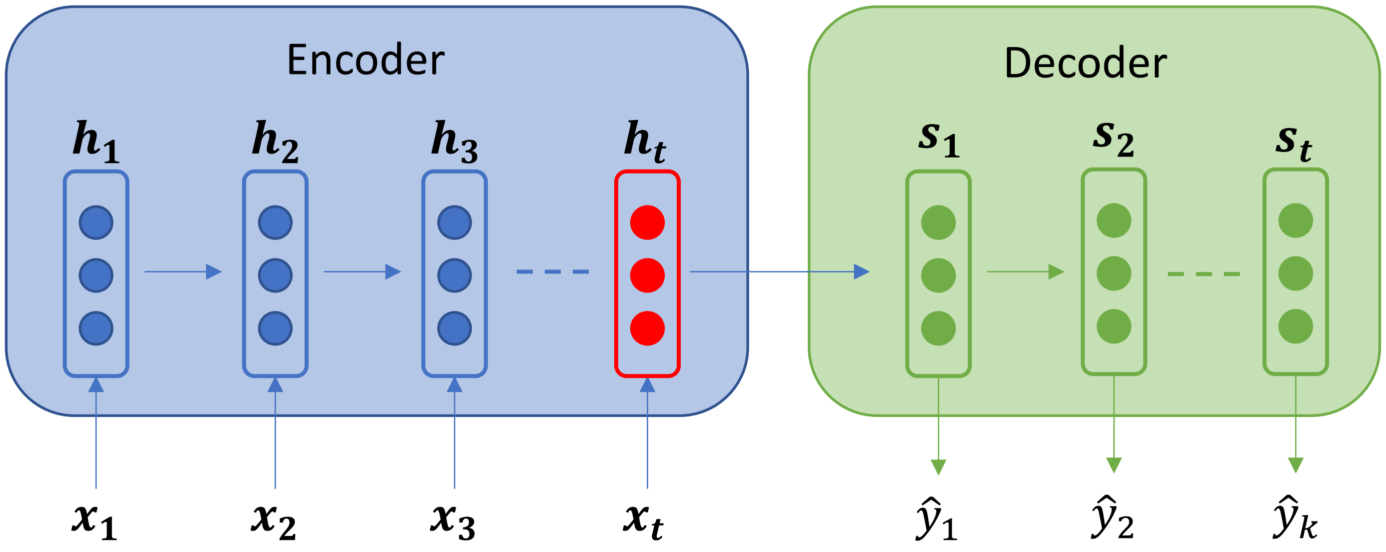
EE7207 Week 9

Attention Mechanisms and Transformers

Bottleneck of encoder-decoder network

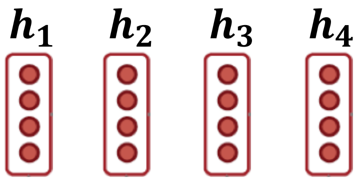
- Encoding of input sequence: a fixed length vector $\boldsymbol{h_t}$
- Need to capture all necessary information of input sequence
- Information bottleneck, especially when input sequence is long



Attention: thinking process

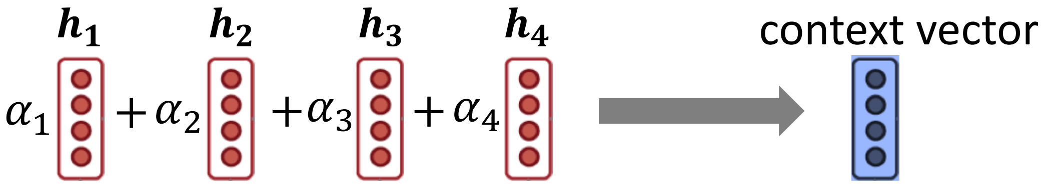
How to solve the bottleneck problem?

Instead of only using only $\boldsymbol{h_4}$, let's use all encoder hidden states!



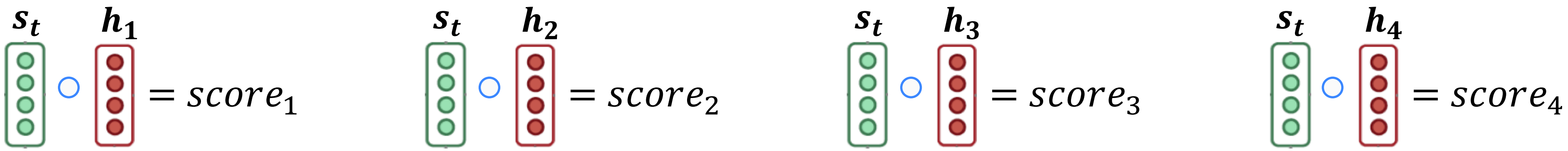
How do we deal with variable length input sequence?

Let's do a weighted sum of all encoder hidden states!



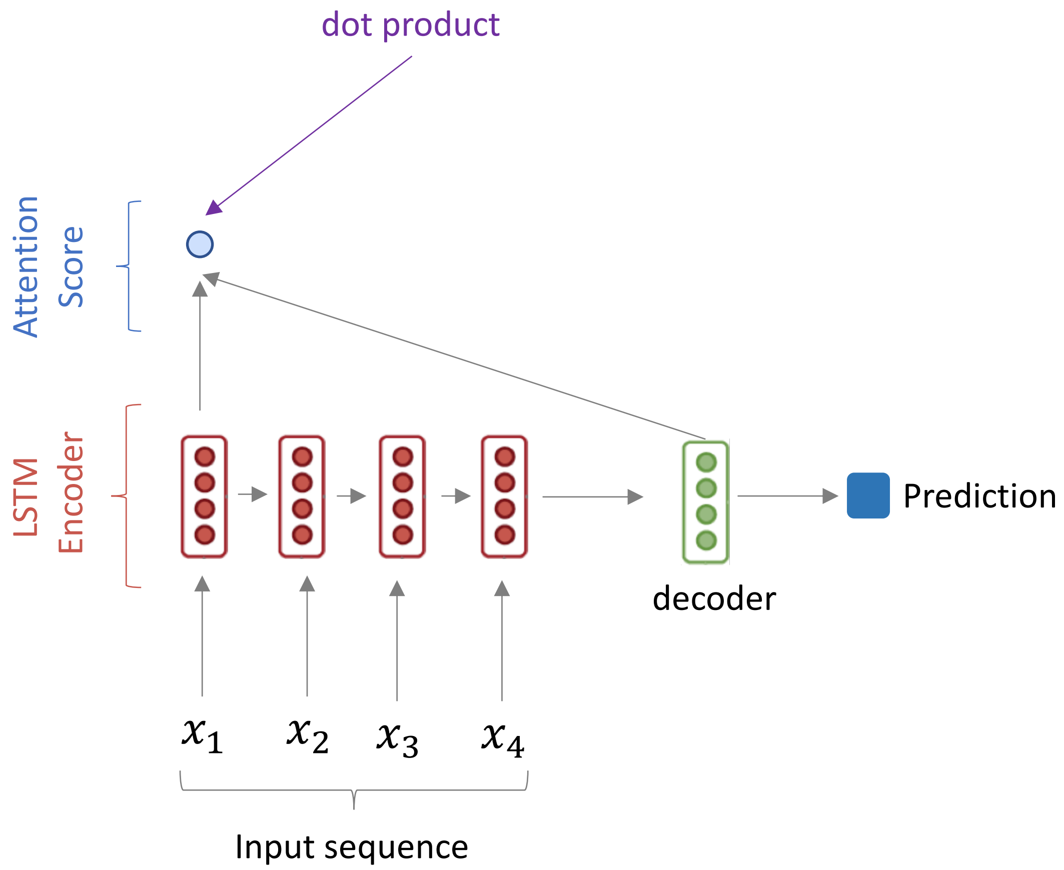
How do we get the weights α_i?

