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

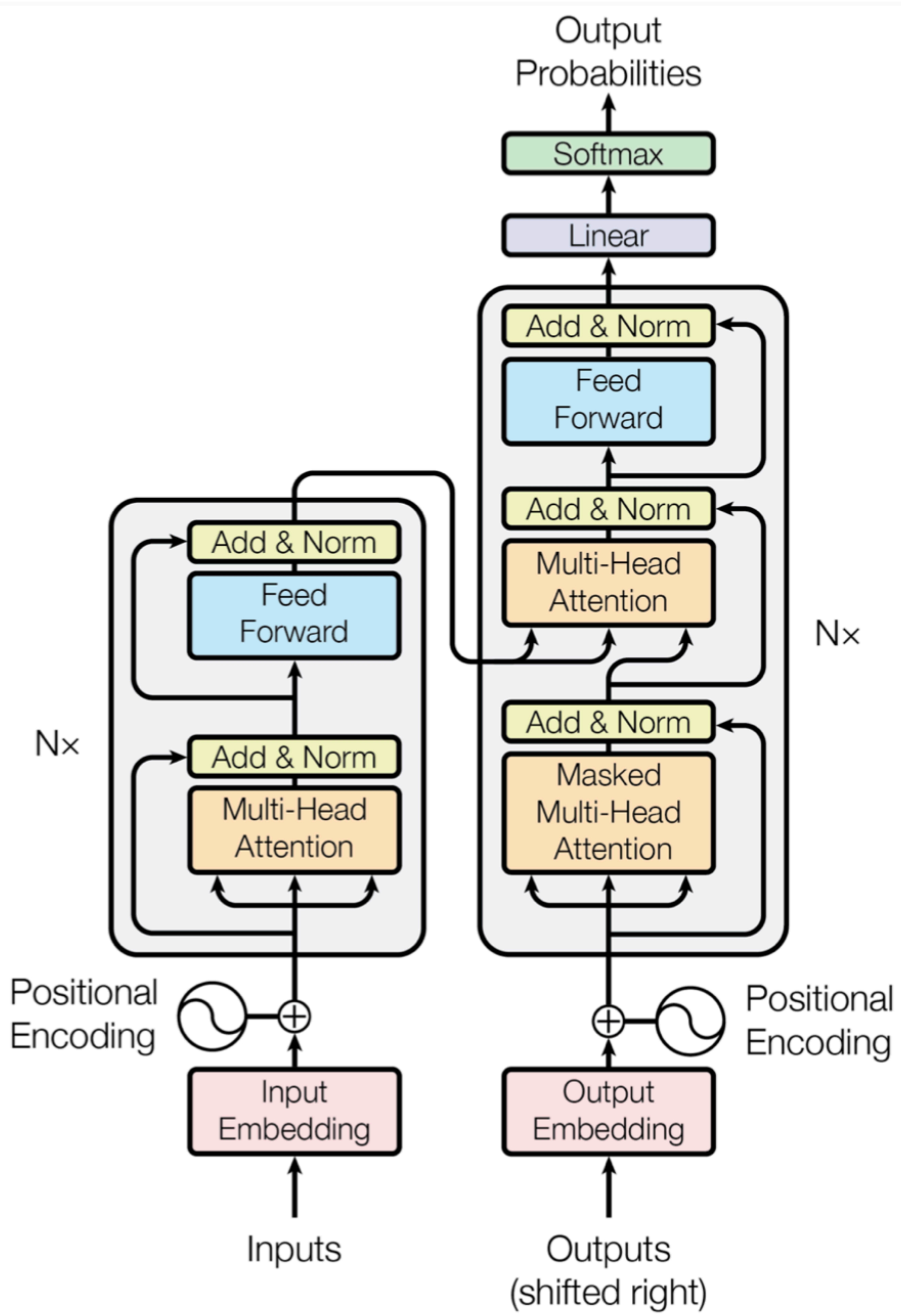
COMP 5212
Machine Learning
Lecture 20

Transformers, VAEs

Junxian He

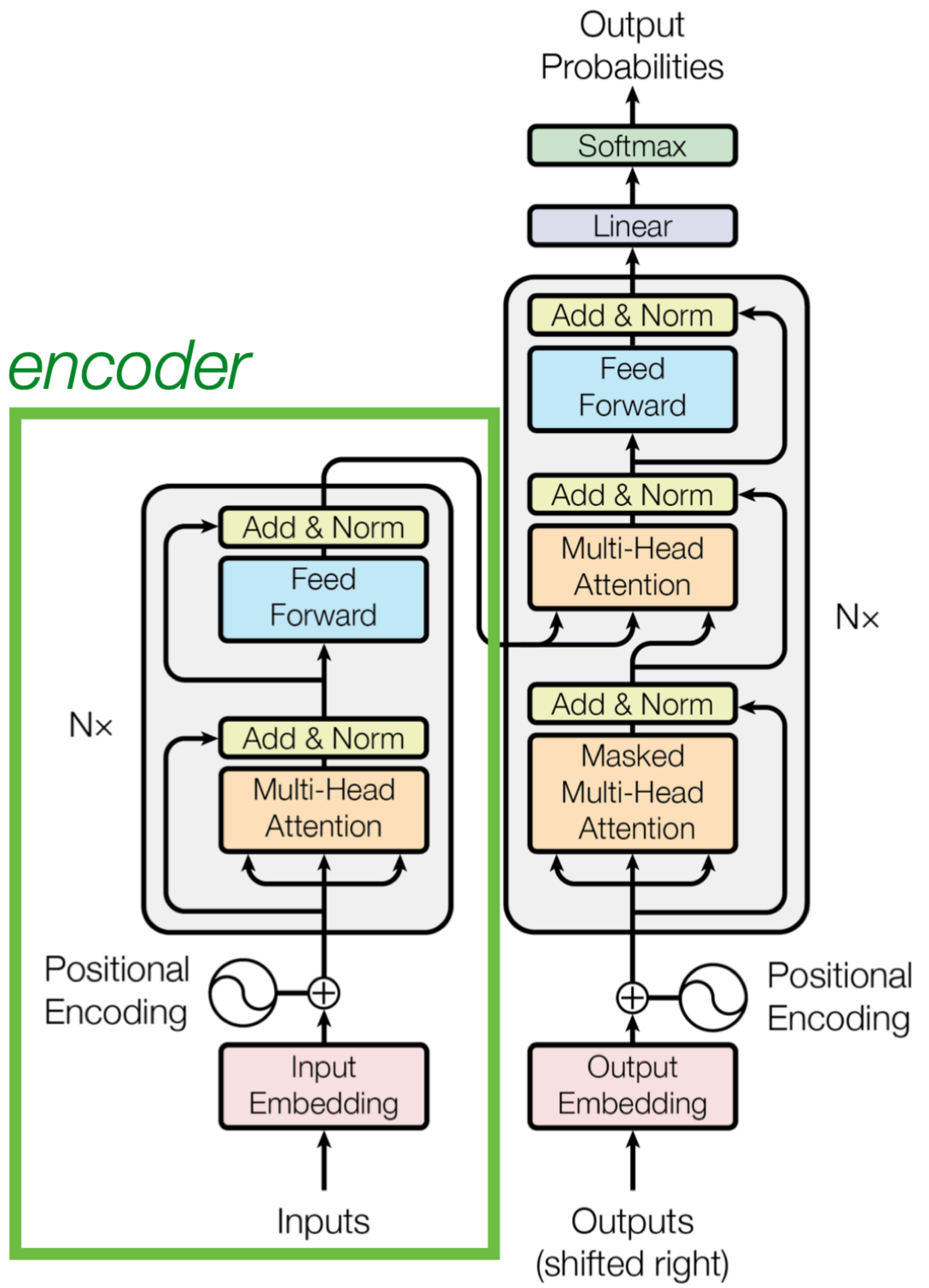
Nov 19, 2024

Transformer



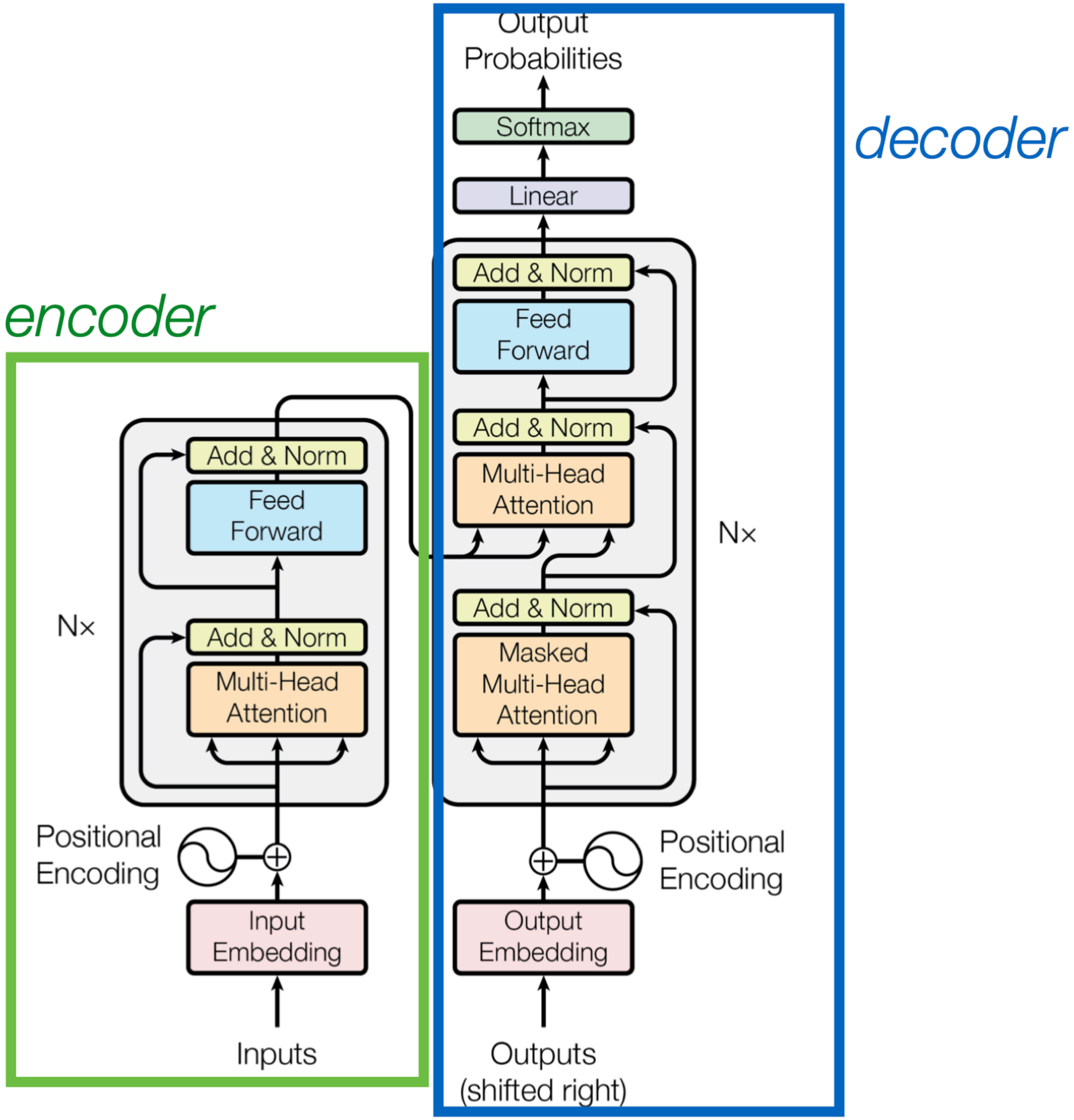
Vaswani et al. Attention is All You Need. NeurIPS 2017.

Encoder



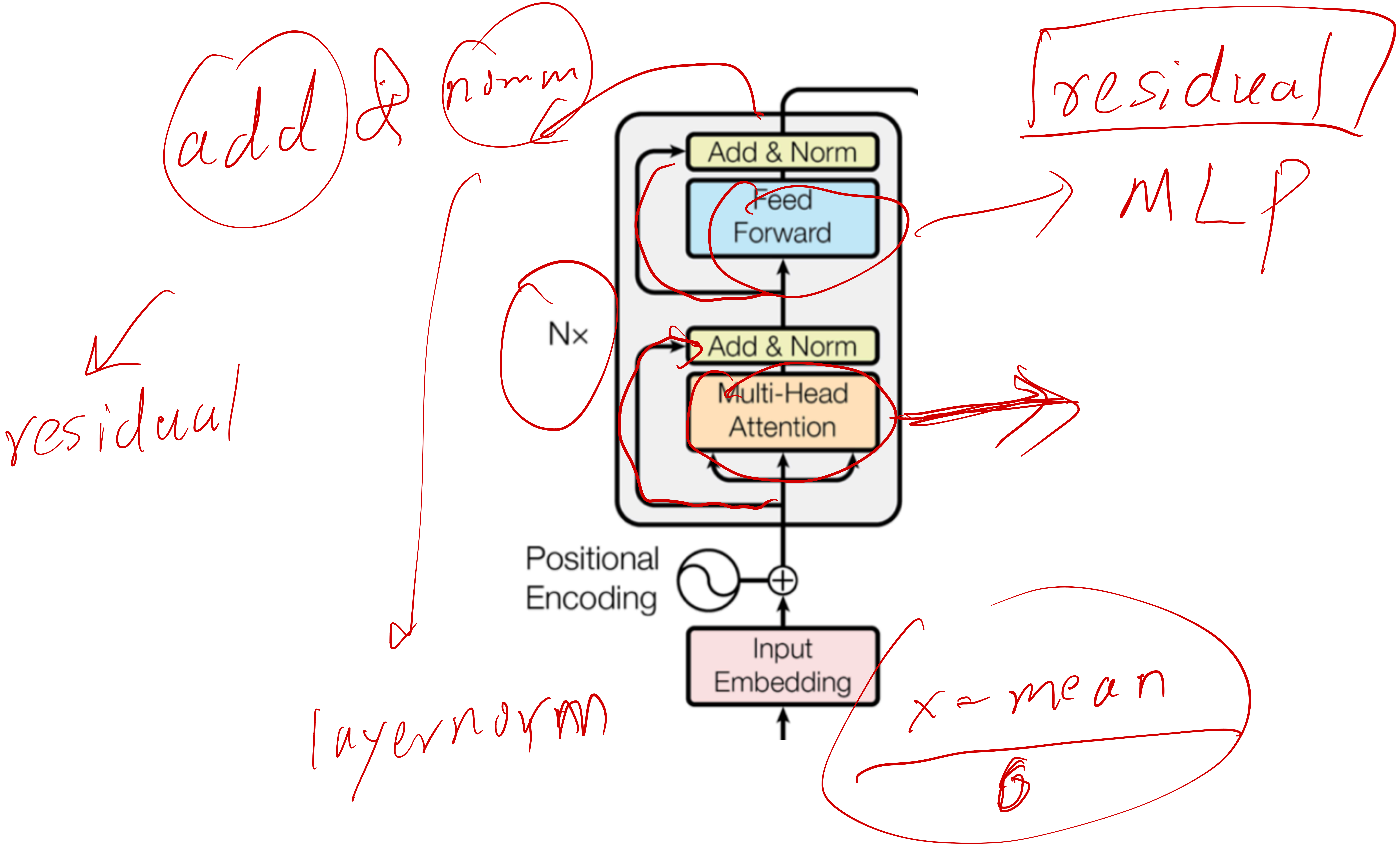
Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
encoder
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

Decoder

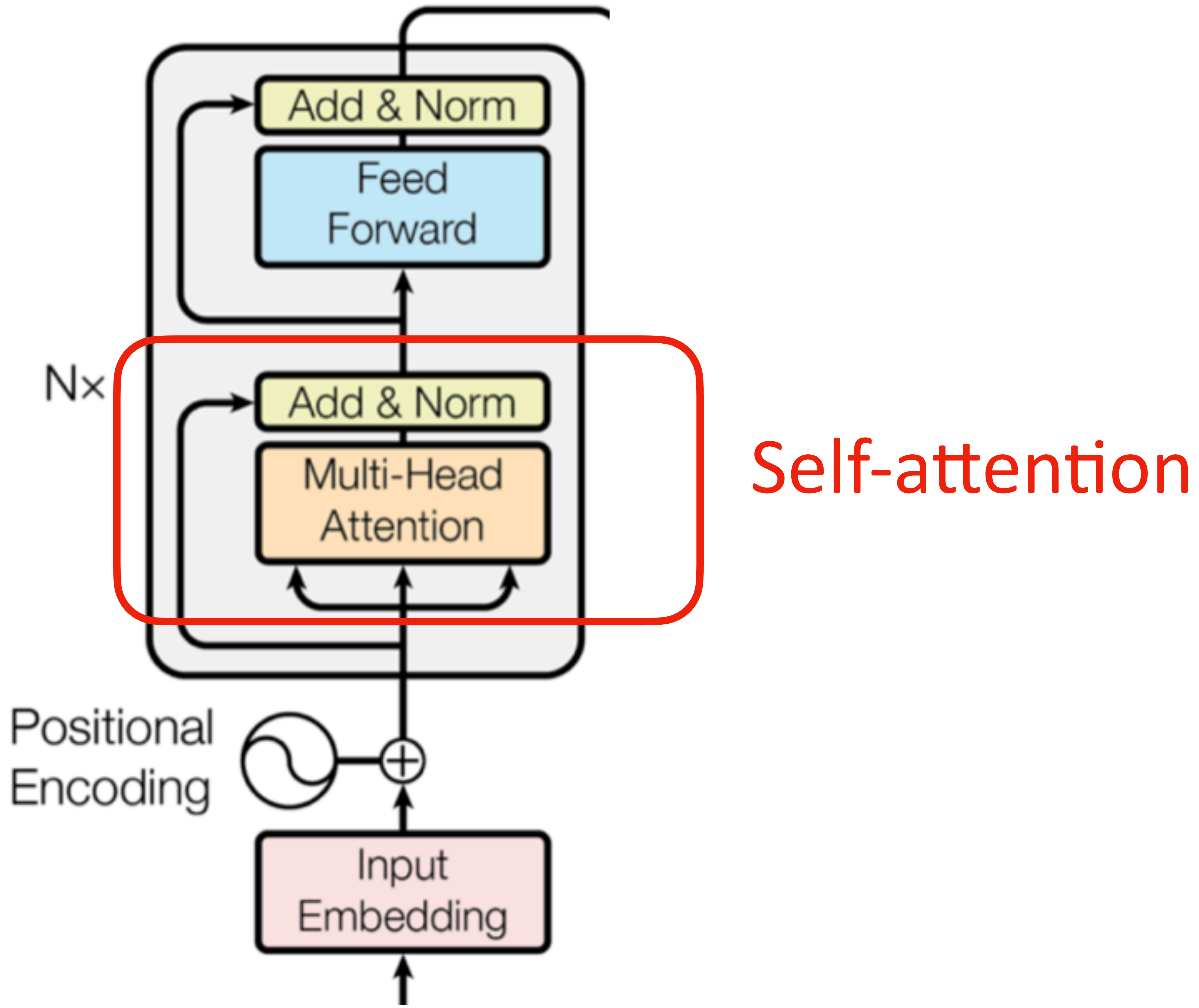


Output
Probabilities
Softmax
decoder
Linear
Add & Norm
Feed
Forward
encoder
Add & Norm
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Nx
Add & Norm
Add & Norm
Masked
Multi-Head
Attention
Multi-Head
Attention
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted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

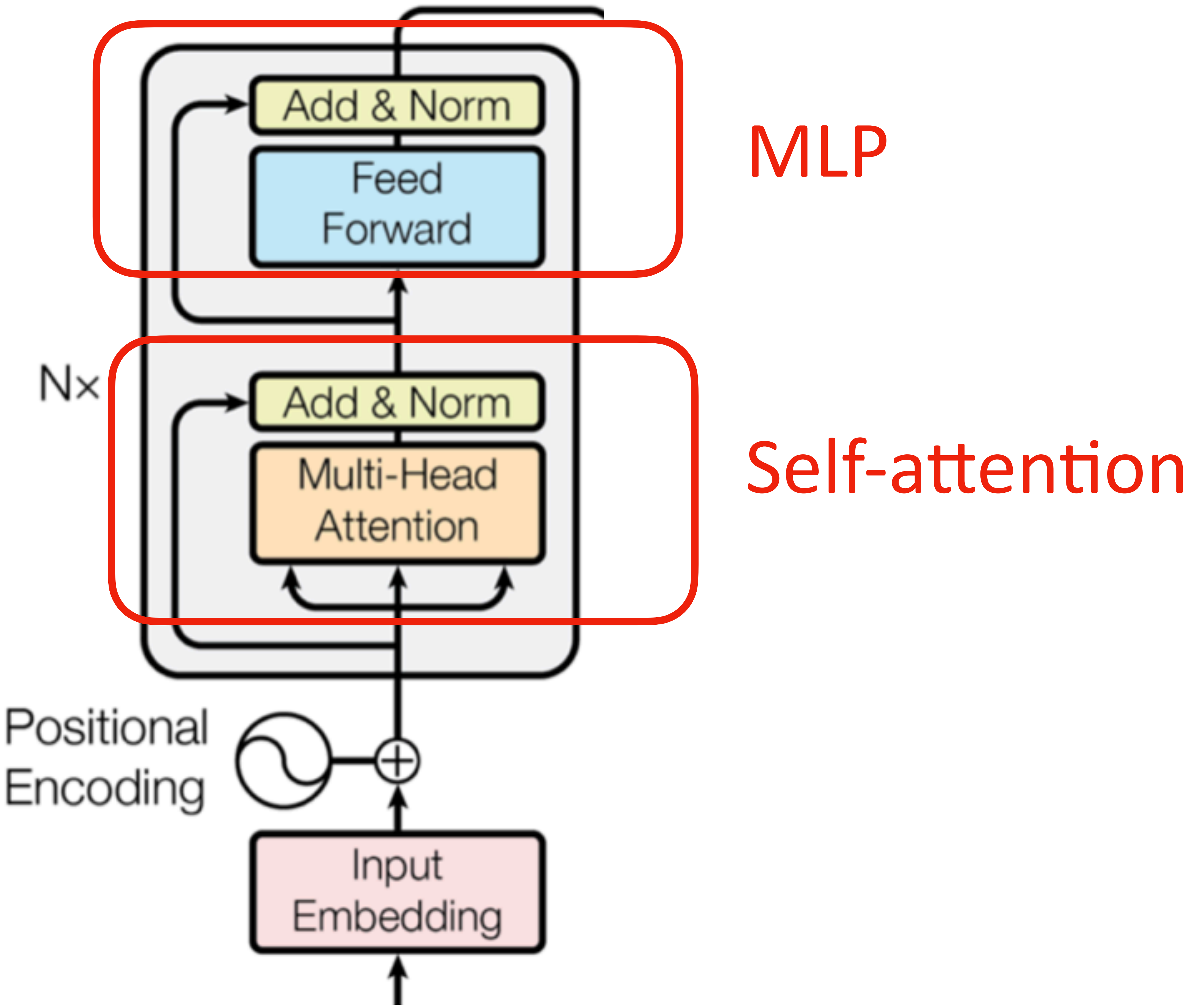
Transformer Encoder



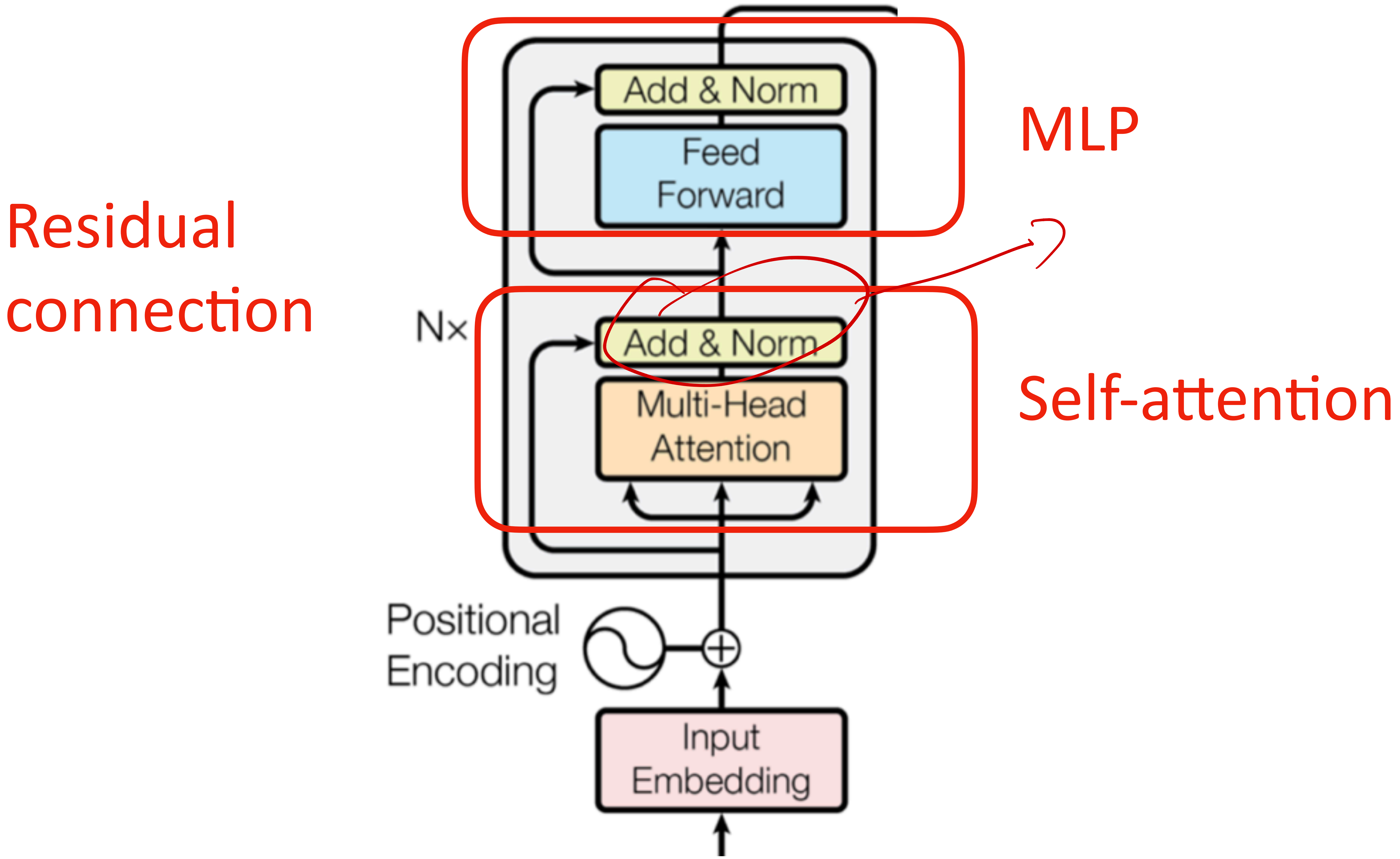
Transformer Encoder



Transformer Encoder

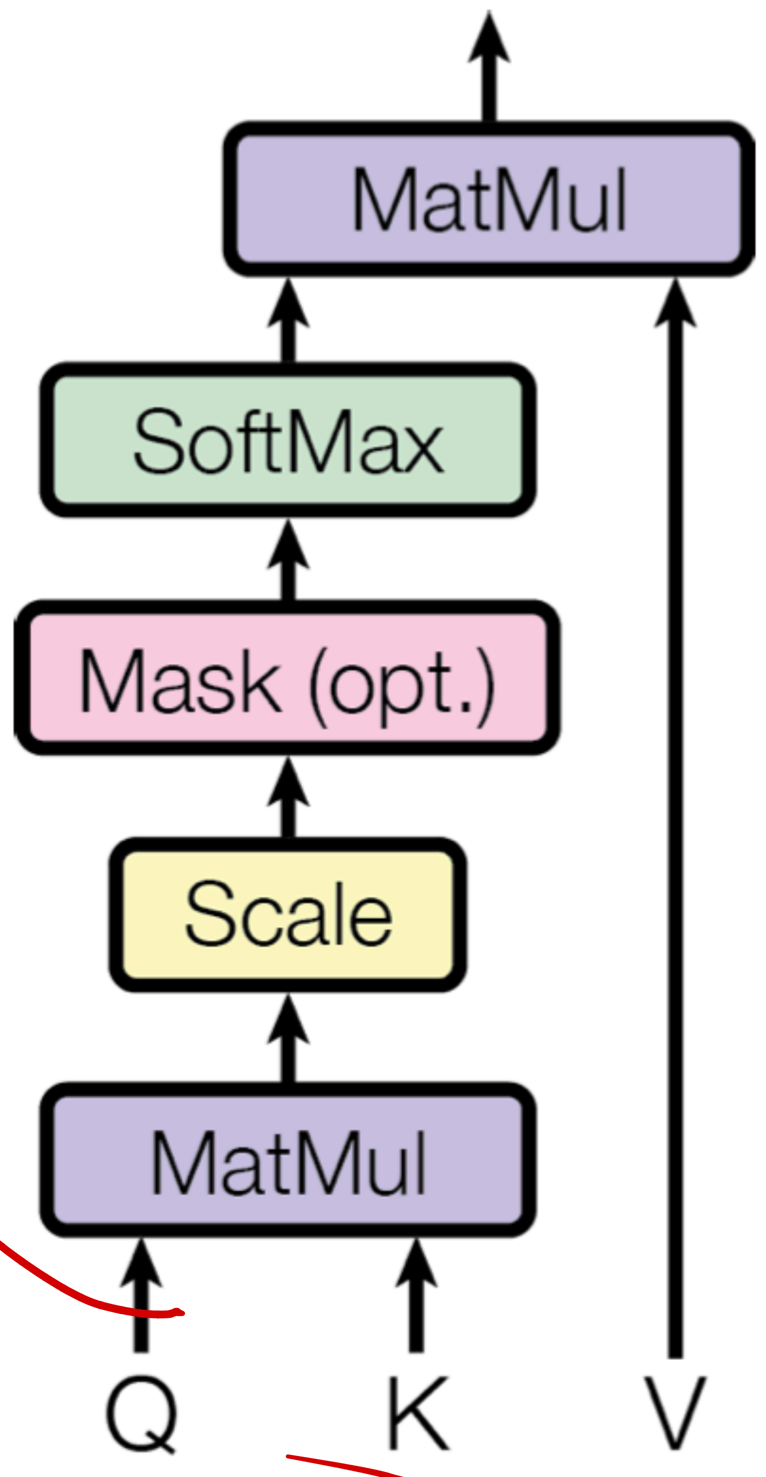


Transformer Encoder



What is Attention

Scaled Dot-Product Attention



Q: Query
K: key
V: value

What is Attention

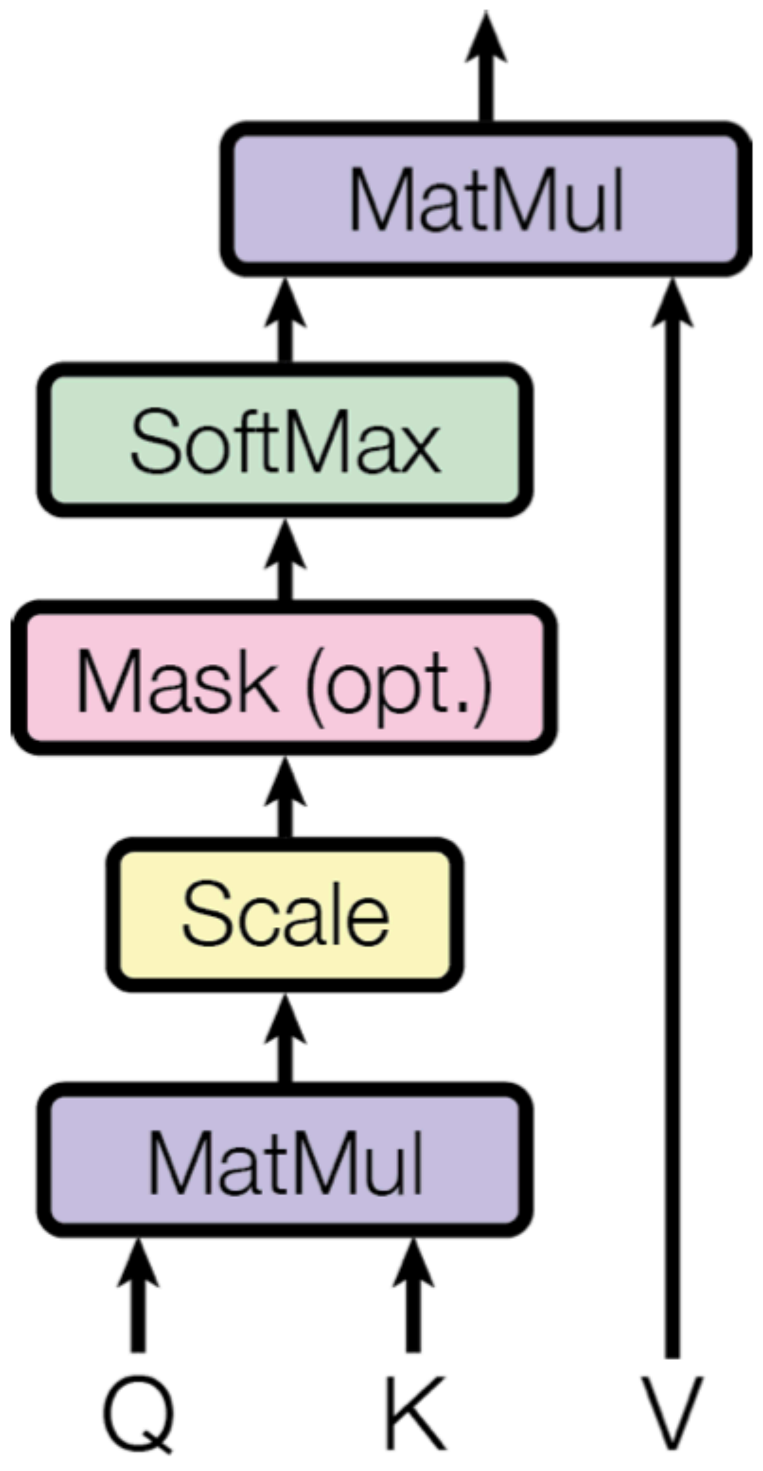
$Q \in R^{n \times d}$ $K \in R^{m \times d}$ $V \in R^{m \times d}$

