Introduction to Deep Learning

Lecture 19
Transformers

Liangze Li

Kateryna Shapovalenko

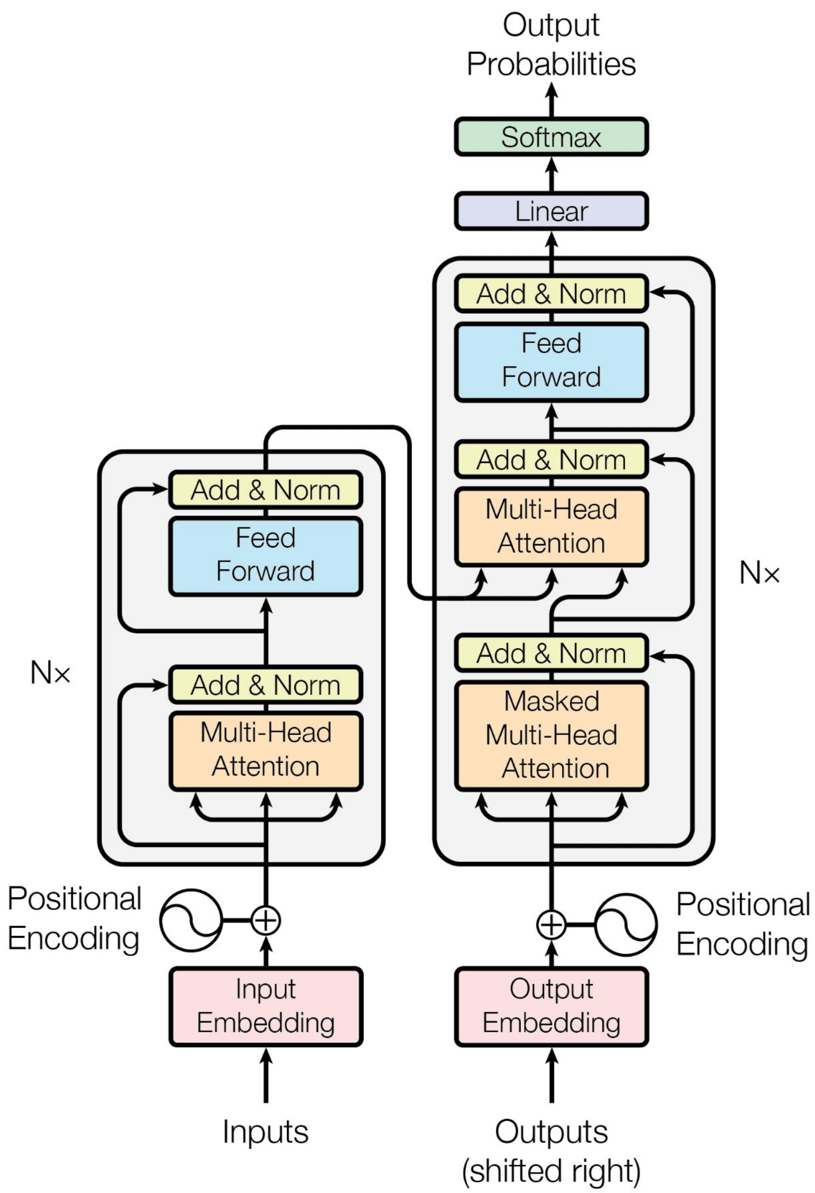
11-785, Spring 2024

Attendance poll @1585

Table of contents

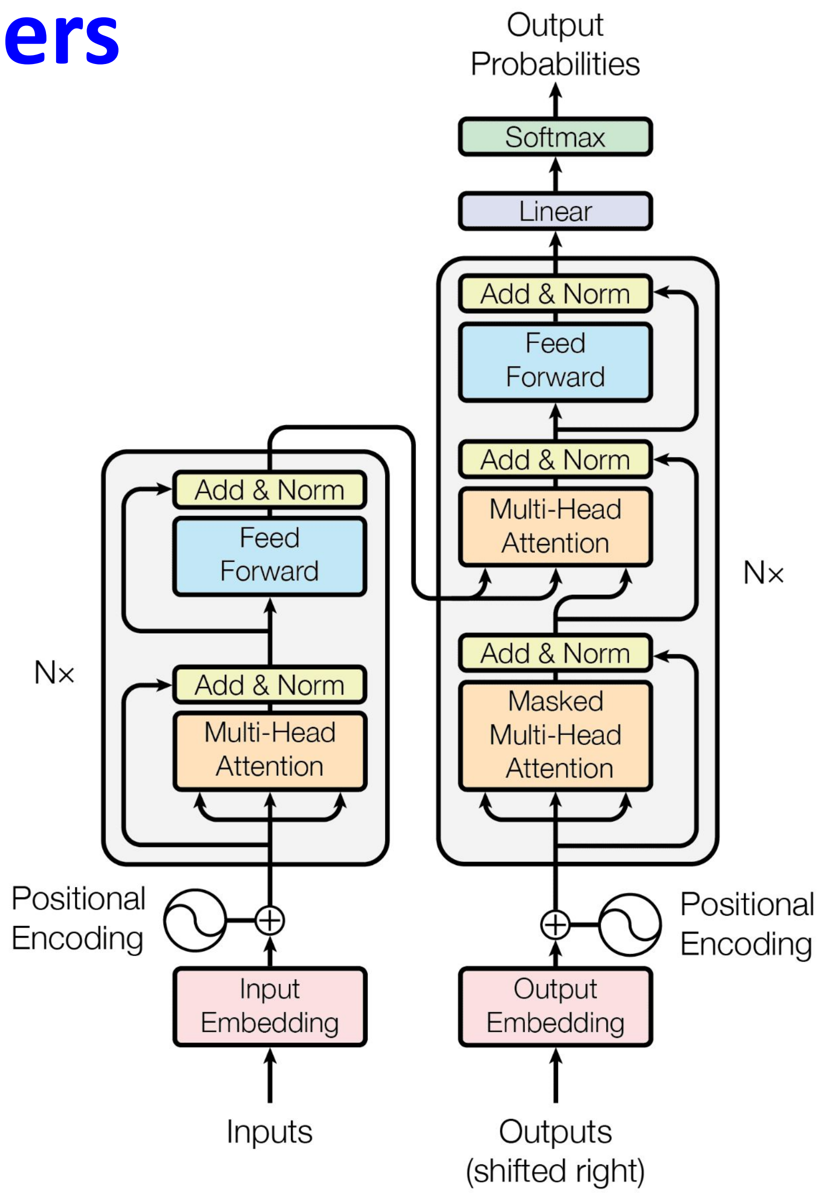
Part 1

Transformer Architecture



Transformers

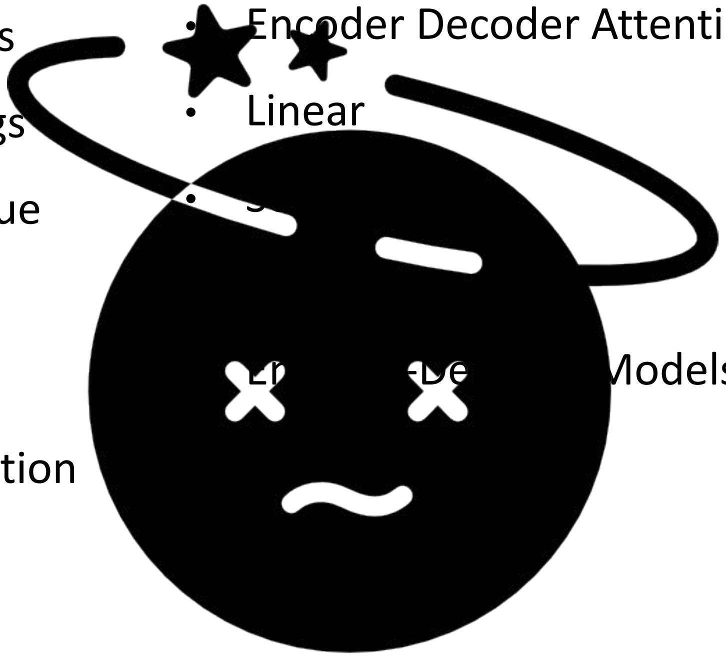
- Tokenization
- Input Embeddings
- Position Encodings
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders
- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models

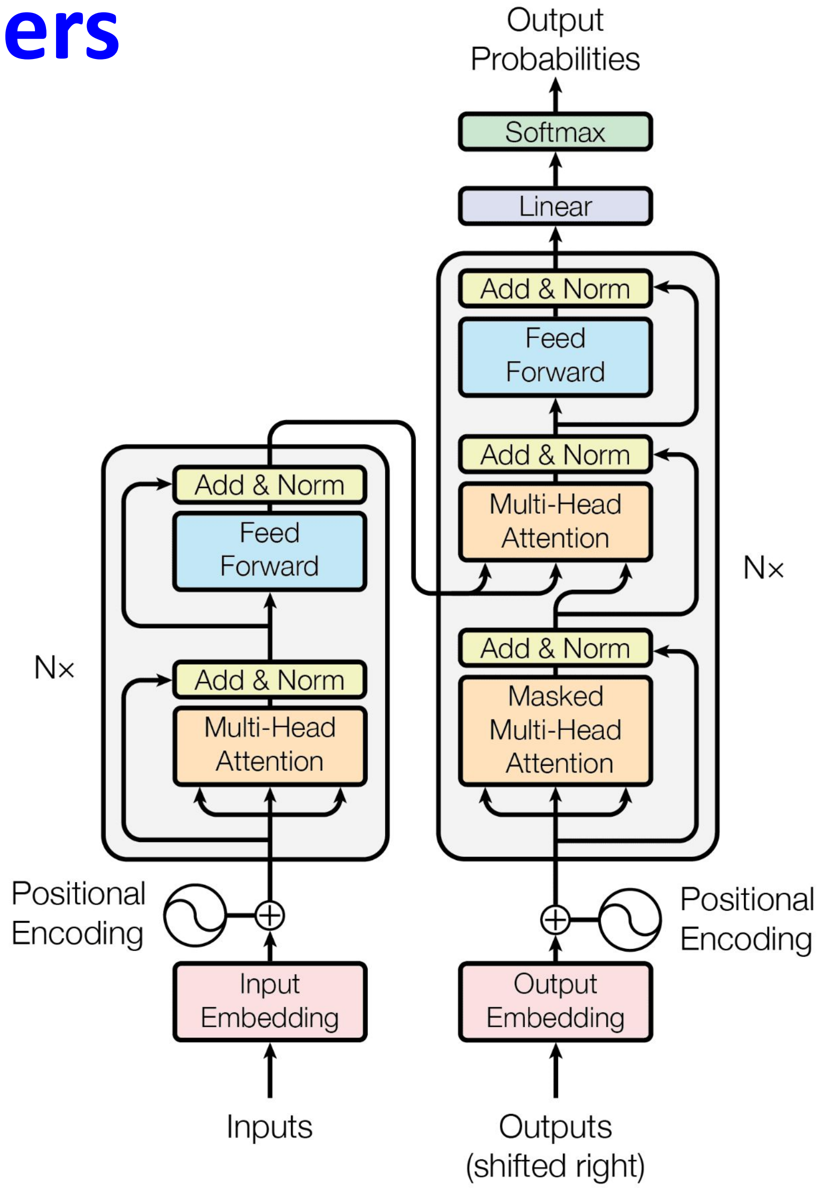


Transformers

- 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
- Models





Machine Translation

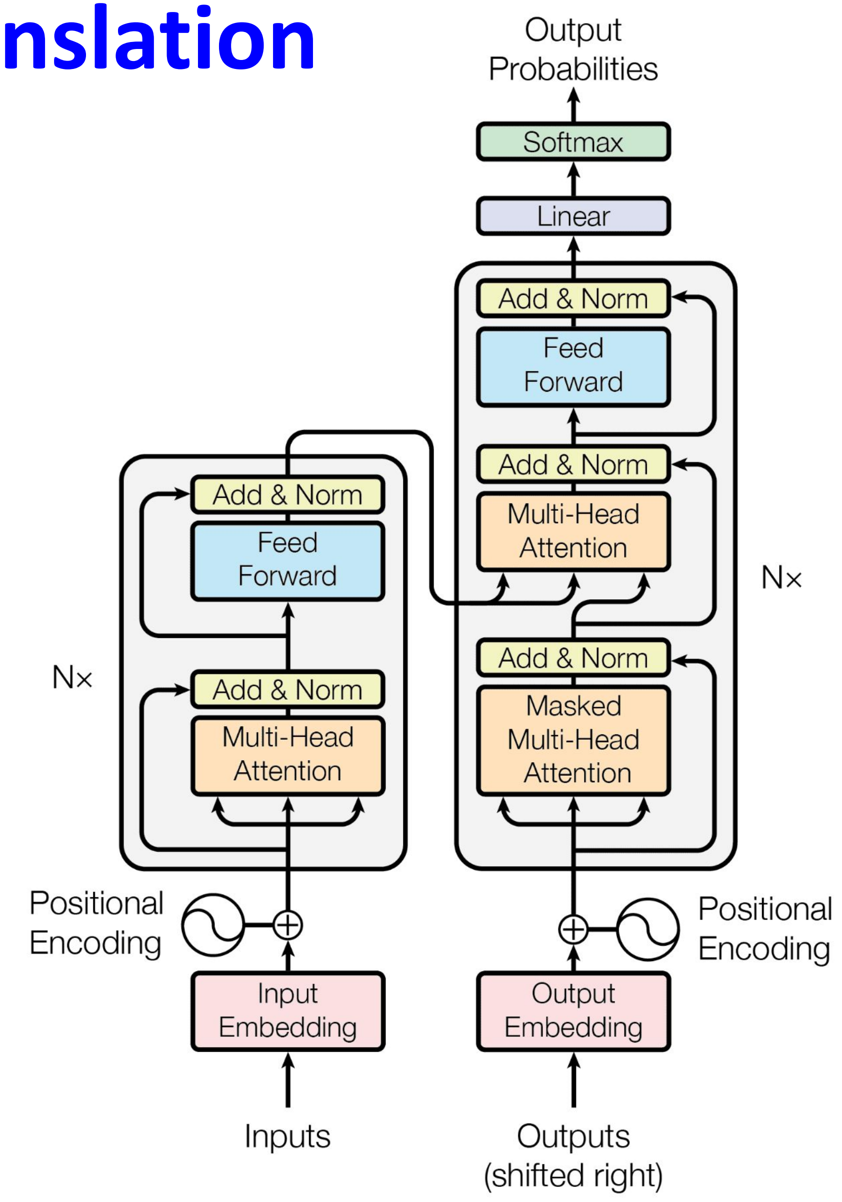
Targets

Ich habe einen Apfel gegessen

⬆

Inputs

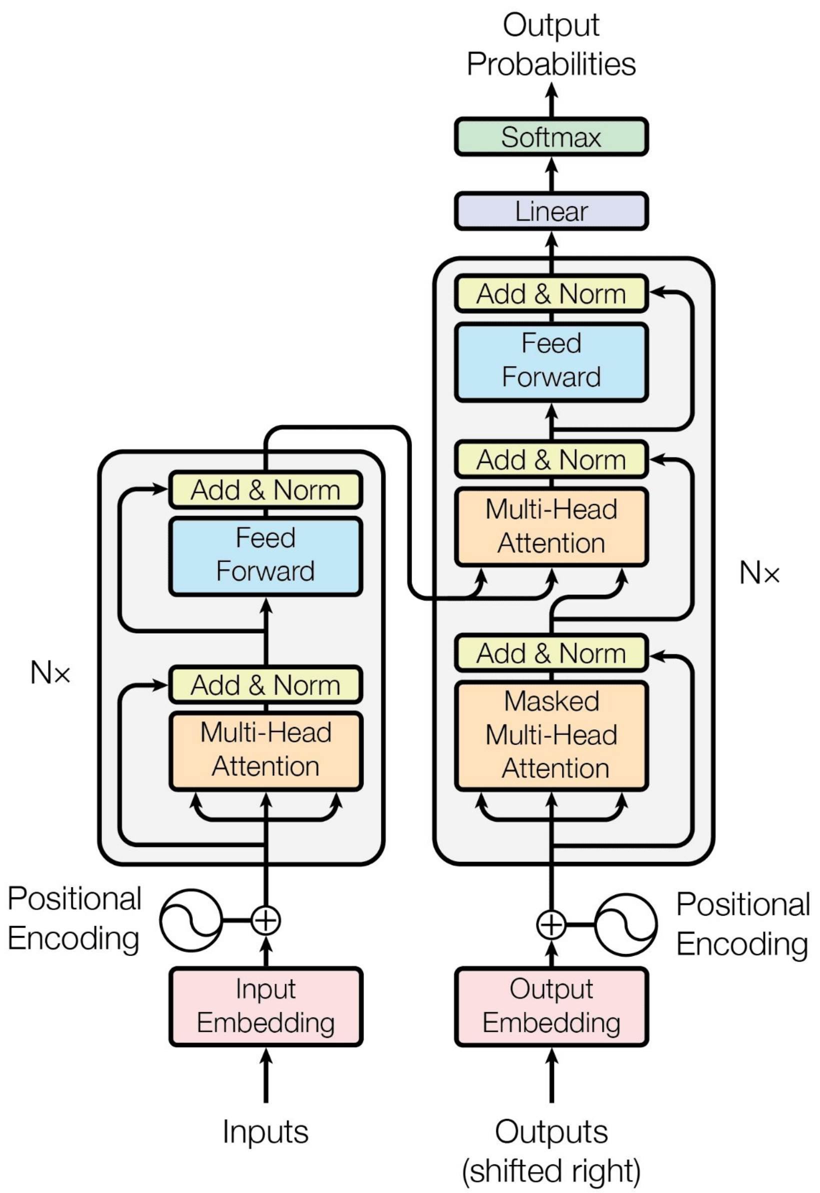
I ate an apple



Inputs

Processing Inputs

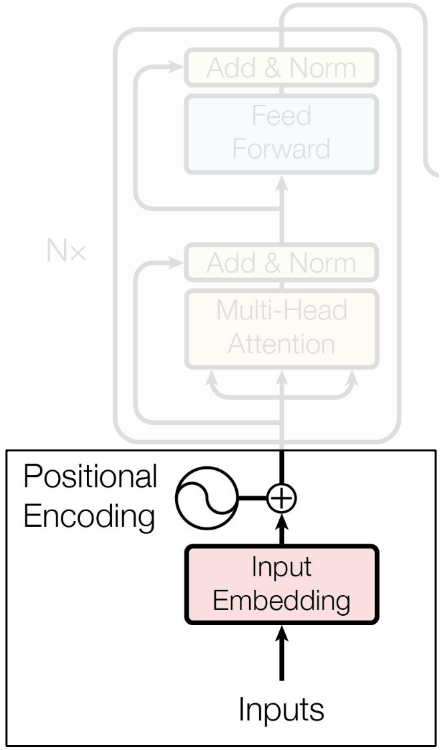
Inputs
I ate an apple



Tokenization

Tokenizer (split into individual words)

I ate an apple

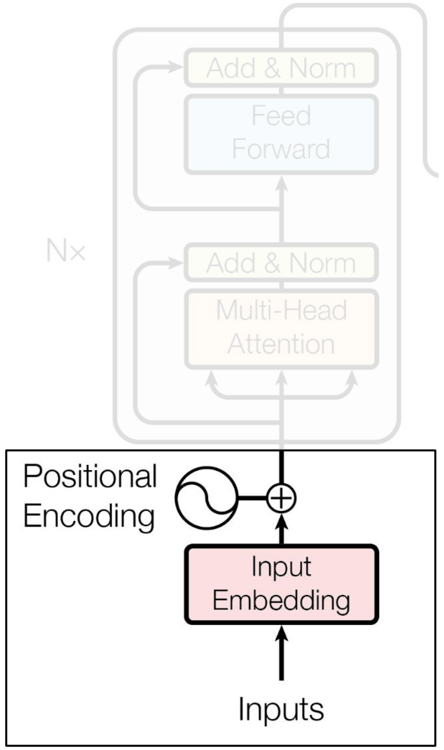


Tokenization

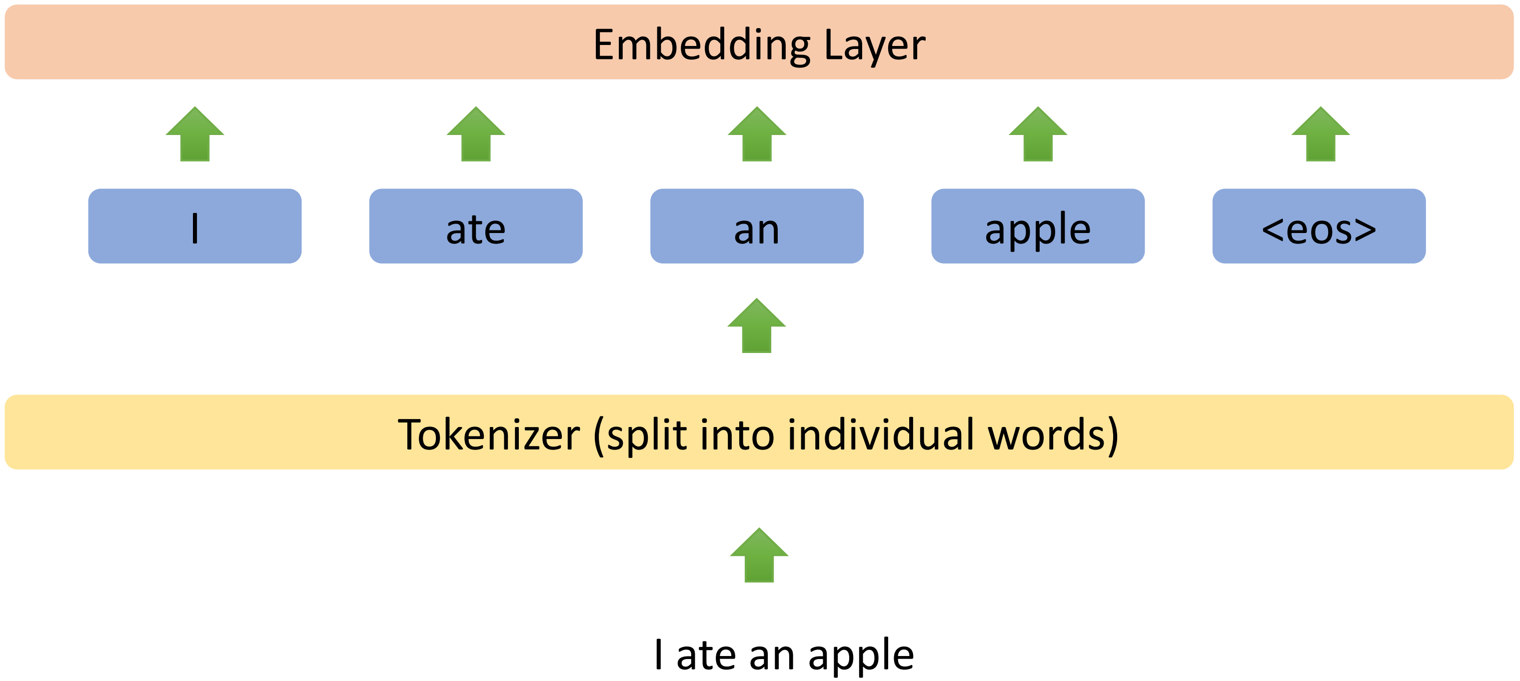
I | ate | an | apple | <eos>

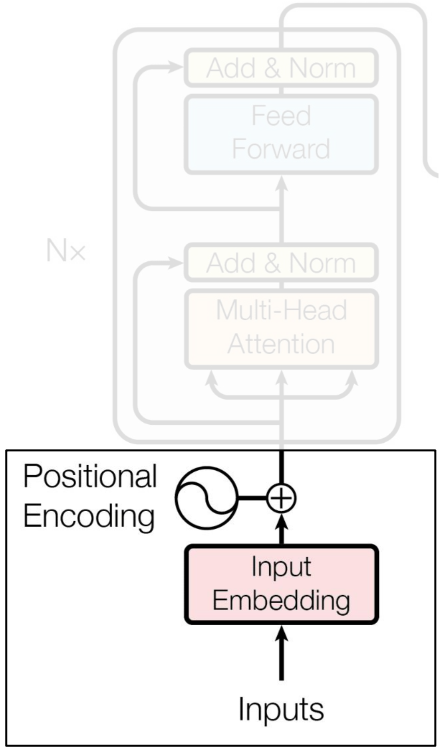
Tokenizer (split into individual words)

I ate an apple



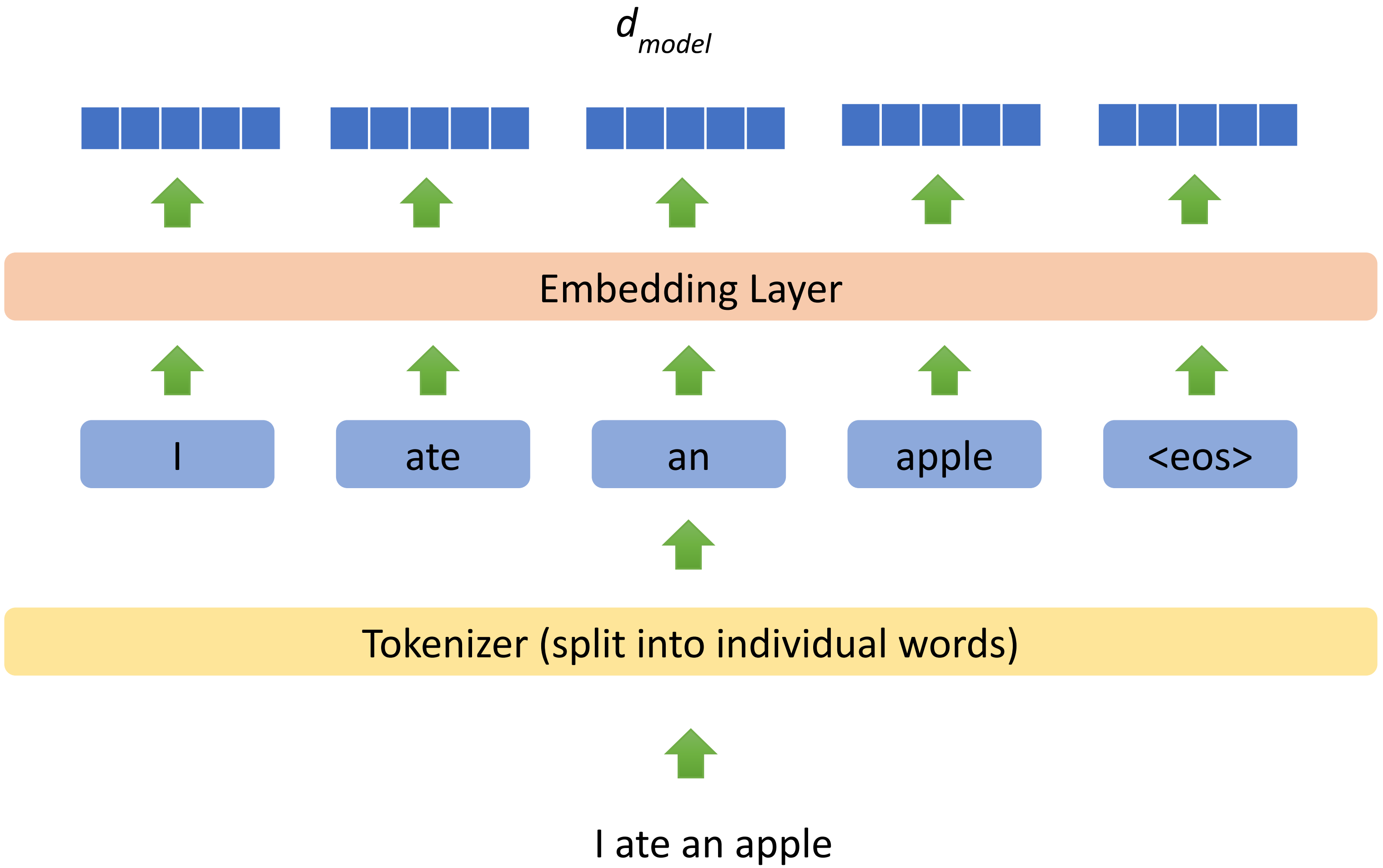
Input Embeddings

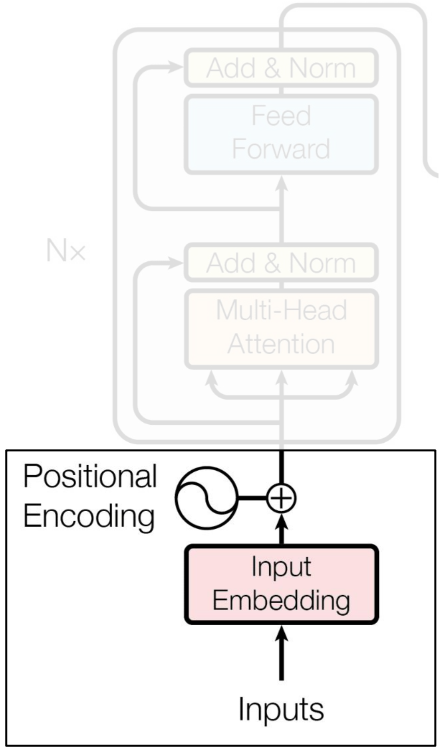




Generate Input Embeddings

Input Embeddings

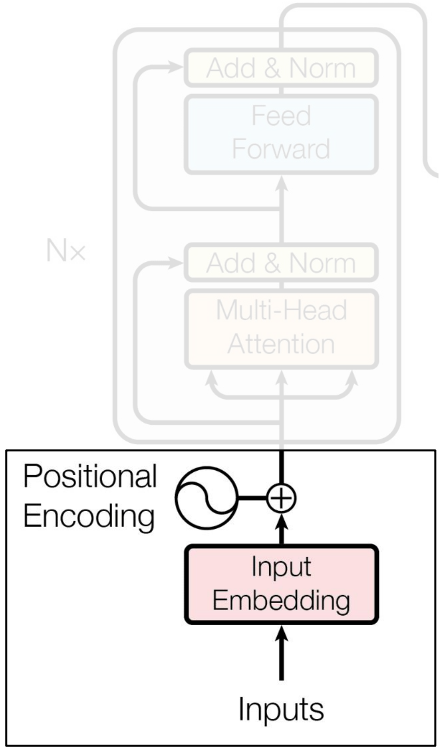




Generate Input Embeddings

Position Encodings

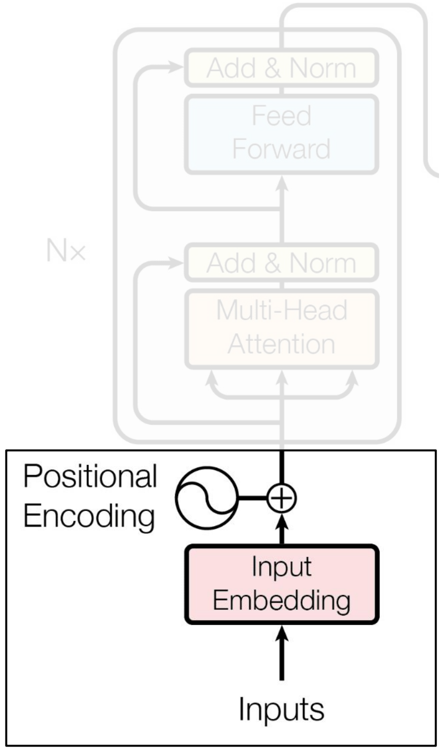
I   ate   an   apple   <eos>



Position Encodings

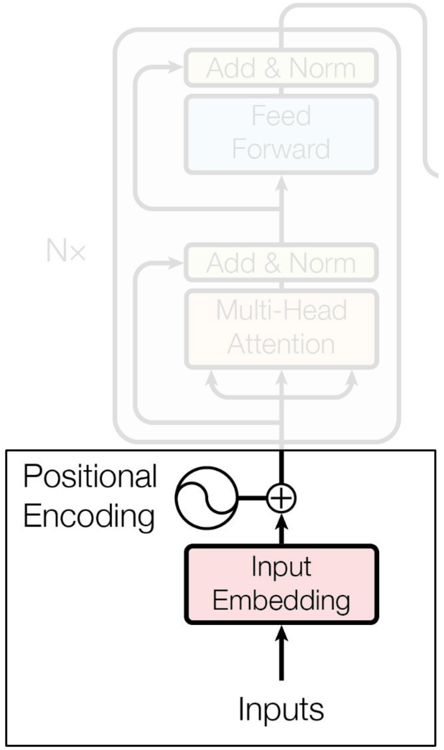
I ate an apple <eos>

apple ate an I <eos>



Position Encodings

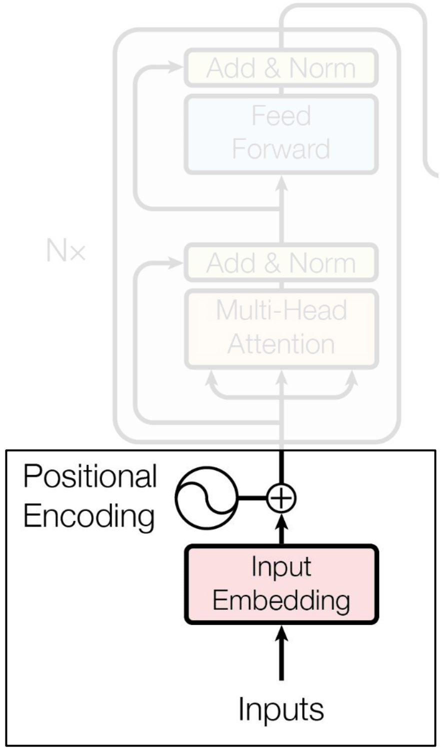
Requirements for Positional Encodings???



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

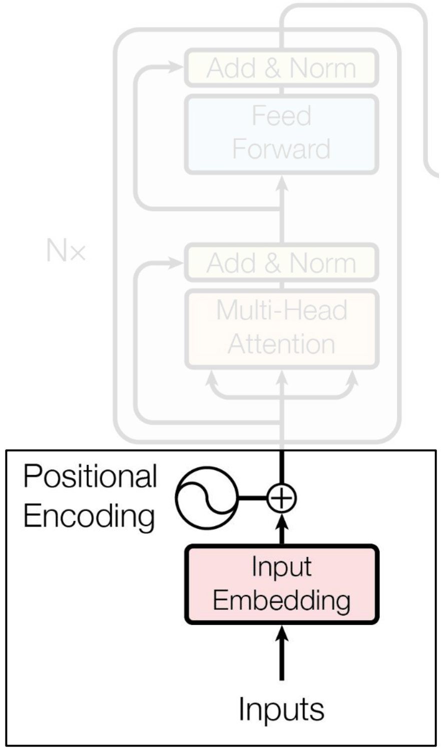
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic

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

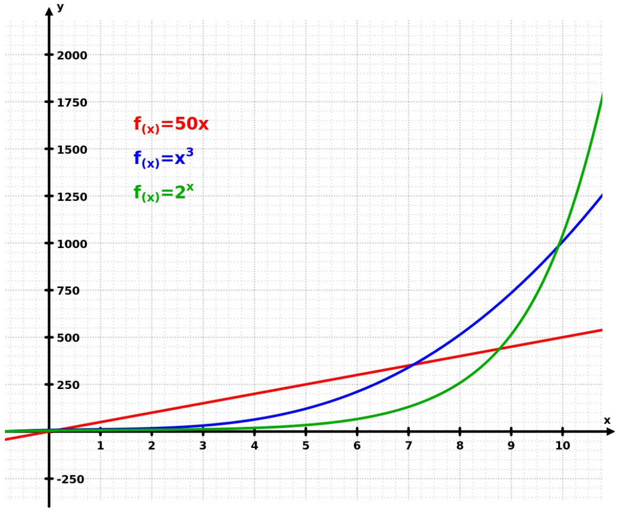
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic

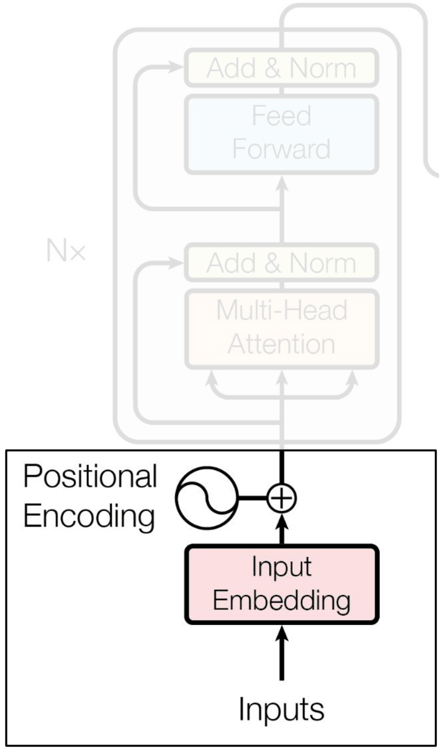
Possible Candidates :

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

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

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





Position Encodings

Requirements for Positional Encodings

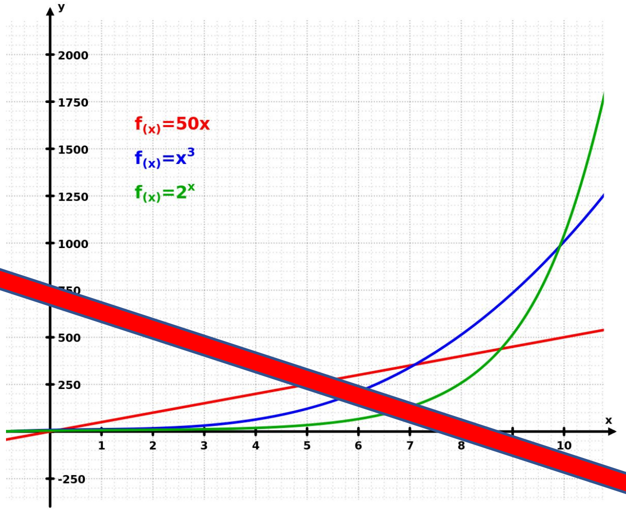
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic

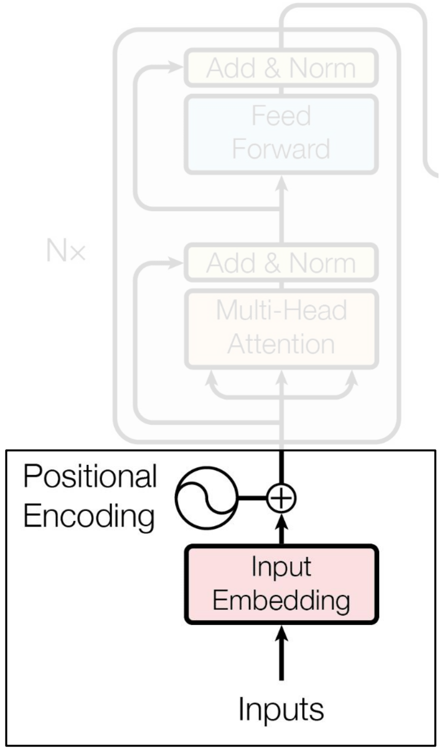
Possible Candidates :

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

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

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





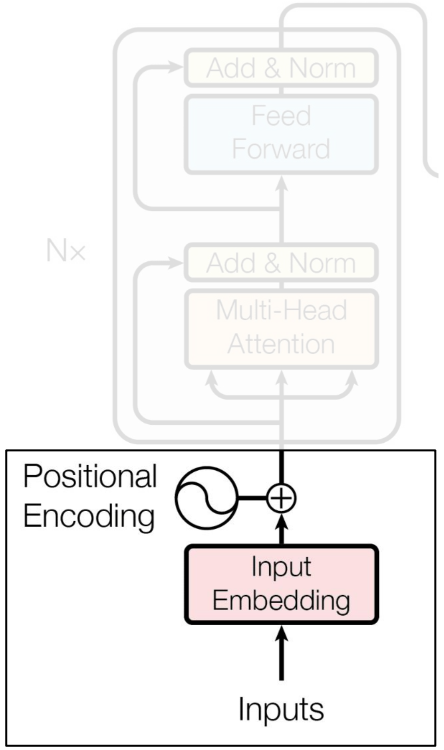
Position Encodings

Requirements for Positional Encodings

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

Possible Candidates

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



Position Encodings

Requirements for Positional Encodings

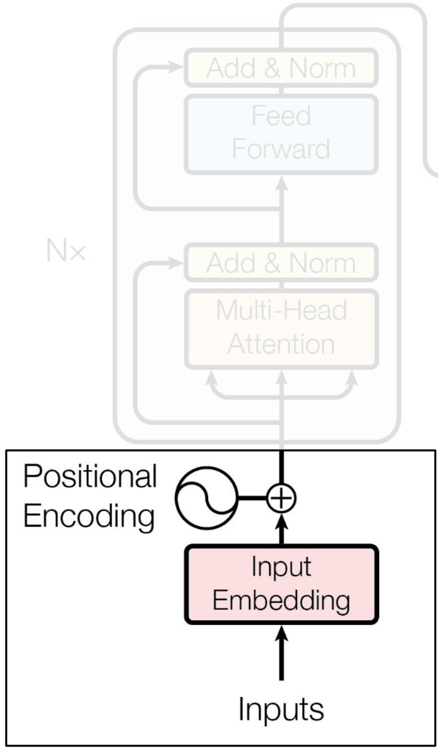
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position – not cyclic
- **Bounded**

Possible Candidates

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

M?

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



Rotary Position Embedding

ROFORMER: ENHANCED TRANSFORMER WITH ROTARY POSITION EMBEDDING

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

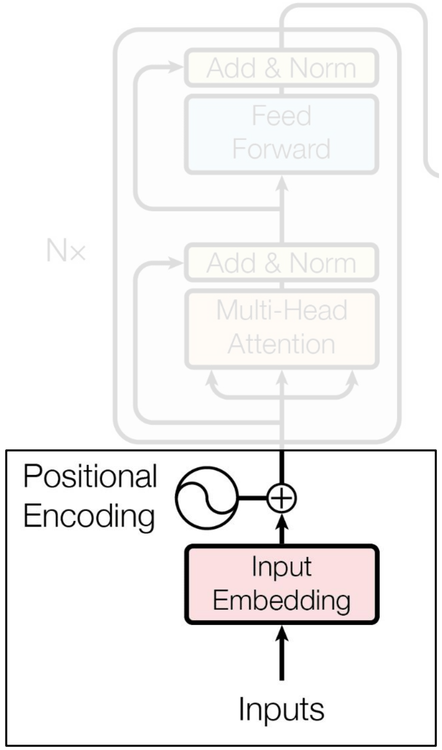


Table 2: Comparing RoFormer and BERT by fine tuning on downstream GLEU tasks.

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

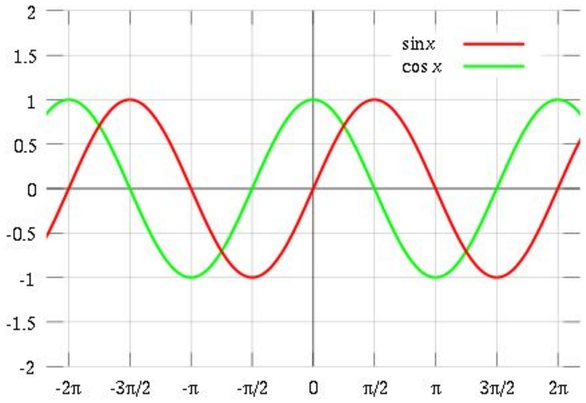
Requirements for Position Encodings

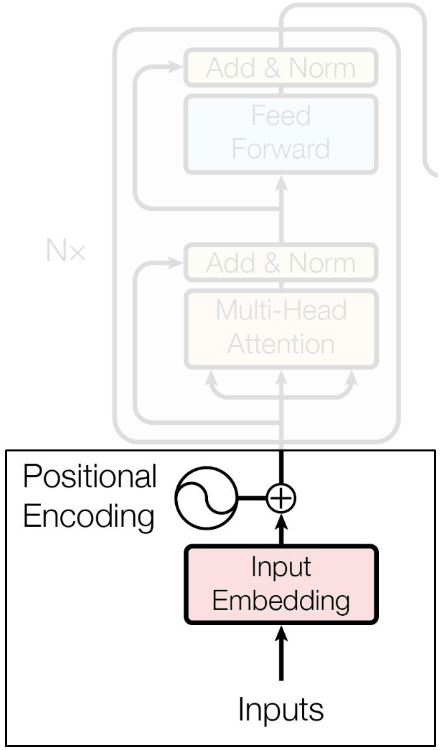
- Some representation of time? (like **seq2seq**?)
- Should be unique for each position
- Bounded

Actual Candidates

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

*cosine(**g(t)**)*





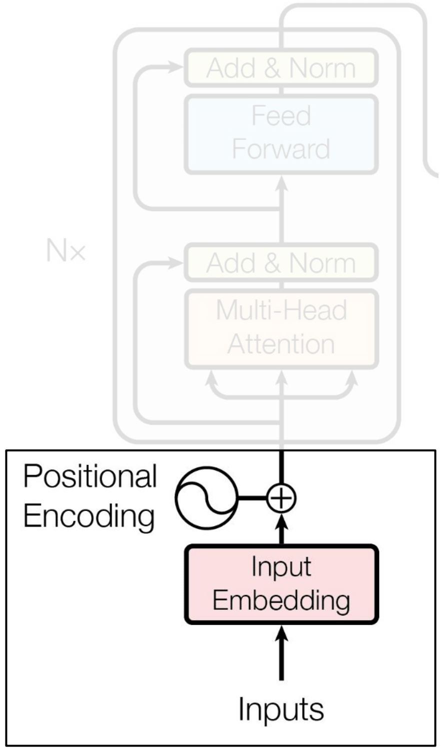
Requirements for g(t)

- 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

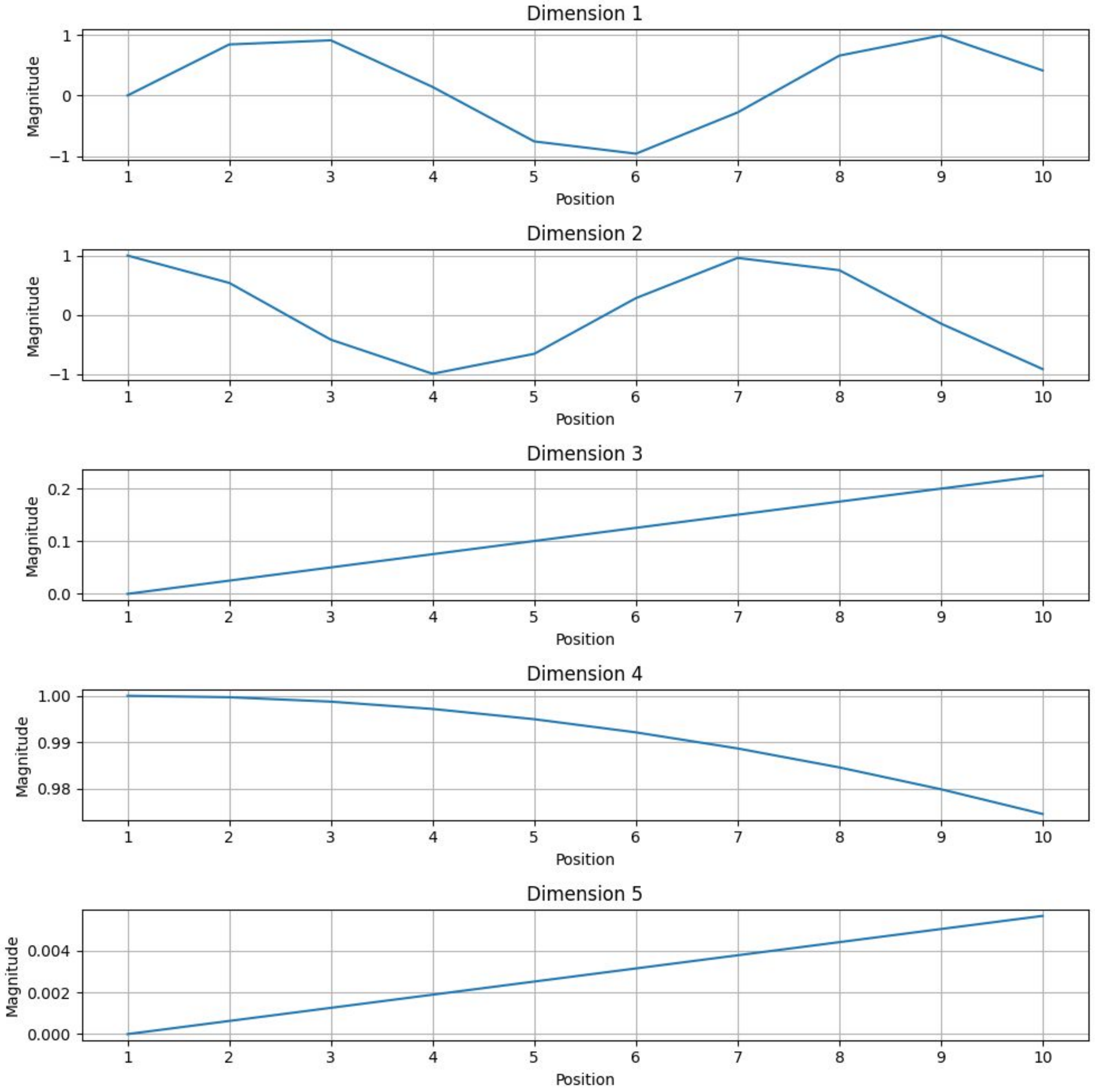
Position Encoding

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

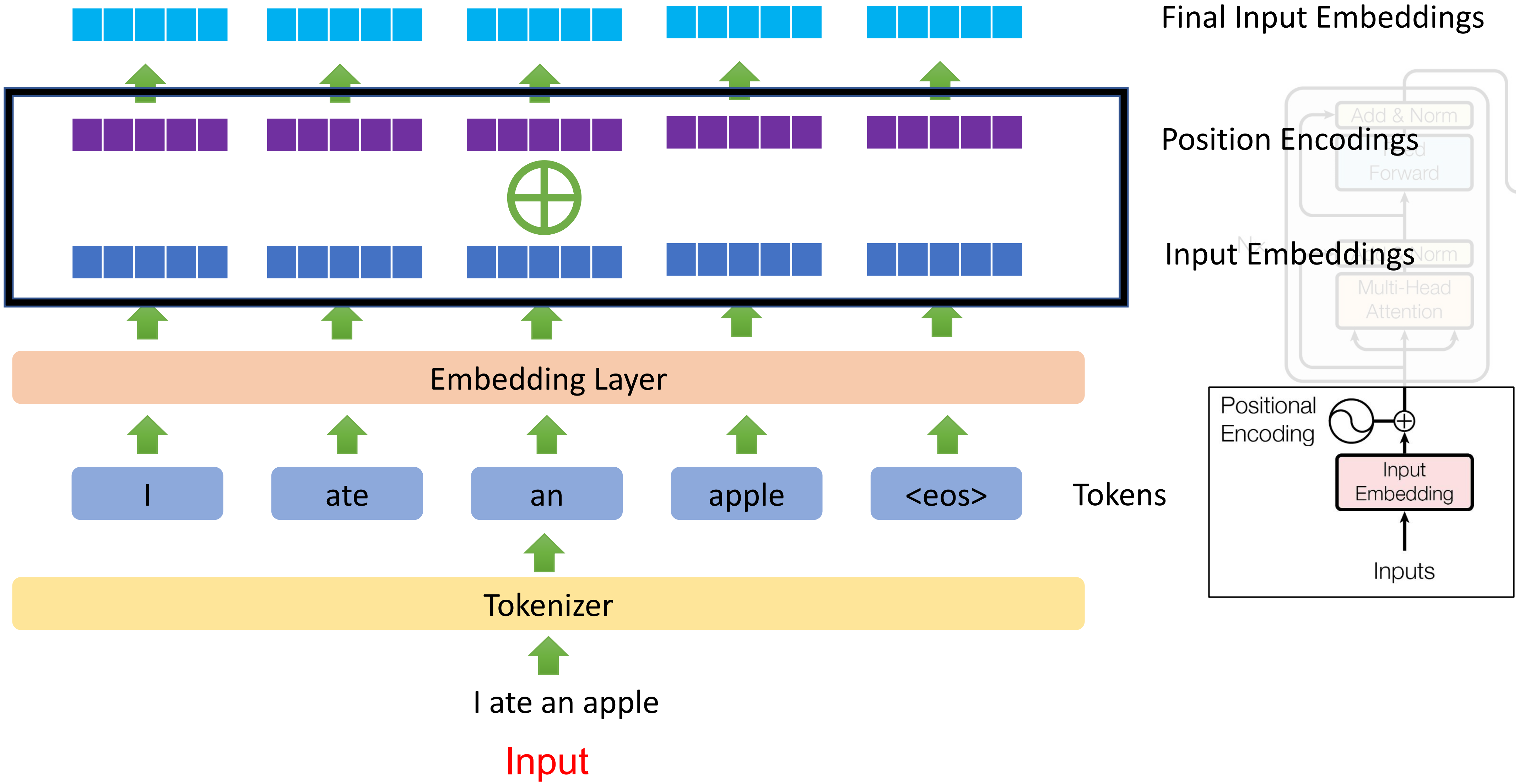
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

