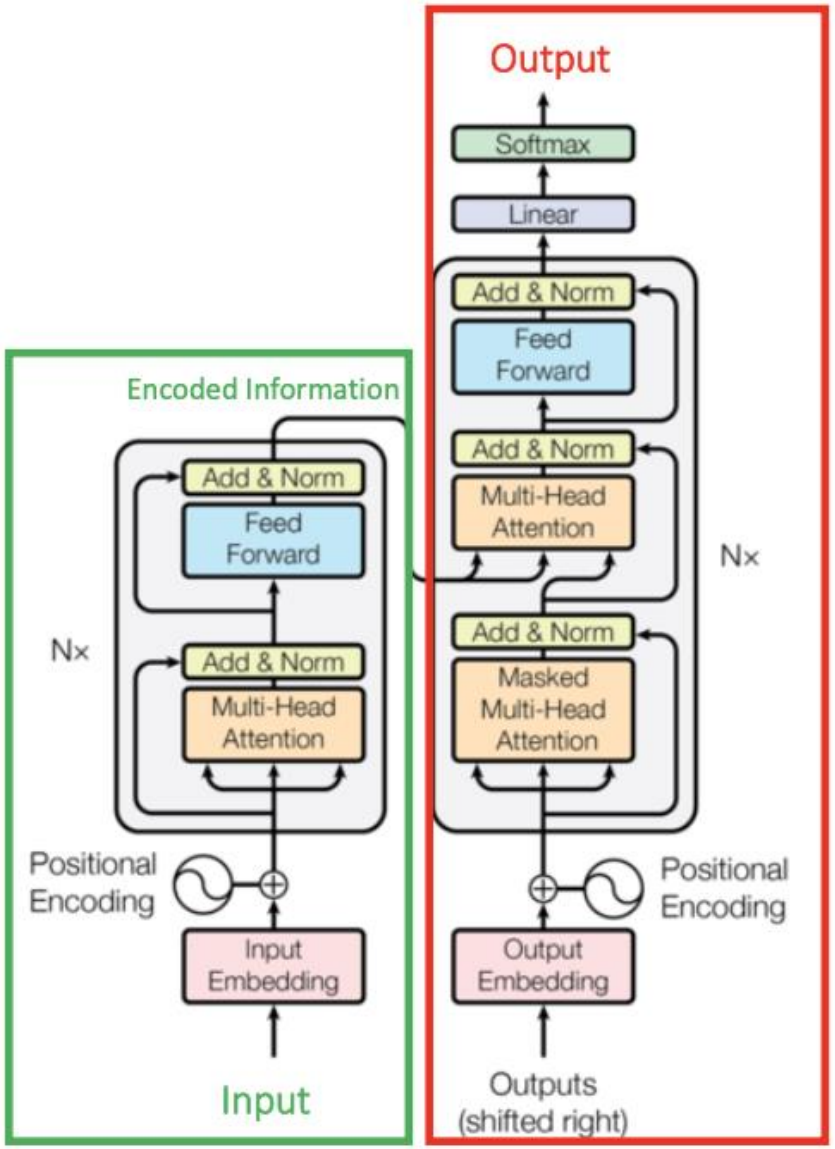
Transformer

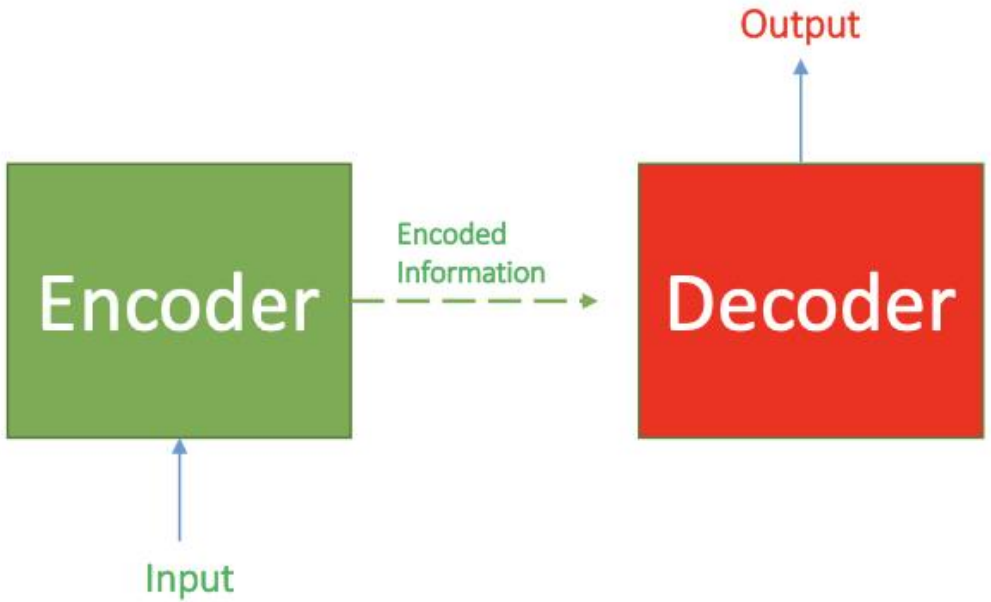
Yuxuan Du

EE 5993 AI Practicum

03/19/2025

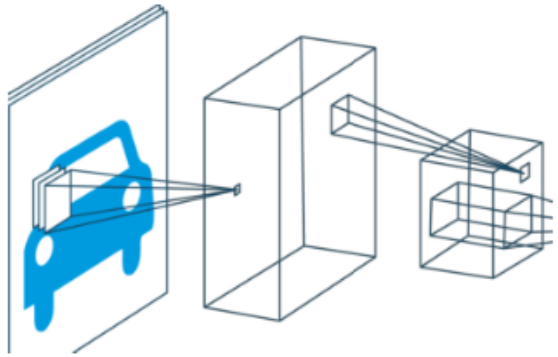
Transformers

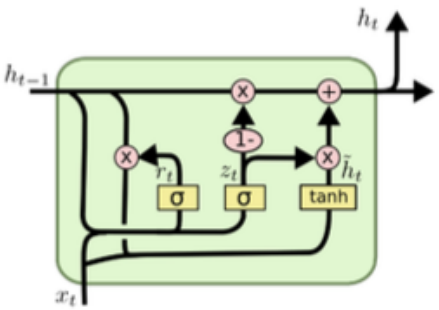




Learning/Inductive Bias

A set of assumptions made by the algorithm in order to generalize a type of training data (such as images or text)





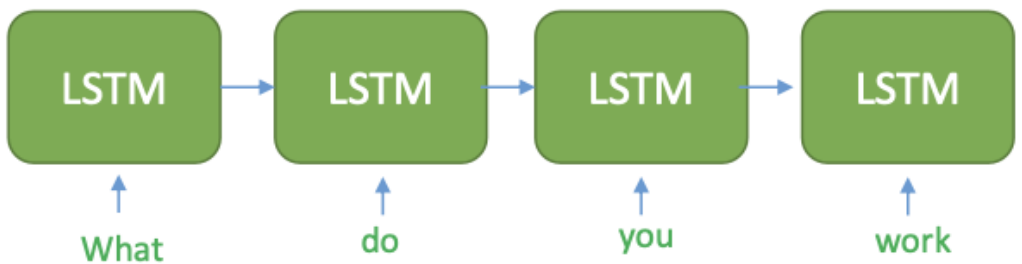
- CNNs have an inductive bias for images and LSTMs have an inductive bias for sequences, this is because of the way they are designed to deal with images or text. Their mechanism encourages them to prioritize solutions with specific properties.
- In other words, they are not general (while MLPs are)
- If we could have an architecture based fully on MLPs (like Transformers), then we can generalize much better
- For example, training a CNN stops improving after 100 epochs while training a transformers continues to improve after 100 epochs