Step 1: dot product

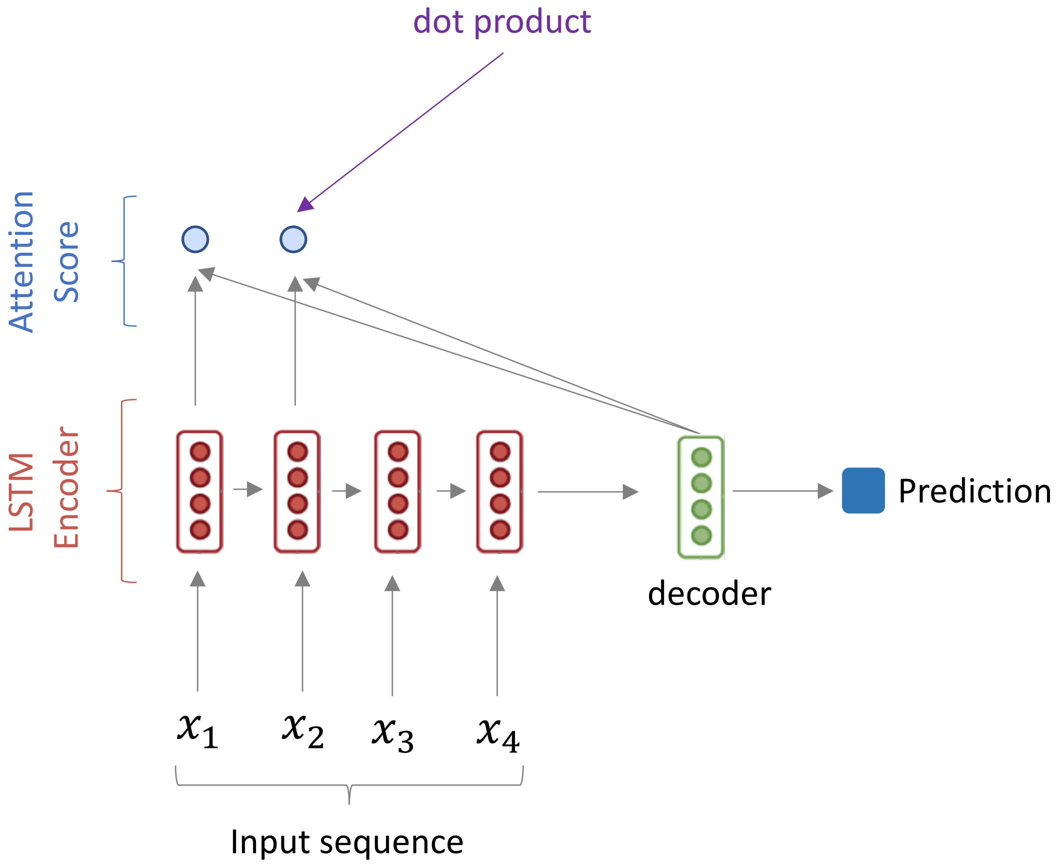


Step 2: softmax $\alpha_i = \frac{\exp(score_i)}{\sum_{j=1}^{4} \exp(score_j)}$

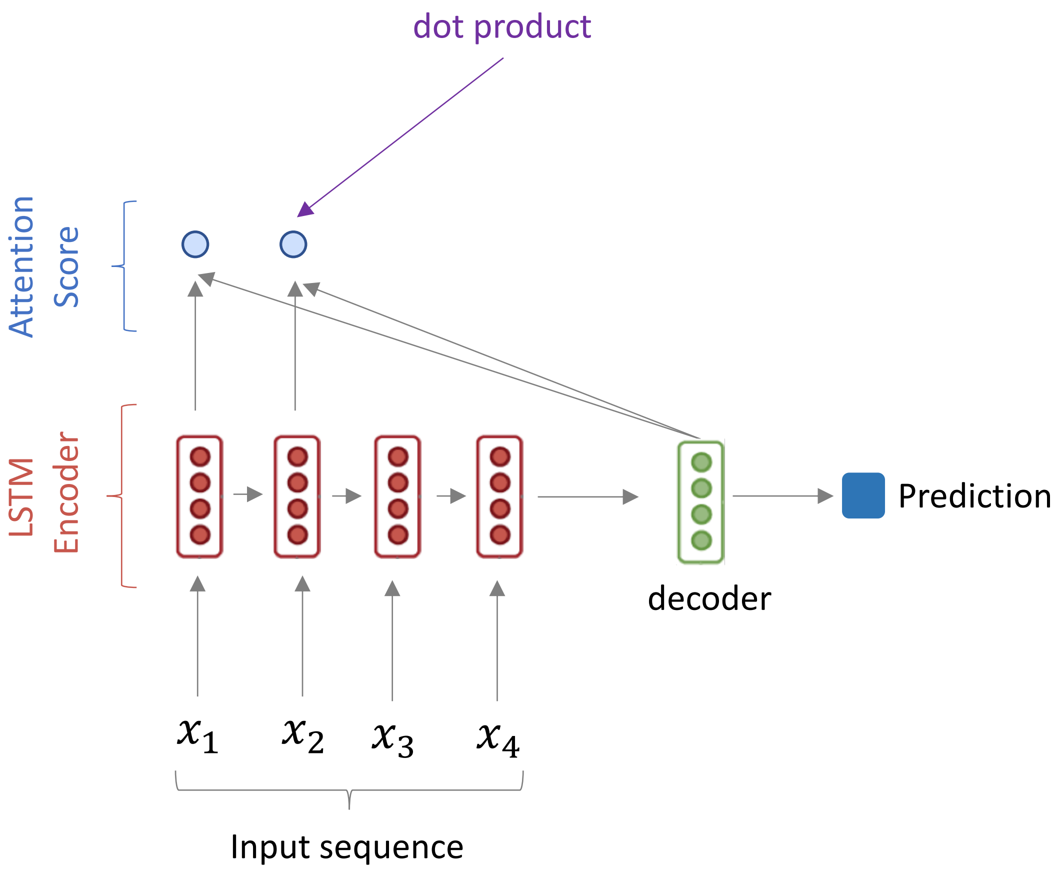
LSTM with attention



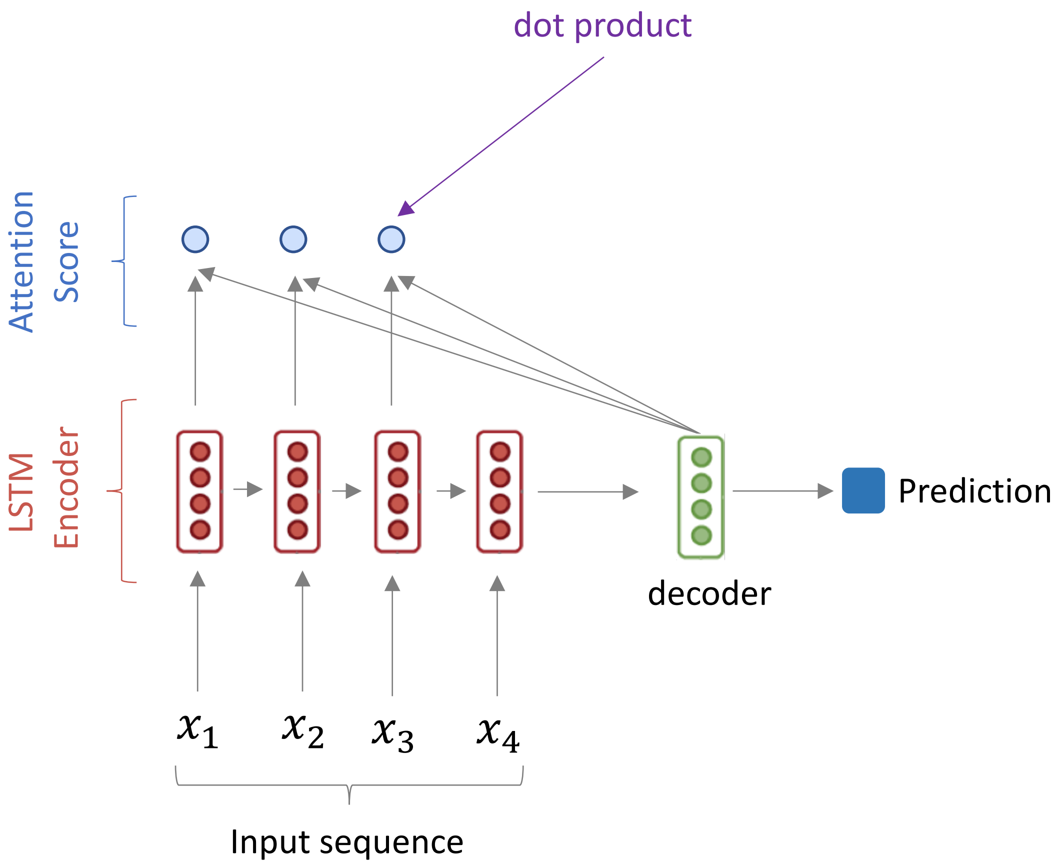
LSTM with attention