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

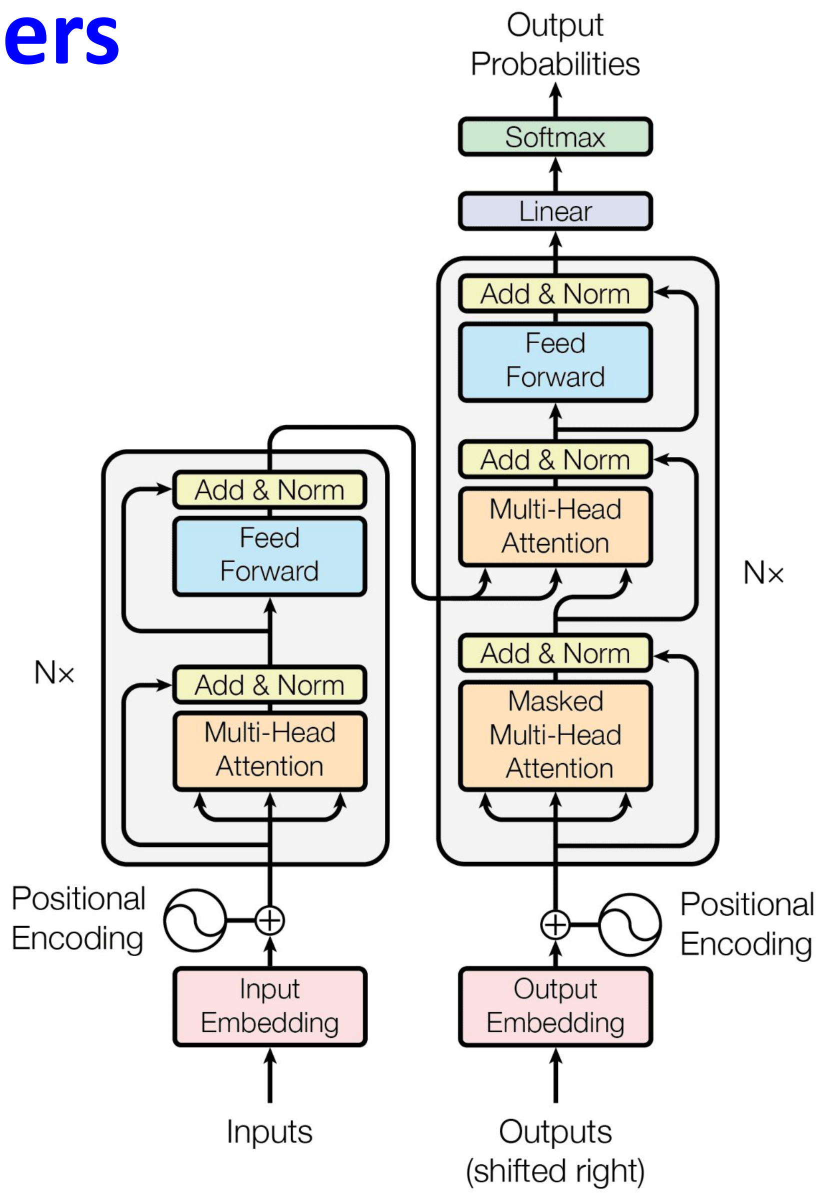


Position Encoding



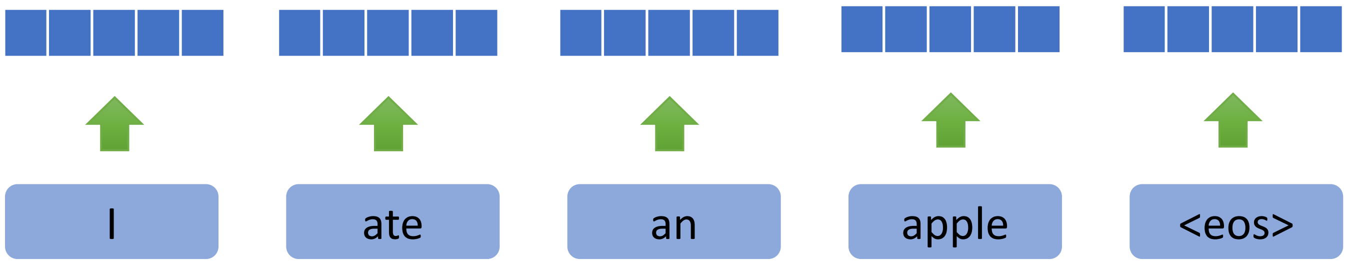
Transformers

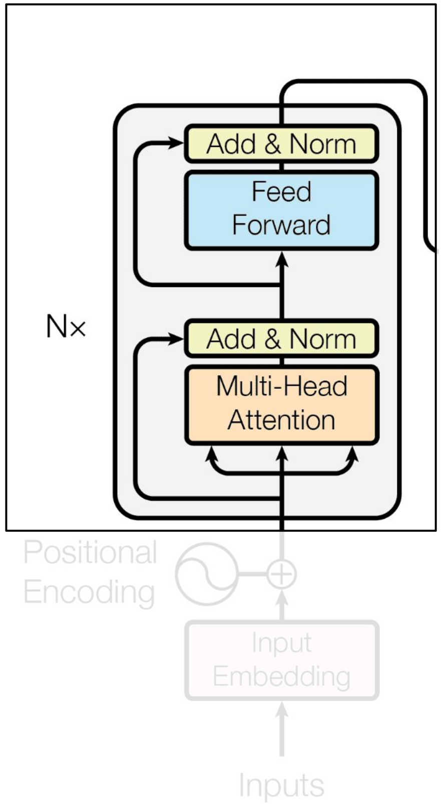
- ✔ **Tokenization**
- ✔ **Input Embeddings**
- ✔ **Position Encodings**
- Query, Key, & Value
- Attention
- Self Attention
- Multi-Head Attention
- Feed Forward
- Add & Norm
- Encoders
- Masked Attention
- Encoder Decoder Attention
- Linear
- Softmax
- Decoders
- Encoder-Decoder Models



Encoder

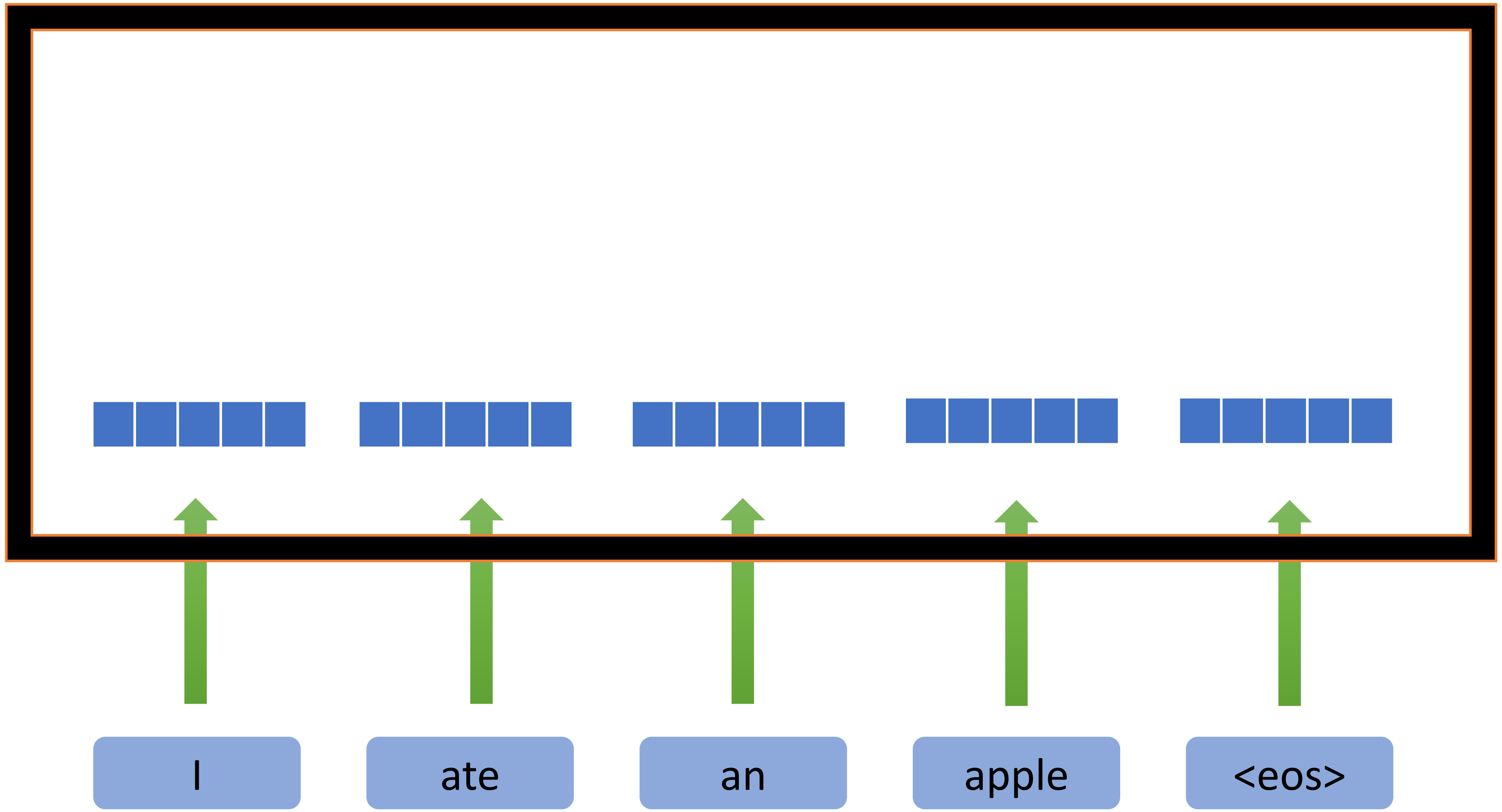
WHERE IS THE CONTEXT ?

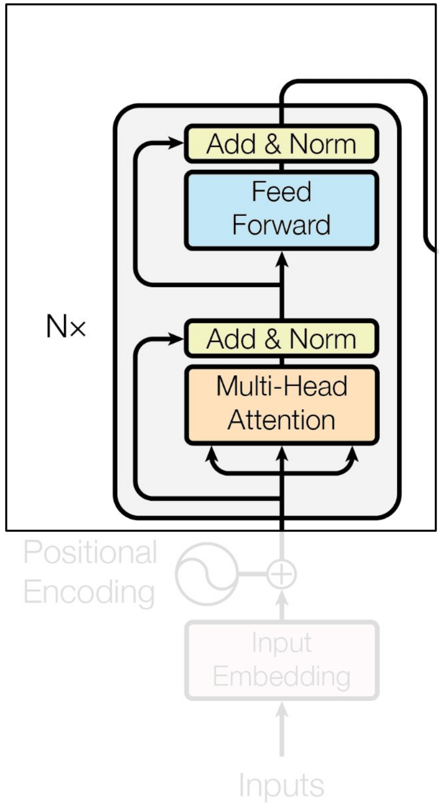




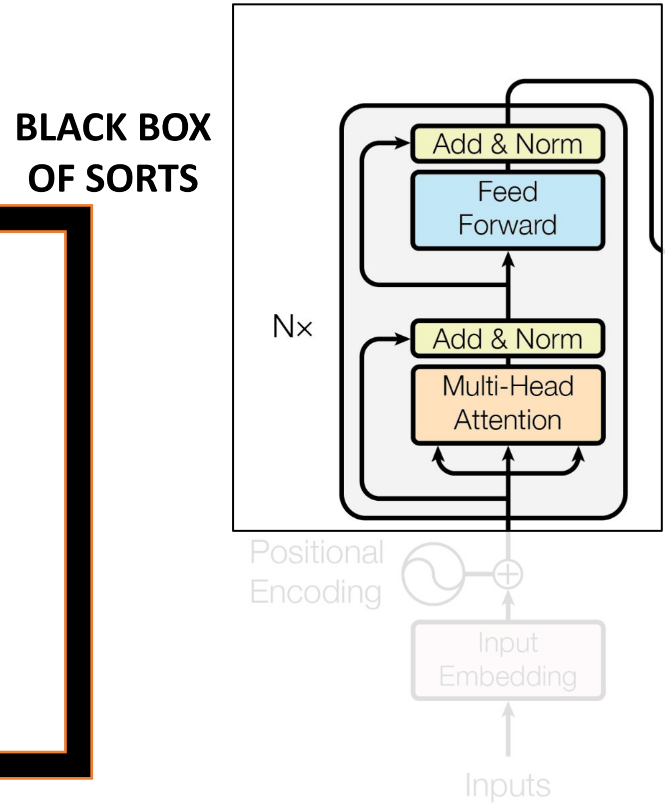
Encoder

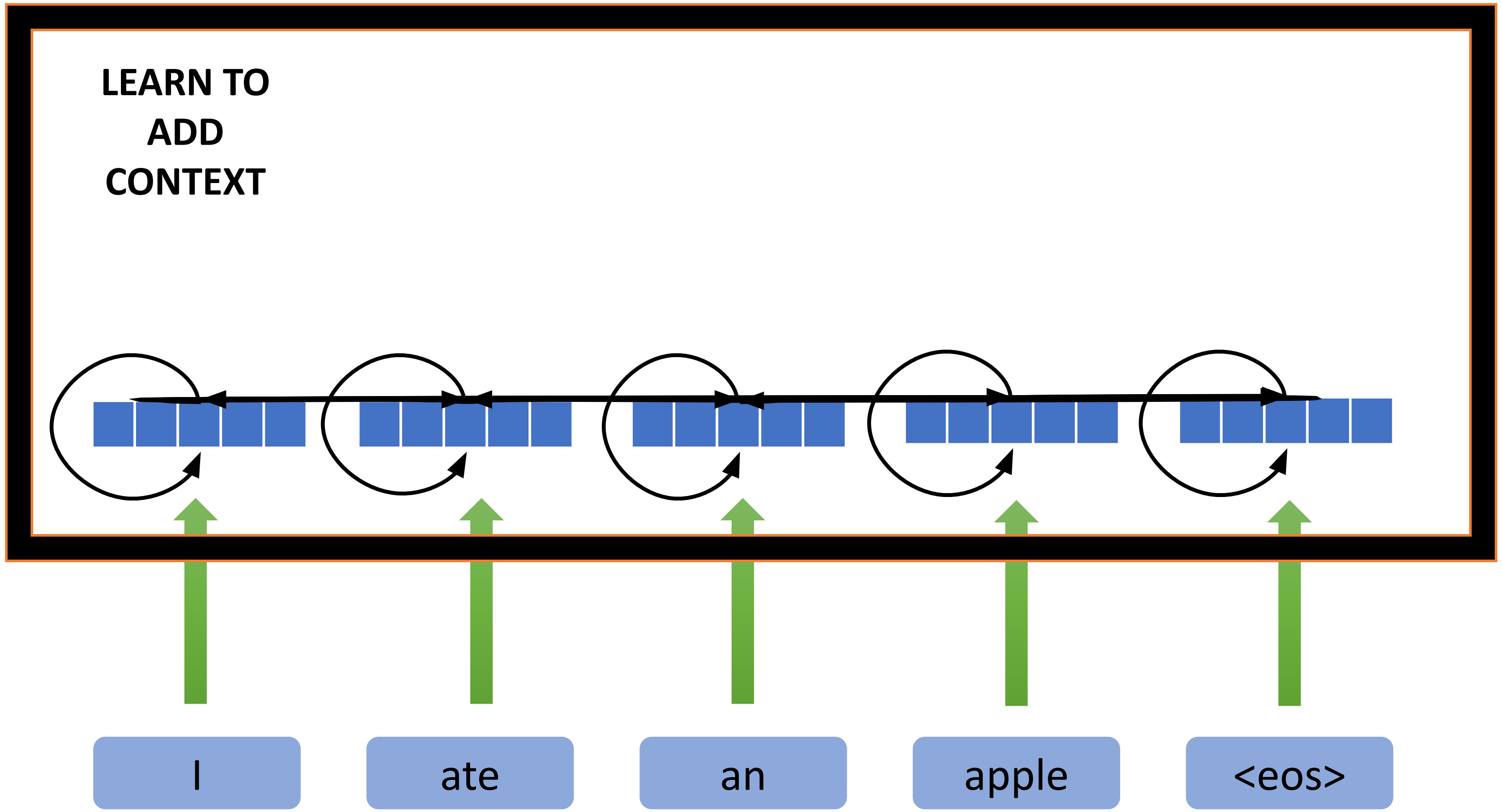
BLACK BOX
OF SORTS



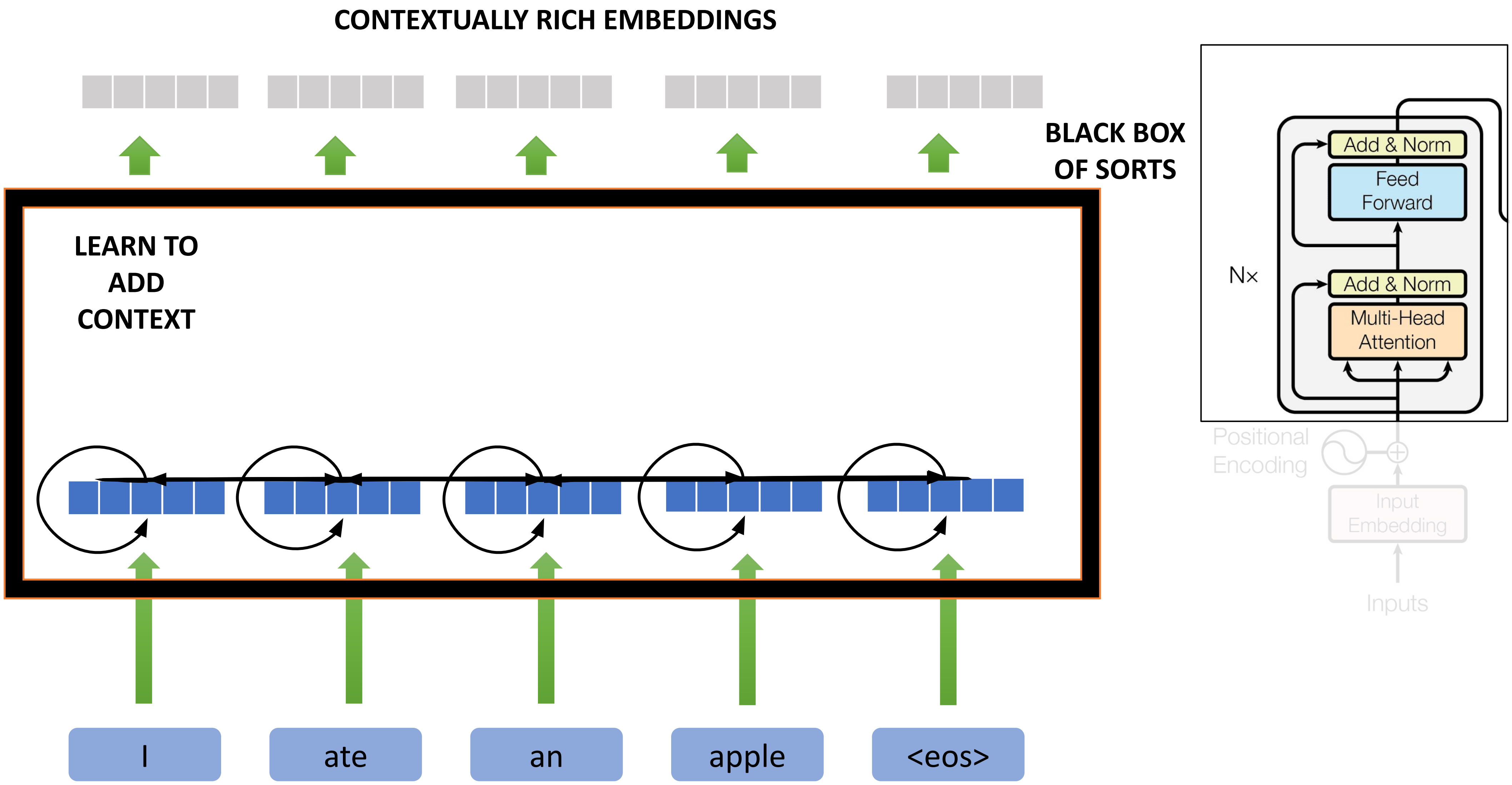


Encoder


BLACK BOX
OF SORTS
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Inputs


LEARN TO
ADD
CONTEXT
I
ate
an
apple
<eos>

Encoder

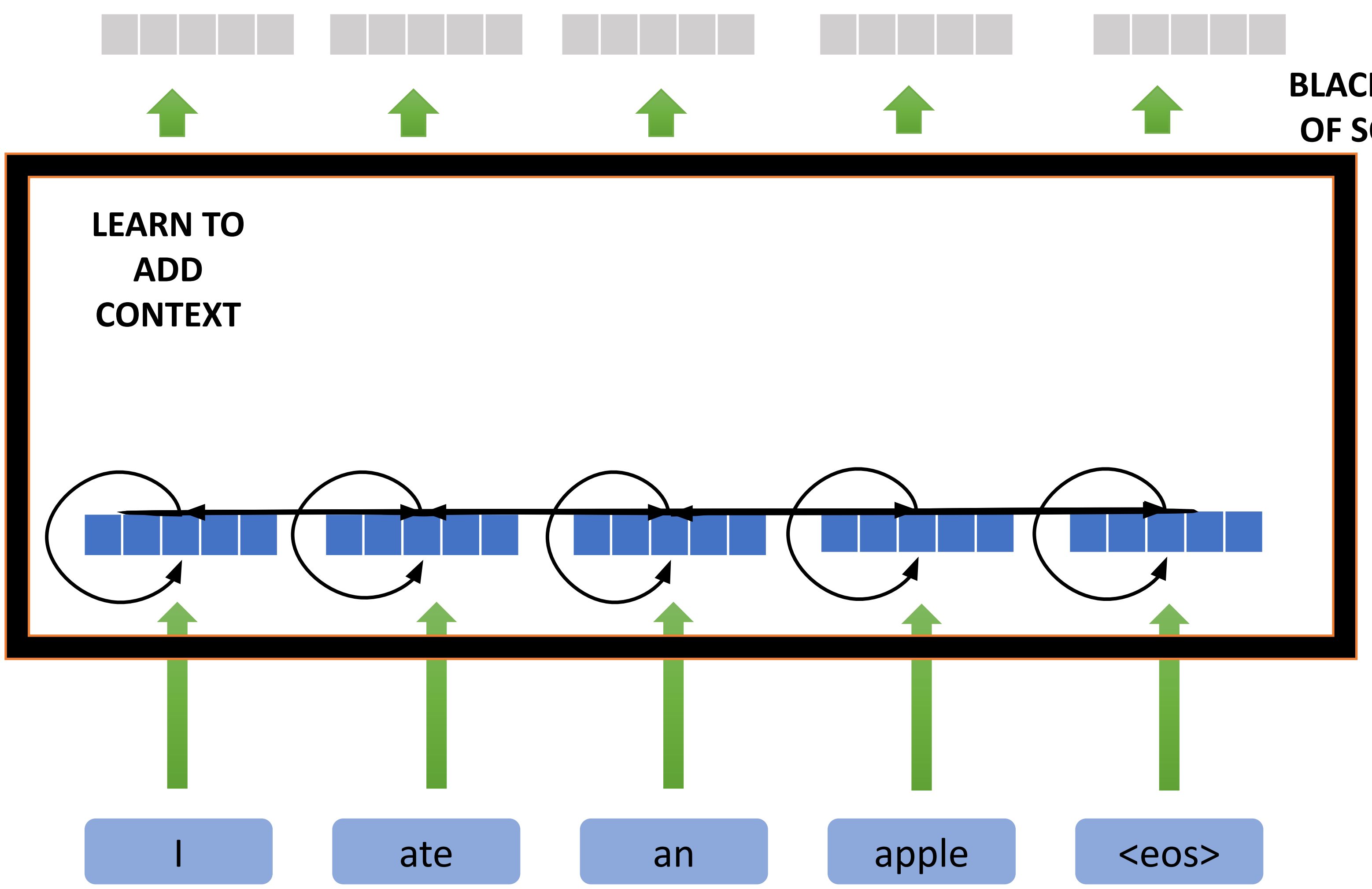


CONTEXTUALLY RICH EMBEDDINGS
BLACK BOX OF SORTS
LEARN TO ADD CONTEXT
I
ate
an
apple
<eos>
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N×
Positional Encoding
Input Embedding
Inputs

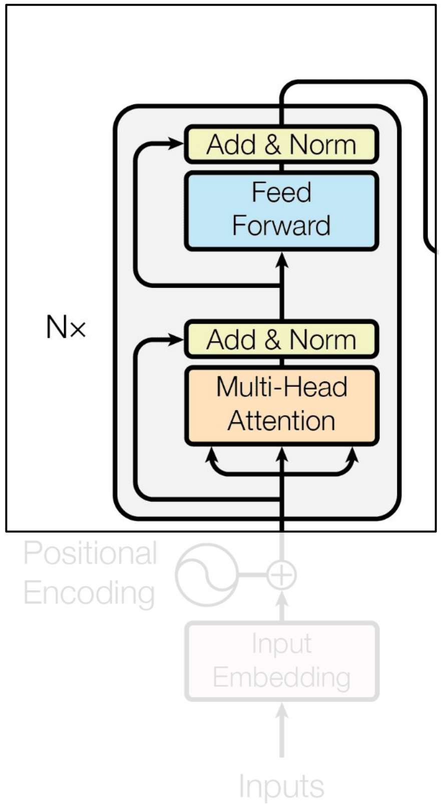
Encoder

$\alpha_{[\,i\,j\,]}$?

CONTEXTUALLY RICH EMBEDDINGS



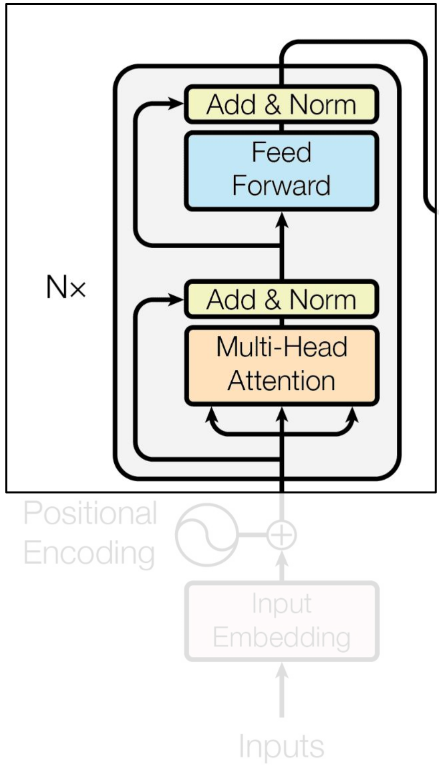
BLACK BOX
OF SORTS



Attention

From lecture 18: $\alpha_{[\,i\;j\,]}$?

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



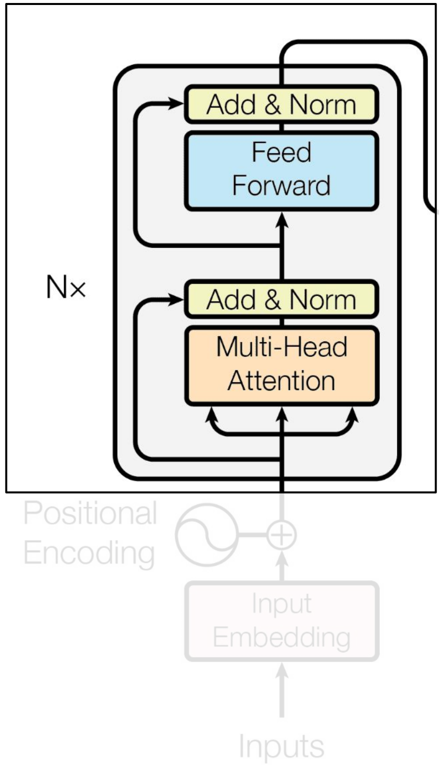
Attention

$\alpha_{[\,i\,j\,]}$?

From lecture 18:

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

- Query
- Key
- Value



Query, Key & Value

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”}

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

{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

{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 !!

