NLP -Section 9

Self-Attention + Transformers

Self-Attention

- Same idea as attention but instead of reflecting the attention on the output, it reflects on itself.
- Creates context embeddings
- Lacks positionality

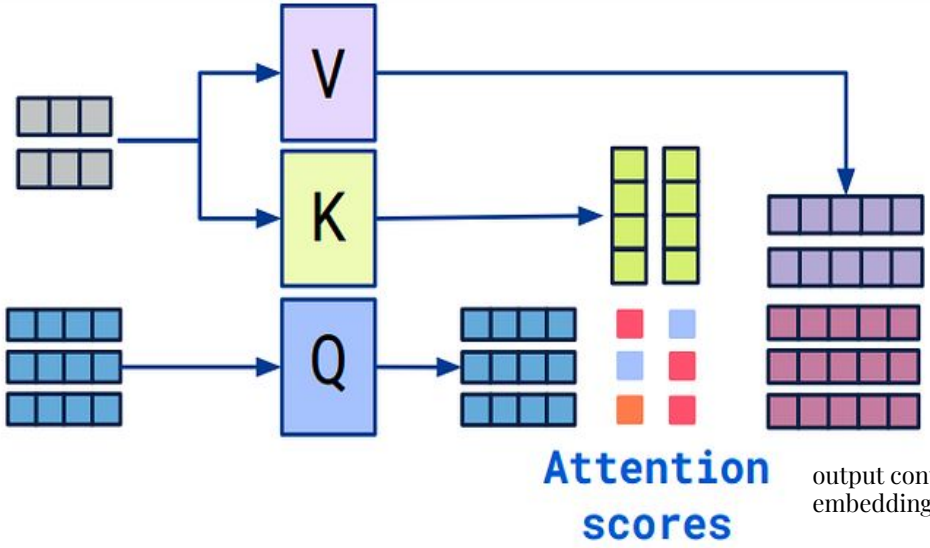
Self Attention - Matrices

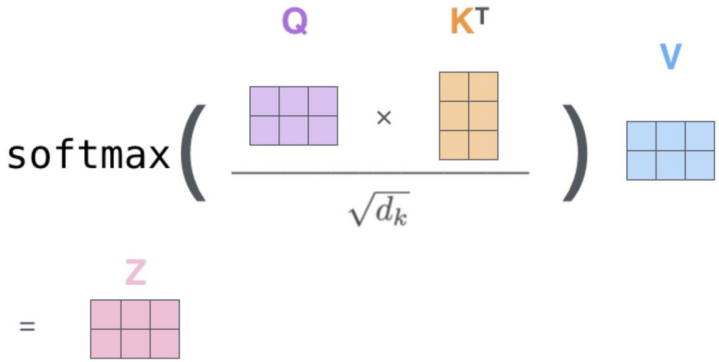
- We create Q, K, V for each token.
- Q (query): A learned projection of the input embedding used to compute attention scores toward all other tokens.
- K (key): A learned projection that interacts with queries so that the model can decide how much attention a token should receive.
- V (value): A learned projection that carries the content to be aggregated based on the attention weights.
- Q, K, and V have # of columns equal to the number of tokens each column representing a token (by small letter and index ex: q1, v3).
- Calculated through Wq*X, Wk*X, Wv*X

Self-Attention - Calculation

- The most straightforward method:

1. Calculate the score by q.k (the dot product) to get a single value
2. Weight the values around zero by dividing by root d then applying softmax
3. Multiply by v to get the output context representation z





