

Introduction to Deep Learning

Lecture 19

Transformers and LLMs

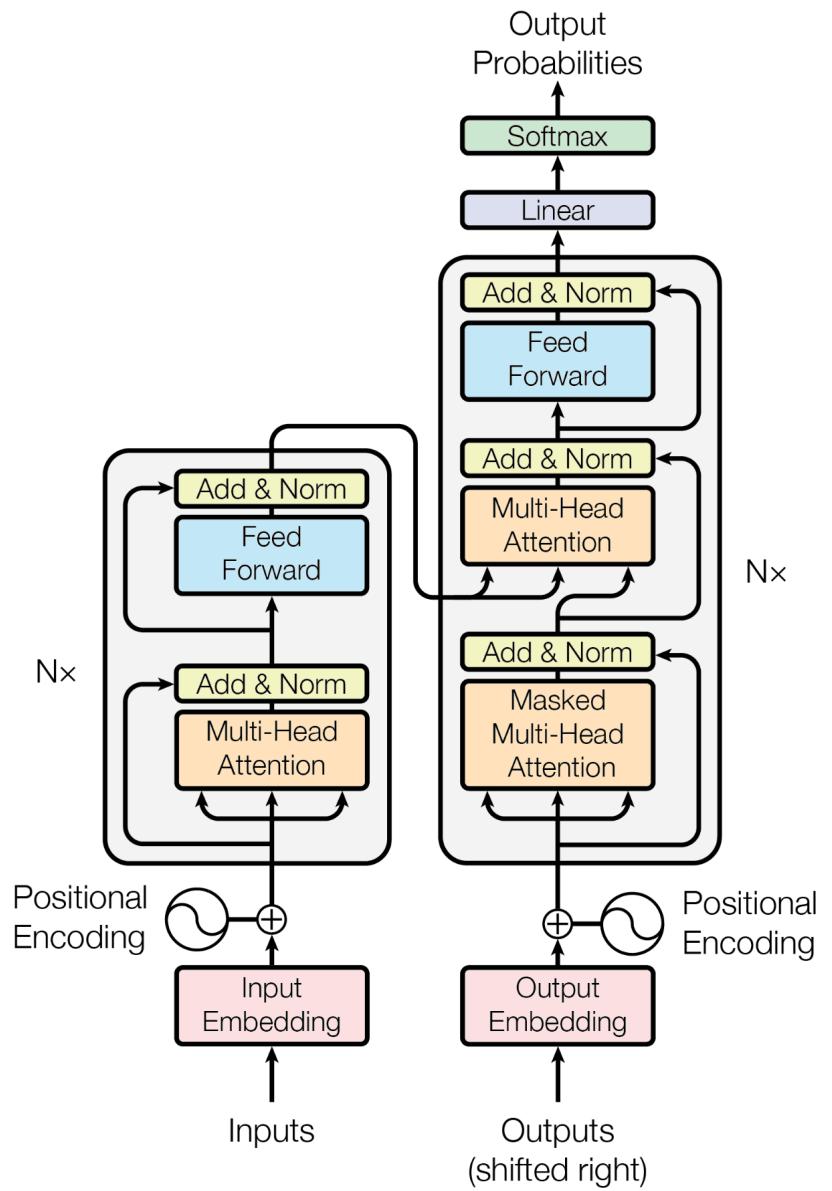
Shikhar Agnihotri

Liangze Li

11-785, Fall 2023

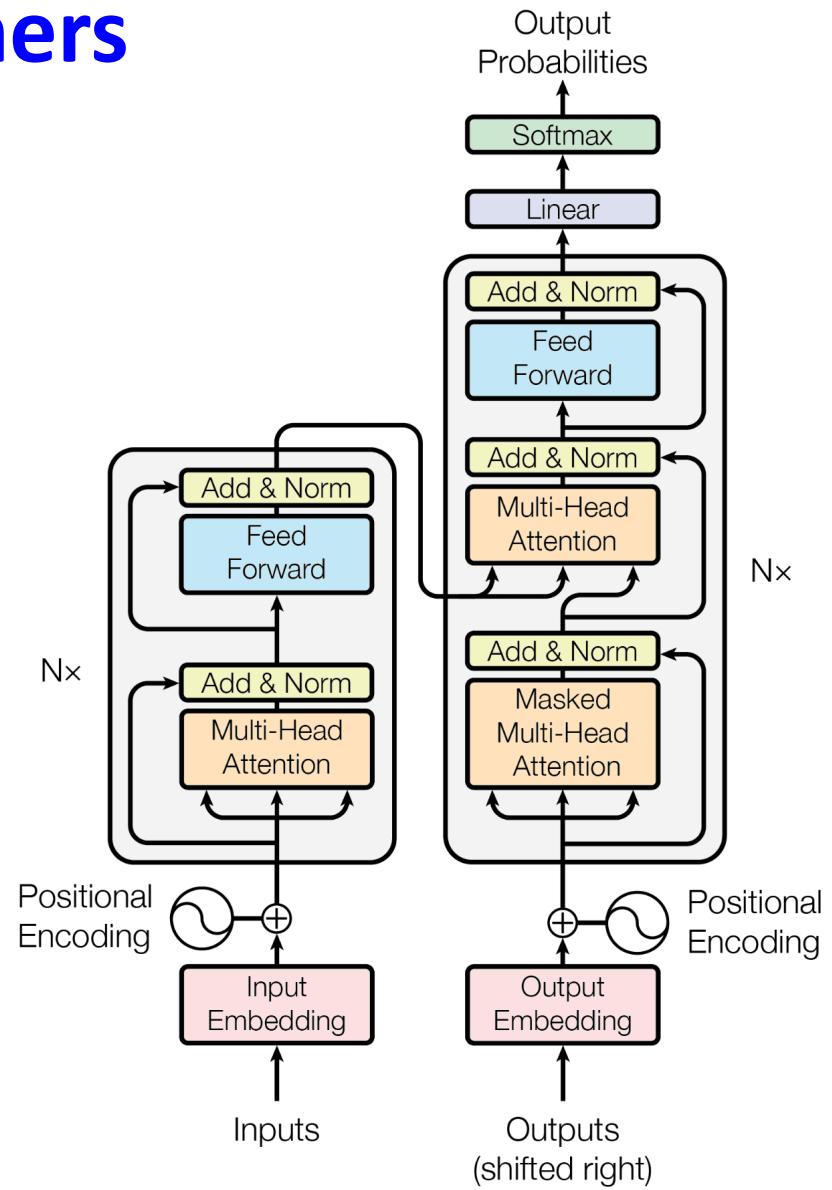
Part 1

Transformers



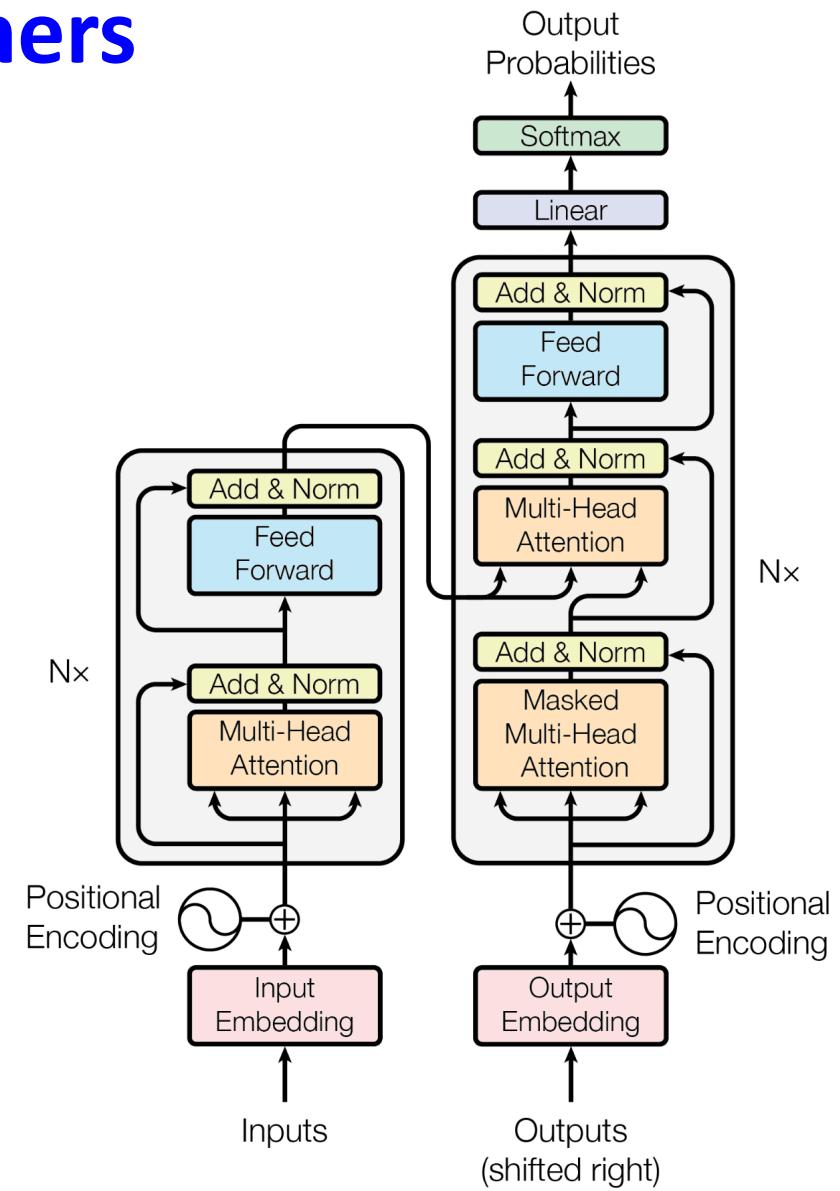
Transformers

- Tokenization
- Input Embeddings
- Position Encodings
- Residuals
- Query
- Key
- Value
- Add & Norm
- Encoder
- Decoder
- Attention
- Self Attention
- Multi Head Attention
- Masked Attention
- Encoder Decoder Attention
- Output Probabilities / Logits
- Softmax
- Encoder-Decoder models
- Decoder only models



Transformers

- Tokenization
 - Input Embeddings
 - Position Encodings
 - Residuals
 - Query
 - Key
 - Value
 - Add & Norm
 - Encoder
 - Decoder
-
- Attention
 - Self Attention
 - Multi Head Attention
 - Masked Attention
 - Encoder Attention
 - Decoder Probabilities / Logits



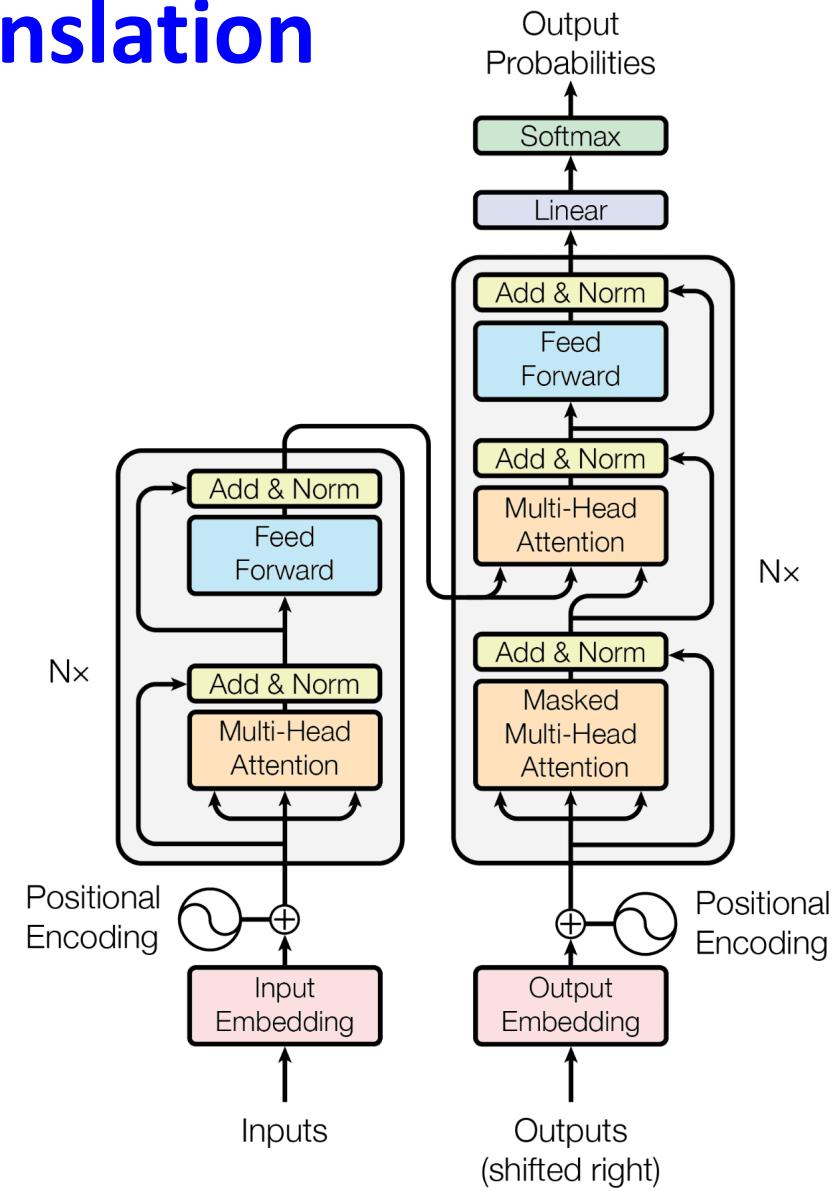
Machine Translation

Targets

Ich have einen apfel gegessen

Inputs

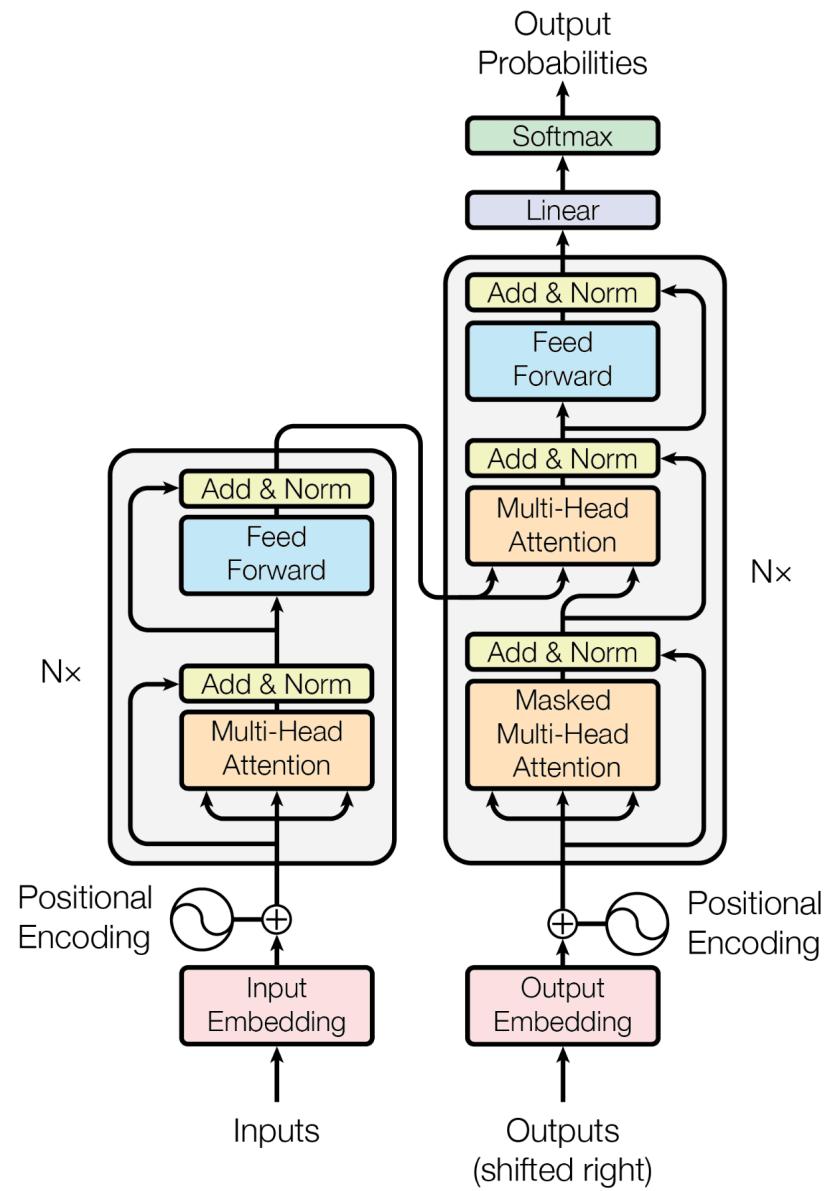
I ate an apple



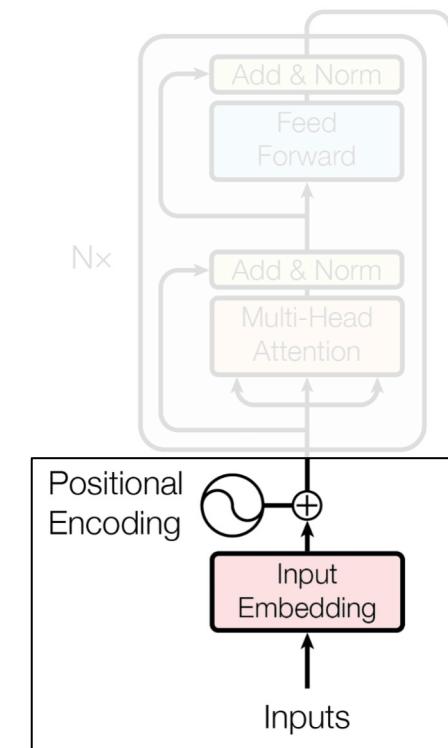
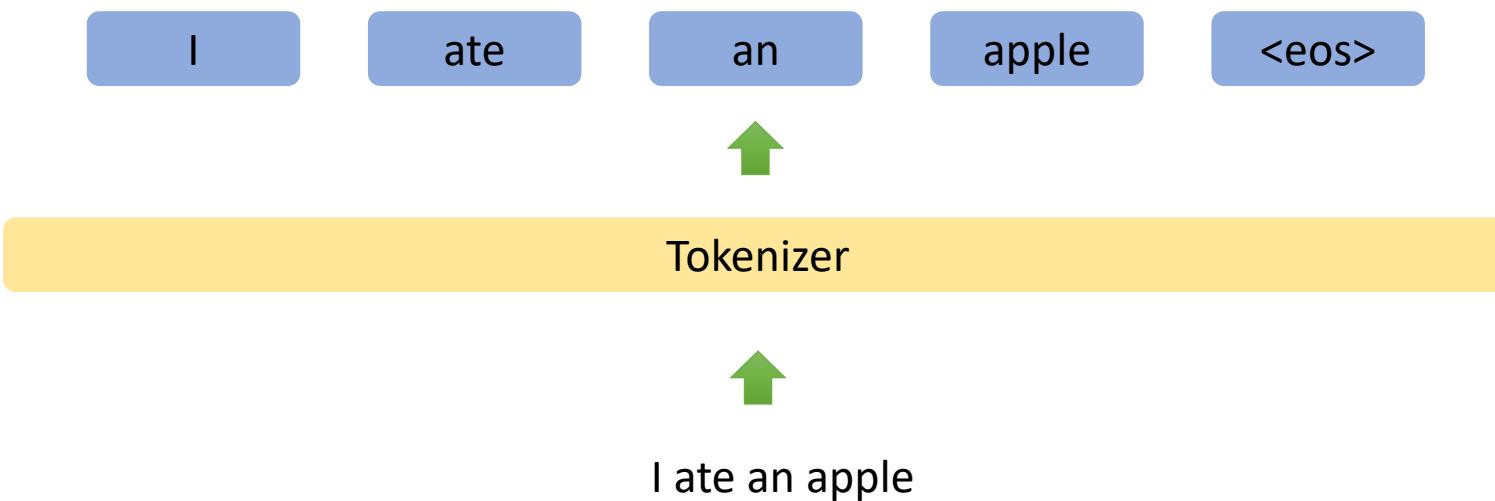
Inputs

Processing Inputs

Inputs
I ate an apple

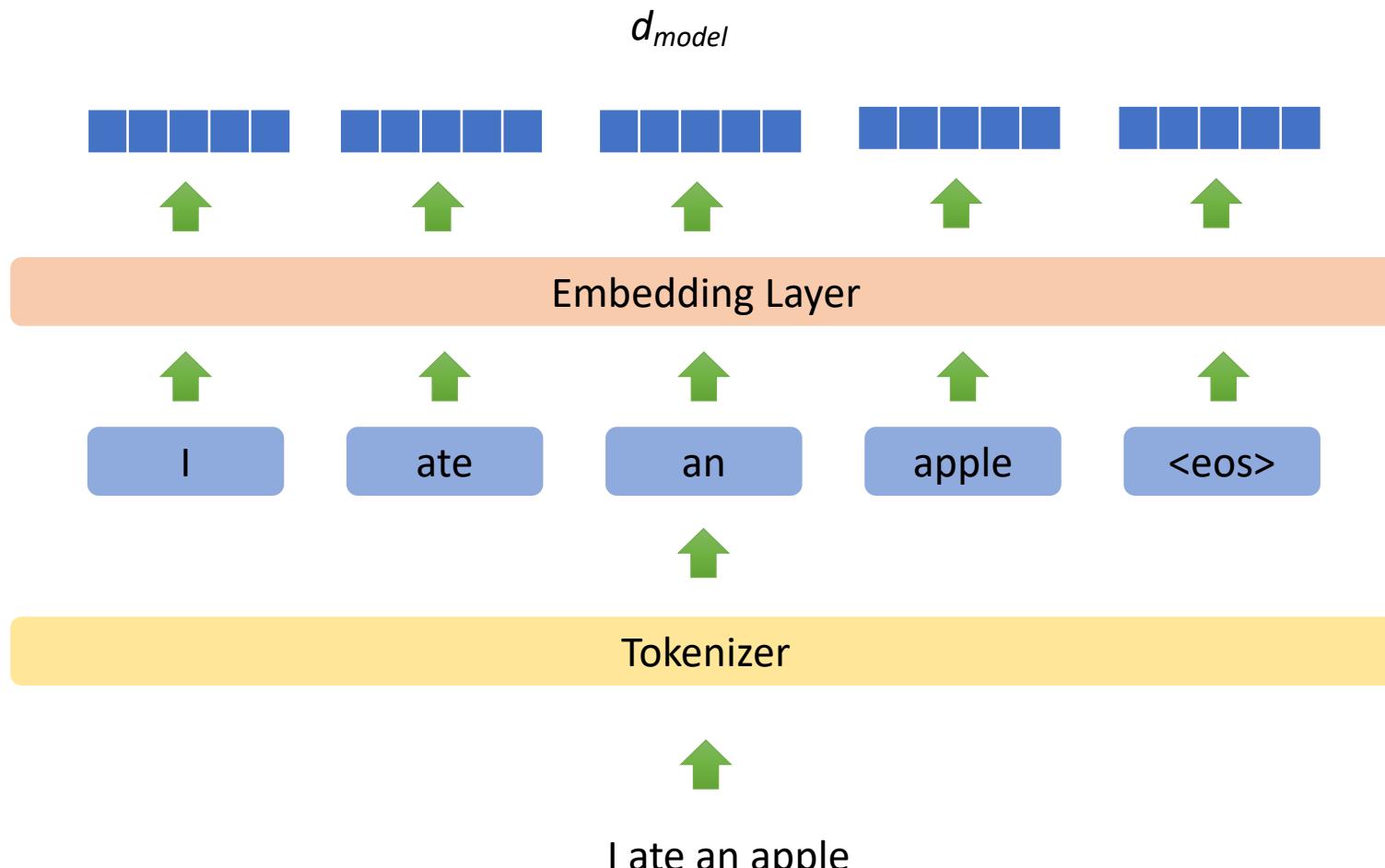


Inputs

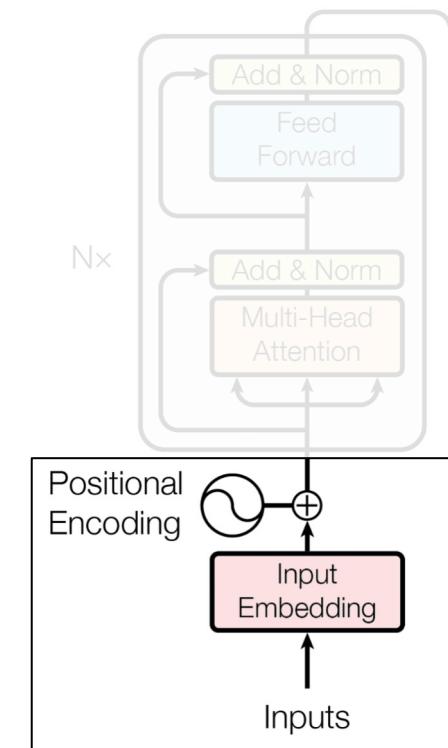


Generate Input Emebeddings

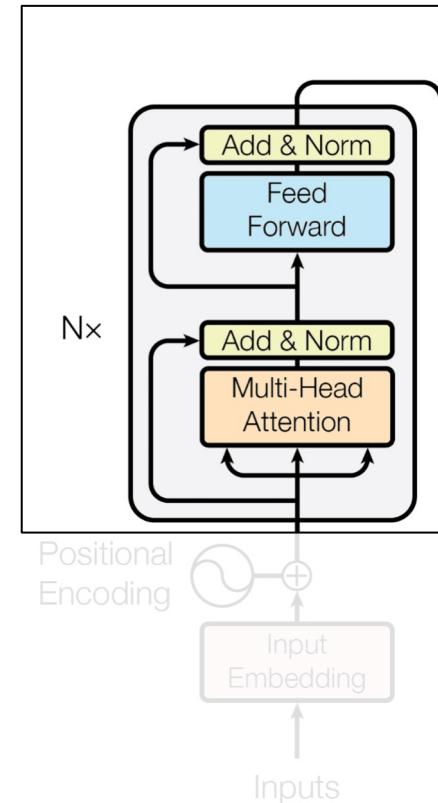
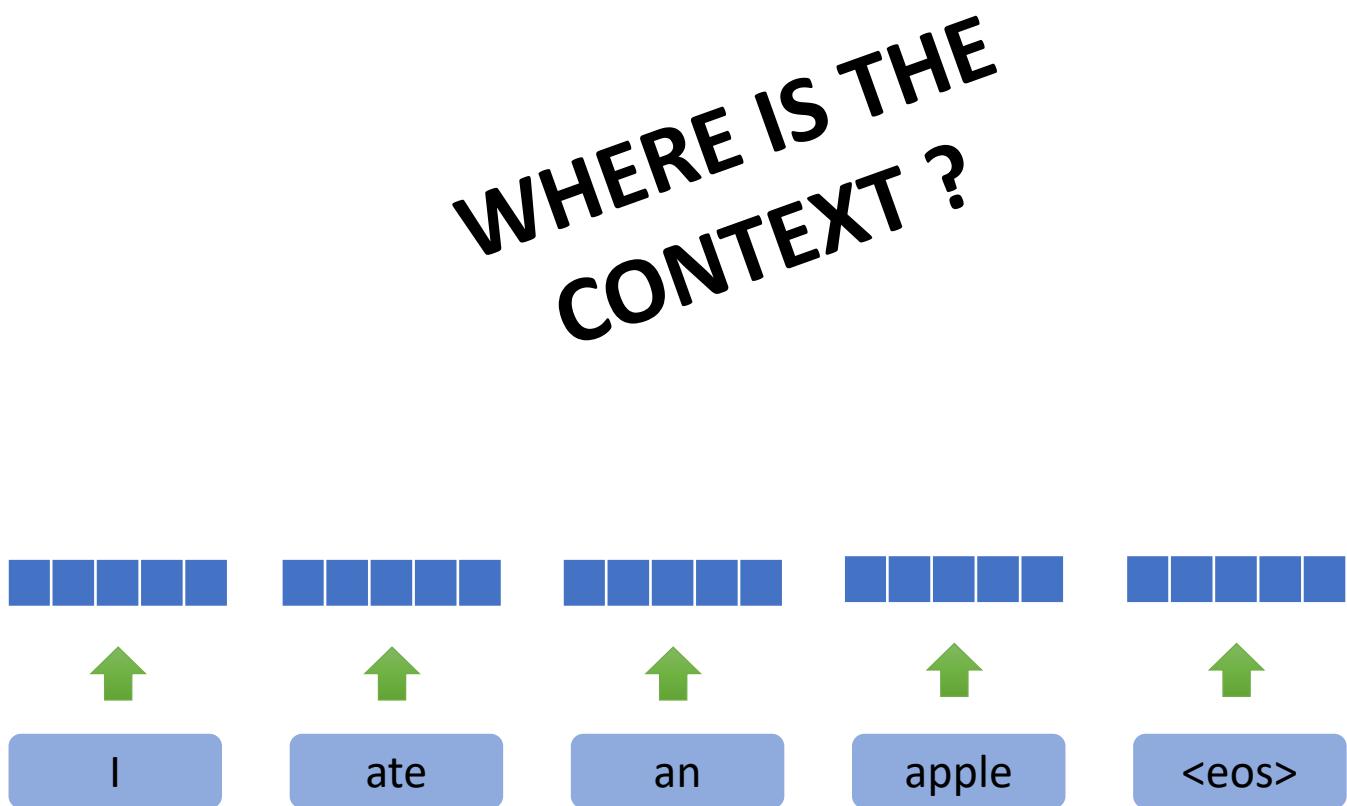
Inputs



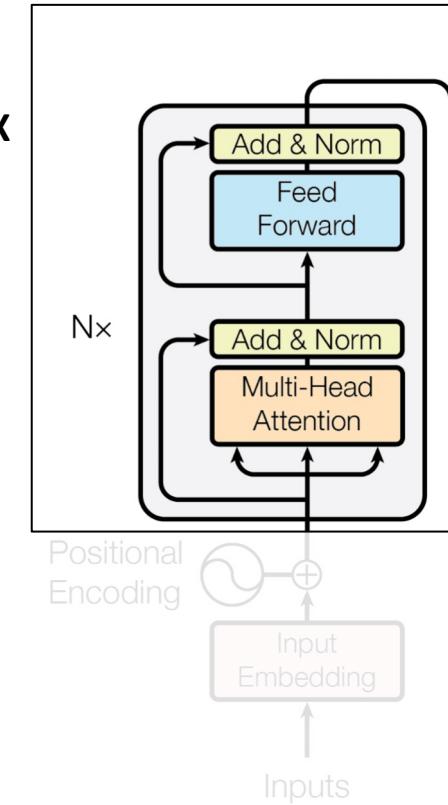
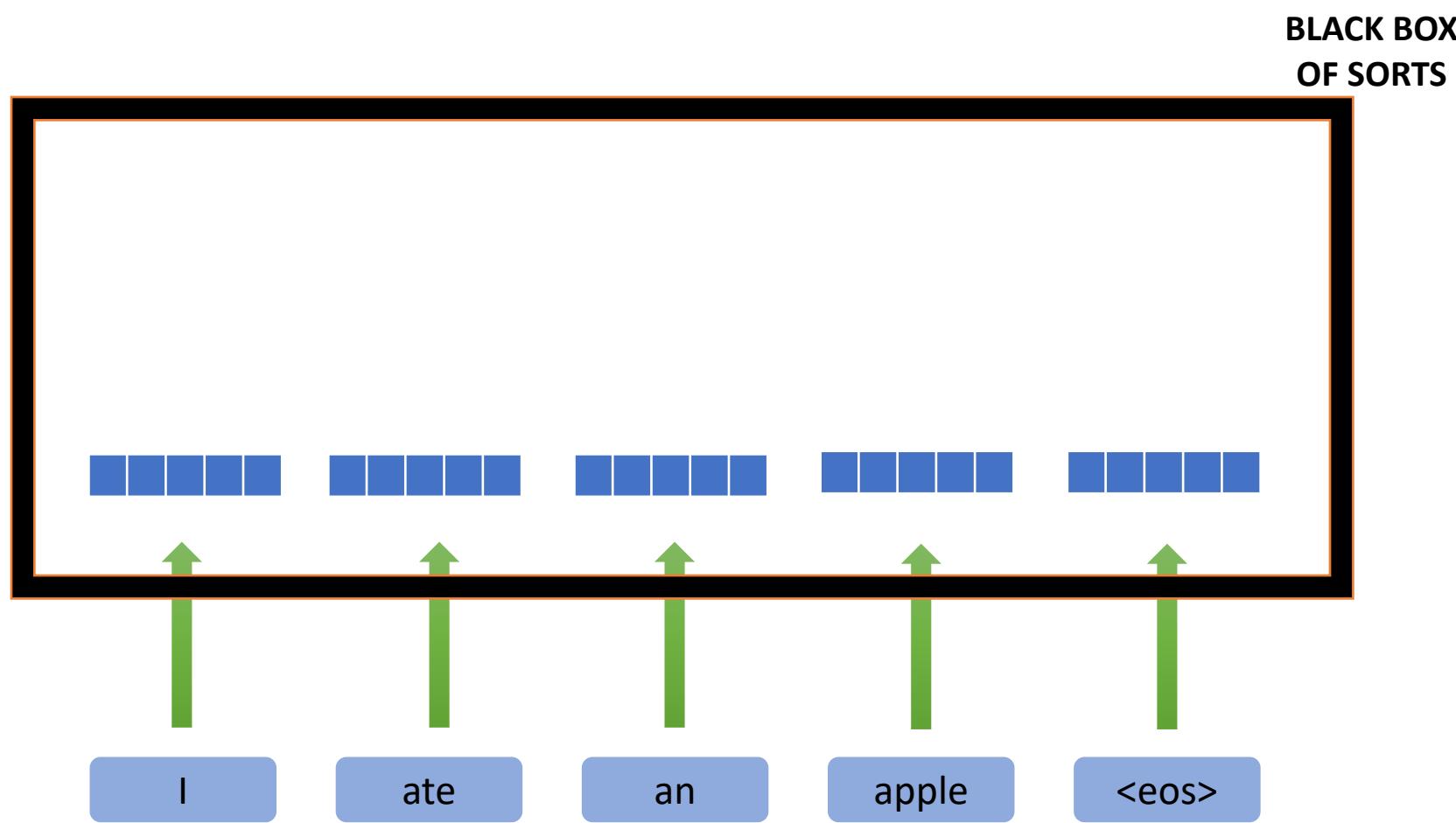
Generate Input Embeedings



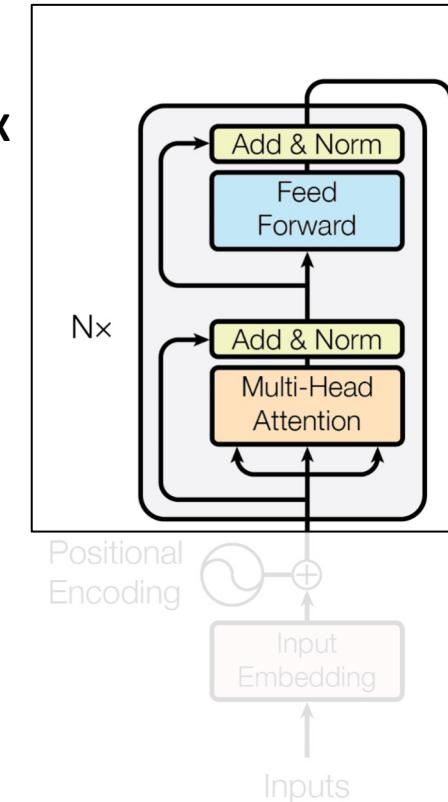
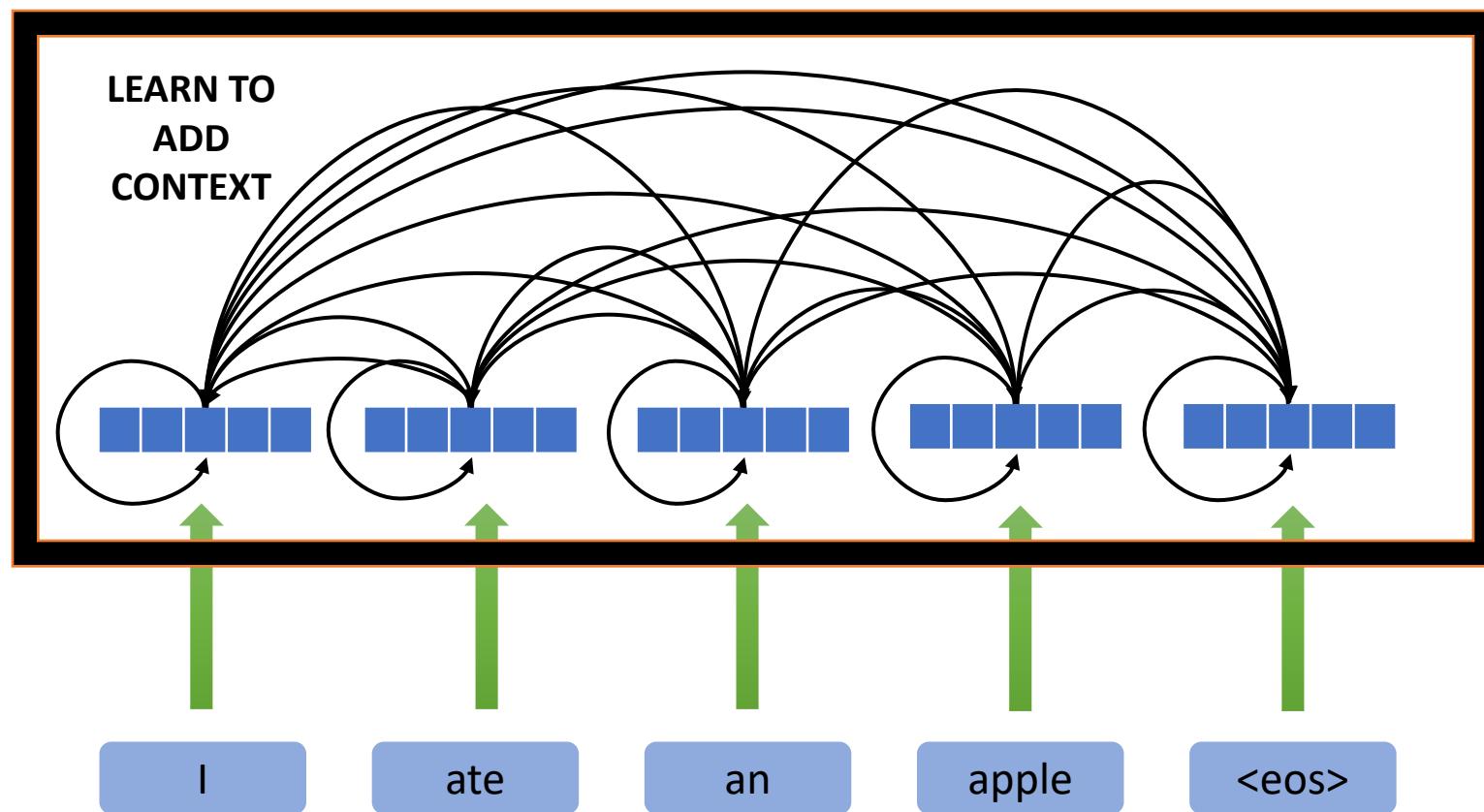
Encoder



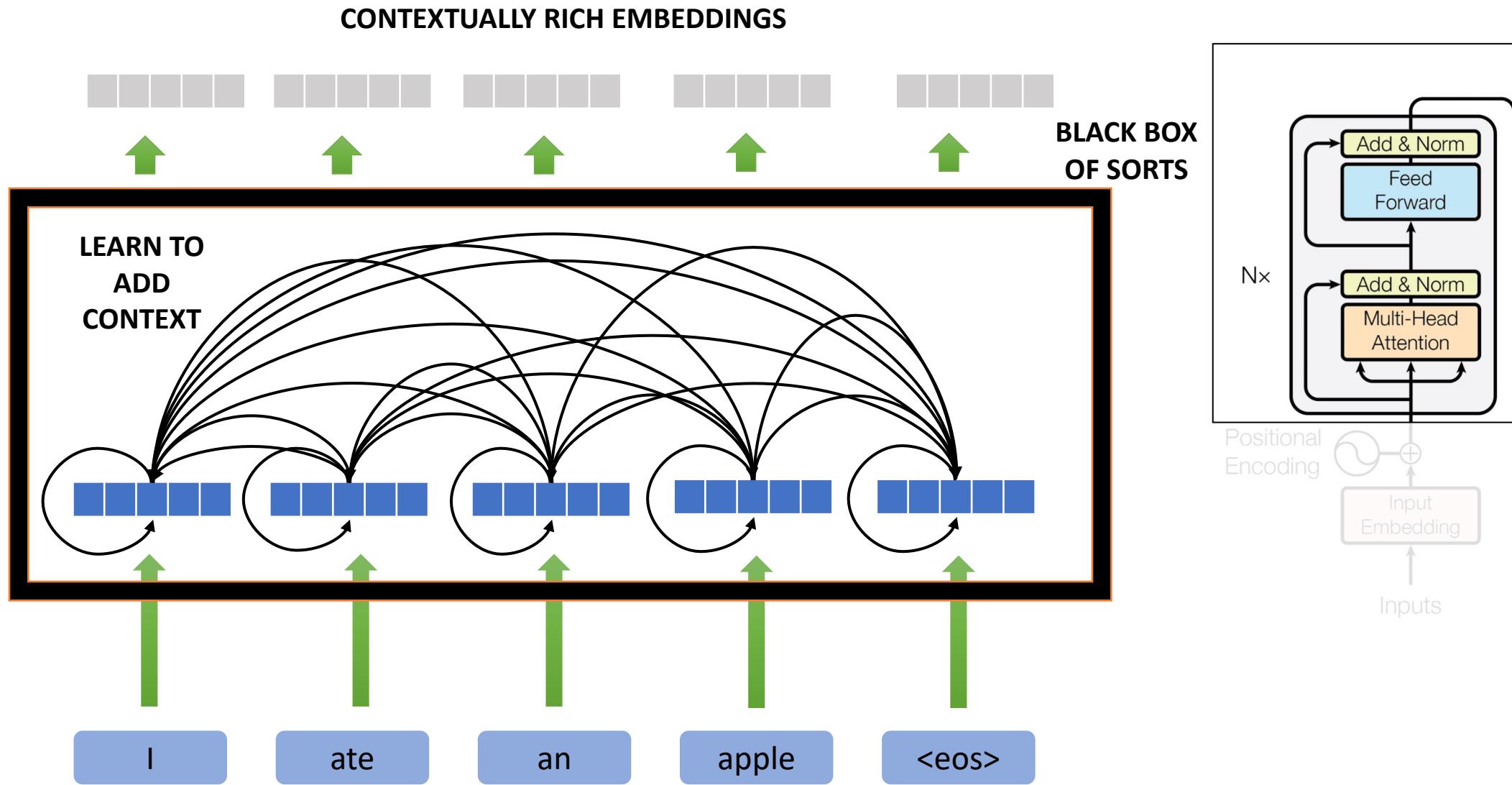
Encoder



Encoder



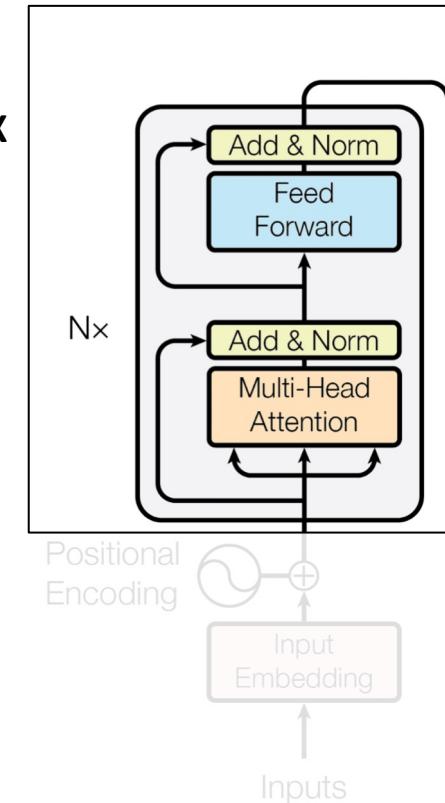
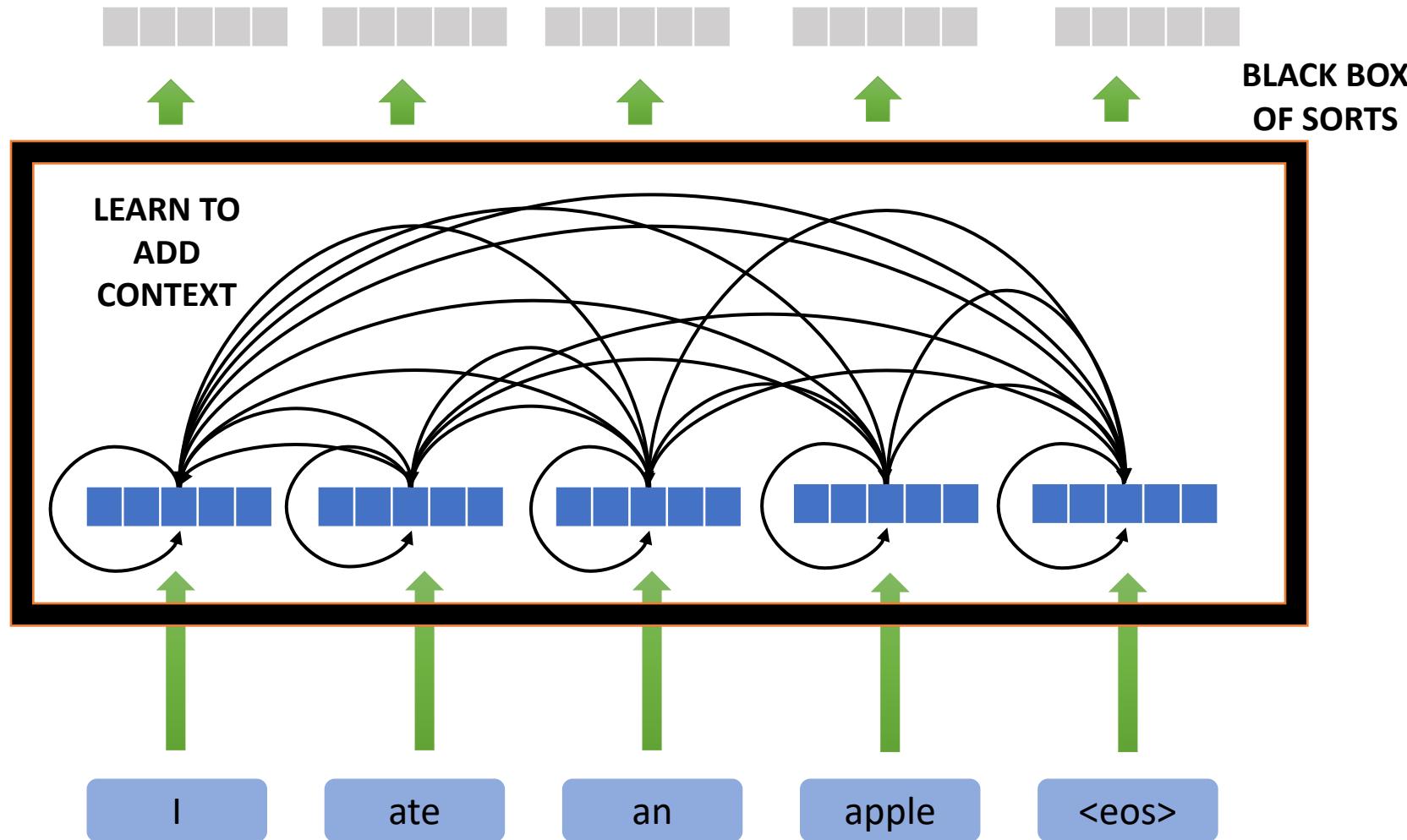
Encoder



Encoder

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

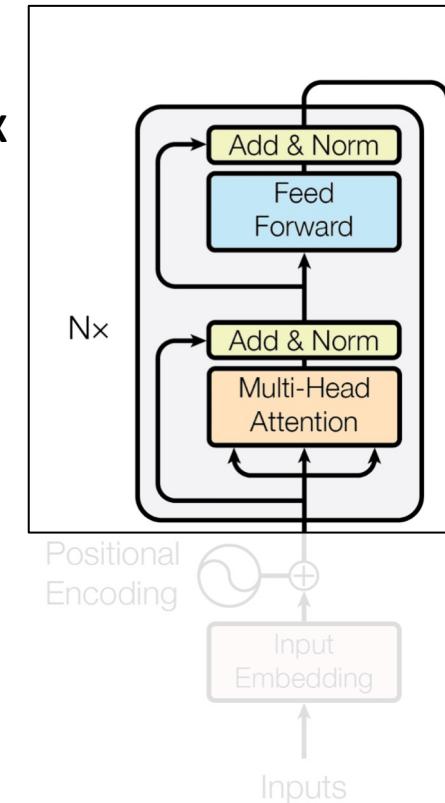
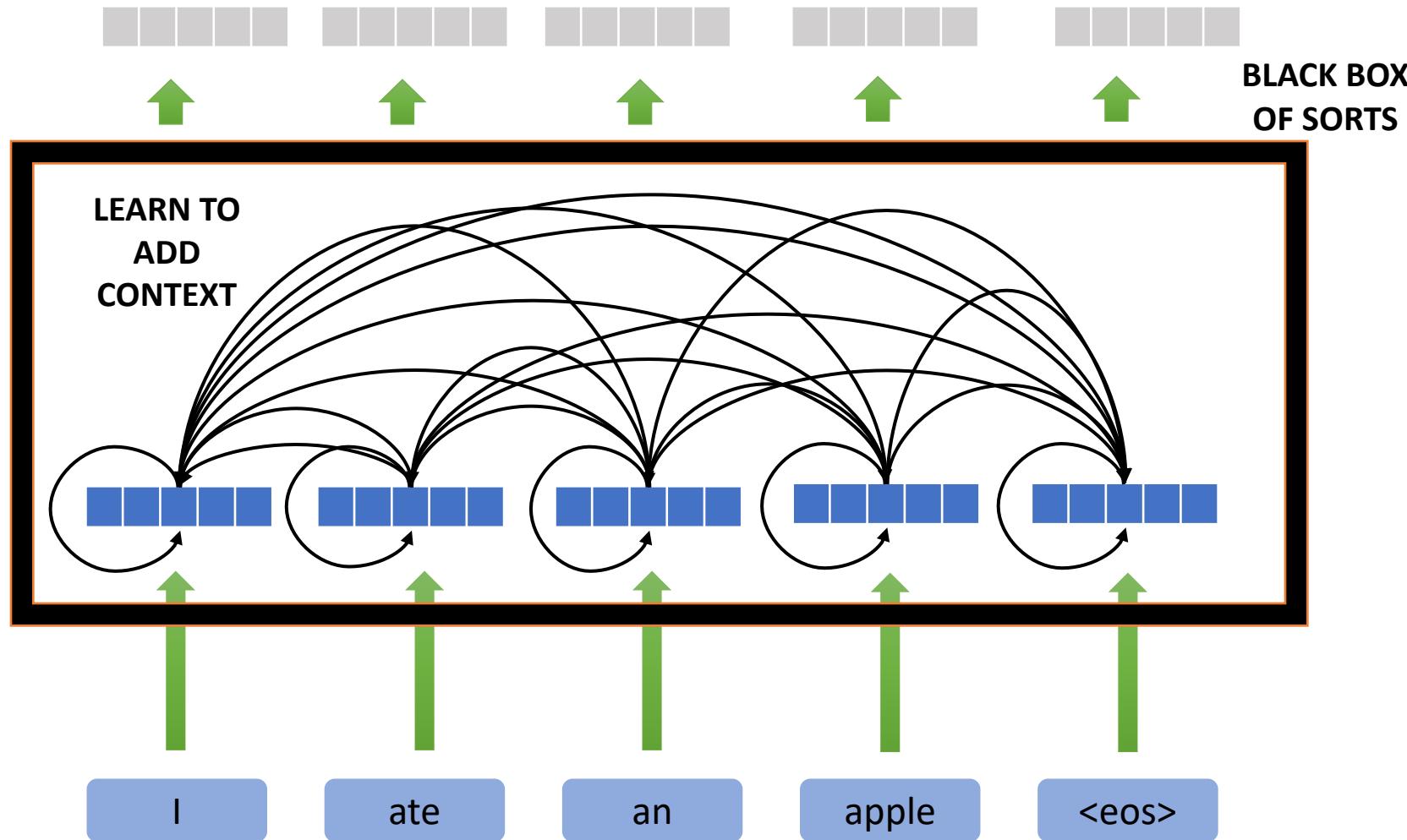
CONTEXTUALLY RICH EMBEDDINGS



Encoder

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

CONTEXTUALLY RICH EMBEDDINGS

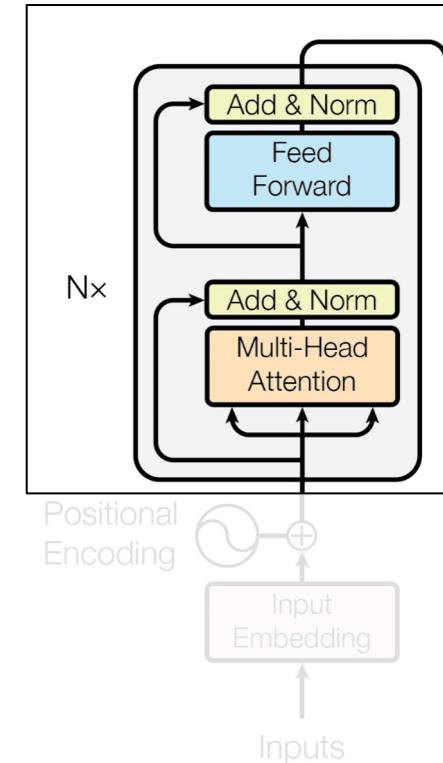


Attention

$\alpha_{[ij]}$?