Q: n different queries, each R^d

K, V: m different (key, value) pairs
R^d

(k, v)

Scaled Dot-Product Attention



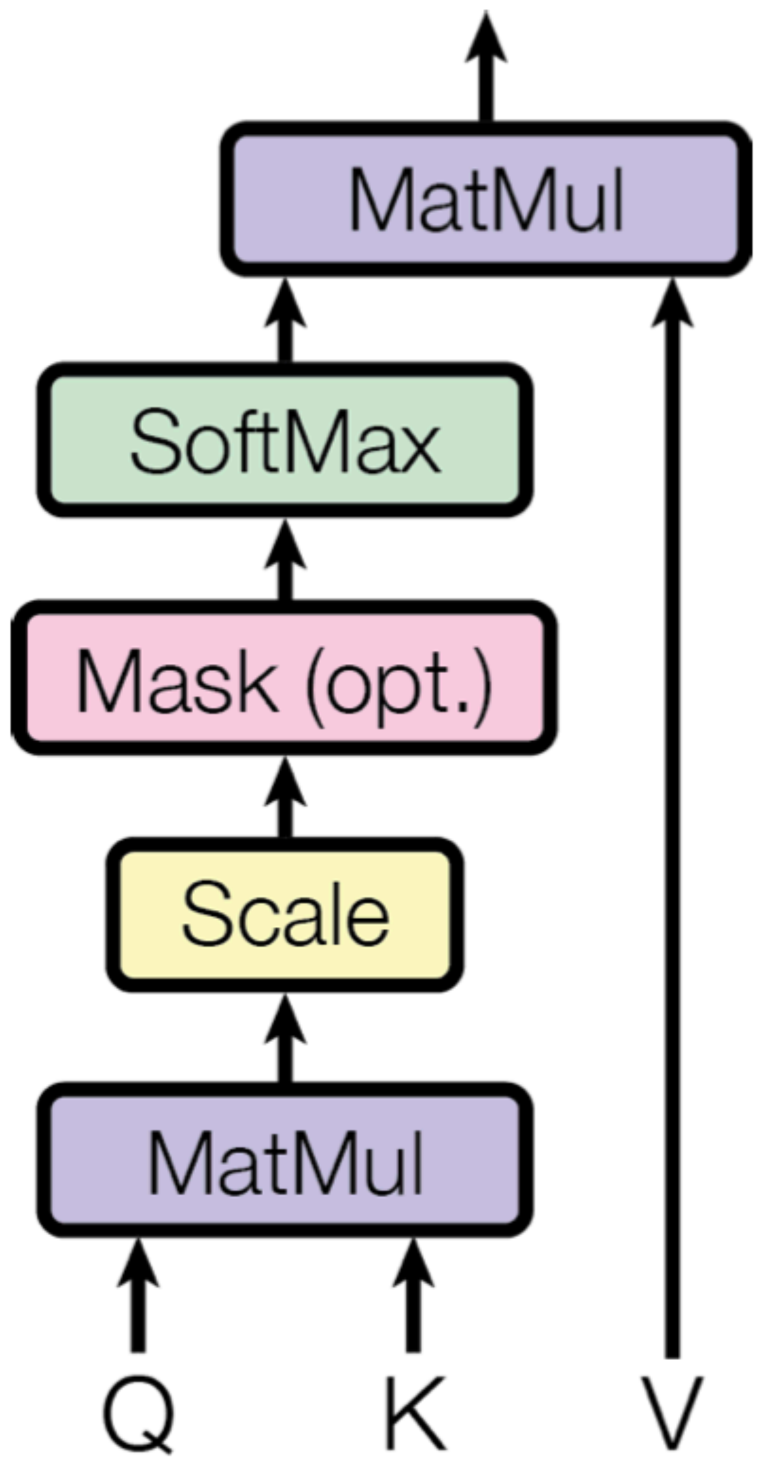
Q: Query
K: key
V: value

What is Attention

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

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

Scaled Dot-Product Attention



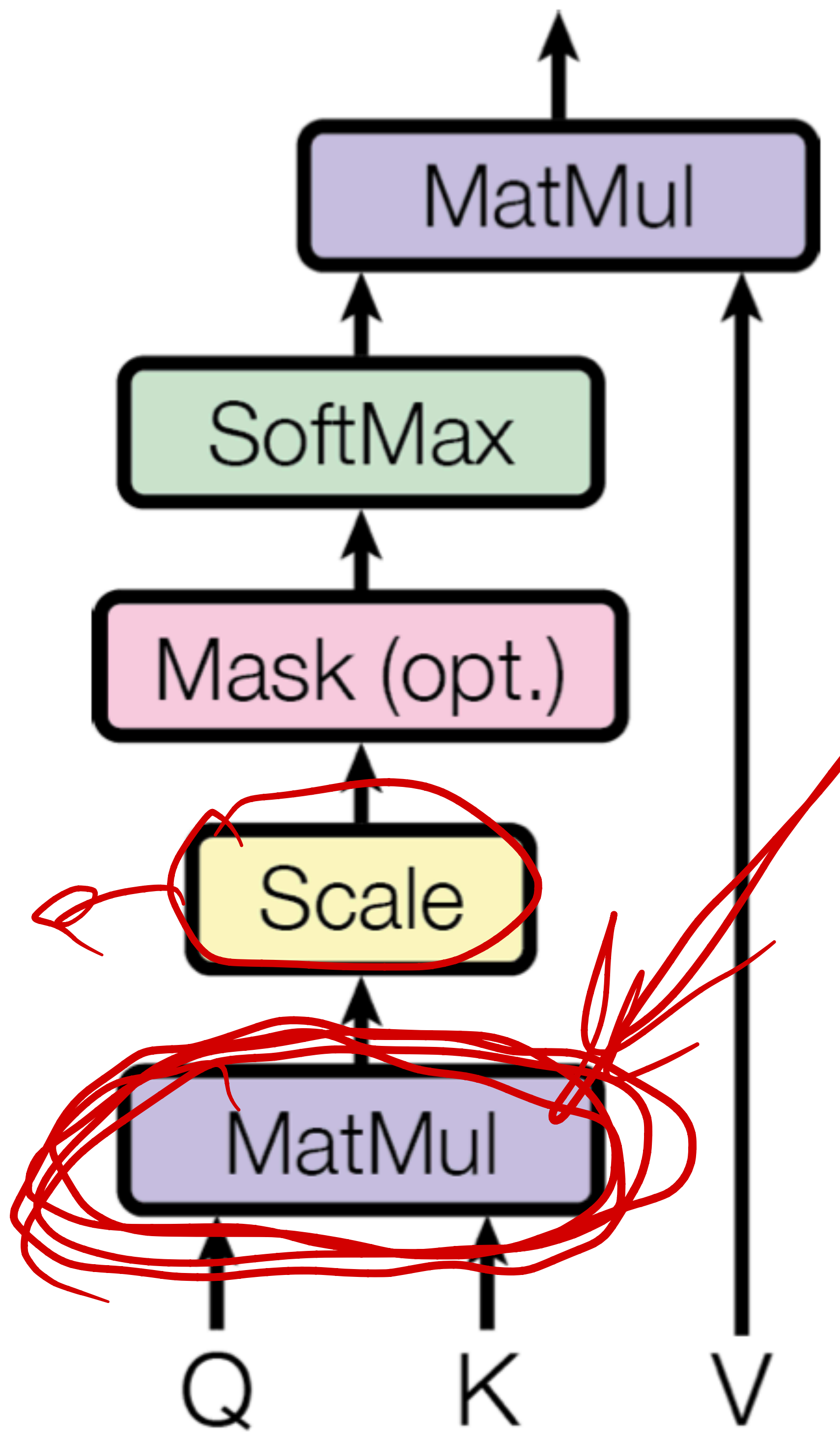
Q: Query
K: key
V: value

What is Attention

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

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

Scaled Dot-Product Attention

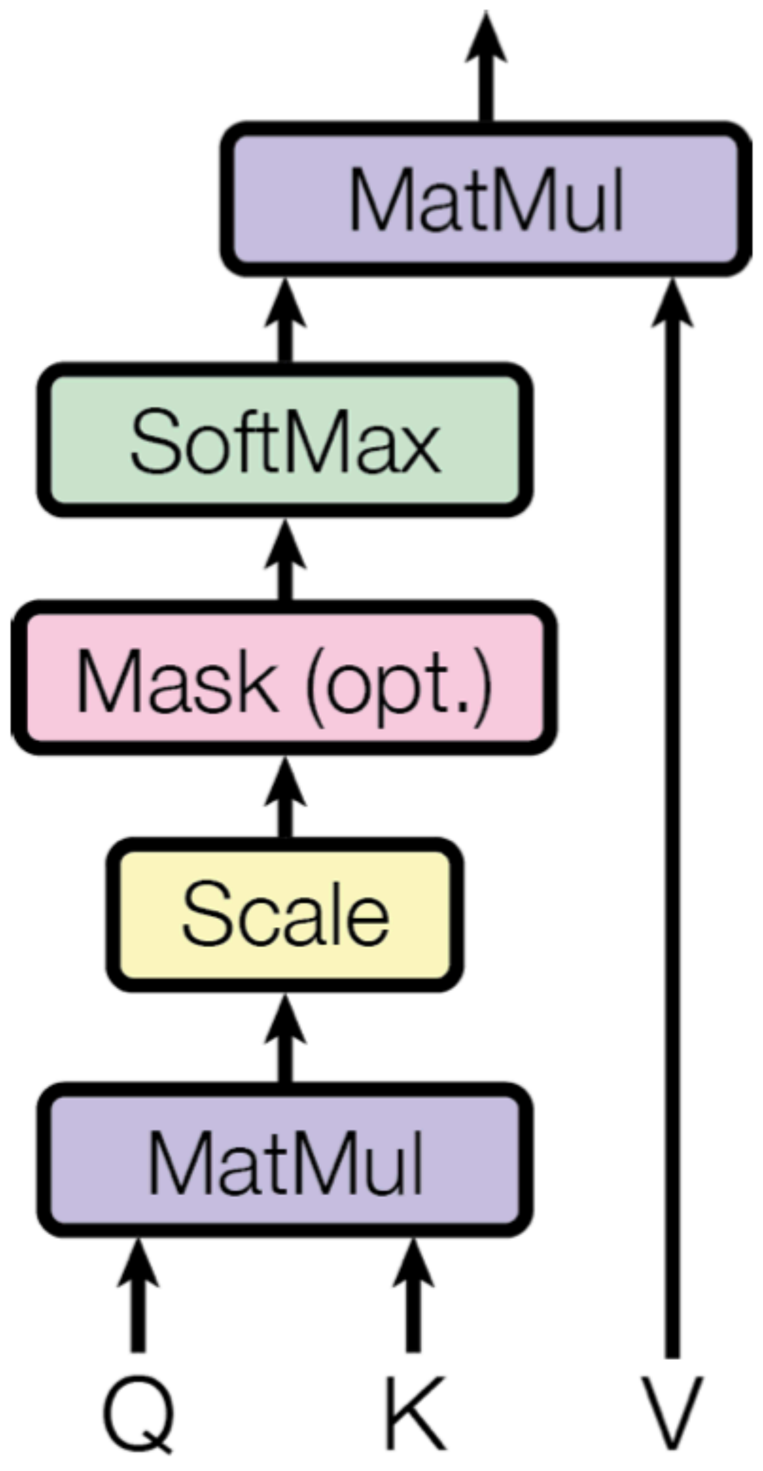


Attention weight = softmax(QK^T)

Q: Query
K: key
V: value

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)

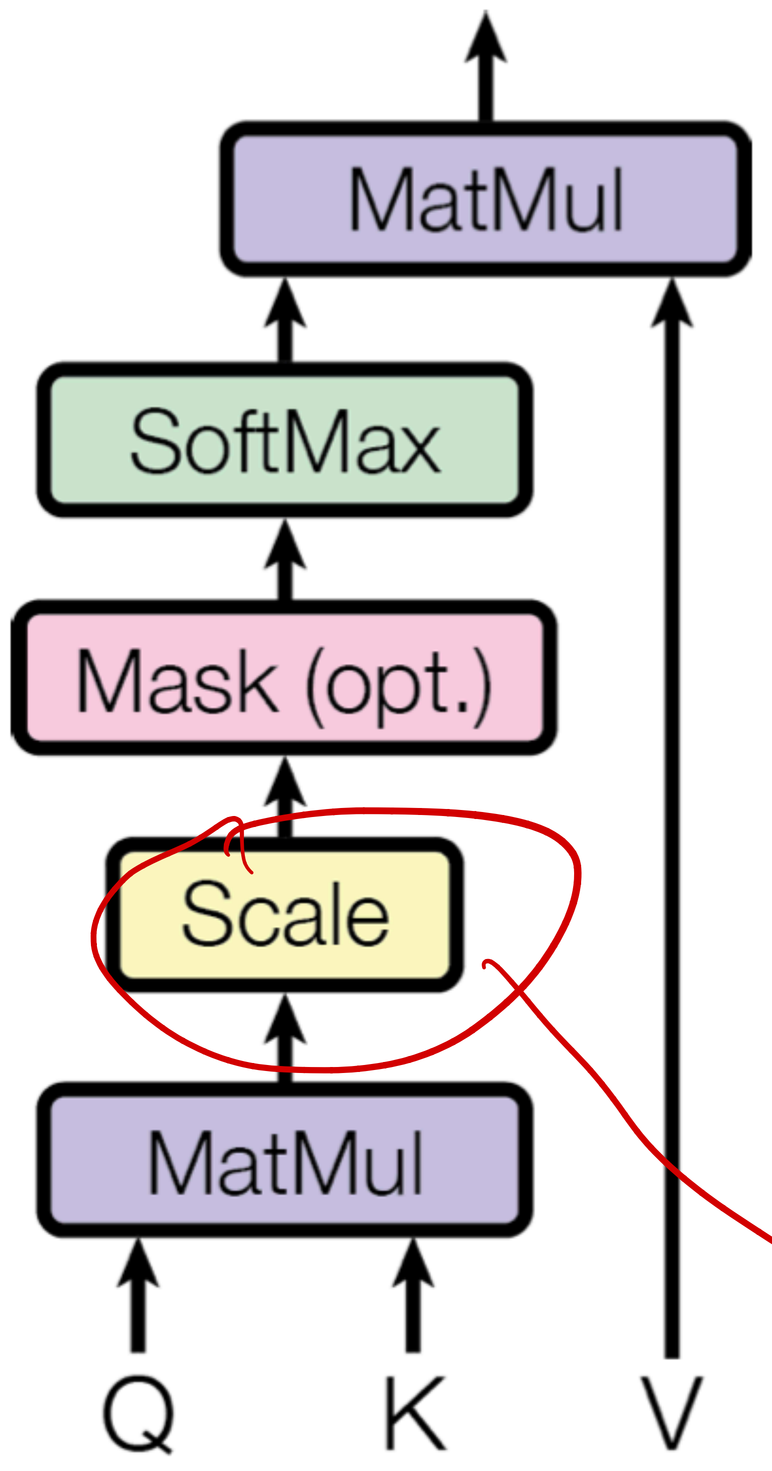
Dot-products grow large in magnitude

What is Attention

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

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

Scaled Dot-Product Attention



Q: Query
K: key
V: value

Attention weight = softmax(QK^T)

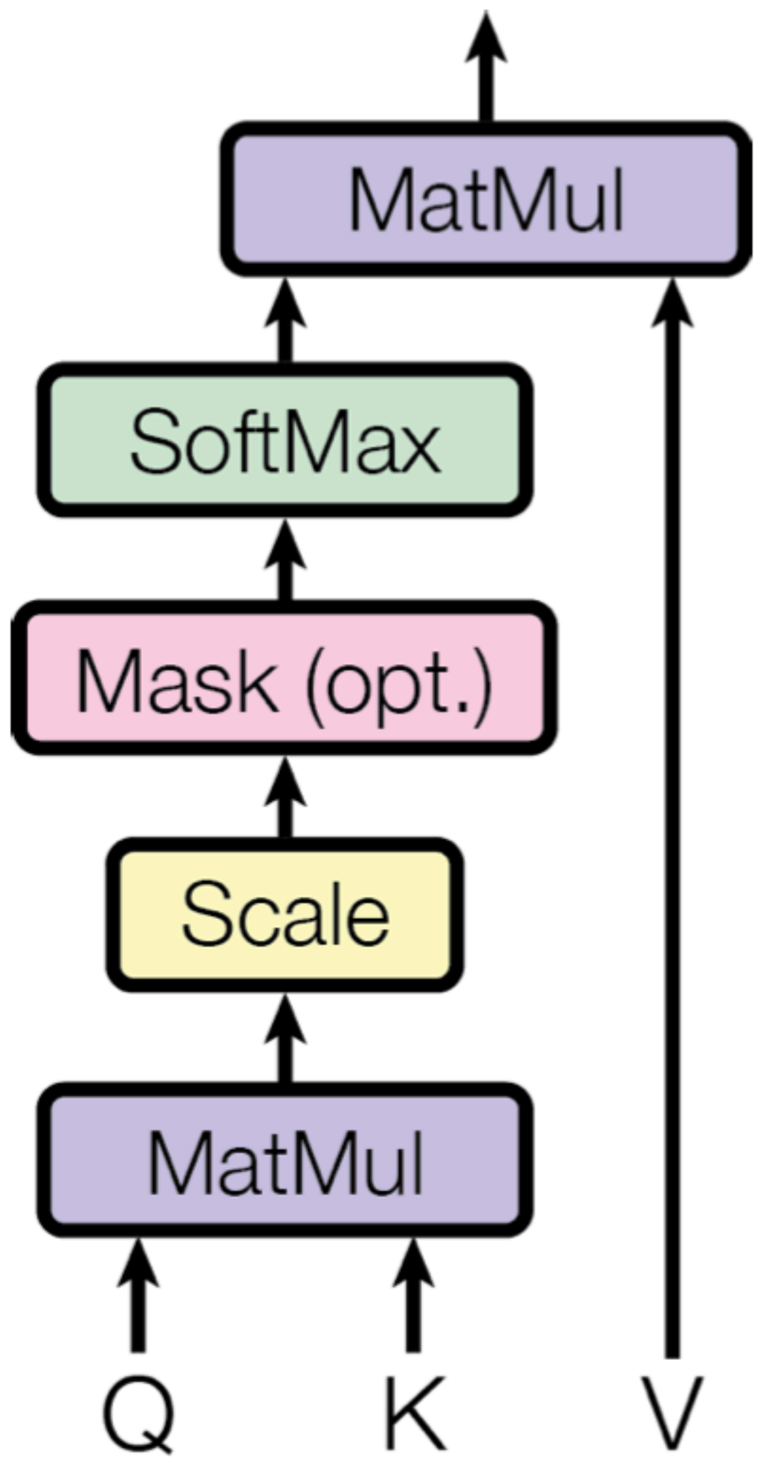
Dot-products grow large in magnitude

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

$$Var\left(\frac{X}{\sqrt{d_k}}\right) = \frac{Var(X)}{d_k}$$

What is Attention

Scaled Dot-Product Attention



Q: Query
K: key
V: value

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

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

Attention weight = softmax(QK^T)

Dot-products grow large in magnitude

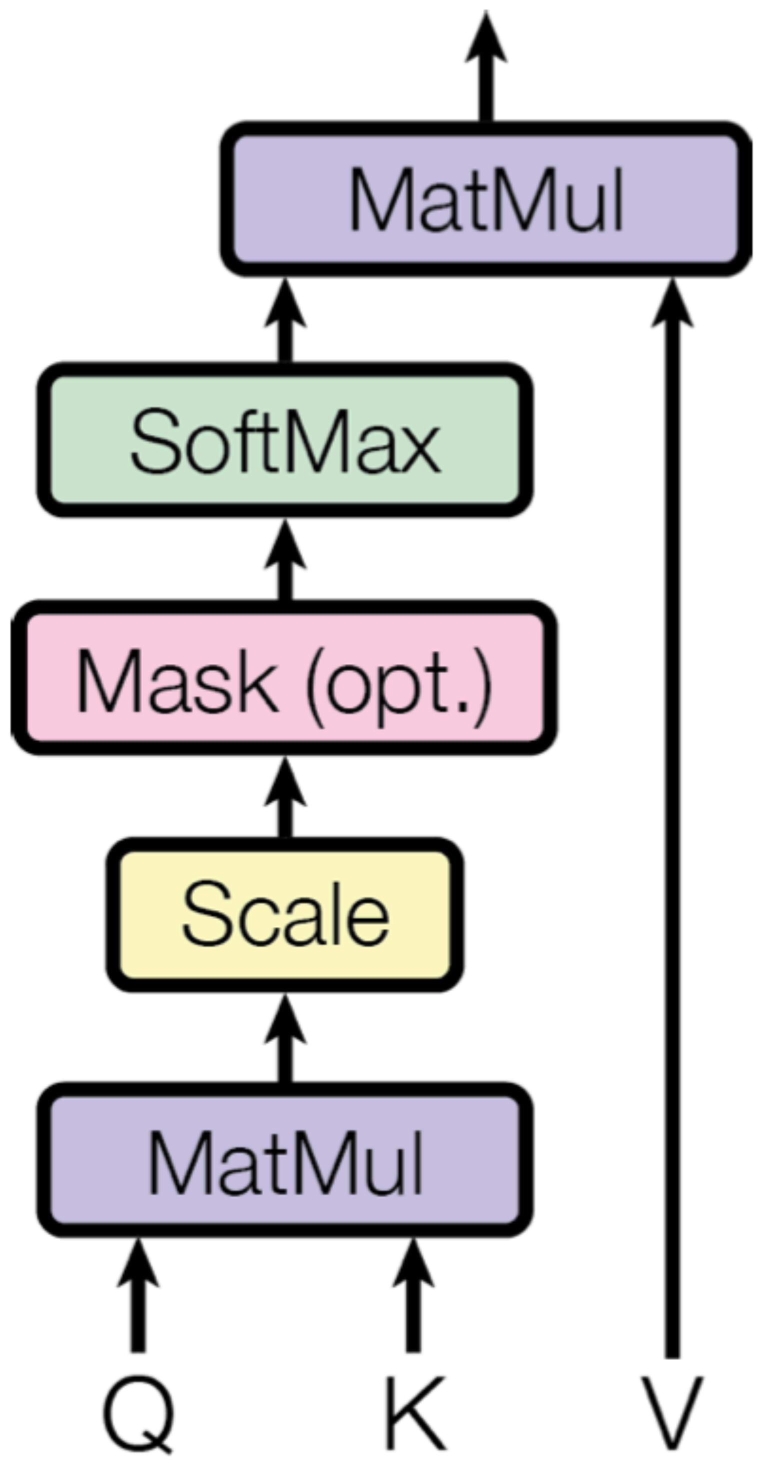
Scaled Attention weight = softmax$(\frac{QK^T}{\sqrt{d_k}})$ Shape is mxn

What is Attention

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

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

Scaled Dot-Product Attention



Q: Query
K: key
V: value

Attention weight = softmax(QK^T)

Dot-products grow large in magnitude

Scaled Attention weight = softmax$\left(\frac{QK^T}{\sqrt{d_k}}\right)$

Shape is mxn

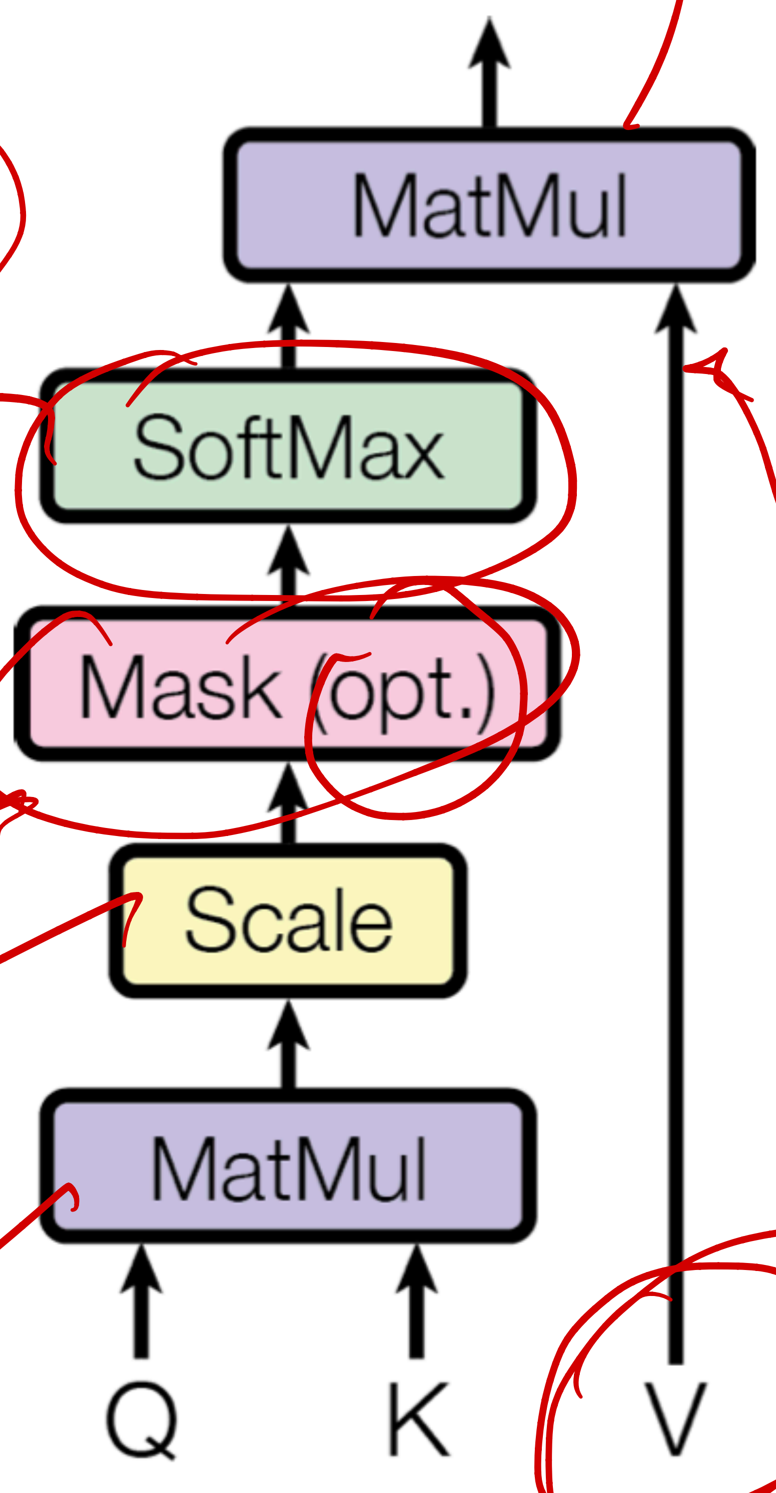
Attention weight represents the strength to “attend” values V

What is Attention

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

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

Scaled Dot-Product Attention



Q: Query
K: key
V: value

Attention weight = softmax(QK^T)

Dot-products grow large in magnitude

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

Attention weight represents the strength to "attend" values V

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

linear combination of all the values

$\mathbb{R}^d$ $\mathbb{R}^d$ $\mathbb{R}^d$

q_1 q_2 K_1 K_2 K_3 ... K_m $n \times d$

V_1 V_2 V_3 ... V_m

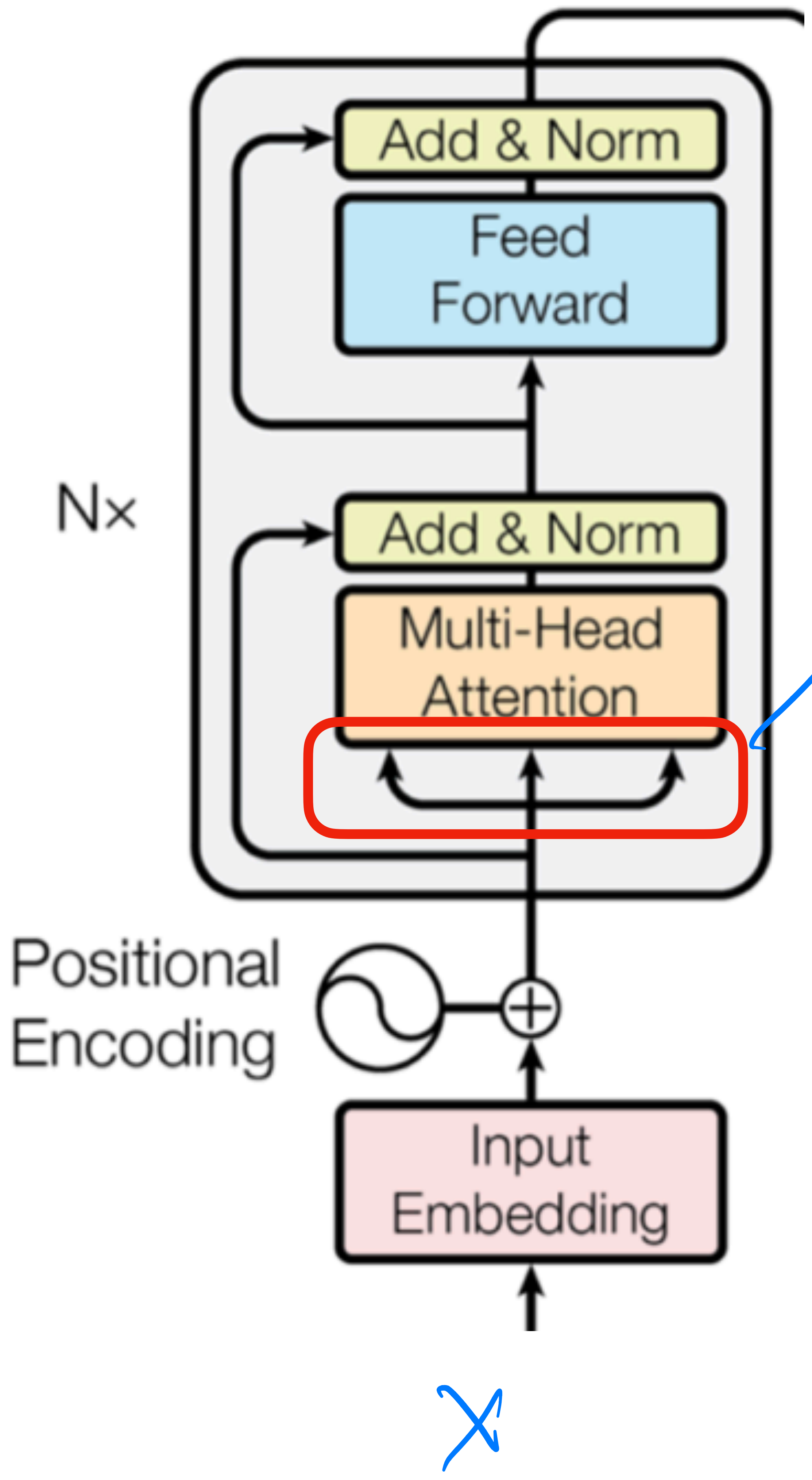
different queries ... q_n $\mathbb{R}^d$

are independent

$$\text{Softmax}\left(\frac{q_1 K_1^T, q_1 K_2^T, q_1 K_3^T \cdots q_1 K_m^T}{\sqrt{d}}\right) = W_1, W_2 \cdots W_m \quad \in \mathbb{R}$$