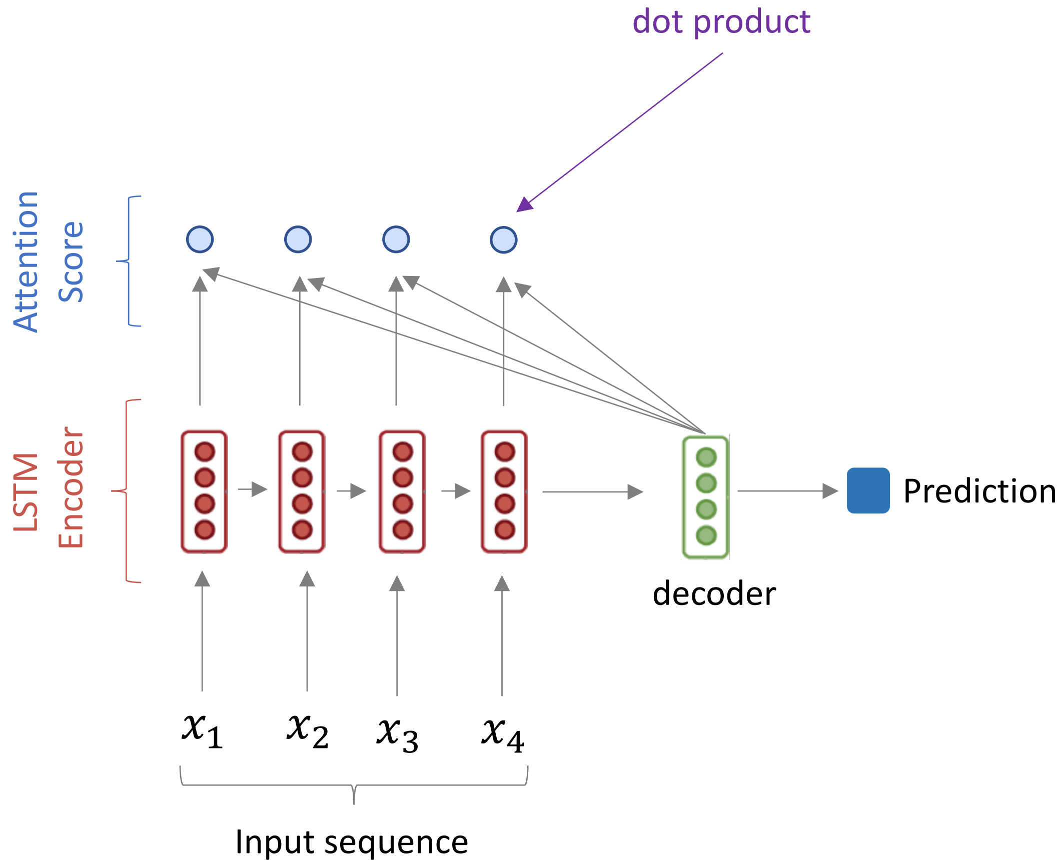
LSTM with attention



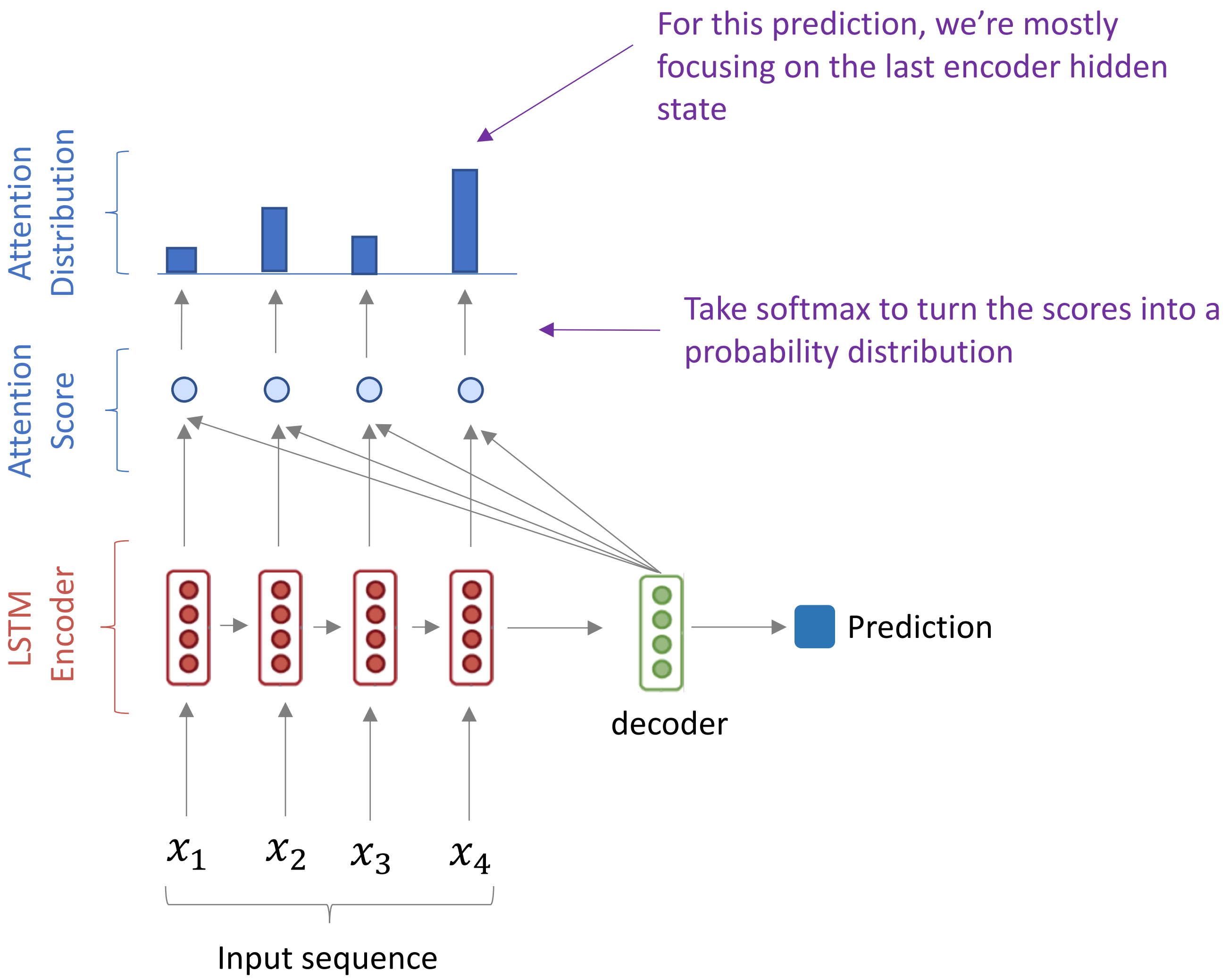
LSTM with attention



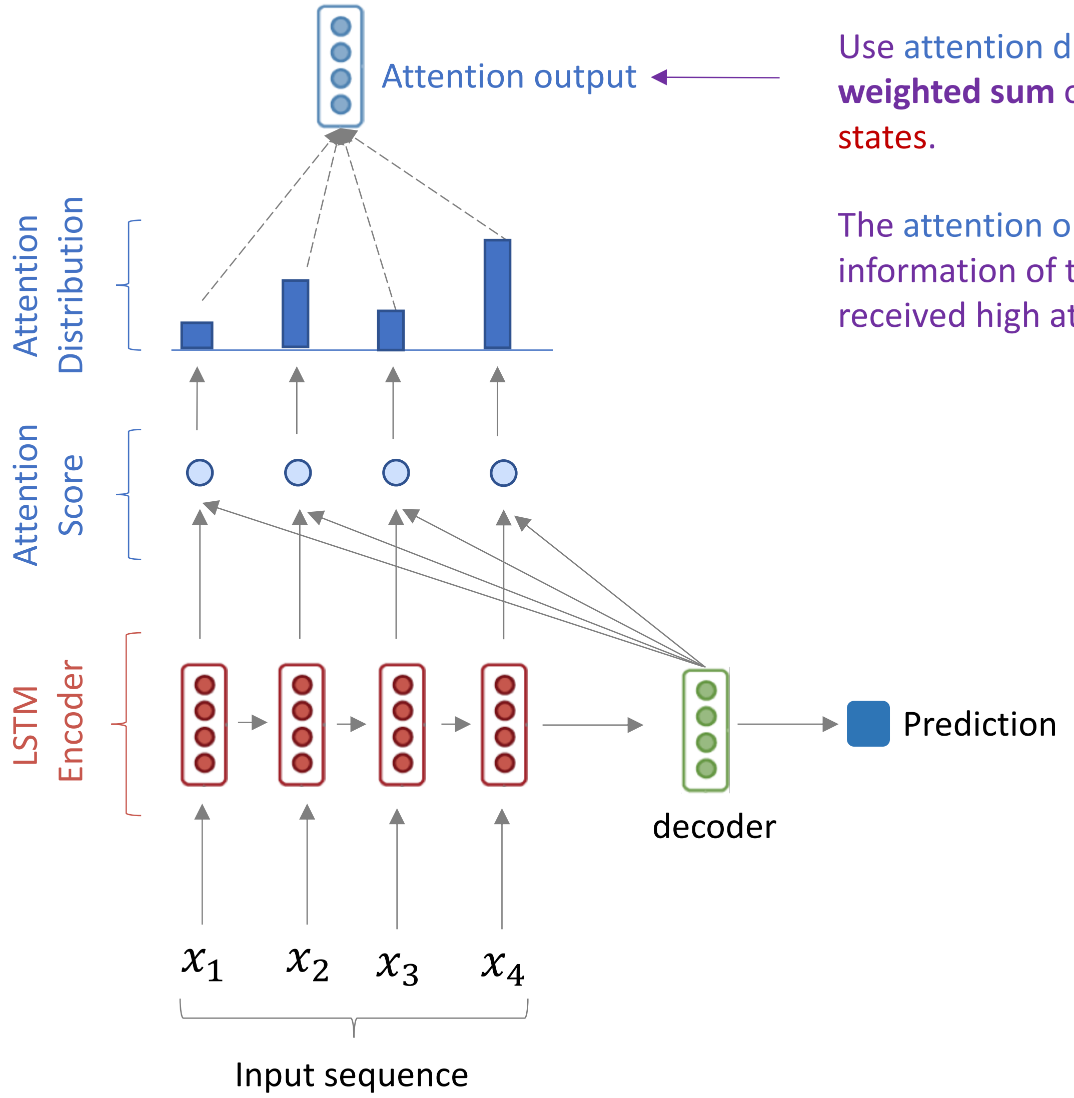
LSTM with attention



LSTM with attention



LSTM with attention