{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”}


```
{"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

1. Search for info

Key

1. Interacts directly with Queries
2. Distinguishes one object from another
3. Identify which object is the most relevant and by how much

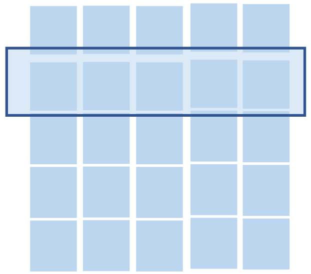
Value

1. Actual details of the object
2. More fine grained

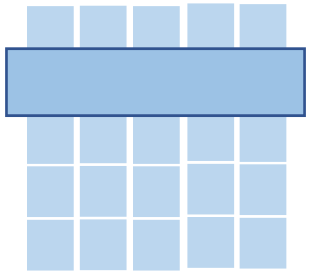
Attention

Query

Key Value
Store

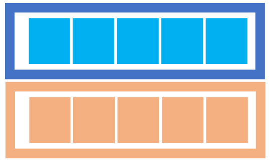


Key



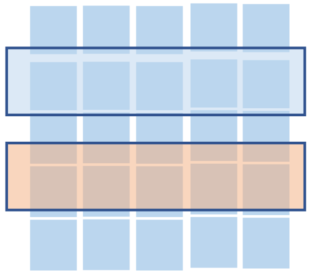
Value

Attention

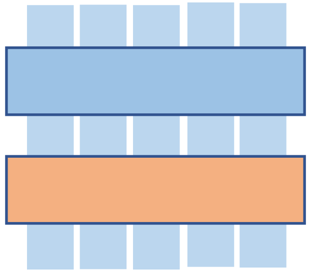


Query

Key Value
Store



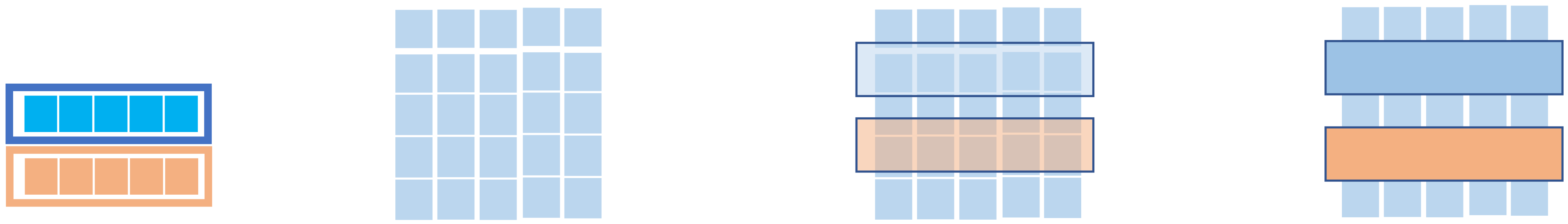
Key



Value

Attention

Done at the same time !!



Query

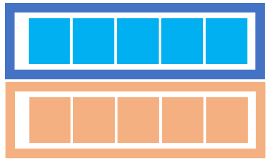
Key Value Store

Key

Value

Attention

Parallelizable !!!

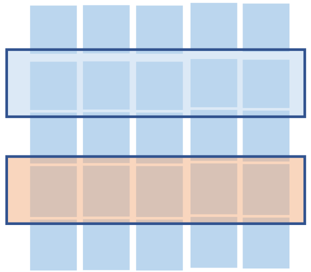


Query

Q

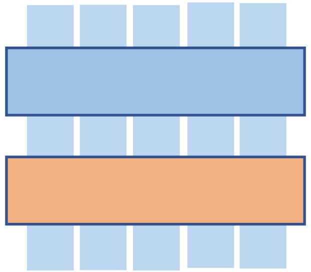
Key Value
Store

QK^T



Key

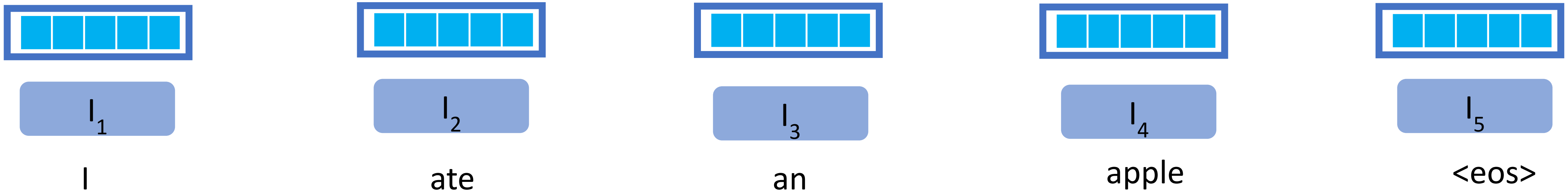
$softmax(\frac{QK^T}{\sqrt{d}})$



Value

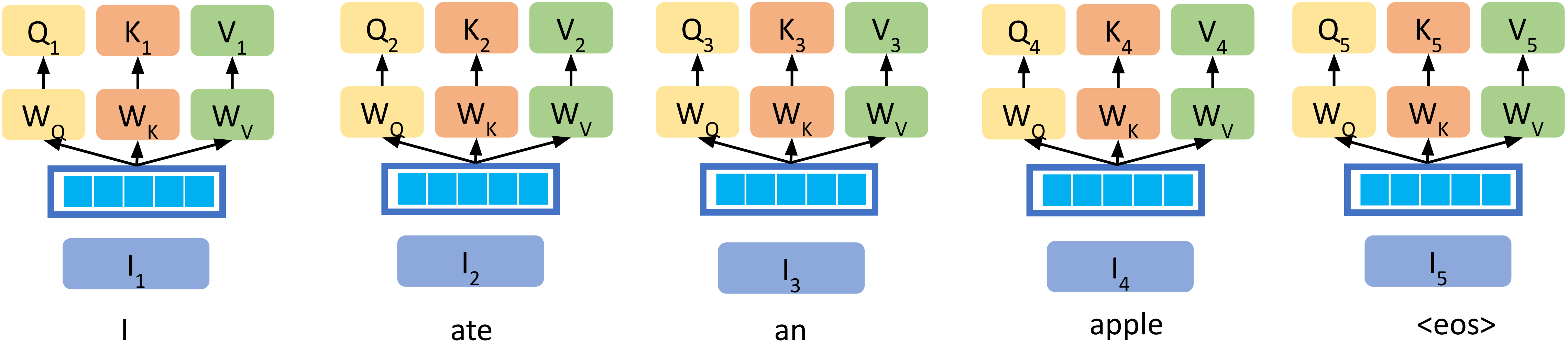
$softmax(\frac{QK^T}{\sqrt{d}})V$

Attention



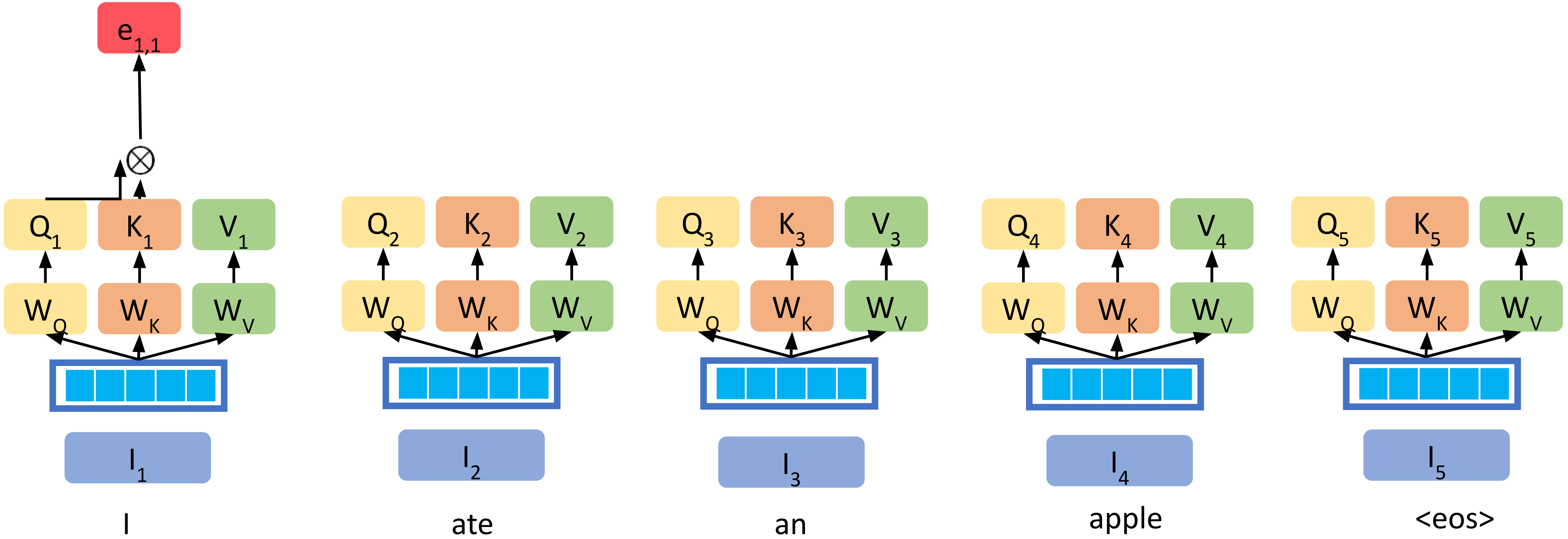
Attention

Dimensions across QKV have been dropped for brevity



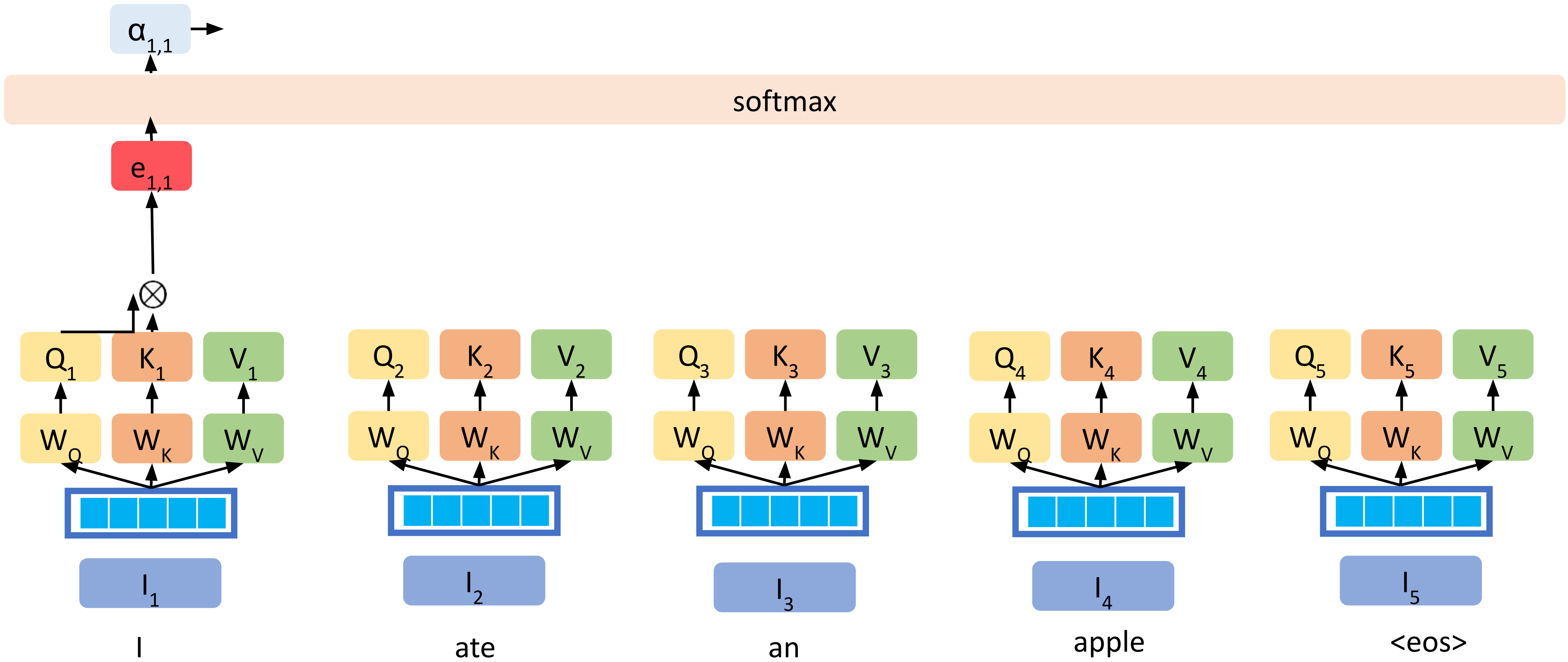
Attention

Dimensions across QKV have been dropped for brevity



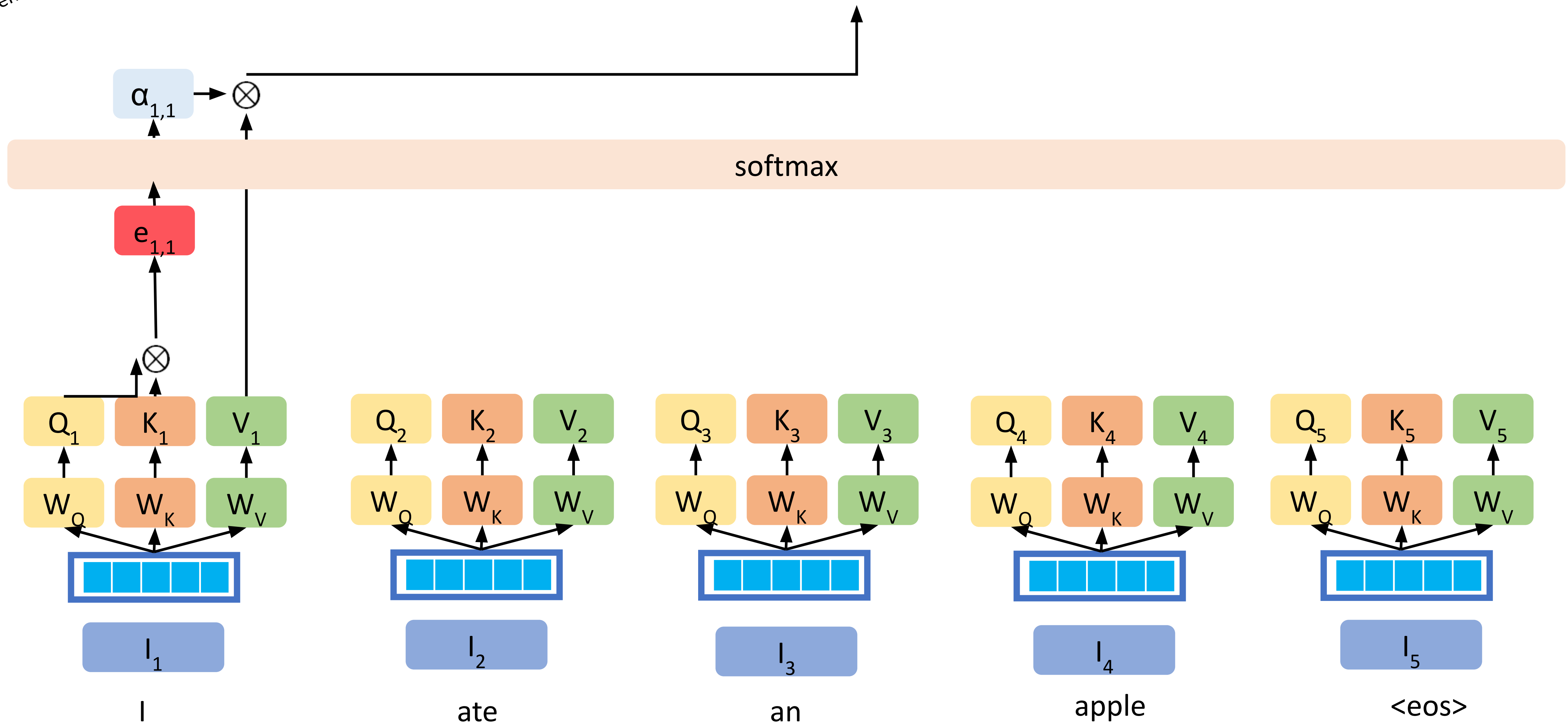
Attention

Dimensions across QKV have been dropped for brevity



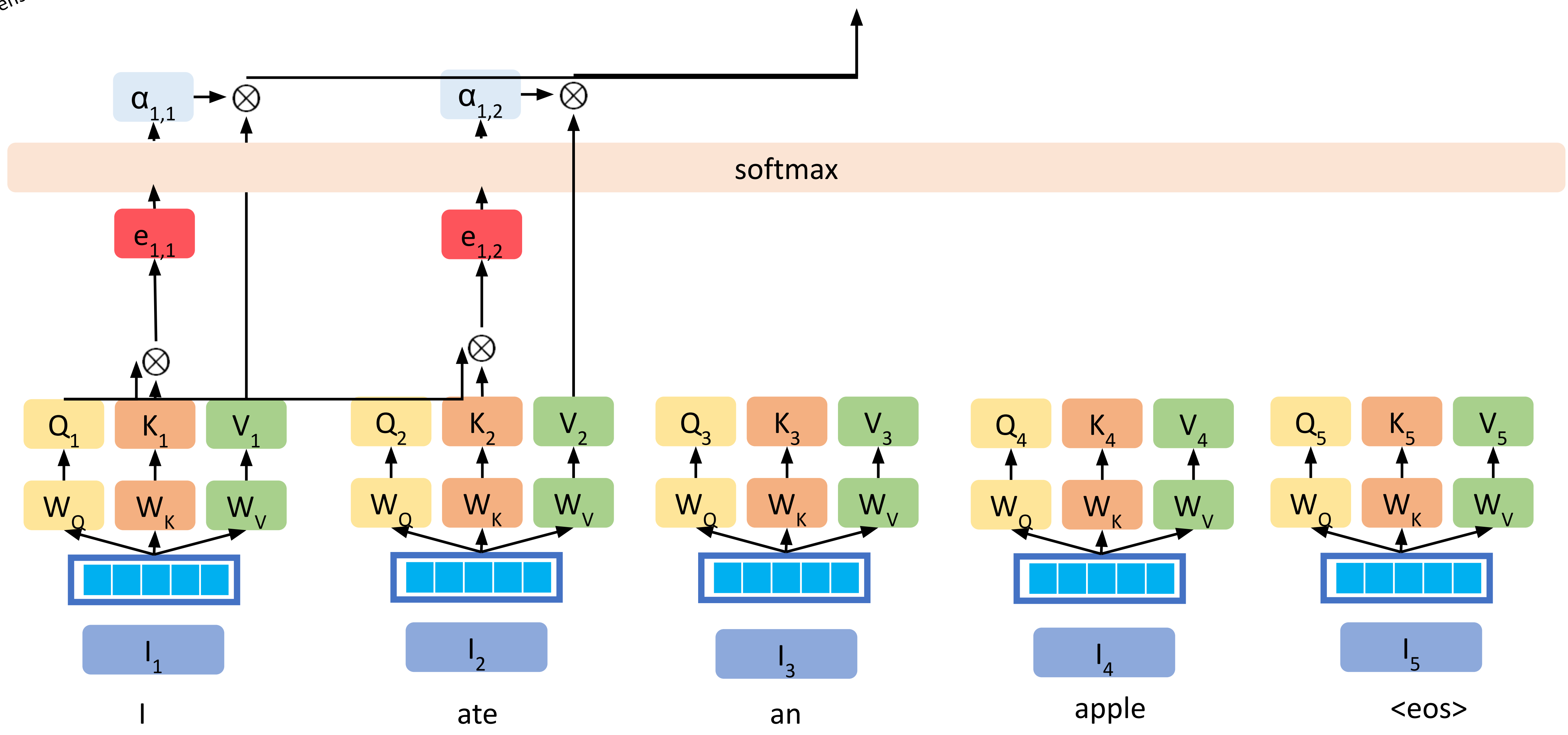
Attention

Dimensions across QKV have been dropped for brevity

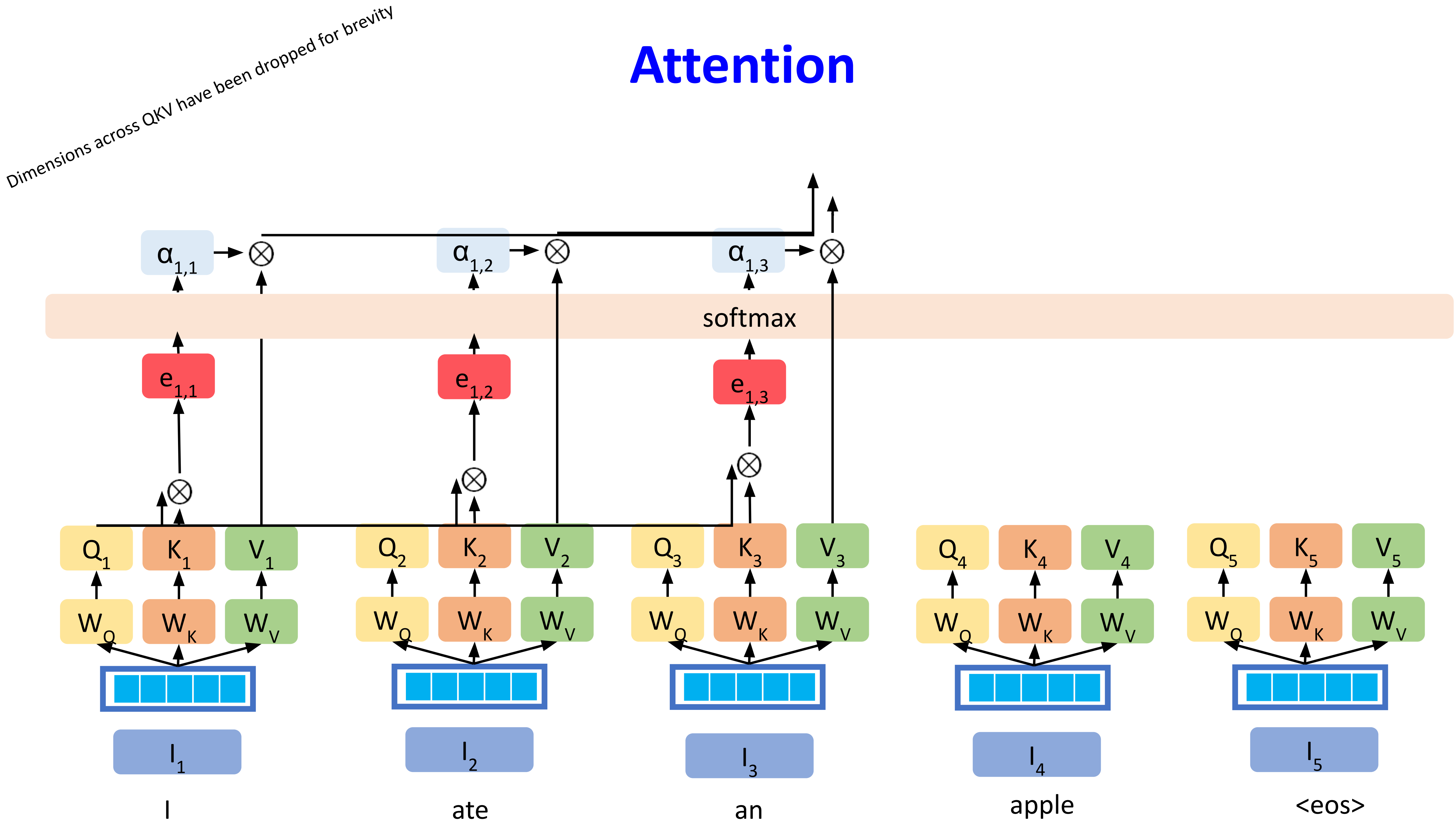


Attention

Dimensions across QKV have been dropped for brevity



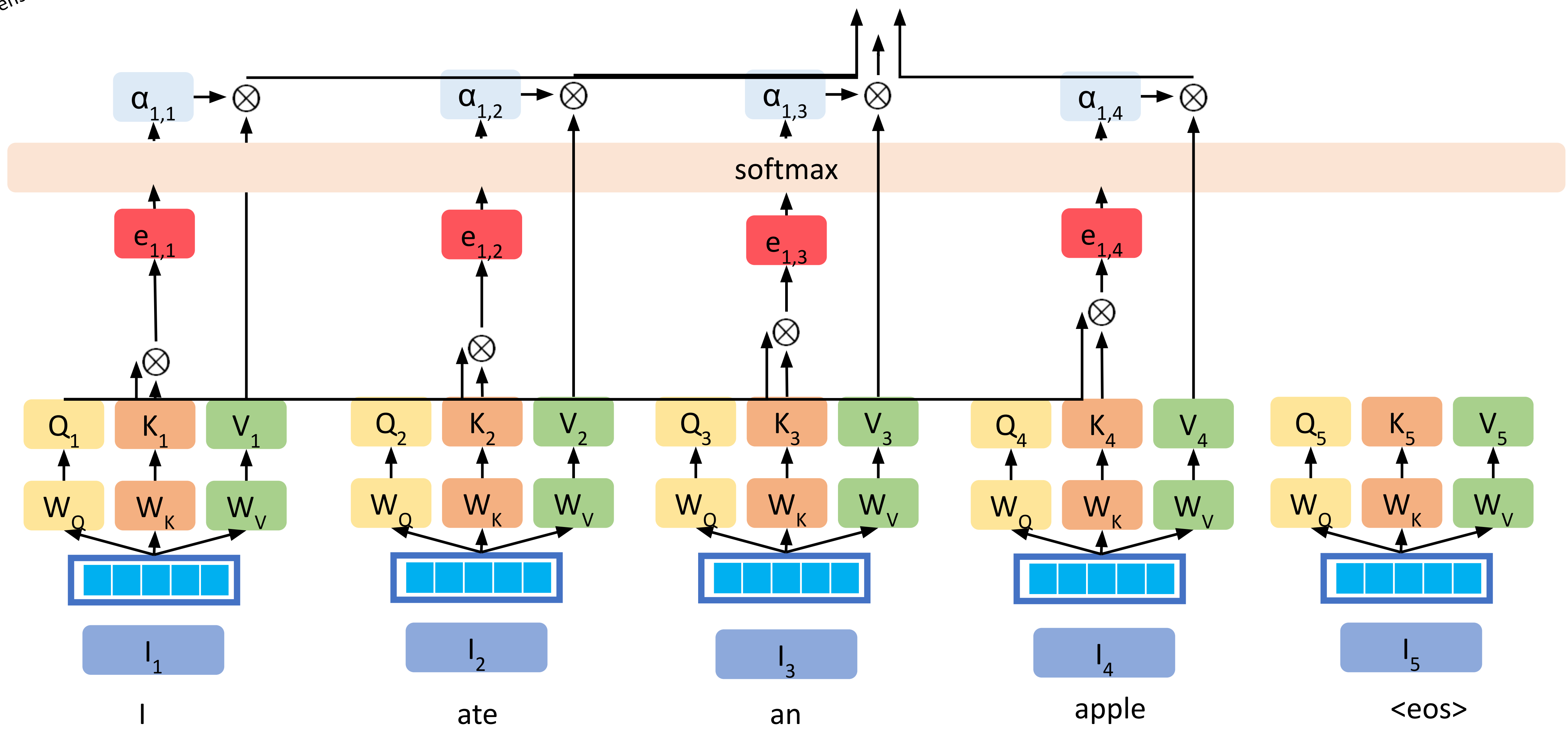
Attention



Dimensions across QKV have been dropped for brevity
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
softmax
$e_{1,1}$
$e_{1,2}$
$e_{1,3}$
Q_1 K_1 V_1
Q_2 K_2 V_2
Q_3 K_3 V_3
Q_4 K_4 V_4
Q_5 K_5 V_5
W_Q W_K W_V
I_1
I_2
I_3
I_4
I_5
I
ate
an
apple
<eos>

Dimensions across QKV have been dropped for brevity

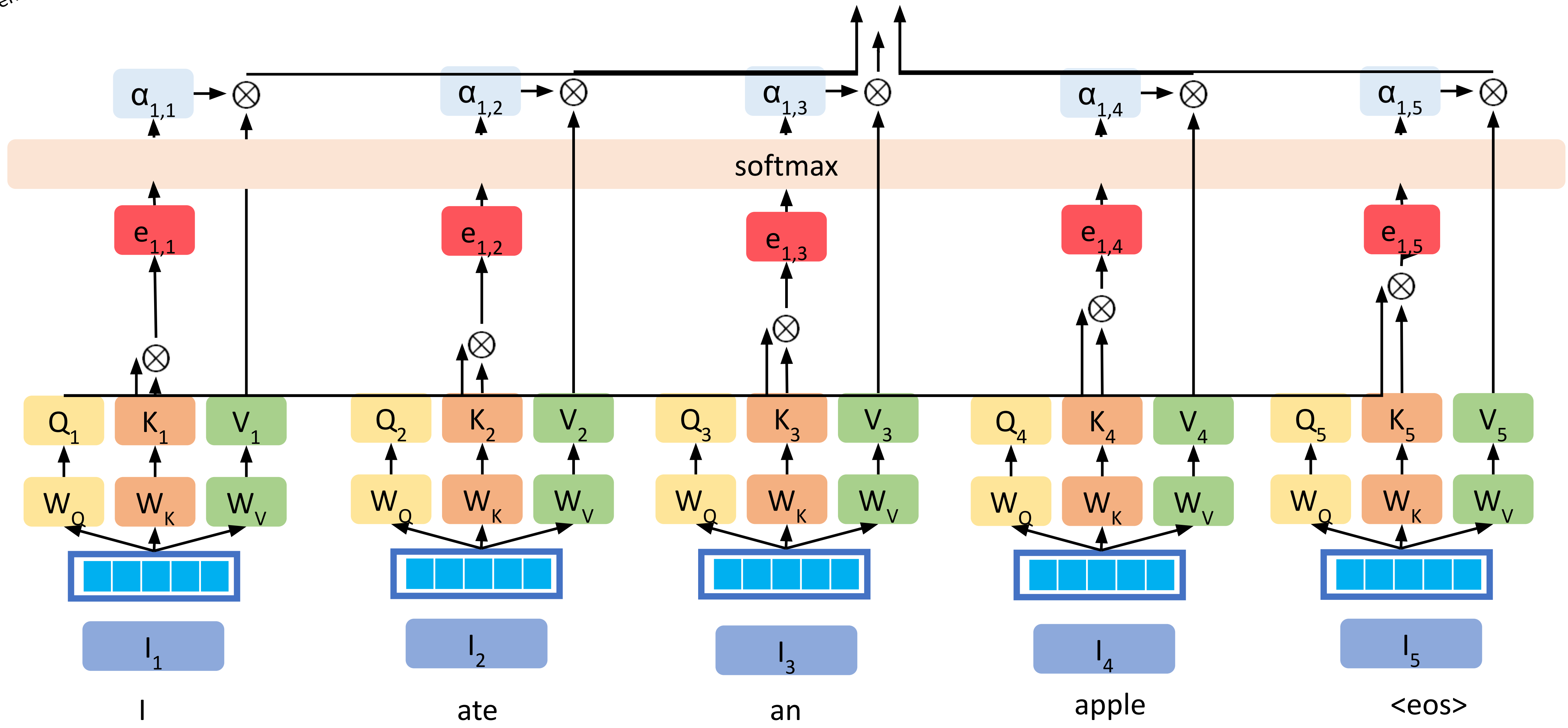
Attention



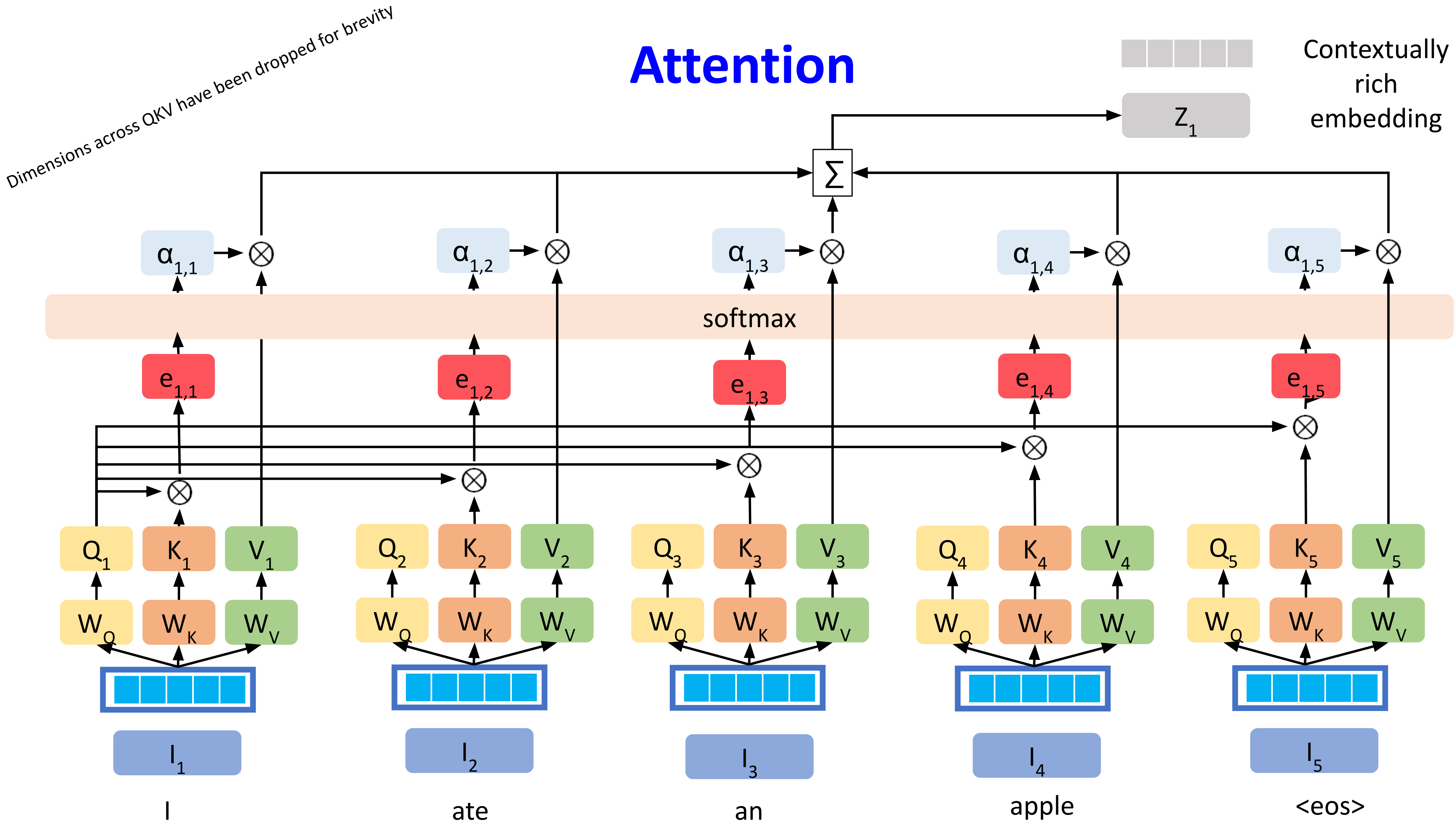
α1,1
α1,2
α1,3
α1,4
softmax
e1,1
e1,2
e1,3
e1,4
Q1 K1 V1
Q2 K2 V2
Q3 K3 V3
Q4 K4 V4
Q5 K5 V5
WQ WK WV
I1
I2
I3
I4
I5
I
ate
an
apple
<eos>

Dimensions across QKV have been dropped for brevity

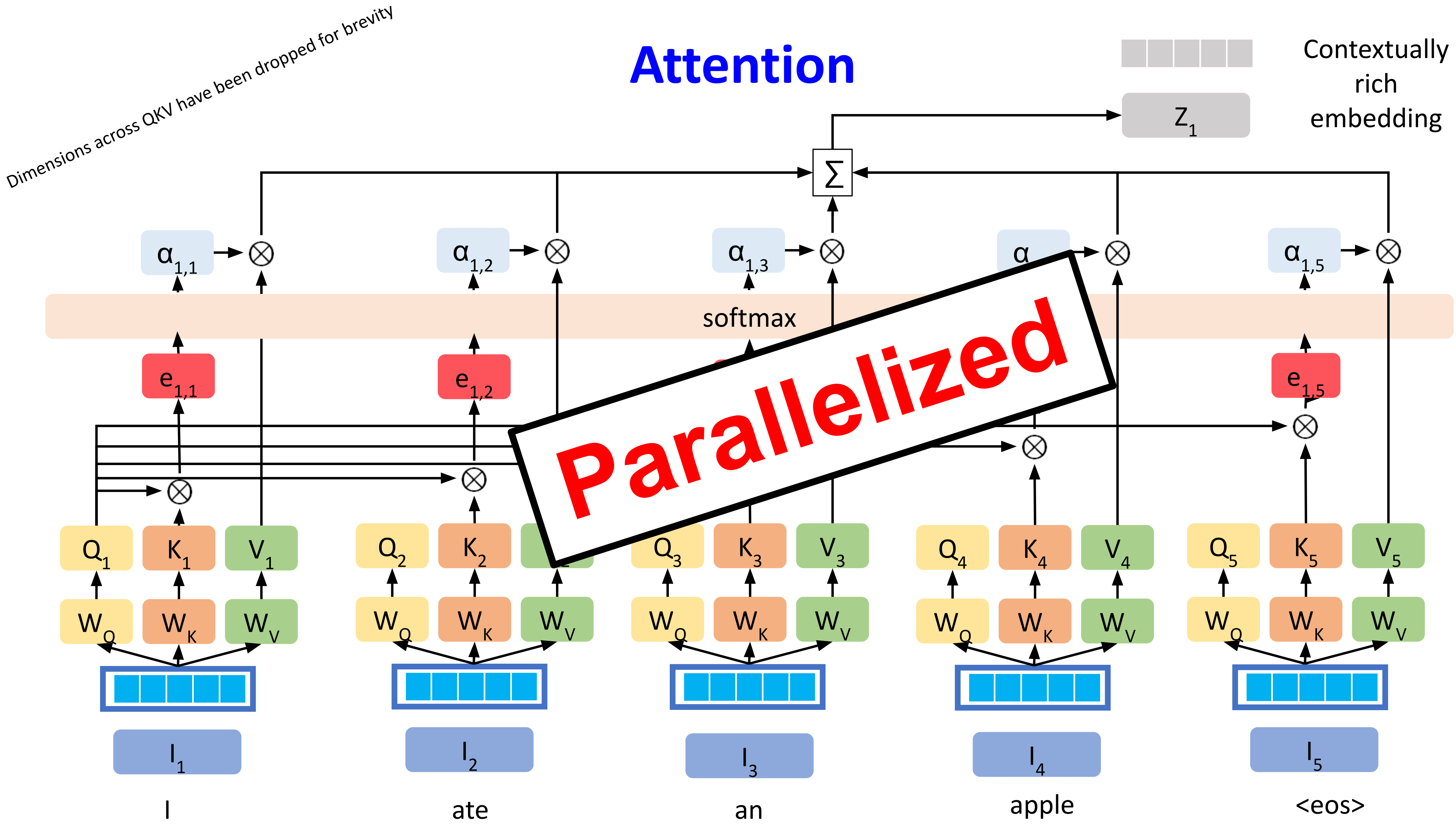
Attention



Attention

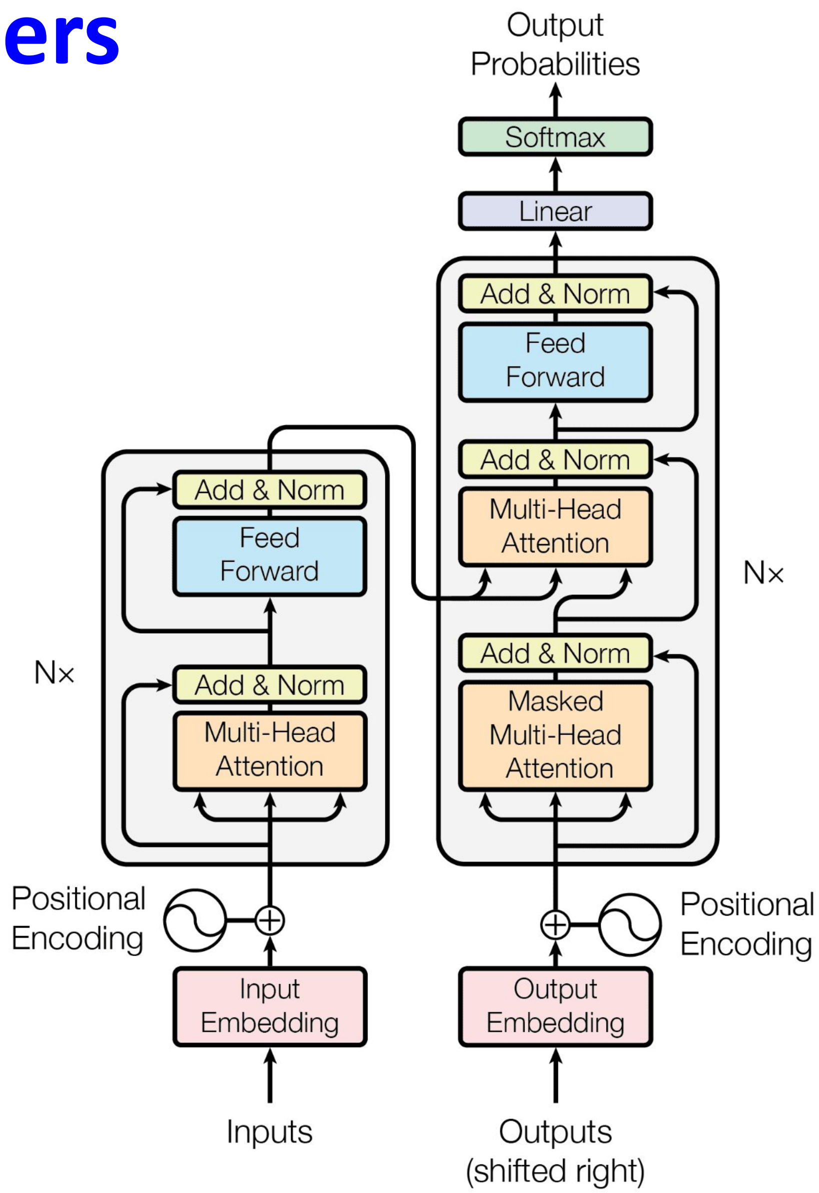

Dimensions across QKV have been dropped for brevity
Contextually rich embedding
Z_1
Σ
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
$\alpha_{1,5}$
softmax
$e_{1,1}$
$e_{1,2}$
$e_{1,3}$
$e_{1,4}$
$e_{1,5}$
Q_1 K_1 V_1
Q_2 K_2 V_2
Q_3 K_3 V_3
Q_4 K_4 V_4
Q_5 K_5 V_5
W_Q W_K W_V
I_1 I_2 I_3 I_4 I_5
I
ate
an
apple
<eos>

Attention


Dimensions across QKV have been dropped for brevity
Contextually rich embedding
Z1
Σ
α1,1
α1,2
α1,3
α1,5
softmax
e1,1
e1,2
e1,5
Parallelized
Q1 K1 V1
Q2 K2
Q3 K3 V3
Q4 K4 V4
Q5 K5 V5
WQ WK WV
I1 I2 I3 I4 I5
I
ate
an
apple
<eos>

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



Poll 1 - @1581

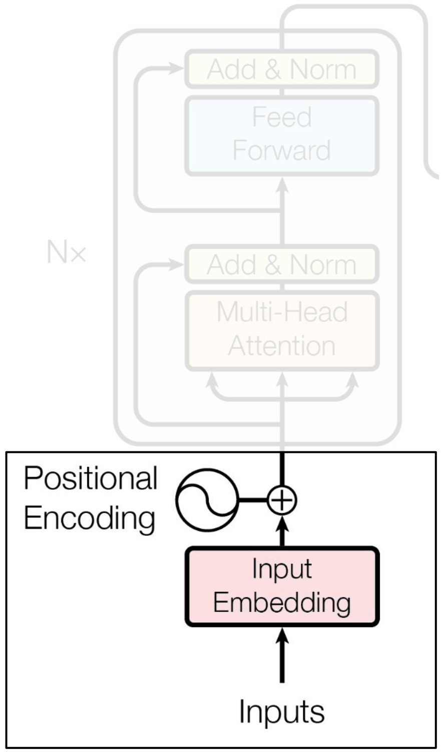
Which of the following are true about attention?

a. To calculate attention weights for input $\mathbf{I}_2$, you would use key $\mathbf{k}_2$, and all queries

b. To calculate attention weights for input $\mathbf{I}_2$, you would use query $\mathbf{q}_2$, and all keys

c. We scale the QK^T product to bring attention weights in the range of [0,1]

d. We scale the QK^T product to allow for numerical stability



Poll 1 - @1581

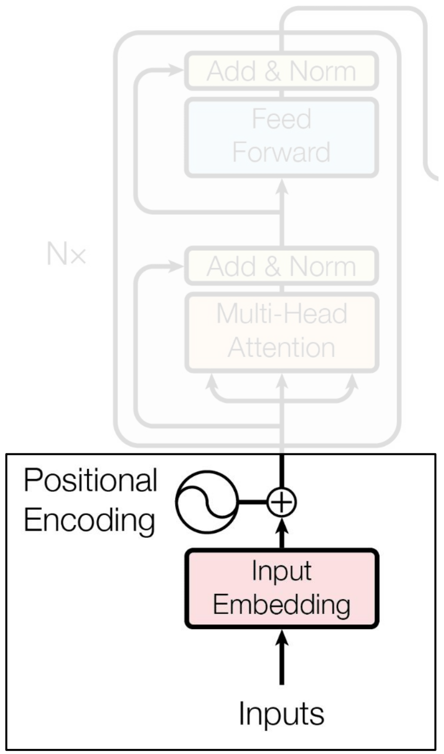
Which of the following are true about attention?

a. To calculate attention weights for input $\mathbf{I}_2$, you would use key $\mathbf{k}_2$, and all queries

b. **To calculate attention weights for input I_2, you would use query q_2, and all keys**

c. We scale the QK^T product to bring attention weights in the range of [0,1]

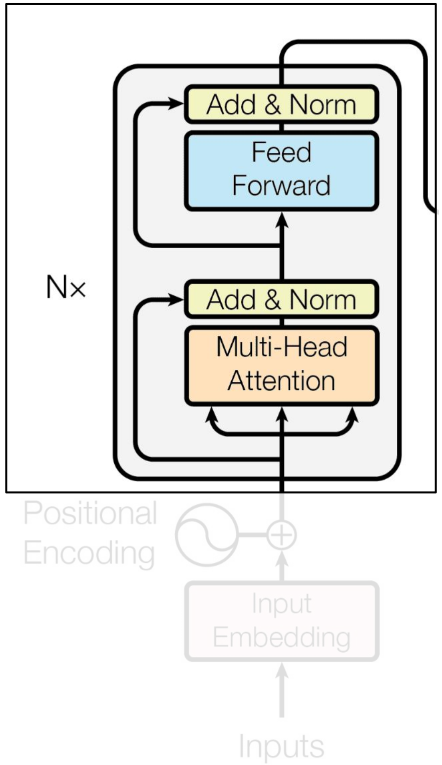
d. **We scale the QK^T product to allow for numerical stability**



Self Attention

From lecture 18:

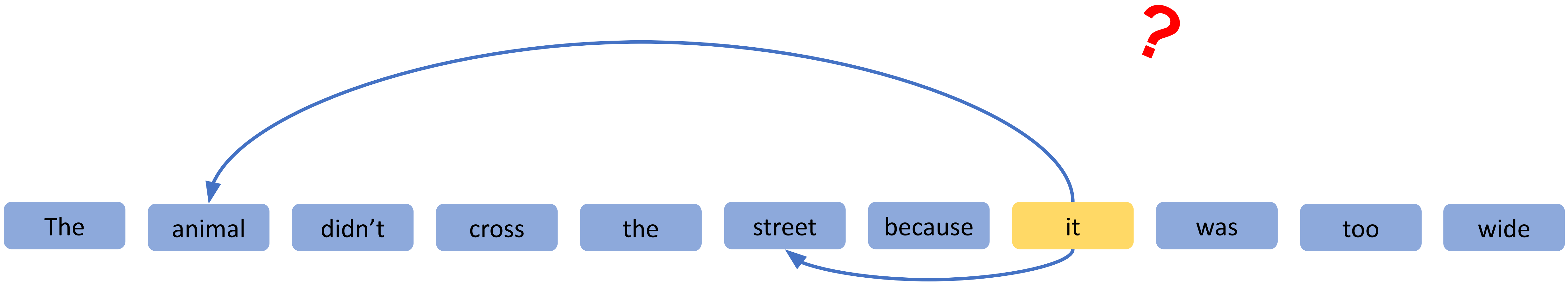
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Self Attention

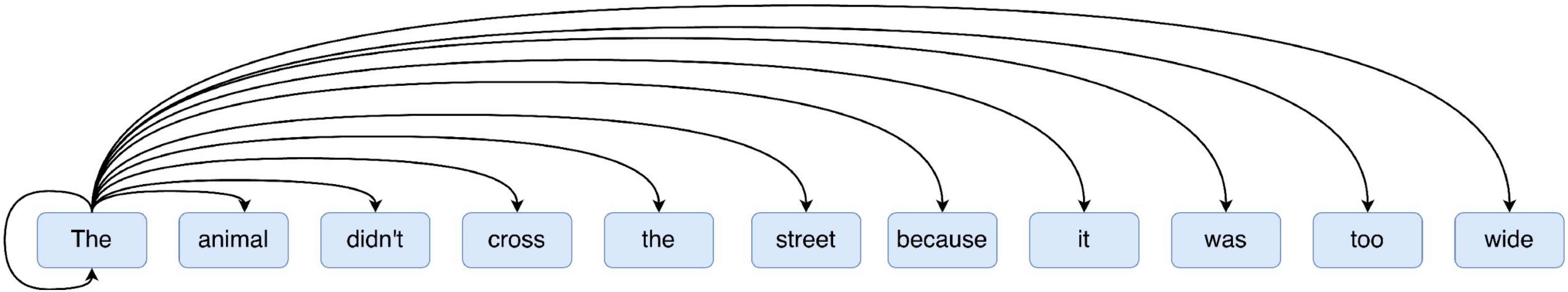
The animal didn't cross the street because it was too wide

Self Attention

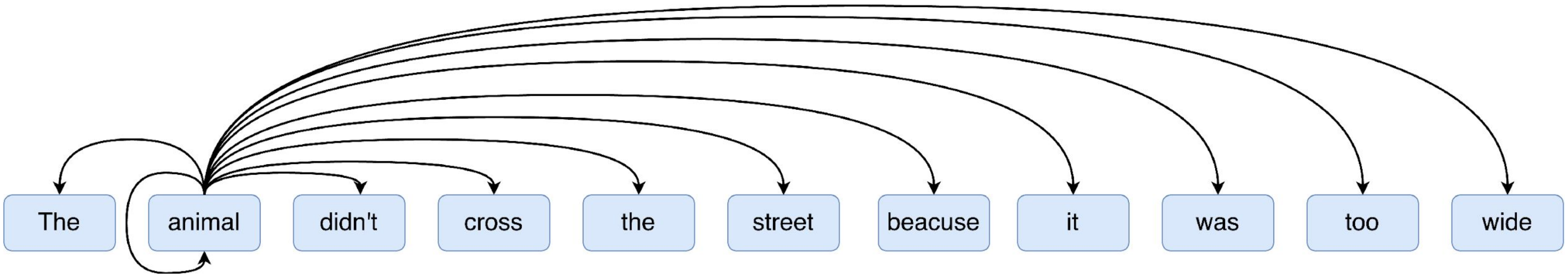


coreference resolution?

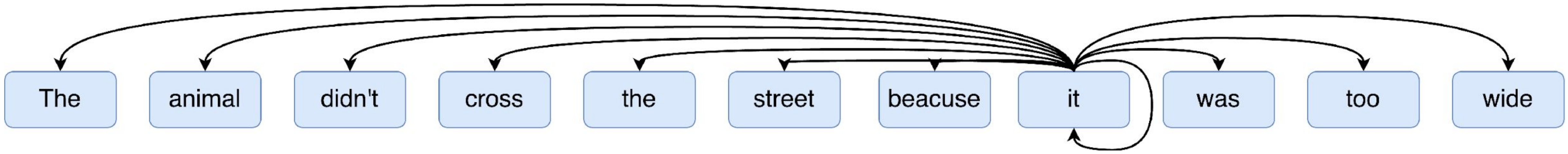
Self Attention



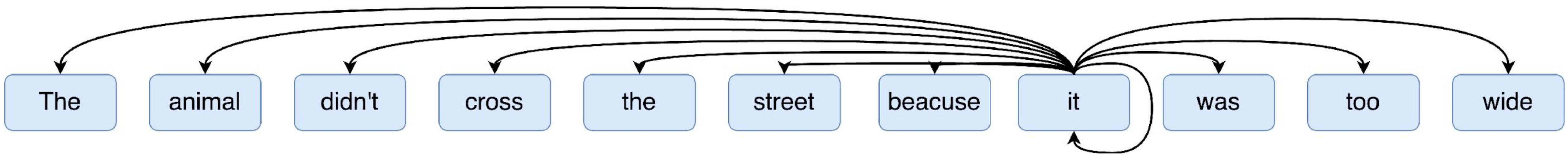
Self Attention



Self Attention



Self Attention



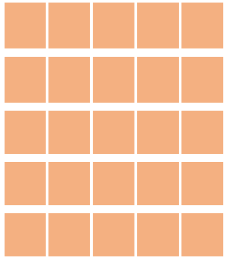
SELF

Query Inputs = Key Inputs = Value Inputs

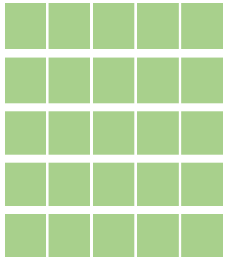
Self Attention

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

W_Q

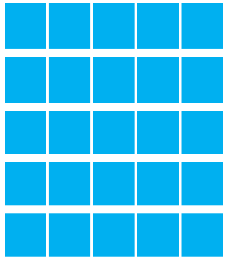


W_K

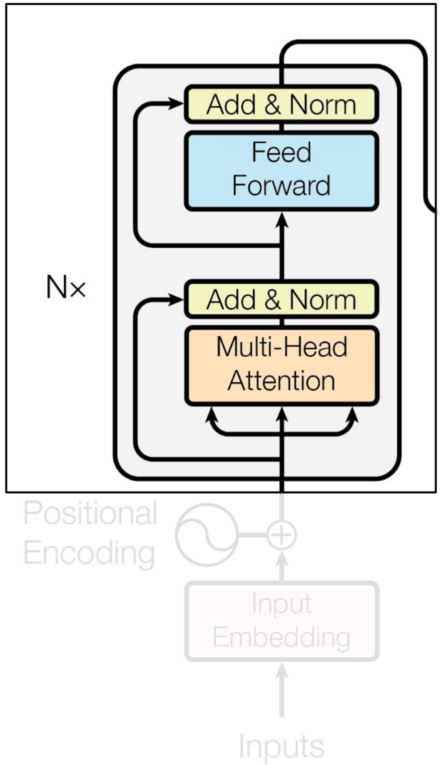


W_V

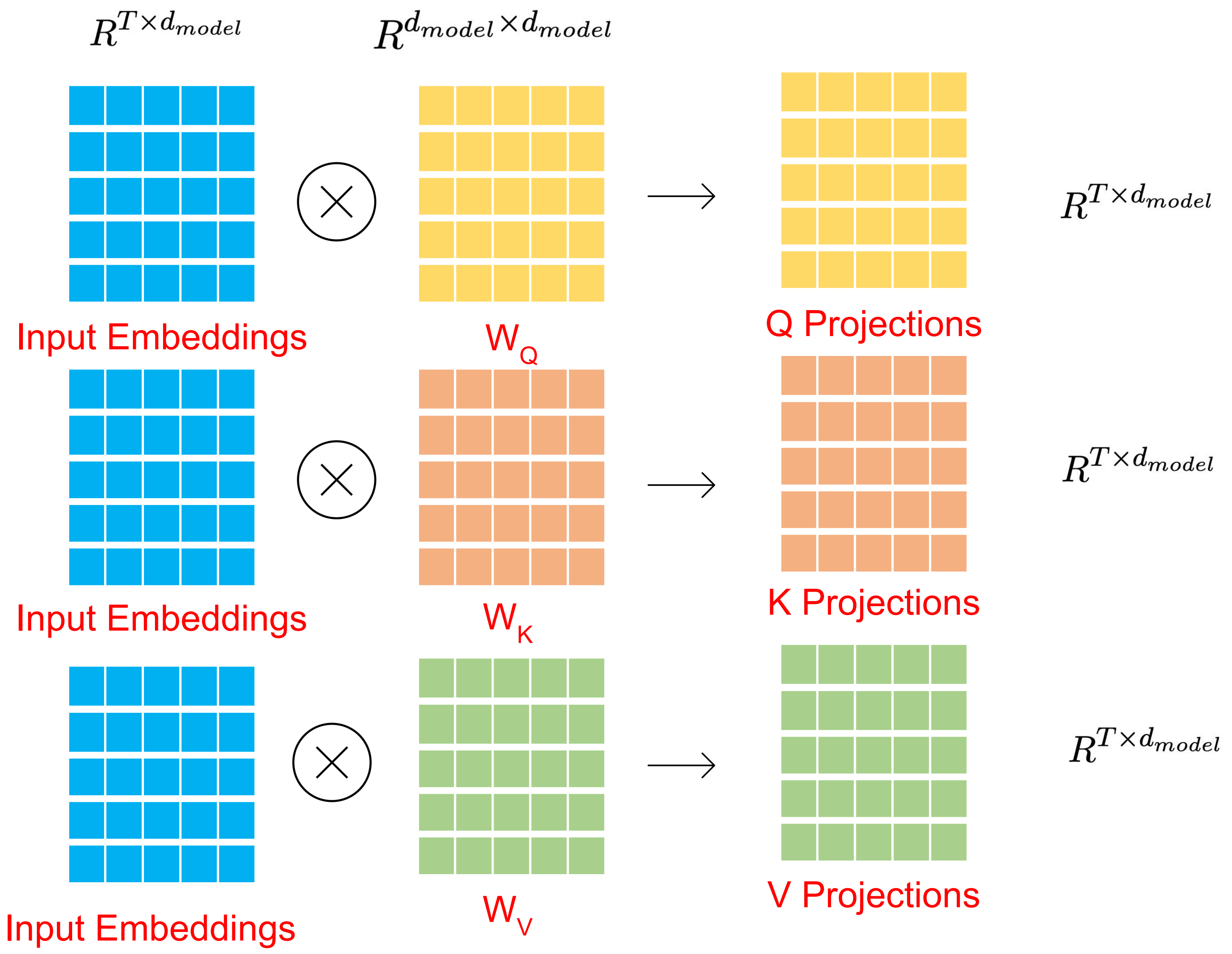
$R^{T \times d_{model}}$

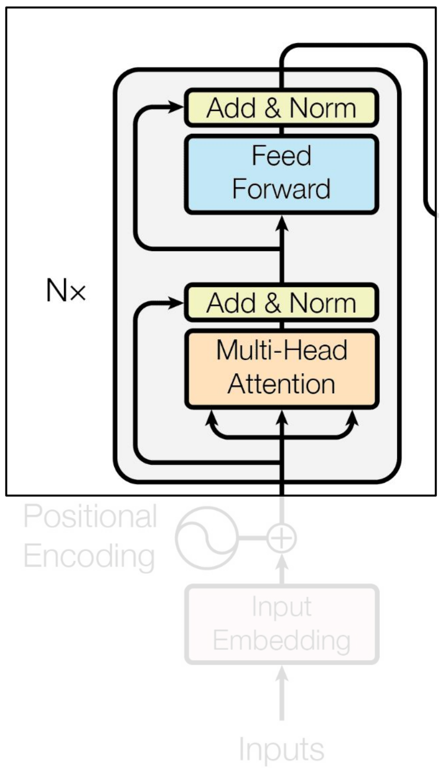


Input Embeddings



Self Attention



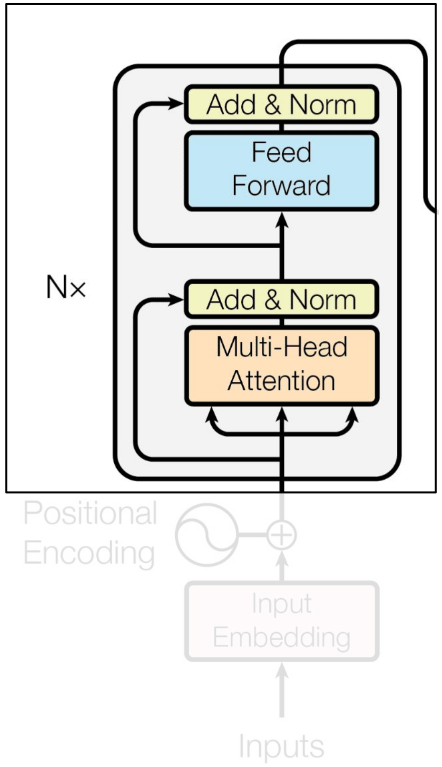


Self Attention

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

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

$R^{T\times T}$



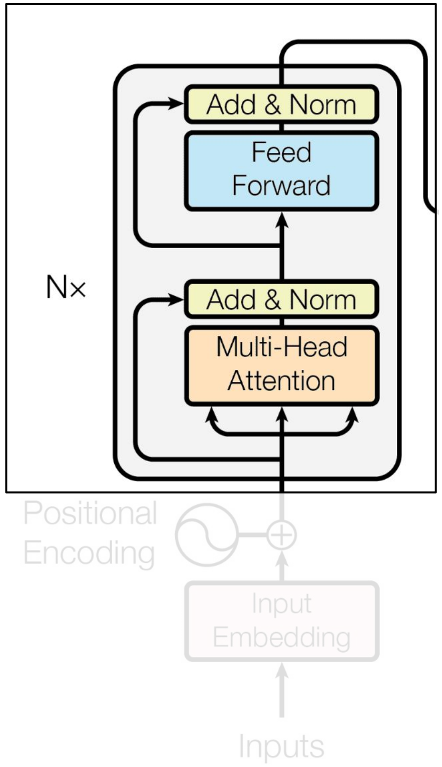
Self Attention

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

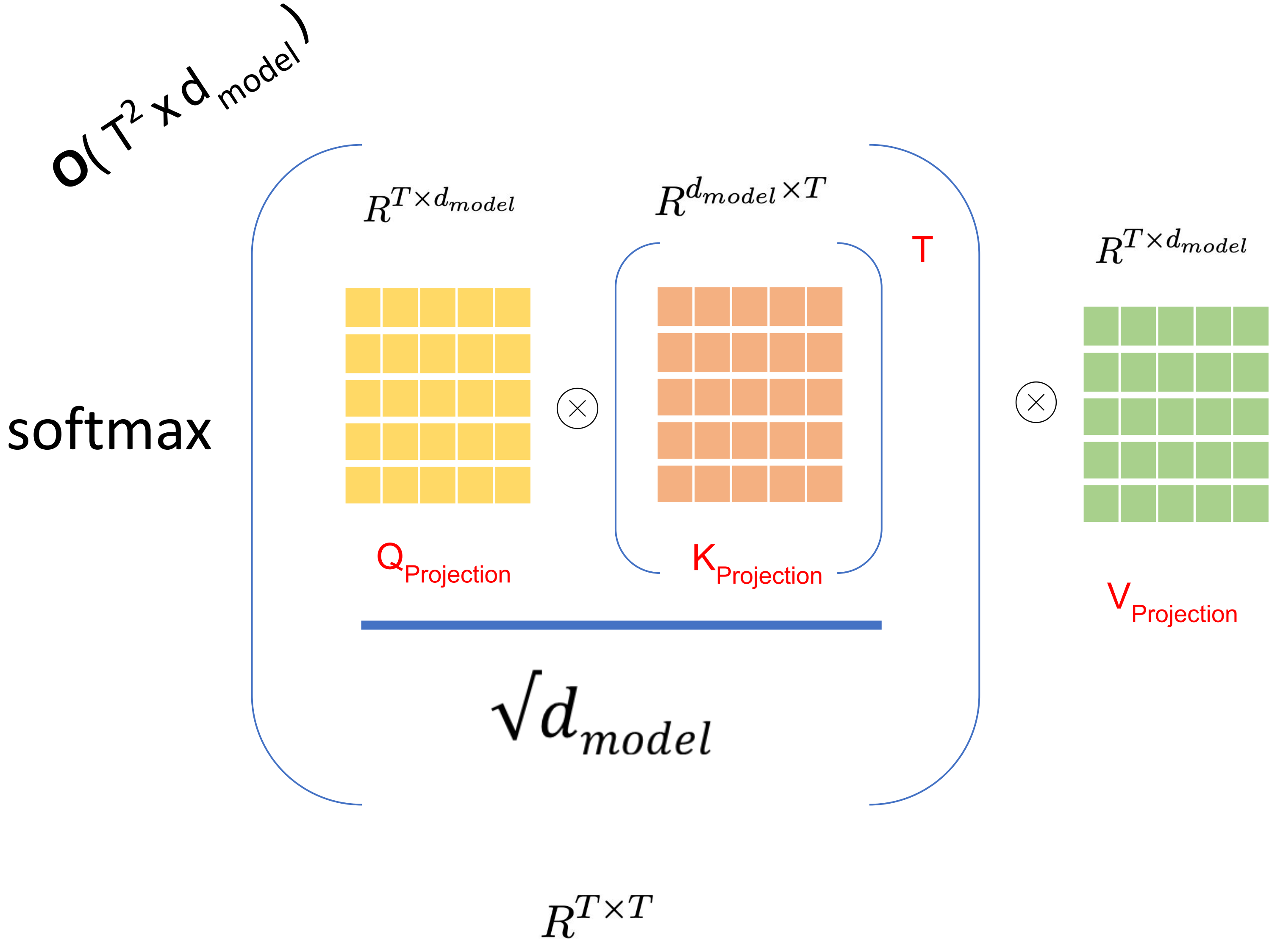
$R^{T \times d_{model}}$ $\quad$ $R^{d_{model} \times T}$

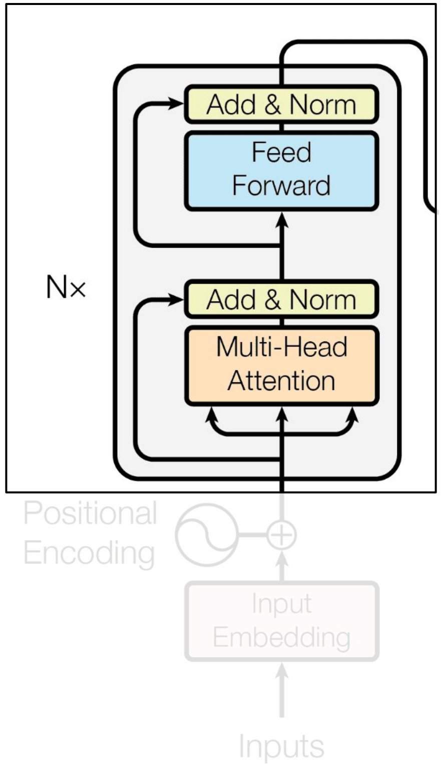
$$\text{softmax}\left(\frac{Q_{Projection} \otimes K_{Projection}^{T}}{\sqrt{d_{model}}} \right)$$

$R^{T \times T}$



Self Attention

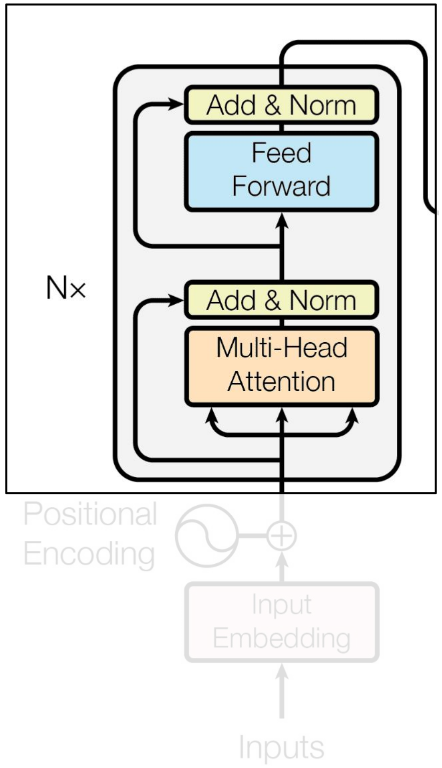




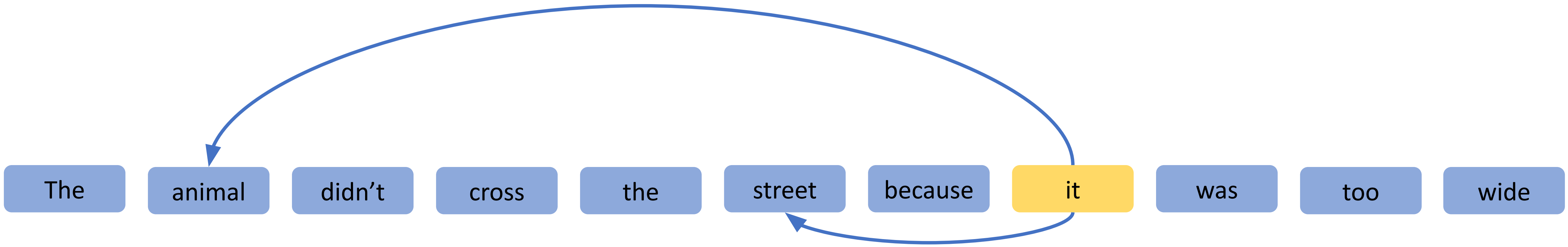
Self Attention

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

Attention: Z

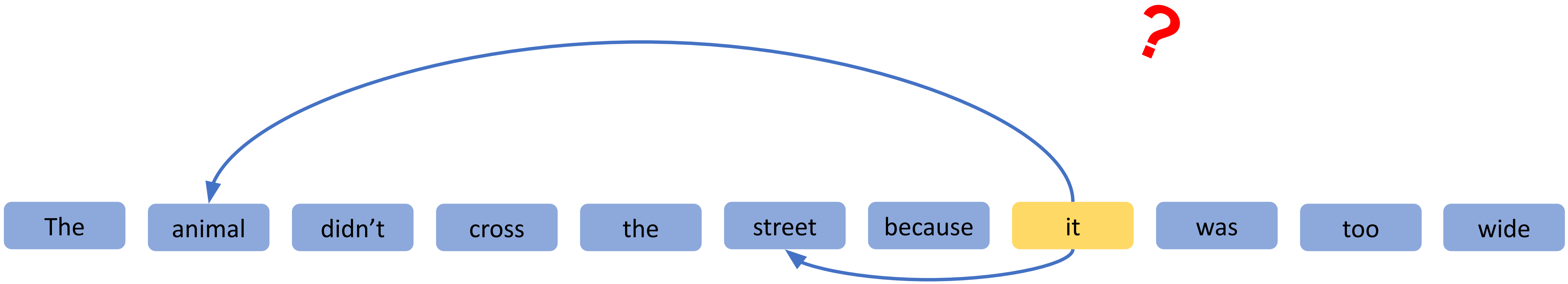


Self Attention



Coreference resolution

Self Attention



Sentence boundaries ?

Coreference resolution ✓

Context ?