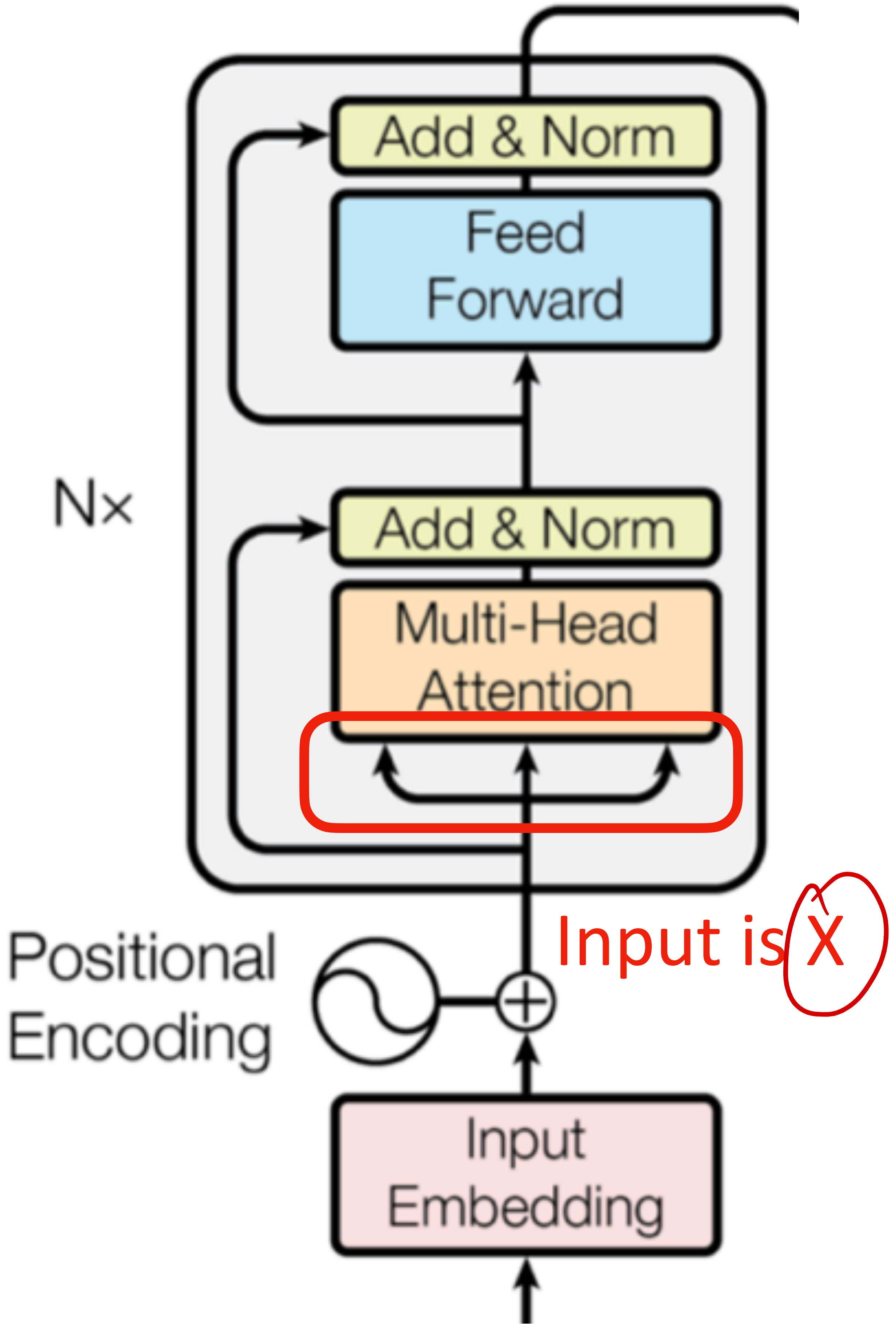
$$\text{output} = W_1 \cdot V_1 + W_2 \cdot V_2 + W_3 V_3 \cdots W_m V_m$$

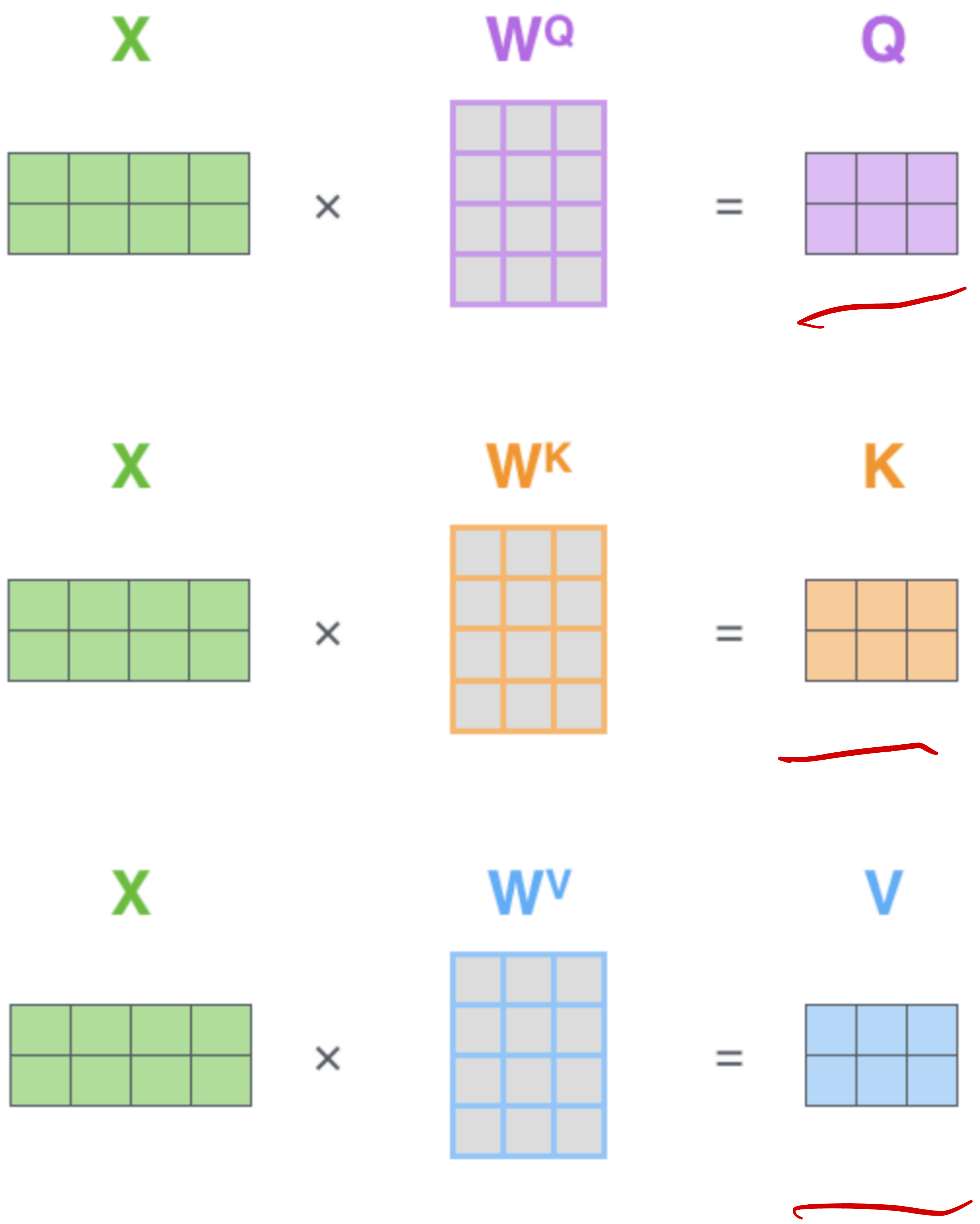
Q, K, V



What are Q, K, V in the transformer

Self-Attention

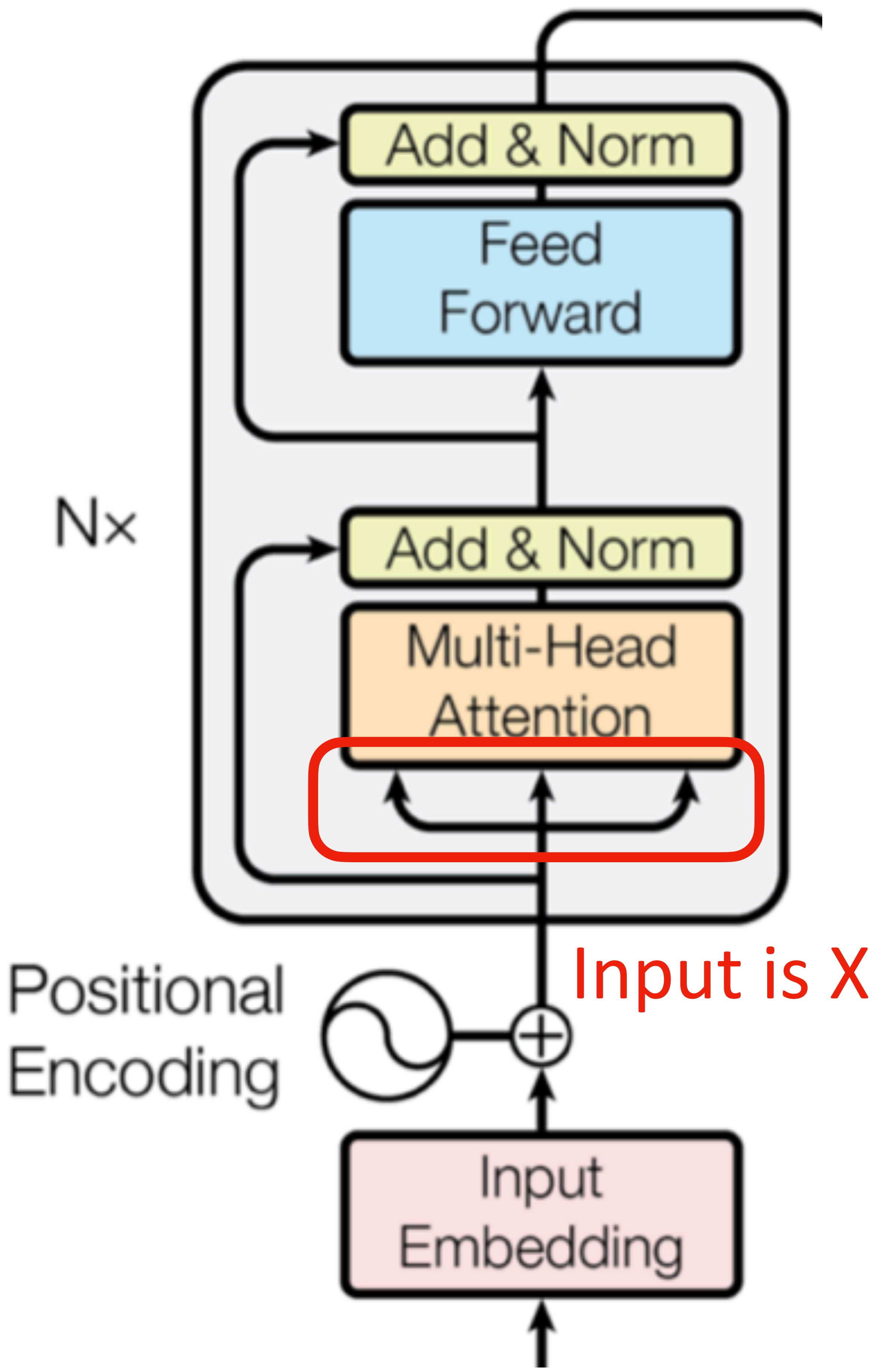


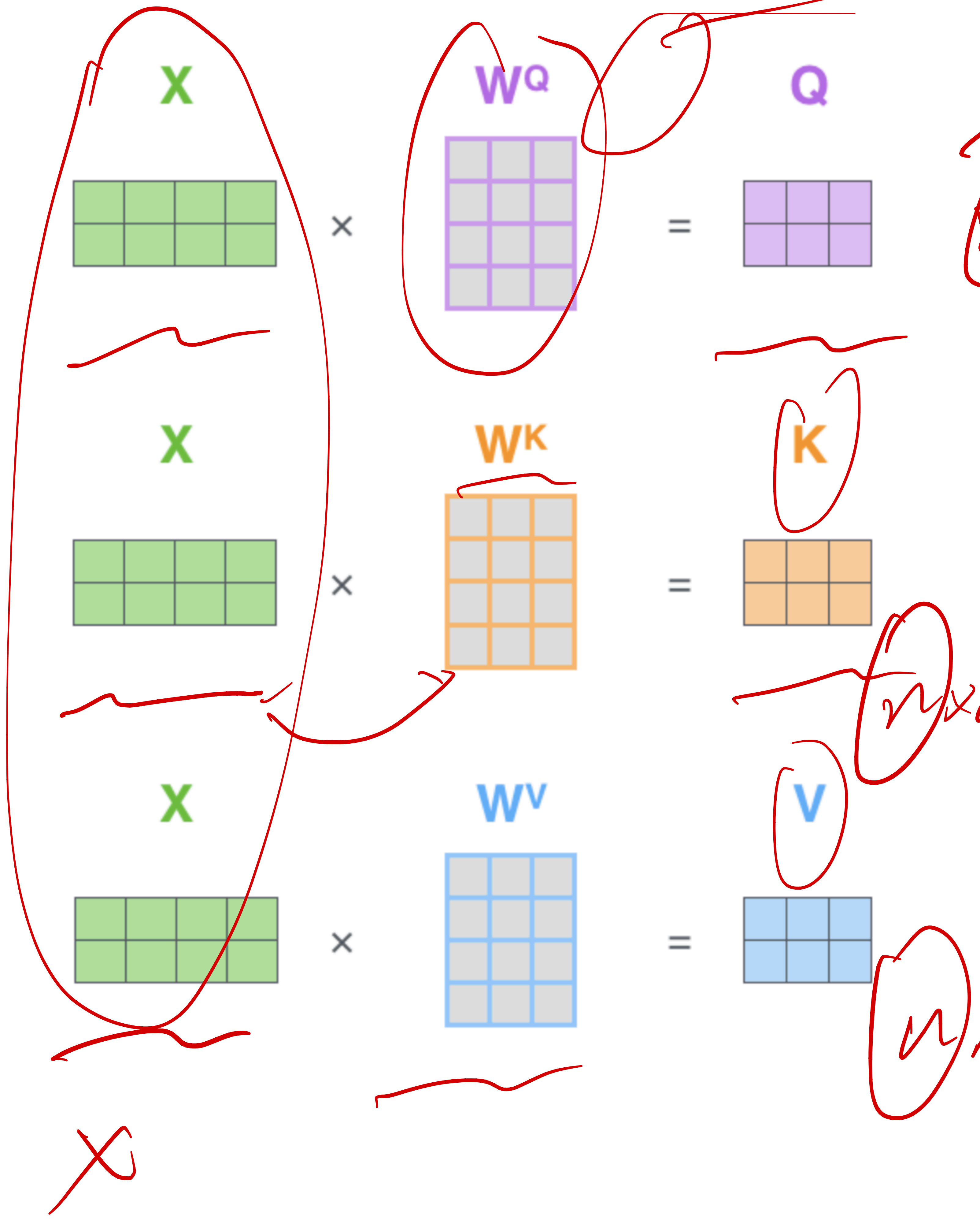


8

Jay Alammar. The Illustrated Transformer.

Self-Attention

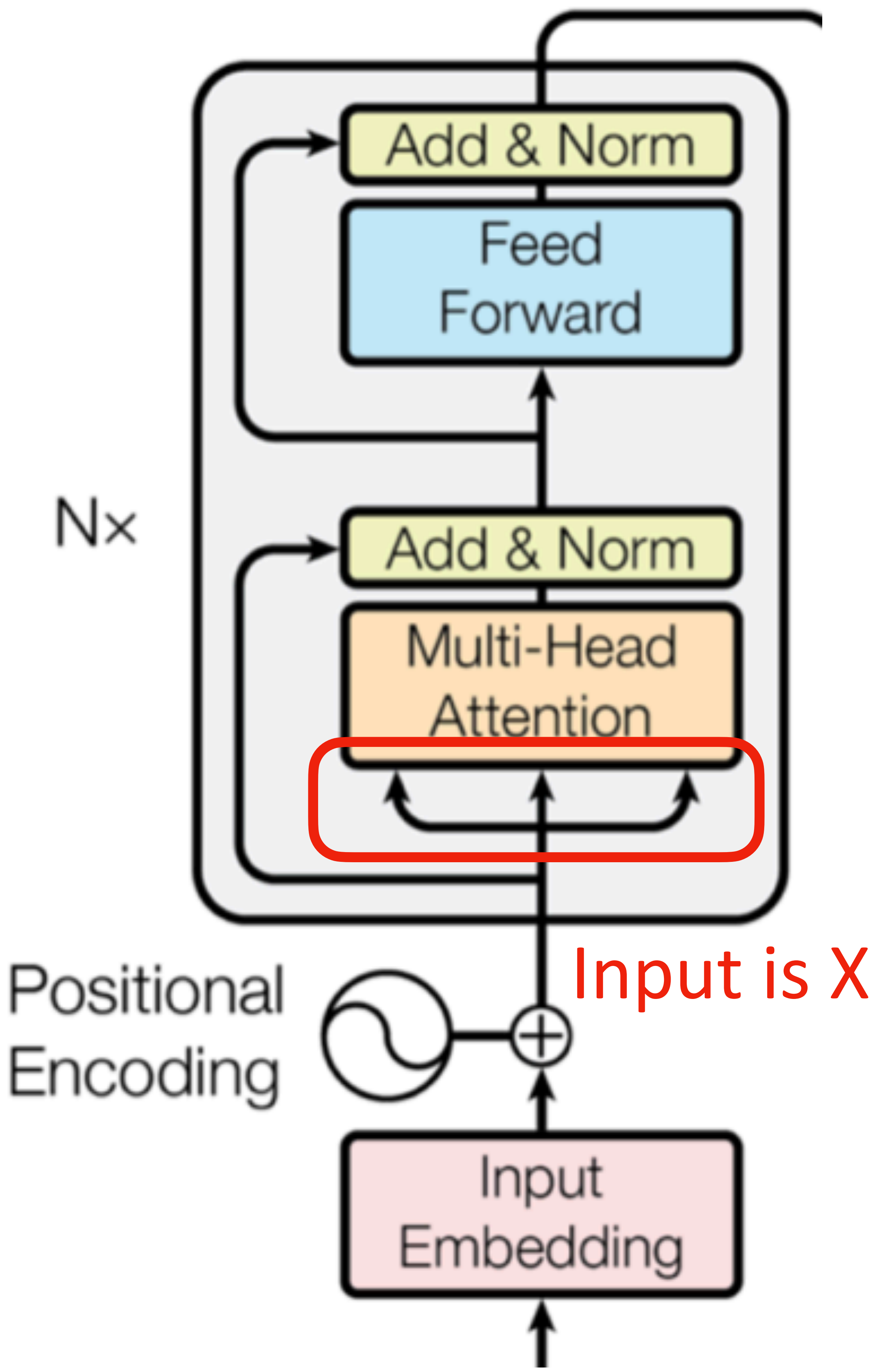




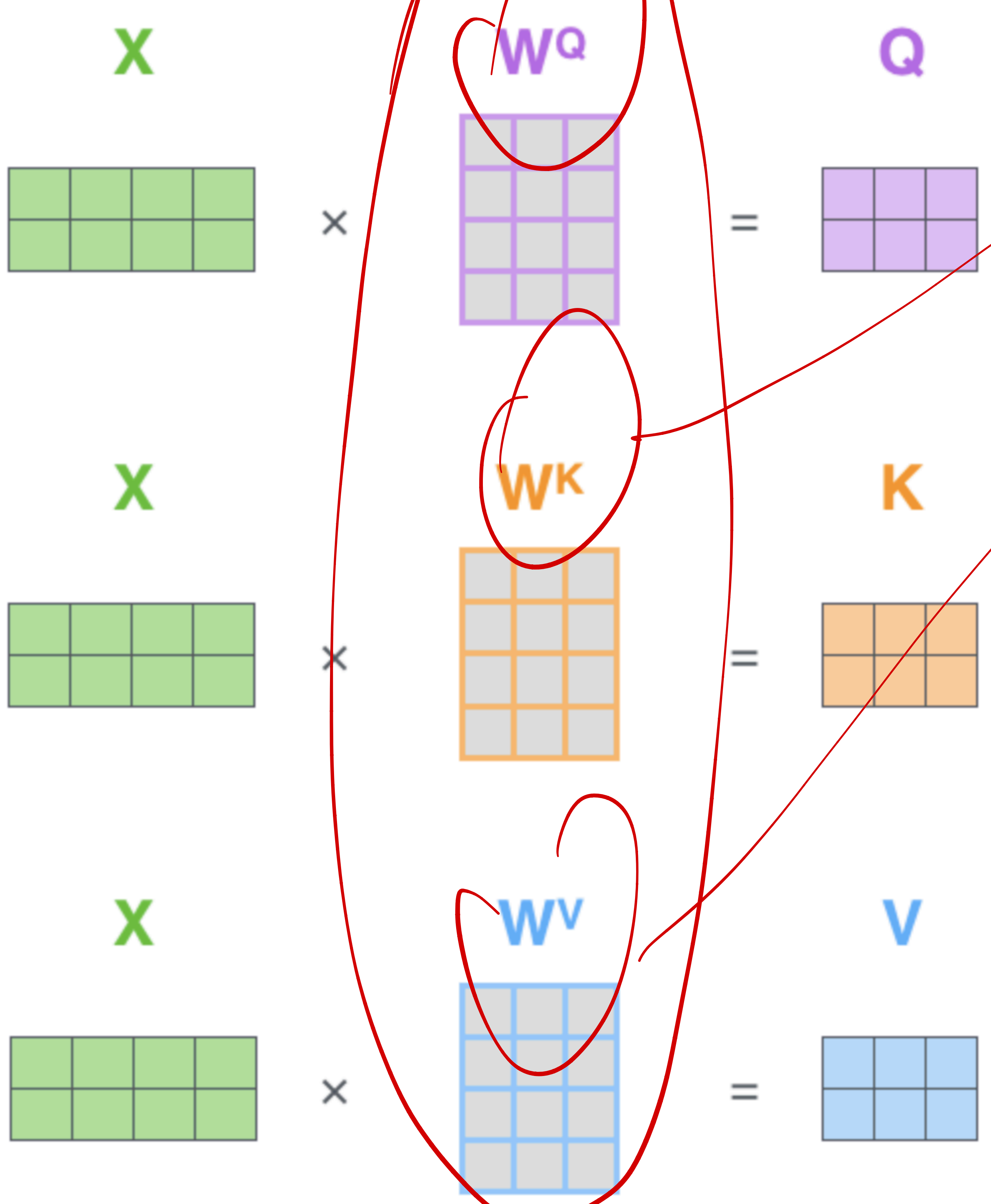
Query, key, and value are from the same input, thus it is called "self"-attention

Jay Alammar. The Illustrated Transformer.

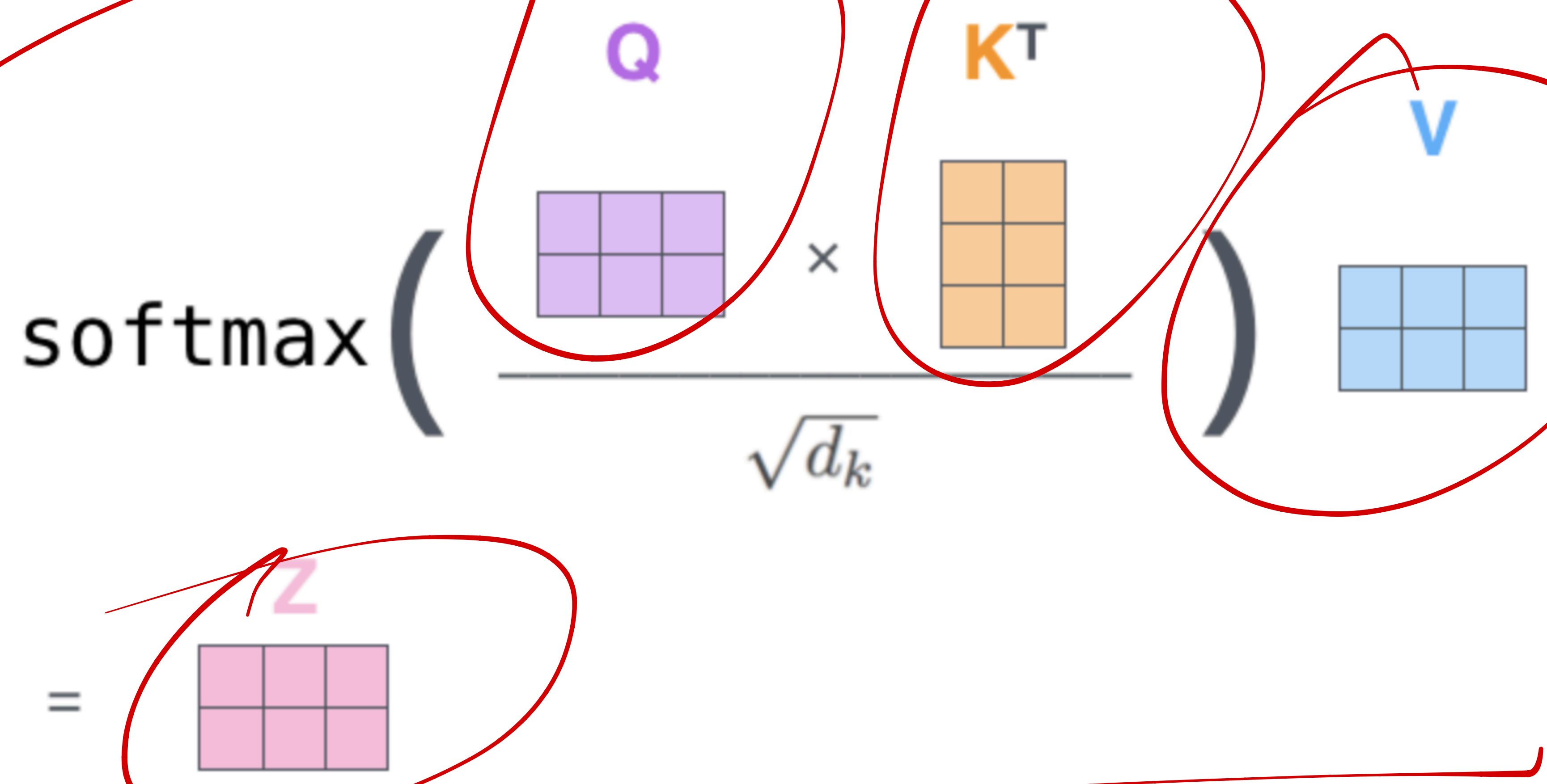
Self-Attention



8

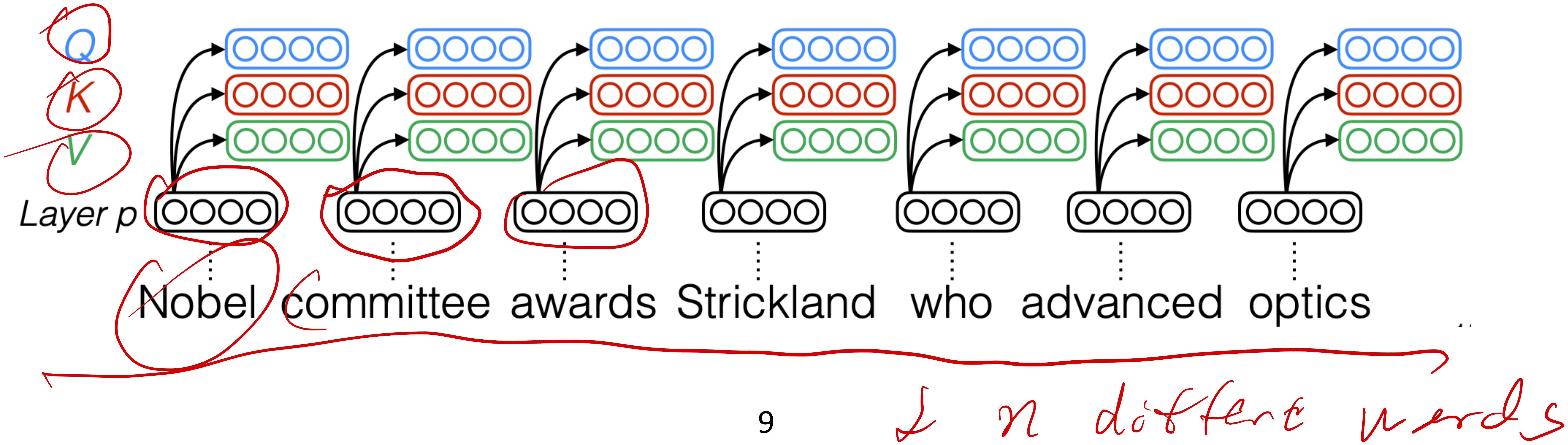


Query, key, and value are from the same input, thus it is called "self"-attention



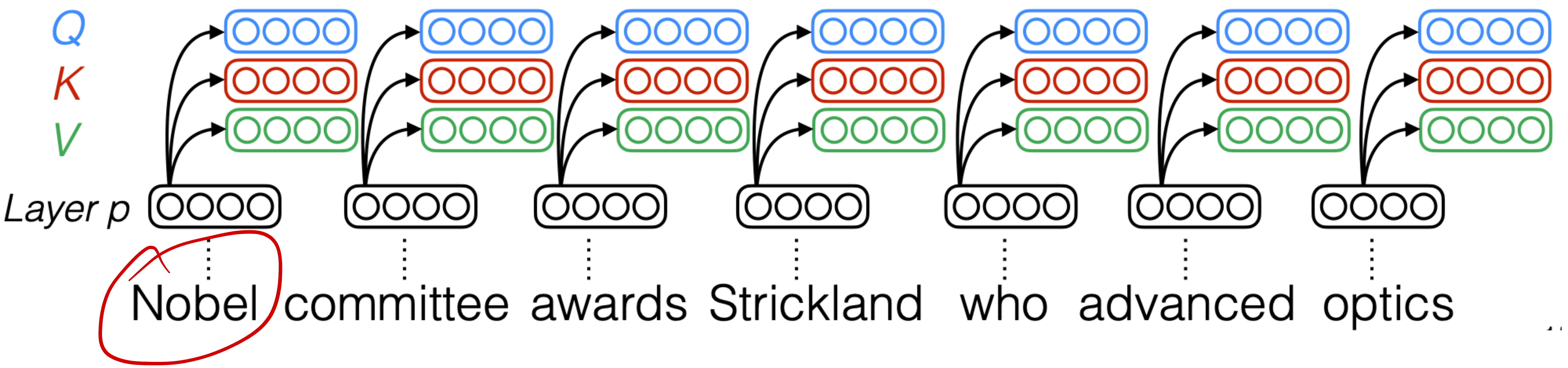
Jay Alammar. The Illustrated Transformer.