From lecture 18:

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



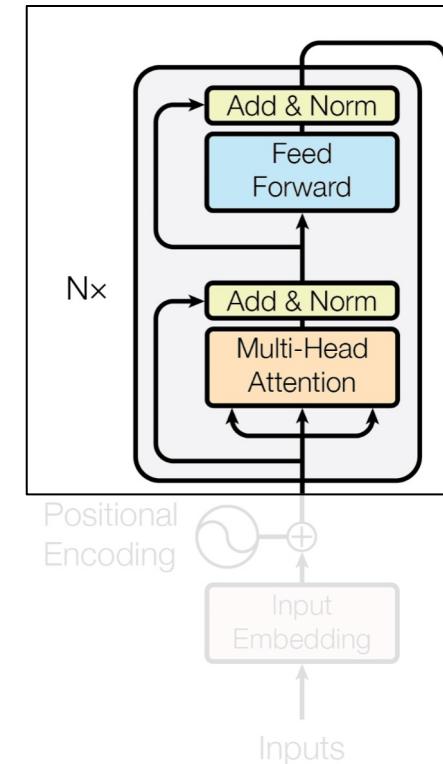
Attention

$\alpha_{[ij]}$?

From lecture 18:

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

- 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

```
{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

{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", ...}}
```

Query, Key & Value

{Query: "Order details of order_104"}
OR
{Query: "Order details of order_106"}

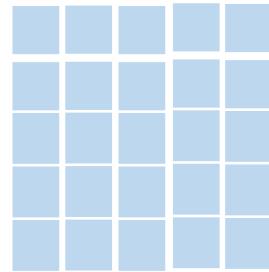
```
{"order_100": {"items": "a1", "delivery_date": "a2", ...},  
 "order_101": {"items": "b1", "delivery_date": "b2", ...},  
 "order_102": {"items": "c1", "delivery_date": "c2", ...},  
 "order_103": {"items": "d1", "delivery_date": "d2", ...},  
 "order_104": {"items": "e1", "delivery_date": "e2", ...},  
 "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	Key	Value
1. Search for info	1. Interacts directly with Queries 2. Distinguishes one object from another 3. Identify which object is the most relevant and by how much	1. Actual details of the object 2. More fine grained

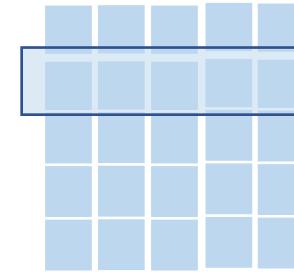
Attention



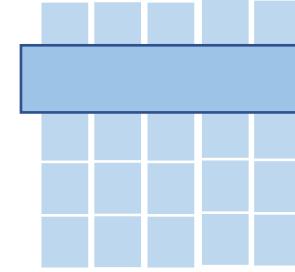
Query



Key Value Store

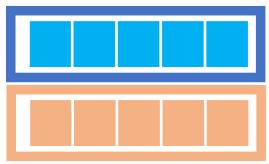


Key

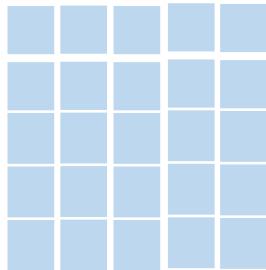


Value

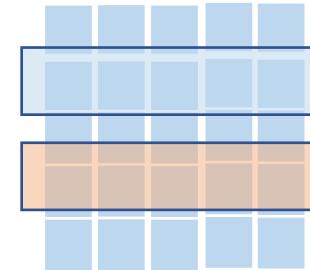
Attention



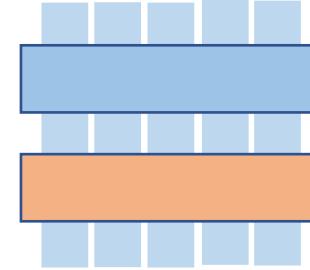
Query



Key Value Store



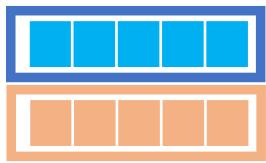
Key



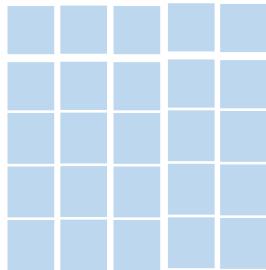
Value

Attention

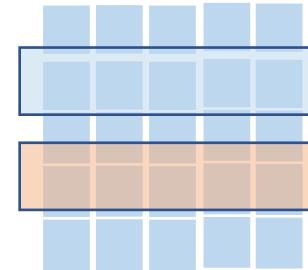
Done at the same time !!



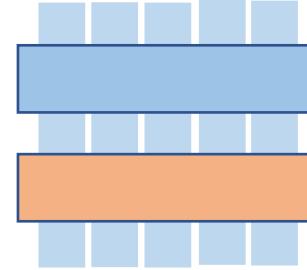
Query



Key Value Store



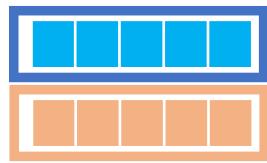
Key



Value

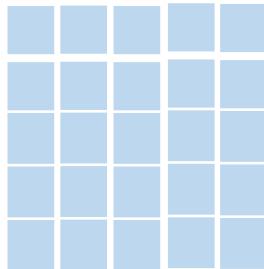
Attention

Parallelizable !!!



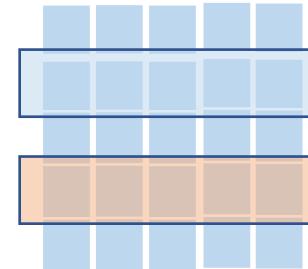
Query

$$Q$$



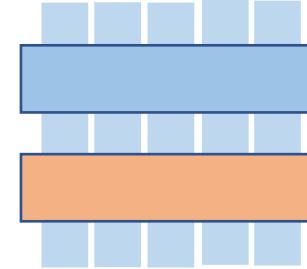
Key Value Store

$$QK^T$$



Key

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

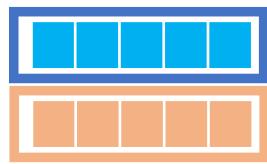


Value

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

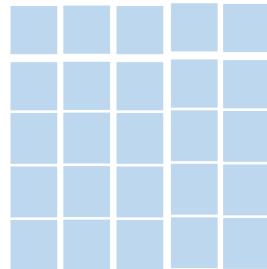
Attention

Parallelizable !!!



Query

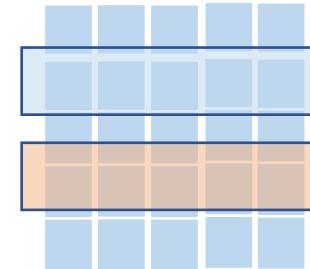
$$Q$$



Key Value Store

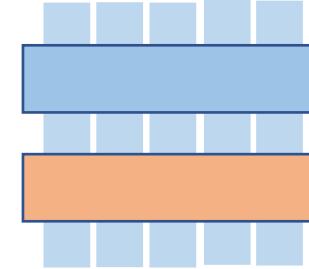
$$QK^T$$

Attention Filter



Key

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



Value

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

Attention



I_1

I



I_2

ate



I_3

an



I_4

apple

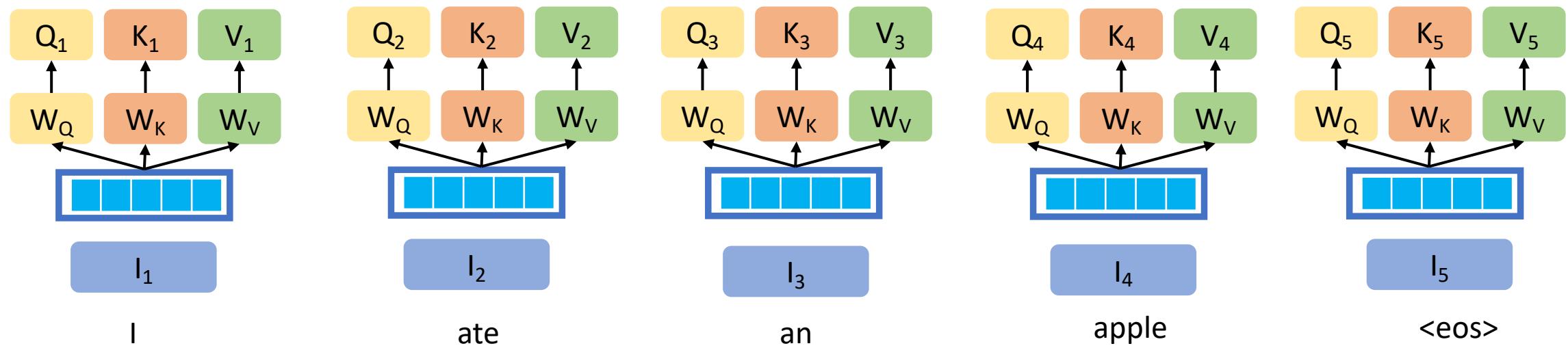


I_5

<eos>

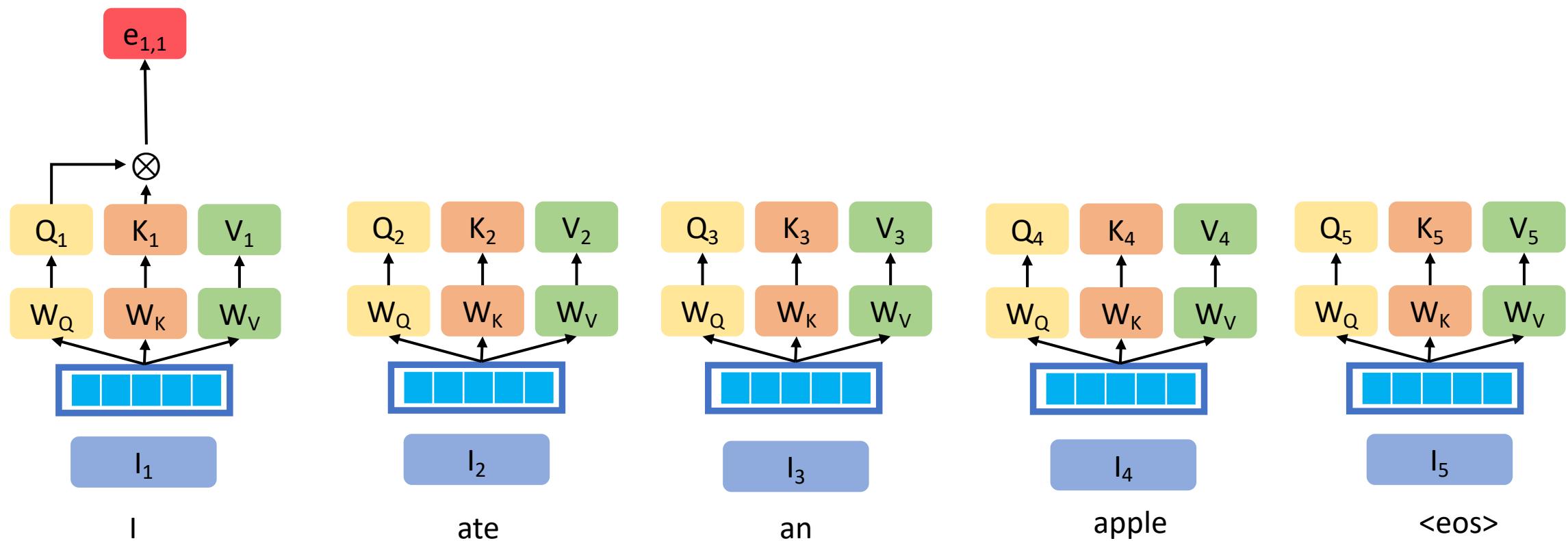
Dimensions across QKV have been dropped for brevity

Attention



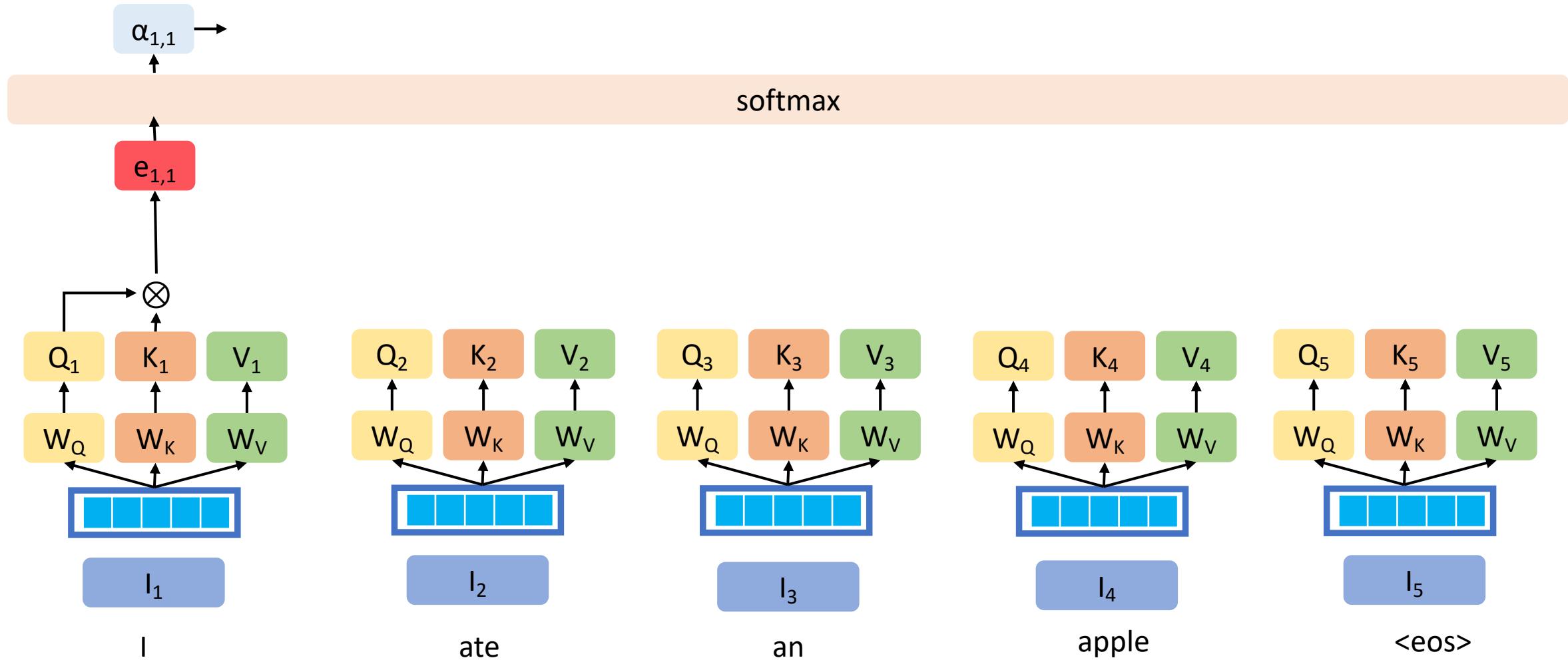
Attention

Dimensions across QKV have been dropped for brevity

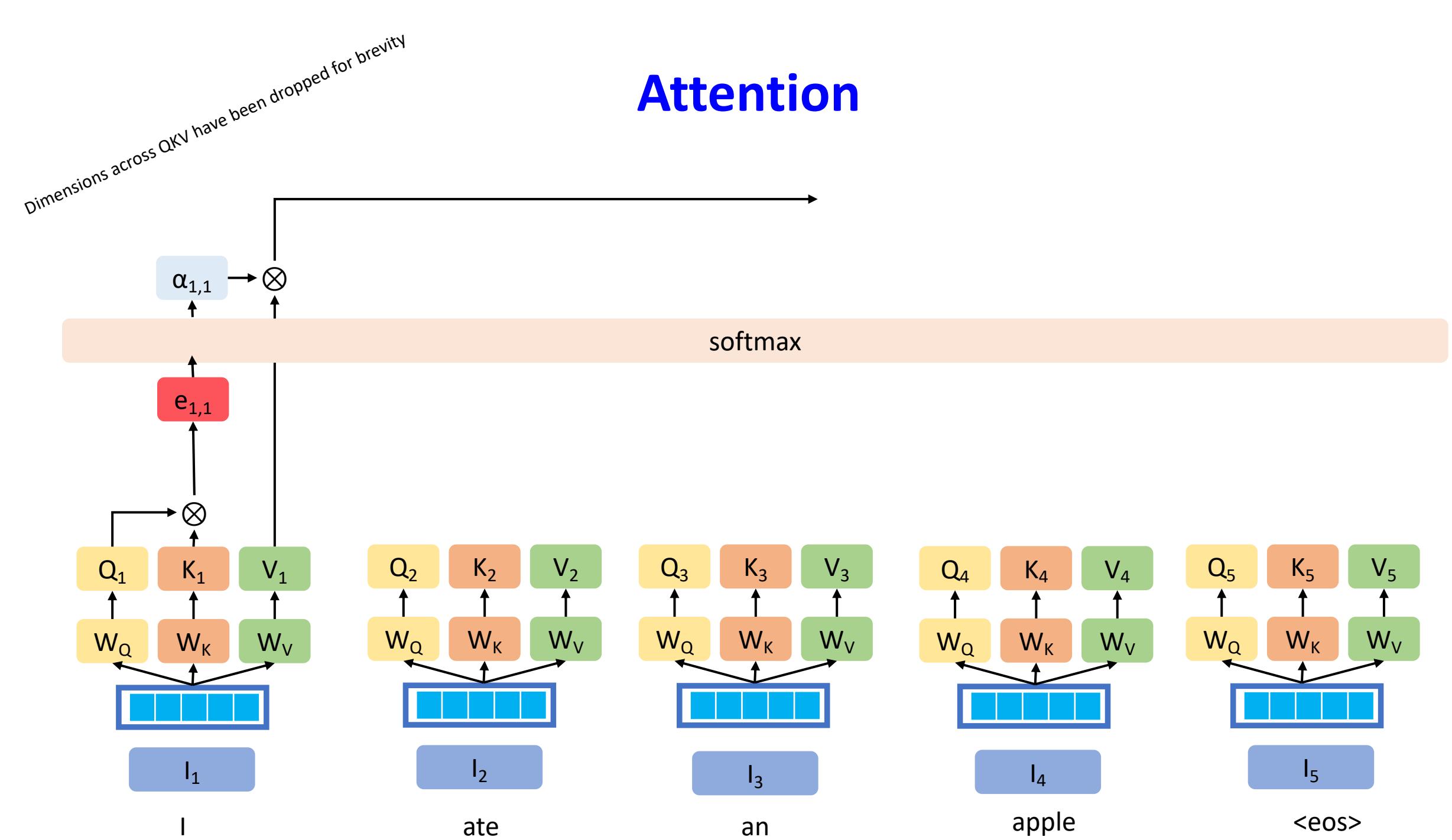


Attention

Dimensions across QKV have been dropped for brevity

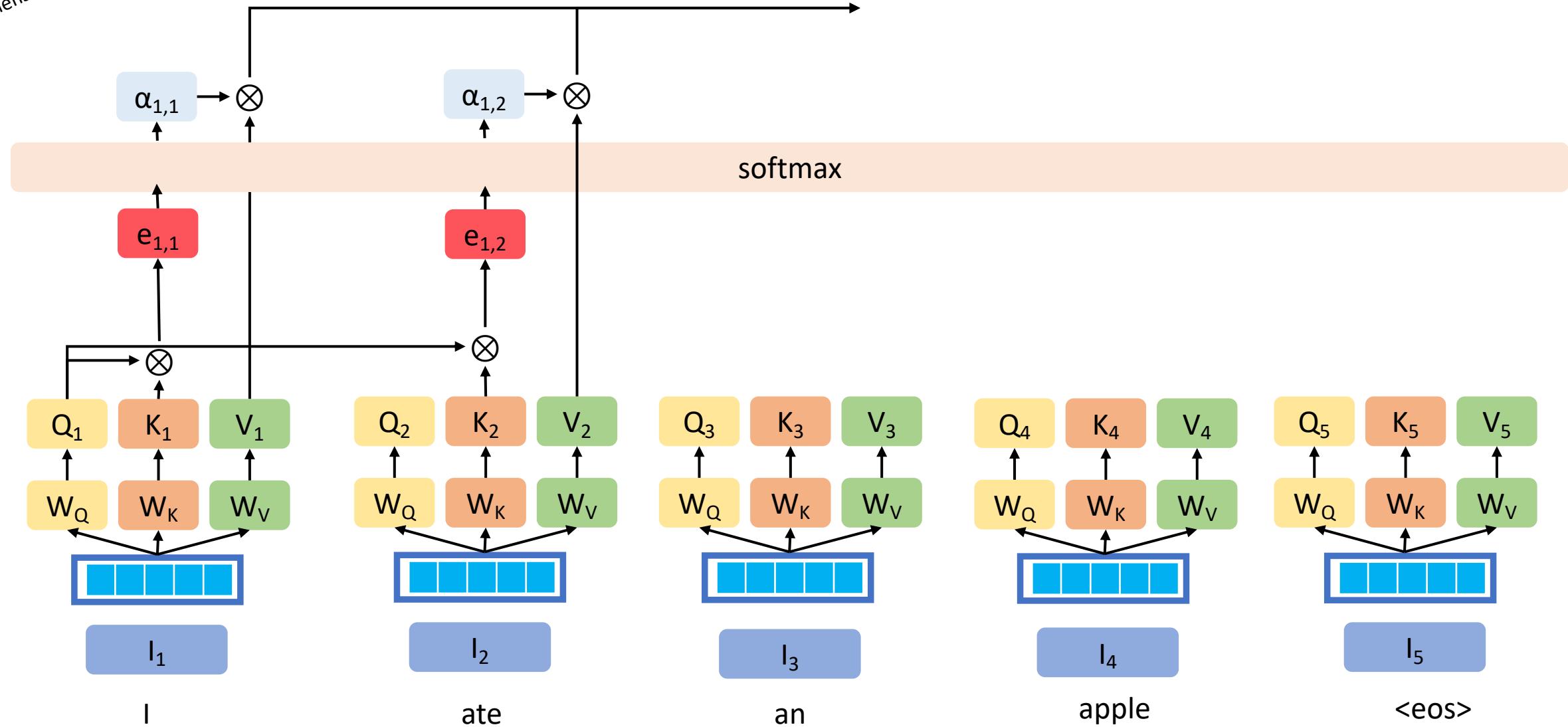


Attention

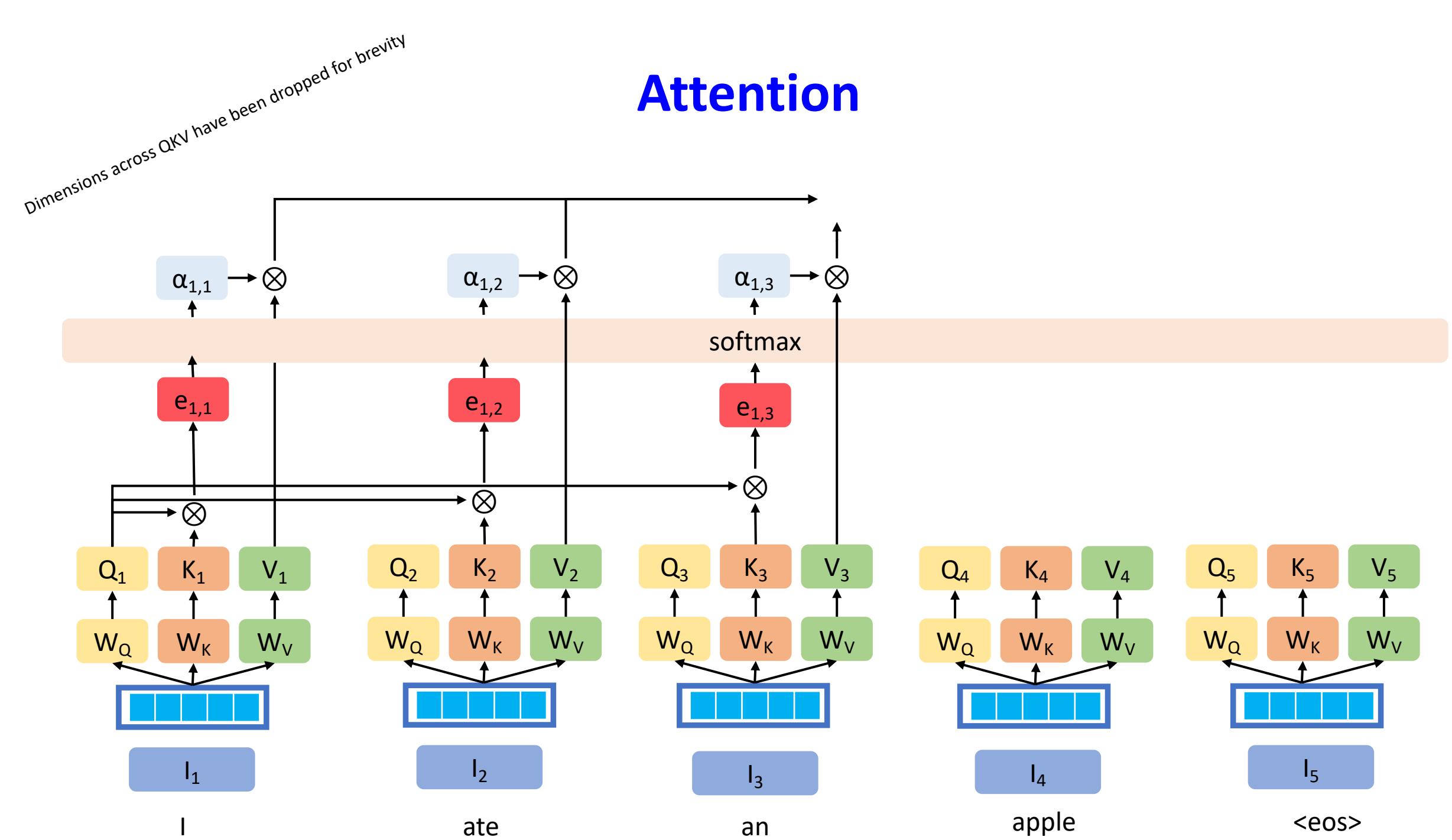


Attention

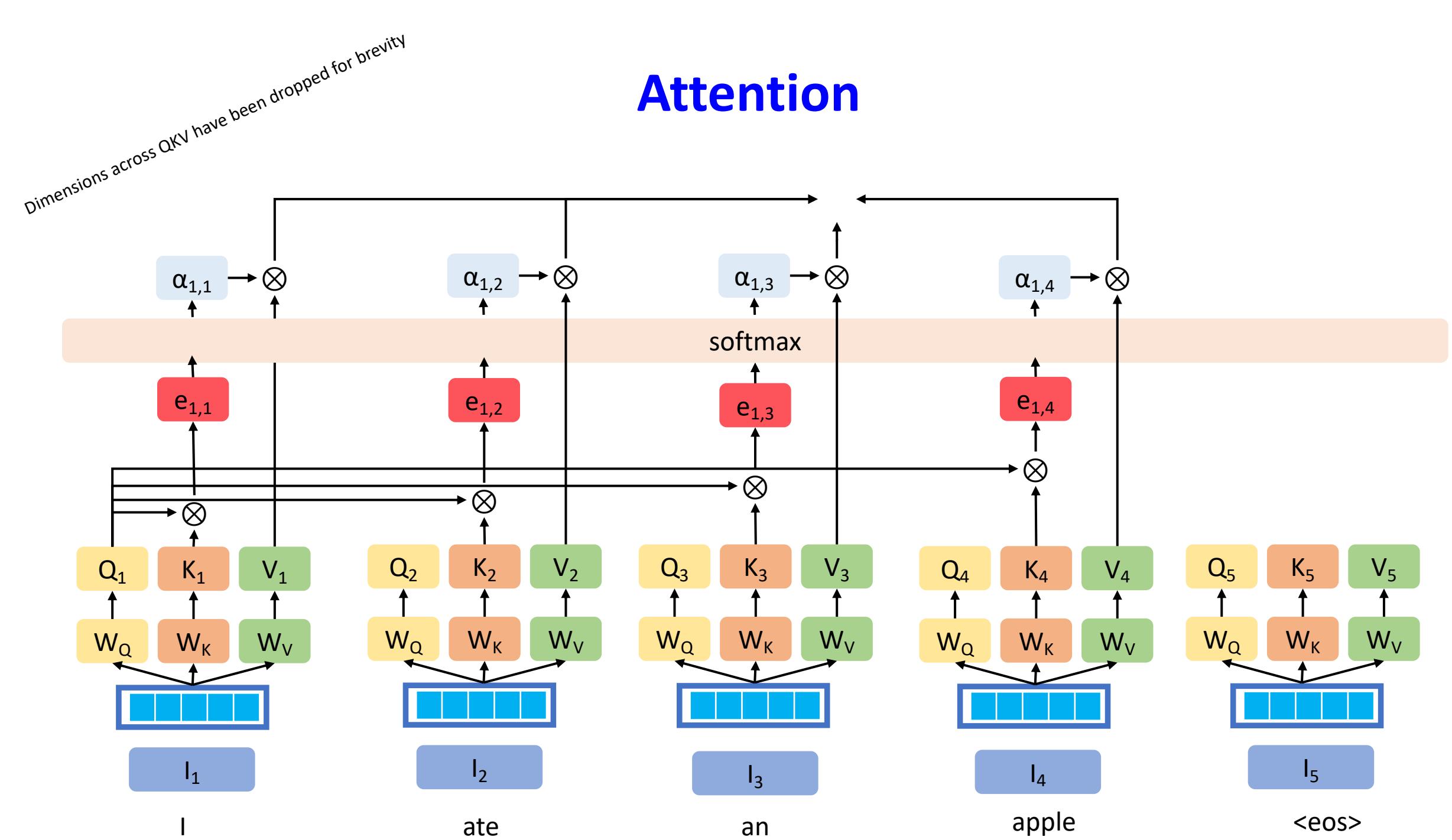
Dimensions across QKV have been dropped for brevity



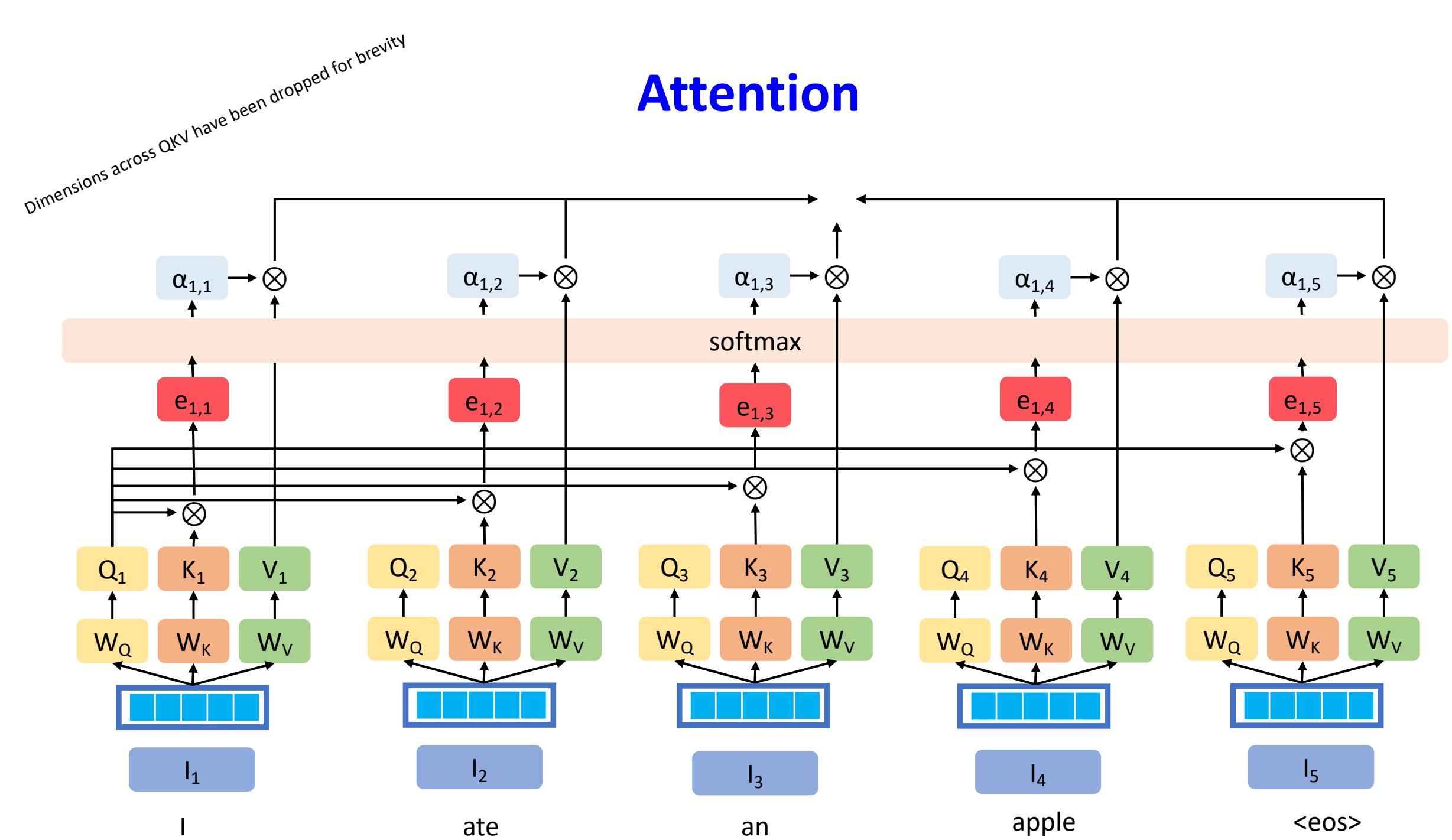
Attention



Attention



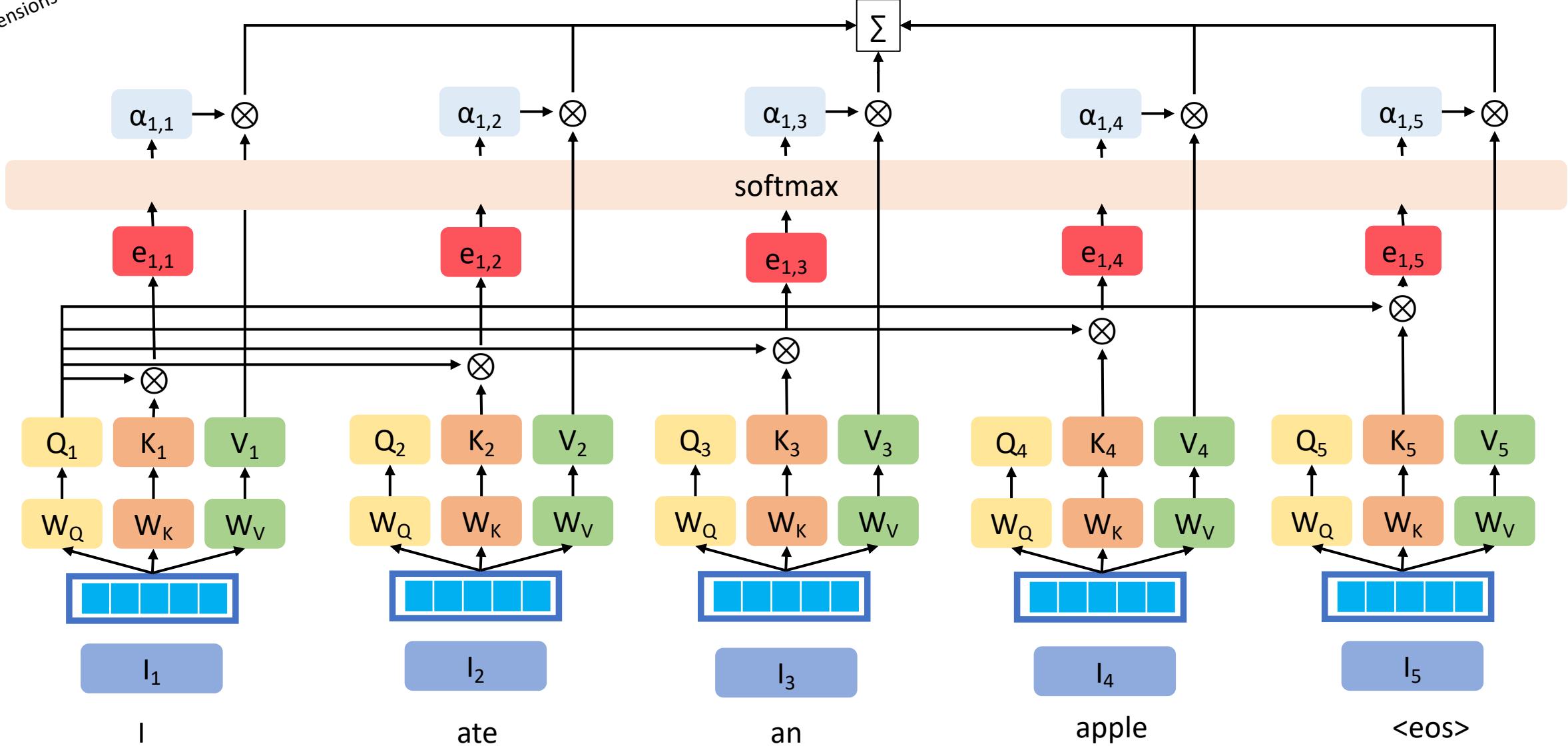
Attention



Contextually
rich
embedding

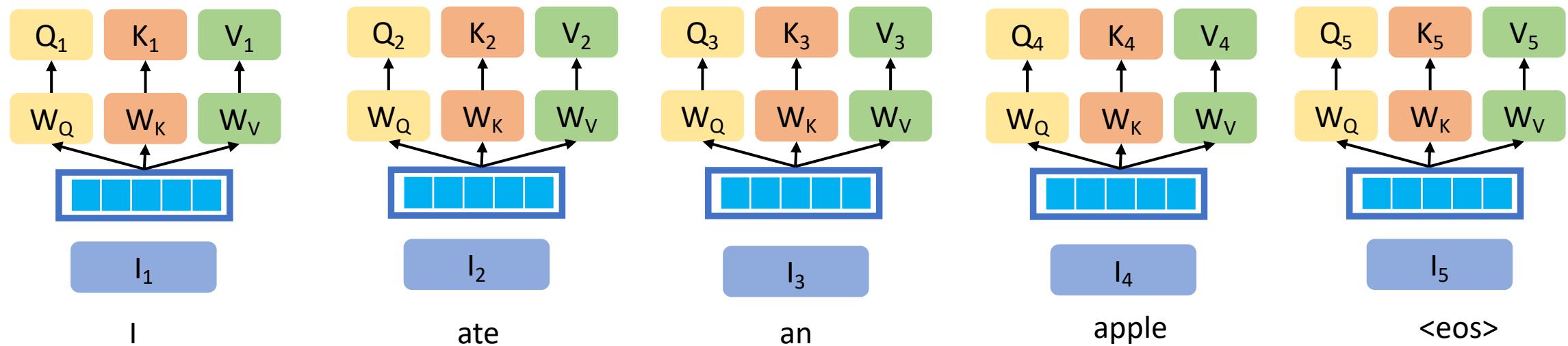
Attention

Dimensions across QKV have been dropped for brevity



Dimensions across QKV have been dropped for brevity

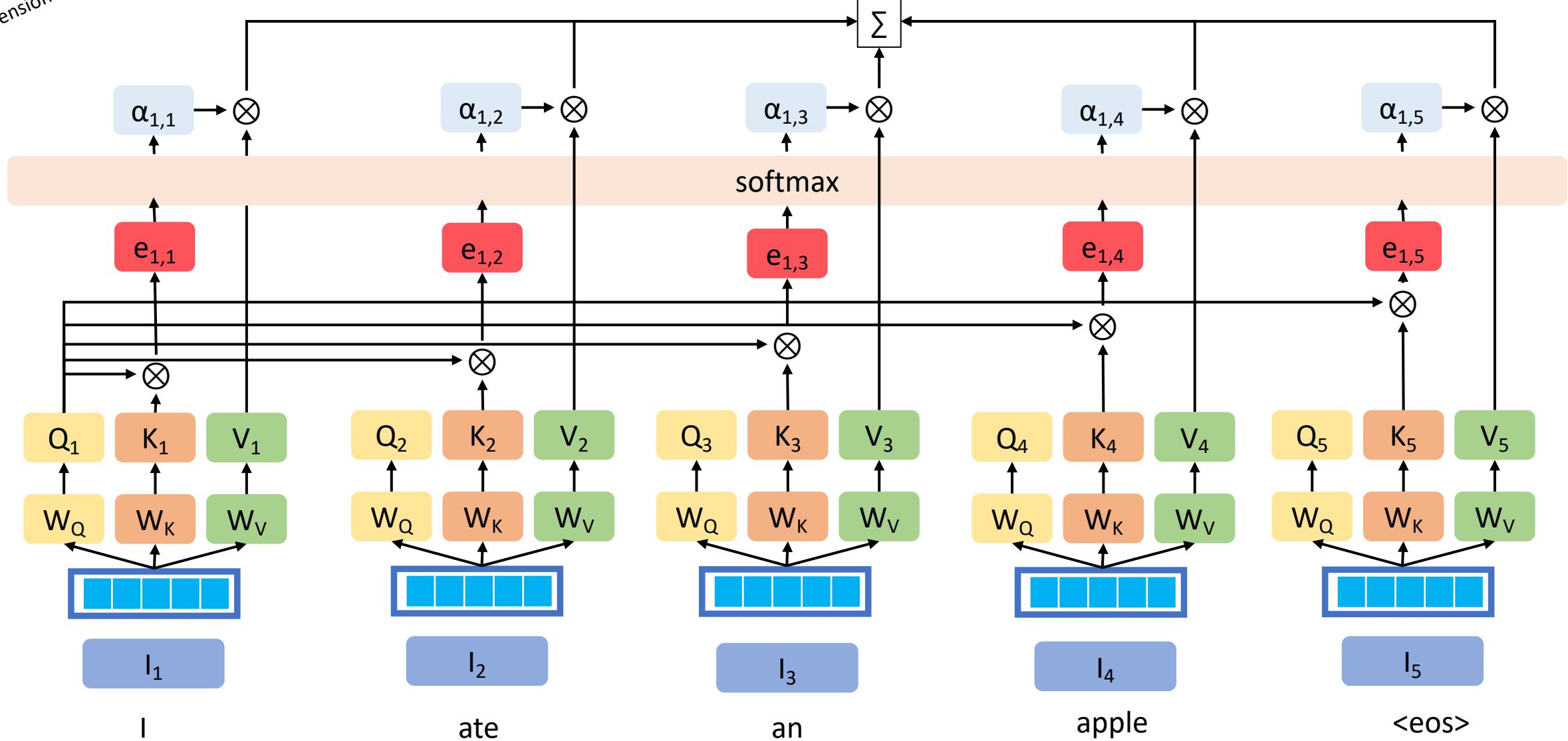
Attention



Contextually
rich
embedding

Attention

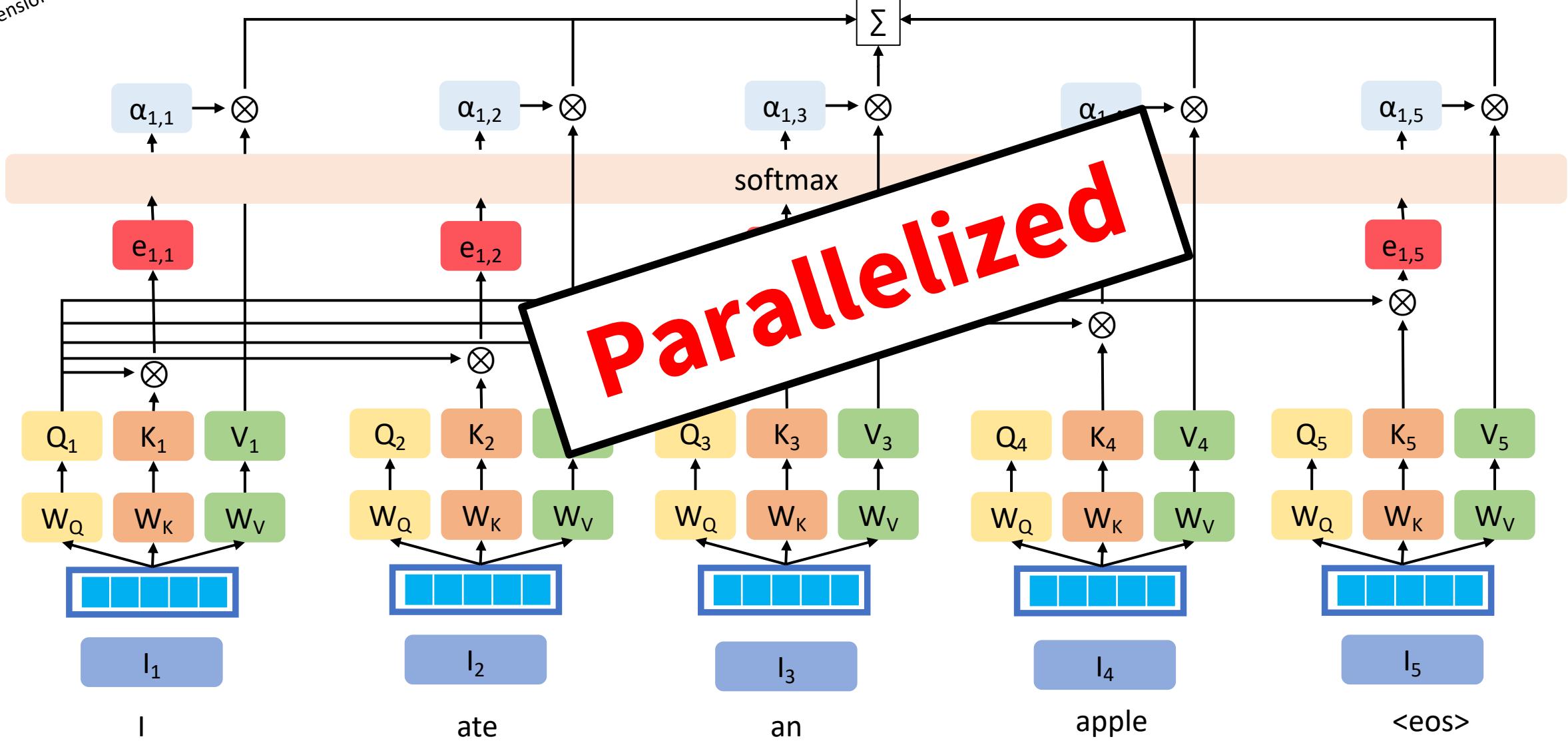
Dimensions across QKV have been dropped for brevity



Contextually
rich
embedding

Attention

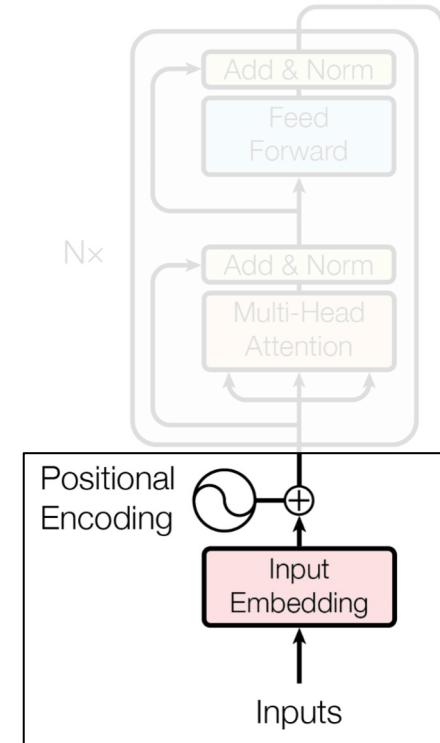
Dimensions across QKV have been dropped for brevity



Poll 1 @1296

Which of the following are true about attention? (Select all that apply)

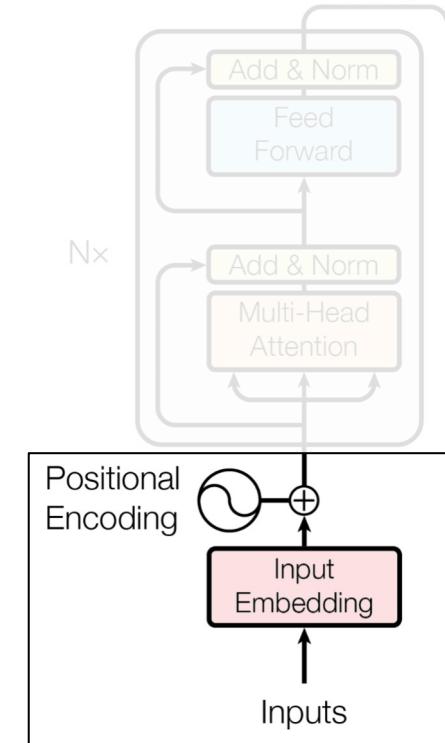
