# Transformers

Lara J. Martin (she/they)

https://laramartin.net/interactive-fiction-class

*Slides modified from Dr. Daphne Ippolito*

# Learning Objectives

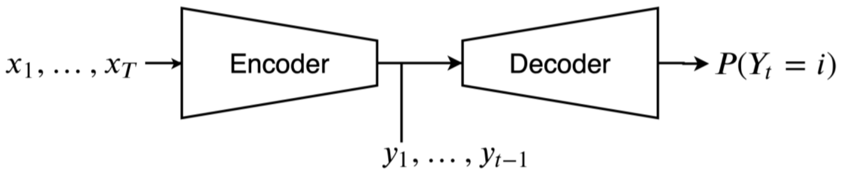
Intuit what query, key, and value components are in the transformer algorithm

Distinguish encoder-decoder attention from self-attention

Investigate what information self-attention might capture

Compare sequence-to-sequence RNNs to transformers

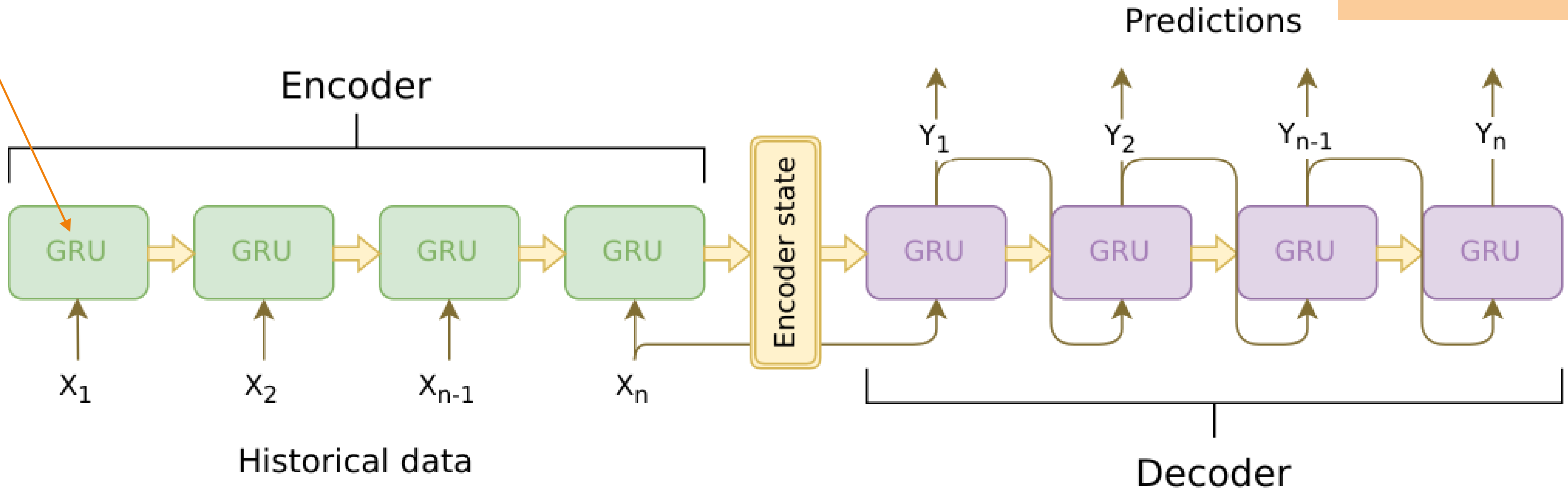
# What is a language model?




# Review:
# Sequence-to-Sequence / Encoder-Decoder Models

What are the inputs & outputs?

Can be LSTM



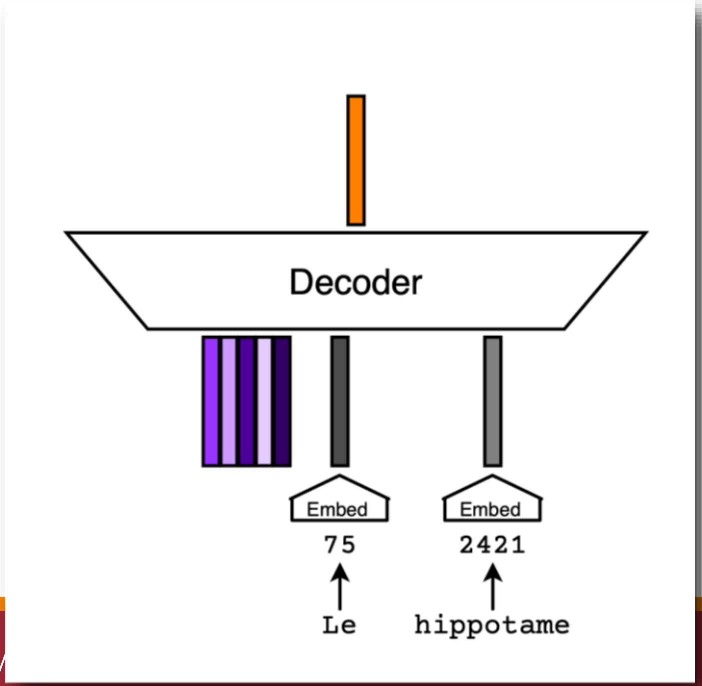

https://jeddy92.github.io/JEddy92.github.io/ts_seq2seq_intro/

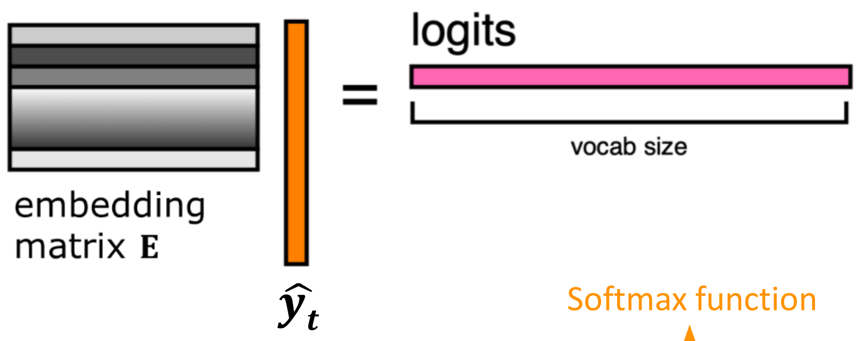
I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Conference on Advances in Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2014, pp. 3104–3112. https://proceedings.neurips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

# Review: Turning $\widehat{yt}$ into a Probability Distribution

We can multiply the predicted embedding $\widehat{yt}$ by our vocabulary embedding matric to get a score for each vocabulary word. These scores are referred to as logits.

The softmax function then lets us turn the logits into probabilities.



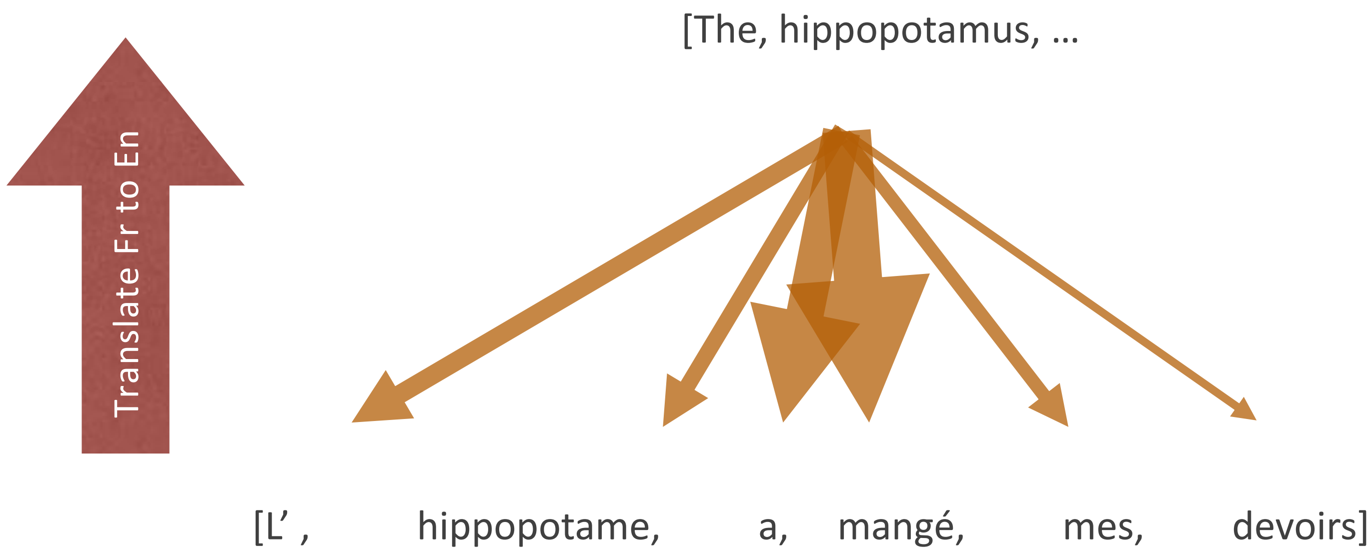




Softmax function

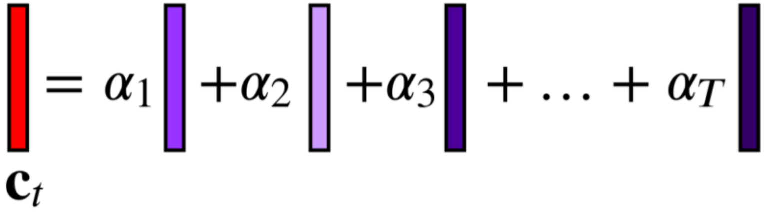
$$P(Y_t = i | \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \boxed{\frac{\exp(\mathbf{E}\hat{\boldsymbol{y}}_{t[i]})}{\sum_j \exp(\mathbf{E}\hat{\boldsymbol{y}}_{t[i]})}}$$

# Review: Attention

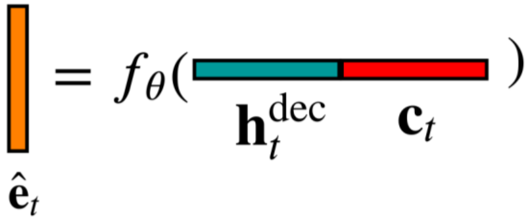
**Better approach:** an attention mechanism



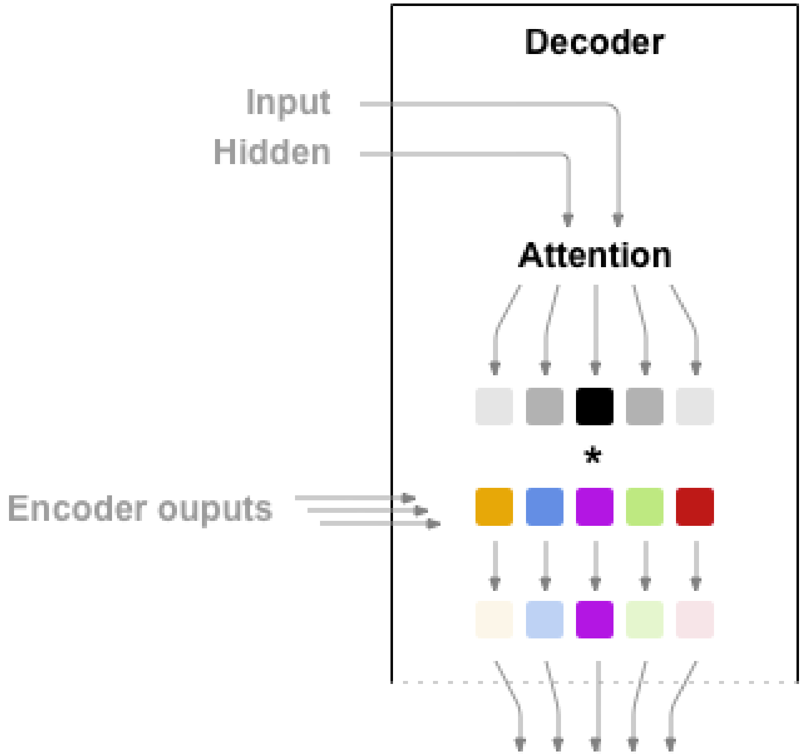

Compute a linear combination of the encoder hidden states.




Decoder's prediction at position *t* is based on both the context vector and the hidden state outputted by the RNN at that position.



# Review: Attention Decoder




https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

# Review: What are some of the limitations of RNNs?

# Transformers

Since 2018, the field has rapidly standardized on the Transformer architecture

## Attention Is All You Need

**Ashish Vaswani*** Google Brain avaswani@google.com
**Noam Shazeer*** Google Brain noam@google.com
**Niki Parmar*** Google Research nikip@google.com
**Jakob Uszkoreit*** Google Research usz@google.com
**Llion Jones*** Google Research llion@google.com
**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu
**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com
**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com

**Abstract**

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

# Neural Language Model Timeline

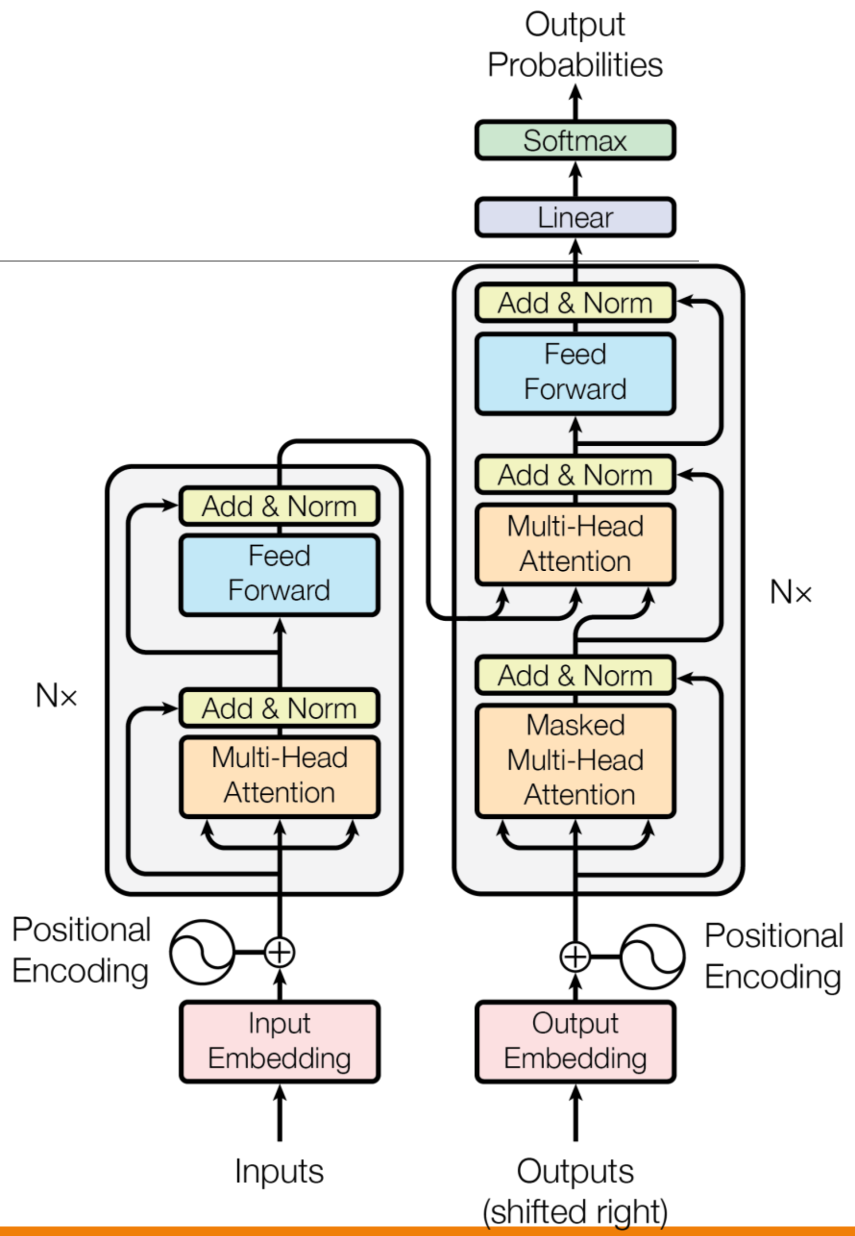
| | +attention -recurrence | | data++ parameters++ | data++ parameters++ | +RL | +??? |
|---|---|---|---|---|---|---|

| Seq2Seq RNN (2014) | Transformers (2017) | GPT-2 (2019) | GPT-3 (2020) | ChatGPT (InstructGPT) (2022) | GPT-4 (2023) |
|---|---|---|---|---|---|
| Sequence to Sequence Learning with Neural Networks<br>Ilya Sutskever, Oriol Vinyals, & Quoc V. Le | Attention Is All You Need<br>Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, & Illia Polosukhin | Language Models are Unsupervised Multitask Learners<br>Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, & Ilya Sutskever | Language Models are Few-Shot Learners<br>Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, et al. | Training language models to follow instructions with human feedback<br>Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, et al. | GPT-4 Technical Report<br>OpenAI |