Self-Attention

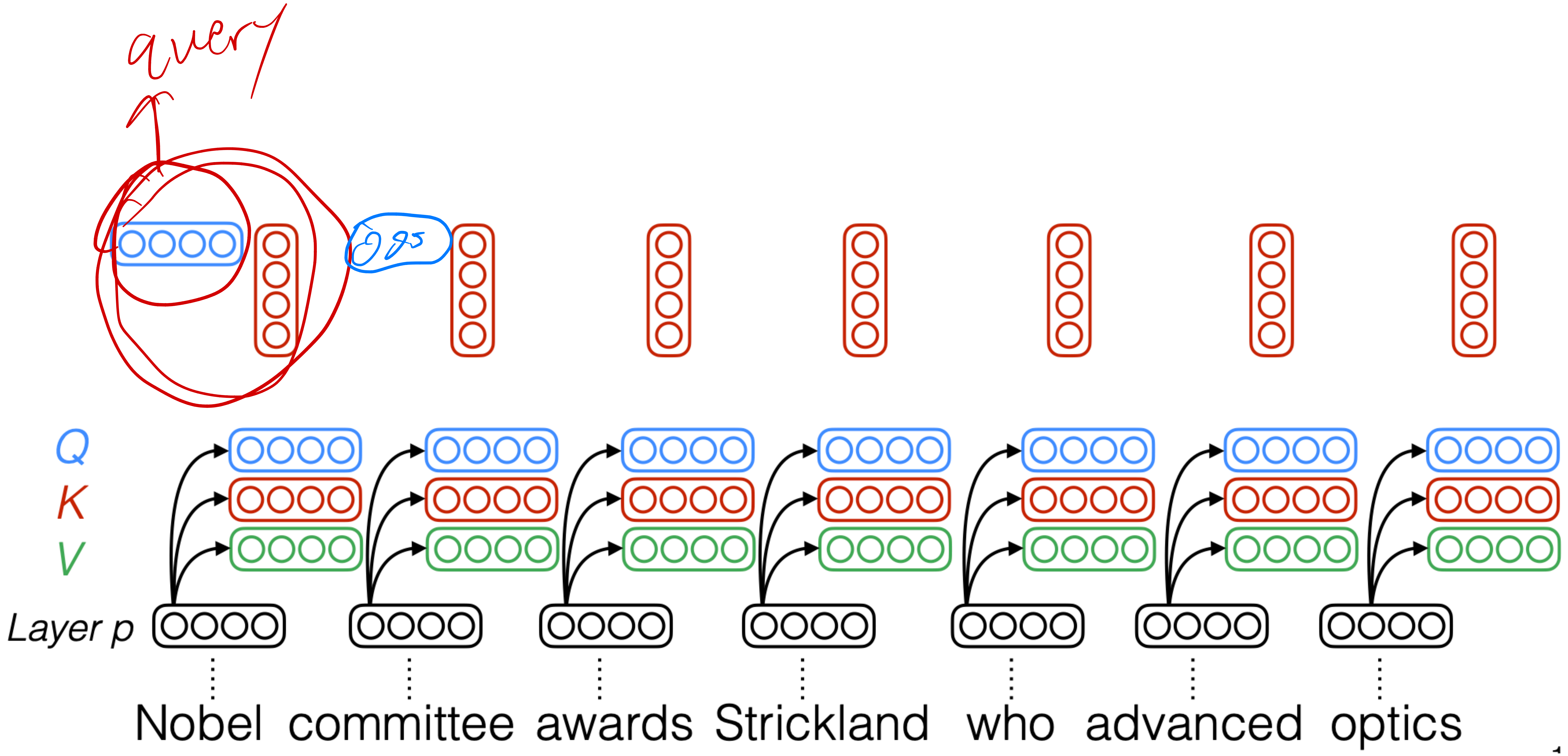


Self-Attention

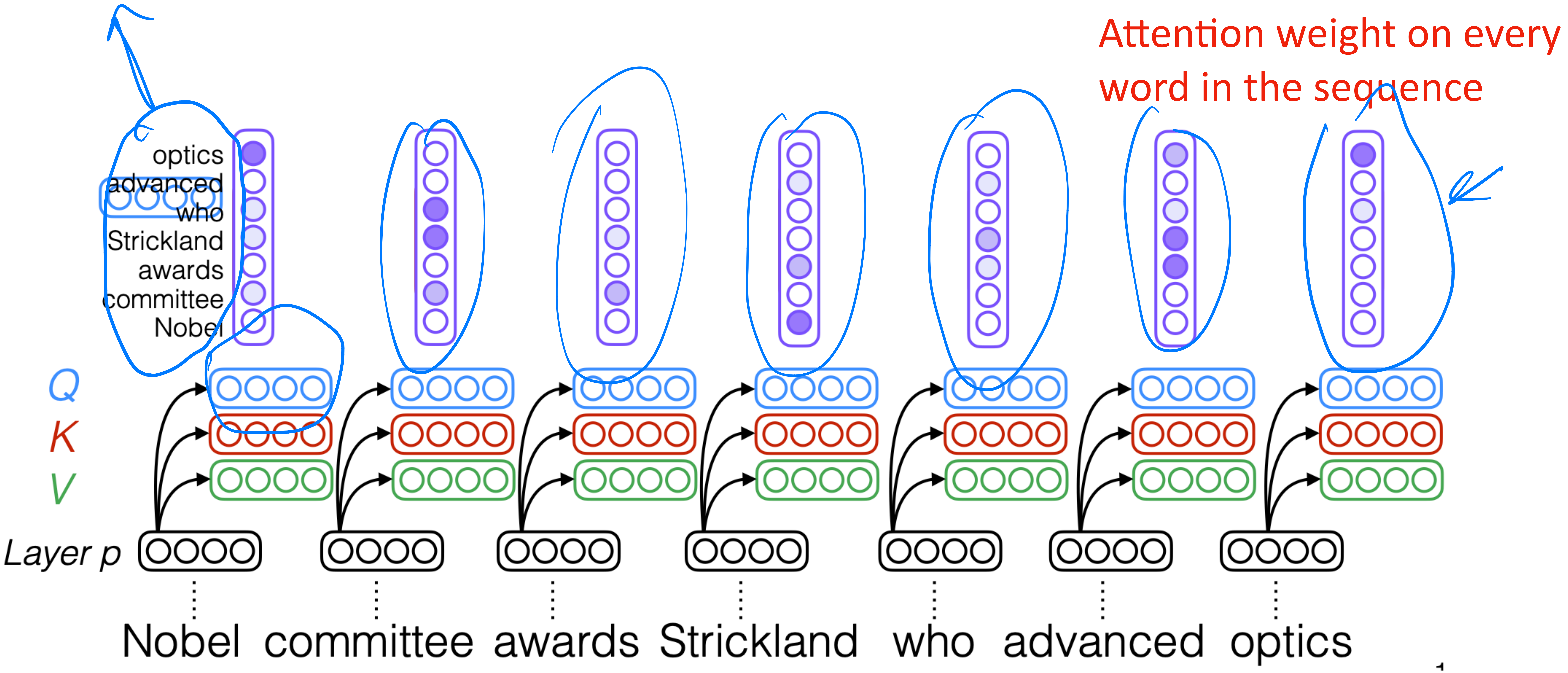
At each step, the attention computation attends to all steps in the input example



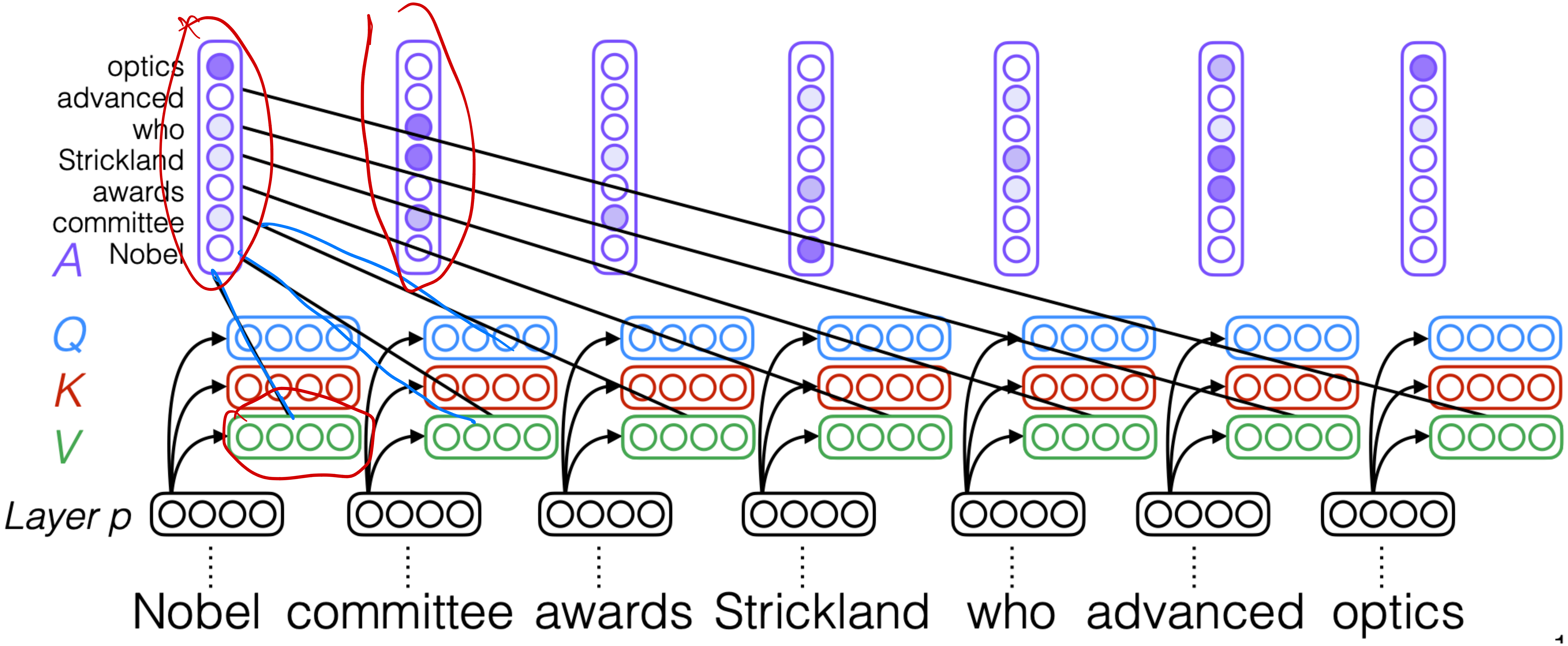
Self-Attention



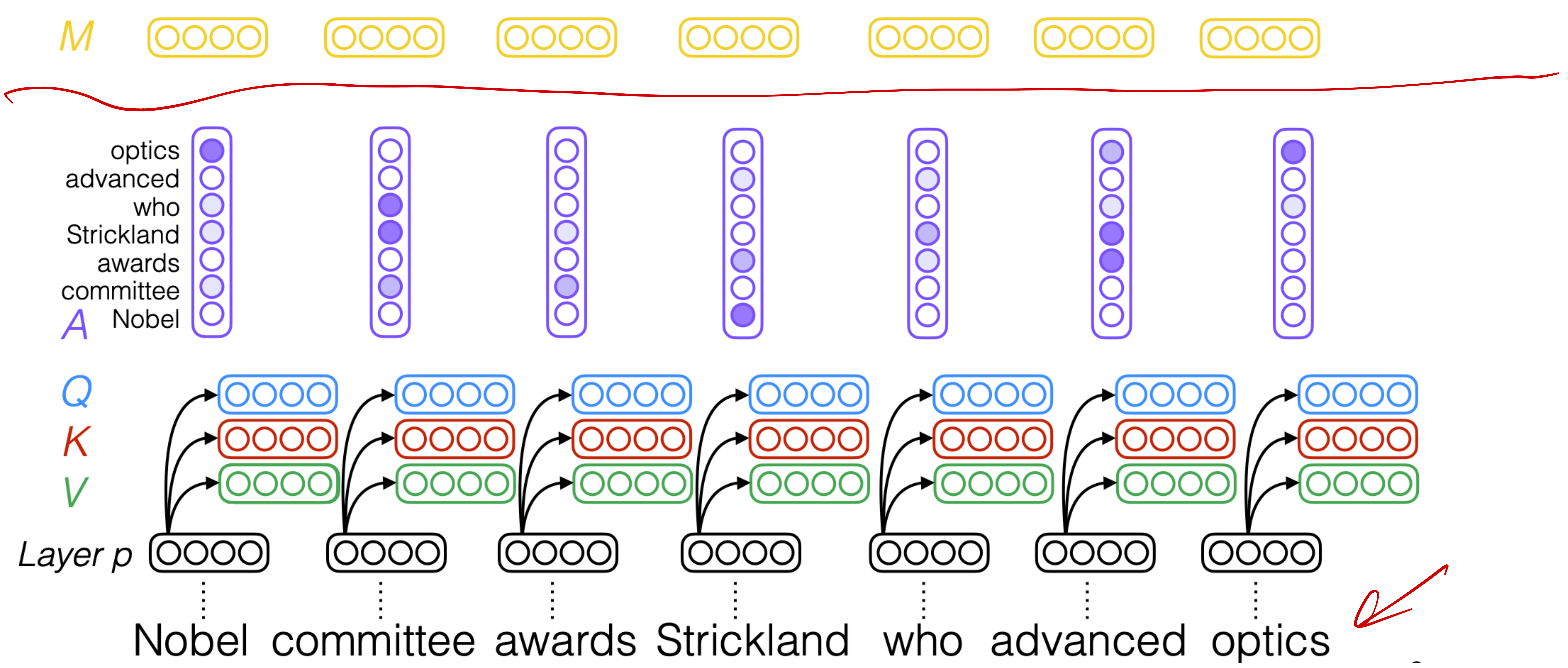
Self-Attention



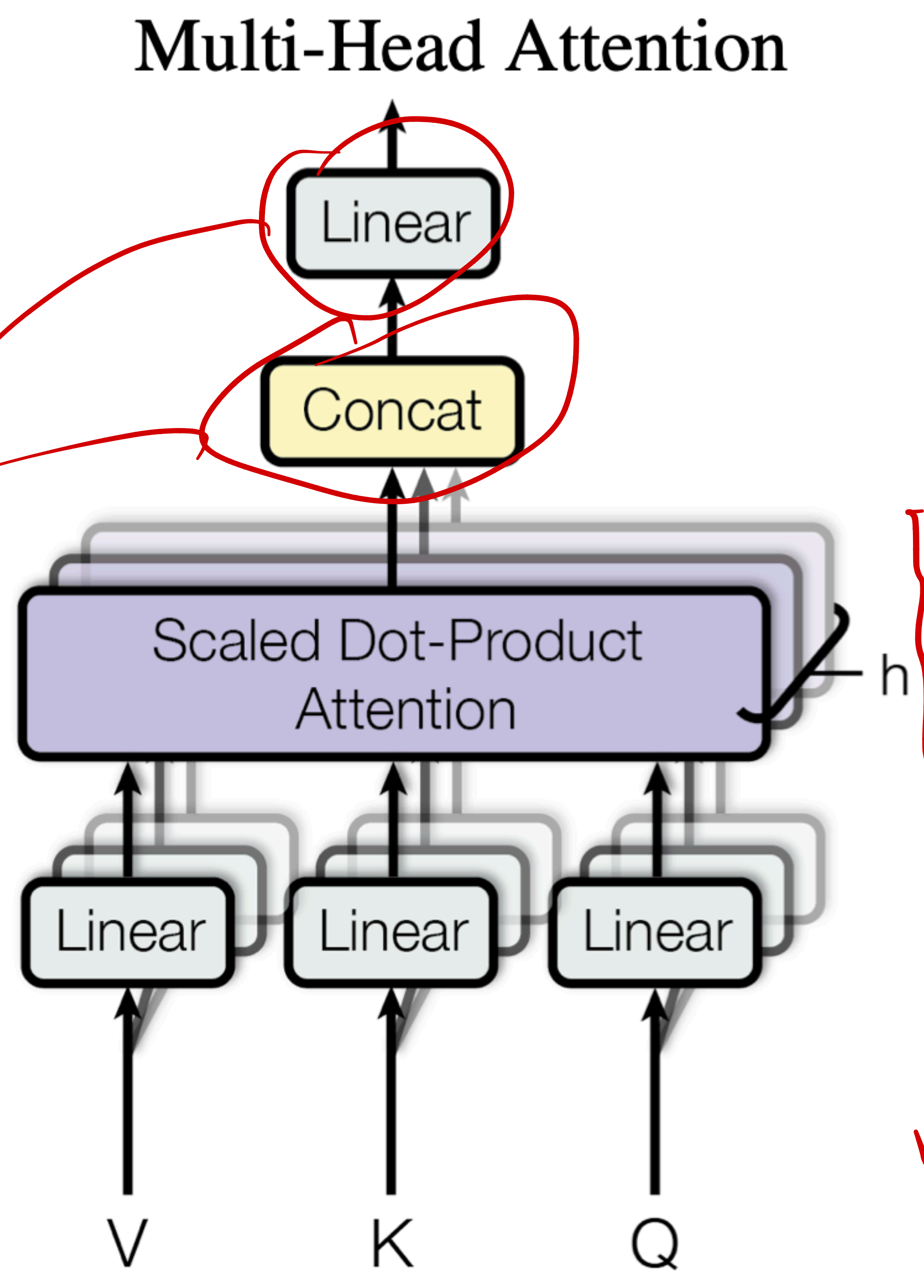
Self-Attention



Self-Attention



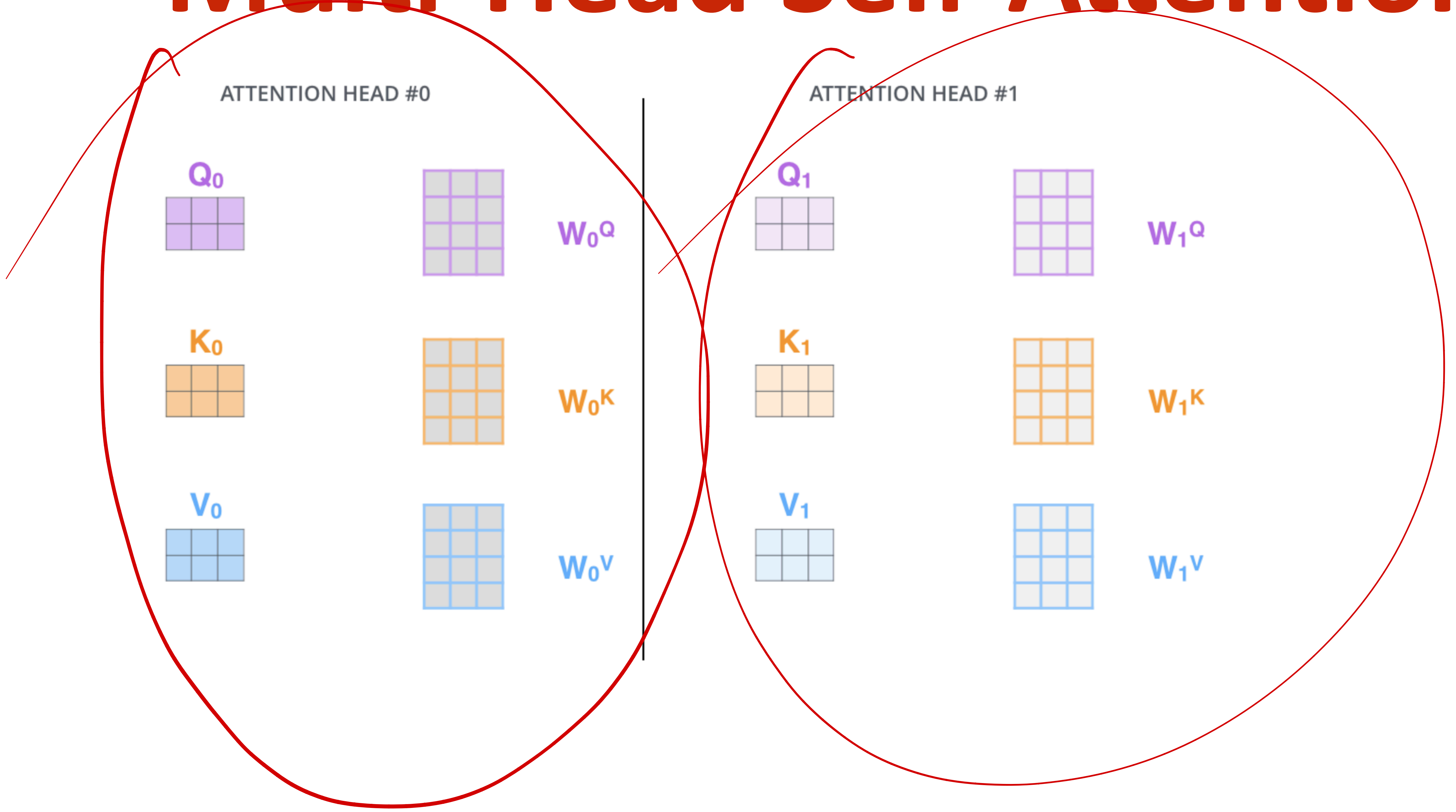
Multi-Head Attention



Multi-Head Self-Attention

Jay Alammar. The Illustrated Transformer.

Multi-Head Self-Attention



Jay Alammar. The Illustrated Transformer.

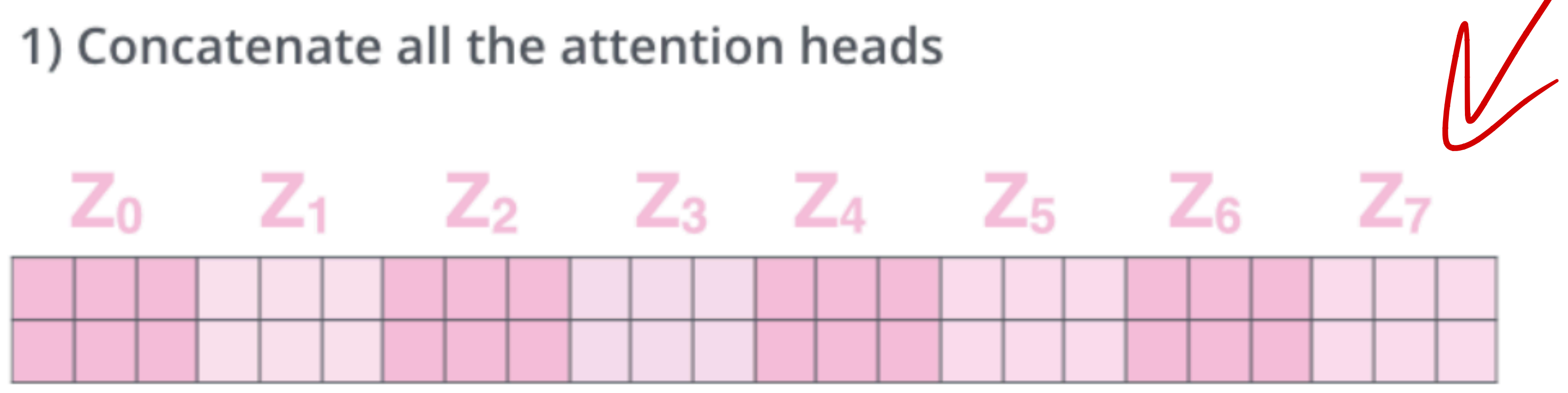
Multi-Head Self-Attention

Jay Alammar. The Illustrated Transformer.

Multi-Head Self-Attention

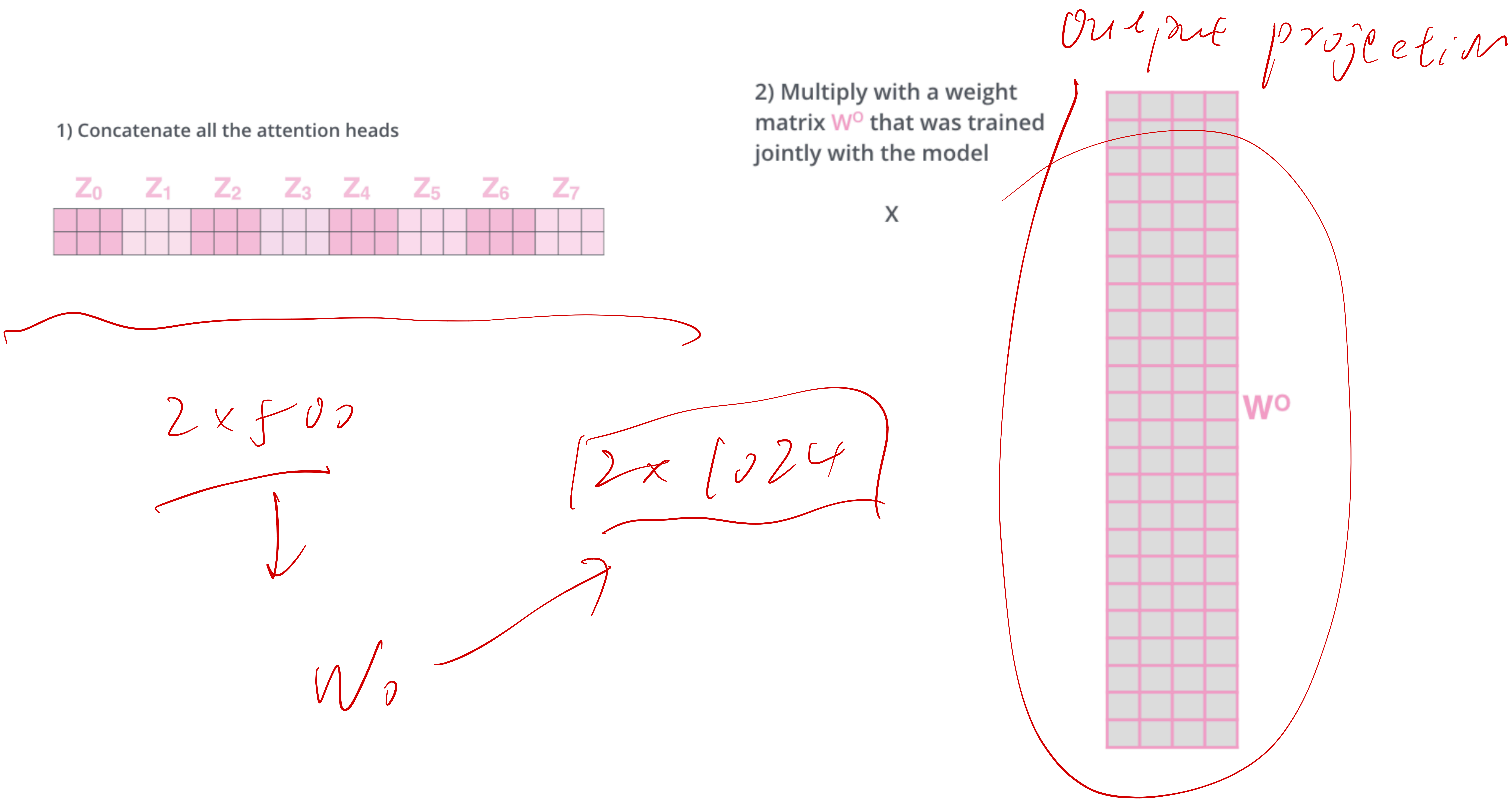
Jay Alammar. The Illustrated Transformer.

Multi-Head Self-Attention



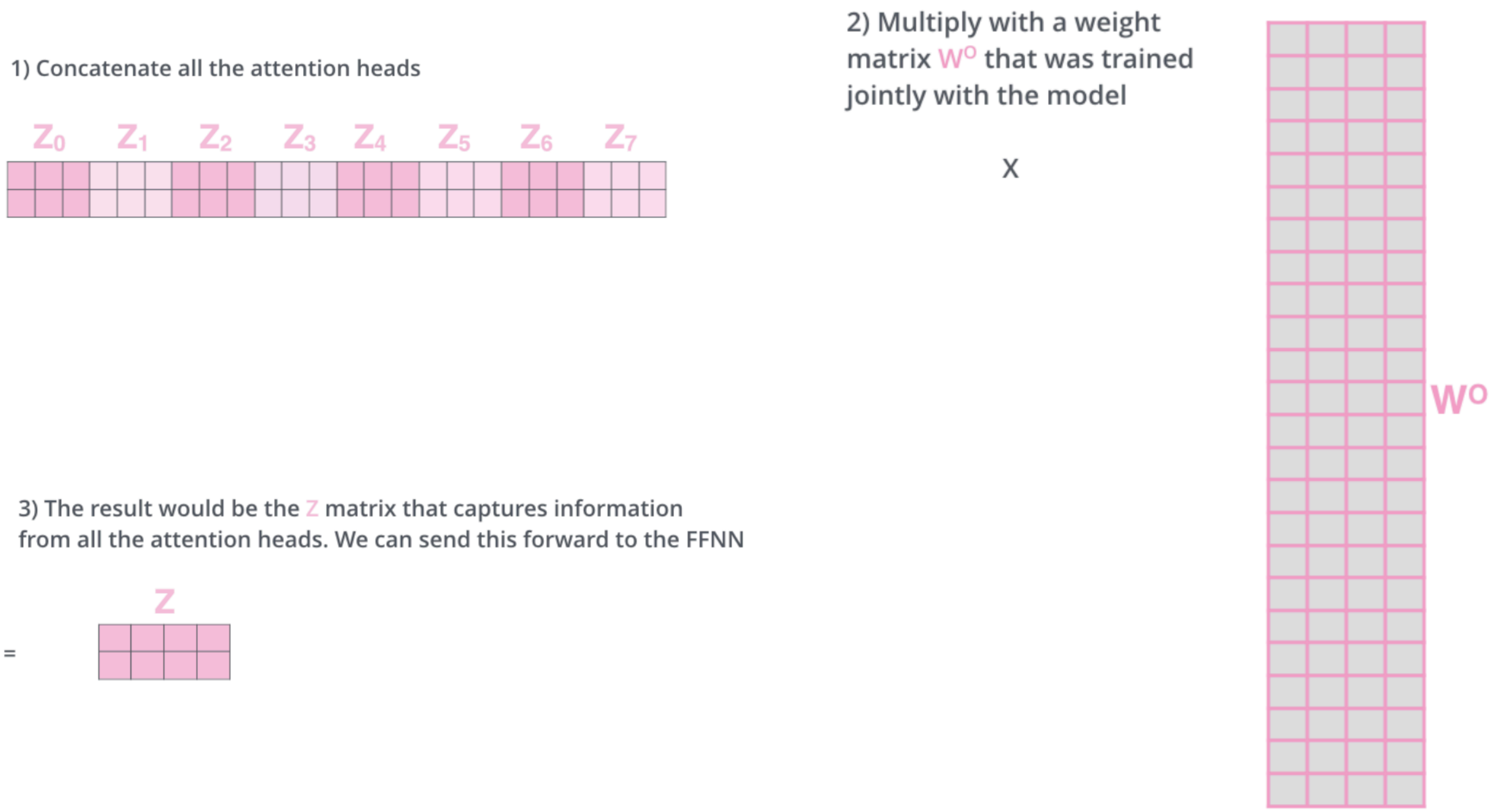
Jay Alammar. The Illustrated Transformer.

Multi-Head Self-Attention



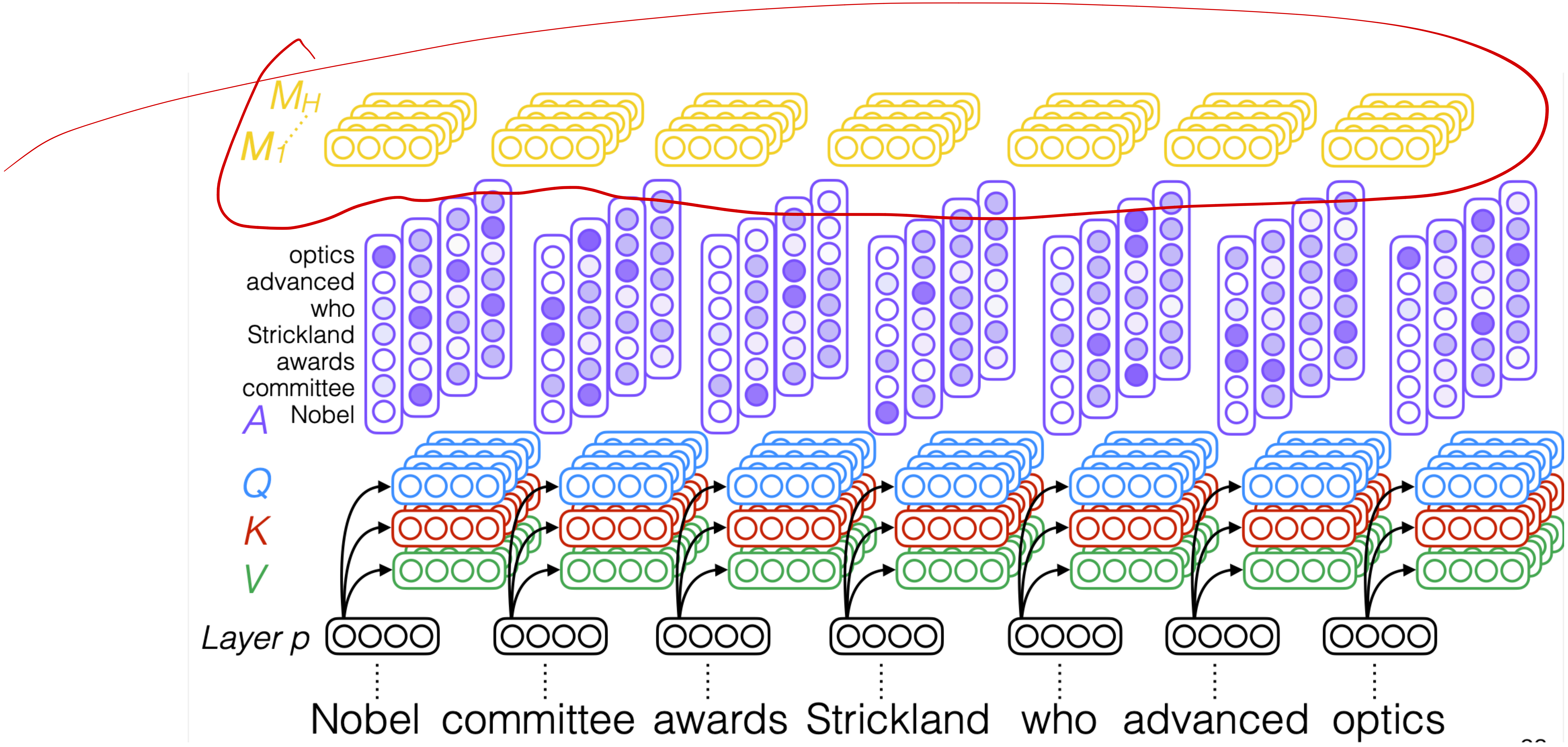
Jay Alammar. The Illustrated Transformer.

Multi-Head Self-Attention

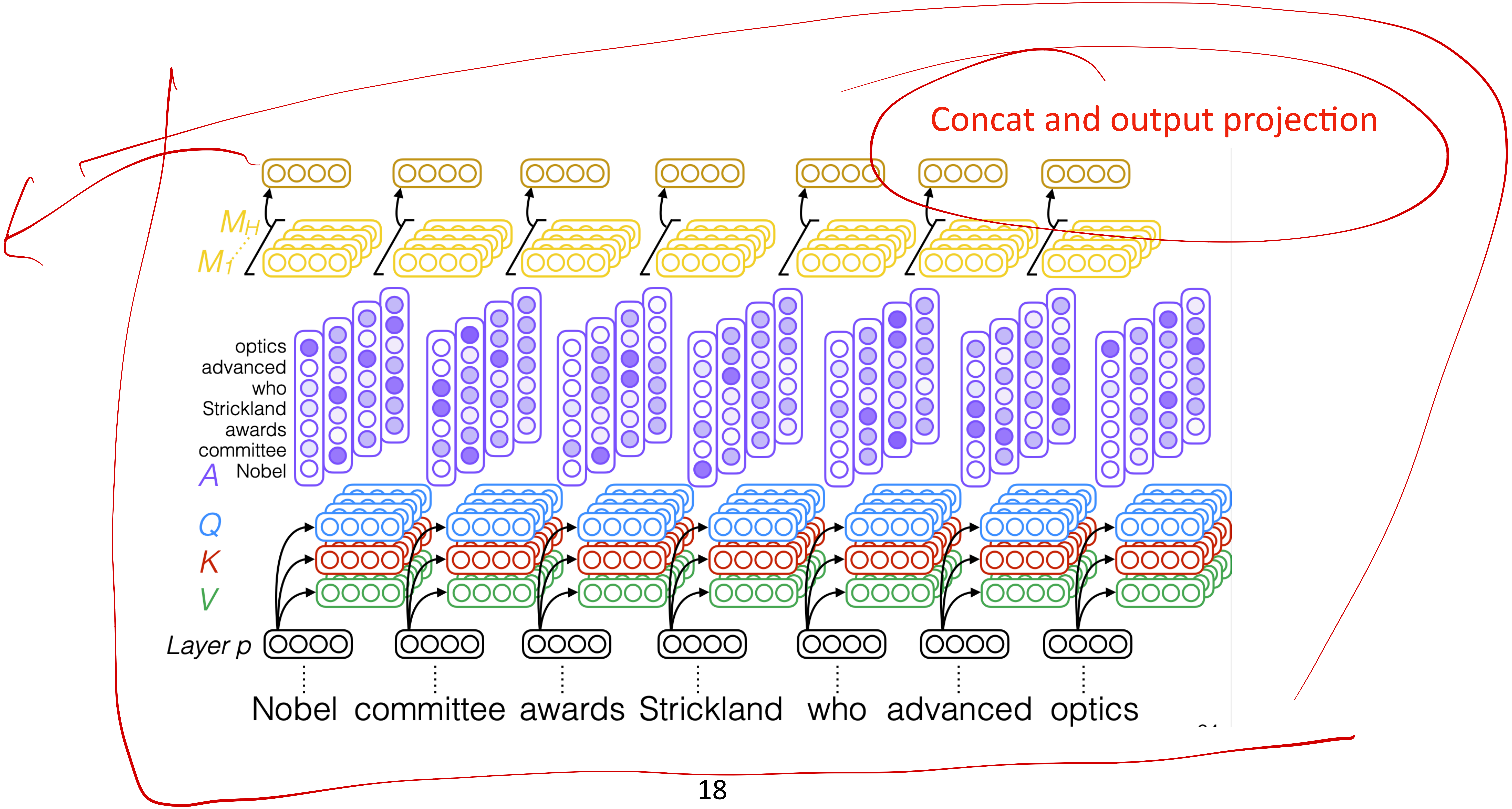


Jay Alammar. The Illustrated Transformer.