Three keypoints of the Transformer

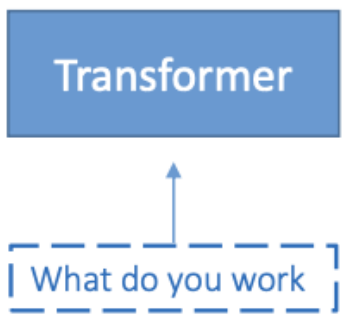
- *Not* sequential like RNNs, all the input (ex. sentence) is fed once through the model and calculation is performed one time.
- Attention is generated from the model's own input (self-attention)
- More than one attention is generated each time (multi-head attention)

1.Not Sequential

RNNs

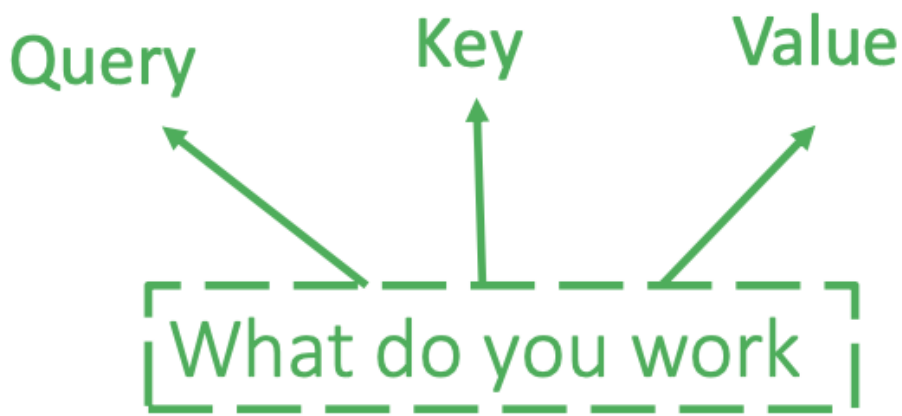


Transformers



2.Self-attention

- The attention is generated from the model's own input → Query, Key and Value is generated from the model's input

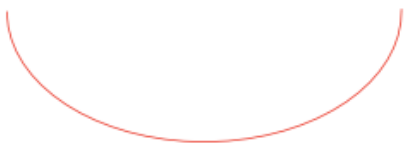


How is self-attention useful?

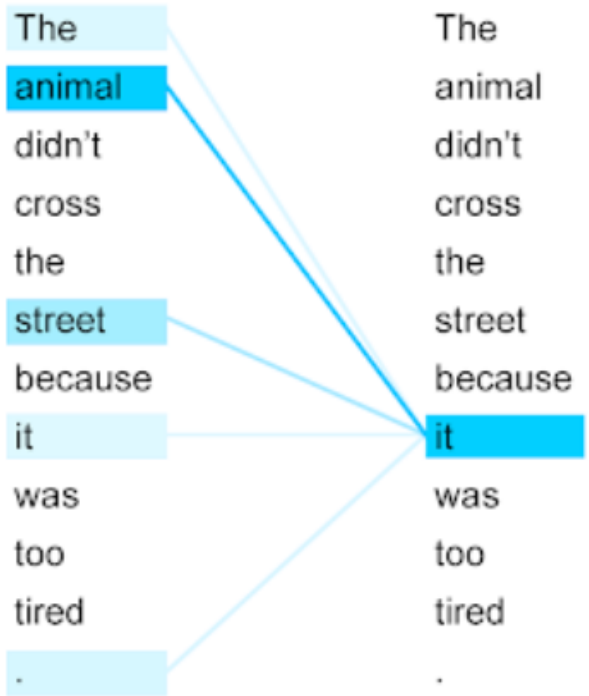
- Consider two sentences:

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

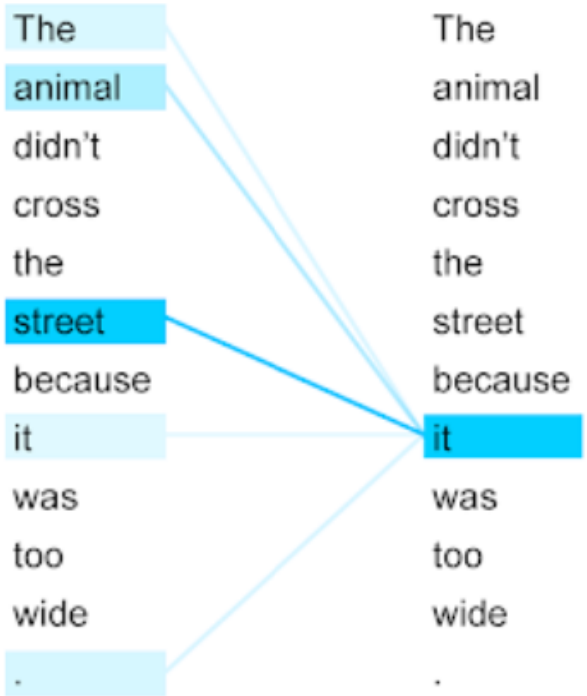
The animal didn't cross the street because it was too wide