The self-attention calculation in matrix form

Transformers Architecture - Positional Encoding

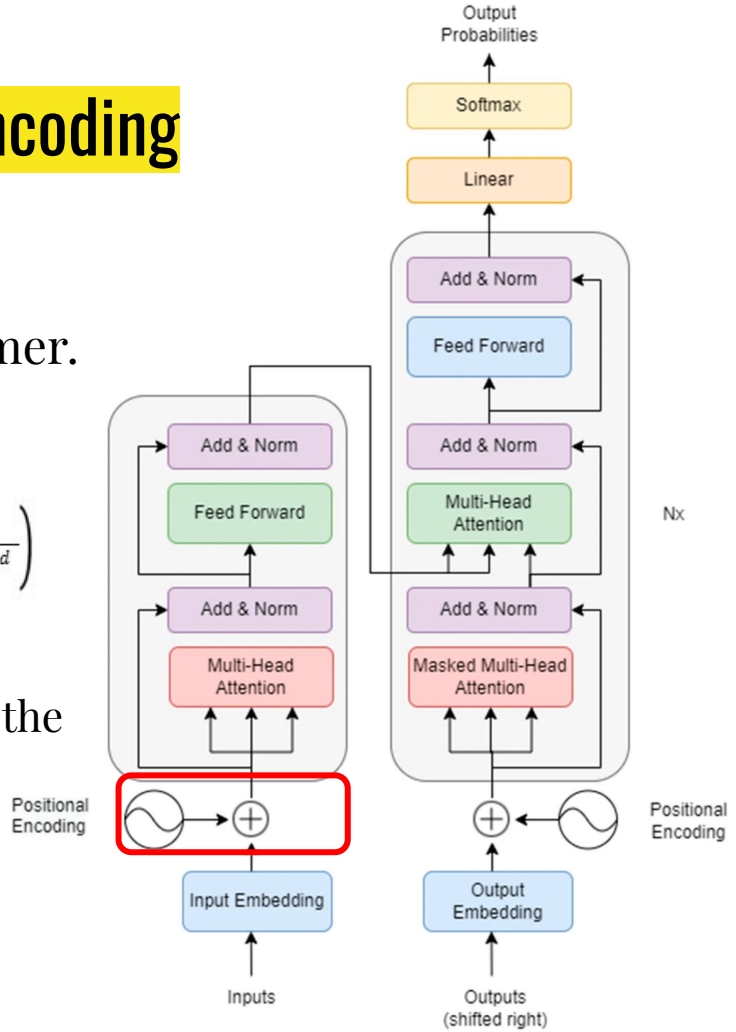
- Positional Encoding can be learned (new architectures) or sinusoidal (original transformer.
- Sinusoidal where:

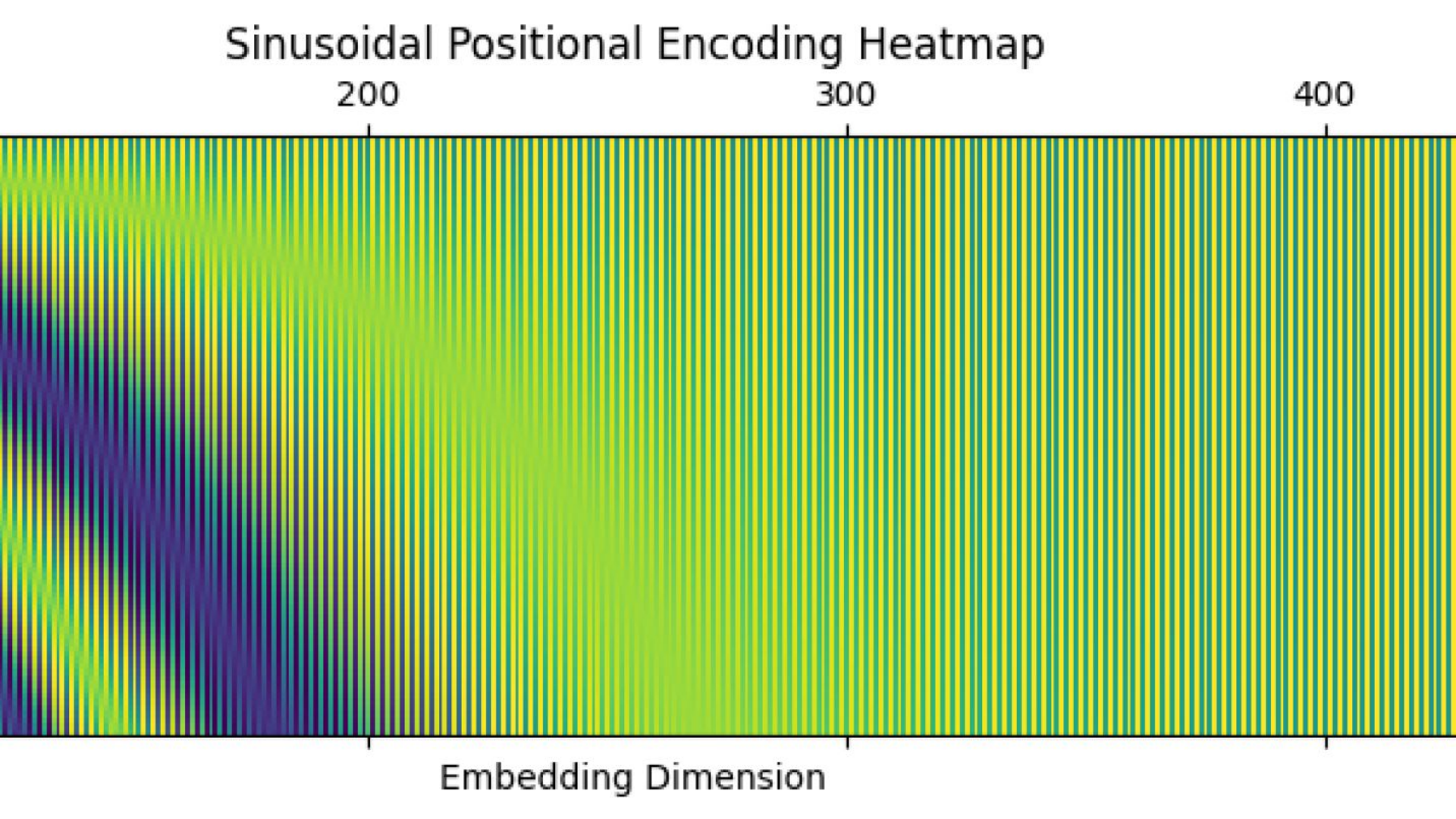
$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE(pos, 2i + 1) = cos\left(\frac{pos}{10000^{2i/d}}\right)$$