- a. To calculate attention weights for input I_2 , you would use key k_2 , and all queries
- b. To calculate attention weights for input I_2 , you would use query q_2 , and all keys
- c. We scale the QK^T product to bring attention weights in the range of $[0,1]$
- d. We scale the QK^T product to allow for numerical stability



Poll 1 @1296

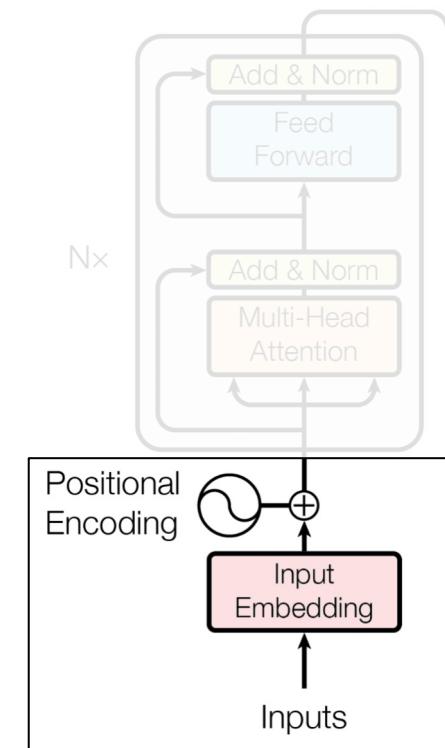
Which of the following are true about attention? (Select all that apply)

- a. To calculate attention weights for input I_2 , you would use key 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**



Positional Encoding

I ate an apple <eos>



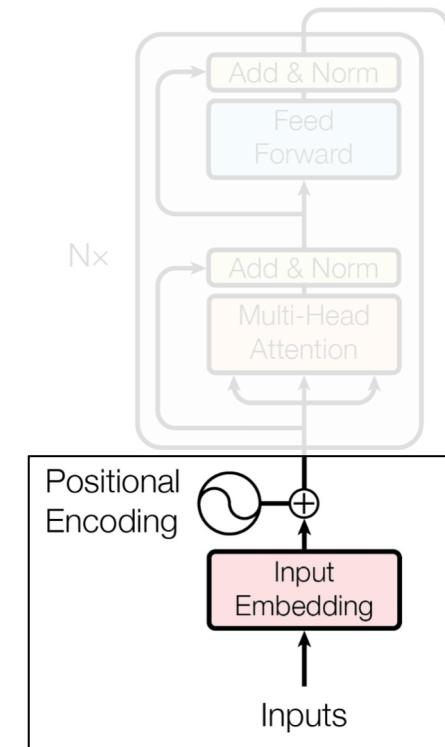
Positional Encoding

I ate an apple <eos>



apple ate an I <eos>

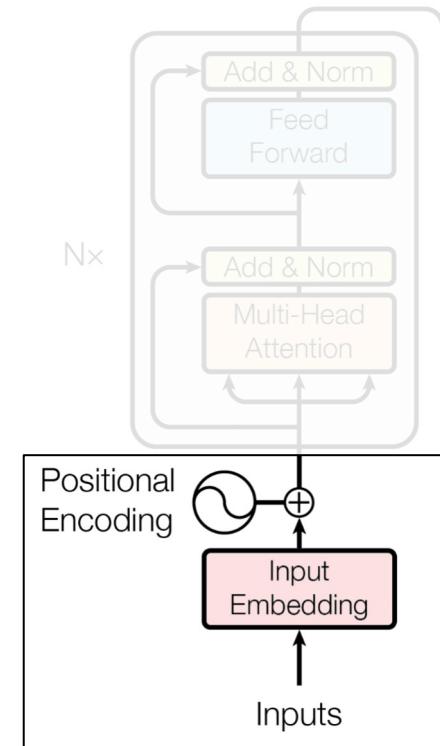
Positional Encoding



Positional Encoding

Requirements for Positional Encodings

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



Positional Encoding

Positional Encoding

Requirements for Positional Encodings

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

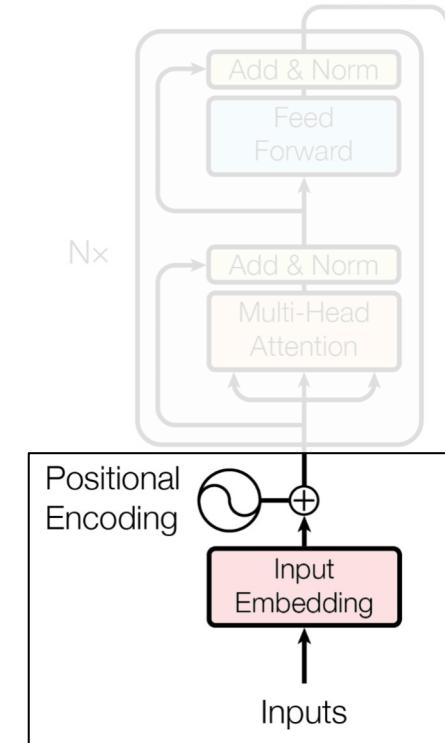
Possible Candidates :

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

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

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

Positional Encoding



Positional Encoding

Requirements for Positional Encodings

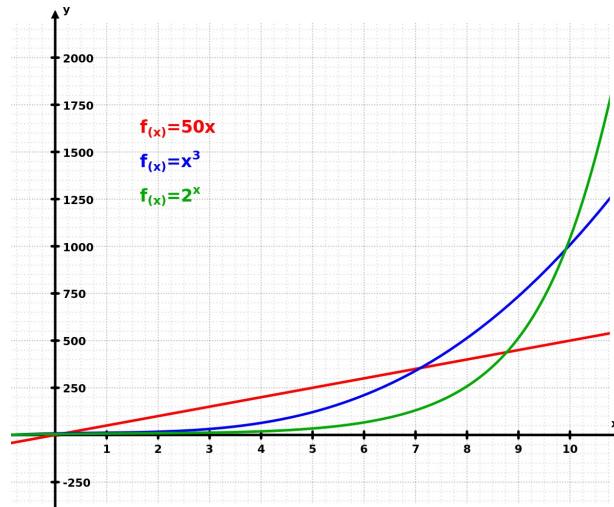
- Some representation of time ? (like seq2seq ?)
- Should be unique for each position – not cyclic

Possible Candidates :

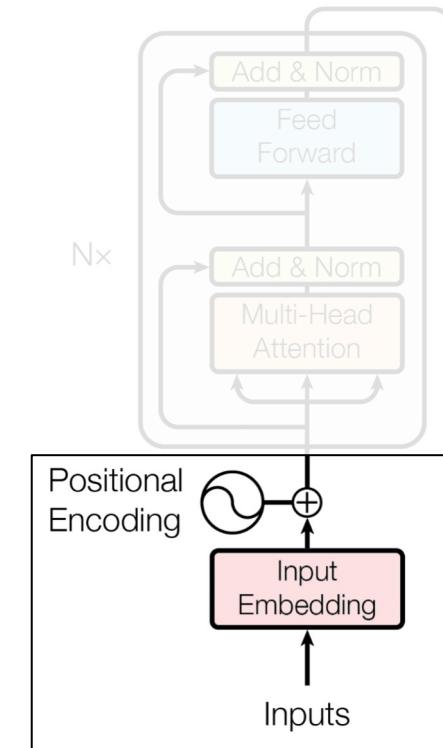
$$P_{t+1} = P_t + \Delta c$$

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

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



Positional Encoding



Positional Encoding

Requirements for Positional Encodings

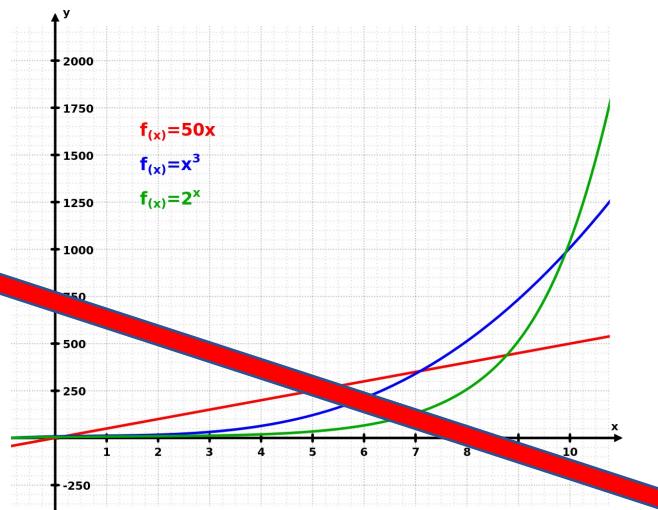
- Some representation of time ? (like seq2seq ?)
- Should be unique for each position – not cyclic
- **Bounded**

Possible Candidates :

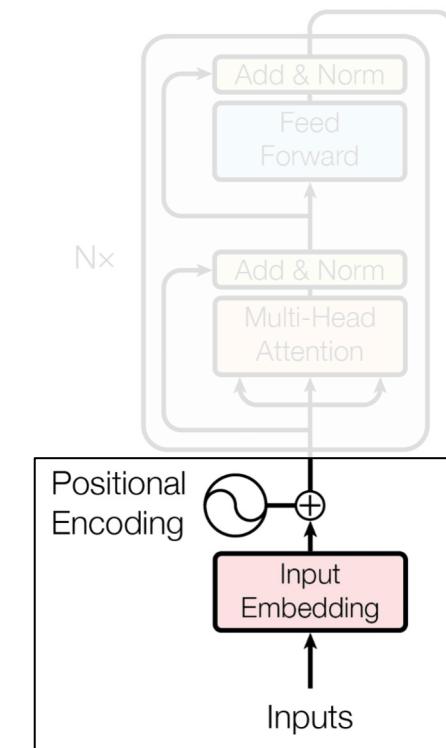
$$P_{t+1} = P_t + \Delta c$$

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

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



Positional Encoding



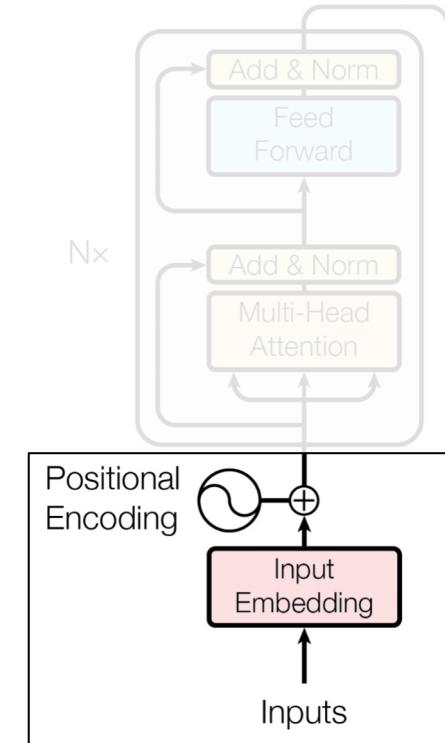
Positional Encoding

Requirements for Positional Encodings

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

Possible Candidates :

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



Positional Encoding

Positional Encoding

Requirements for Positional Encodings

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

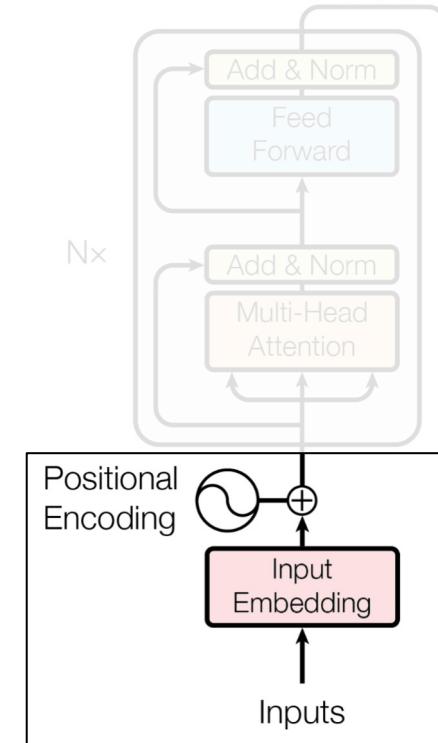
Possible Candidates :

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

M ?

1. **Should be a unitary matrix**
2. **Magnitudes of eigen value should be 1 -> norm preserving**

Positional Encoding



Positional Encoding

Requirements for Positional Encodings

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

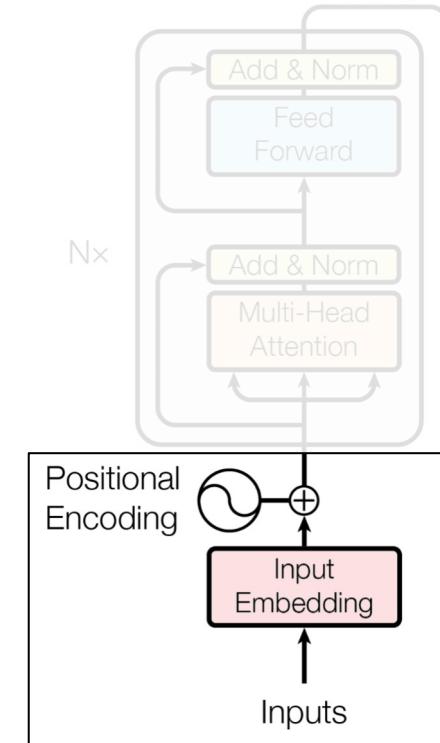
Possible Candidates :

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

M

1. **The matrix can be learnt**
2. **Produces unique rotated embeddings each time**

Positional Encoding



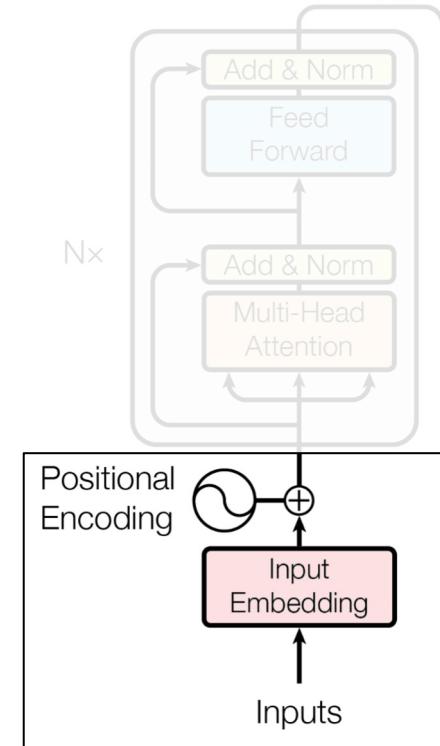
Rotary Positional Embedding

RoFORMER: ENHANCED TRANSFORMER WITH ROTARY POSITION EMBEDDING

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

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

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



[REF: Rotary Positional Embeddings](#)

Positional Encoding

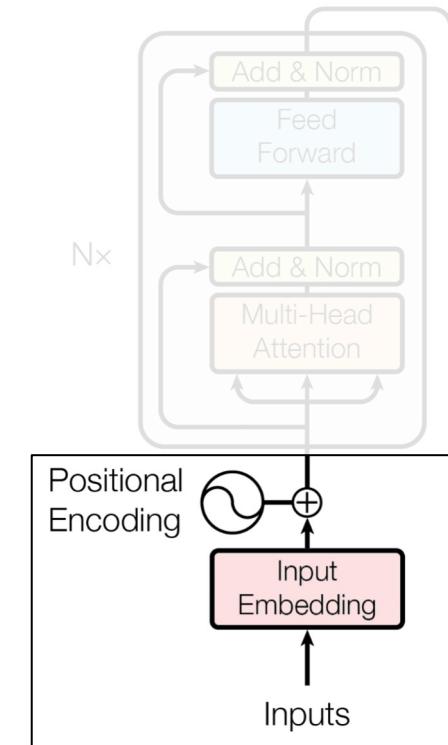
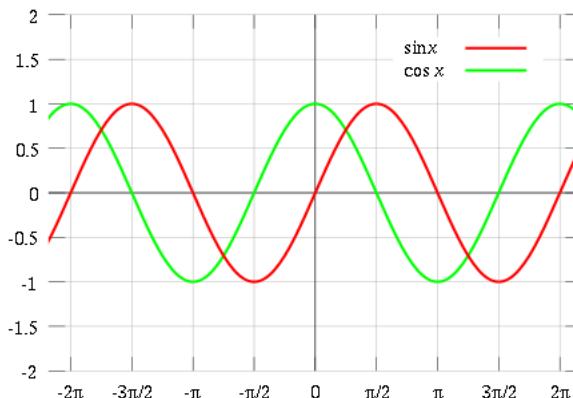
Requirements for Positional Encodings

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

Actual Candidates :

$\sin(g(t))$

$\cos(g(t))$



Positional Encoding

Positional Encoding

Requirements for $g(t)$

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