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



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

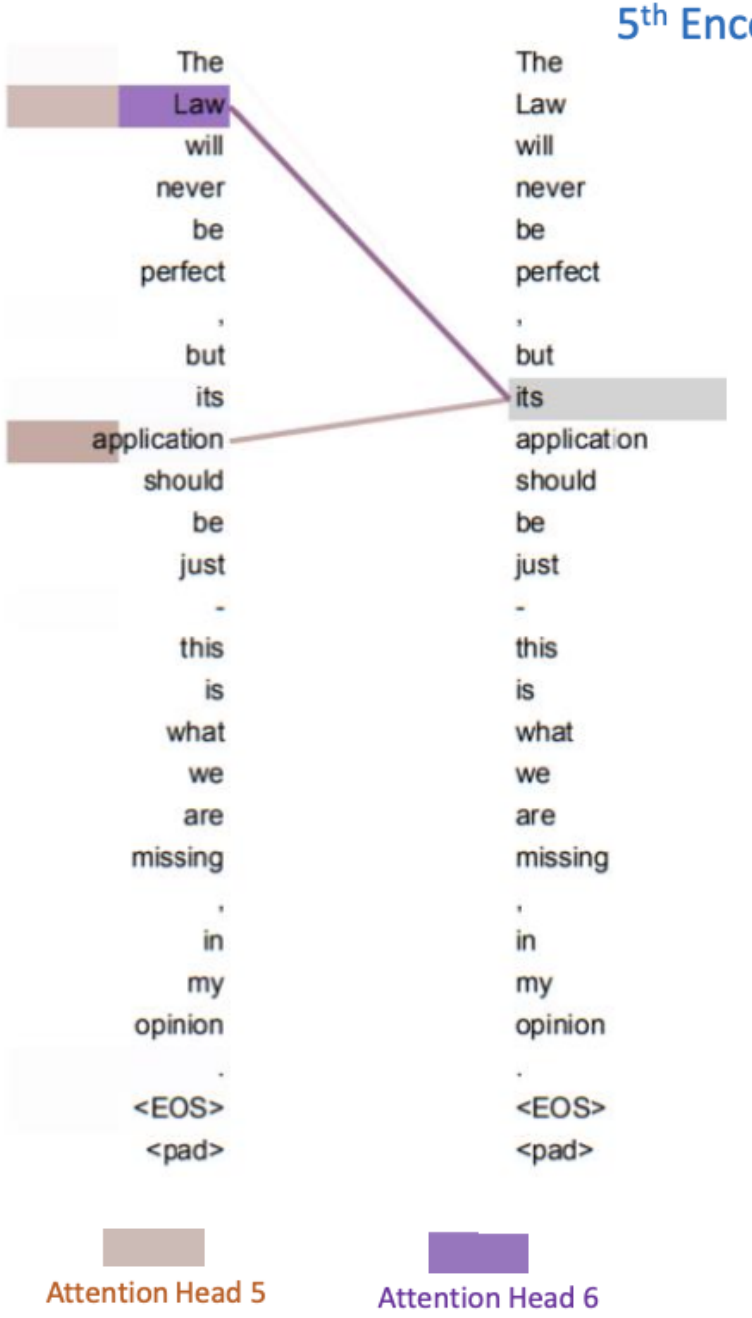


https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

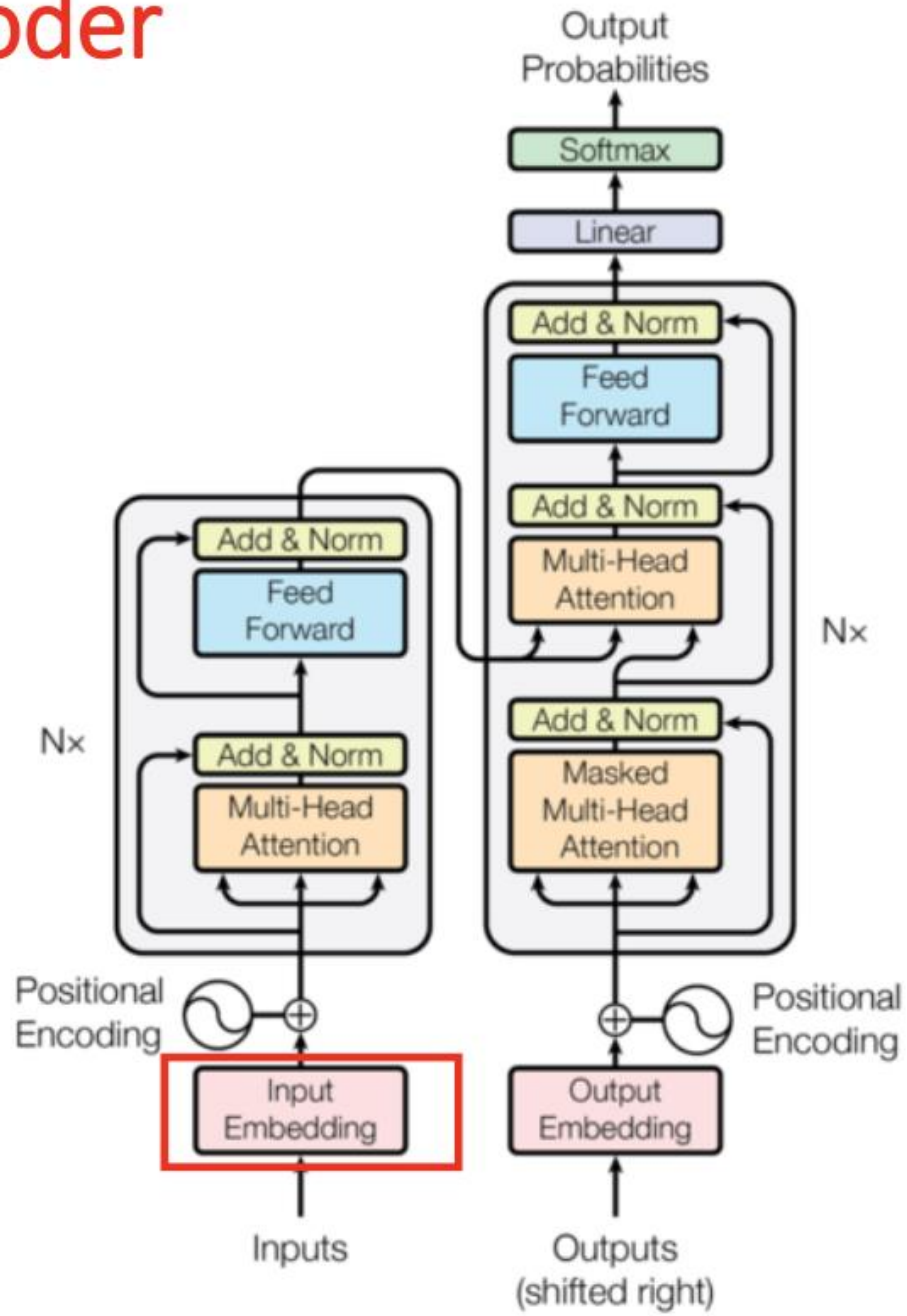
5th Layer of the Encoder for one out of 8 attention heads

3.Multi-Head attention

- This gives the model the advantage of focusing on different words *h ways* (*h* is the number of heads). It broadens the model's capability to focus on different positions and gives the attention layer multiple different representations (we'll have *h* different Queries, Keys and Values).



Let's start: Encoder



Sentences of Variable Length and Masking

- Since sentences are of variable length, in order to load the sentences in batches through our model, we need to set a maximum length and pad the sentences that are shorter than the maximum length.
- We also don't want the model to attend over the padded words, so we need to set a mask for each sentence.

Data is a matrix of shape: `(batch_size, max_len)` → `(4,9)`

Hi	my	name	is	John	nice	to	meet	you
Artificial	Intelligence	is	ruled	by	the	chain	rule	0
How	will	you	go	to	campus	0	0	0
Lions	live	in	grasslands	0	0	0	0	0

Masks is a matrix of shape: `(batch_size, max_len)` → `(4,9)`

```
[ [ 1 1 1 1 1 1 1 1 1 ],
  [ 1 1 1 1 1 1 1 1 0 ],
  [ 1 1 1 1 1 1 0 0 0 ],
  [ 1 1 1 1 0 0 0 0 0 ]]
```

Embedding all the words in our sentence

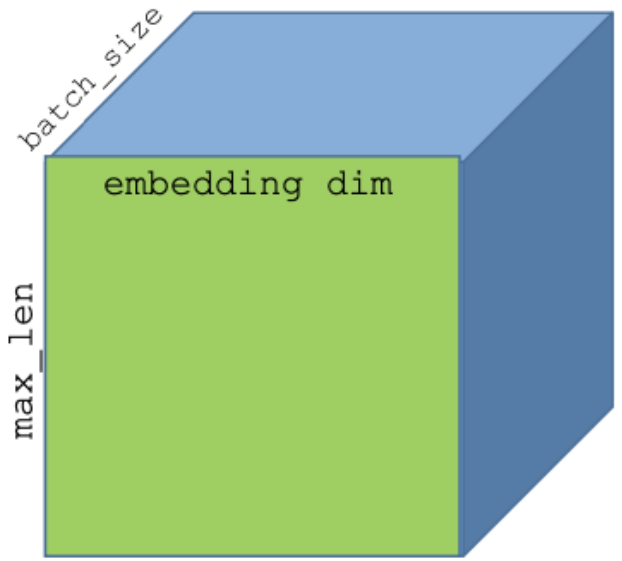
- We first use an embedding layer to embed all the words in our sentence. The dimensionality of the word embedding will be the dimensionality of our model. Let's take the word embedding dimension as 8 for this example.