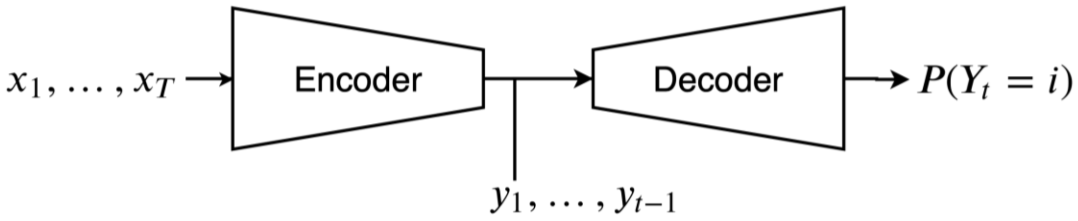
# Transformers

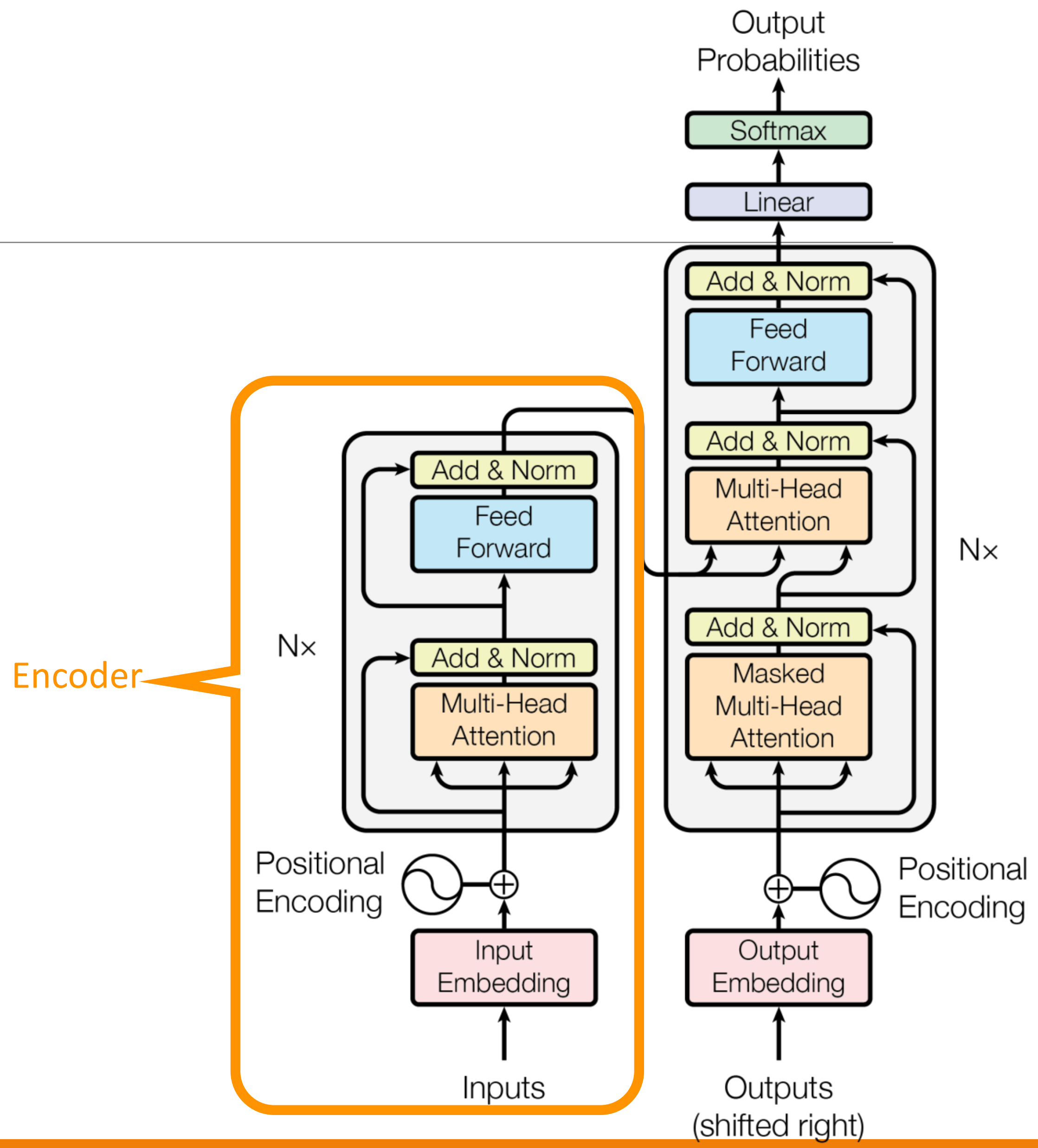
The Transformer is a **non-recurrent** non-convolutional (feed-forward) neural network designed for language understanding