Multi-head Self-Attention

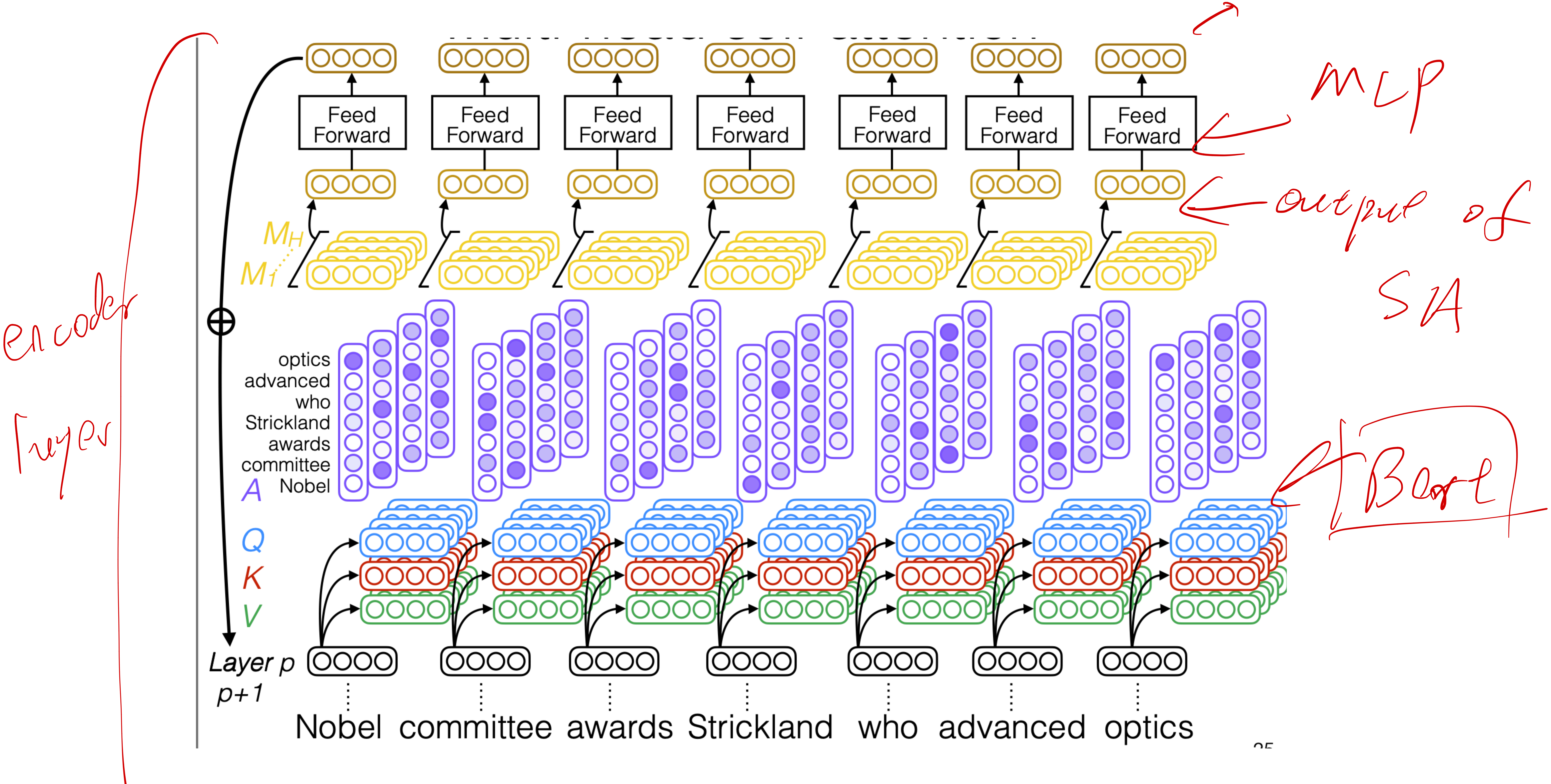


Multi-head Self-Attention

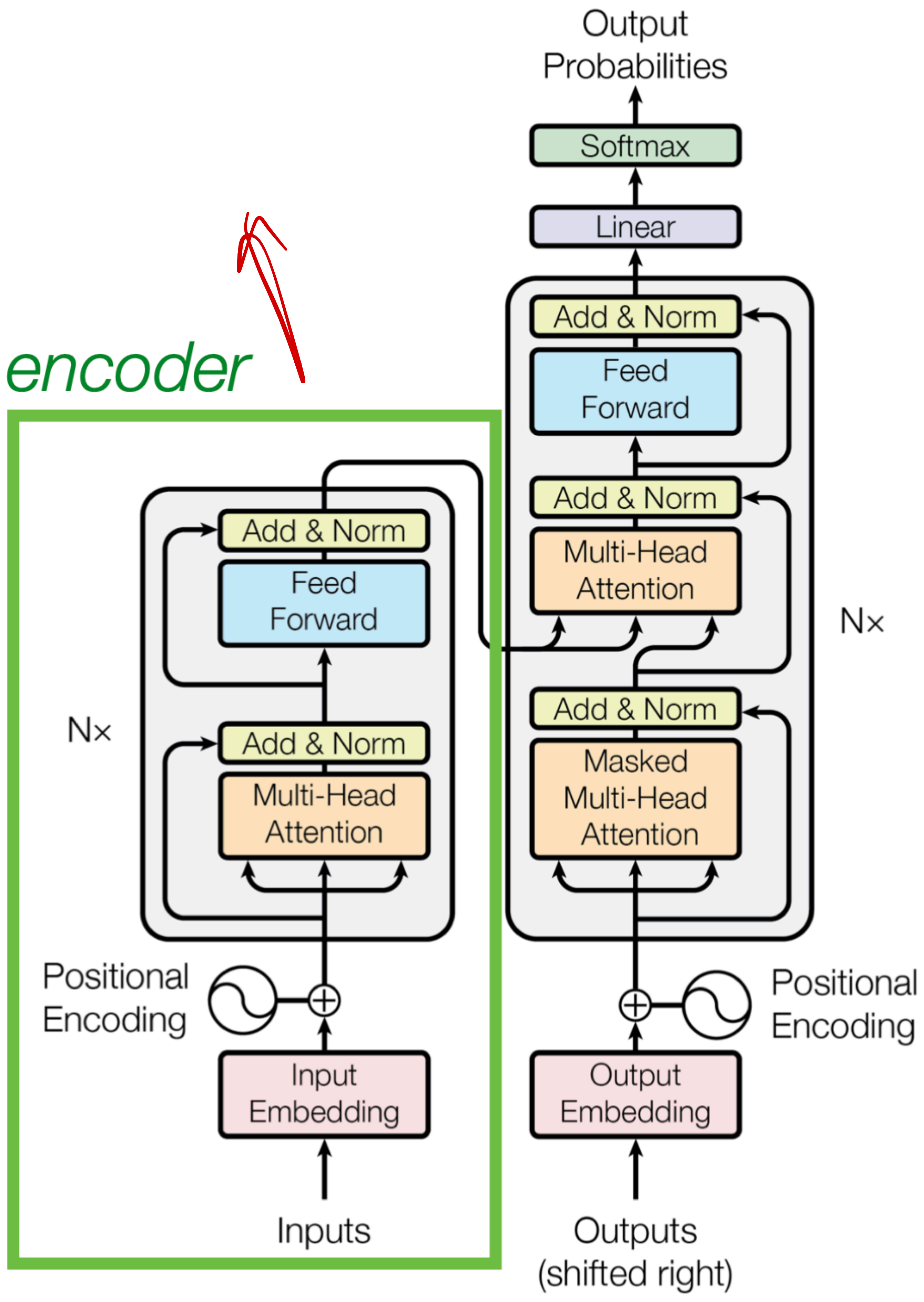


Slides by Emma Strubell

Multi-head Self-Attention + FFN

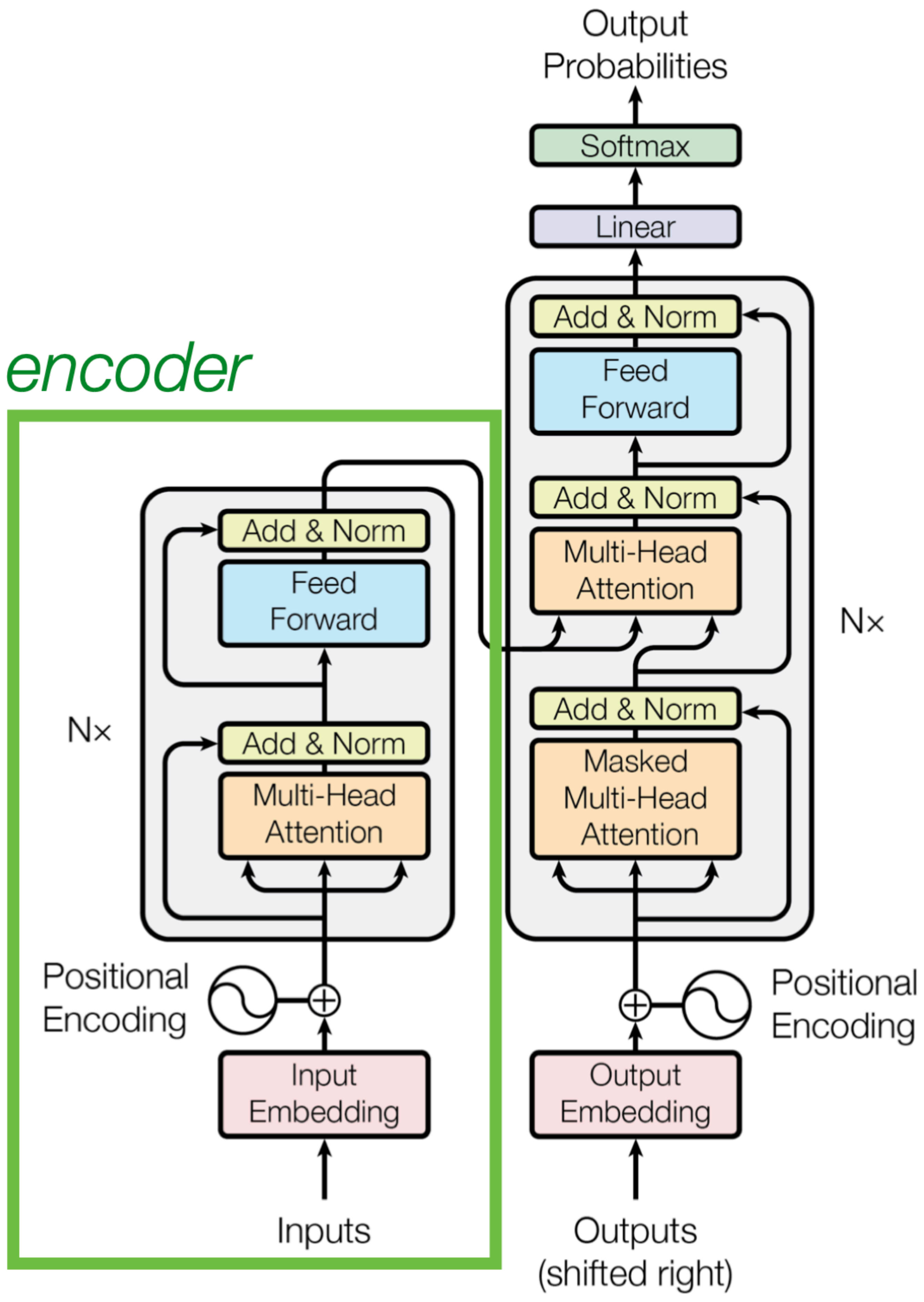


Transformer Encoder

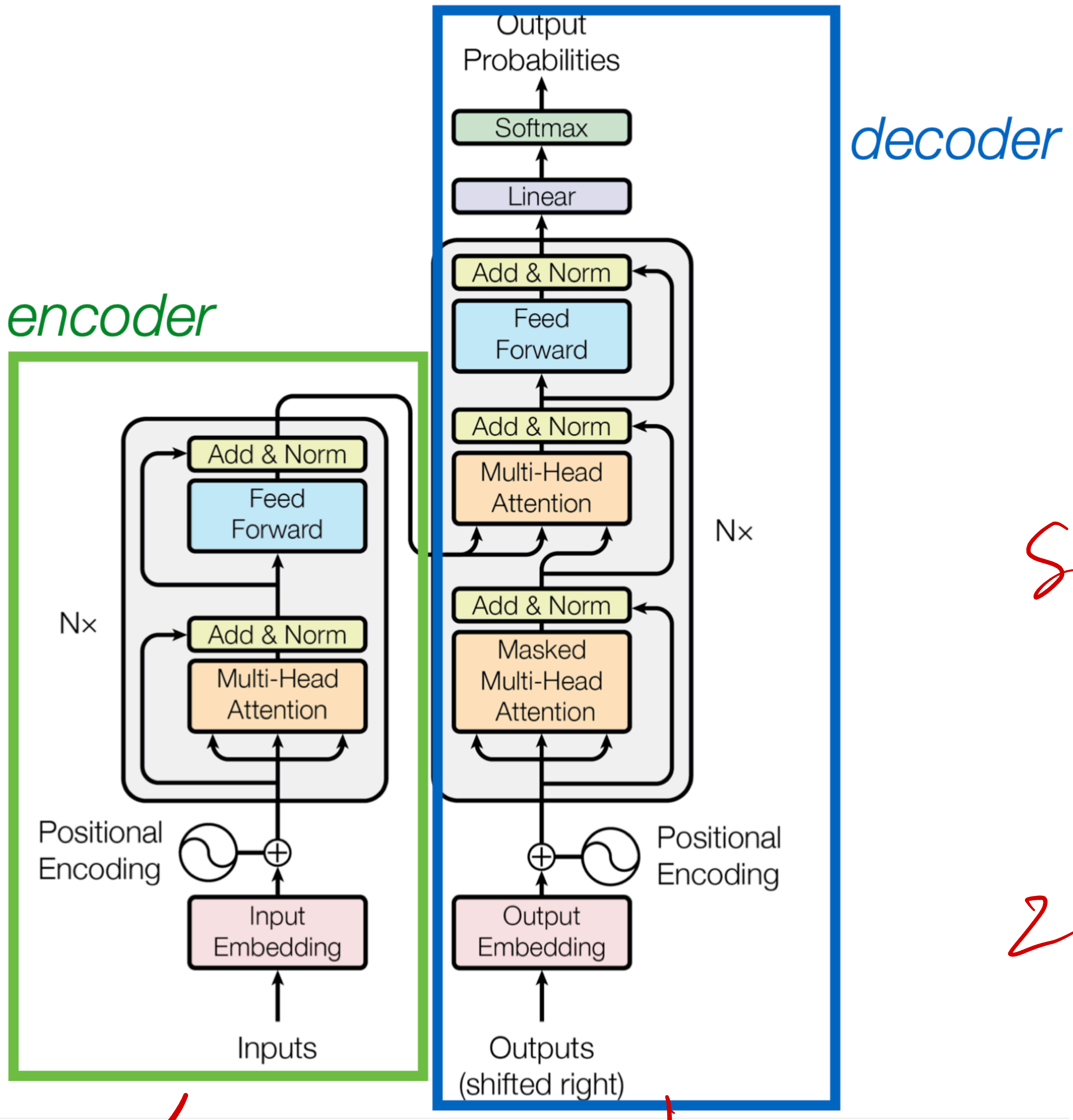


Currently we only cover the encoder side

Transformer Encoder



Currently we only cover the encoder side

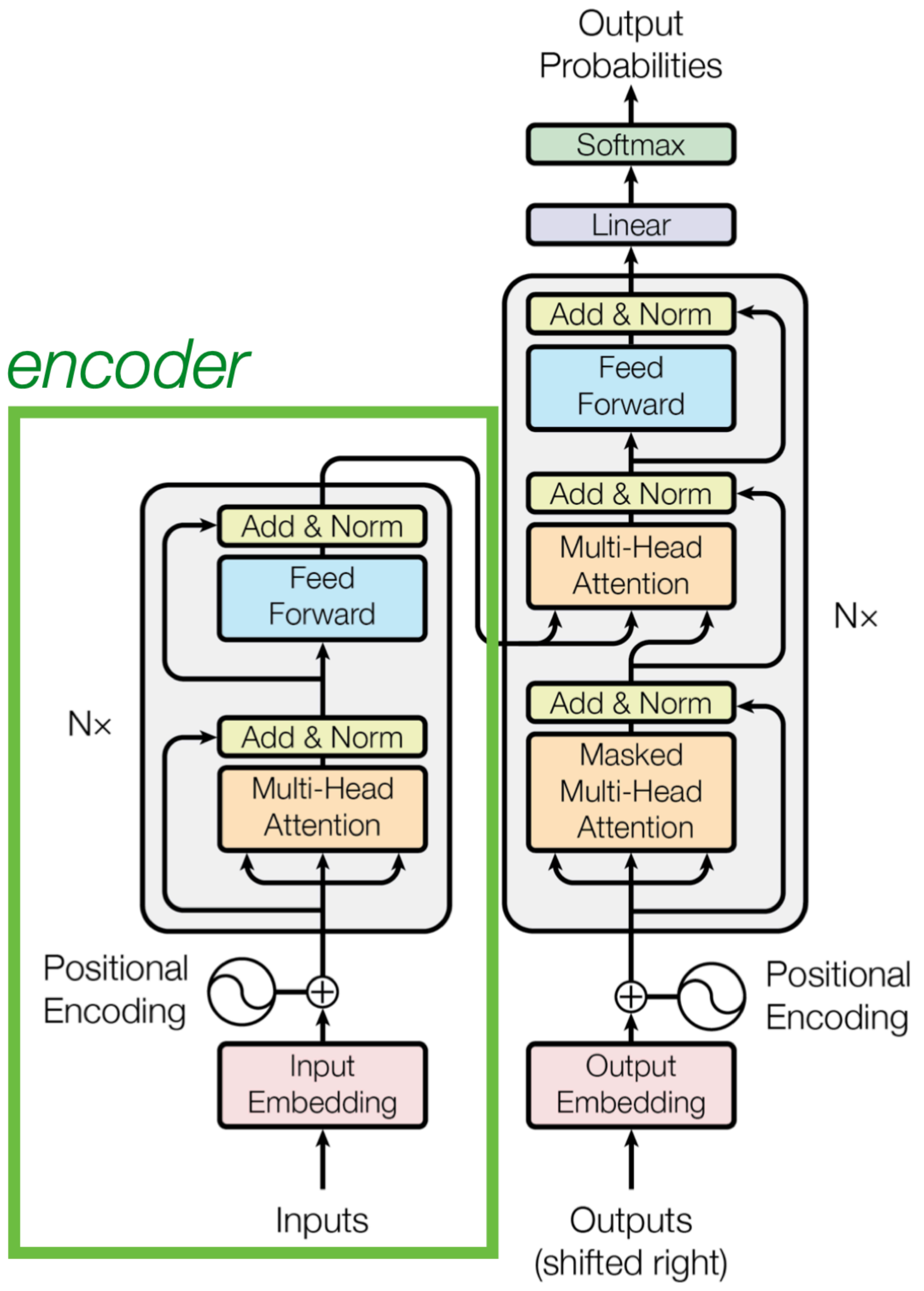


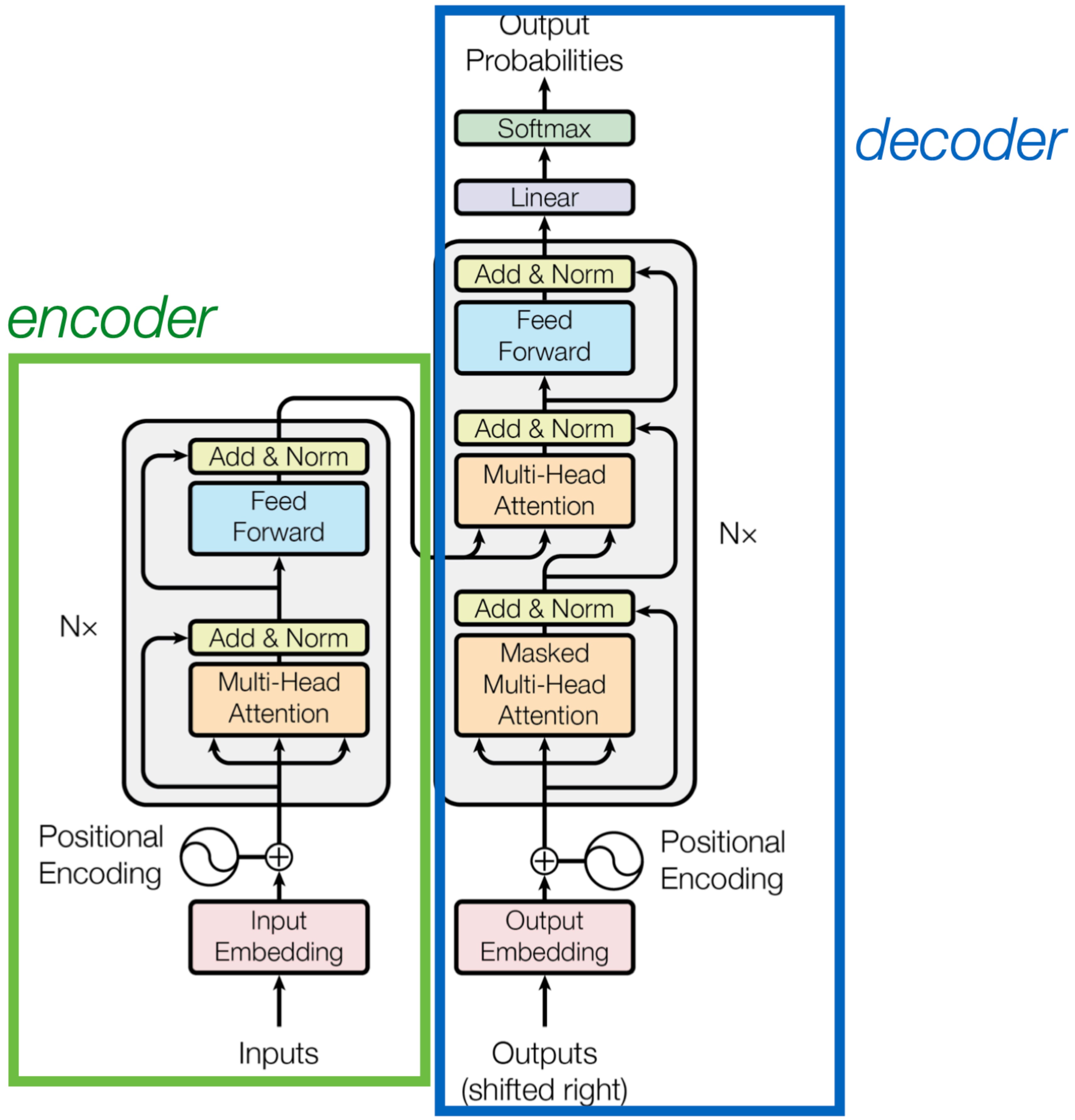
seq2seq

2RNN

Transformer Encoder

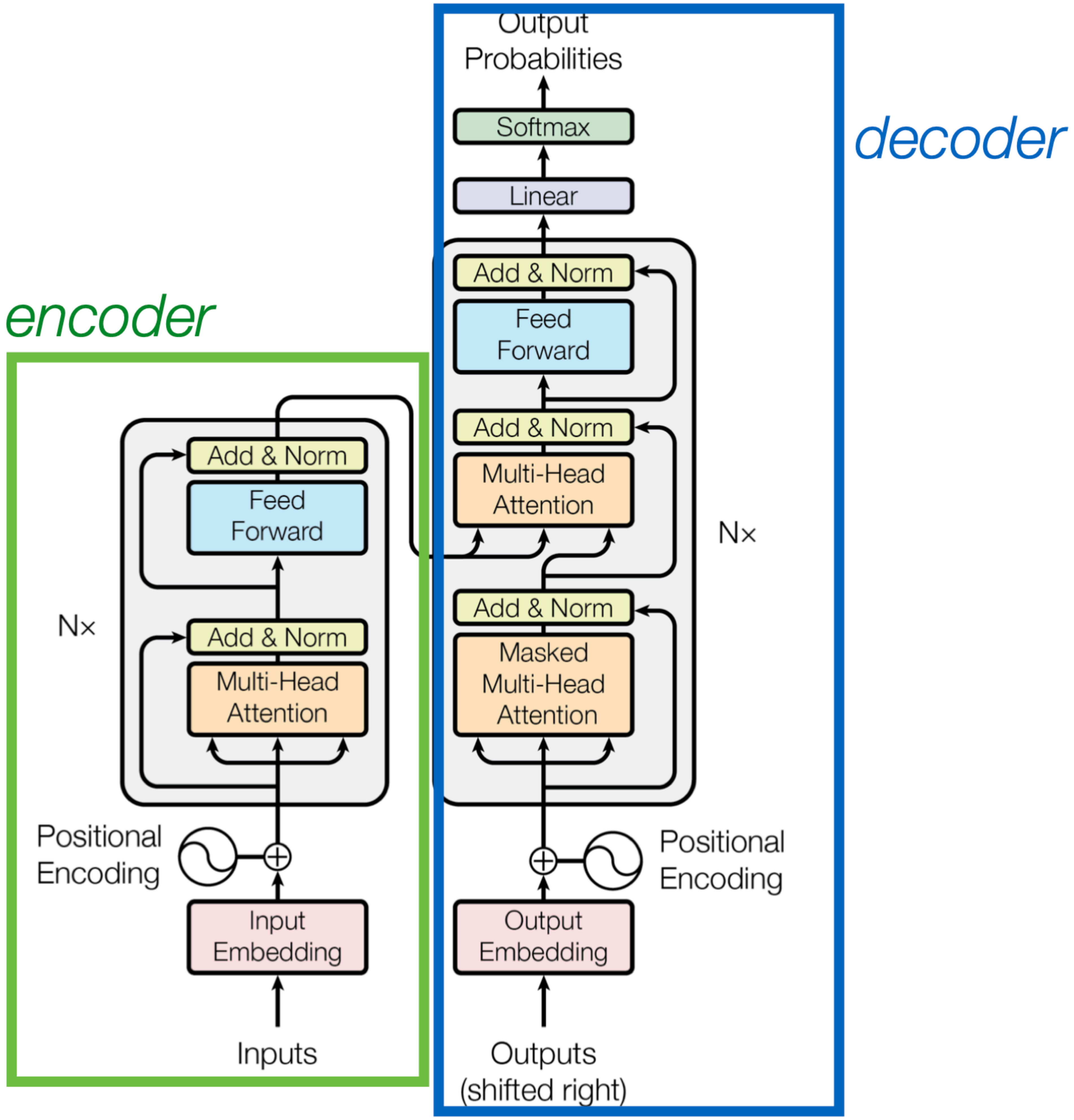
Currently we only cover the encoder side



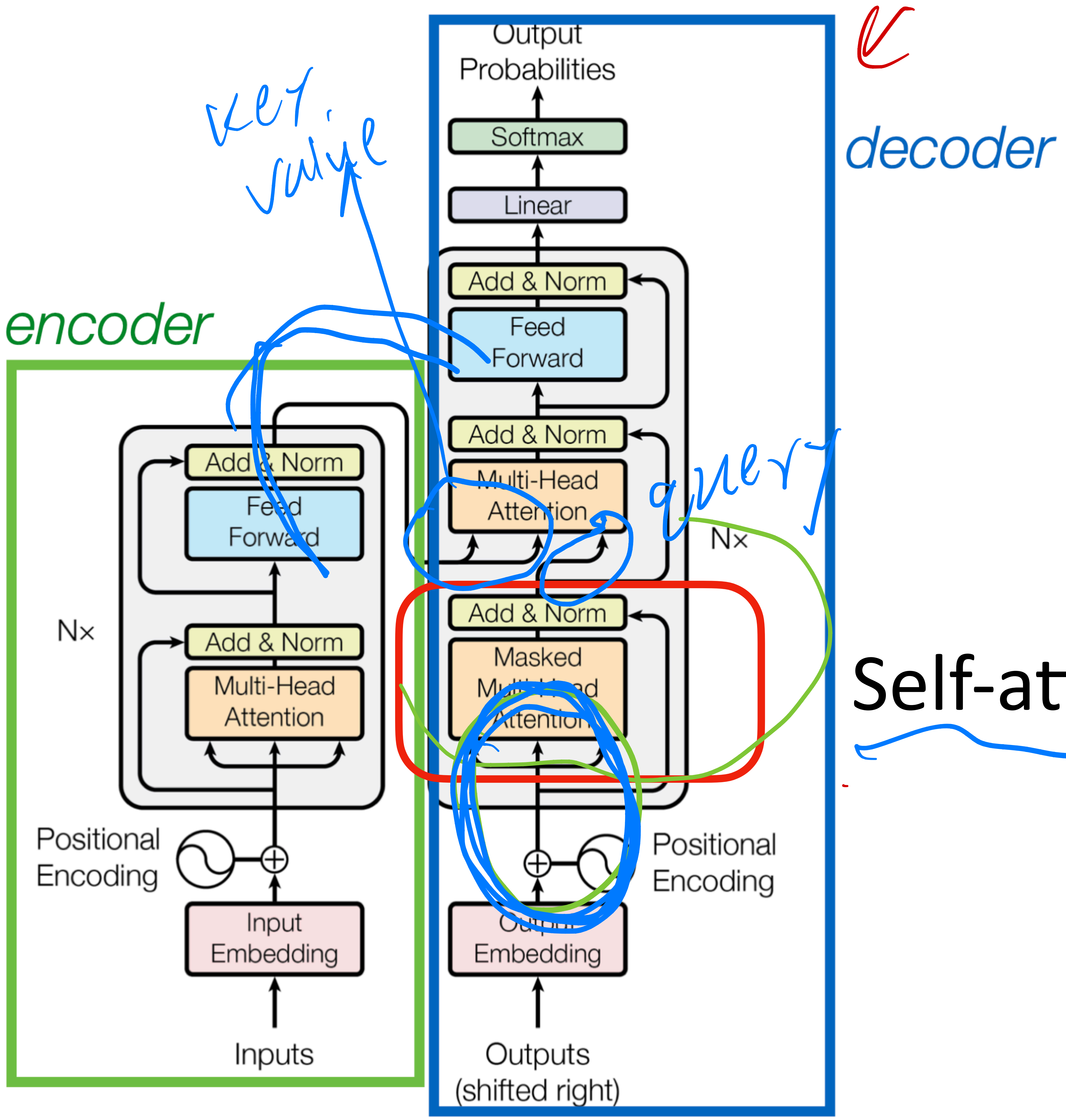


This encoder-decoder arch is originally proposed as a seq2seq arch, for classification tasks, often only encoder is used. And language models often only have a decoder