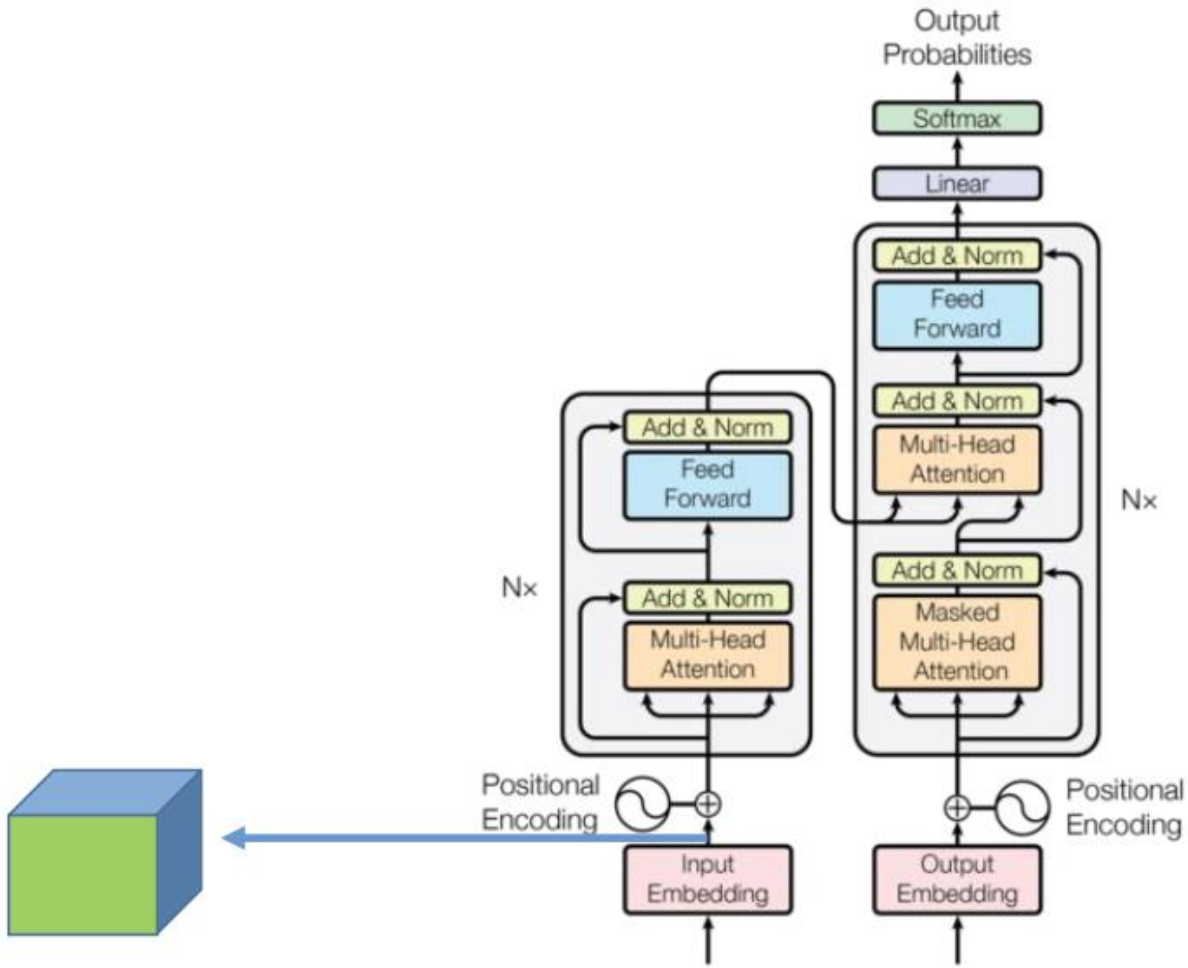
Embeddings

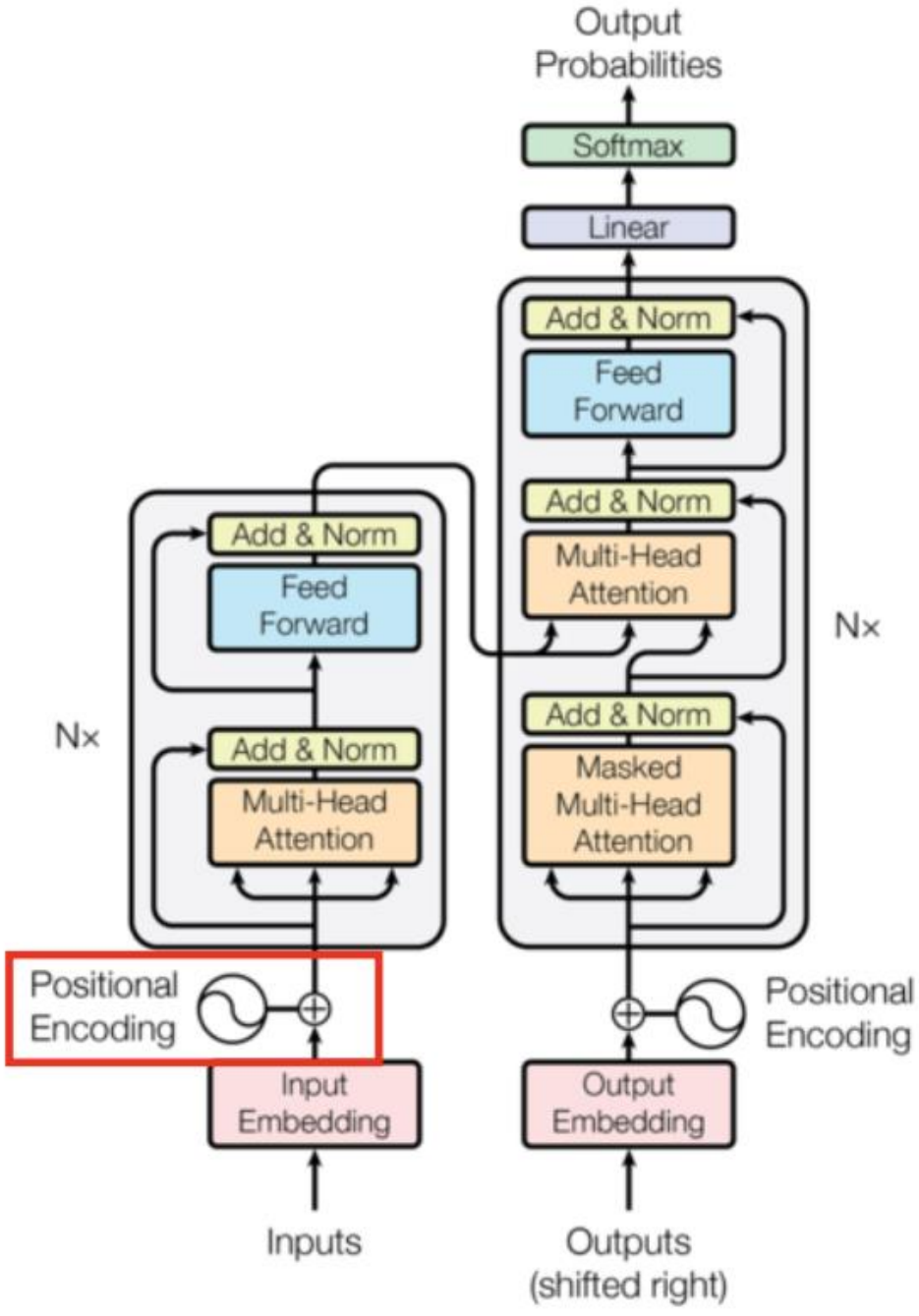
Hi								
my								
name								
is								
john								
nice								
to								
meet								
you								