- introduces self-attention in addition to encoder-decoder attention

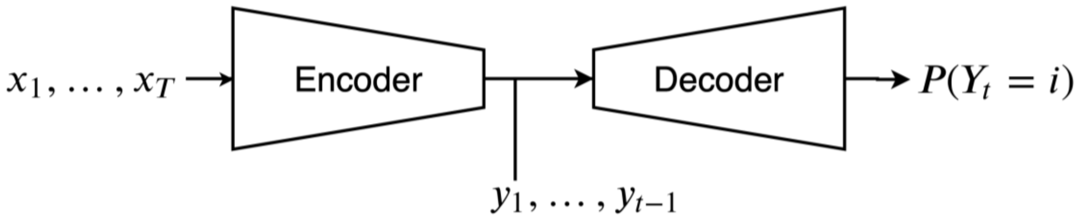


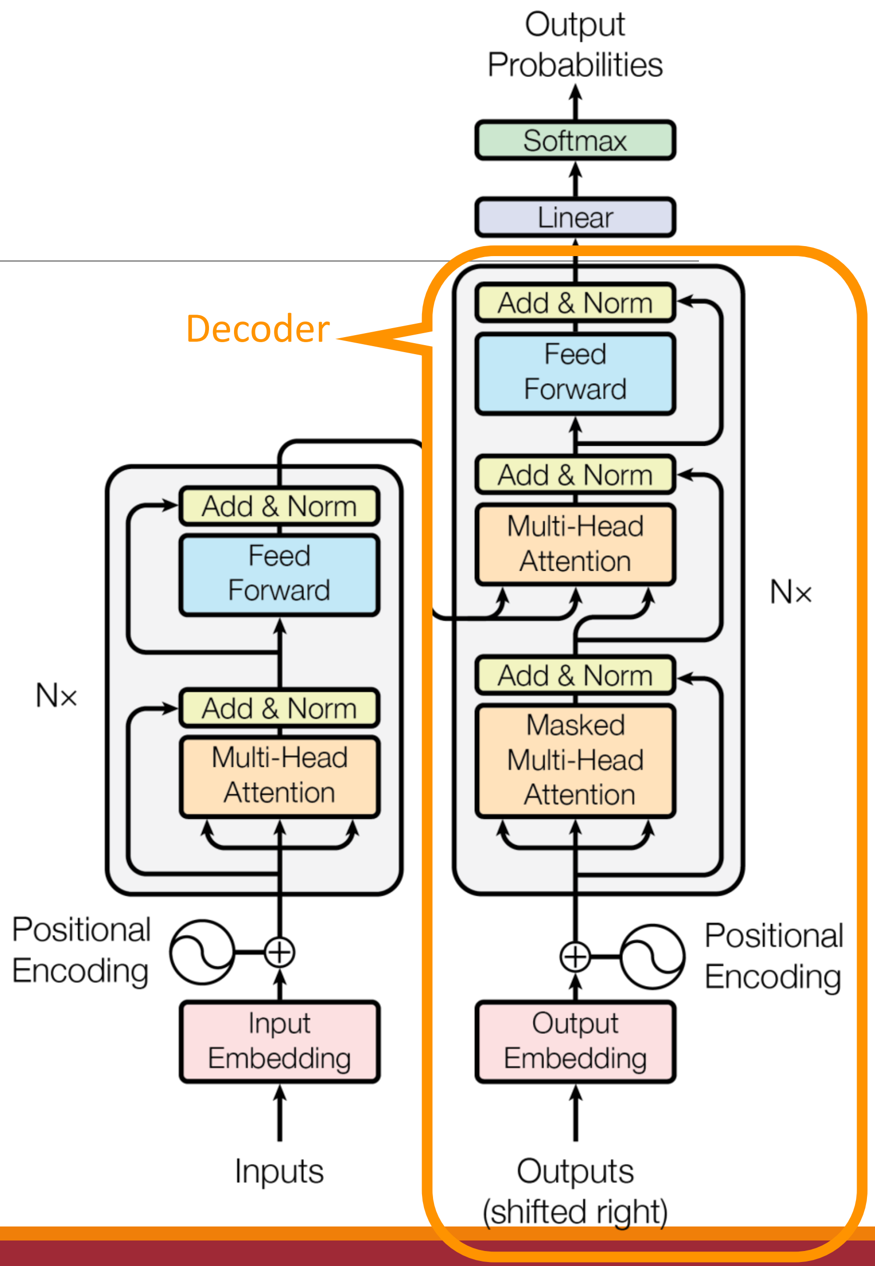
# Transformers



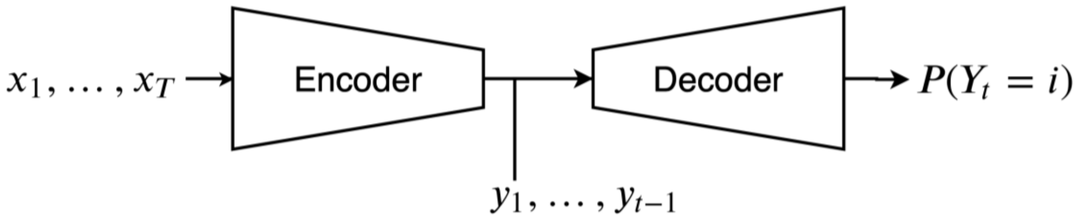



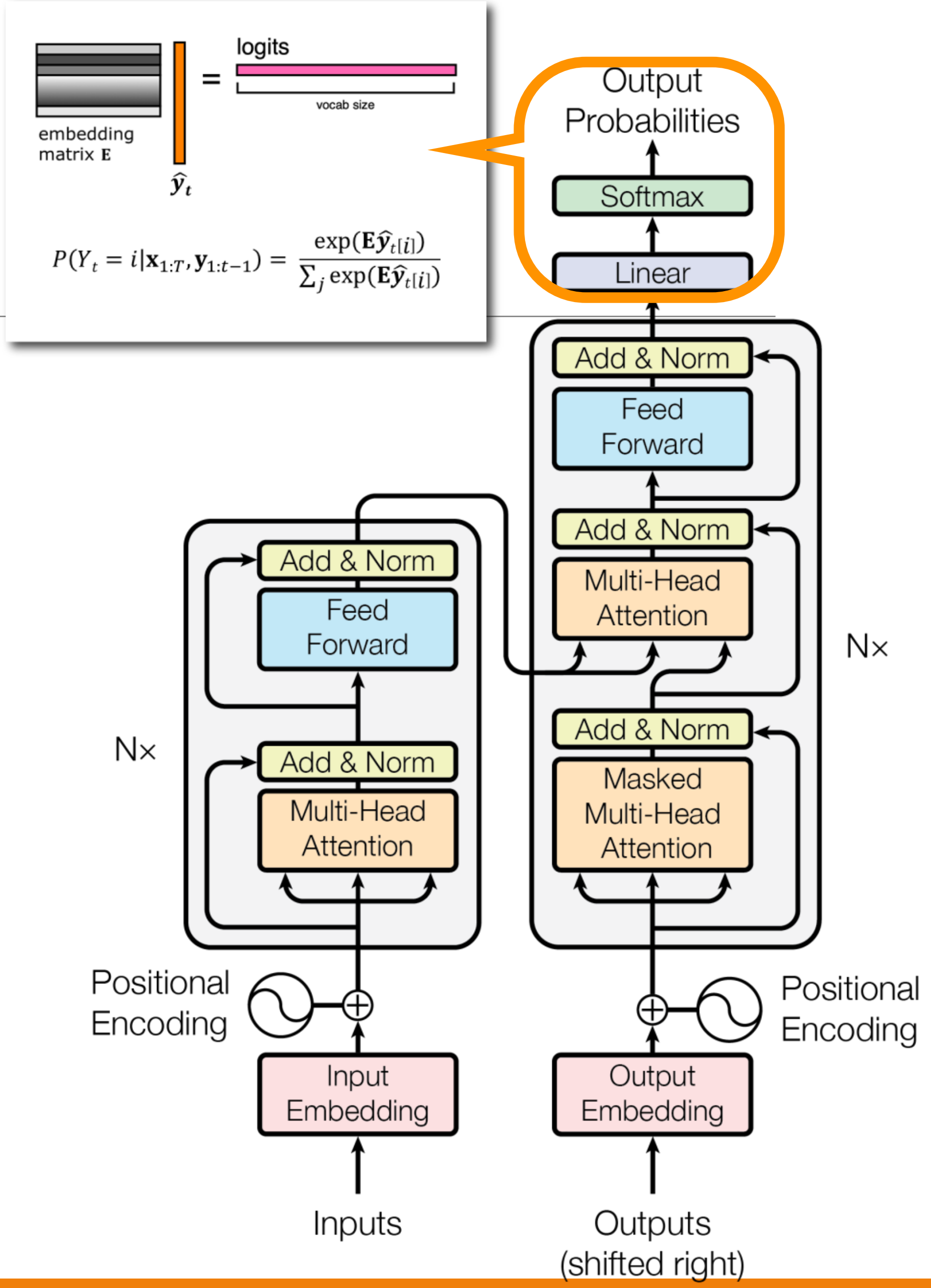
# Transformers



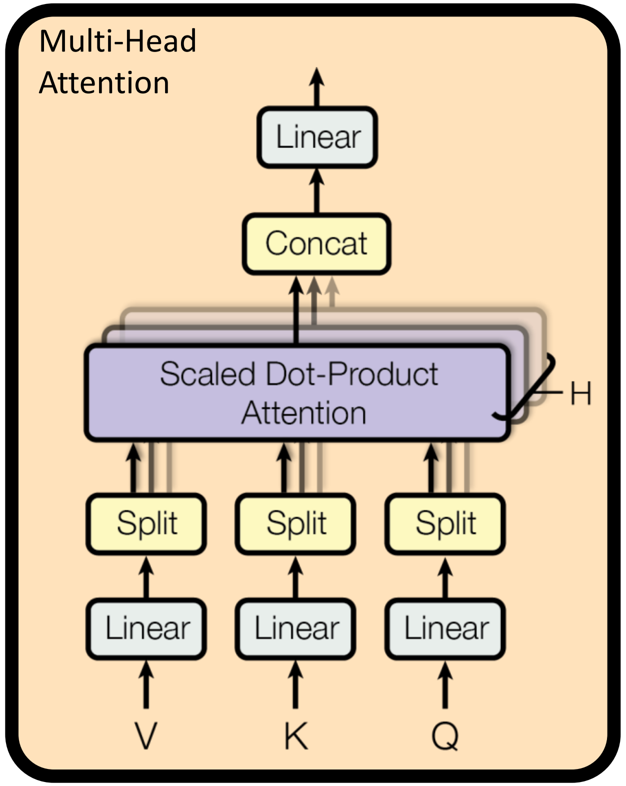



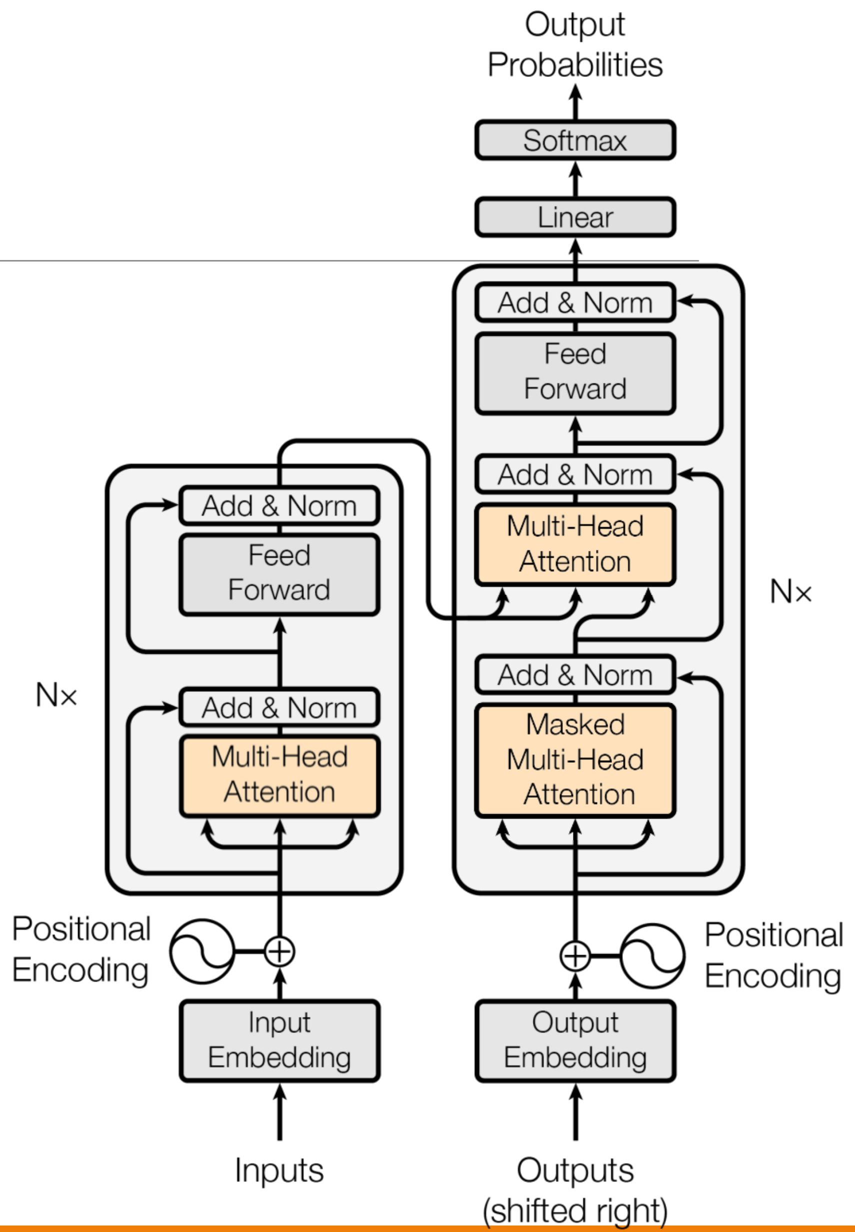
# Transformers



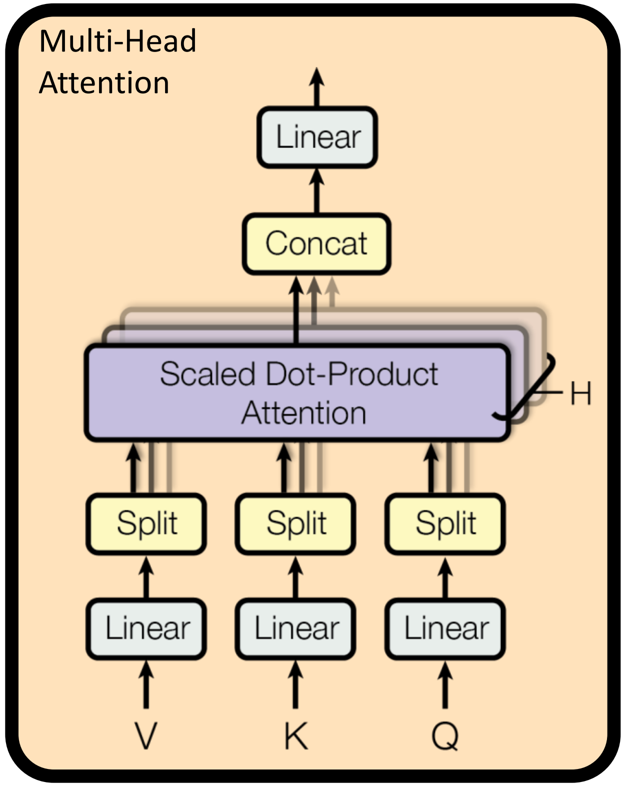



# Attention Mechanism

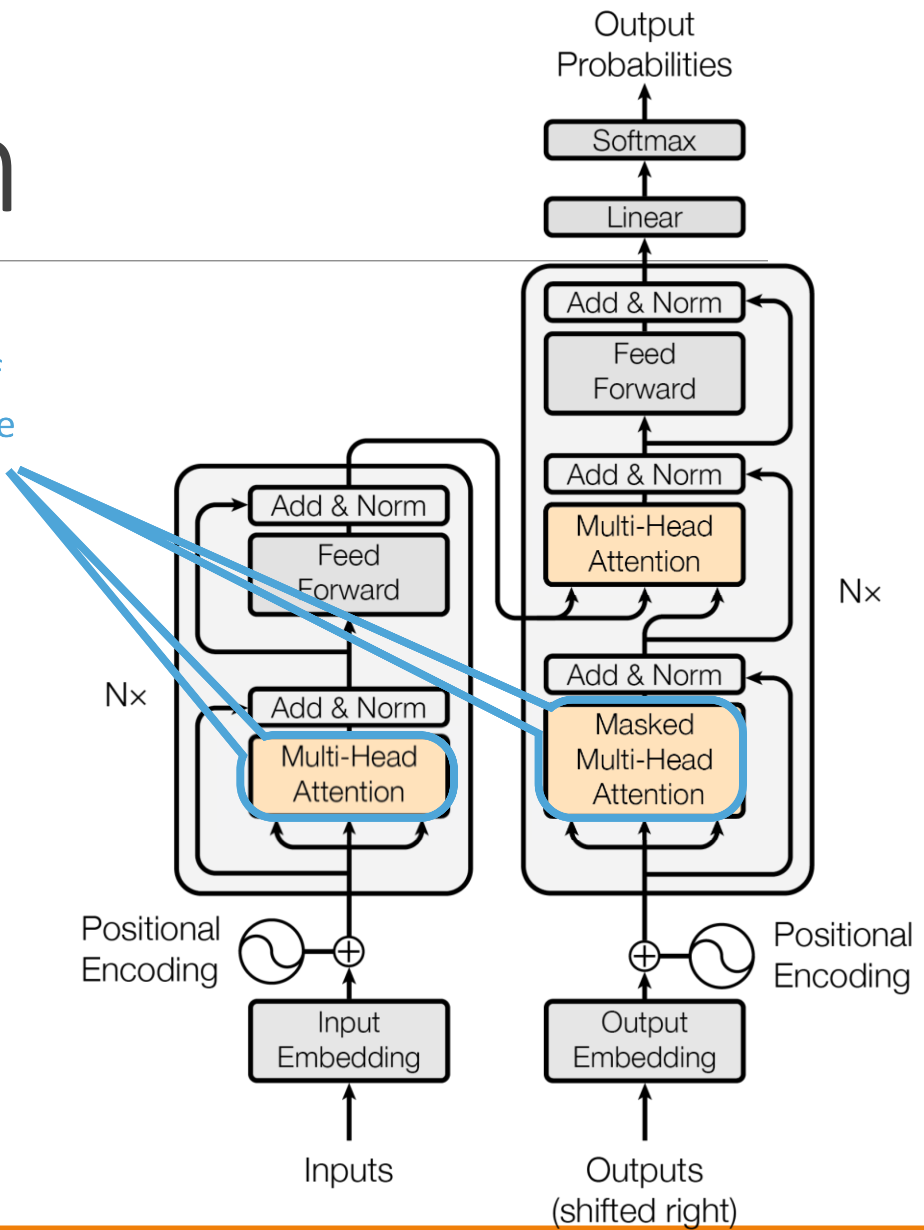


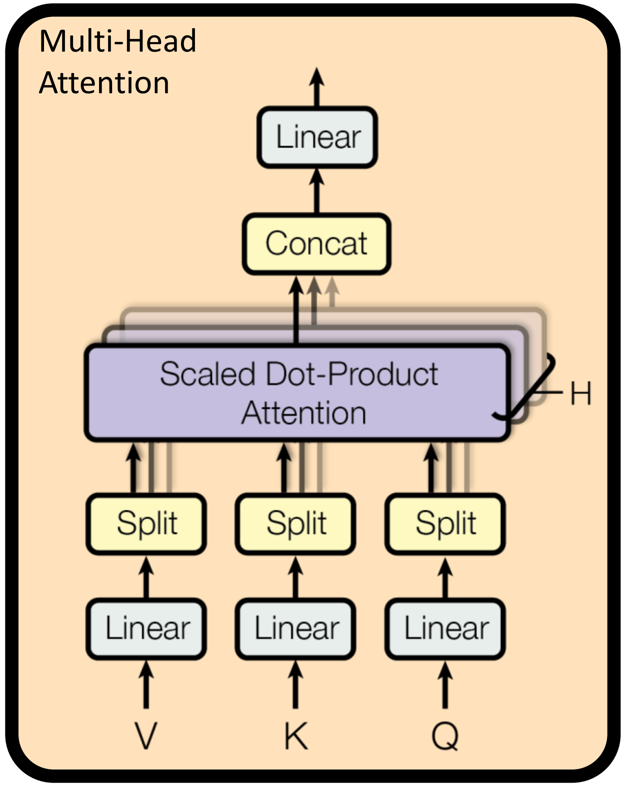
# Multi-Head Attention



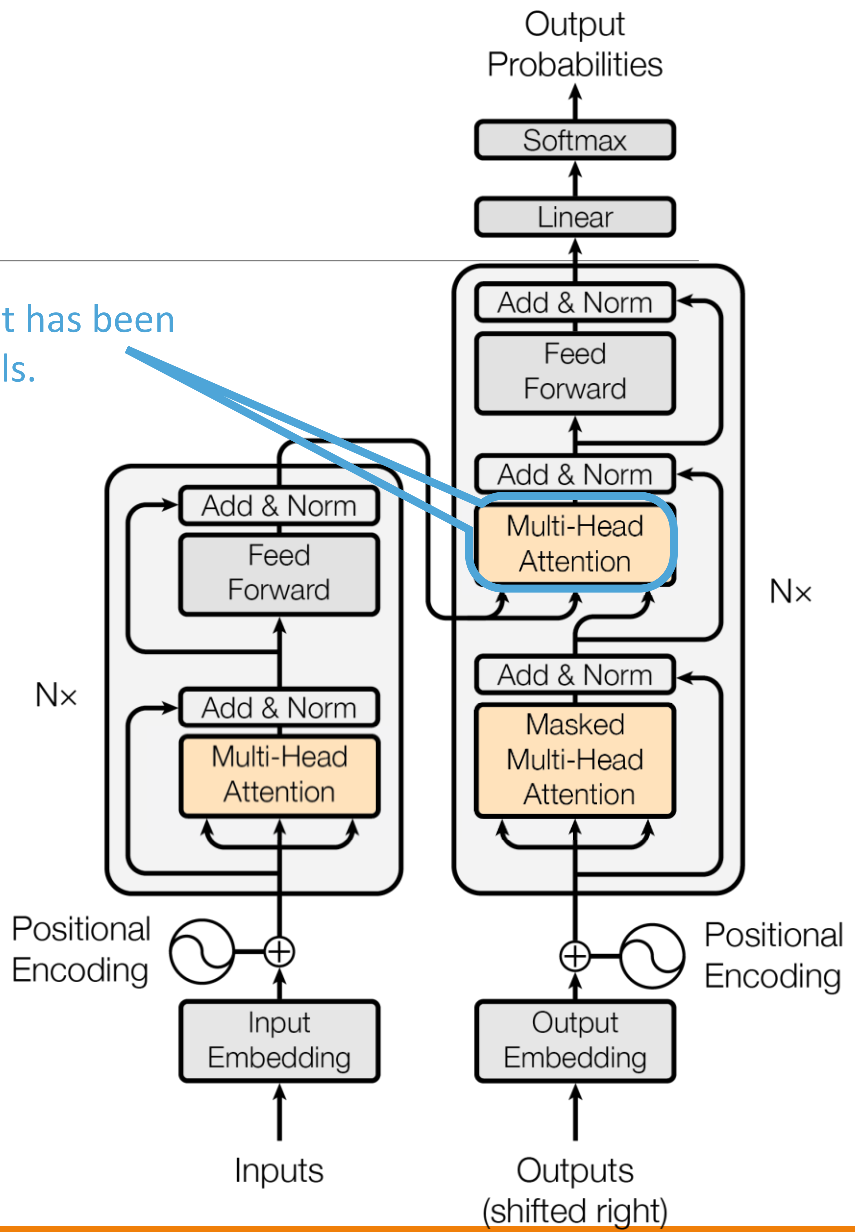

Self-attention between a sequence of hidden states and that same sequence of hidden states.



