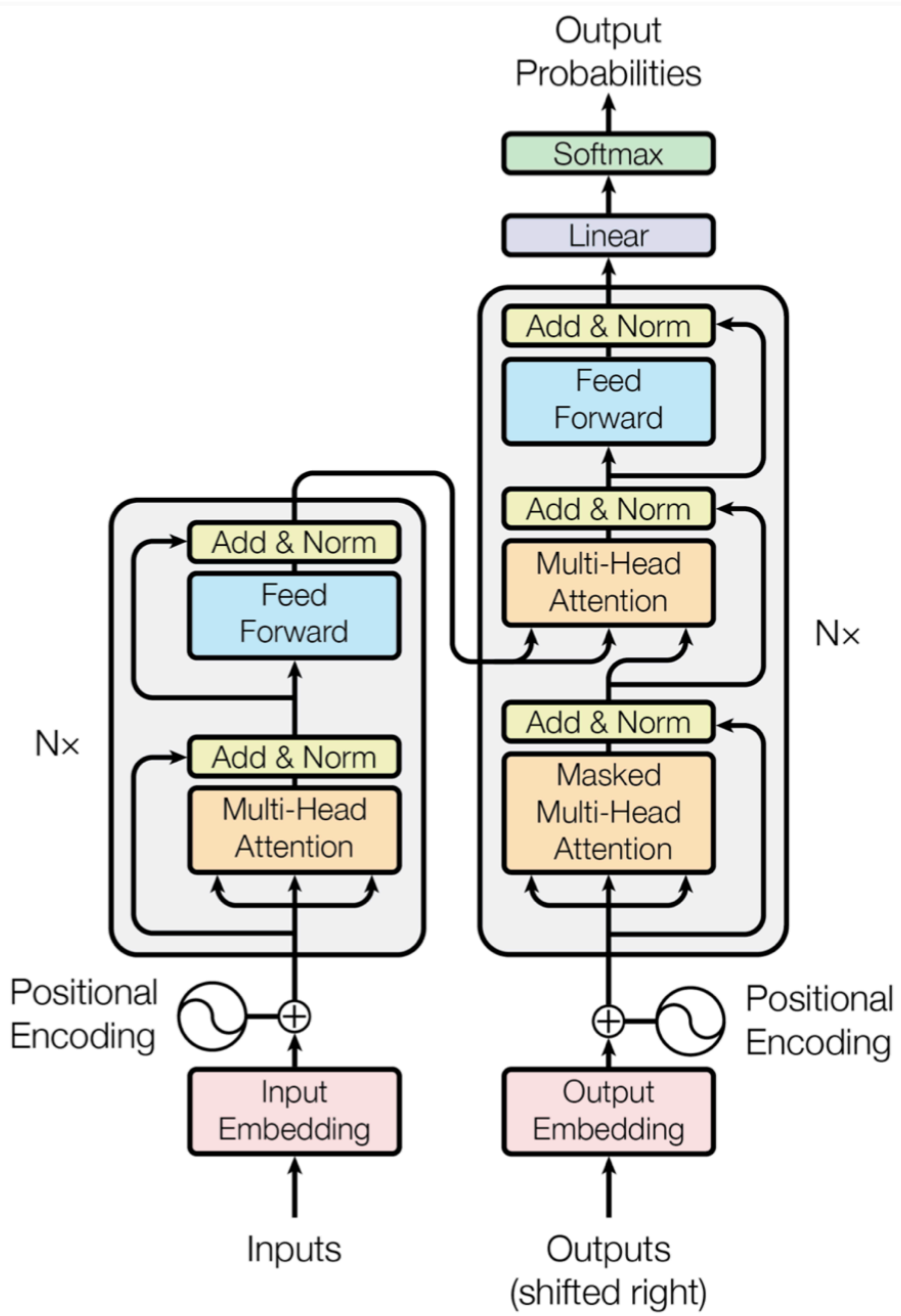
香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMP 5212
Machine Learning
Lecture 20

Transformers, VAEs

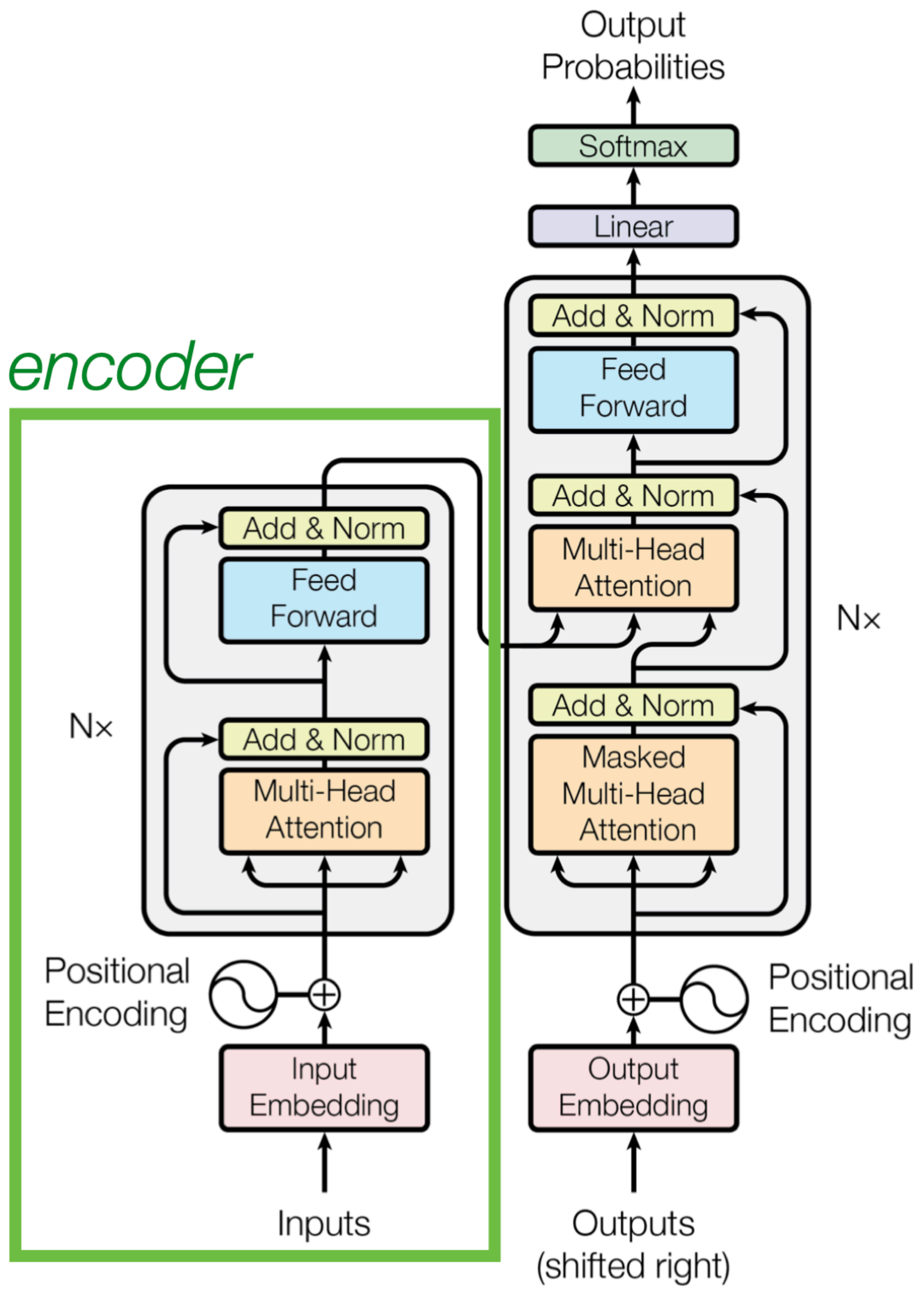
Junxian He
Apr 19, 2024

Transformer



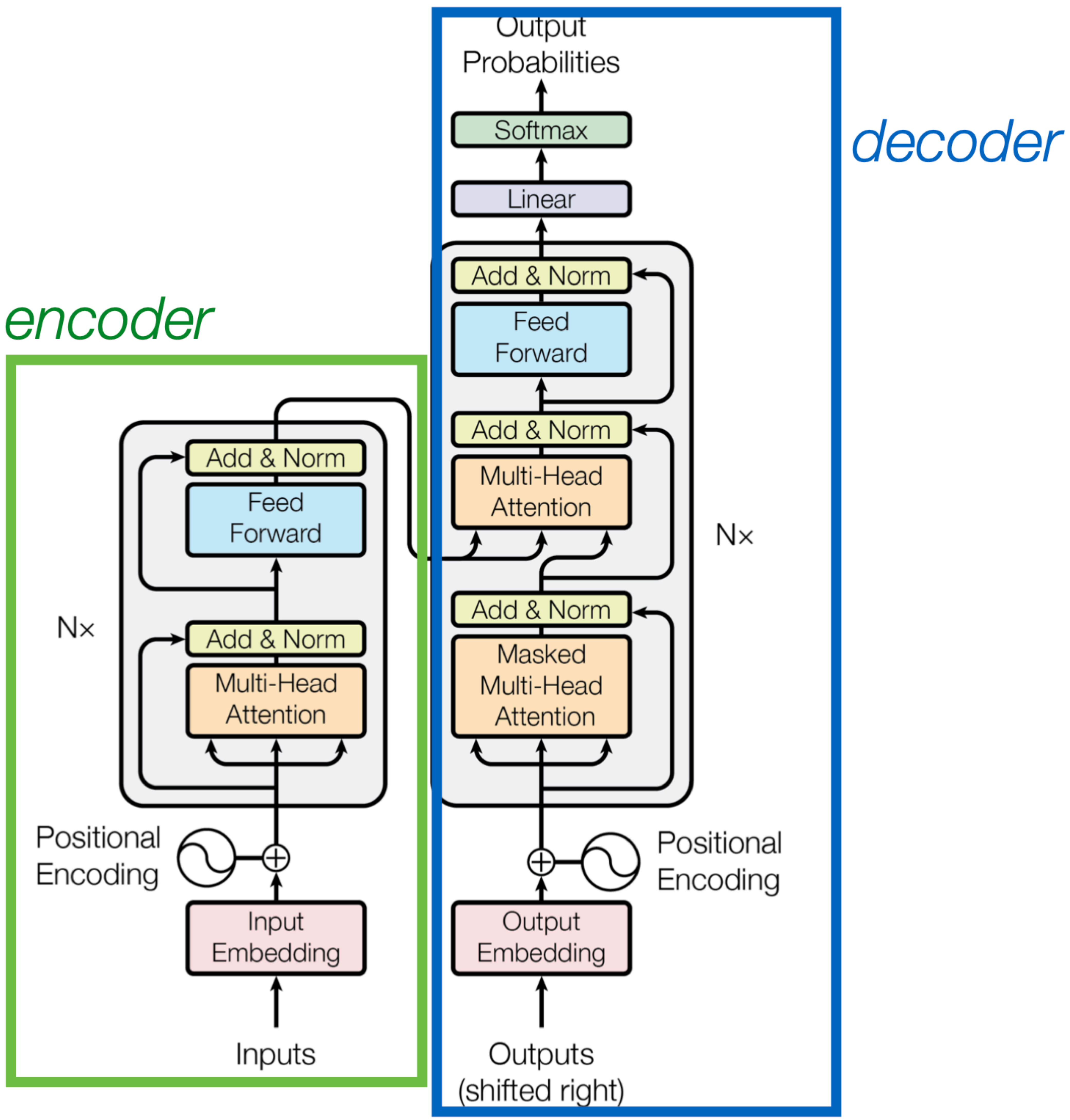
Vaswani et al. Attention is All You Need. NeurIPS 2017.

Encoder



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

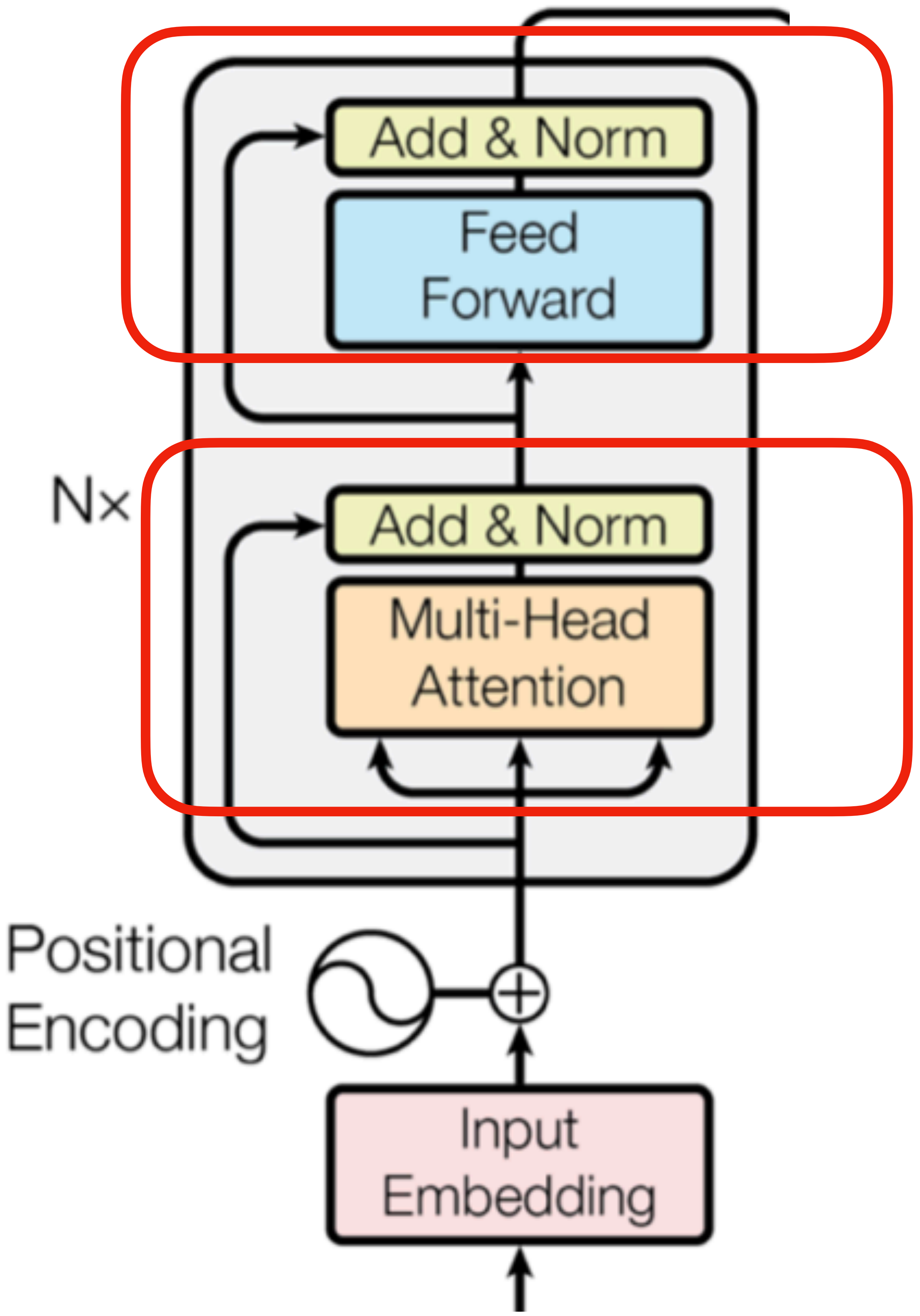
Decoder



decoder
encoder
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
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