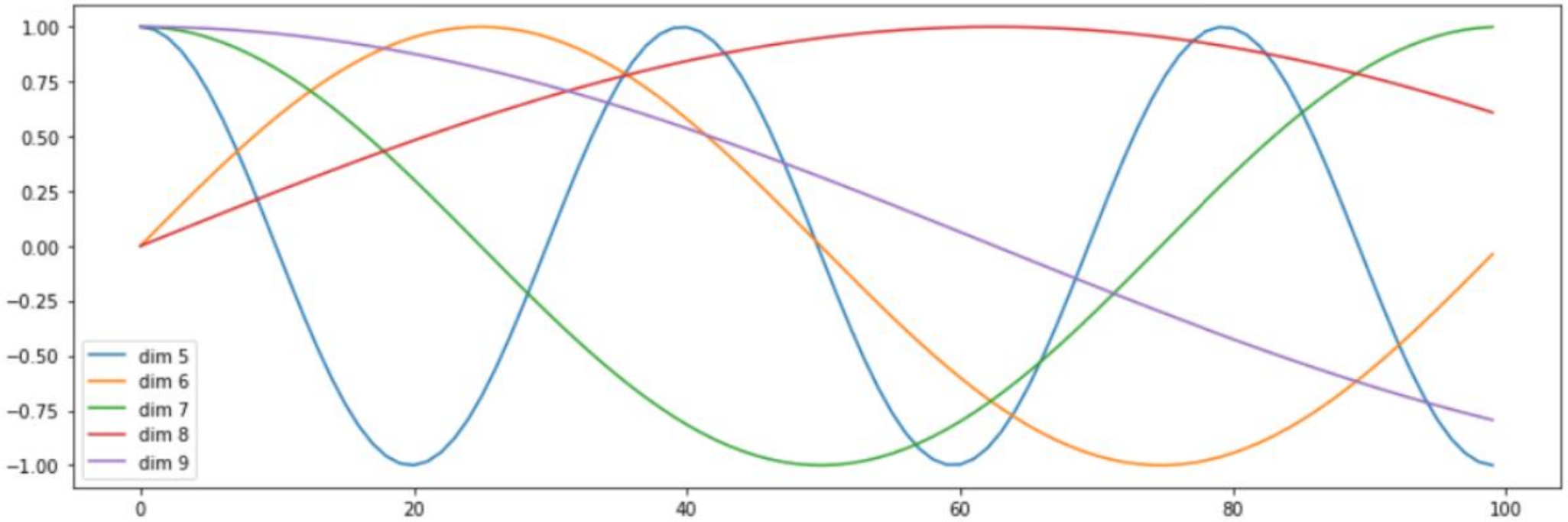
Dataset is now of shape:

`(batch_size, max_len, embedding dim)`

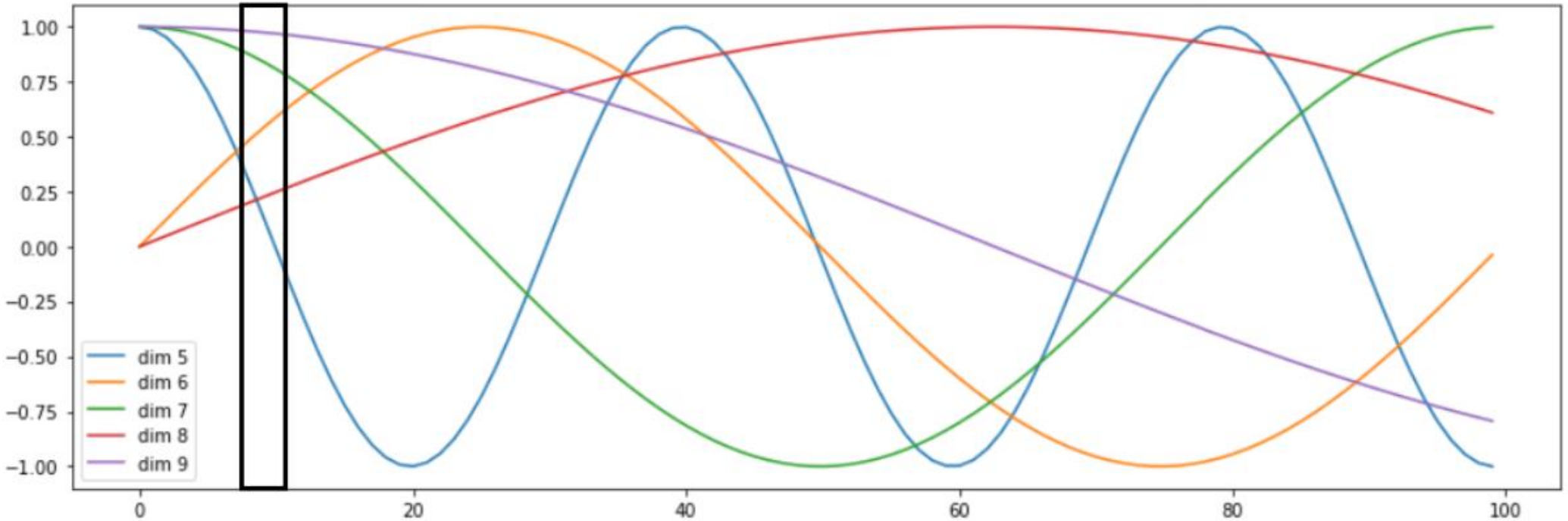



Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding

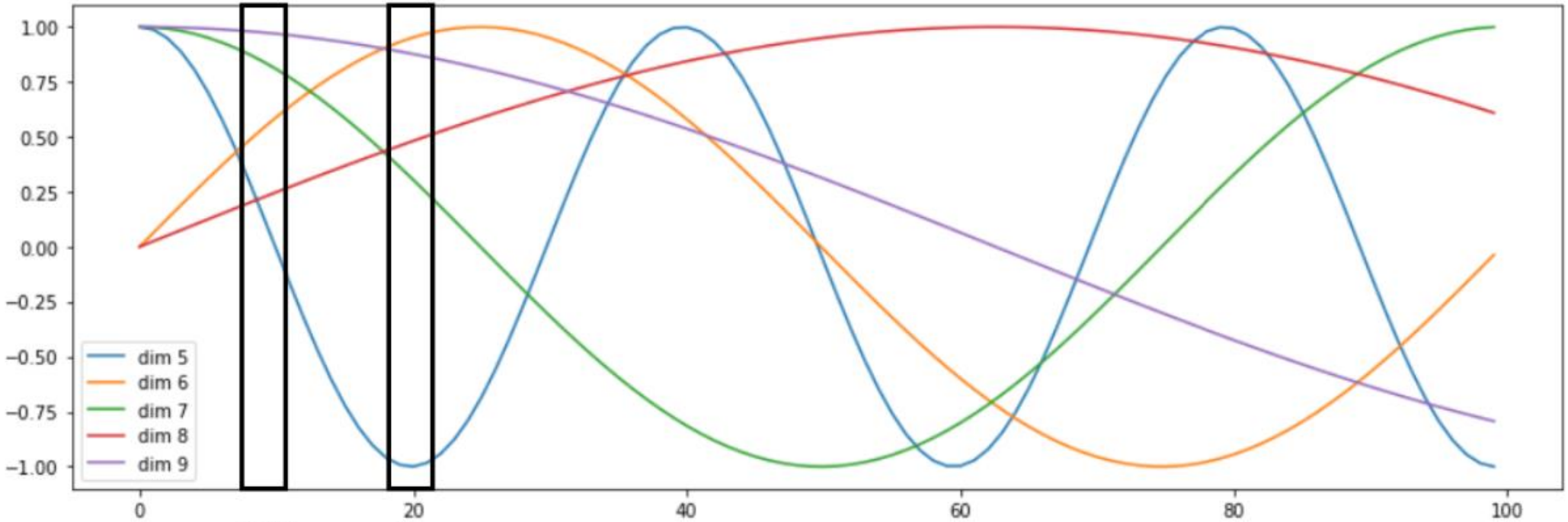

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20
40
60
80
100
dim 5
dim 6
dim 7
dim 8
dim 9

