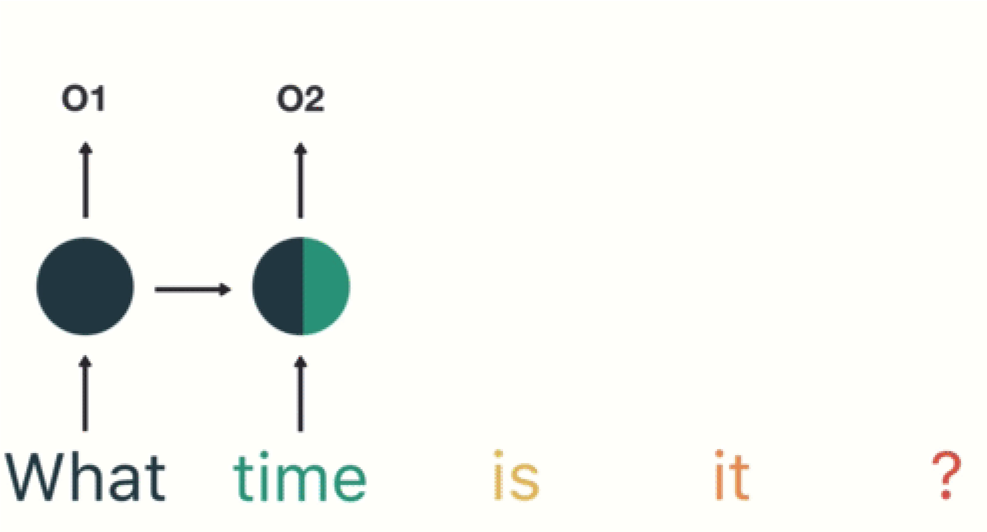
CENG 506 Deep Learning

Lecture 11 – Transformers

Slides were mostly prepared using Visual Guide to
Transformer Neural Networks videos by Hedu AI
(https://www.youtube.com/c/HeduMathematicsofIntelligence/videos)

Refresher

RNN:

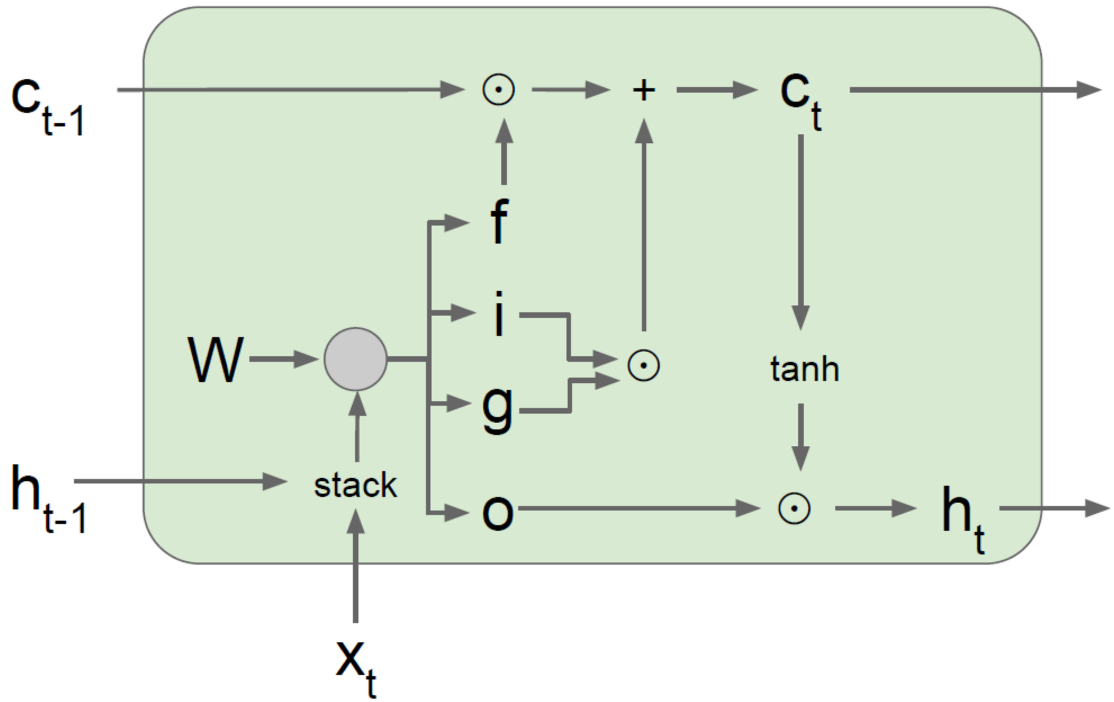


Problems with RNN:

- Problem of vanishing/exploding gradients
- They take input sequentially one by one, which does not use GPU's well, which are designed for parallel computation.

Refresher

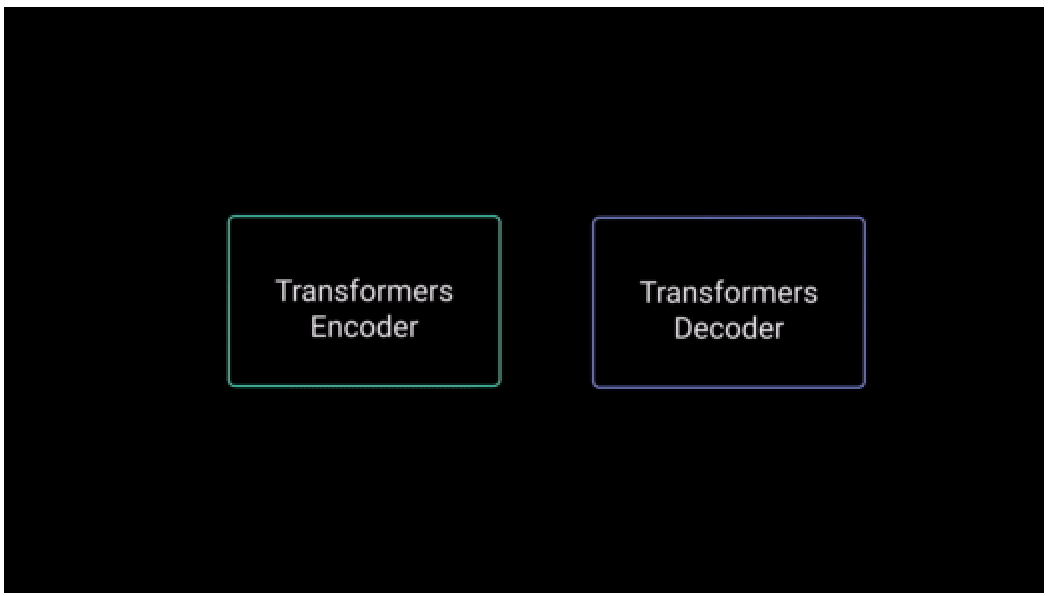
LSTM:



Problems with LSTM:

- Better than RNNs, but slower than RNNs
- Still no parallel computation.

Transformers



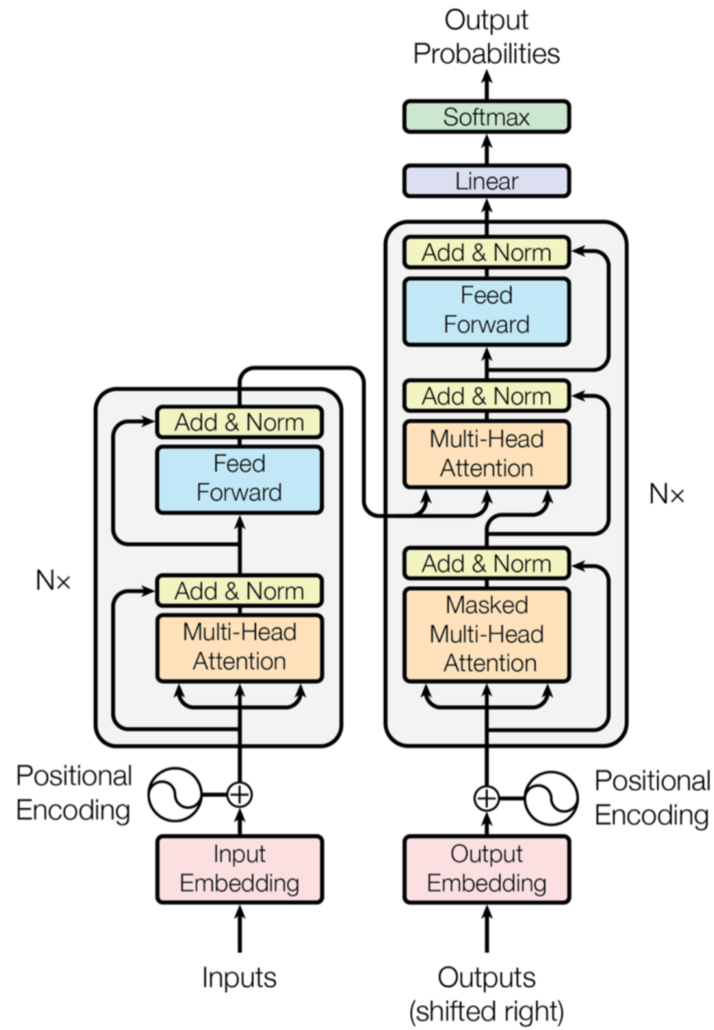


Figure 1: The Transformer - model architecture.

