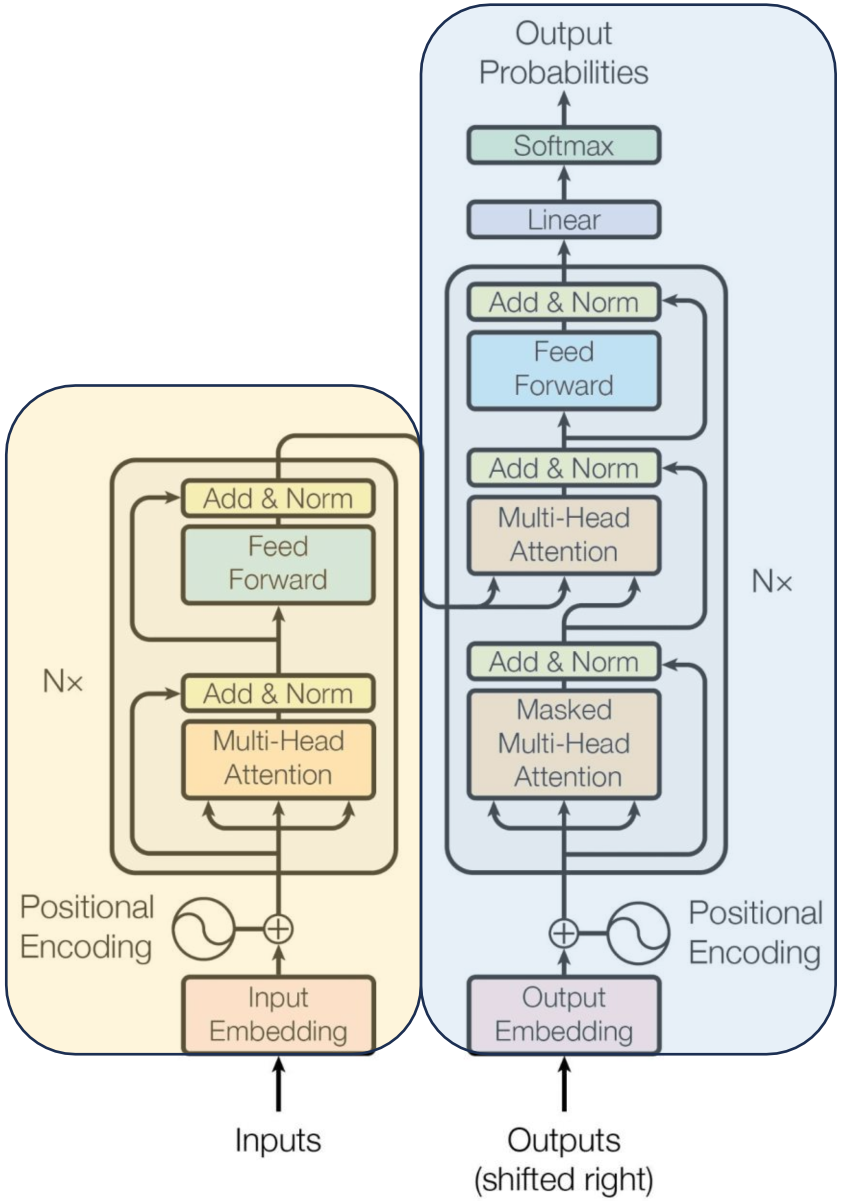
Input – input tokens
Output – hidden states

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

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