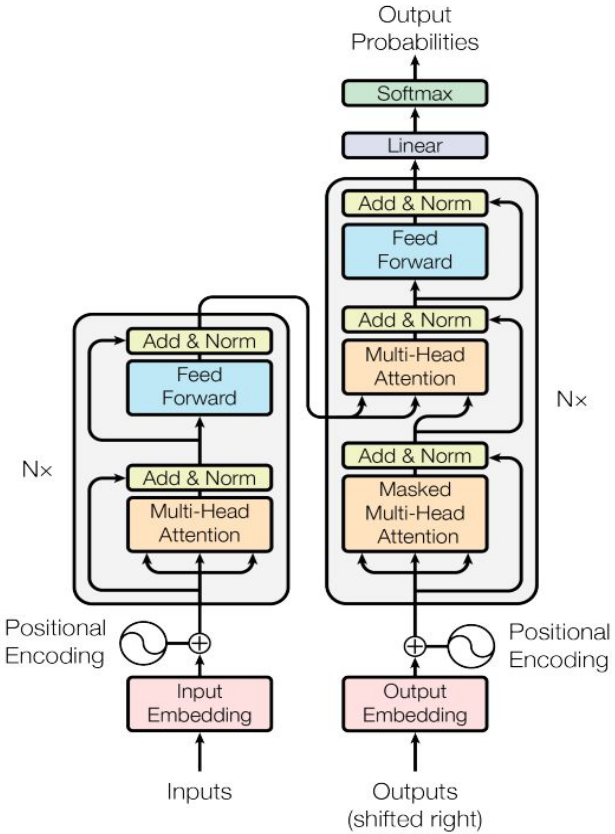
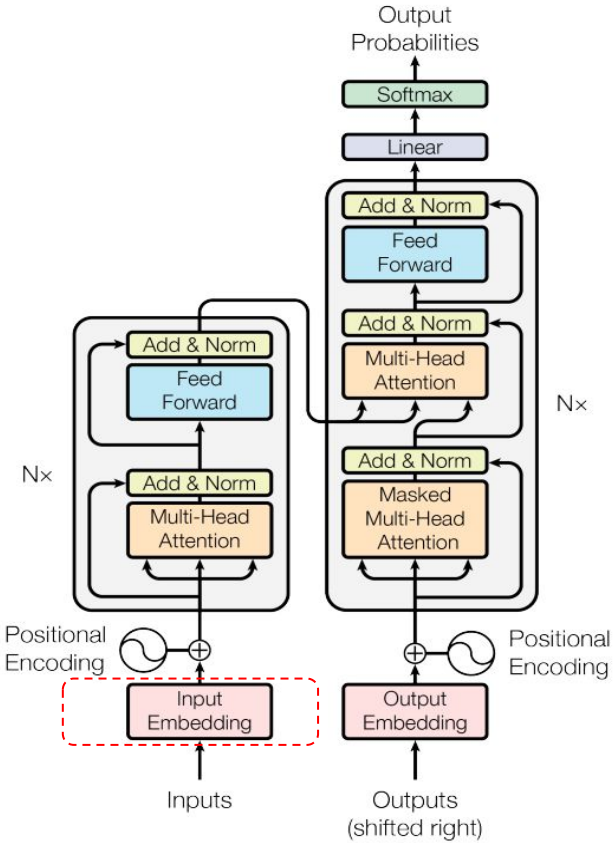
Transformers - Architecture

1. Introduction
2. The Architecture of the Transformer
3. Encoder
    - 3.1. Input Embedding
    - 3.2. Positional Encoding
    - 3.3. Multi- Headed Attention
    - 3.4. Self-attention
    - 3.5. Layer Normalisation
    - 3.6. Point-wise feed forward
4. Decoder