Transformer Encoder

Residual connection

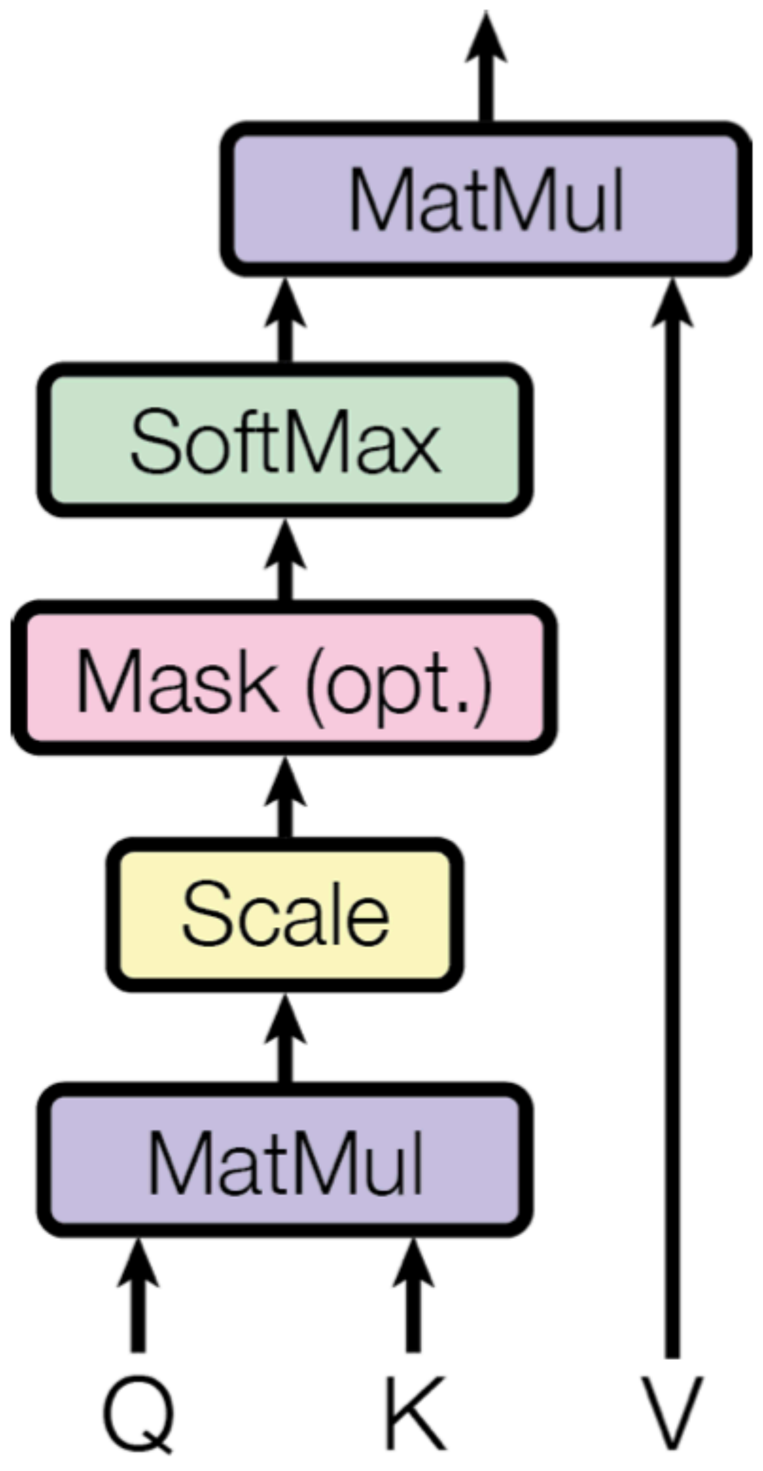


MLP

Self-attention

What is Attention

Scaled Dot-Product Attention



Q: Query
K: key
V: value

$$Q \in R^{n \times d} \qquad K \in R^{m \times d} \qquad V \in R^{m \times d}$$

We have n queries, m (key, value) pairs

Attention weight = softmax(QK^T)

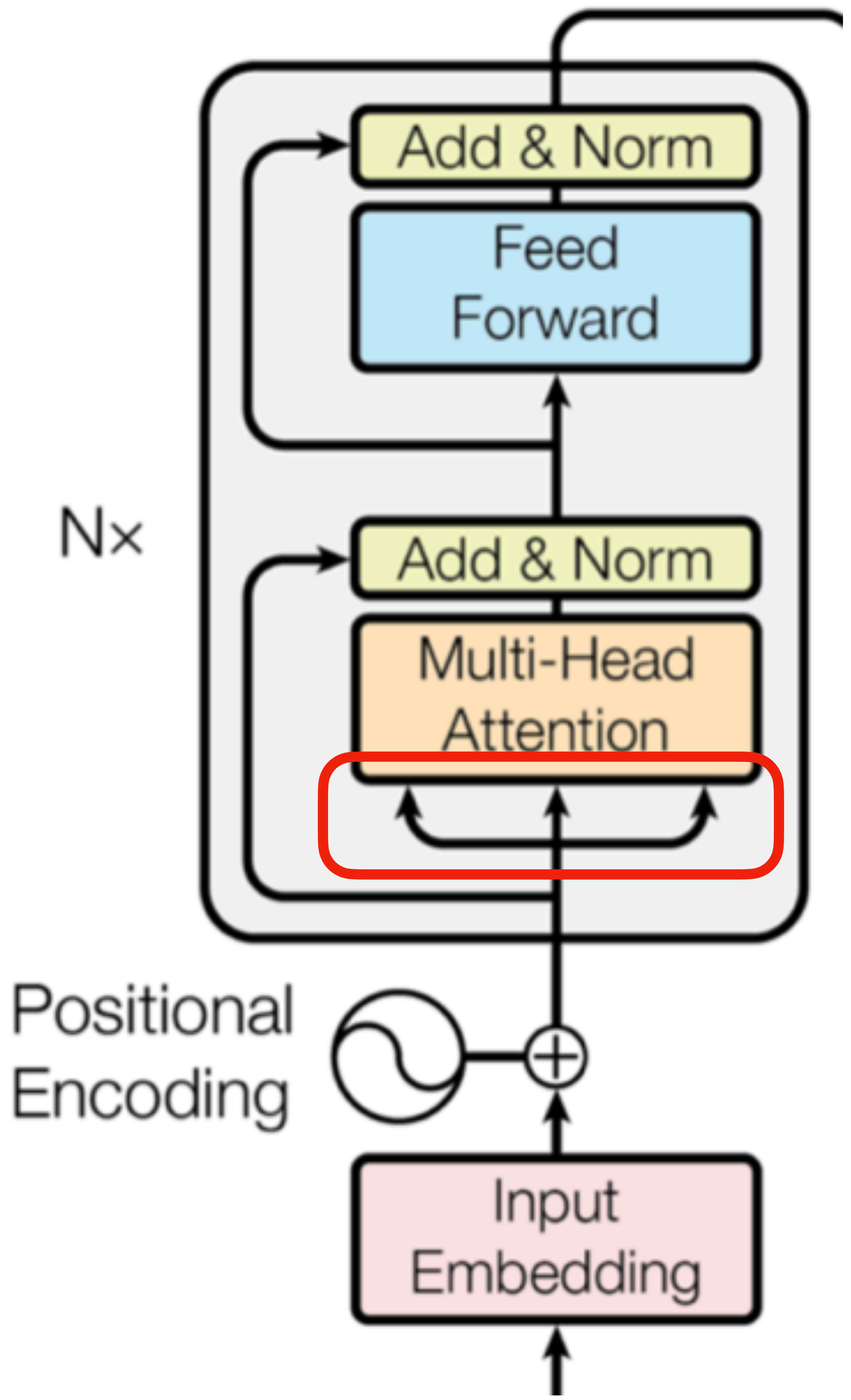
Dot-products grow large in magnitude

Scaled Attention weight = softmax$(\frac{QK^T}{\sqrt{d_k}})$ Shape is mxn

Attention weight represents the strength to "attend" values V

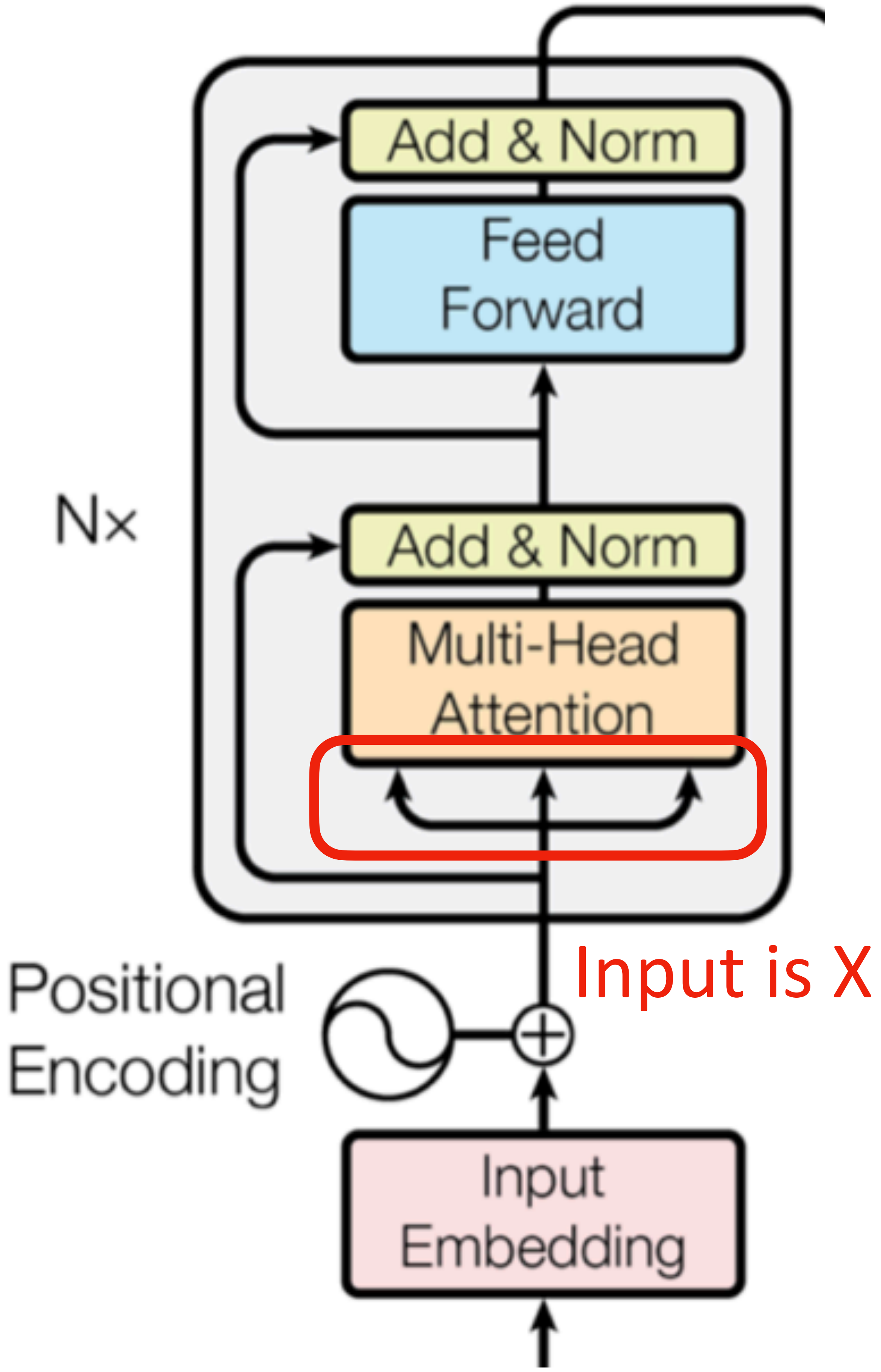
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