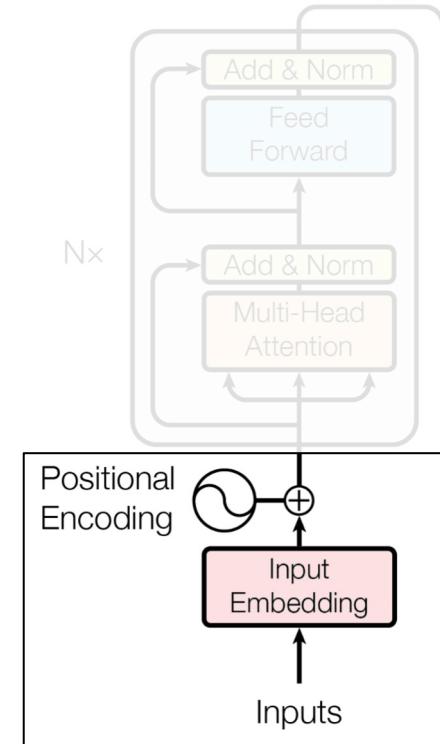
pos \rightarrow idx of the token in input sentence

i \rightarrow ith dimension out of d

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

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

Positional Encoding



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

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

Positional Encoding

Requirements for $g(t)$

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

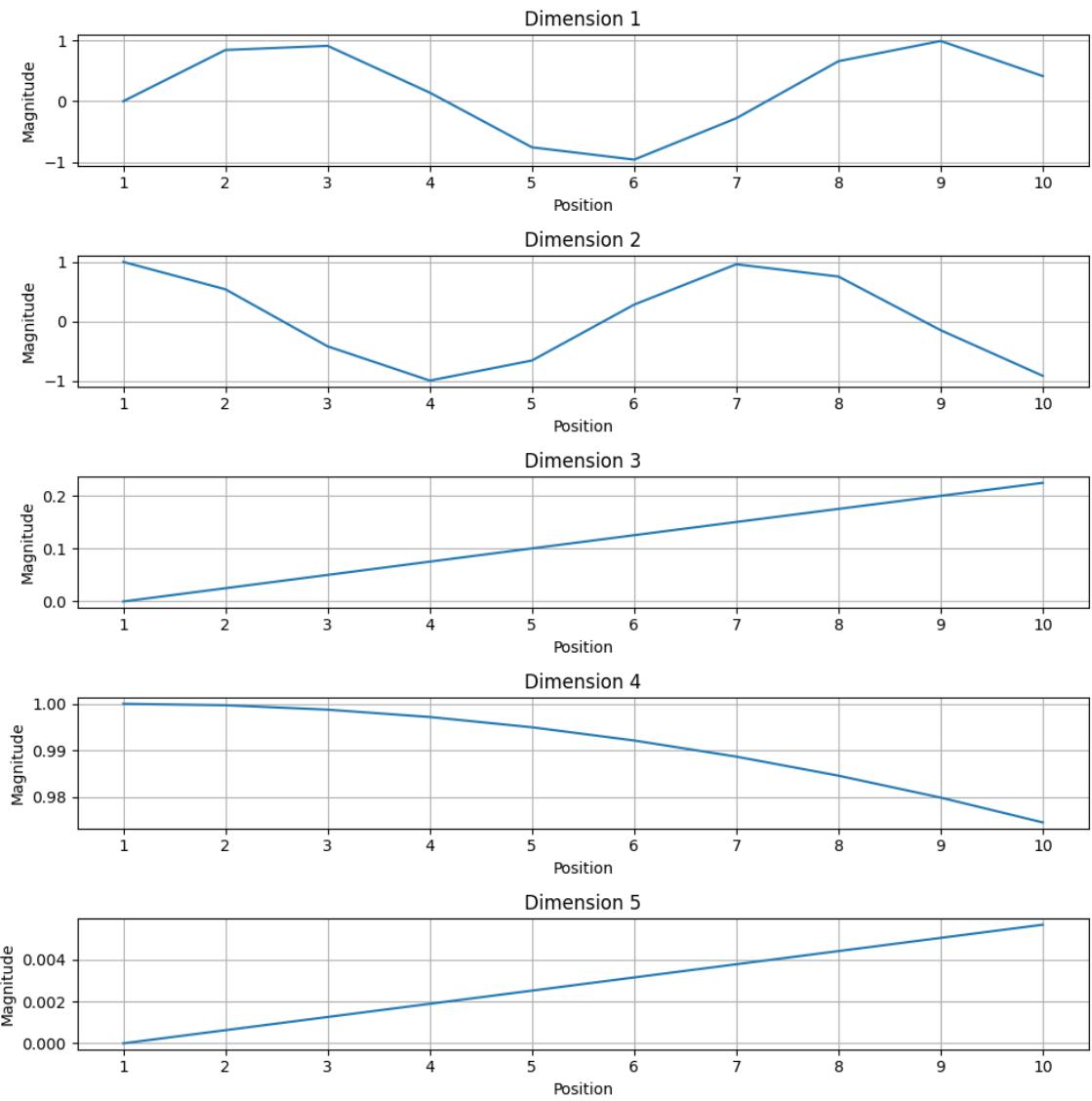
pos \rightarrow idx of the token in input sentence

i \rightarrow ith dimension out of d

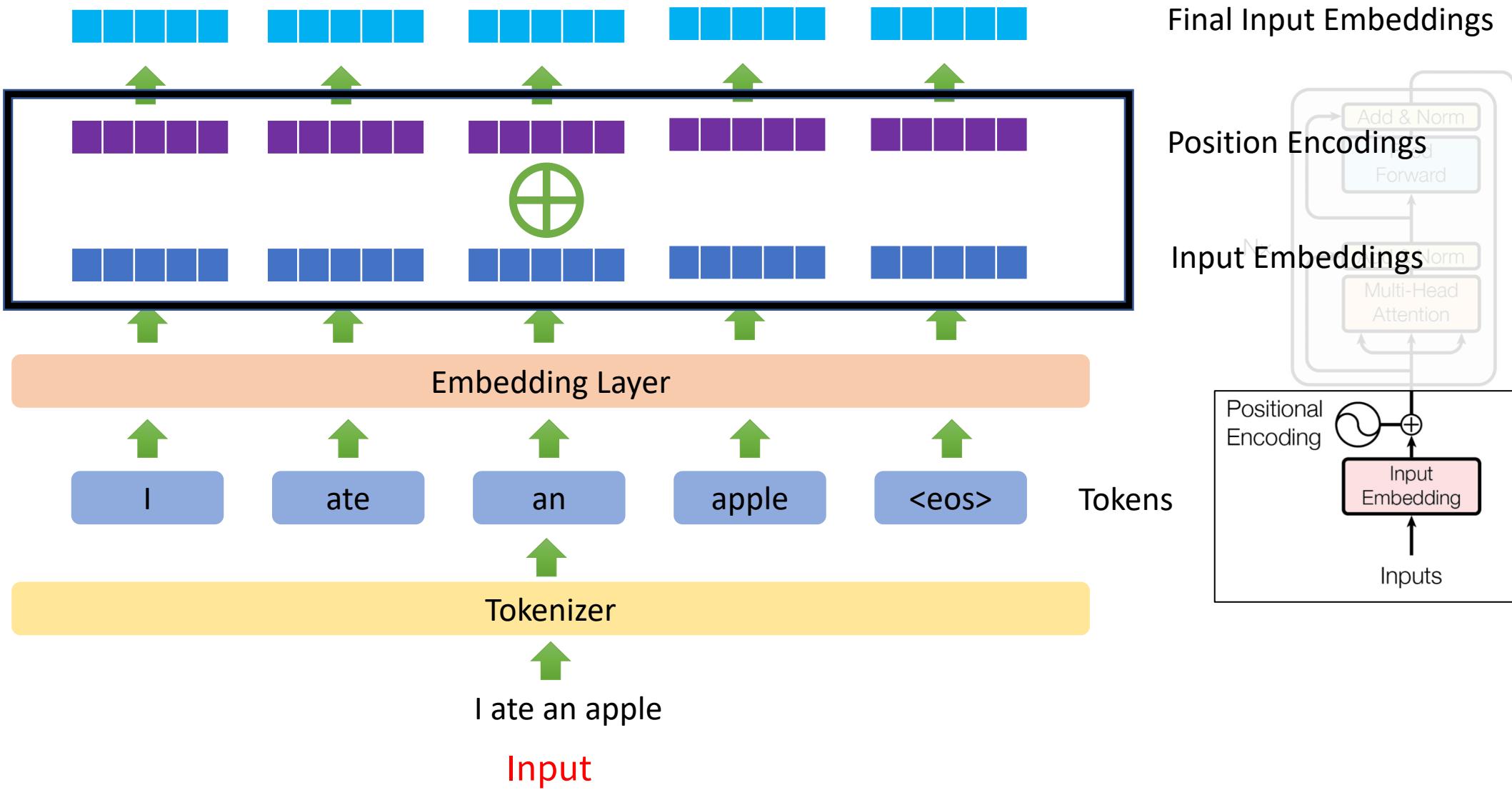


Positional Encoding:

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



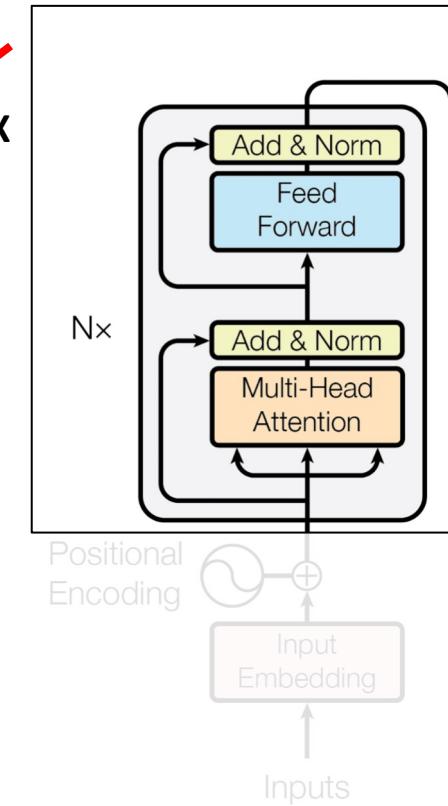
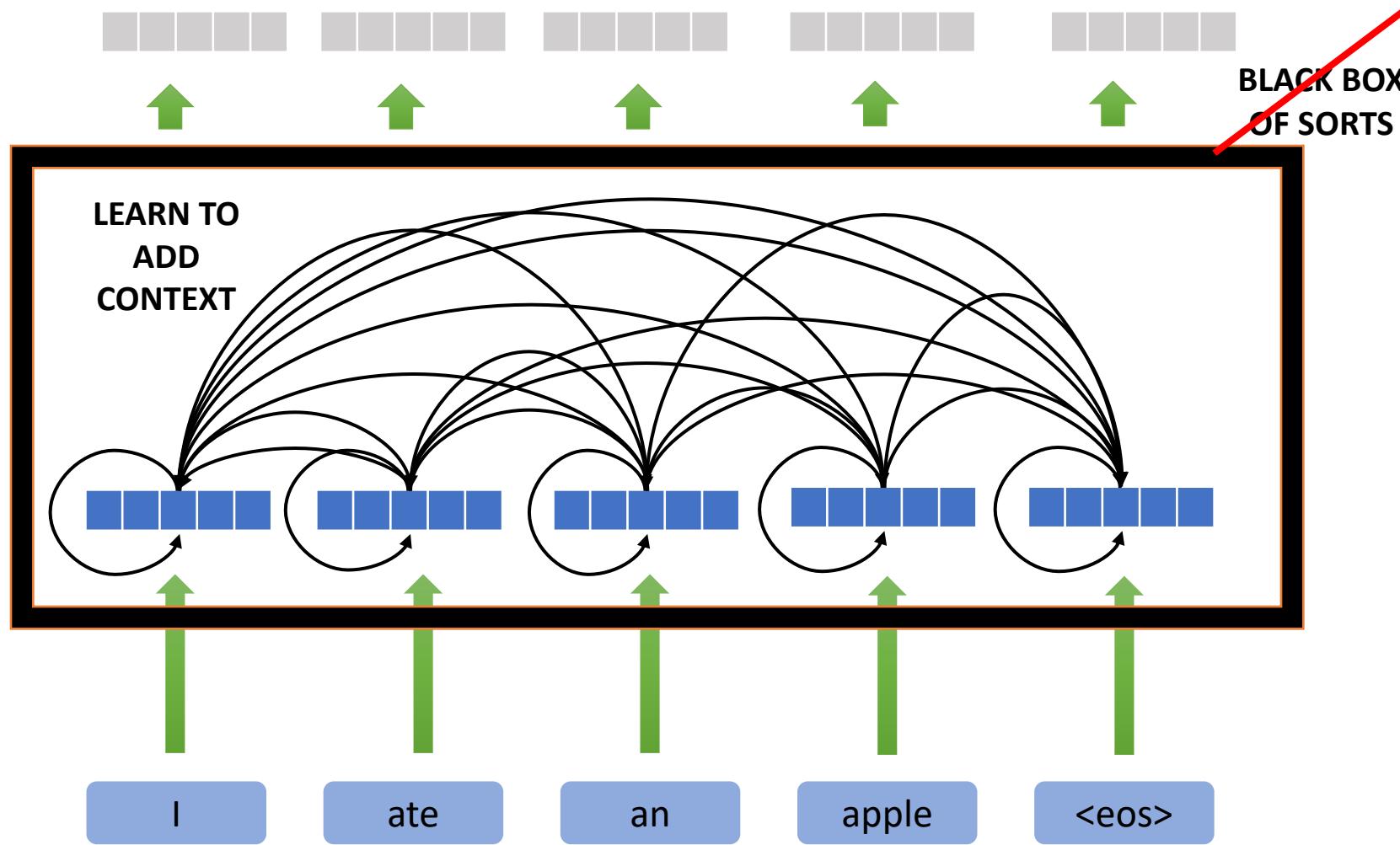
Positional Encoding



Encoder

$\alpha_{[i,j]}$ Σ

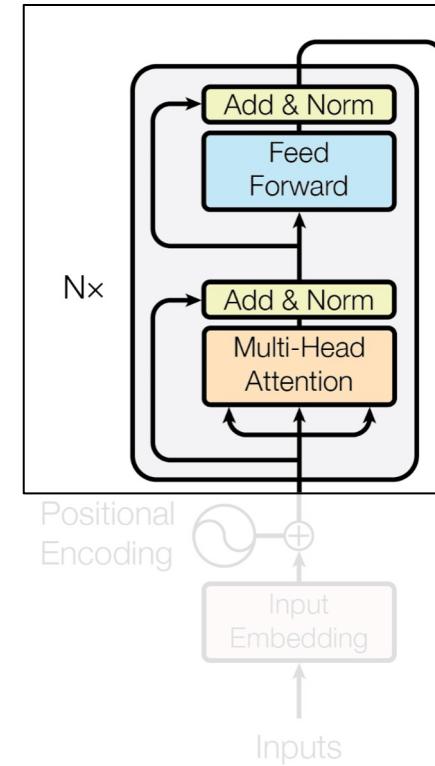
CONTEXTUALLY RICH EMBEDDINGS



Self Attention

From lecture 18:

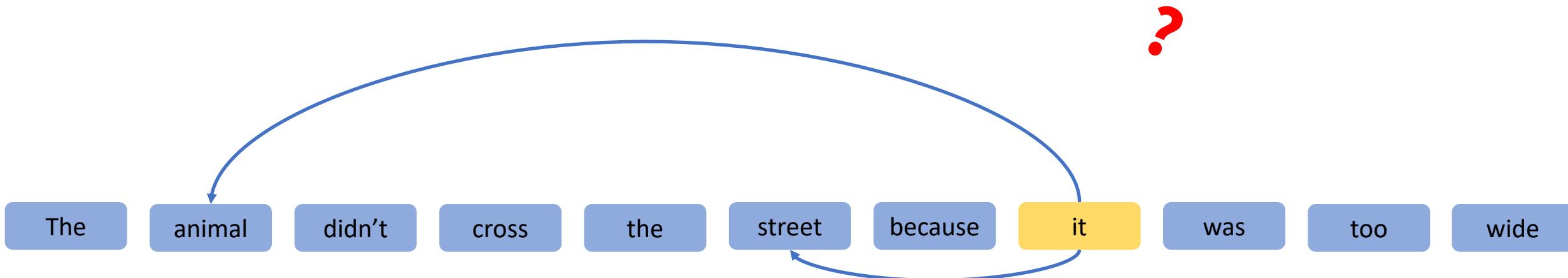
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



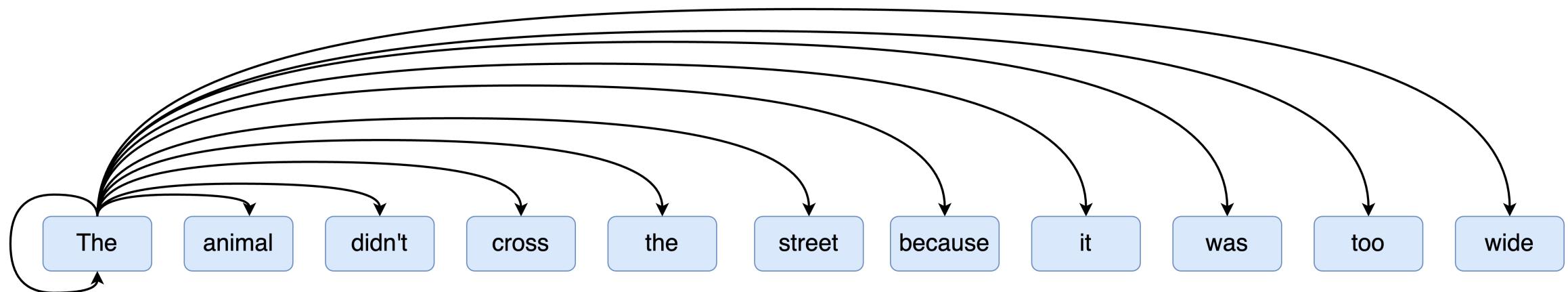
Self Attention

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

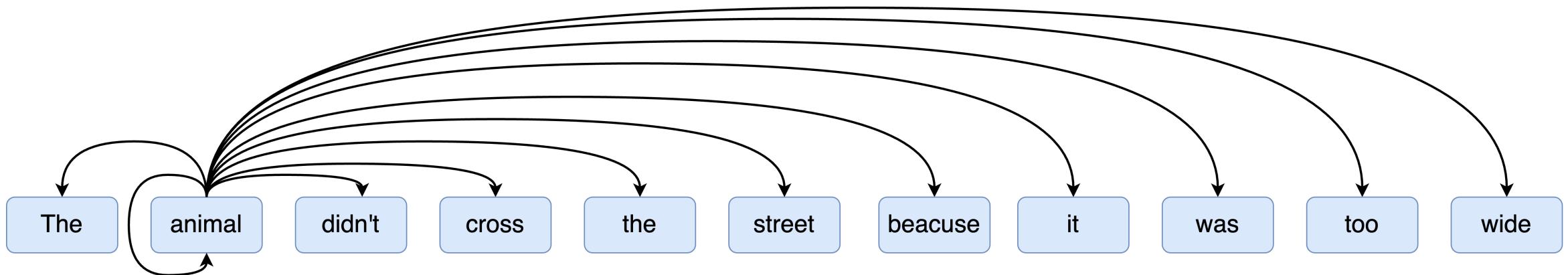
Self Attention



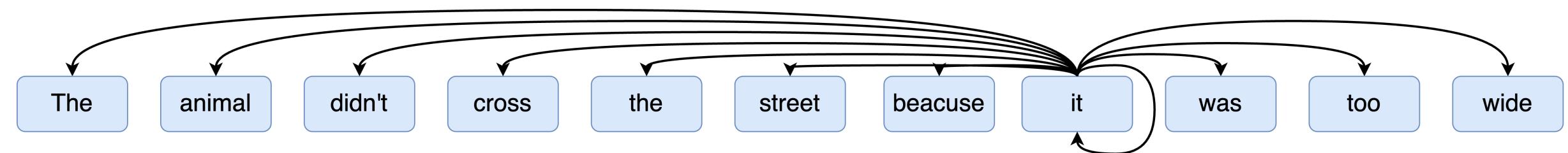
Self Attention



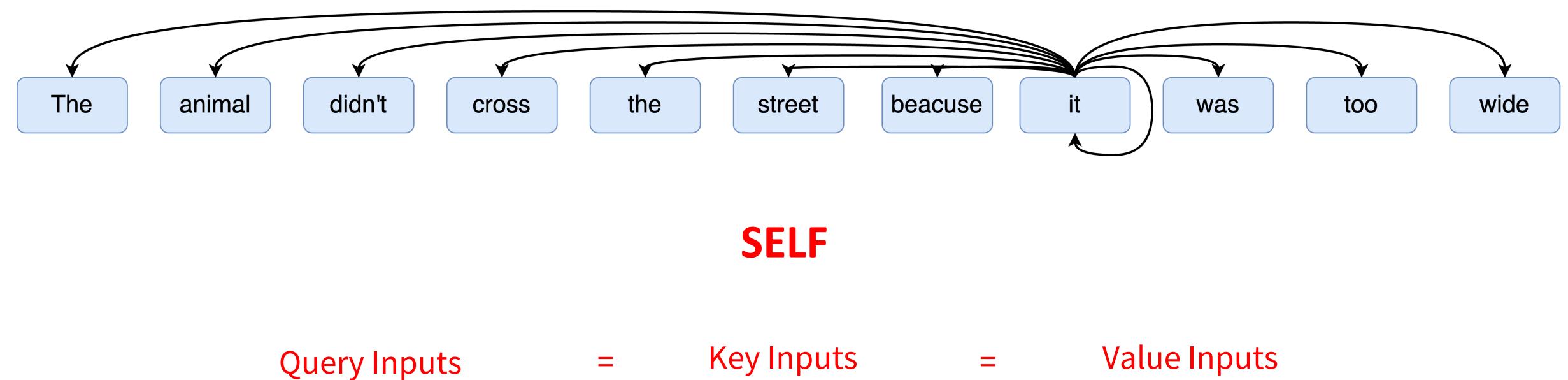
Self Attention



Self Attention

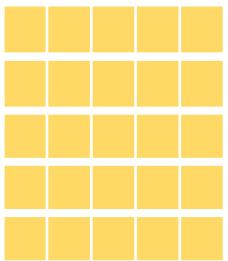


Self Attention

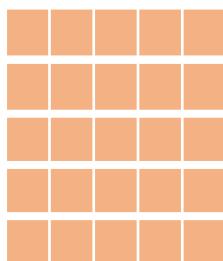


Self Attention

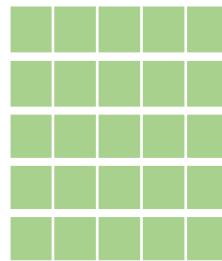
$$R^{d_{model} \times d_{model}}$$



W_Q

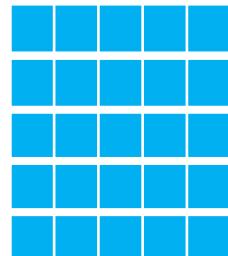


W_K

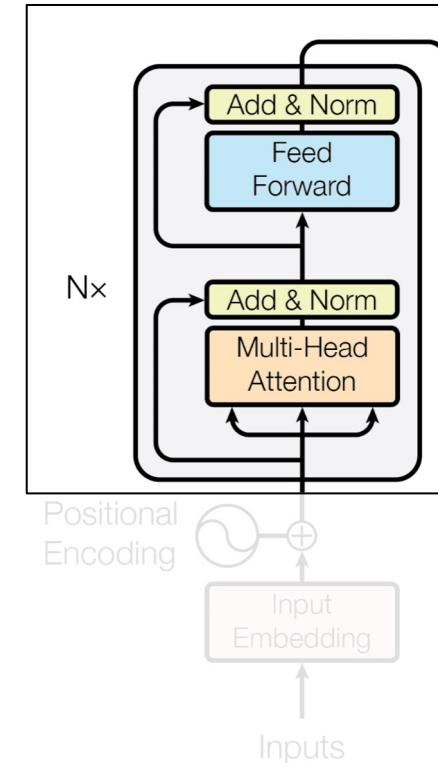


W_V

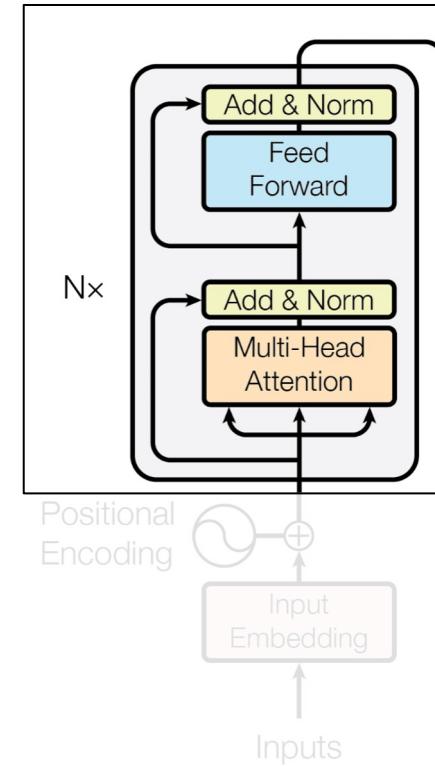
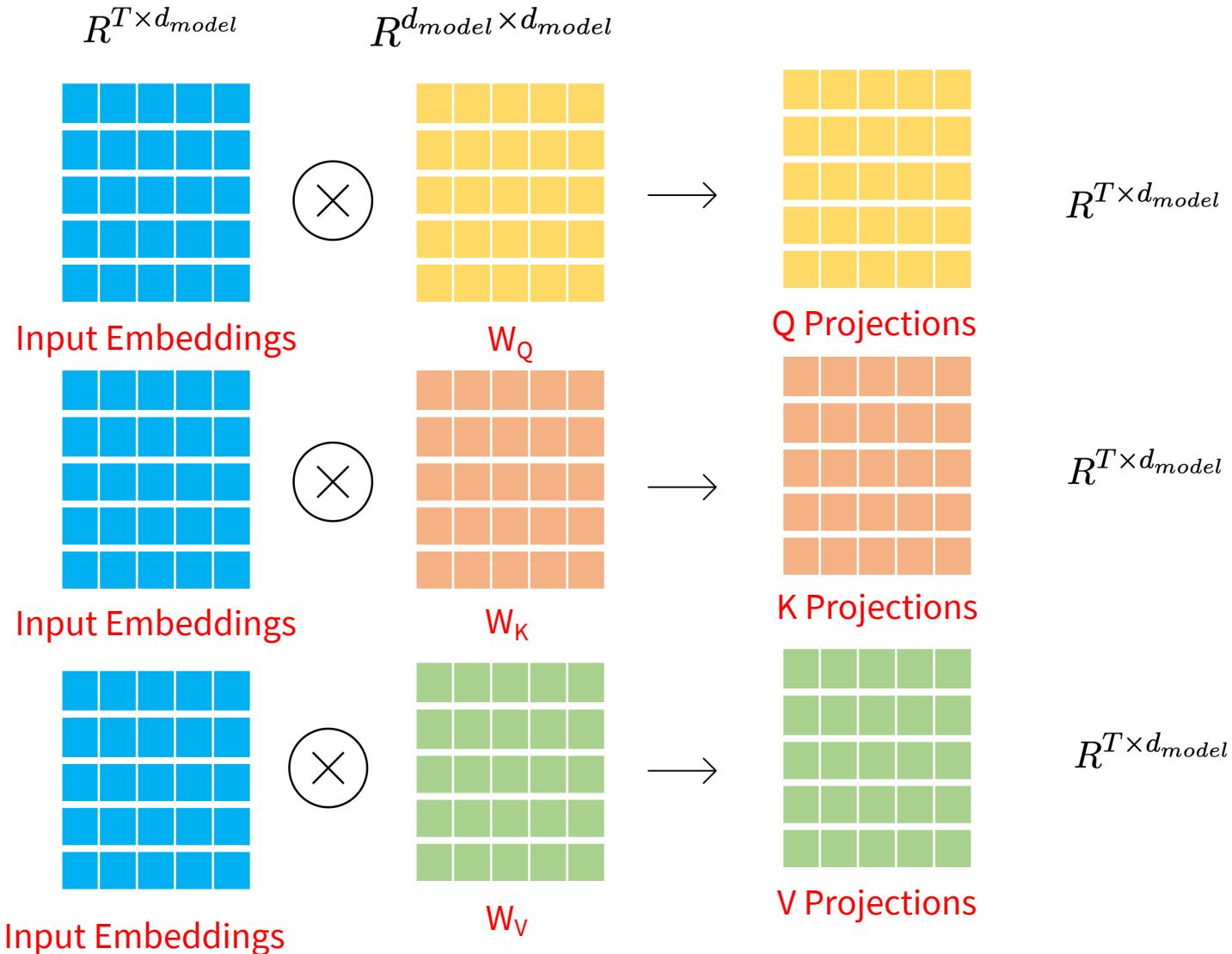
$$R^{T \times d_{model}}$$



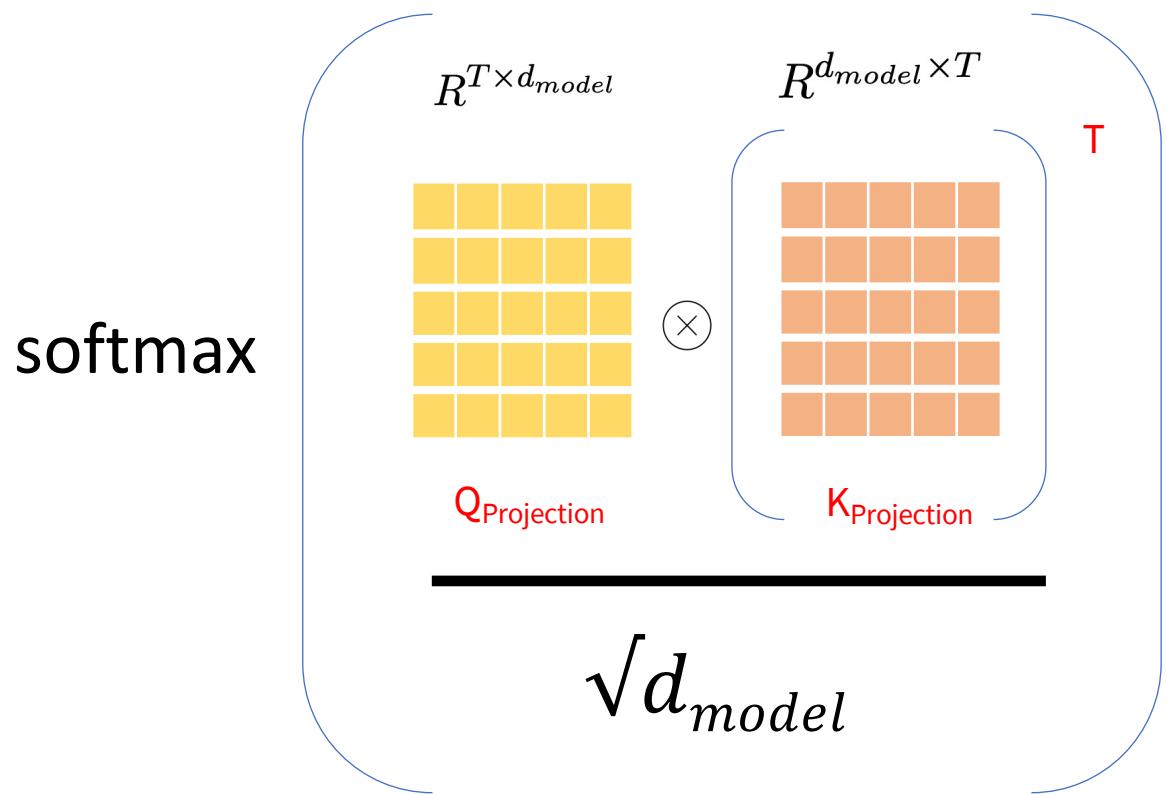
Input Embeddings



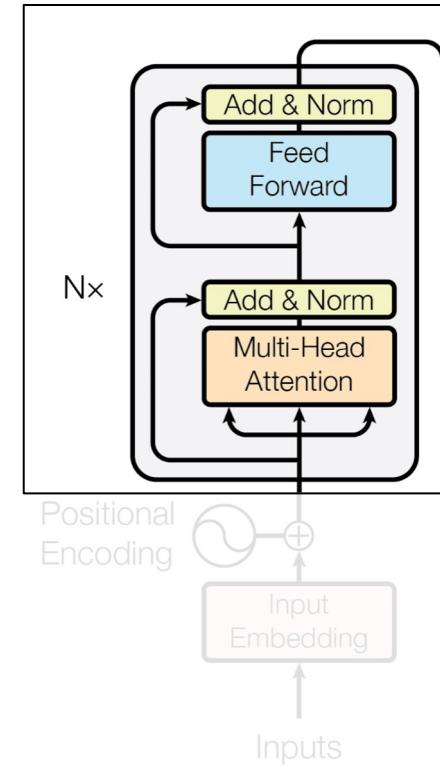
Self Attention



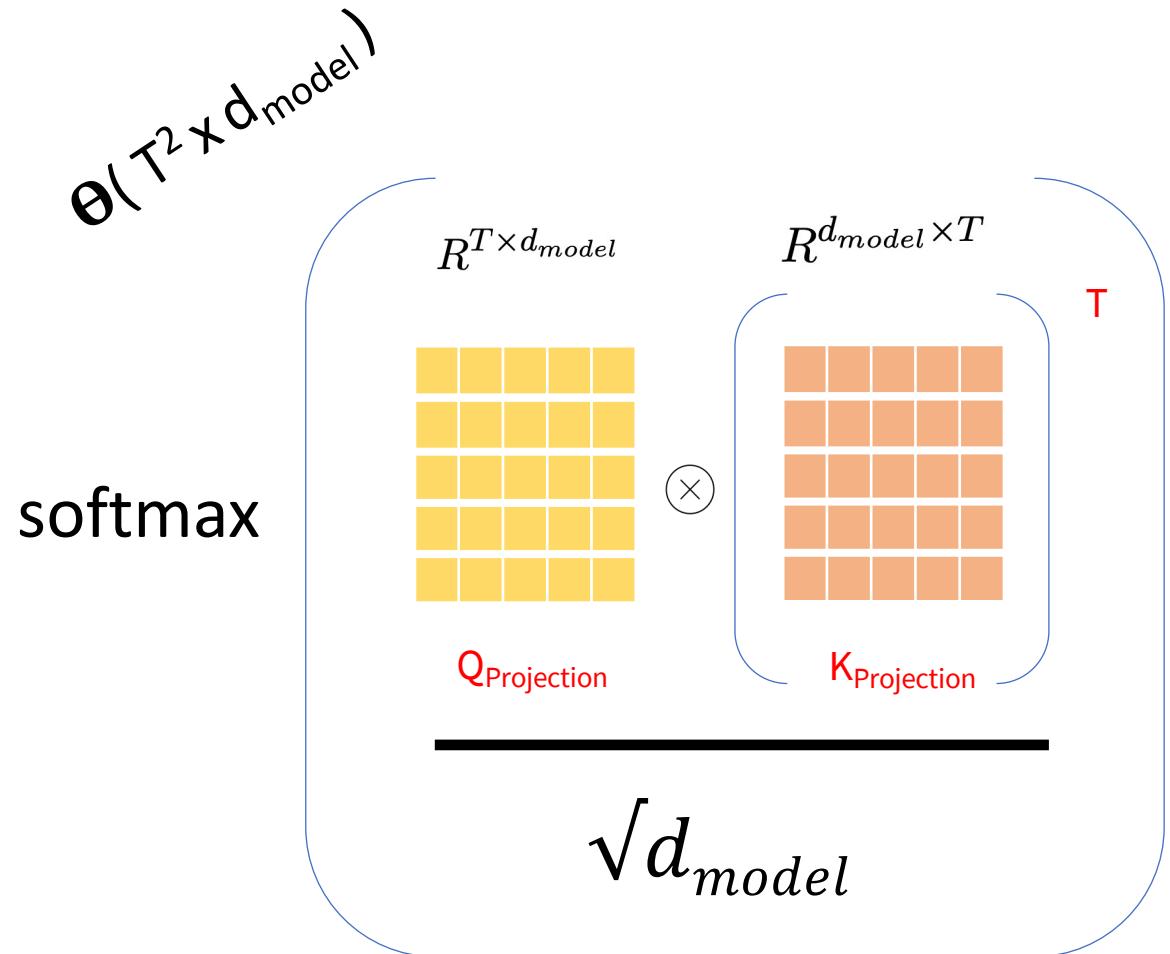
Self Attention



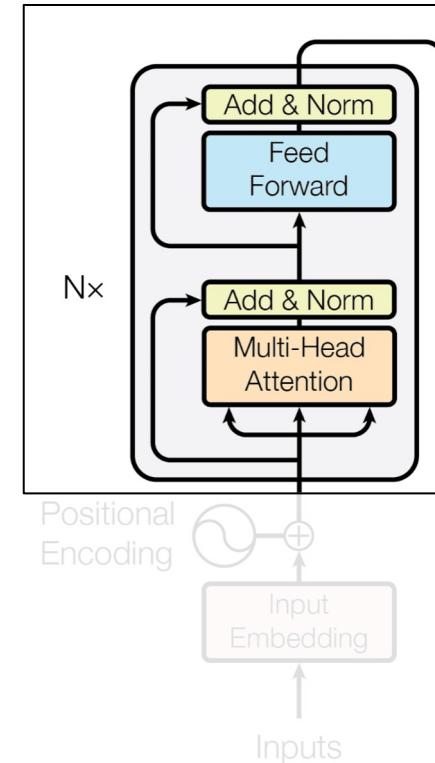
$$R^{T \times T}$$



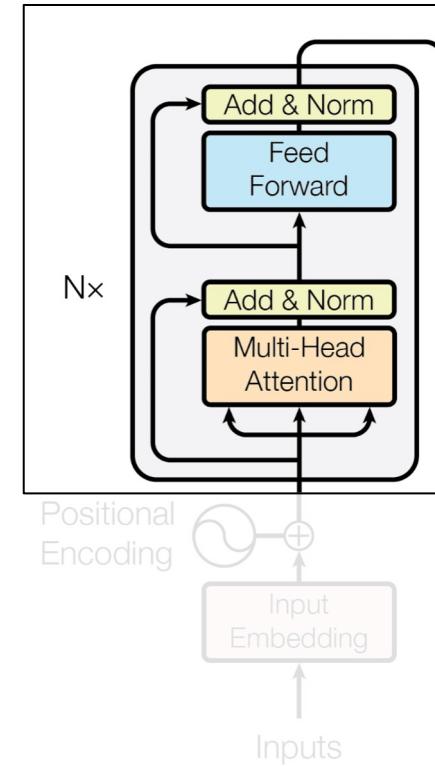
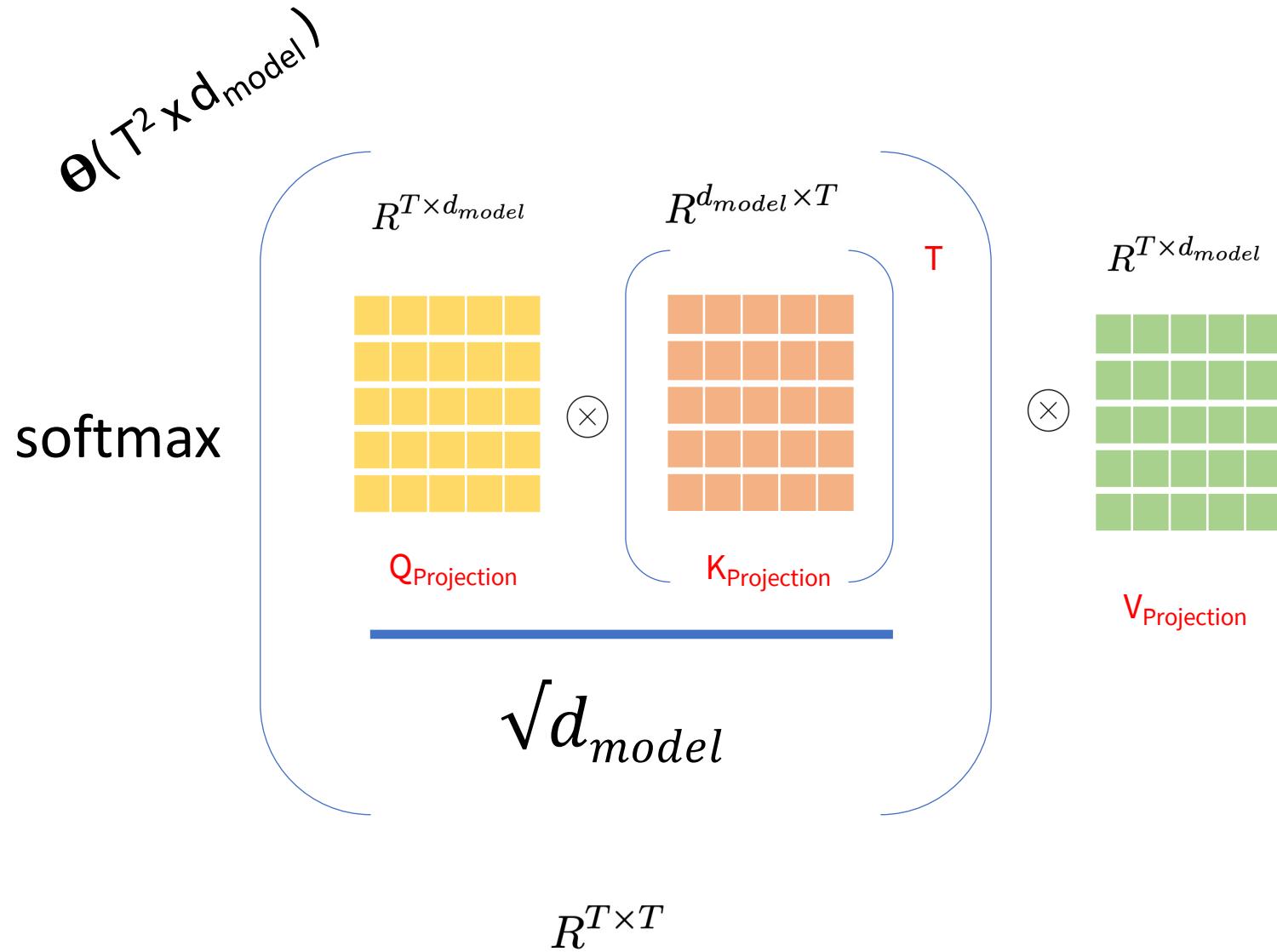
Self Attention



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

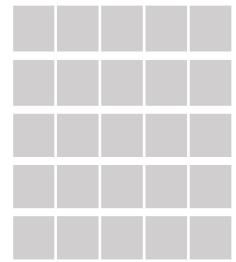


Self Attention

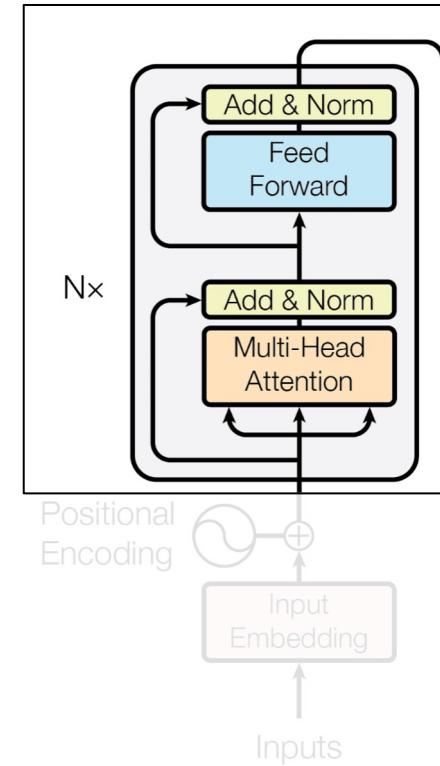


Self Attention

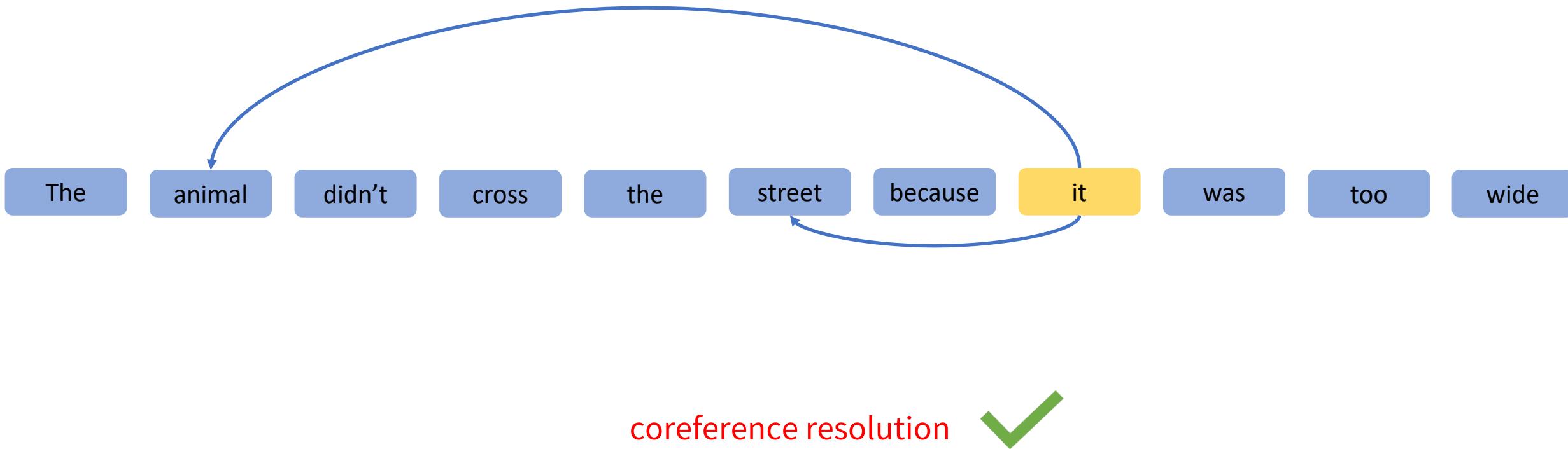
$$R^{T \times d_{model}}$$



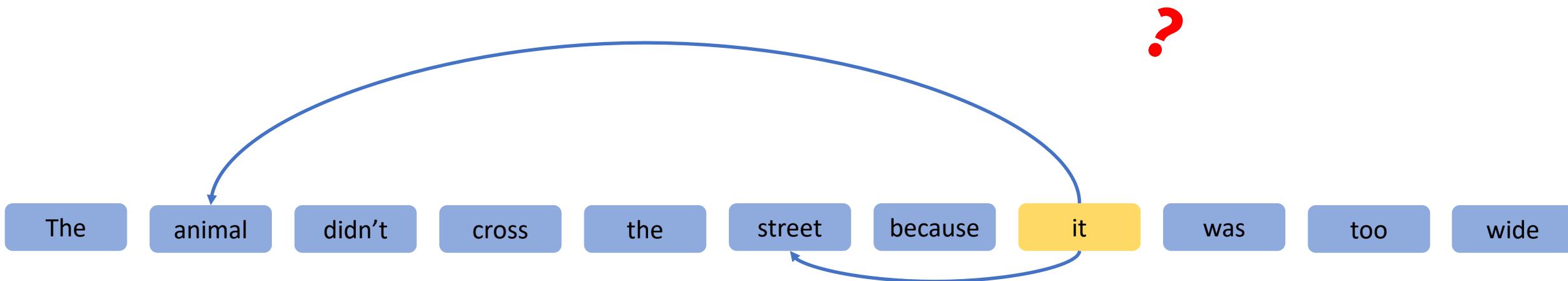
Attention: Z



Self Attention



Self Attention



Sentence boundaries ?

coreference resolution



Context ?

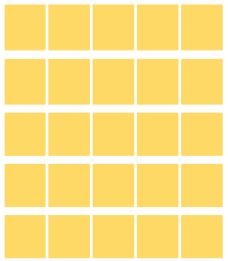
Semantic relationships ?

Part of Speech ?

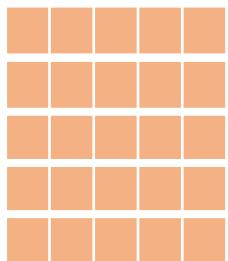
Comparisons ?

Self Attention

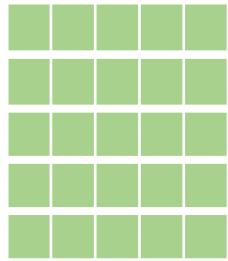
$$R^{d_{model} \times d_{model}}$$



W_Q

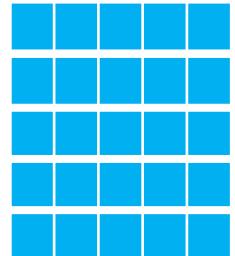


W_K

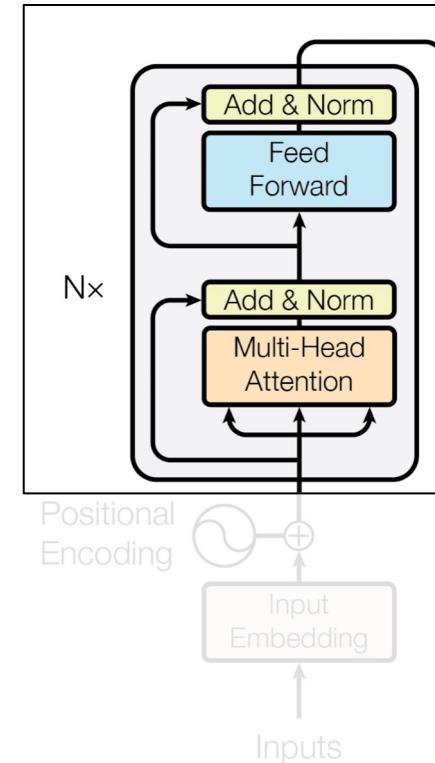


W_V

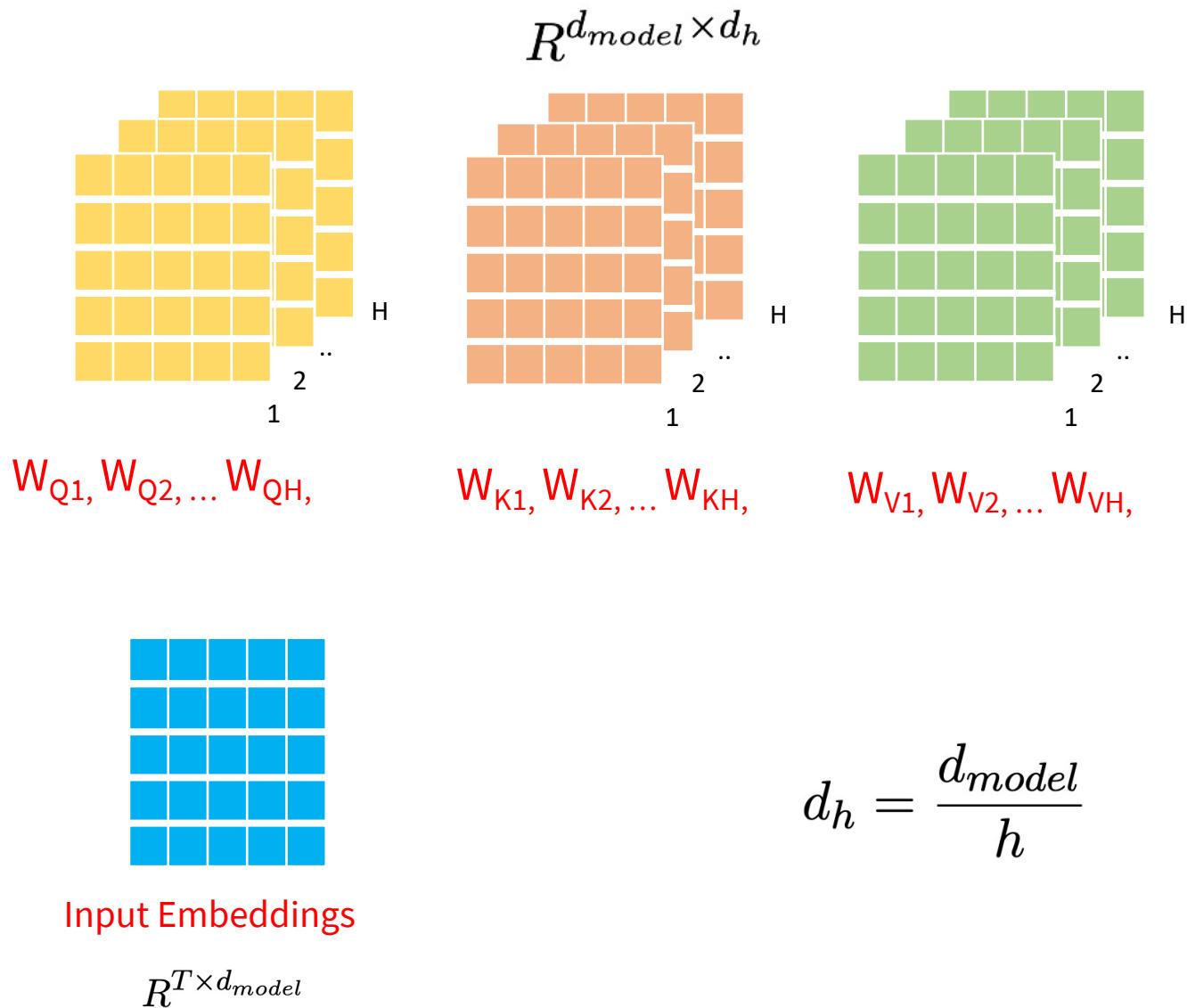
$$R^{T \times d_{model}}$$



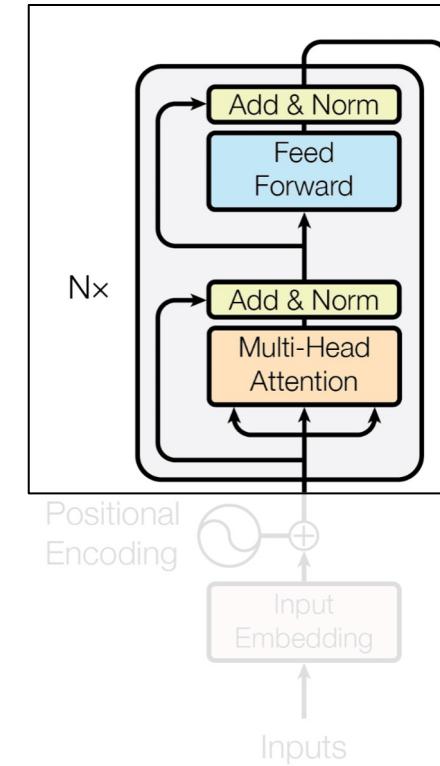
Input Embeddings



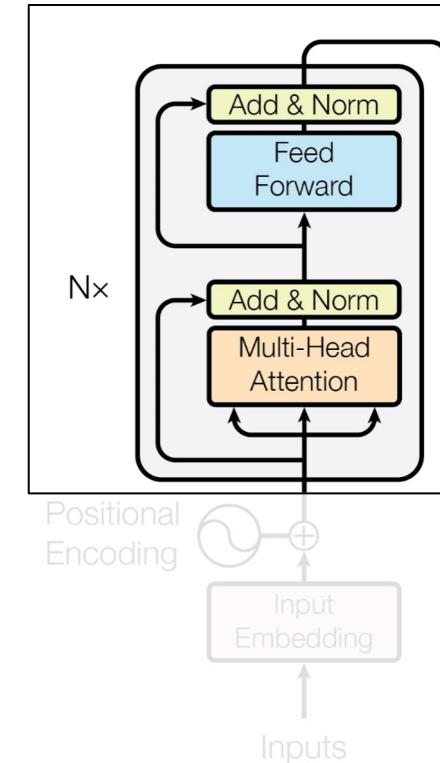
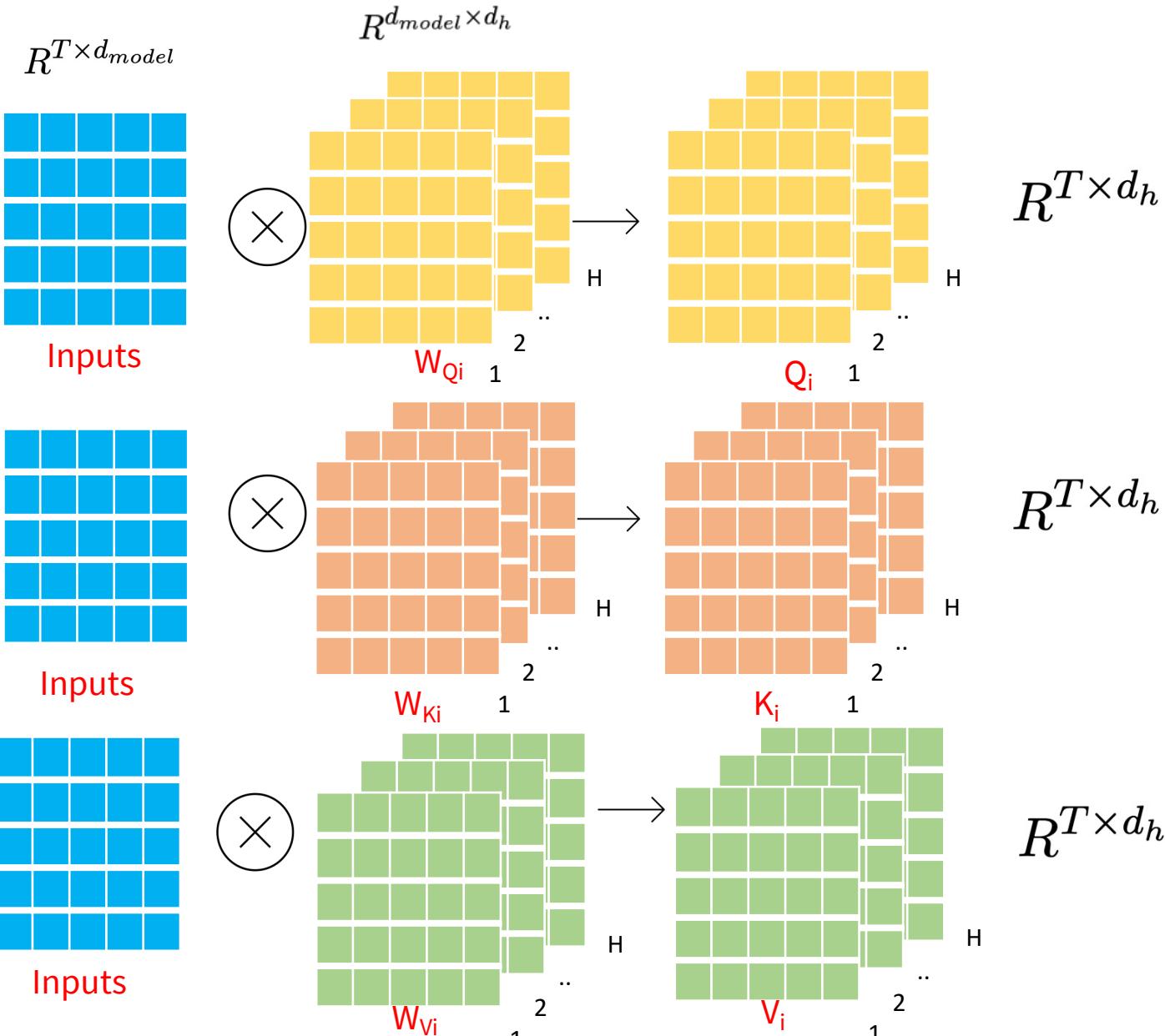
Multi-Head Attention



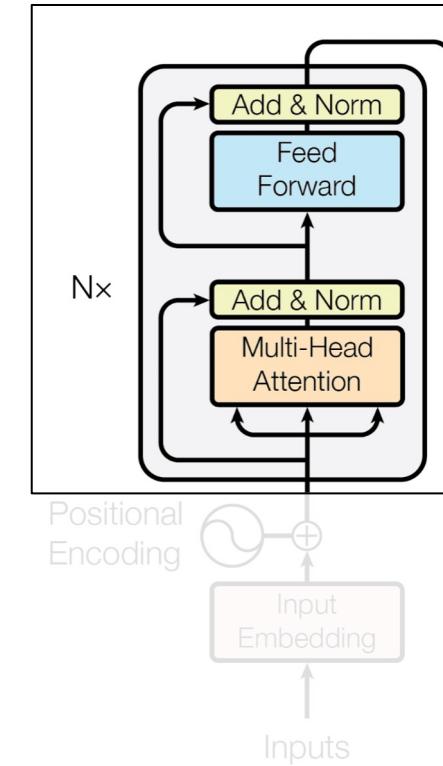
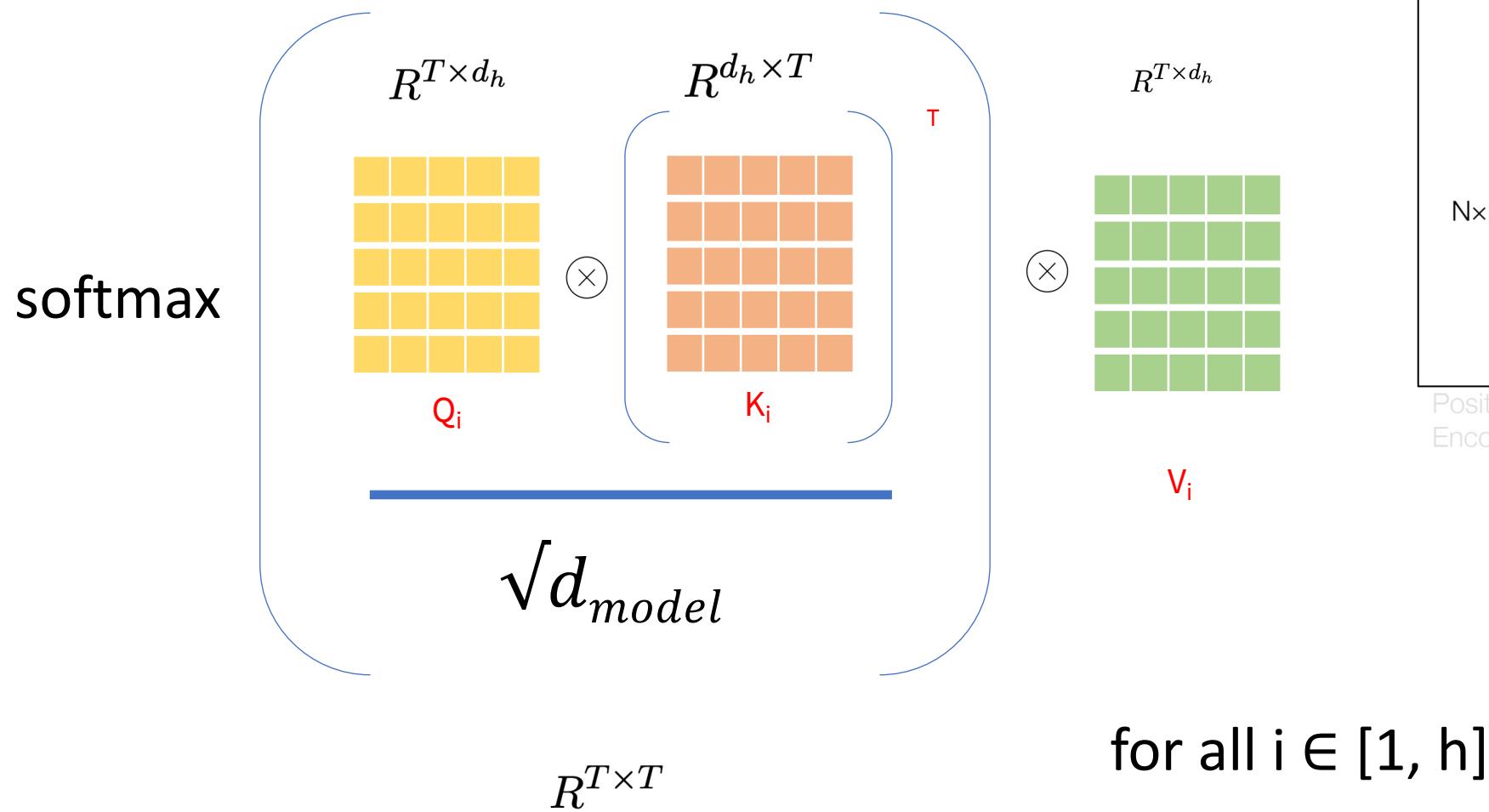
$$d_h = \frac{d_{model}}{h}$$



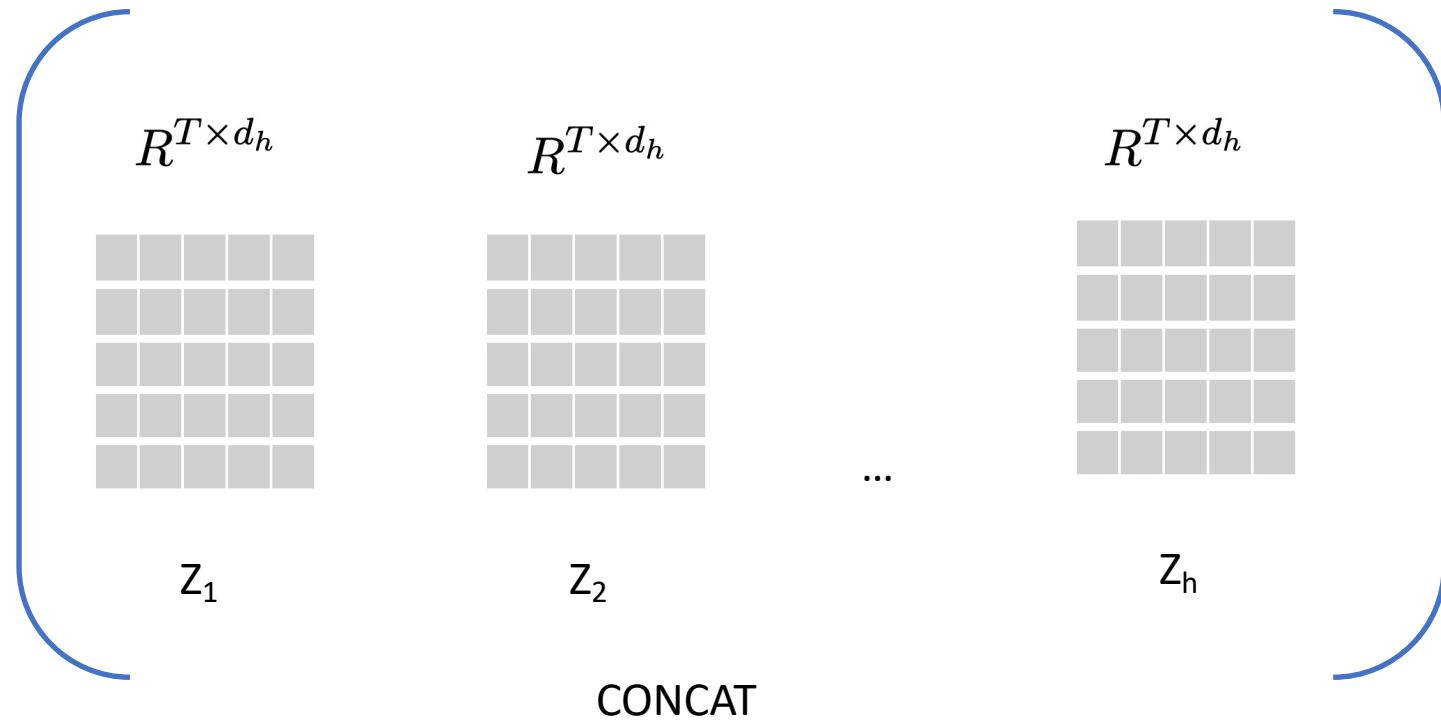
Multi-Head Attention



Multi-Head Attention



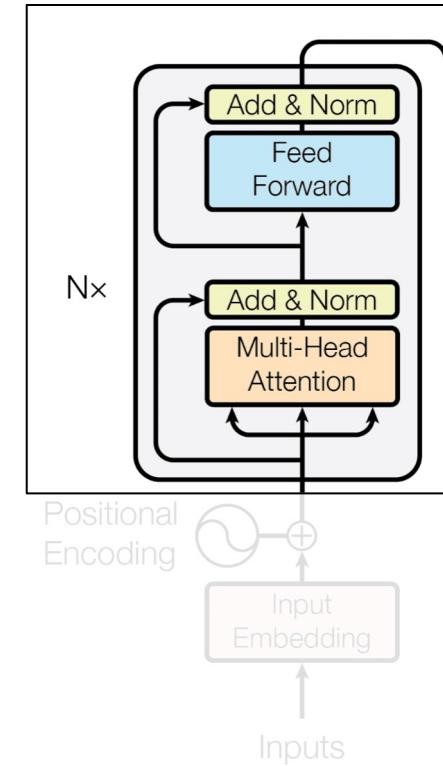
Multi-Head Attention



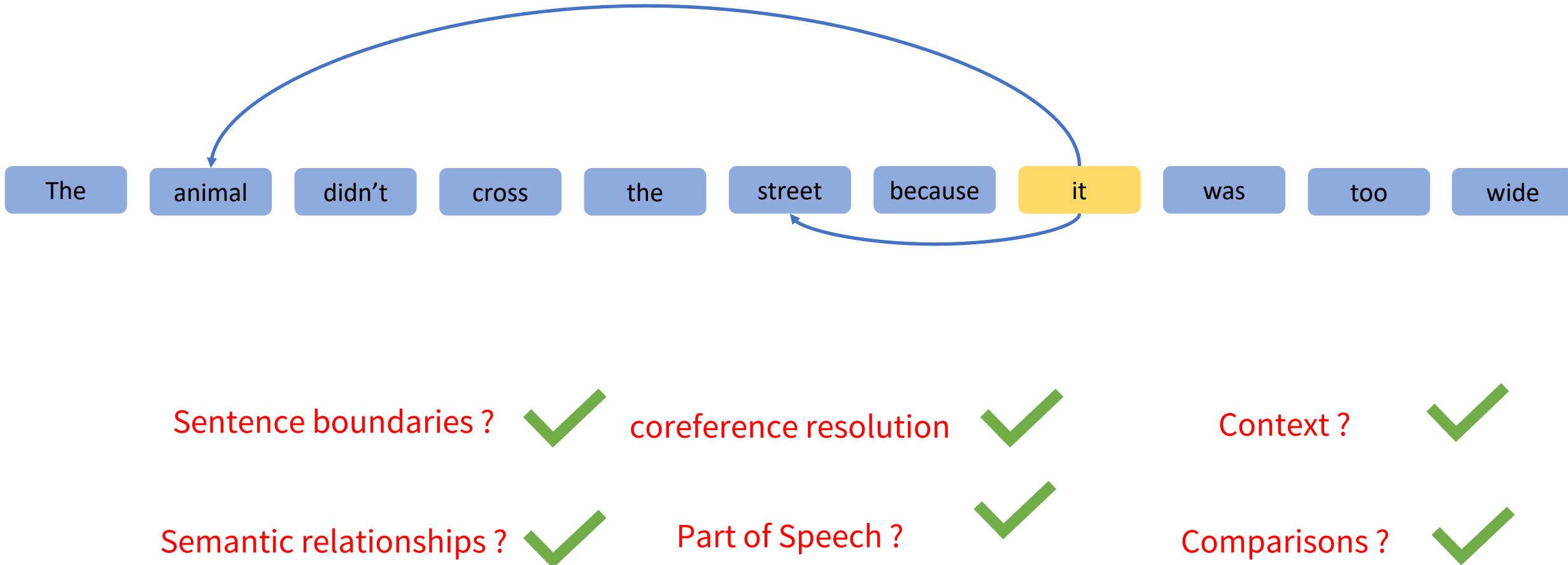
Multi Head Attention : Z

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

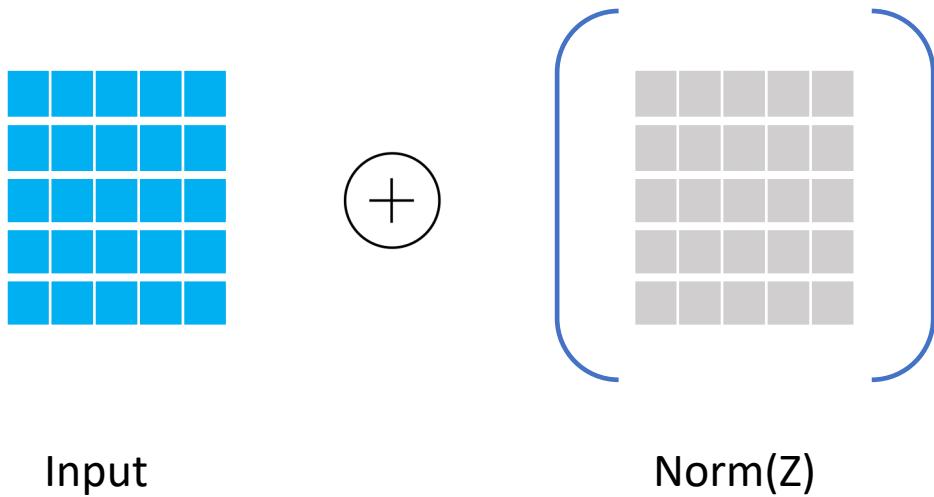
$$R^{T \times d_{model}}$$



Multi-Head Attention



Add & Norm

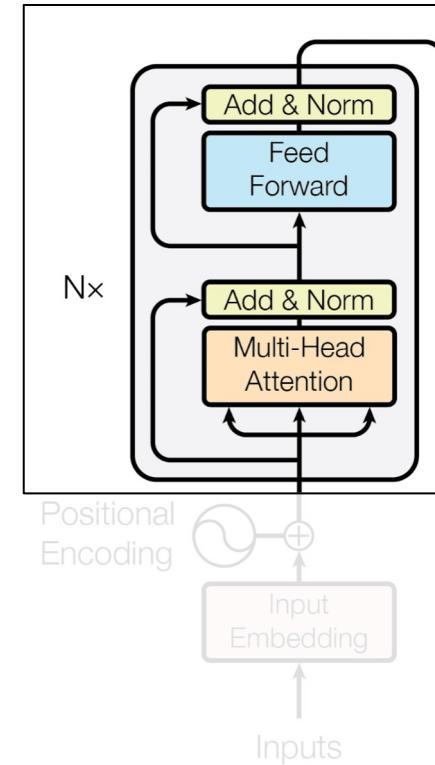


Normalization(Z)

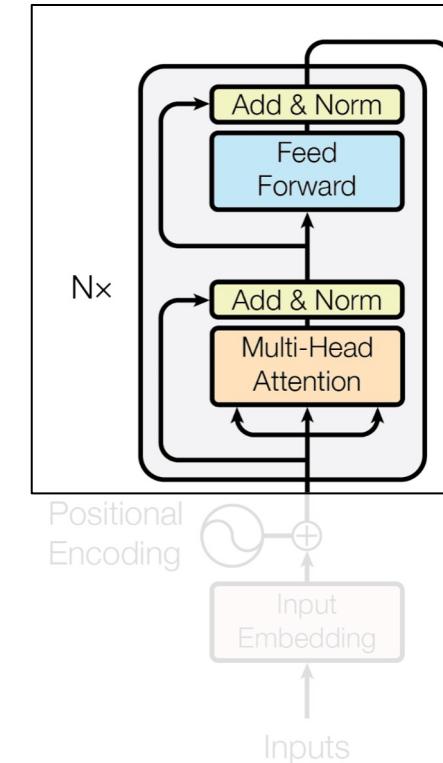
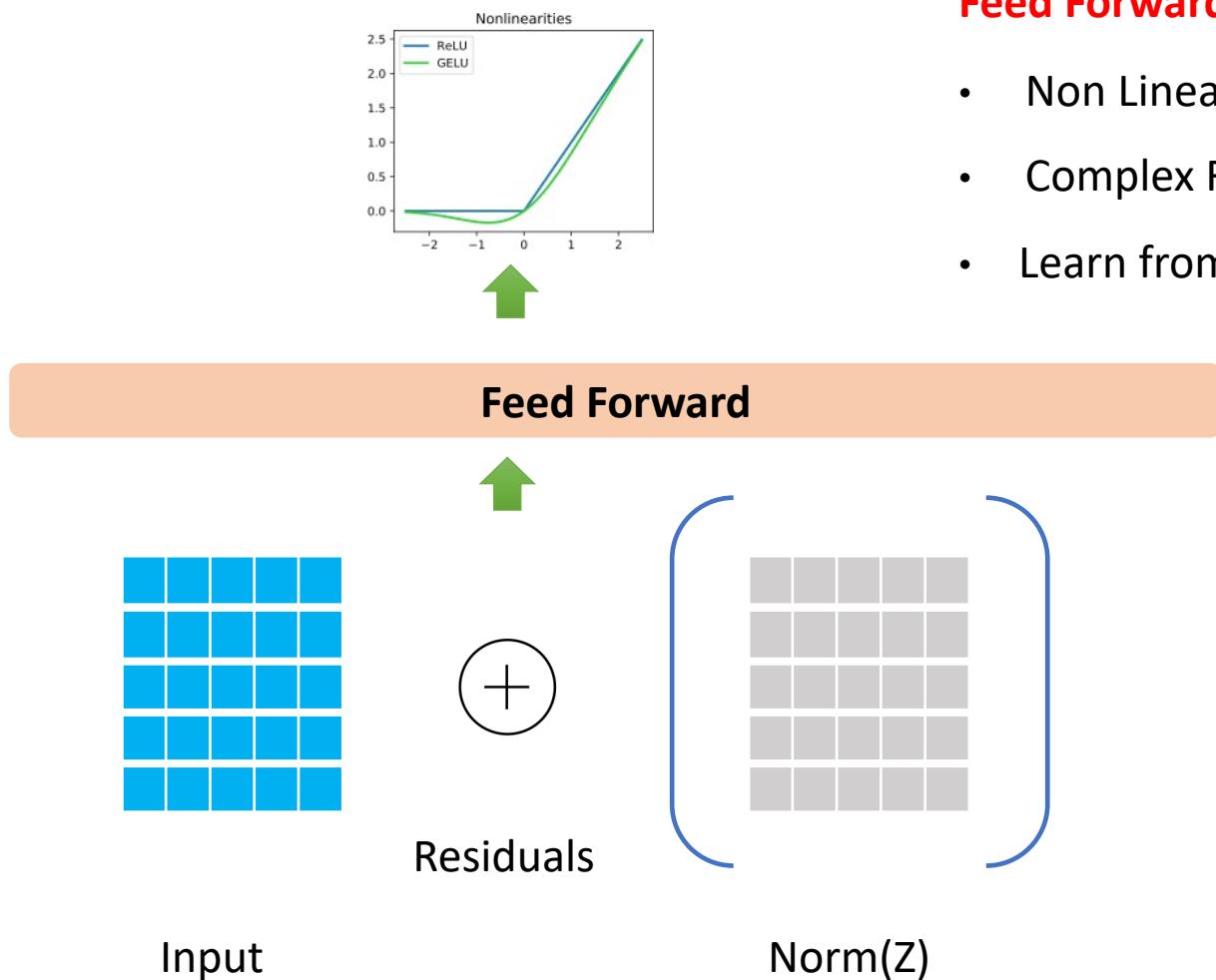
- Mean 0, Std dev 1
- Stabilizes training
- Regularization effect

Add -> Residuals

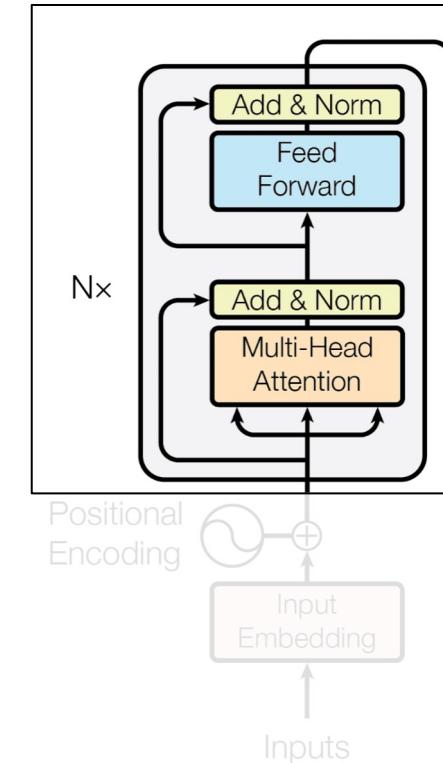