1. pos is the position
2. i is the embedding dimension
3. d is the dimension of the embedding dimension

sin calculates the even dimensions and cosine calculates the odd dimensions.




Sinusoidal Positional Encoding Heatmap
200
300
400
Embedding Dimension

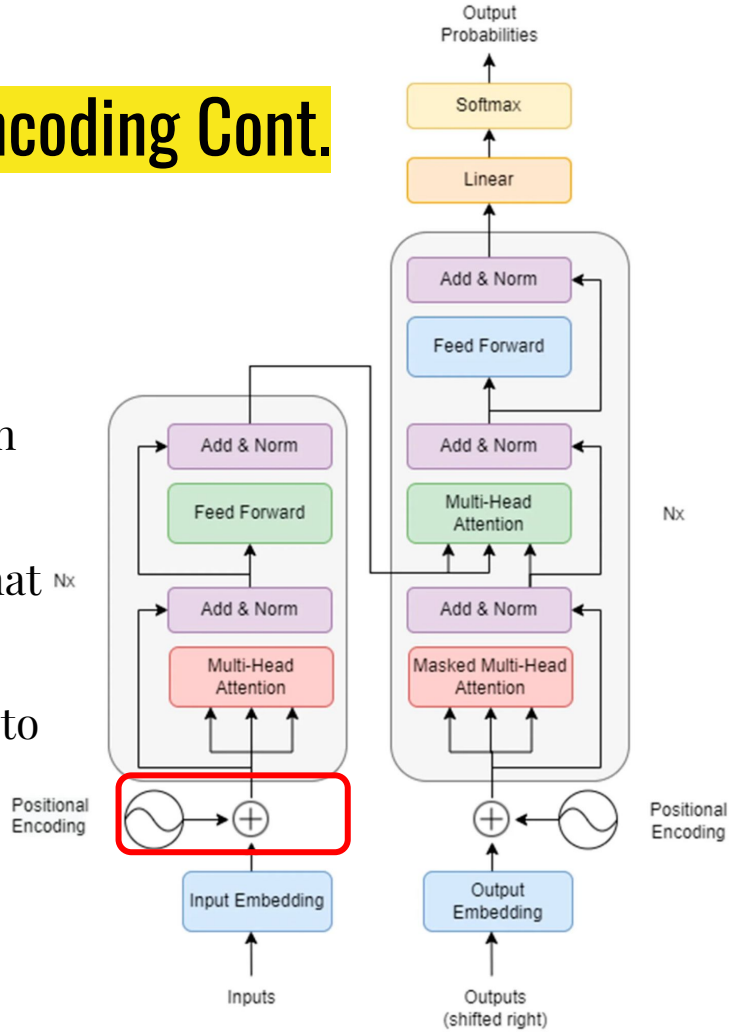
Transformers Architecture - Positional Encoding Cont.