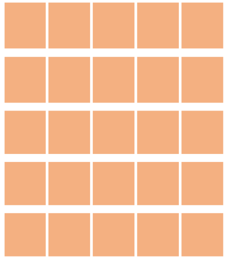
Semantic relationships ?

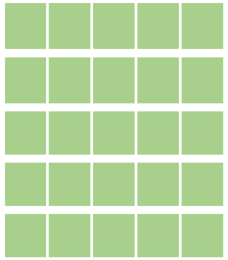
Part of Speech ?

Comparisons ?

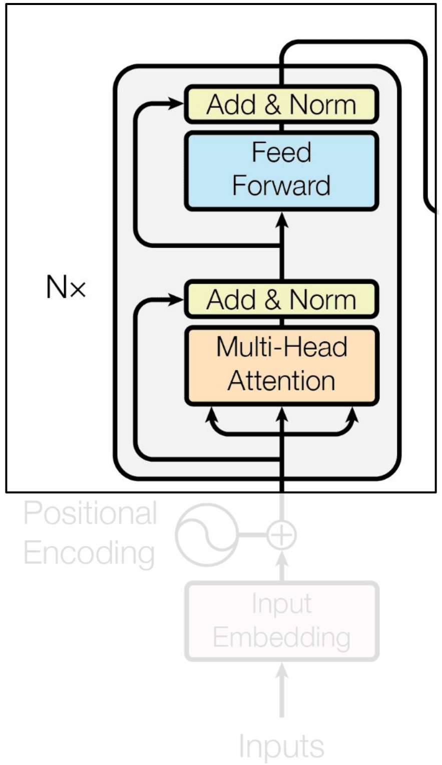
Self Attention

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

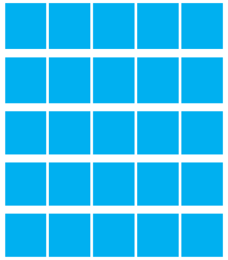




W_Q W_K W_V



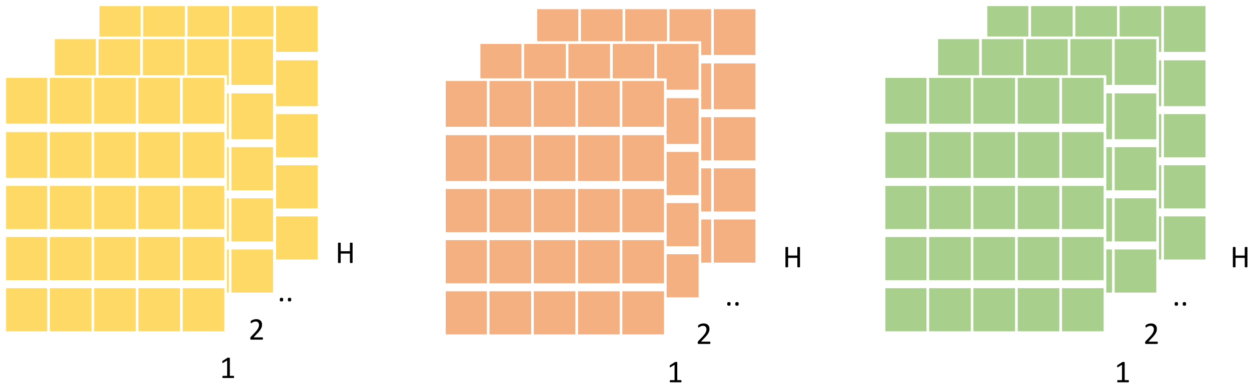
$$R^{T \times d_{model}}$$



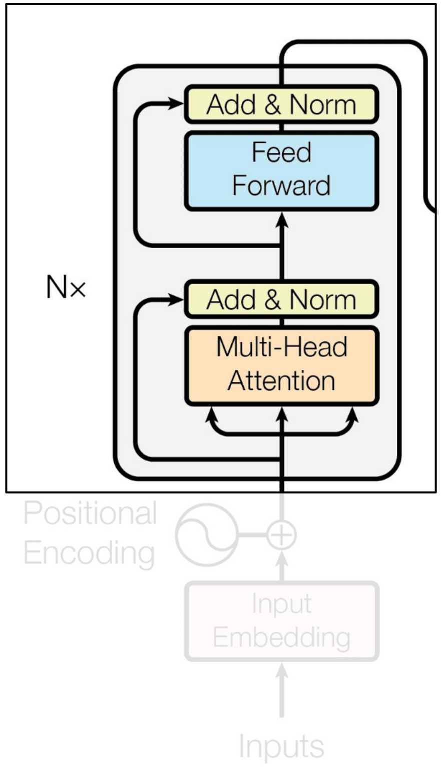
Input Embeddings

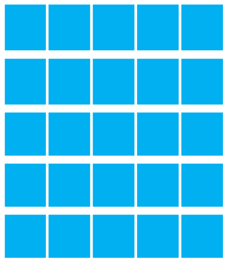
Multi-Head Attention

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



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



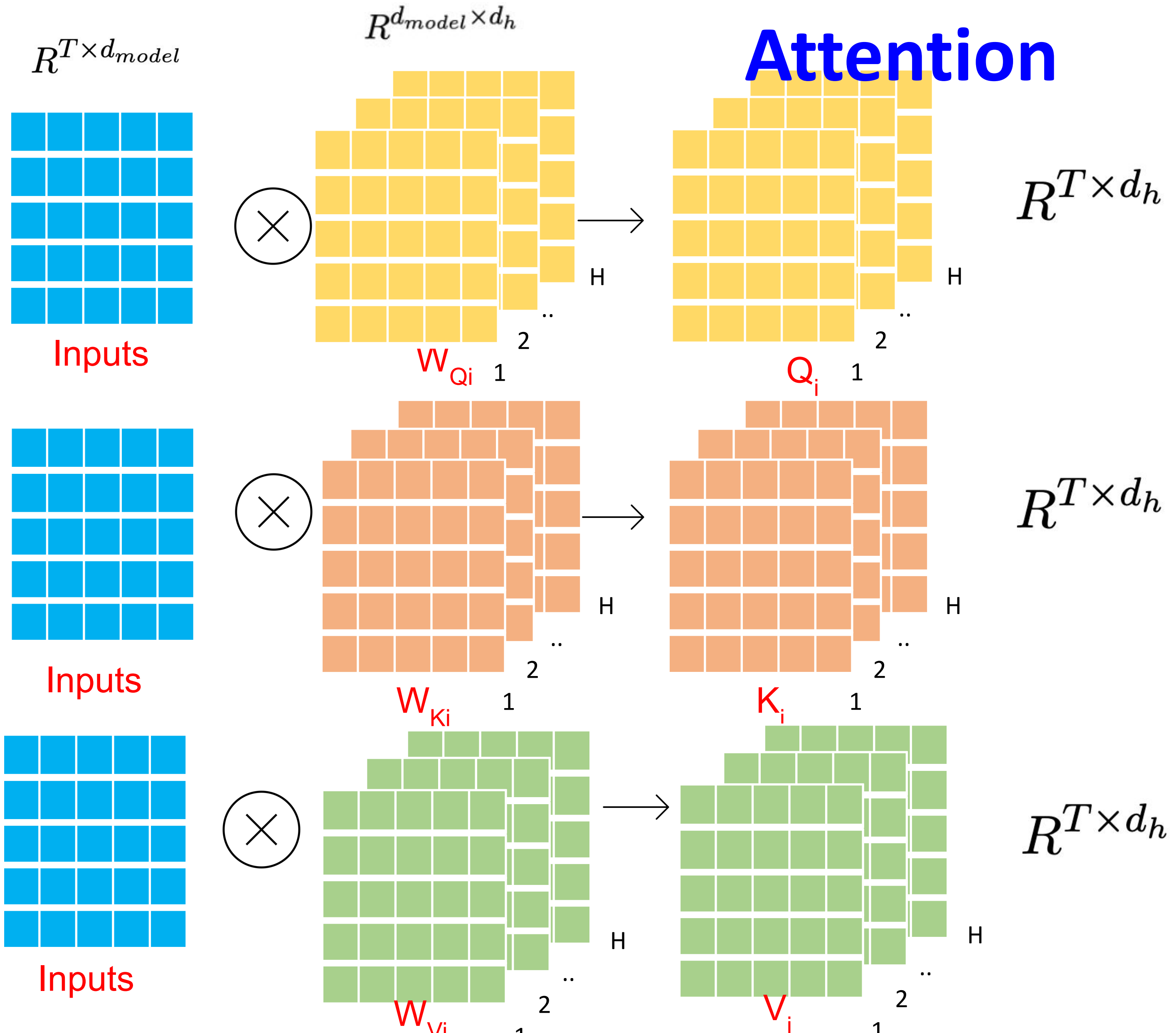


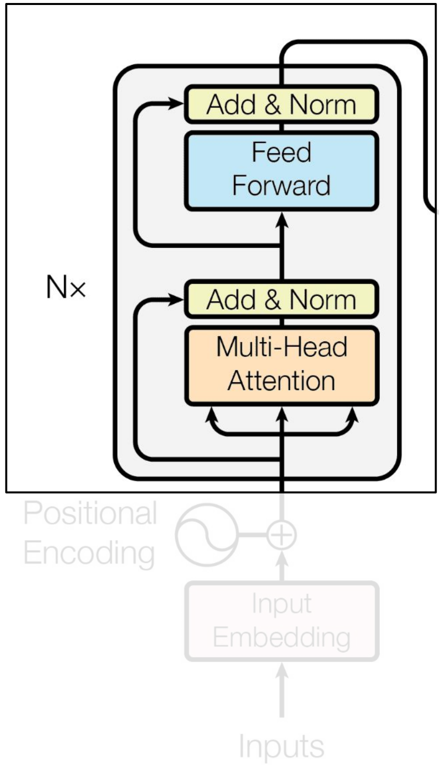
Input Embeddings

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

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

Multi-Head Attention



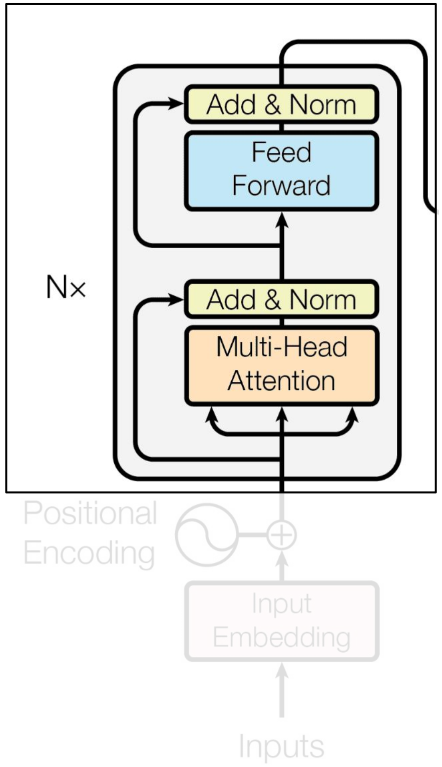


Multi-Head Attention

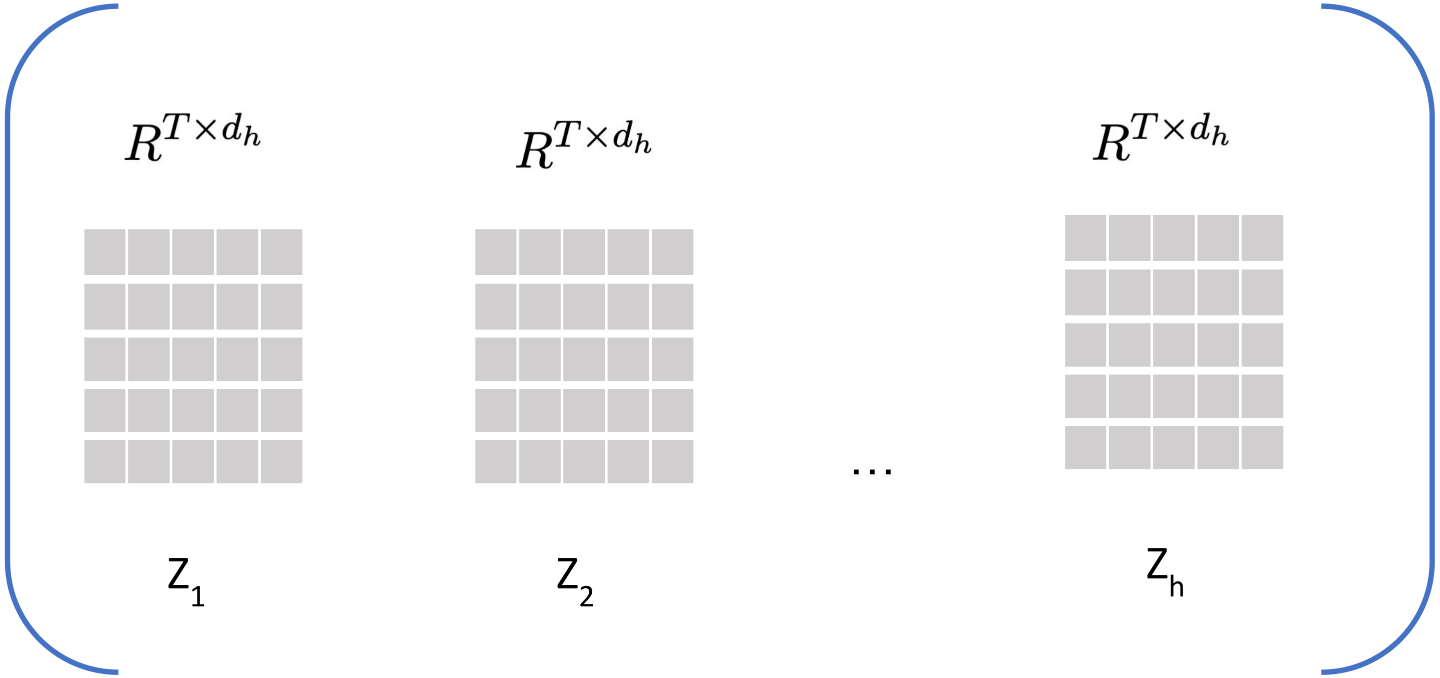
$$\text{softmax}\left(\frac{\overset{R^{T\times d_h}}{Q_i} \otimes \overset{R^{d_h\times T}}{(K_i)^T}}{\sqrt{d_{model}}}\right) \otimes \overset{R^{T\times d_h}}{V_i}$$

$R^{T\times T}$

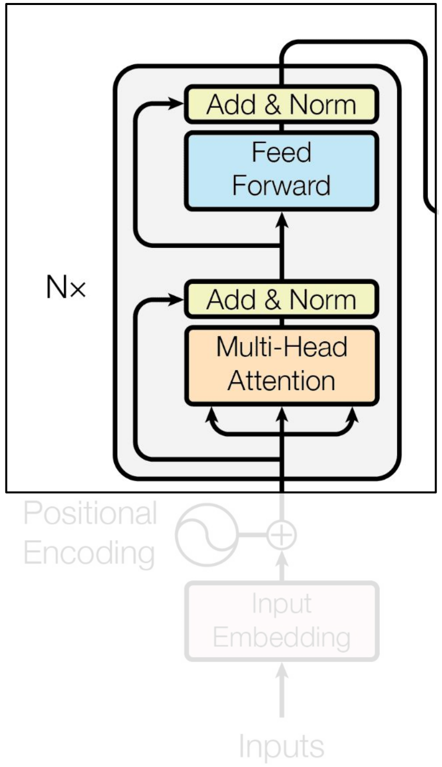
for all i ∈ [1, h]



Multi-Head Attention



CONCAT

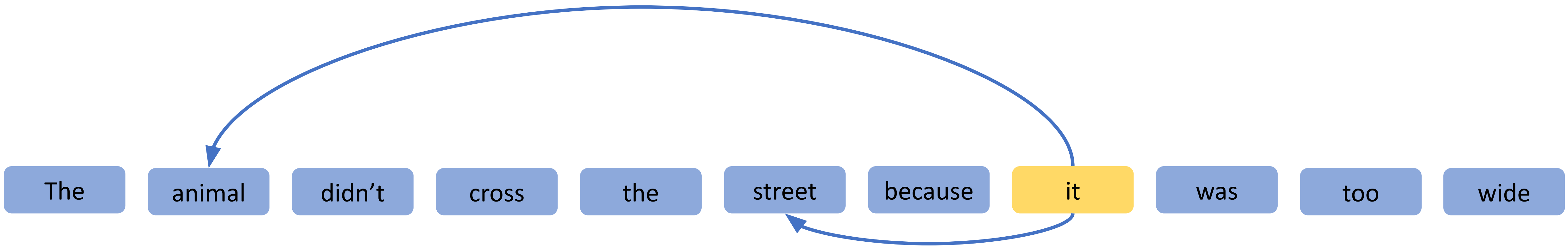


Multi Head Attention : Z

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

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

Multi-Head Attention



Sentence boundaries ✓

Coreference resolution ✓

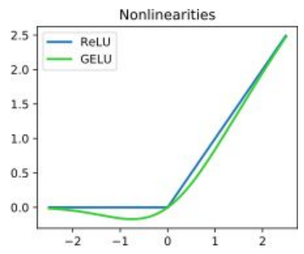
Context ✓

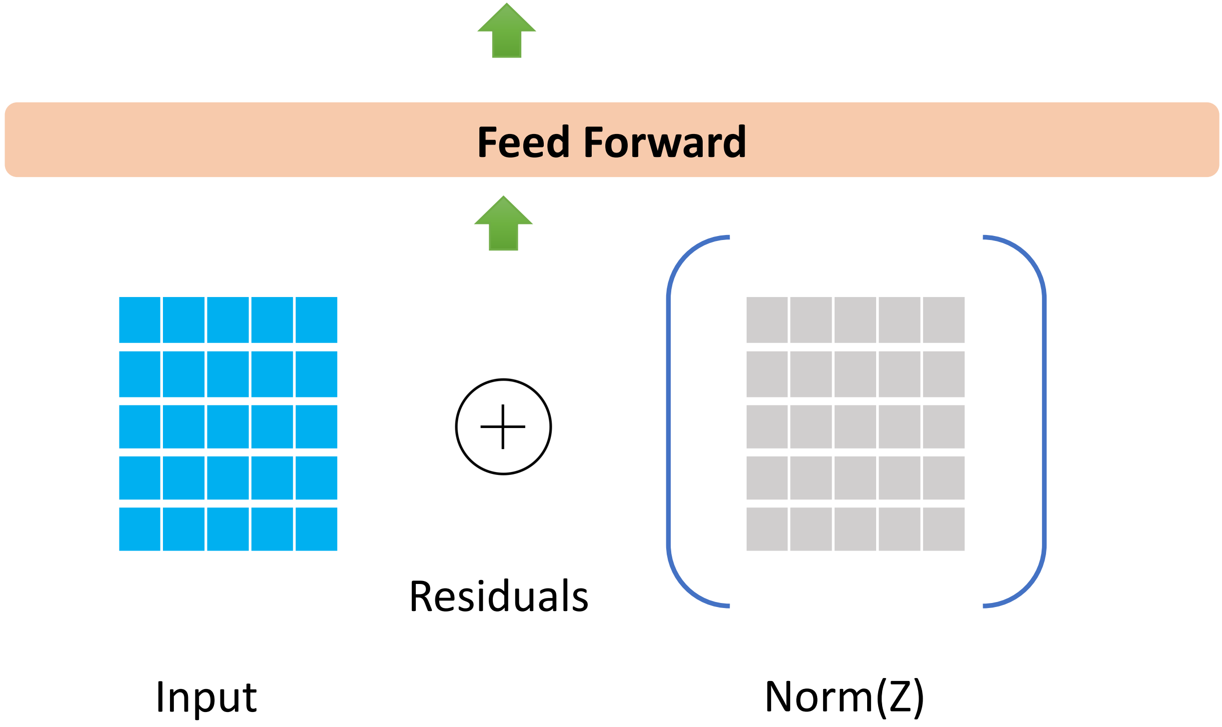
Semantic relationships ✓

Part of speech ✓

Comparisons ✓

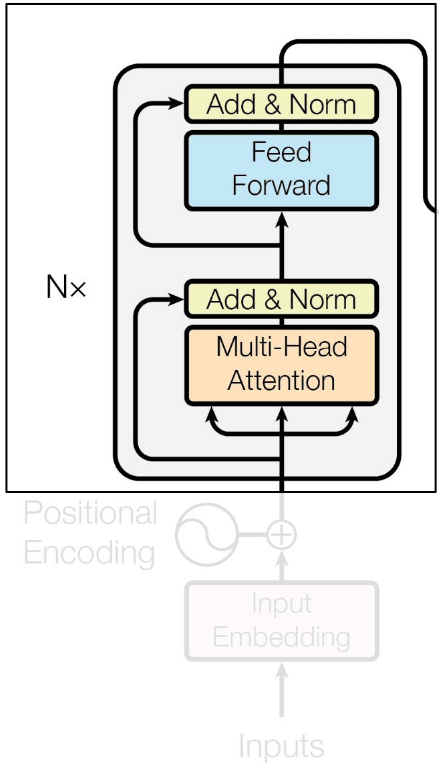
Feed Forward



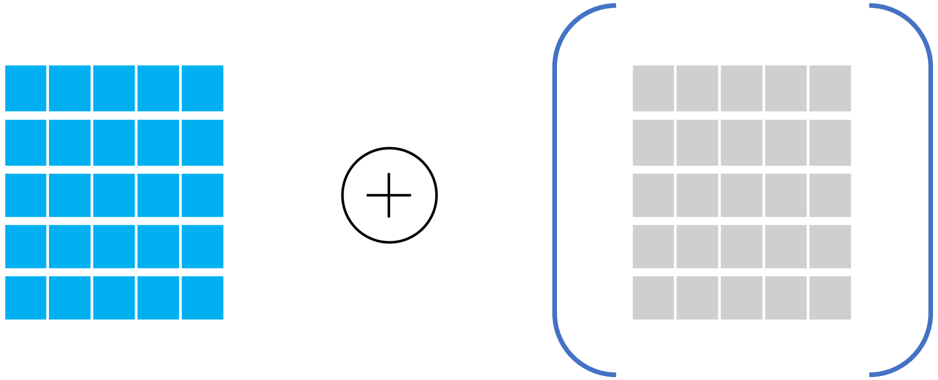


Feed Forward

- Non Linearity
- Complex Relationships
- Learn from each other

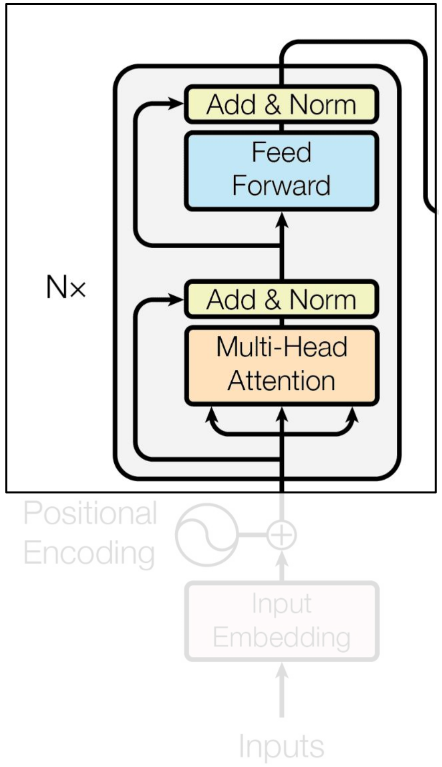


Add & Norm



Input

Norm(Z)



Normalization

Mean 0, Std dev 1

Stabilizes training

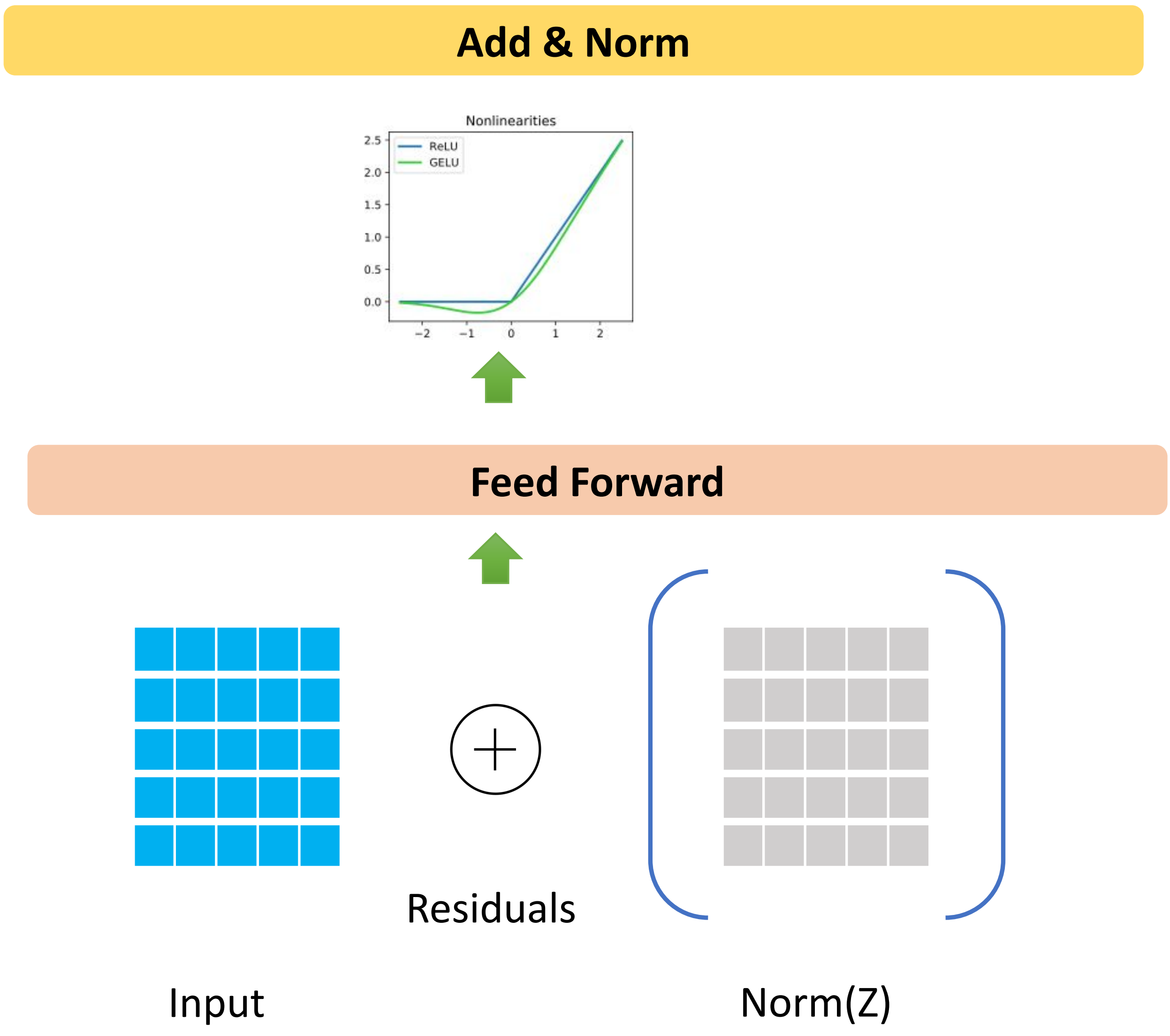
Regularization effect

Add Residuals

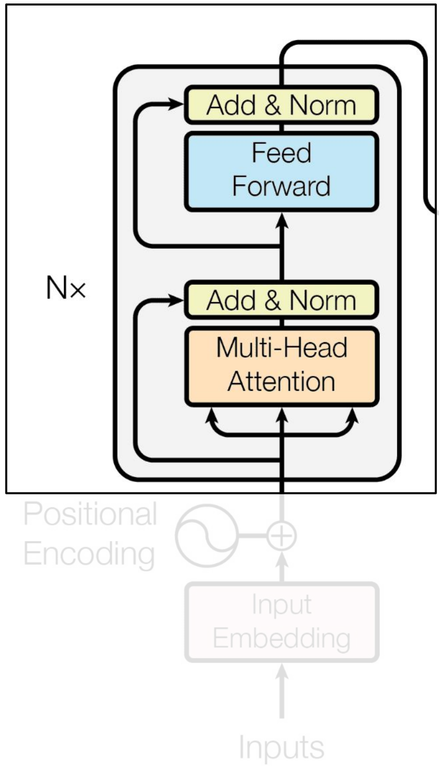
Avoid vanishing gradients

Train deeper networks

Add & Norm



Add & Norm
Nonlinearities
ReLU
GELU
2.5
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2
Feed Forward
Residuals
Input
Norm(Z)

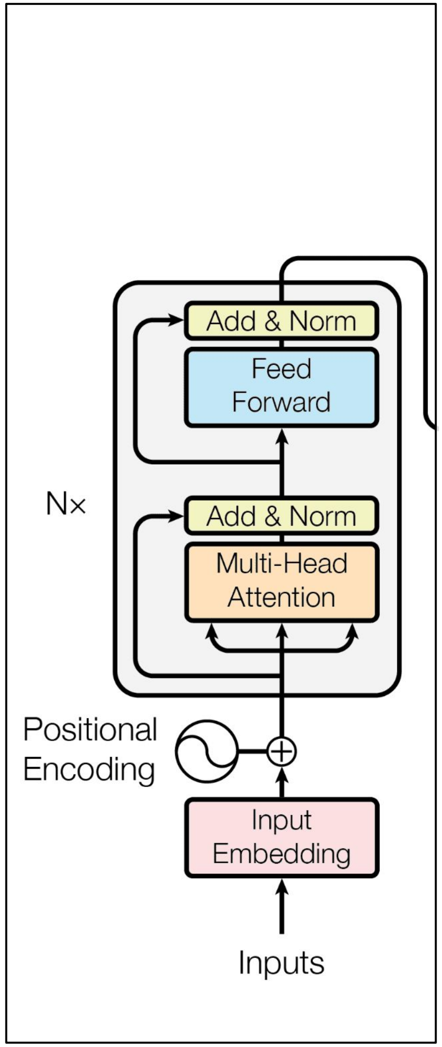


Add & Norm
Feed Forward
N×
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

Encoders

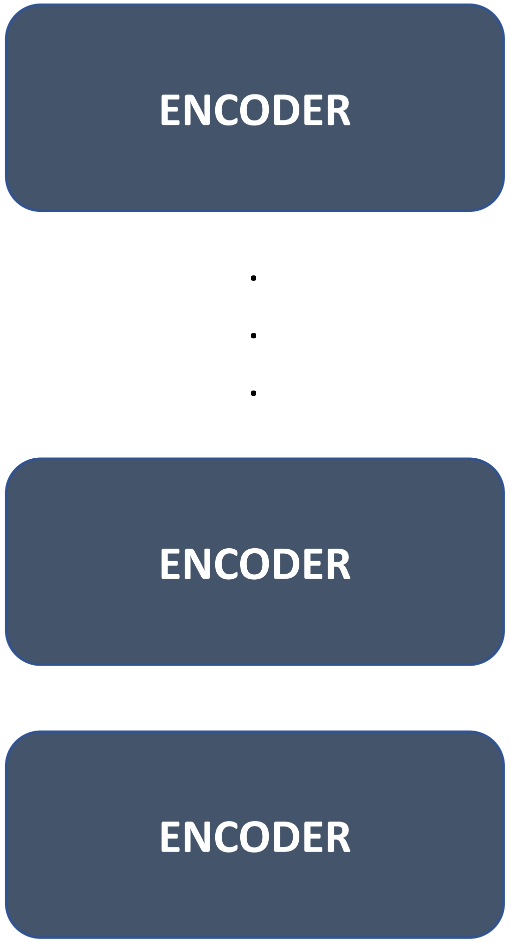
Encoder

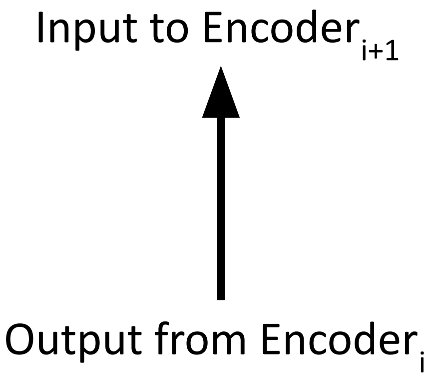
ENCODER

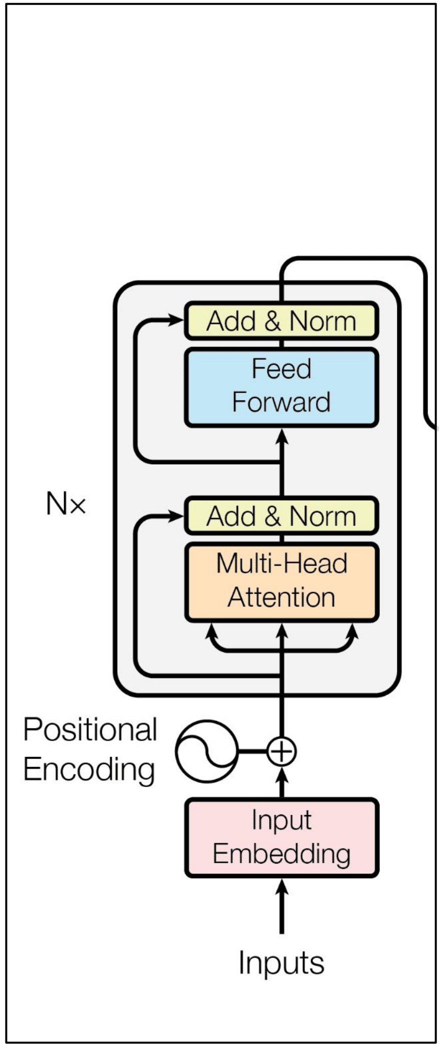


Encoders

Encoder



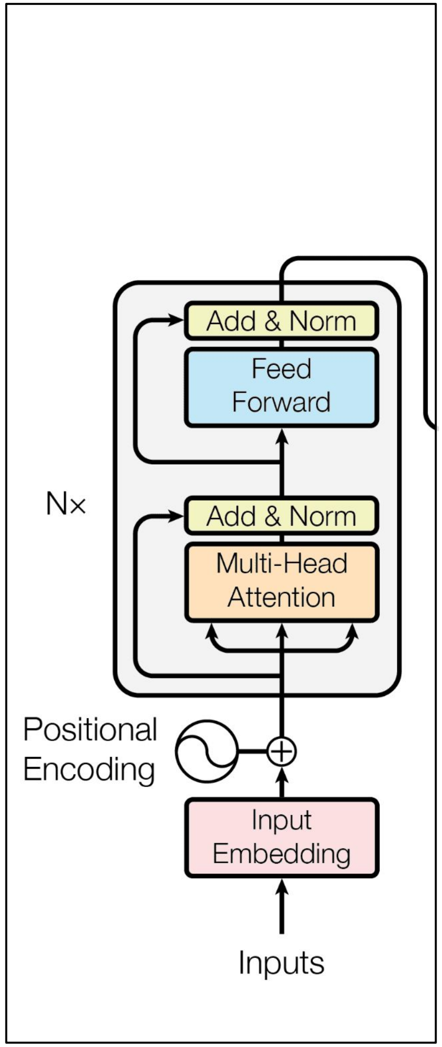




Transformers

- ✔ 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



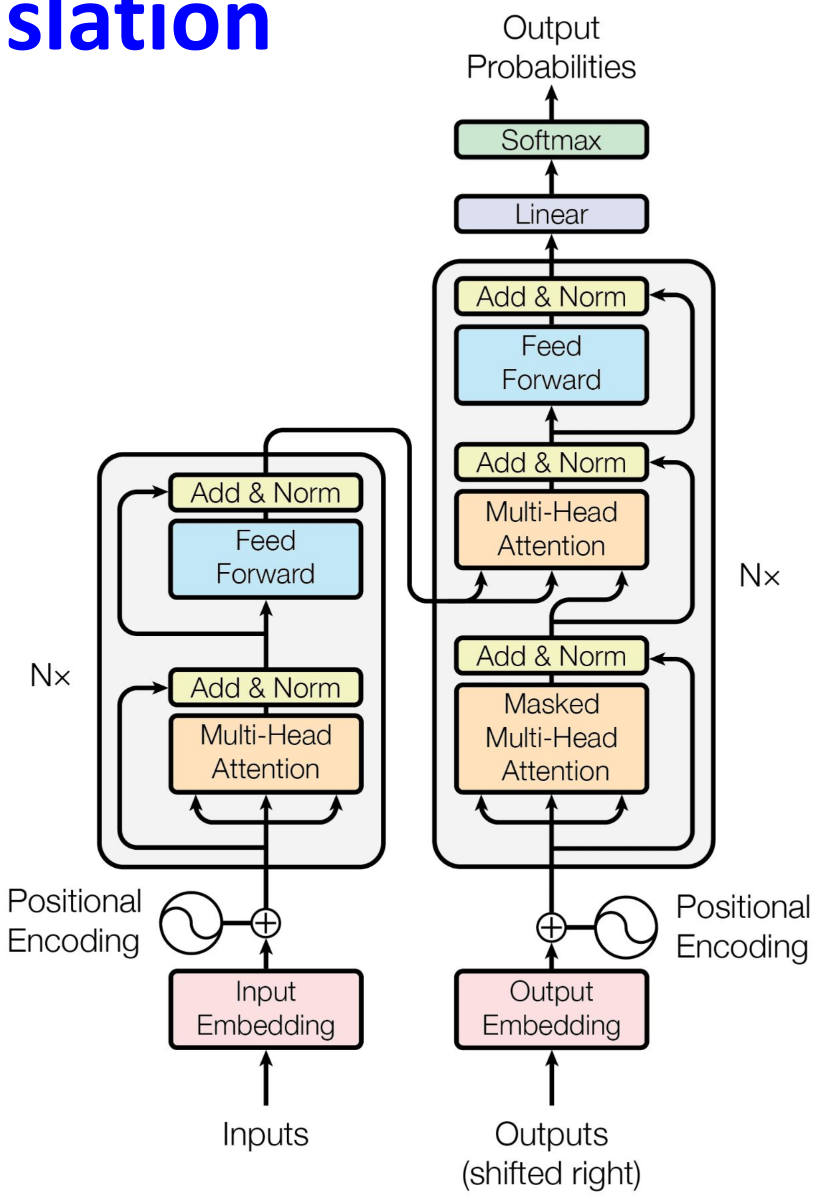
Machine Translation

Targets

Ich habe einen Apfel gegessen

Inputs

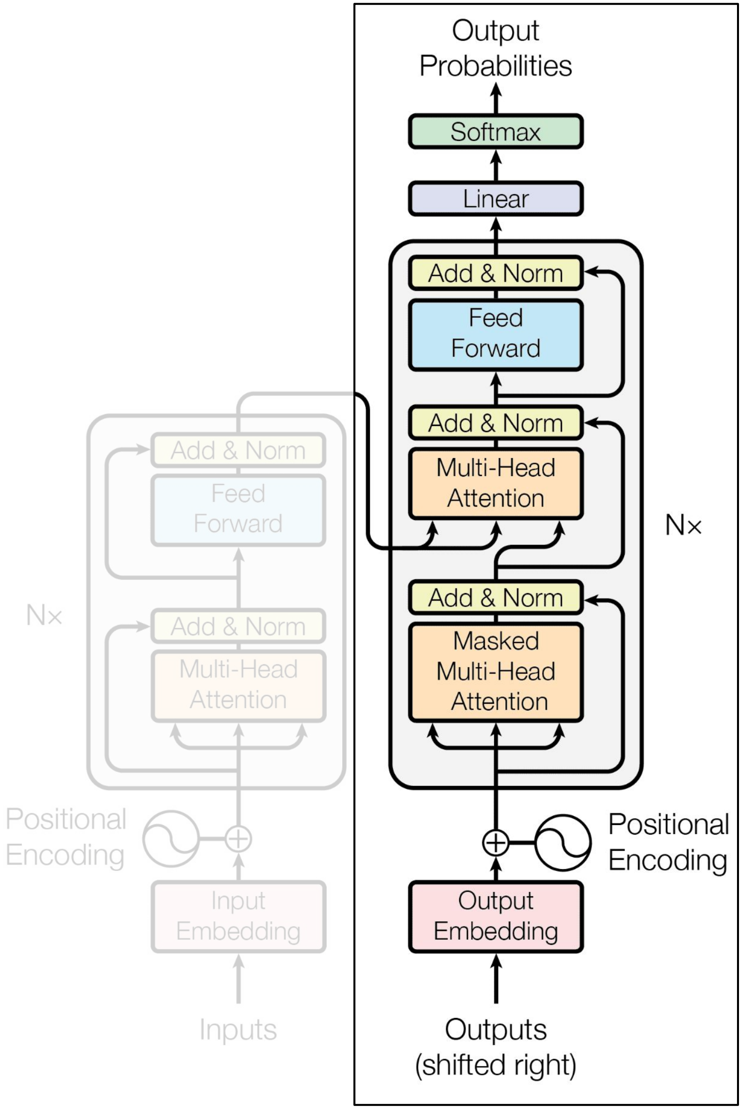
I ate an apple



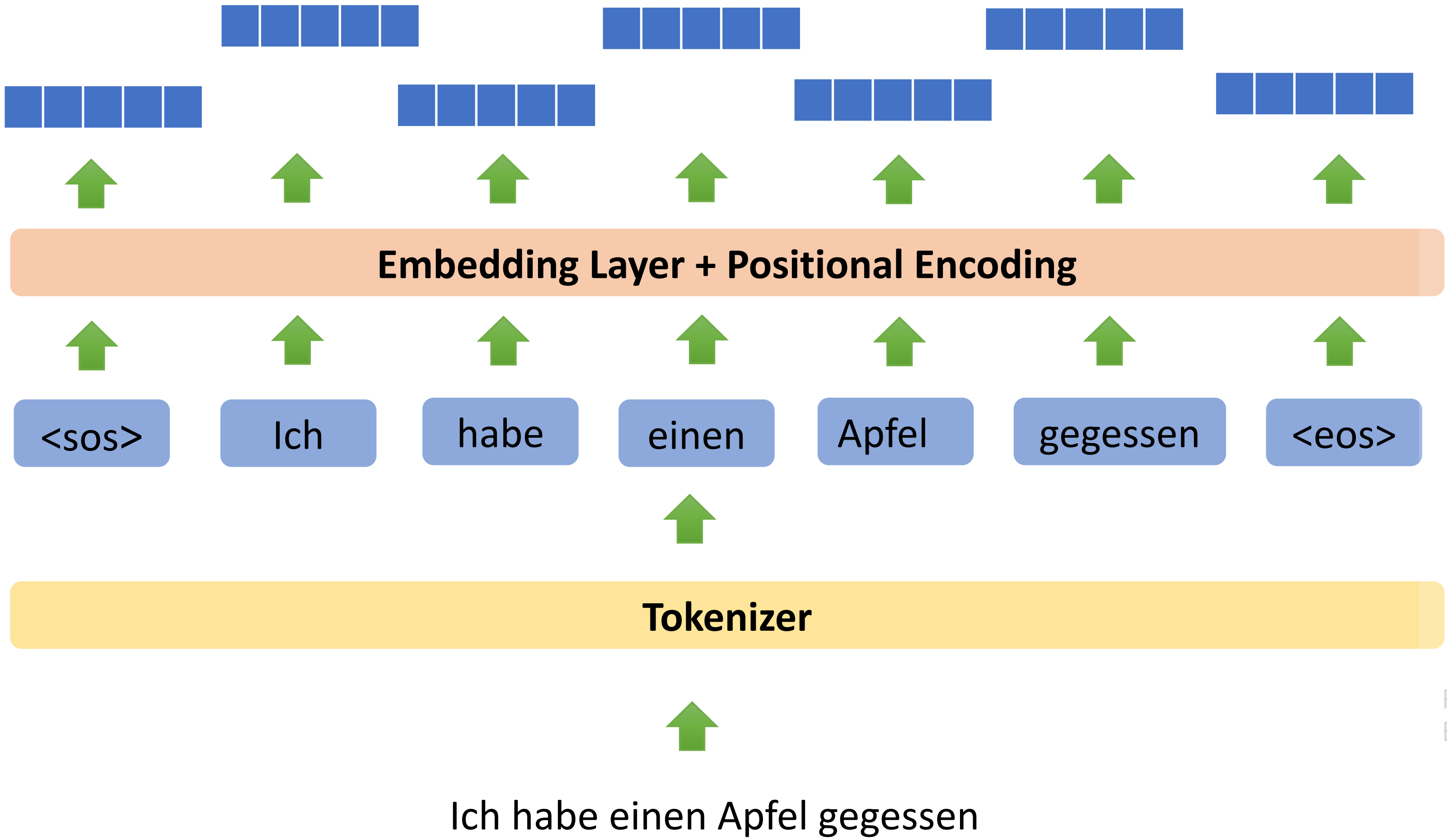
Targets

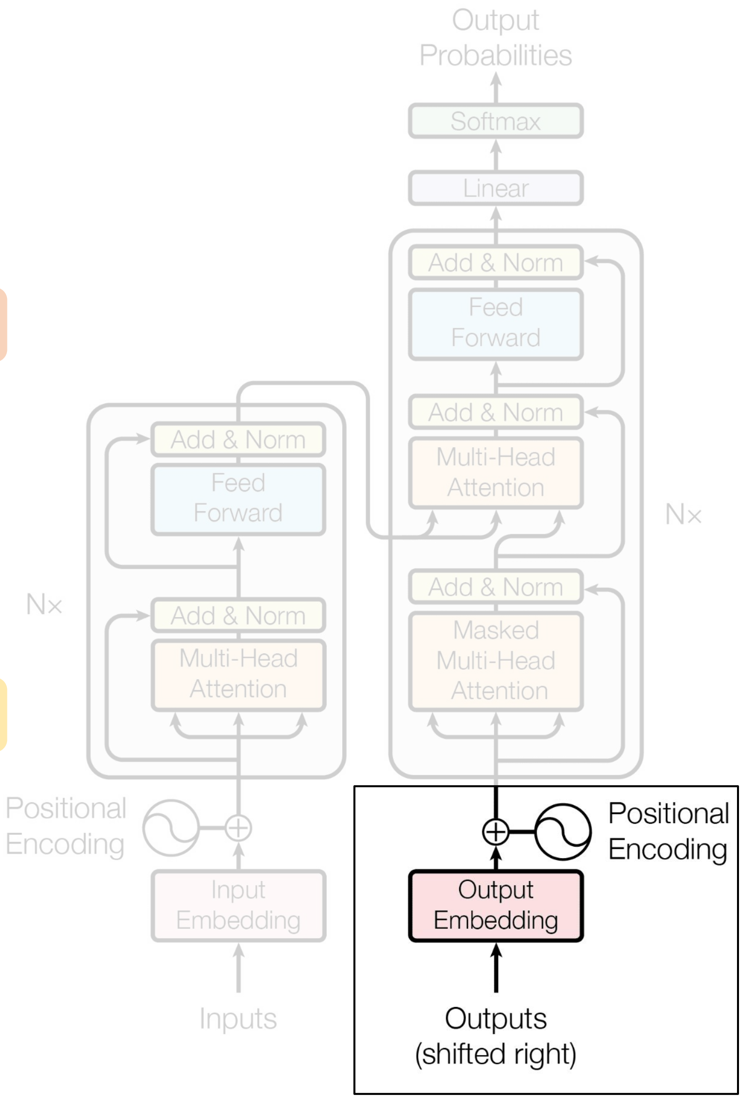
Targets

Ich habe einen Apfel gegessen



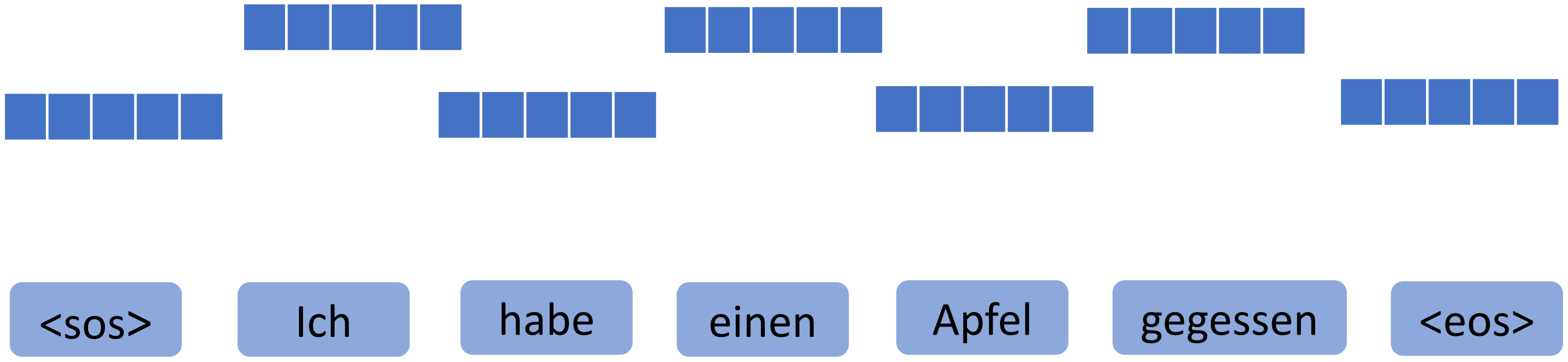
Targets

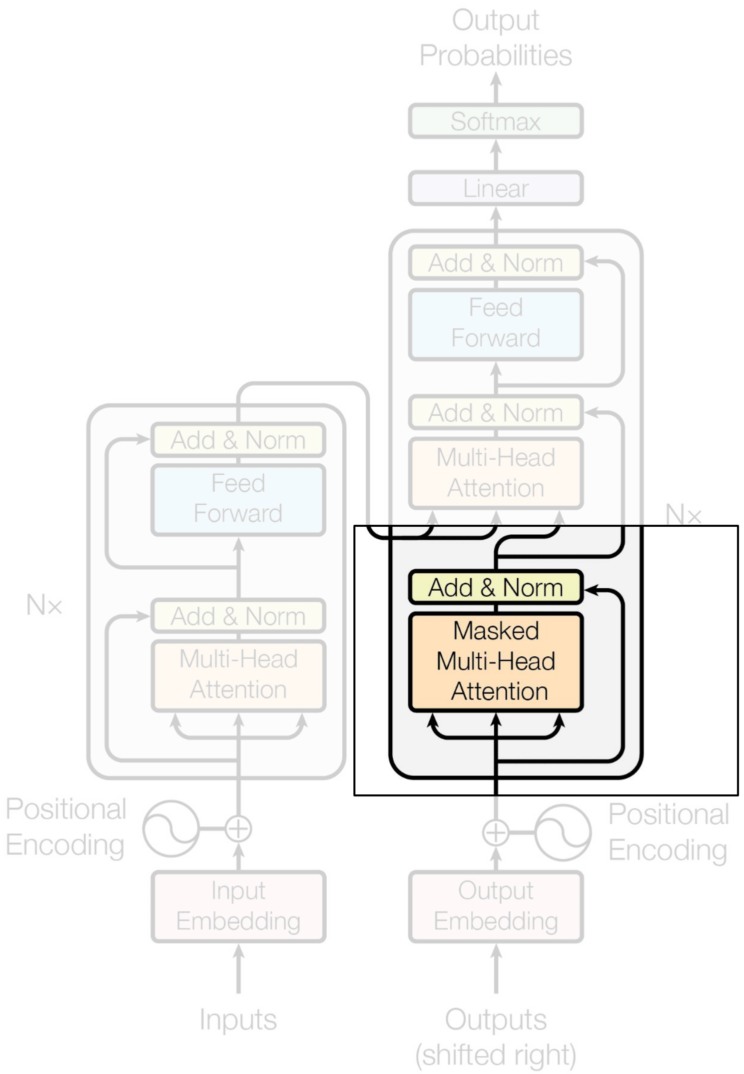




Generate Target Embeddings

Masked Multi Head Attention

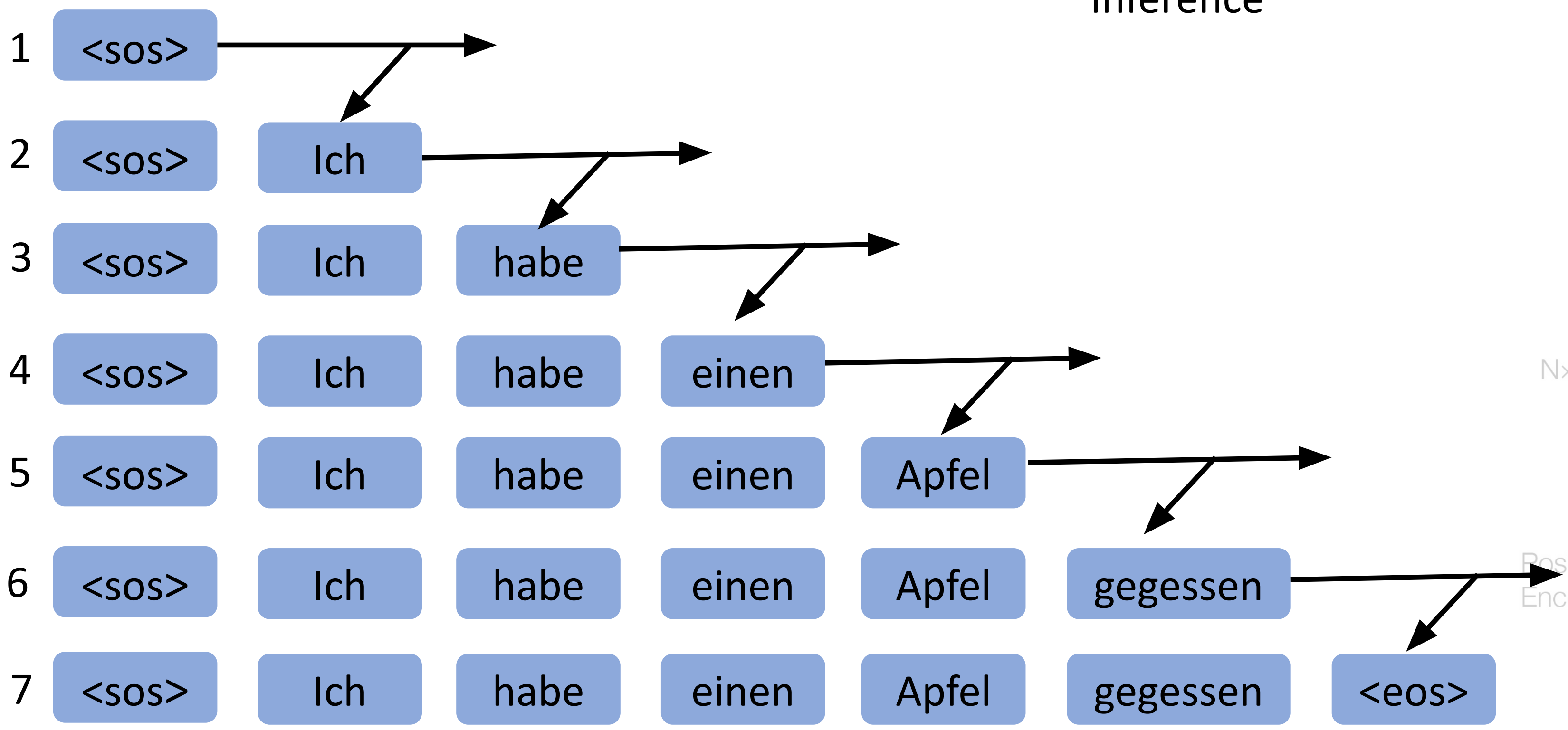


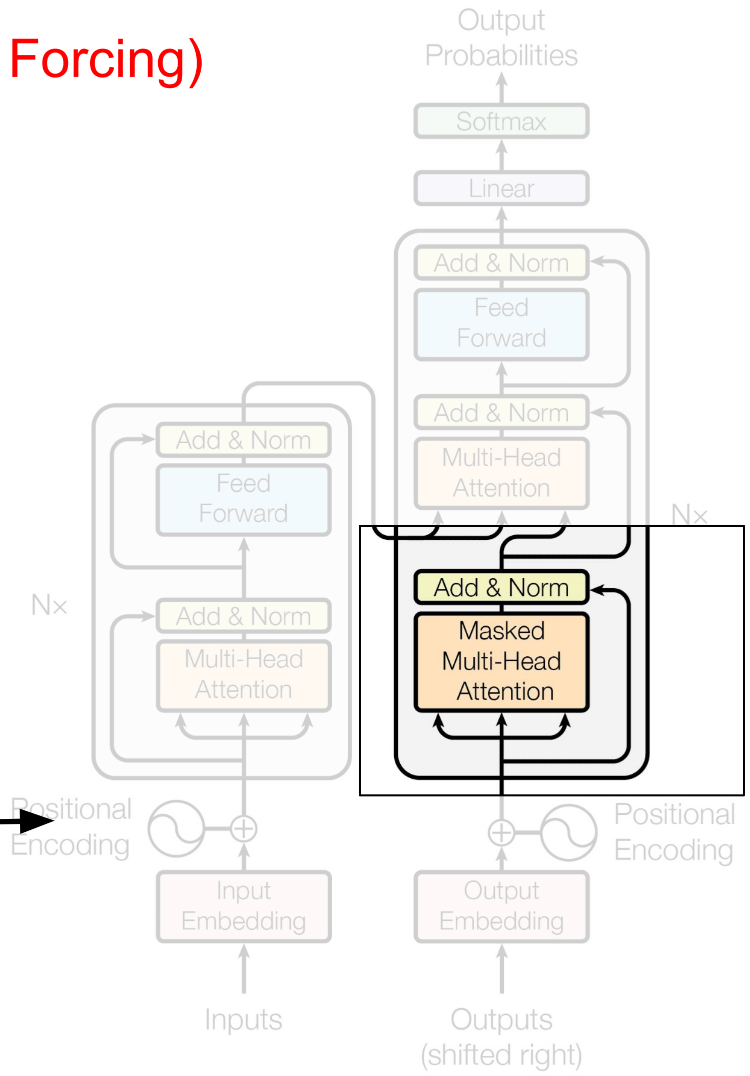


Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

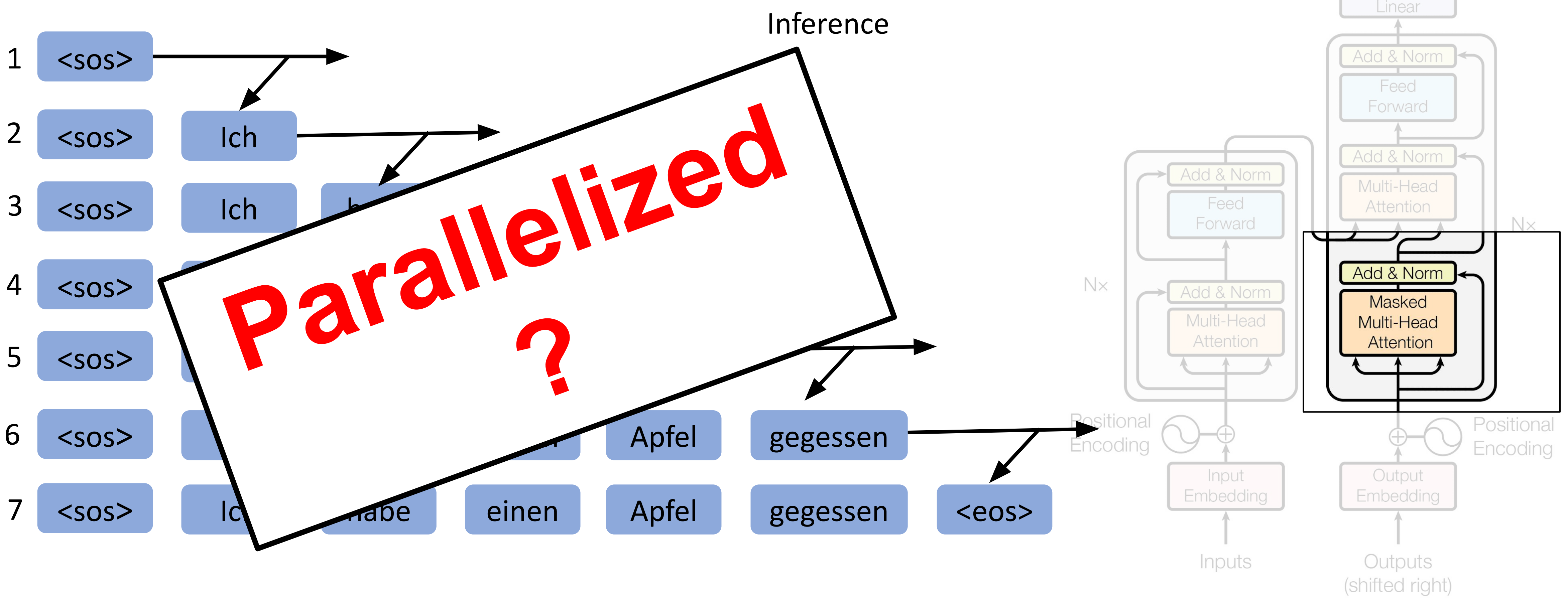
Inference





Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

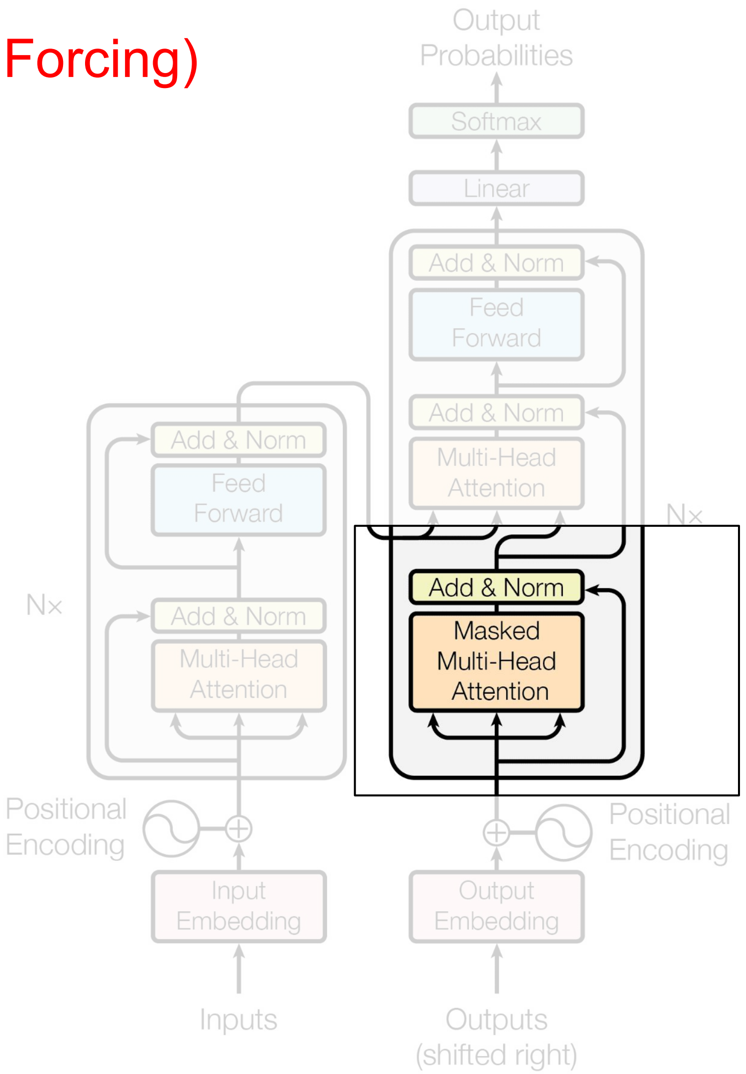


Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

Training

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



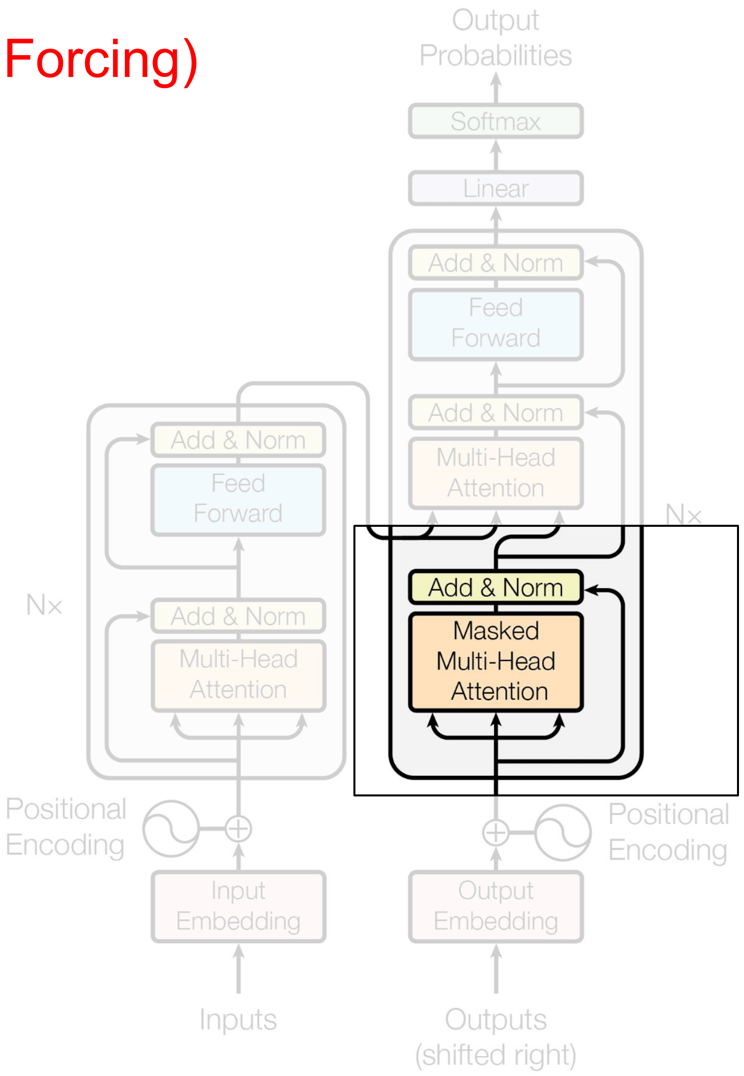
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

Training

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

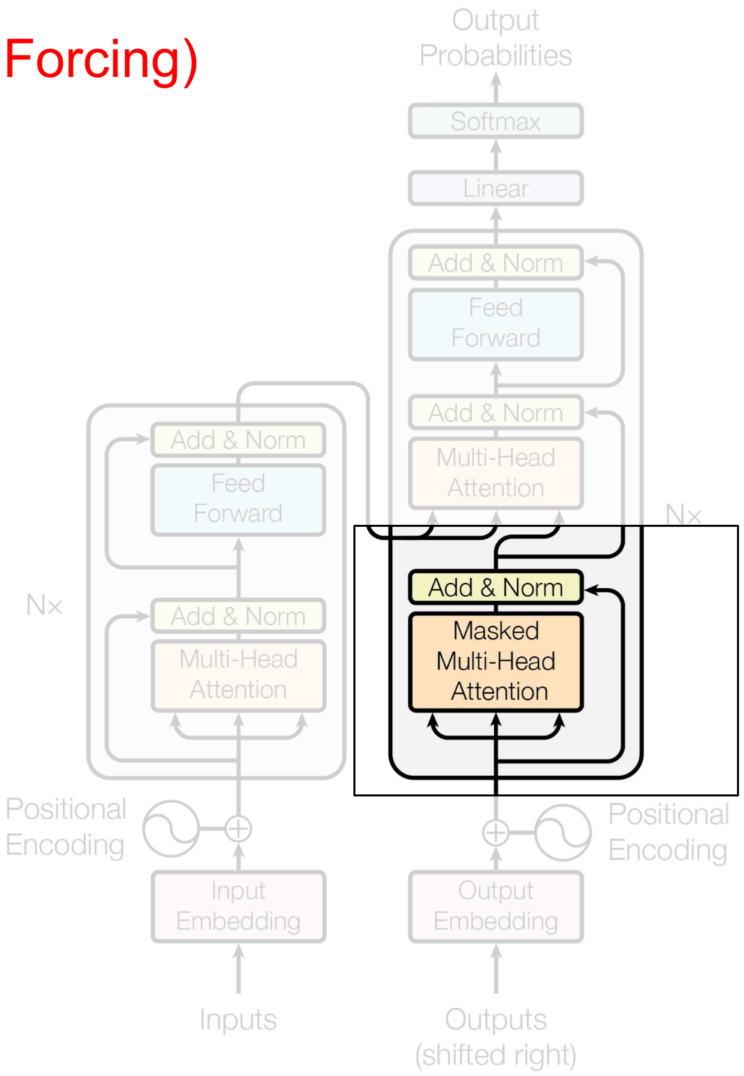
Outputs at time T should only pay attention to outputs until time T-1



Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

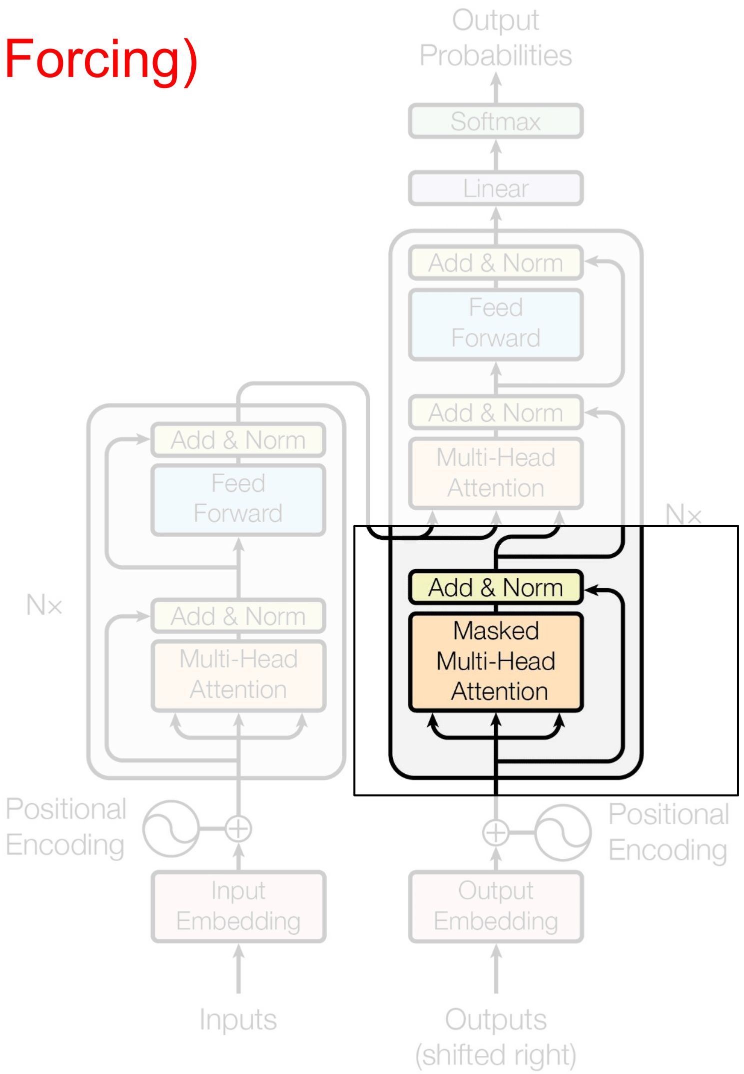
1	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
2	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
3	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
4	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
5	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
6	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
7	<sos>	Ich	habe	einen	Apfel	gegessen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



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>

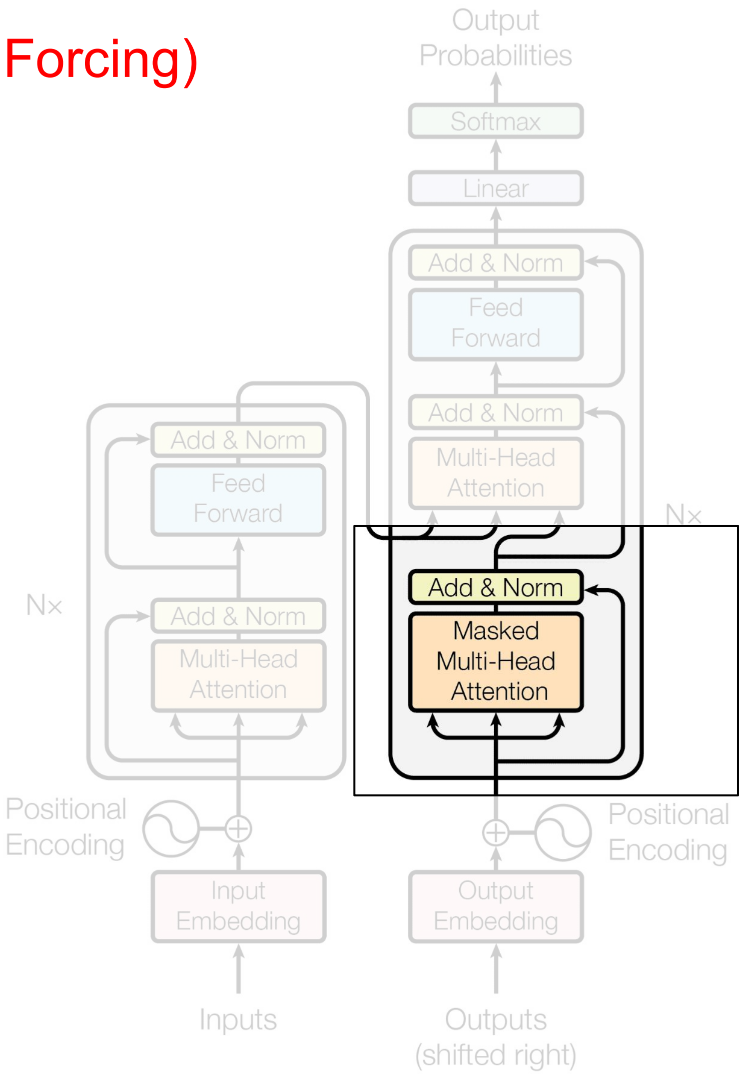


Mask the available attention values ?

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>

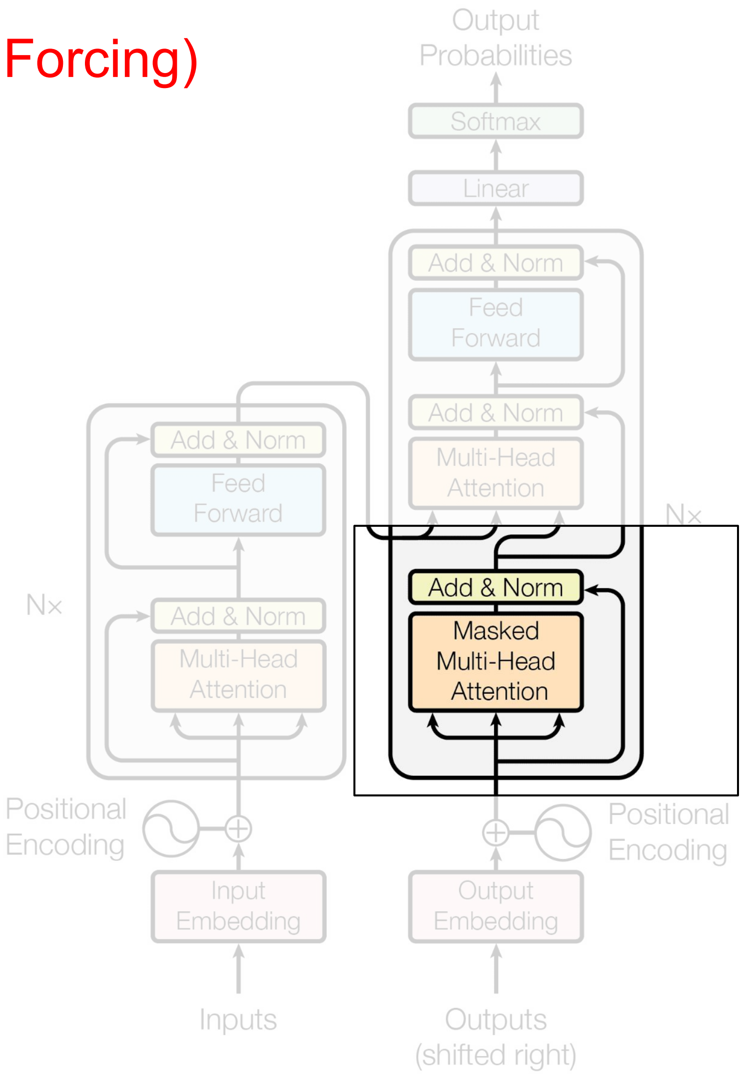


Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

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

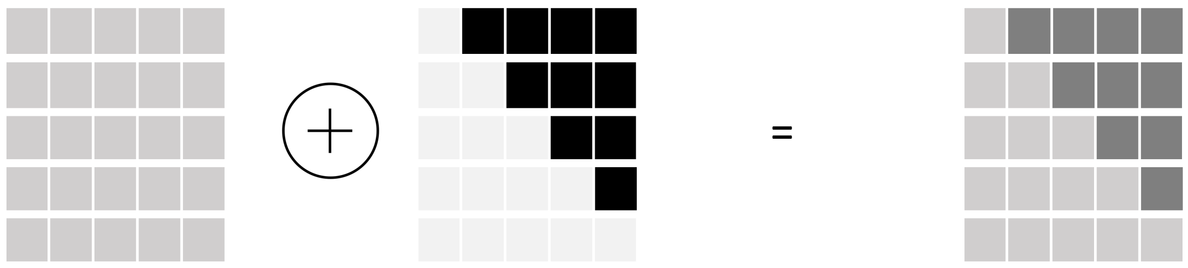
Softmax - ∞ -> 0



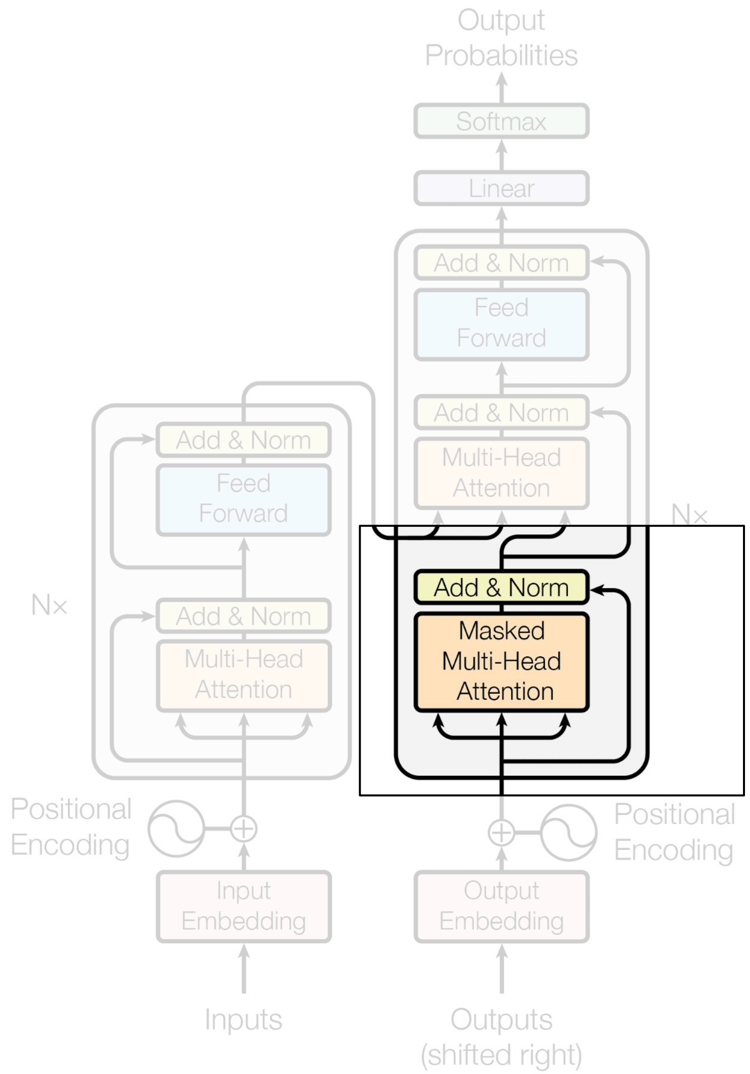
Masked Multi Head Attention

Masked Multi Head Attention

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



QK^T Attention Mask: M Masked Attention

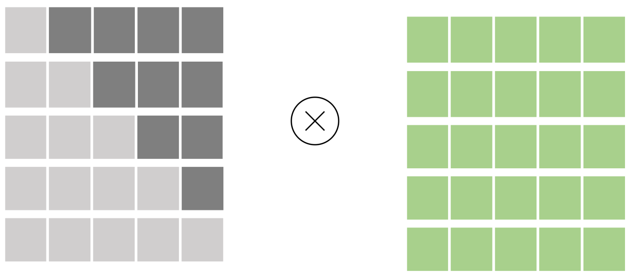


Masked Multi Head Attention : Z’

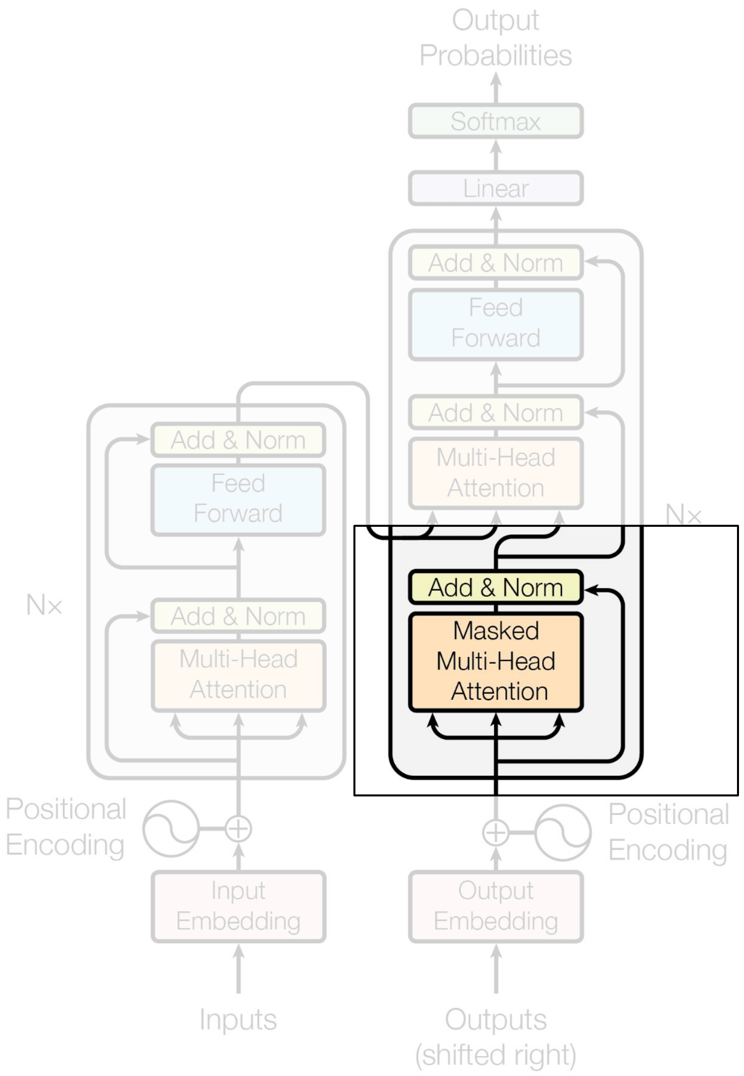
Masked Multi Head Attention

Masked Multi Head Attention

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

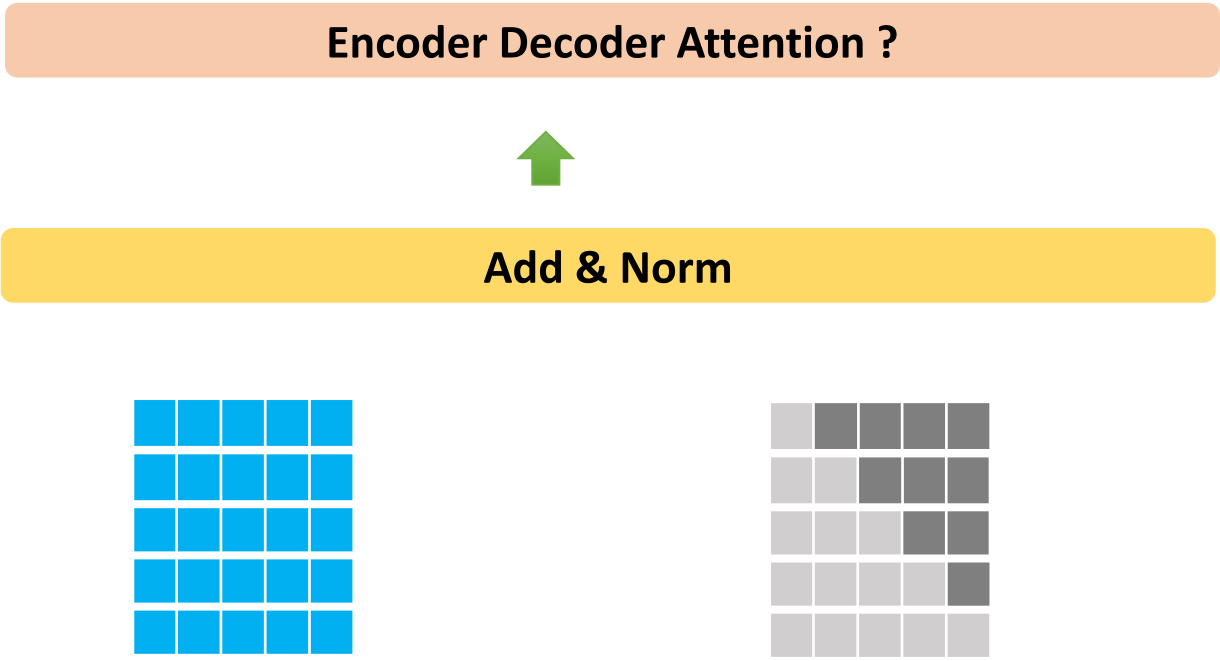


Masked Attention $\quad$ Values



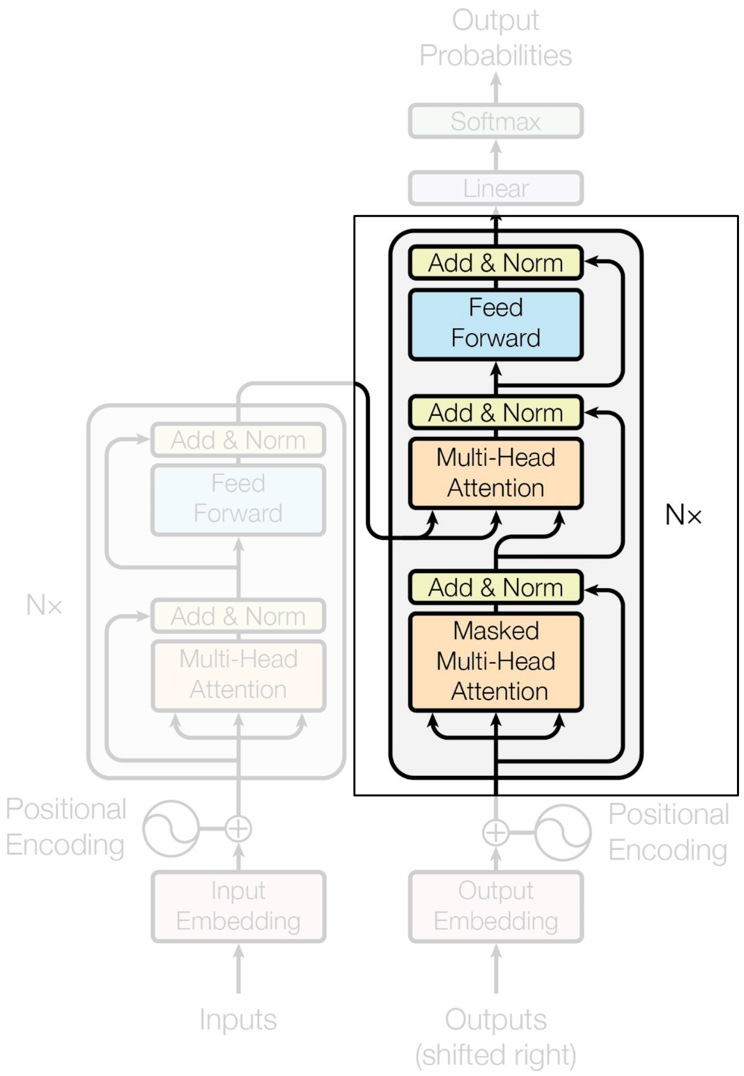
Masked Multi Head Attention : Z’

Encoder Decoder Attention



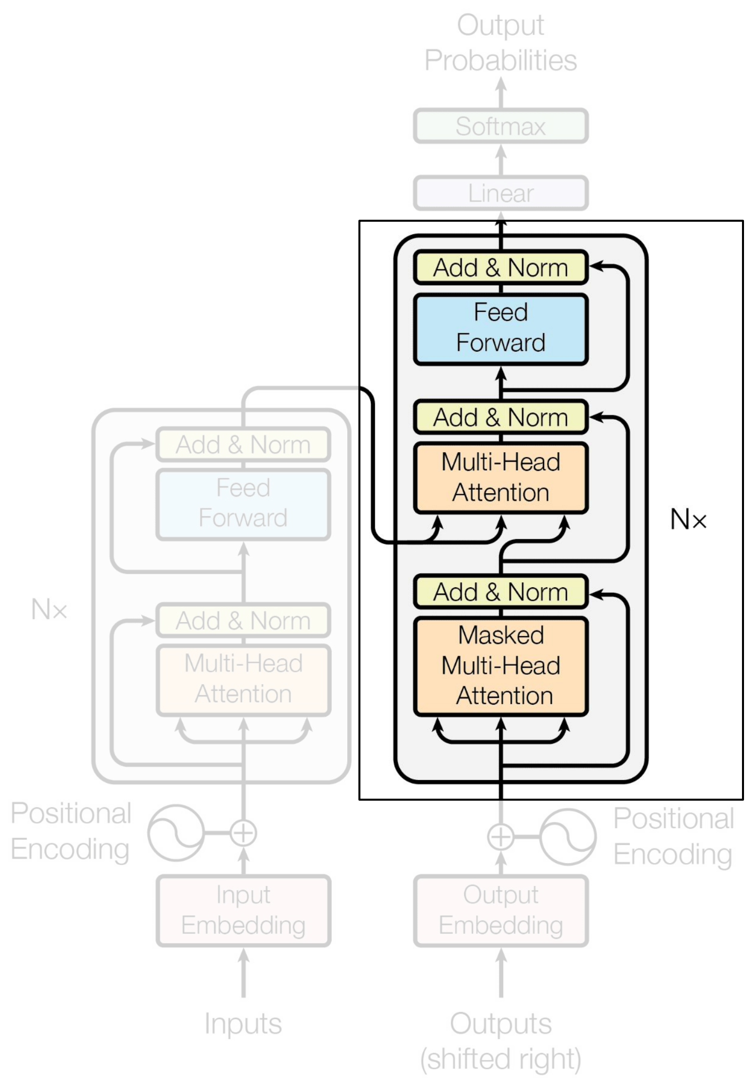
Input

Norm(Z’)



Encoder Decoder Attention

Encoder Decoder Attention ?



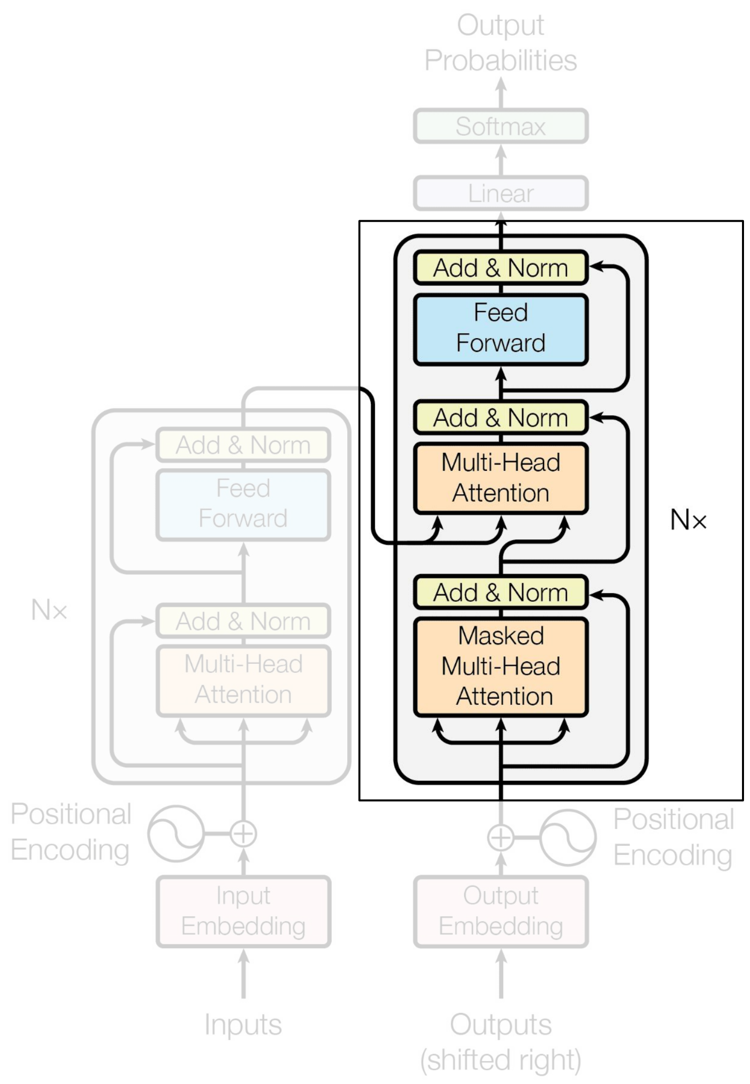
Encoder Decoder Attention

<u>Encoder **Self** Attention</u>

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

<u>Decoder **Masked Self** Attention</u>

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



Attention

{Key, Value store}

{Query: “Order details of order_104”}

{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", ...}}
```

