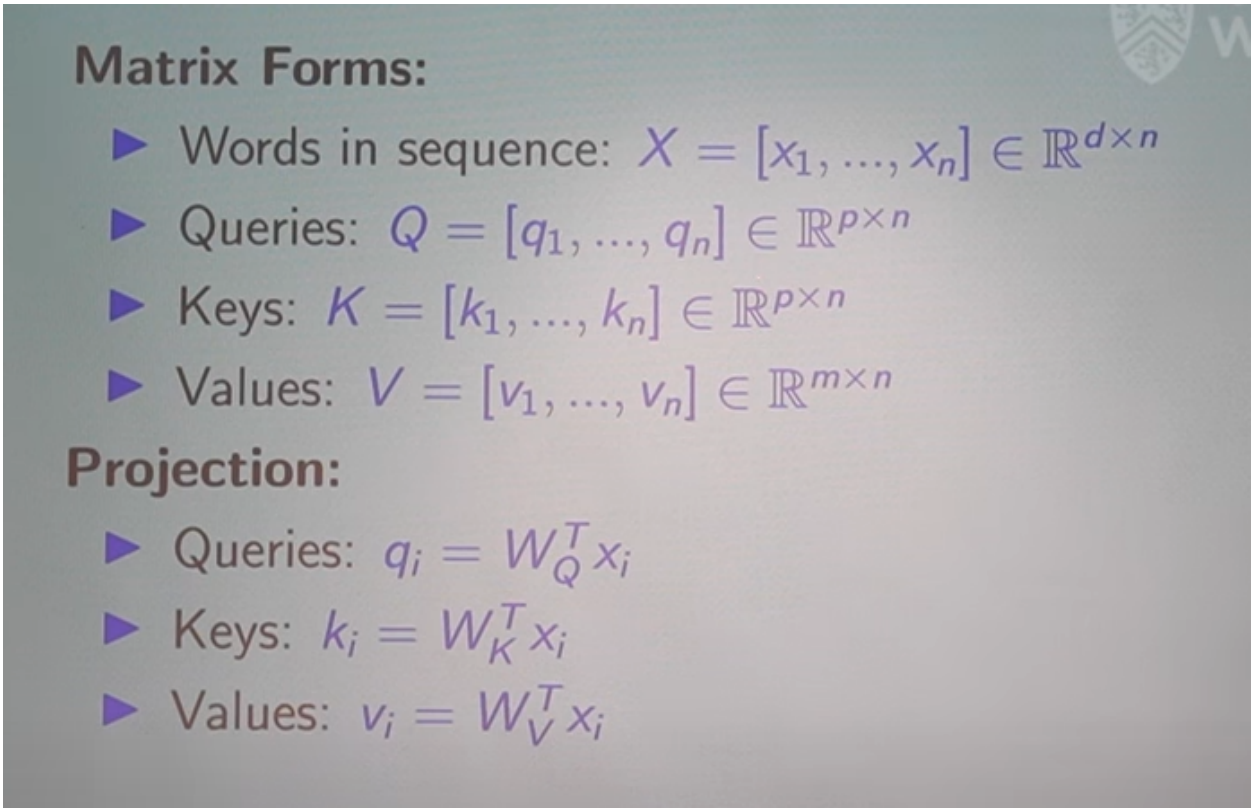
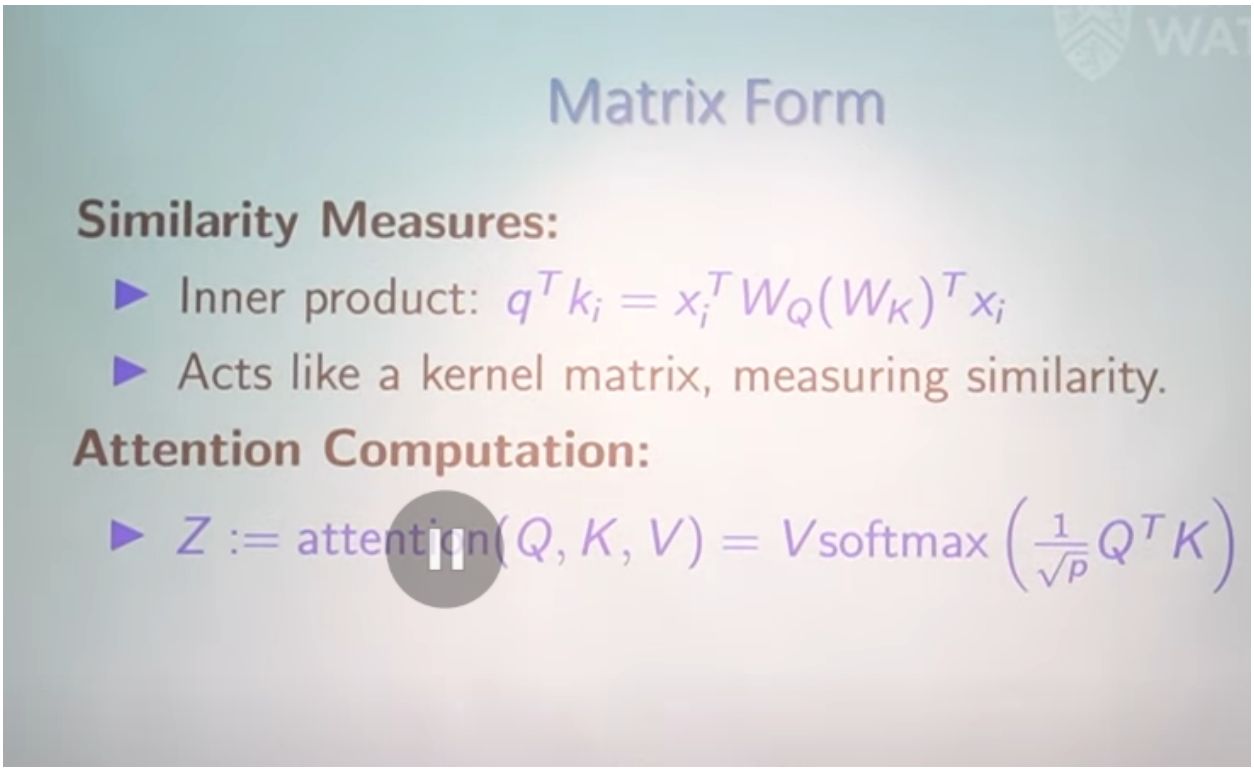
Lecture 10: Transformers

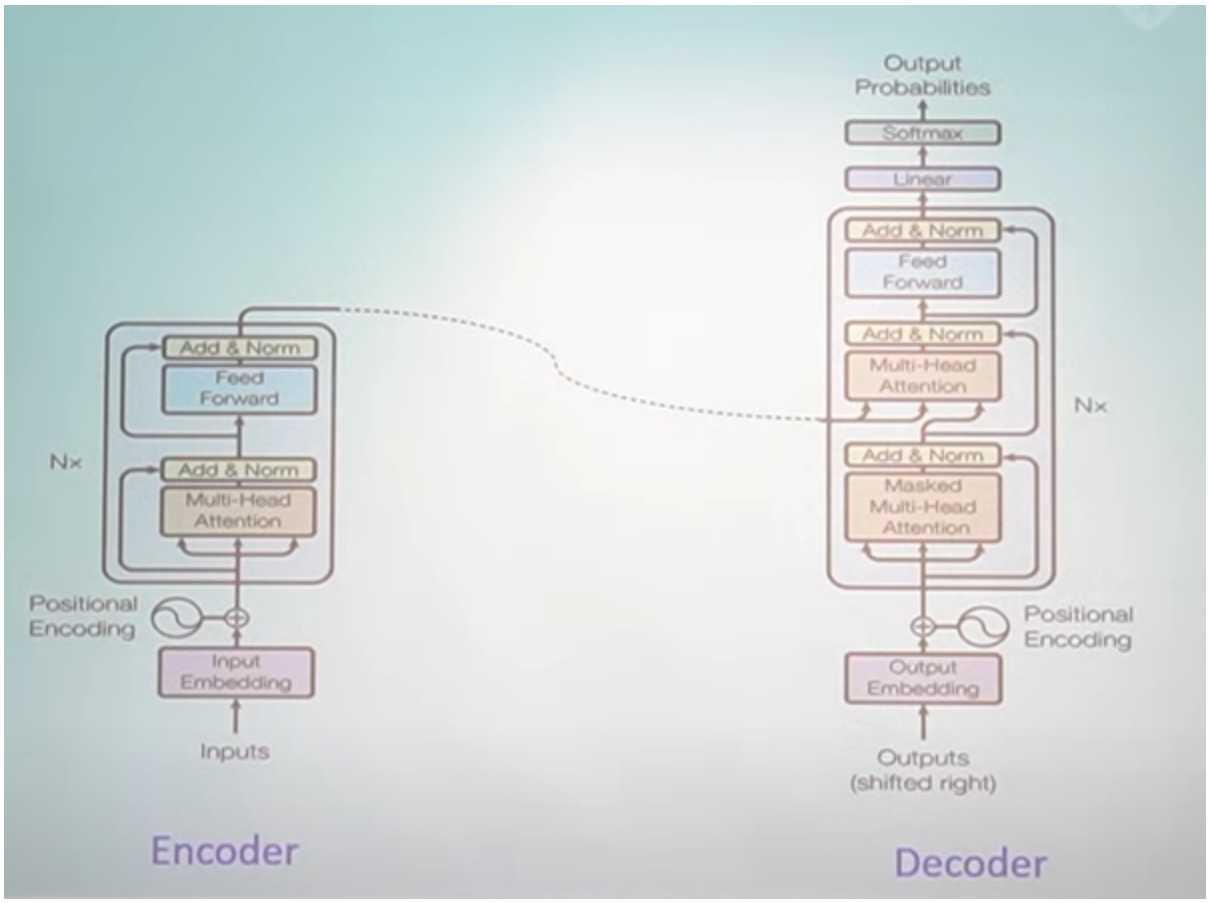


q and k need to be in same dimension, p. But v can be any dimension, say m as shown here.