- Learned:

❖ Each position gets a vector embedding.
This vector embedding is learned during training such as any other weighted matrix

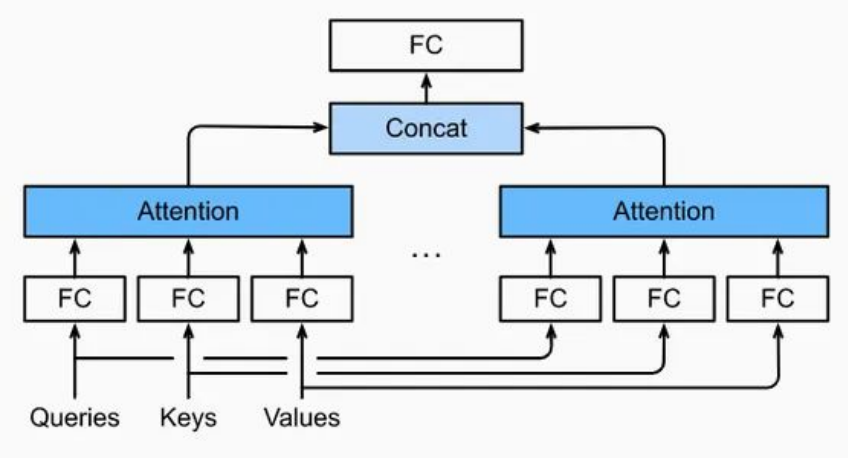
❖ pos 1 will have a constant embedding regardless of what is at the position.

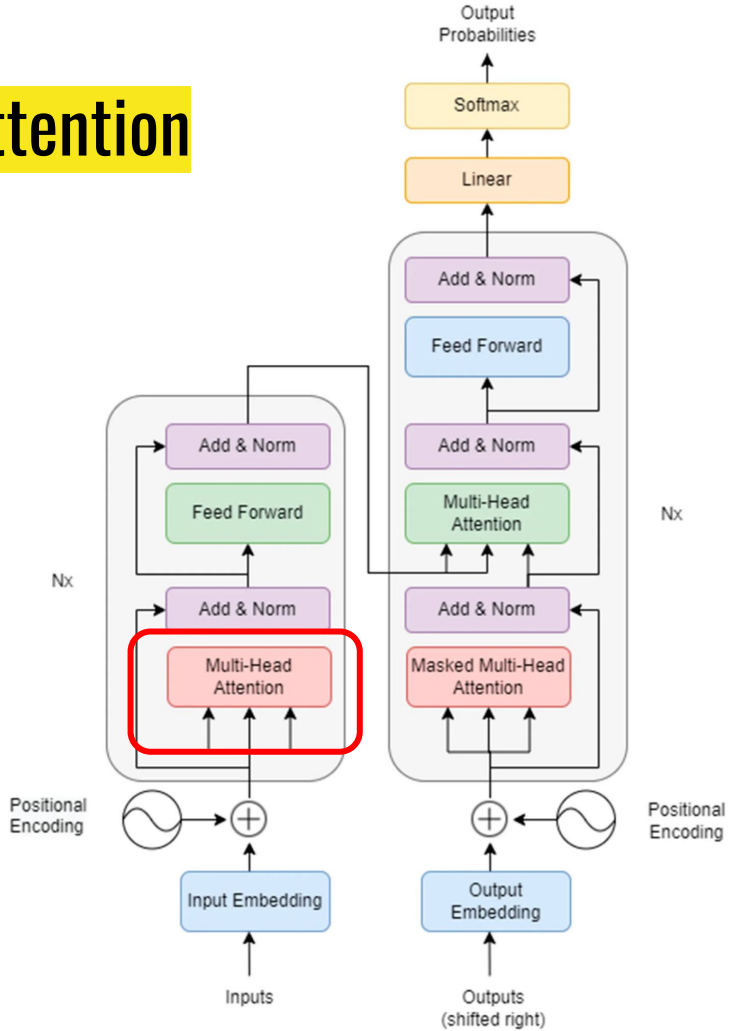
❖ This is simpler and makes the model not constrained to sinusoidal representation.