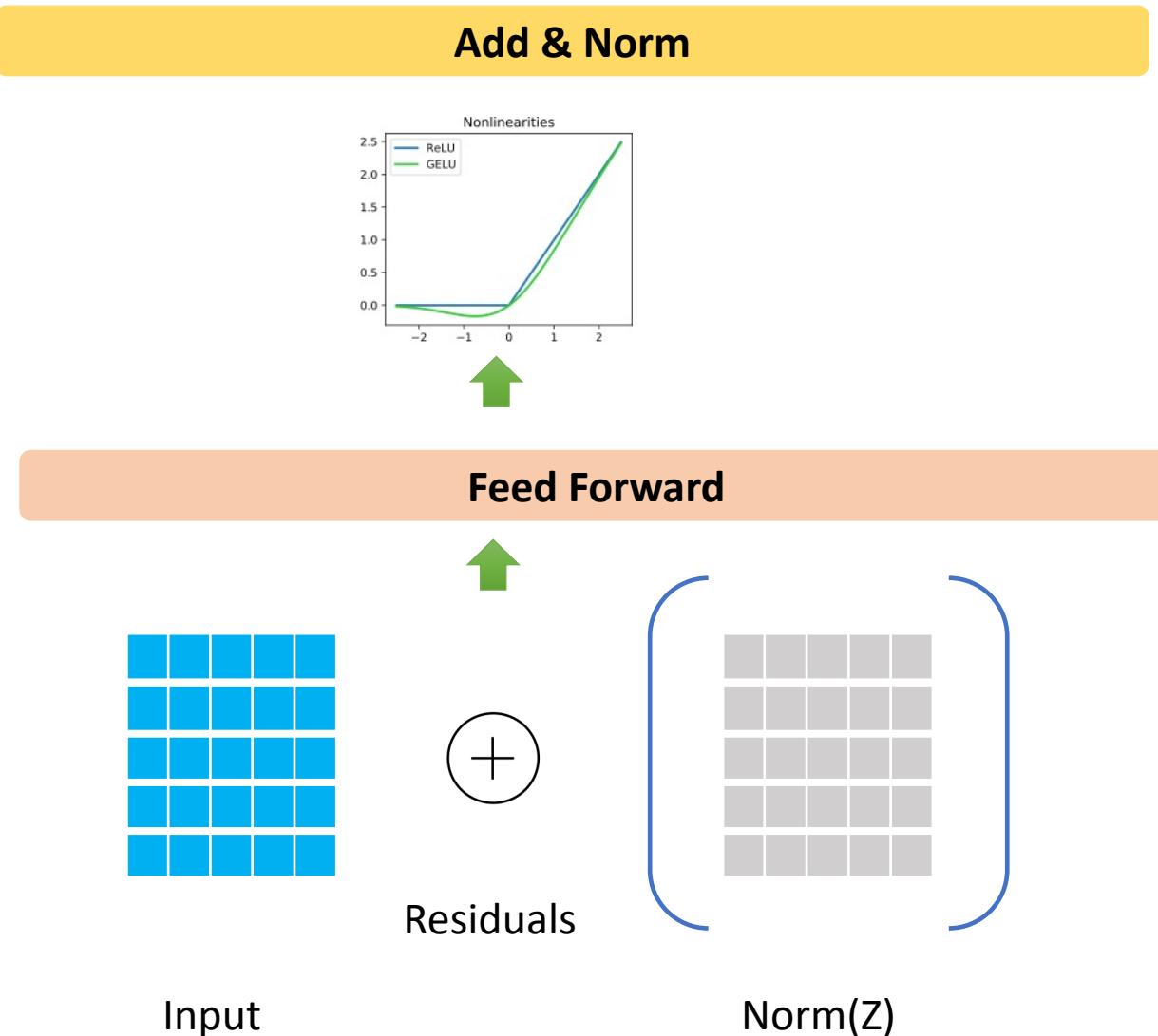
- Avoid vanishing gradients
- Train deeper networks



Feed Forward

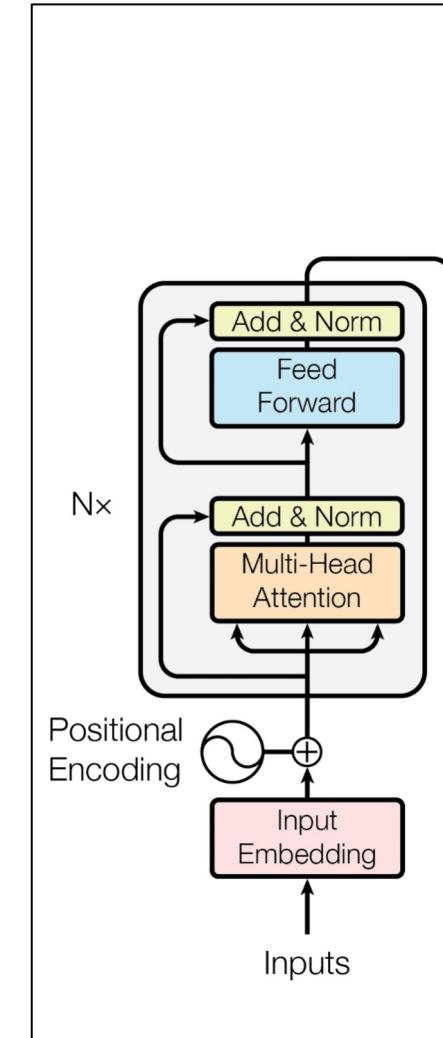


Add & Norm



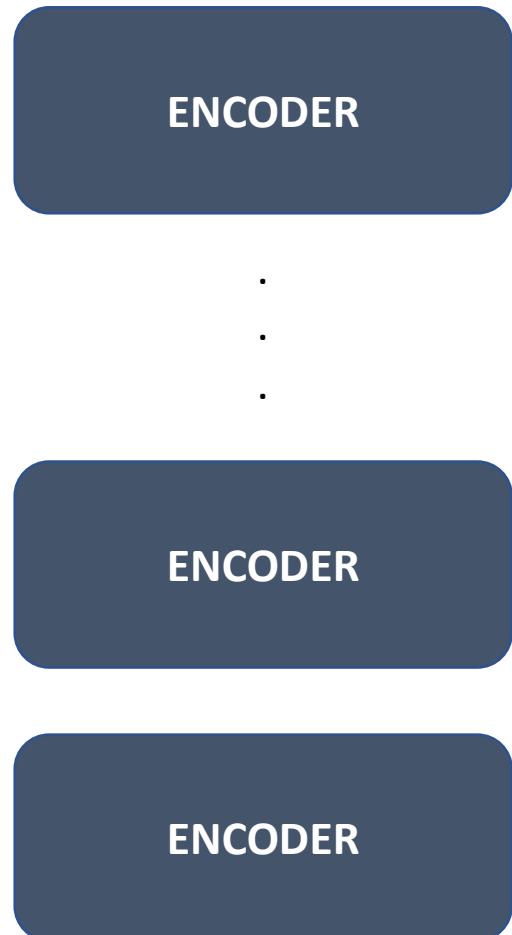
Encoders

Encoder

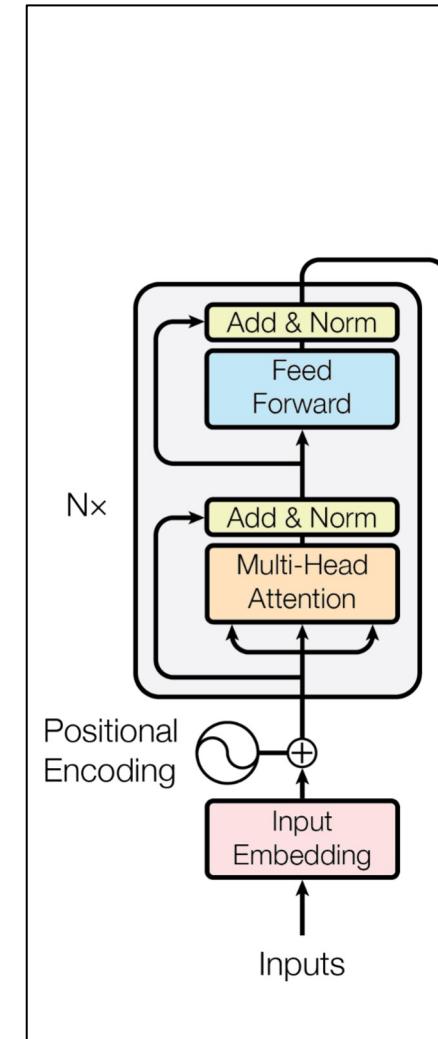


Encoders

Encoder

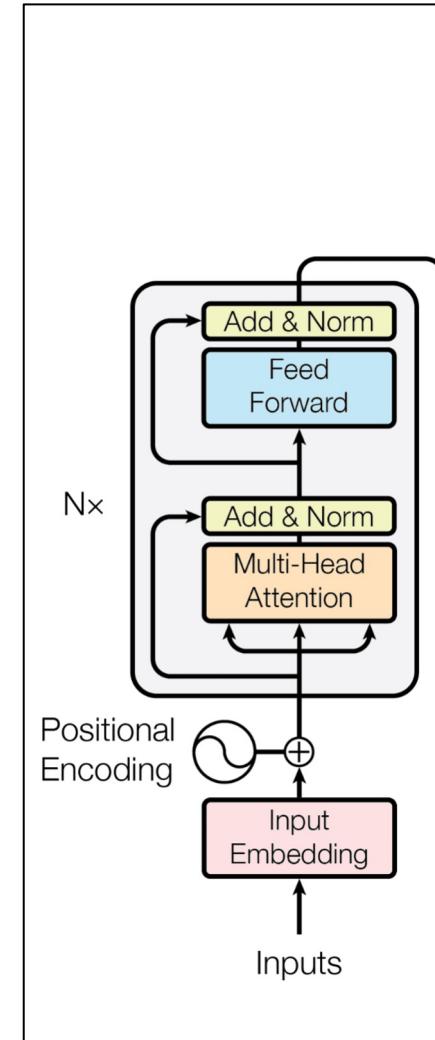


Input to Encoder_{i+1}
↑
Output from Encoder_i



Transformers

- ✓ Tokenization
- ✓ Input Embeddings
- ✓ Position Encodings
- ✓ Residuals
- ✓ Query
- ✓ Key
- ✓ Value
- ✓ Add & Norm
- ✓ Encoder
- Decoder
- ✓ Attention
- ✓ Self Attention
- ✓ Multi Head Attention
 - Masked Attention
 - Encoder Decoder Attention
 - Output Probabilities / Logits
 - Softmax
 - Encoder-Decoder models
 - Decoder only models



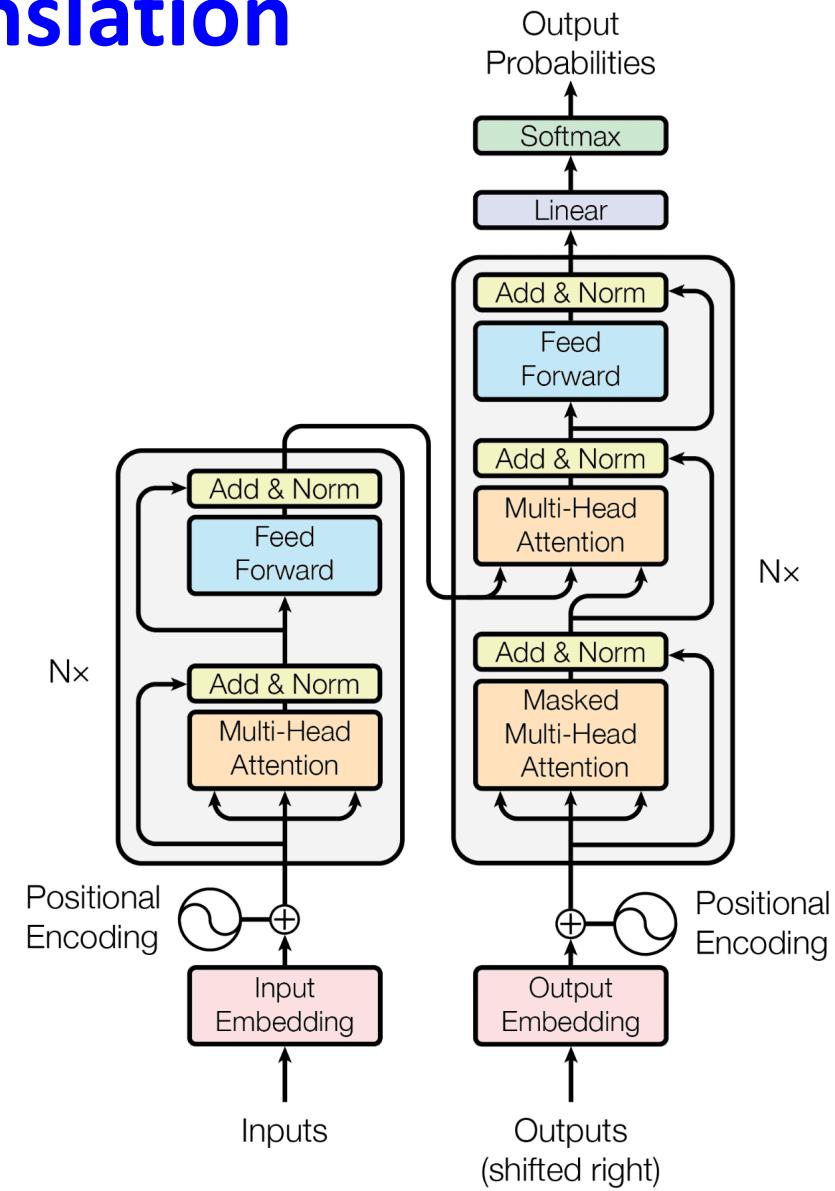
Machine Translation

Targets

Ich have einen apfel gegessen

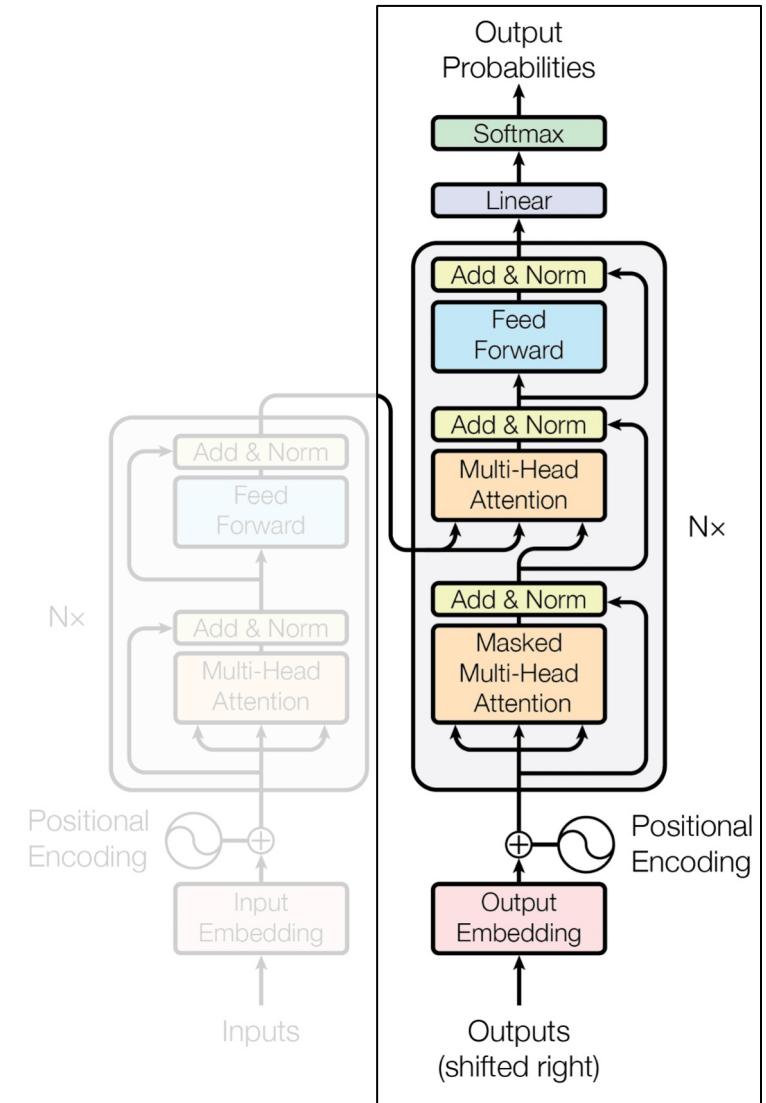
Inputs

I ate an apple

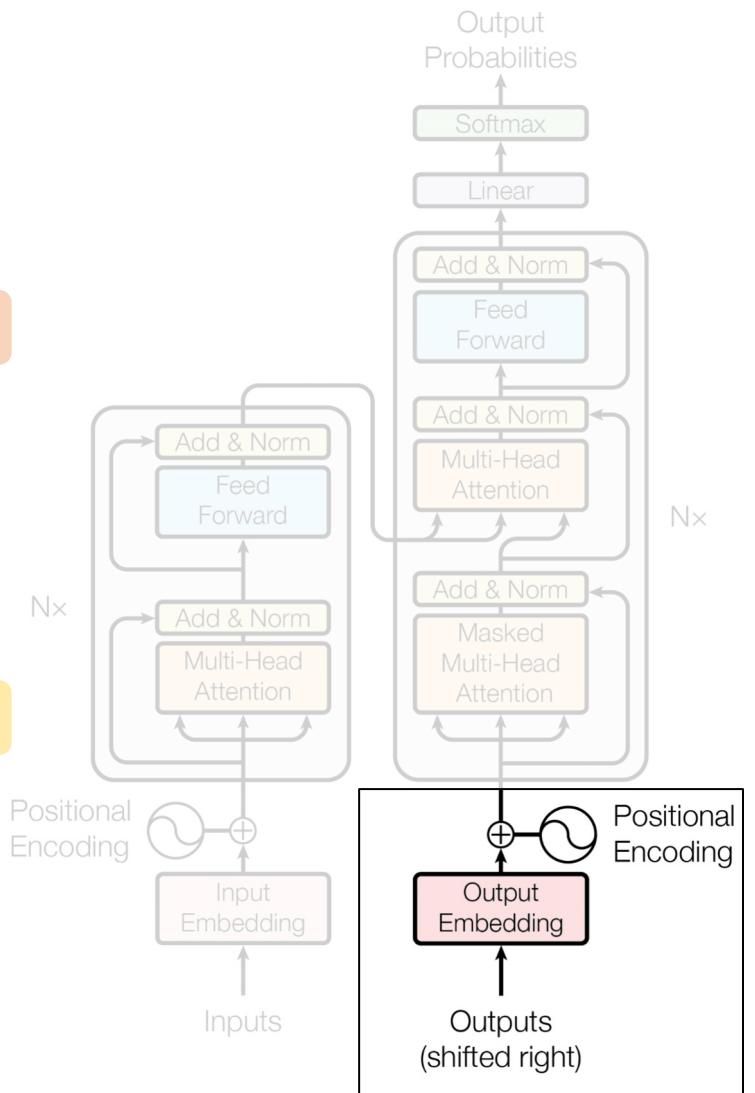
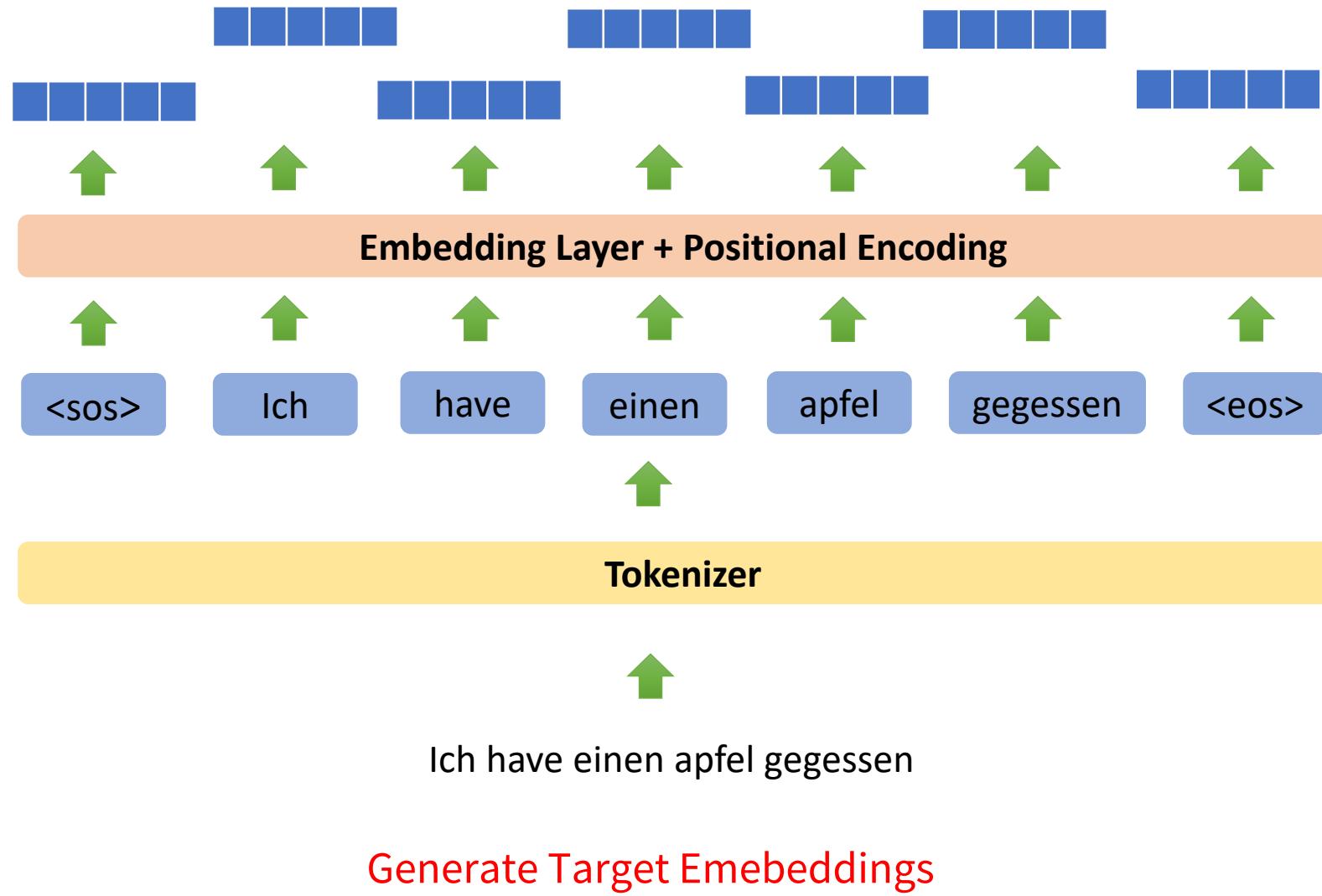


Targets

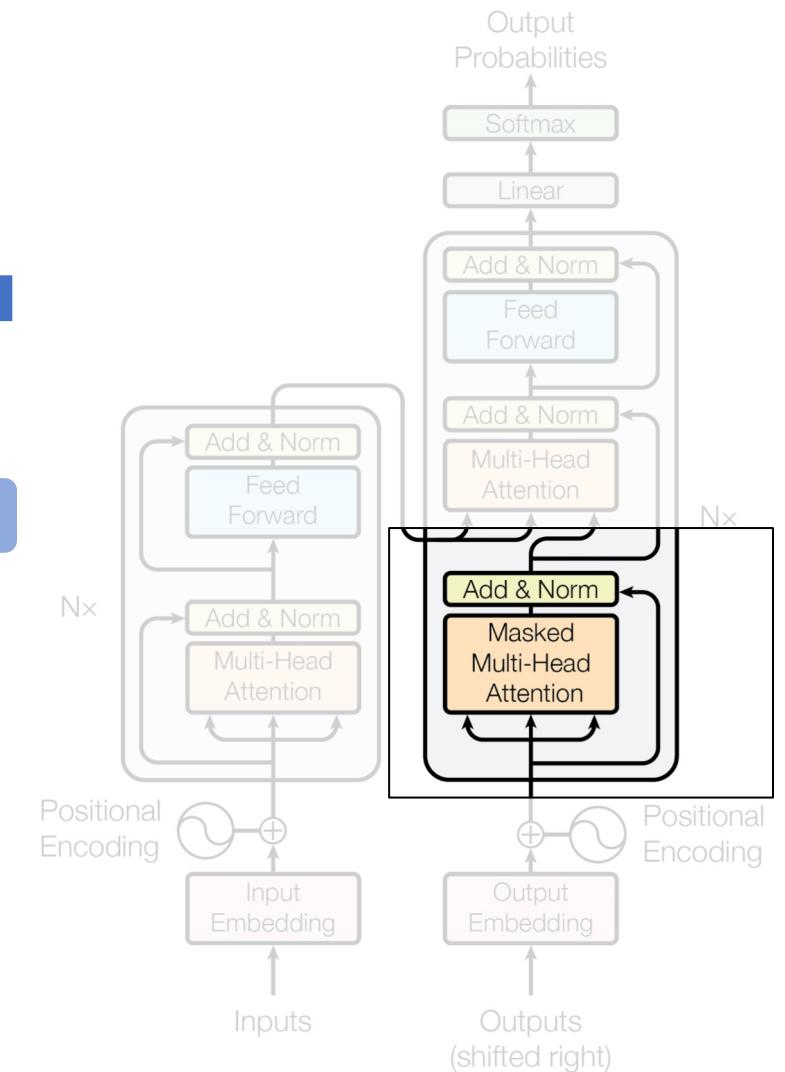
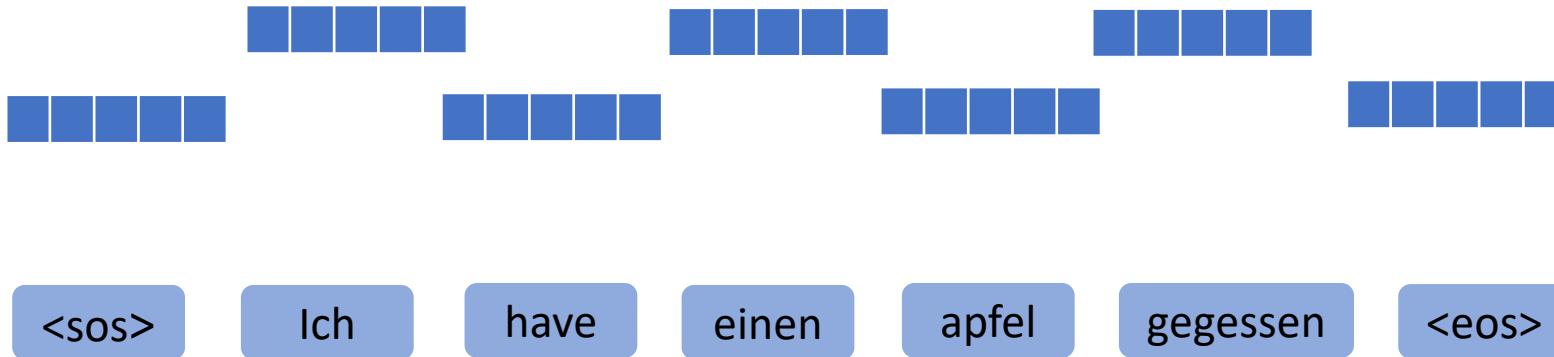
Targets
Ich have einen apfel gegessen



Targets

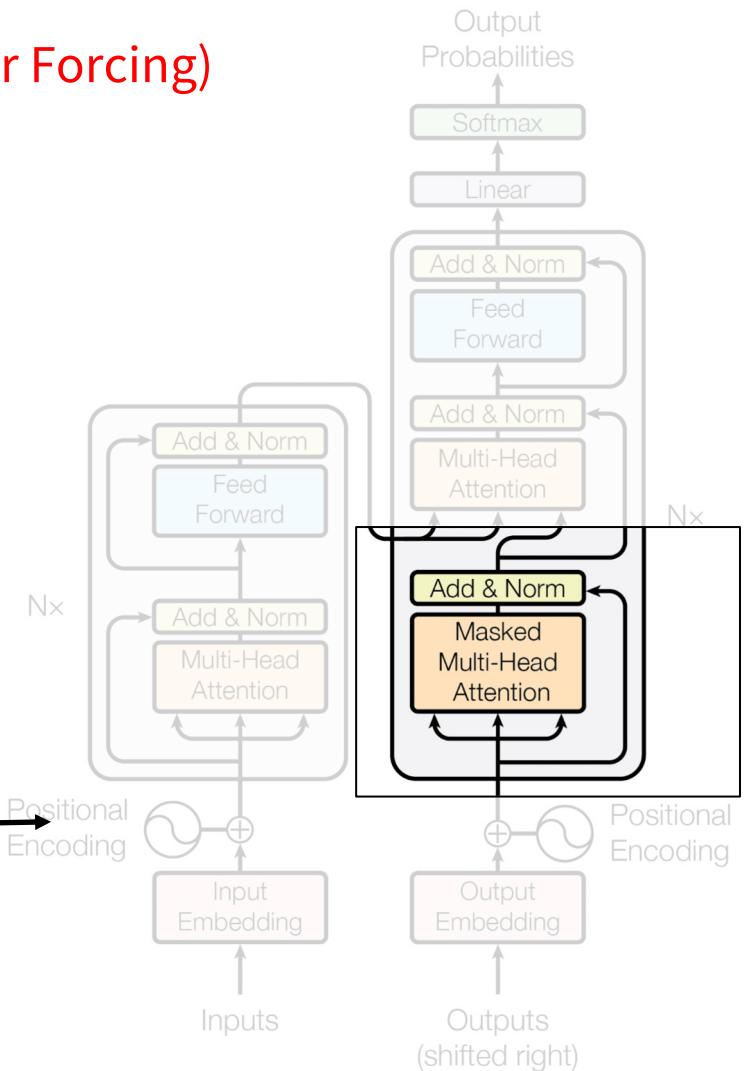
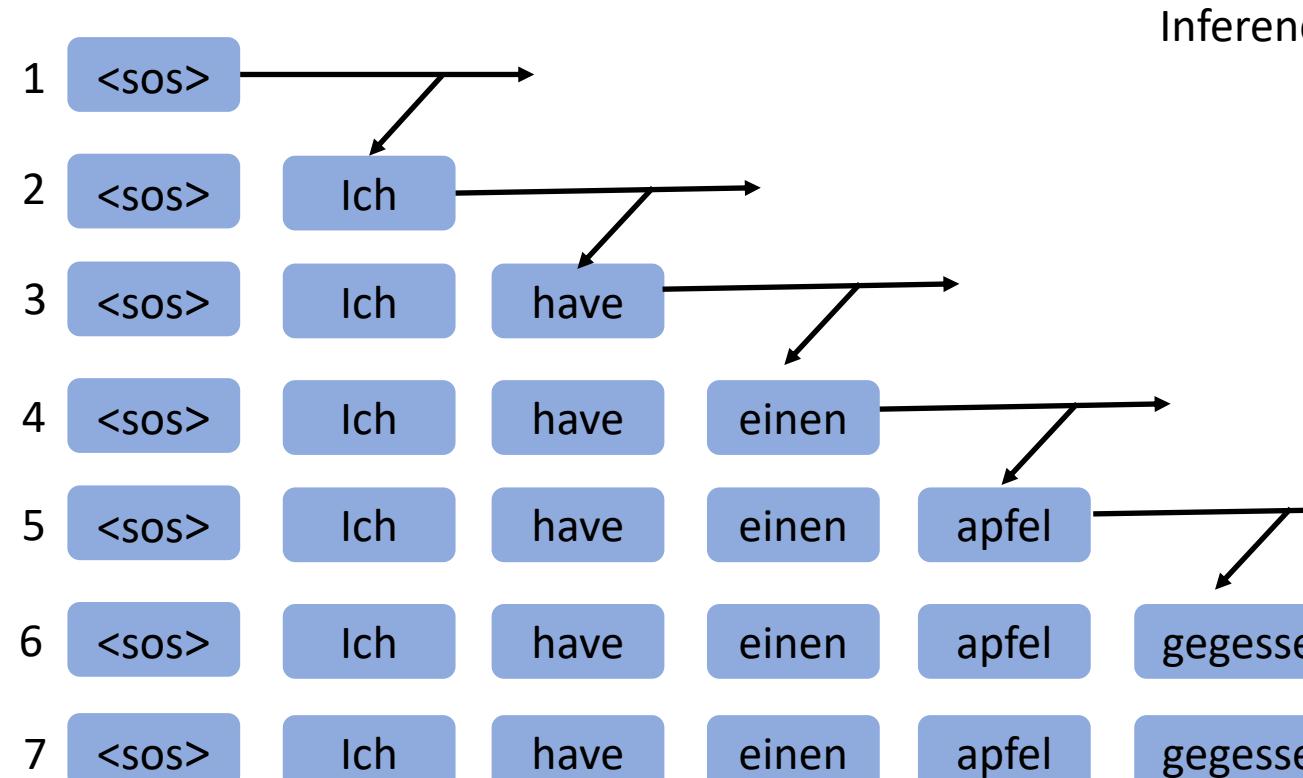


Masked Multi Head Attention



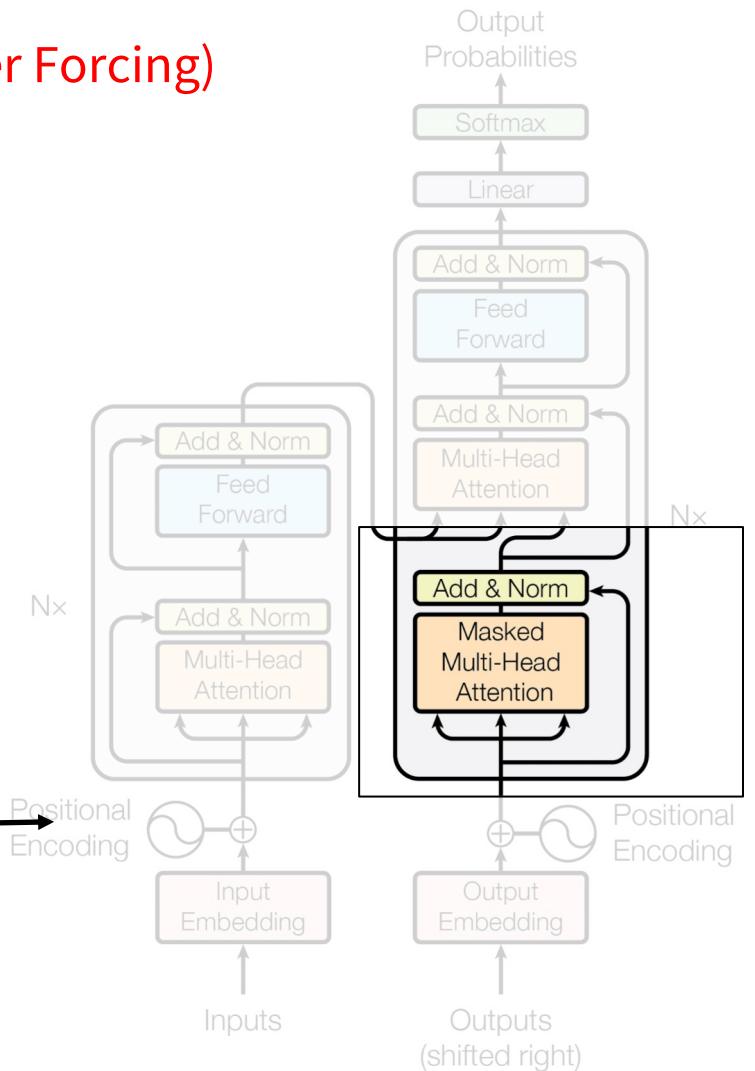
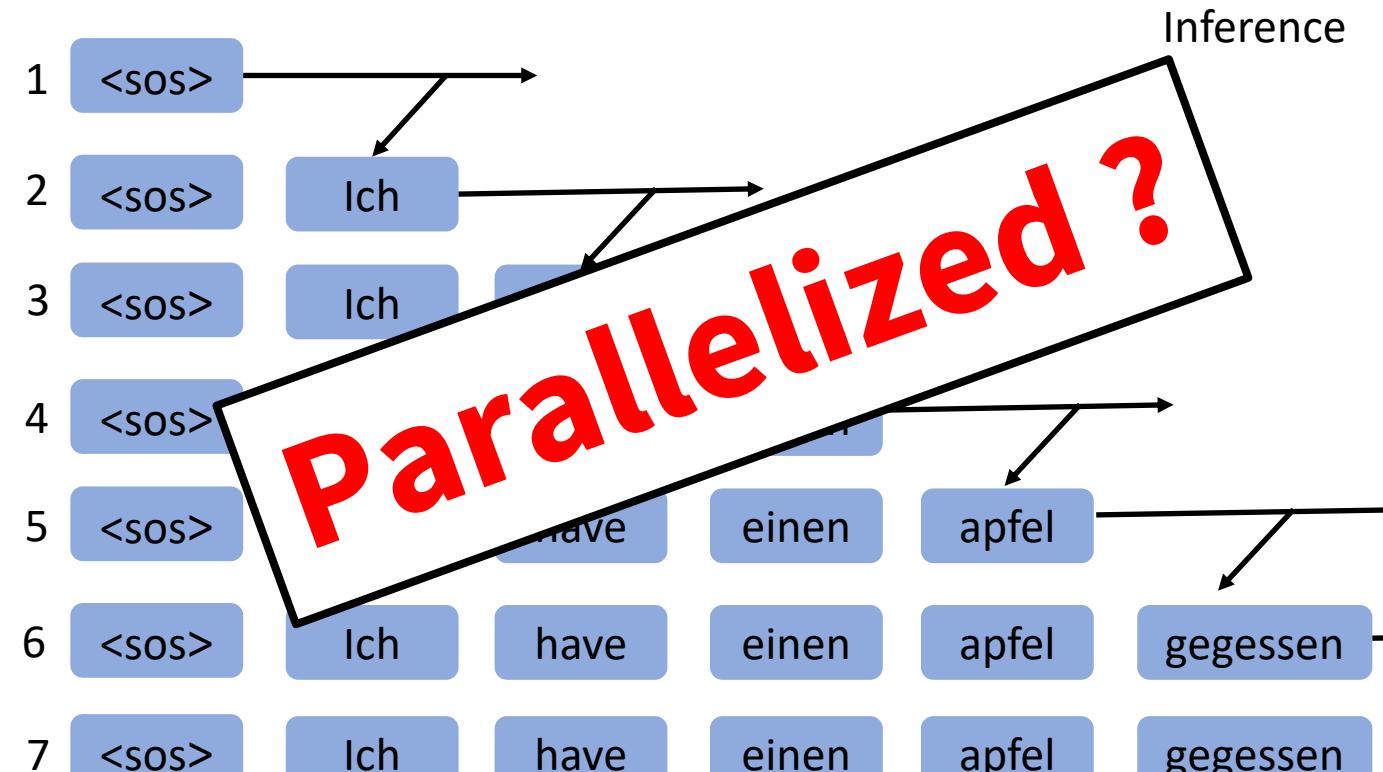
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)



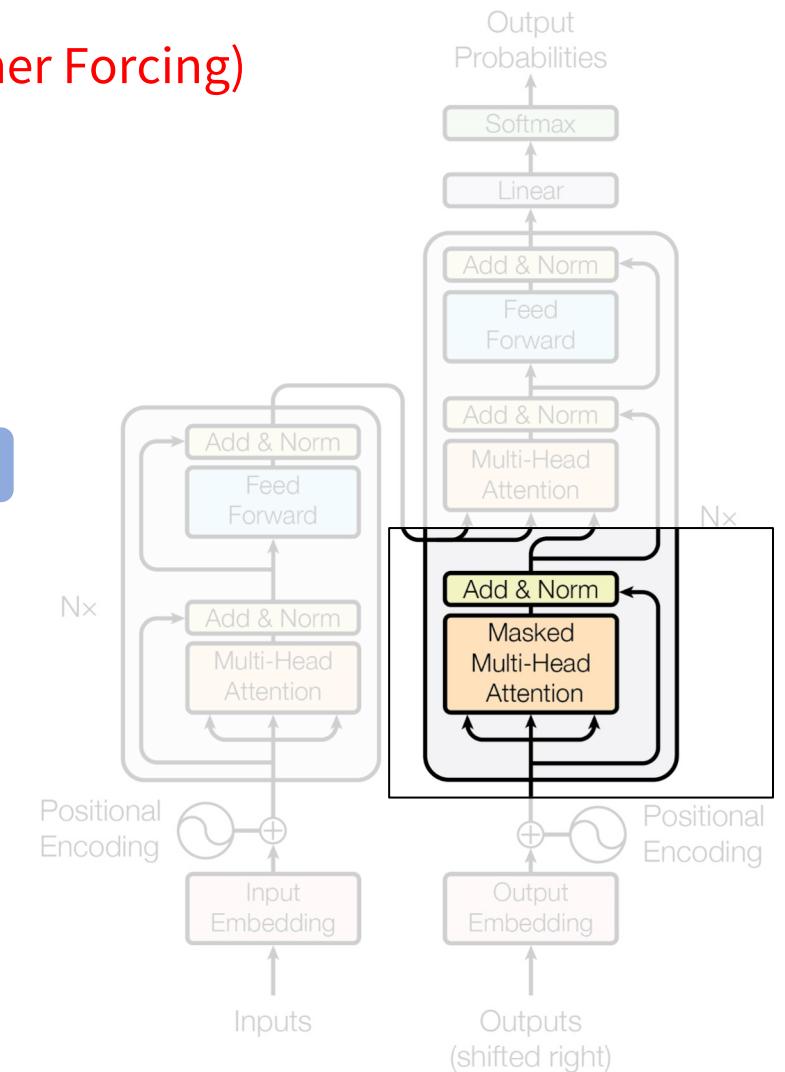
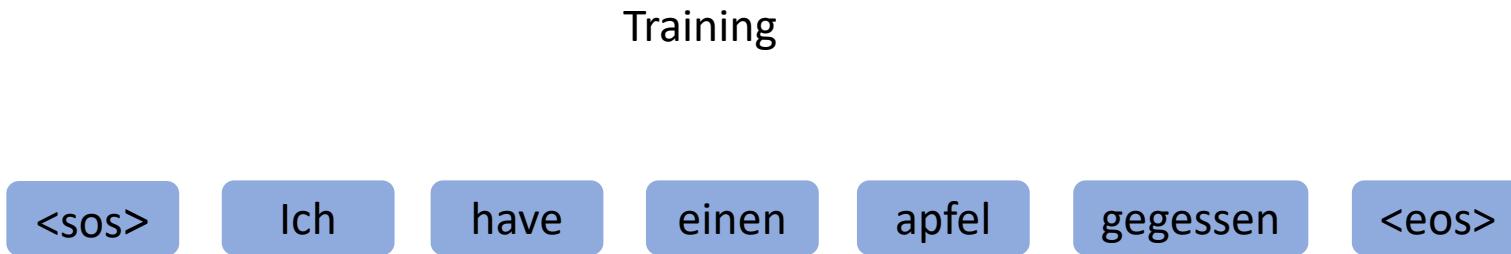
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)



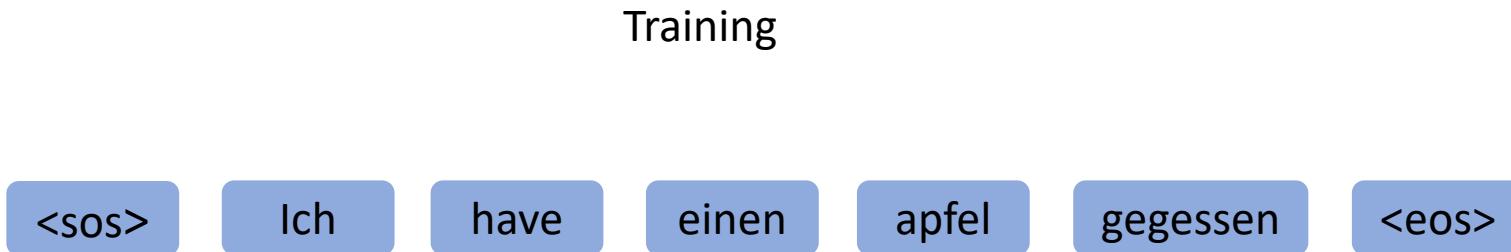
Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

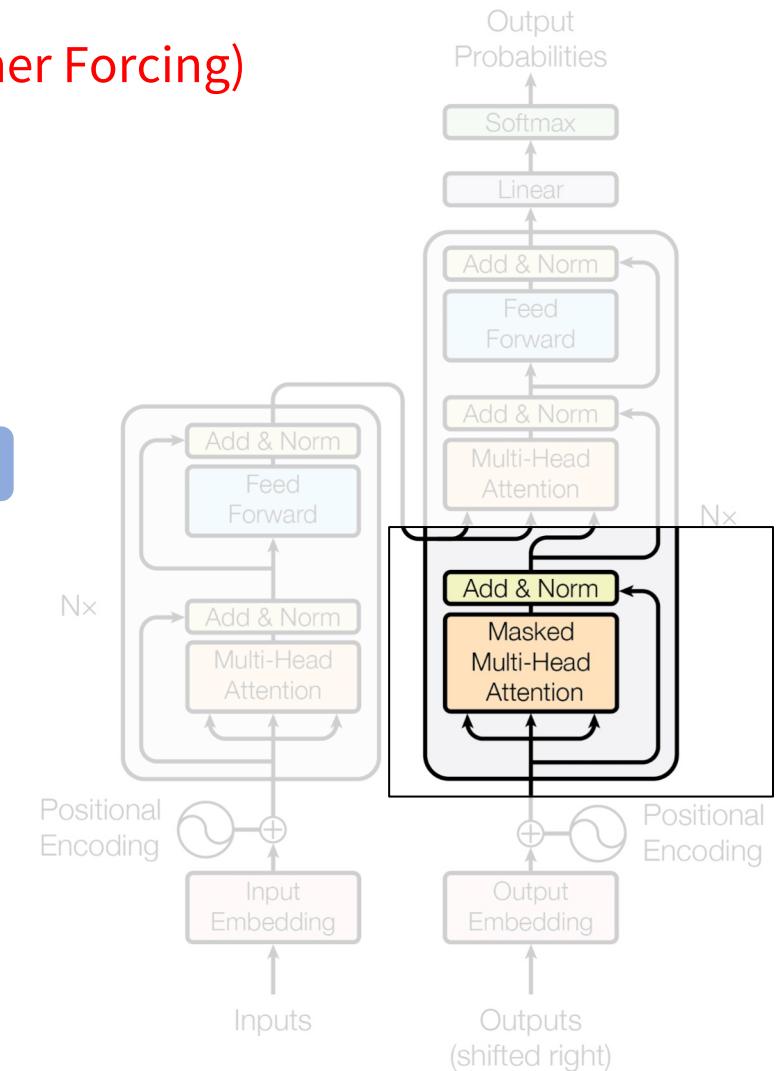


Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)



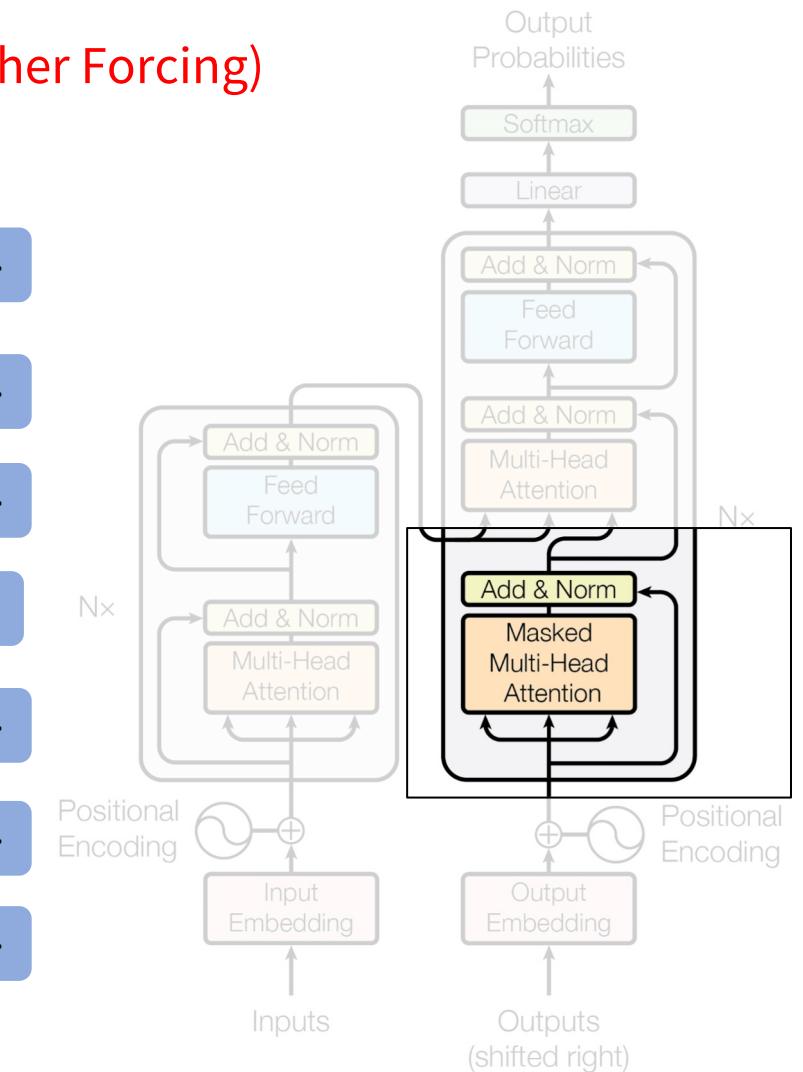
*Outputs at time T should only pay attention to outputs
until time $T-1$*



Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

1	<sos>	Ich	have	einen	apfel	gegessen	<eos>
2	<sos>	Ich	have	einen	apfel	gegessen	<eos>
3	<sos>	Ich	have	einen	apfel	gegessen	<eos>
4	<sos>	Ich	have	einen	apfel	gegessen	<eos>
5	<sos>	Ich	have	einen	apfel	gegessen	<eos>
6	<sos>	Ich	have	einen	apfel	gegessen	<eos>
7	<sos>	Ich	have	einen	apfel	gegessen	<eos>

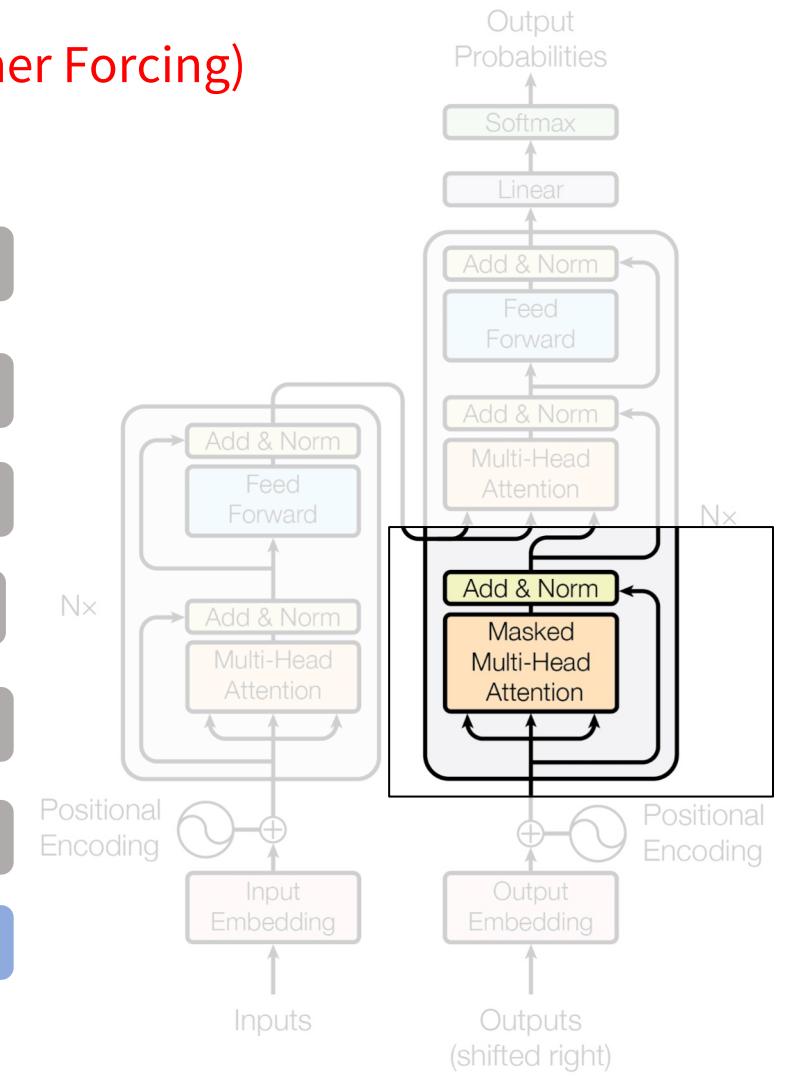


Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

1	<sos>	Ich	have	einen	apfel	gegessen	<eos>
2	<sos>	Ich	have	einen	apfel	gegessen	<eos>
3	<sos>	Ich	have	einen	apfel	gegessen	<eos>
4	<sos>	Ich	have	einen	apfel	gegessen	<eos>
5	<sos>	Ich	have	einen	apfel	gegessen	<eos>
6	<sos>	Ich	have	einen	apfel	gegessen	<eos>
7	<sos>	Ich	have	einen	apfel	gegessen	<eos>

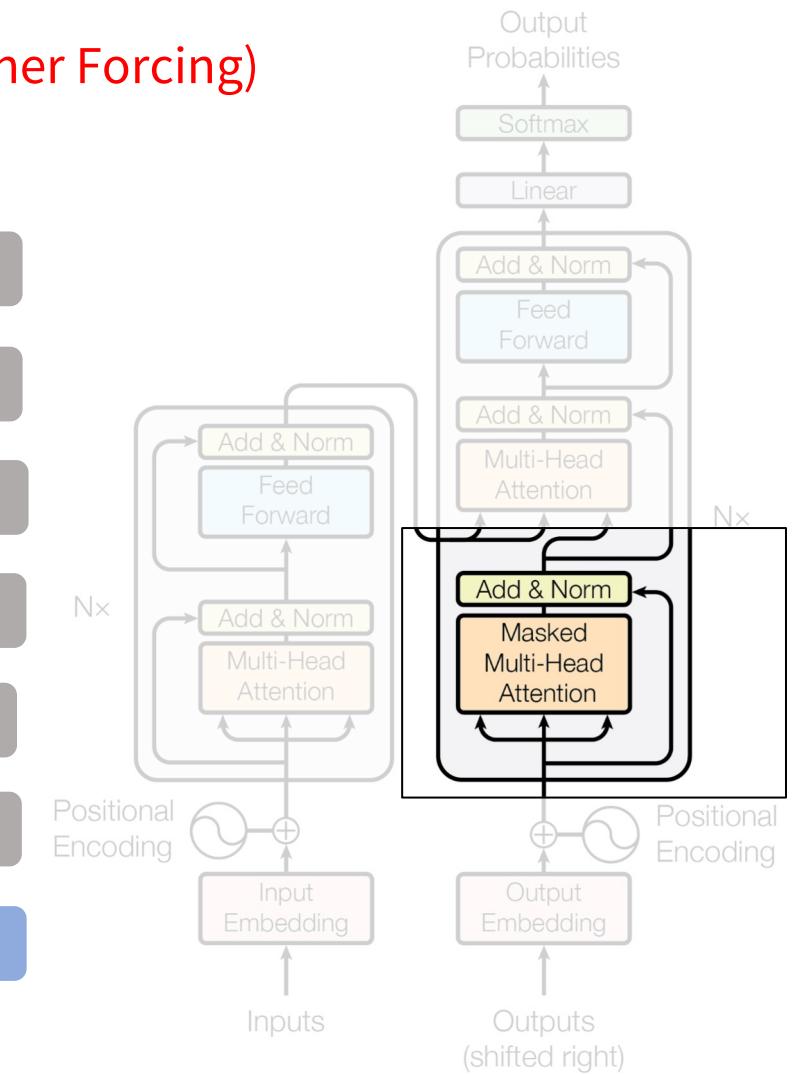
Mask the available attention values ?



Masked Multi Head Attention

Decoding step by step (using Teacher Forcing)

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

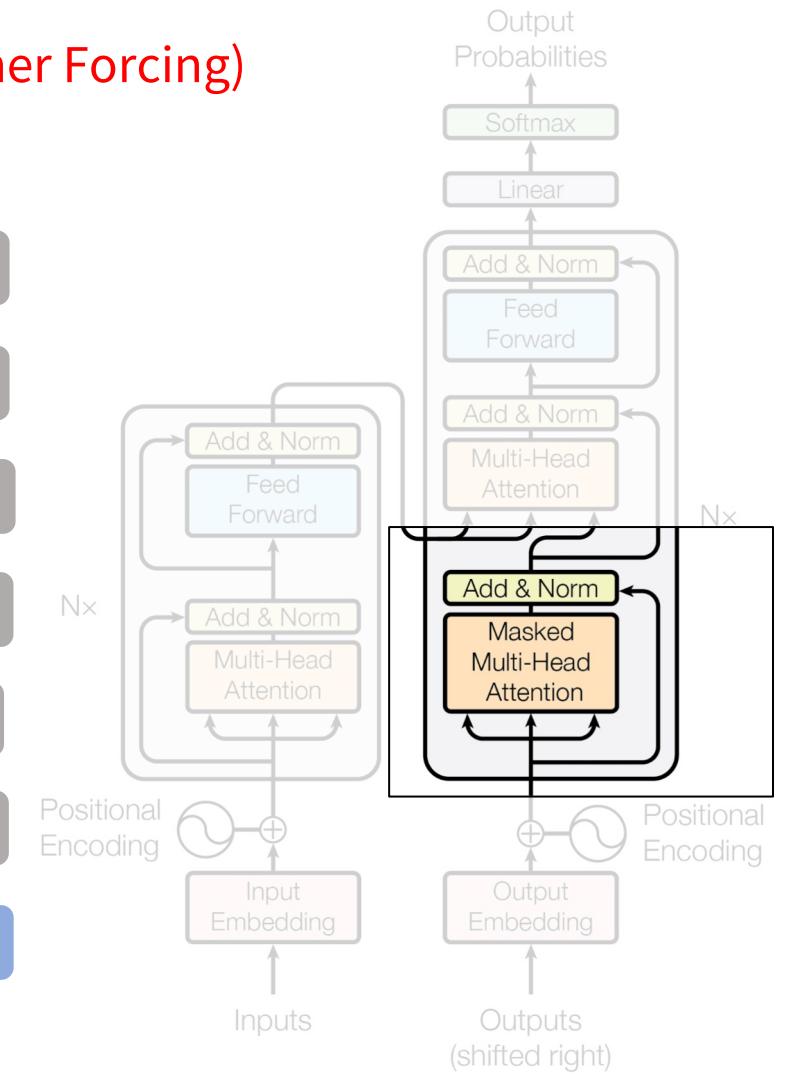


Masked Multi Head Attention

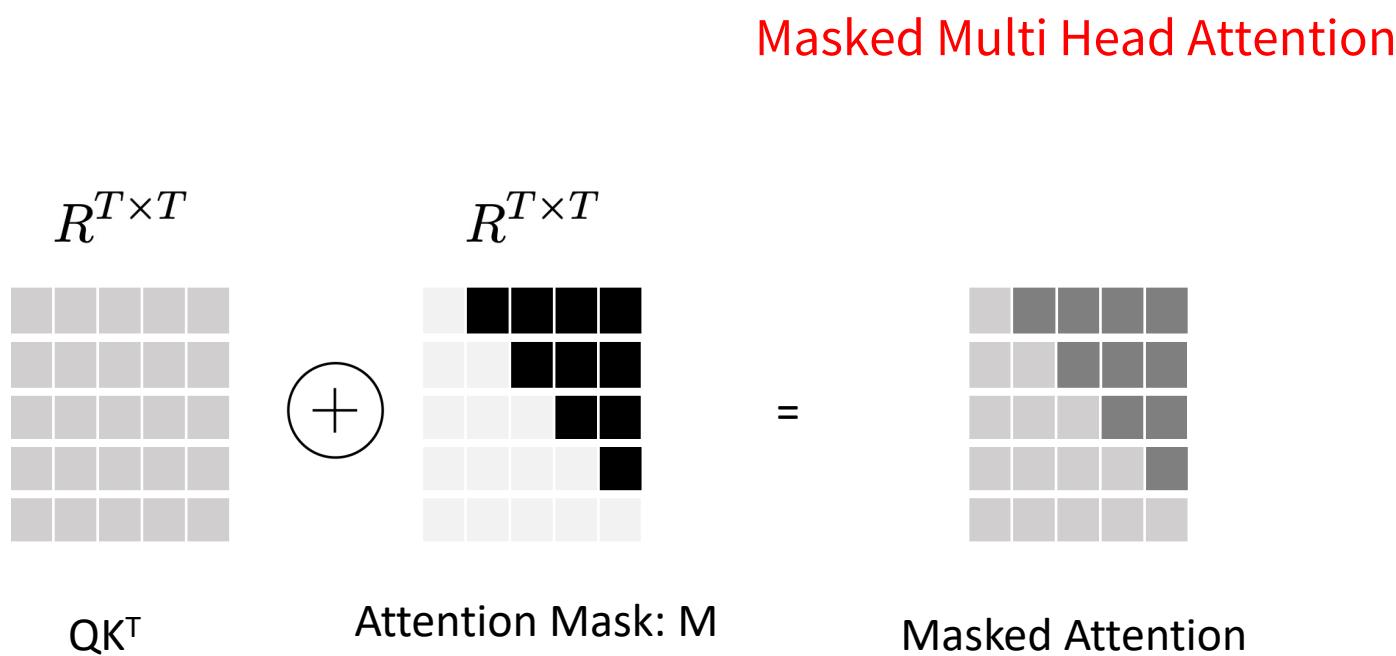
Decoding step by step (using Teacher Forcing)

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

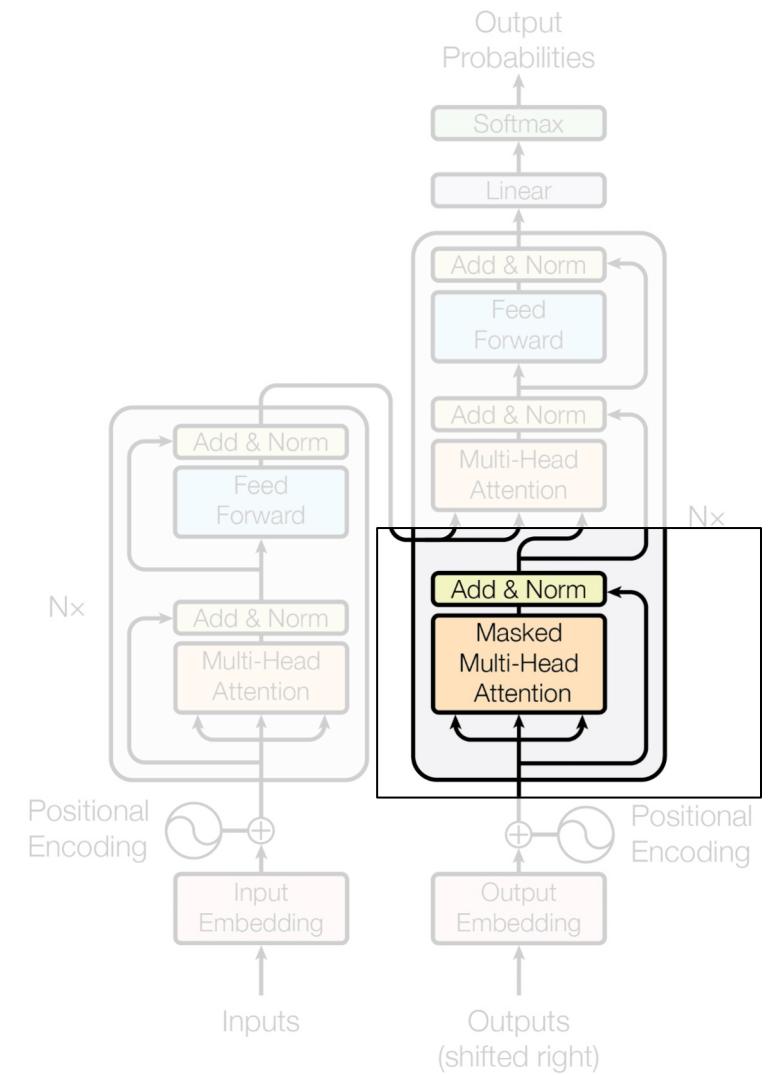
Softmax -> - ∞ -> 0



Masked Multi Head 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 Values

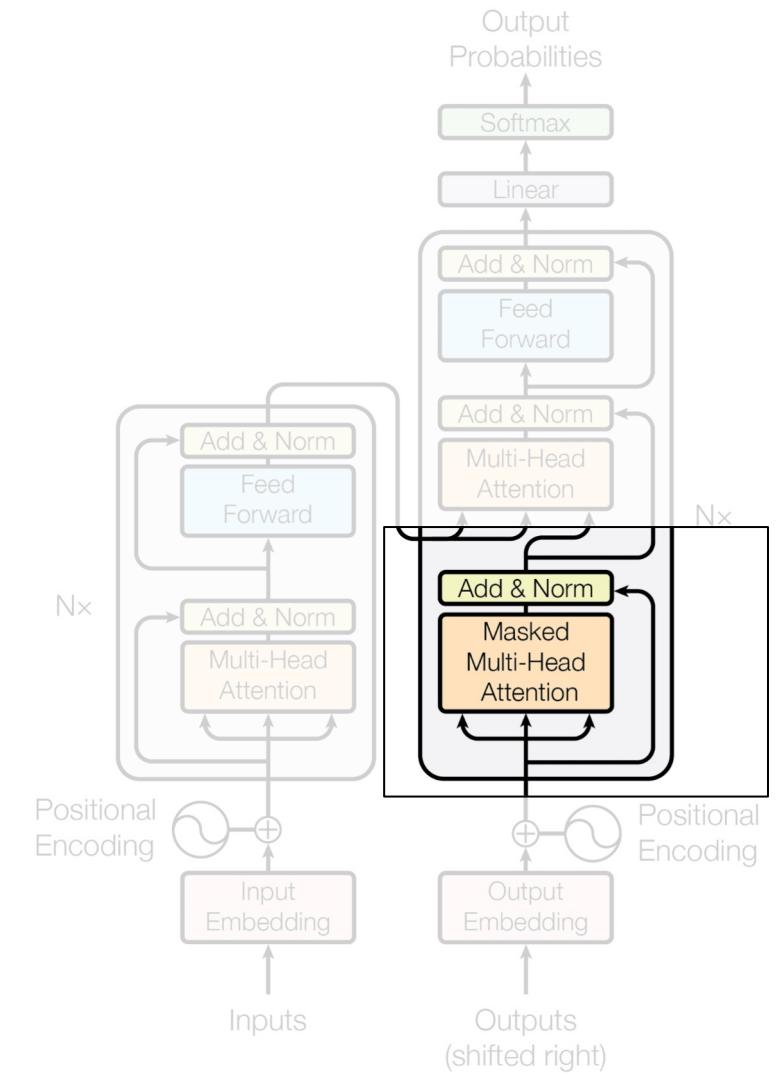
$R^{T \times T}$

$R^{T \times d_h}$

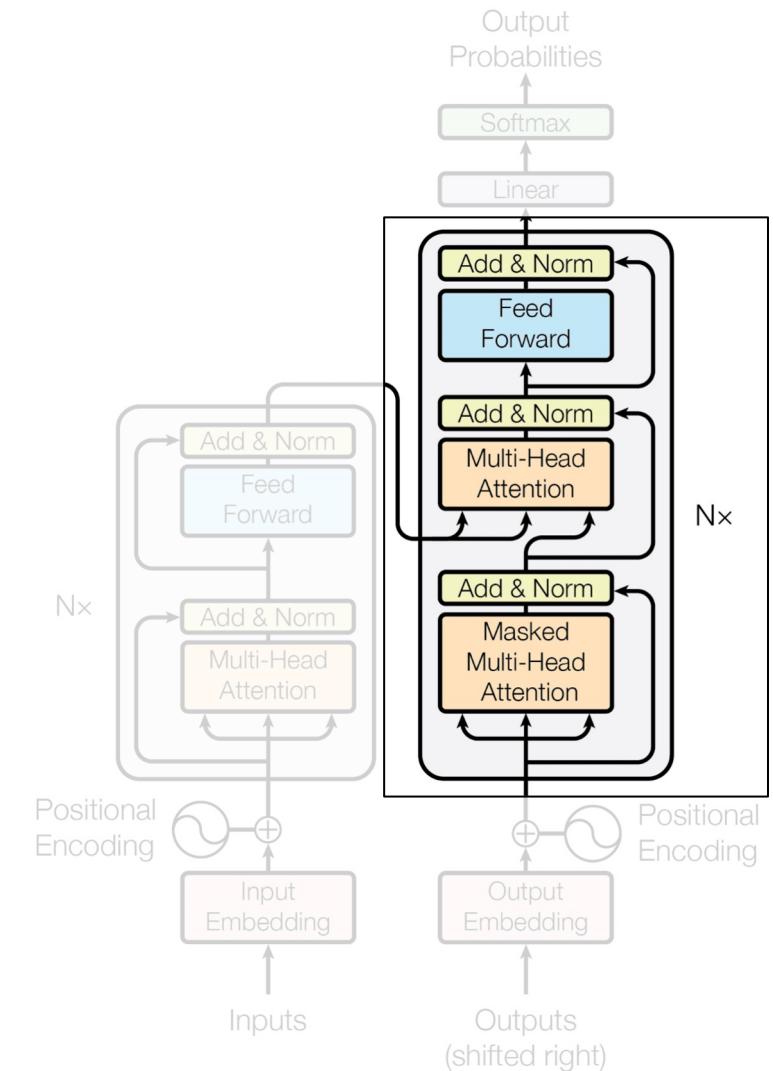
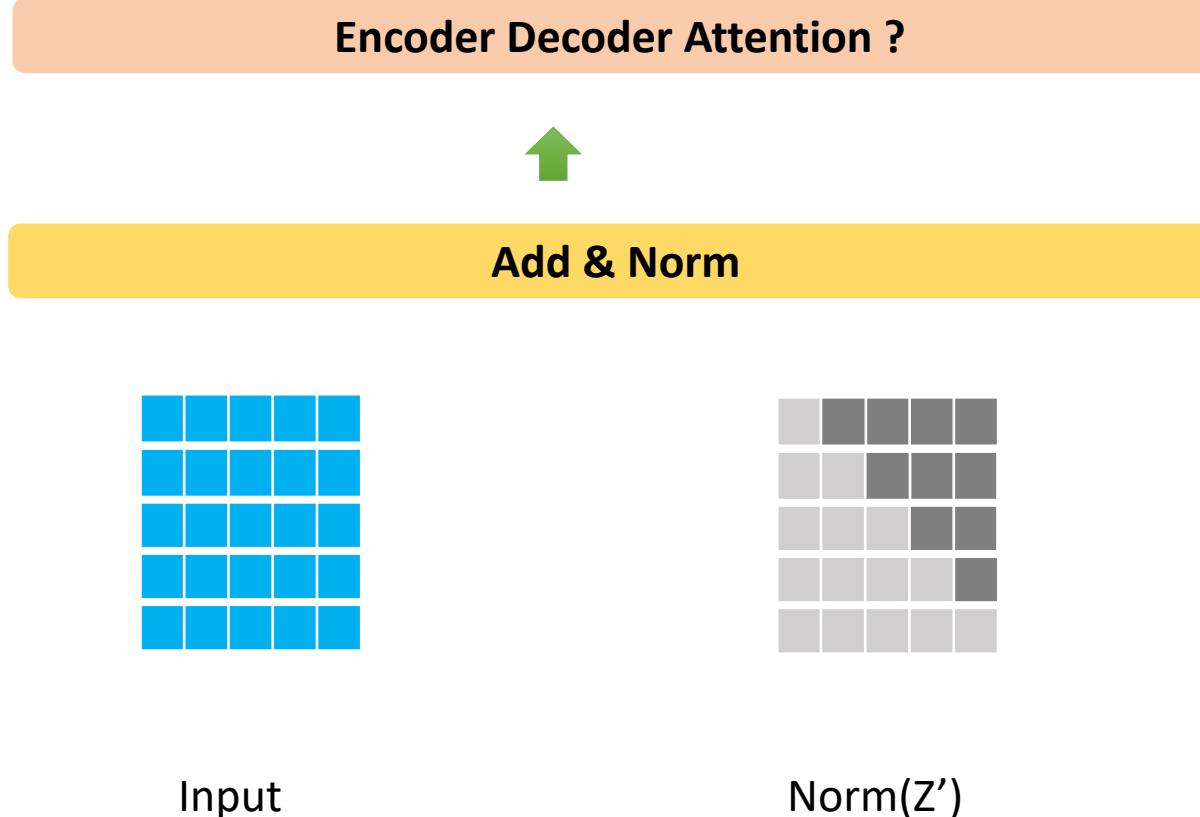
Masked Attention

Values

Masked Multi Head Attention : Z'

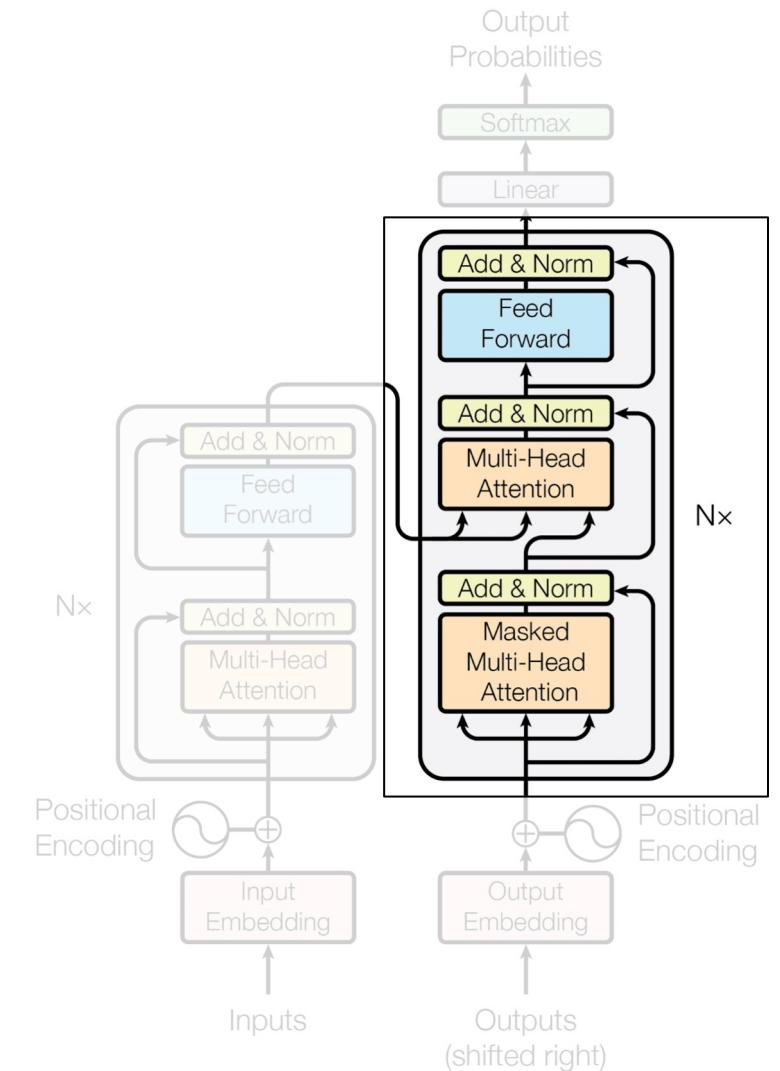


Encoder Decoder Attention



Encoder Decoder Attention

Encoder Decoder Attention ?



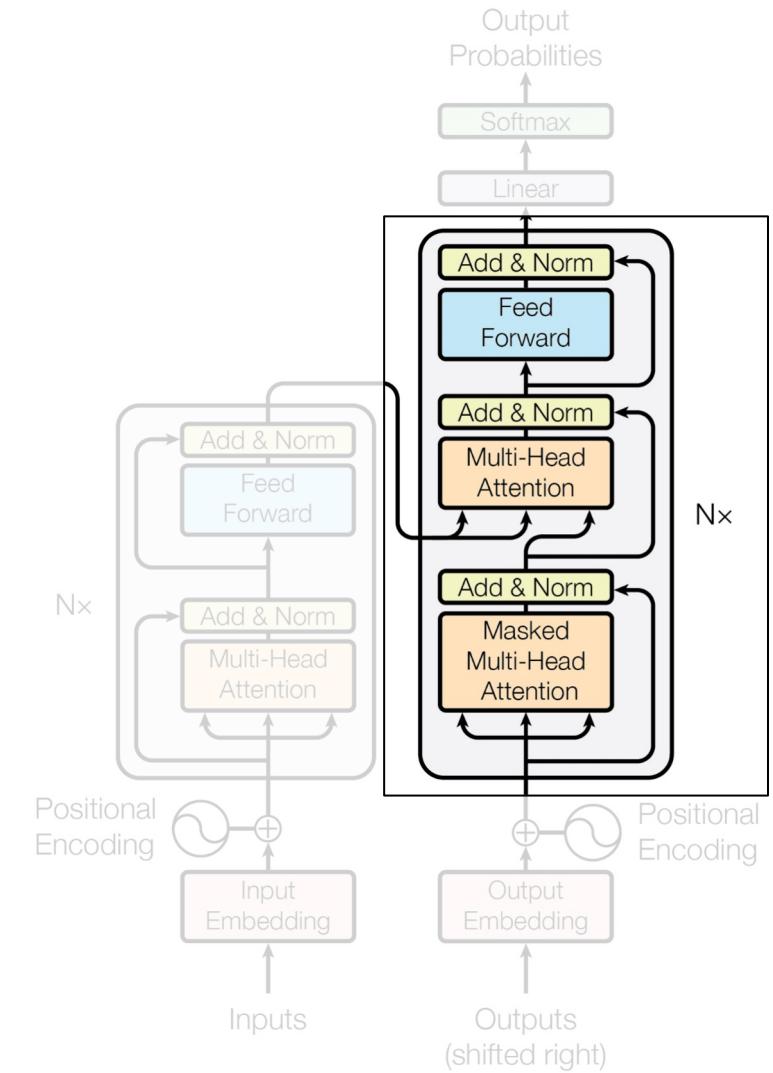
Encoder Decoder Attention

Encoder Self Attention

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

Decoder Masked Self Attention

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



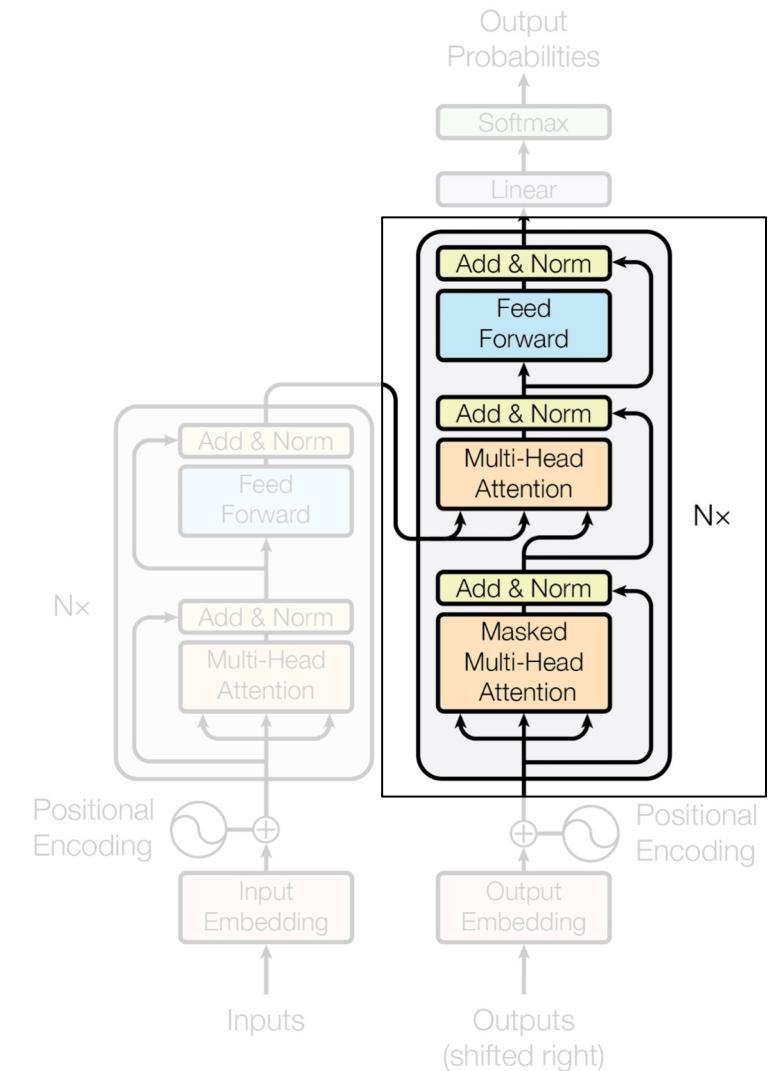
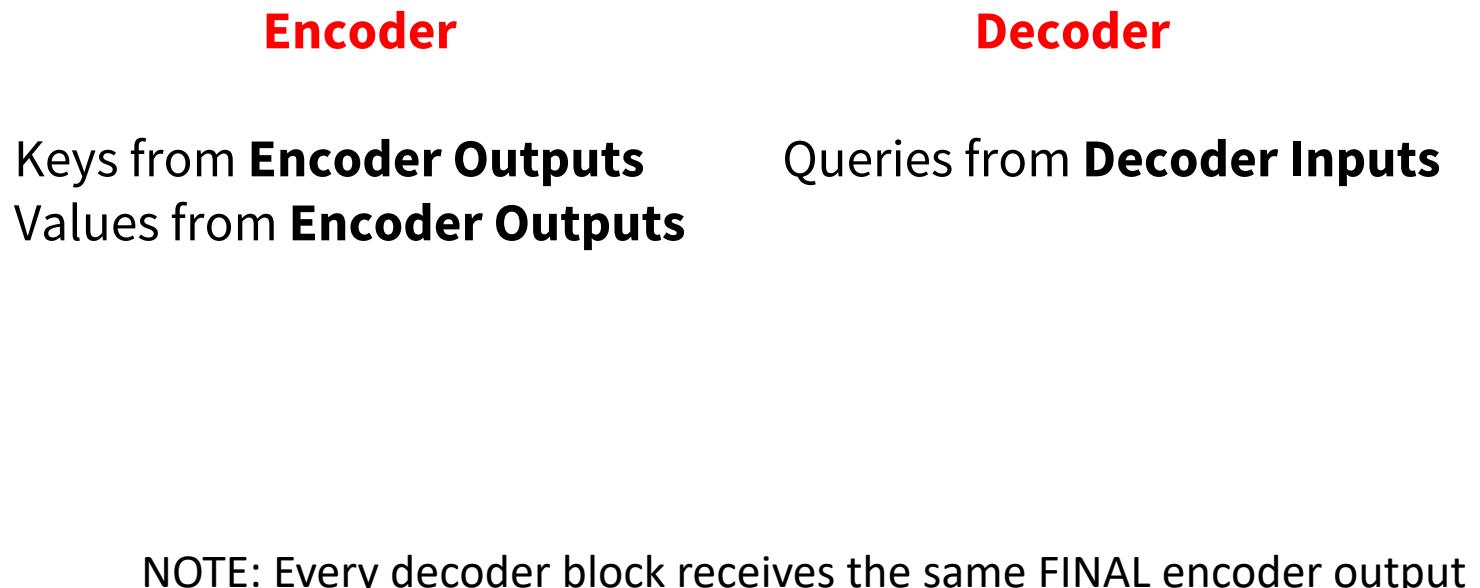
Attention