Transformers



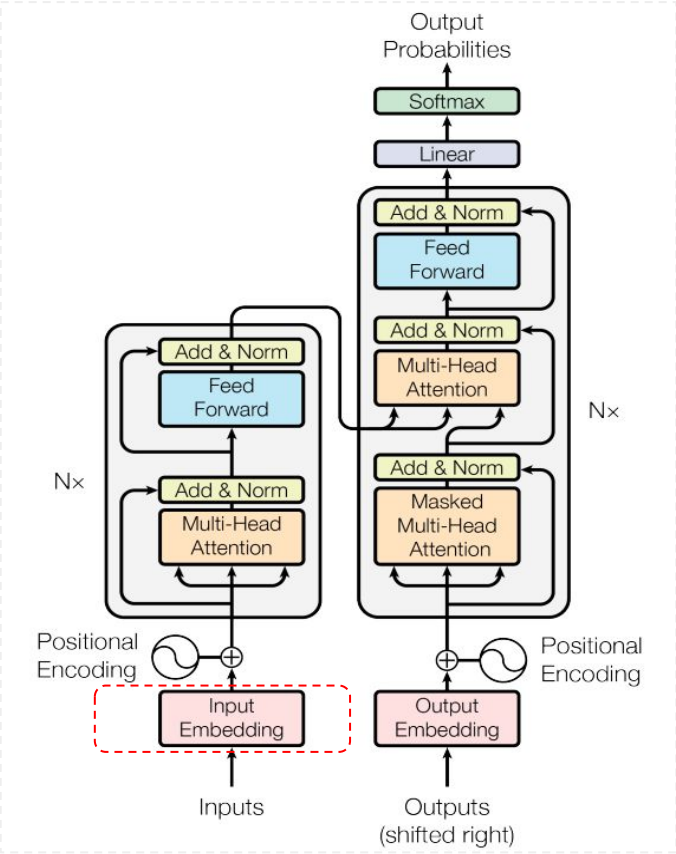
Encoder



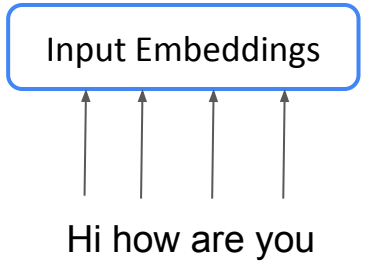
1. Input Embedding

Hi how are you

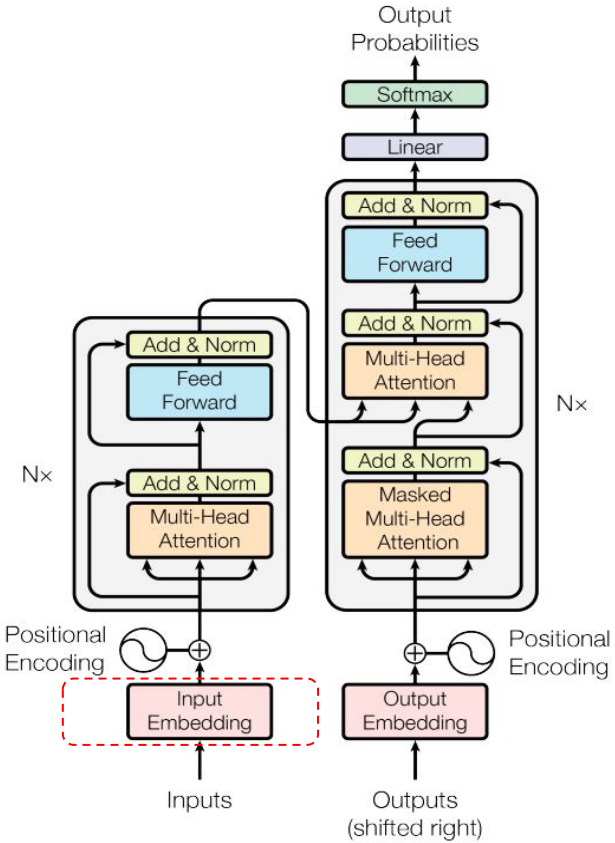
Encoder



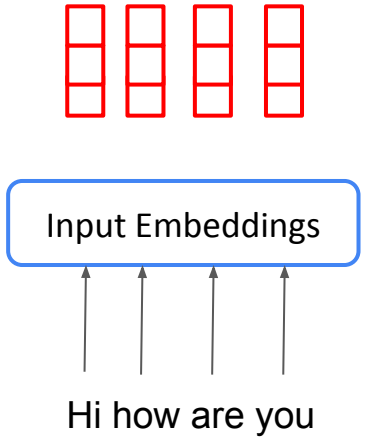
1. Input Embedding



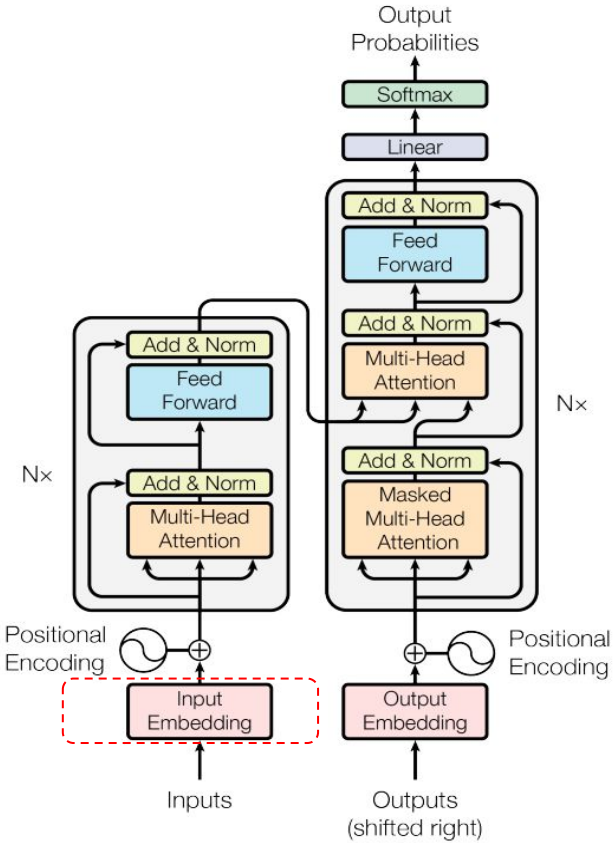
Encoder



1. Input Embedding



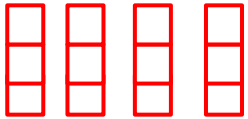
Encoder



1. Input Embedding

Hi =

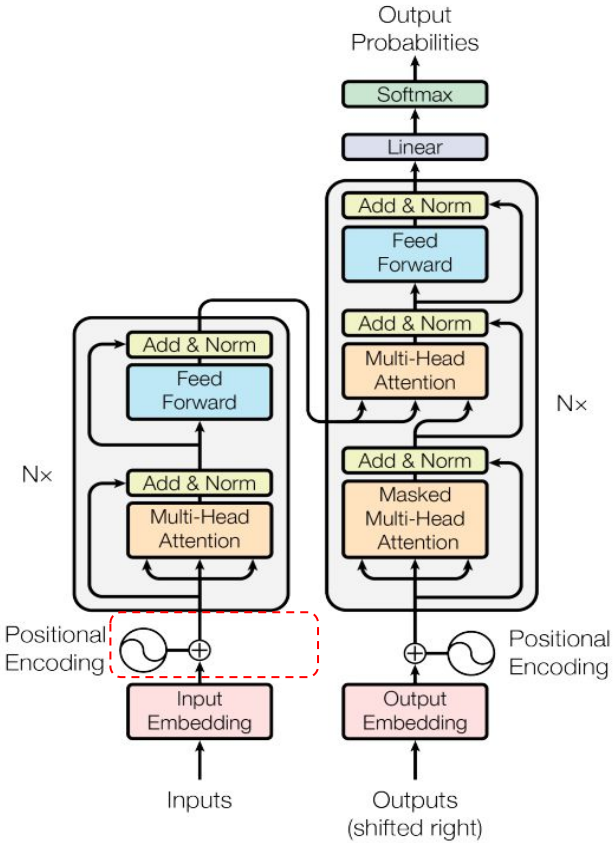
0.1
0.8
0.4



Input Embeddings

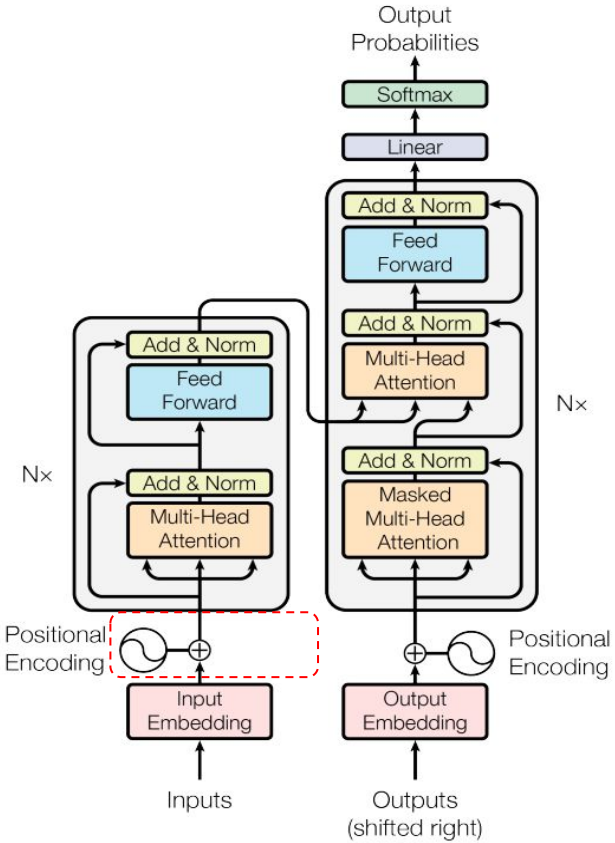
Hi how are you

Encoder

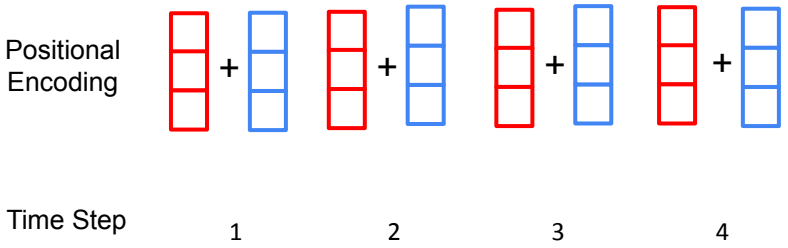


2. Positional Encoding

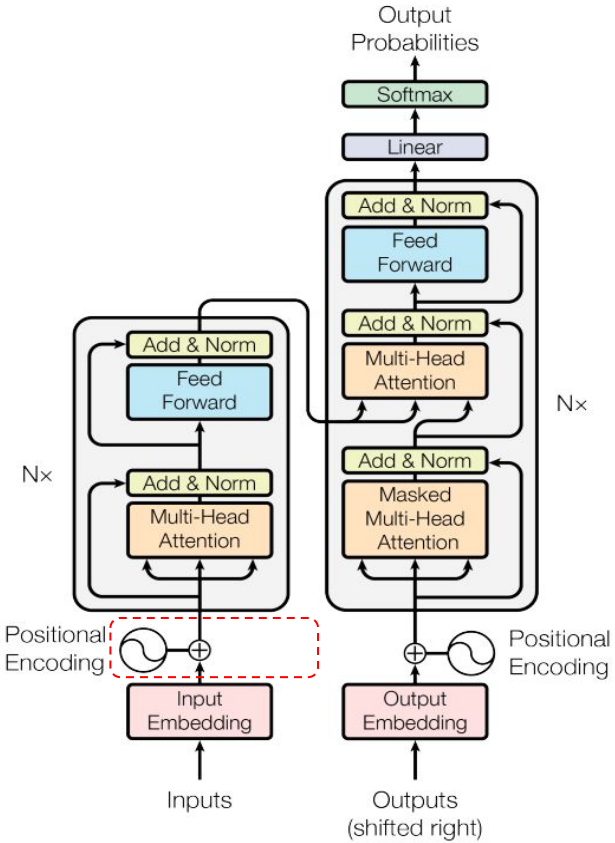
Transformers



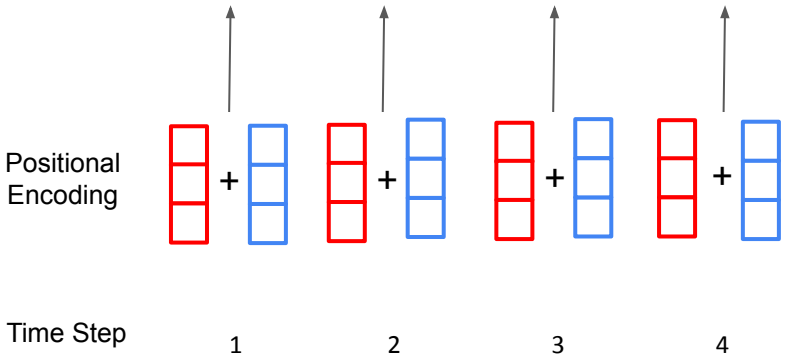
2. Positional Encoding



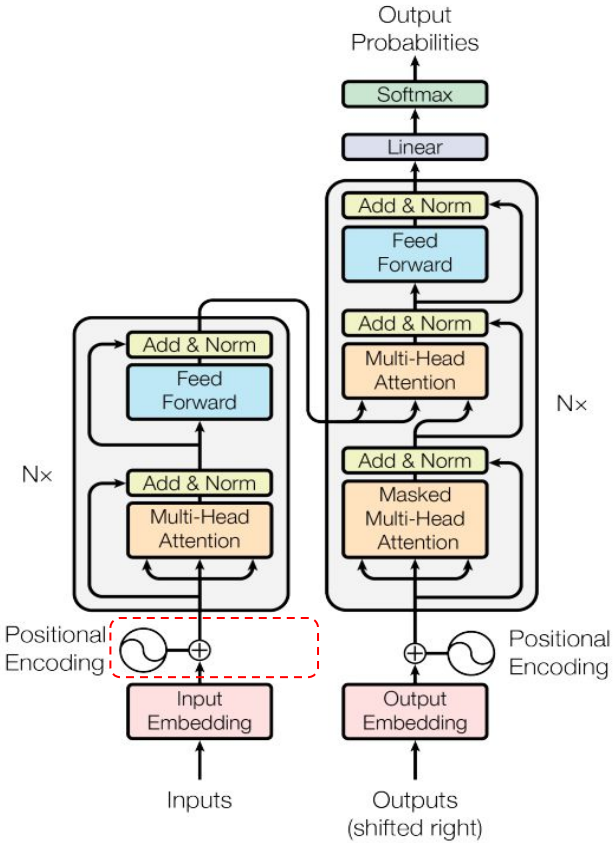
Transformers



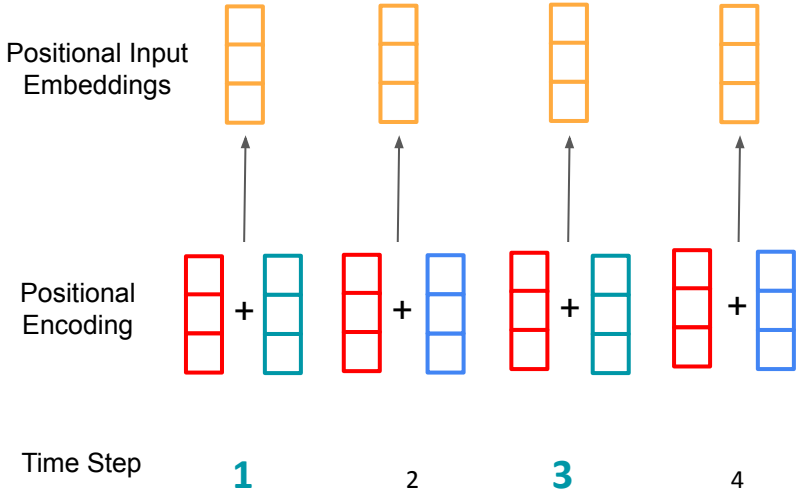
2. Positional Encoding



Encoder

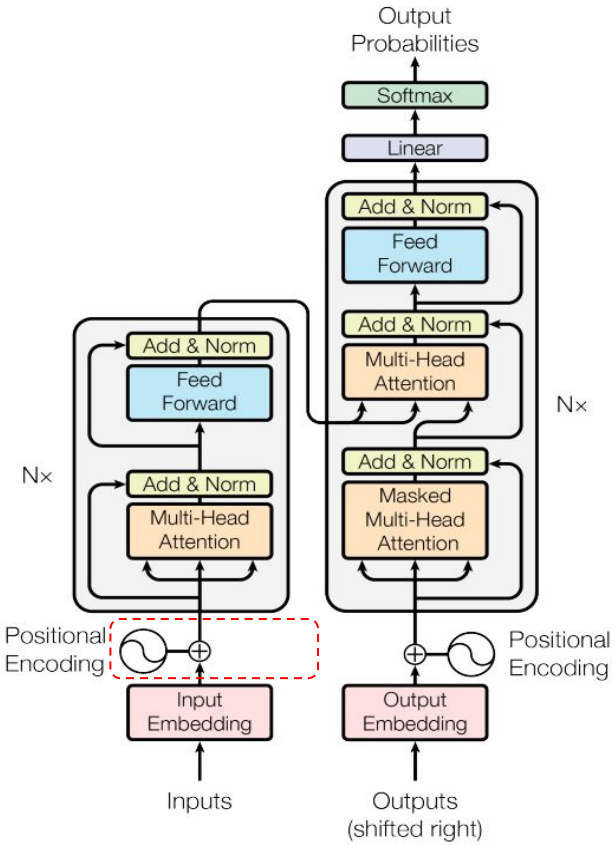


2. Positional Encoding

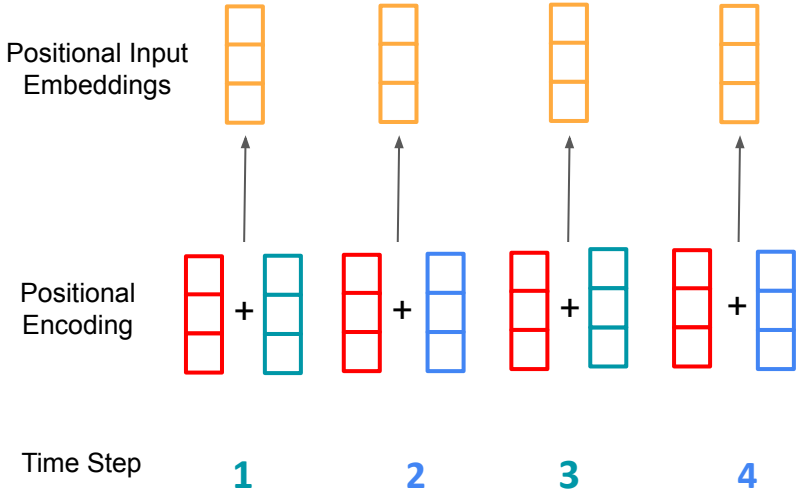


$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Encoder



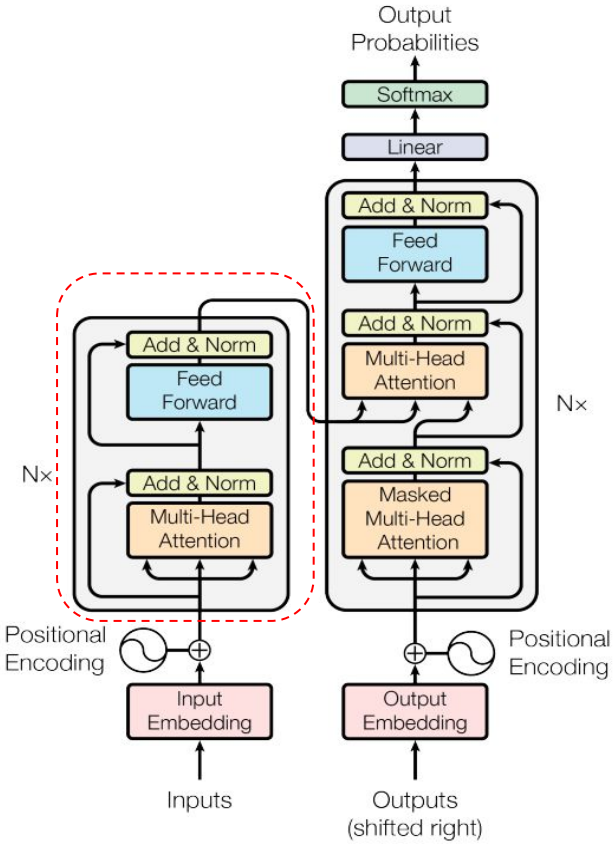
2. Positional Encoding



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

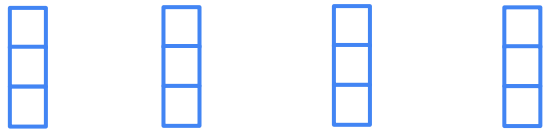
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