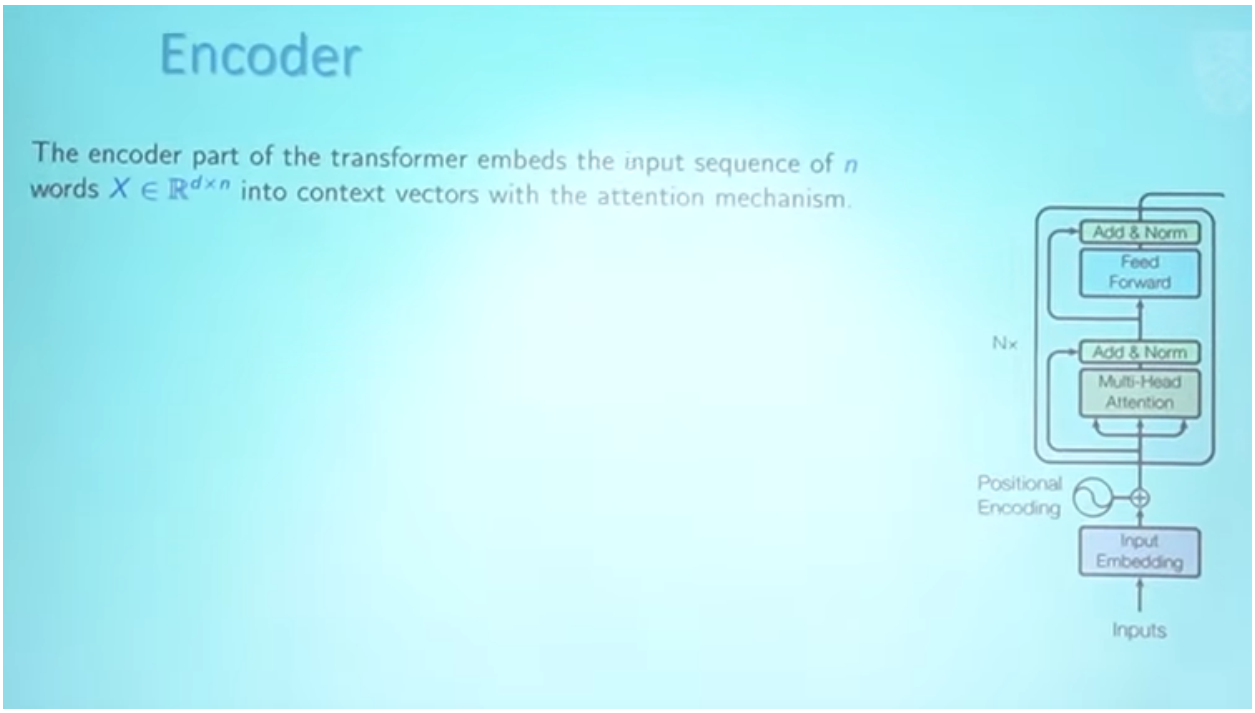


Transformer

Transformer is a model based on Attention.

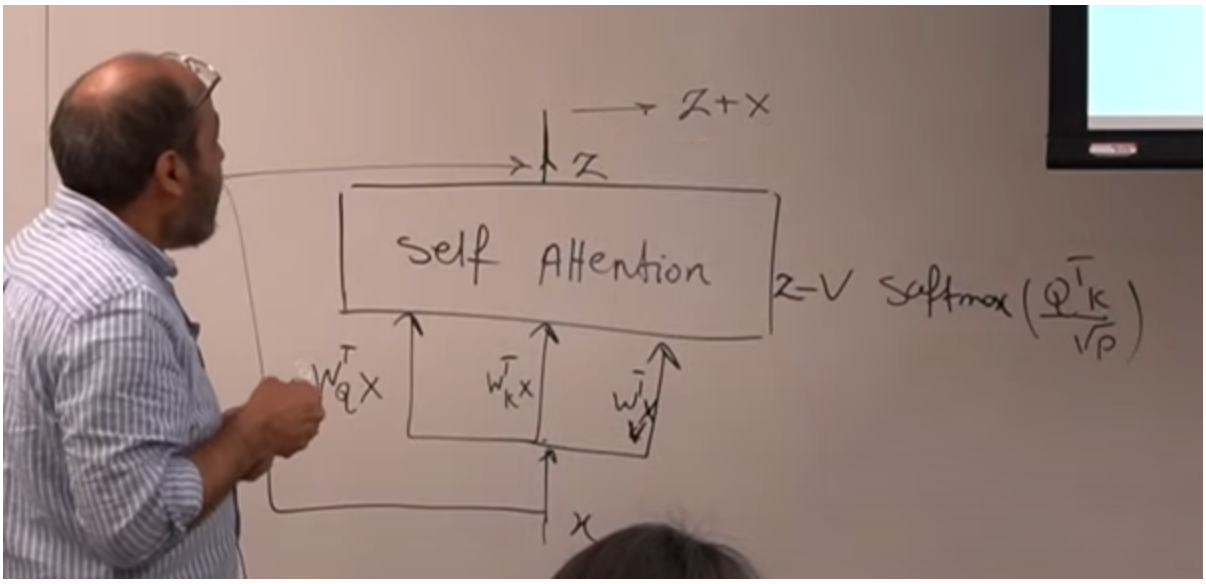

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs
Encoder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