Transformer Decoder in Seq2Seq

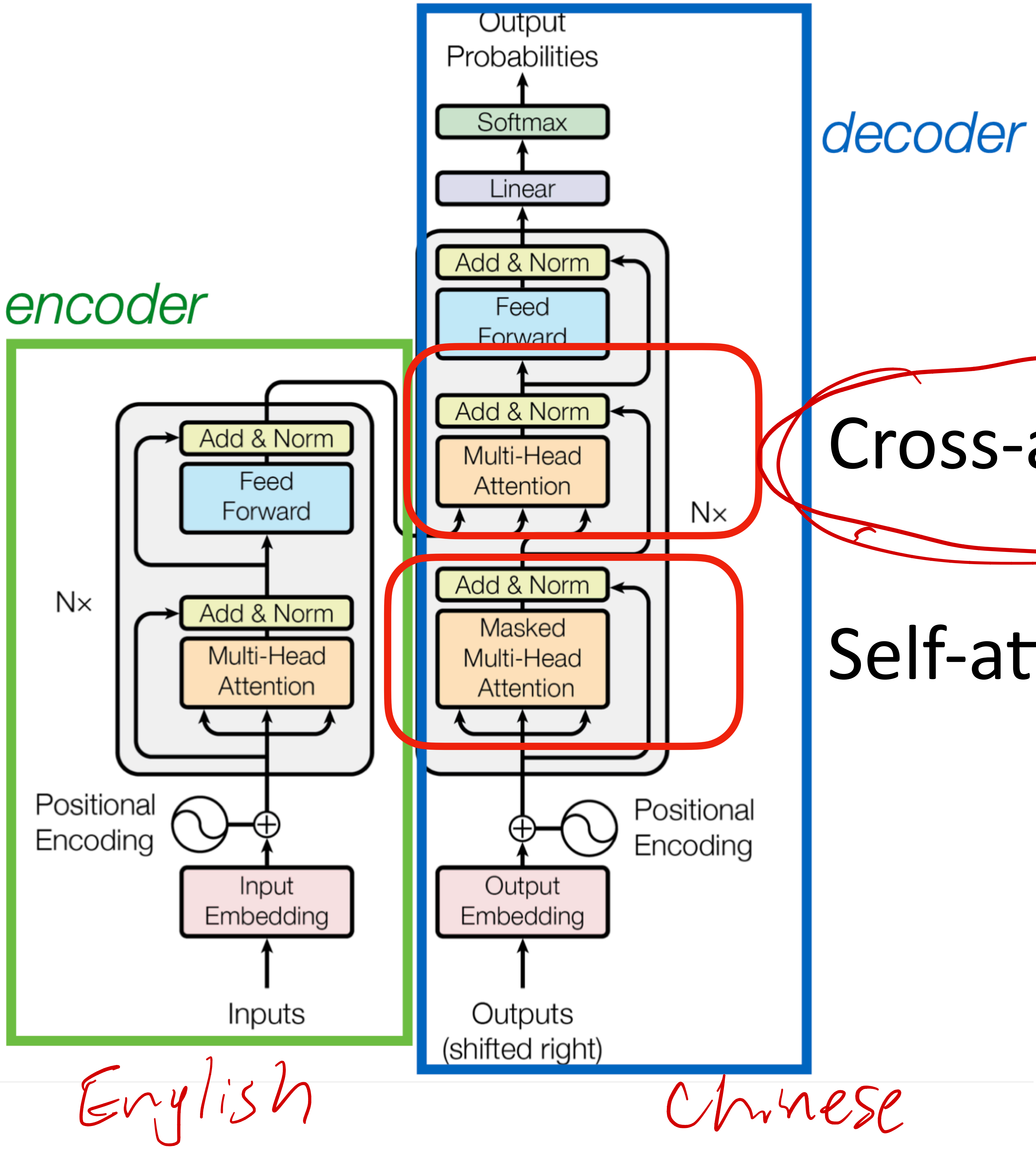


Transformer Decoder in Seq2Seq



Self-attention

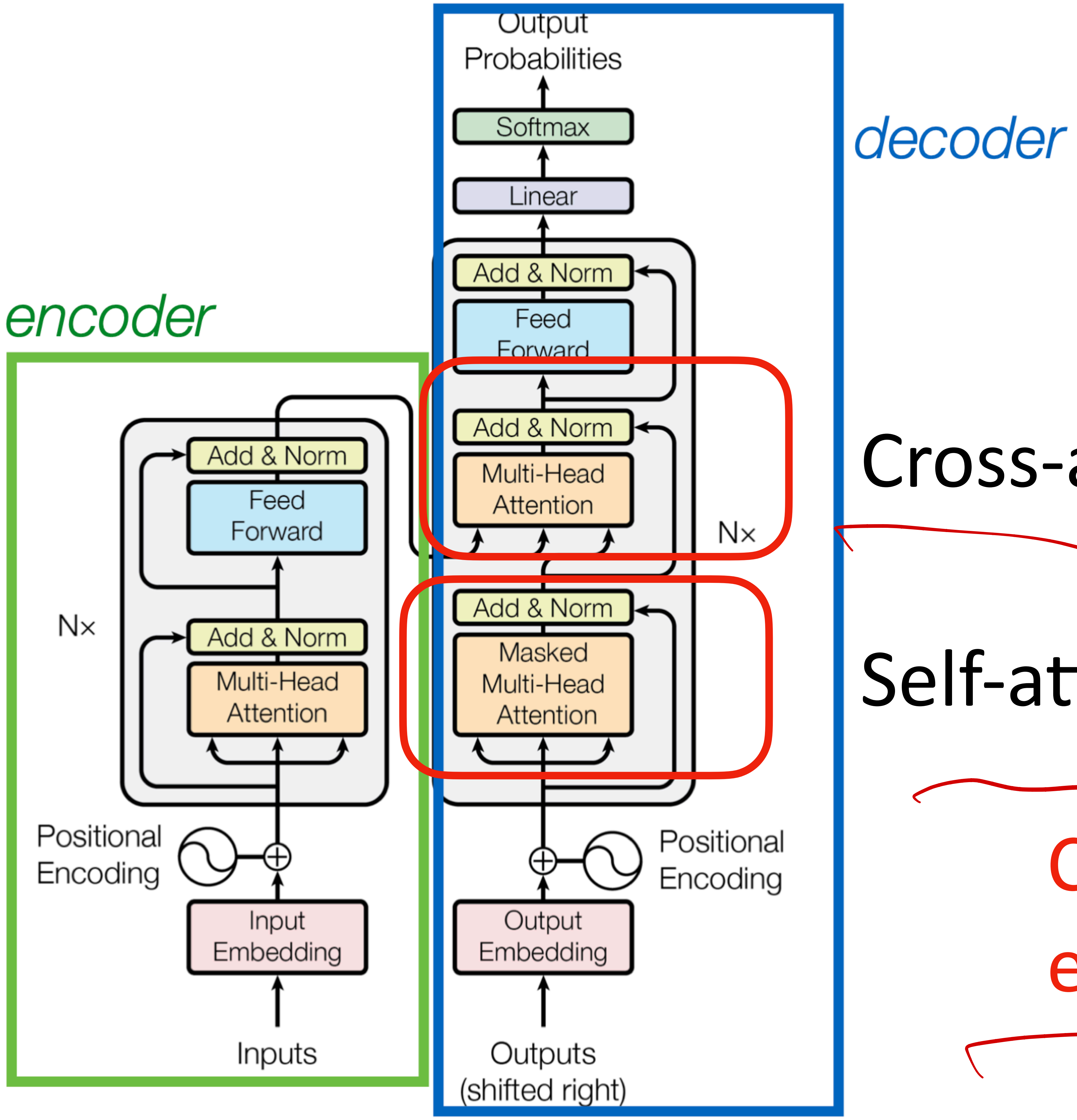
Transformer Decoder in Seq2Seq



Cross-attention

Self-attention

Transformer Decoder in Seq2Seq

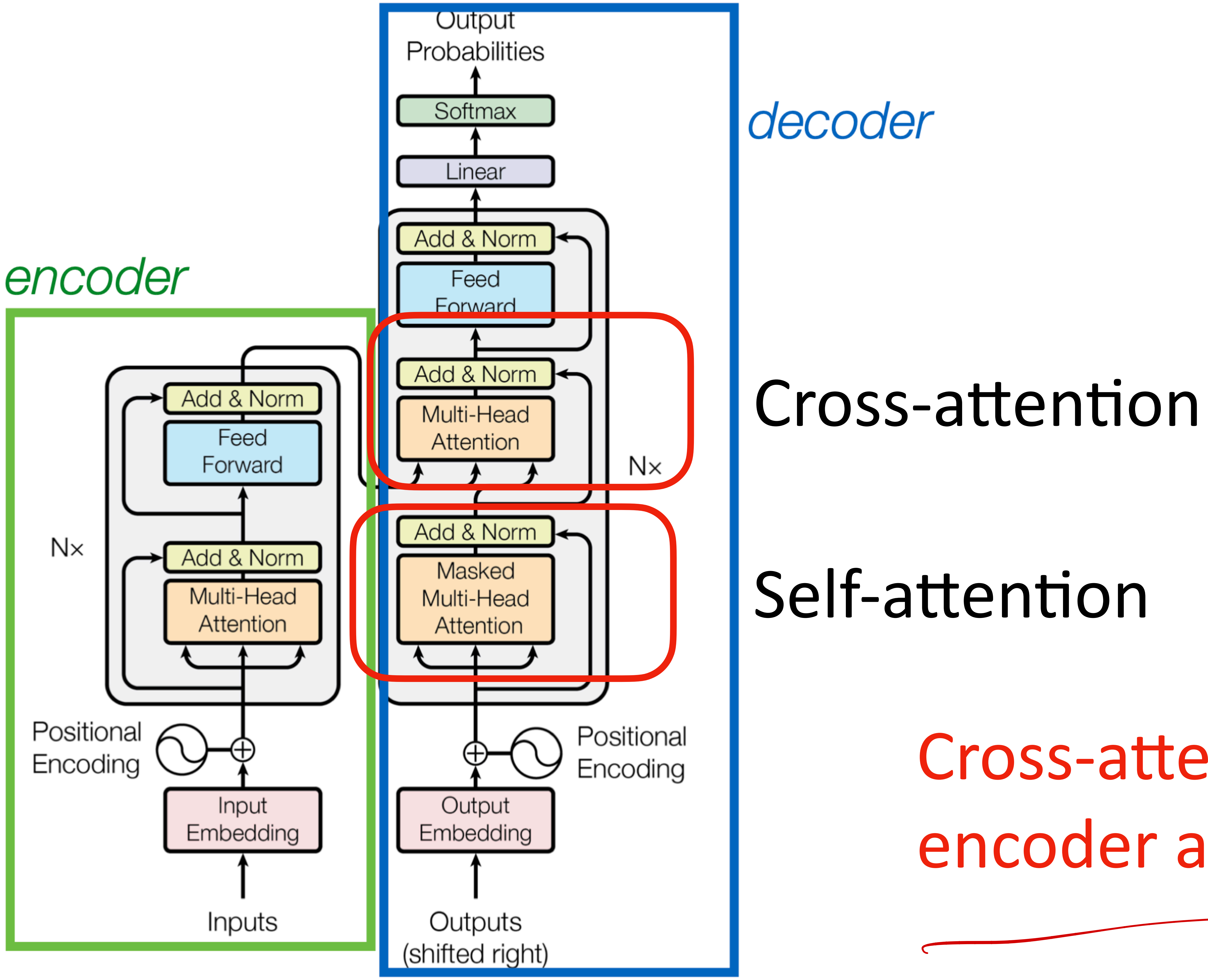


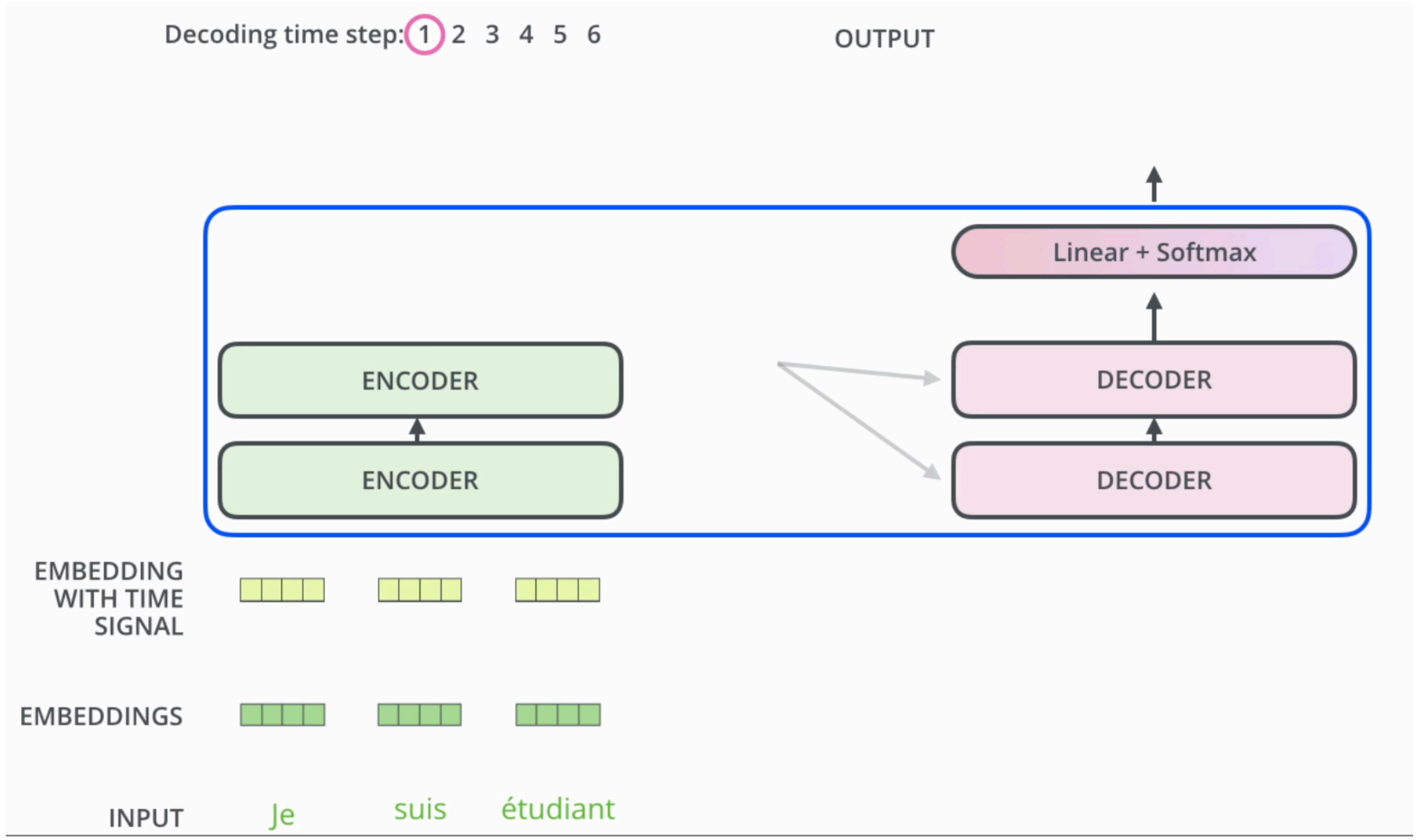
Cross-attention

Self-attention

Cross-attention uses the output of encoder as input

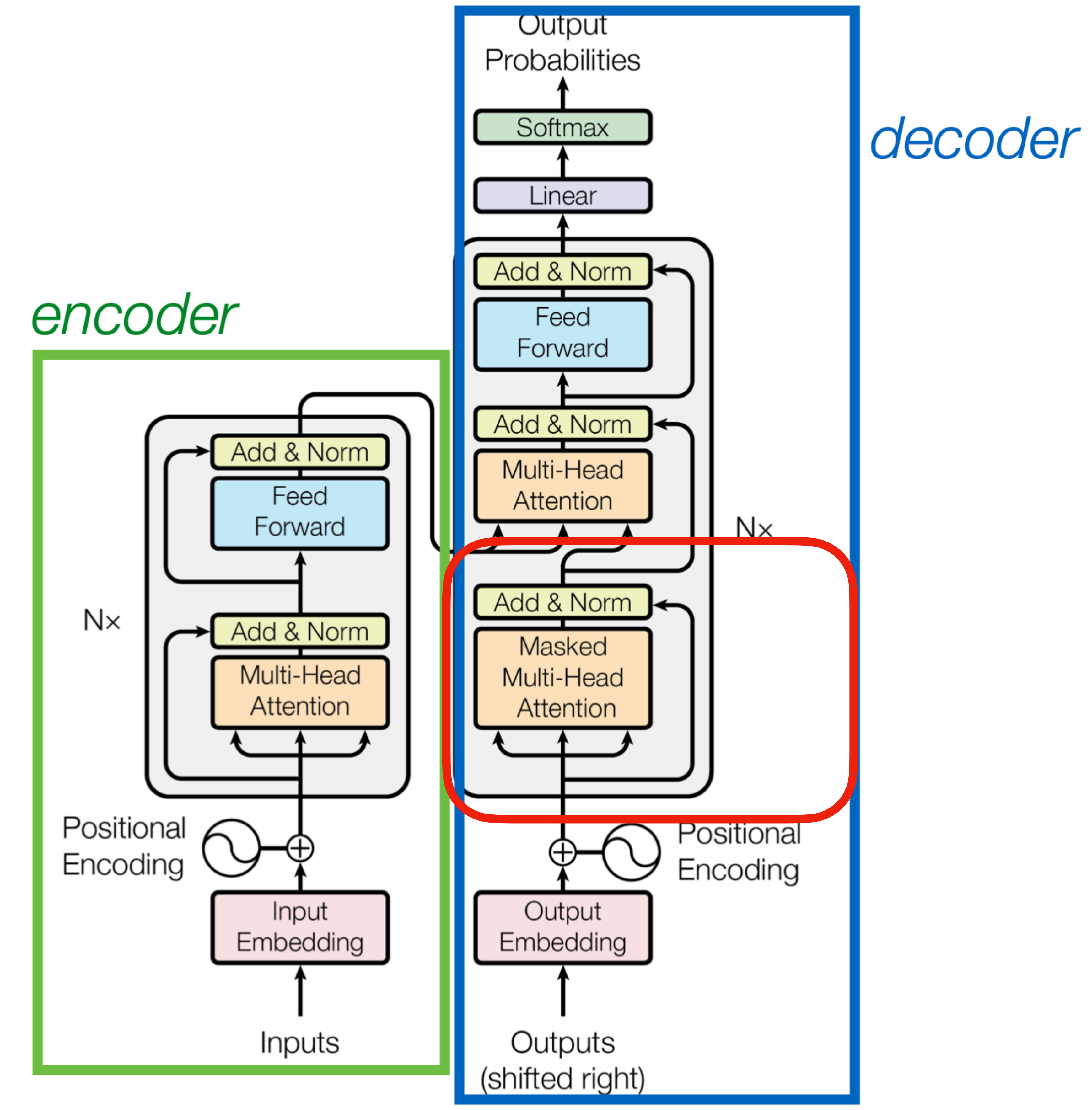
Transformer Decoder in Seq2Seq



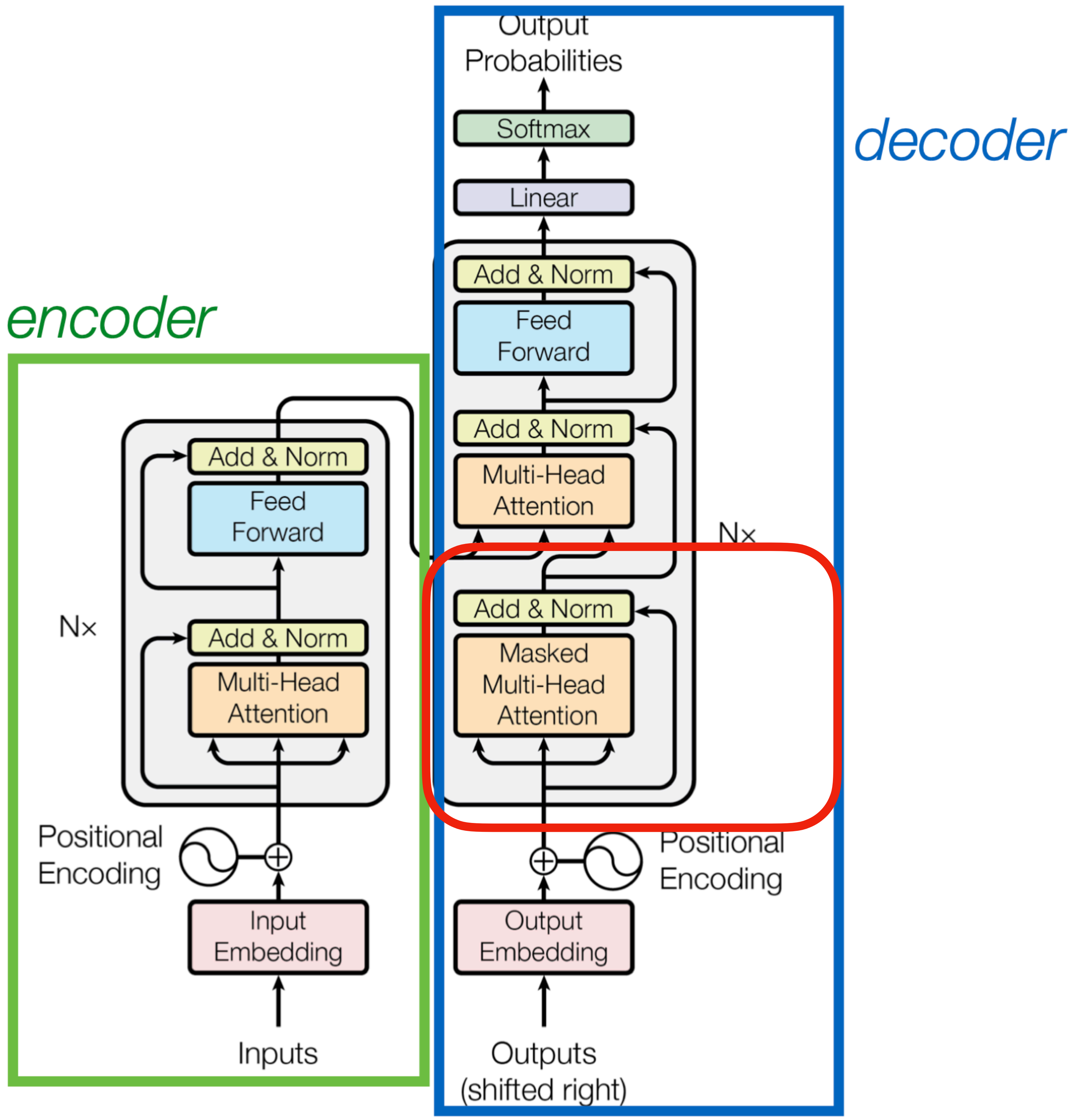


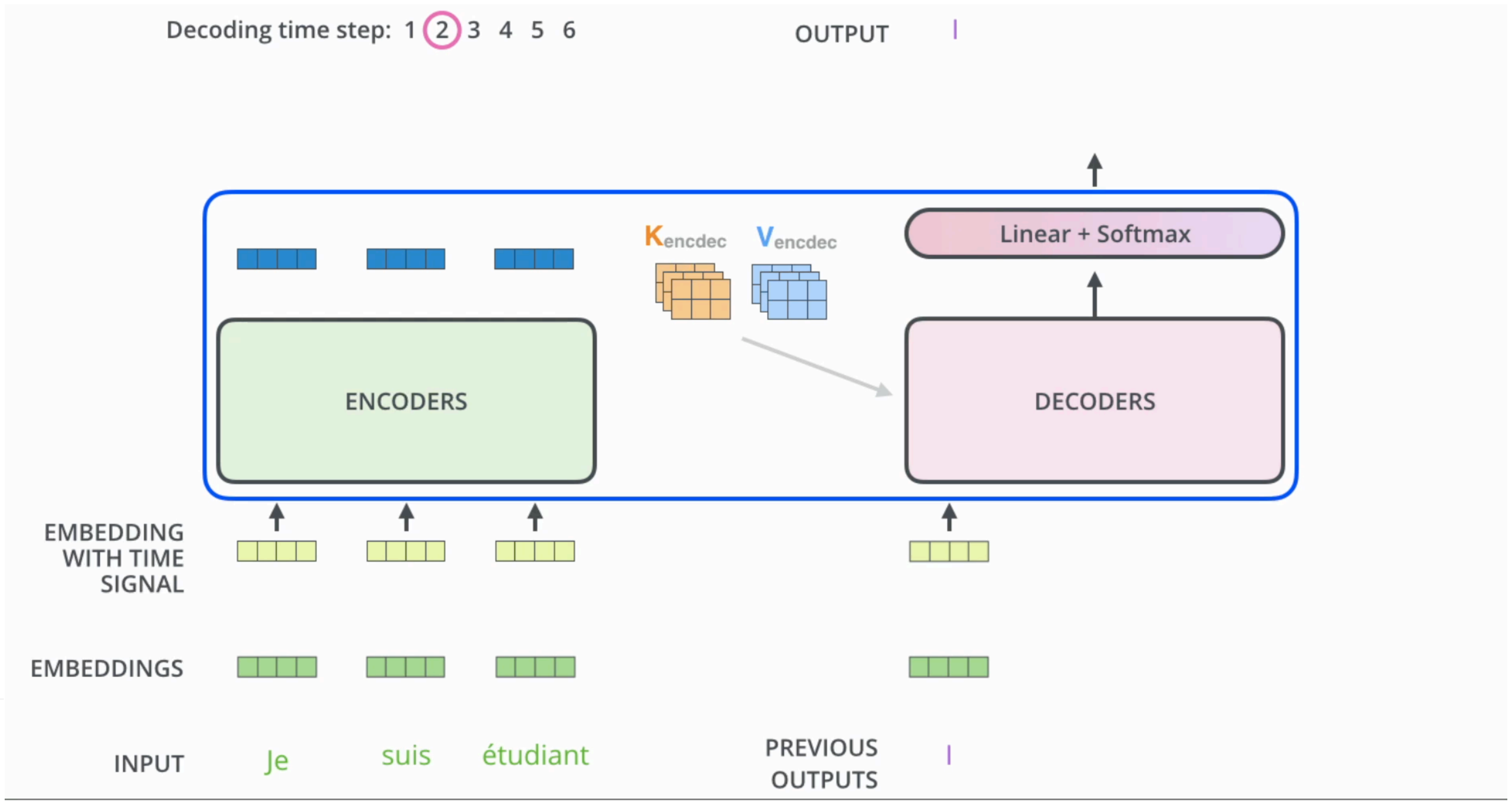
Cross-attention uses the output of encoder as input