Use attention distribution to taka a **weighted sum** of the encoder hidden states.

The attention output mostly contains information of the hidden states that received high attention.

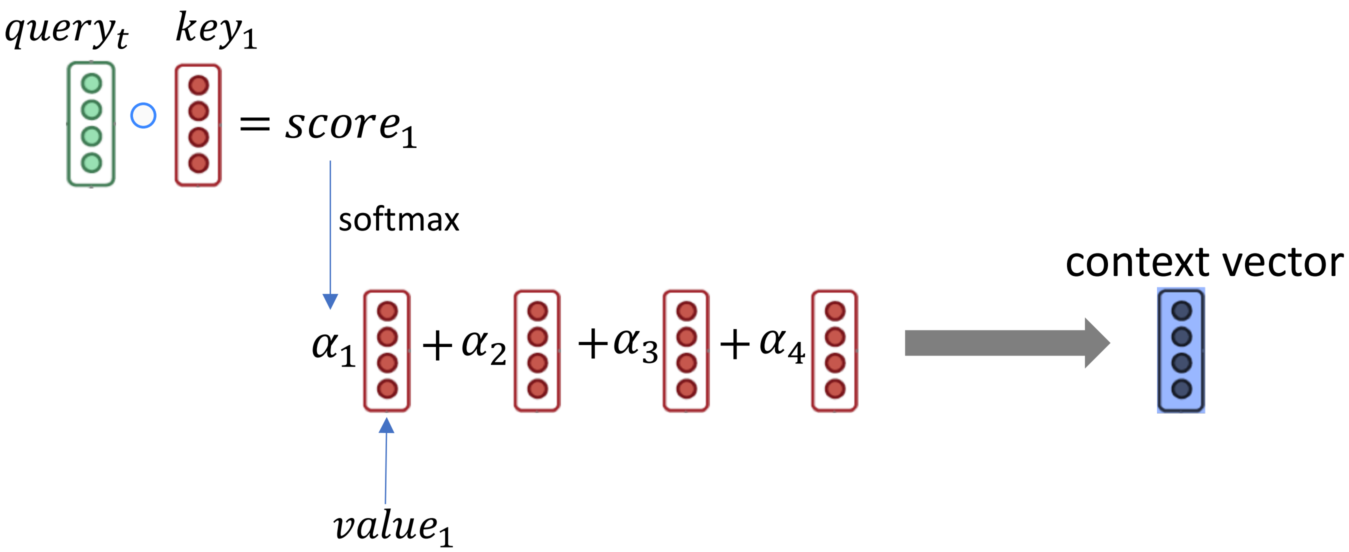
Another way to compute attention score: key-query-value attention