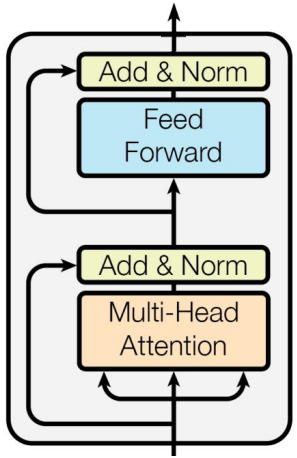
Encoder



2. Encoder Layer

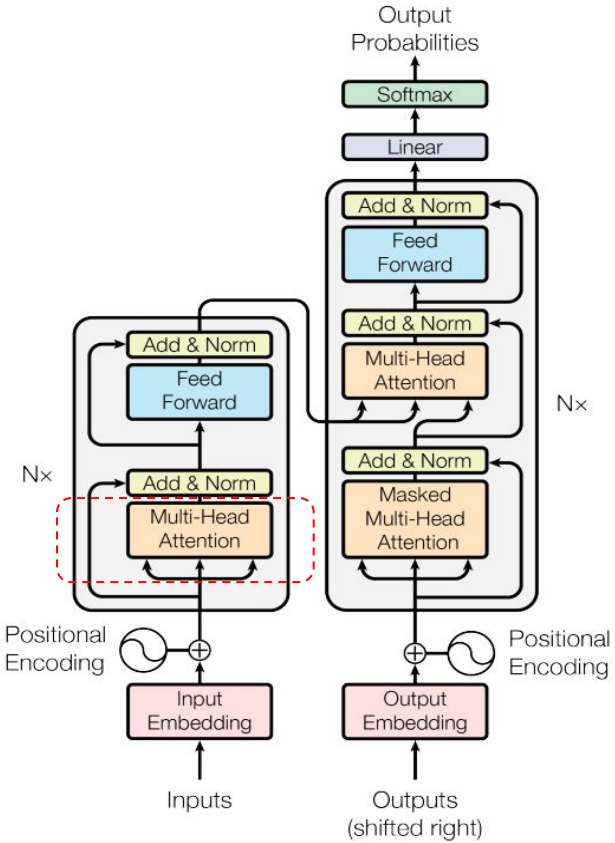
Encoder input Representation





Positional input Embeddings

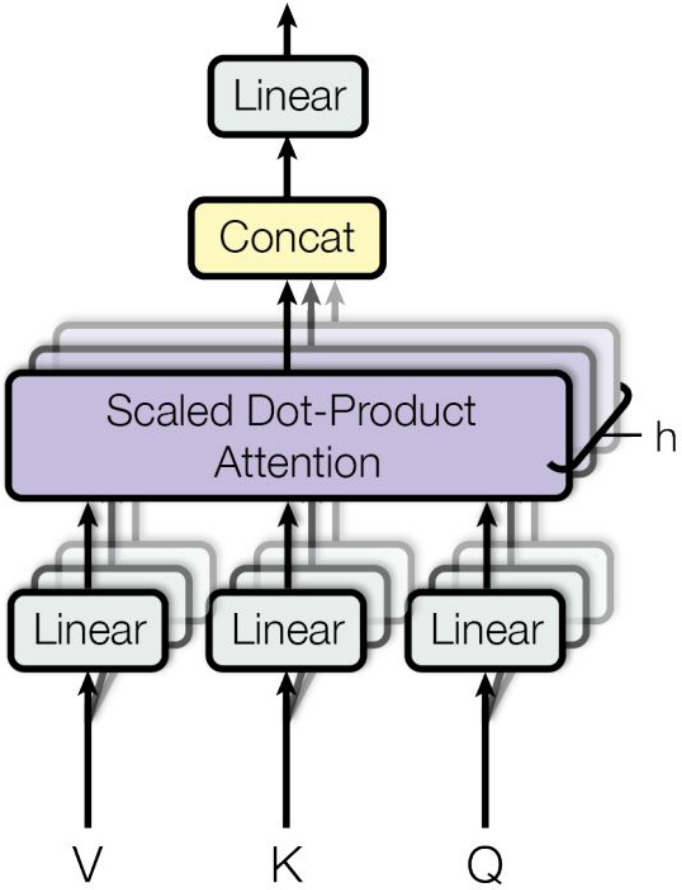
Encoder



3. Multi- Headed Attention

Hi how are you

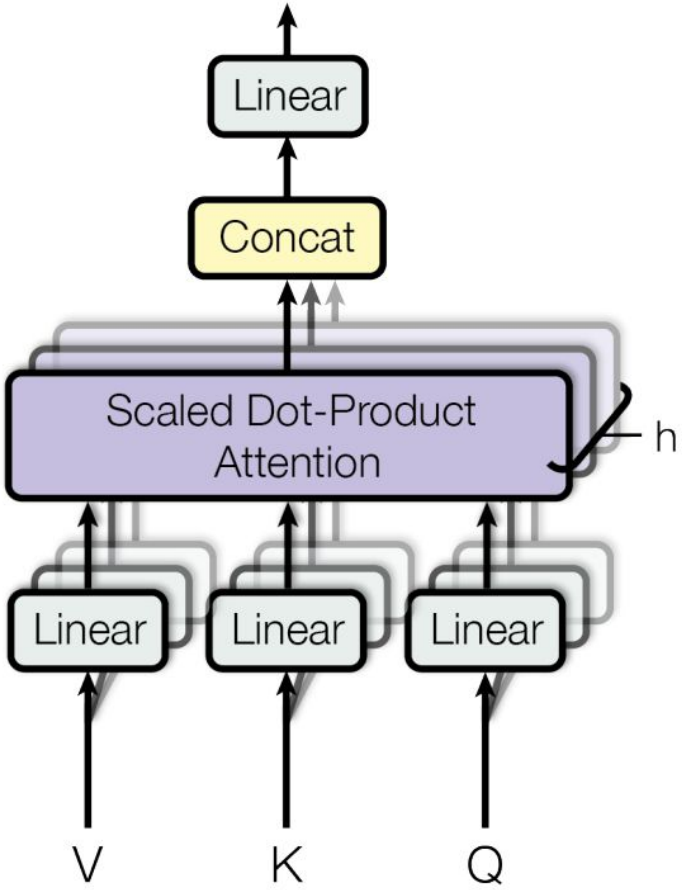
Encoder



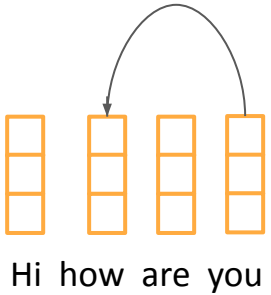
3. Multi- Headed Attention

Hi how are you

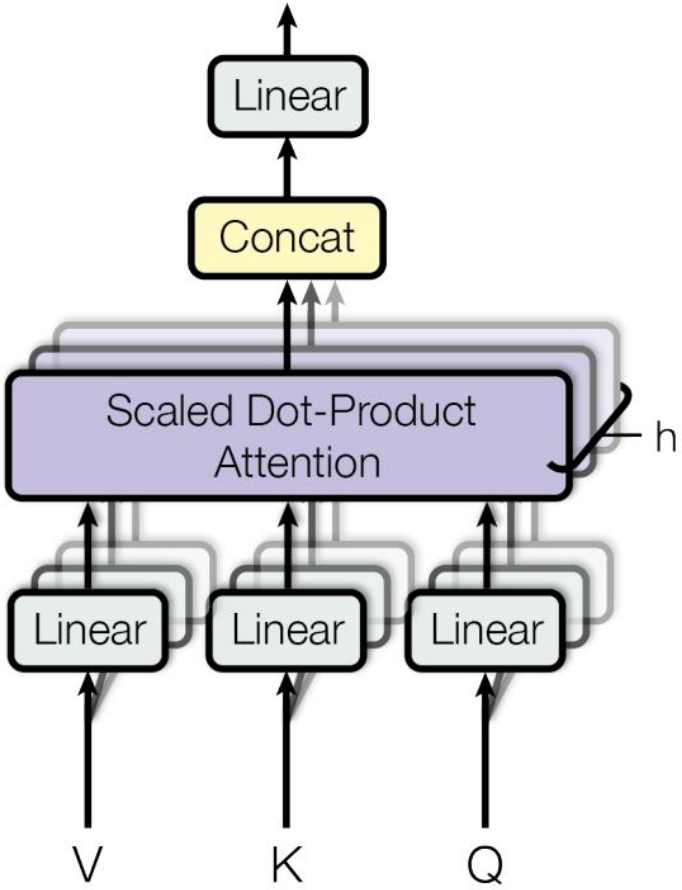
Encoder



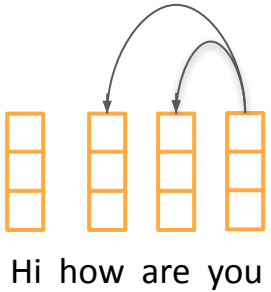
3. Multi- Headed Attention



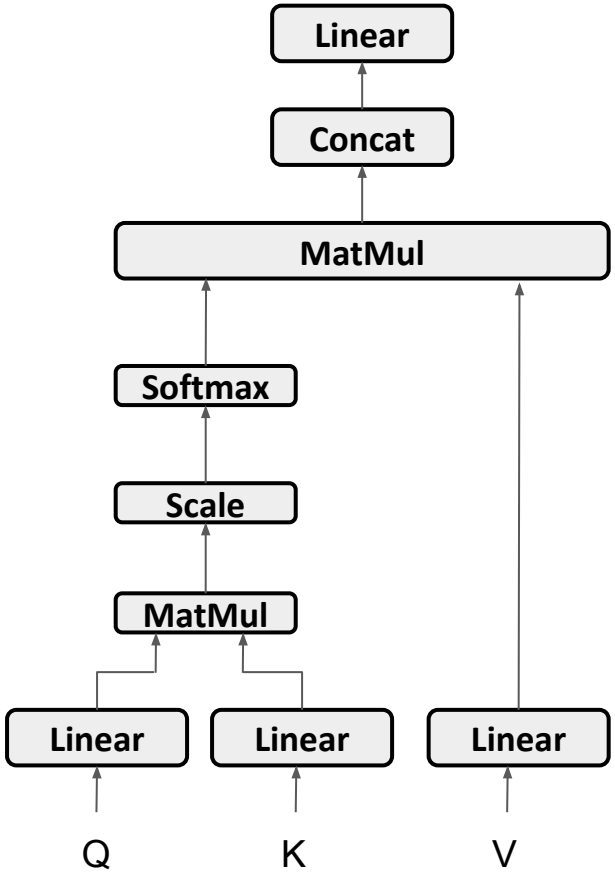
Encoder



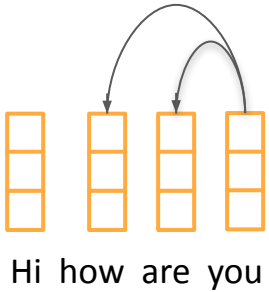
3. Multi- Headed Attention



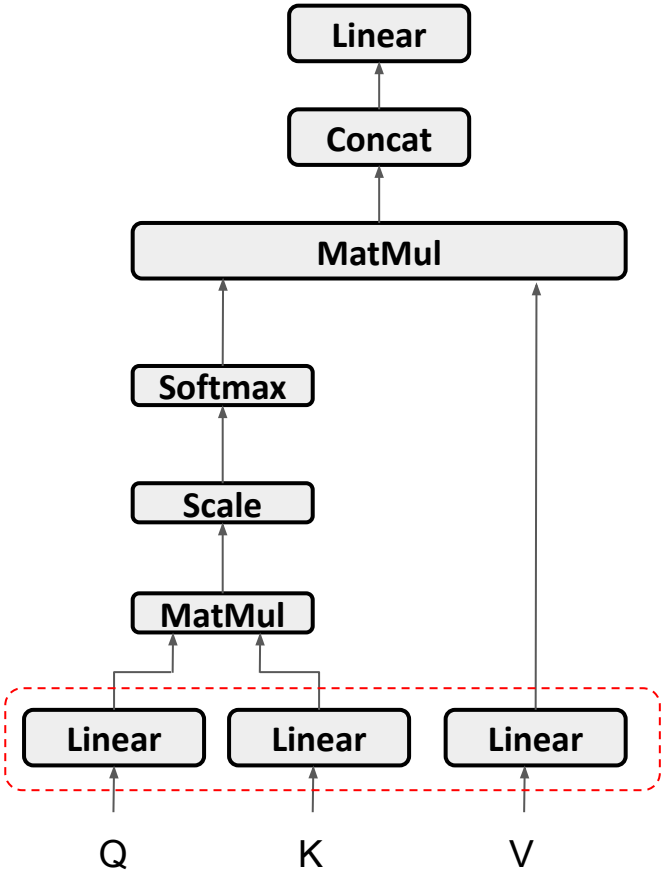
Encoder



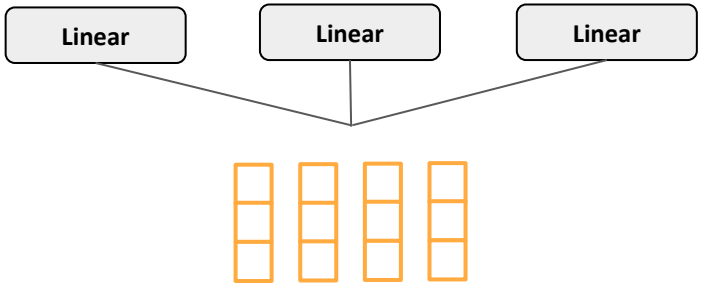
3. Multi- Headed Attention



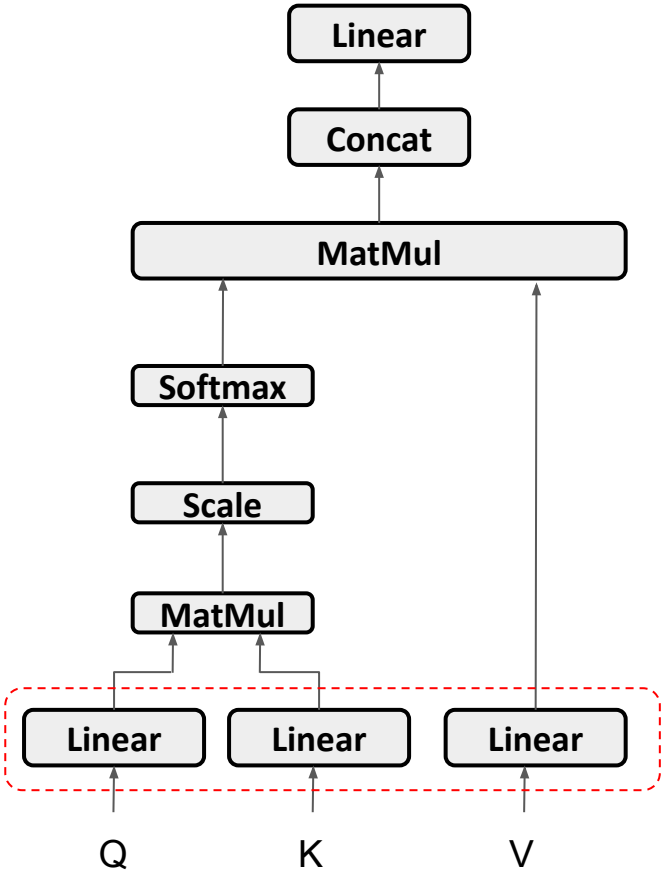
Encoder



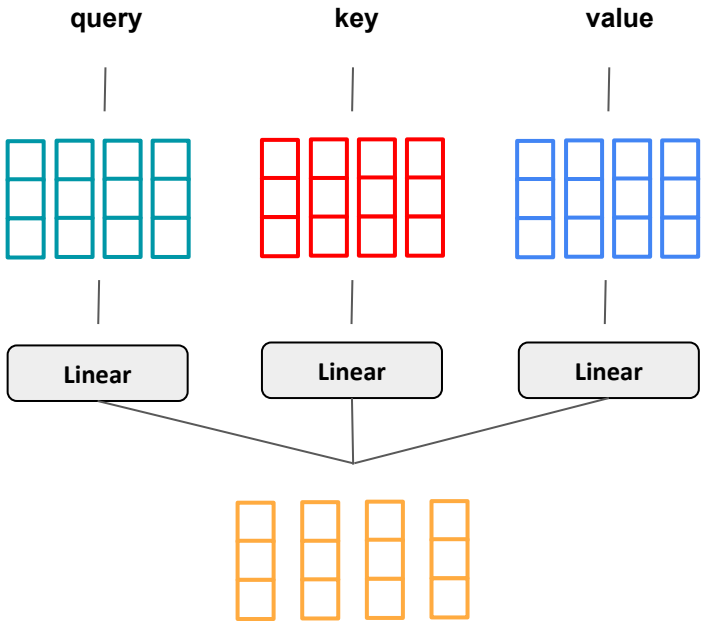
3. Multi- Headed Attention



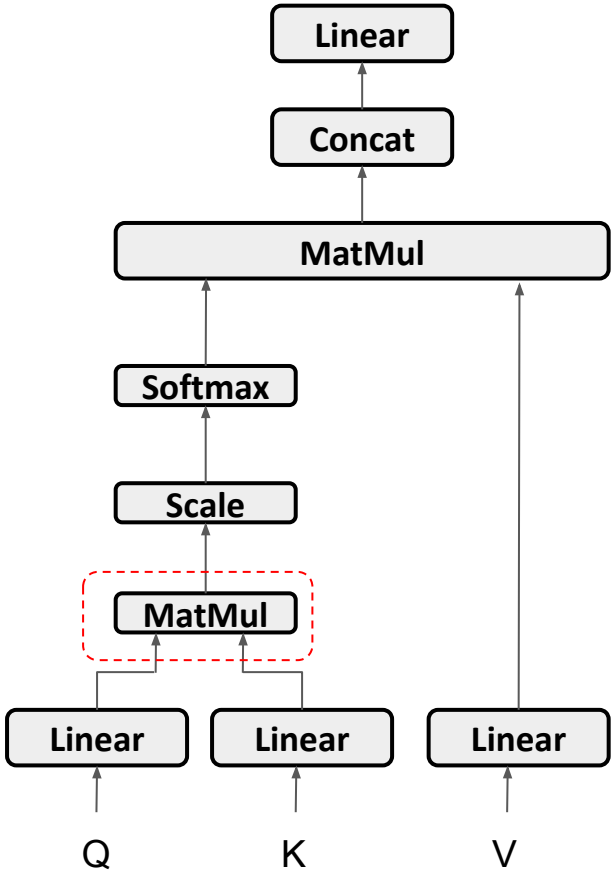
Transformers



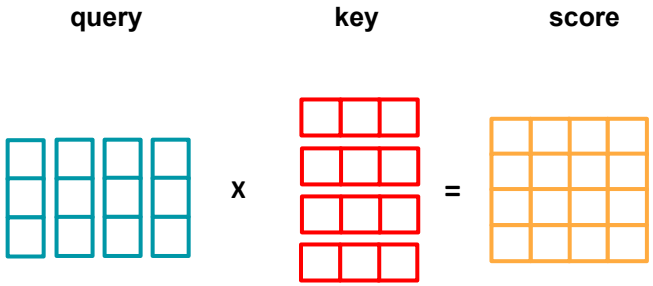
3. Multi- Headed Attention

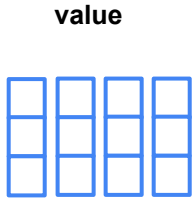


Transformers

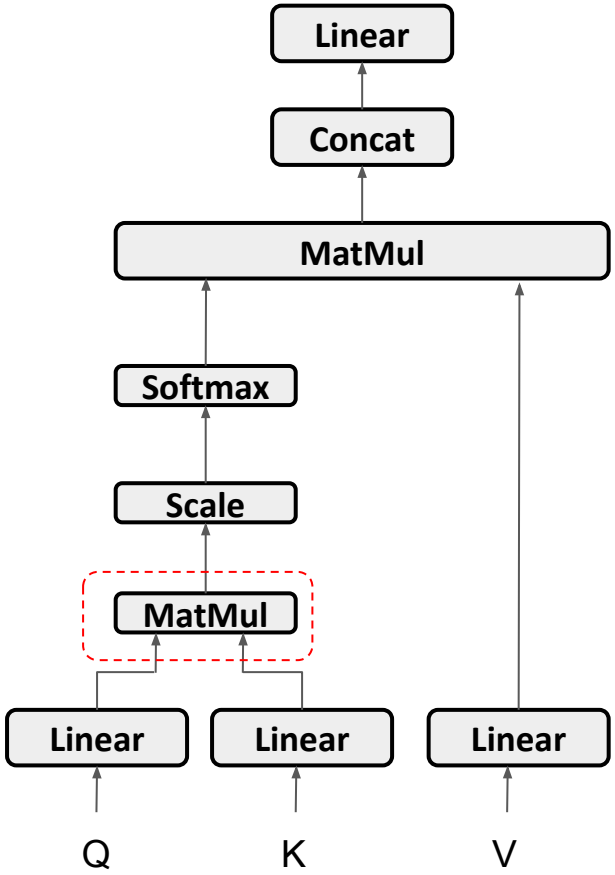


3. Multi- Headed Attention





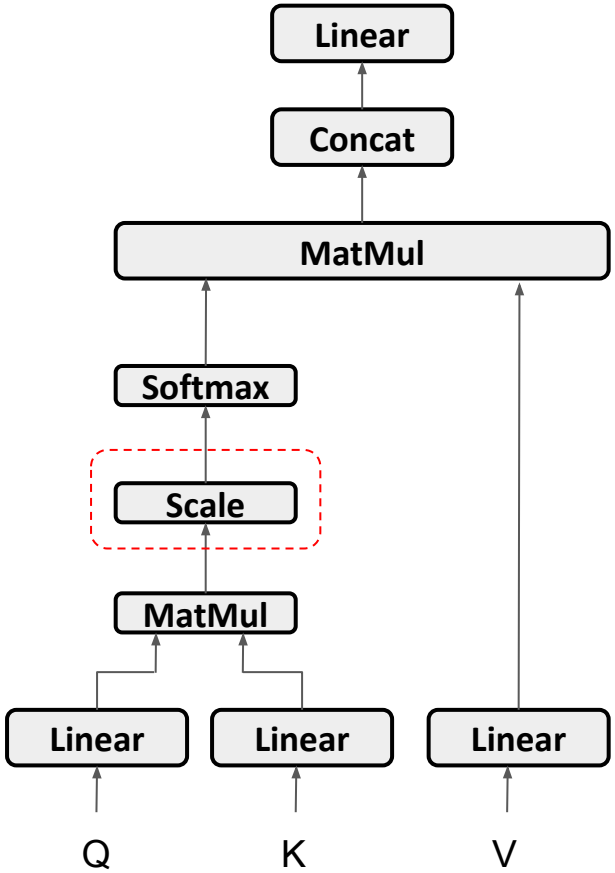
Encoder



3. Multi- Headed Attention

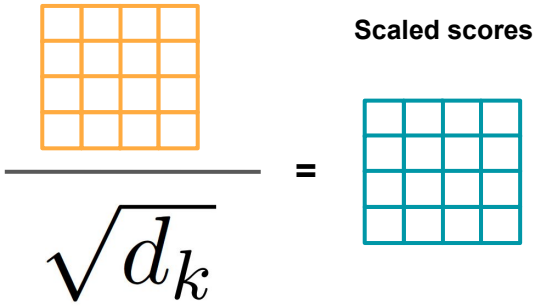
	Hi	how	are	you
Hi	96	25	8	10
how	25	87	29	65
are	8	29	89	52
you	10	65	52	90