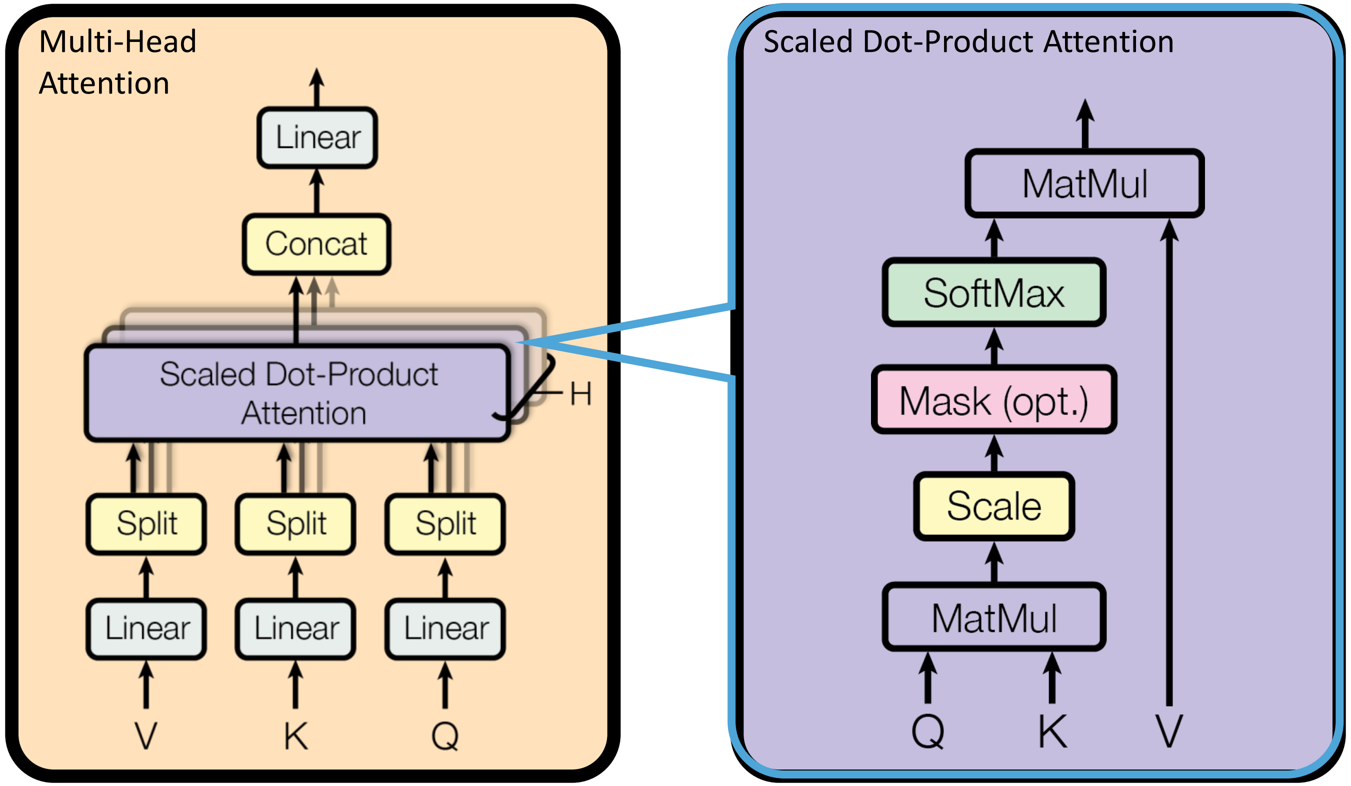
# Multi-Head Attention

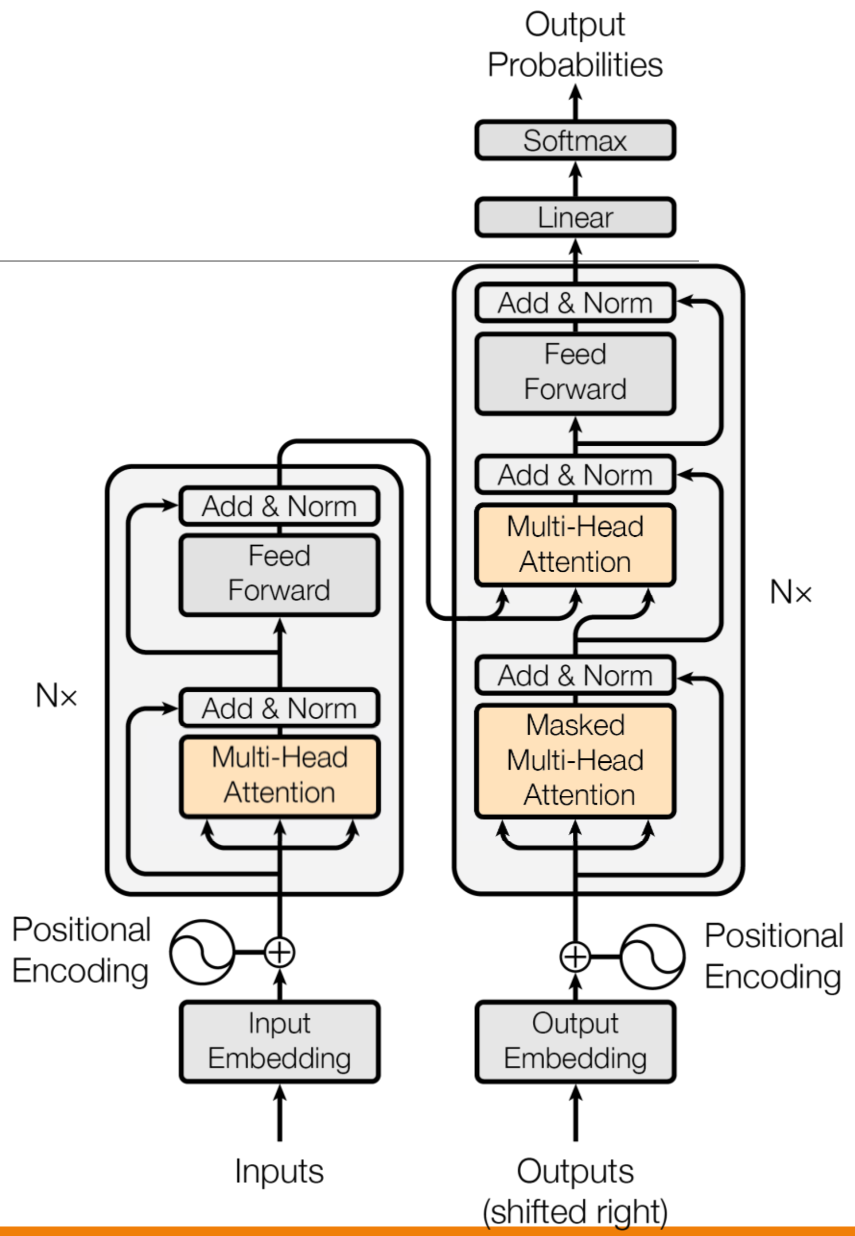



Encoder-decoder attention, like what has been standard in recurrent seq2seq models.

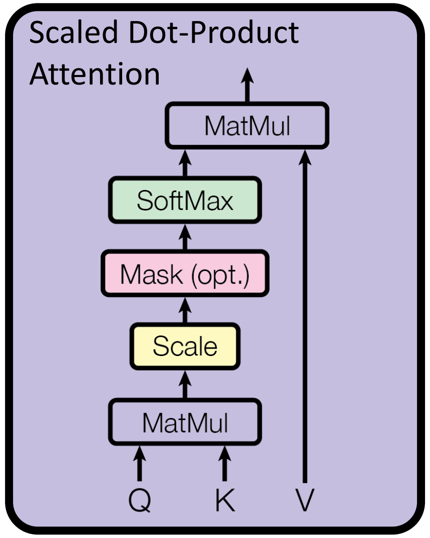


# Attention Mechanism



# Scaled Dot-Product Attention



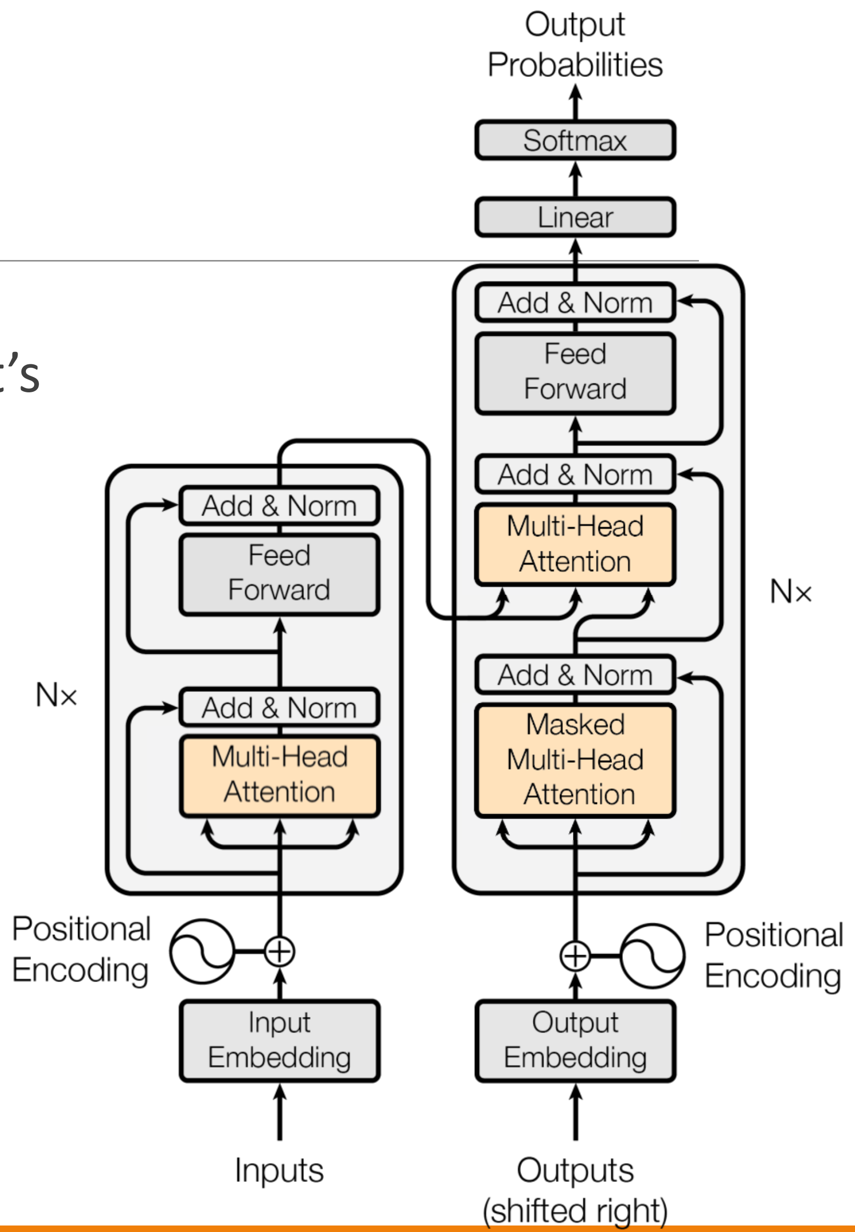

The scaled dot-product attention mechanism is almost identical to the one we looked at, but let's turn it into matrix multiplications.

The query: Q ∈ $R^{Txdk}$

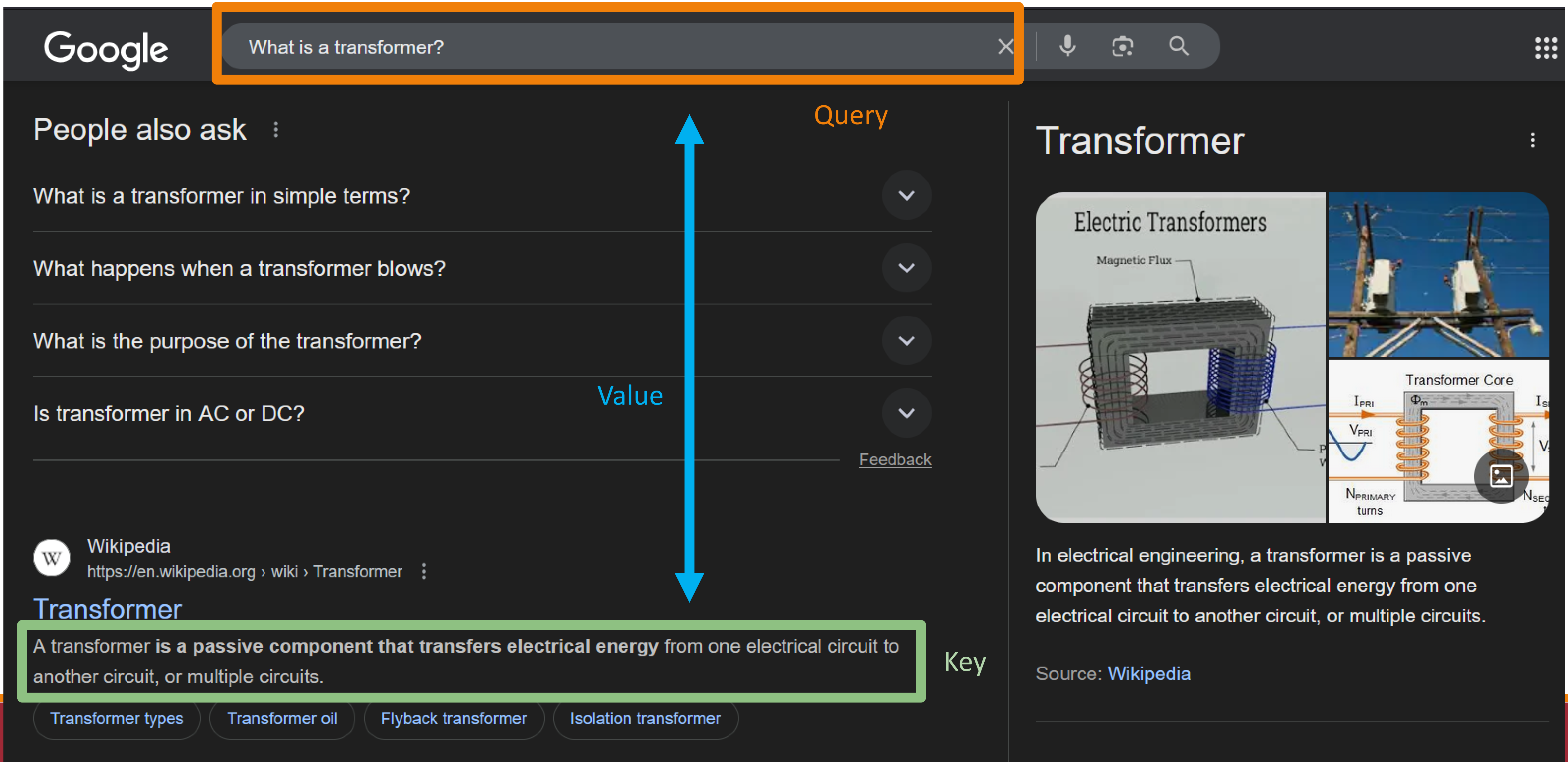
The key: K ∈ $R^{T'xdk}$

The value: V ∈ $R^{Txdk}$

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



# An analogy…



Google
What is a transformer?
Query
People also ask
What is a transformer in simple terms?
What happens when a transformer blows?
What is the purpose of the transformer?
Is transformer in AC or DC?
Value
Feedback
Wikipedia
https://en.wikipedia.org › wiki › Transformer
Transformer
A transformer is a passive component that transfers electrical energy from one electrical circuit to another circuit, or multiple circuits.
Key
Transformer types
Transformer oil
Flyback transformer
Isolation transformer
Transformer
Electric Transformers
Magnetic Flux
Transformer Core
In electrical engineering, a transformer is a passive component that transfers electrical energy from one electrical circuit to another circuit, or multiple circuits.
Source: Wikipedia