Figure source: Vaswani et al., "Attention Is All You Need", NeurIPS 2017

Transformers

How it works in high level:

Transformers

1) Before ‘attention’
2) Encoder
3) Decoder

Before ‘attention’:

a) Input

b) Embedding layer

c) Positional encoding

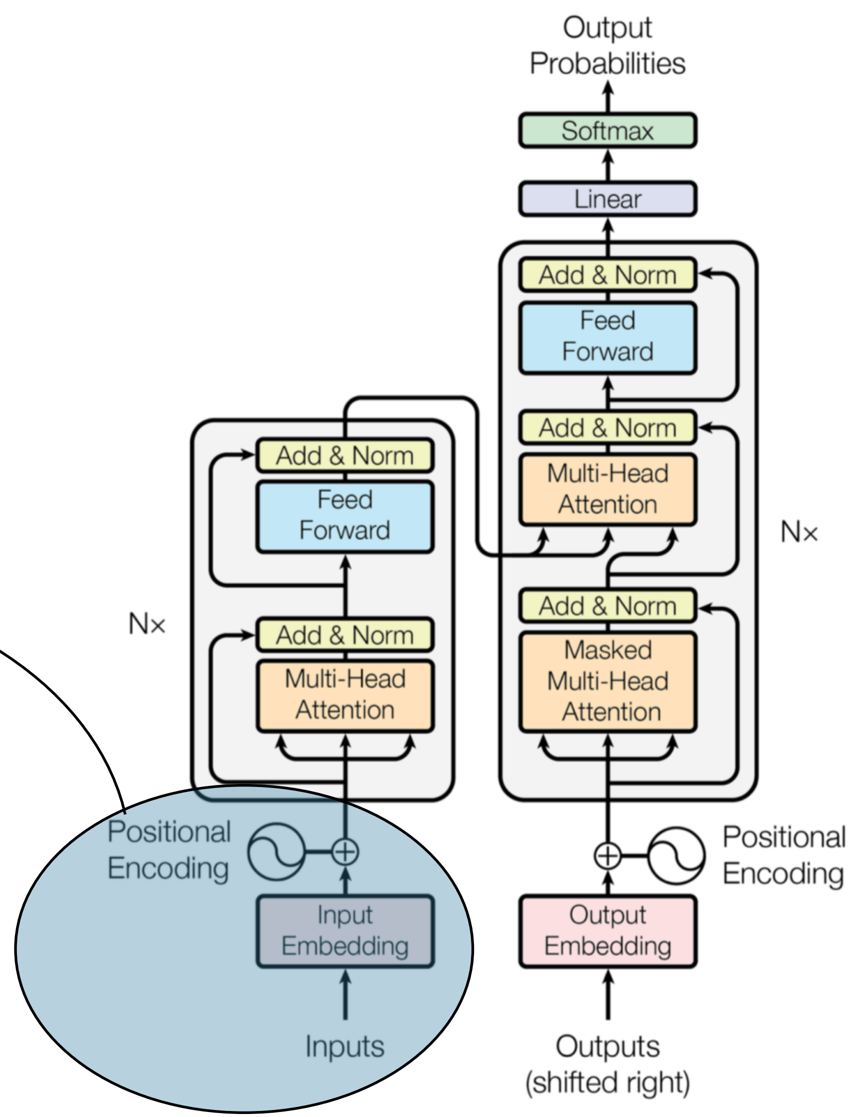


Figure 1: The Transformer - model architecture.

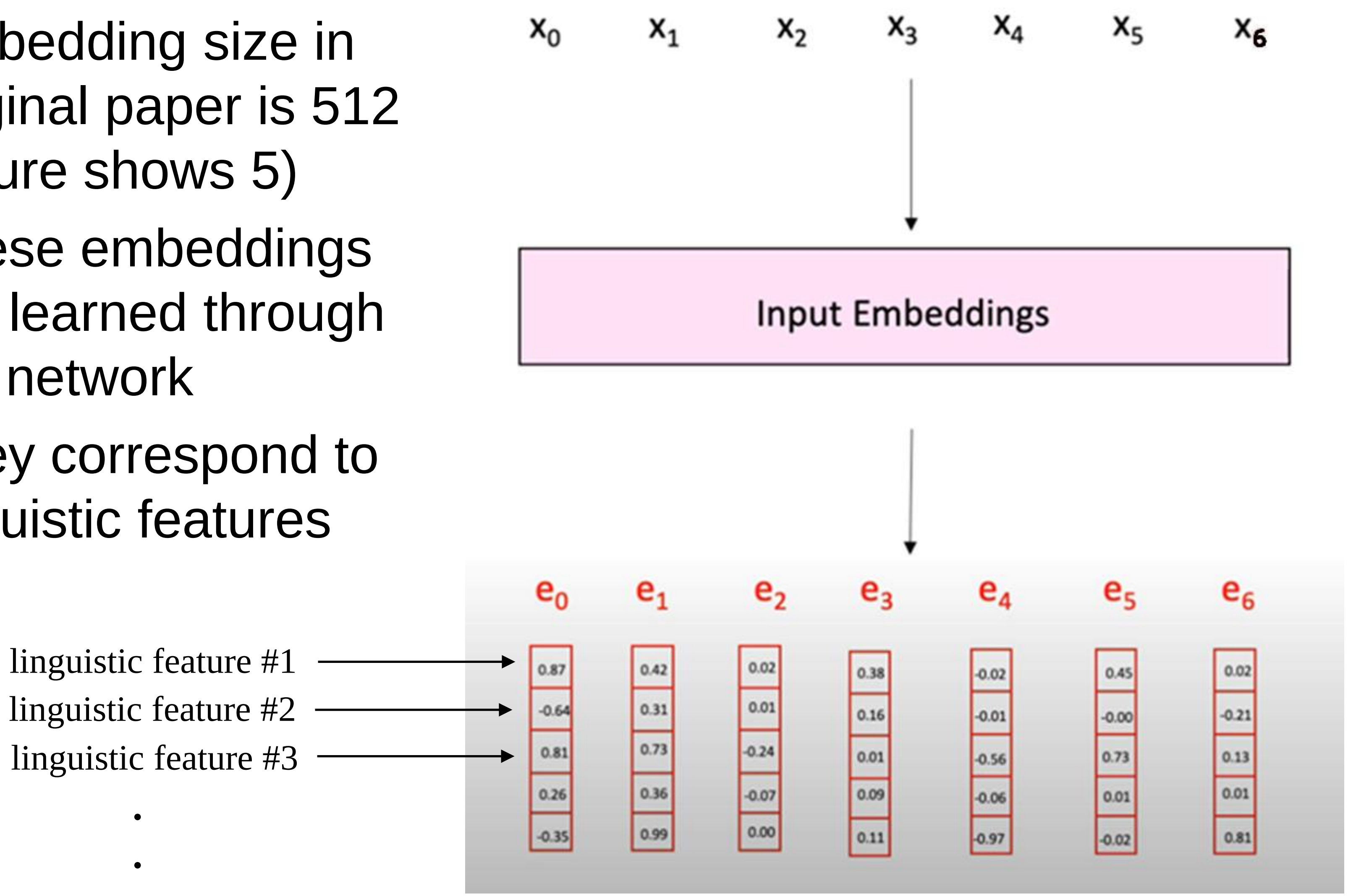
Vaswani et al., “Attention Is All You Need”, NeurIPS 2017

Transformers: Input

We assign a numeric index to each word.

Transformers: Embedding layer

- Embedding size in original paper is 512 (figure shows 5)
- These embeddings are learned through the network
- They correspond to linguistic features



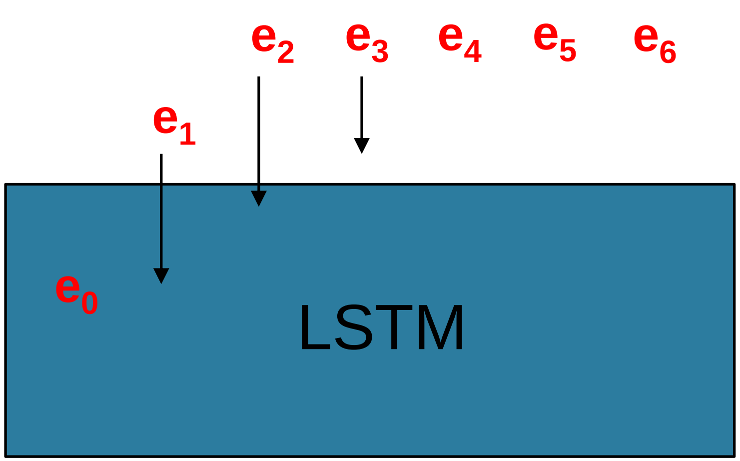
Transformers: Positional Encoding

Why the order matters:

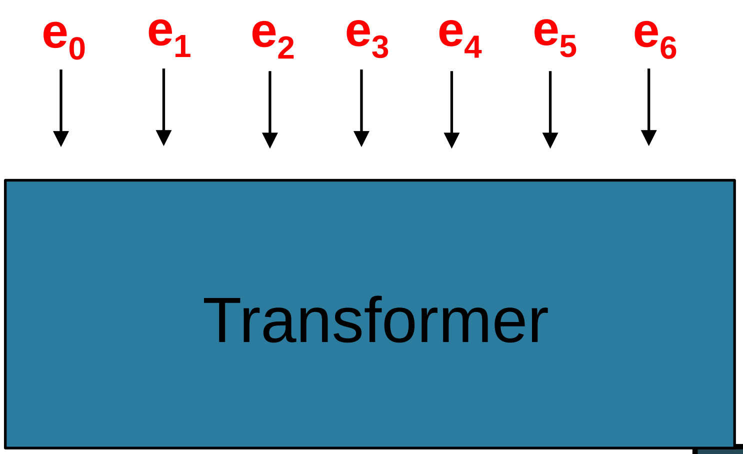
Even though she did not win the award, she was satified.

Even though she did win the award, she was not satified.

In RNN/LSTM we feed 1-by-1,
no need for pos. encoding.



In transformers,
we feed altogether



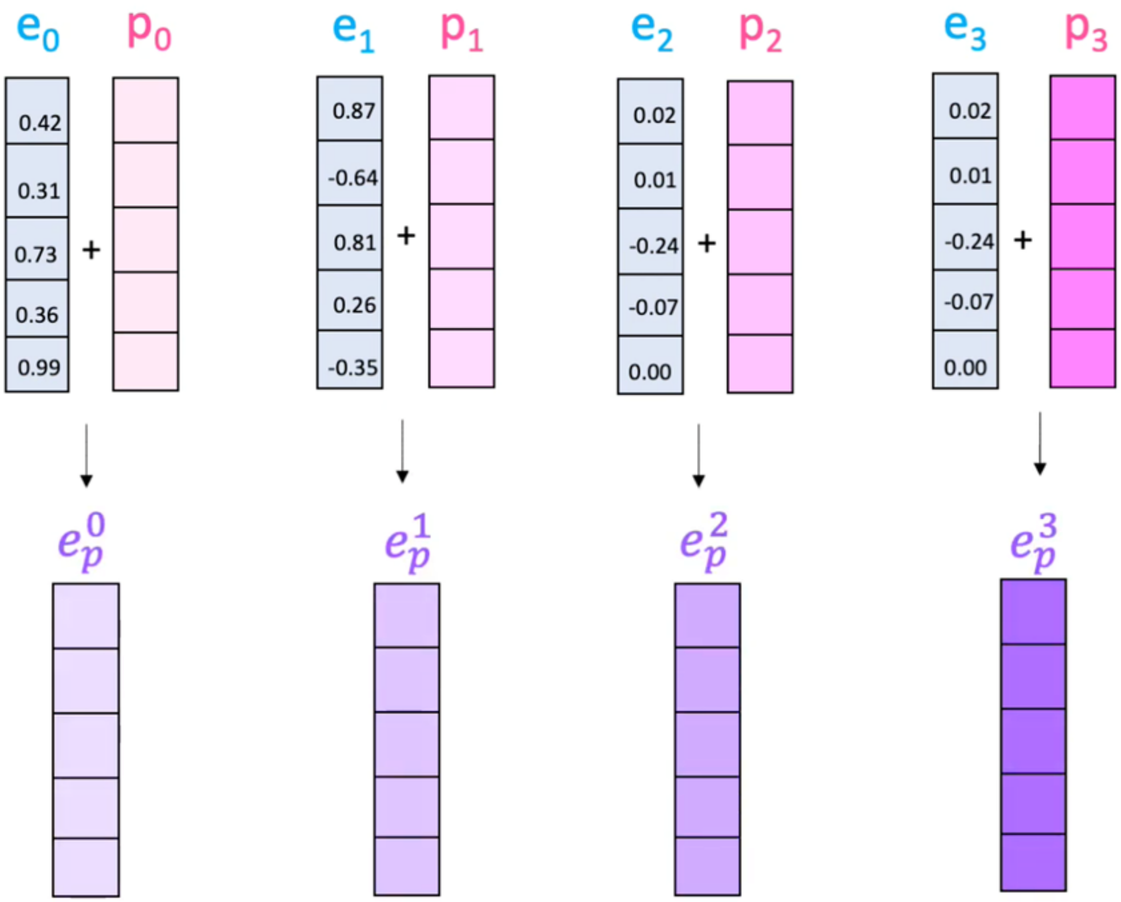
Transformers: Positional Encoding

What are these position embeddings?