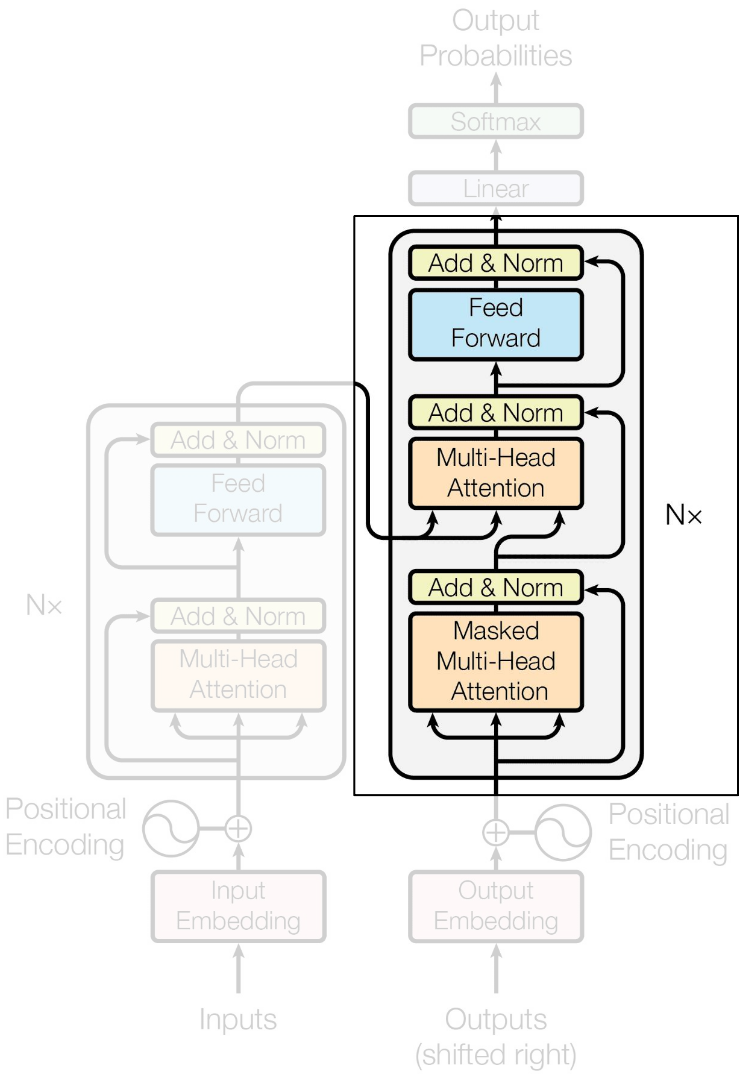
Encoder Decoder Attention

Encoder

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

Decoder

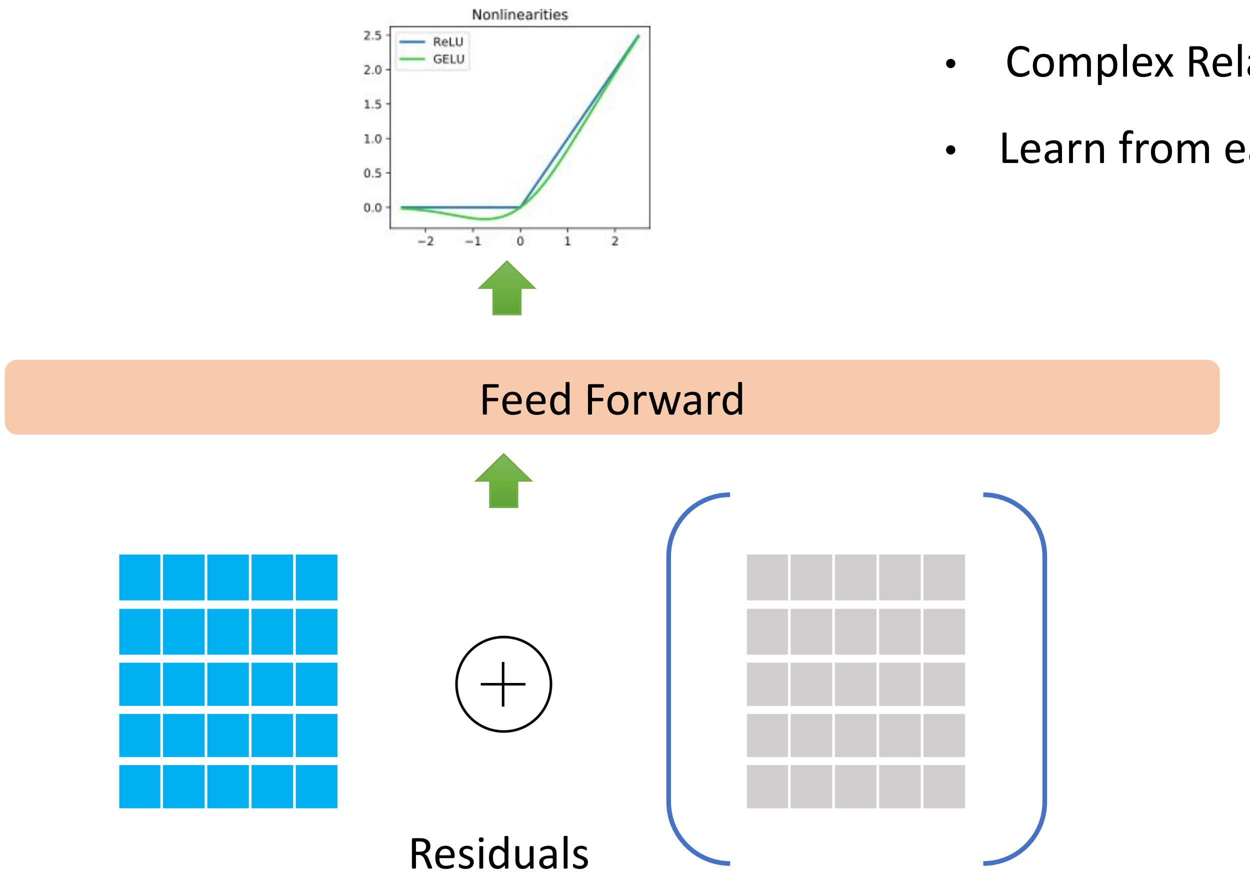
Queries from **Decoder Inputs**



NOTE: Every decoder block receives the same FINAL encoder output

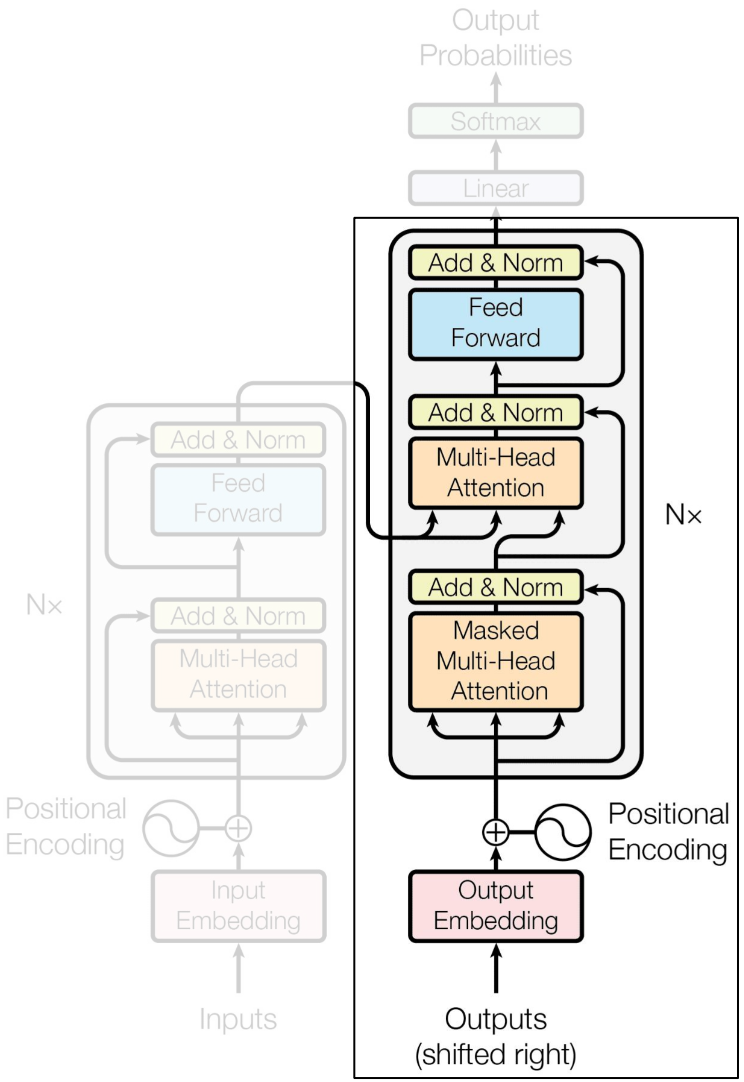
Encoder Decoder Attention

- Non Linearity
- Complex Relationships
- Learn from each other



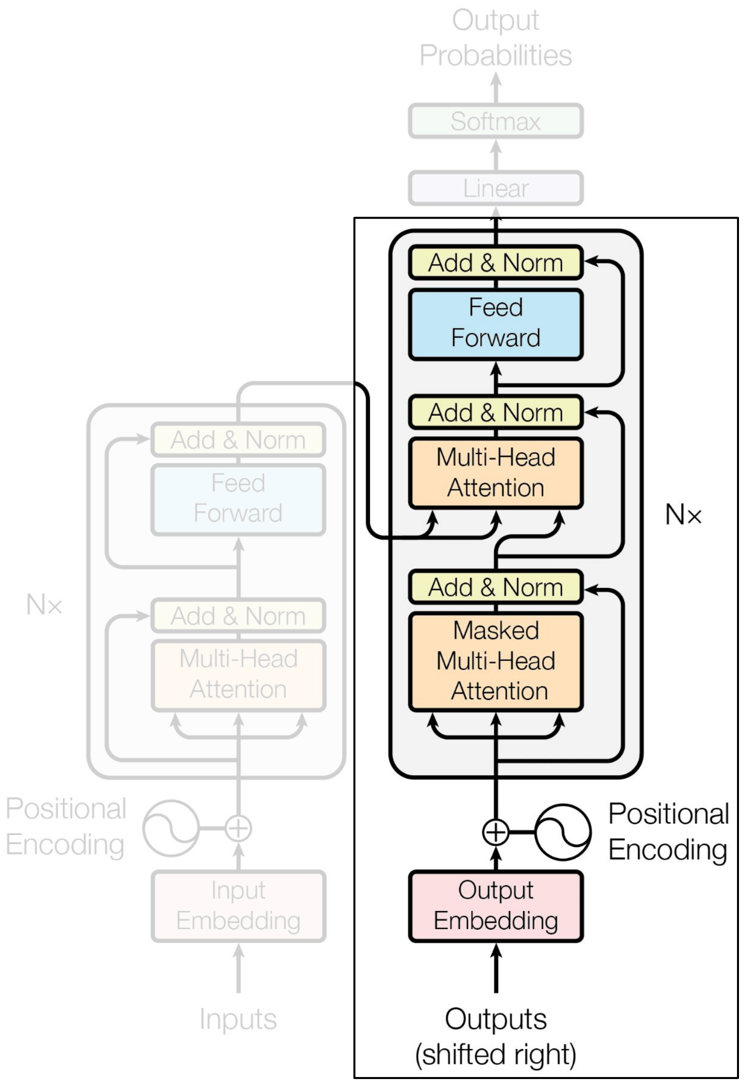
Add n Norm Decoder Self Attn

Norm(Z'')

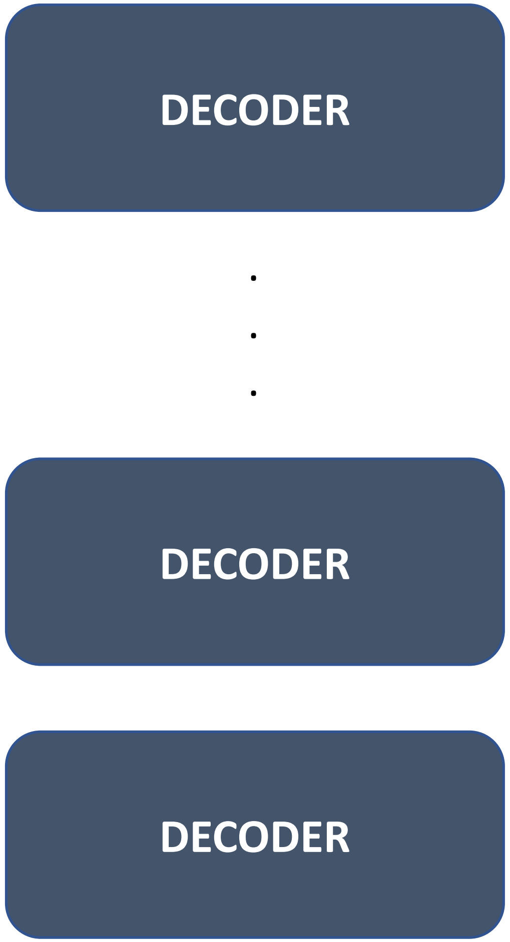


Decoder

DECODER

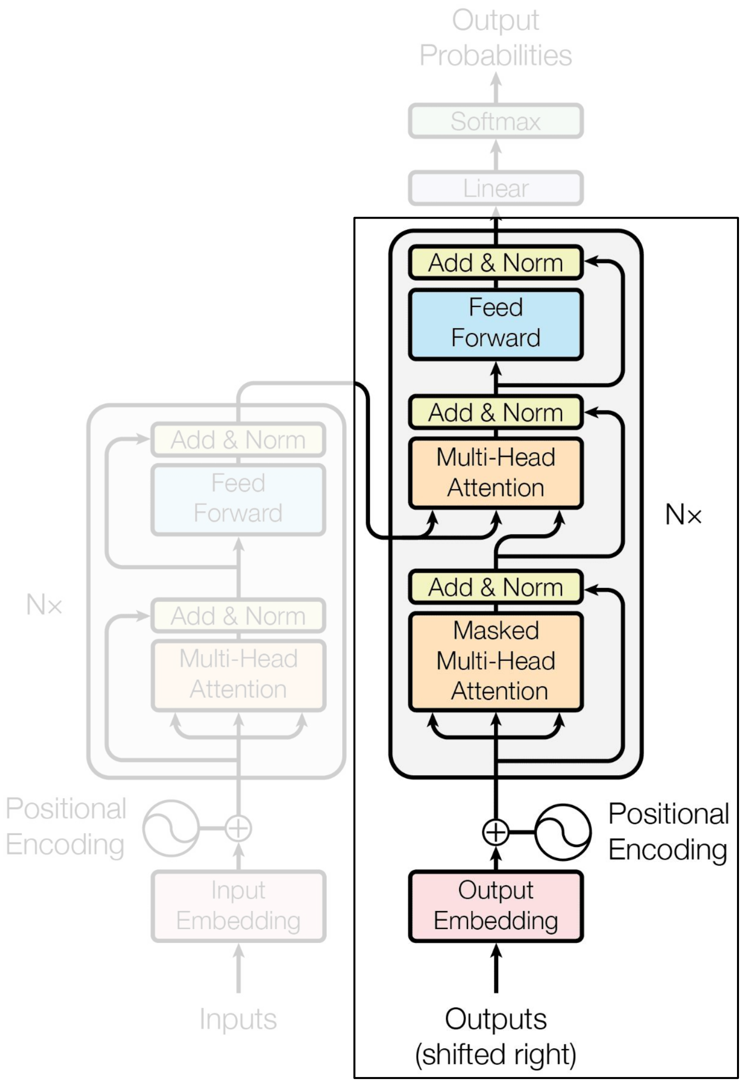


Decoder



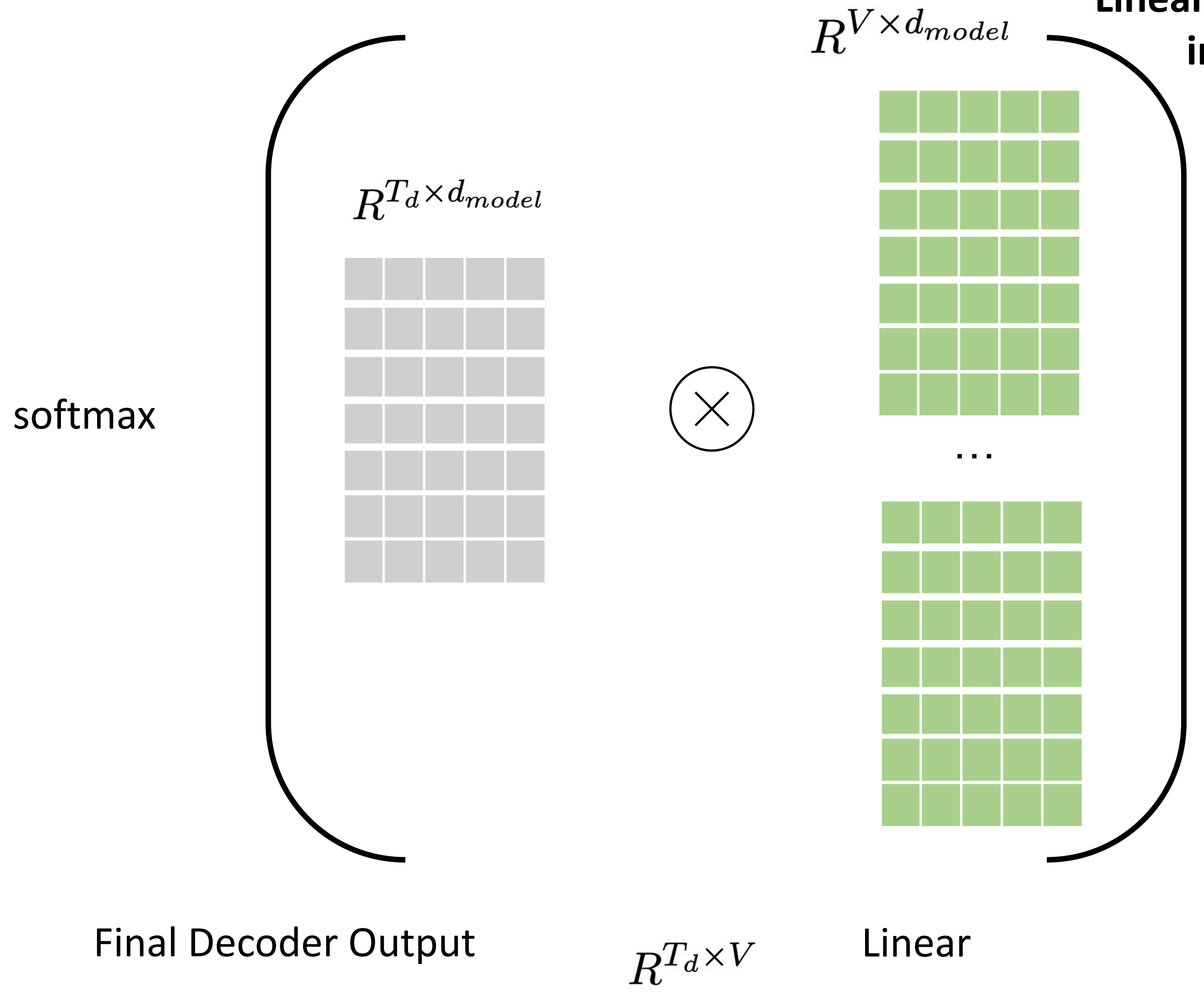
$R^{T_d \times d_{model}}$

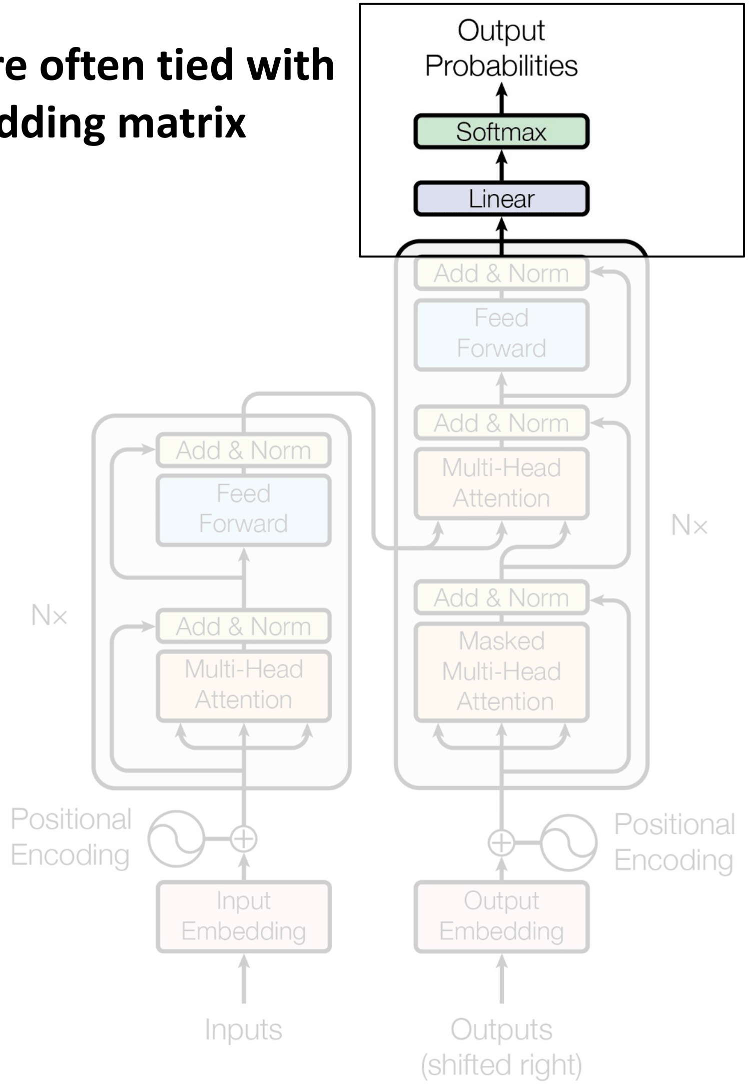
Decoder output



Linear

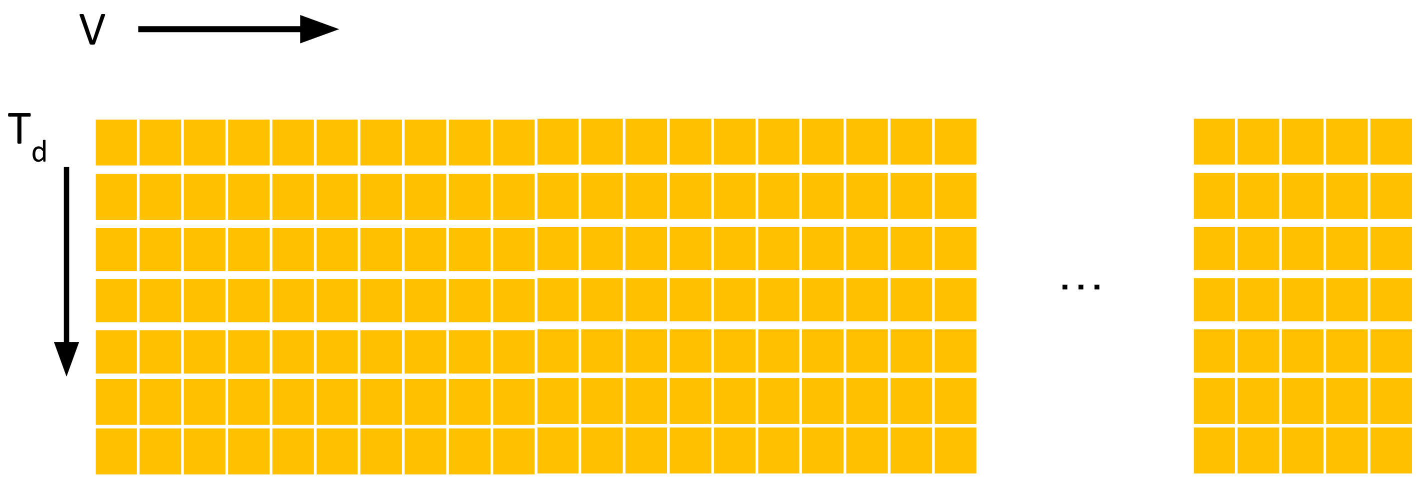
Linear weights are often tied with input embedding matrix