# 3Blue1Brown Explanation of Q,K,V (~6 minutes)

https://youtu.be/eMlx5fFNoYc?si=1sXvOHytbTUPqnE8&t=366

6:06 - 9:28 = 3:22

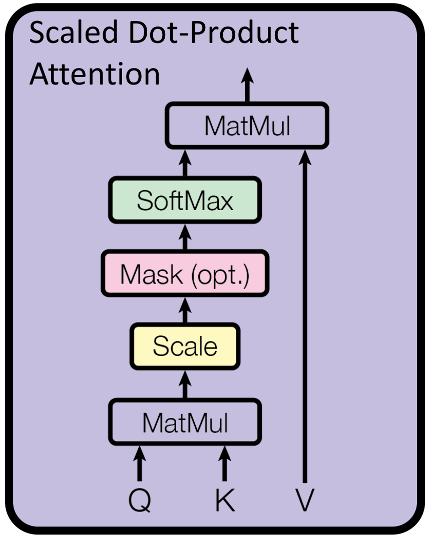
And then skip ahead to values

https://youtu.be/eMlx5fFNoYc?t=790&si=uNLE2TOpFtxkdDEj

13:10 – 15:43

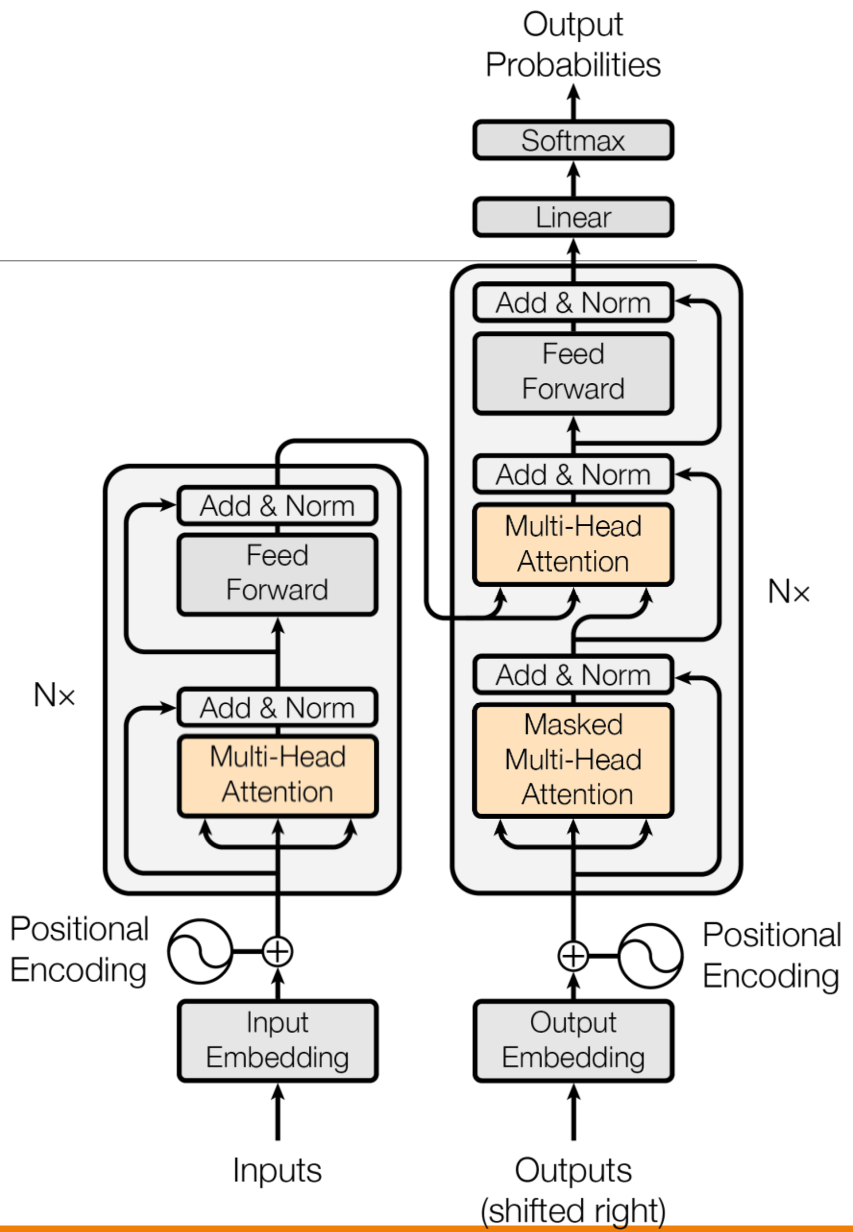
# Scaled Dot-Product Attention

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



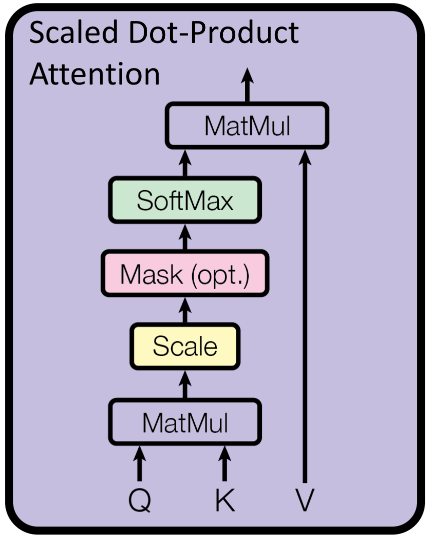

The rough algorithm:

- For each vector in Q (query matrix), take the linear sum of the vectors in V (value matrix)
- The amount to weigh each vector in V is dependent on how "similar" that vector is to the query vector
- "Similarity" is measured in terms of the dot product between the vectors



# Scaled Dot-Product Attention

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

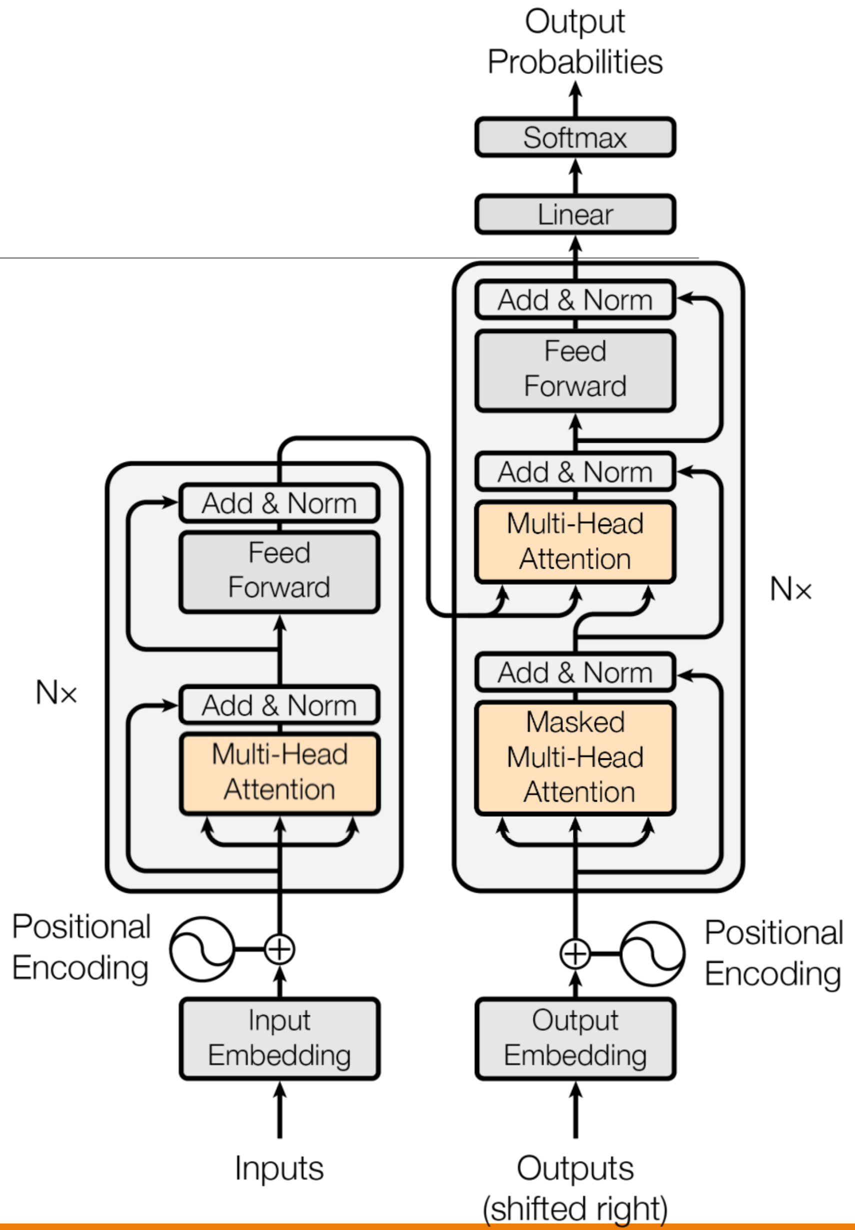



**For self-attention:**

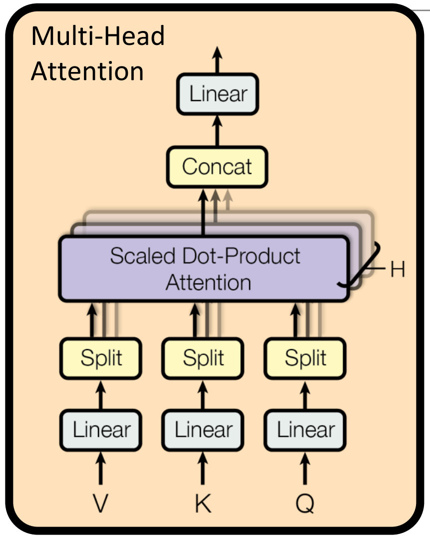
Keys, queries, and values all come from the outputs of the previous layer

**For encoder-decoder attention:**

Keys and values come from encoder's final output. Queries come from the previous decoder layer's outputs.



# Multi-Head Attention



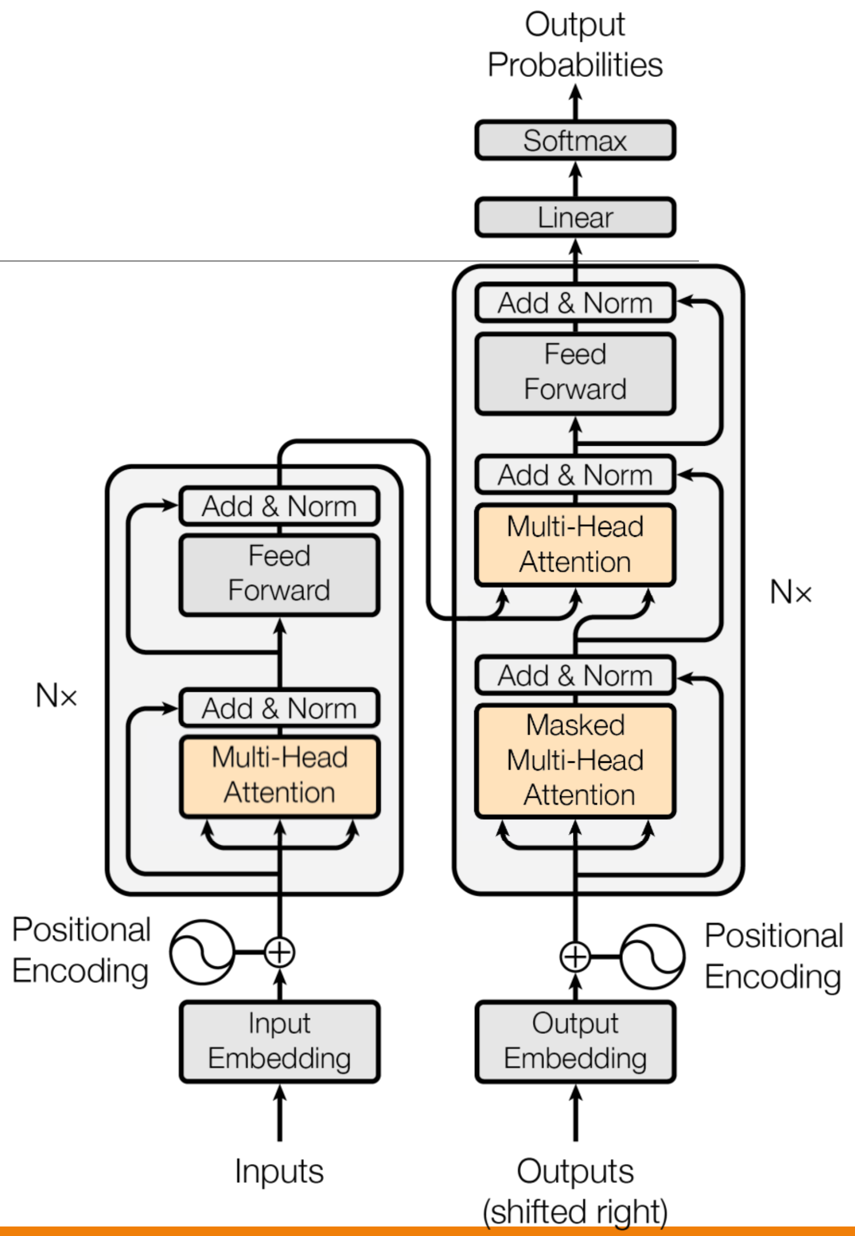

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

$$\text{MultiHeadAtt}(\mathbf{Q,K,V}) = \text{Concat}(\text{head}_1, \ldots \text{head}_h)\mathbf{W}^O$$

Instead of operating on **Q**, **K**, and **V** mechanism projects each input into a smaller dimension. This is done h times.

The attention operation is performed on each of these "heads," and the results are concatenated.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.