Transformers

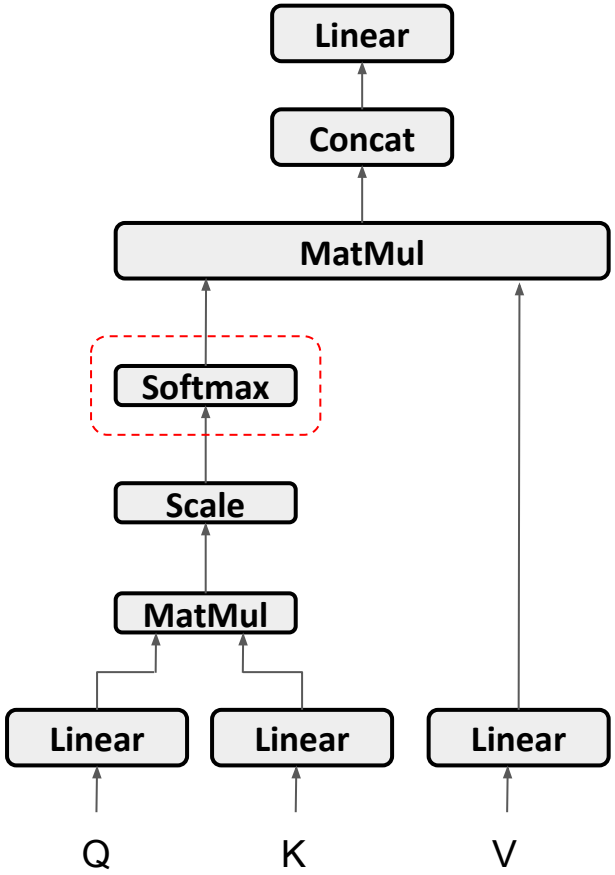


3. Multi- Headed Attention

3.1 Self-Attention



Encoder



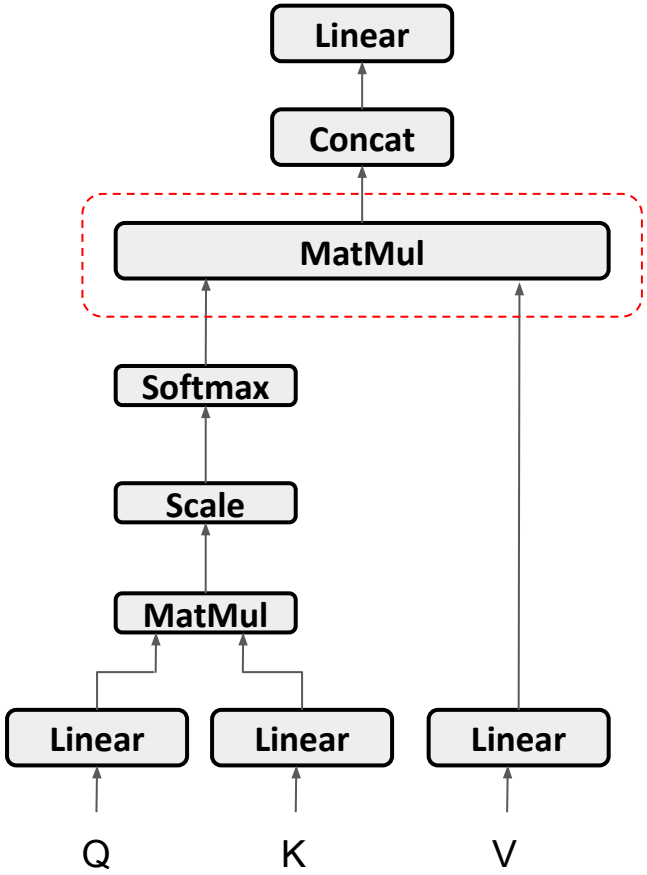
3. Multi- Headed Attention

$$\text{Softmax}\left(\boxed{}\right) =$$

attention weights

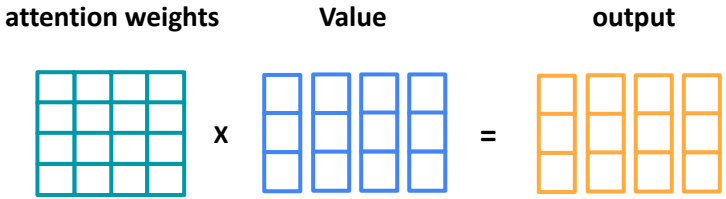
	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.2	0.5	0.1	0.2
are	0.1	0.2	0.6	0.1
you	0.1	0.2	0.4	0.3