Masked Attention



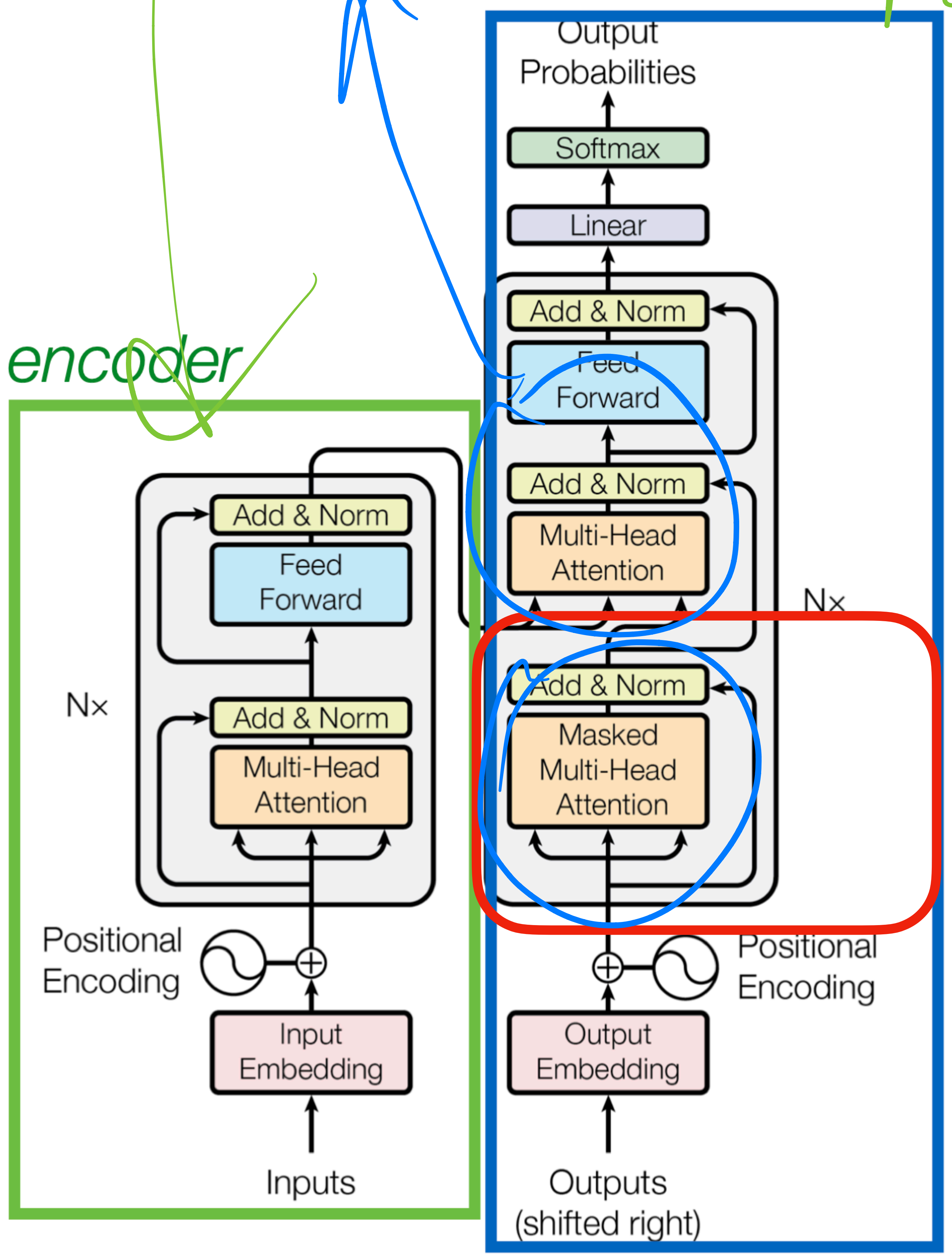
Masked Attention

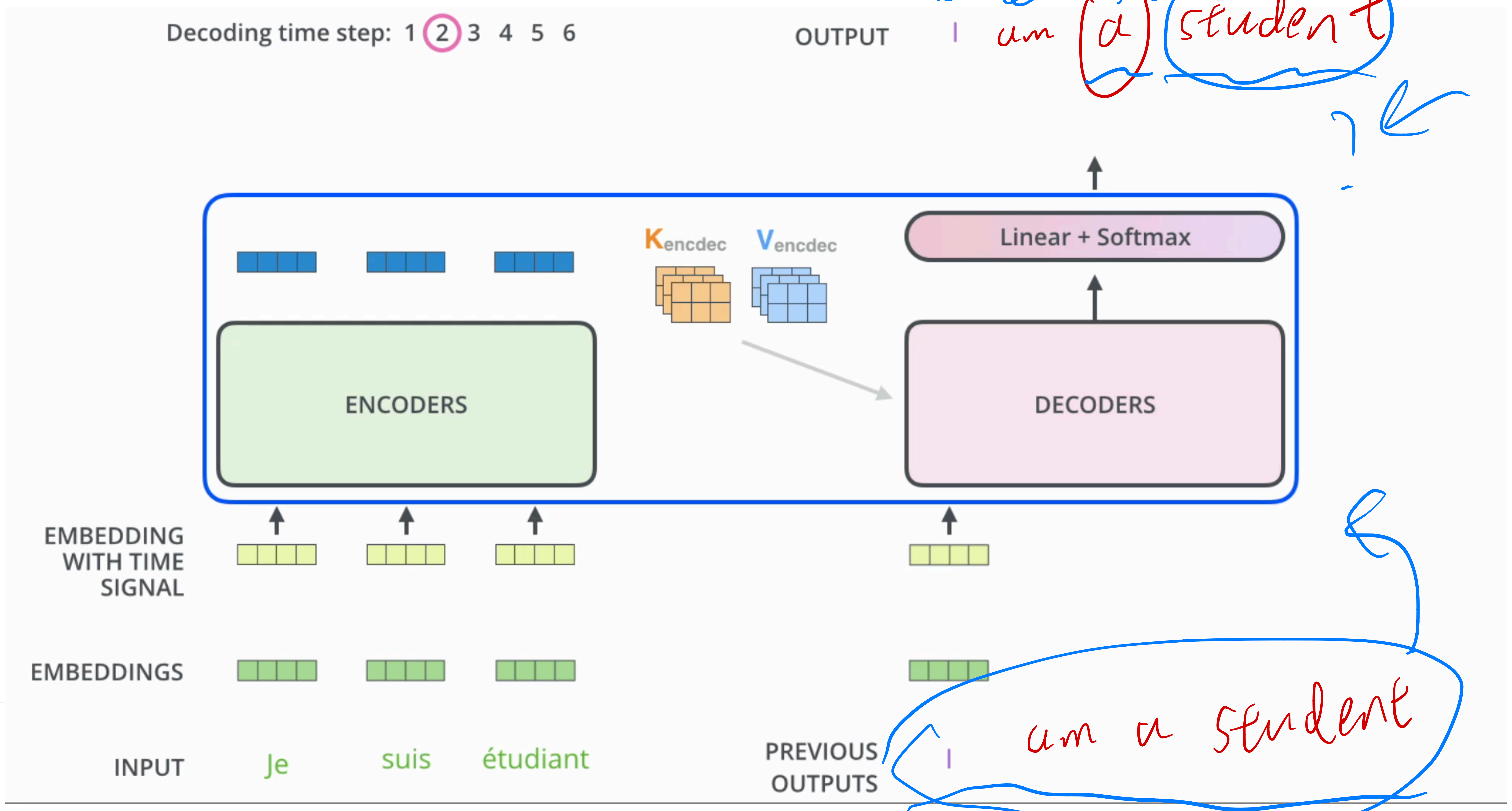




Masked Attention

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



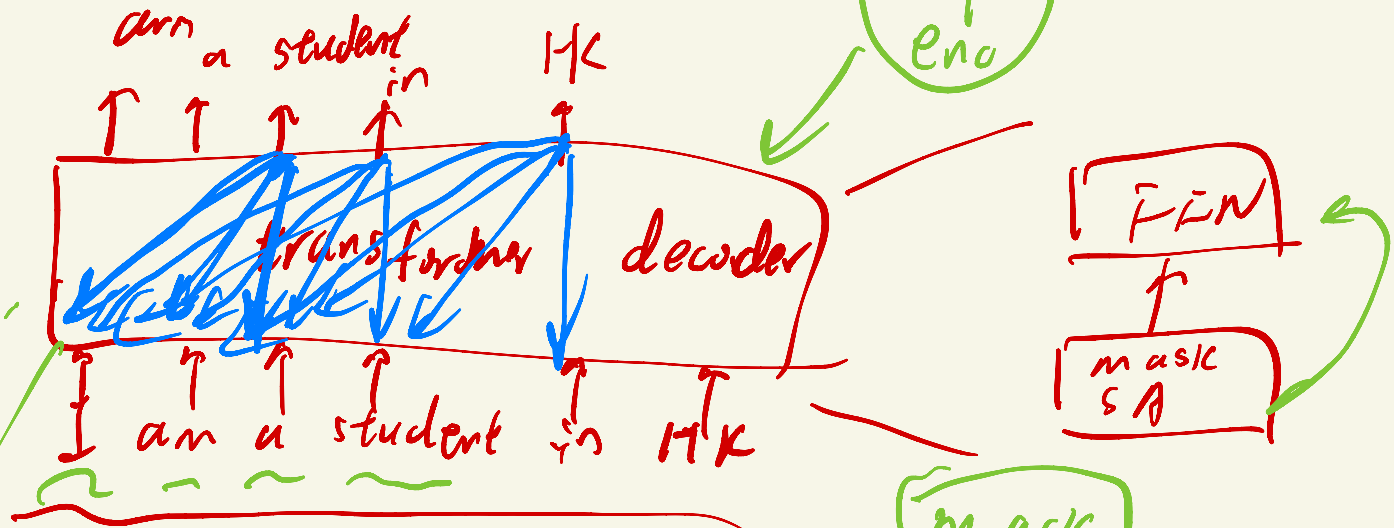


ChatGPT

next word prediction

mask

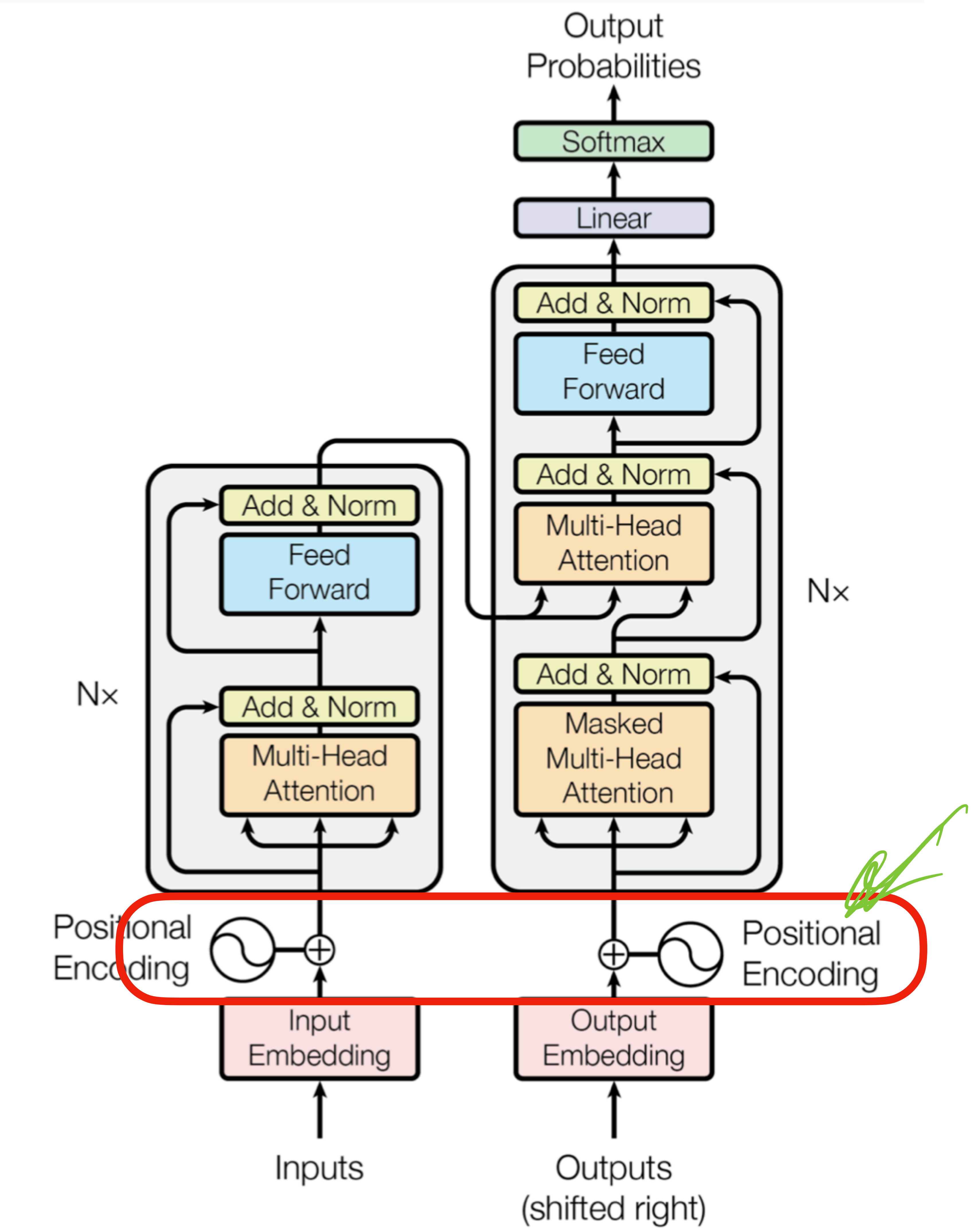
dec ↑ enc



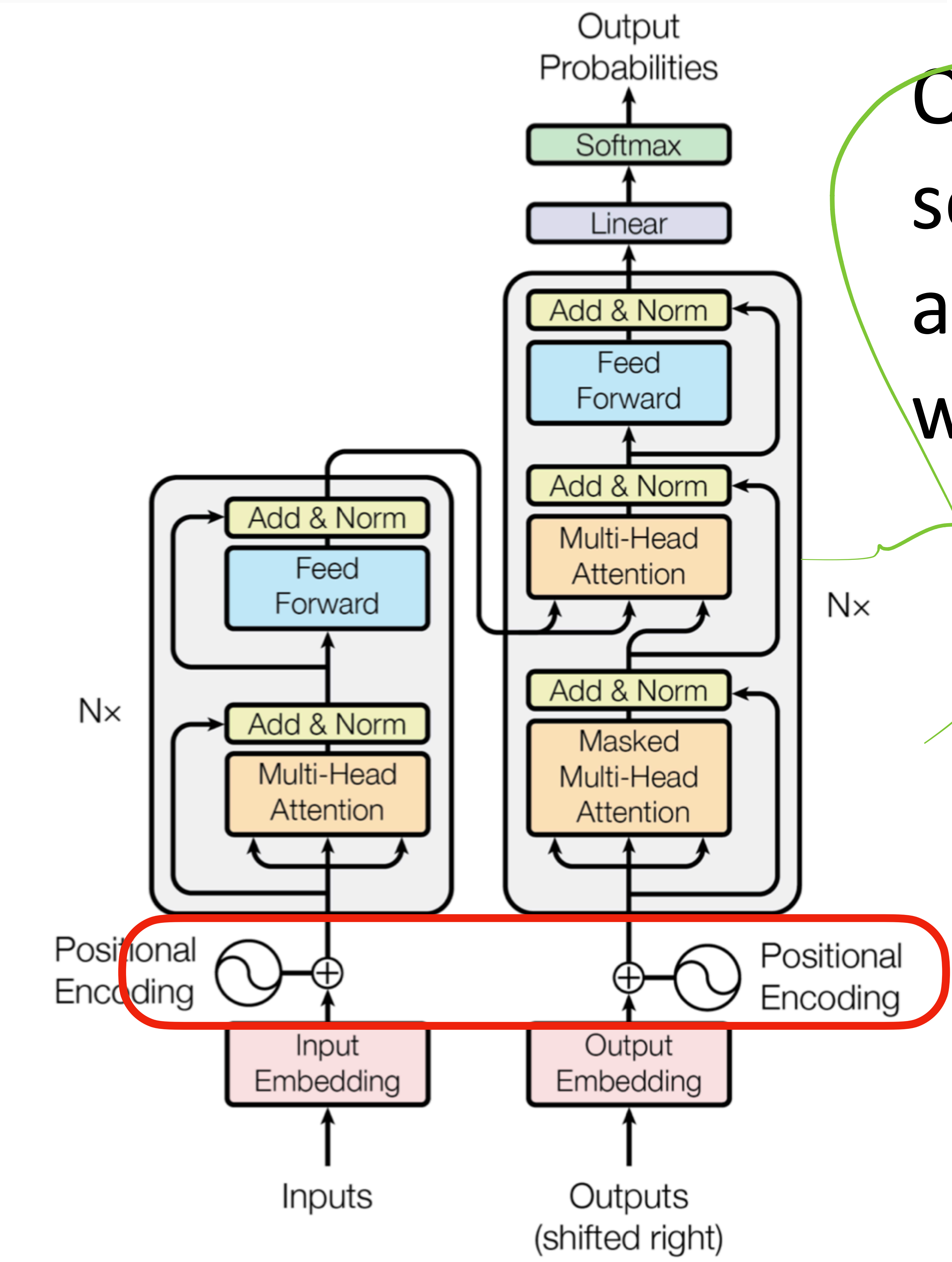
mask

Uni directional attention

Position Embeddings



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?

z_1 z_2 z_3 z_4

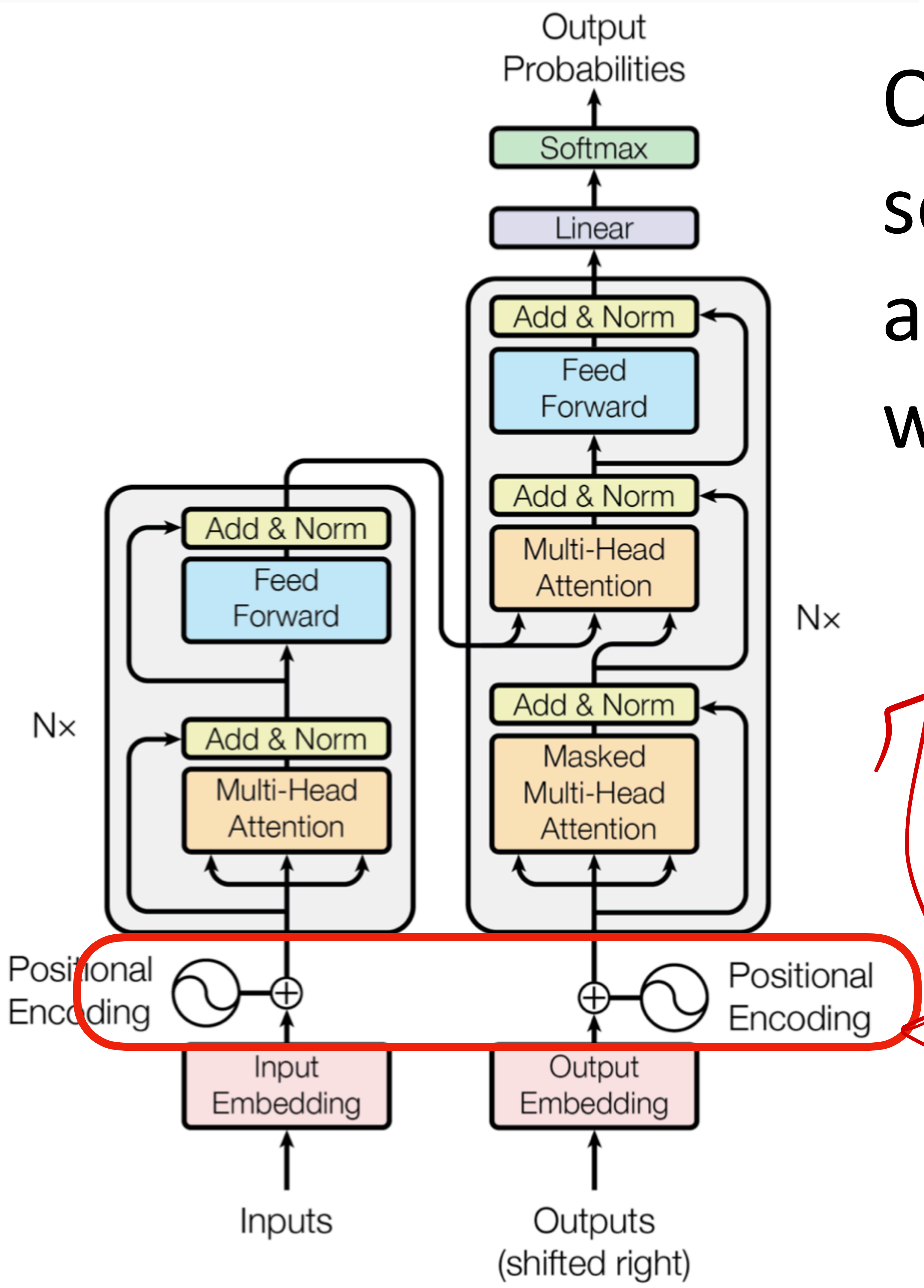
I am a student

$z_4 = z_1'$

z_1' z_2' z_3' z_4'

student a am I

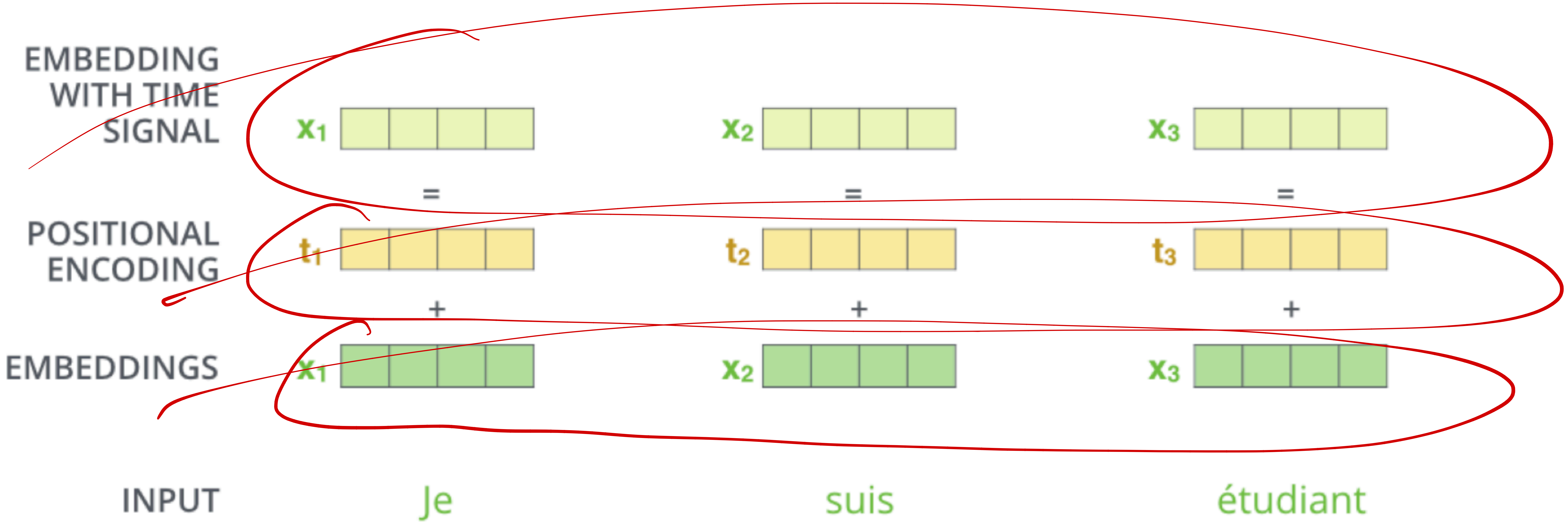
Position Embeddings



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

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

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

Positional encoding is a 512d vector

i = a particular dimension of this vector

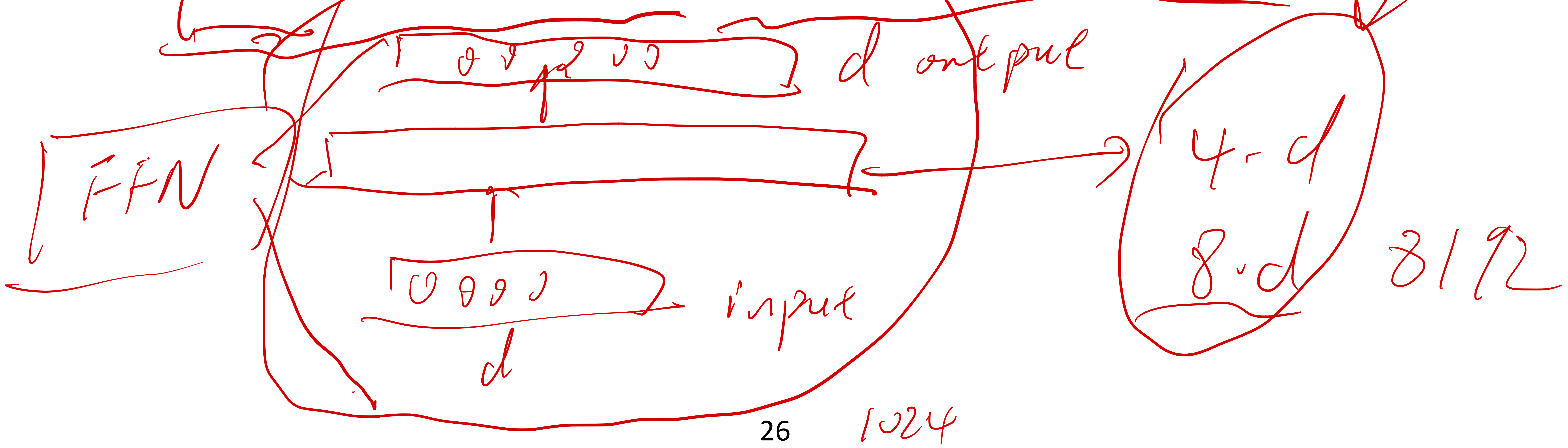
pos = dimension of the word

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.



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

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

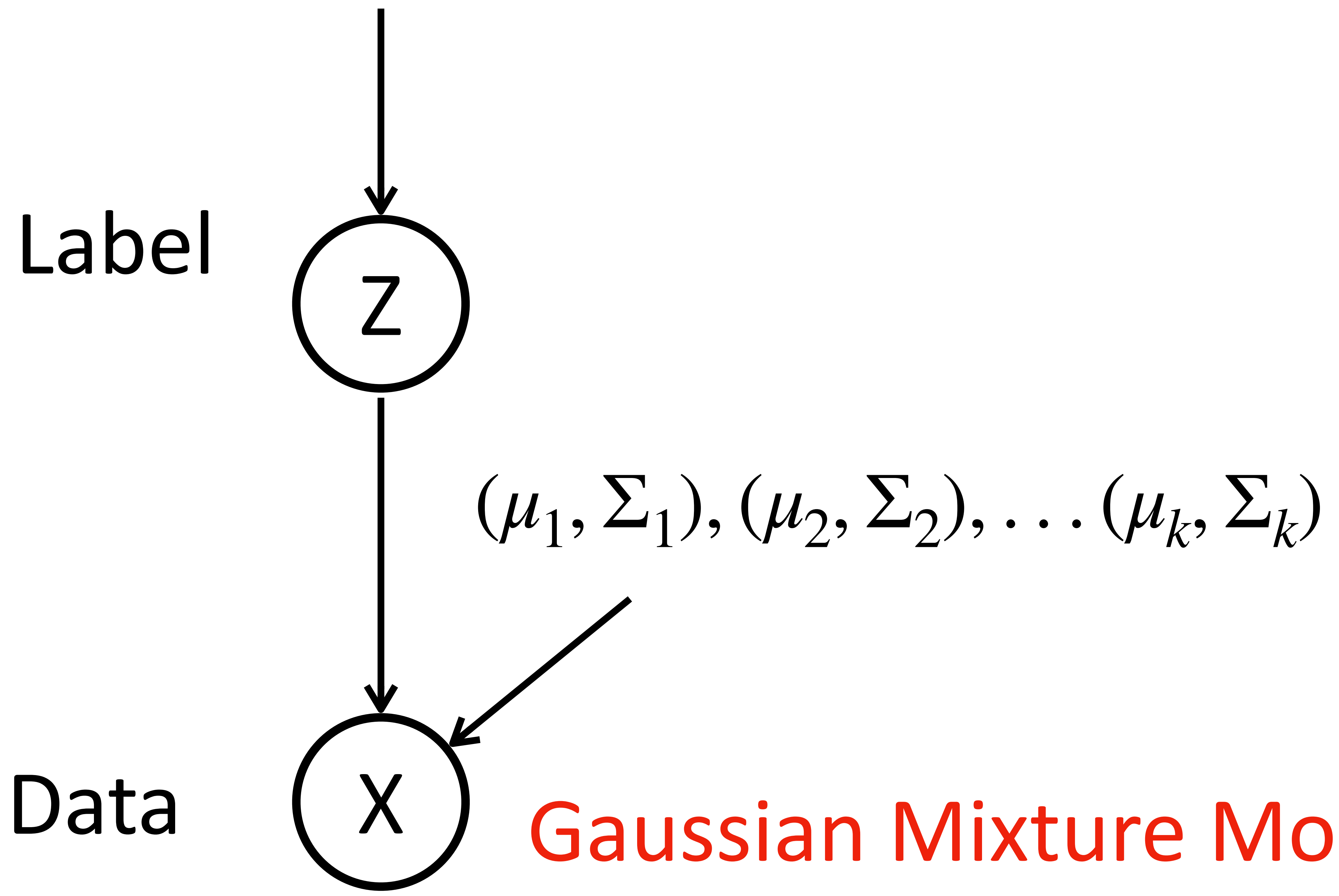
Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

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

Variational Autoencoders

VAE is a Generative Model

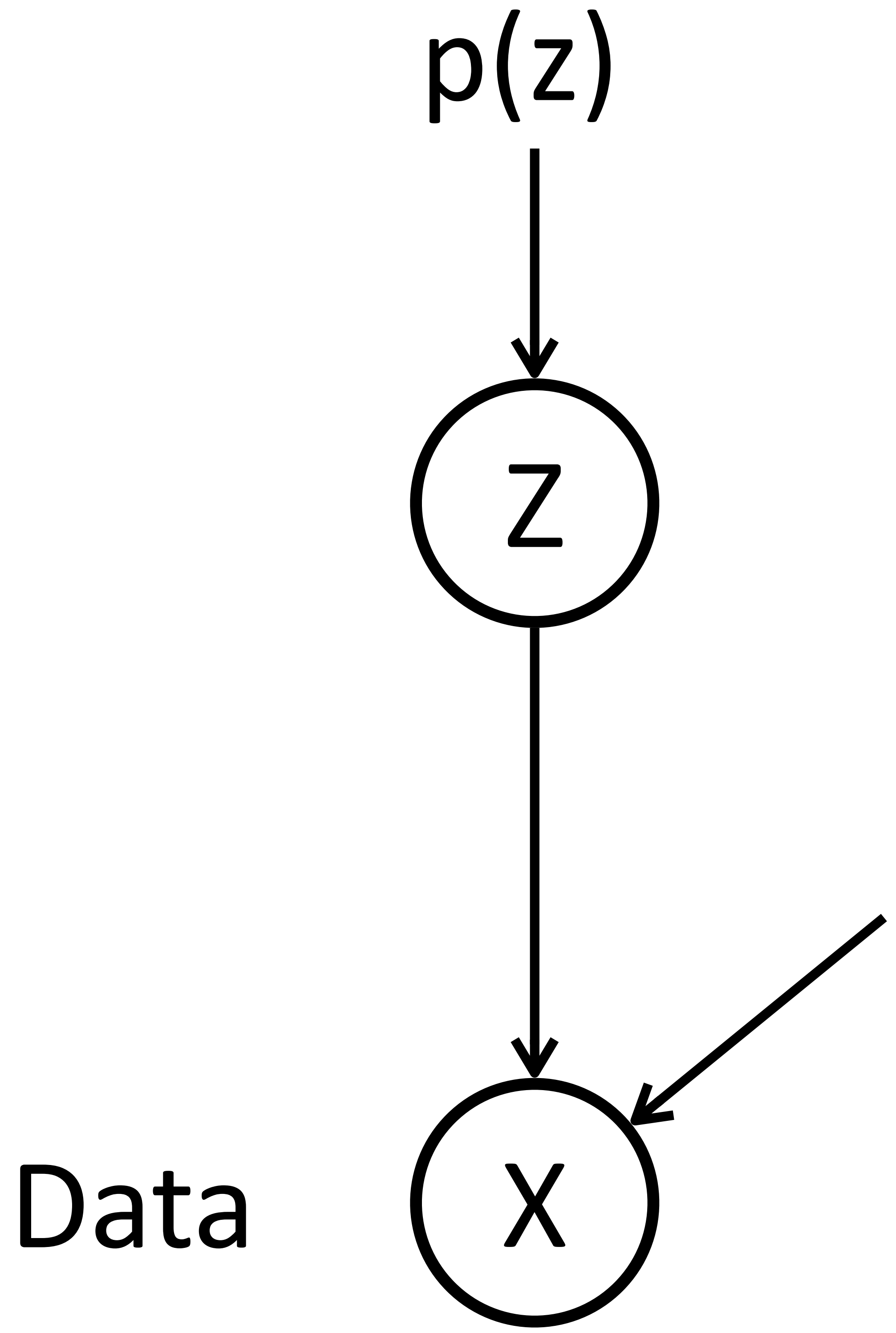
p(z): multinomial , k classes(e.g. uniform)



Gaussian Mixture Model (GMM)

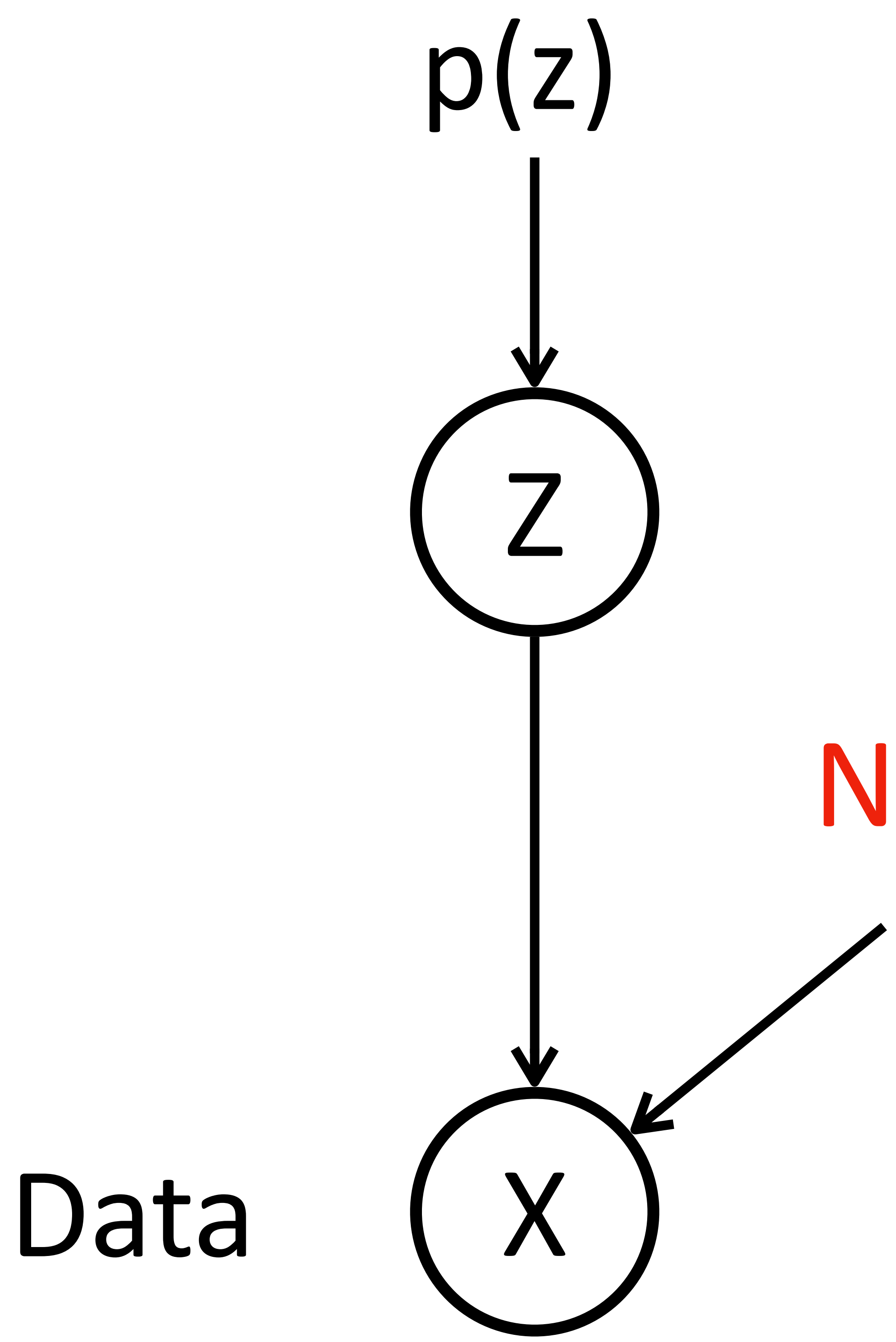
The VAE Model

p(z) is a normal distribution in most cases



The VAE Model

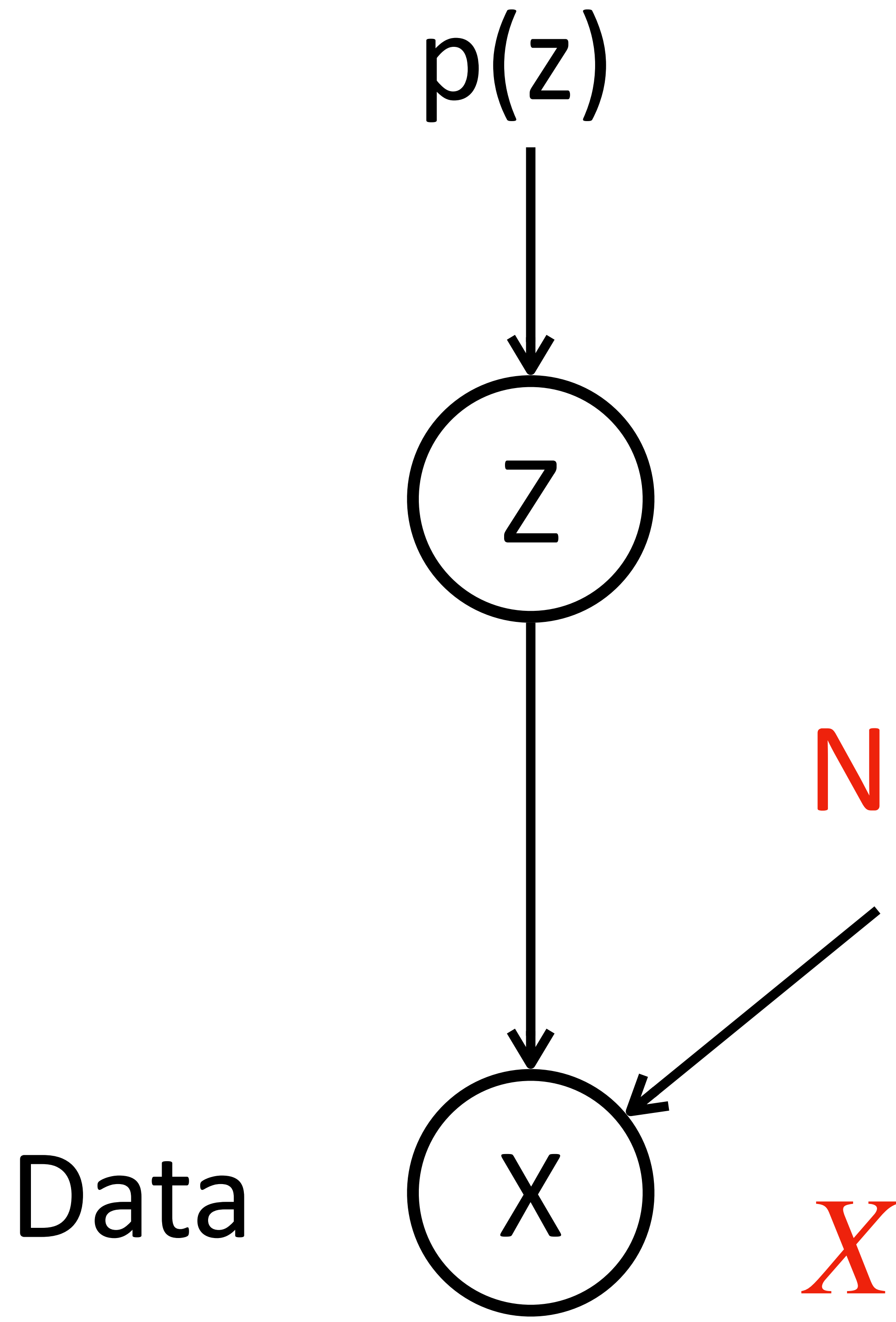
p(z) is a normal distribution in most cases