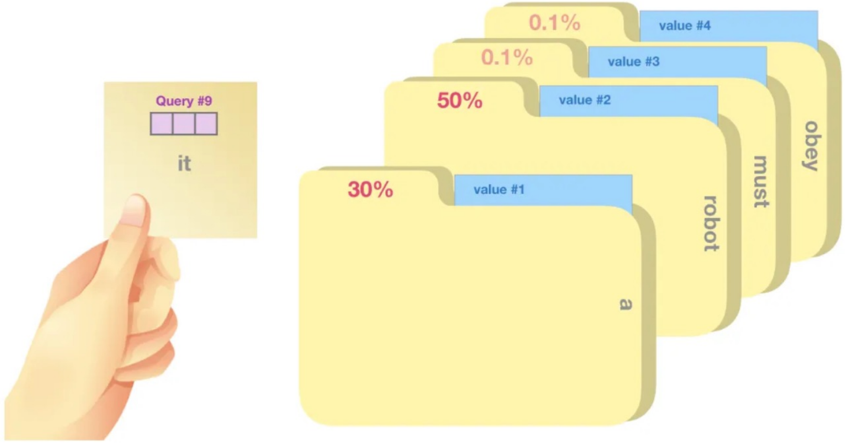
Inspired by information retrieval

Limitation of dot-product attention:

- What if the dimensions of decoder hidden states and encoder hidden states differ
- Question for later: Can dot-product attention support multi-head attentions?