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

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

Encoding proposed in the original paper, Vaswani et al. 2017.

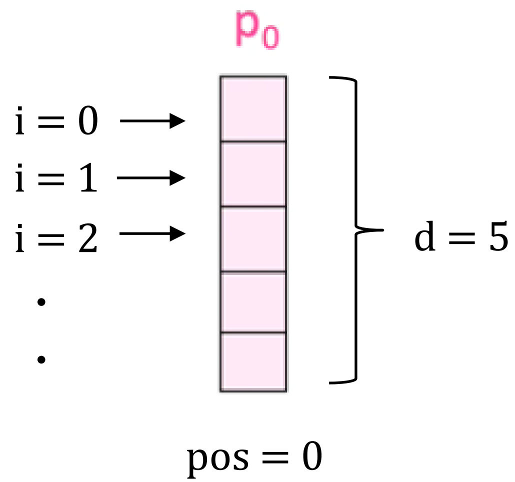


Transformers: Positional Encoding

What are these position embeddings?

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

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

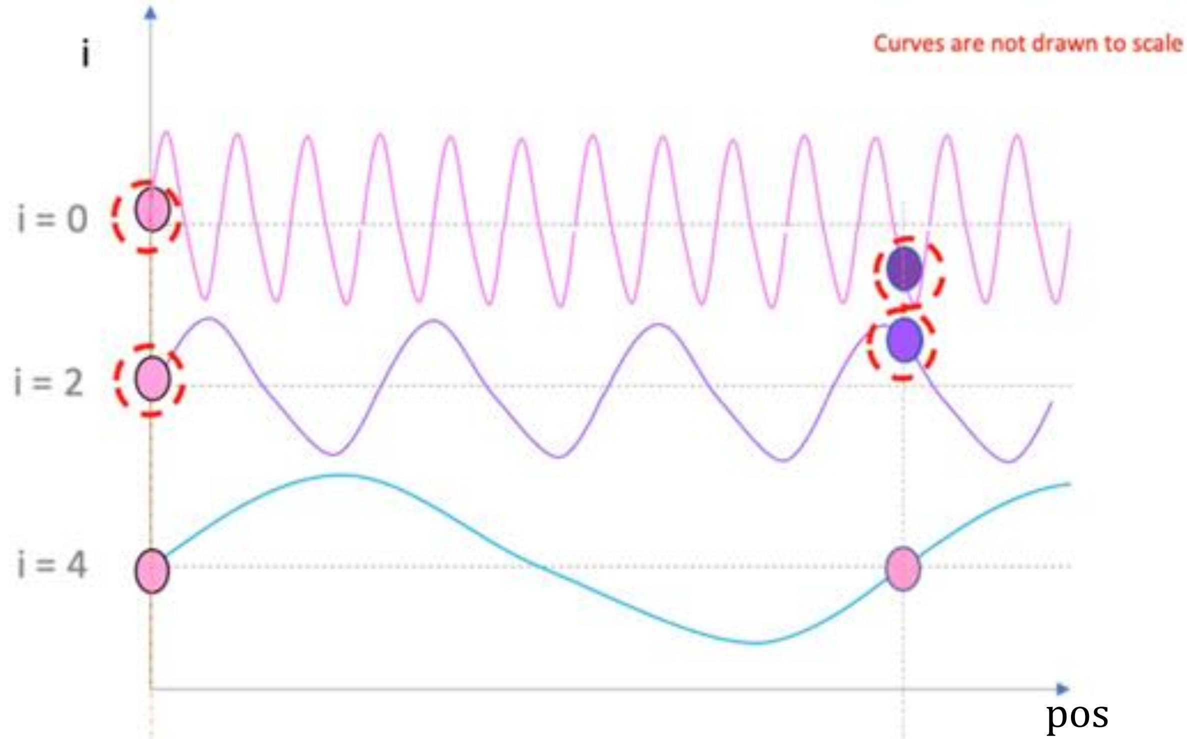


Transformers: Positional Encoding

What are these position embeddings?

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

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

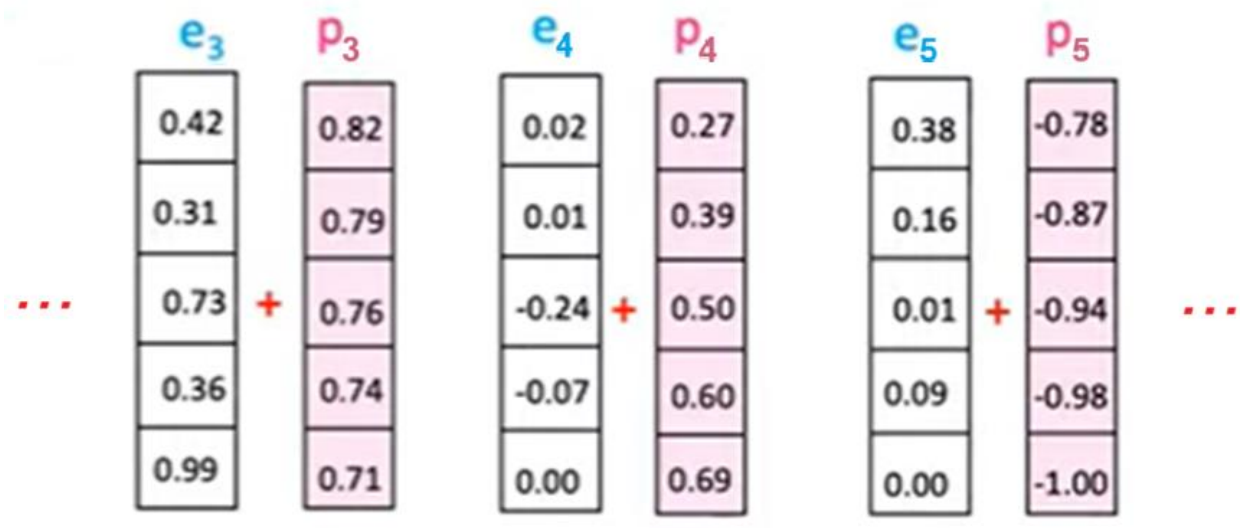


Transformers: Positional Encoding

What are these position embeddings?

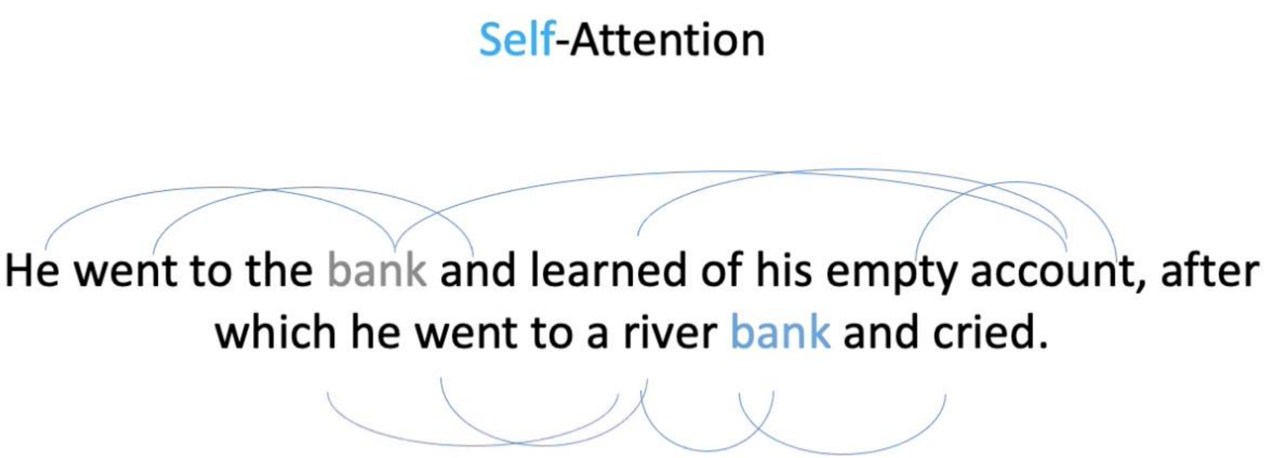
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

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



Encoder

Self-attention:



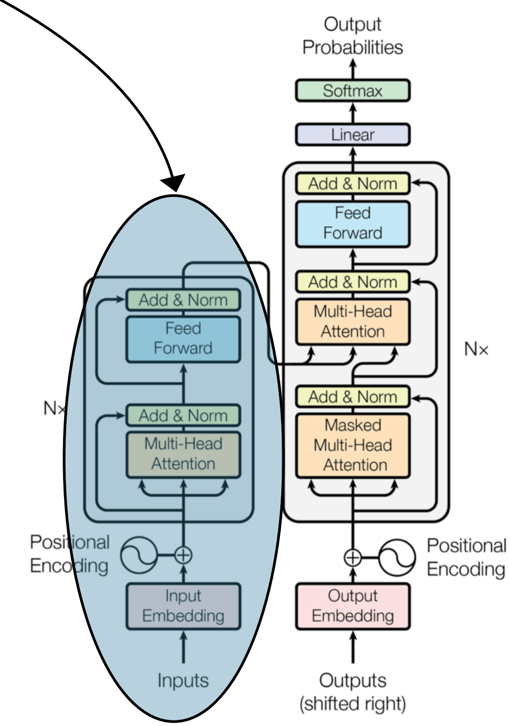
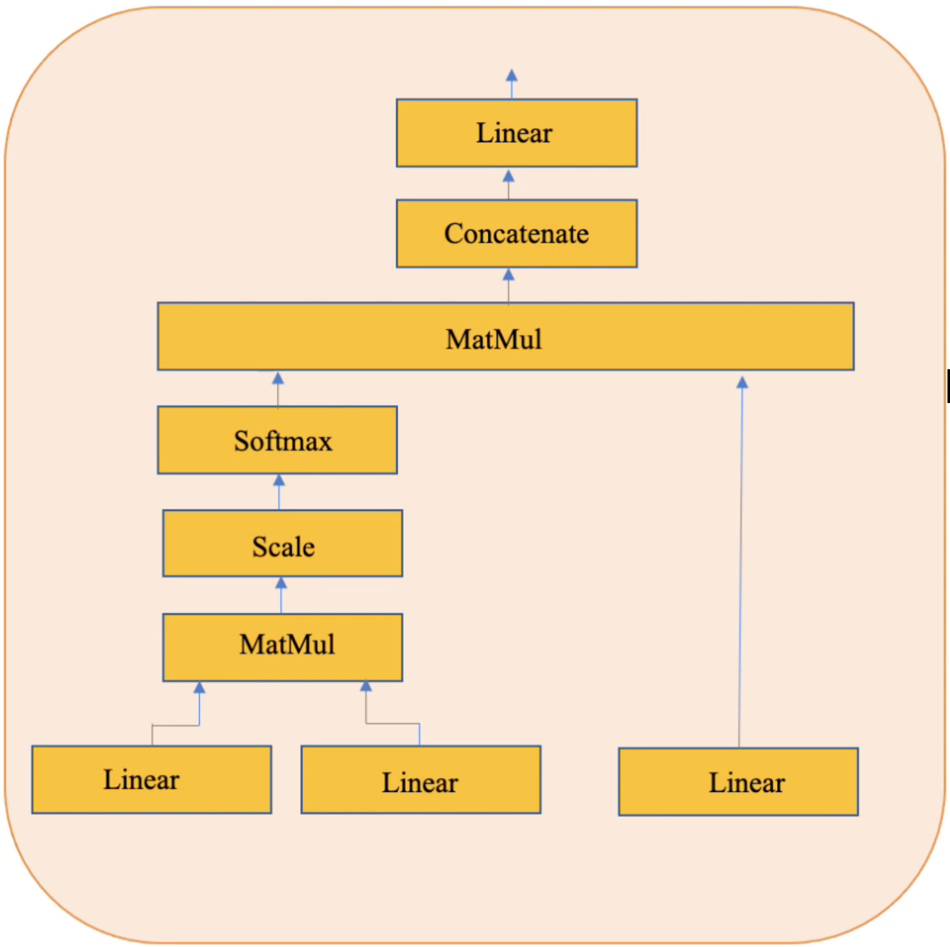


Figure 1: The Transformer - model architecture.

Encoder

Multi-head attention:



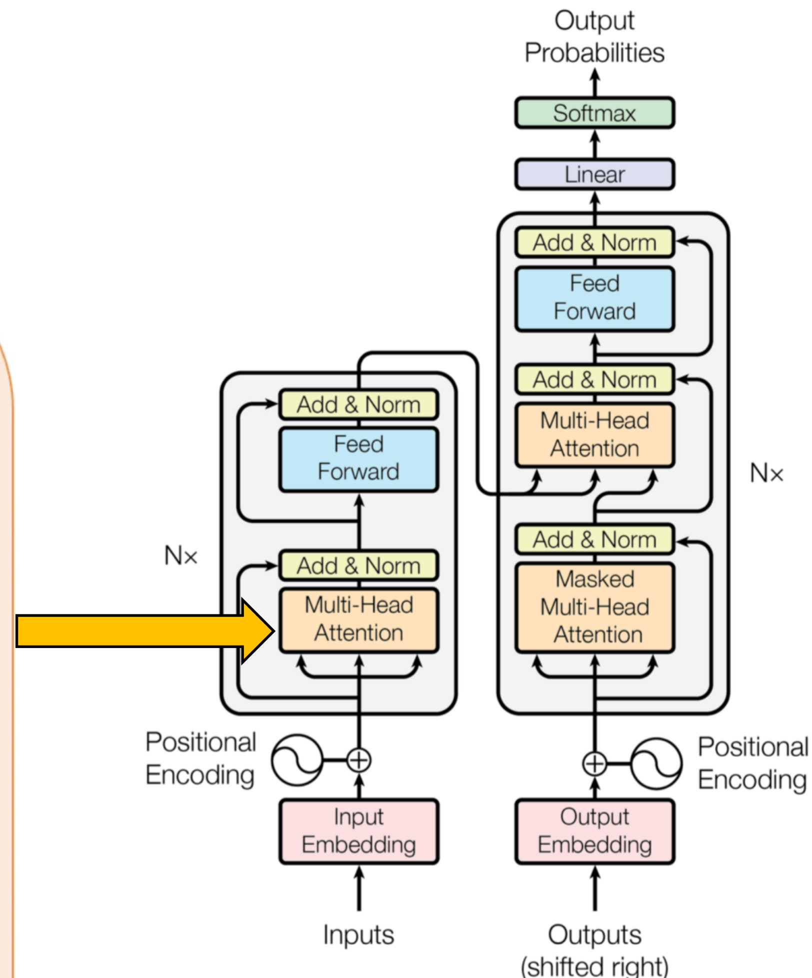
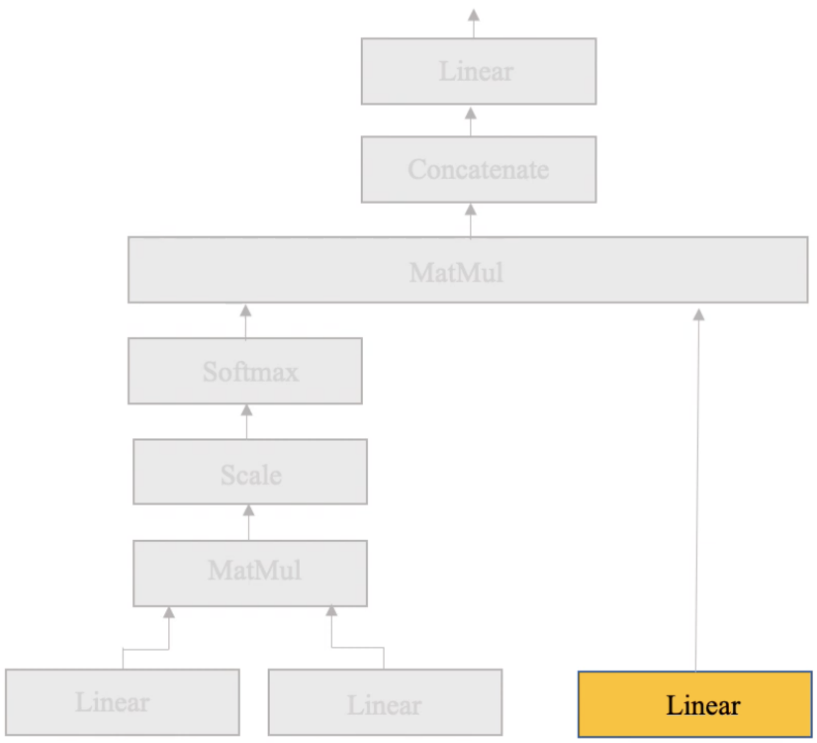


Figure 1: The Transformer - model architecture.