Encoder

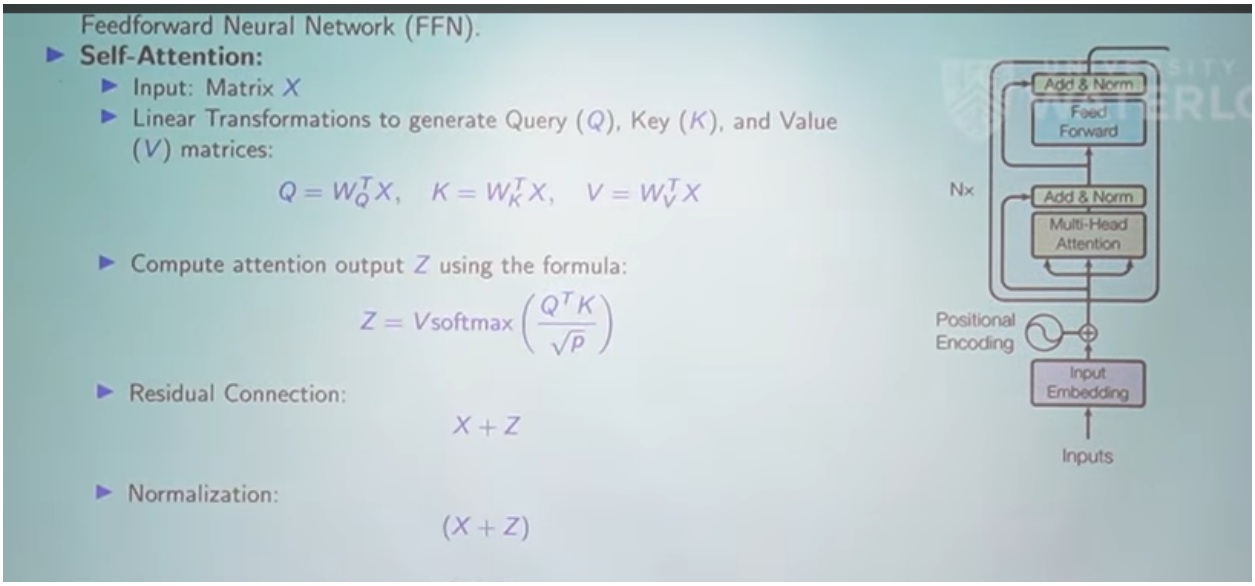


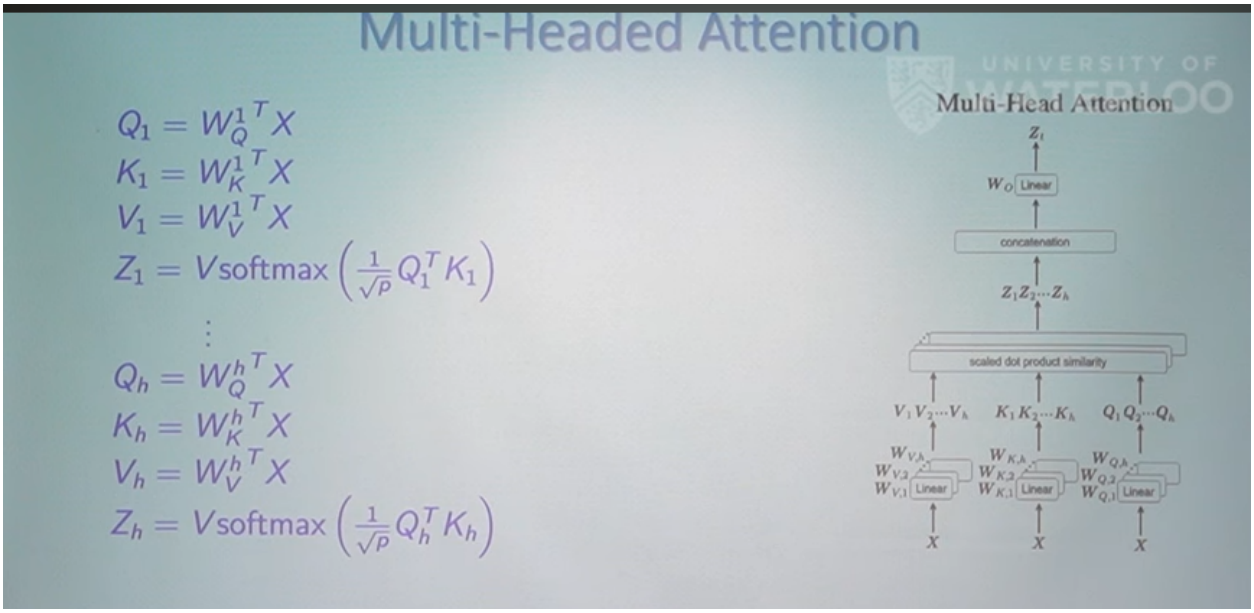
Encoder has 2 main components: Self Attention and Feed Forward Neural Network

Self Attention and Add-Norm with residual connection

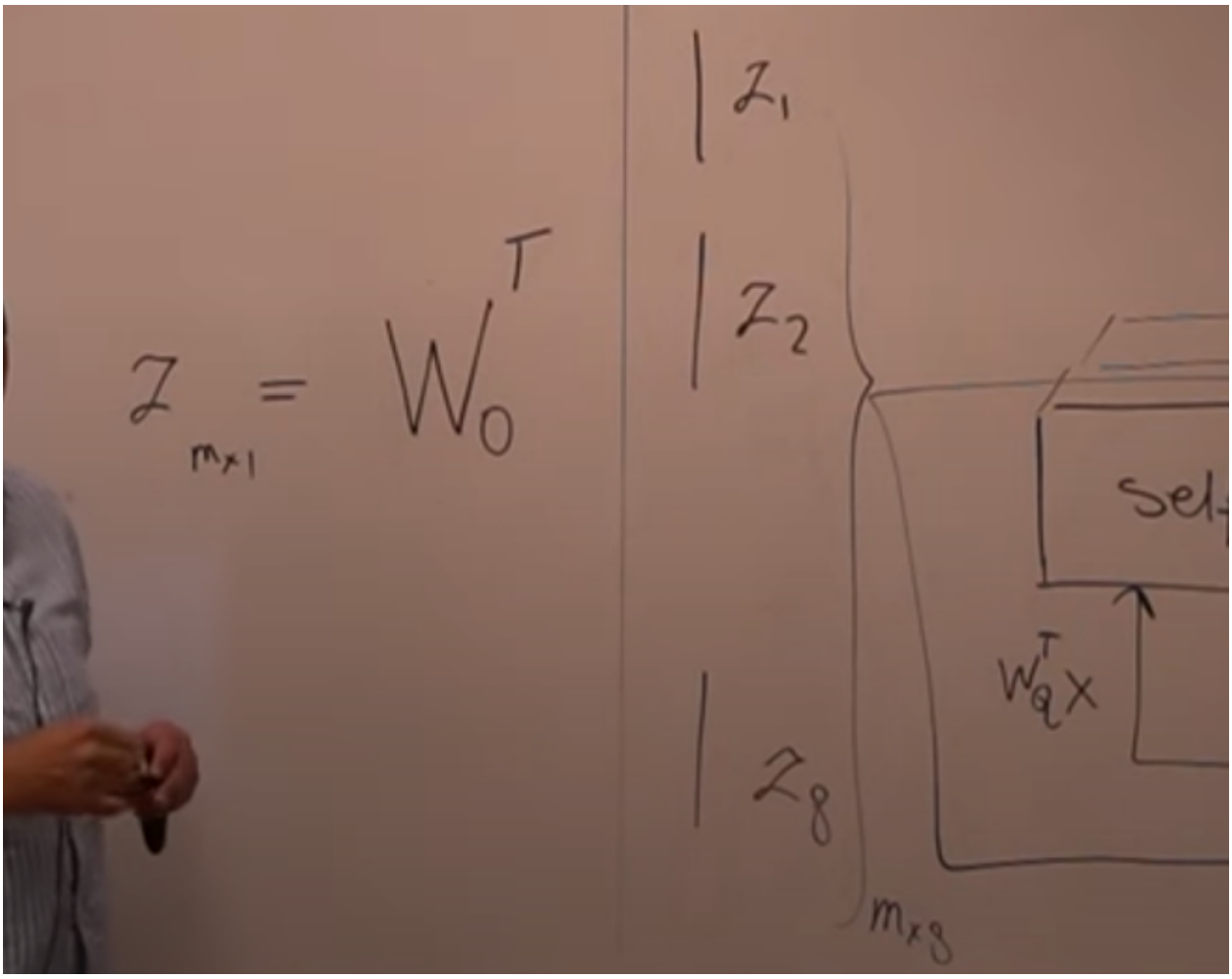


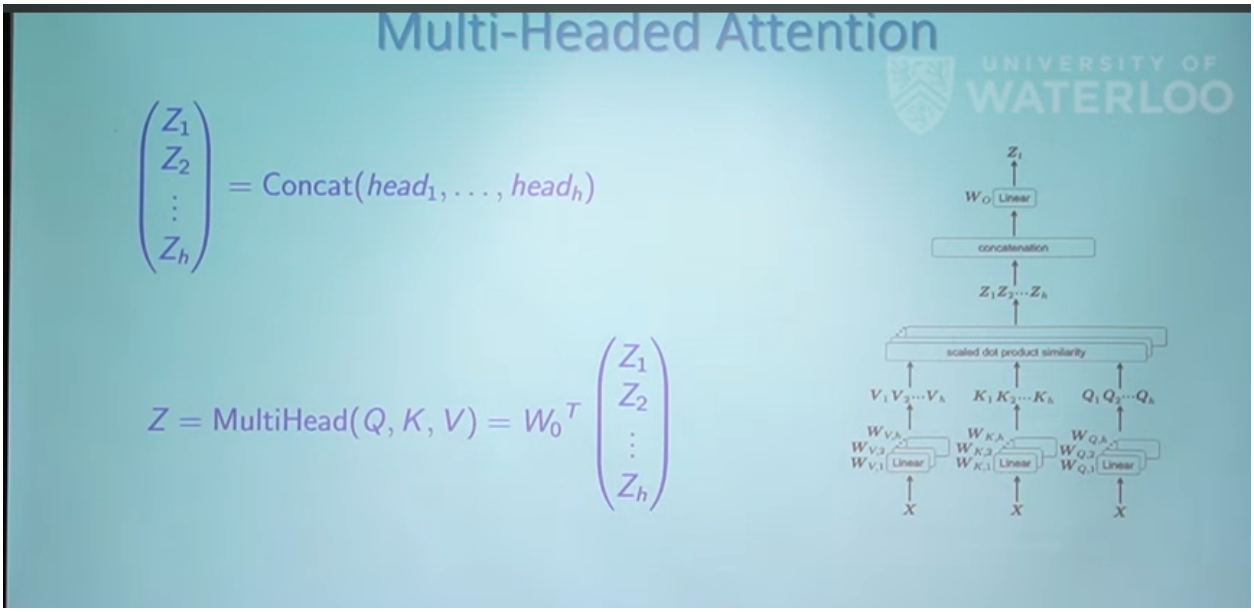
Why do we add X back to Z? Z contains information regarding the relation between words and what meaning they convey but we also need to preserve the individual identity of those words. For that reason X is later added to Z. After that we normalize the sum. The connection which connects X to Z for this sum is called Residual Connection