Encoder

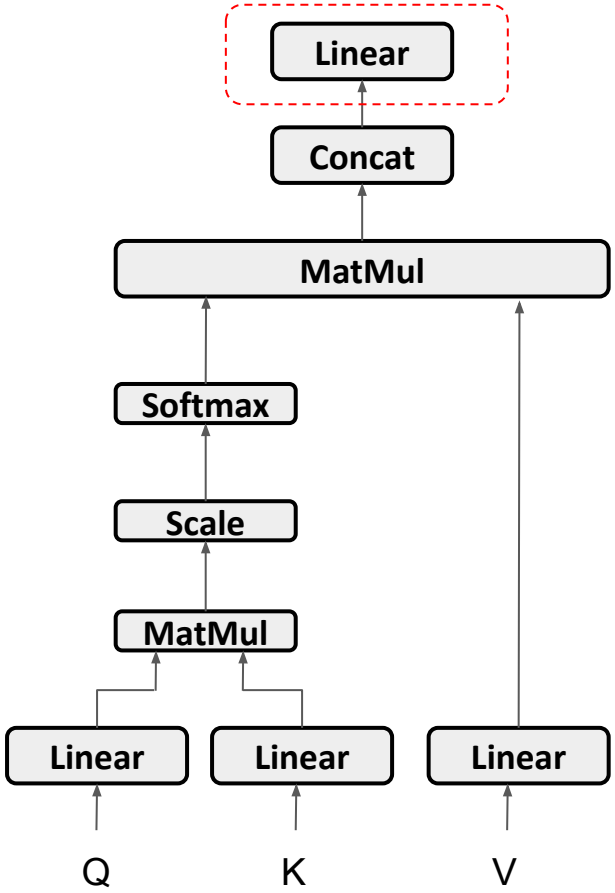


3. Multi- Headed Attention

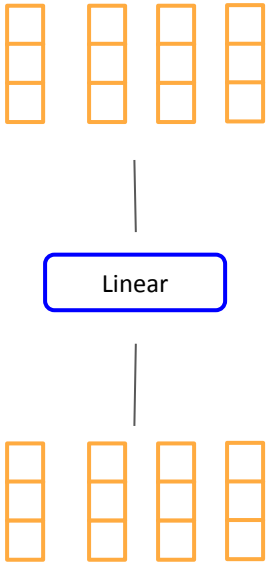
3.1 Self-Attention



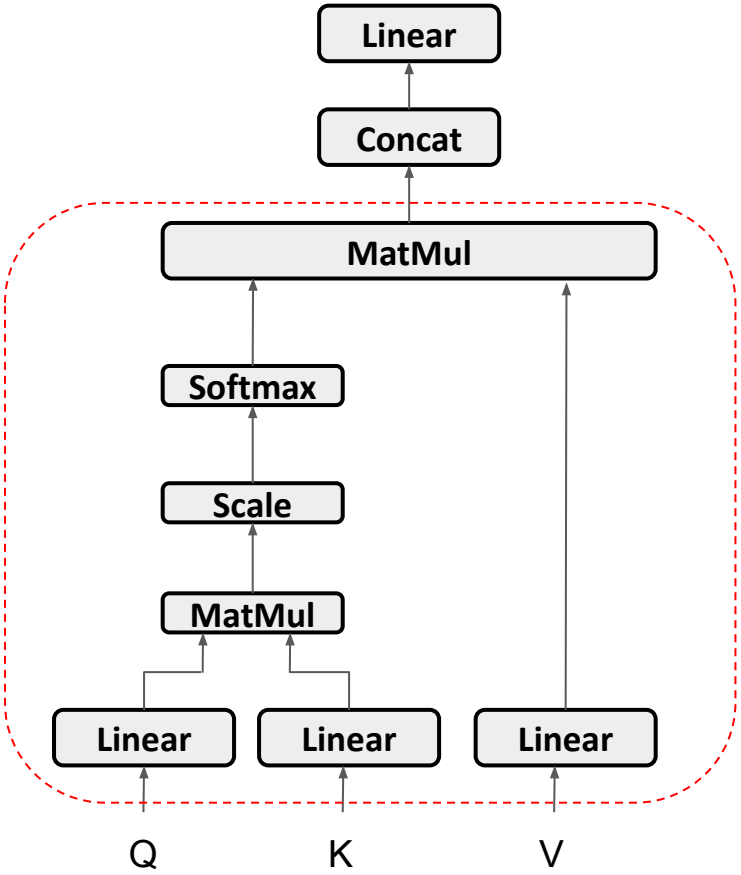
Encoder



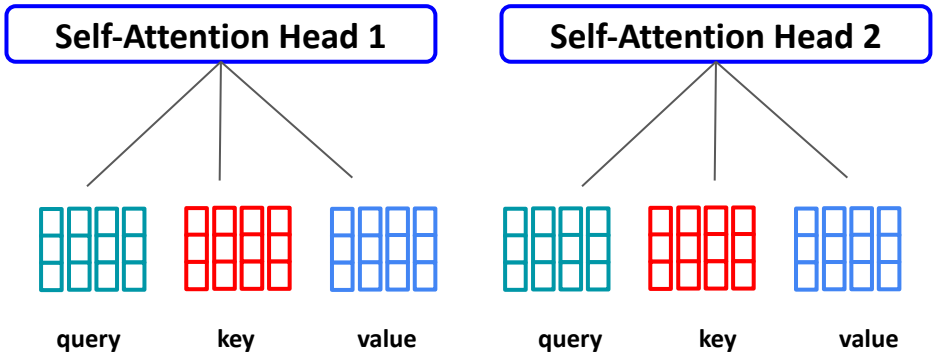
3. Multi- Headed Attention



Encoder

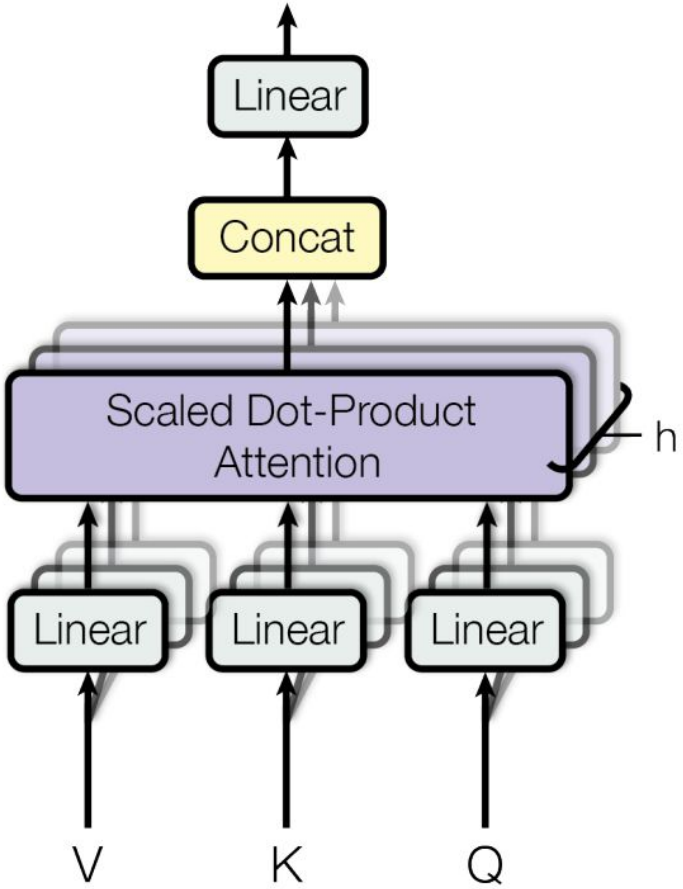


3. Multi- Headed Attention

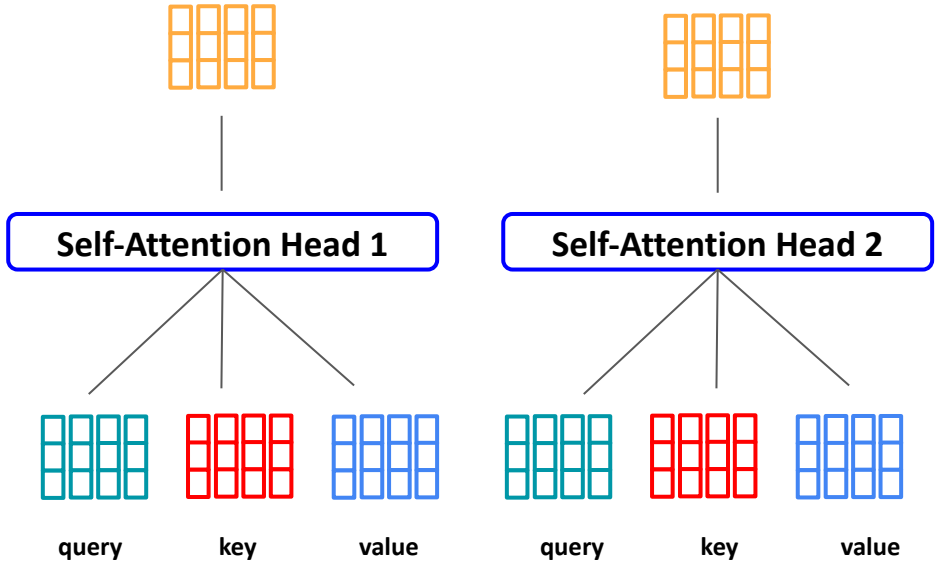


N = 2

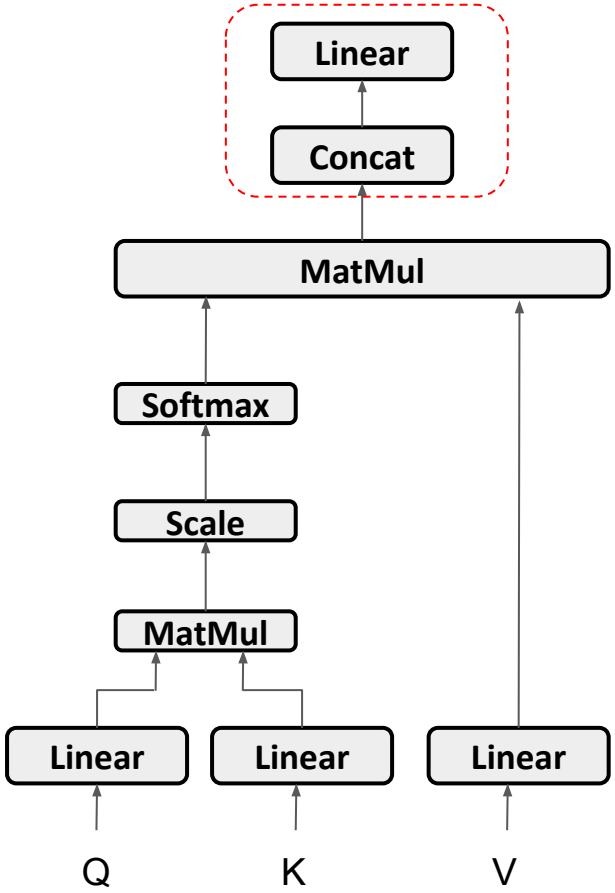
Encoder



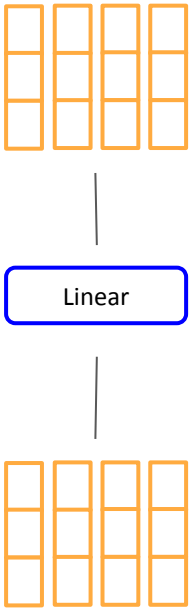
3. Multi- Headed Attention



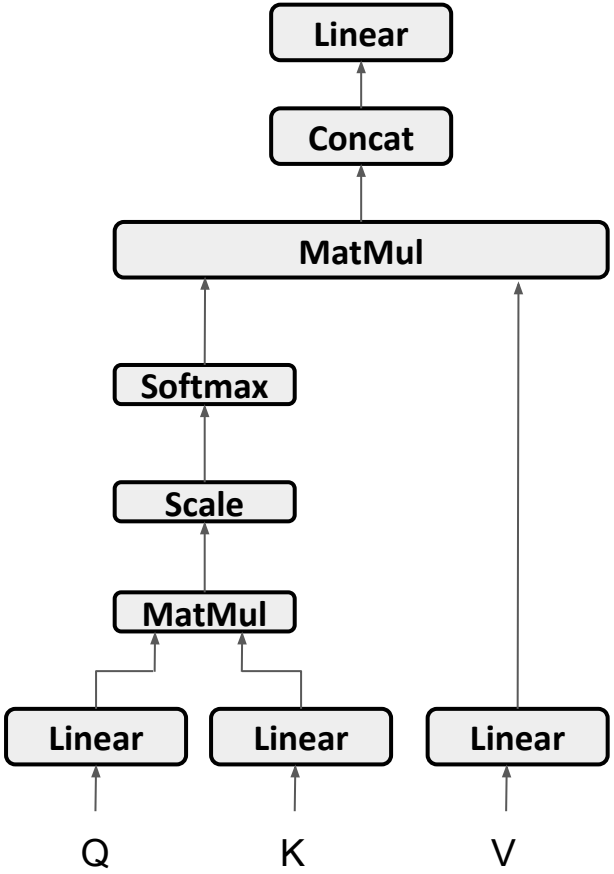
Encoder



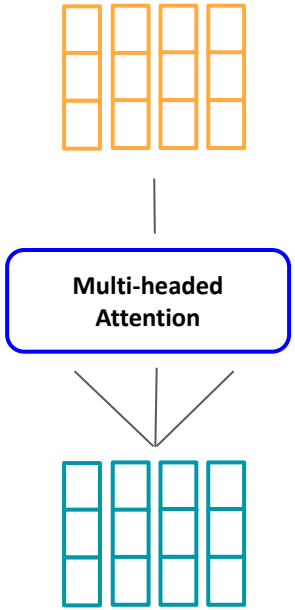
3. Multi- Headed Attention



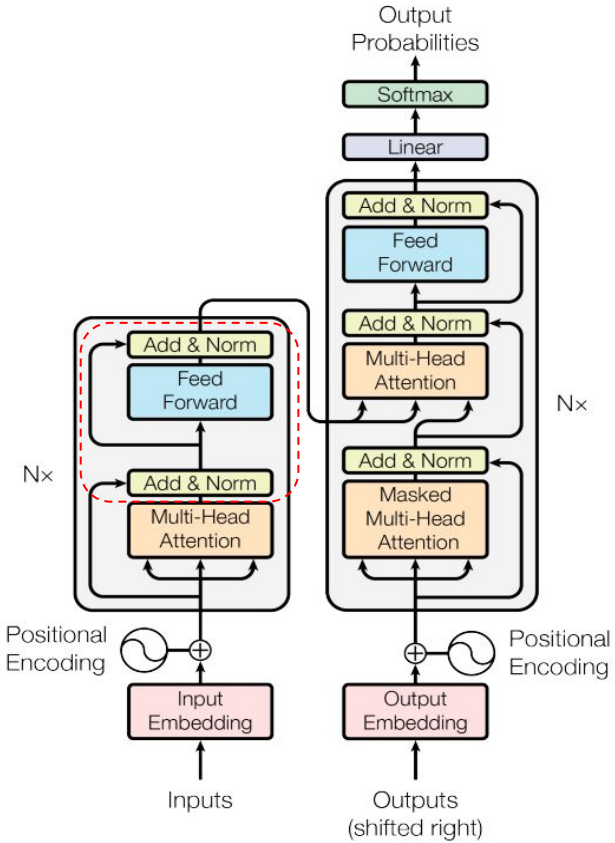
Encoder



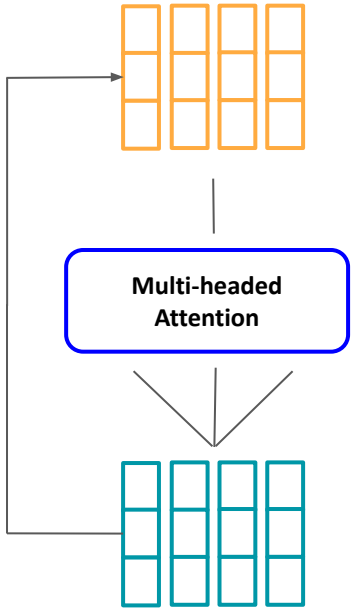
3. Multi- Headed Attention



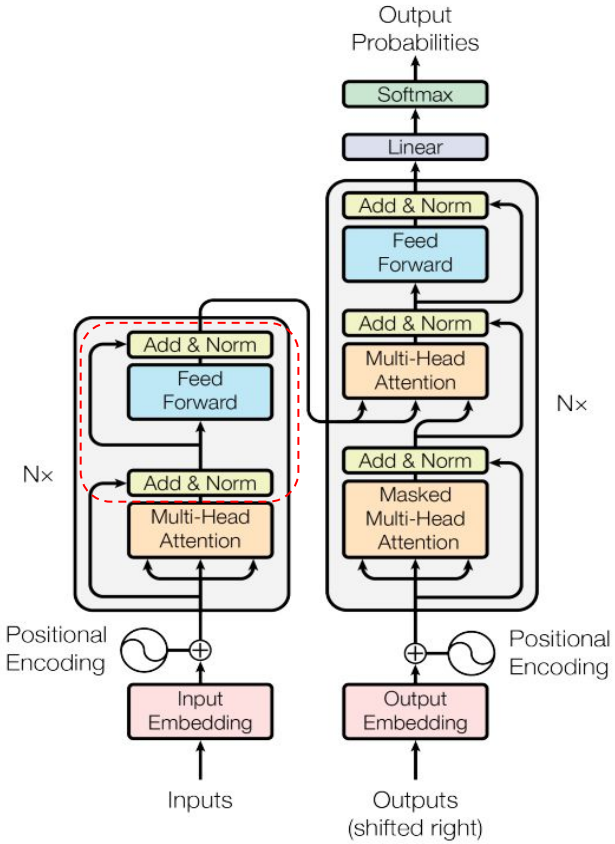
Encoder



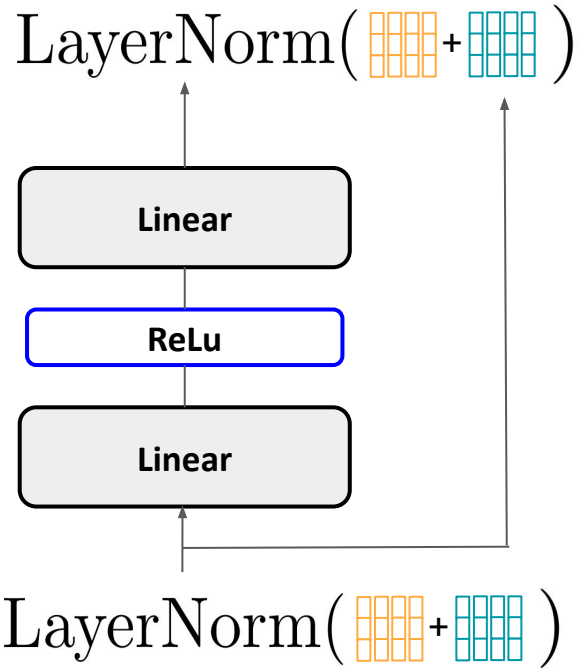
4. Residual Connection



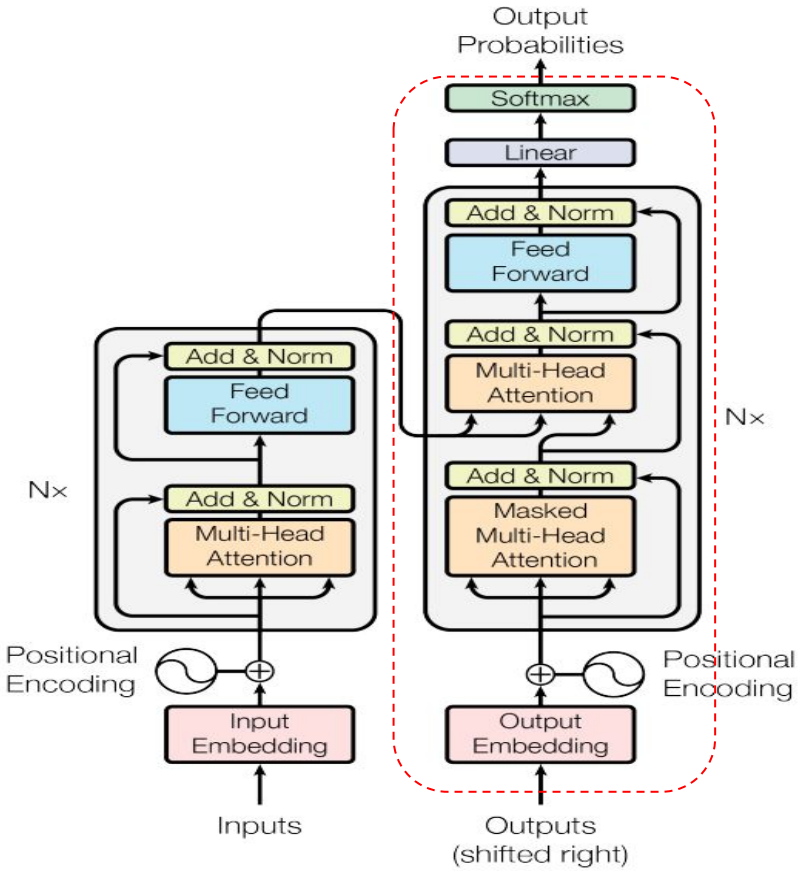
Encoder



5. Layer Normalisation and Point-wise feed forward

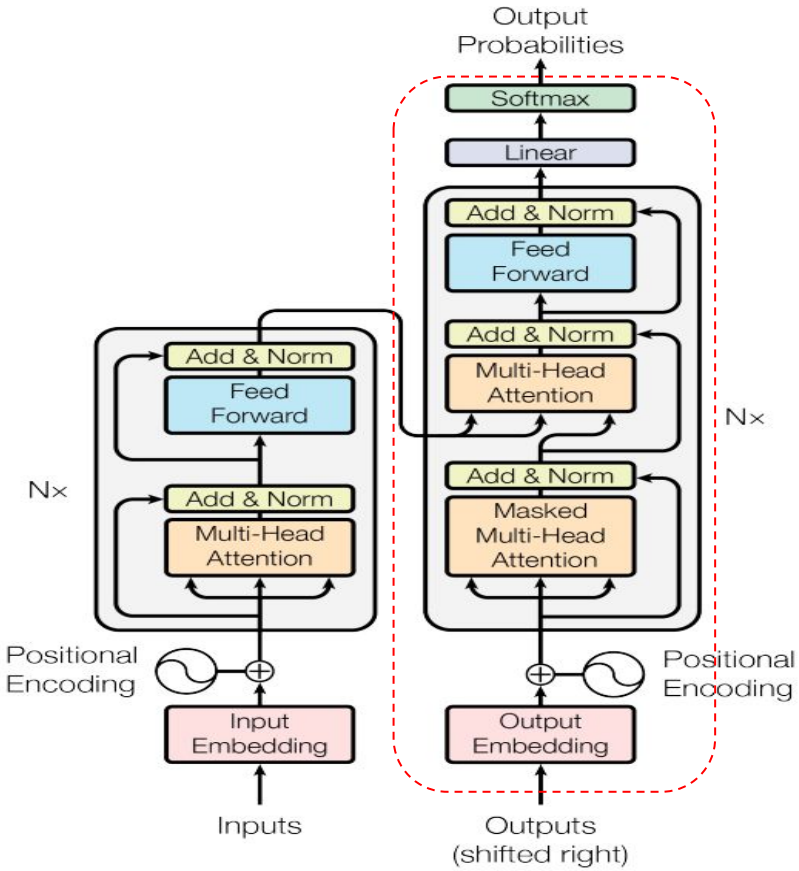


Decoder



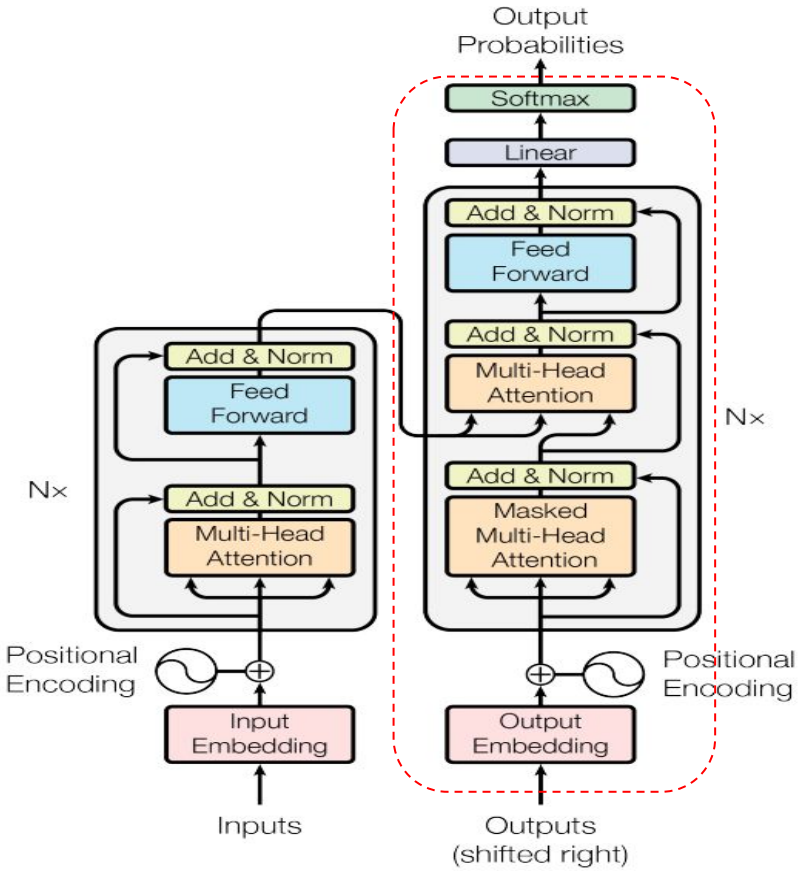
- The decoder‘s job is to generate text sequences