Encoder

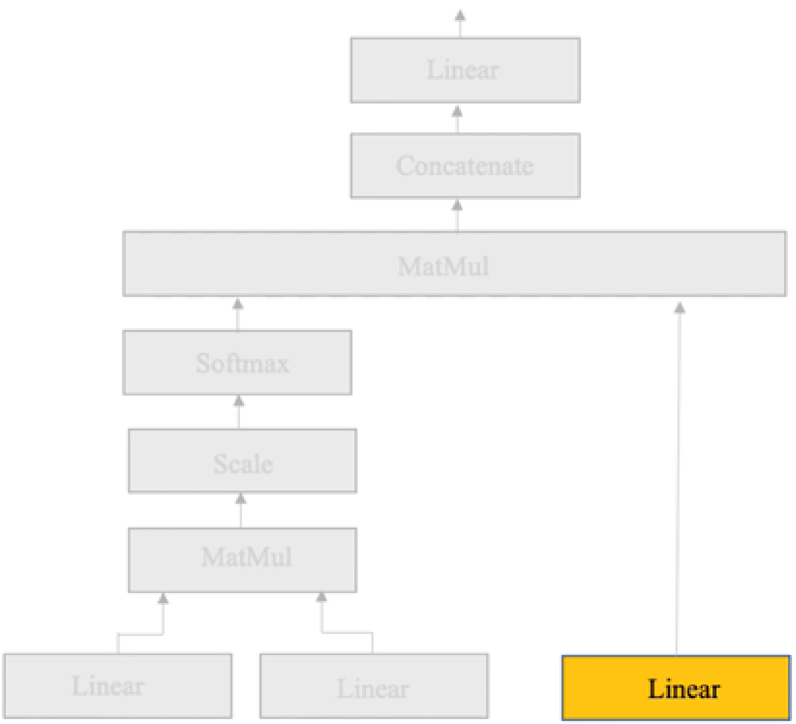


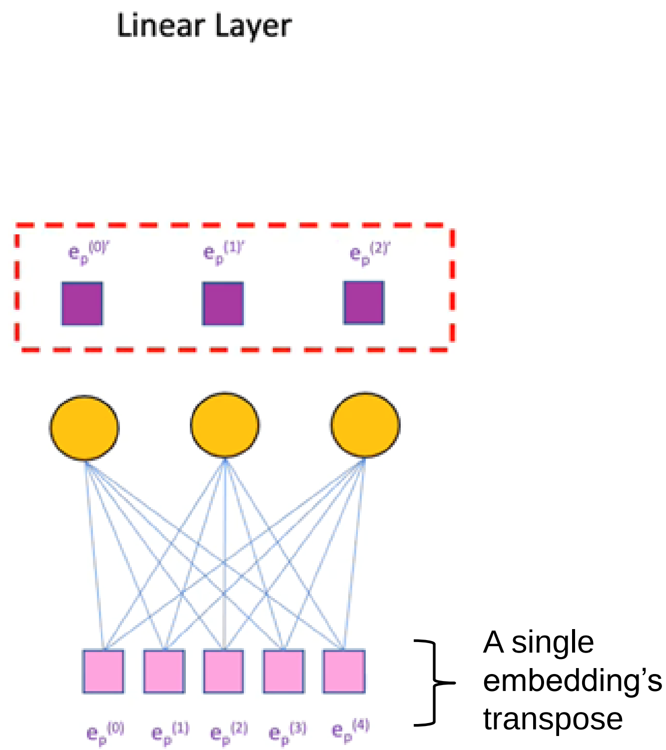
Linear Layer

i) Mapping inputs onto the outputs

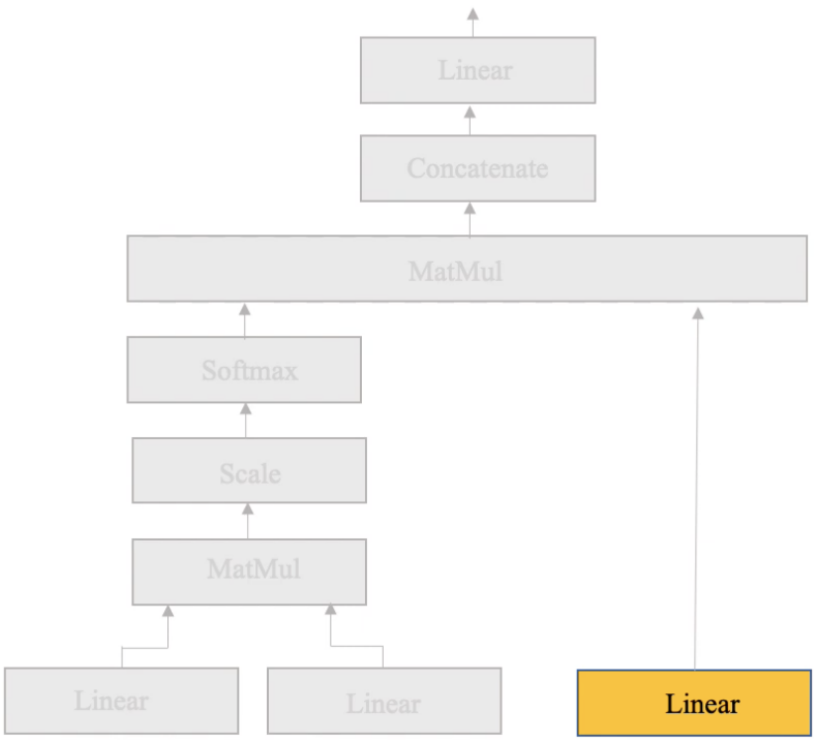
ii) Changing matrix/vector dimensions

Encoder

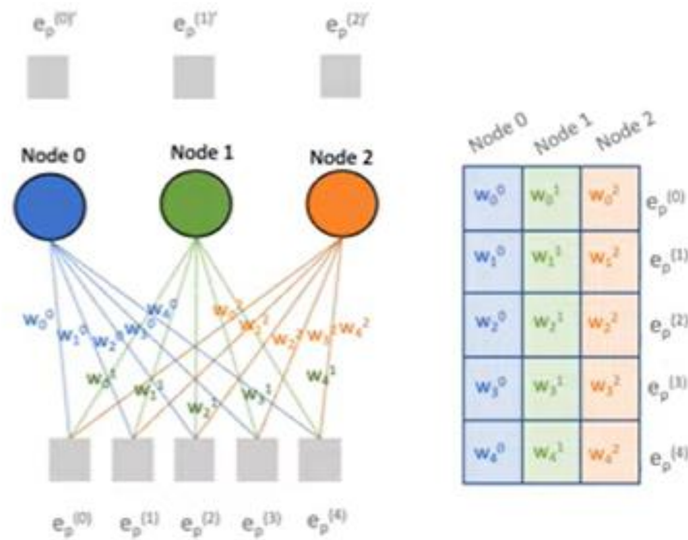

Linear
Concatenate
MatMul
Softmax
Scale
MatMul
Linear
Linear
Linear


Linear Layer
$e_p^{(0)'}$
$e_p^{(1)'}$
$e_p^{(2)'}$
$e_p^{(0)}$ $e_p^{(1)}$ $e_p^{(2)}$ $e_p^{(3)}$ $e_p^{(4)}$
A single embedding's transpose