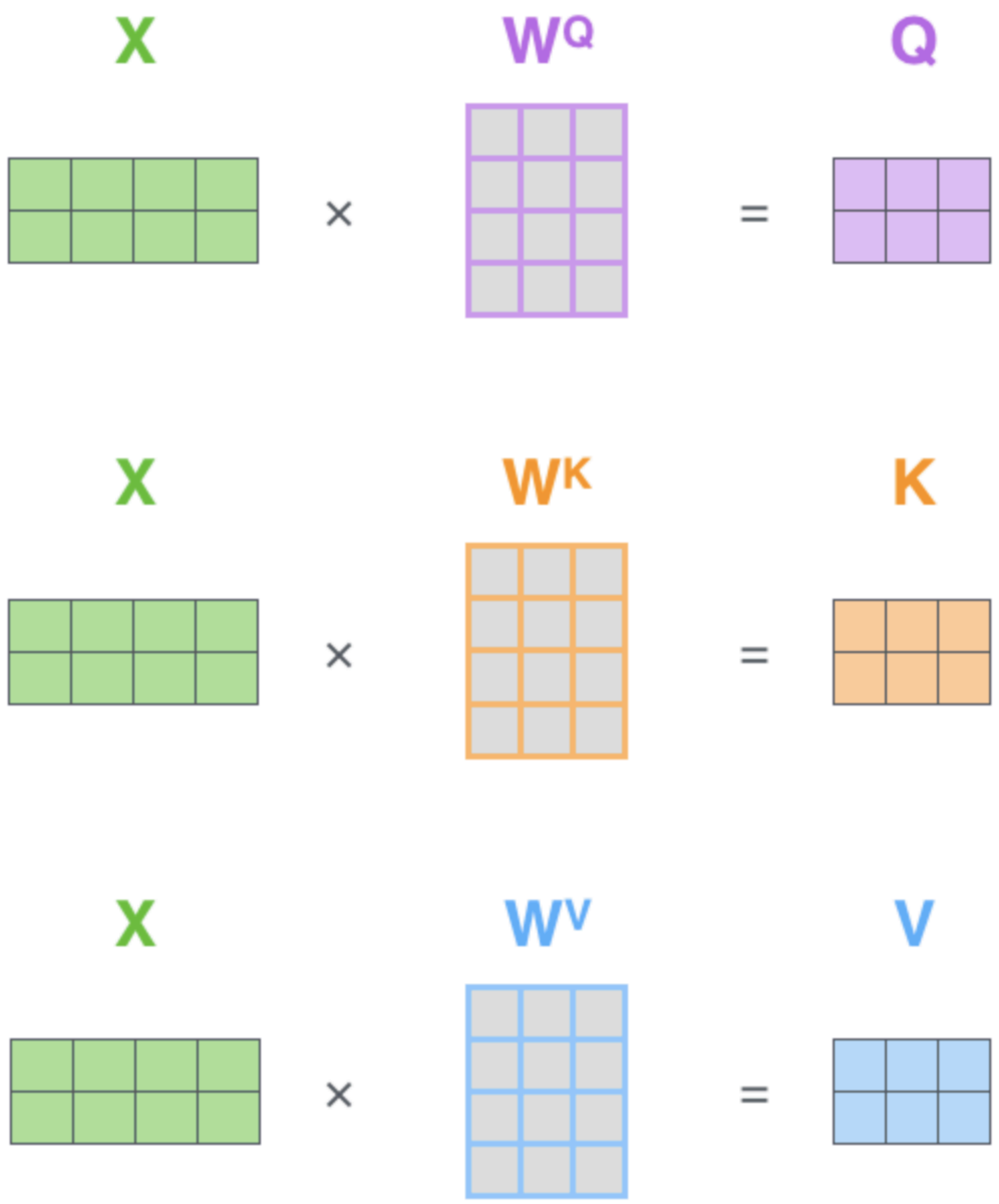
Q, K, V



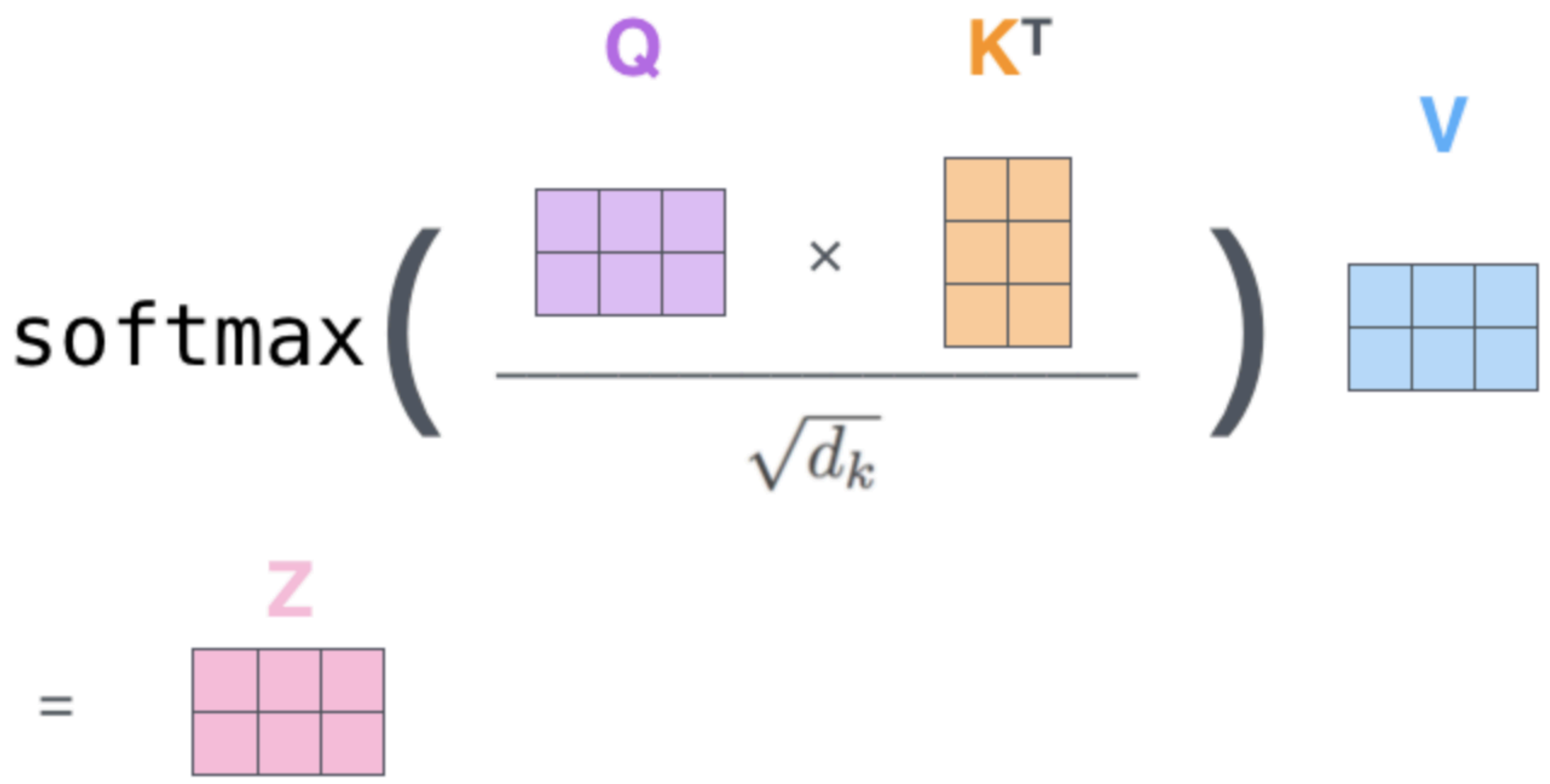
What are Q, K, V in the transformer

Self-Attention





Query, key, and value are from the same input, thus it is called "self"-attention

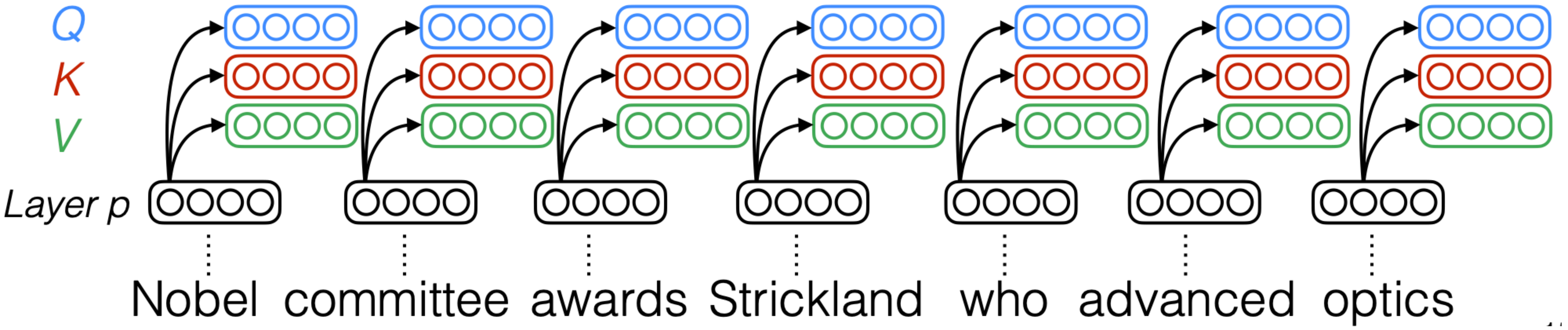


Jay Alammar. The Illustrated Transformer.

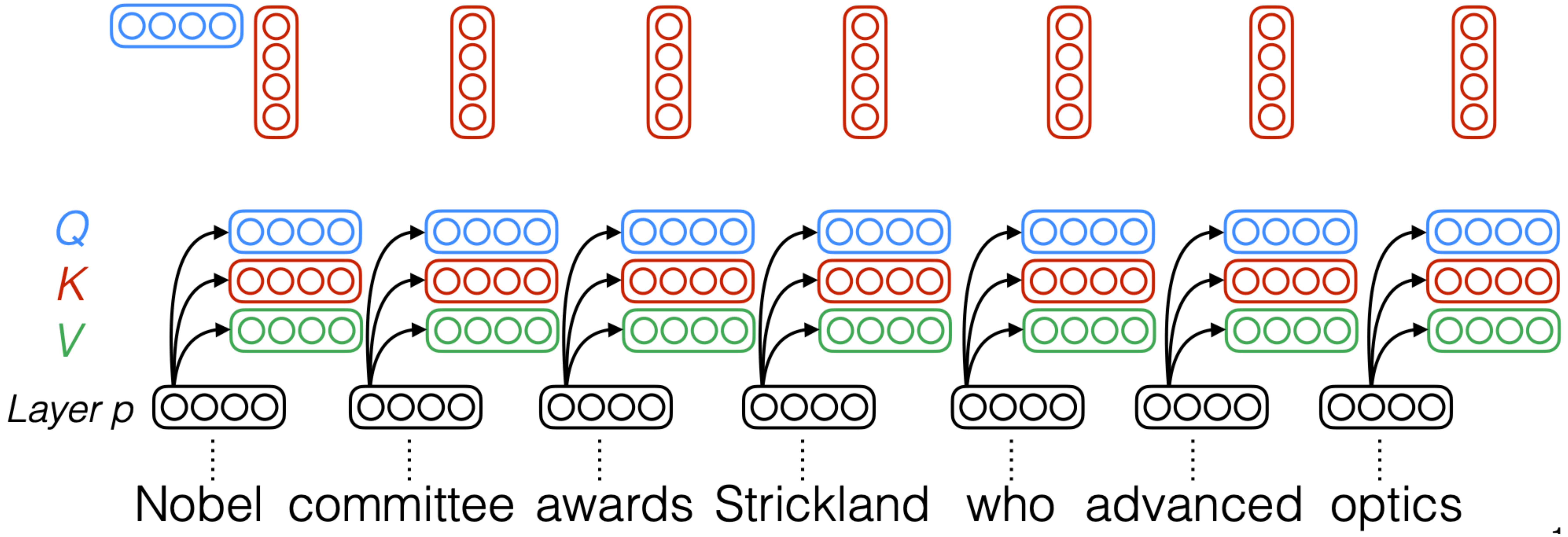
8

Self-Attention

At each step, the attention computation attends to all steps in the input example