Decoder



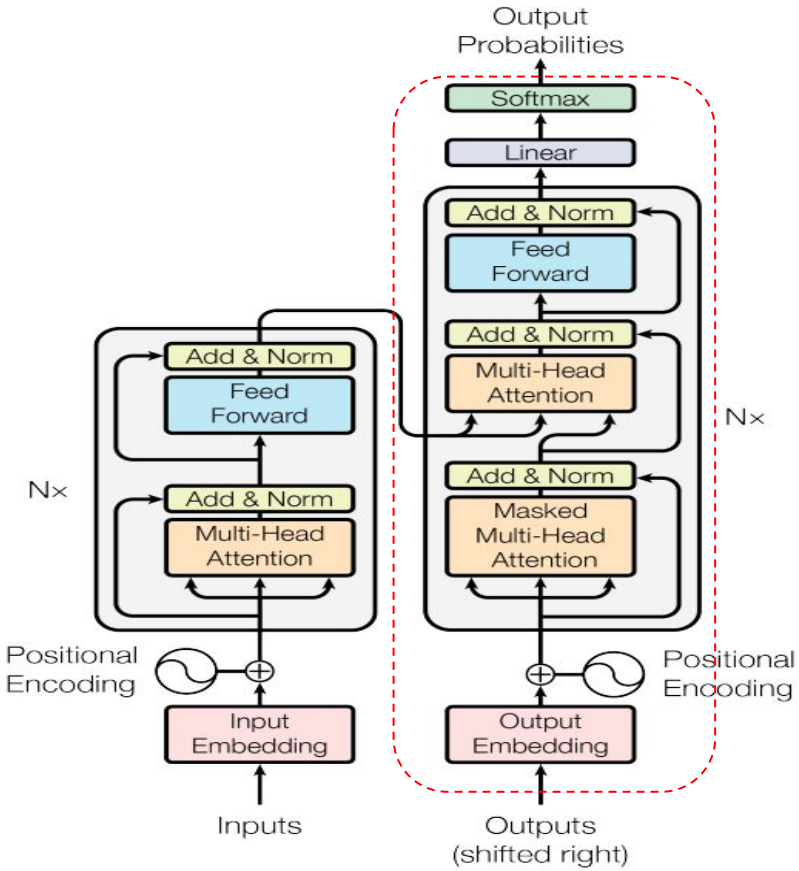
- The decoder‘s job is to generate text sequences
- The decoder has similar sub layers as the encoder
    - Two multi-headed attention layers

Decoder



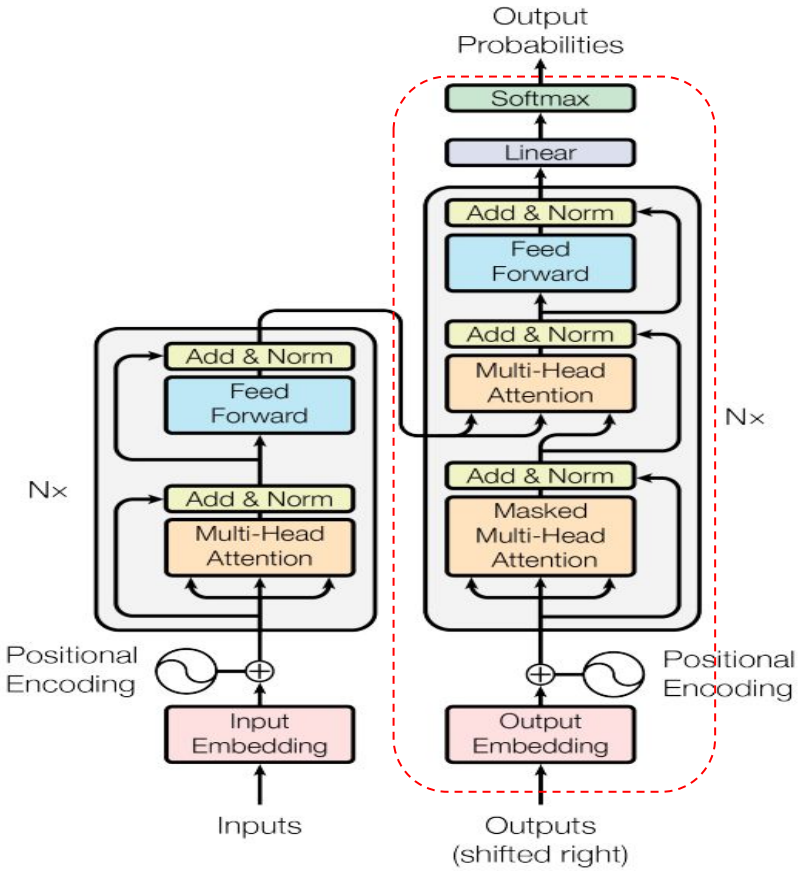
- The decoder‘s job is to generate text sequences
- The decoder has similar sub layers as the encoder
  - Two multi-headed attention layers
  - A point-wise feed-forward layer with residual connections

Decoder



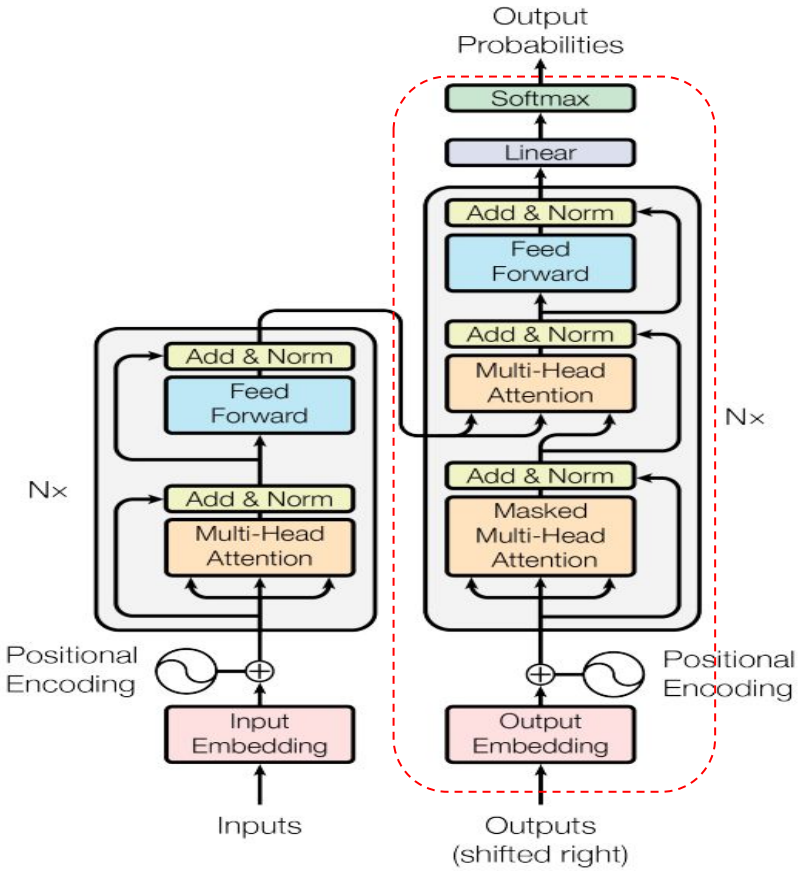
- The decoder‘s job is to generate text sequences
- The decoder has similar sub layers as the encoder
  - Two multi-headed attention layers
  - A point-wise feed-forward layer with residual connections
  - Layer normalization after each sub layer

Decoder



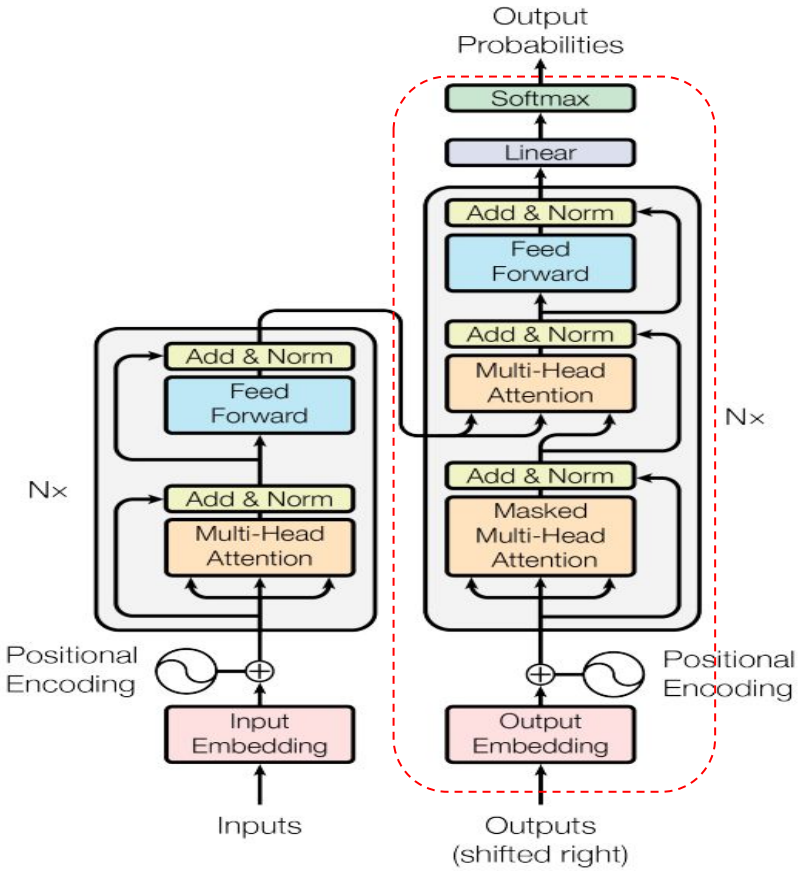
- The decoder‘s job is to generate text sequences
- The decoder has similar sub layers as the encoder
  - Two multi-headed attention layers
  - A point-wise feed-forward layer with residual connections
  - Layer normalization after each sub layer
- A linear layer that acts like a classifier

Decoder



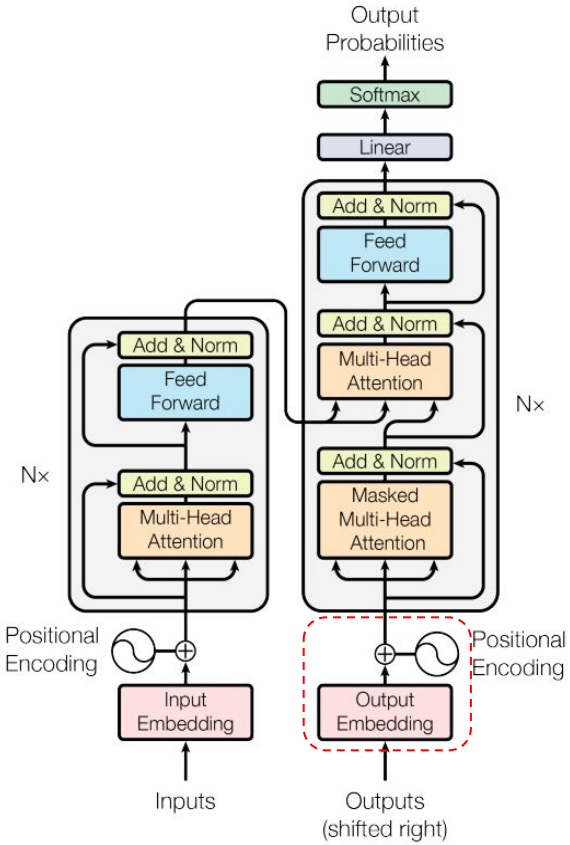
- The decoder‘s job is to generate text sequences
- The decoder has similar sub layers as the encoder
  - Two multi-headed attention layers
  - A point-wise feed-forward layer with residual connections
  - Layer normalization after each sub layer
- A linear layer that acts like a classifier
- The decoder is autoregressive

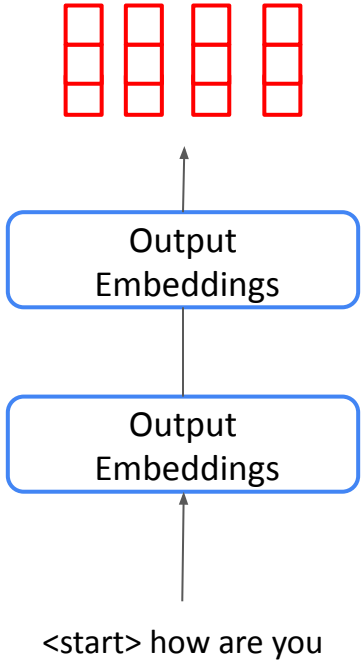
Decoder



- The decoder‘s job is to generate text sequences
- The decoder has similar sub layers as the encoder
  - Two multi-headed attention layers
  - A point-wise feed-forward layer with residual connections
  - Layer normalization after each sub layer
- A linear layer that acts like a classifier
- The decoder is auto regressive
  - takes in the list of previous outputs as inputs and the encoder outputs