Key-Query-Value attention:





Transformers

Attention Is All You Need

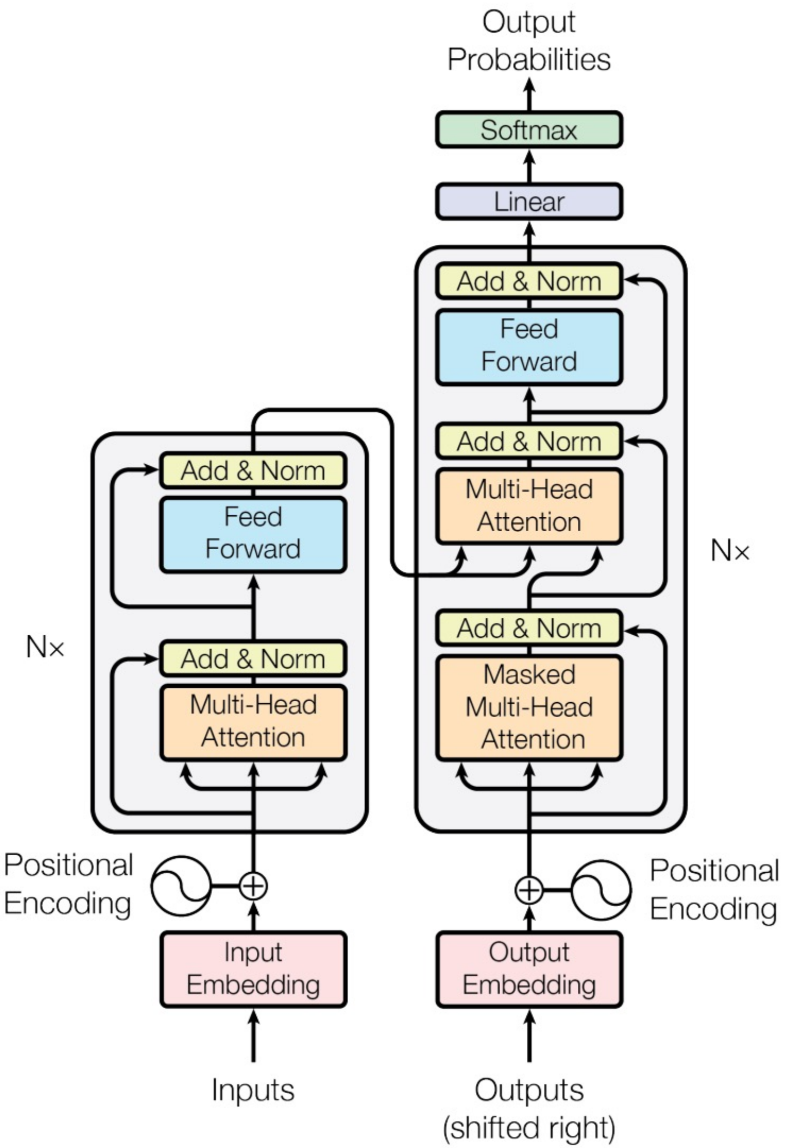
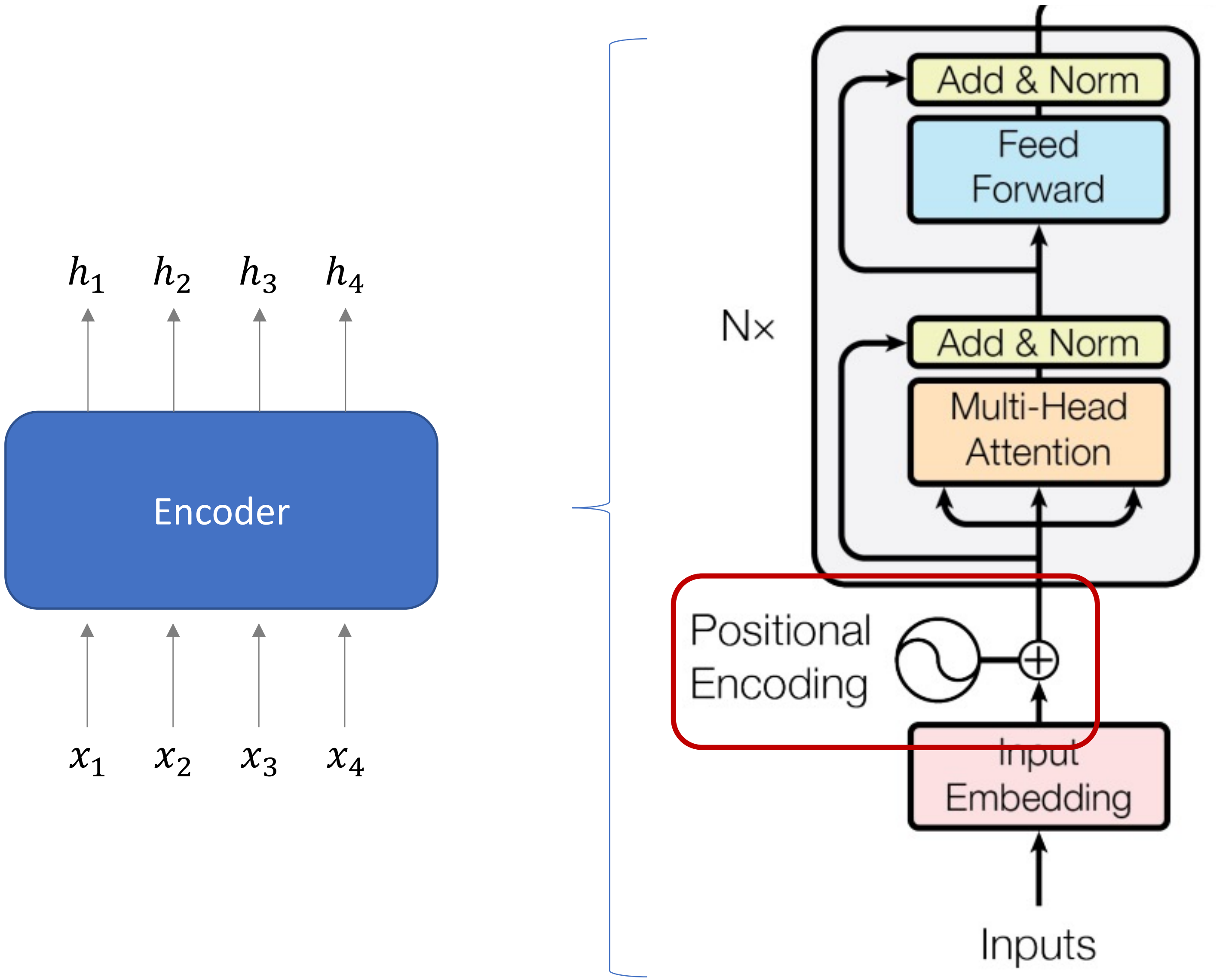