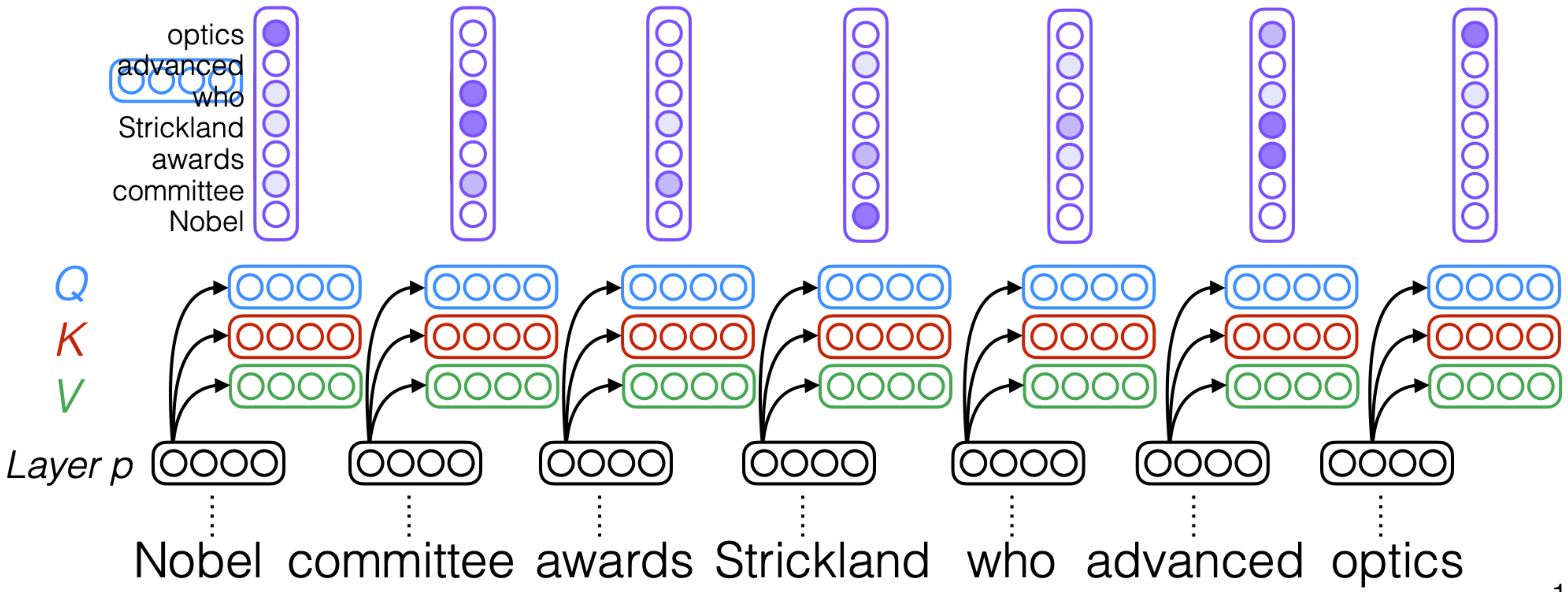


Self-Attention

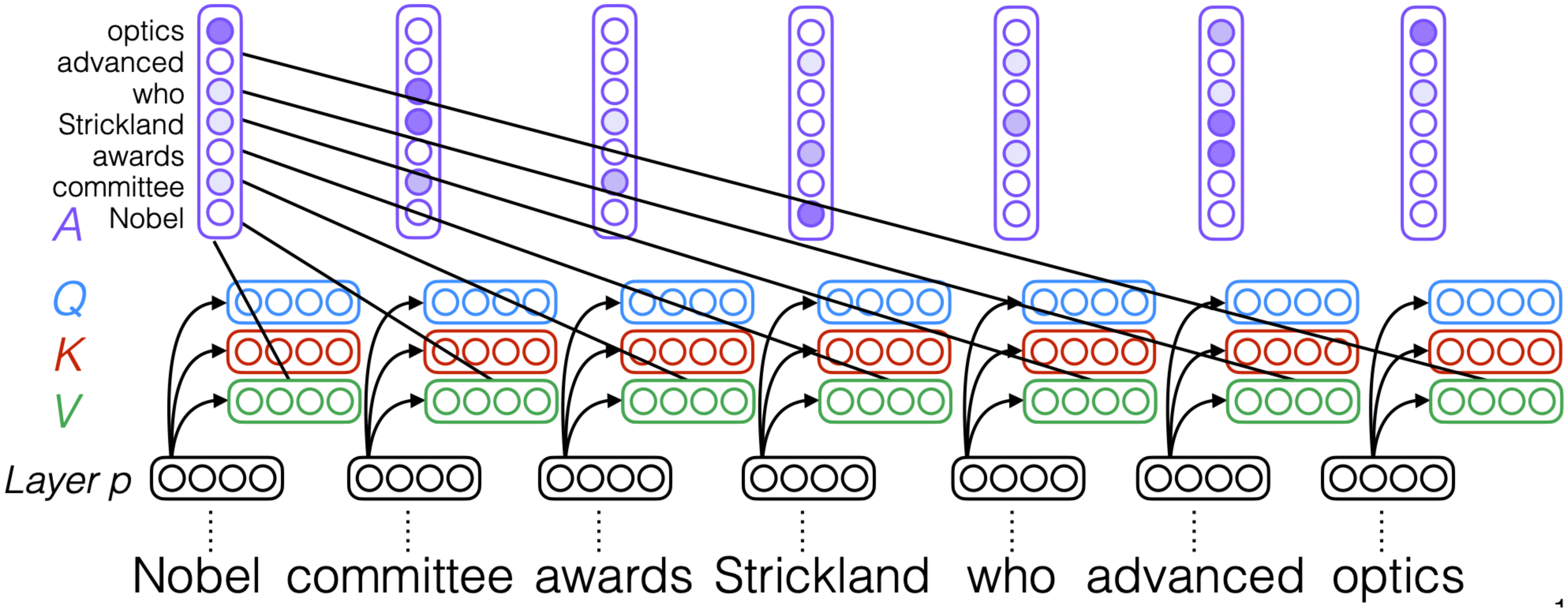


Self-Attention

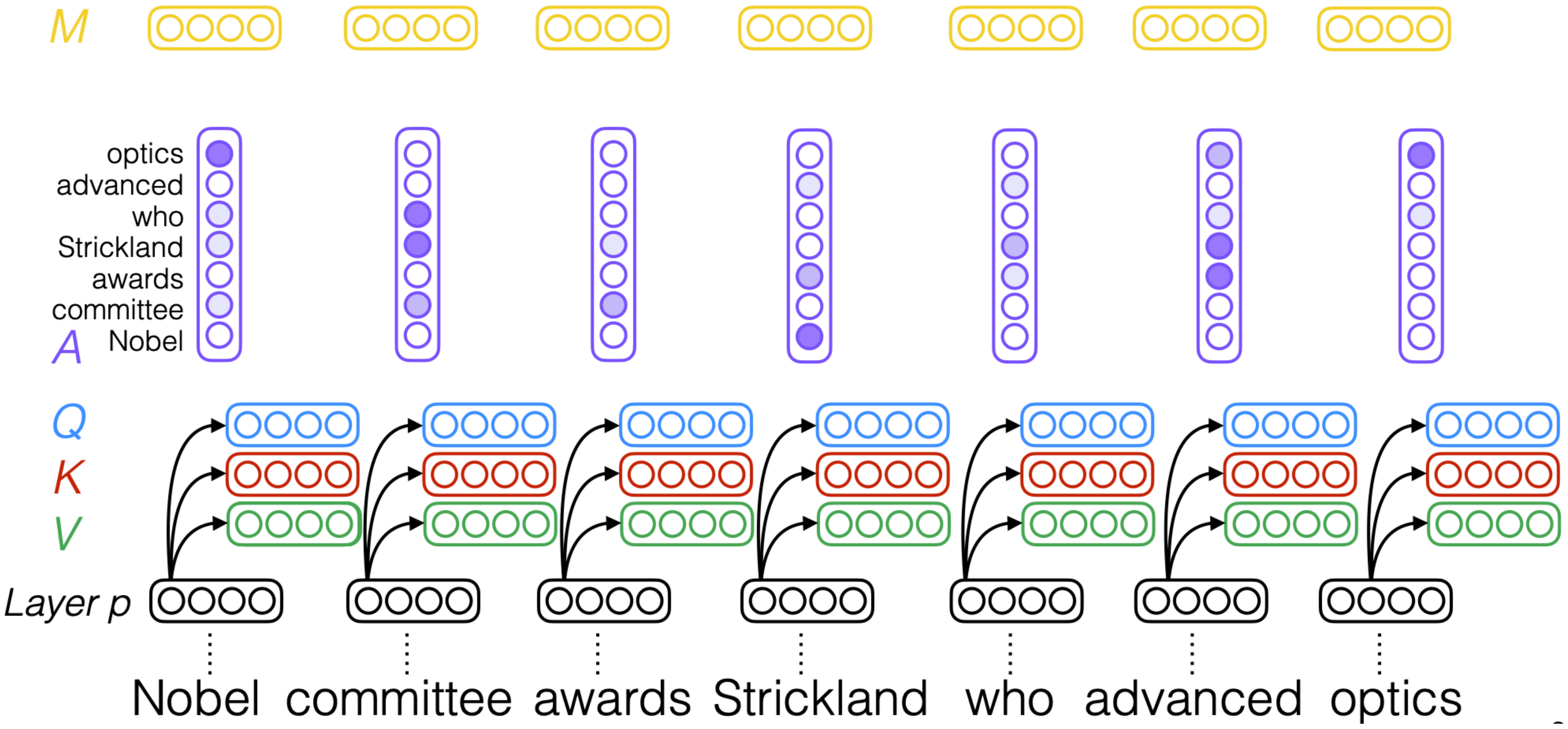
Attention weight on every word in the sequence