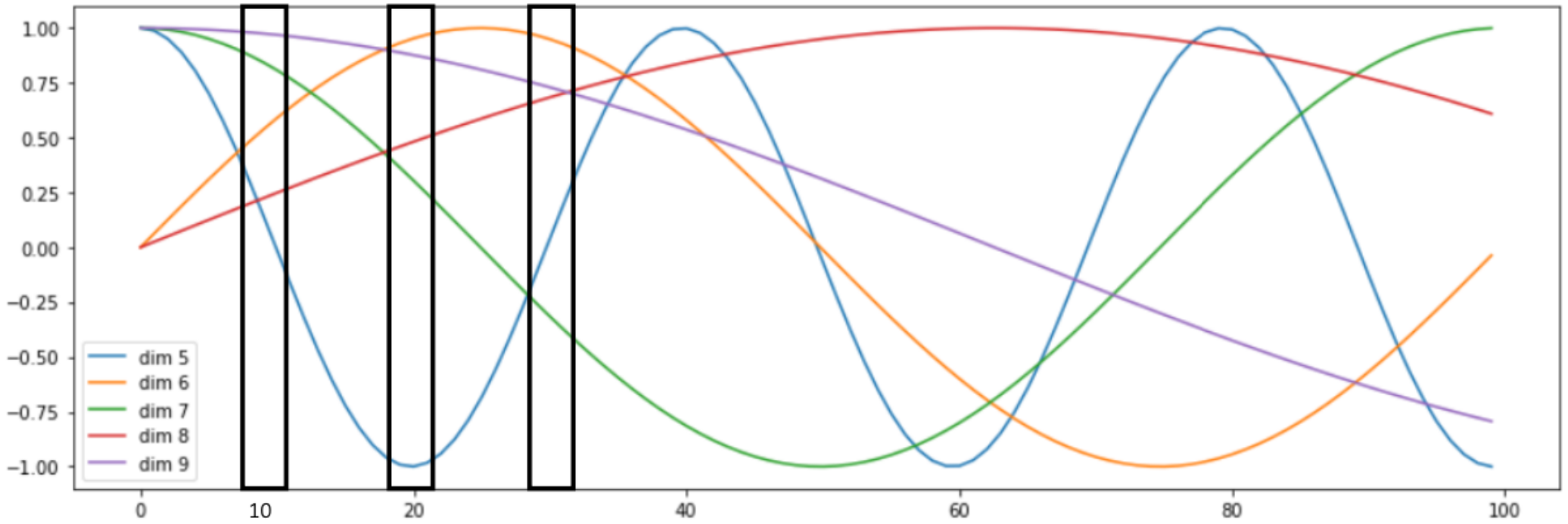
5-dimension



0.99
0.8
0.5
0.22
0.22



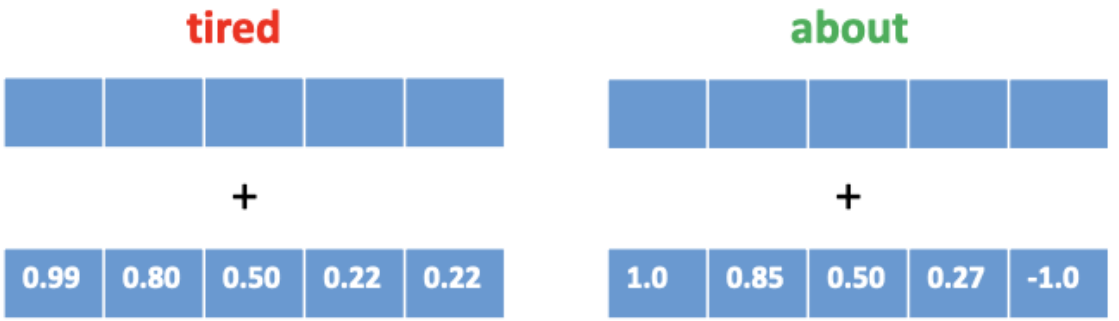
0.99	1
0.8	0.85
0.5	0.5
0.22	0.27
0.22	-1