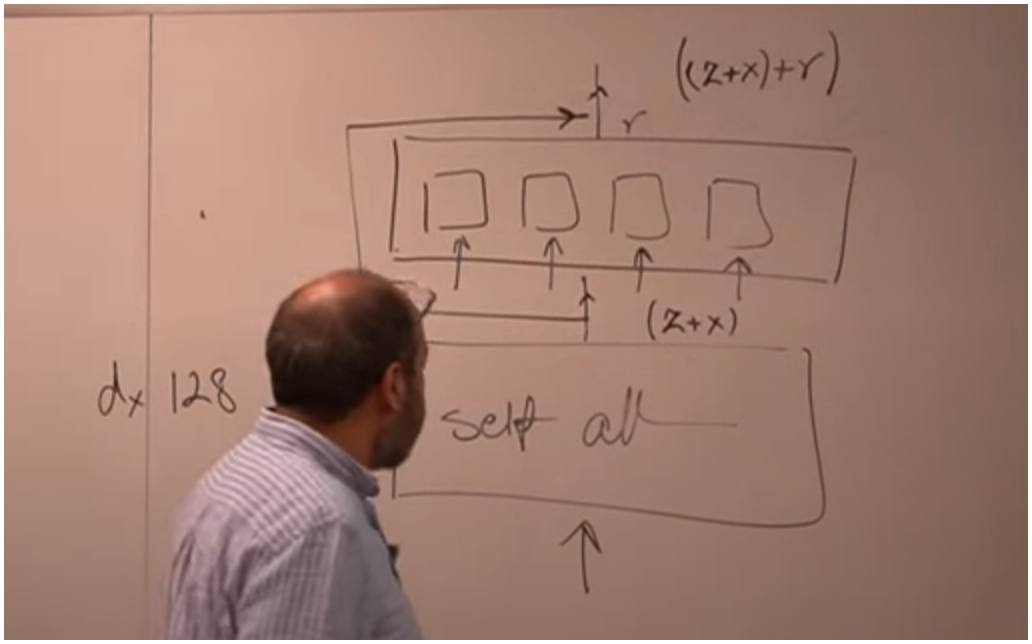
Instead of just one Attention layer, we actually have h of them, each being added to X and then normalized.



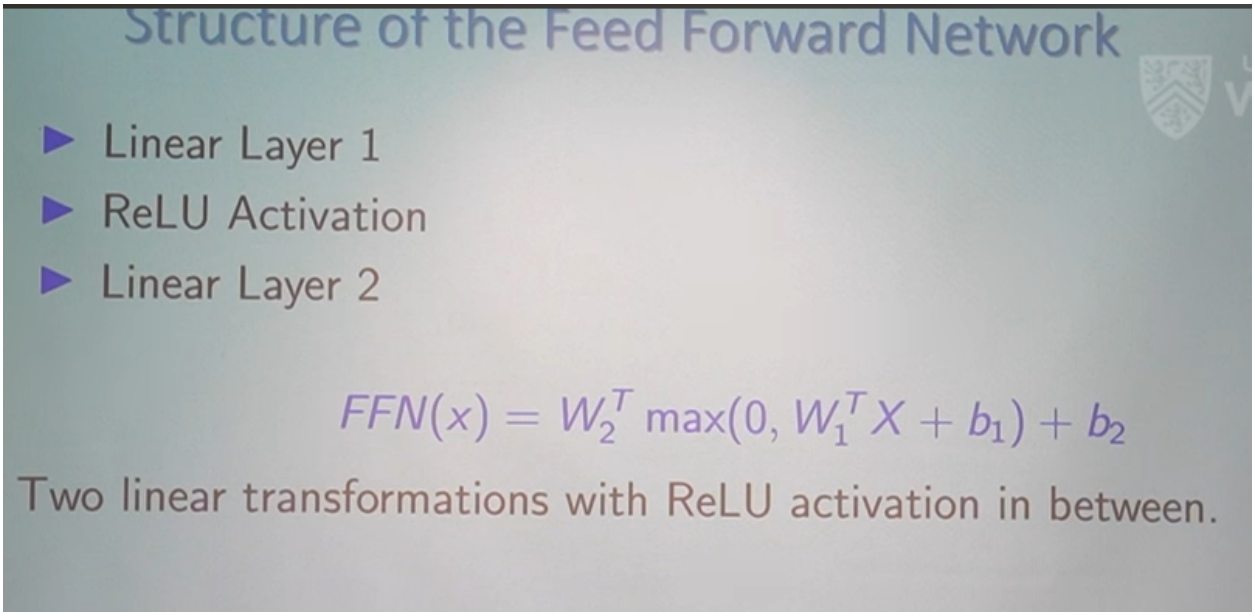


W0 is another parameter that will be trained via backpropagation

Feed Forward Neural Network



Suppose we can pass 128 words/tokens at a time to the model. Then dimensionality of X is dx128, of Z is mx128. Now each of these 128 Z+X is passed to a feed forward NN. Same NN is used for all the positions(tokens). Conceptually we have a NN for every position and and each NN has same weights. The output of this Feed Forward NN is r and we now add Z+X to it.



The reason for adding this Feed Forward NN could be to add more non-linearity to the model. But a more important purpose would be to making some sort of balance between global information and local information. Attention contains global information of relation between words, their composition. But we also need local information of individual words. Although that is done by the Residual connection in Self Attention but a seperate NN for every position acts like another step to refine this local information.

An analogy can help understand this better. Consider a classroom with a teacher and many students. Each student is a position.

In self attention, the students interact with each other and discuss topics while the teacher’s role is to pay attention. The teacher identifies groups that give different ideas.

In feed forward, there is an individual assessment of every student. Only group discussion is not enough to evaluate a student. The teachers role now is to assess students based on their individual performance.

A Classroom Analogy

- Attention Mechanism - Classroom Discussion:
- **Interactions:** Students (positions) in a classroom engaging in a discussion, sharing ideas, and interacting.
- Teacher's Role: The teacher (attention mechanism) observes who is interacting with whom, gaining a global understanding of the discussion dynamics.

A Classroom Analogy

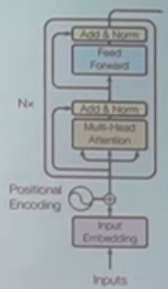
- Feed-Forward Network - Individual Assessment:
- **Teacher's Role**: The teacher (FFN) interacts with each student (position) independently, assessing their understanding and knowledge.
- **Independent Processing**: Each student is evaluated individually, akin to how the FFN processes each position independently.
- **Outcome**: Enhanced understanding and refined representation of each student's performance, akin to the FFN refining representations at each position.

A Classroom Analogy

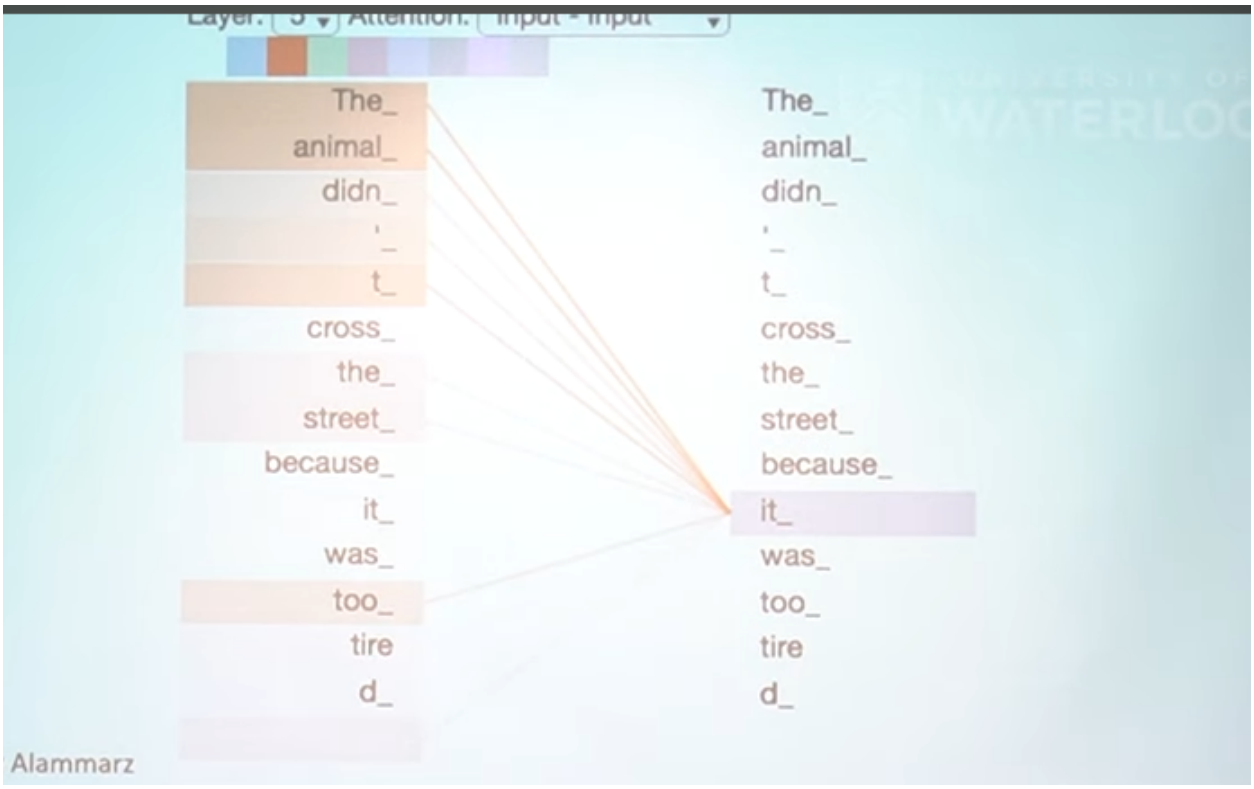
- Synergy of Attention and FFN:
- **Holistic Understanding**: The combination of global interaction observation (attention) and individual assessment (FFN) provides a holistic understanding of both group dynamics and individual performances.
- **Balanced Processing**: A balanced approach to processing global relationships and local, position-specific information, leading to richer representations and enhanced learning.