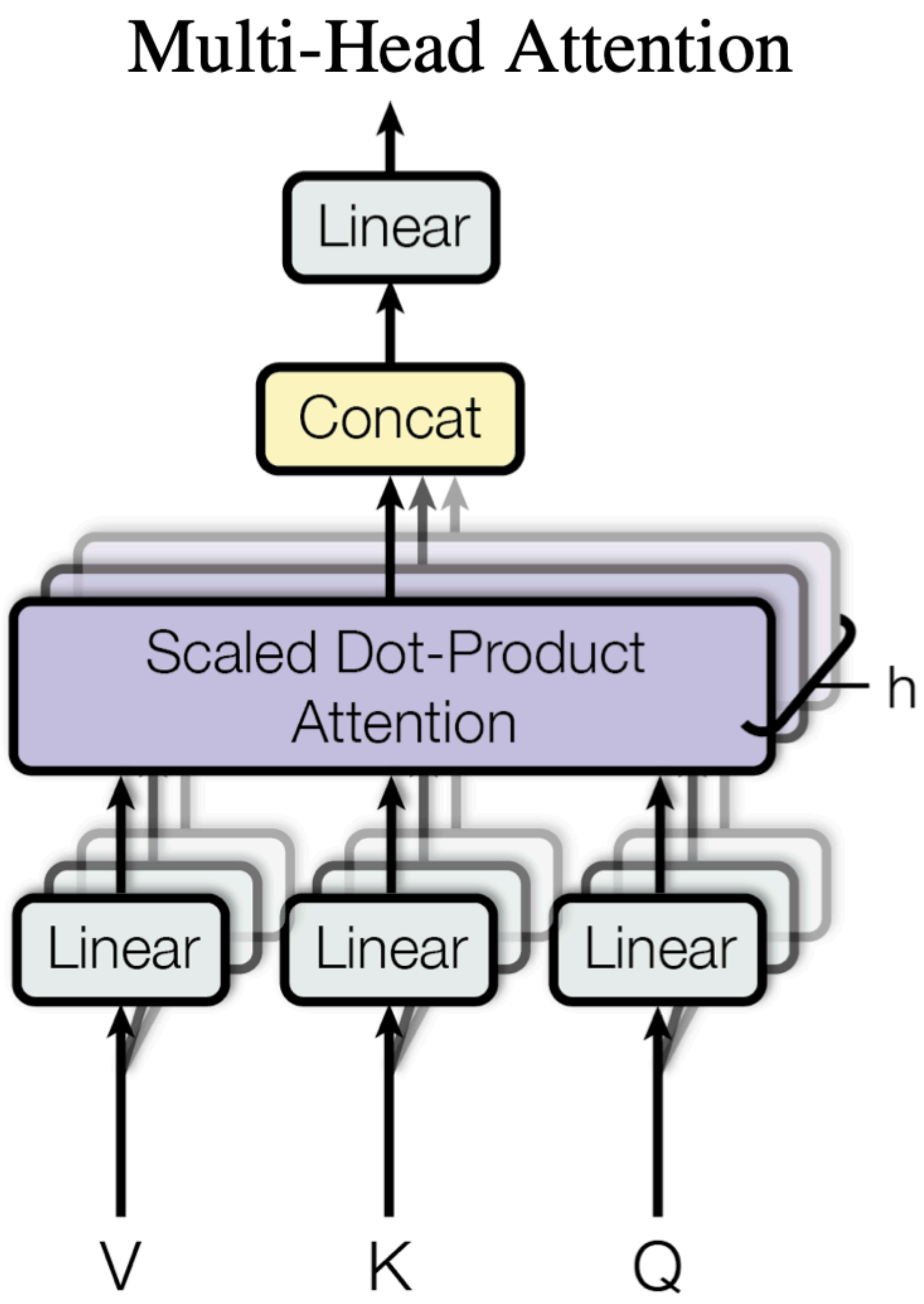
Self-Attention



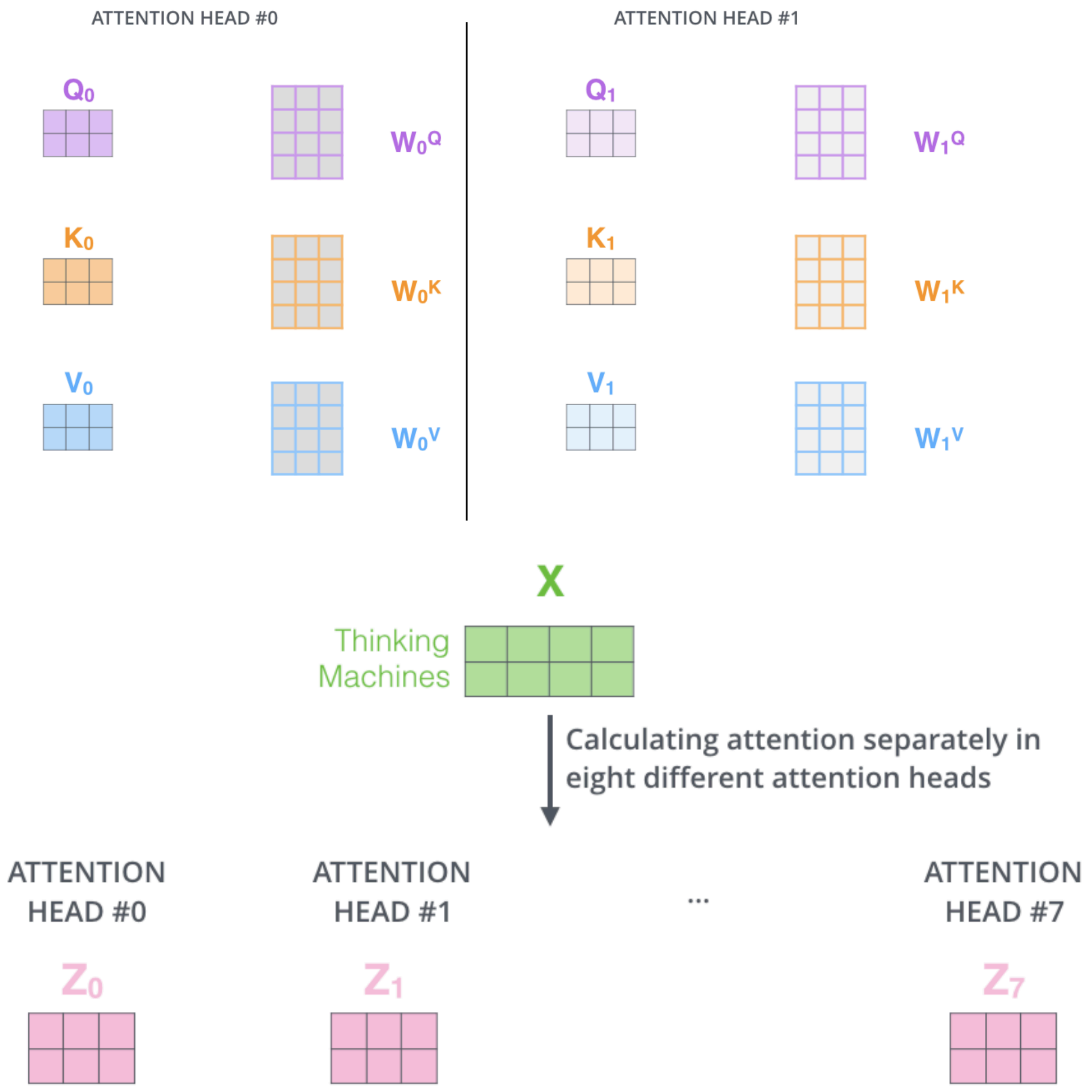
Self-Attention



Multi-Head Attention

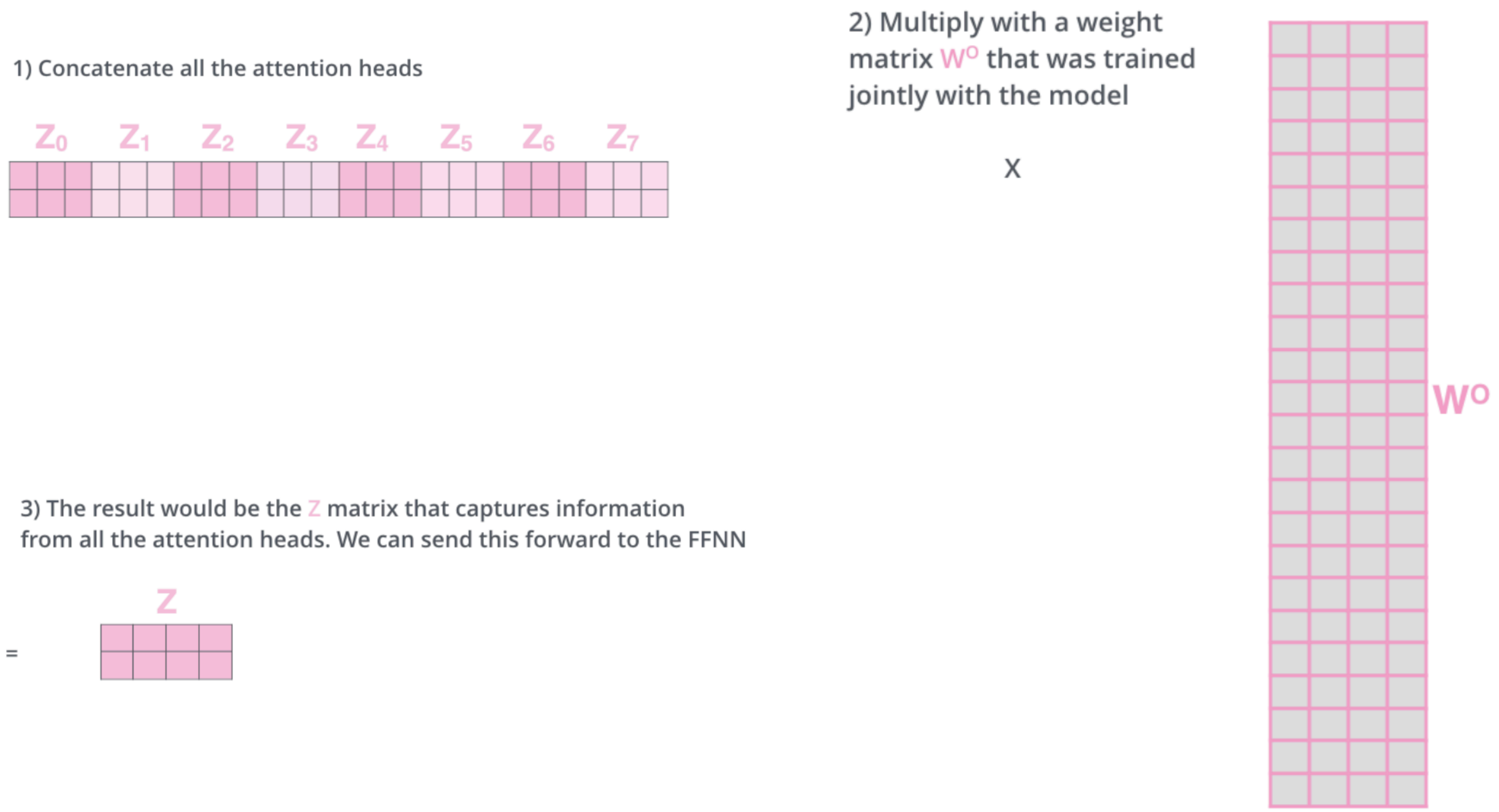


Multi-Head Self-Attention



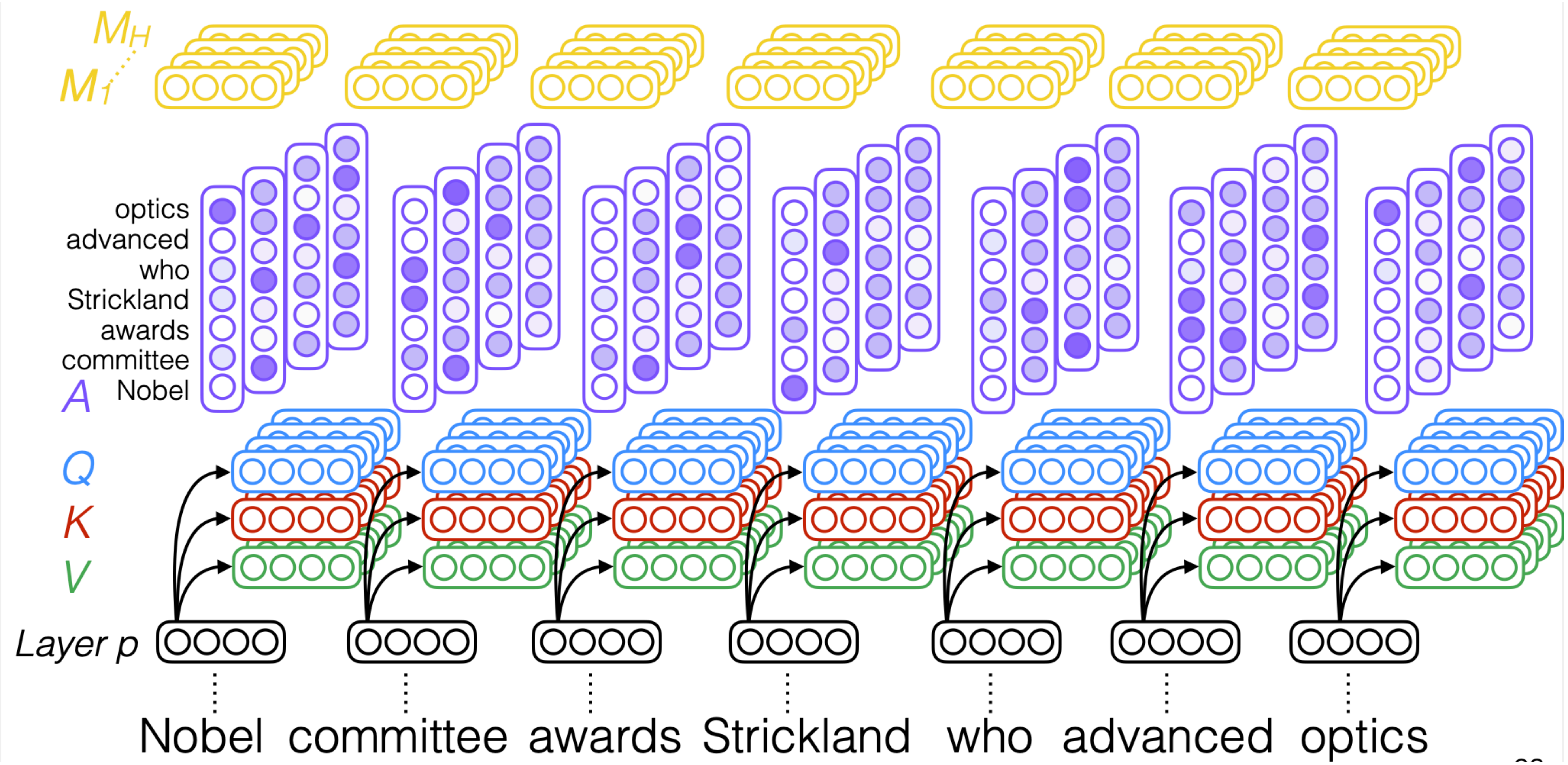
Jay Alammar. The Illustrated Transformer.

Multi-Head Self-Attention



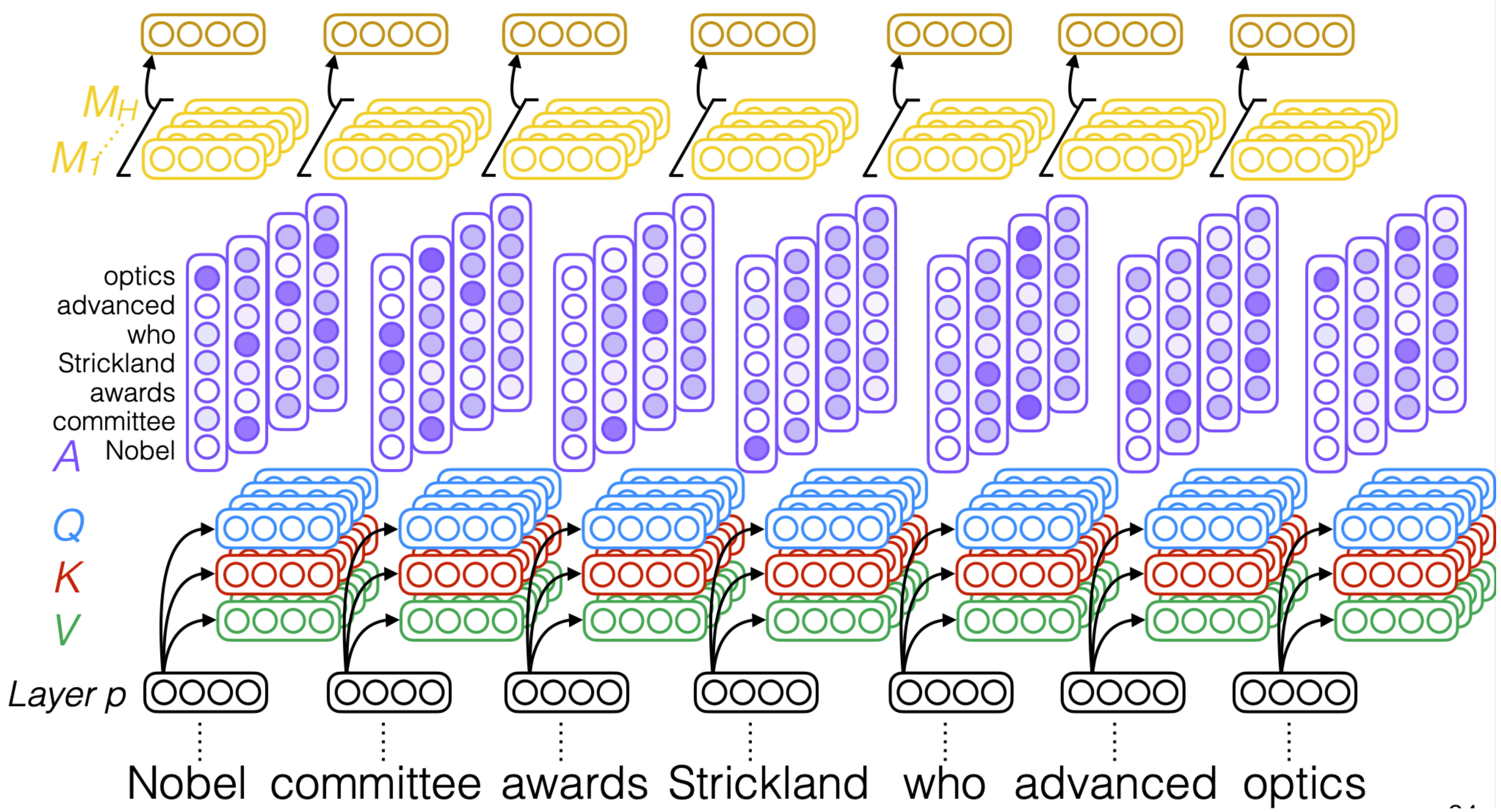
Jay Alammar. The Illustrated Transformer.