Transformers Architecture - Multihead Attention

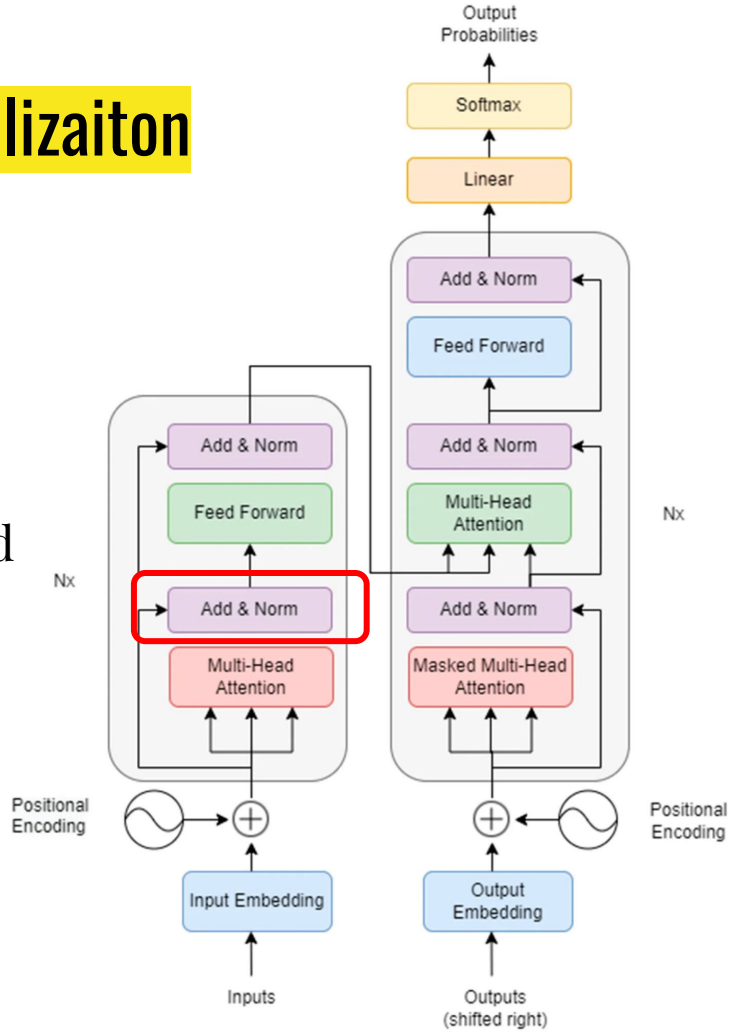
- Instead of relying on one attention head, calculate the attention multiple time and concatenate the results.
- Return to the original dimension by processing through the feed forward layer.





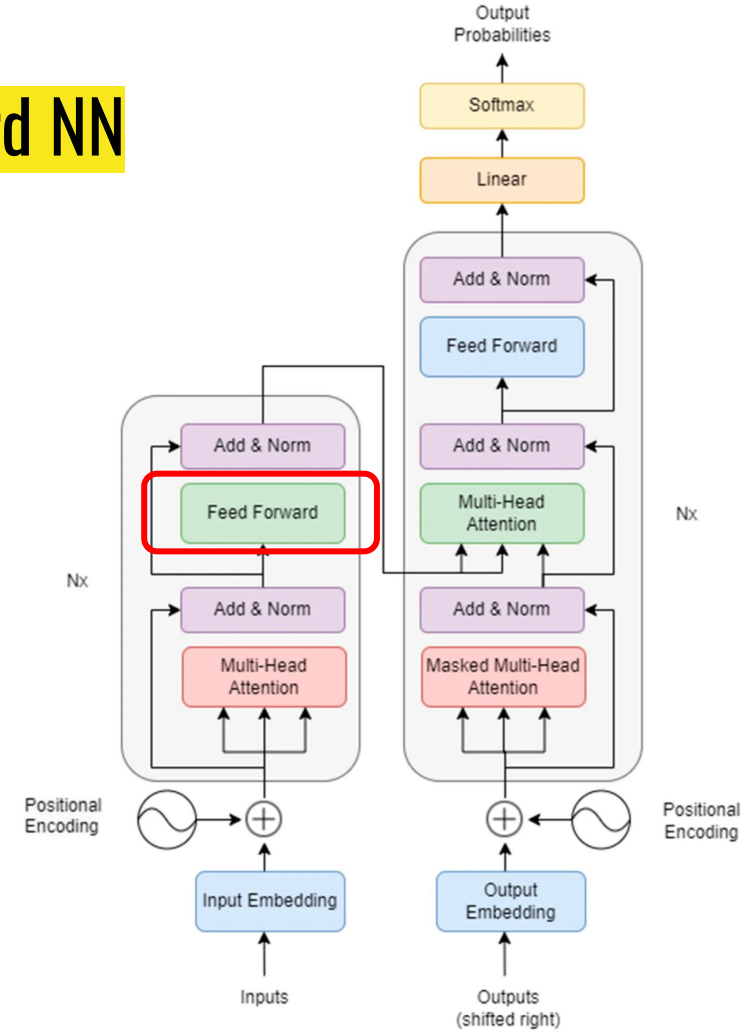
Transformers Architecture - Layer Normalizaiton

- $\frac{x - \mu(x)}{\sigma(x)}$
- Helps normalize the outputs of the multihead attention layer.