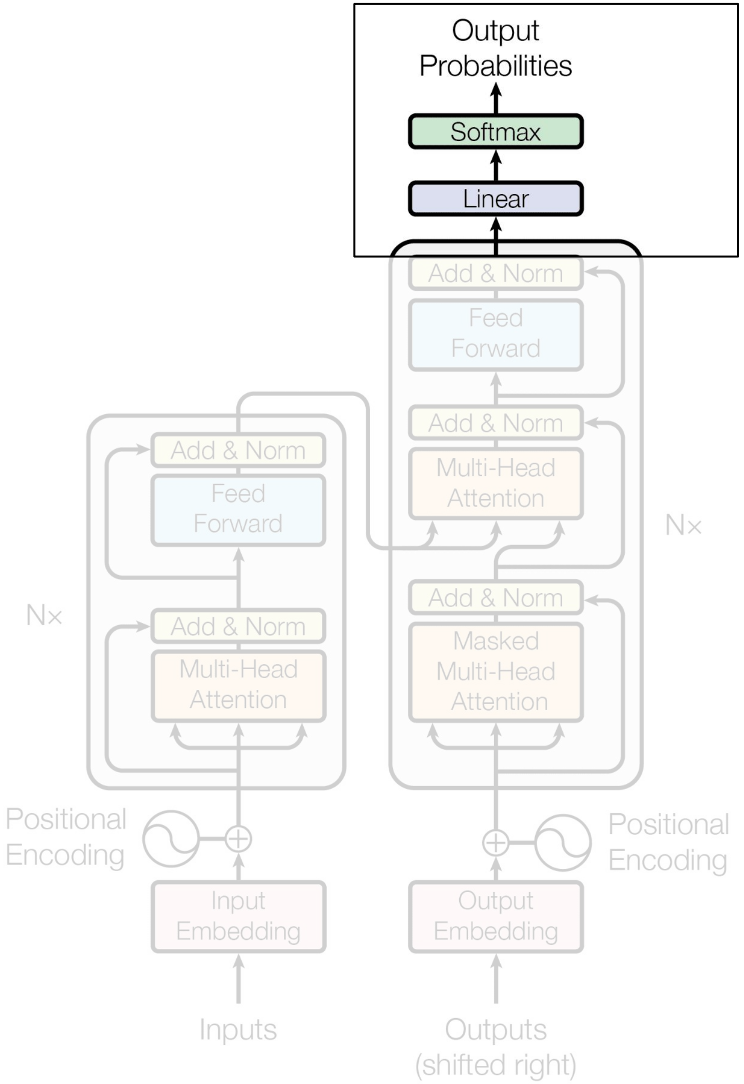


Softmax

Output Probabilities



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



Poll 2 - @1580

Which of the following are true about transformers?

a. Transformers can always be run in parallel
b. Transformer decoders can only be parallelized during training
c. Queries, keys, and values are obtained by splitting the input into 3 equal segments
d. Multihead attention might help transformers find different kinds of relations between tokens
e. Decoder outputs provide attention queries and keys, while the values come from the encoder

Poll 2 - @1580

Which of the following are true about transformers?

a. Transformers can always be run in parallel

b. Transformer decoders can only be parallelized during training

c. Queries, keys, and values are obtained by splitting the input into 3 equal segments

d. Multihead attention might help transformers find different kinds of relations between tokens

e. Decoder outputs provide attention queries and keys, while the values come from the encoder

Transformers

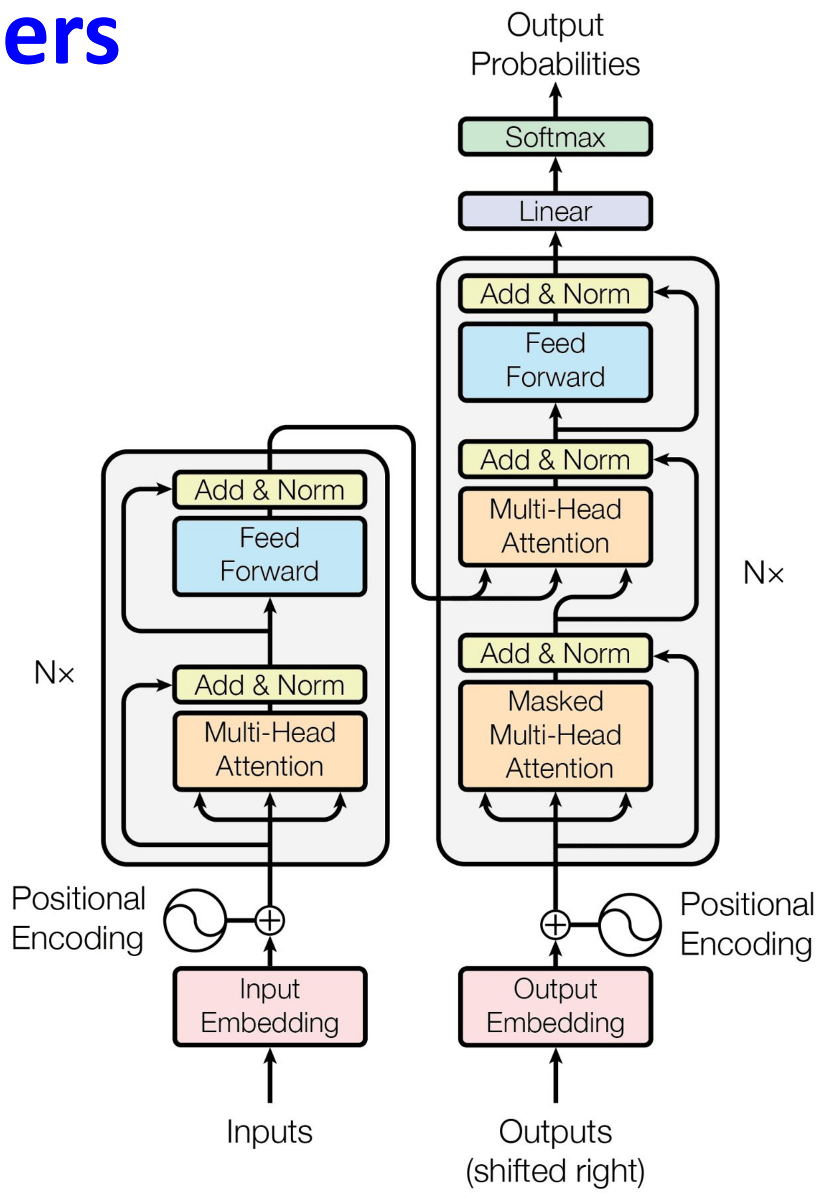
Targets

Ich habe einen Apfel gegessen

Inputs

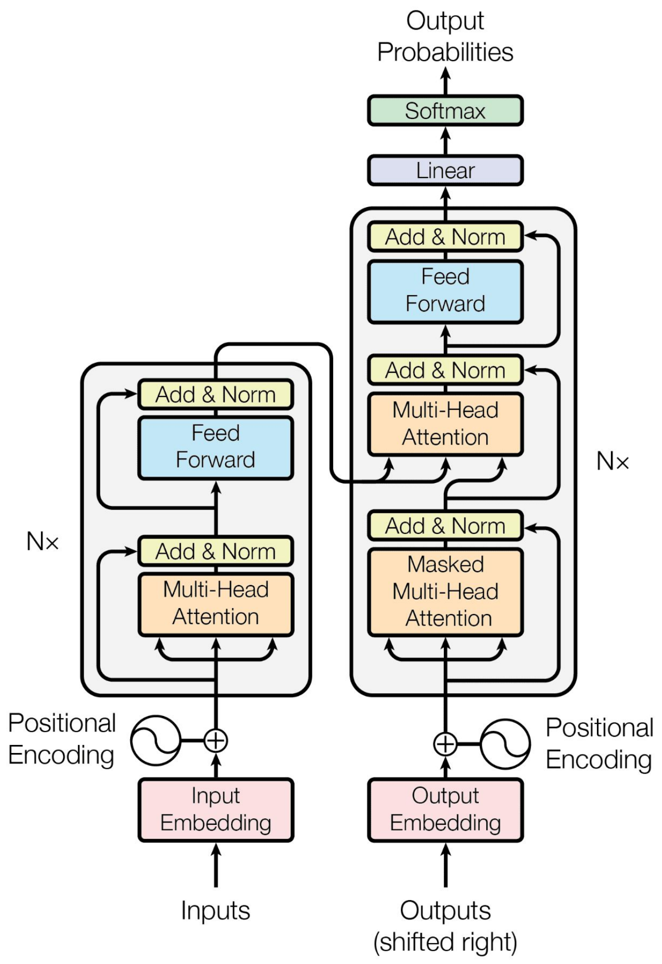
I ate an apple

Machine Translation

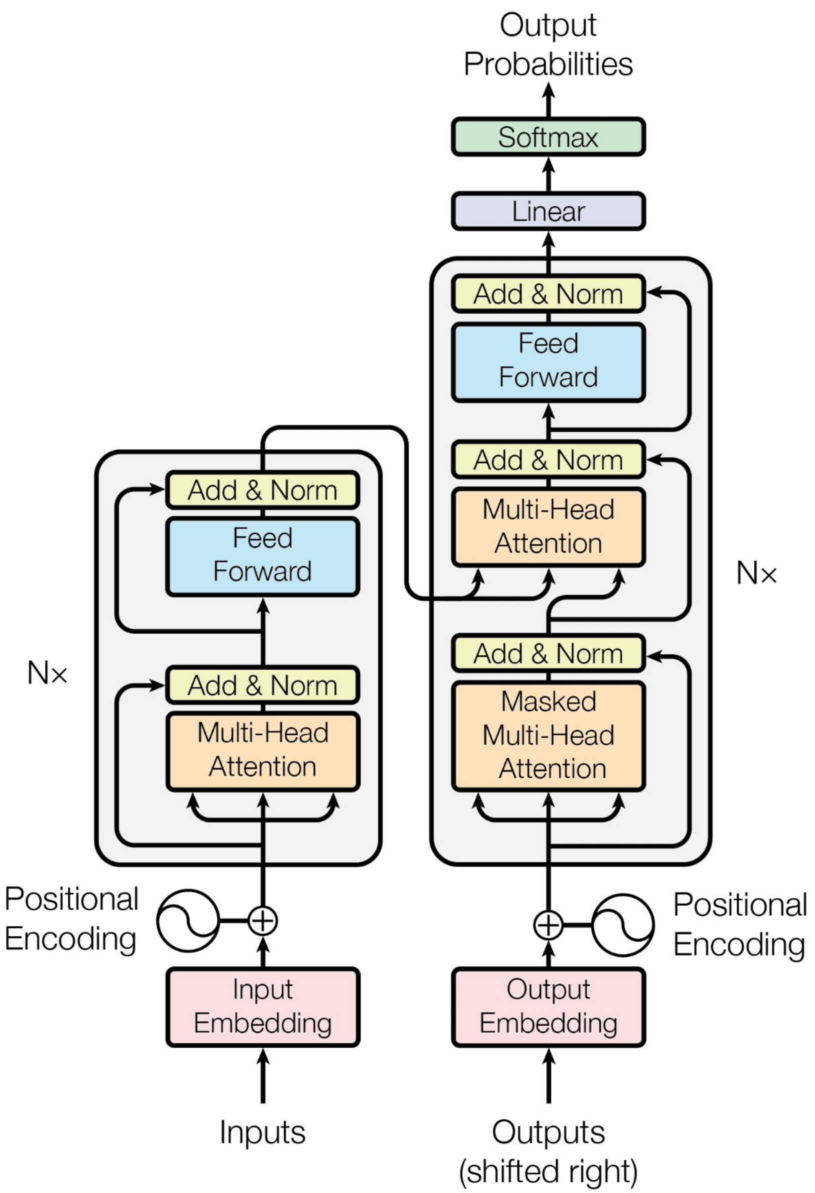


Transformers

- ✔ 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

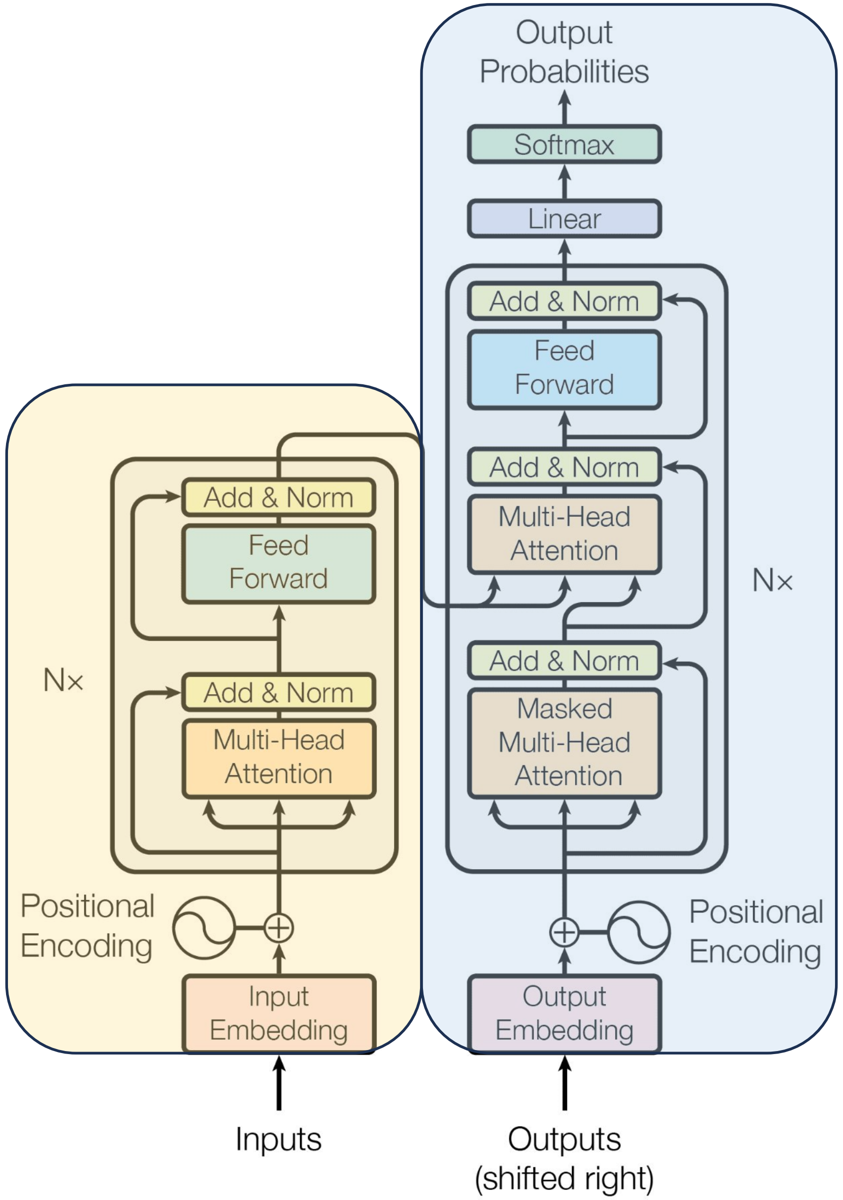


Transformers



Transformers

Representation

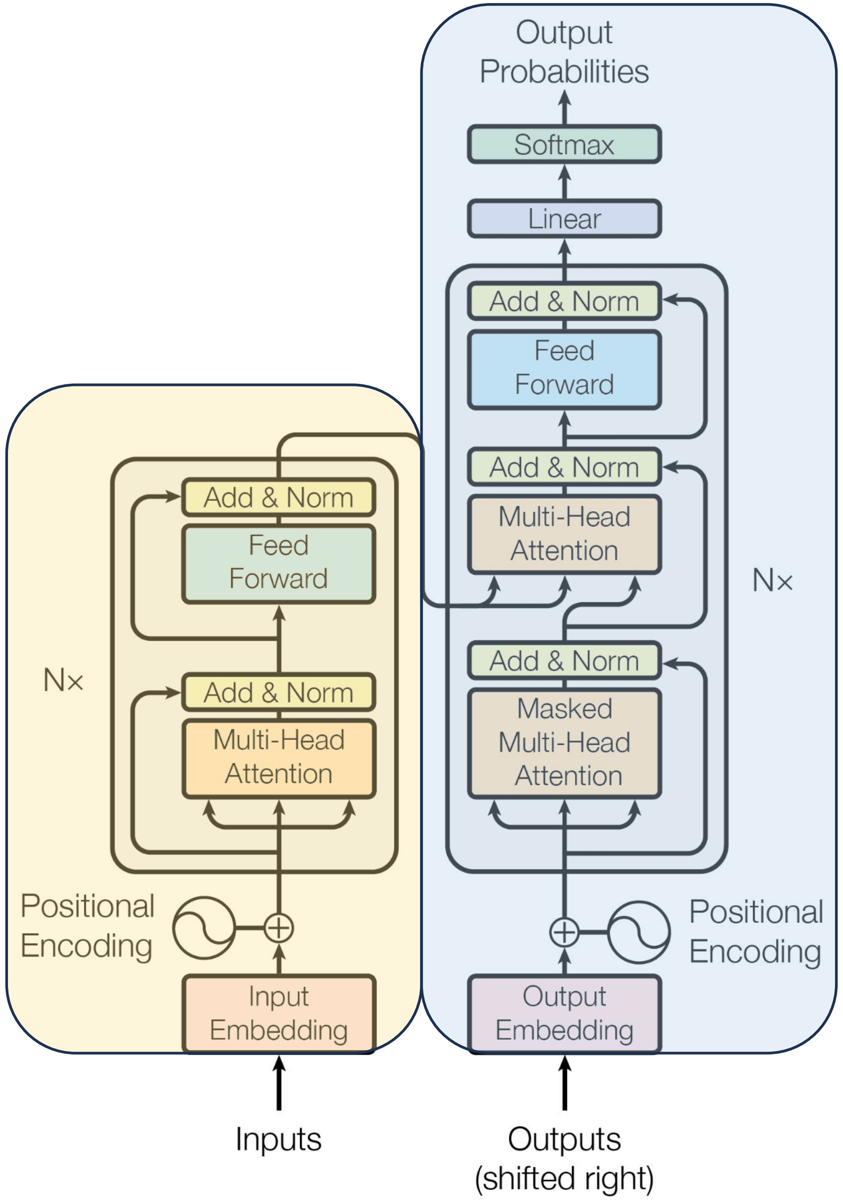


Generation

Transformers

Input – input tokens
Output – hidden states

Representation



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

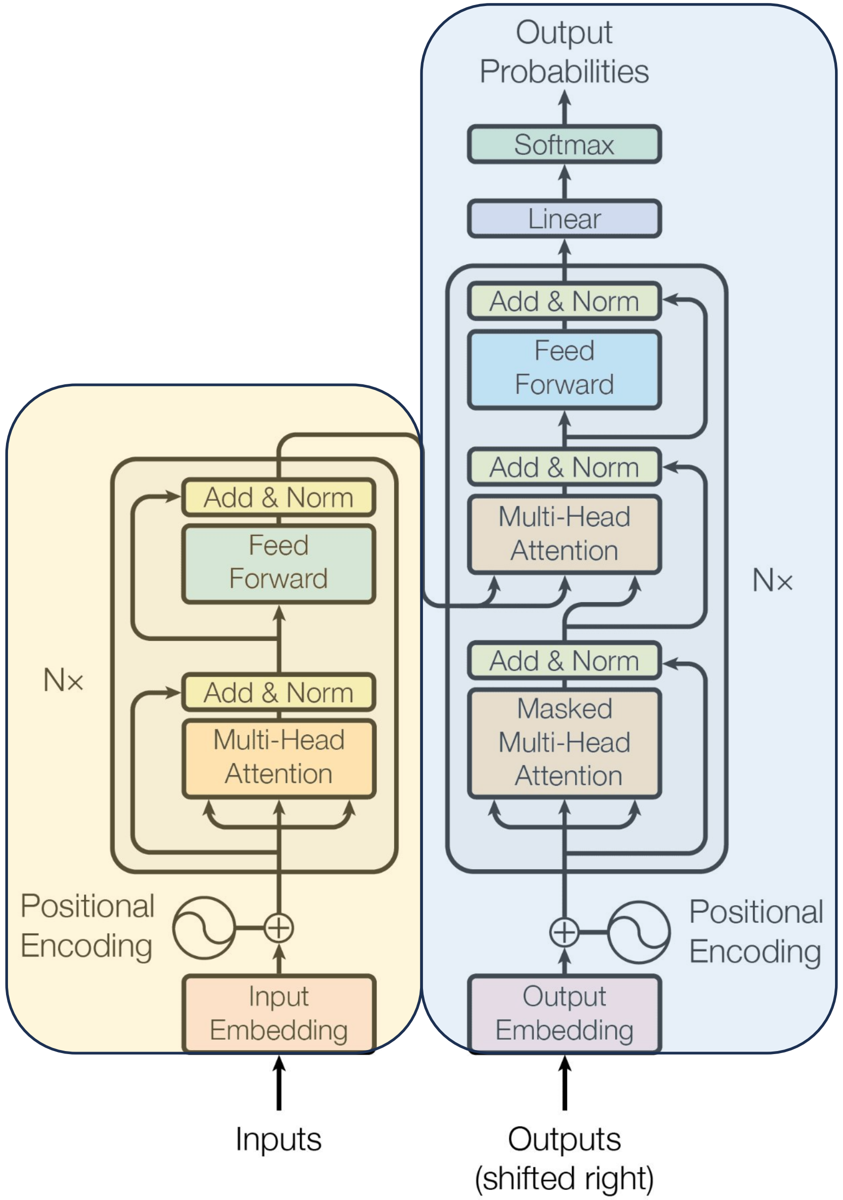
Generation

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

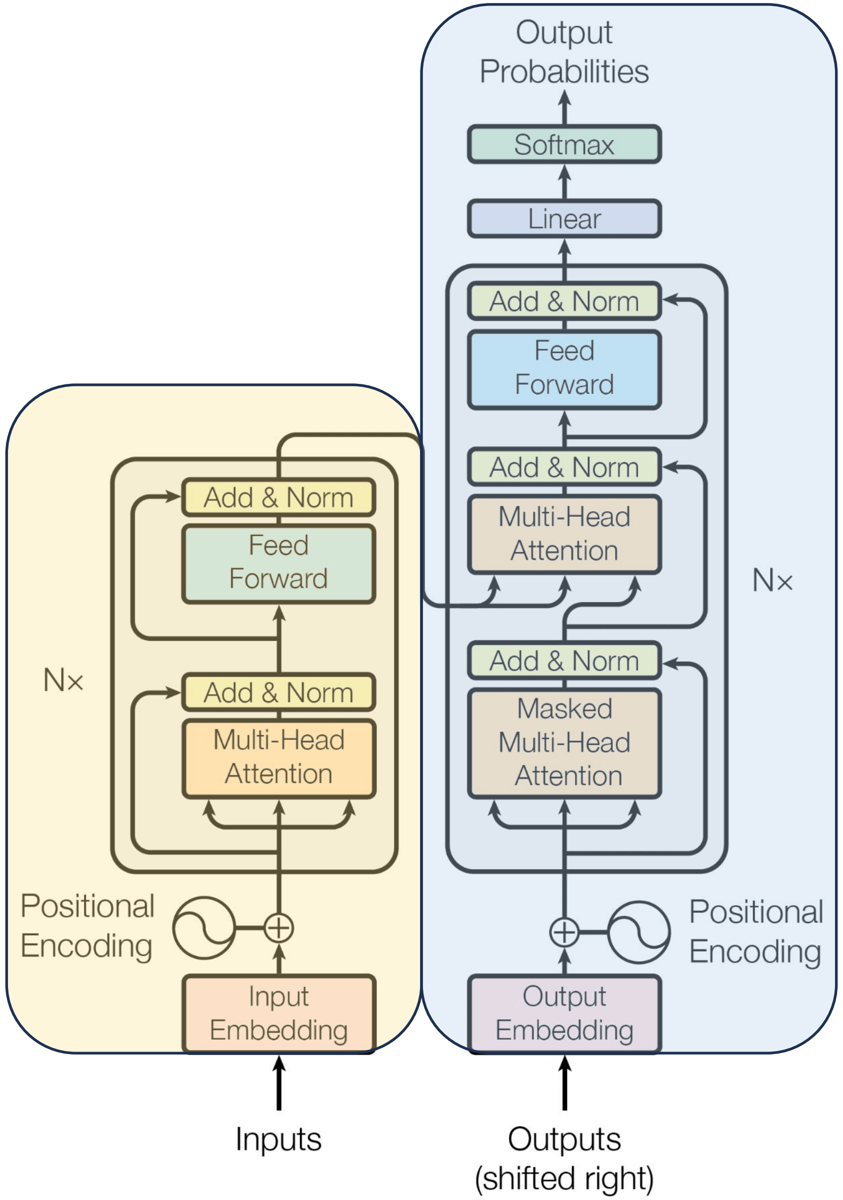
Transformers

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

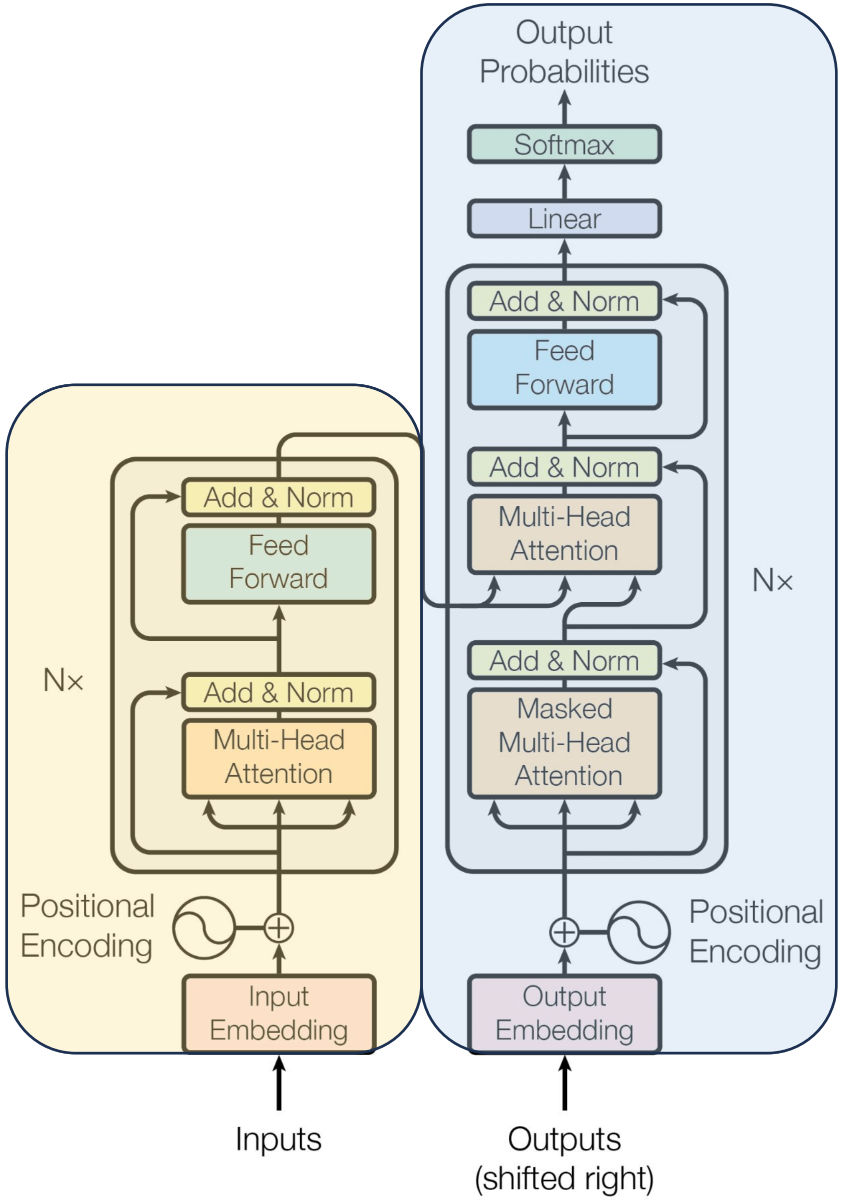
Input – input tokens
Output – hidden states

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

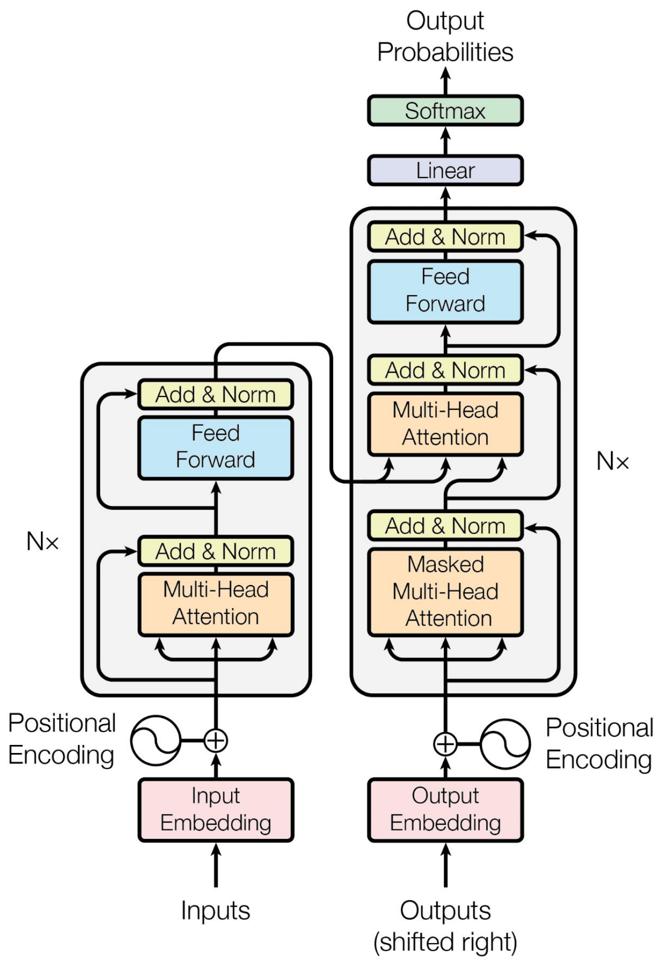
Model is auto-regressive with previous timesteps' outputs

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



Part 2

Pre-training and Fine-tuning

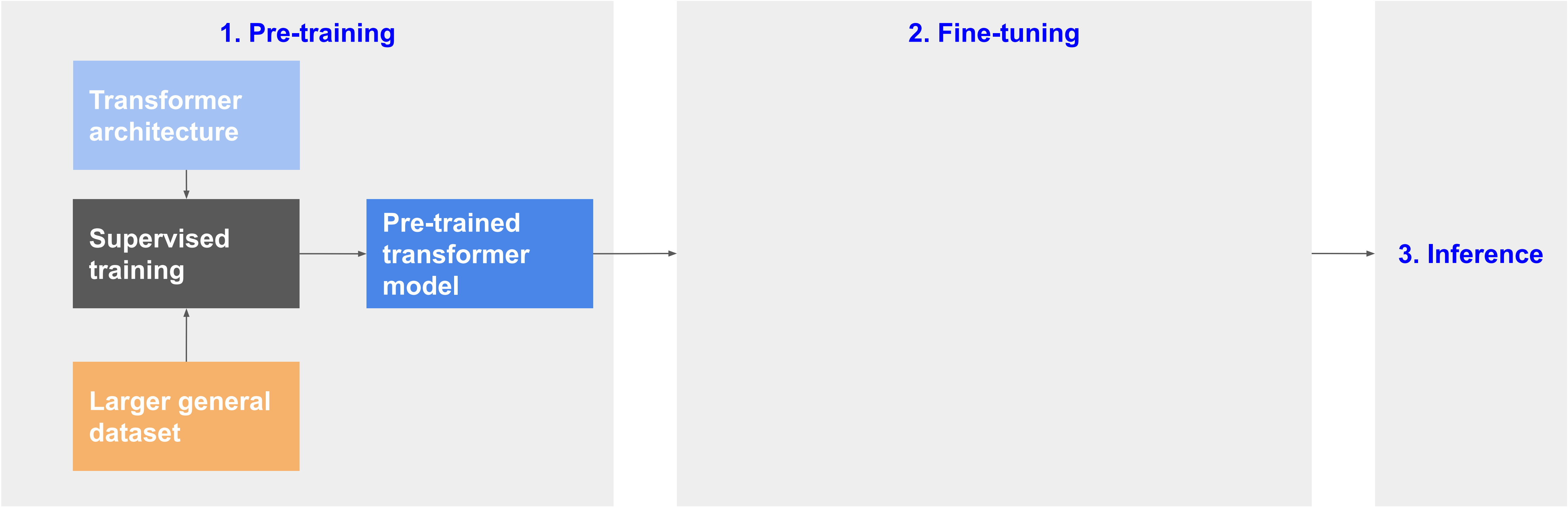
How to train and fine-tune transformers

1. Training → 2. Inference

How to train and fine-tune transformers

1. Pre-training → 2. Fine-tuning → 3. Inference

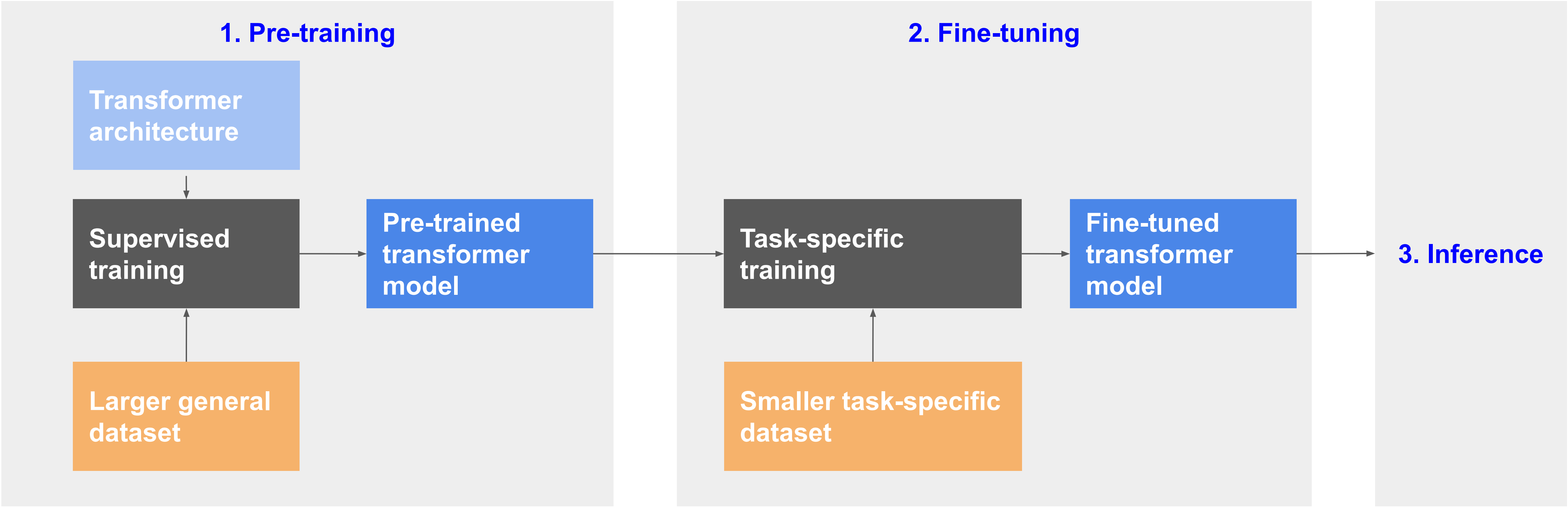
How to train and fine-tune transformers



Lot's of data, learn general things. May serve as a parameter initialization.

Usually requires significant computational resources and time.

How to train and fine-tune transformers



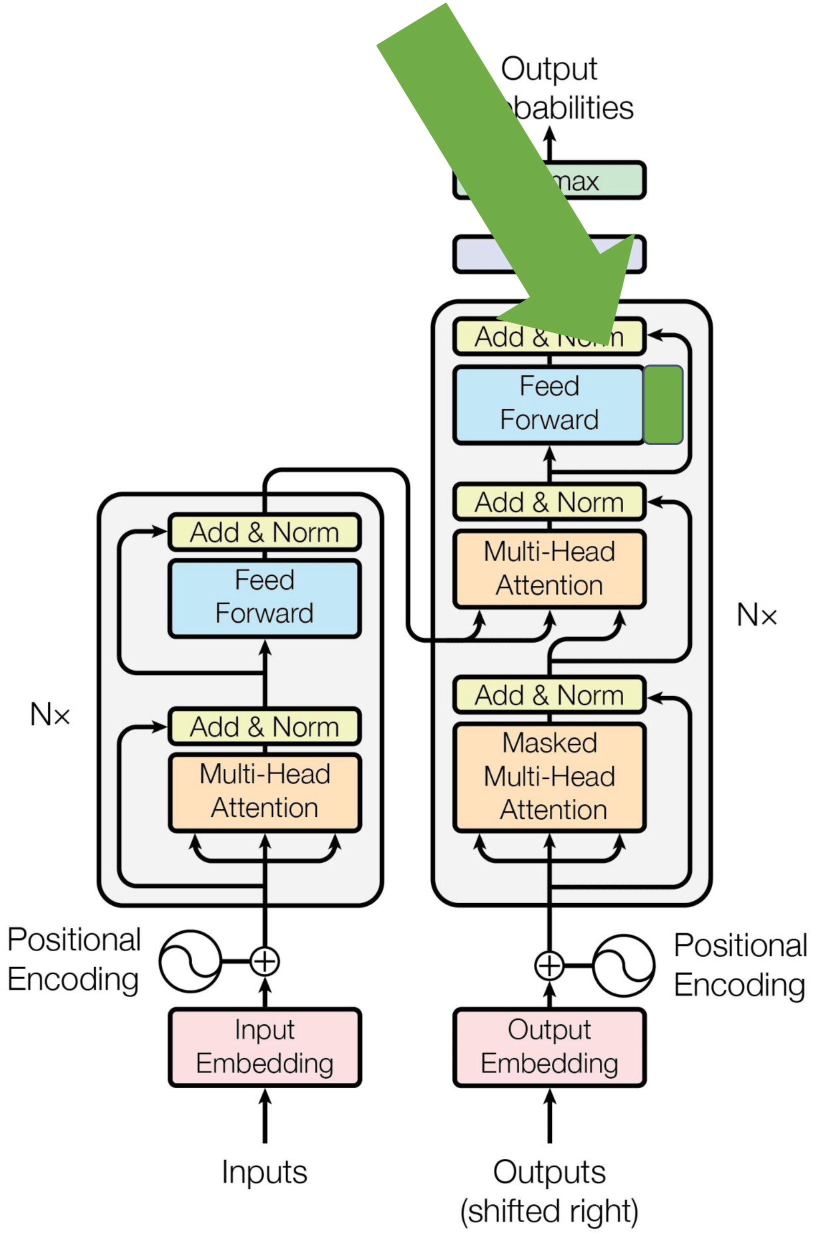
Lot’s of data, learn general things. May serve as a parameter initialization.

Usually requires significant computational resources and time.

Adaptation to the specific task.

Potentially less computationally intensive.

Parameter-Efficient Fine-Tuning Techniques



LoRA: https://arxiv.org/abs/2106.09685
BitFit: https://arxiv.org/abs/2106.10199

Parameter-Efficient Fine-Tuning Techniques

LoRA (Lower-Rank Adaptation)

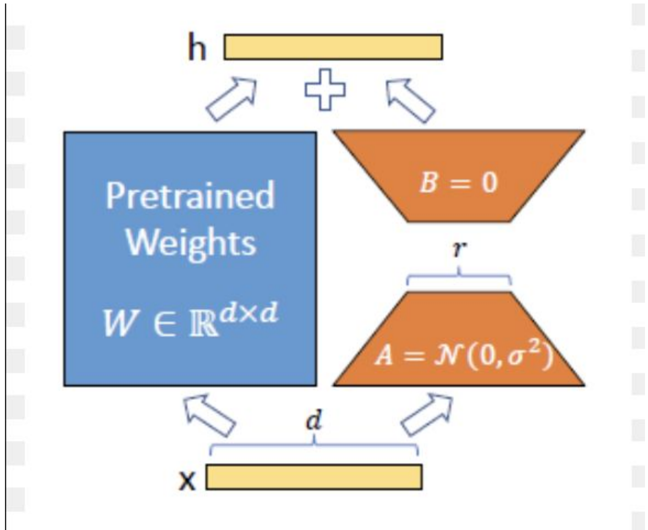
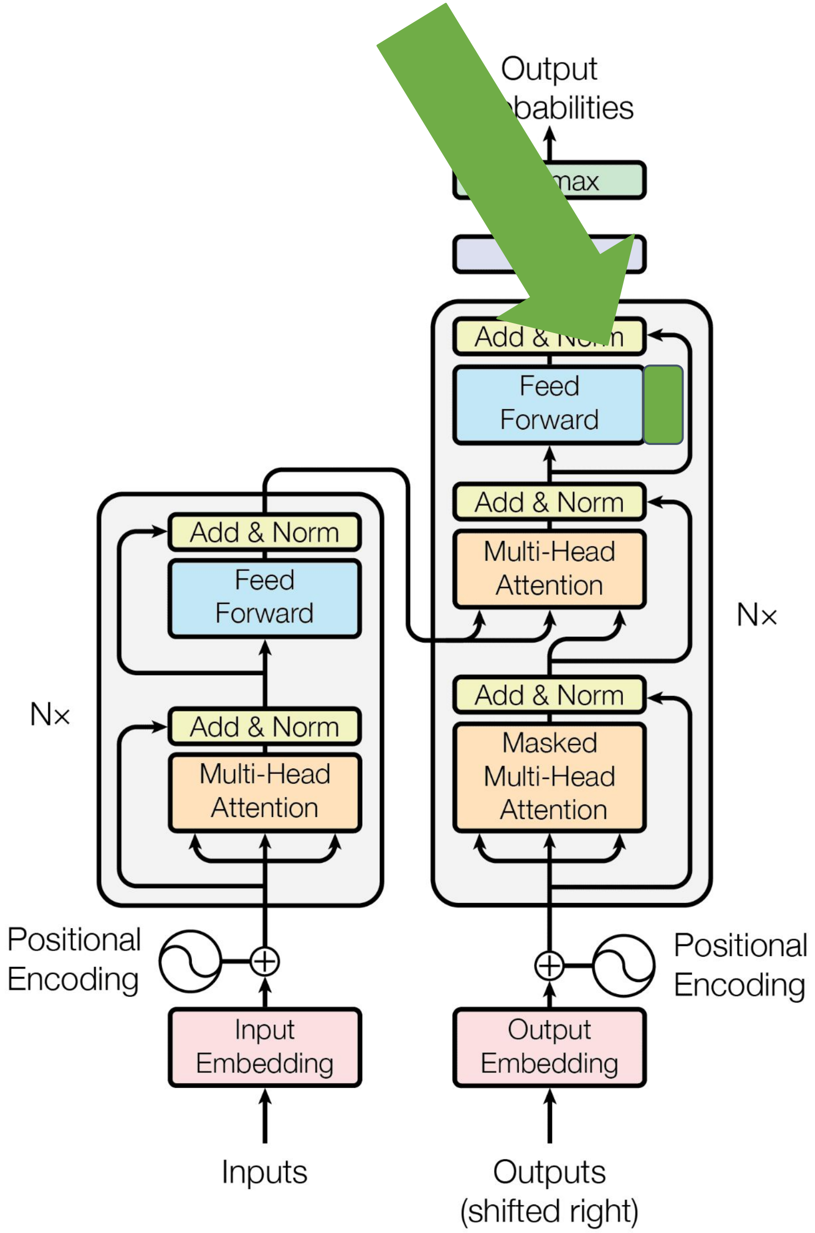


Figure 1: Our reparametrization. We only train A and B.



LoRA: https://arxiv.org/abs/2106.09685
BitFit: https://arxiv.org/abs/2106.10199

Parameter-Efficient Fine-Tuning Techniques

LoRA (Lower-Rank Adaptation)

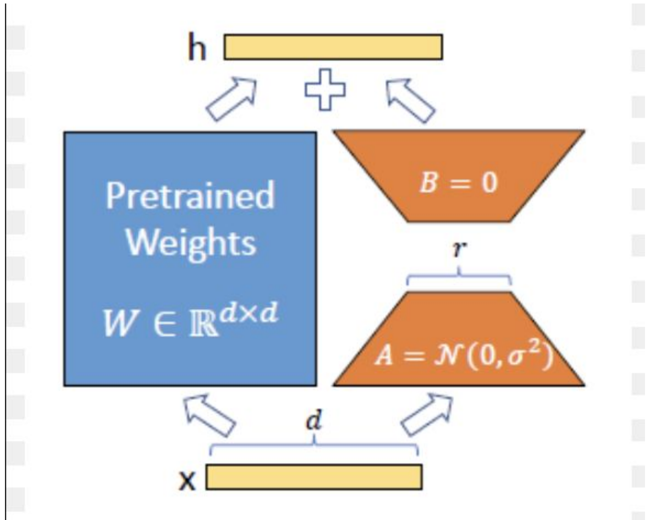
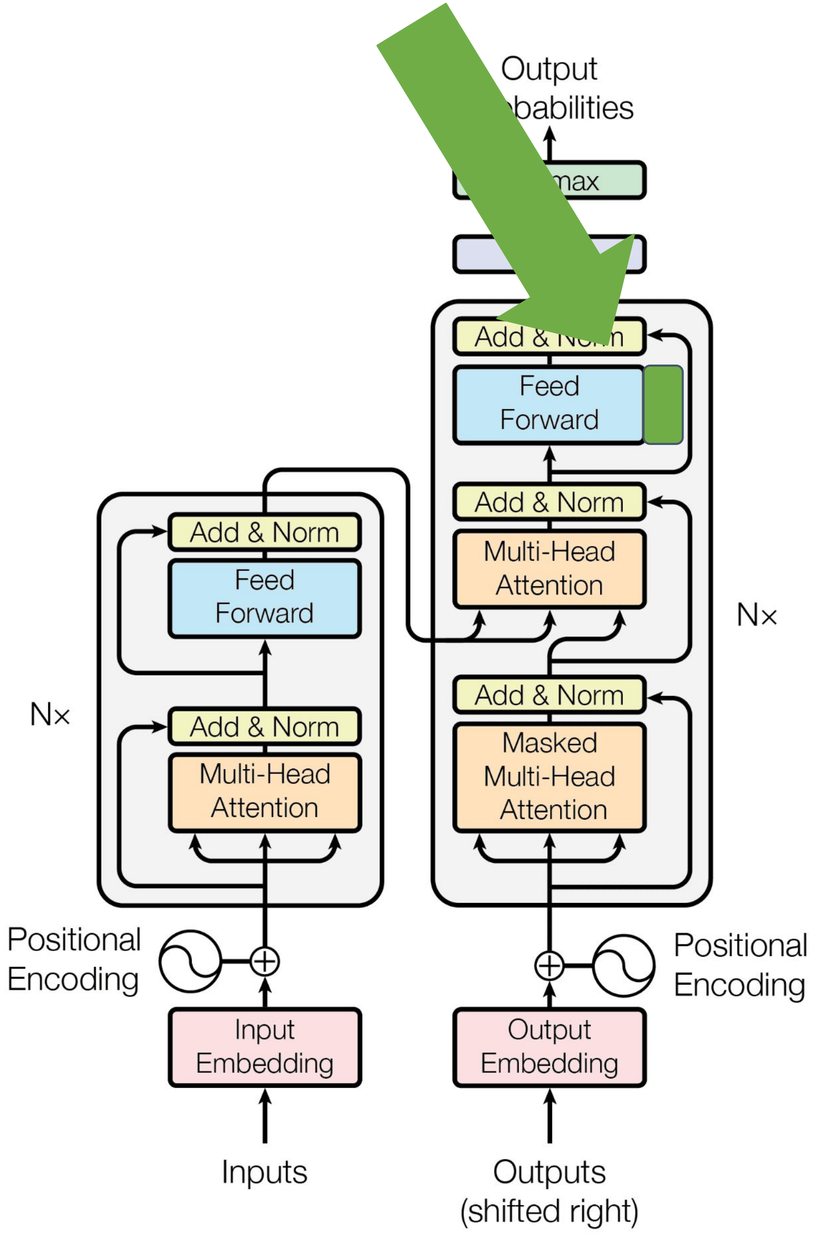


Figure 1: Our reparametrization. We only train A and B.