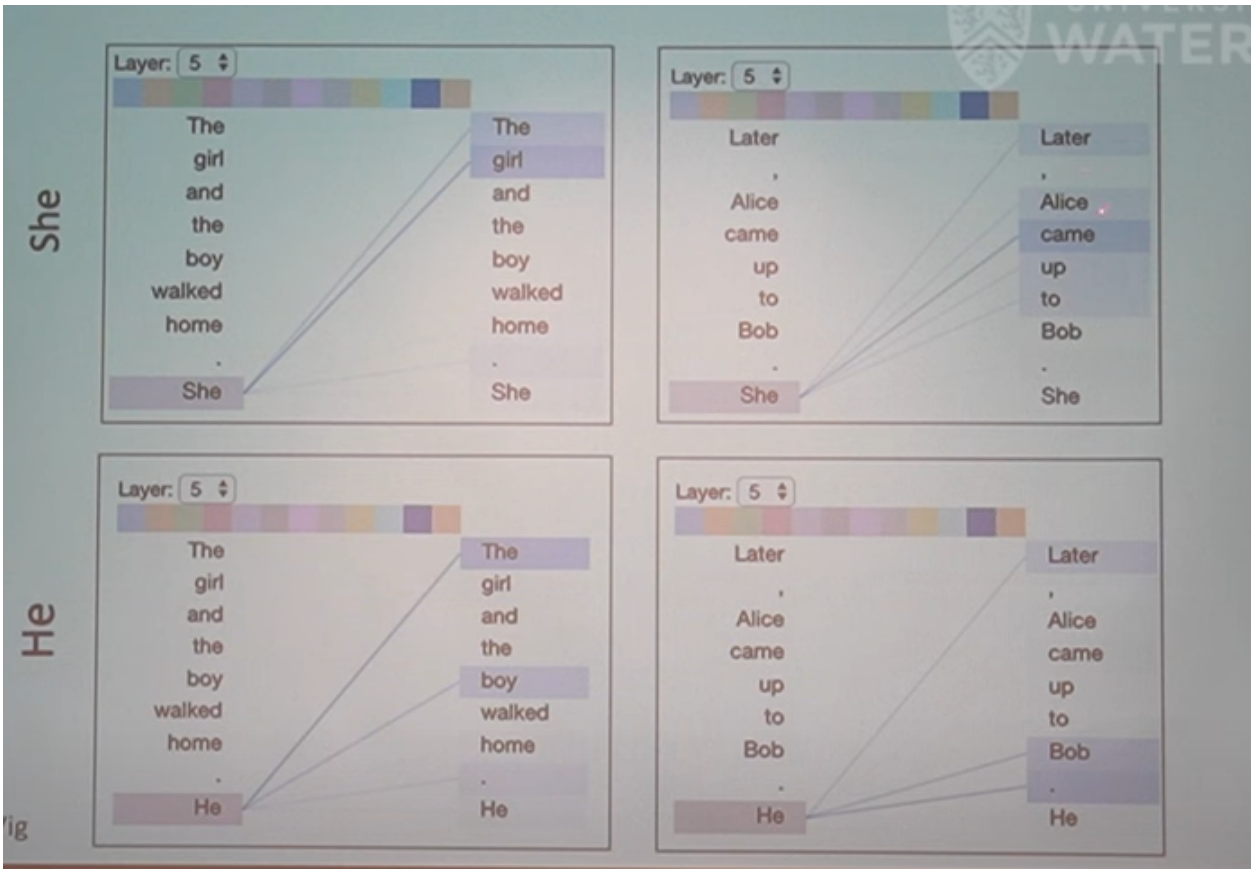
Encoder

If the output of the FFN is denoted by R, then a residual connection is established from the output of the previous layer $(X + Z)$ to the output of the FFN, resulting in $(X + Z) + R$.This will be normalized $((X + Z) + R)$to form the output of the encoder.



Visualize


The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
Alammarz


She
He
Layer: 5
The girl and the boy walked home . She
Later , Alice came up to Bob . She
The girl and the boy walked home . He
Later , Alice came up to Bob . He

Decoder

Decoder has 3 parts:

Masked Multi-Head Attention

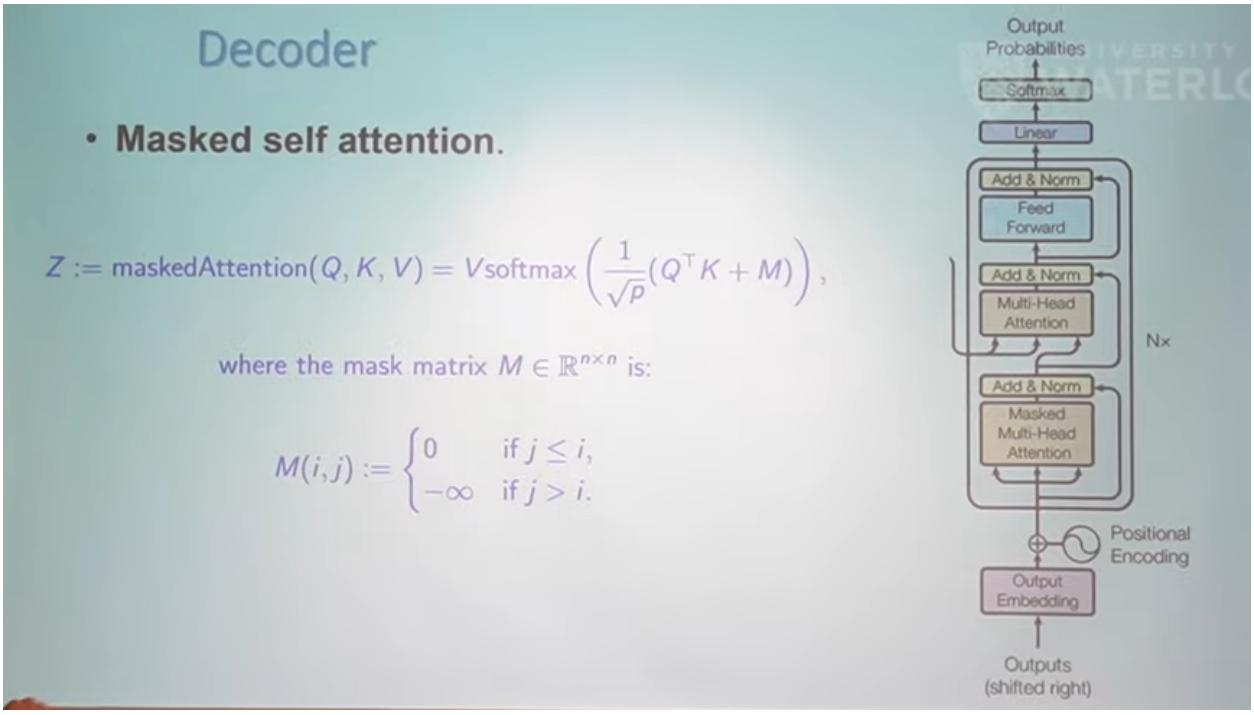
Cross Attention

Feed Forward NN

Masked Self Attention

In self attention, we find similarity of a word with every other word in the input sequence, be it before or after that particular word. But in case of decoder, we are generating a sequence, we don’t have words after the word we are currently predicting. We only have words predicted till that time. So we want to attend to only these words and mask the tokens after the current word.