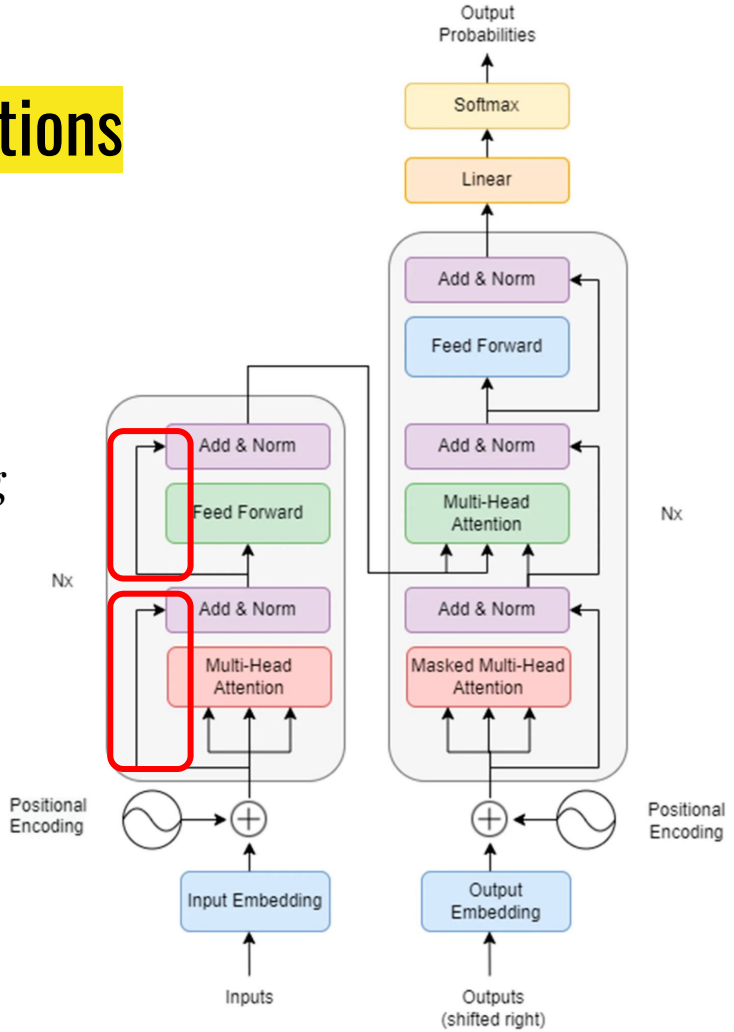
Transformers Architecture - Feed Forward NN

- Feed forward NN introduces the regular NN linear layers with activation functions.
- Preserves the original input shape



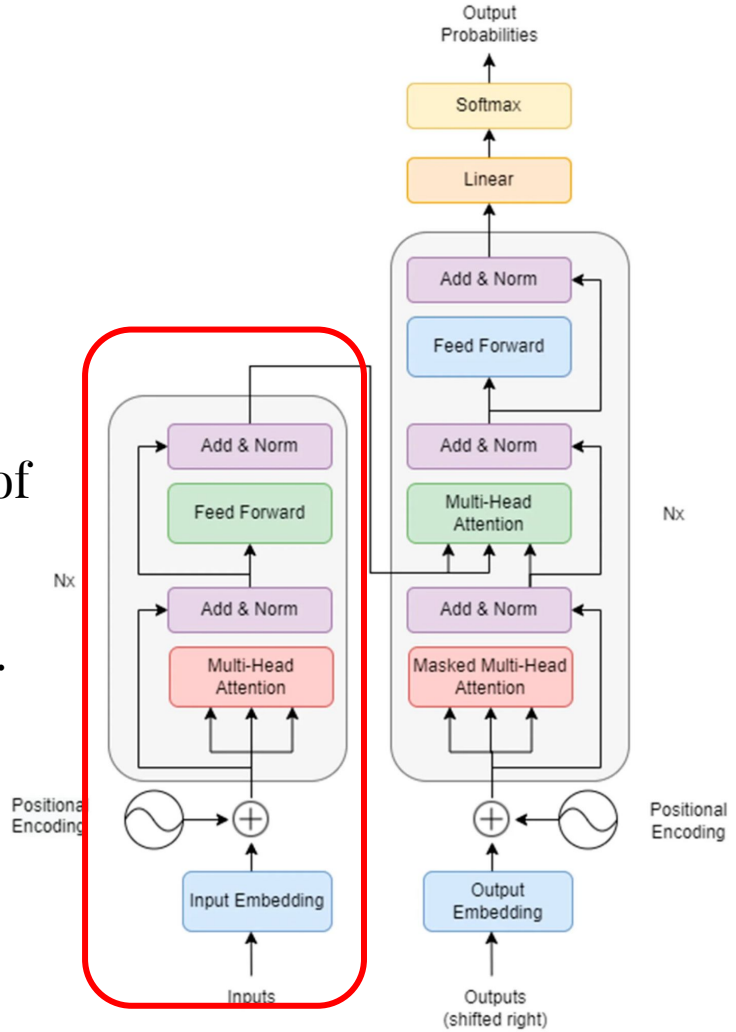
Transformers Architecture - Skip Connections

- As in ResNets, add the output of the previous block to the next block.
- This helps smooth the loss and make training easier.