```
{Query: "Order details of order_104"}  
  
{Query: "Order details of order_106"}
```

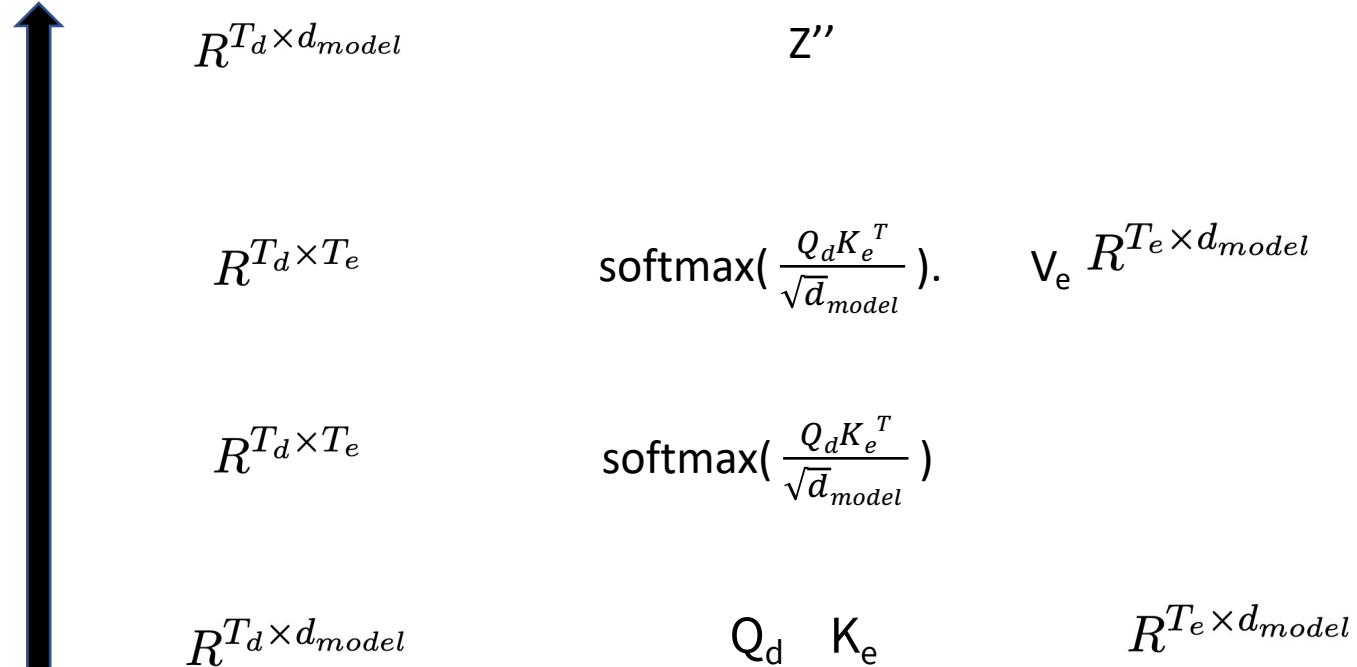
{Key, Value store}