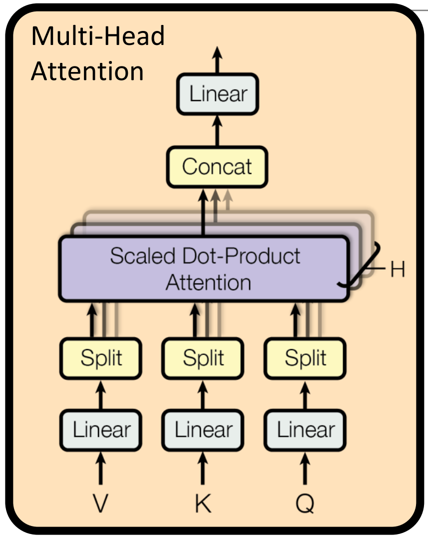
# Knowledge Check

Run first three cells of the Visualizing Attention with BertViz notebook

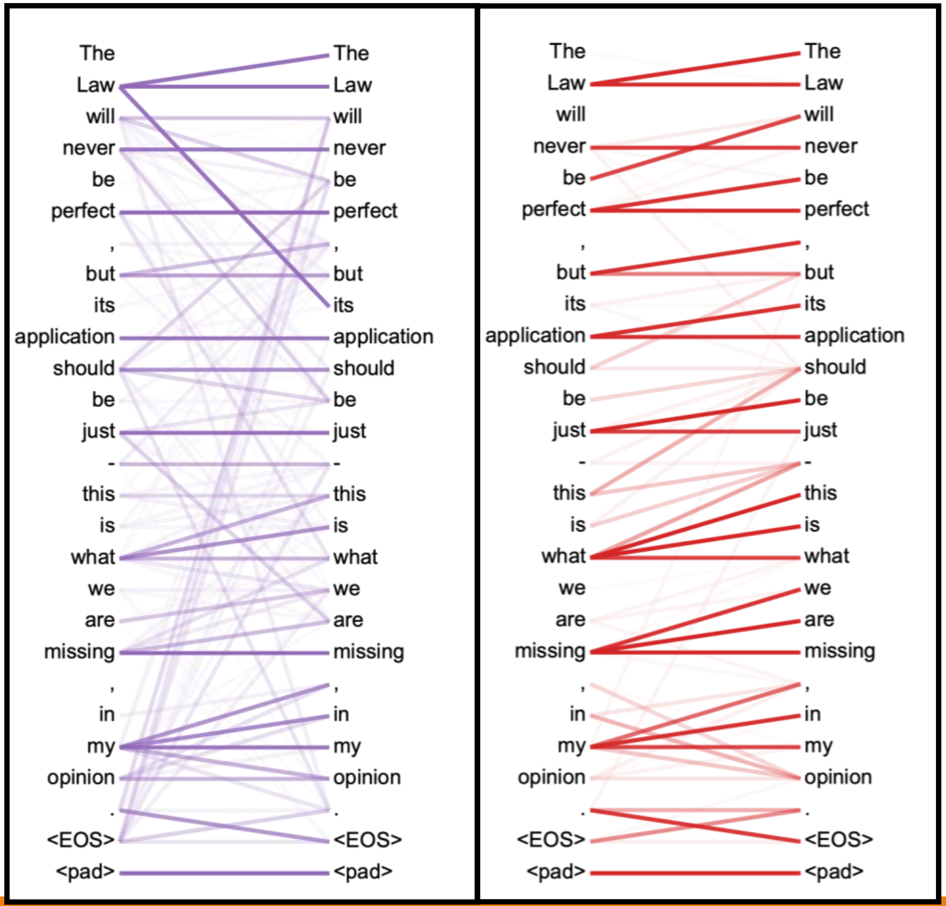
For the cell visualizing the "cat sentence", look at the different layers of attention. (You can change the layer using the drop-down menu next to "Layer").

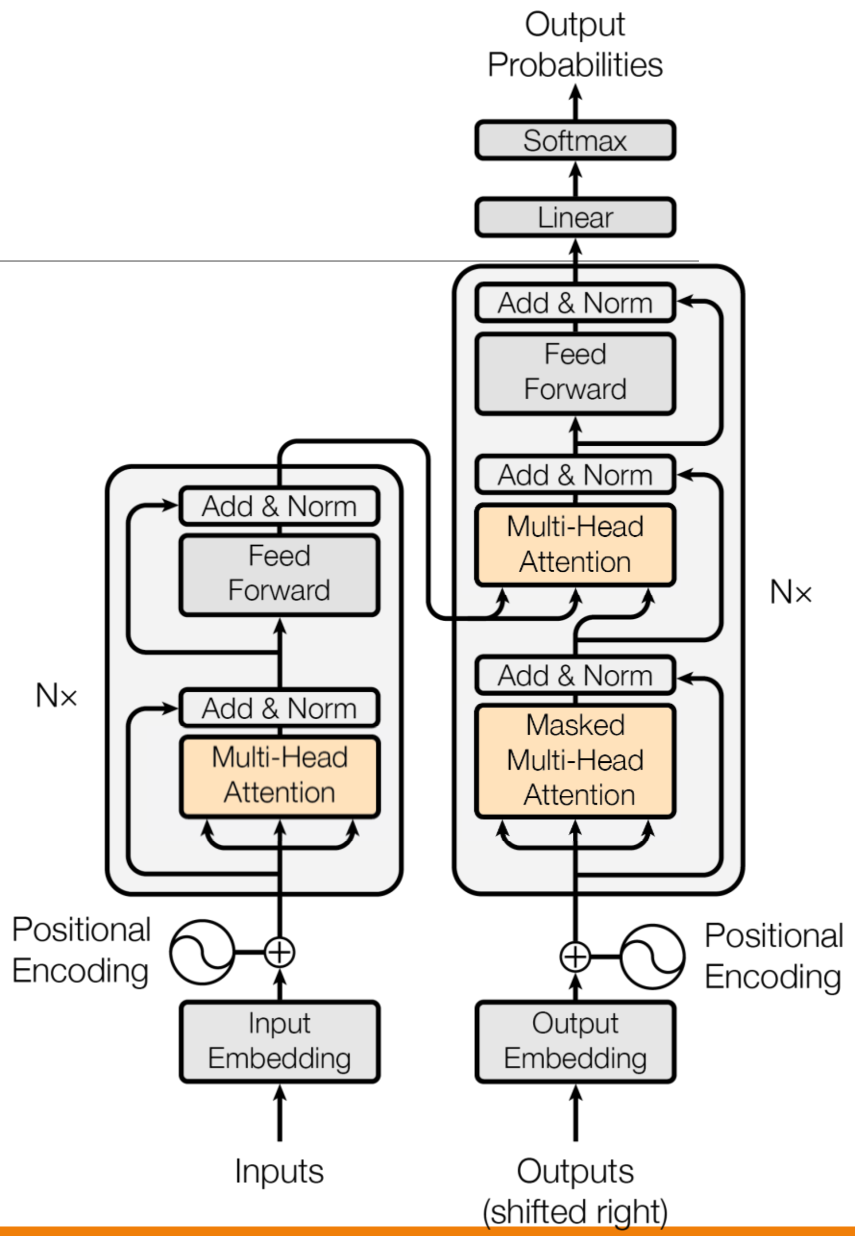
1. Are there any patterns that you see between the layers? (e.g., What words are connected to what other words for each layer?)
2. Come up with a guess for what type of information each layer could be capturing.
3. Change the sentence on the `inputs = tokenizer.encode()` line and run the cell again. Does this break what you thought for question #2? Explain.

# Multi-Head Attention




Two different self-attention heads:

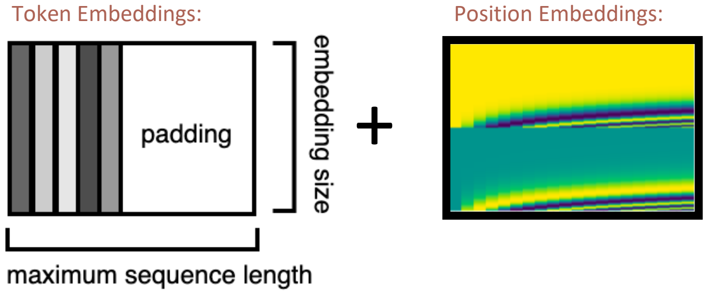


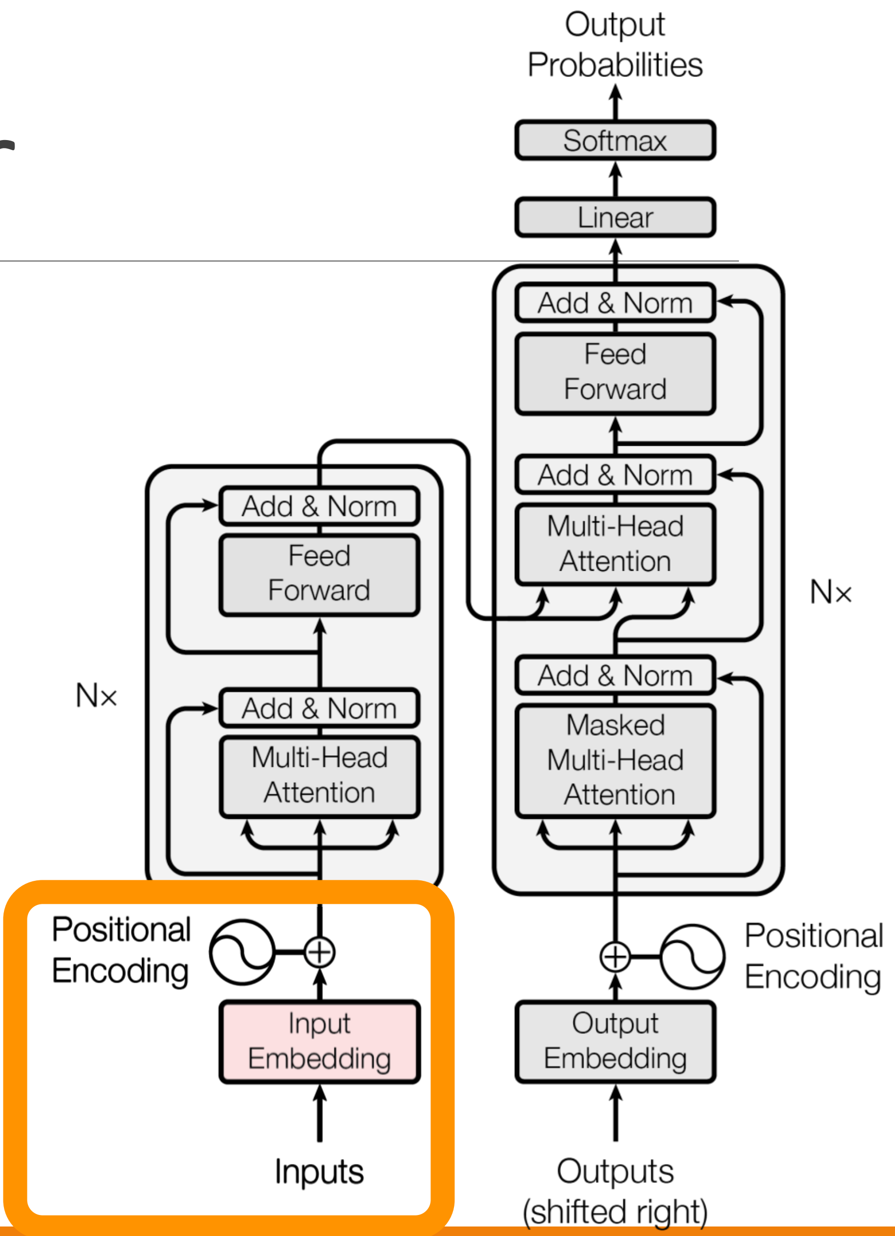
# Inputs to the Encoder

The input into the encoder looks like:

$\mathbf{H}_0^{\text{enc}} =$




$=$ token embeddings + position embeddings



# How does the transformer compare to the seq2seq RNN?

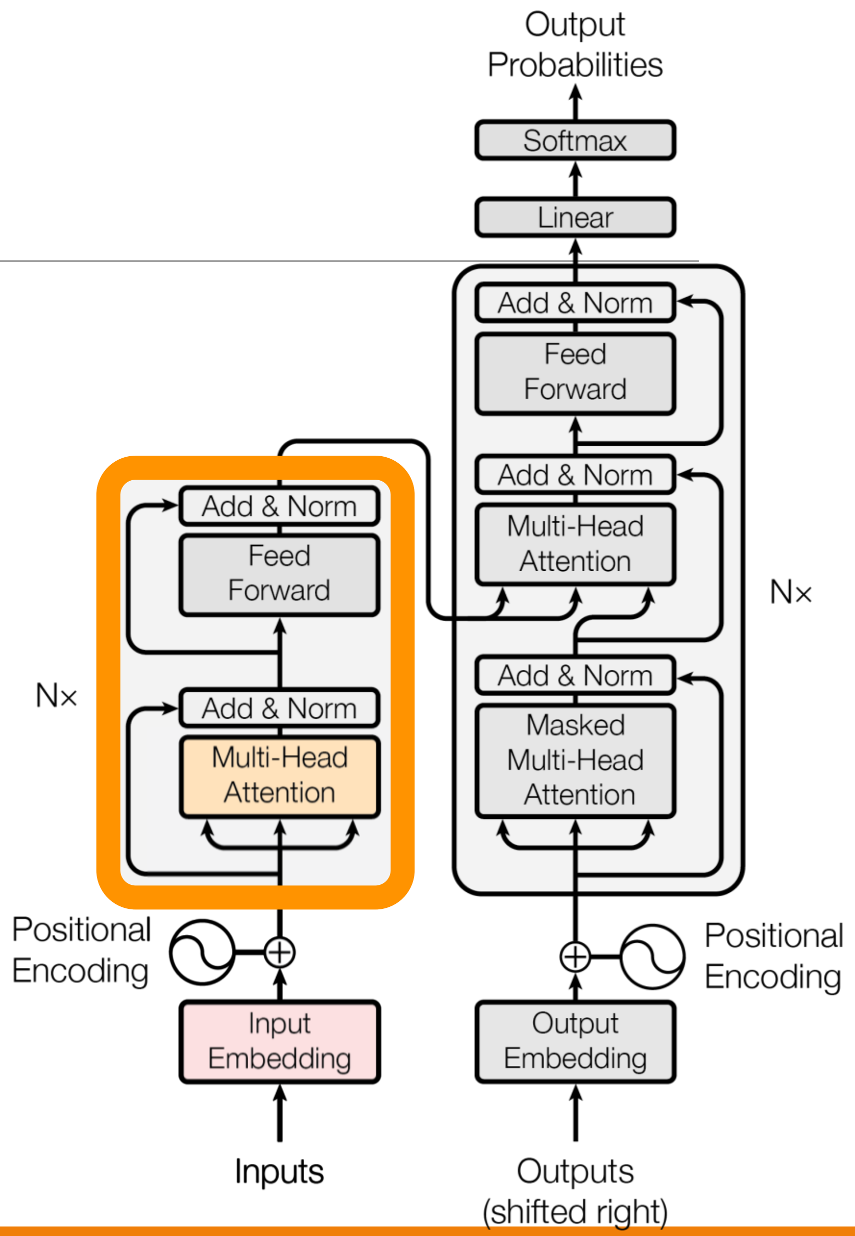
# Think-Pair-Share

Why do you think we don’t need recurrence anymore (i.e., why is “attention all you need”)?