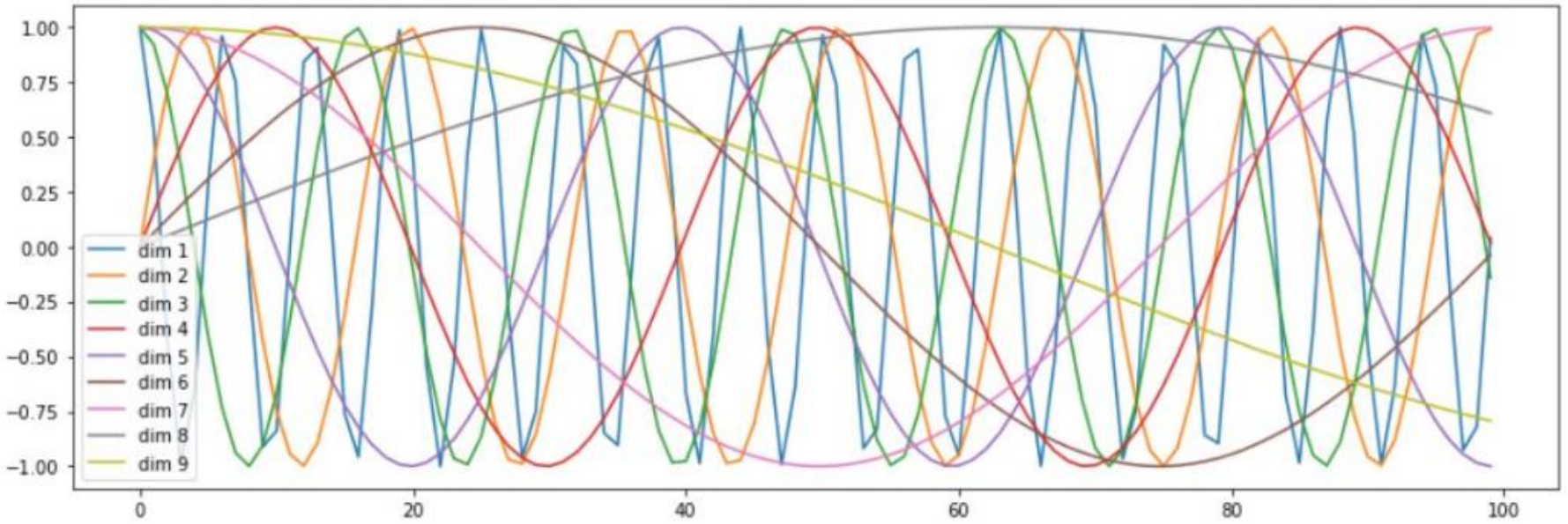


0.99	1	0.9
0.8	0.85	0.73
0.5	0.5	0.7
0.22	0.27	0
0.22	-1	-0.30

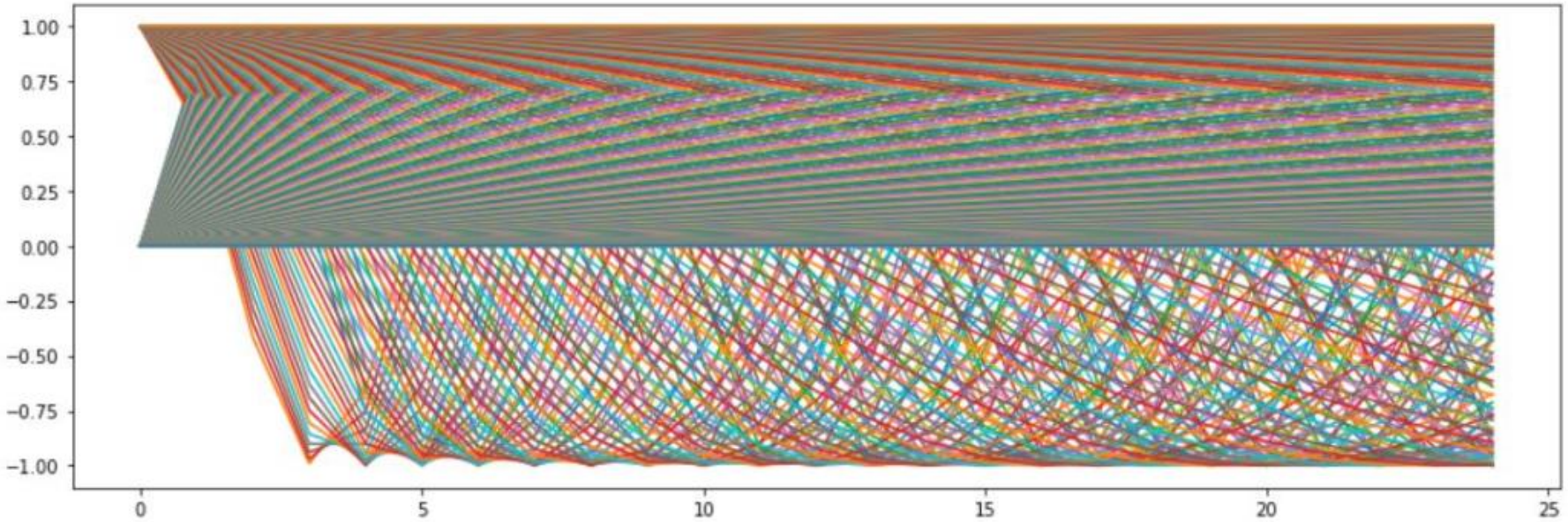
Each of these encode the position of the word. Now we add these encoding to our word embedding

Sentence: When I came home last night I was very tired from the football match so I instantly slept. How about you?





9-dimension



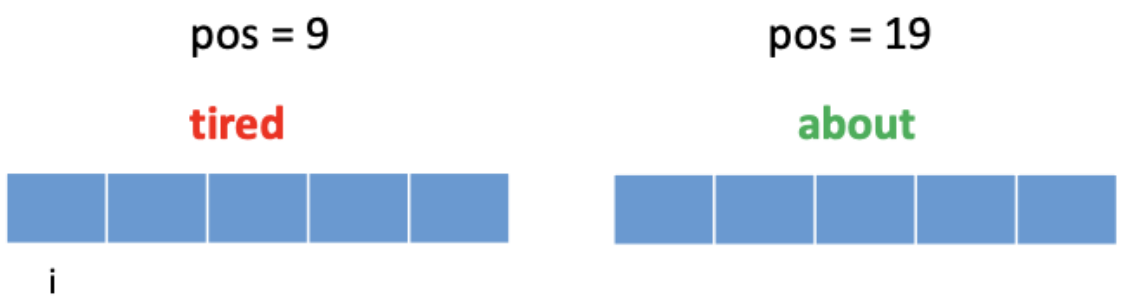
512-dimension

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

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

pos – the position of the word in the sentence
i – the dimension in each word

Sentence: When I came home last night I was very tired from the football match so I instantly slept. How about you?



Assume the dimension of the word is 10

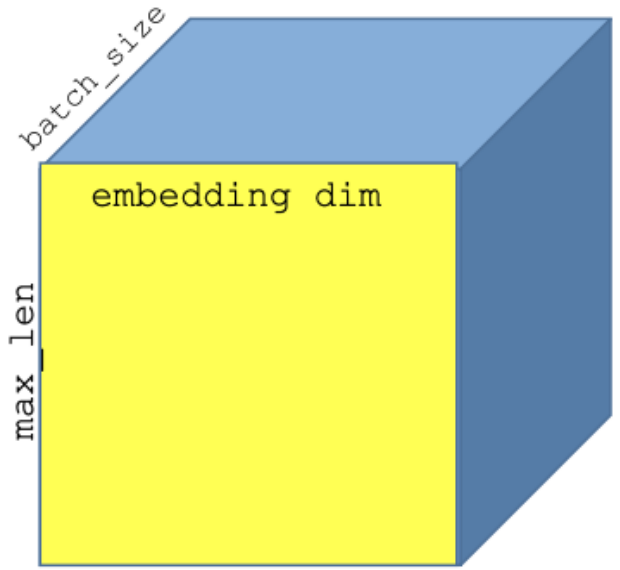
2i	0	2	4	6	8
2i+1	1	3	5	7	9

sin	0	2	4	6	8
cos	1	3	5	7	9

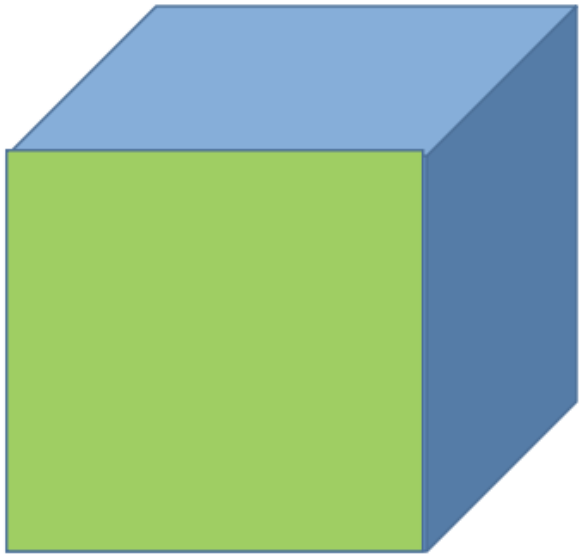
0	1	2	3	4	5	6	7	8	9
sin	cos	sin	cos	sin	cos	sin	cos	sin	cos

Create the Positional Encoding Matrix

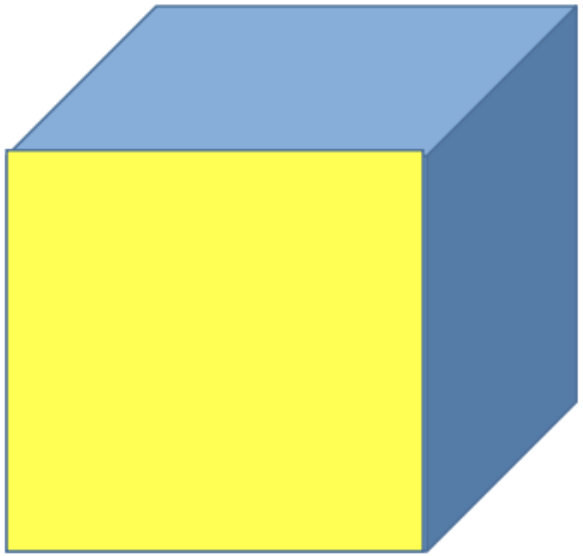
Note that the dimensionality of the positional encoding NEED to be the same with the dimensionality for the word embeddings, since we'll add them together.