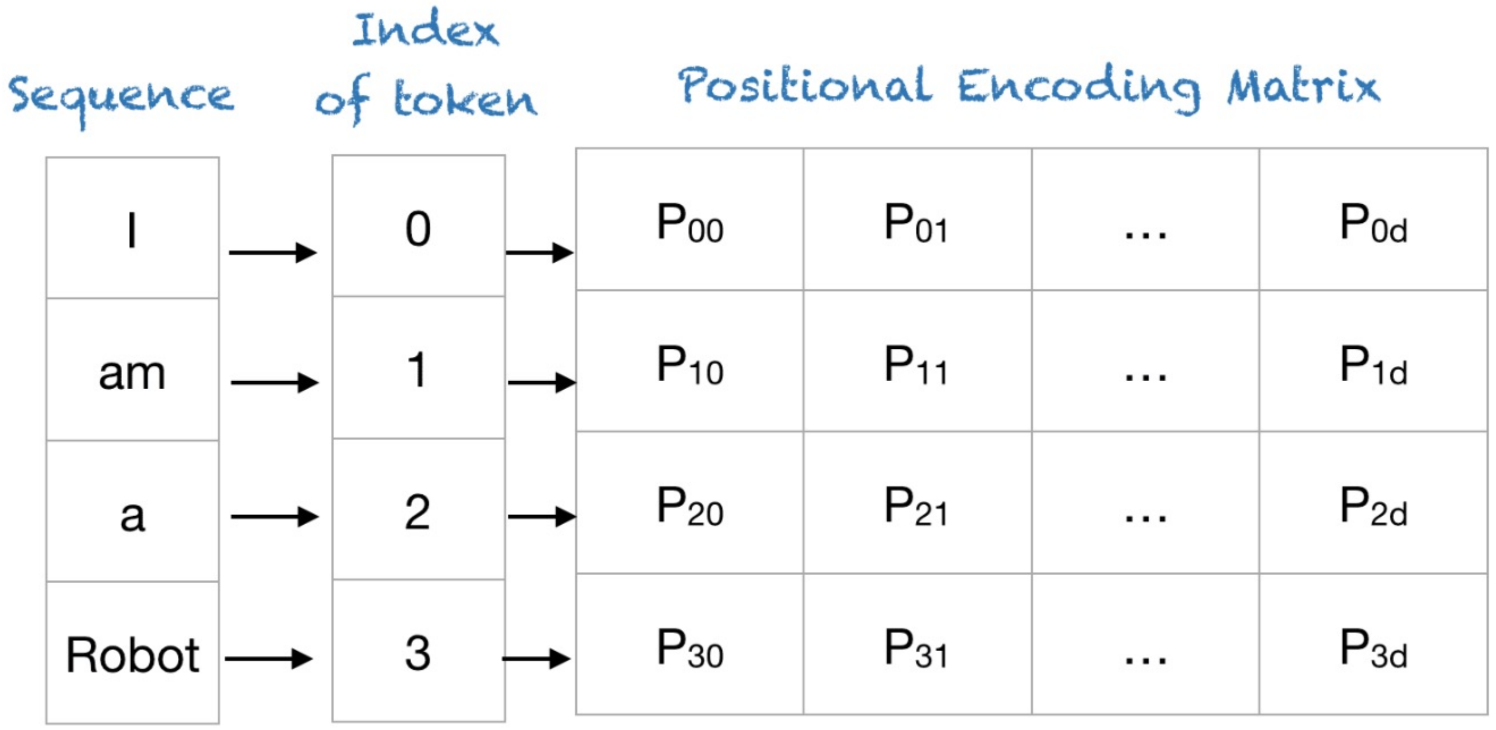
Figure 1: The Transformer - model architecture.

https://doi.org/10.48550/arXiv.1706.03762

Encoder


h_1 h_2 h_3 h_4
Encoder
x_1 x_2 x_3 x_4
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

Positional Encoding



Positional Encoding Matrix for the sequence 'I am a robot'

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Positional Encoding

$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$
$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

input sequence of length

Here:

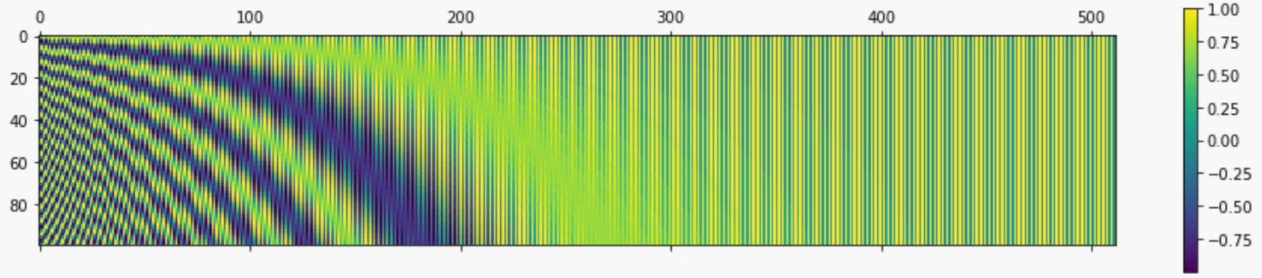
k: Position of an object in the input sequence, $0 \leq k < L/2$

d: Dimension of the output embedding space

$P(k, j)$: Position function for mapping a position k in the input sequence to index (k, j) of the positional matrix

n: User-defined scalar, set to 10,000 by the authors of Attention Is All You Need.