BitFit

$$\mathbf{Q}^{m,\ell}(\mathbf{x}) = \mathbf{W}_q^{m,\ell}\mathbf{x} + \mathbf{b}_q^{m,\ell}$$
$$\mathbf{K}^{m,\ell}(\mathbf{x}) = \mathbf{W}_k^{m,\ell}\mathbf{x} + \mathbf{b}_k^{m,\ell}$$
$$\mathbf{V}^{m,\ell}(\mathbf{x}) = \mathbf{W}_v^{m,\ell}\mathbf{x} + \mathbf{b}_v^{m,\ell}$$



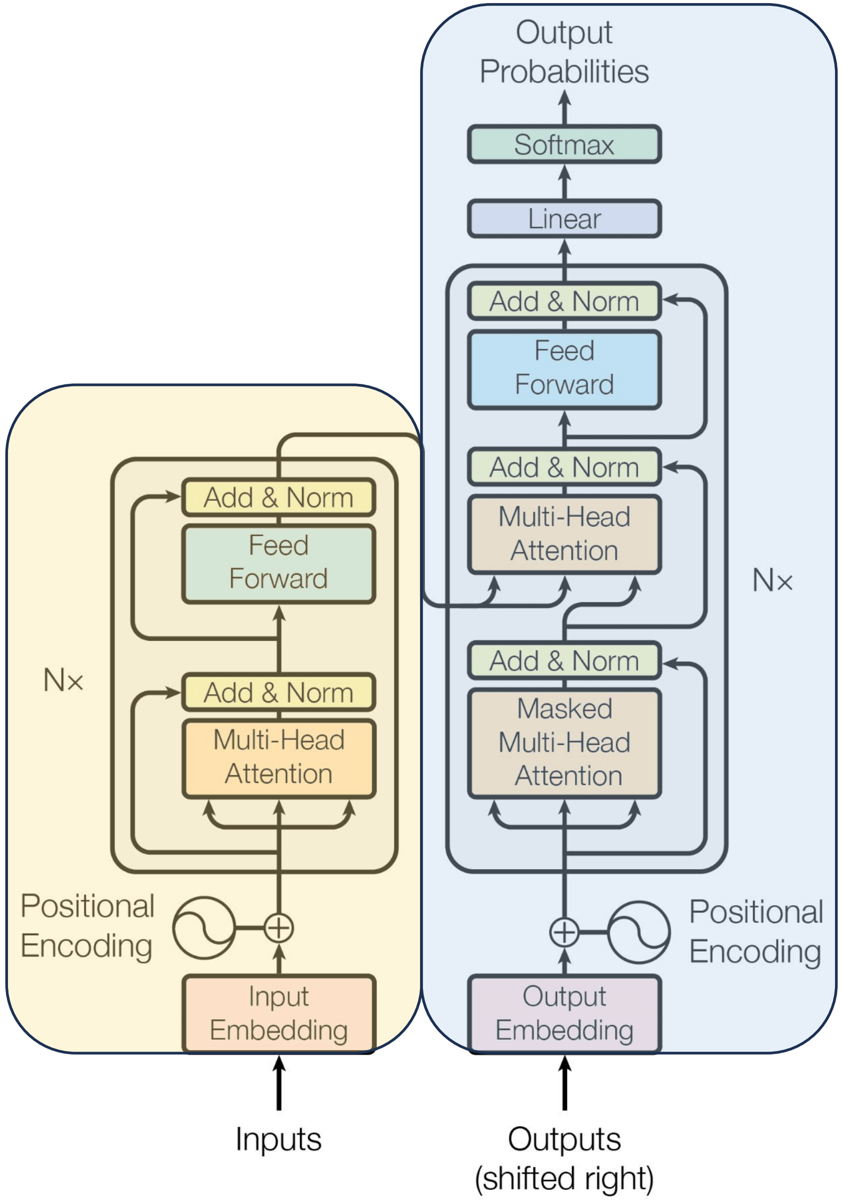
LoRA: https://arxiv.org/abs/2106.09685
BitFit: https://arxiv.org/abs/2106.10199

Part 3

Transformer Applications

Transformers

Representation / Encoder



Generation / Decoder

Data Modalities

- Language (see Part 4 of the lecture)
- Vision
- Audio
- … and many other modalities (e.g., biological/physiological signals, etc.)
- Multimodal (>2 data modalities)

Computer Vision

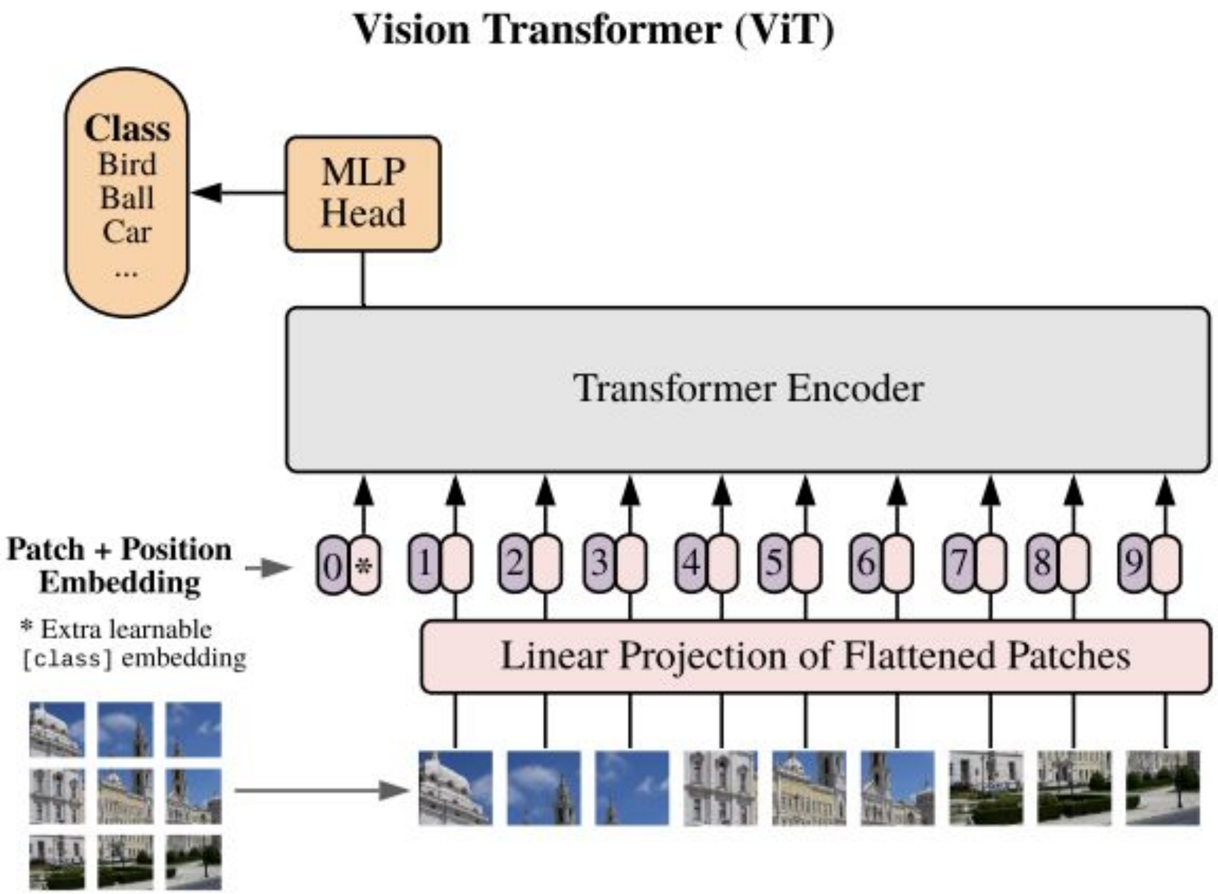
1. In computer vision convolutional architectures remain largely dominant.
2. Inspired by NLP successes, multiple works try introducing combining CNN-like architectures with self-attention or replacing the convolutions entirely.
3. However, they faced challenges with performance and scaling.
4. Key breakthrough - Vision Transformer (ViT) released in 2020

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

Computer Vision - Tokenization

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

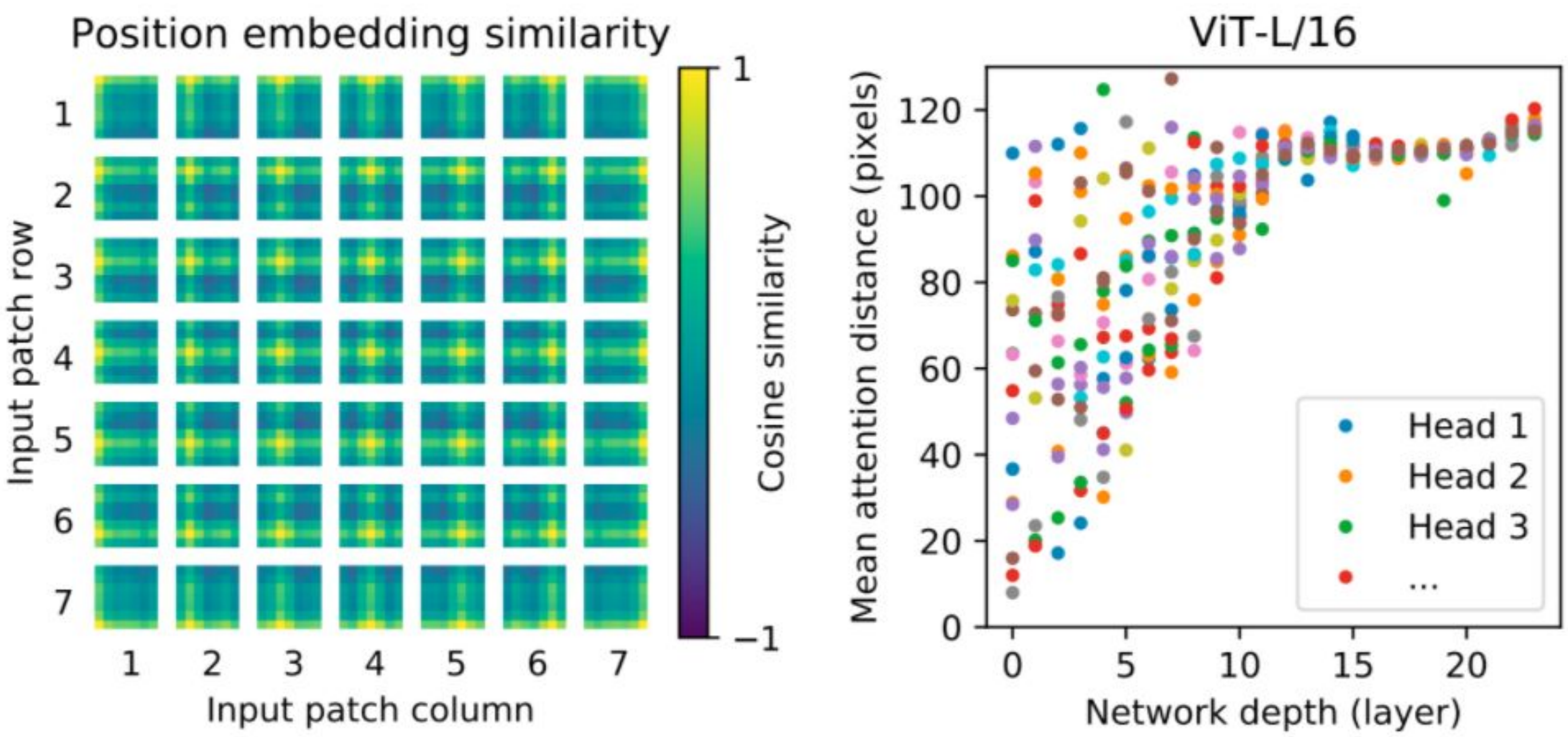
Vision Transformer (ViT) Model Architecture



1. Split an image into fixed-size patches (16x16 pixels).
2. Tokenize each path (linear projection of flattened patches).
3. Add position embedding.
4. Feed the resulting sequence of vectors to a standard Transformer encoder.
5. For classification, add an extra learnable“classification token” to the sequence.

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

ViT - Learning Patterns

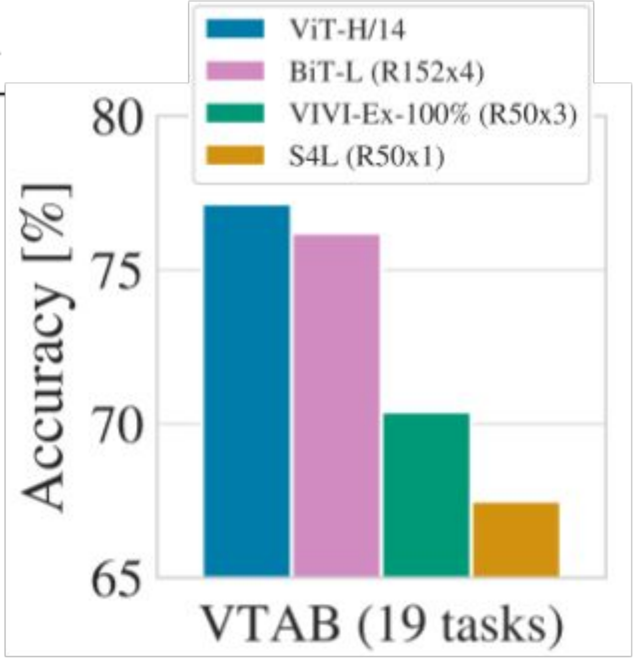


- ViT learns the grid like structure of the image patches via its position embeddings.
- The lower layers contain both global and local features, the higher layers contain only global features.

[Transformers for Image Recognition at Scale](#)

ViT Performance

	ViT-H	Previous SOTA
ImageNet	88.55	88.5
ImageNet-ReaL	90.72	90.55
Cifar-10	99.50	99.37
Cifar-100	94.55	93.51
Pets	97.56	96.62
Flowers	99.68	99.63



- ViT model attains state-of-the-art performance on multiple popular benchmarks, including 88.55% top-1 accuracy on ImageNet and 99.50% on CIFAR-10

[Transformers for Image Recognition at Scale](#)

Audio

- Similar to the computer vision but with spectrograms instead of images.

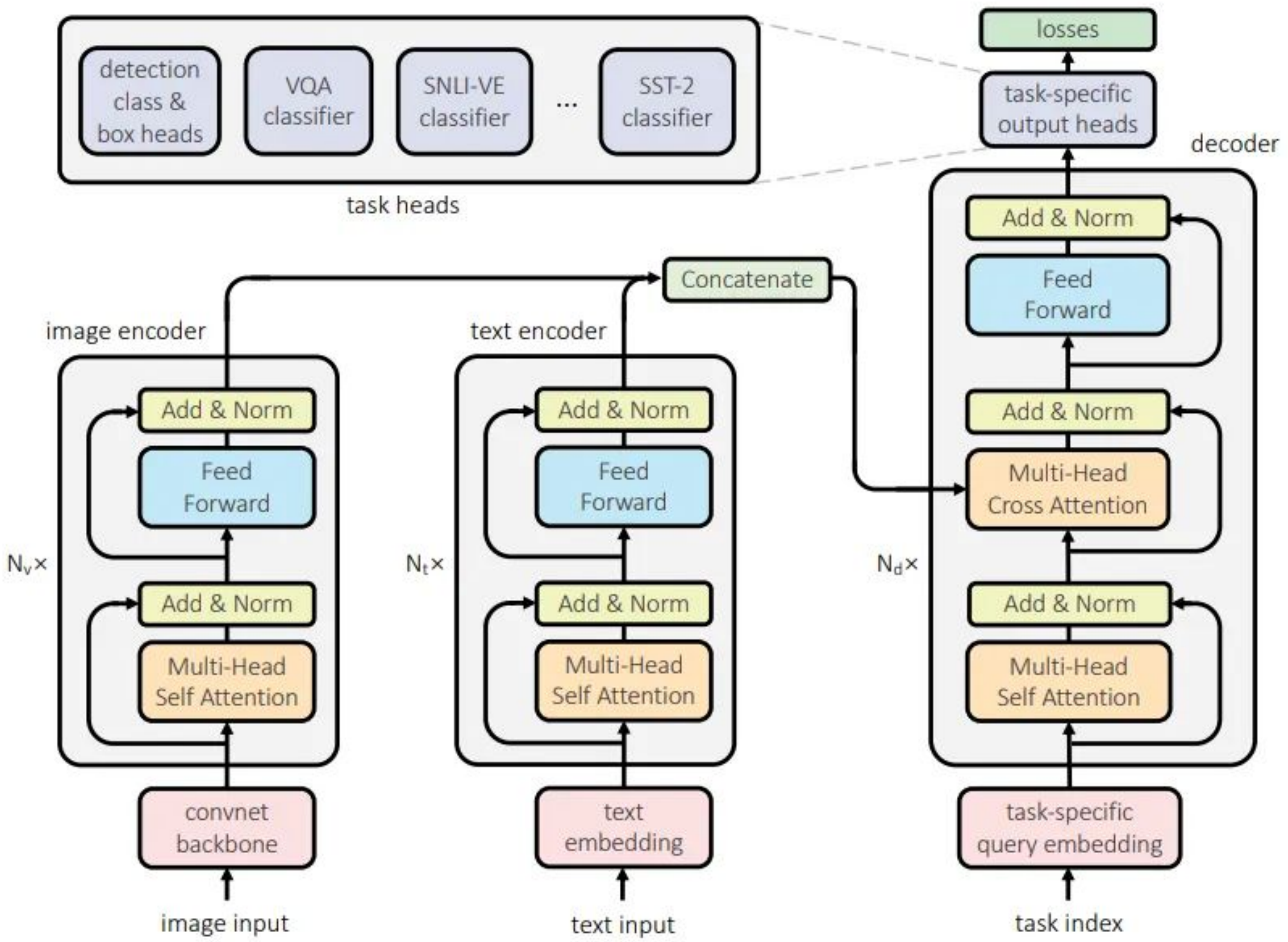
- Exists as encoder-decoder variants or as an encoder-only variant with CTC loss.

- Could be augmented with the CNN.

[Conformer: Convolution-augmented Transformer for Speech Recognition]

[AST: Audio Spectrogram Transformer]

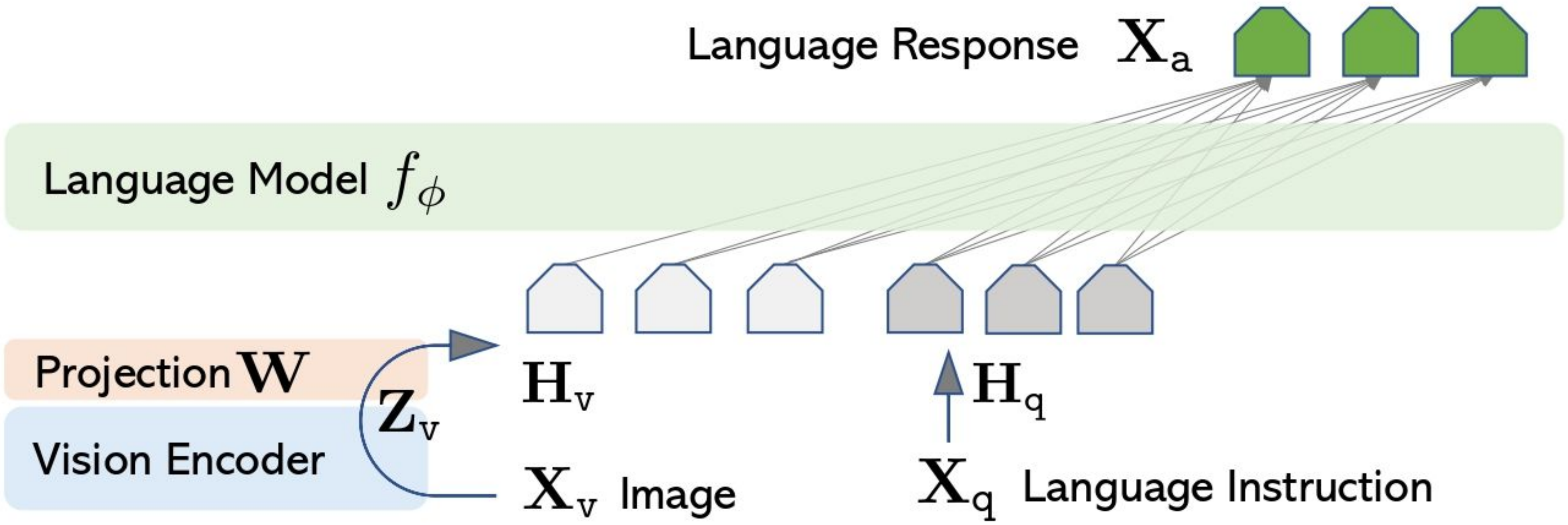
Multimodal Transformer - UniT



1. UniT handles 7 tasks ranging from object detection to vision-and language reasoning and natural language understanding.
2. Components:
    - An image encoder to encode the visual inputs.
    - A text encoder to encode the language inputs.
    - A joint decoder with per-task query embedding.
    - Task-specific heads to make the final outputs for each task.

UniT: Multimodal Multitask Learning with a Unified Transformer

Multimodal Transformer - LLaVA



[Visual Instruction Tuning (LLaVA - Large Language and Vision Assistant)]

Multimodal Transformer - LLaVA

Start a new conversation, and the history is cleared.

User

Do you know who drew this painting?

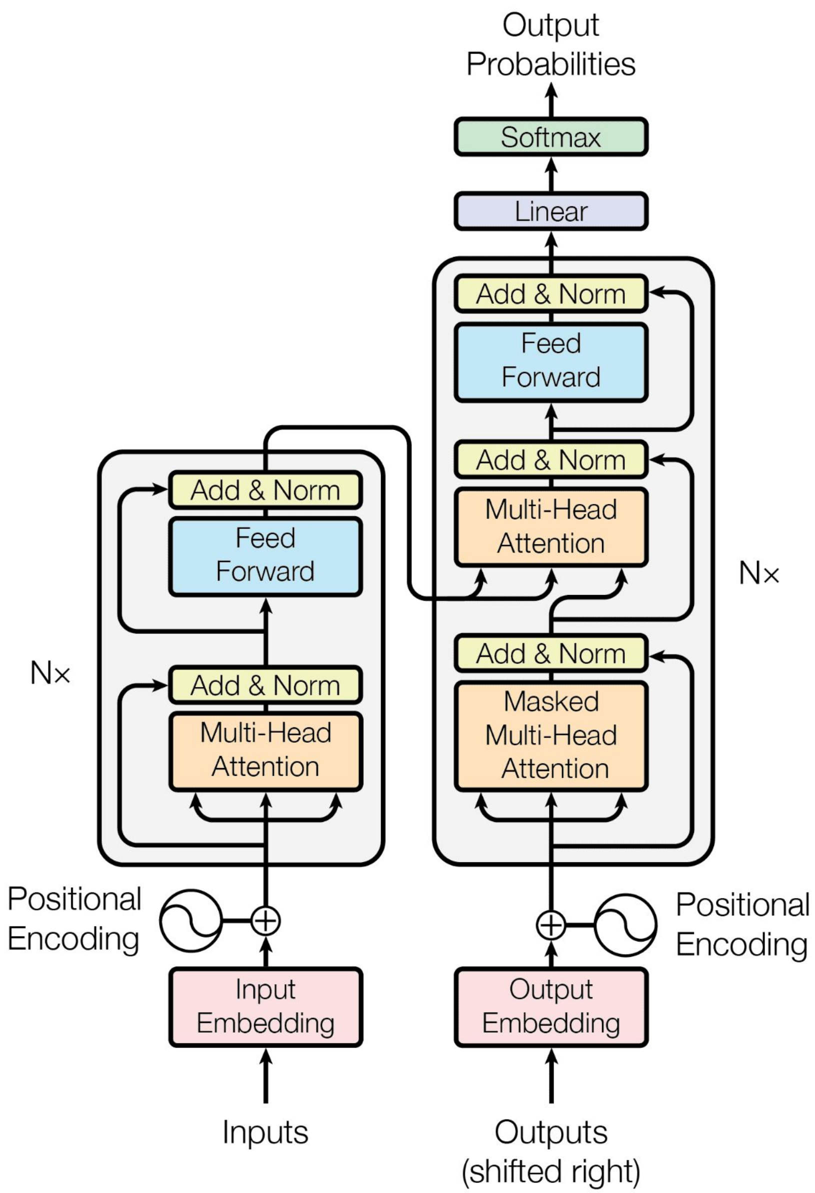
LLaVA

The painting depicts a dog in a humorous situation, where it appears to be wearing a woman's dress and standing in front of a landscape. Although the artist of the painting is not explicitly identified, it could be a reference to the famous artist Da Vinci, known for his artistic works, including the Mona Lisa. The painting is likely a creative and comical take on the traditional portrait style, replacing a human figure with a dog dressed in a woman's clothing, making it an interesting and amusing visual piece.

[Visual Instruction Tuning (LLaVA - Large Language and Vision Assistant)]

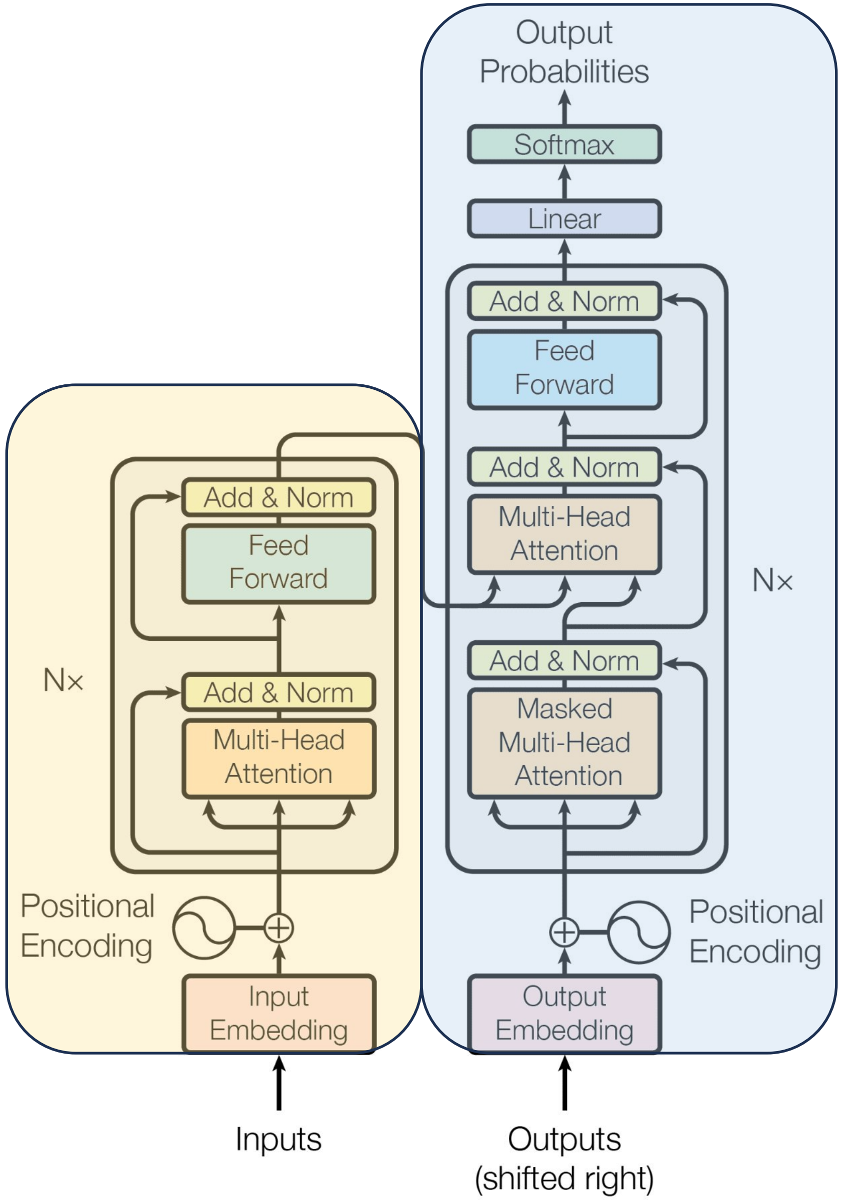
Part 4

Large Language Models



Transformers, mid-2017

Representation

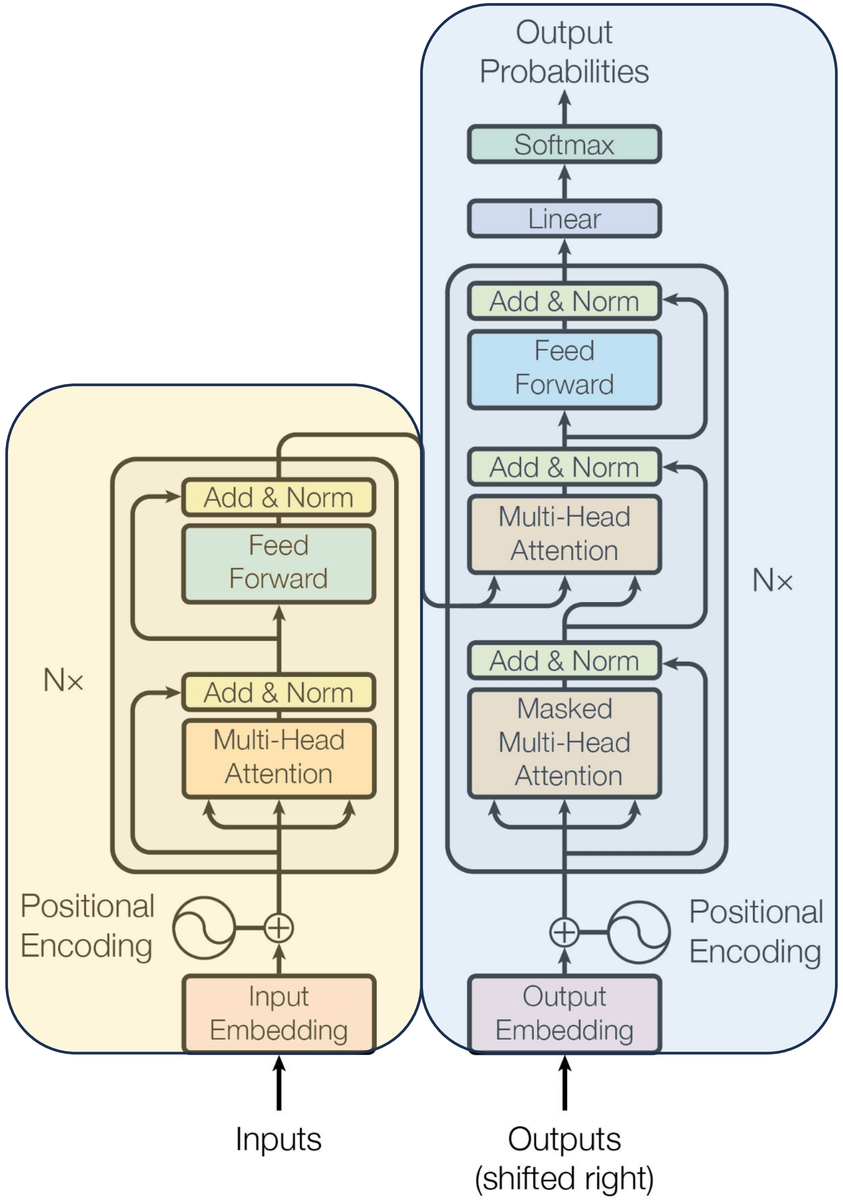


Generation

2018 – Inception of the LLM Era

BERT
Oct 2018

Representation



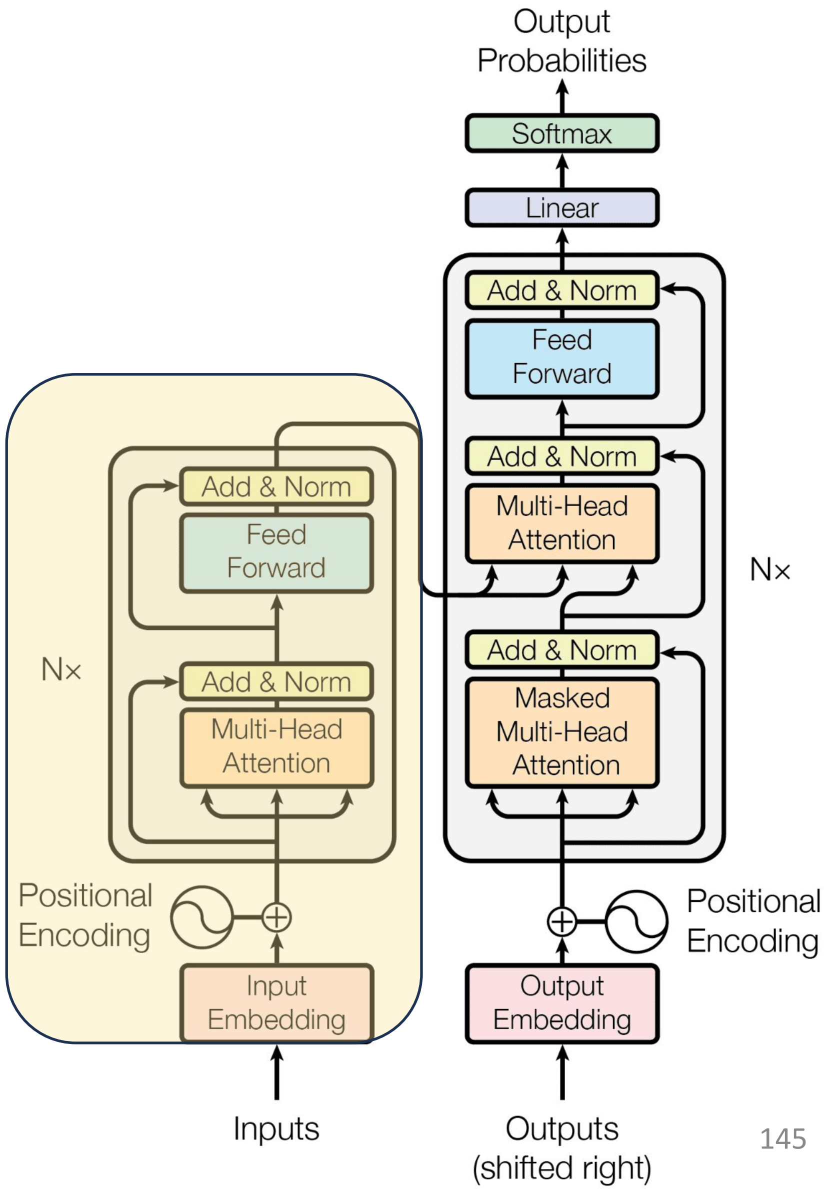
GPT
June 2018

Generation

BERT - Bidirectional Encoder Representations

- One of the biggest challenges in LM-building used to be the lack of task-specific training data.

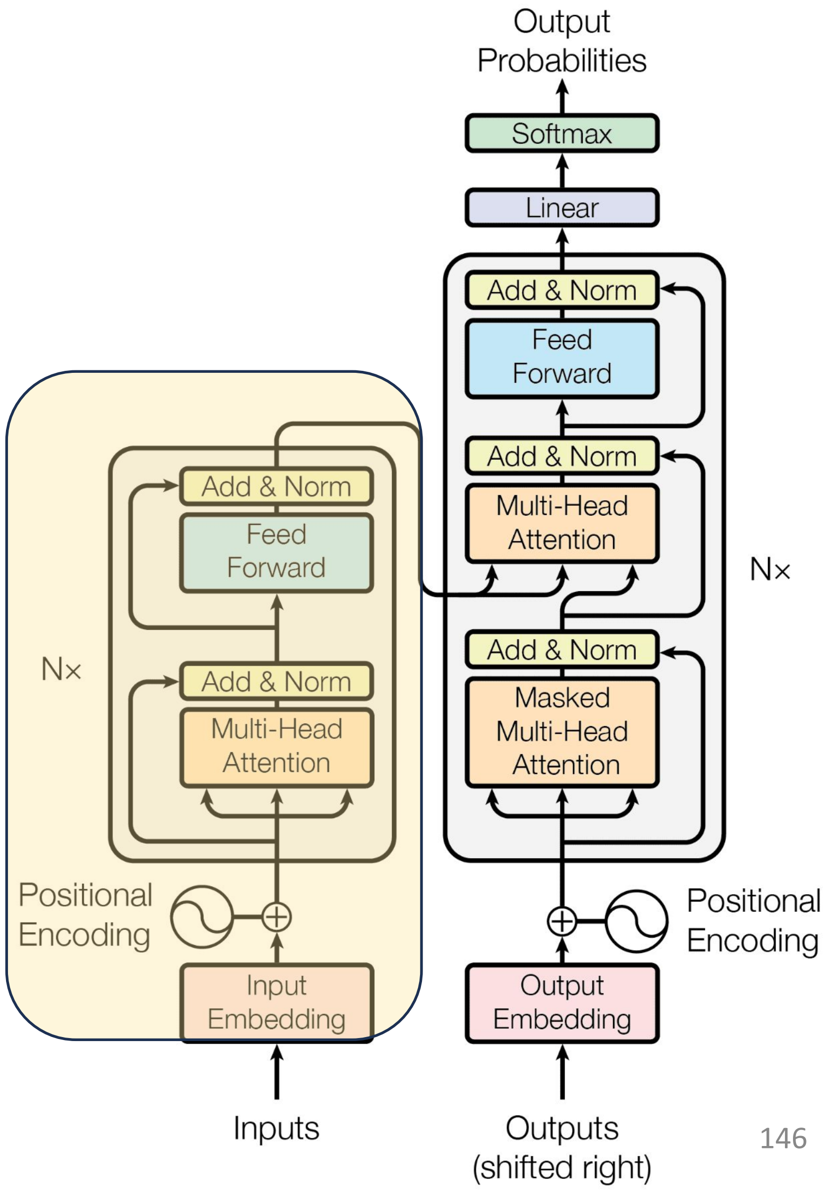
- What if we learn an effective representation that can be applied to a variety of downstream tasks?
  - Word2vec (2013)
  - GloVe (2014)



BERT - Bidirectional Encoder Representations

BERT Pre-Training Corpus:

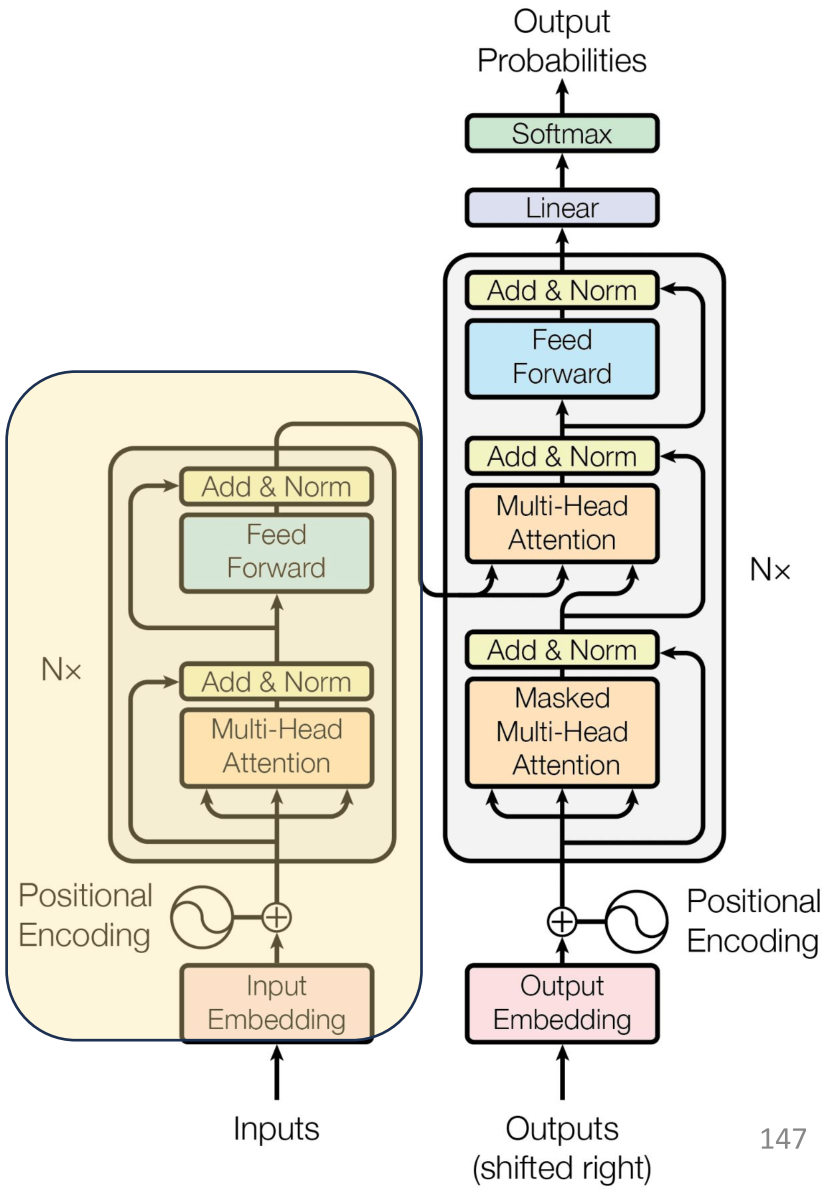
- English Wikipedia - 2,500 million words
- Book Corpus - 800 million words



BERT - Bidirectional Encoder Representations

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)

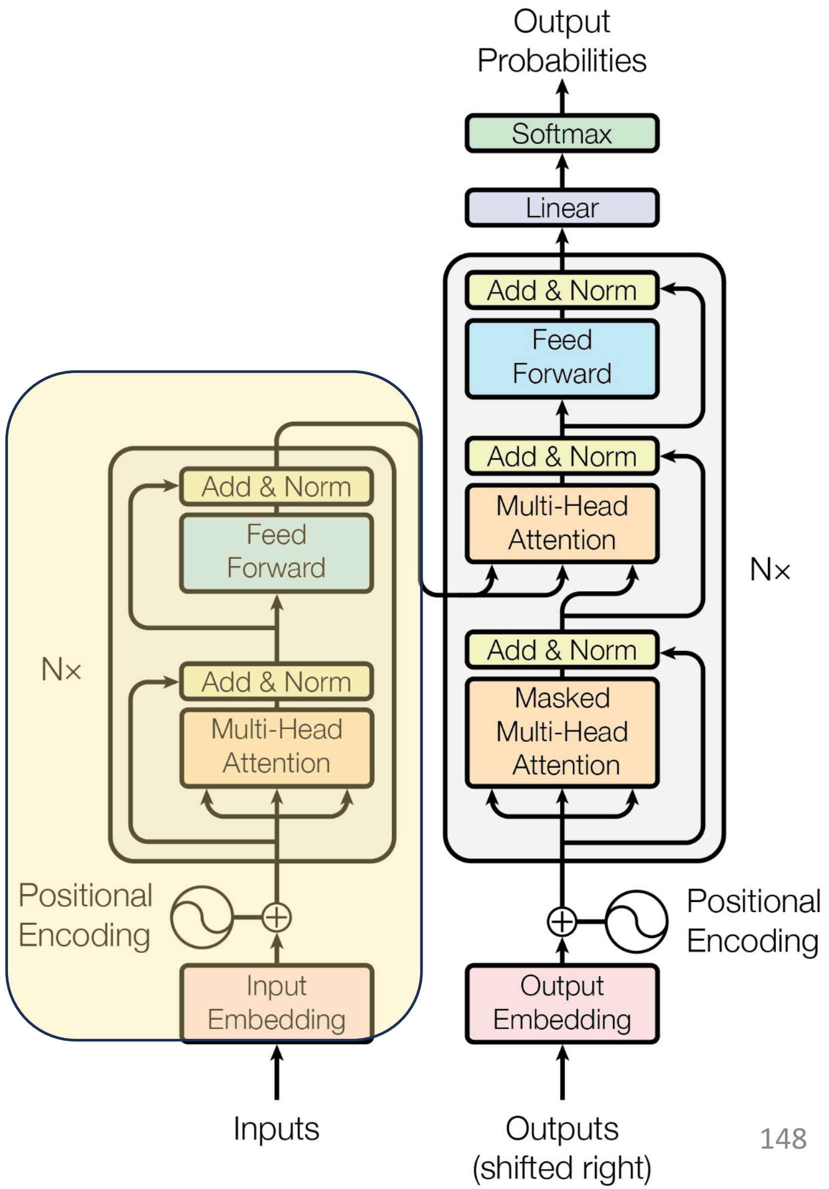


BERT - Bidirectional Encoder Representations

BERT Pre-Training Corpus:
- English Wikipedia - 2,500 million words
- Book Corpus - 800 million words

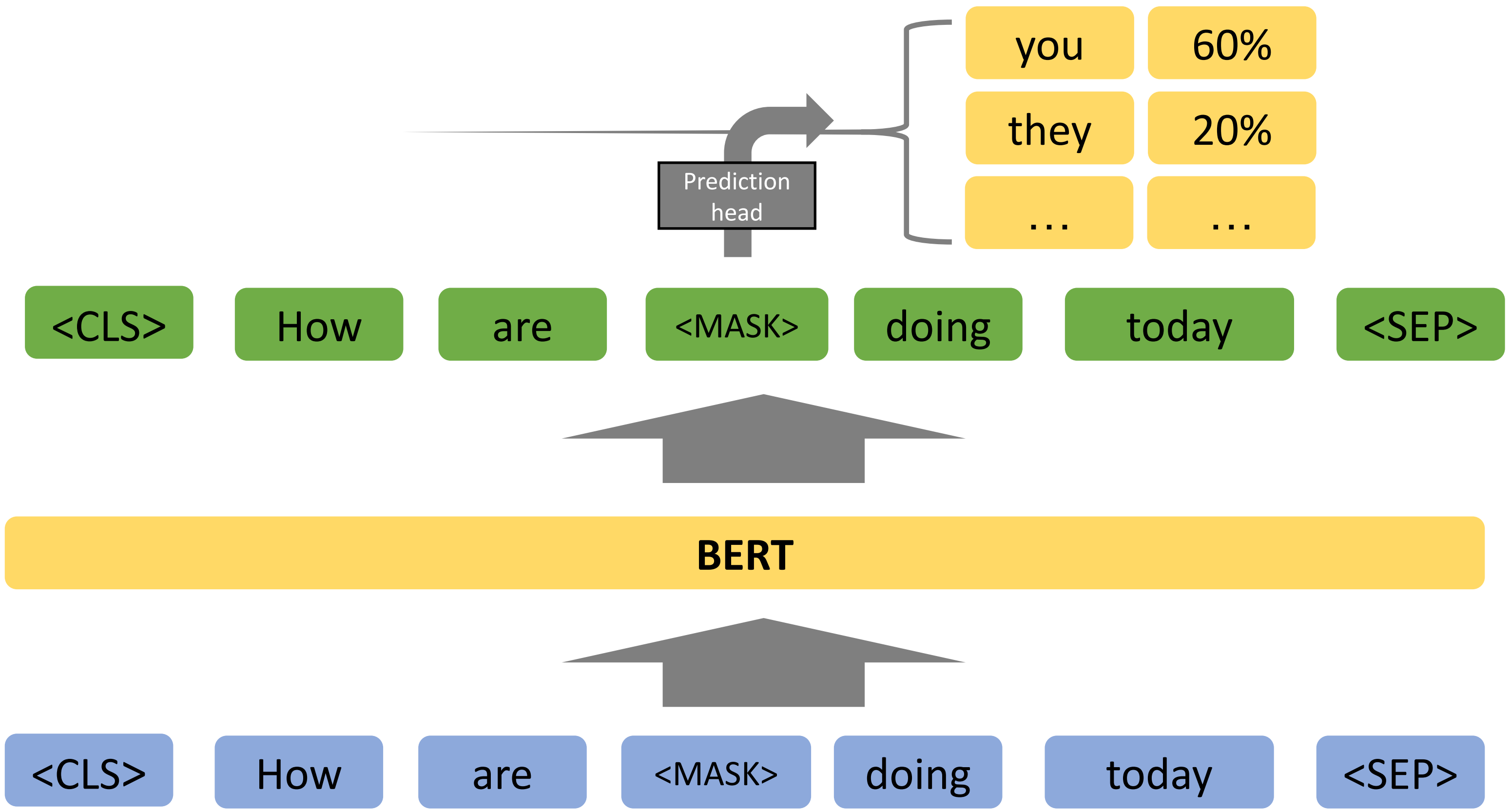
BERT Pre-Training Tasks:
- MLM (Masked Language Modeling)
- NSP (Next Sentence Prediction)

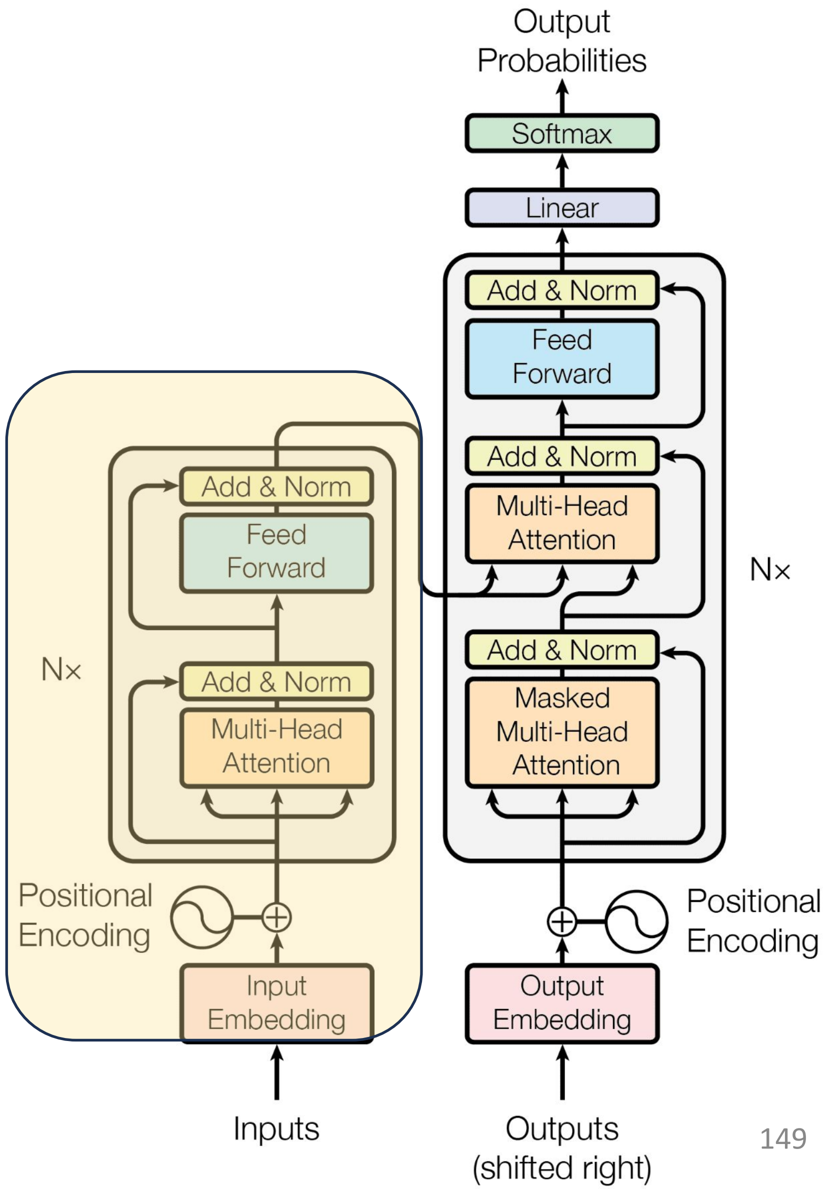
BERT Pre-Training Results:
- BERT-Base – 110M Params
- BERT-Large – 340M Params



BERT - Bidirectional Encoder Representations

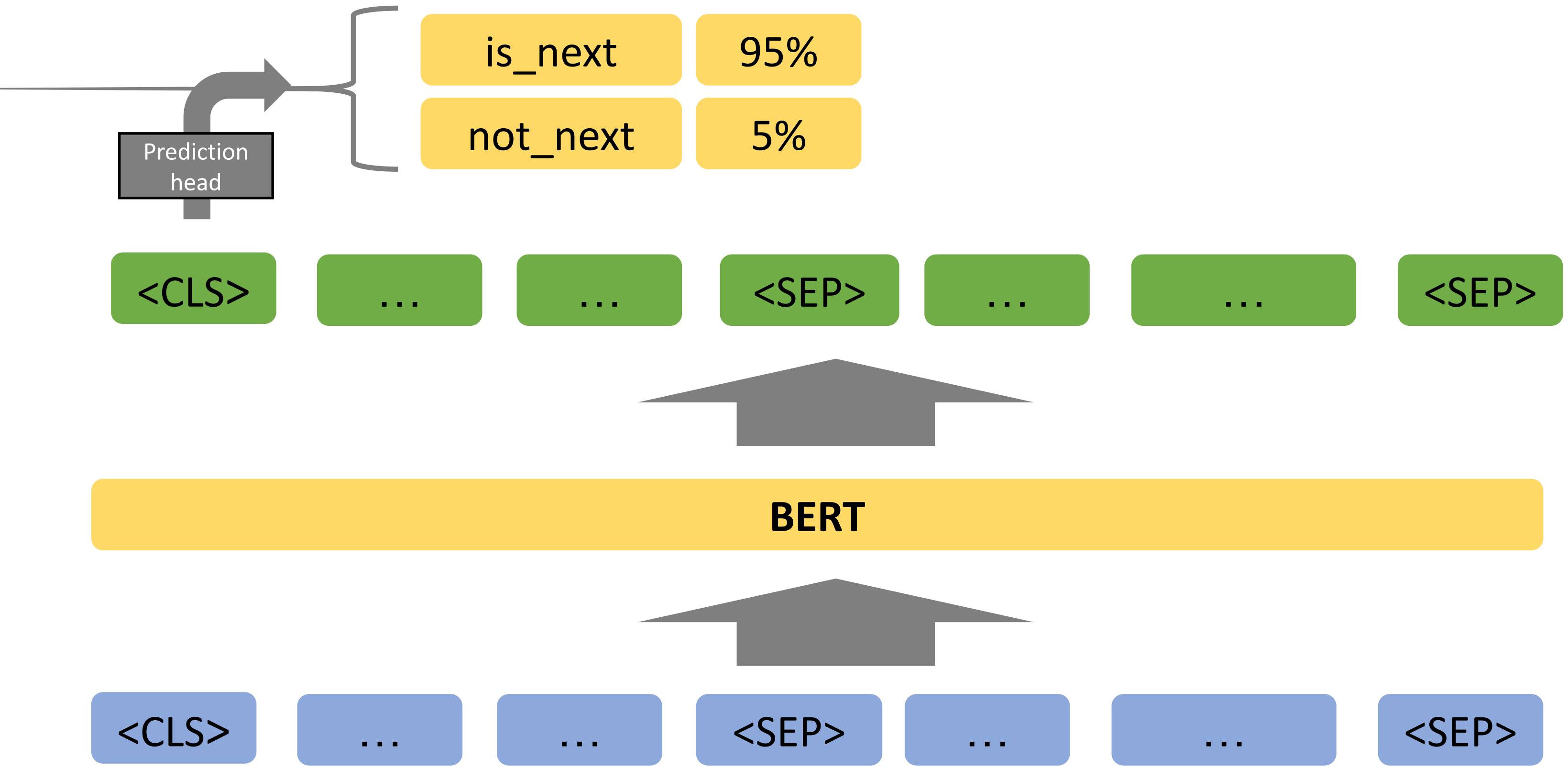
MLM (Masked Language Modeling)

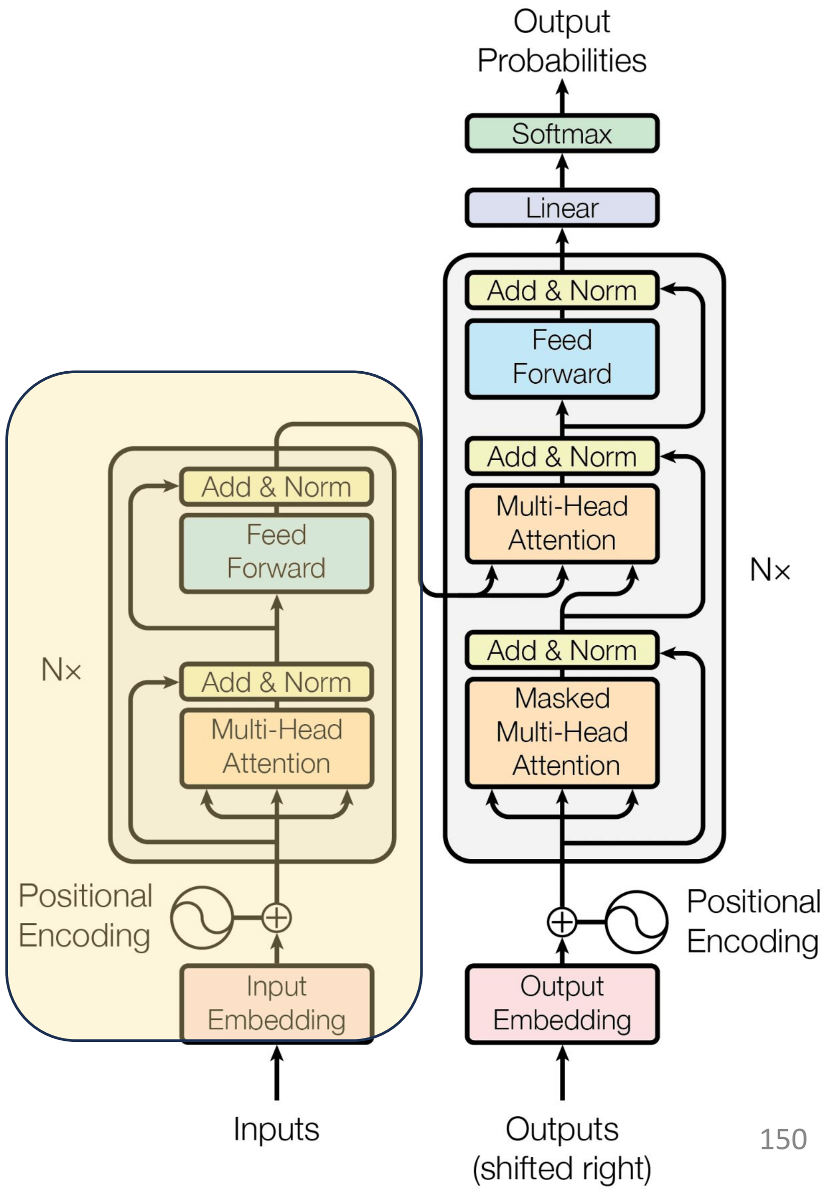




BERT - Bidirectional Encoder Representations

NSP (Next Sentence Prediction)

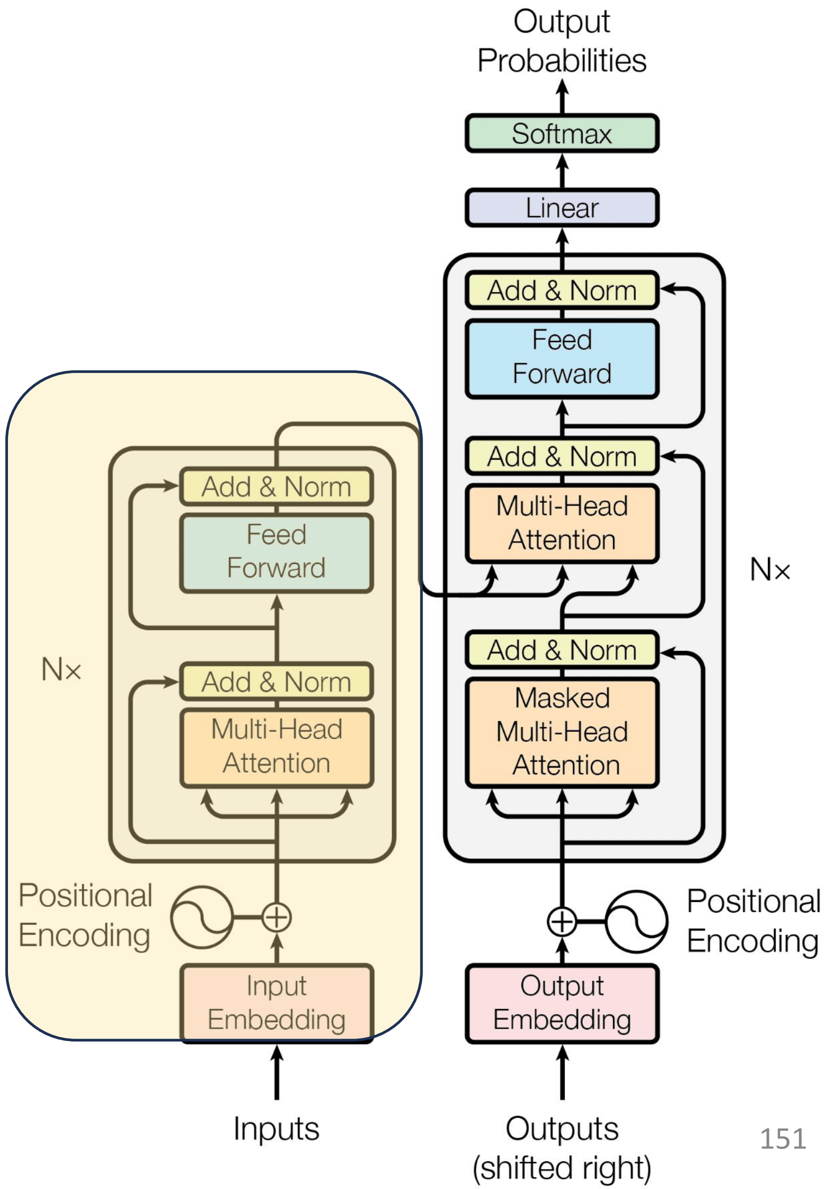




BERT - Bidirectional Encoder Representations

BERT Fine-Tuning:

- Simply add a task-specific module after the last encoder layer to map it to the desired dimension.
  - *Classification Tasks:*
    - *Add a feed-forward layer on top of the encoder output for the [CLS] token*
  - *Question Answering Tasks:*
    - *Train two extra vectors to mark the beginning and end of answer from paragraph*
  - ...