Decoder

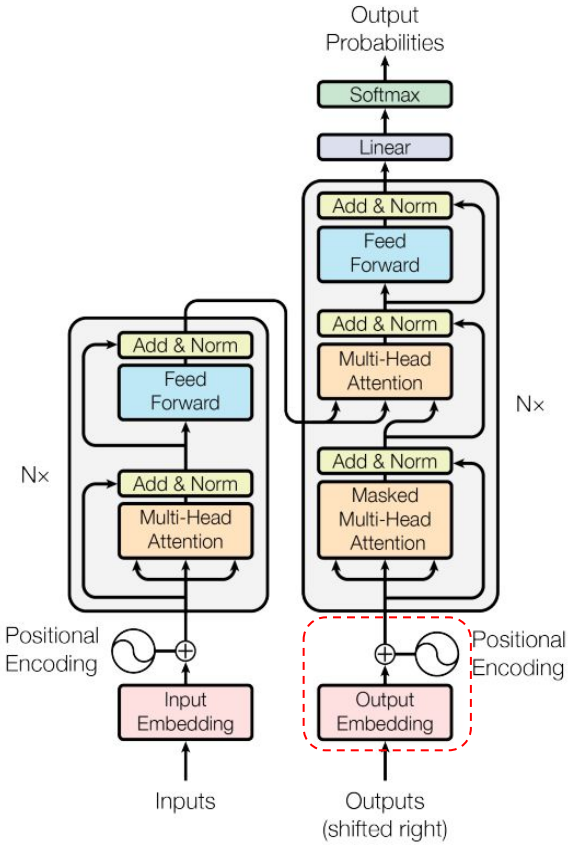


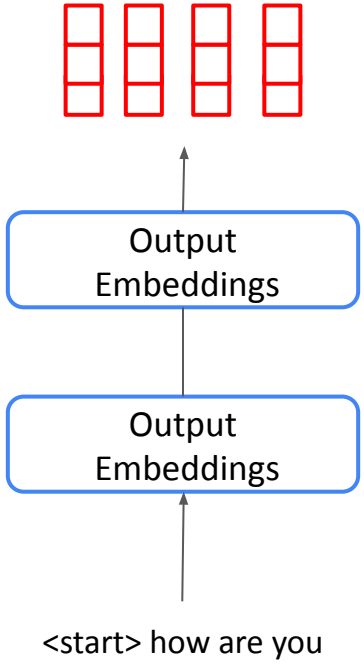


6. Output Embedding and Positional Encoding

- The input goes through an embedding layer

Decoder

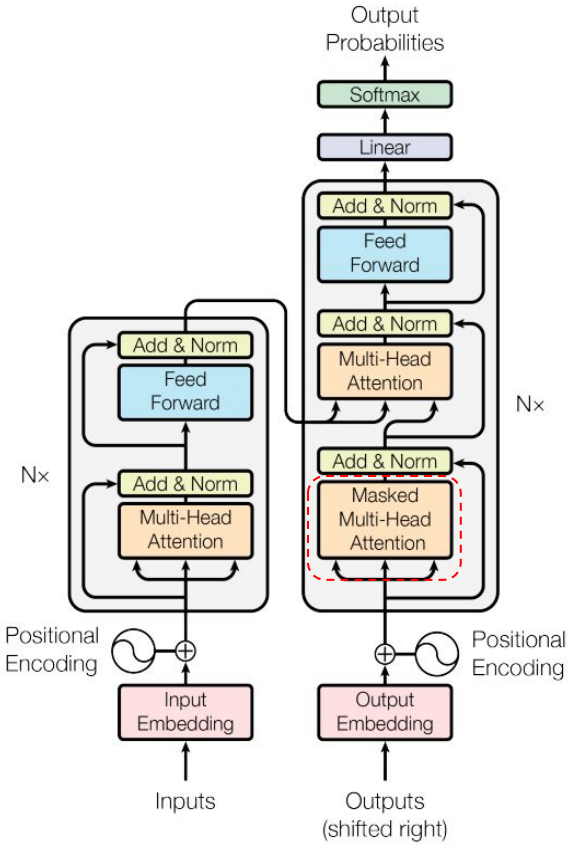


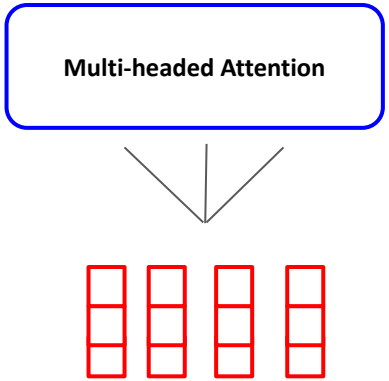


6. Output Embedding and Positional Encoding

- The input goes through an embedding layer
- Positional encoding layer

Decoder

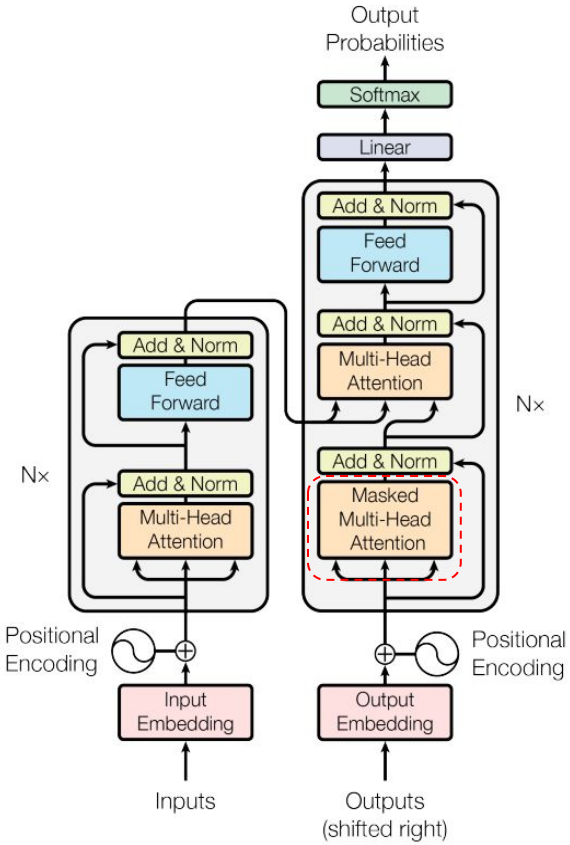


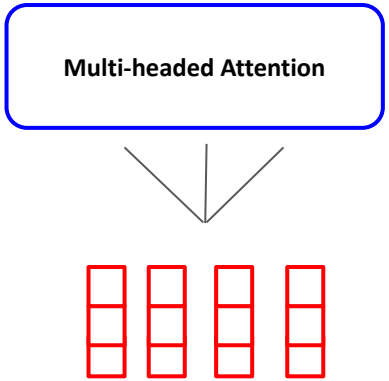


7. Decoder Multi-Head Attention 1

- Positional embeddings get fed into the first multi-headed attention layer

Decoder

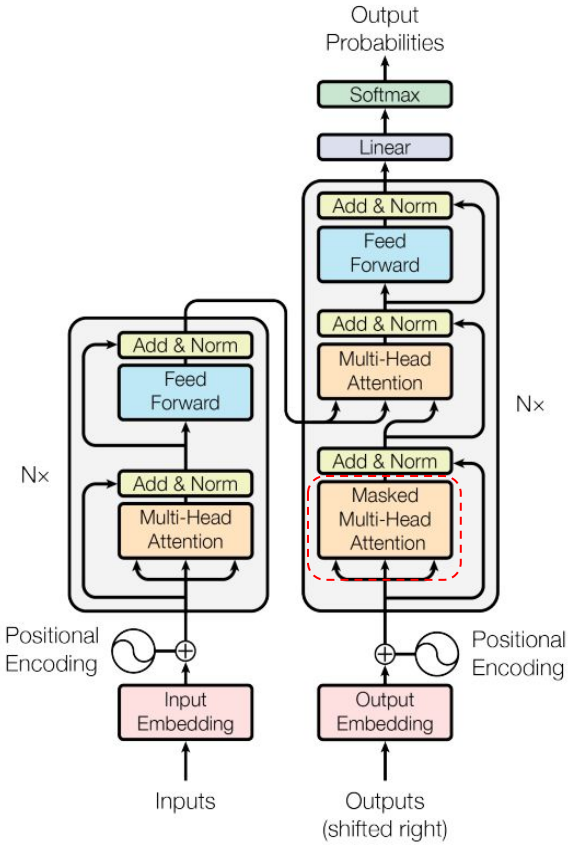


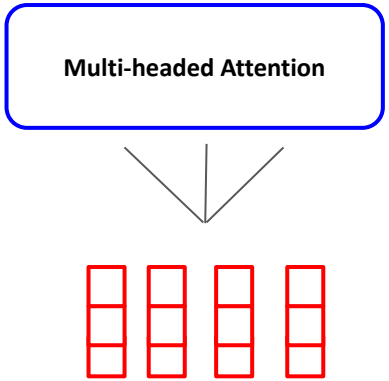


7. Decoder Multi-Head Attention 1

- Positional embeddings get fed into the first multi-headed attention layer
- Assign attention scores for the decoder's input

Decoder

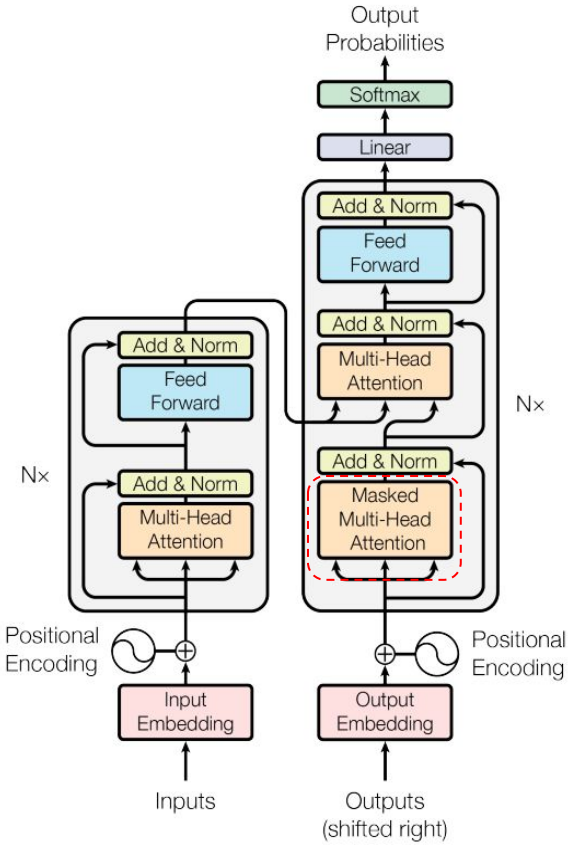


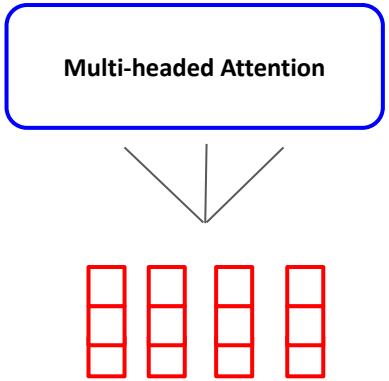


7. Decoder Multi-Head Attention 1

- Positional embeddings get fed into the first multi-headed attention layer
- Assign attention scores for the decoder's input
- This multi-headed attention layer operates slightly different

Decoder

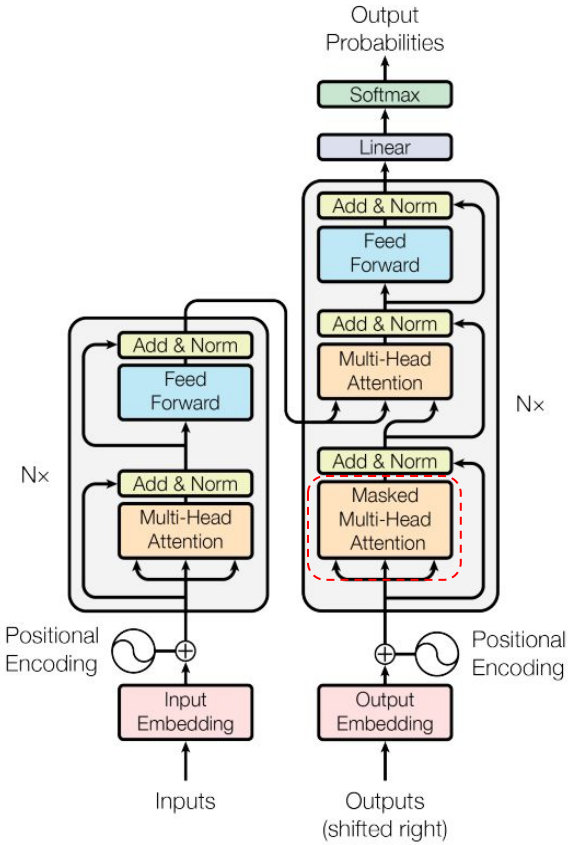


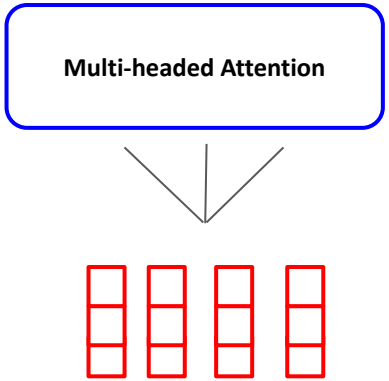


7. Decoder Multi-Head Attention 1

- Positional embeddings get fed into the first multi-headed attention layer
- Assign attention scores for the decoder's input
- This multi-headed attention layer operates slightly different
- The decoder is autoregressive

Decoder



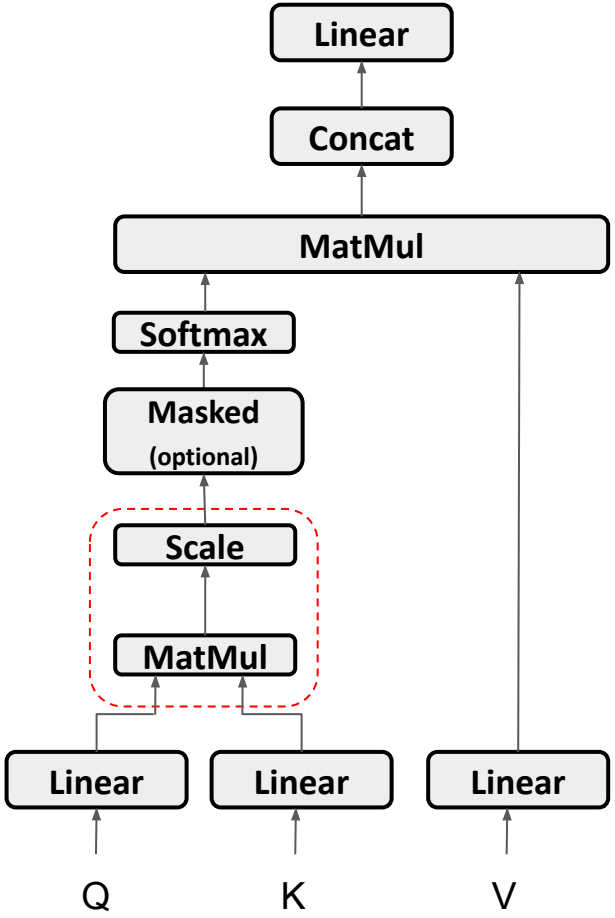


7. Decoder Multi-Head Attention 1

- Positional embeddings get fed into the first multi-headed attention layer
- Assign attention scores for the decoder's input
- This multi-headed attention layer operates slightly different
- The decoder is autoregressive
- No conditioning on future tokens