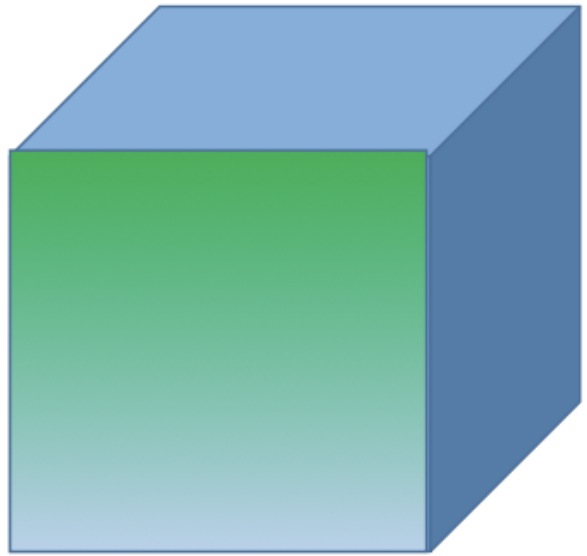


Embeddings

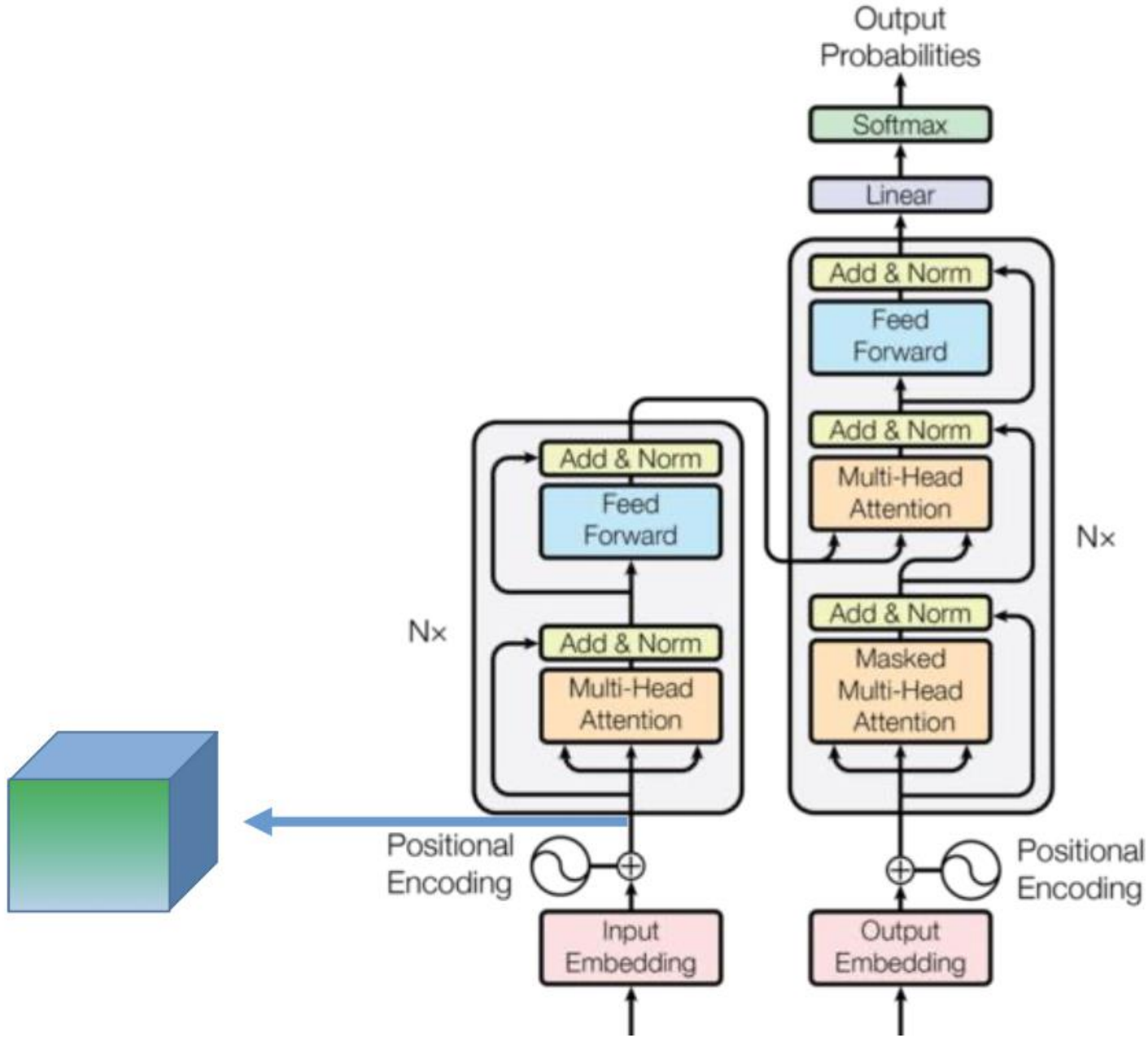


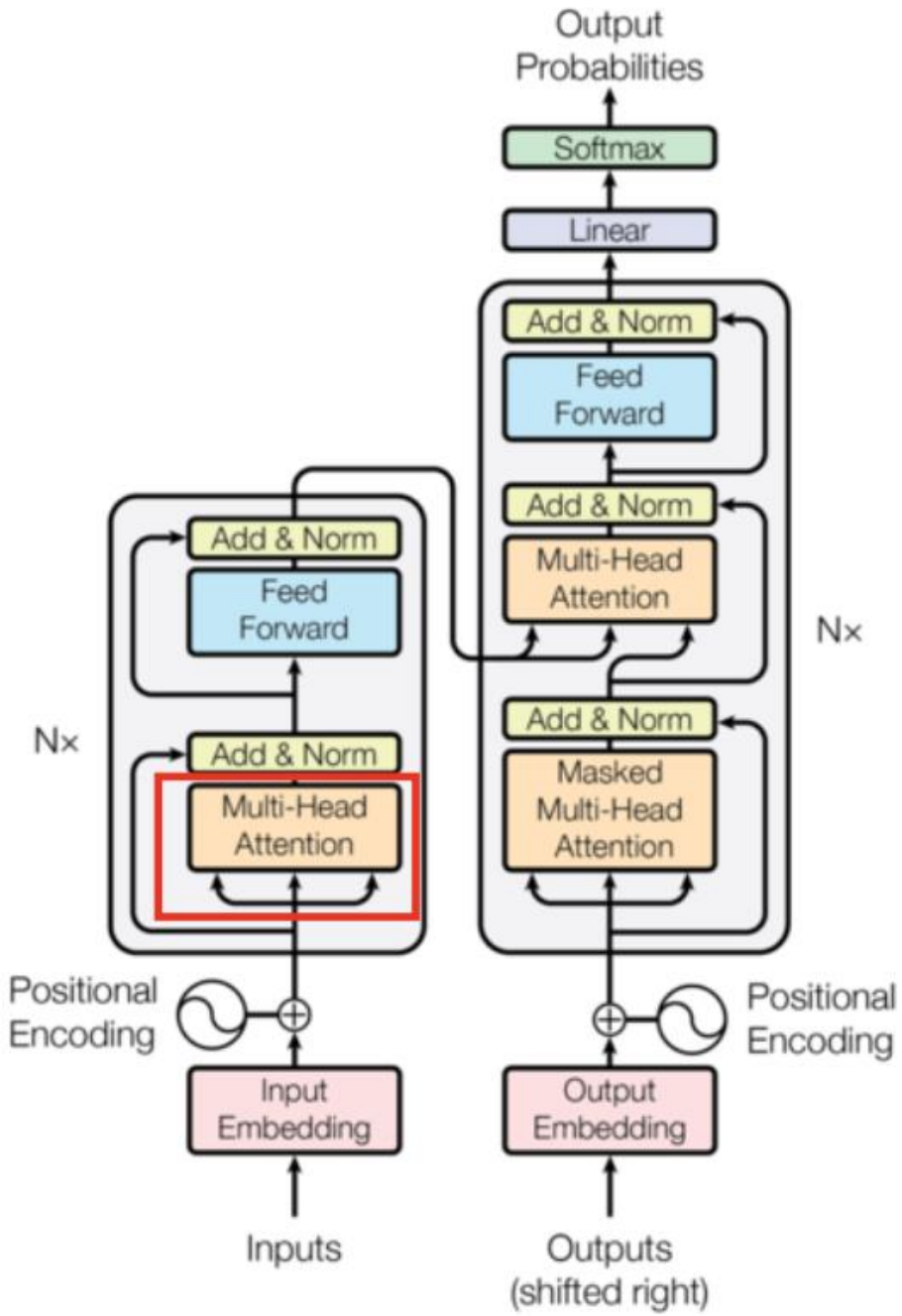
Positional Encoding





Realize that we gave more weights to the **embeddings**. We need to make sure that we don't loose the meaning when we add the positional encoding


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)