Neural Networks

The VAE Model

p(z) is a normal distribution in most cases

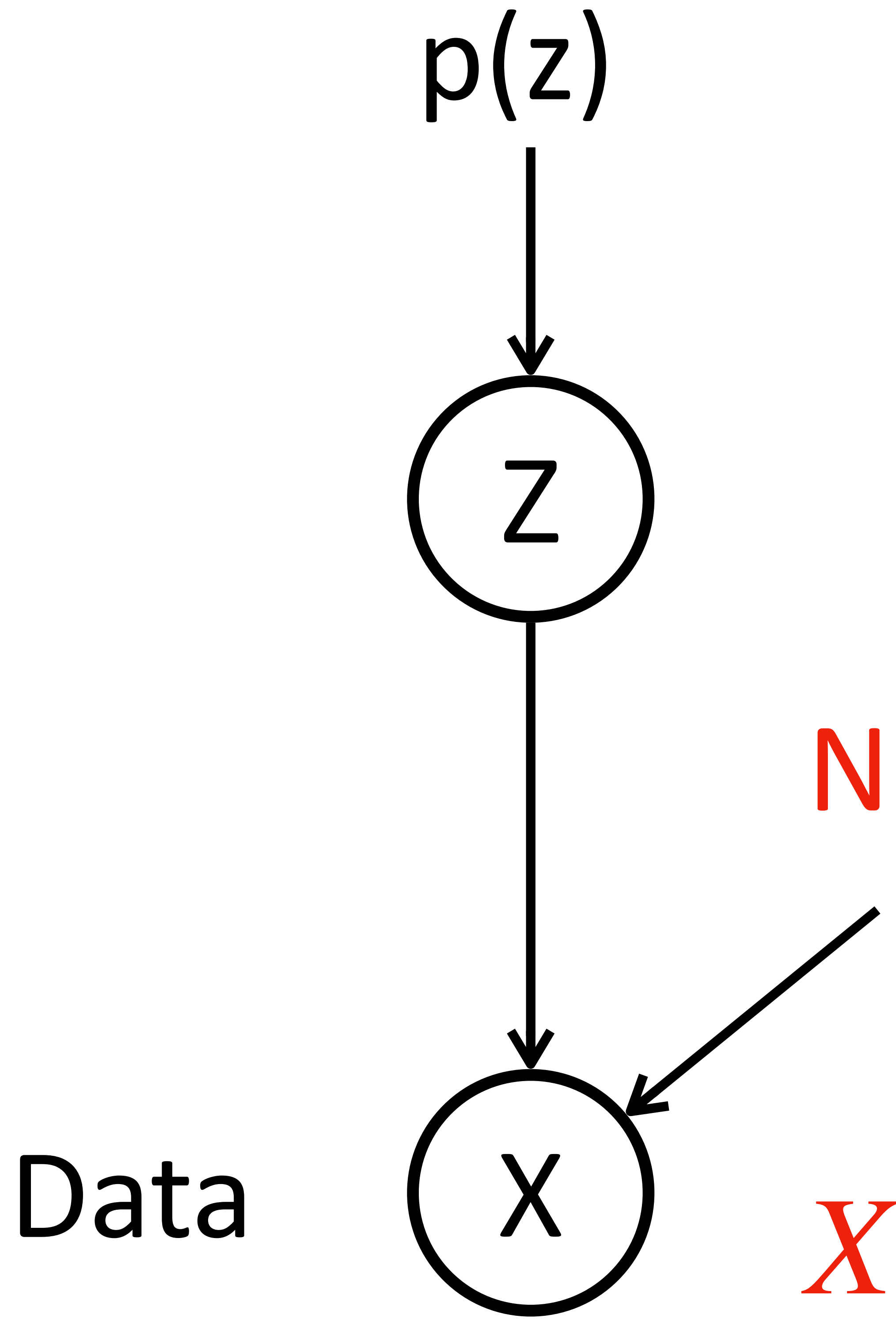


Neural Networks

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

The VAE Model

p(z) is a normal distribution in most cases

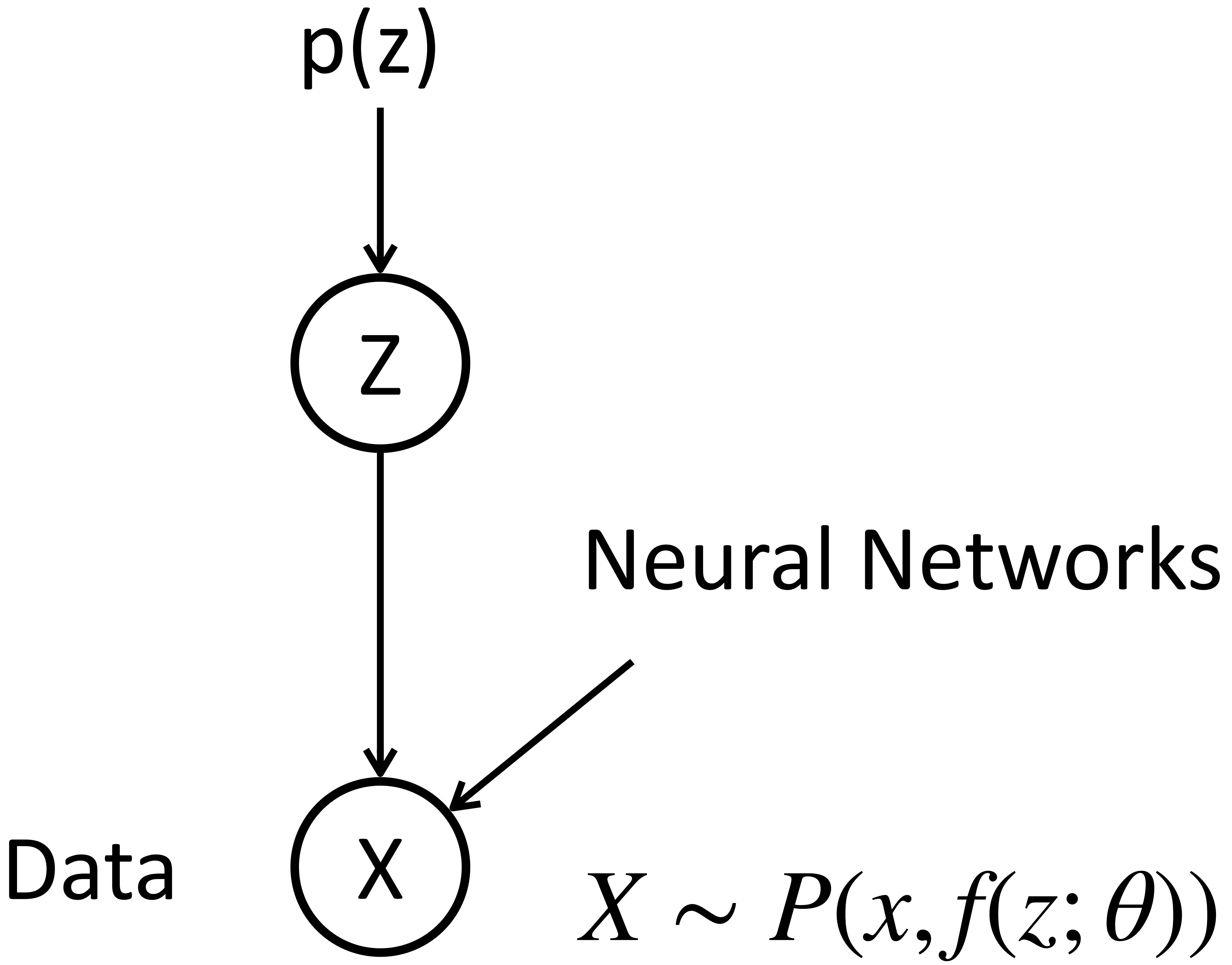


Neural Networks

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

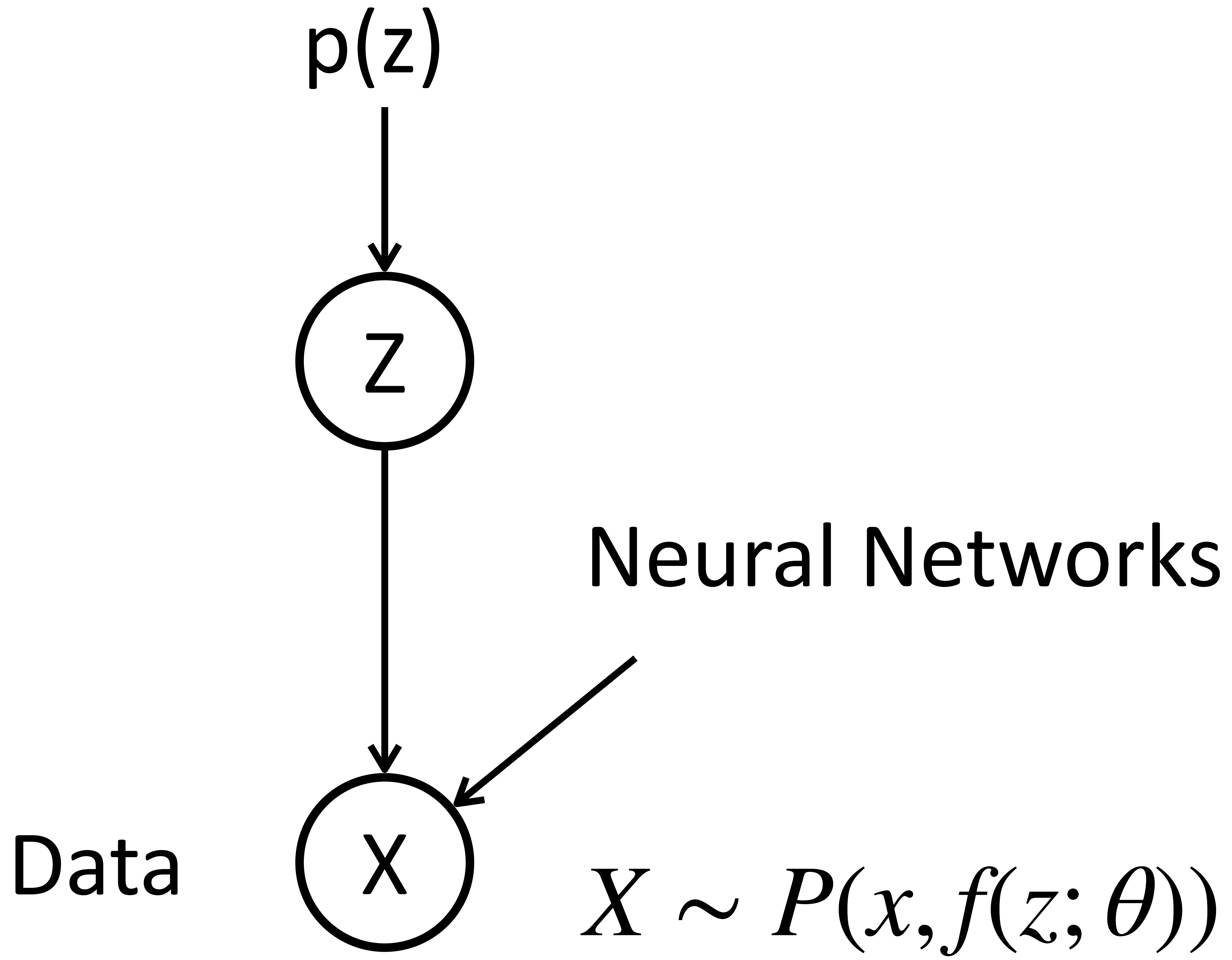
f is a neural network taking Z as input

Training



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

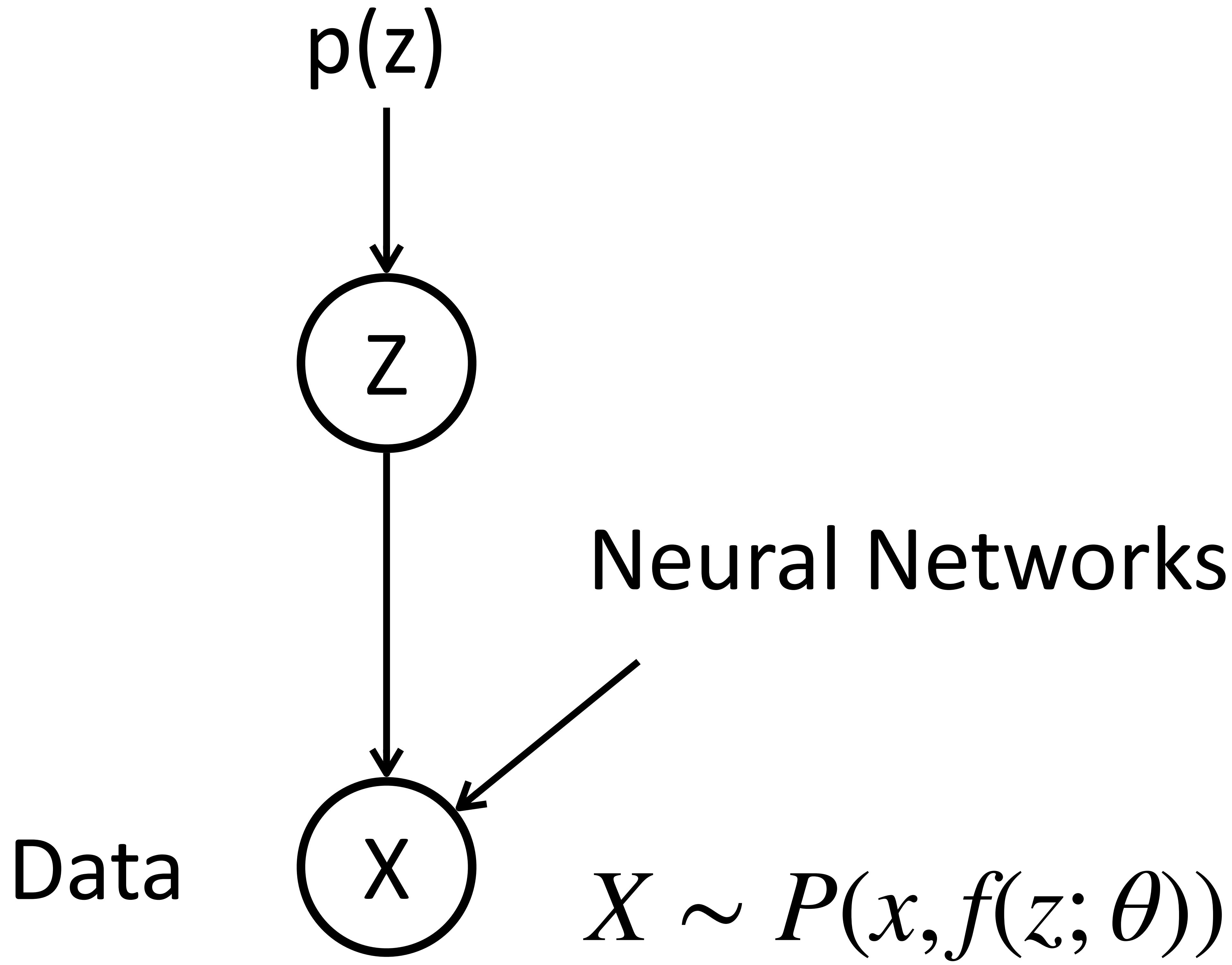
Training



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

How to train the model? Can we do MLE?

Training

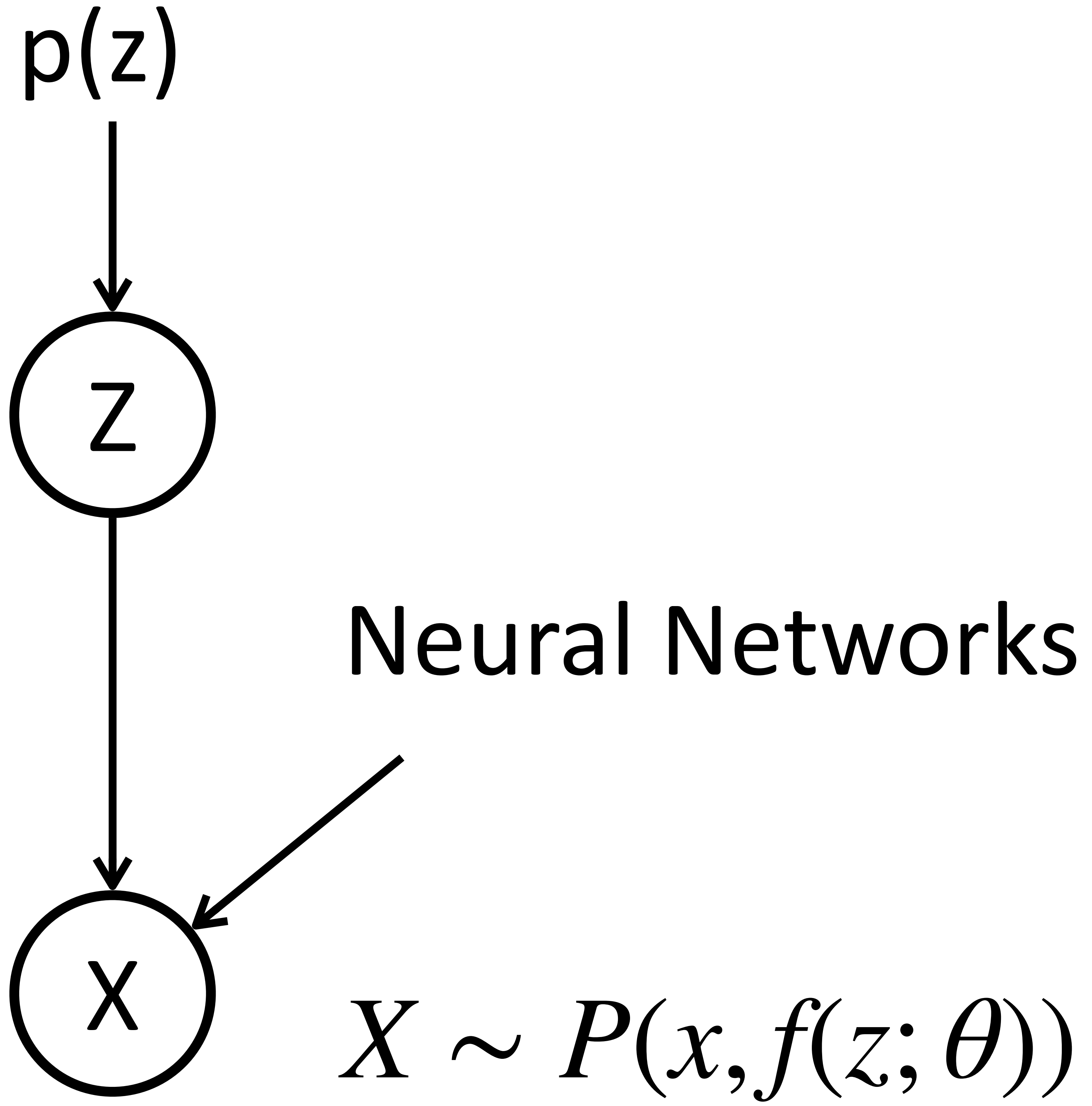


$$X \sim P(x, f(z; \theta))$$

How to train the model? Can we do MLE?

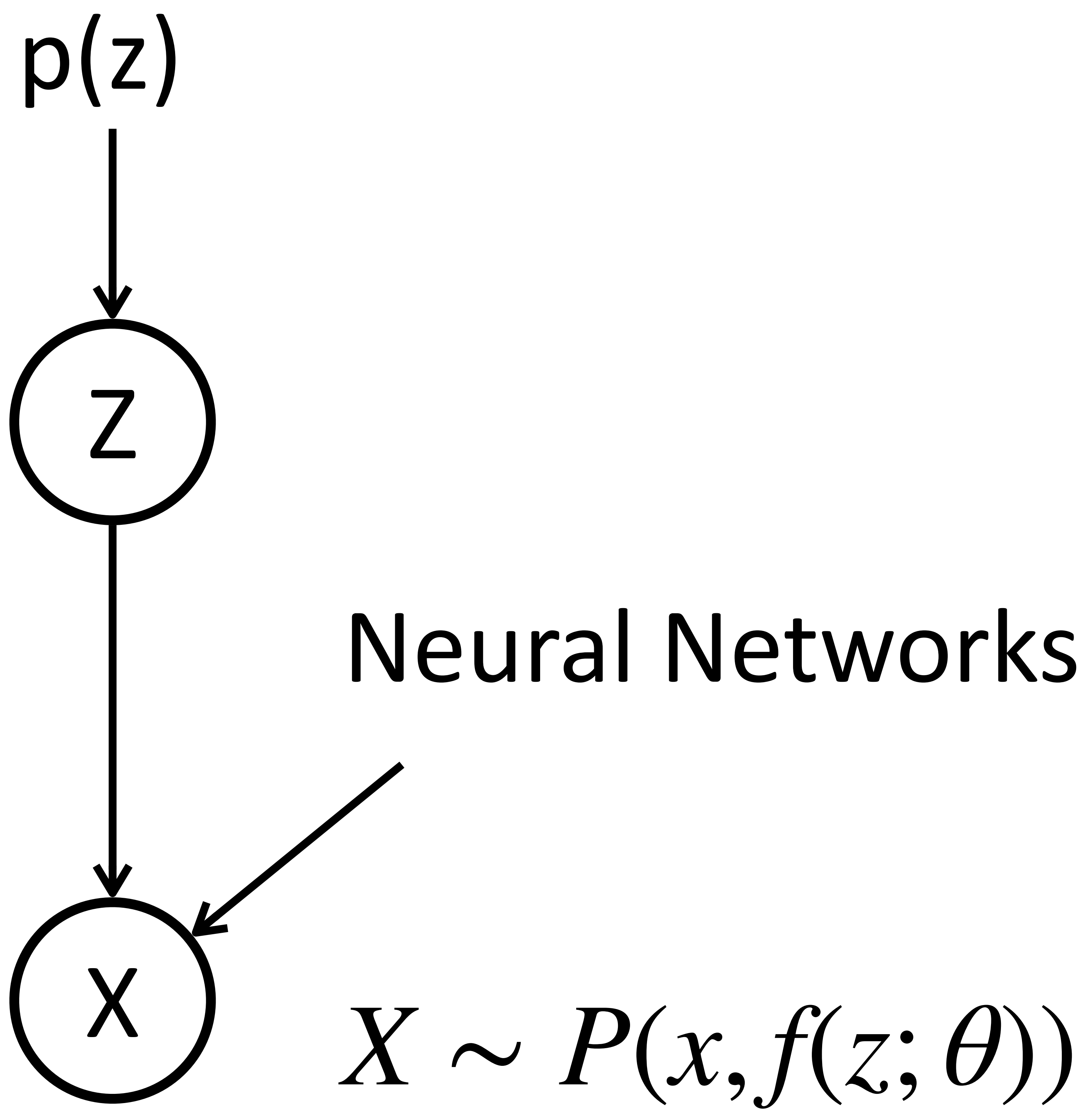
Intractable P(X), EM algorithm?

Let's try EM



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

Let's try EM

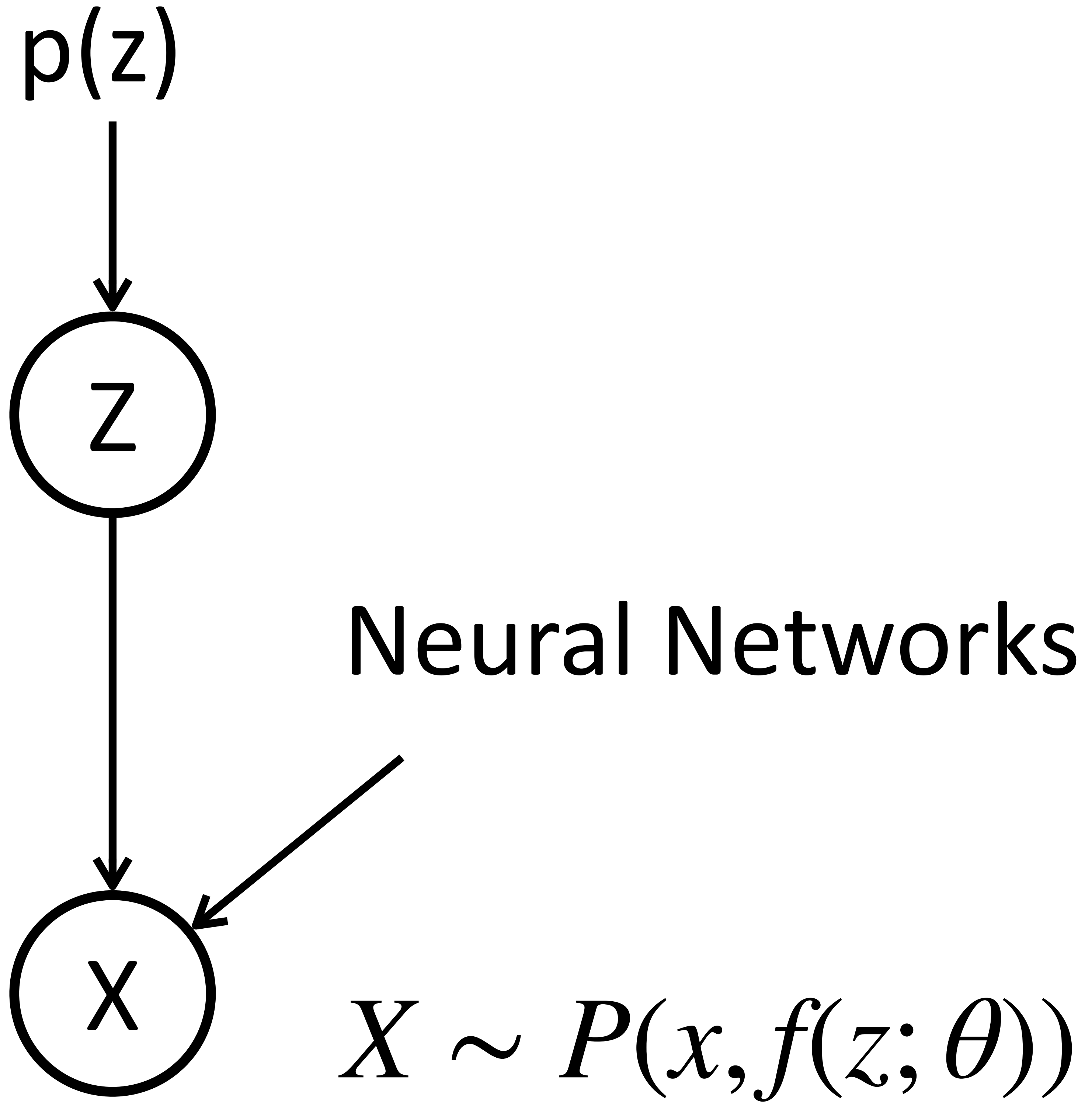


$$X \sim P(x, f(z;\theta))$$

E-Step: compute P(z|x)

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

Let's try EM



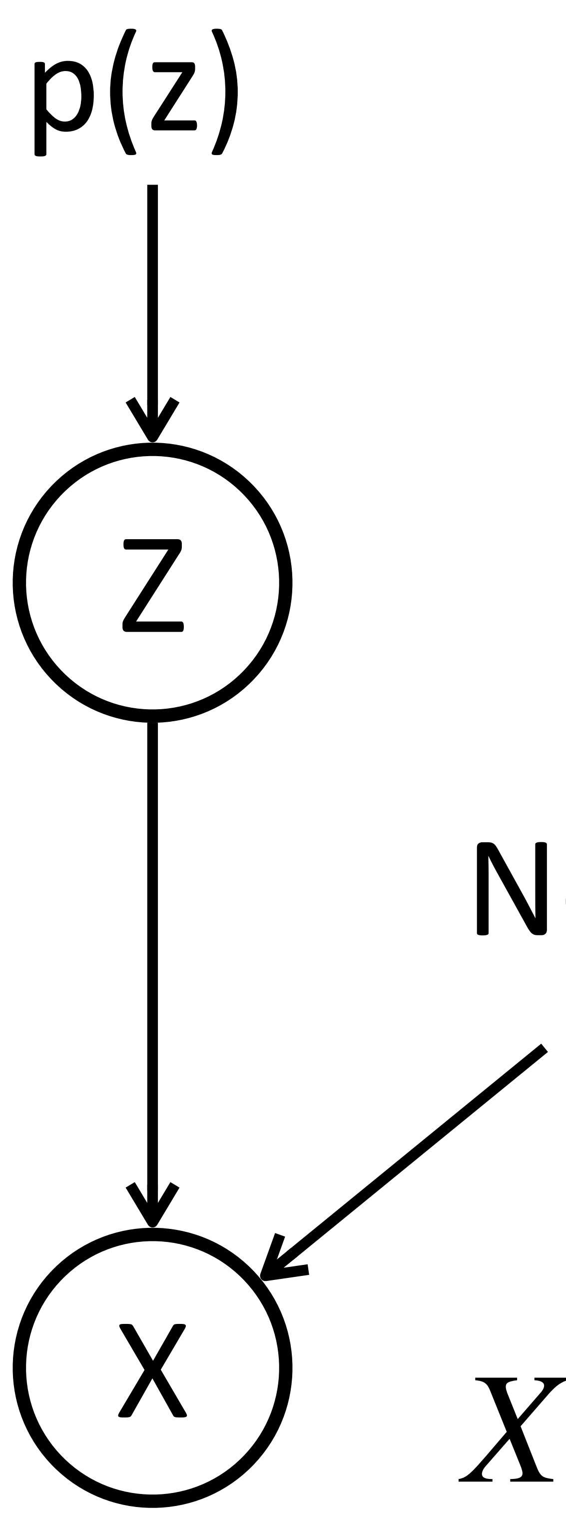
$$X \sim P(x, f(z;\theta))$$

E-Step: compute P(z|x)

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

This is ok?

Let's try EM



Neural Networks

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

E-Step: compute P(z|x)

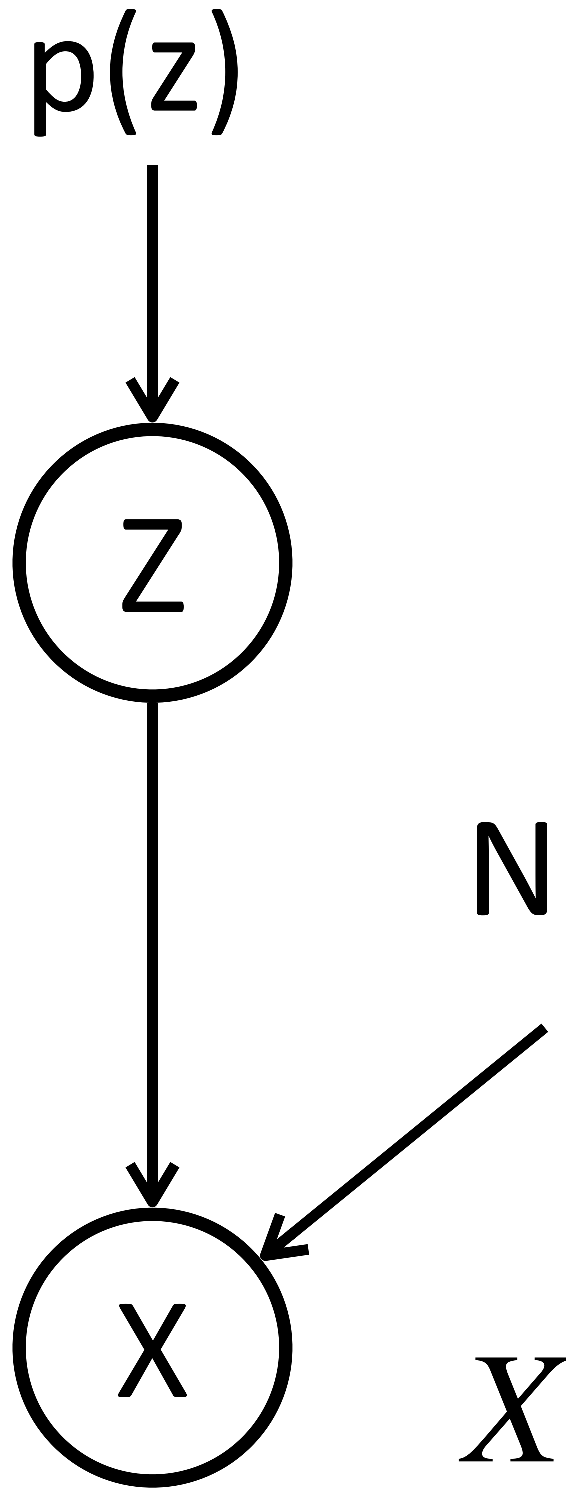
$$Q(z) = P(z|x) \propto P(z)P(x|z)$$

This is ok?

M-Step: the ELBO objective

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

Let's try EM



Neural Networks

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

E-Step: compute P(z|x)

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

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 $P(z|x)$

Approximate Posterior

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

$q(z\,|\,x;\phi)$ to approximate $p(z\,|\,x;\theta)$

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?

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

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

Recap: ELBO

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

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

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

Recap: ELBO

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

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

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

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

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

Recap: ELBO

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

What is $\mathrm{argmax}_{Q(z)}$ 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:

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

Variational Inference

Training VAEs

E-Step:

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

M-Step:

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

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

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:

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

M-Step:

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

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:

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

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

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