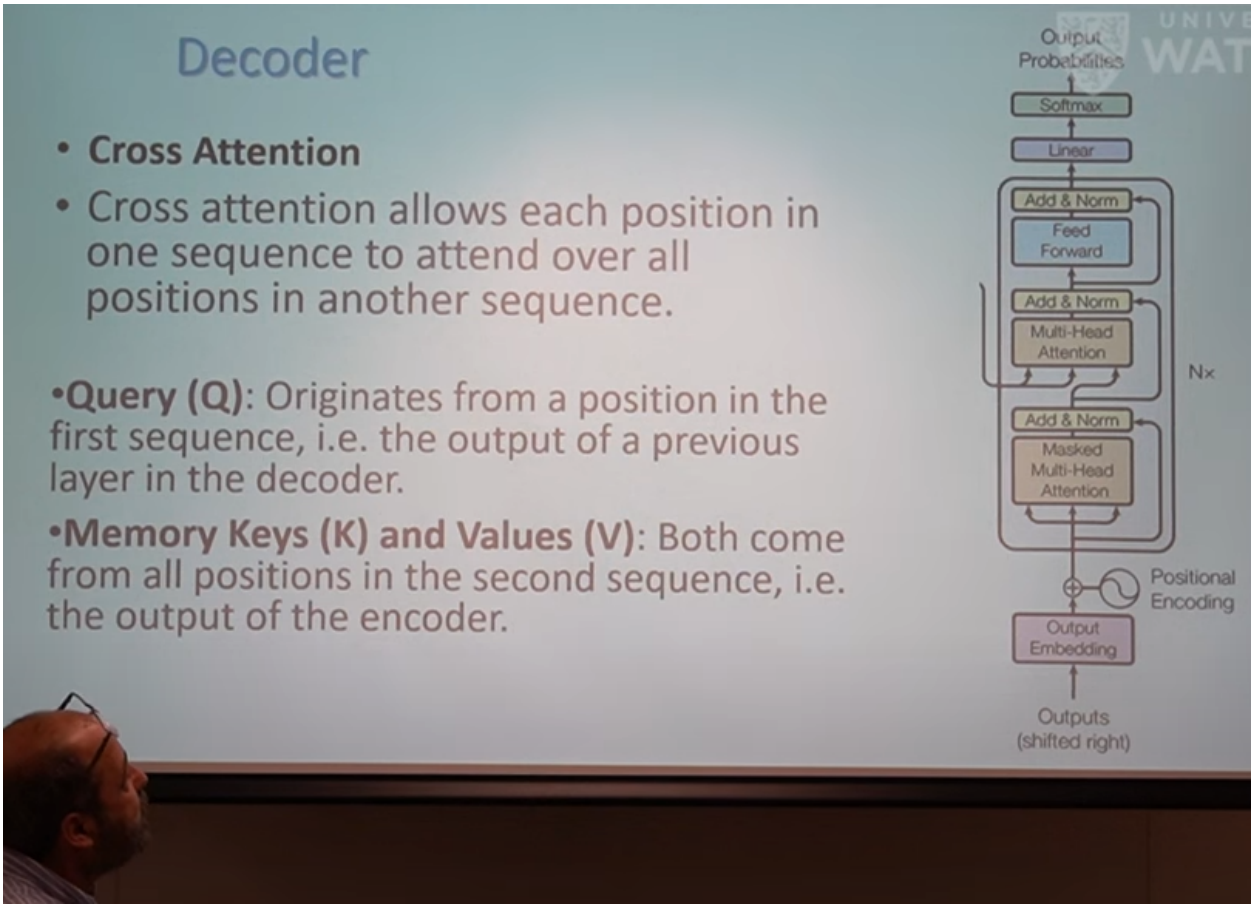
In order to do this, we add a matrix M to the dot product of Q and K such that it keeps similarity scores of the between the words predicted till now and eliminates the other scores(scores for words that are going to be predicted).



After this it is the usual residual connection and Add & Norm layer.

Cross Attention

We want to find the similarity between the output of Masked Attention, which acts as query, and the output of the encoder, which acts as key and value.



As we have done earlier, here also residual connection and Add & Norm layer is added.

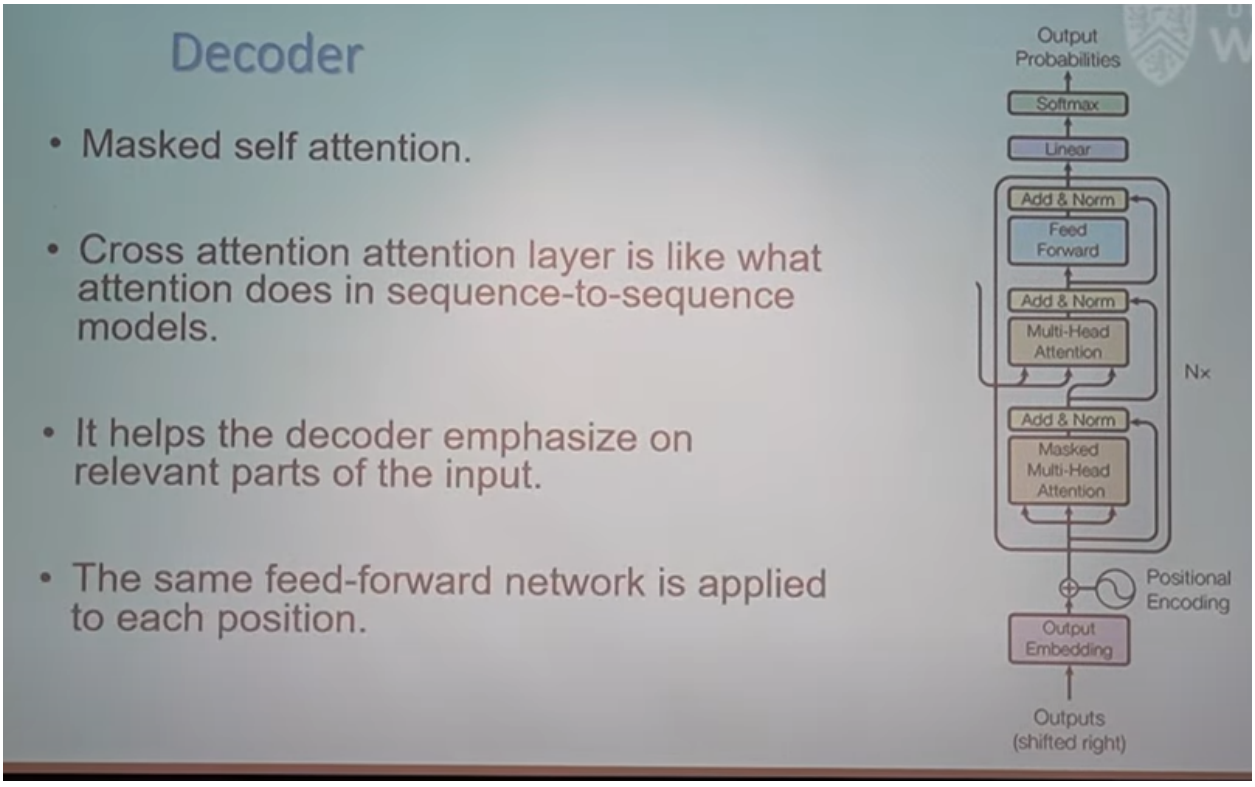
Feed Forward Neural Network

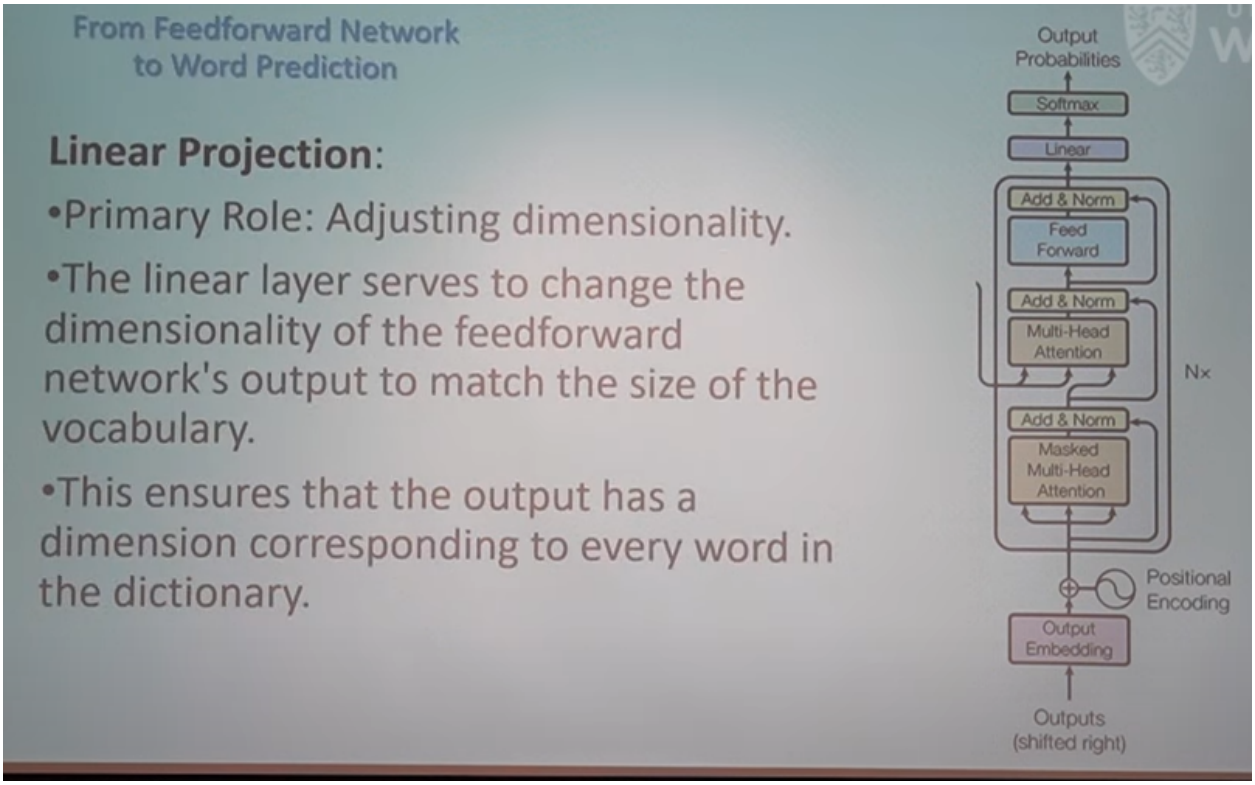
This Feed Forward Neural Network is the same as we had seen in the Encoder.