Multi-head Self-Attention



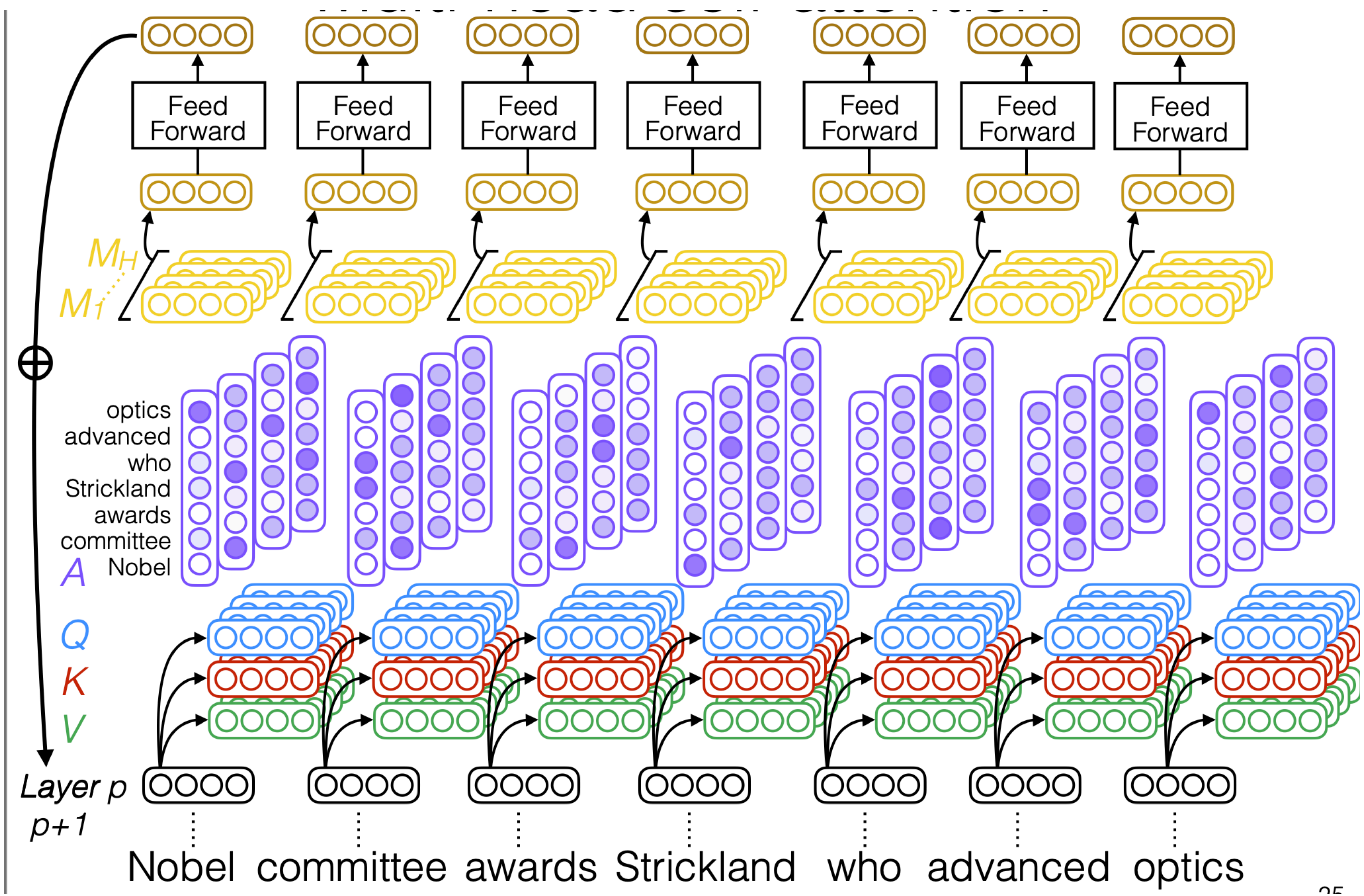
Multi-head Self-Attention

Concat and output projection

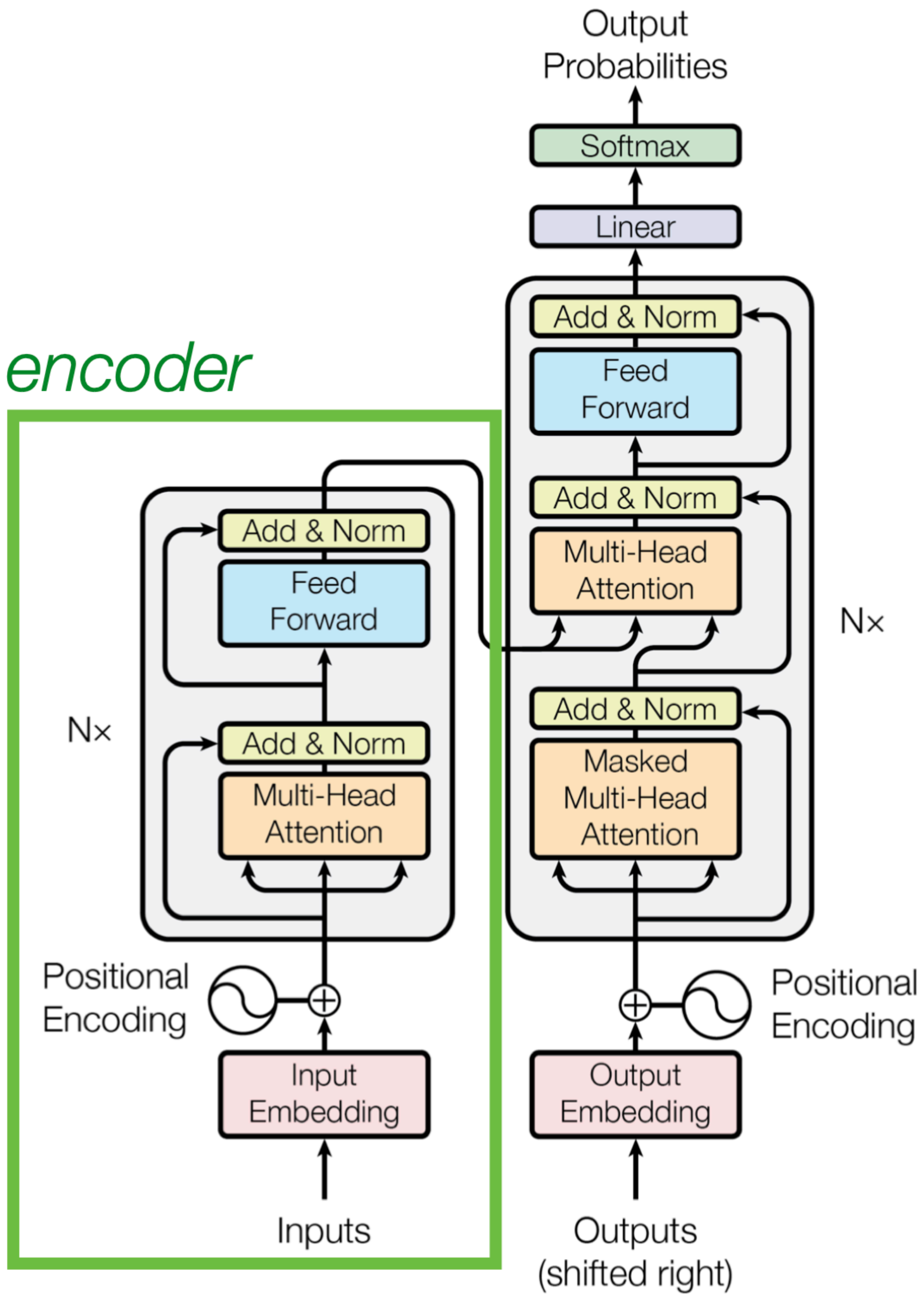


Slides by Emma Strubell

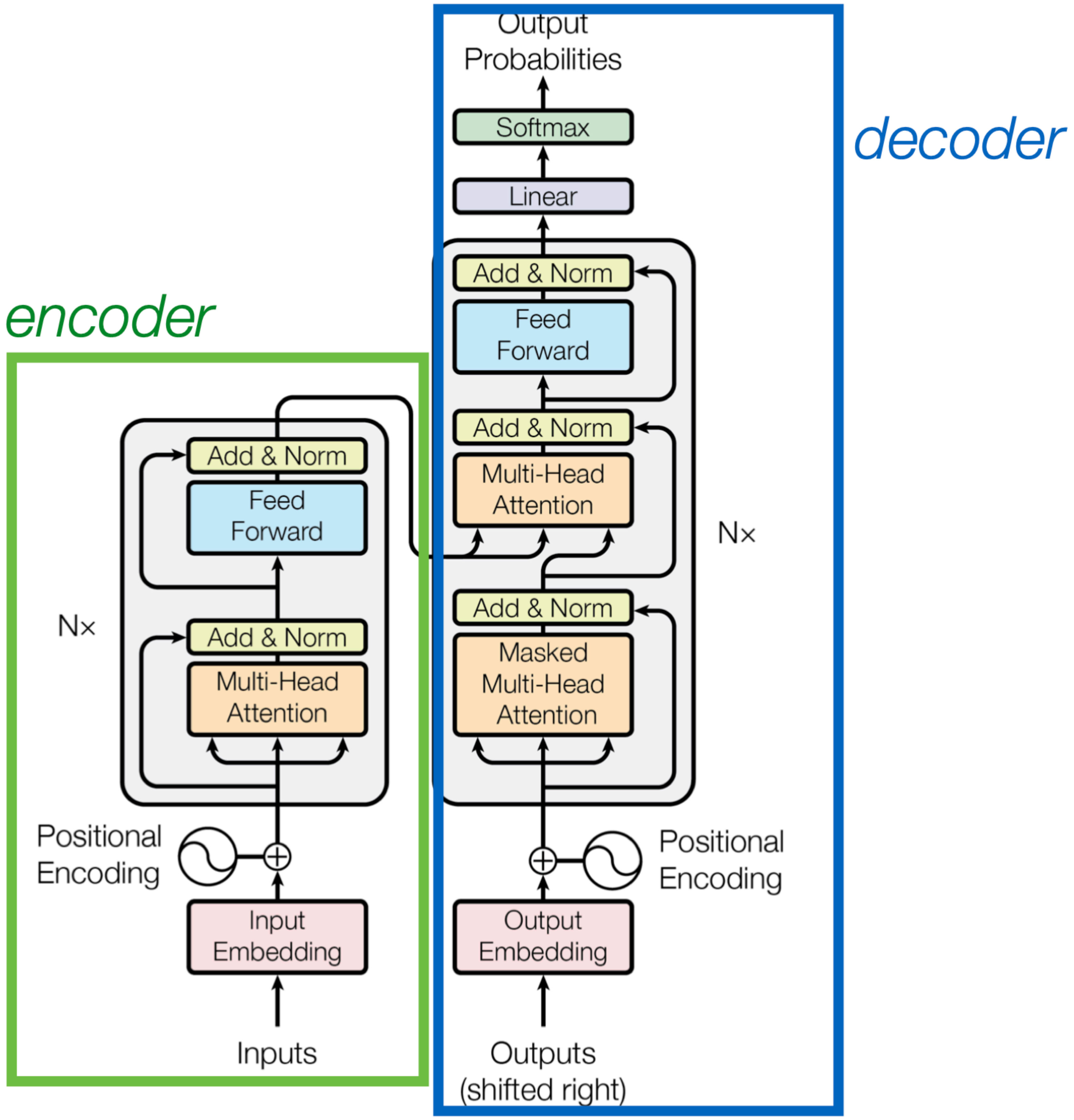
Multi-head Self-Attention + FFN



Transformer Encoder

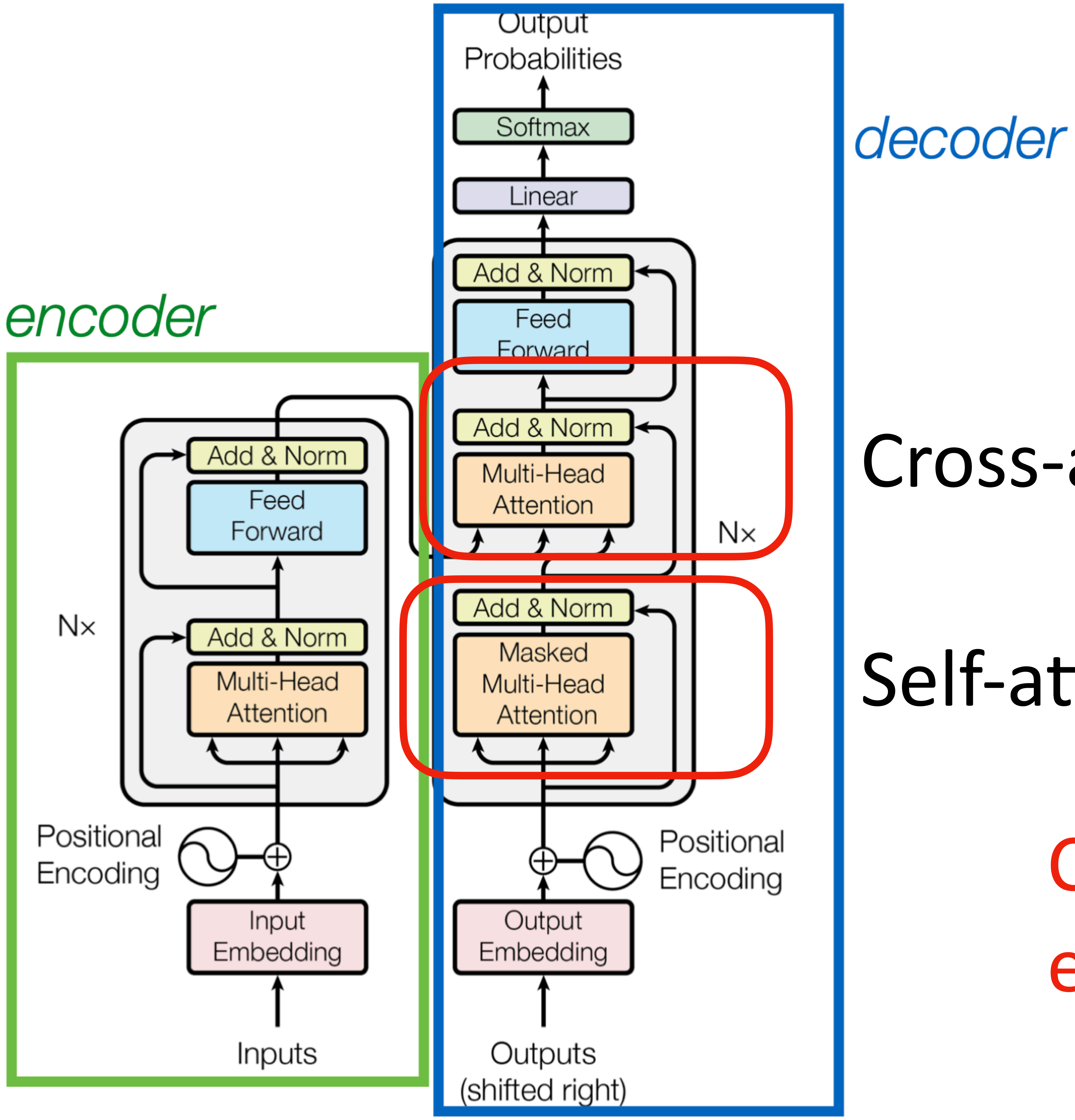


Currently we only cover the encoder side



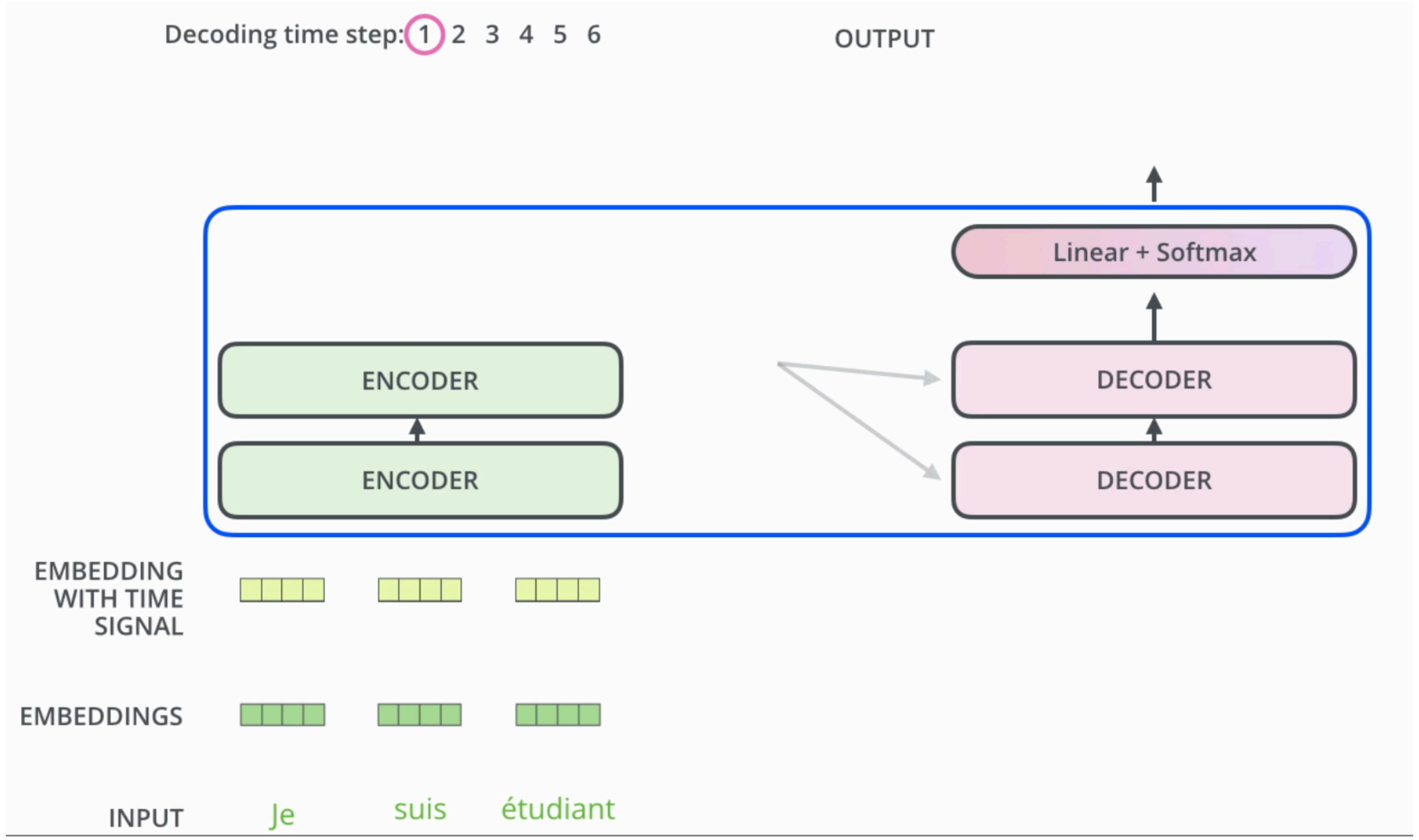
This encoder-decoder arch is originally proposed as a seq2seq arch, for classification tasks, often only encoder is used. And language models often only have a decoder