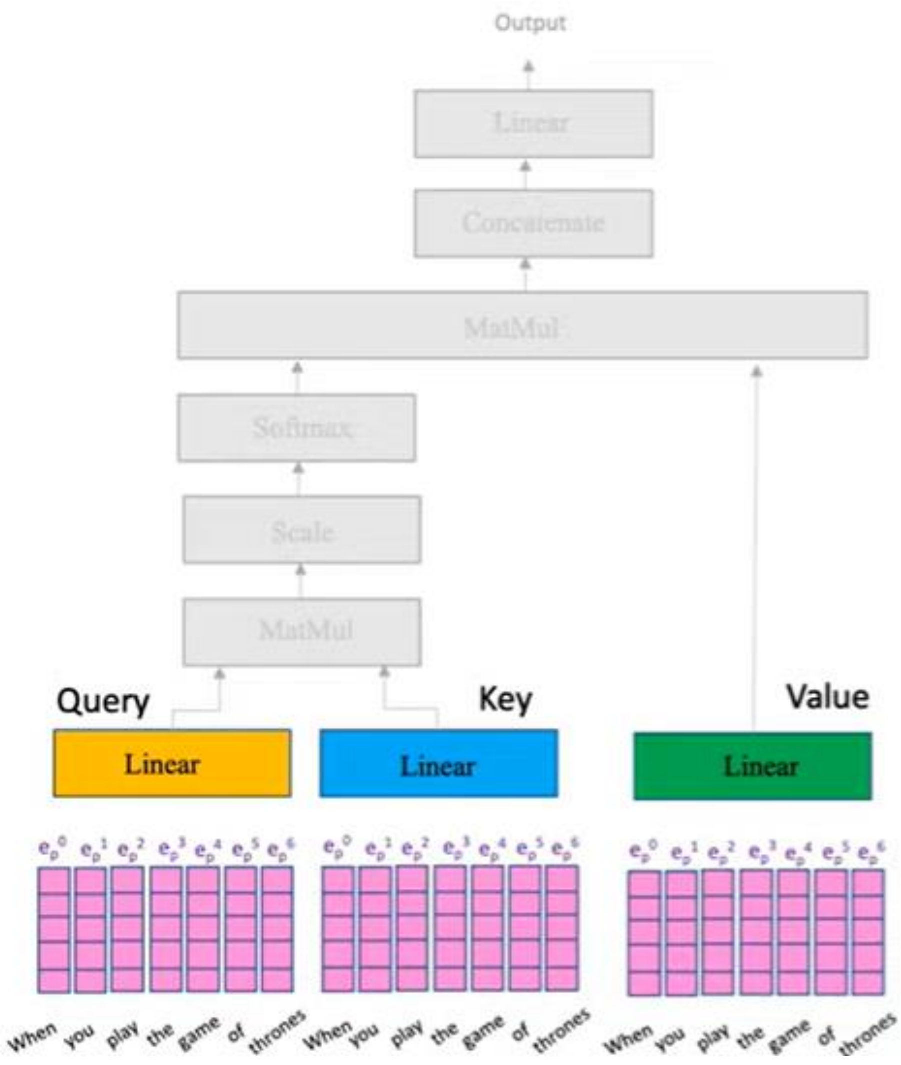
Encoder



Linear Layer



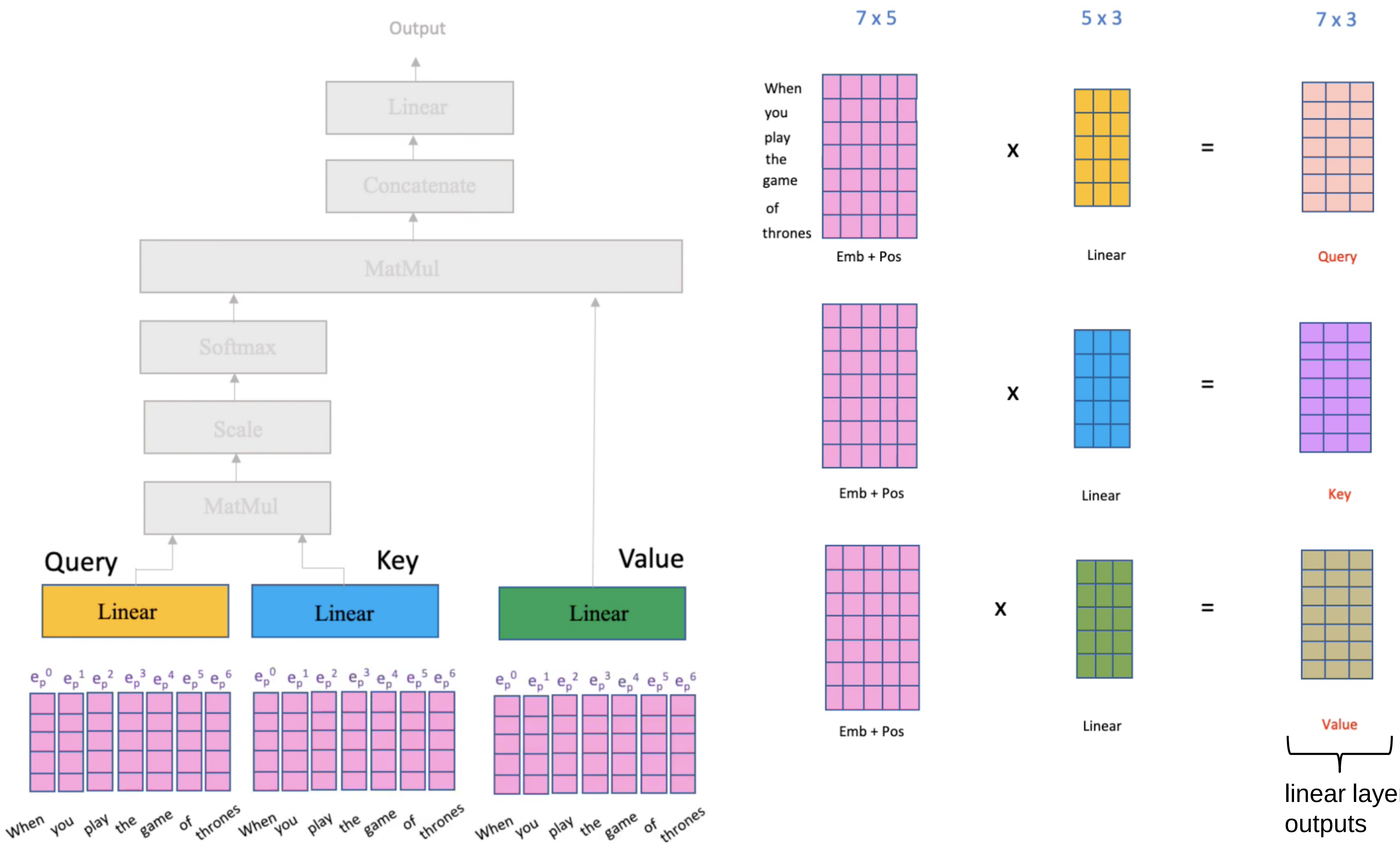
Encoder



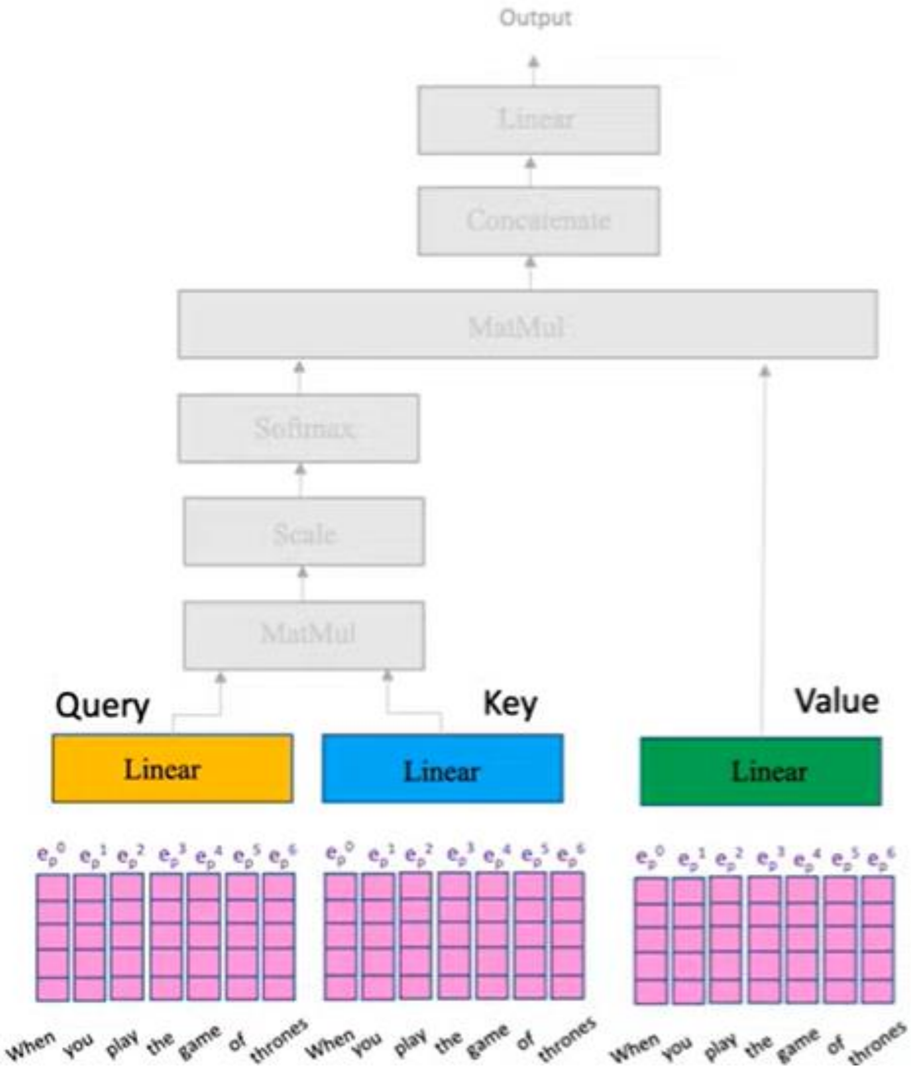
What do we feed into **Query**, **Key** and **Value**?

The same embeddings, because it's self-attention! (stay tuned)

Encoder


Output
Linear
Concatenate
MatMul
Softmax
Scale
MatMul
Query
Key
Value
Linear
Linear
Linear
When you play the game of thrones
7 x 5
5 x 3
7 x 3
When
you
play
the
game
of
thrones
x
=
Emb + Pos
Linear
Query
Key
Value
linear layer outputs

Encoder

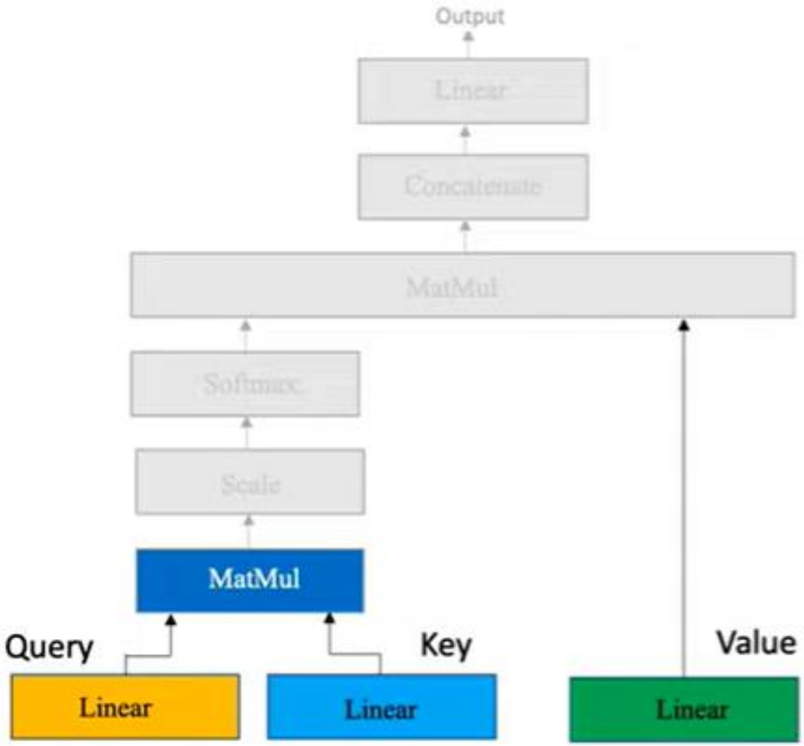


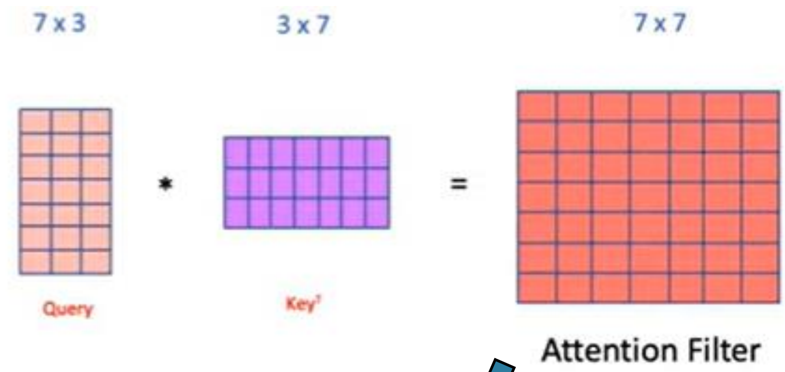
We want **Query** and **Key** (linear layer outputs) to be similar.

We measure that with cosine similarity:

$$sim(Q, K) = \frac{Q \cdot K^T}{scaling}$$

Encoder

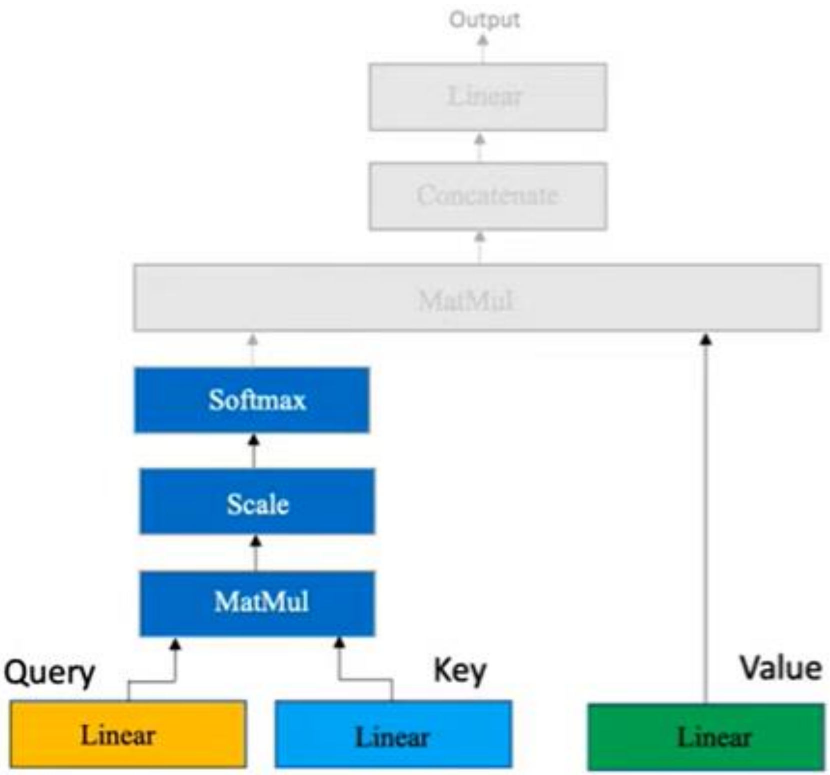




	When	you	play	the	game	of	thrones
When	89	20	41	10	55	78	59
you	90	98	81	22	87	15	32
play	29	81	95	10	90	30	92
the	10	22	67	12	88	40	89
game	22	70	90	56	98	44	80
of	10	15	30	40	44	44	59
thrones	59	72	92	90	13	59	99