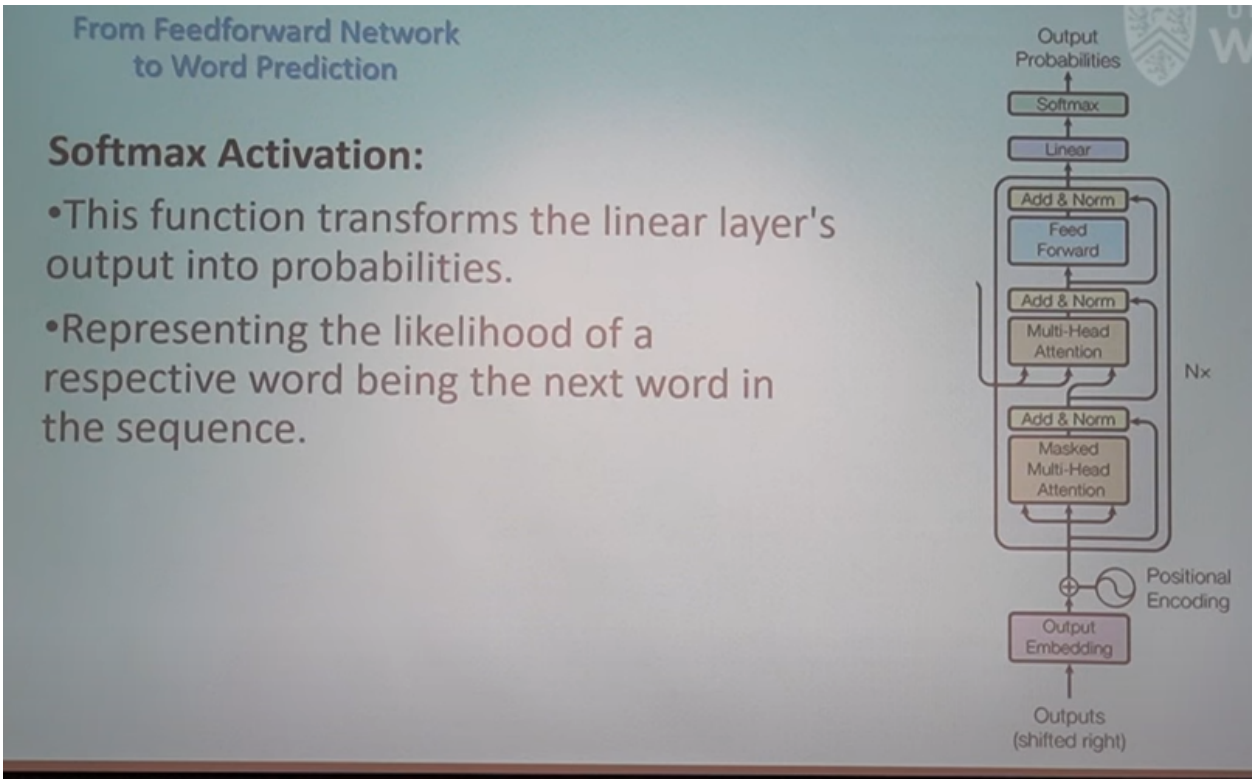
Note: We have multi-head attention. So we actually have h attention layers. Apart from that we have N blocks of all the three components which are projected linearly towards the end.

Now the Z obtained at the end, will be multiplied by W0 to change the dimensionality such that we have the same dimensionality as the number of words in the vocabulary. Why do we need this? The next word we are going to predict is going to be from our vocabulary. So we want the probability of every word. This probability is obtained as output of the Softmax layer.

Note: Transformer is a Encoder-Decoder model wherein it maps the source domain to the target domain. The Encoder encodes the source domain, passes it to the decoder which then gives the representations in the target domain.







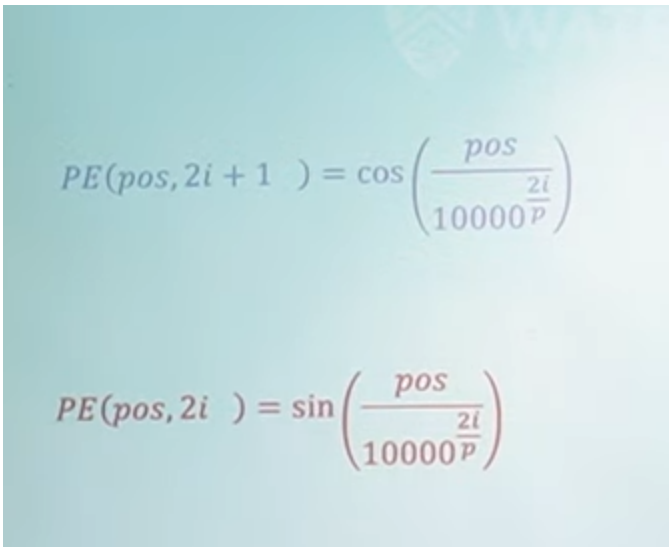
Positional Embedding

So far the model has no sense of word/token ordering in our sequence.

Positional embedding is a way to encode the positions of tokens in our input sequence. These encodings can then be added to the word embeddings.

There are many ways to obtain position embeddings. One way would be to learn them via backpropagation. However, there is a much simpler method to obtain them and the performance difference isn’t much.

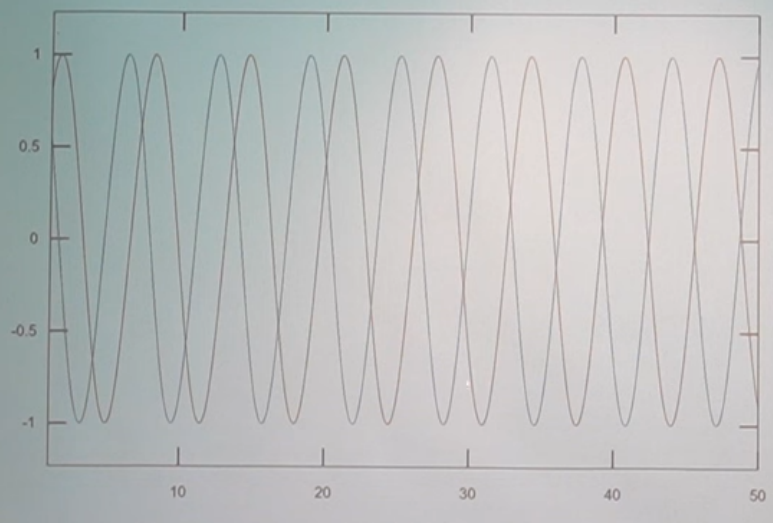
The position encoding vector could be a binary vector. [00..0] for 1st word, [100..0] 2nd word, [0100..0] 3rd word, [1100..0] 4th word and so on. But the paper suggests keeping continuous values in the position vector instead of 1s and 0s in the vector. To ensure this they use sine and cosine functions. Cosine fills the odd entries in the vector while sine fills the odd entries.



pos = position of word/token
i = entry number in position vector

Now as the value of i increases the frequency reduces. This means that the starting entries in the vector change more frequently than the latter entries.