Transformer Decoder in Seq2Seq



Cross-attention

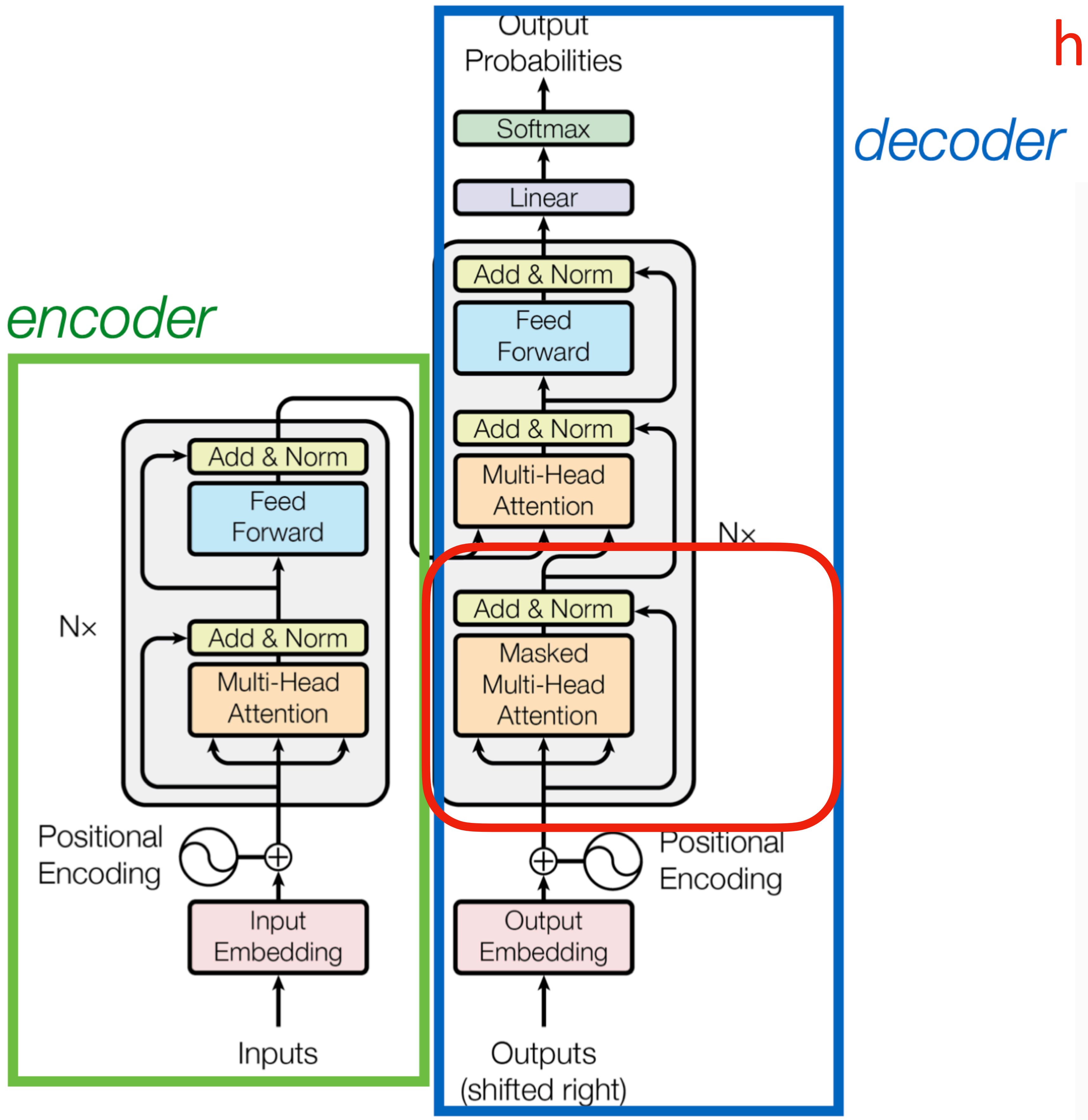
Self-attention

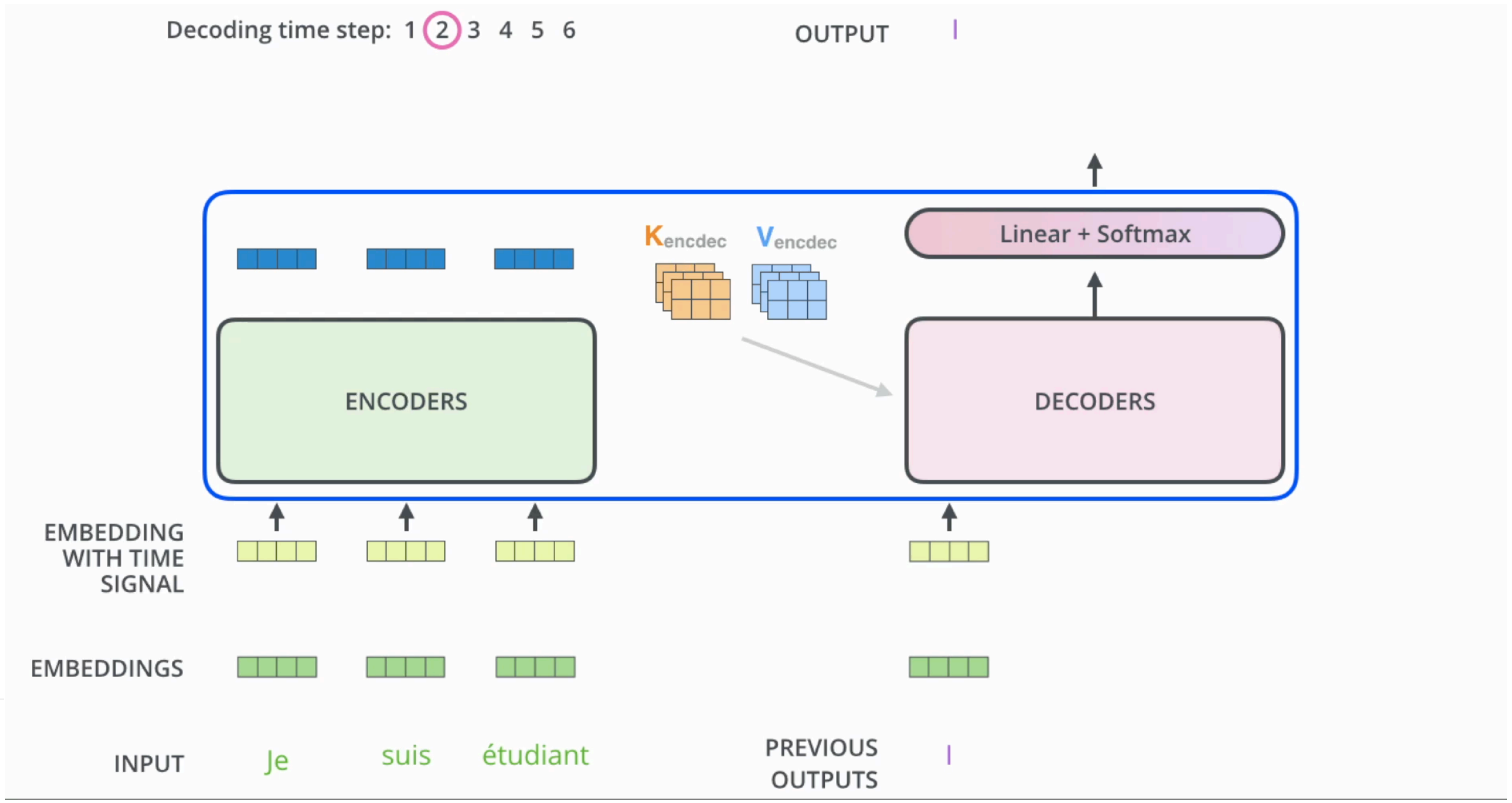


Cross-attention uses the output of encoder as input

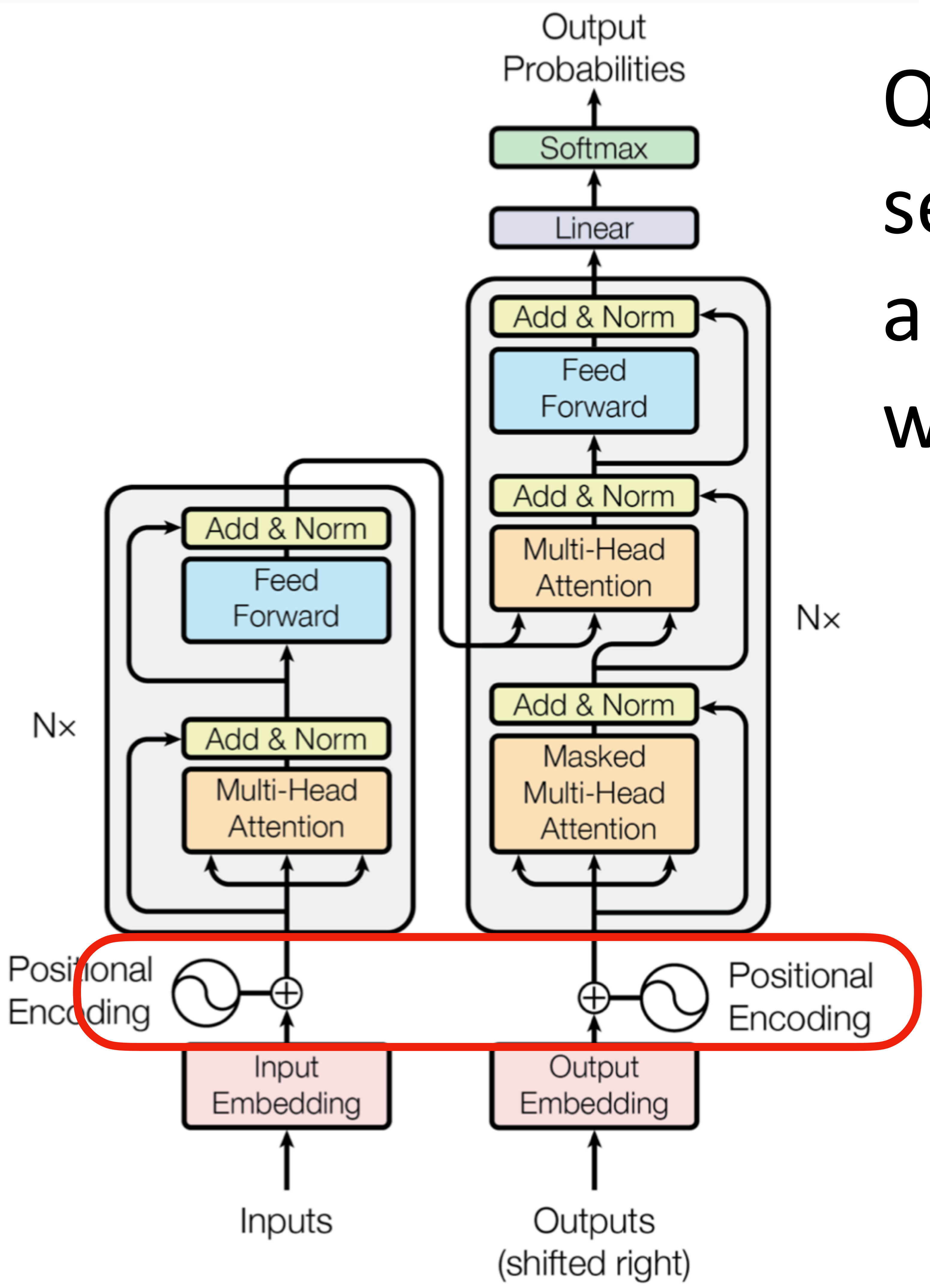
Masked Attention

Typical attention attends to the entire sequence, while masked attention only attends to the ones on the left because future words have not been generated





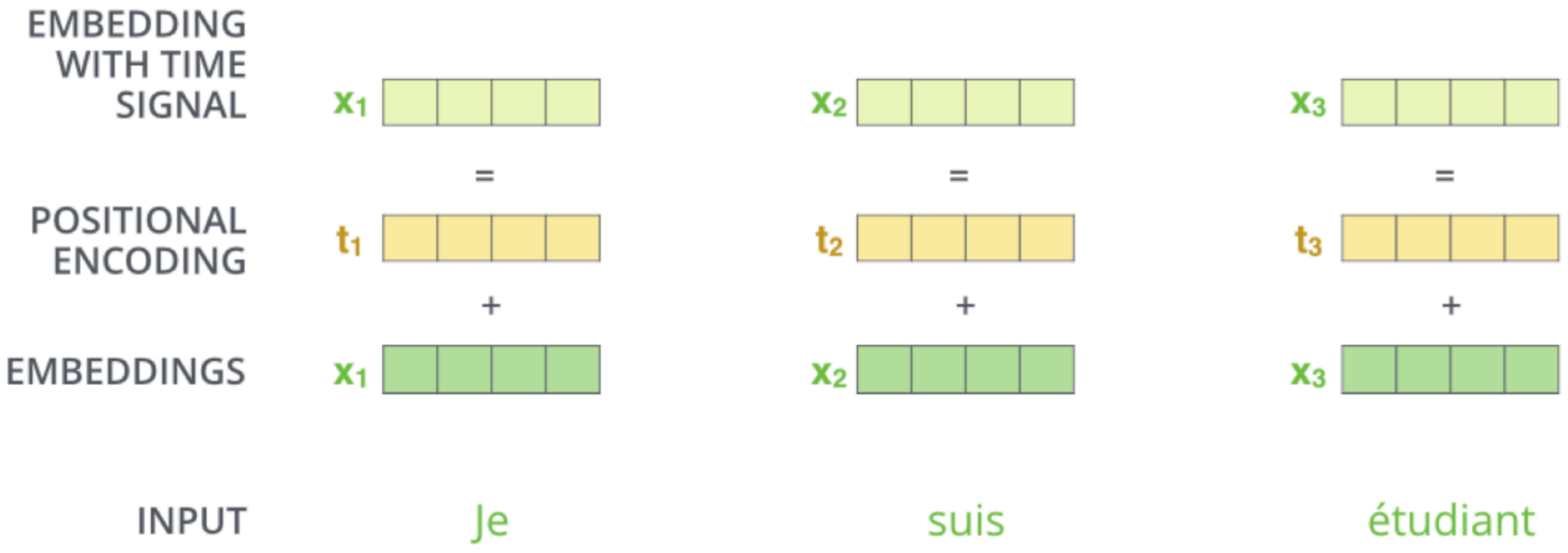
Position Embeddings



Question: If we shuffle the order of words in the sequence, will that change the attention output and feed forward output of the corresponding word?