That is why it is self-attention

Encoder



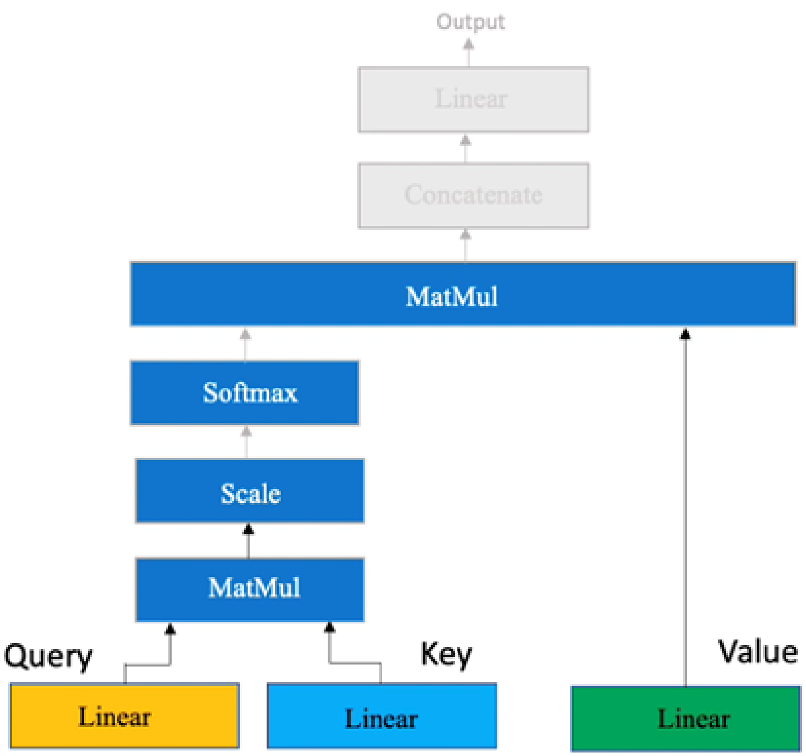
7 x 7

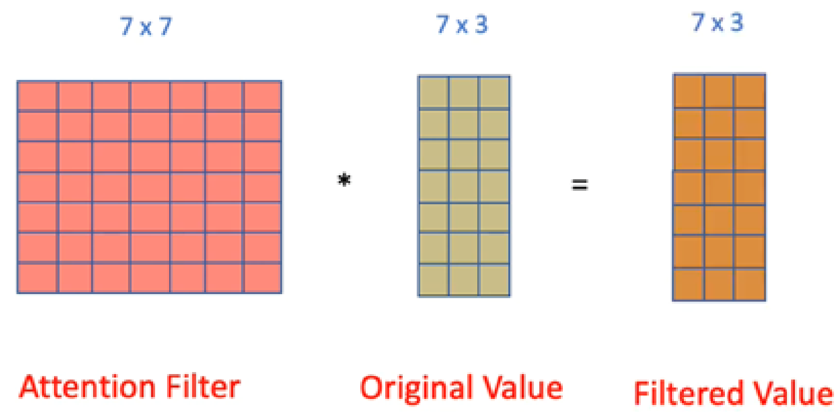
	When	you	play	the	game	of	thrones
When	1.00	0.00	0.00	0.00	0.00	0.00	0.00
you	0.00	0.97	0.03	0.00	0.00	0.00	0.00
play	0.00	0.00	0.68	0.00	0.10	0.00	0.22
the	0.00	0.00	0.00	0.65	0.14	0.00	0.21
game	0.00	0.00	0.03	0.02	0.72	0.00	0.23
of	0.00	0.00	0.00	0.00	0.00	1.00	0.00
thrones	0.00	0.00	0.05	0.03	0.17	0.00	0.75

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_i)}$$

$$\text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right)$$

Encoder





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

Encoder

Multiplying attention filter with the original input gives something like a version of the input where important parts are emphasized.

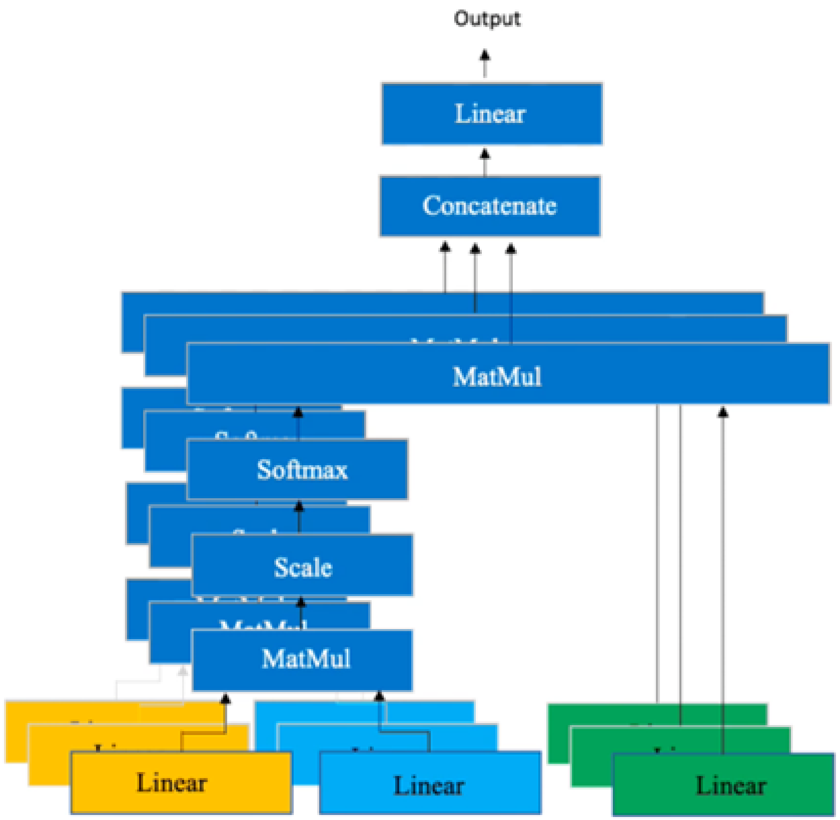
Q: What is at the left of the table in the image?
A: Chair

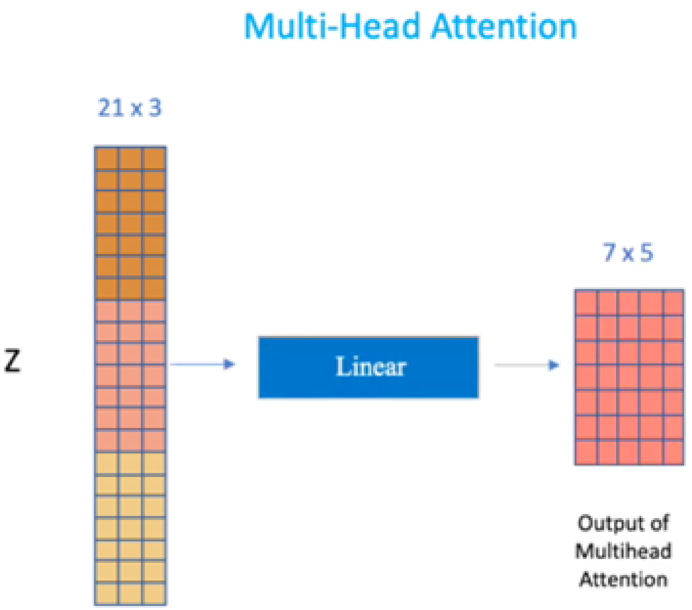
Q: What color are the bed sheets in the image?
A: Red

Q: What is the color of the cake?
A: White

Q: What are standing next to the flower pot?
A: Dogs

What is Multi-head Attention?





Multiple attention filters (different sets of trained weights) focus on different linguistics phenomena.

Decoder

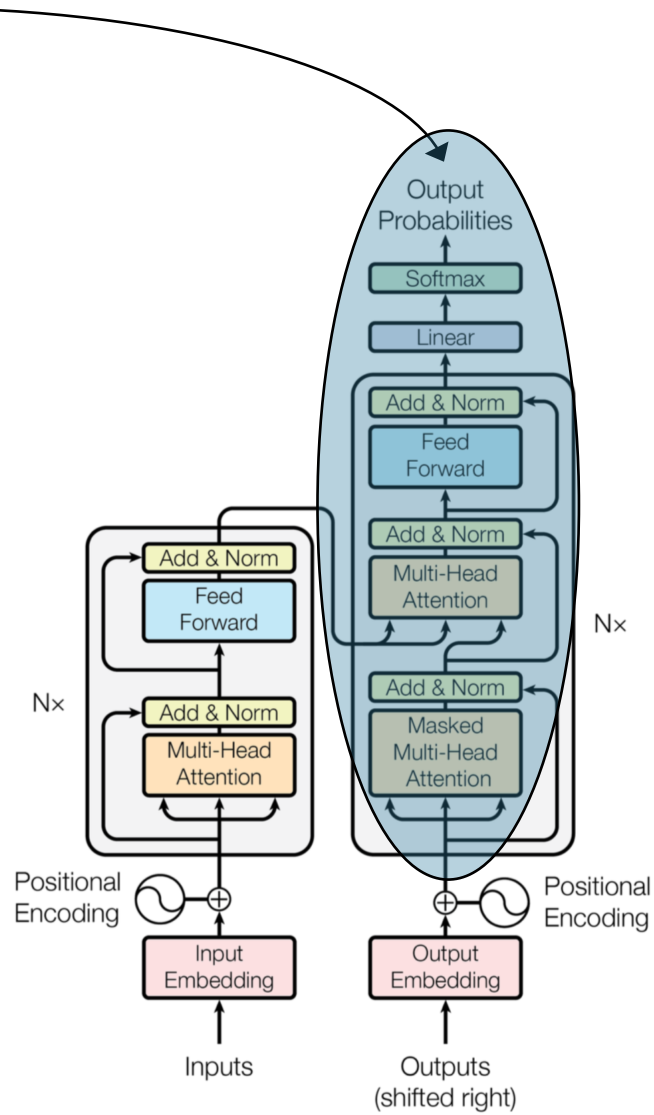
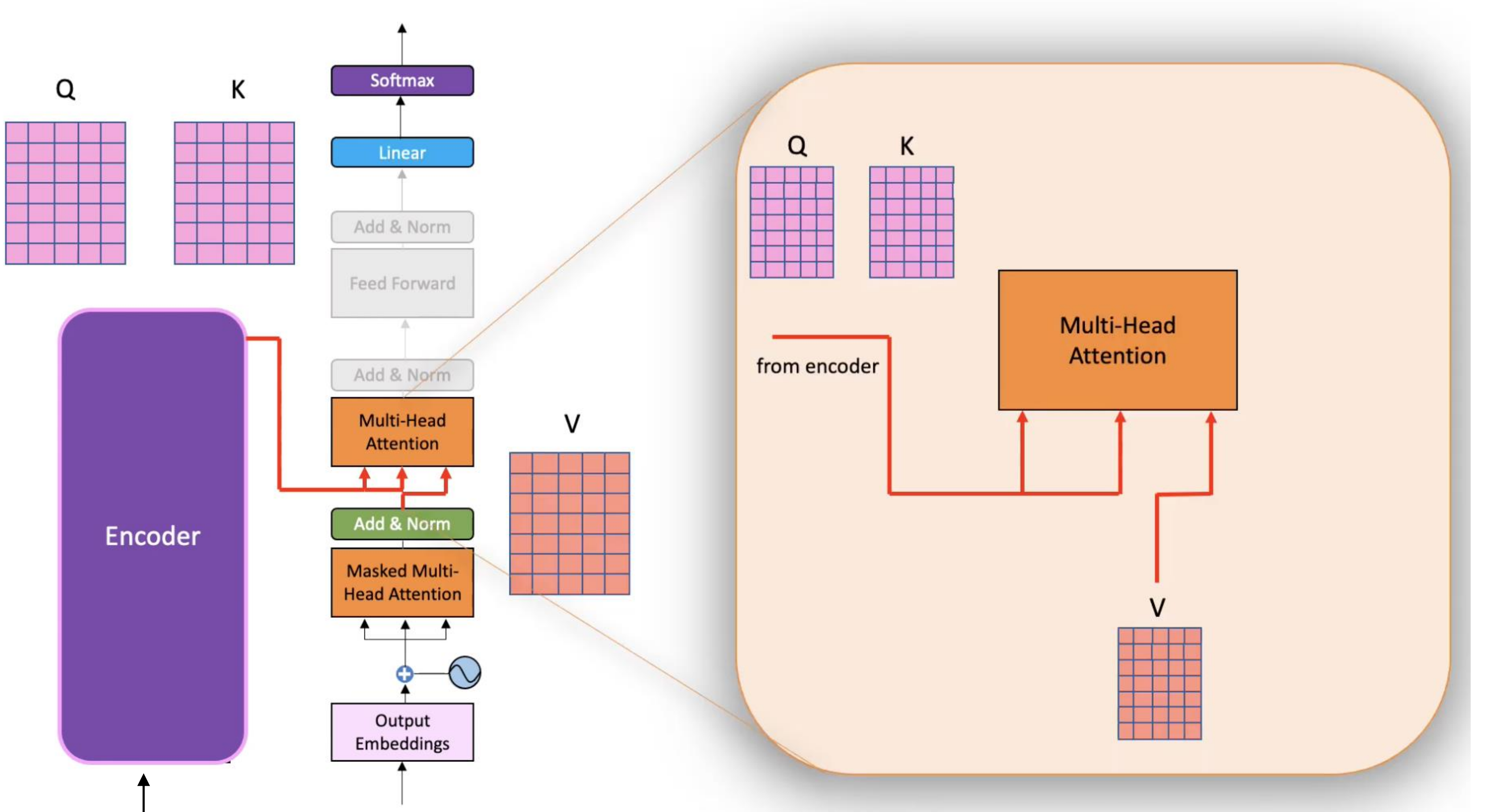