Decoder

7. Decoder Multi-Head Attention 1

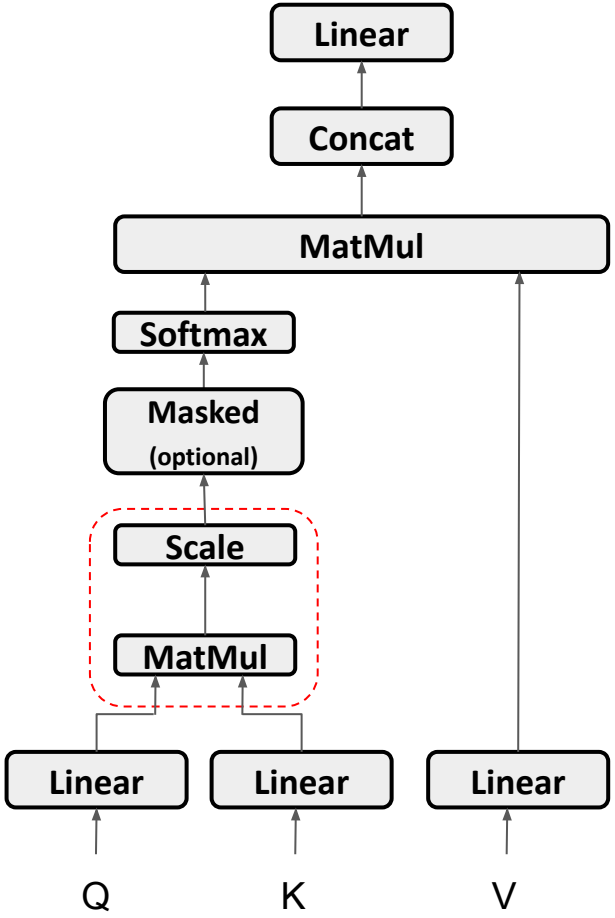


Attention weights

	<Start>	I	am	fine
<Start>	0.7	0.1	0.1	0.1
I	0.2	0.5	0.1	0.2
am	0.1	0.2	0.6	0.1
fine	0.1	0.2	0.4	0.3

Decoder

7. Decoder Multi-Head Attention 1

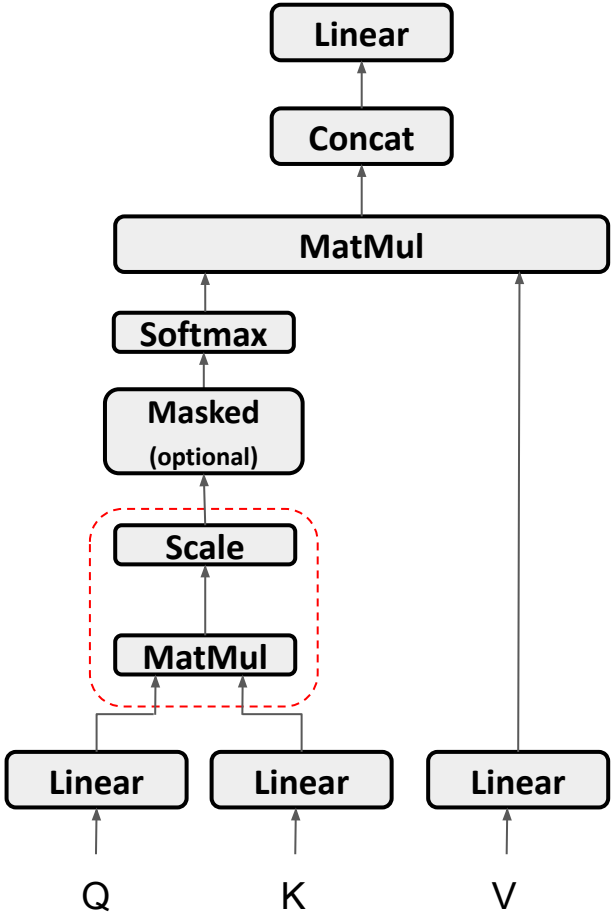


Attention weights

	<Start>	I	am	fine
<Start>	0.7	0.1	0.1	0.1
I	0.2	0.5	0.1	0.2
am	0.1	0.2	0.6	0.1
fine	0.1	0.2	0.4	0.3

Decoder

7. Decoder Multi-Head Attention 1

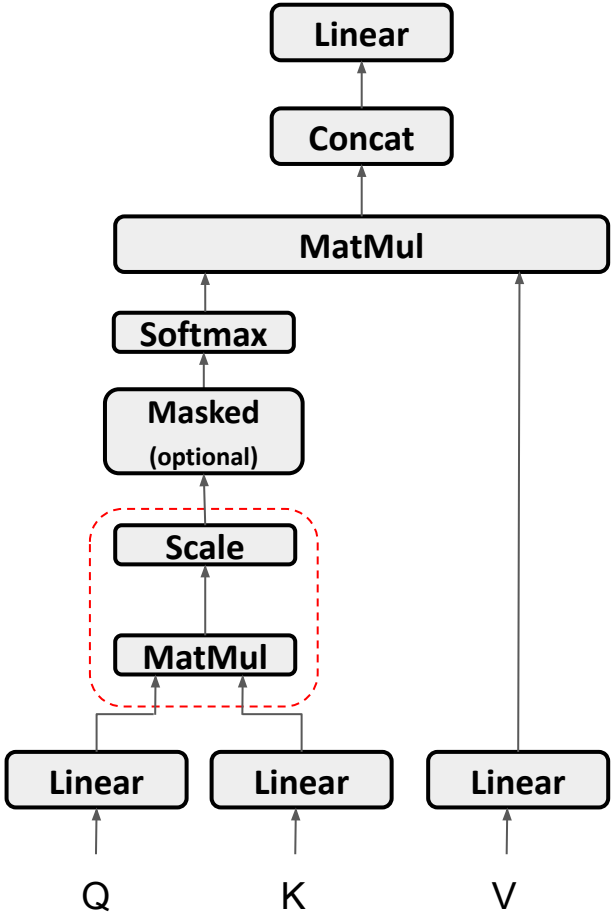


Attention weights

	<Start>	I	am	fine
<Start>	0.7	0.1	0.1	0.1
I	0.2	0.5	0.1	0.2
am	0.1	0.2	0.6	0.1
fine	0.1	0.2	0.4	0.3

Decoder

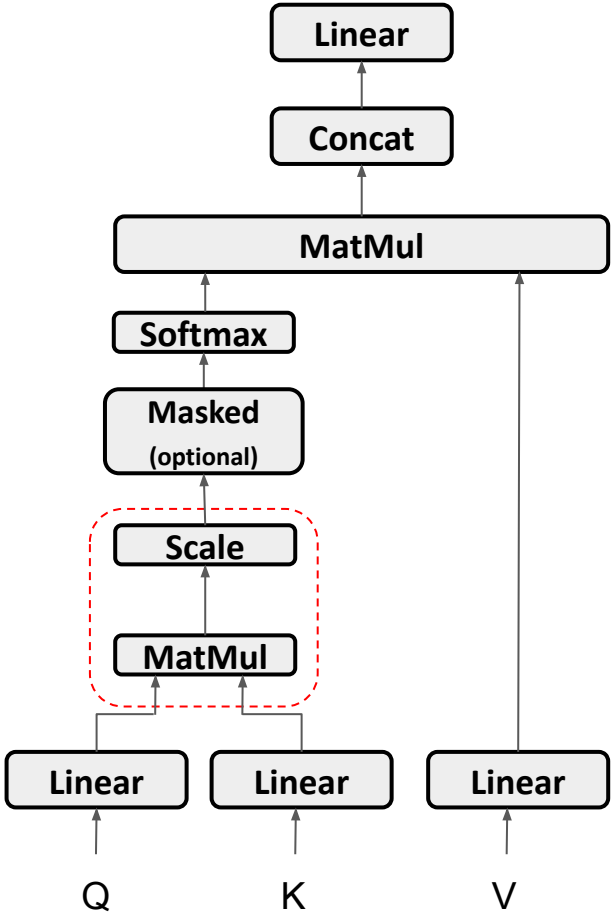
7. Decoder Multi-Head Attention 1

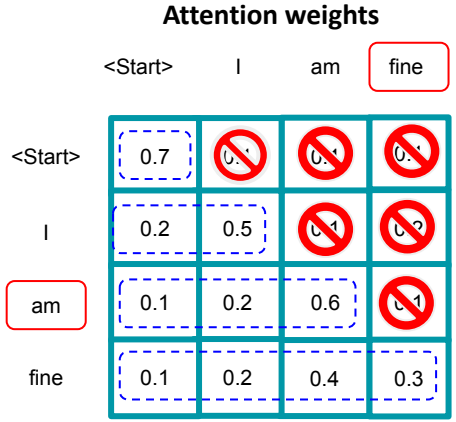


Attention weights

	<Start>	I	am	fine
<Start>	0.7	0.1	0.1	0.1
I	0.2	0.5	0.1	0.2
am	0.1	0.2	0.6	0.1
fine	0.1	0.2	0.4	0.3

Decoder



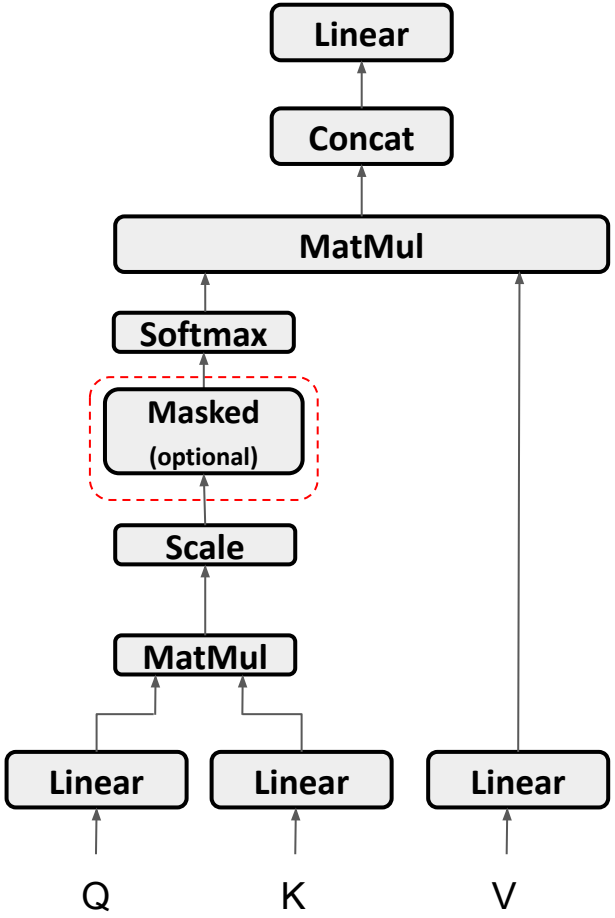


7. Decoder Multi-Head Attention 1

- How can we prevent the encoder from looking at future tokens ?

Decoder

7. Decoder Multi-Head Attention 1



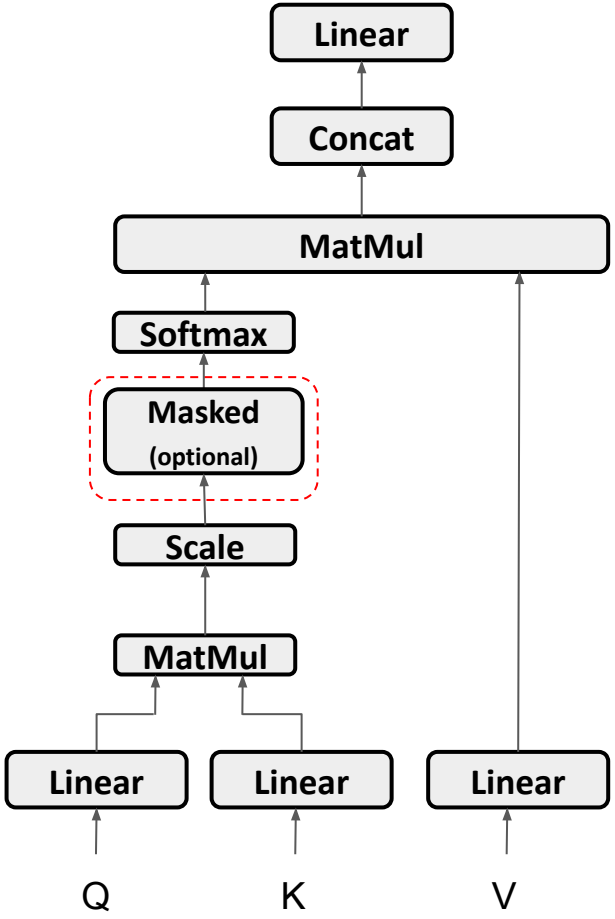
Attention weights

	<Start>	I	am	fine
<Start>	0.7	0.1	0.1	0.1
I	0.2	0.5	0.1	0.2
am	0.1	0.2	0.6	0.1
fine	0.1	0.2	0.4	0.3

- This method is called **masking**

Decoder

7. Decoder Multi-Head Attention 1
7.1 Look-Ahead mask



Scaled scores

0.7	0.1	0.1	0.1
0.2	0.5	0.1	0.2
0.1	0.2	0.6	0.1
0.1	0.2	0.4	0.3

+

Look-Ahead mask

0	-inf	-inf	-inf
0	0	-inf	-inf
0	0	0	-inf
0	0	0	0.3

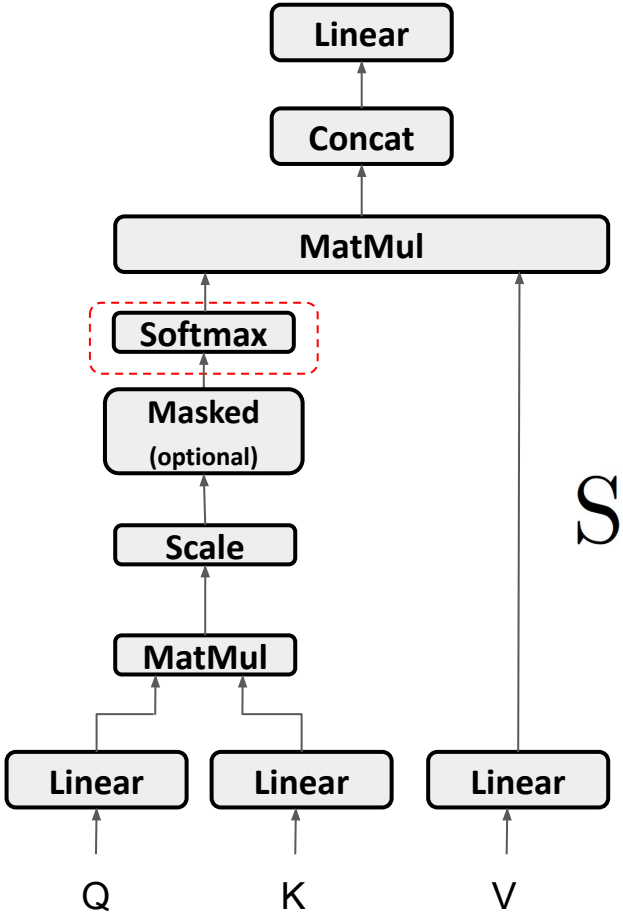
=

Masked Scores

0.7	-inf	-inf	-inf
0.2	0.5	-inf	-inf
0.1	0.2	0.6	-inf
0.1	0.2	0.4	0.3

Decoder

7. Decoder Multi-Head Attention 1
7.1 Look-Ahead mask



Masked Scores

$$\text{Softmax}\left(\begin{bmatrix} 0.7 & -inf & -inf & -inf \\ 0.2 & 0.5 & -inf & -inf \\ 0.1 & 0.2 & 0.6 & -inf \\ 0.1 & 0.2 & 0.4 & 0.3 \end{bmatrix}\right) =$$

	<Start>	I	am	fine
<Start>	1	0	0	0
I	0.37	0.63	0	0
am	0.26	0.31	0.43	0
fine	0.21	0.26	0.26	0.26