Position embeddings are added to each word embedding, otherwise our model is unaware of the position of a word

Positional Encoding



Transformer Positional Encoding

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

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

Positional encoding is a 512d vector
i = a particular dimension of this vector
pos = dimension of the word
d_model = 512

Complexity

Layer Type	Complexity per Layer	Sequential Operations
Self-Attention	$O(n^2 \cdot d)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$

n is sequence length, d is embedding dimension.

Restricted self-attention means not attending all words in the sequence, but only a restricted field

Square complexity of sequence length is a major issue for transformers to deal with long sequence

Auto-Encoding Variational Bayes

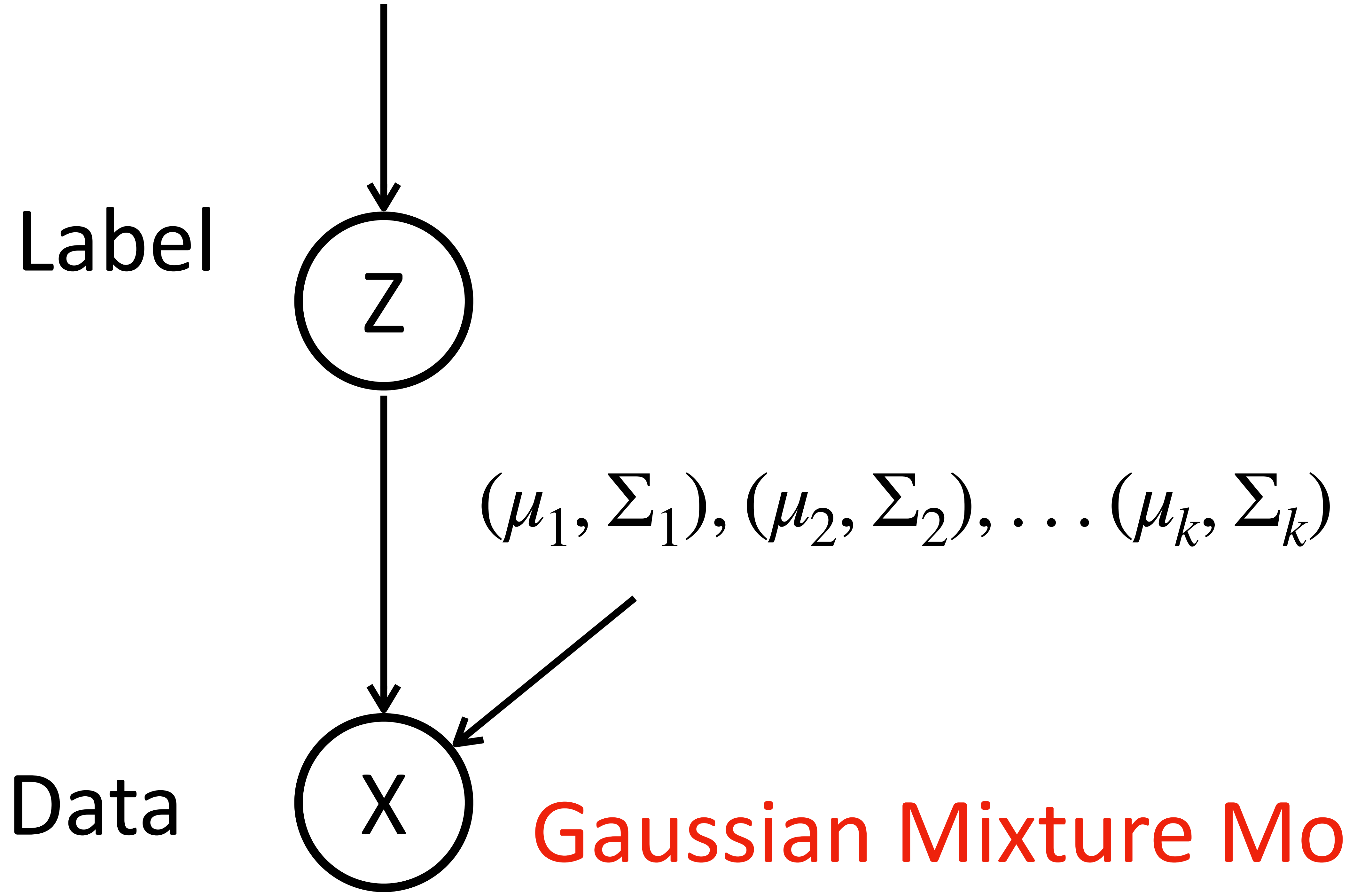
Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Variational Autoencoders

VAE is a Generative Model

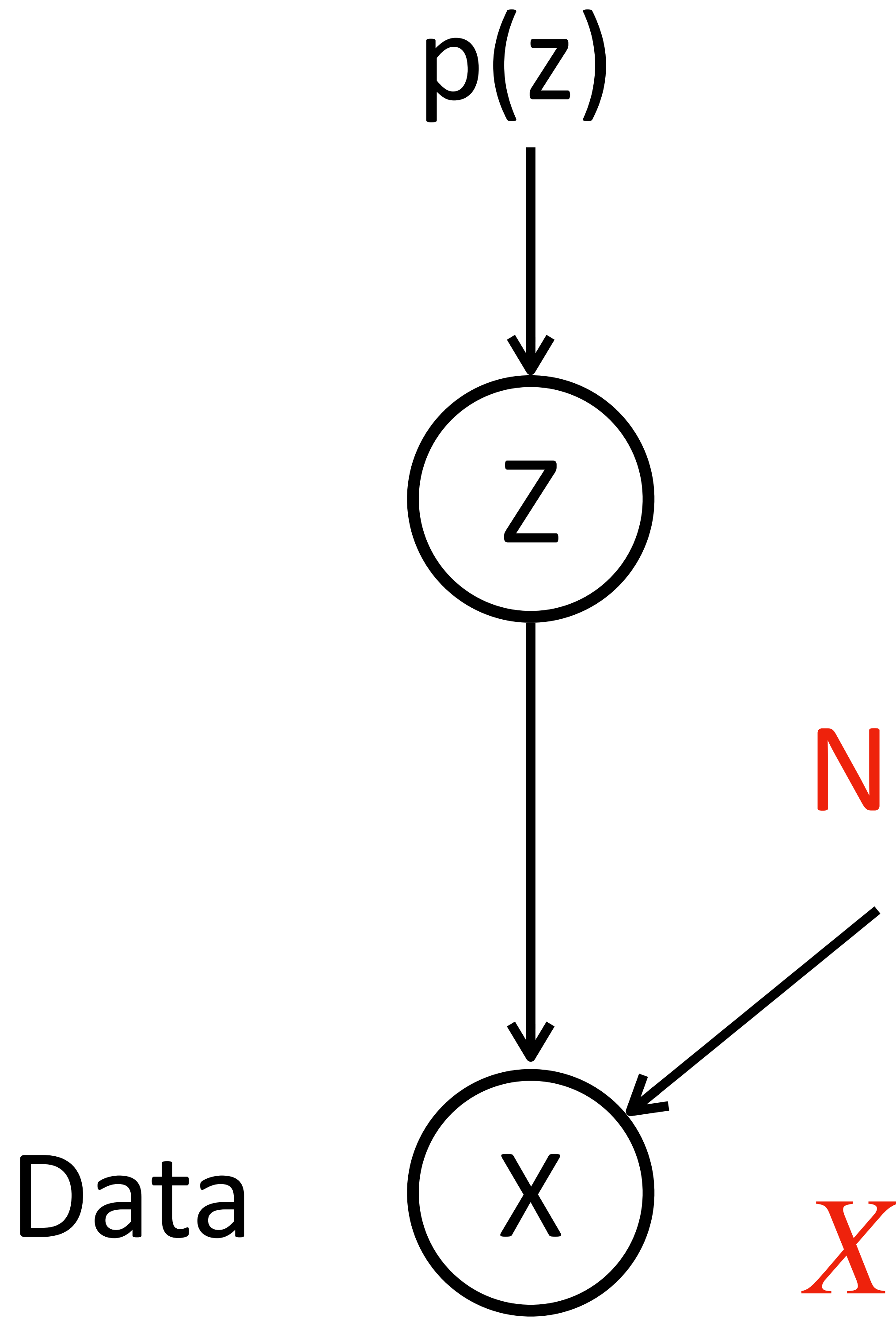
p(z): multinomial , k
classes(e.g. uniform)



Gaussian Mixture Model (GMM)

The VAE Model

p(z) is a normal distribution in most cases



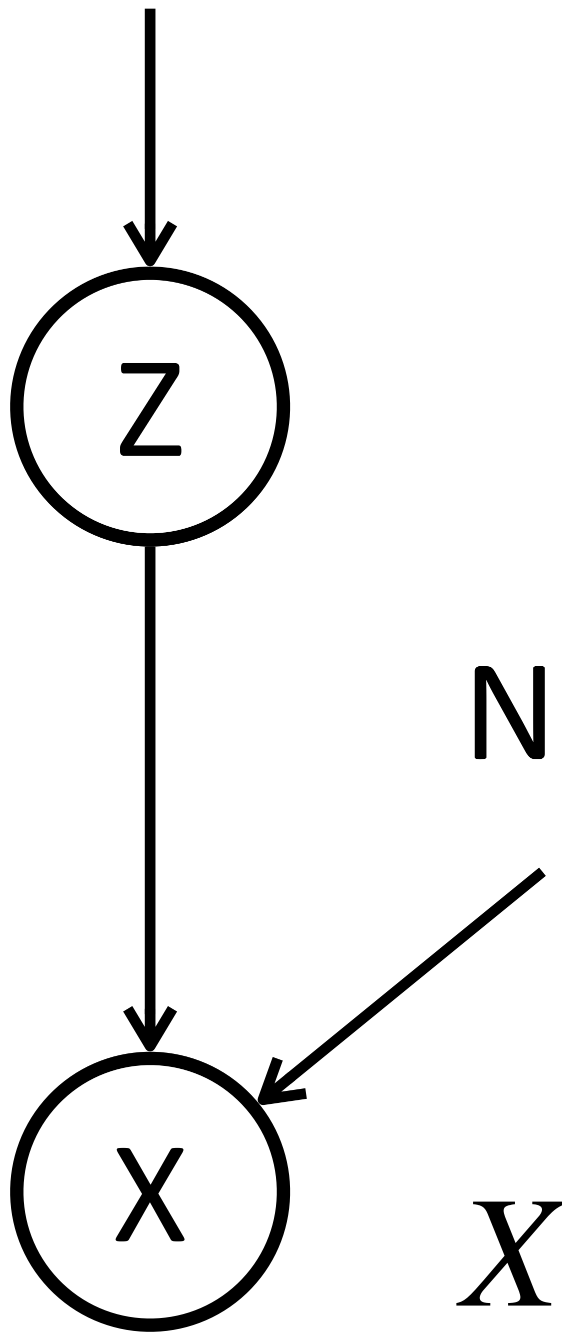
Neural Networks

$$X \sim P(x, f(z; \theta))$$

f is a neural network taking Z as input

Training

p(z)



Neural Networks

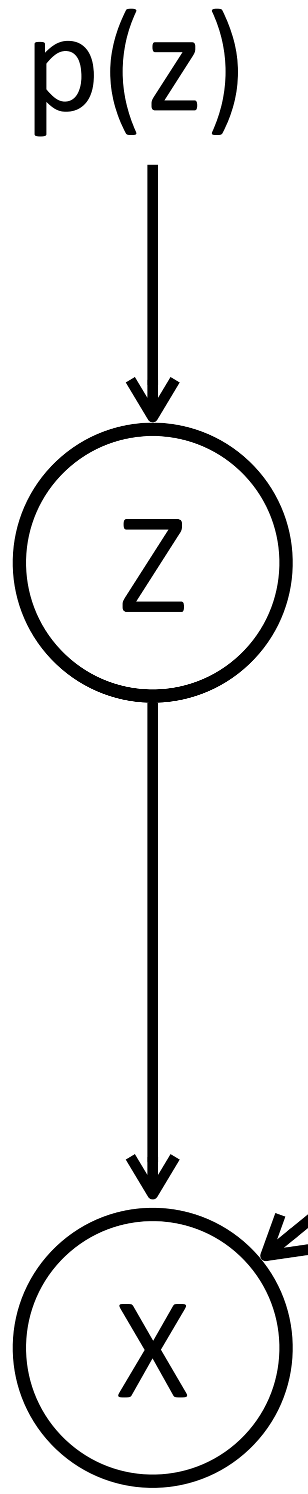
Data

$$X \sim P(x, f(z;\theta))$$

How to train the model? Can we do MLE?

Intractable P(X), EM algorithm?

Let's try EM



Neural Networks

$$X \sim P(x, f(z;\theta))$$

E-Step: compute P(z|x)

$$Q(z) = P(z|x) \propto P(z)P(x|z)$$

This is ok?

M-Step: the ELBO objective

$$\text{argmax}_\theta \sum_z Q(z) \log p(x, z; \theta) = \text{argmax}_\theta \mathbb{E}_{z \sim Q(z)} \log p(x, z; \theta)$$

In most cases, we cannot do the sum, and cannot easily sample from Q(z) either

Approximate Posterior

We need an easy-to-sample distribution to approximate P(z|x)

$q(z \mid x; \phi)$ to approximate $p(z \mid x; \theta)$ Why conditioned on x?

ϕ is the parameter for the approximate function, θ is the generative model parameter

How to train $q(z \mid x; \phi)$, what would be the loss to find ϕ?

Recap: ELBO

$$\text{ELBO}(x; Q, \theta) = \sum_z Q(z) \log \frac{p(x, z; \theta)}{Q(z)}$$

What is $\text{argmax}_{Q(z)}$ELBO$(x; Q, \theta)$?

ELBO is maximized when Q(z) is equal to p(z|x)

Therefore, we can approximate the true posterior by maximizing ELBO:

$$\text{argmax}_\phi \sum_z q(z \,|\, x; \phi) \log \frac{p(x, z; \theta)}{q(z \,|\, x; \phi)}$$

Variational Inference

Training VAEs

E-Step:

$$\text{argmax}_{\phi} \sum_z q(z|x;\phi) \log \frac{p(x,z;\theta)}{q(z|x;\phi)}$$

M-Step:

$$\text{argmax}_{\theta} \sum_z q(z|x;\phi) \log \frac{p(x,z;\theta)}{q(z|x;\phi)}$$

Same objective, different parameters to optimize

Because we use approximate rather than exact posterior, it is also called Variational EM

Training VAEs

E-Step:

$$\text{argmax}_\phi \sum_z q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

Can we do gradient descent over ϕ?

M-Step:

$$\text{argmax}_\theta \sum_z q(z \mid x; \phi) \log \frac{p(x, z; \theta)}{q(z \mid x; \phi)}$$

We use MC sampling to approximate expectation and use gradient descent to optimize θ