Positional Encoding



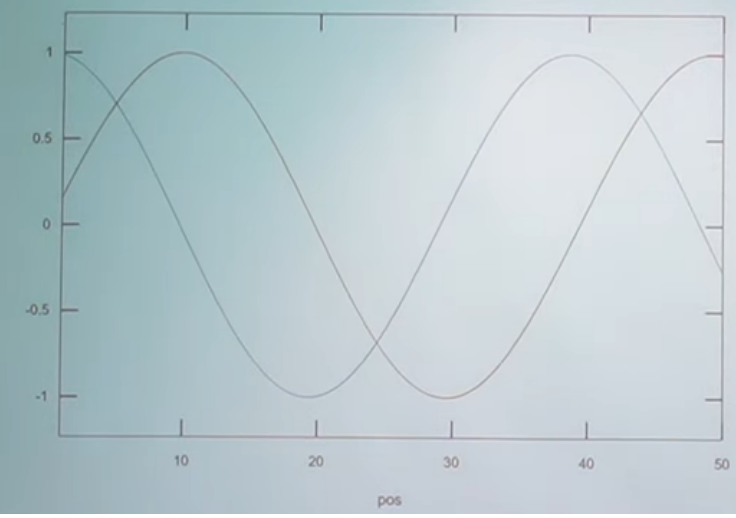
$i = 0$

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

$i = 1$

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

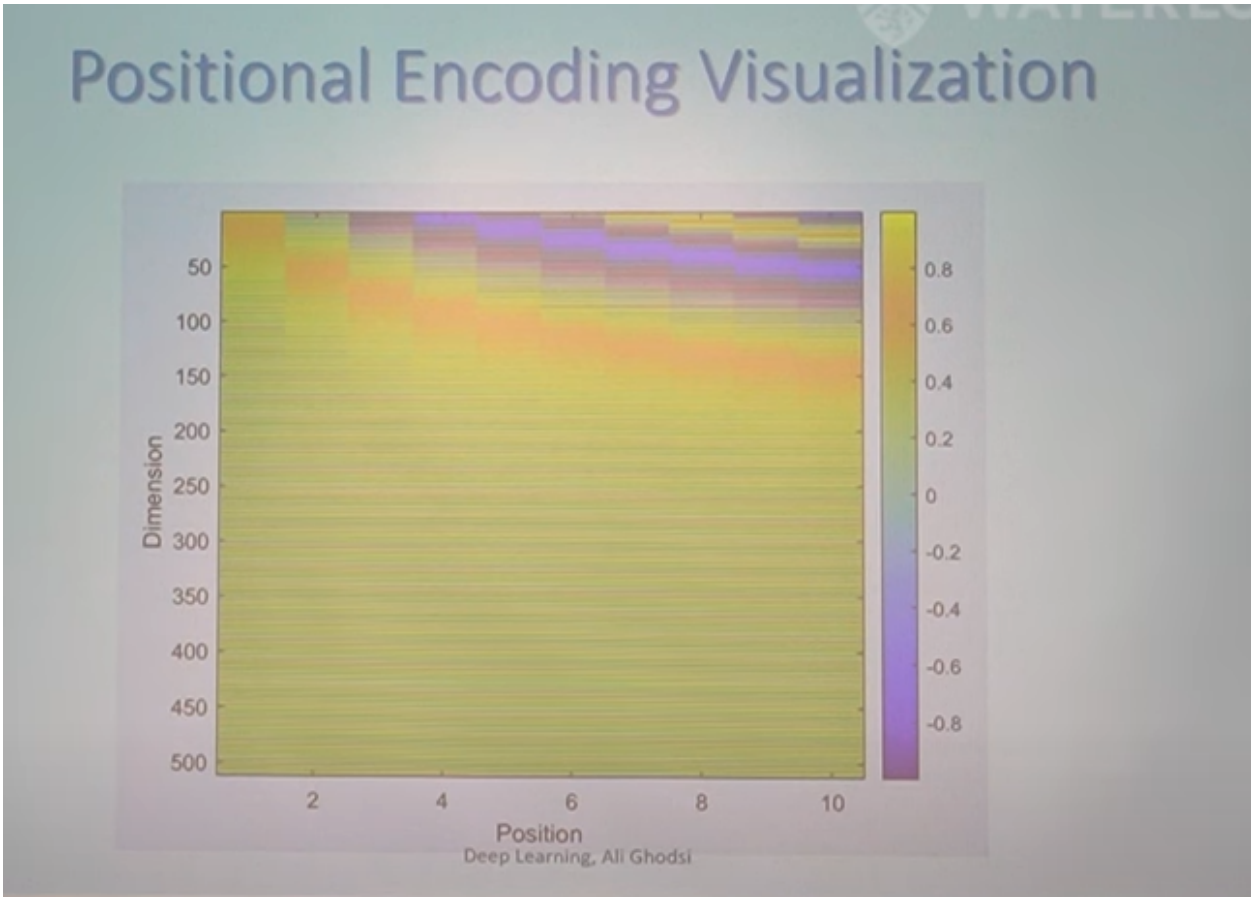
Positional Encoding



$i = 50$

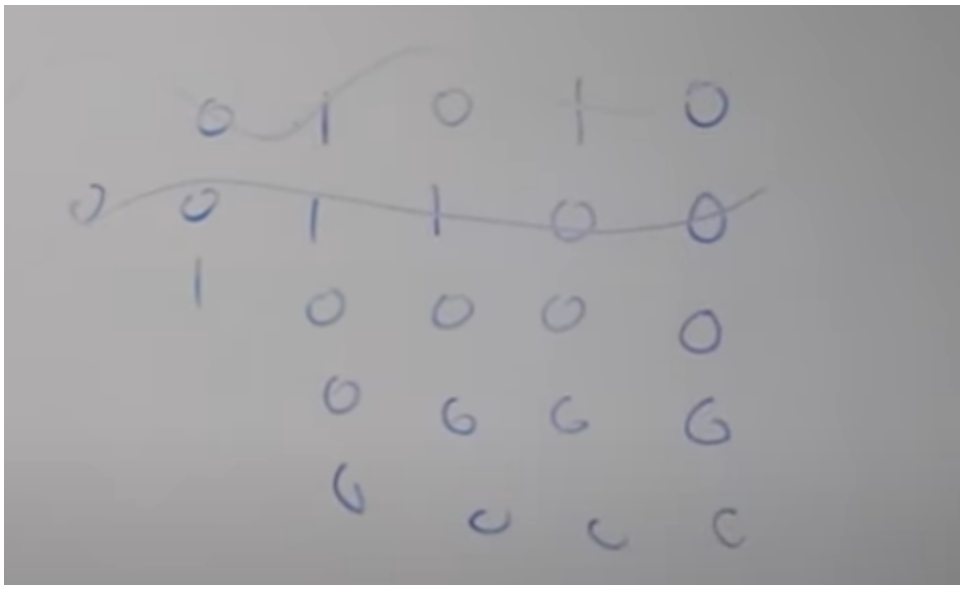
$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{p}}}\right)$$

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

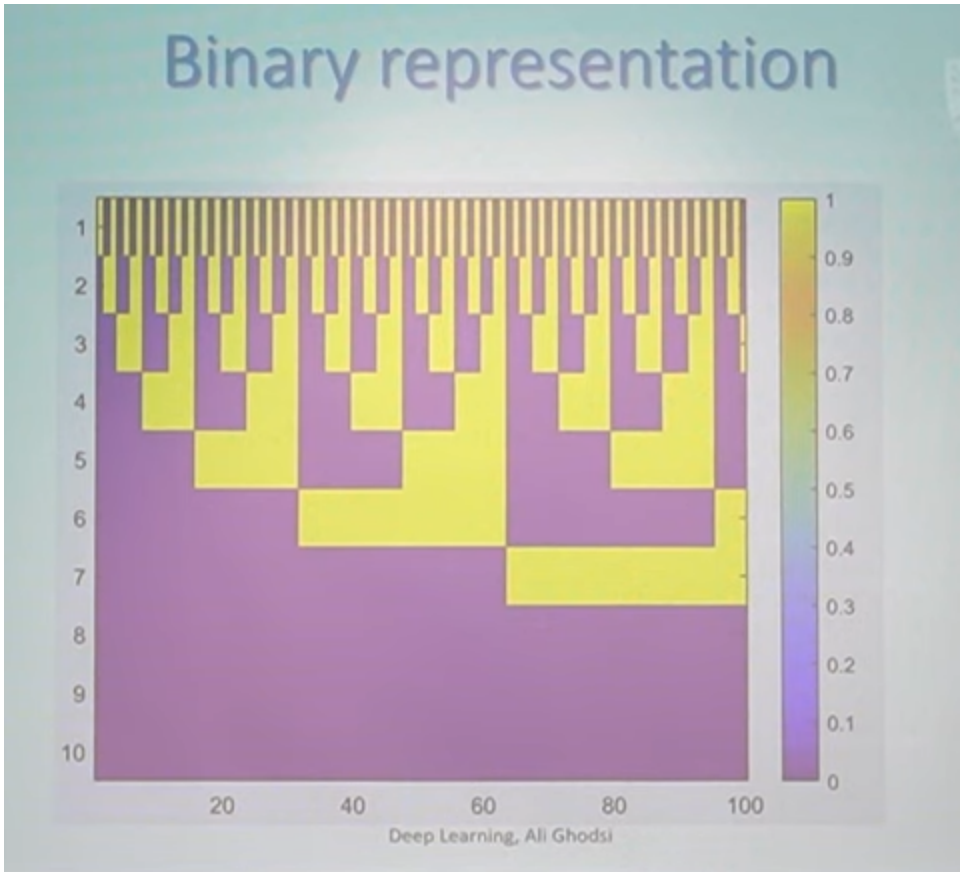


The blue here refers to 1 in one hot vectors.

Note: Positional embedding is more aligned with the notion of binary encoding.



The frequency of change in bits is higher in initial entries of vector.



Note: Positional encoding is added to the word embeddings before passing it to the respective layers in the encoder and decoder

for learning.

Sneak-peak into the next lecture:
BERT is a stack of encoders while GPT is a stack of decoders.

With BERT you can find word representation of sentences while with GPT you can generate text.