```
{"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", ...}}
```

Encoder Decoder Attention



Encoder Decoder Attention



$$\text{softmax}\left(\frac{Q_d K_e^T}{\sqrt{d_{model}}}\right).$$

$$\text{softmax}\left(\frac{Q_d K_e^T}{\sqrt{d_{model}}}\right)$$

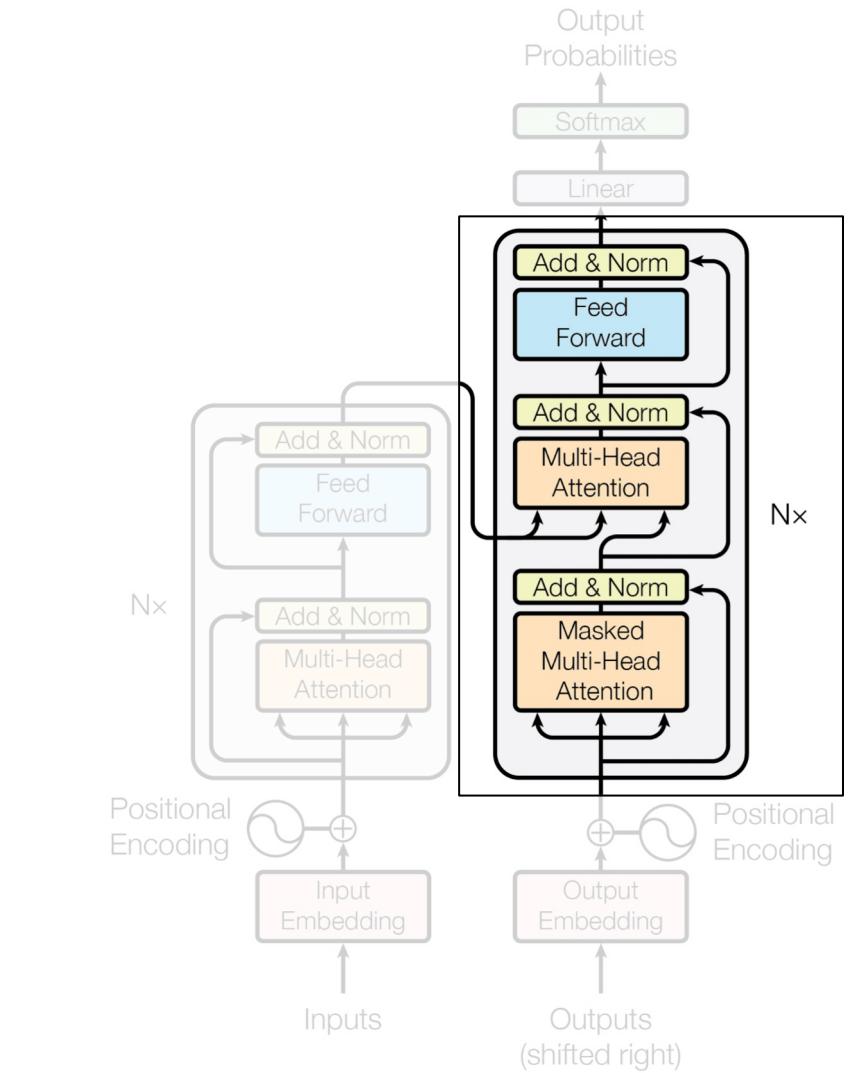
$$Q_d \quad K_e$$

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

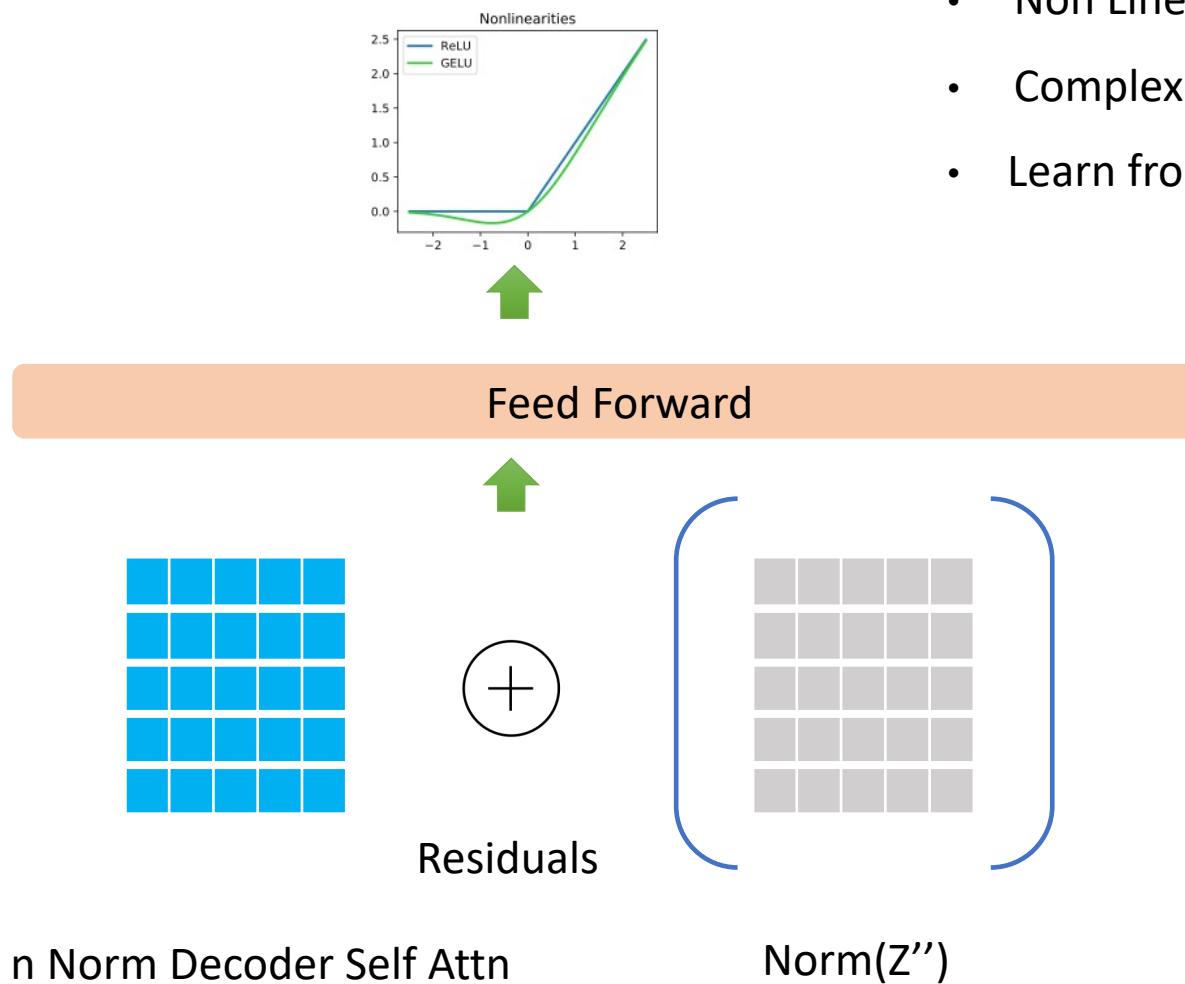
Z''

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

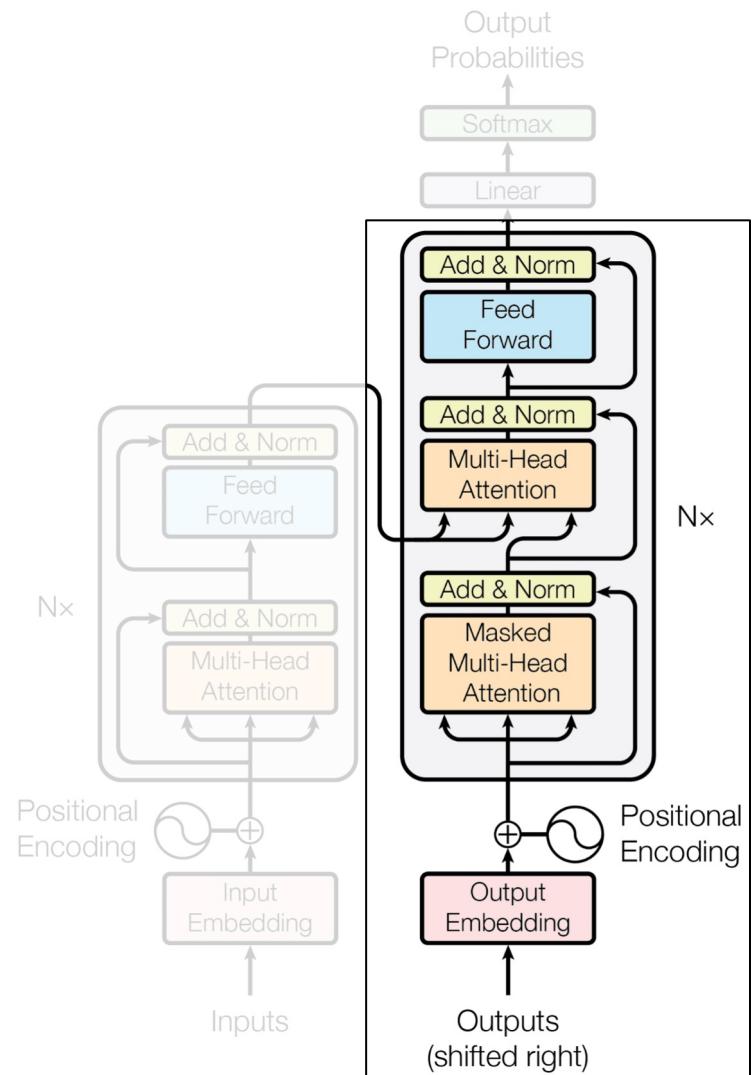
$$V_e \quad R^{T_e \times d_{model}}$$



Encoder Decoder Attention

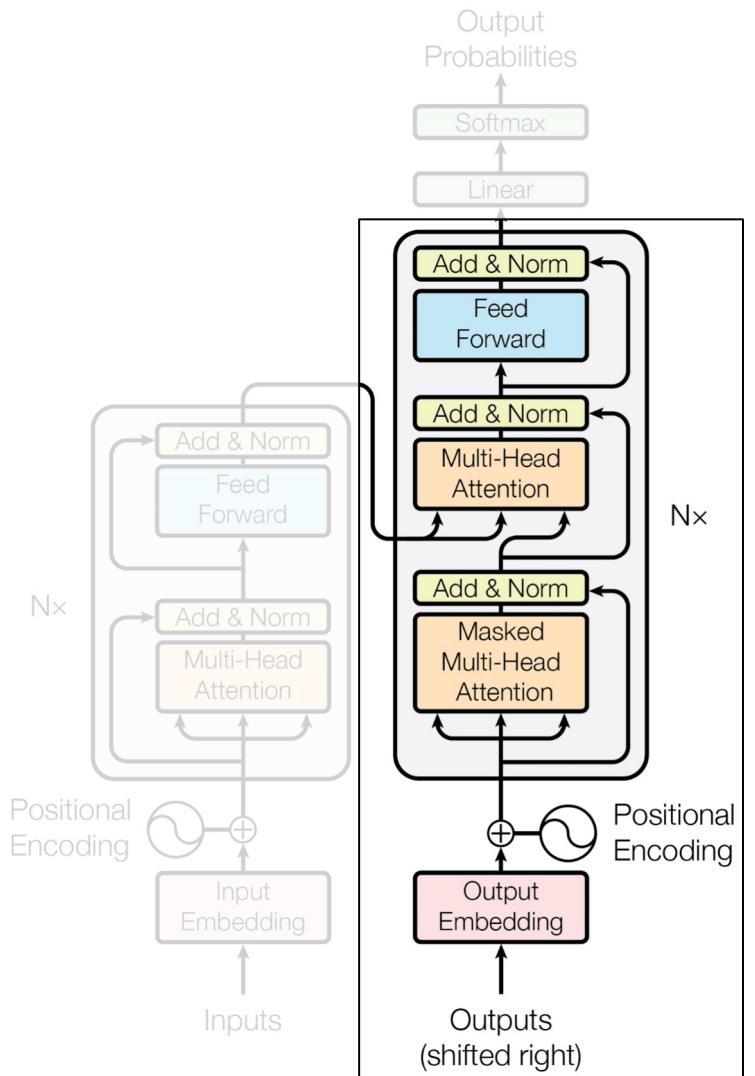


- Non Linearity
- Complex Relationships
- Learn from each other

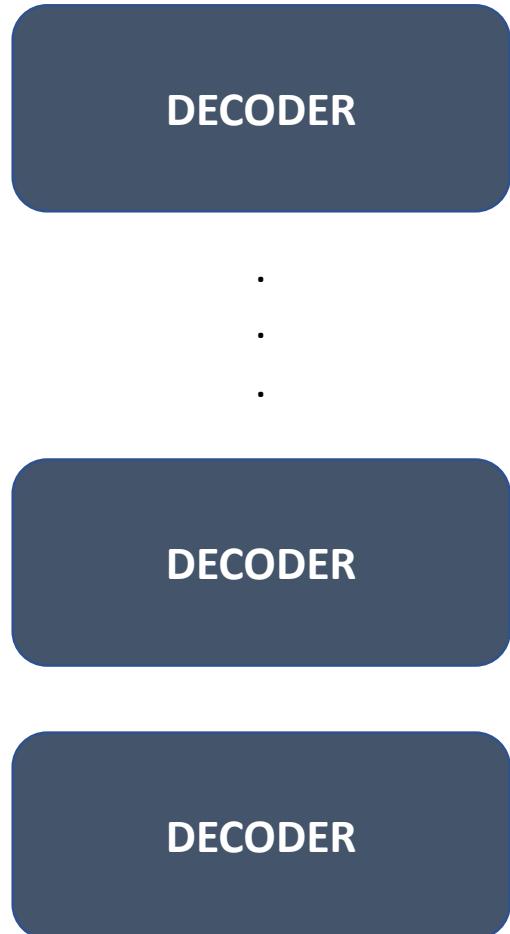


Decoder

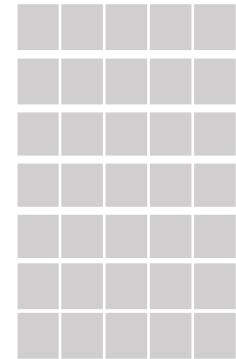
DECODER



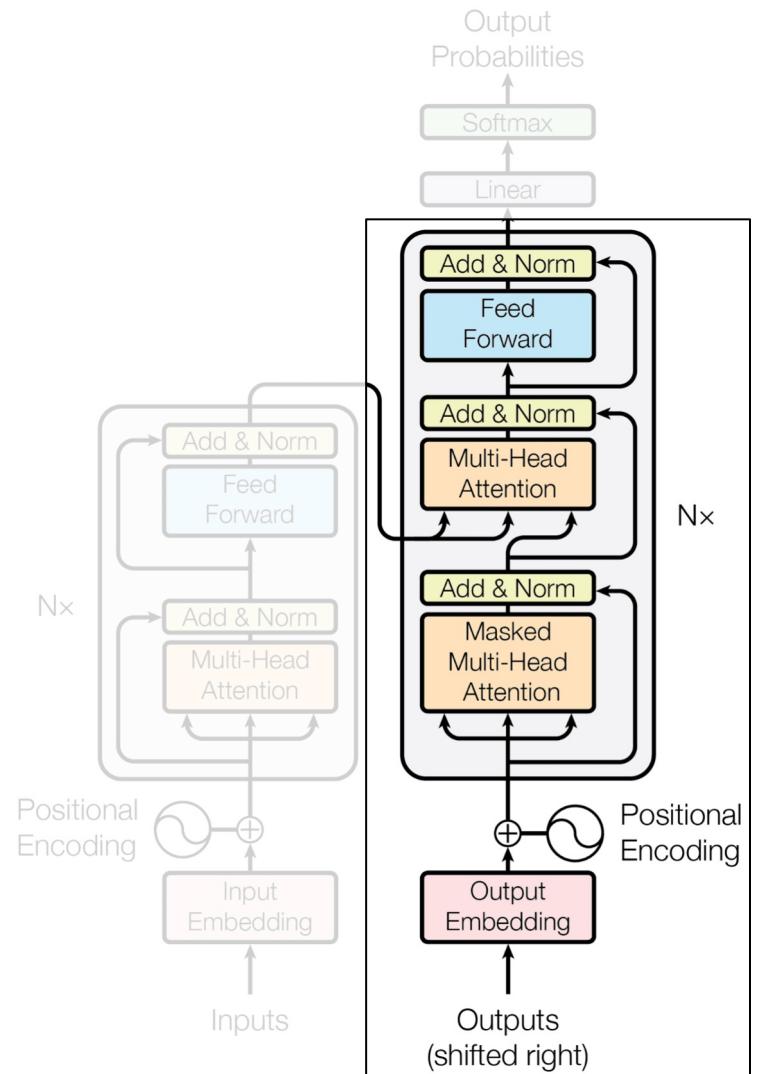
Decoder



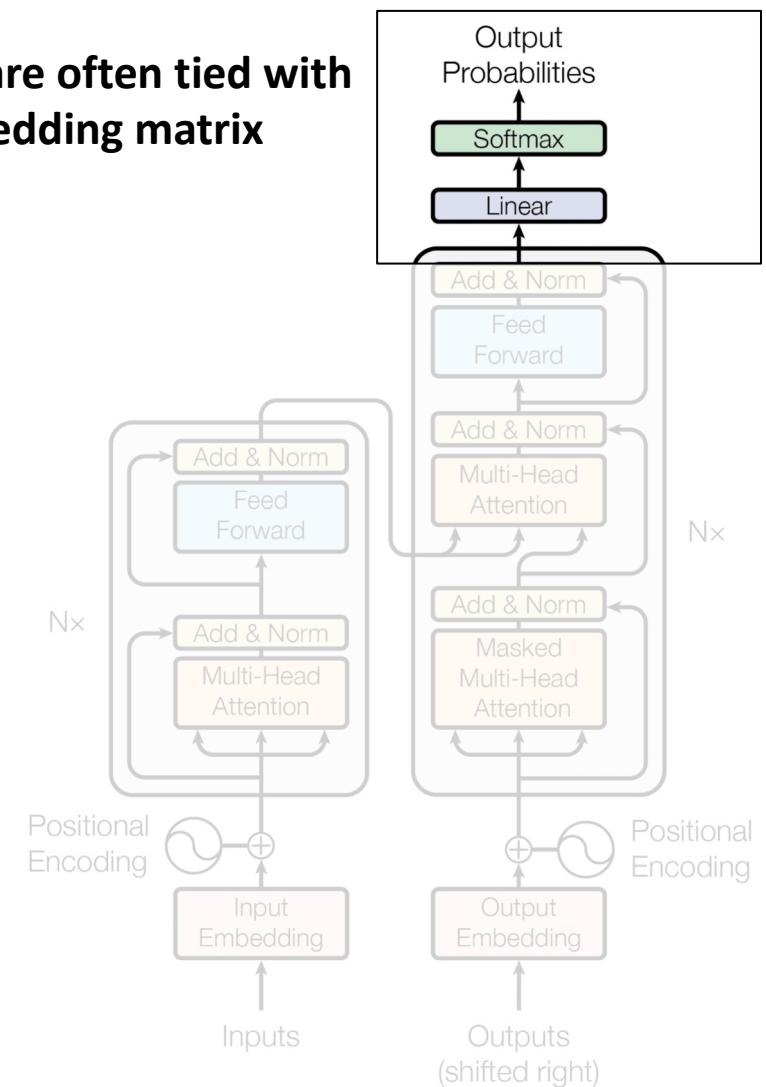
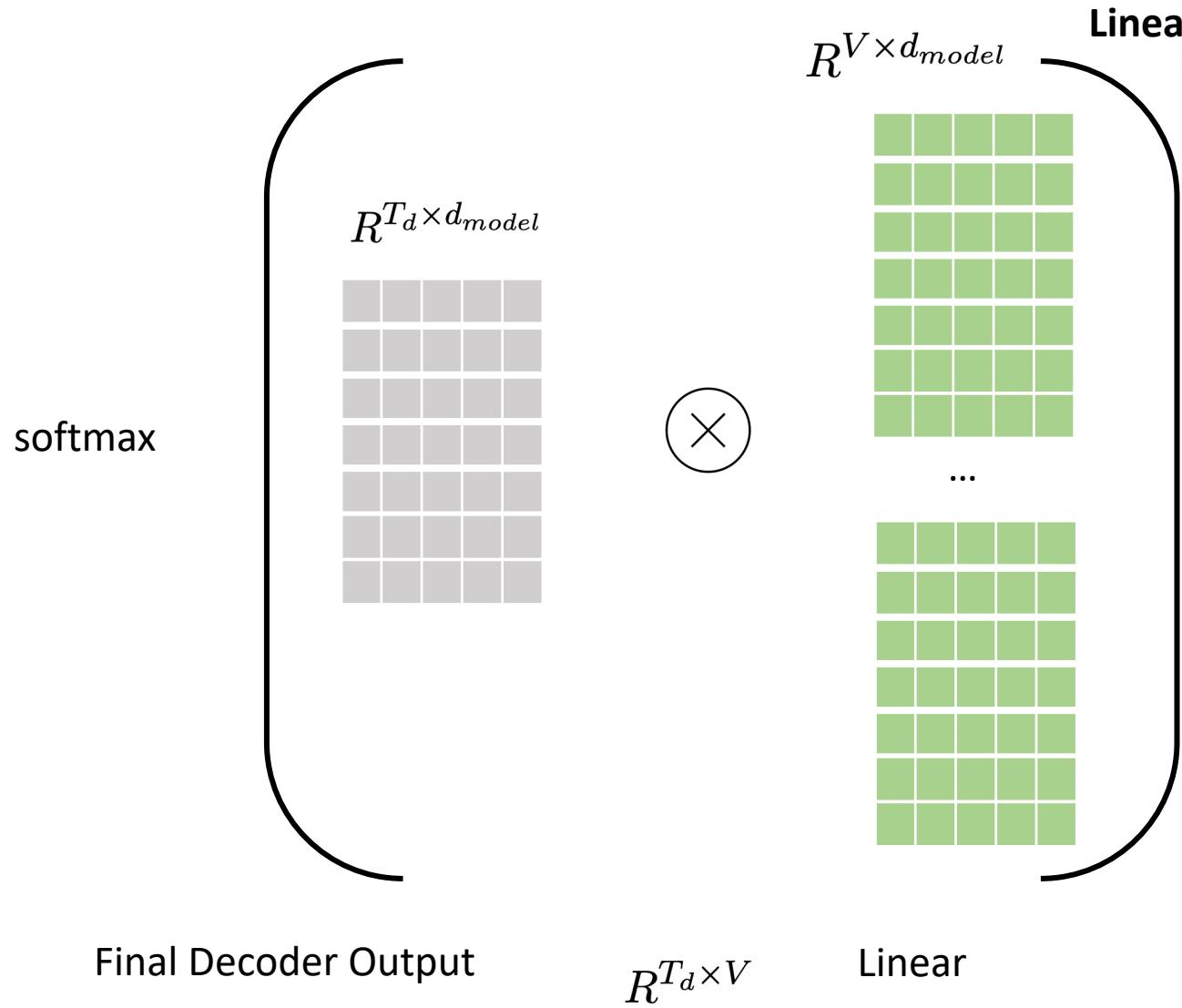
$$R^{T_d \times d_{model}}$$



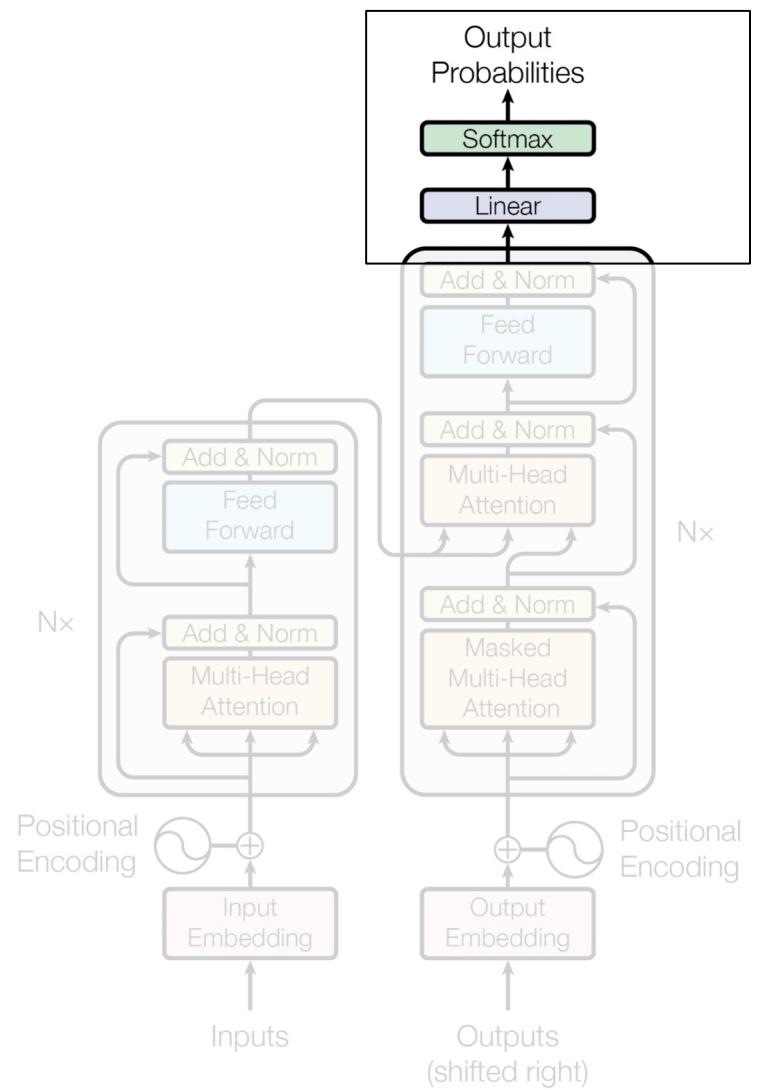
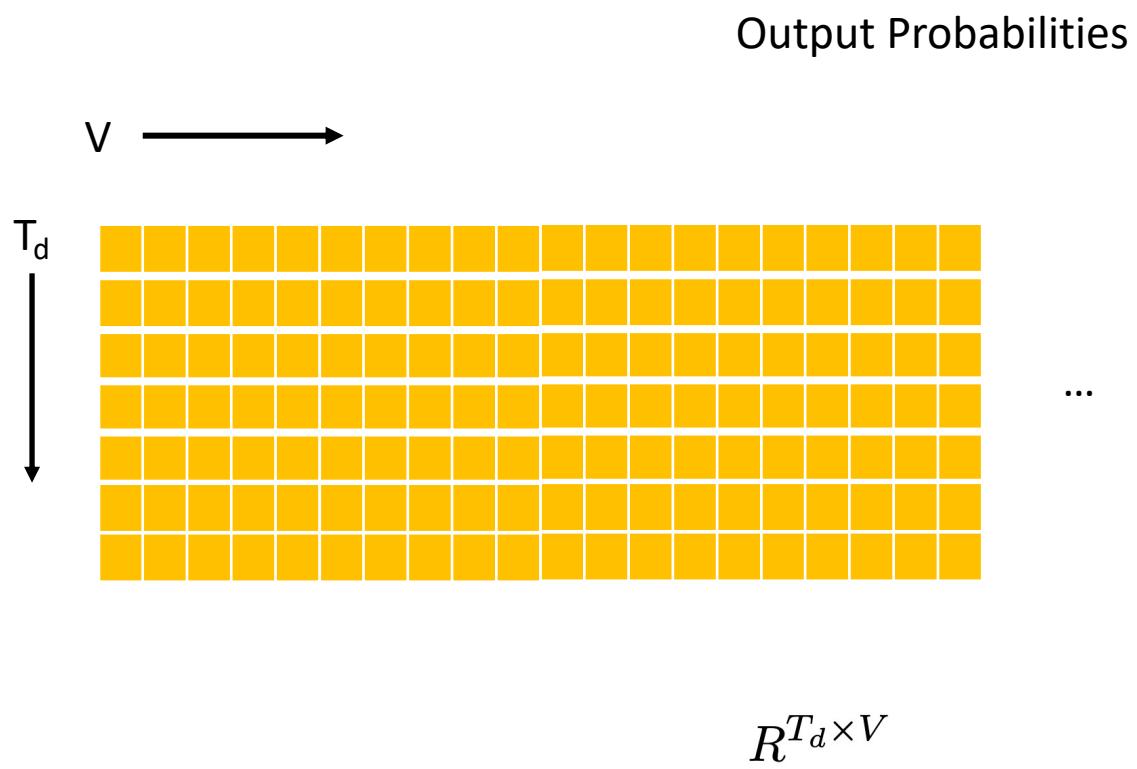
Decoder output



Linear



Softmax



Poll 2 (@1297)

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. Positional encodings help parallelize the transformer encoder
- d. Queries, keys, and values are obtained by splitting the input into 3 equal segments
- e. Multiheaded attention helps transformers find different kinds of relations between the tokens
- f. During decoding, decoder outputs function as queries and keys while the values come from the encoder

Poll 2 (@1126)

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. Positional encodings help parallelize the transformer encoder
- d. Queries, keys, and values are obtained by splitting the input into 3 equal segments
- e. **Multiheaded attention helps transformers find different kinds of relations between the tokens**
- f. During decoding, decoder outputs function as queries and keys while the values come from the encoder

Transformers

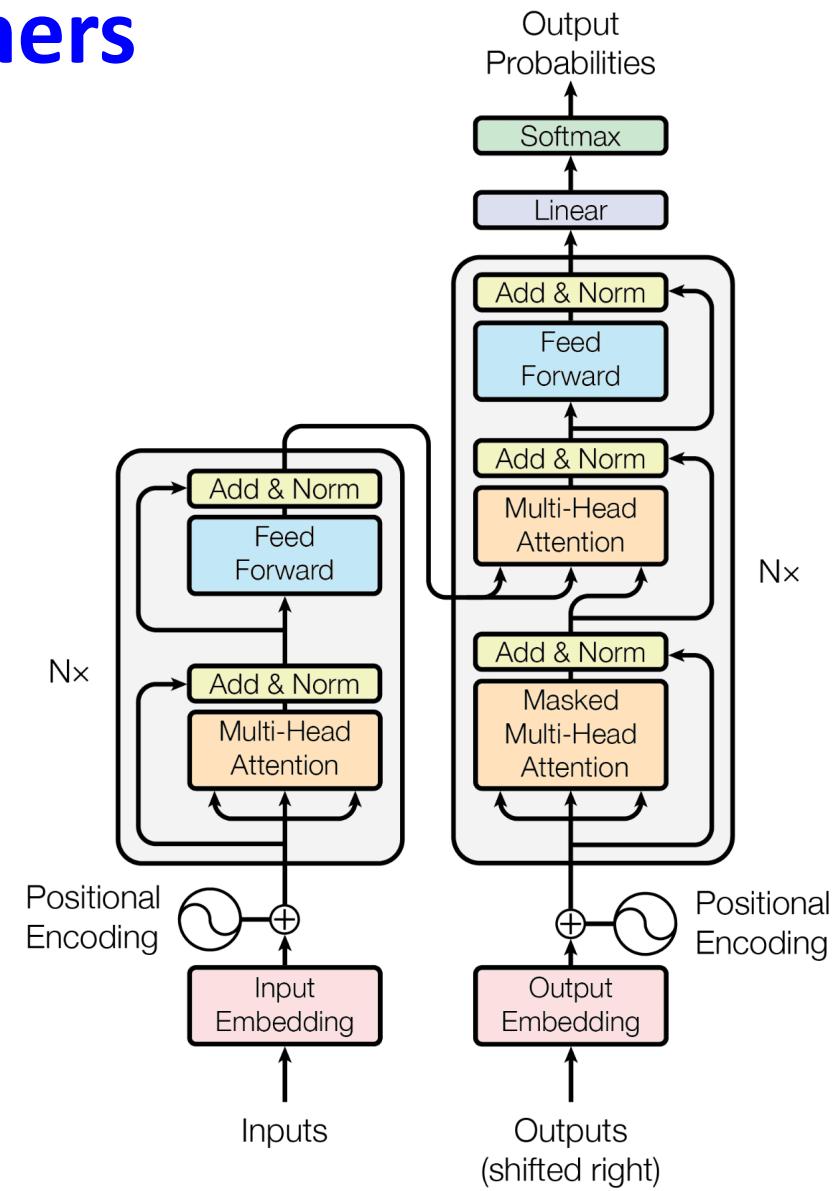
Targets

Ich have einen apfel gegessen

Inputs

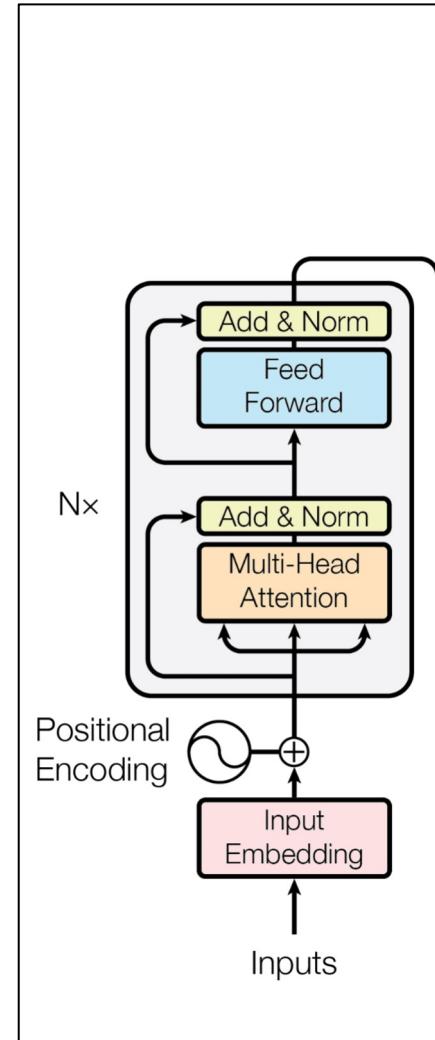
I ate an apple

Machine Translation



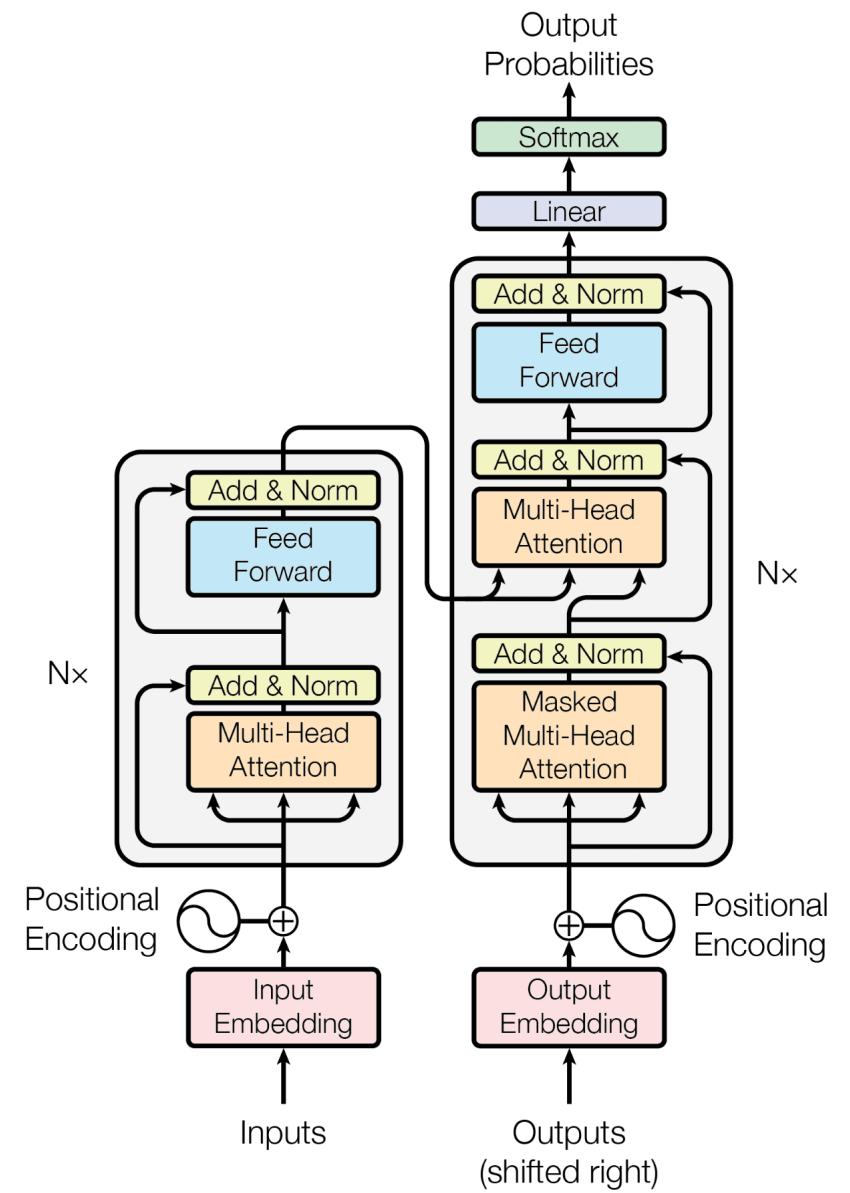
Transformers

- ✓ Tokenization
- ✓ Input Embeddings
- ✓ Position Encodings
- ✓ Residuals
- ✓ Query
- ✓ Key
- ✓ Value
- ✓ Add & Norm
- ✓ Encoder
- ✓ Decoder
- ✓ Attention
- ✓ Self Attention
- ✓ Multi Head Attention
- ✓ Masked Attention
- ✓ Encoder Decoder Attention
- ✓ Output Probabilities / Logits
- ✓ Softmax
 - Encoder-Decoder models
 - Decoder only models

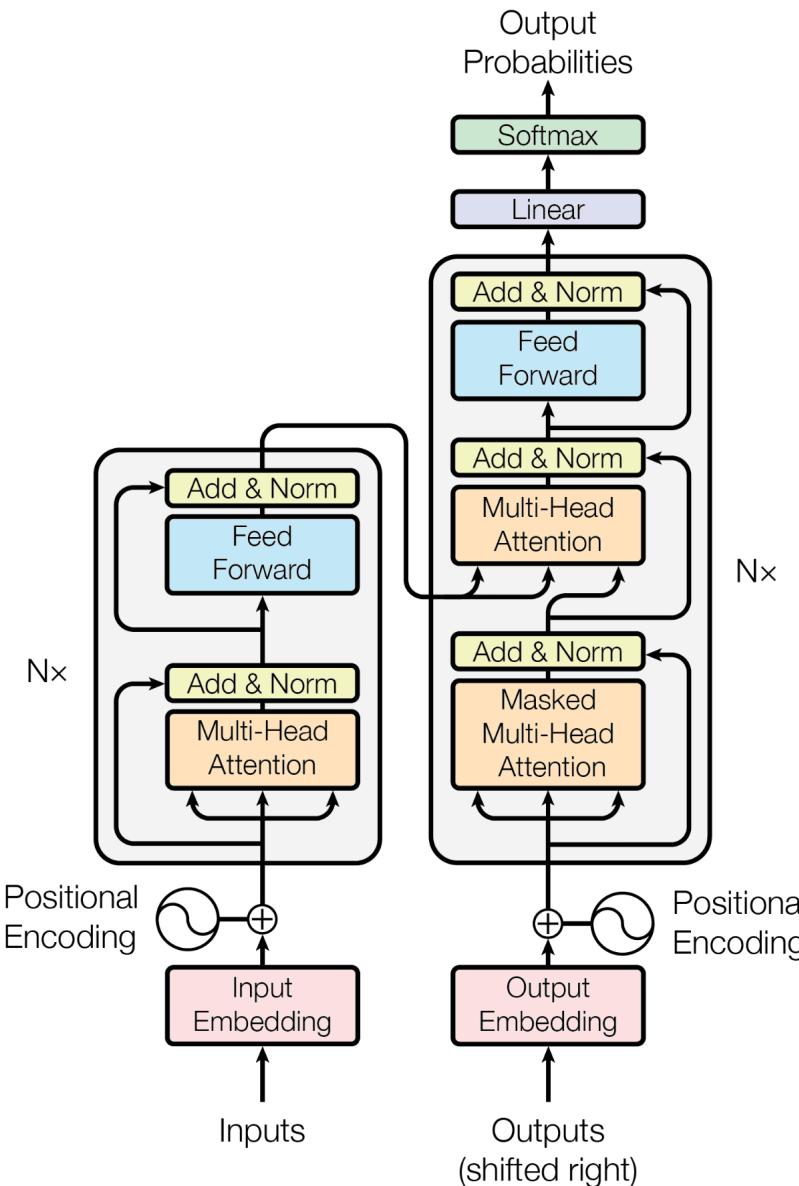


Part 2

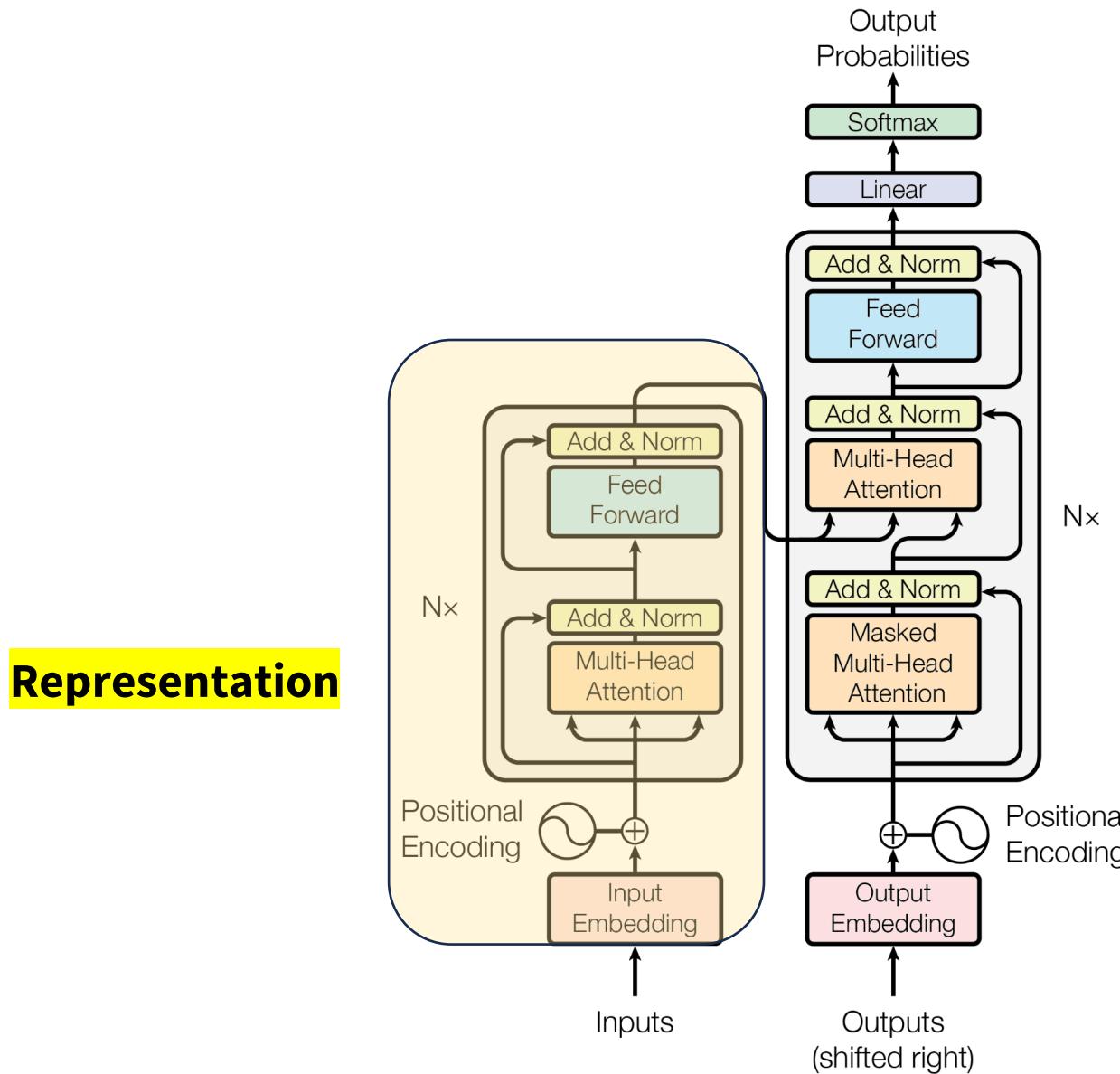
LLM



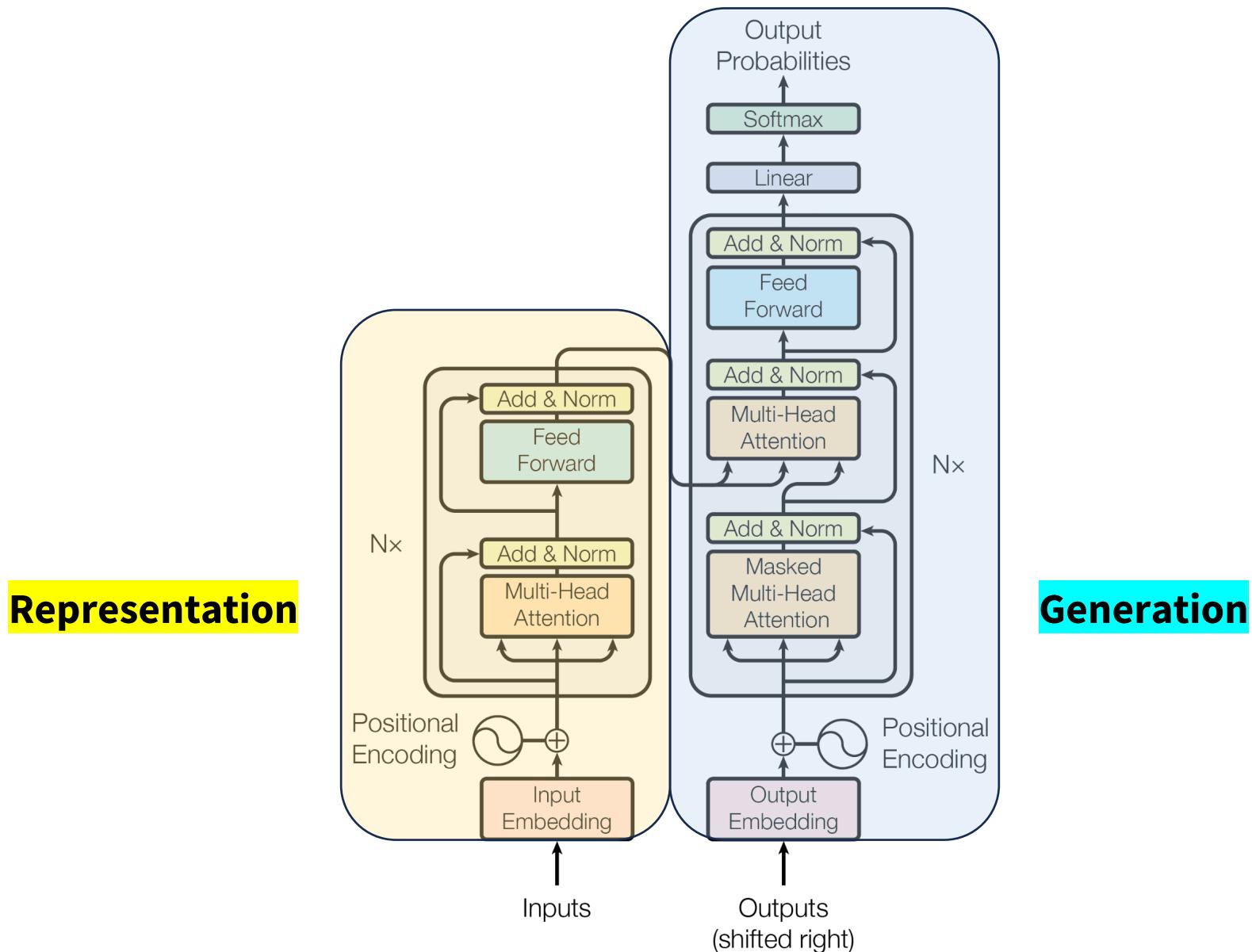
Transformers, mid-2017



Transformers, mid-2017



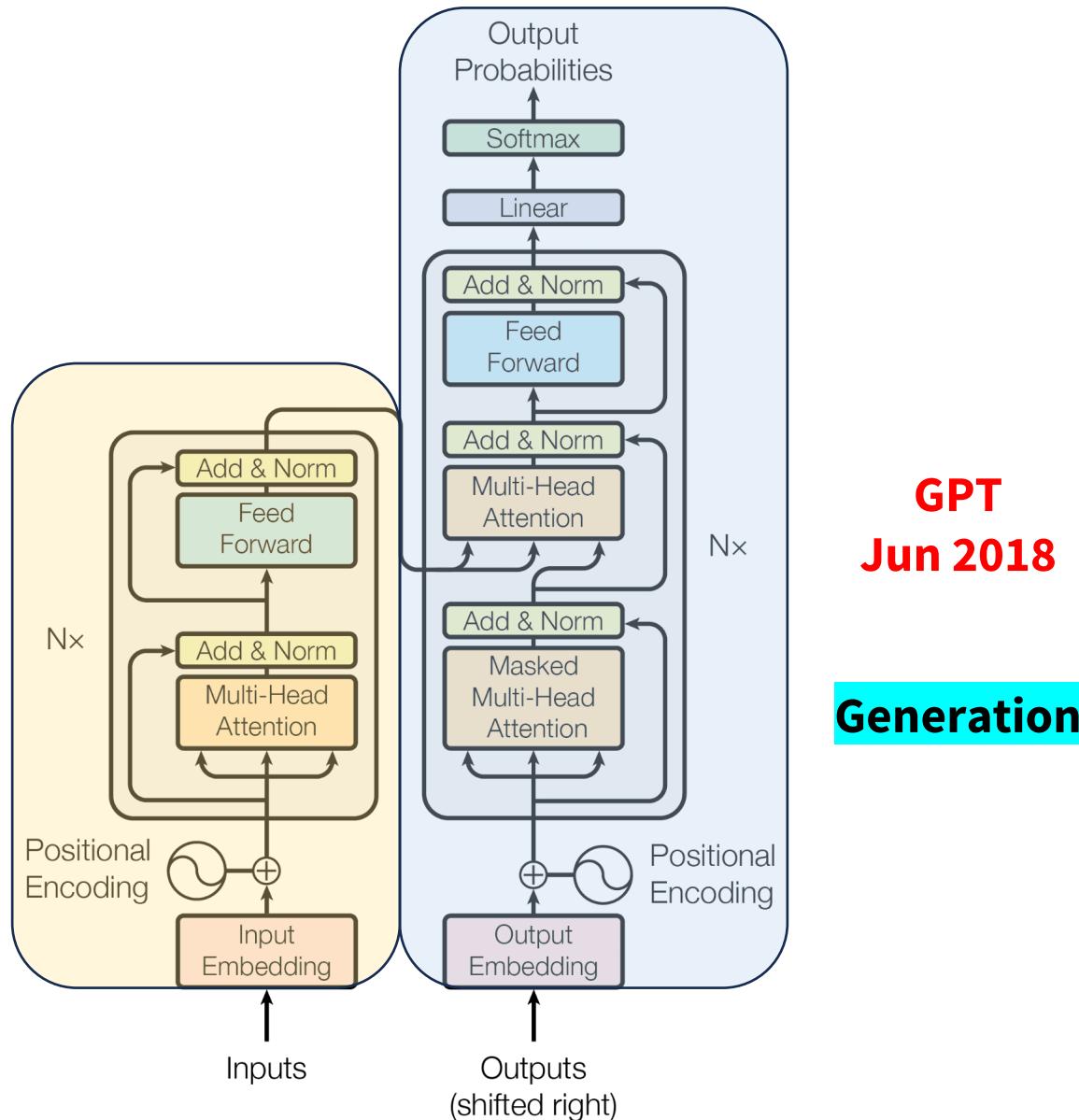
Transformers, mid-2017



2018 – The Inception of the LLM Era

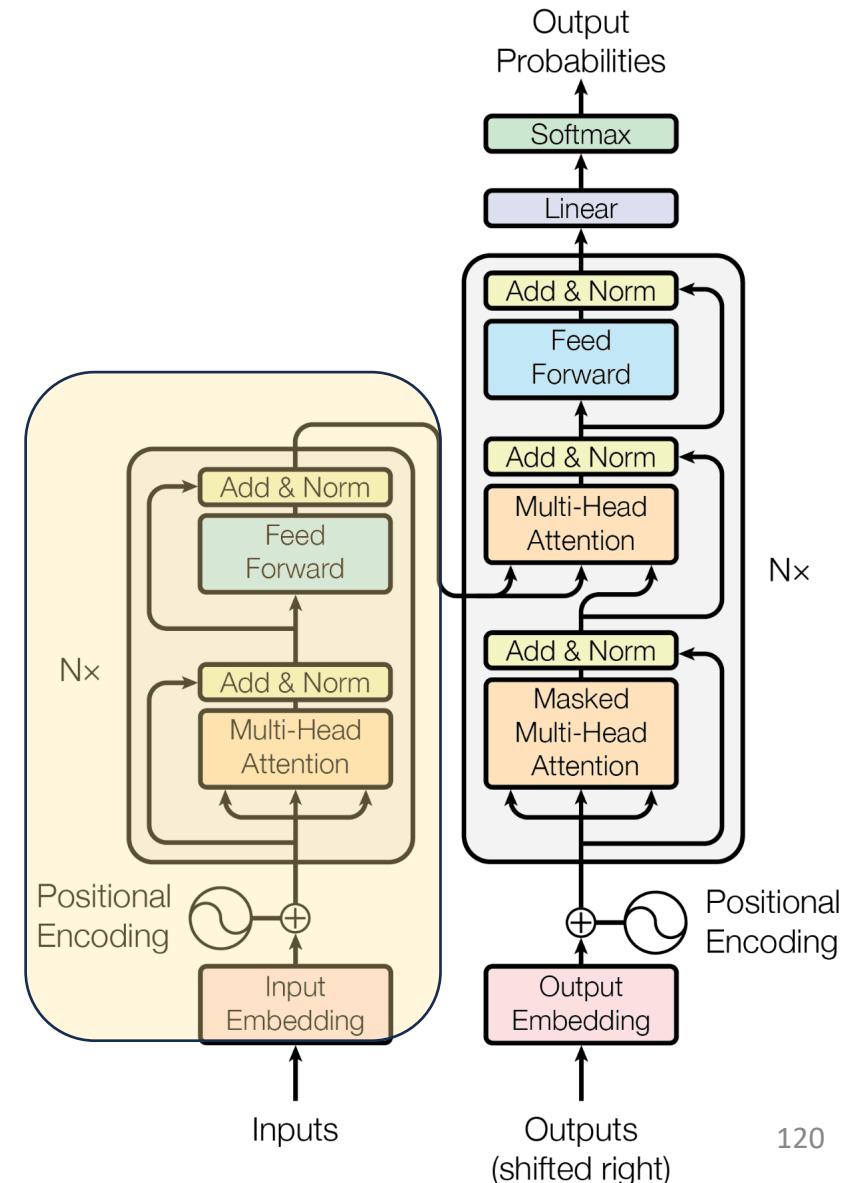
BERT
Oct 2018

Representation



BERT - Bidirectional Encoder Representations

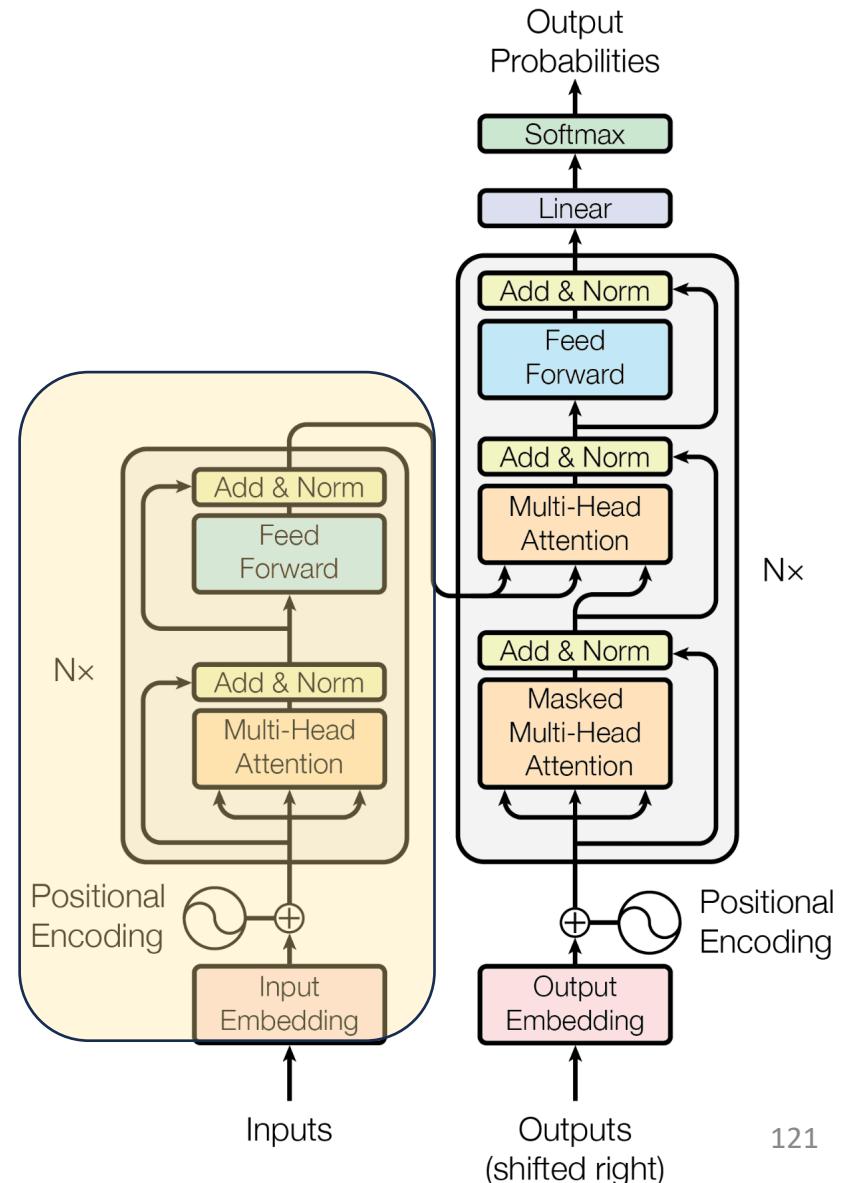
- One of the biggest challenges in LM-building used to be the lack of task-specific training data.
- What if we learn an effective representation that can be applied to a variety of downstream tasks?
 - Word2vec (2013)
 - GloVe (2014)



BERT - Bidirectional Encoder Representations

BERT Pre-Training Corpus:

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



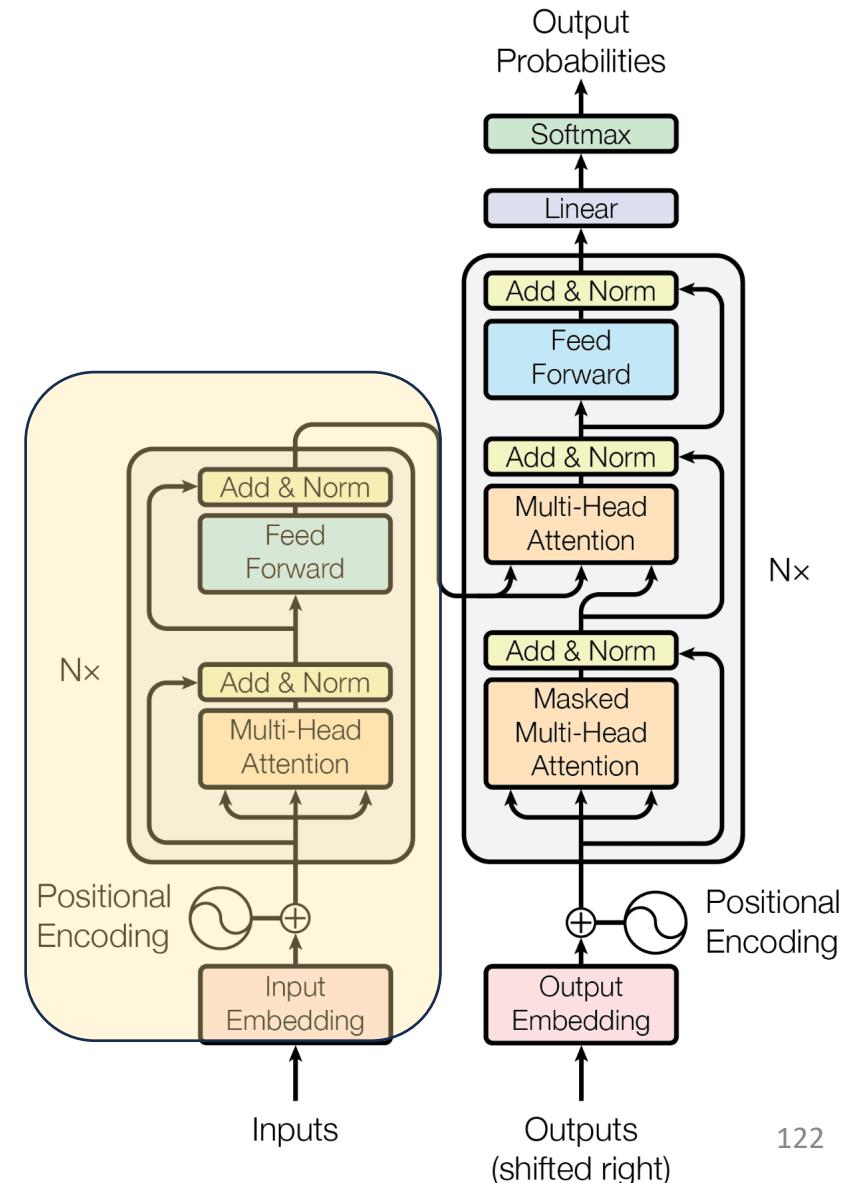
BERT - Bidirectional Encoder Representations

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:

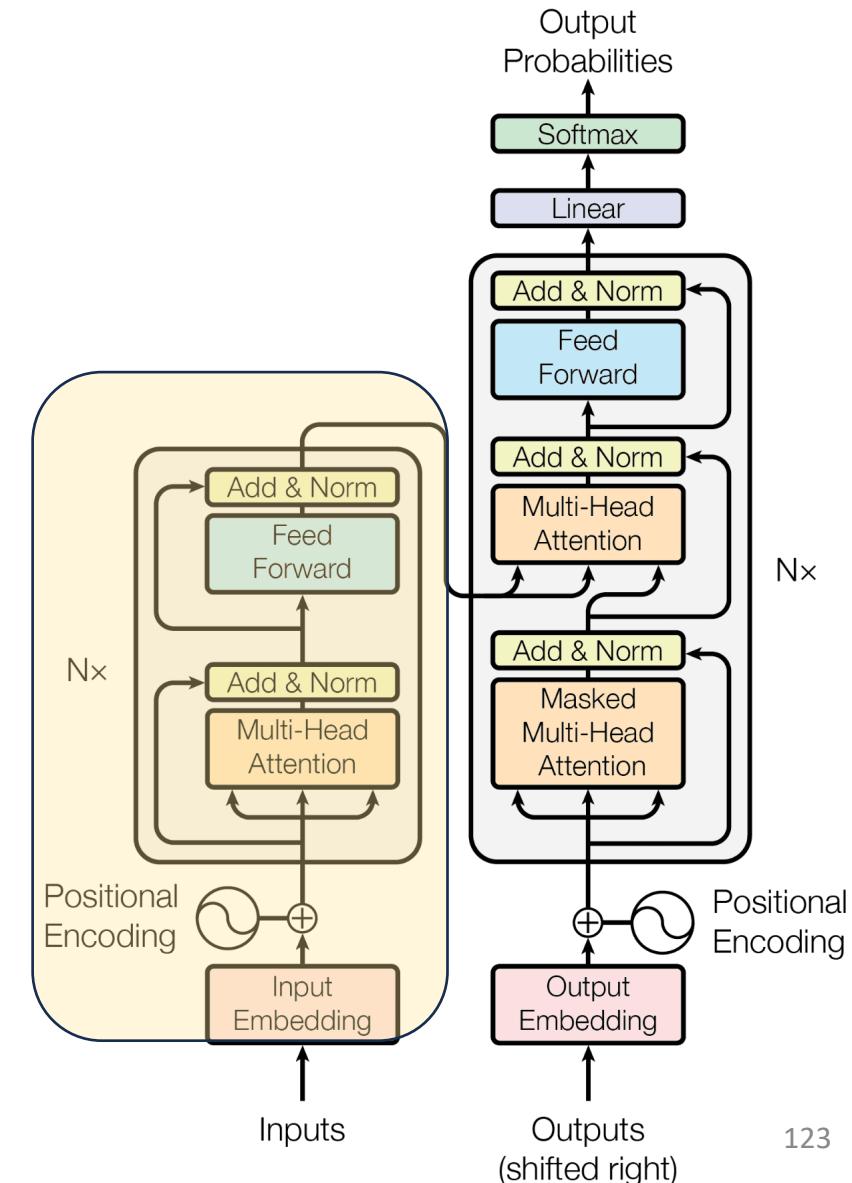
- English Wikipedia - 2,500 million words
- Book Corpus - 800 million words

BERT Pre-Training Tasks:

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

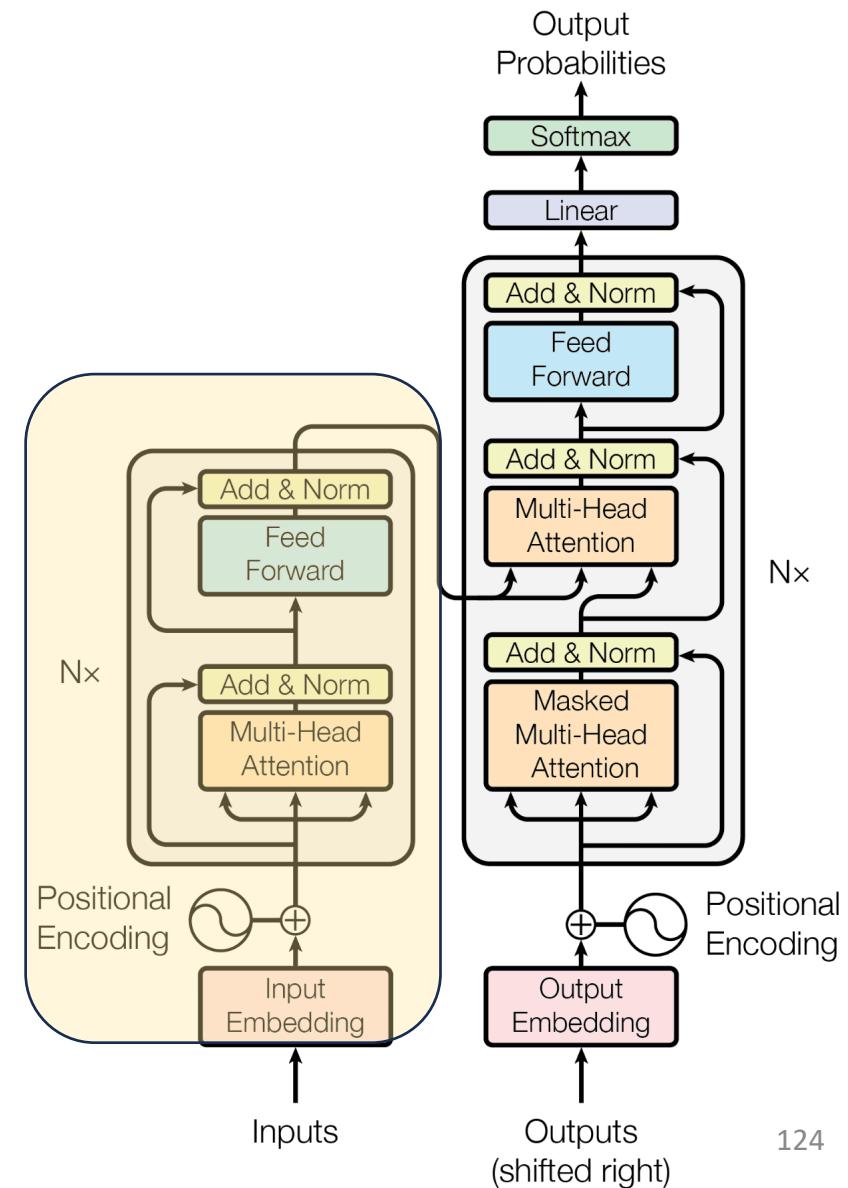
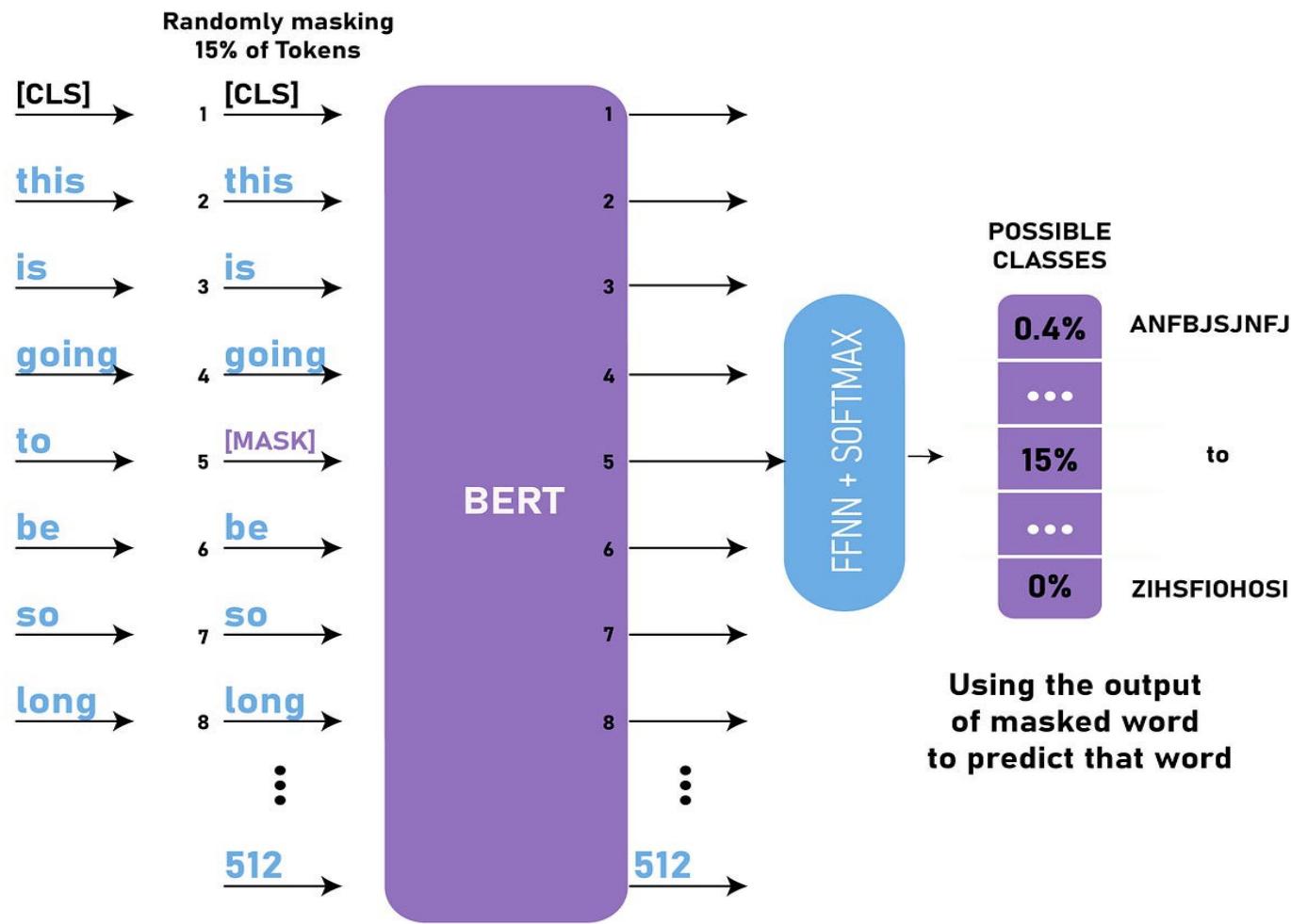
BERT Pre-Training Results:

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



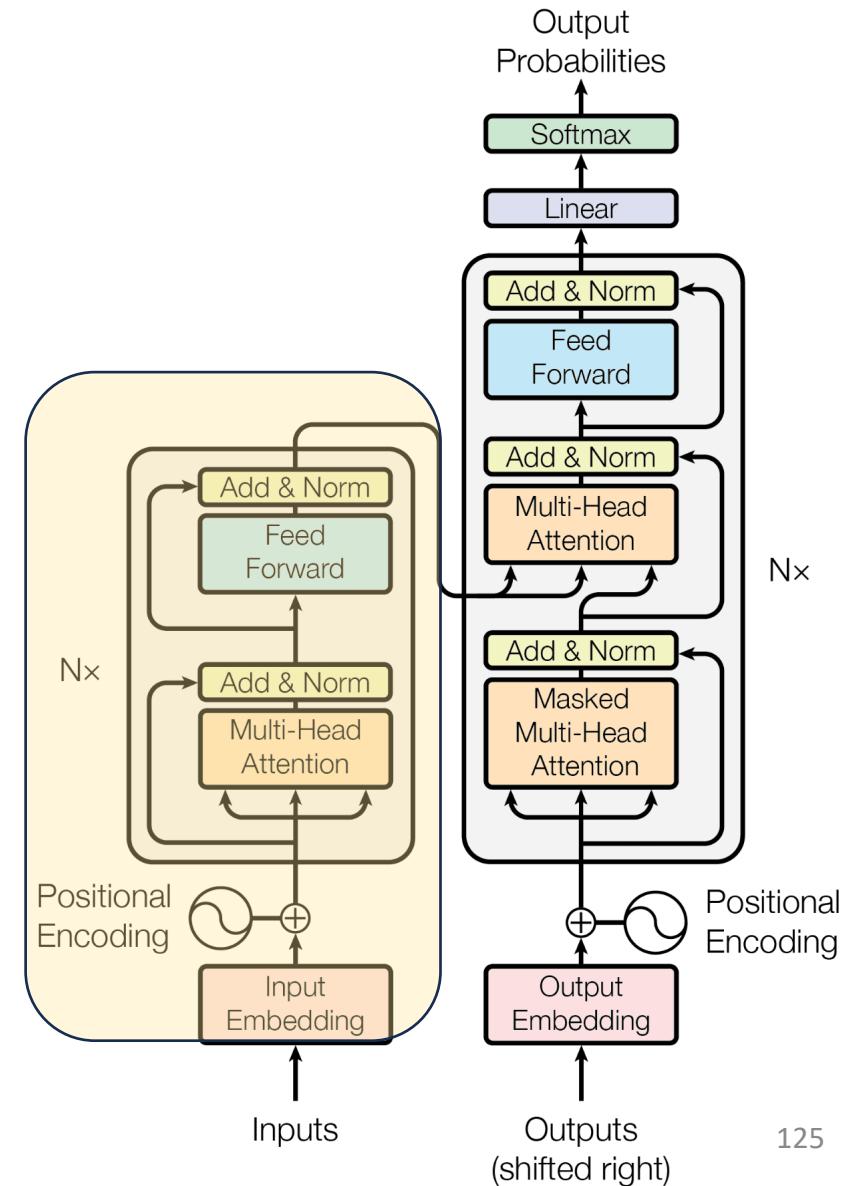
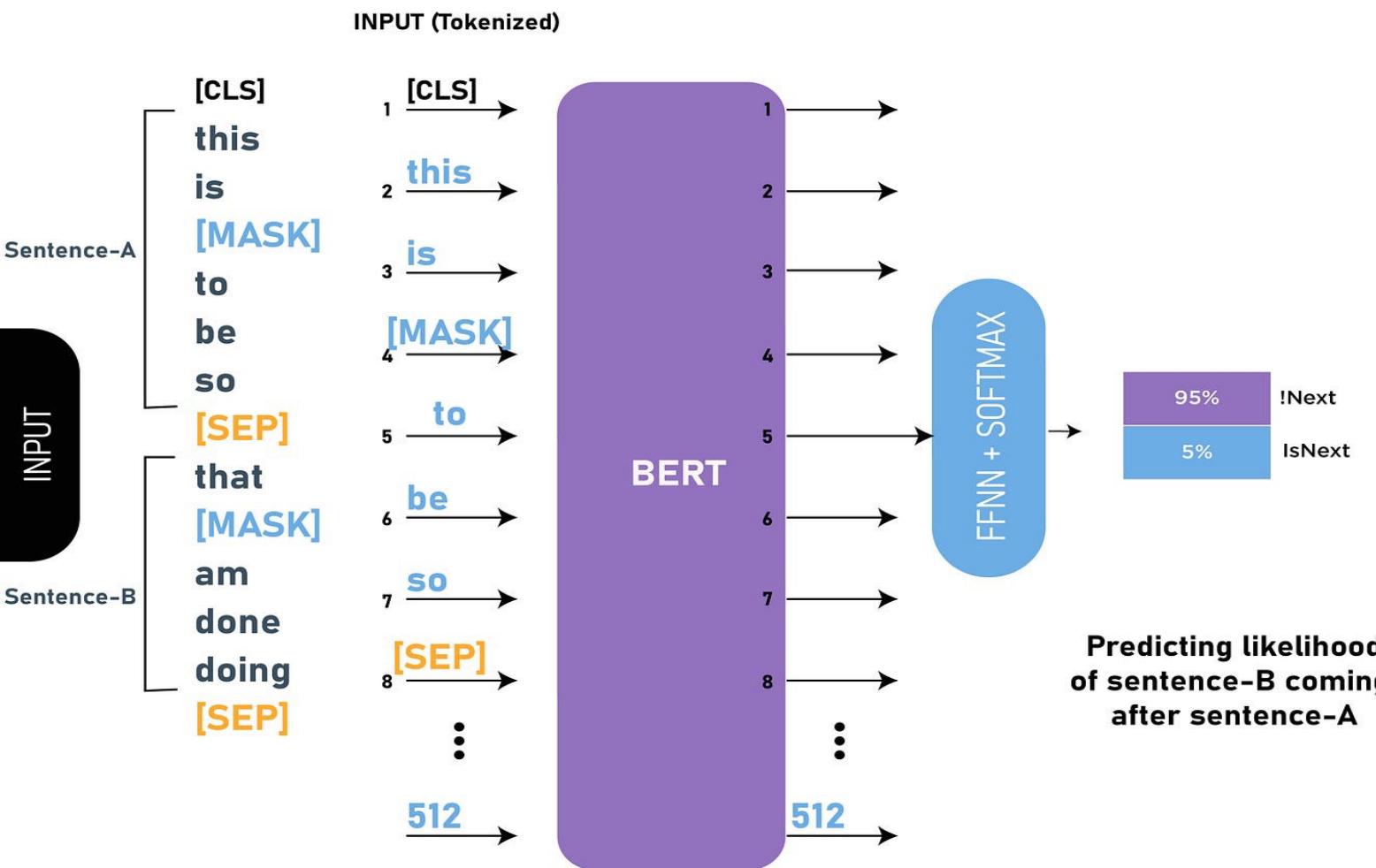
BERT - Bidirectional Encoder Representations

MLM (Masked Language Modeling)



BERT - Bidirectional Encoder Representations

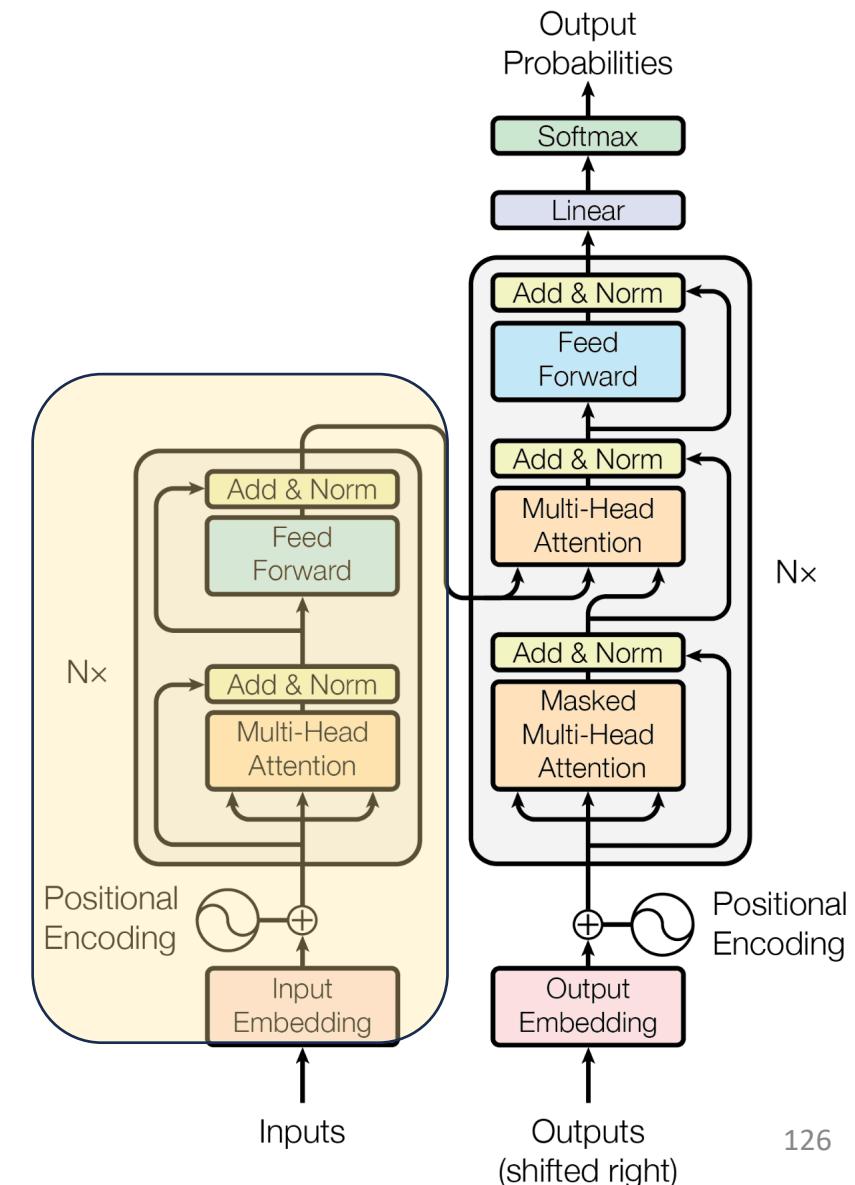
NSP (Next Sentence Prediction)



BERT - Bidirectional Encoder Representations

BERT Fine-Tuning:

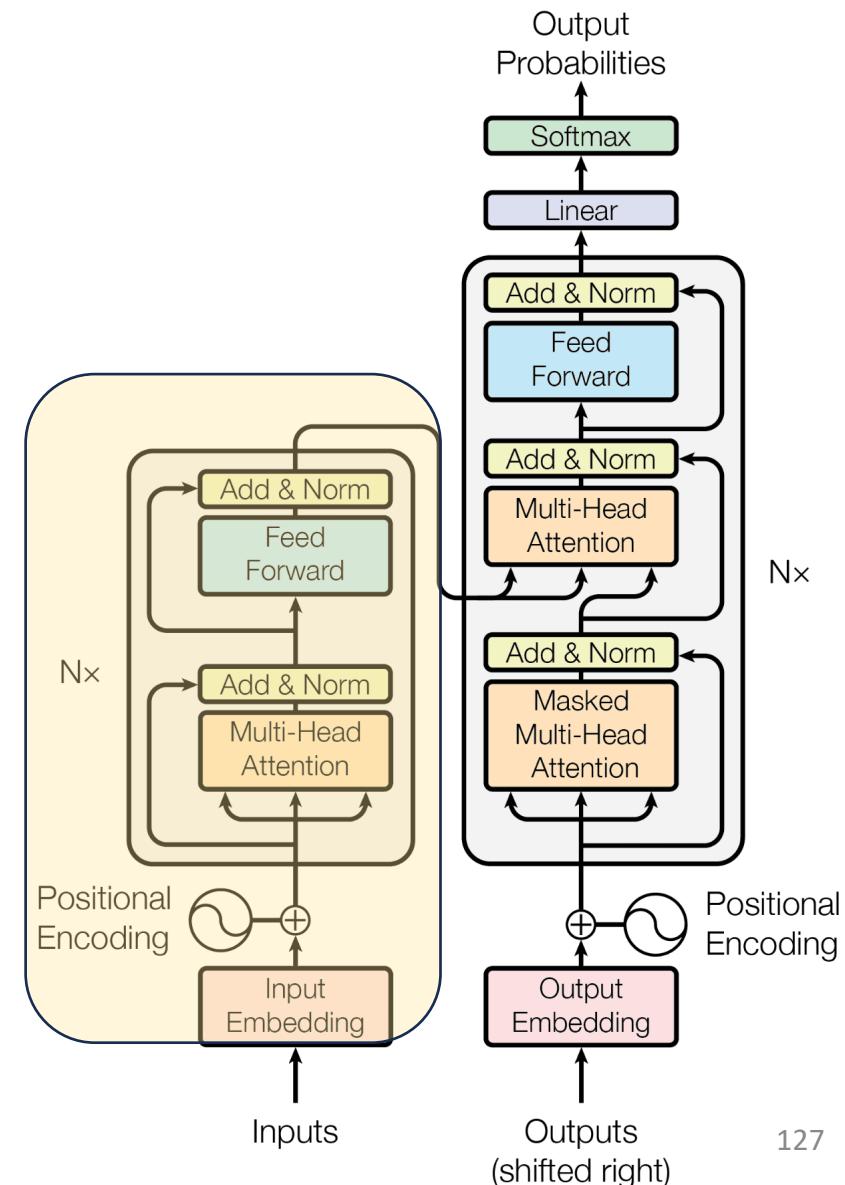
- Simply add a task-specific layer after the last encoder layer outputs.
- Classification Tasks:
 - Add a feed-forward layer on top of the encoder output for the [CLS] token
- Question Answering Tasks:
 - Train two extra vectors to mark the beginning and end of answer from paragraph
- ...



BERT - Bidirectional Encoder Representations

BERT Evaluation:

- General Language Understanding Evaluation (GLUE)
 - Sentence pair tasks
 - Single sentence classification
- Stanford Question Answering Dataset (SQuAD)



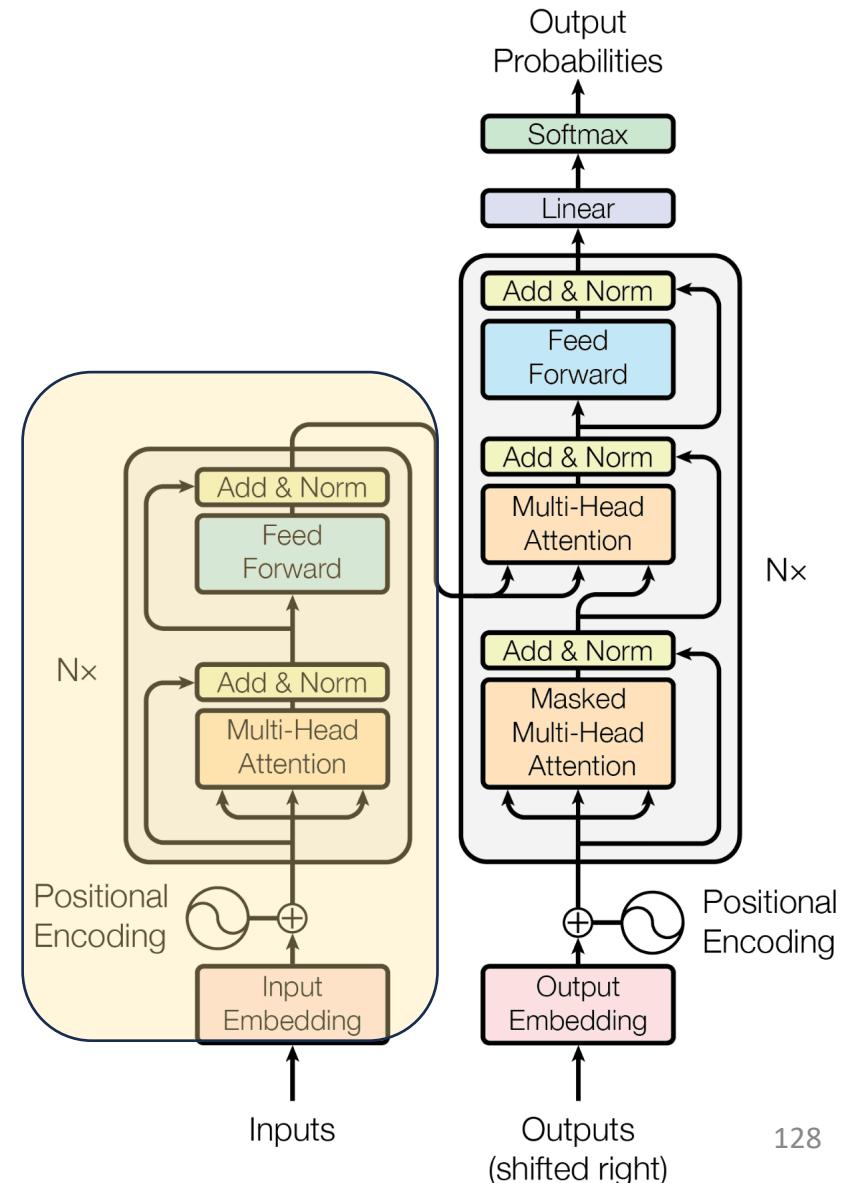
BERT - Bidirectional Encoder Representations

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
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

	System		Dev		Test		
			EM	F1	EM	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							
BERT _{BASE} (Single)	80.8	88.5	-	-			