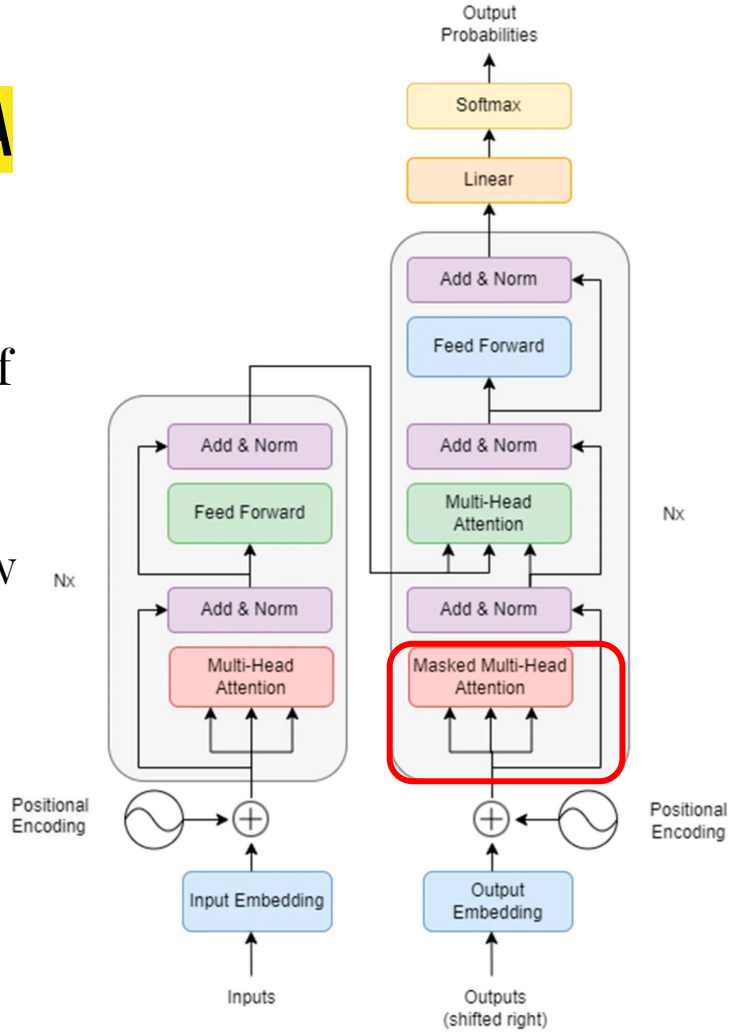
Transformers Architecture - Encoder

- The outputs z can be calculated in parallel which speeds inference using GPUs.
- We can stack more than one encoder on top of each other to introduce complexity to the model and get better contextual embeddings.