Figure 1: The Transformer - model architecture.

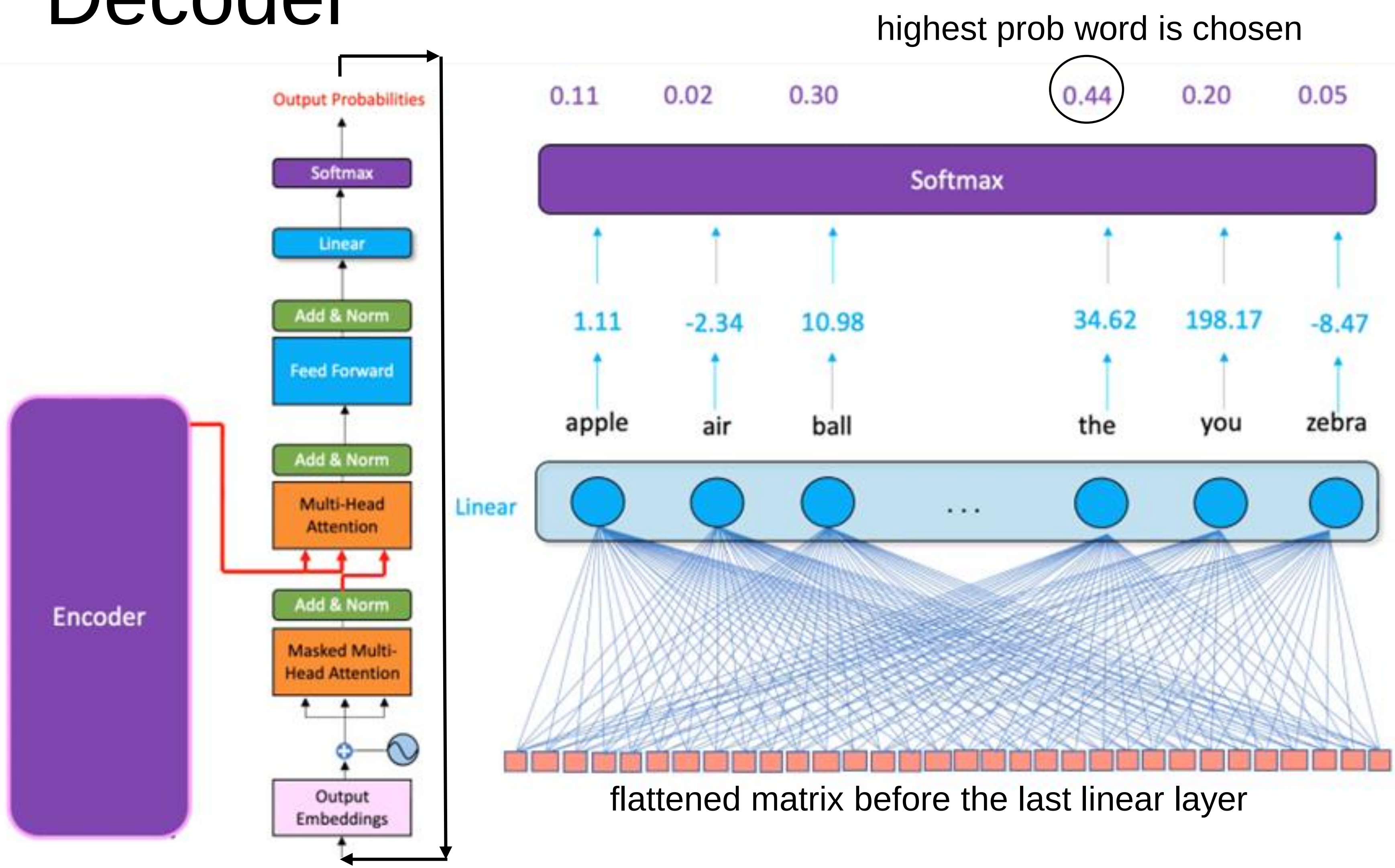
Decoder



Input text

Output text (so far)

Decoder



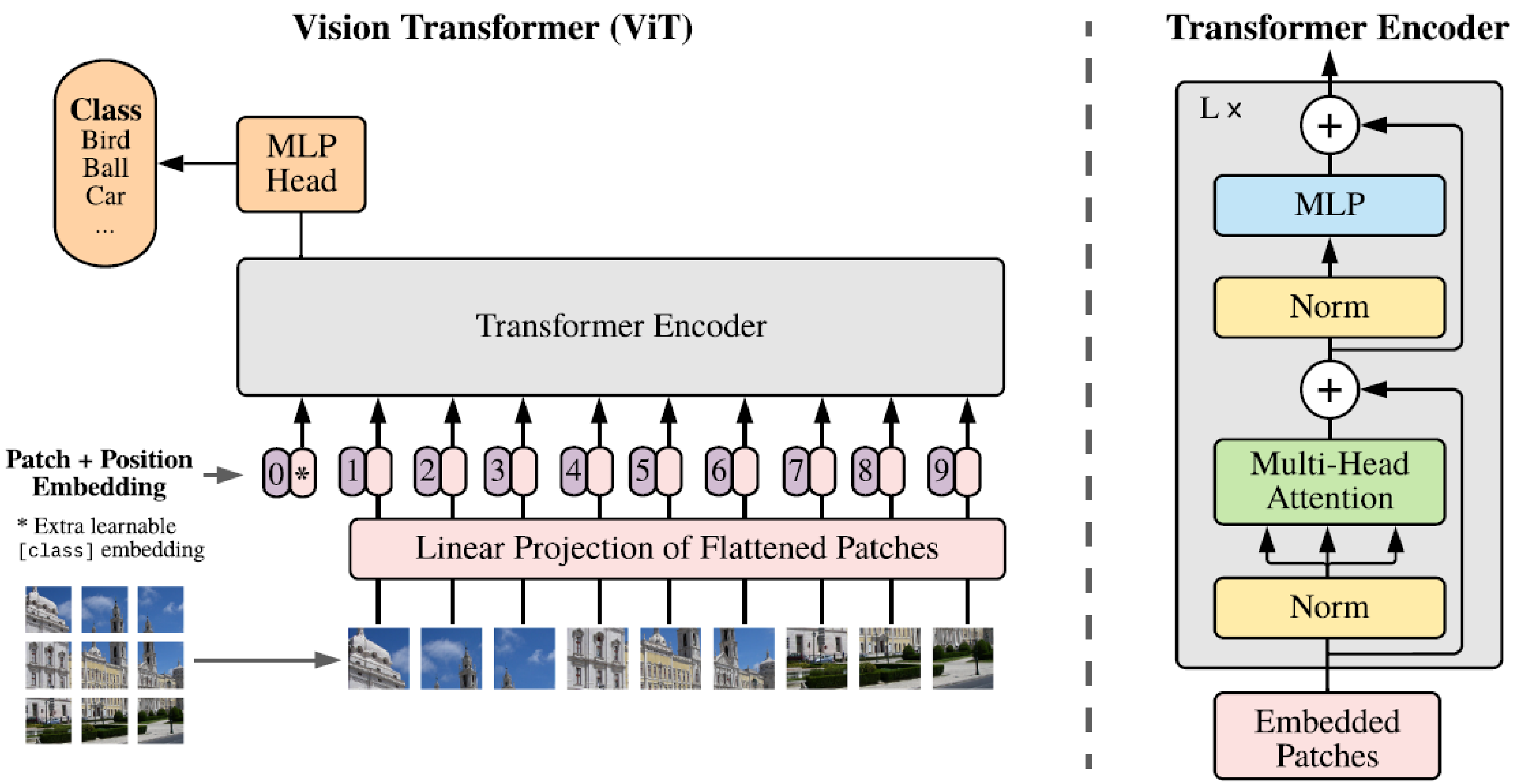
First input to the decoder is usually <start> token.

Implementation

Transformer Layers

`nn.Transformer`	A transformer model.
`nn.TransformerEncoder`	TransformerEncoder is a stack of N encoder layers
`nn.TransformerDecoder`	TransformerDecoder is a stack of N decoder layers
`nn.TransformerEncoderLayer`	TransformerEncoderLayer is made up of self-attn and feedforward network.
`nn.TransformerDecoderLayer`	TransformerDecoderLayer is made up of self-attn, multi-head-attn and feedforward network.

Vision Transformer* (ViT)



* Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, 2020, arXiv:2010.11929v1

So far..

Natural Language Processing
- BERT
- XLNet
- ALBERT
- GPT, GPT-2, GPT-3, GPT-4
- …

Computer Vision
- ViT (Vision Transformer)
- DeiT, BEiT, EsViT
- DETR
- Image GPT
- …

Tasks
- Question answering
- Sentence completion
- Translation
- Dialogue generation
- Summarization
- …

Source1: https://www.topbots.com/leading-nlp-language-models-2020/
Source2: https://huggingface.co/docs/transformers/model_doc/vit