BERT _{LARGE} (Single)	84.1	90.9	-	-			
BERT _{LARGE} (Ensemble)	85.8	91.8	-	-			
BERT _{LARGE} (Sgl.+TriviaQA)	84.2	91.1	85.1	91.8			
BERT _{LARGE} (Ens.+TriviaQA)	86.2	92.2	87.4	93.2			

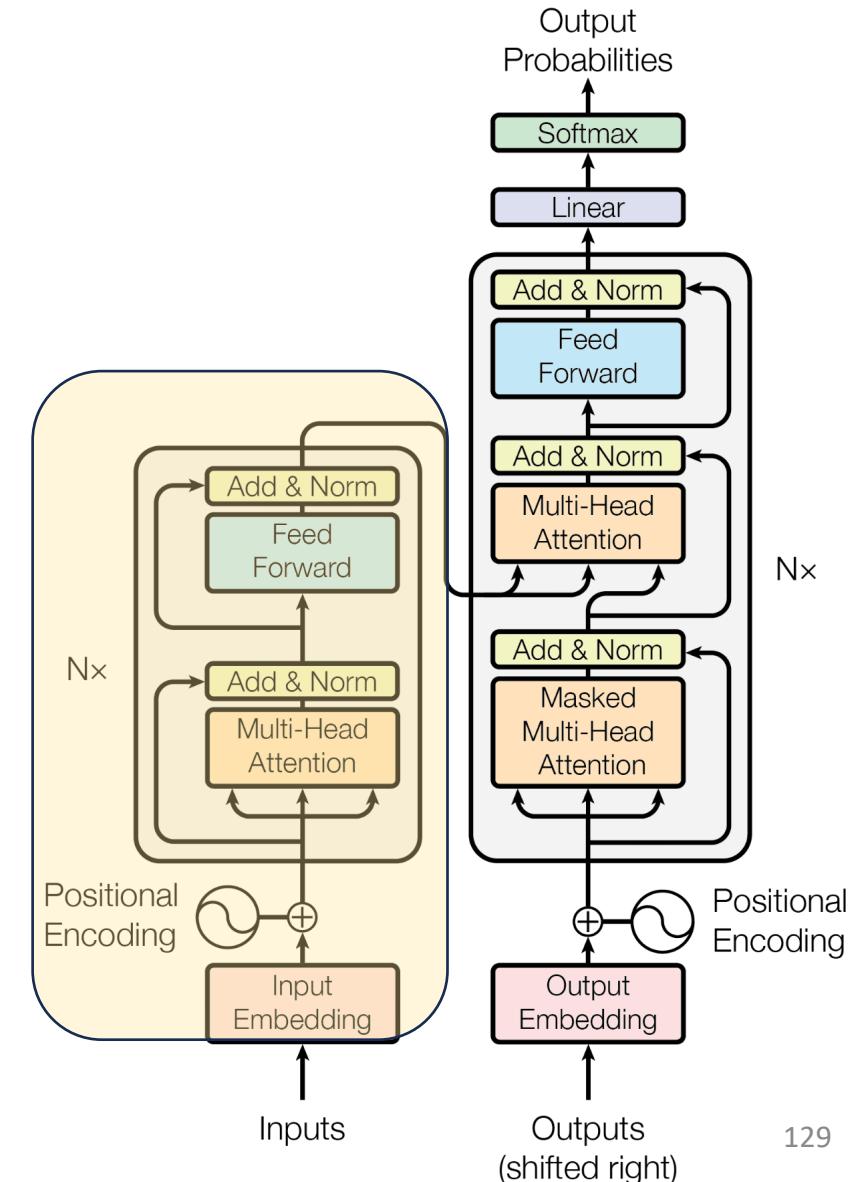
Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.



BERT - Bidirectional Encoder Representations

What is our takeaway from BERT?

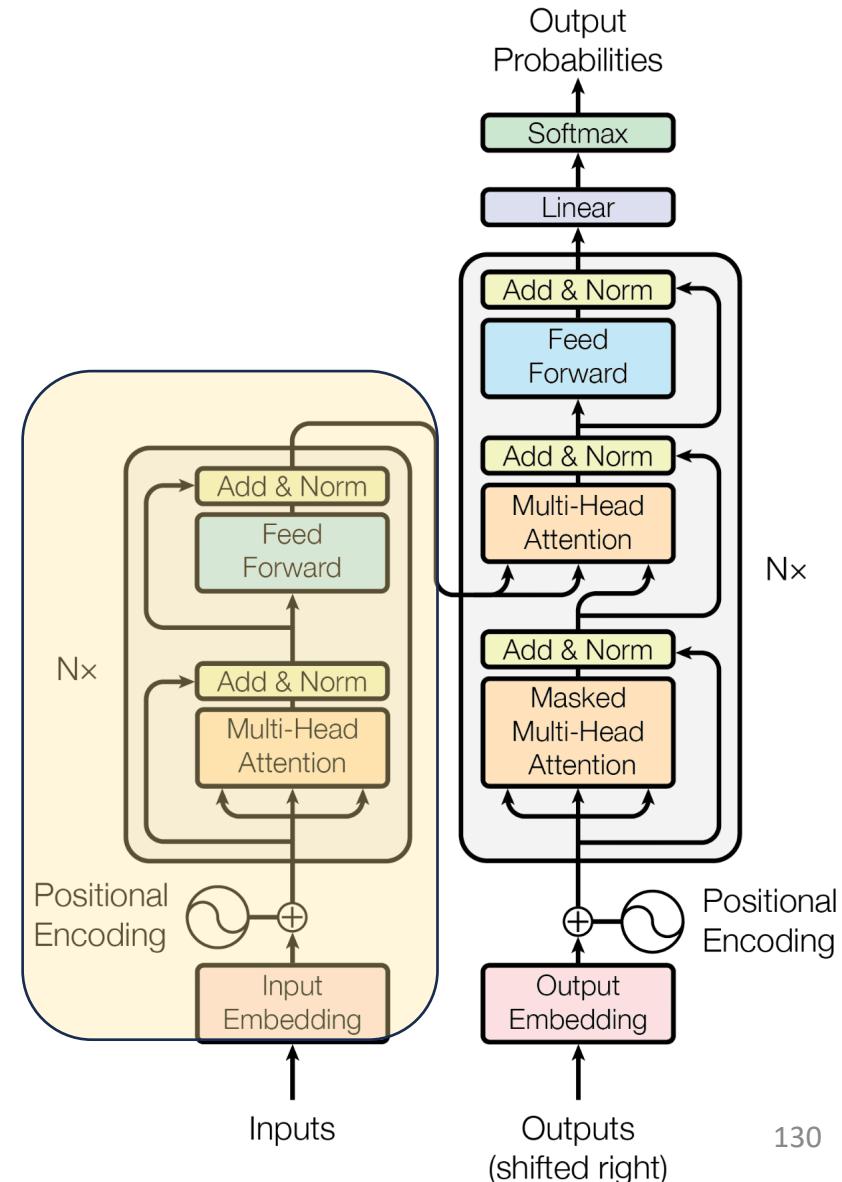
- Pre-training tasks can be invented flexibly...
 - Effective representations can be derived from a flexible regime of pre-training tasks.



BERT - Bidirectional Encoder Representations

What is our takeaway from BERT?

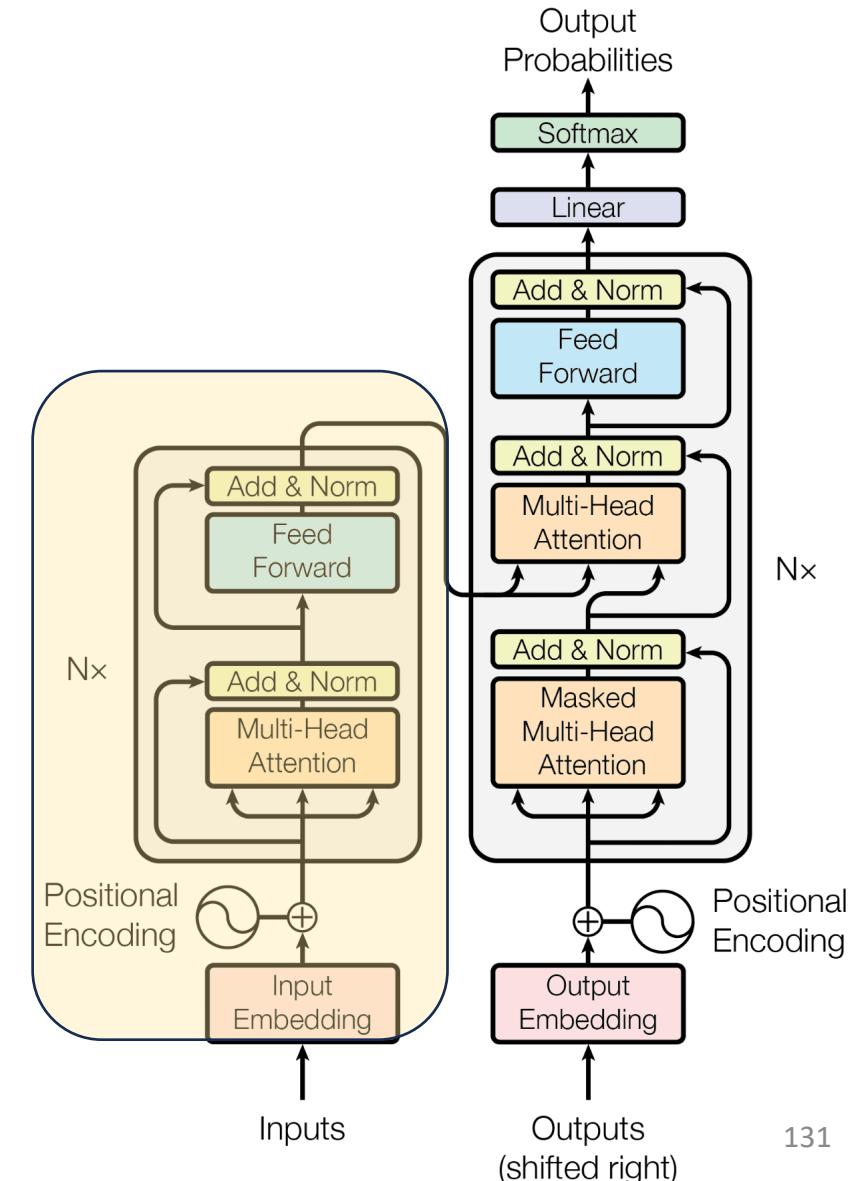
- Pre-training tasks can be invented flexibly...
 - Effective representations can be derived from a flexible regime of pre-training tasks.
 - Different NLP tasks seem to be highly transferable with each other...
 - As long as we have effective representations, that seems to form a general model which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



BERT - Bidirectional Encoder Representations

What is our takeaway from BERT?

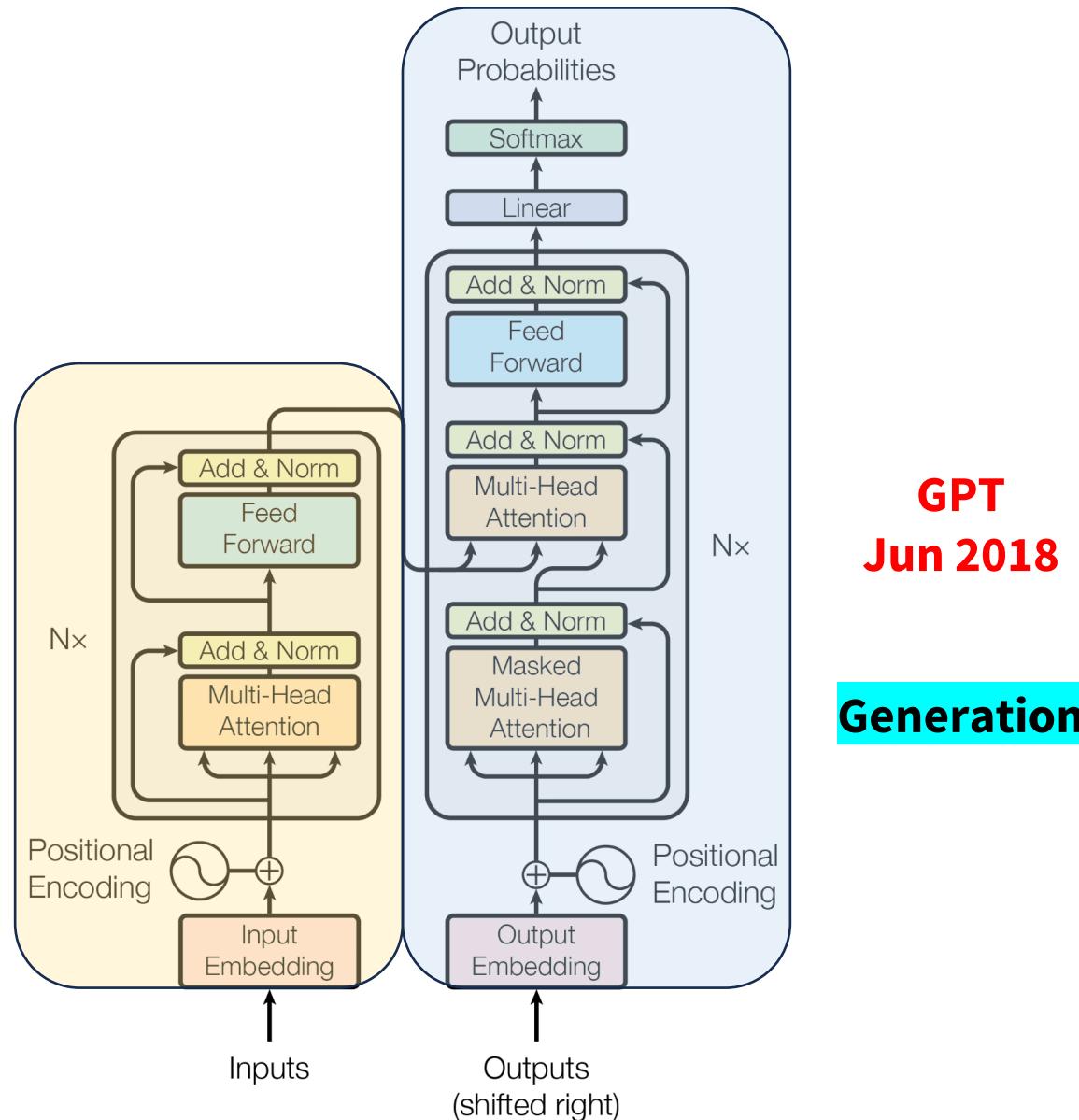
- Pre-training tasks can be invented flexibly...
 - Effective representations can be derived from a flexible regime of pre-training tasks.
- Different NLP tasks seem to be highly transferable with each other...
 - As long as we have effective representations, that seems to form a general model which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



2018 – The Inception of the LLM Era

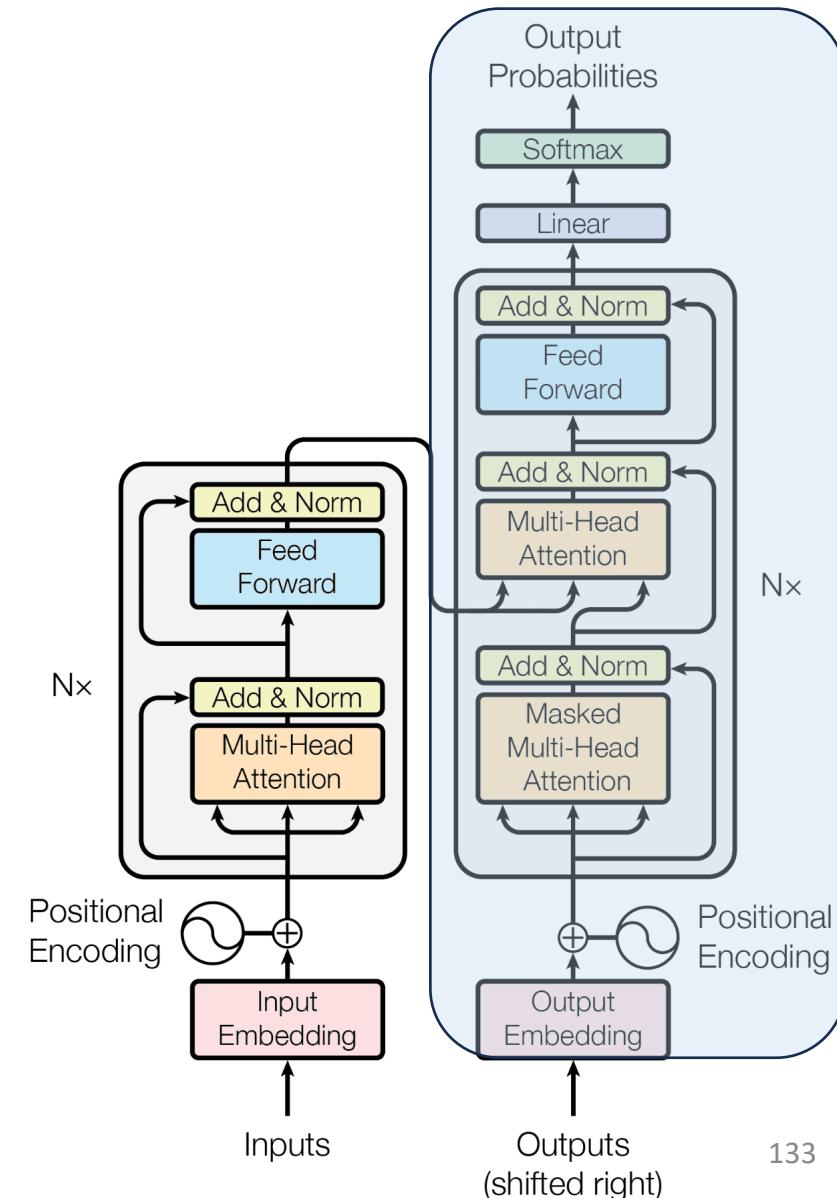
BERT
Oct 2018

Representation



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:

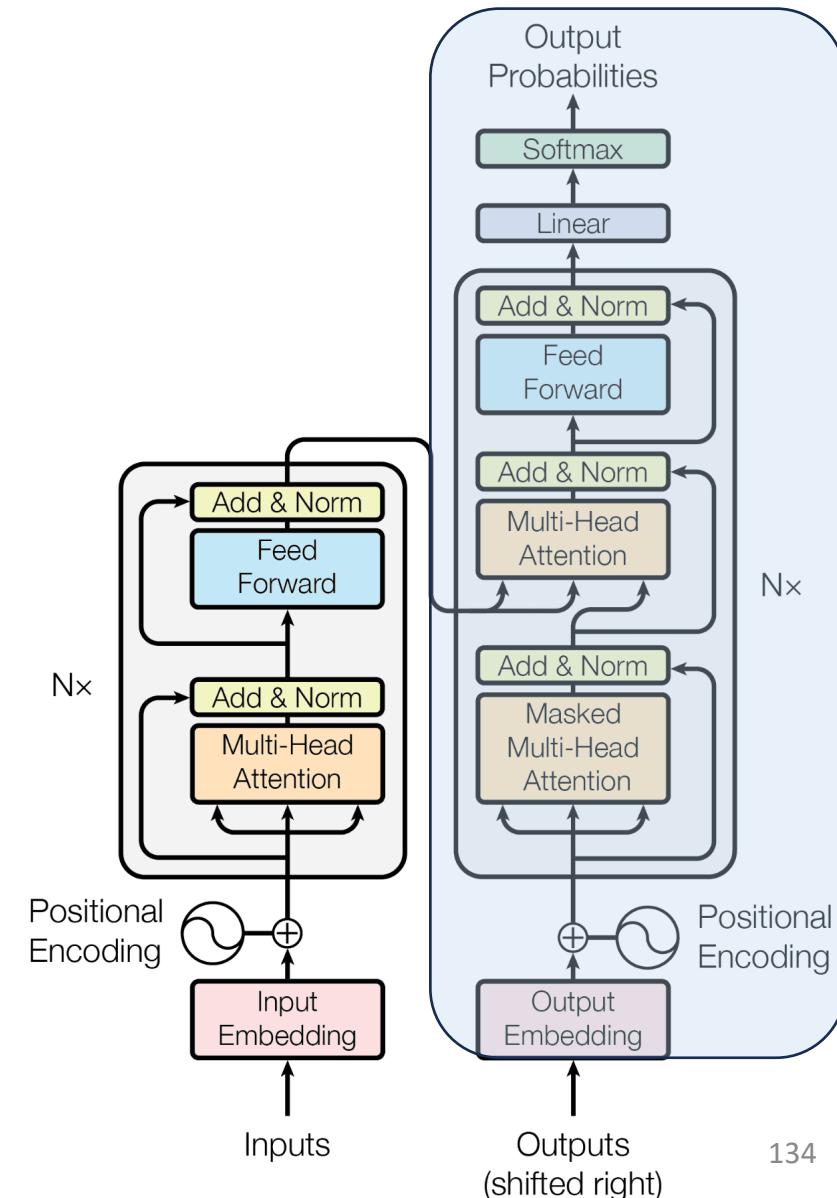
- Similarly, BooksCorpus and English Wikipedia

GPT Pre-Training Tasks:

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

GPT Pre-Training Results:

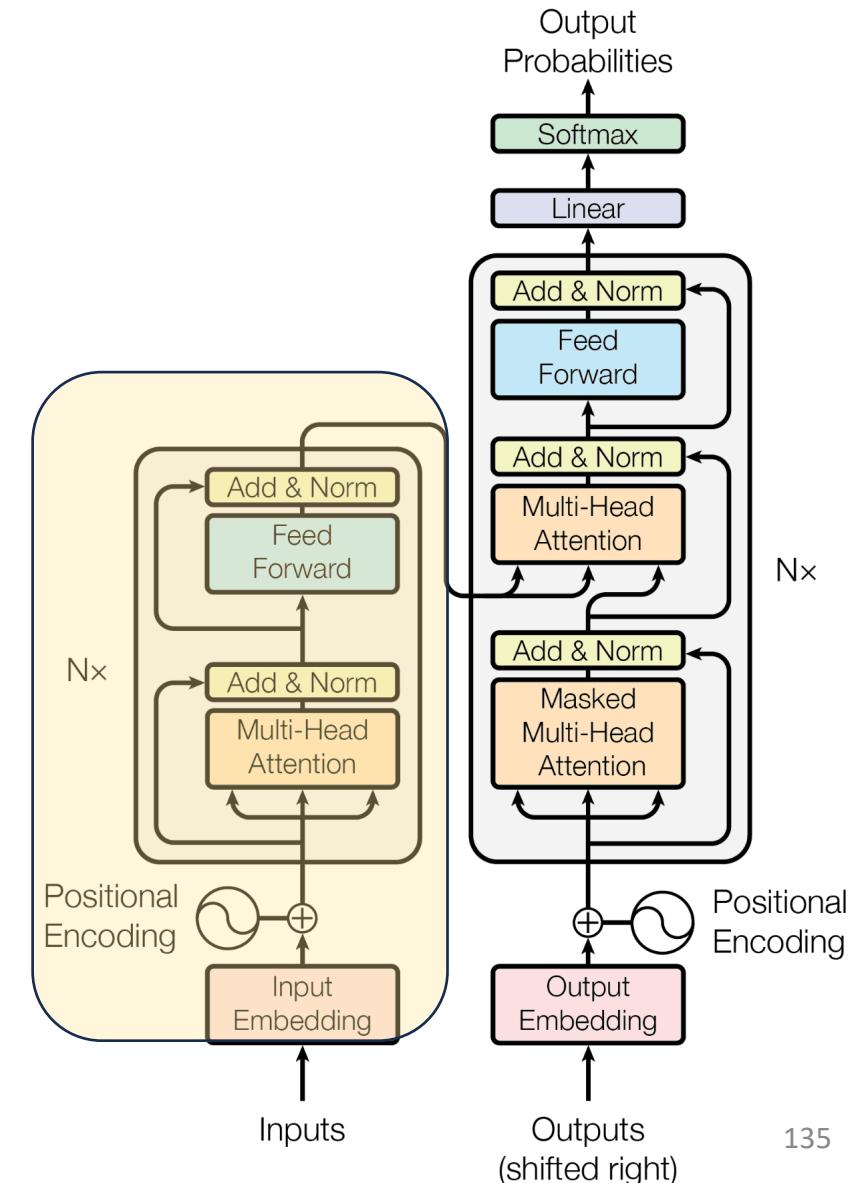
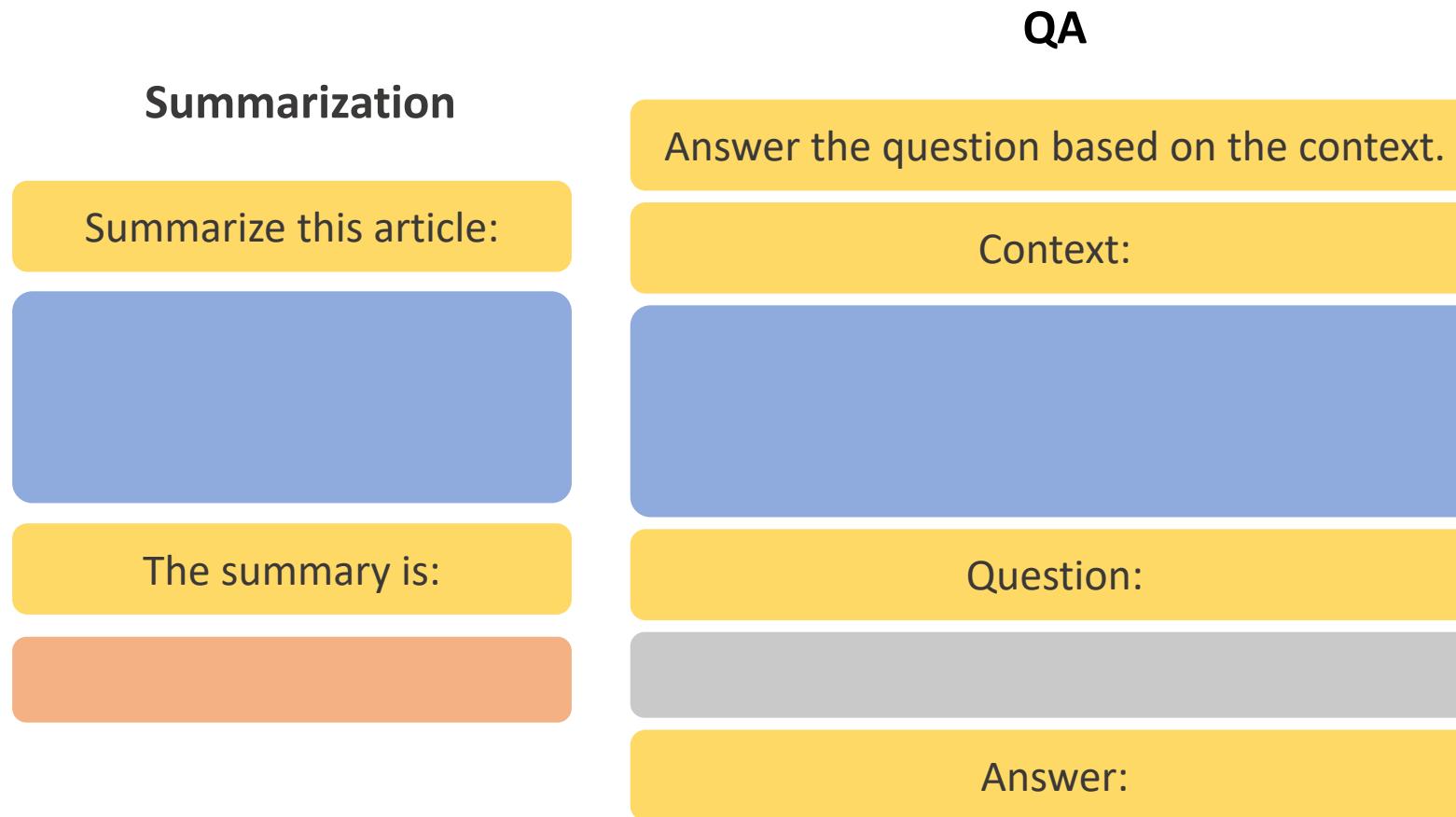
- GPT – 117M Params
 - Similarly competitive on GLUE and SQuAD



GPT – Generative Pretrained Transformer

GPT Fine-Tuning:

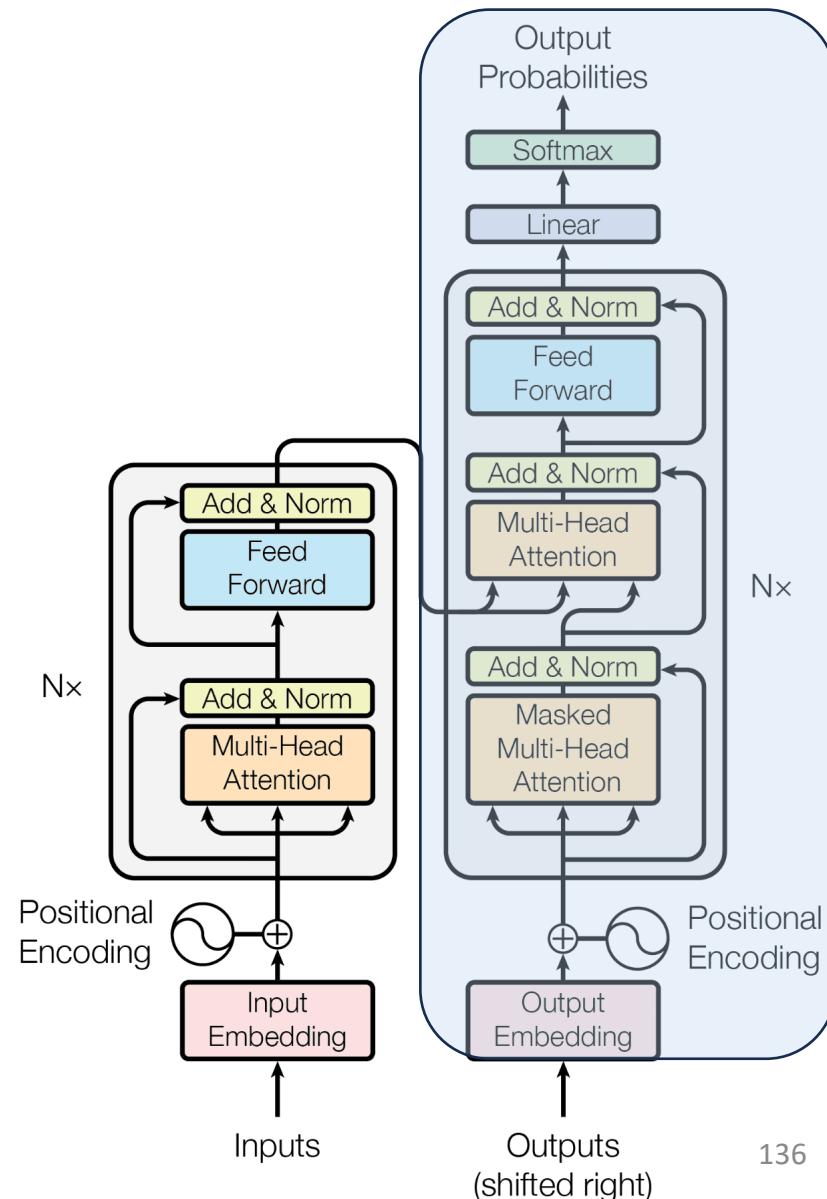
- Prompt-format task-specific text as a continuous stream for the model to fit



GPT – Generative Pretrained Transformer

What is our takeaway from GPT?

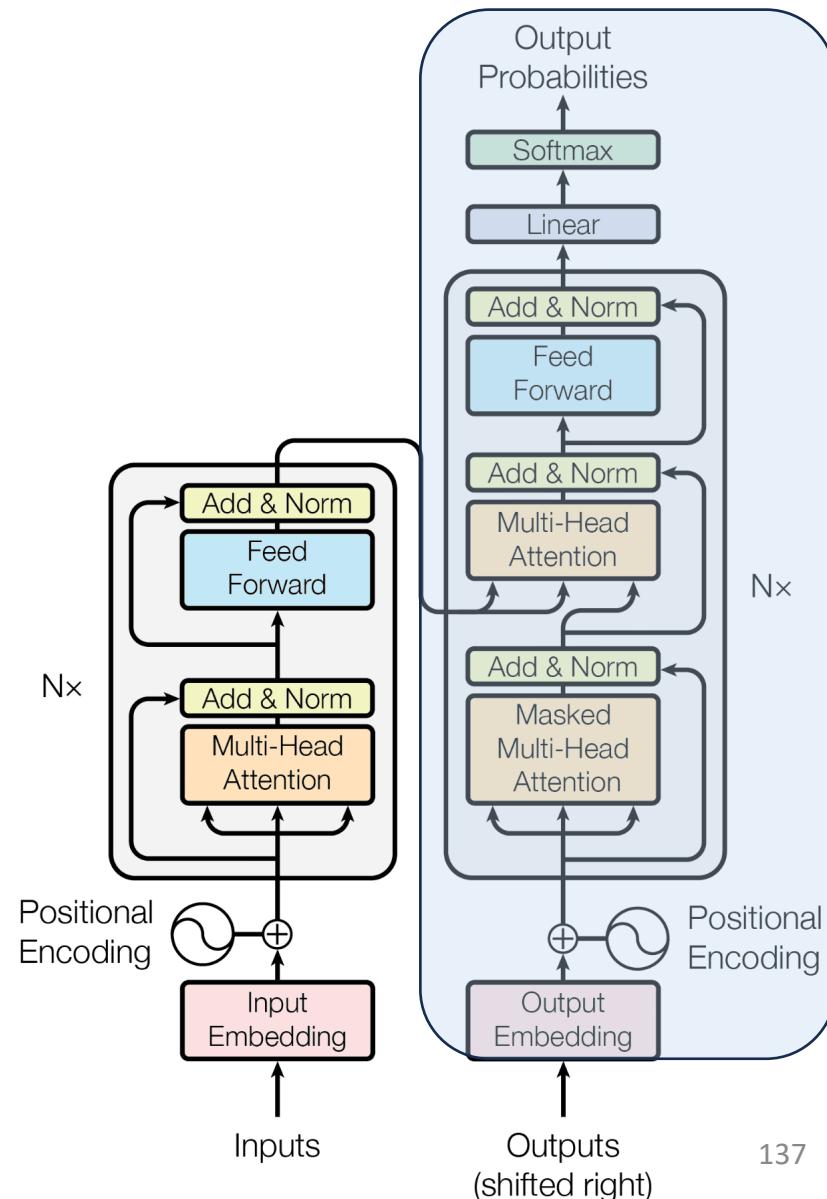
- **The Effectiveness of Self-Supervised Learning**
 - Specifically, the model seems to be able to learn from generating the language *itself*, rather than from any specific task we might cook up.



GPT – Generative Pretrained Transformer

What is our takeaway from GPT?

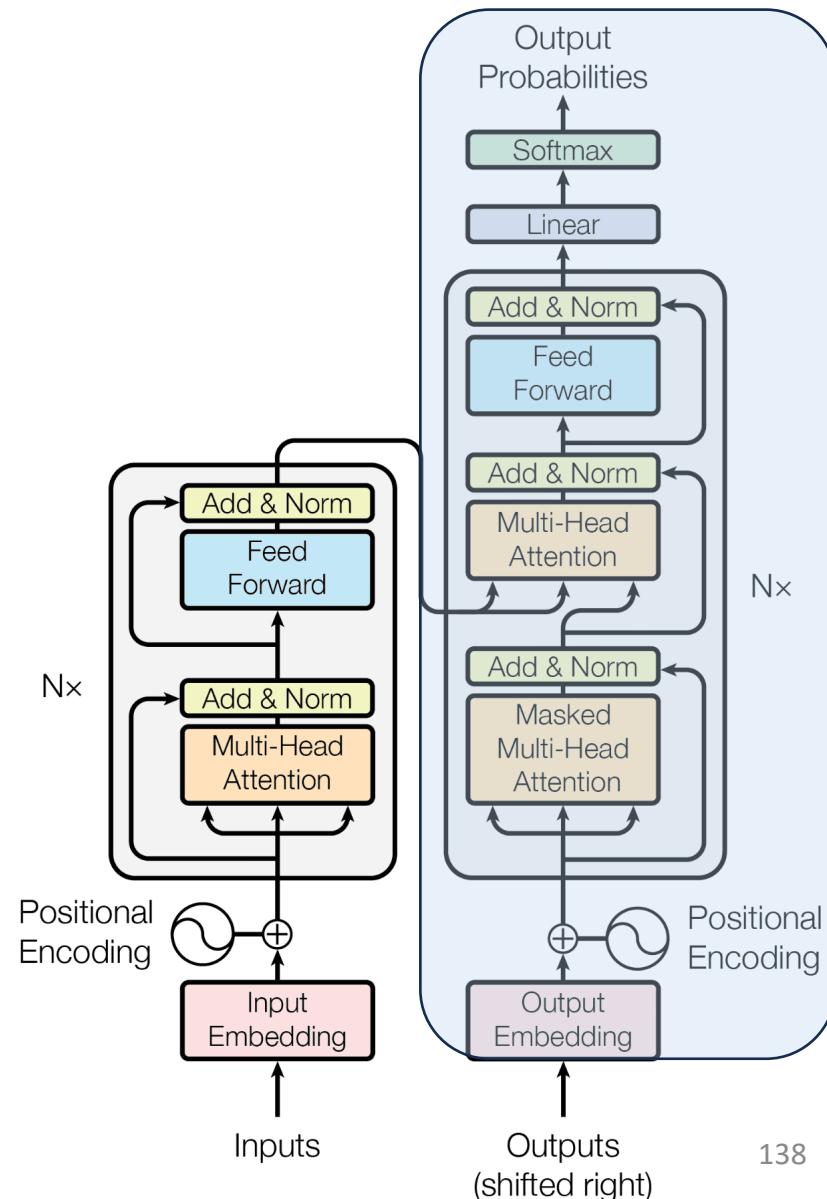
- **The Effectiveness of Self-Supervised Learning**
 - Specifically, the model seems to be able to learn from generating the language *itself*, rather than from any specific task we might cook up.
- **Language Model as a Knowledge Base**
 - Specifically, a generatively pretrained model seems to have 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 @1291

The original GPT's parameter count is closest to

- a. 117
- b. 117K
- c. 117M
- d. 117B

Poll 3 @1291

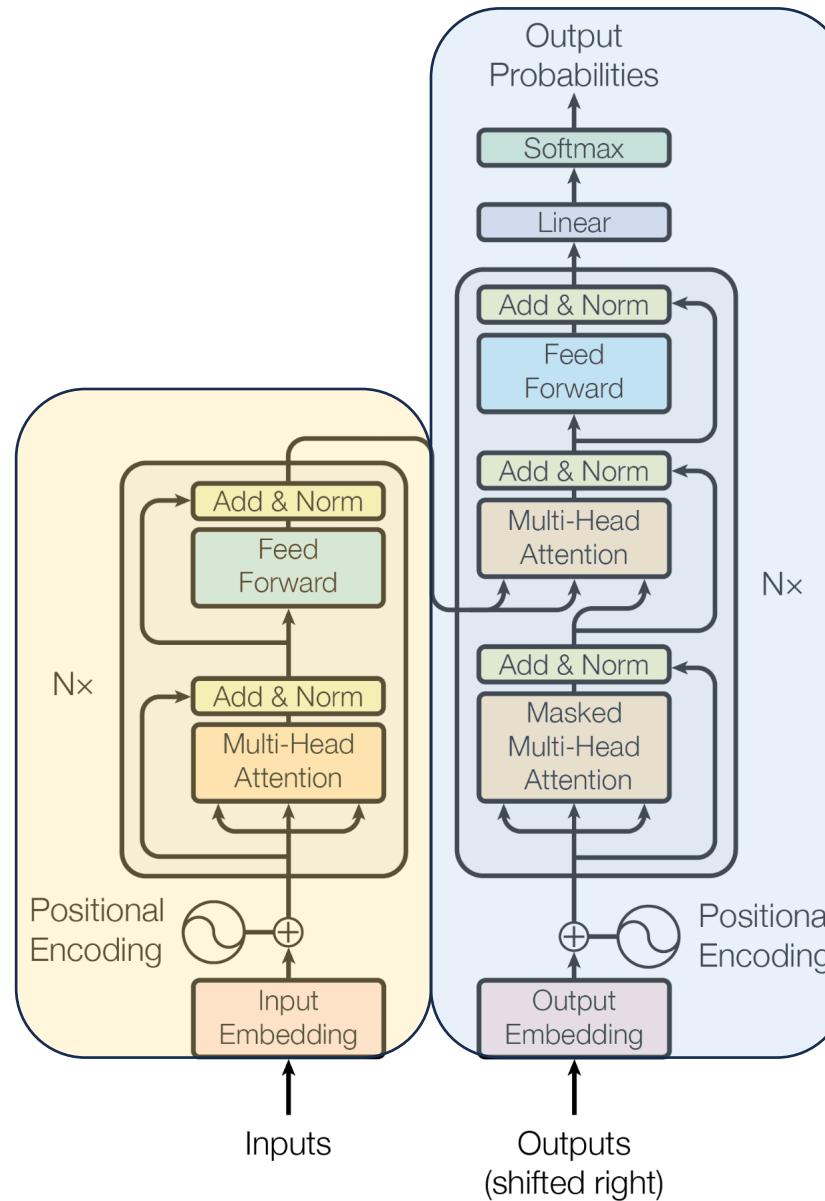
The original GPT's parameter count is closest to

- a. 117
- b. 117K
- c. 117M
- d. 117B

The LLM Era – Paradigm Shift in Machine Learning

BERT
Oct 2018

Representation



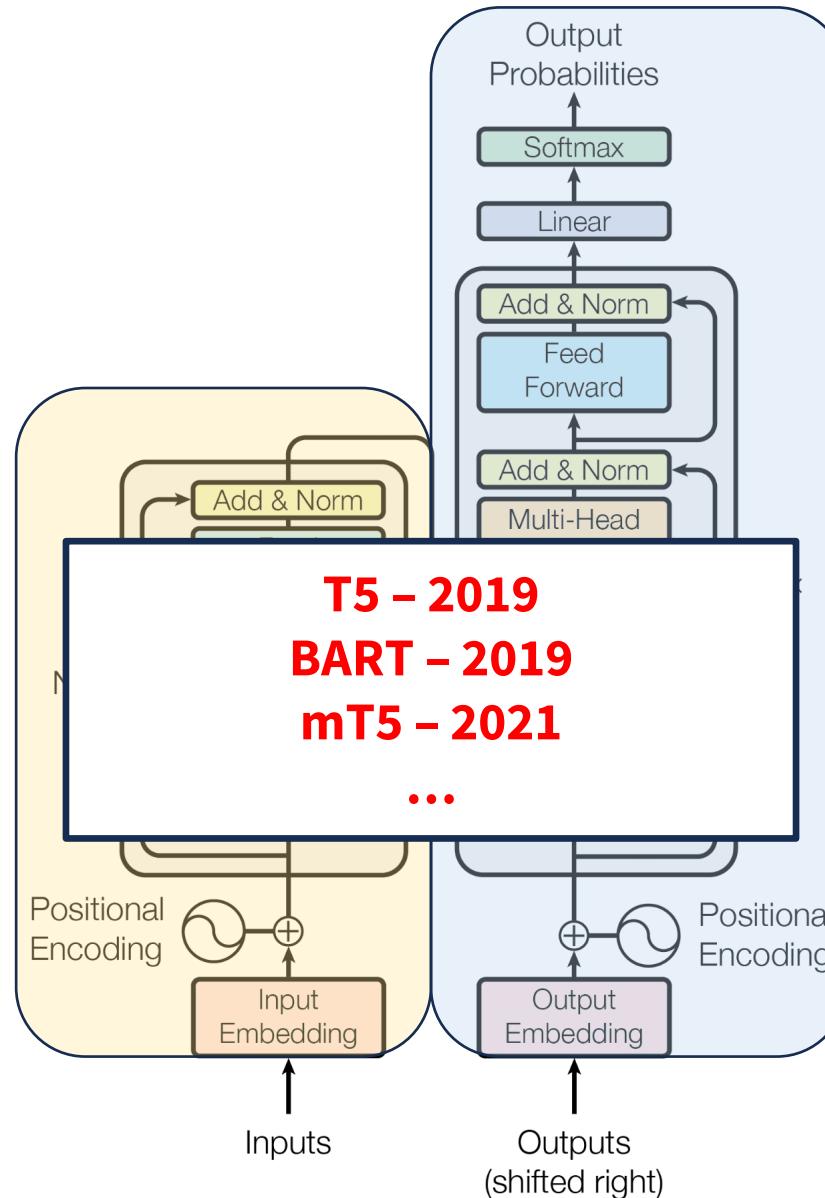
GPT
Jun 2018

Generation

The LLM Era – Paradigm Shift in Machine Learning

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

Representation



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

Generation

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?

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?

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

- 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

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

- Is this all attributable to the power of the attention mechanism?

The LLM Era – Paradigm Shift in Machine Learning

- Is this all attributable to the power of the attention mechanism?
 - Problem in recurrent networks
 - Information is effectively lost during encoding of long sequences
 - Sequential nature encumbers parallel training and favors late timestep inputs

The LLM Era – Paradigm Shift in Machine Learning

- Is this all attributable to the power of the attention mechanism?
 - Problem in recurrent networks
 - Information is effectively lost during encoding of long sequences
 - Sequential nature encumbers parallel training and favors late timestep inputs
 - Solution: Attention mechanism
 - Handling long-range dependencies
 - Parallel training
 - Dynamic attention weights based on inputs

Poll 4 @ 1292

What might be a flaw of our current Transformer-based LLMs?

Answer on Piazza @ 1292

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?
 - 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

Language models just being programmed to try to predict the next word is true, but it's not the dunk some people think it is.

Animals, including us, are just programmed to try to survive and reproduce, and yet amazingly complex and beautiful stuff comes from it.

- Sam Altman