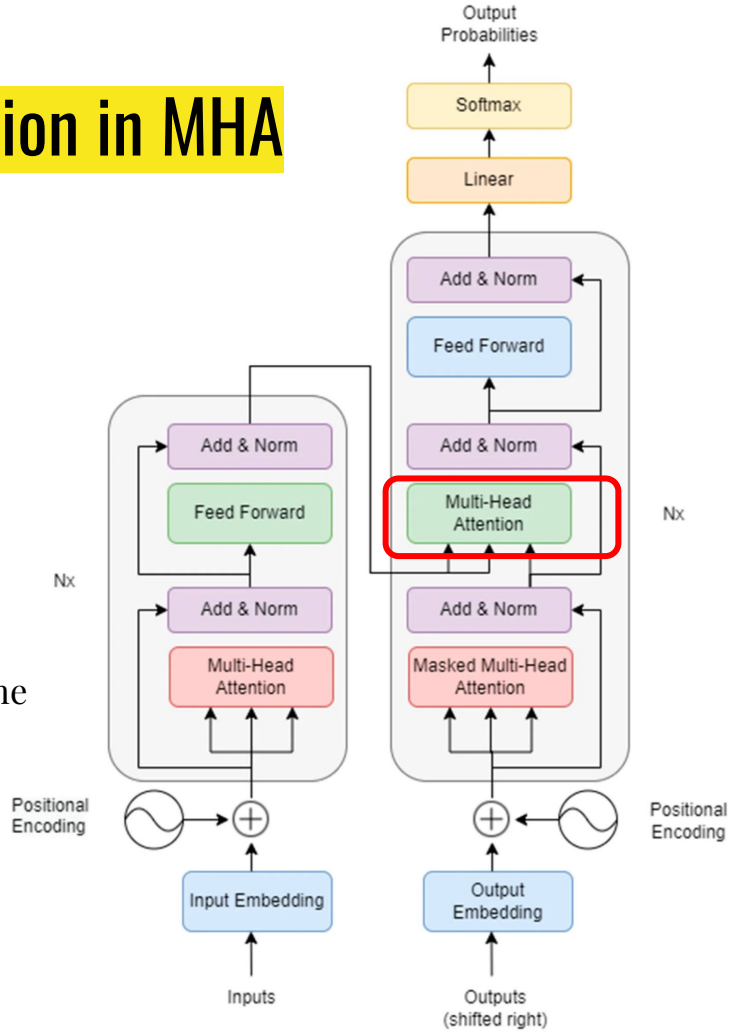
Transformers Architecture - Masked MHA

- Masked multi-head attention is the same as regular attention but “masks” the attention of the outputs.
- “I love dogs.” as the output “I” shouldn’t know that “love” comes after.
- Preserves the “auto regressive” trait.



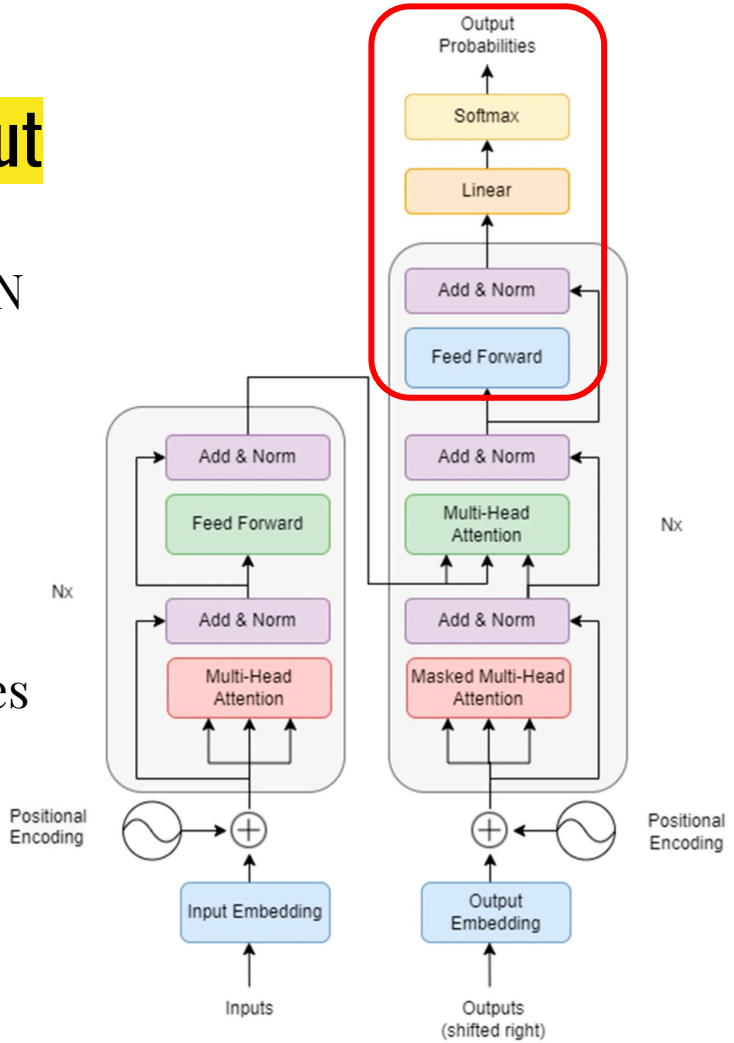
Transformers Architecture - Cross Attention in MHA

- K = encoder output × W_K (Key projection matrix)
- V = encoder output × W_V (Value projection matrix)
- Q = decoder hidden × W_Q
- Then it's normal self-attention. This step combines both the encoder and decoder outputs.
- This can be stacked too, but K and V are constant from the same encoder output.



Transformers Architecture - FNN to Output

- The output of MHA goes through a normal NN with linear layers and activation functions.
- This NN and the linear layer that follows reshape the output based on the task.
- The softmax function converts to probabilities in the one-hot encoding space to be able retrieve the sequence in each step.



Code

- Self-Attention
- Transformer from scratch