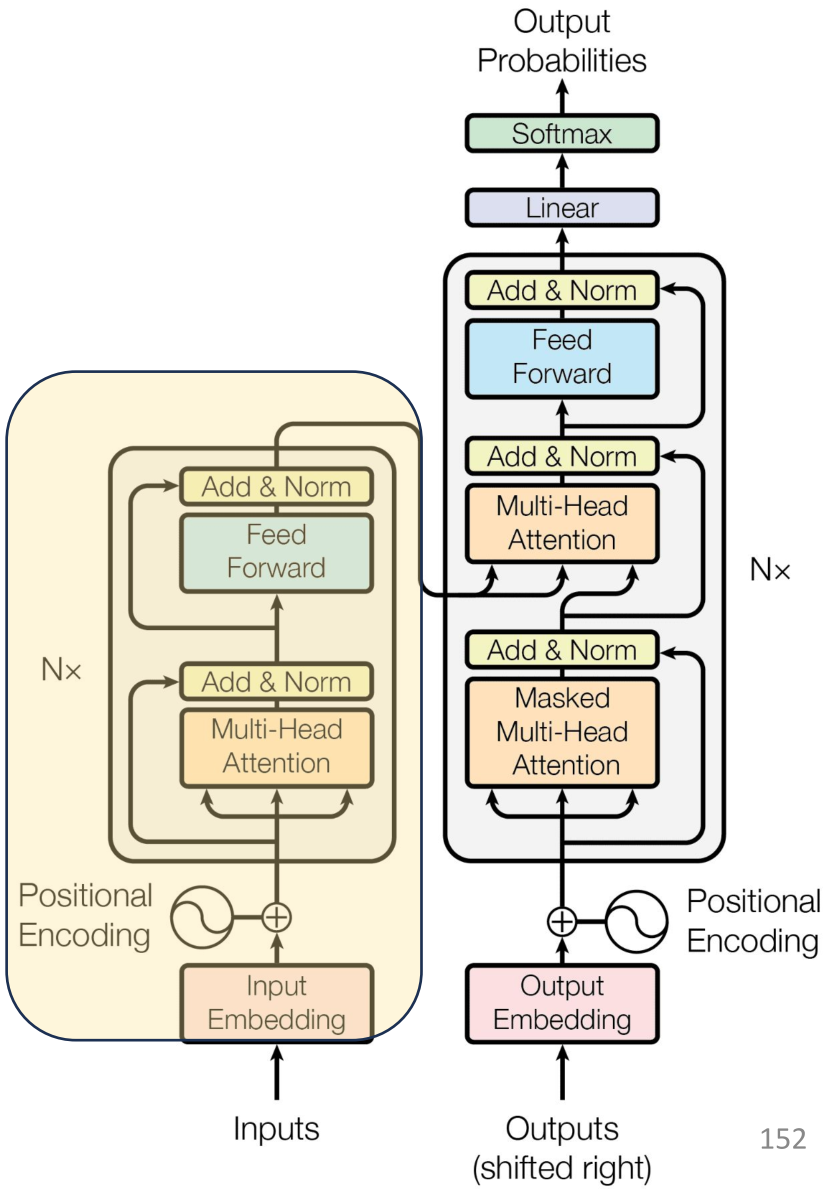


BERT - Bidirectional Encoder Representations

BERT Evaluation:

- General Language Understanding Evaluation (GLUE)
  - Sentence pair tasks
  - Single sentence classification

- Stanford Question Answering Dataset (SQuAD)



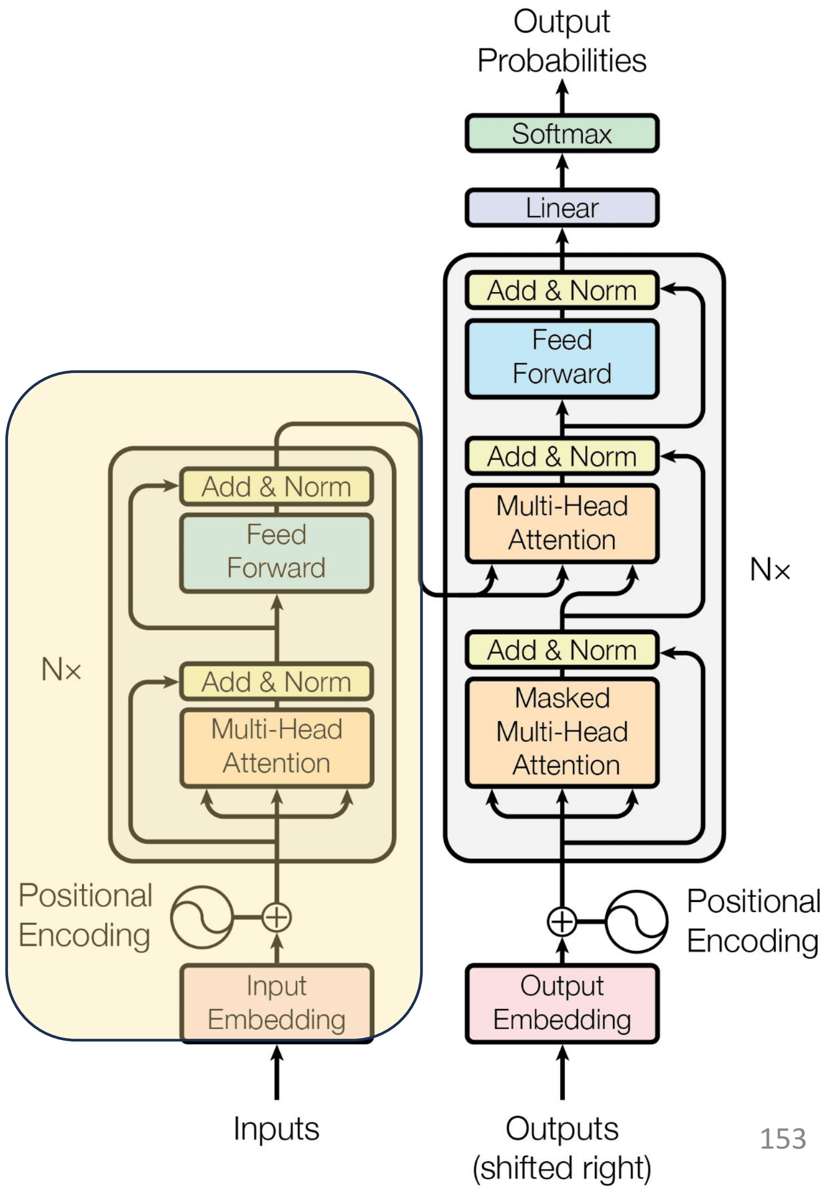
BERT - Bidirectional Encoder Representations

BERT Evaluation:

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
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**

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

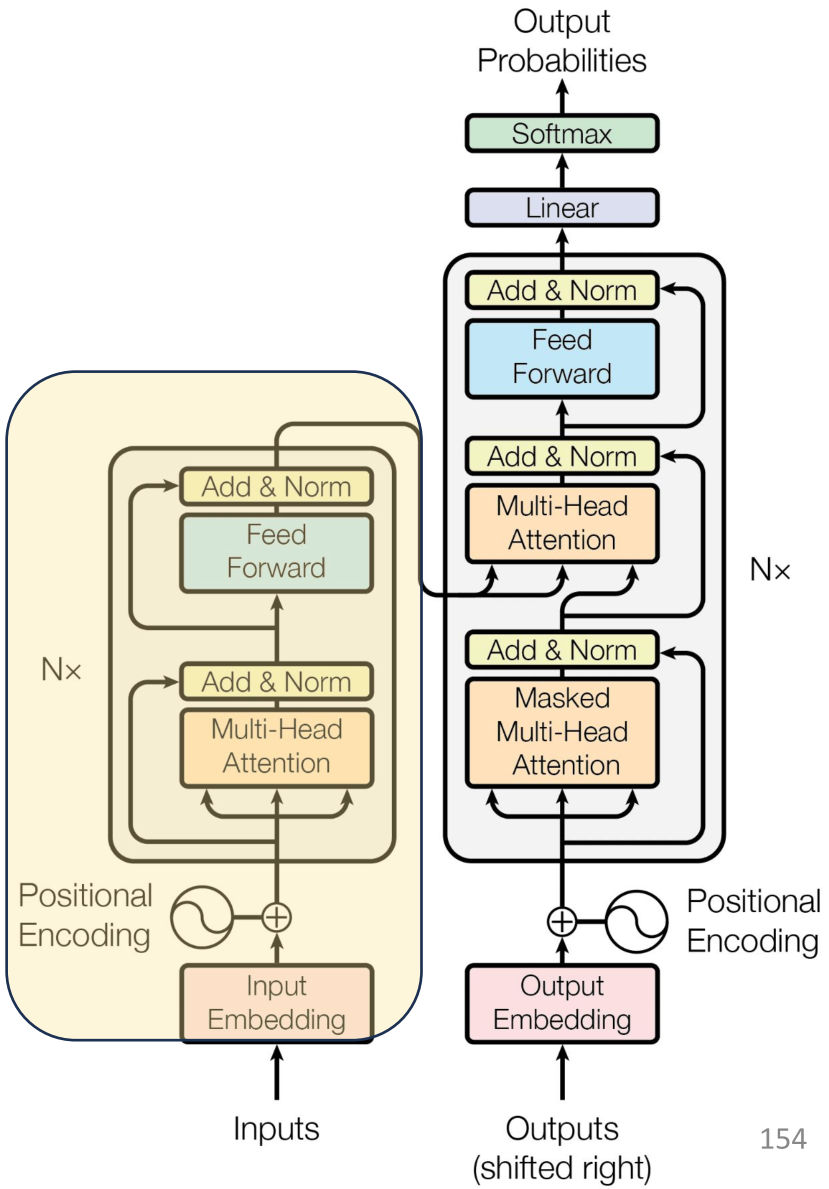
Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.



BERT - Bidirectional Encoder Representations

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.

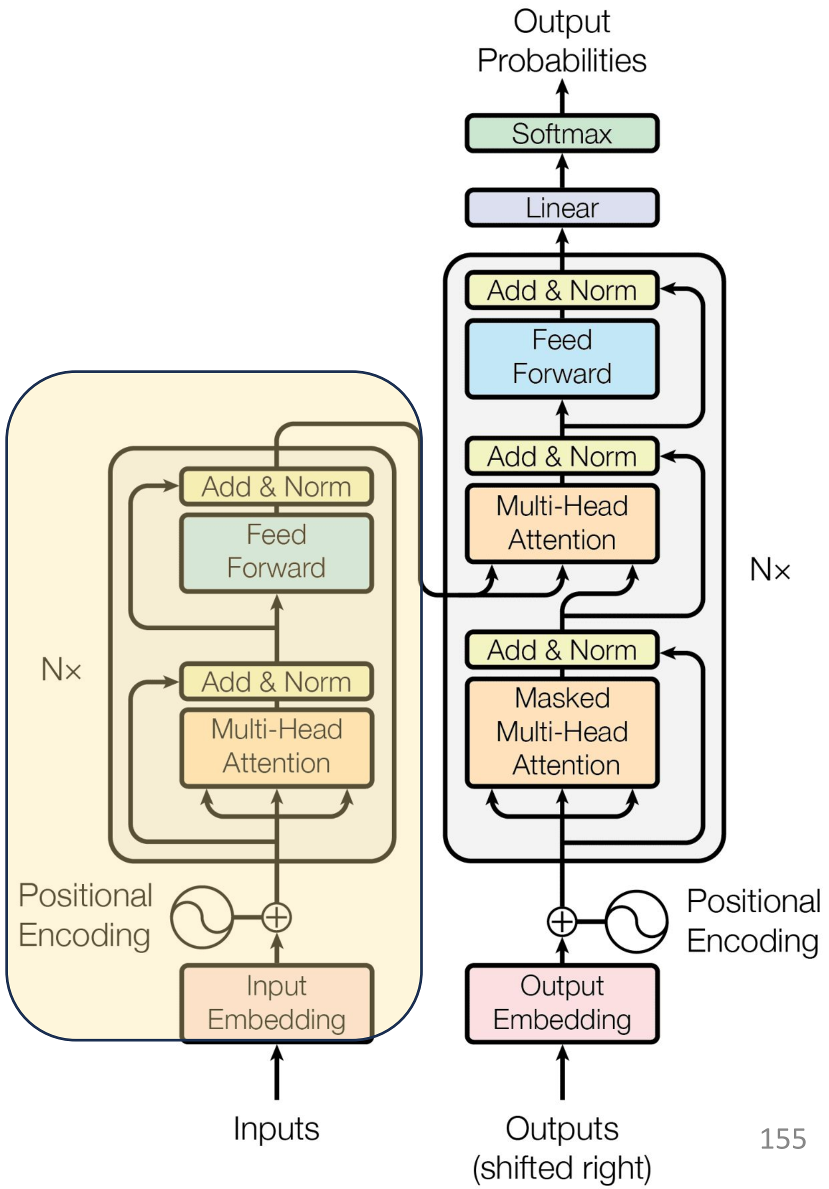


BERT - Bidirectional Encoder Representations

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.



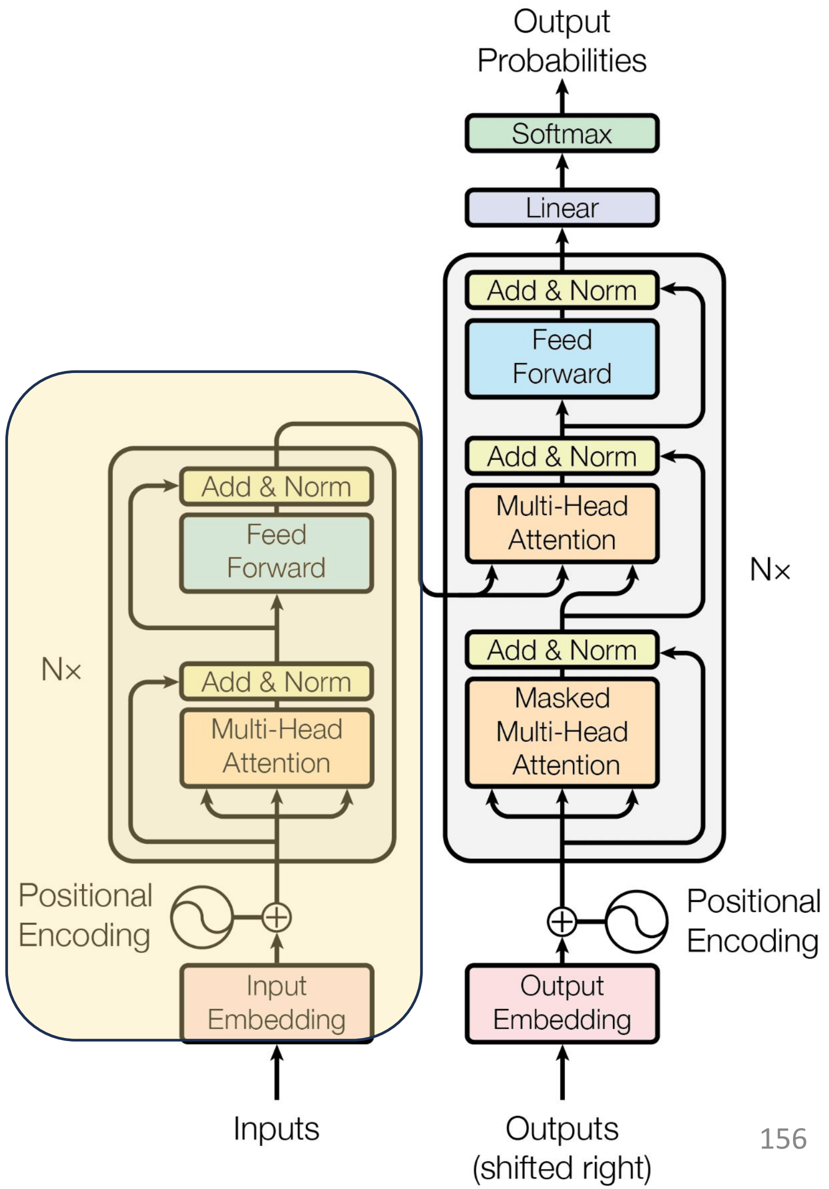
BERT - Bidirectional Encoder Representations

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.

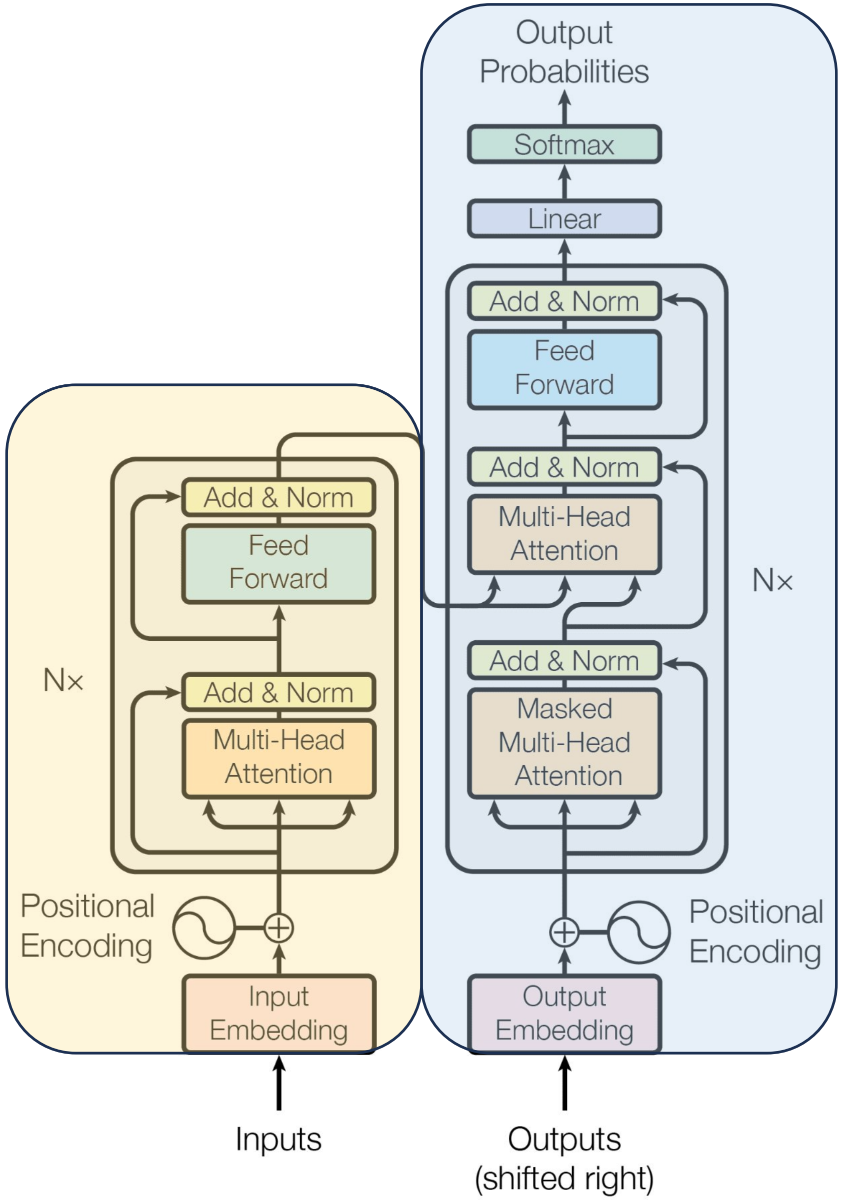
- **And scaling works!!!**
  - 340M was considered large in 2018



2018 – Inception of the LLM Era

BERT
Oct 2018

Representation

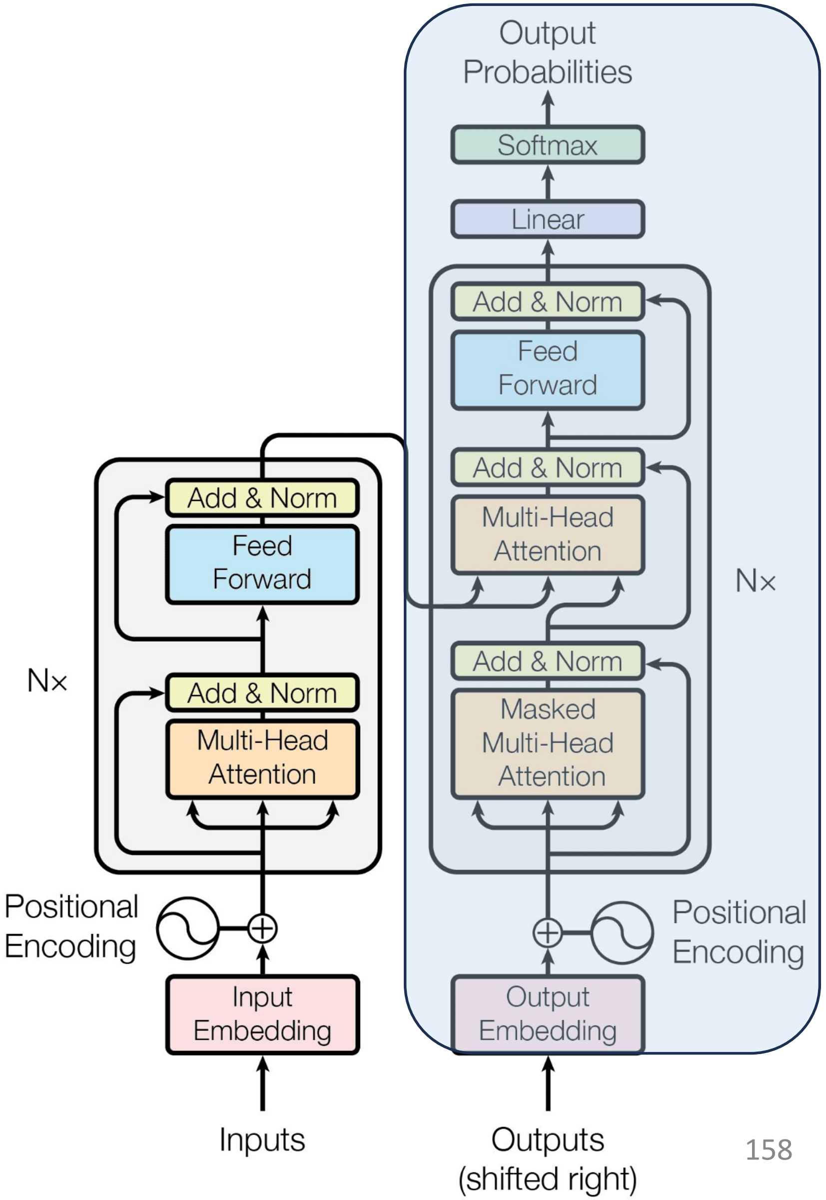


GPT
June 2018

Generation

GPT – Generative Pretrained Transformer

- Similarly motivated as BERT, though differently designed
  - Can we leverage large amounts of unlabeled data to pretrain an LM that understands general patterns?

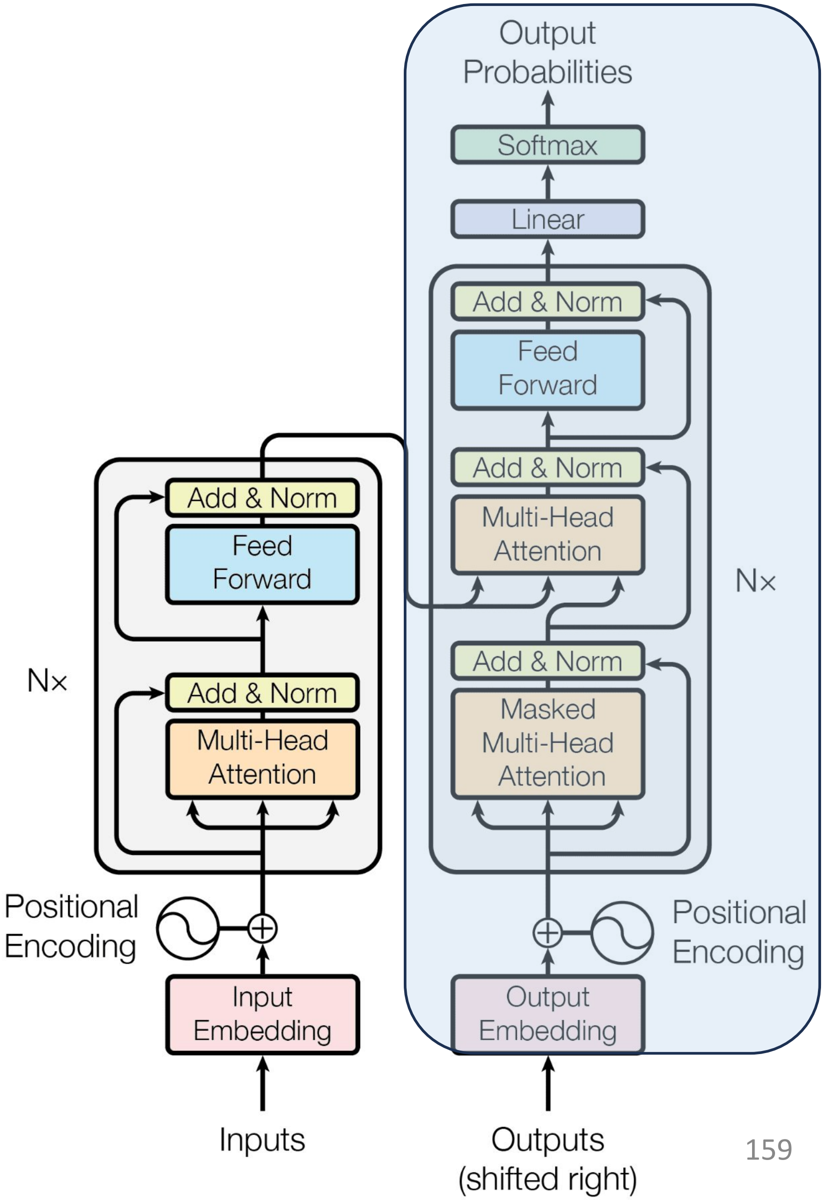


GPT – Generative Pretrained Transformer

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

GPT Fine-Tuning:

- Prompt-format task-specific text as a continuous stream for the model to fit

Summarization

Summarize this article:

The summary is:

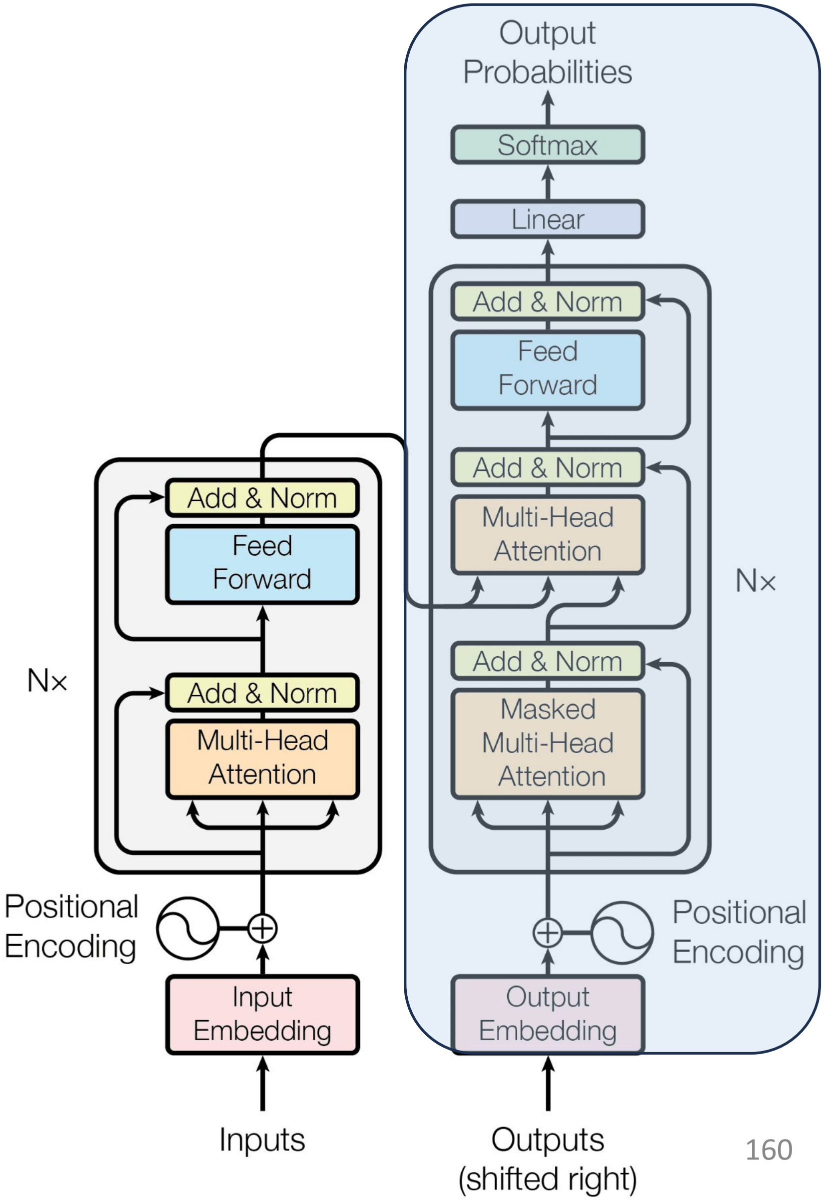
QA

Answer the question based on the context.

Context:

Question:

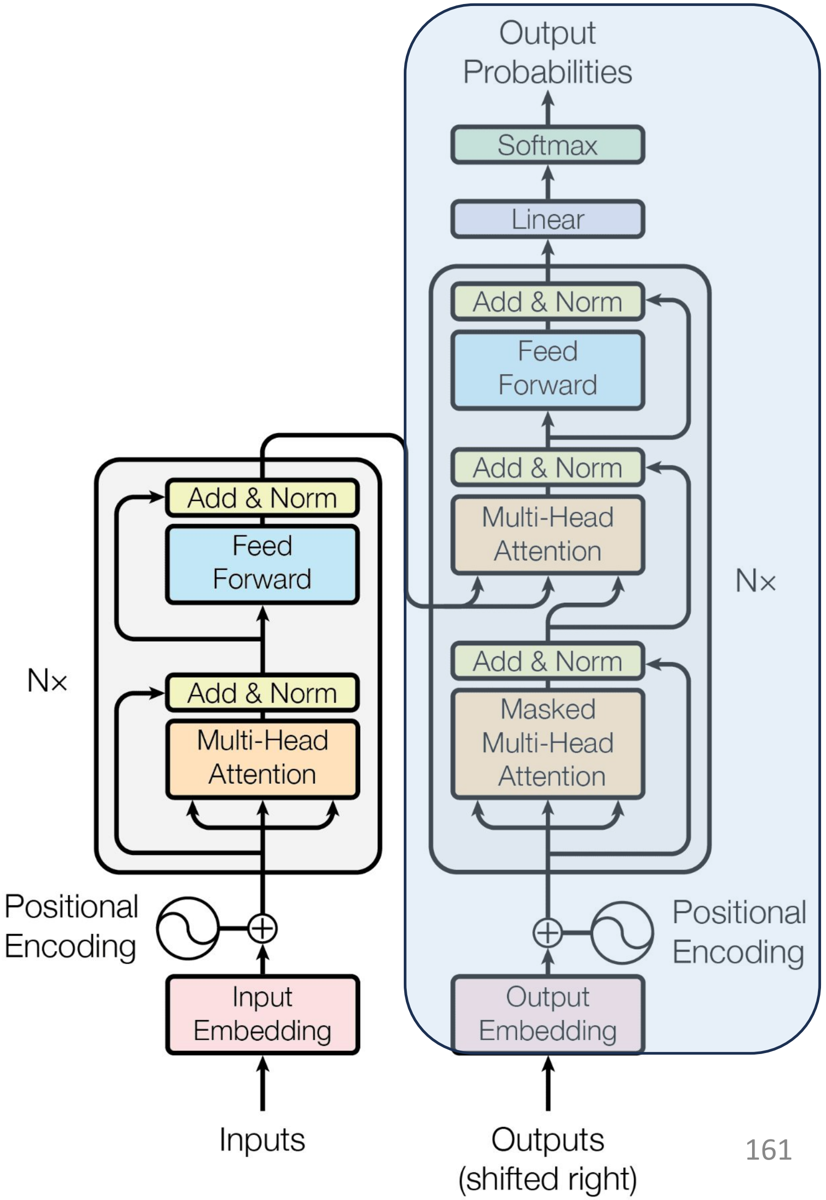
Answer:



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.

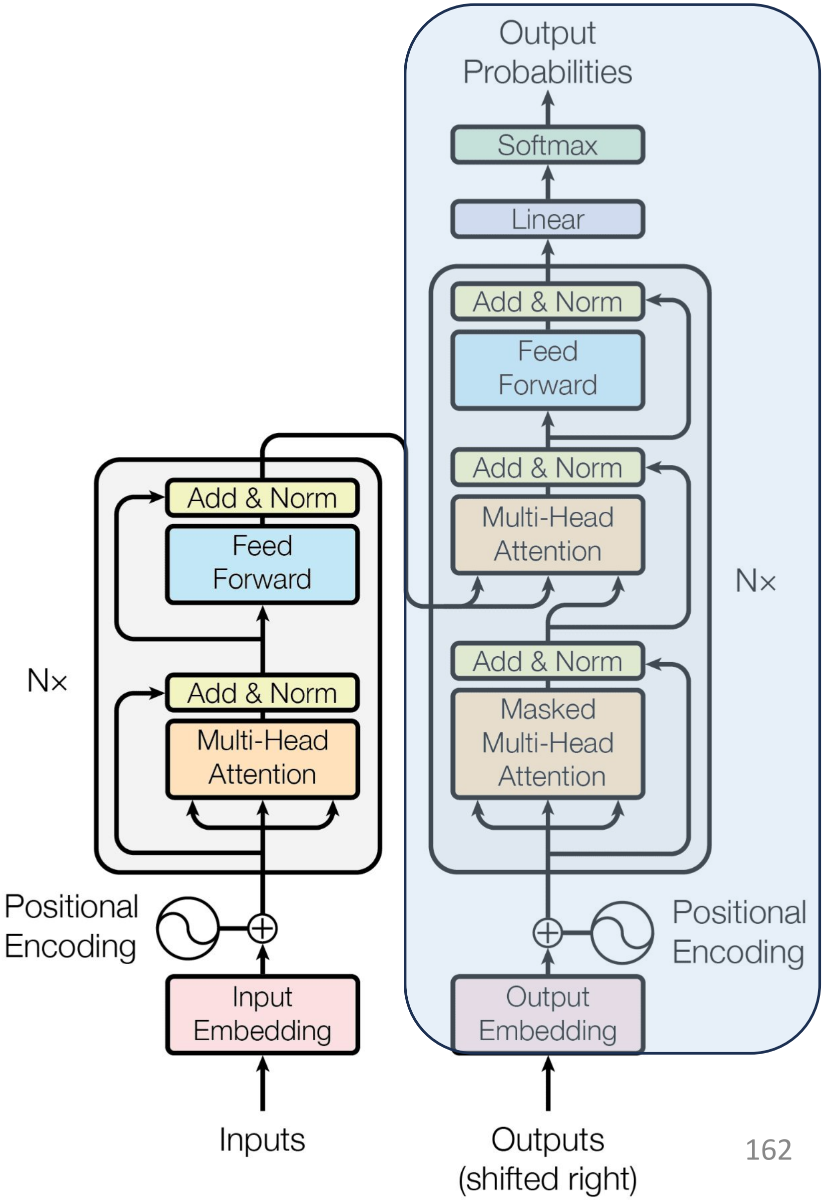


GPT – Generative Pretrained Transformer

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.



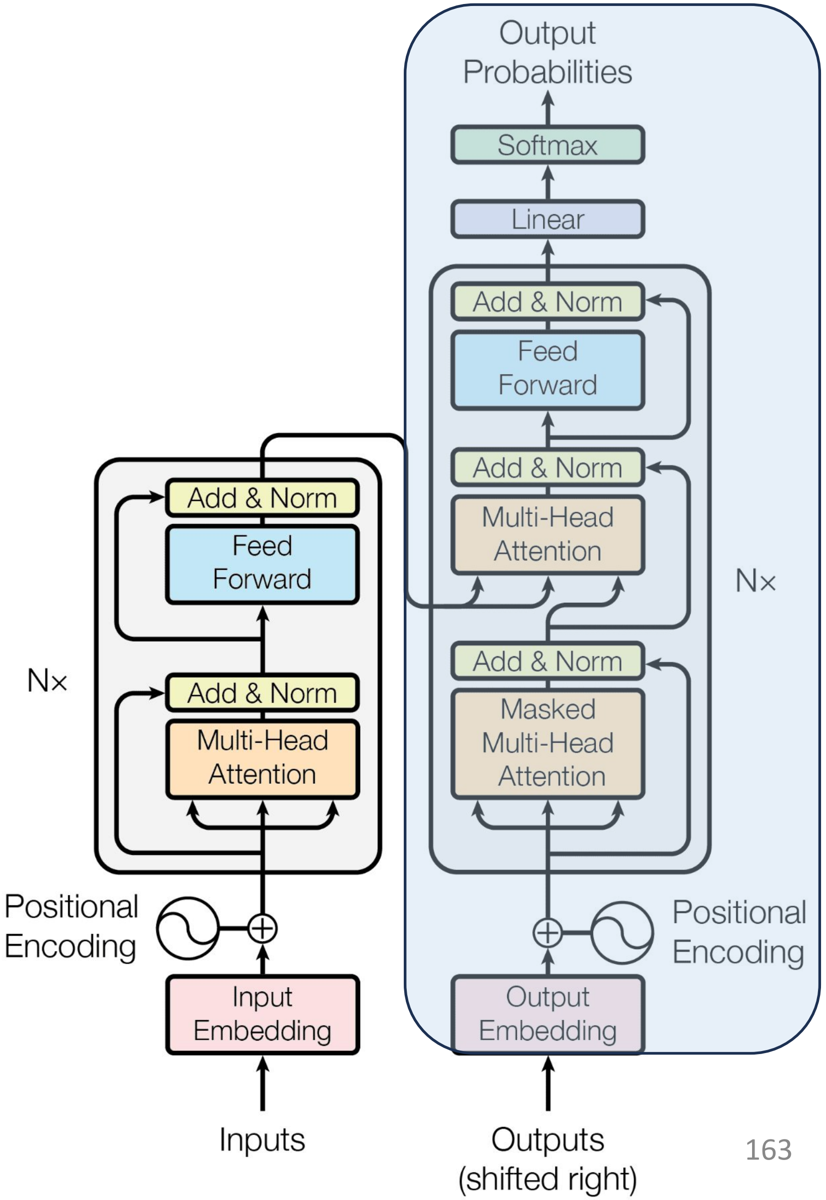
GPT – Generative Pretrained Transformer

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!!!***



Poll 3 - @1579

The original GPT's parameter count is closest to…

A. 117
B. 117K
C. 117M
D. 117B

Poll 3 - @1579

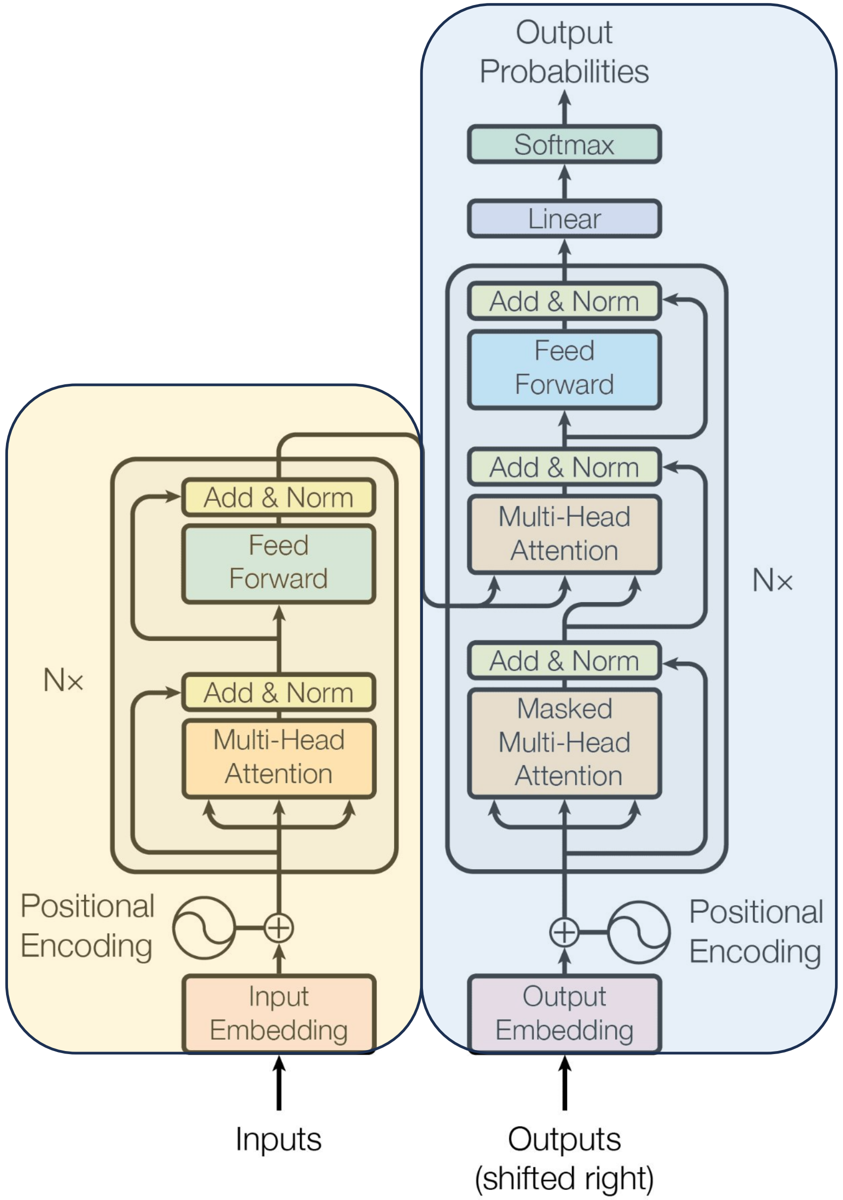
The original GPT's parameter count is closest to…

A. 117
B. 117K
C. **117M**
D. 117B

The LLM Era – Paradigm Shift in Machine Learning

BERT
Oct 2018

Representation



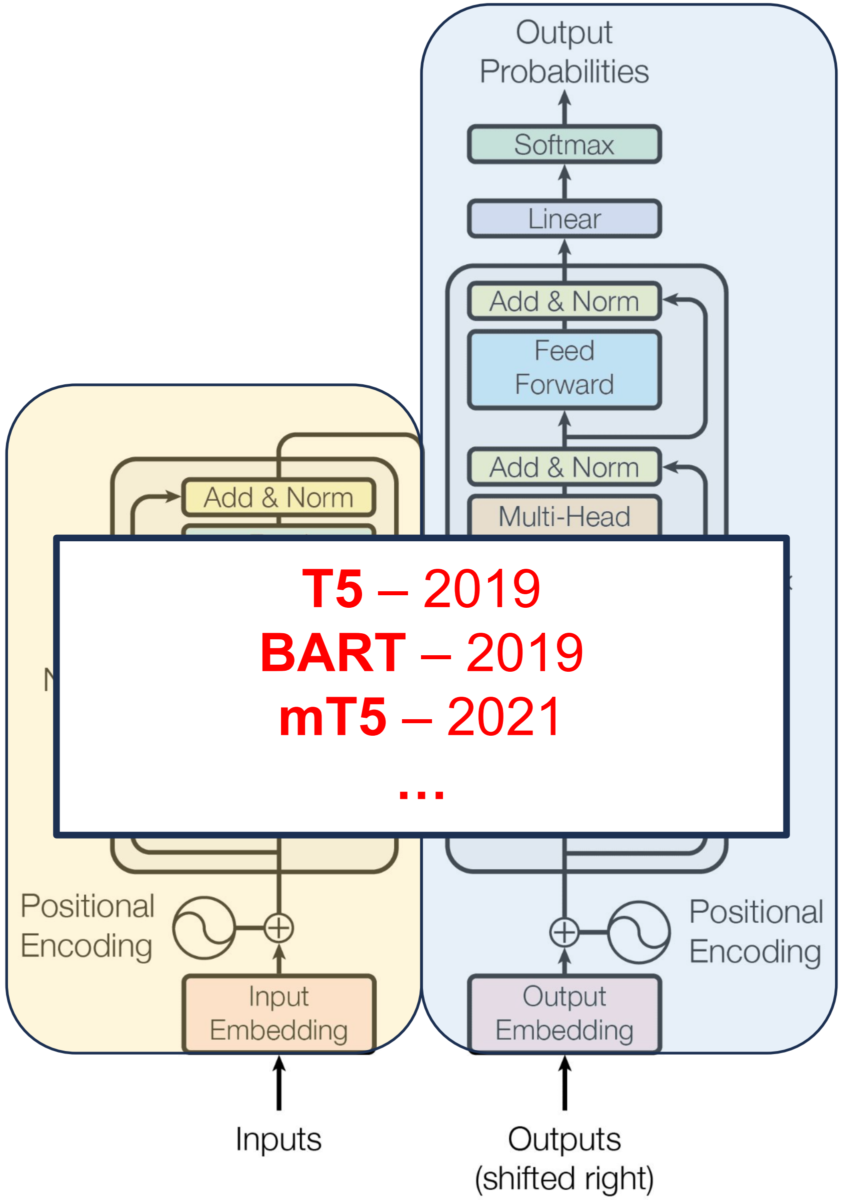
GPT
June 2018

Generation

The LLM Era – Paradigm Shift in Machine Learning

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

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs	Since LLMs
• **Feature Engineering** • How do we design or select the best features for a task?	

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?

Since LLMs

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?

Since LLMs

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs | **Since LLMs**

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
  - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
  - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
  - How do we leverage large scales of unlabeled data out there previously under-leveraged?

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
  - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
  - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
  - How can we make models perform on tasks they are *not* trained on?

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
  - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
  - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
  - How can we make models perform on tasks they are *not* trained on?
- **Prompting**
  - How do we make models understand their task simply by describing it in natural language?

The LLM Era – Paradigm Shift in Machine Learning

From both BERT and GPT, we learn that…

- Transformers seem to provide a new class of generalist models that are capable of capturing knowledge which is more fundamental than task-specific abilities.

Before LLMs

- **Feature Engineering**
  - How do we design or select the best features for a task?
- **Model Selection**
  - Which model is best for which type of task?
- **Transfer Learning**
  - Given scarce labeled data, how do we transfer knowledge from other domains?
- **Overfitting vs Generalization**
  - How do we balance complexity and capacity to prevent overfitting while maintaining good performance?

Since LLMs

- **Pre-training and Fine-tuning**
  - How do we leverage large scales of unlabeled data out there previously under-leveraged?
- **Zero-shot and Few-shot learning**
  - How can we make models perform on tasks they are *not* trained on?
- **Prompting**
  - How do we make models understand their task simply by describing it in natural language?
- **Interpretability and Explainability**
  - How can we *understand* the inner workings of our own models?

The LLM Era – Paradigm Shift in Machine Learning

- **What has caused this paradigm shift?**

The LLM Era – Paradigm Shift in Machine Learning

- **What has caused this paradigm shift?**
  - **Recall: Problem in recurrent networks**
    - Information is effectively lost during encoding of long sequences
    - Sequential nature disables parallel training and favors late timestep inputs

The LLM Era – Paradigm Shift in Machine Learning

- **What has caused this paradigm shift?**

  - **Recall: Problem in recurrent networks**
    - Information is effectively lost during encoding of long sequences
    - Sequential nature disables parallel training and favors late timestep inputs

  - **Solution: Attention is all you need!!!**
    - Handling long-range dependencies
    - Parallel training
    - Dynamic attention weights based on inputs

The LLM Era – Paradigm Shift in Machine Learning

- **Attention and Transformer – is this the end?**

The LLM Era – Paradigm Shift in Machine Learning

- Attention and Transformer – is this the end?
  - Problem in current Transformer-based LLMs??

The LLM Era – Paradigm Shift in Machine Learning

- **Attention and Transformer – is this the end?**

  - **Problem in current Transformer-based LLMs??**
    - True understanding the material vs. memorization and pattern-matching
    - Cannot reliably follow rules – factual hallucination e.g. inability in arithmetic

The LLM Era – Paradigm Shift in Machine Learning

- **Attention and Transformer – is this the end?**
  - **Problem in current Transformer-based LLMs??**
    - True understanding the material vs. memorization and pattern-matching
    - Cannot reliably follow rules – factual hallucination e.g. inability in arithmetic
  - **Solution: ???**

Looking Back

It is true that language models are just programmed to predict the next token…

In fact, all animals, including us, are just programmed to survive and reproduce, and yet amazingly complex and beautiful stuff comes from it.

- Sam Altman*

***Paraphrased**