# If you want more details, check out the following slides

# The Encoder

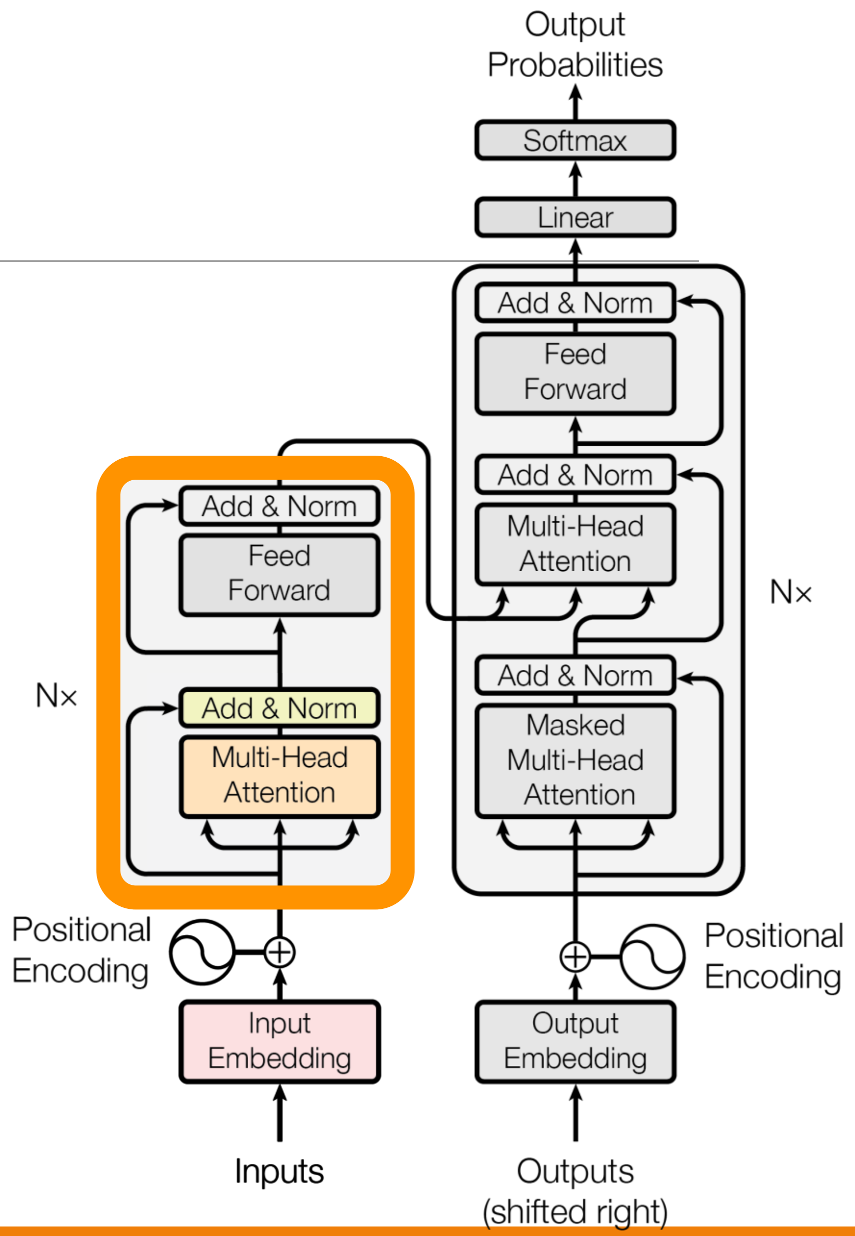
Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$



# The Encoder

$$\boxed{\text{Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm}} = \text{LayerNorm}\left(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{enc}\right)$$



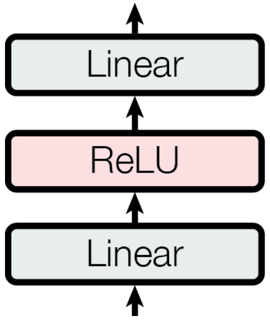
# The Encoder

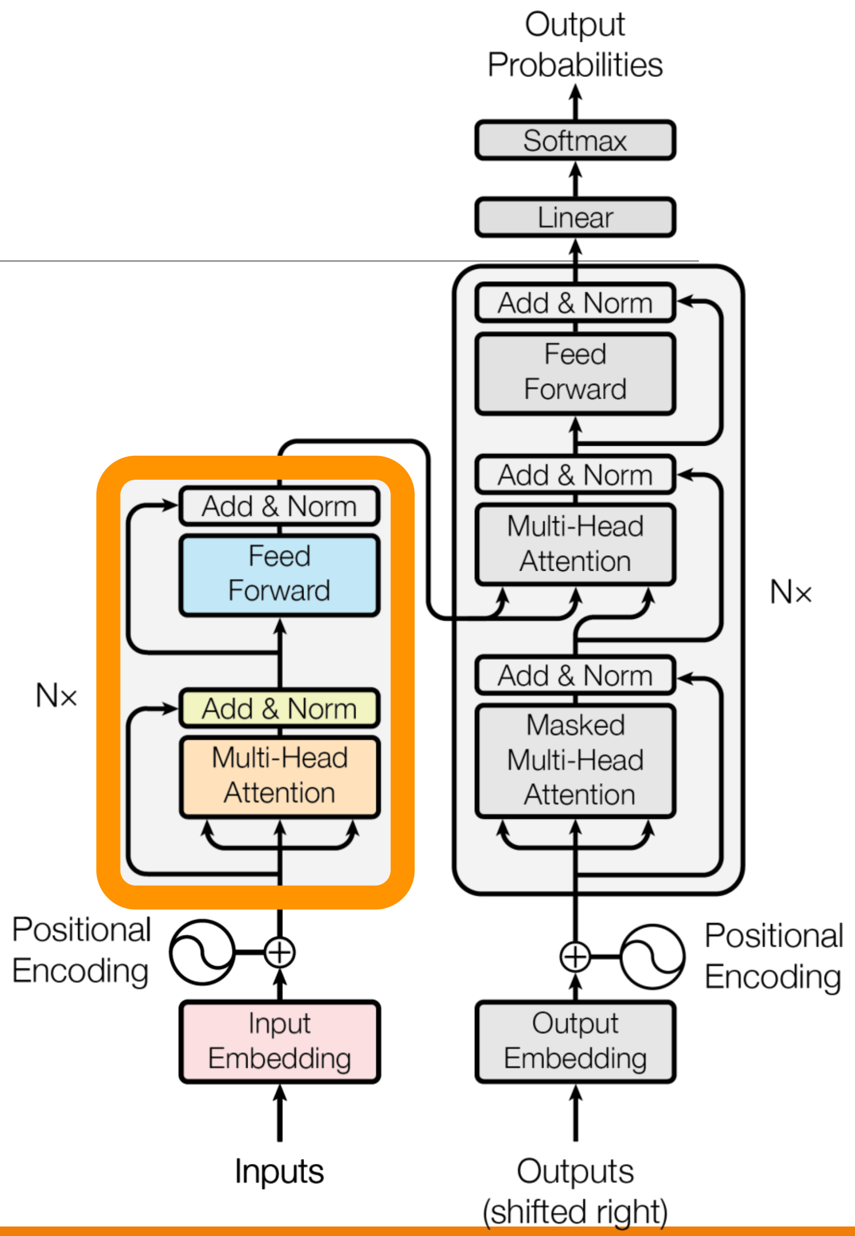
$$\boxed{\text{Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{enc})$$

$$\boxed{\text{Feed Forward}} = max(0, \boxed{\text{Add \& Norm}}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

$\boxed{\text{Feed Forward}}$ <=>



# The Encoder

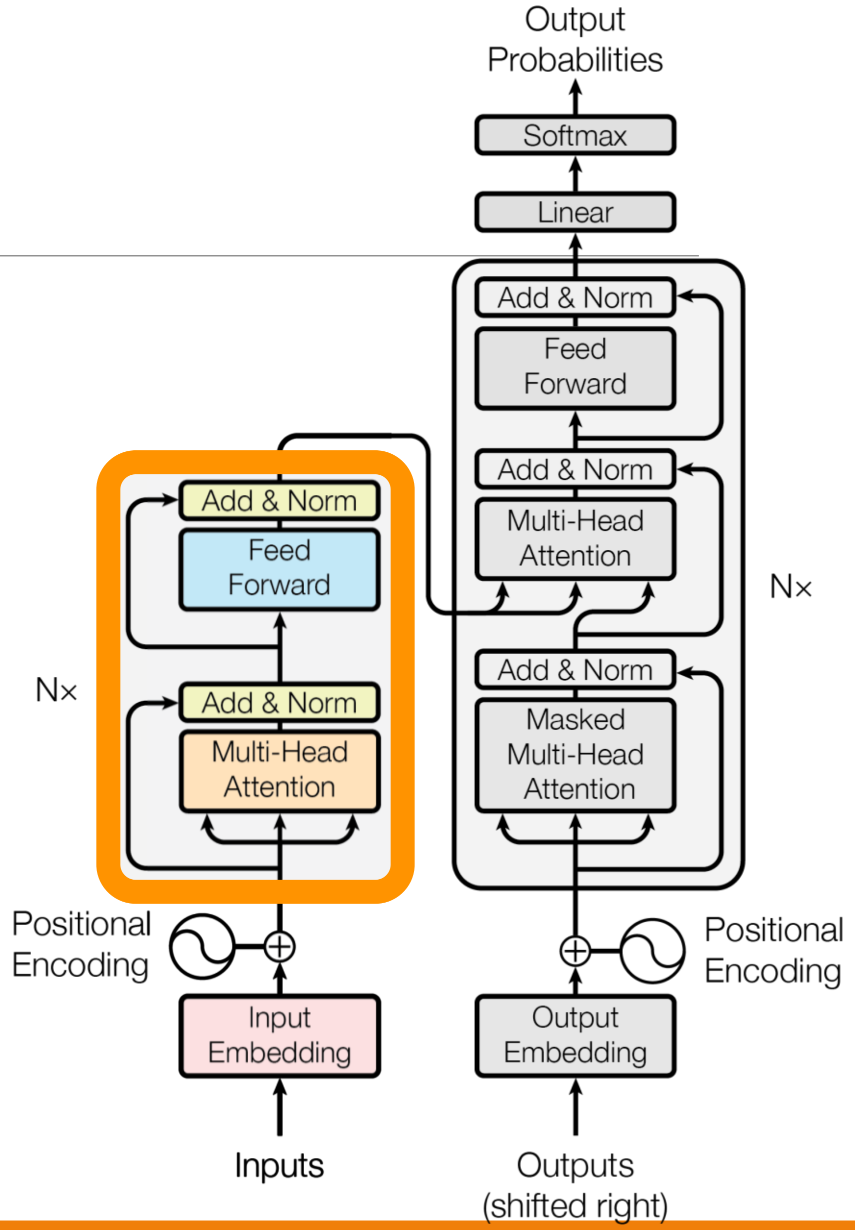
$$\boxed{\text{Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm}} = \text{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{enc})$$

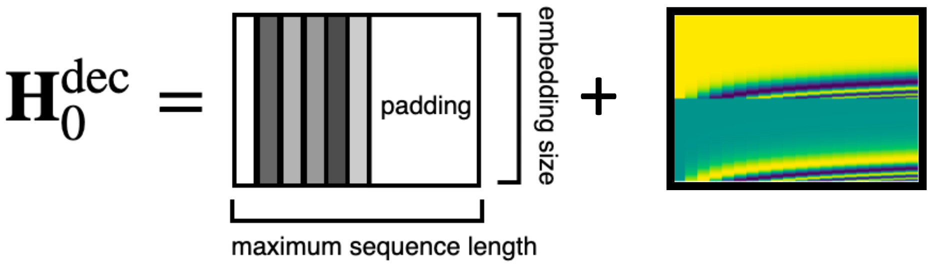
$$\boxed{\text{Feed Forward}} = max(0, \boxed{\text{Add \& Norm}}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

$$\boxed{\text{Add \& Norm (2)}} = \text{LayerNorm}(\boxed{\text{Feed Forward}} + \mathbf{H}_i^{enc})$$

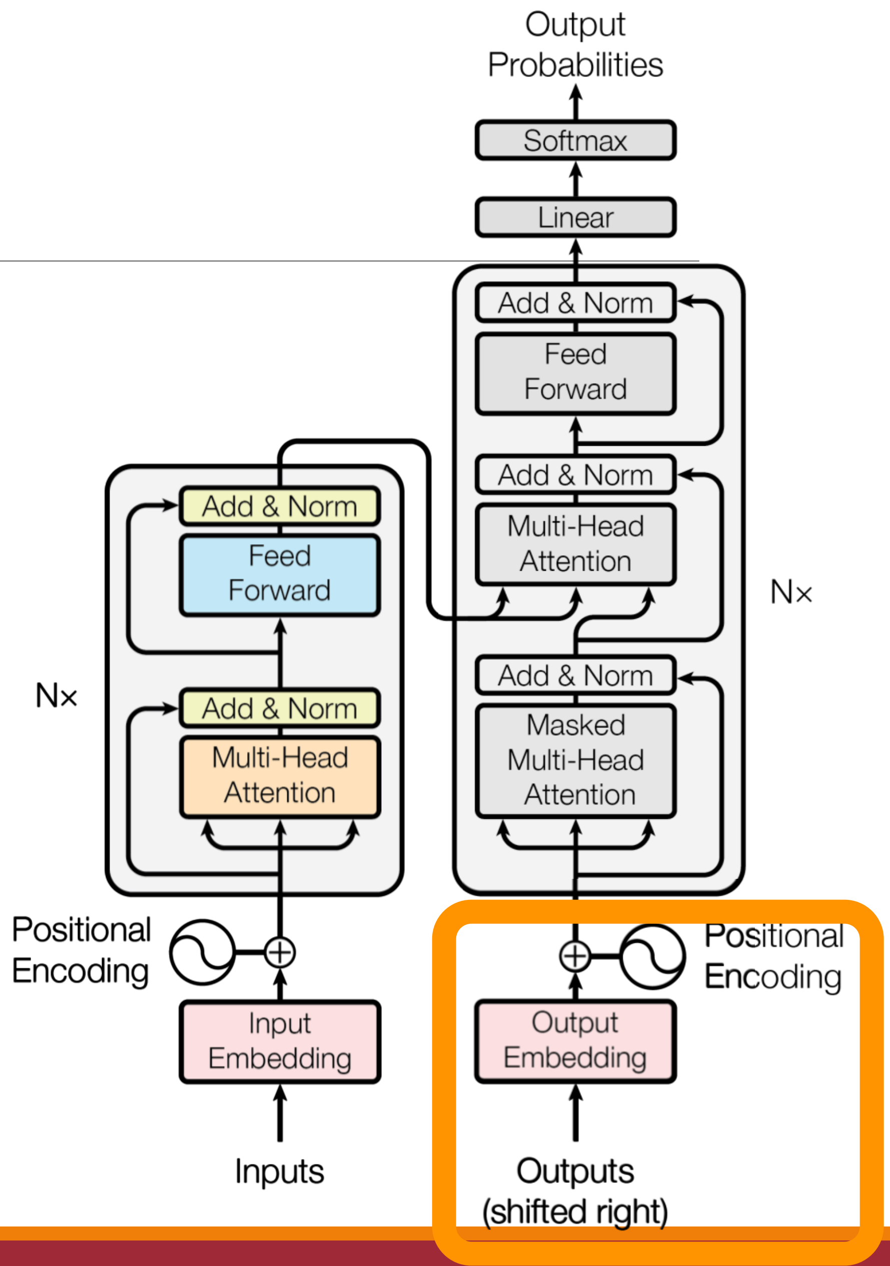
$$\mathbf{H}_{i+1}^{enc} = \boxed{\text{Add \& Norm (2)}}$$



# The Decoder

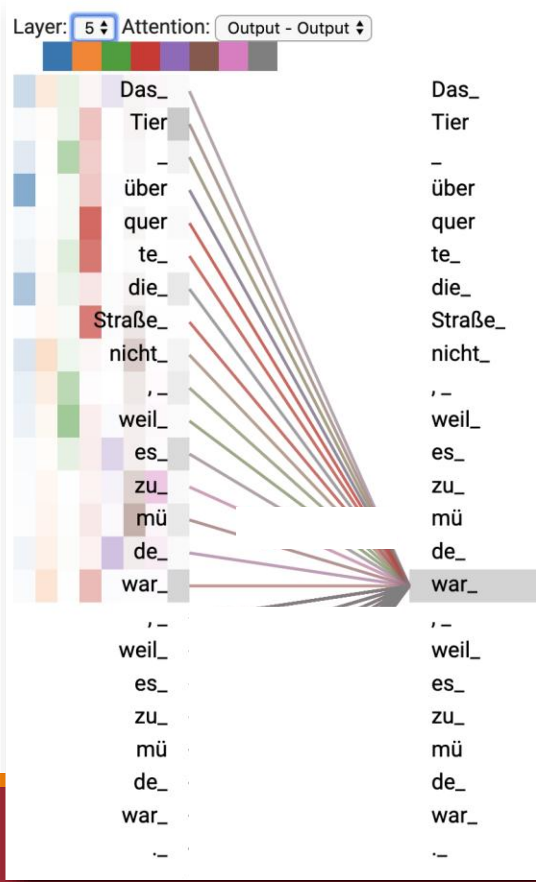


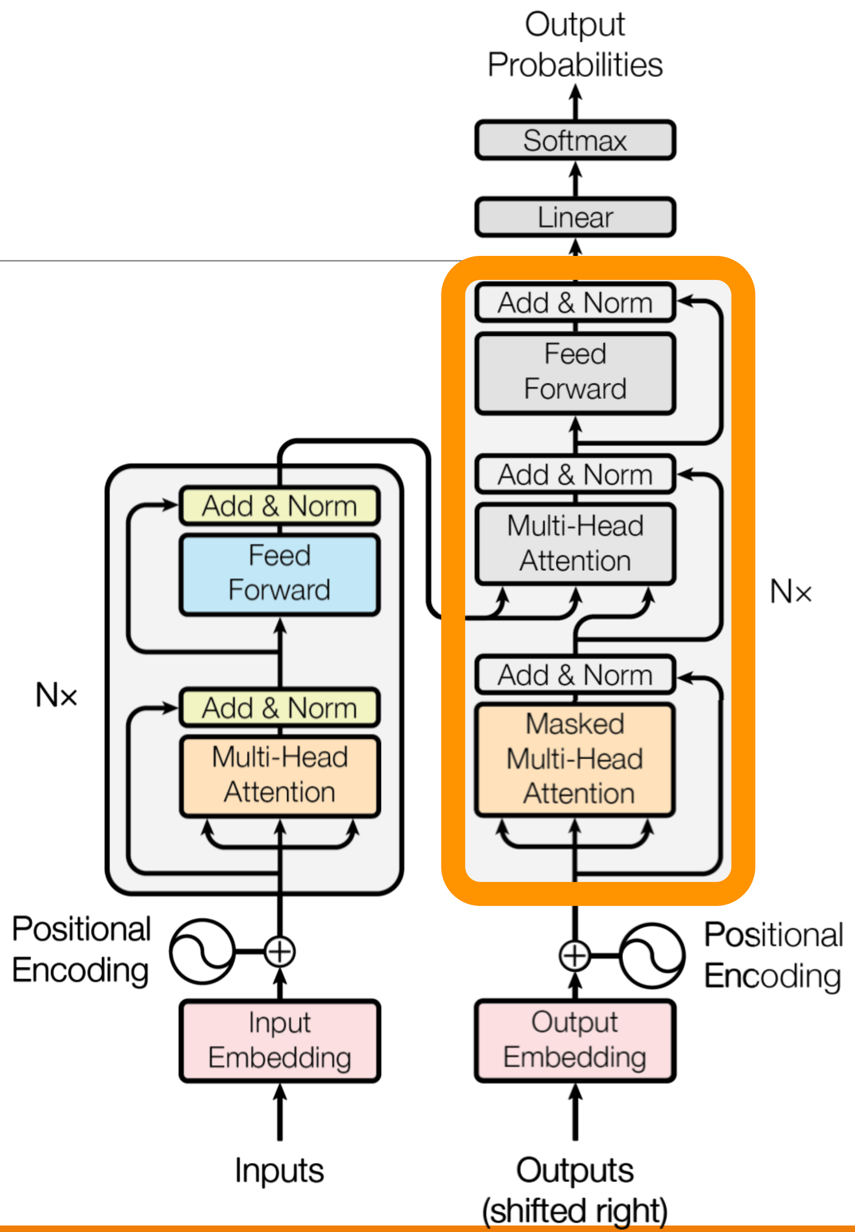

= token embeddings + position embeddings



# The Decoder

Masked Multi-Head Attention $= \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$

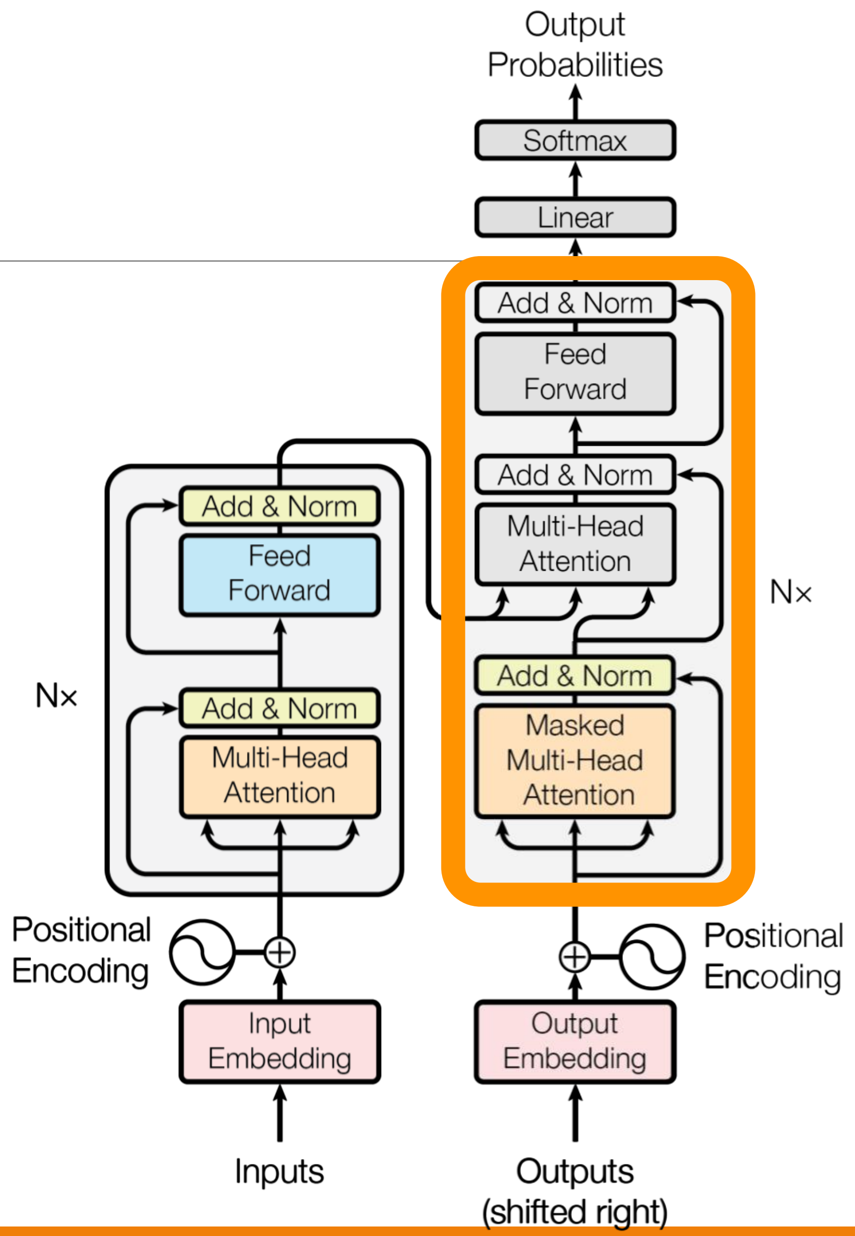


# The Decoder

$\boxed{\text{Masked Multi-Head Attention}} = \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$

$\boxed{\text{Add \& Norm}} = \text{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{dec})$

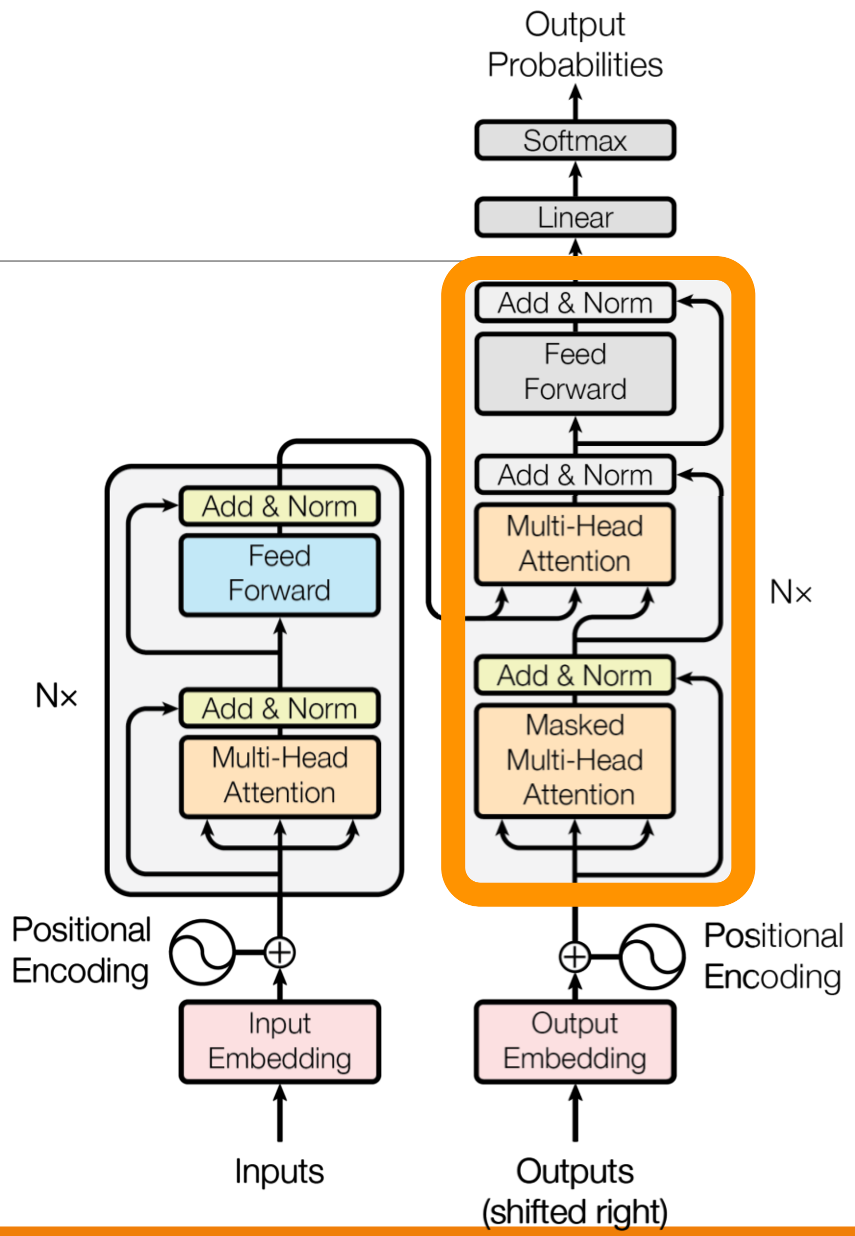


# The Decoder

$\boxed{\text{Masked Multi-Head Attention}} = \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$

$\boxed{\text{Add \& Norm}} = \text{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{dec})$

$\boxed{\text{Enc-Dec Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$



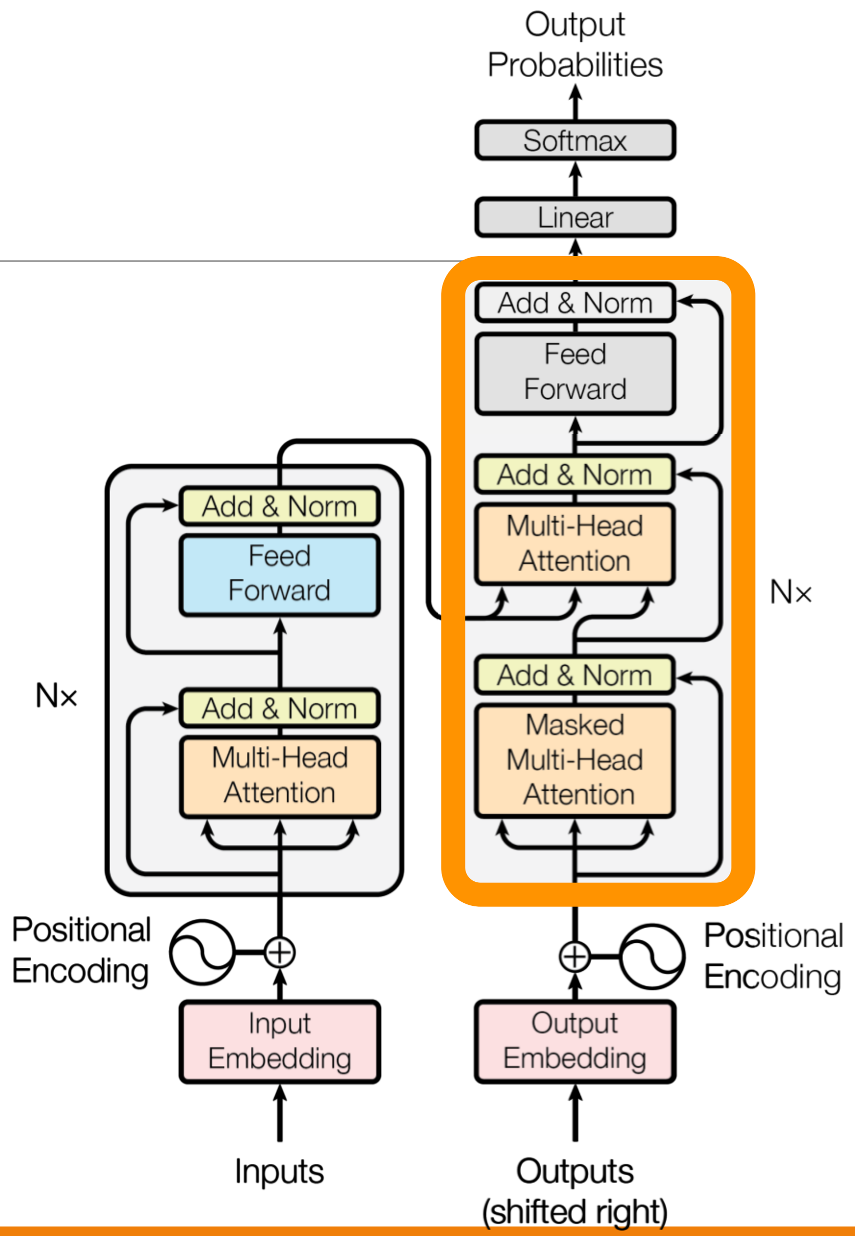
# The Decoder

Masked Multi-Head Attention $= \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$

Add & Norm $= \text{LayerNorm}($ Multi-Head Attention $+ \mathbf{H}_i^{dec})$

Enc-Dec Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$

Add & Norm (2) = LayerNorm( Multi-Head Attention + Add & Norm )



# The Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\boxed{\text{Add \& Norm}} = \text{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{dec})$$

$$\boxed{\text{Enc-Dec Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm (2)}} = \text{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Feed Forward}} = max(0, \boxed{\text{Add \& Norm (2)}}\, \mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

$$\boxed{\text{Add \& Norm (3)}} = \text{LayerNorm}(\boxed{\text{Feed Forward}} + \boxed{\text{Add \& Norm (2)}})$$

$$\mathbf{H}_{i+1}^{dec} = \boxed{\text{Add \& Norm (3)}}$$