i: Used for mapping to column indices $0 \leq i < d/2$, with a single value of i maps to both sine and cosine functions



The positional encoding matrix for n=10,000, d=512, sequence length=100

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Positional Encoding

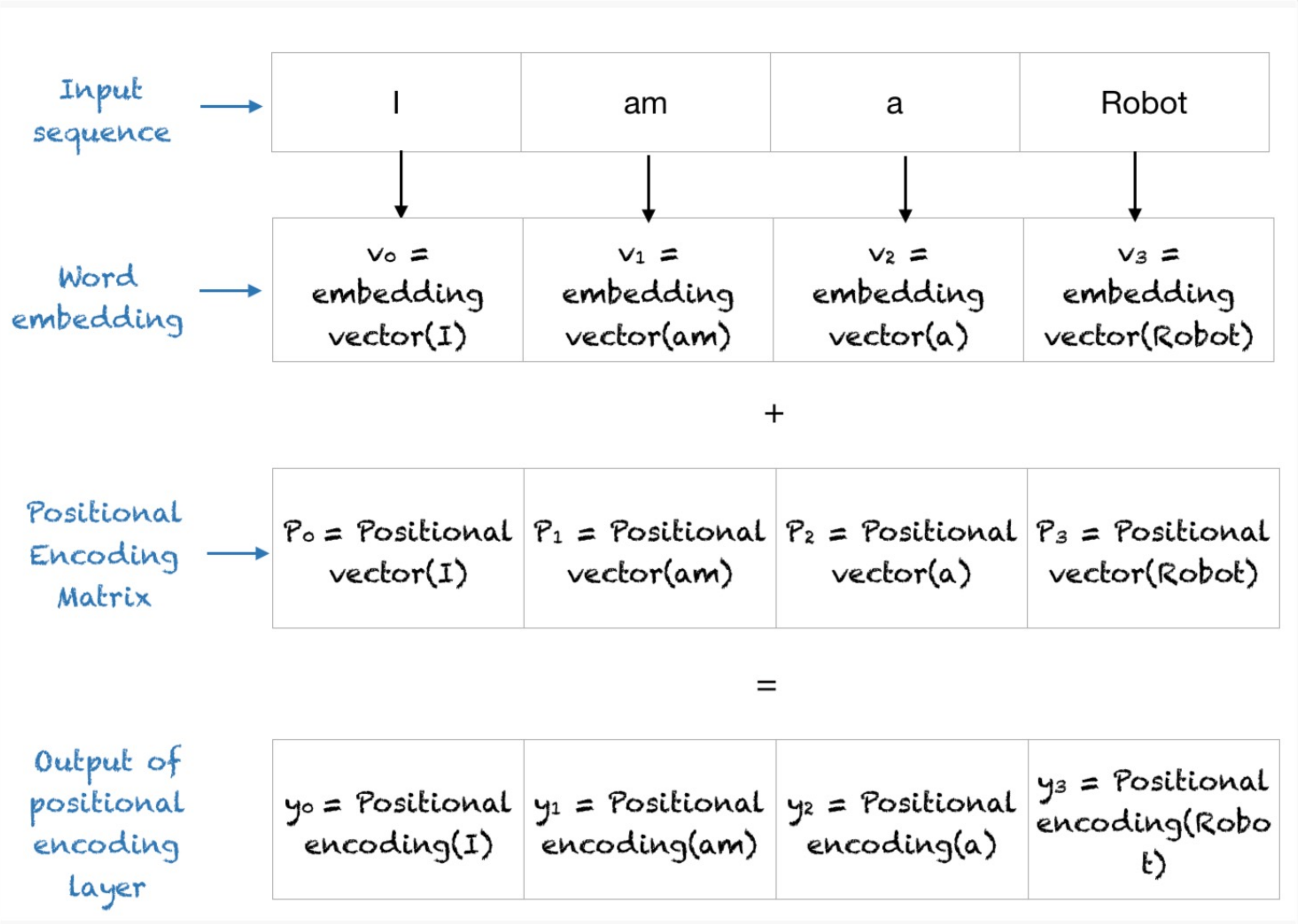
Positional Encoding Matrix with $d=4$, $n=100$

Sequence	Index of token, k	$i=0$	$i=0$	$i=1$	$i=1$
I	0	$P_{00}=\sin(0) = 0$	$P_{01}=\cos(0) = 1$	$P_{02}=\sin(0) = 0$	$P_{03}=\cos(0) = 1$
am	1	$P_{10}=\sin(1/1) = 0.84$	$P_{11}=\cos(1/1) = 0.54$	$P_{12}=\sin(1/10) = 0.10$	$P_{13}=\cos(1/10) = 1.0$
a	2	$P_{20}=\sin(2/1) = 0.91$	$P_{21}=\cos(2/1) = -0.42$	$P_{22}=\sin(2/10) = 0.20$	$P_{23}=\cos(2/10) = 0.98$
Robot	3	$P_{30}=\sin(3/1) = 0.14$	$P_{31}=\cos(3/1) = -0.99$	$P_{32}=\sin(3/10) = 0.30$	$P_{33}=\cos(3/10) = 0.96$

Positional Encoding Matrix for the sequence 'I am a robot'

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

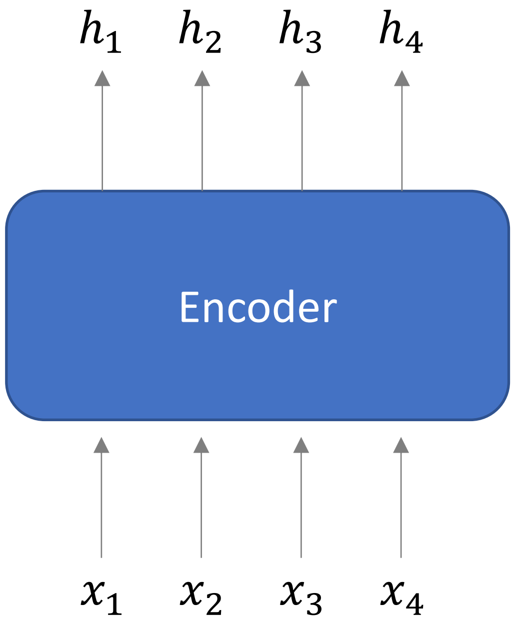
Positional Encoding



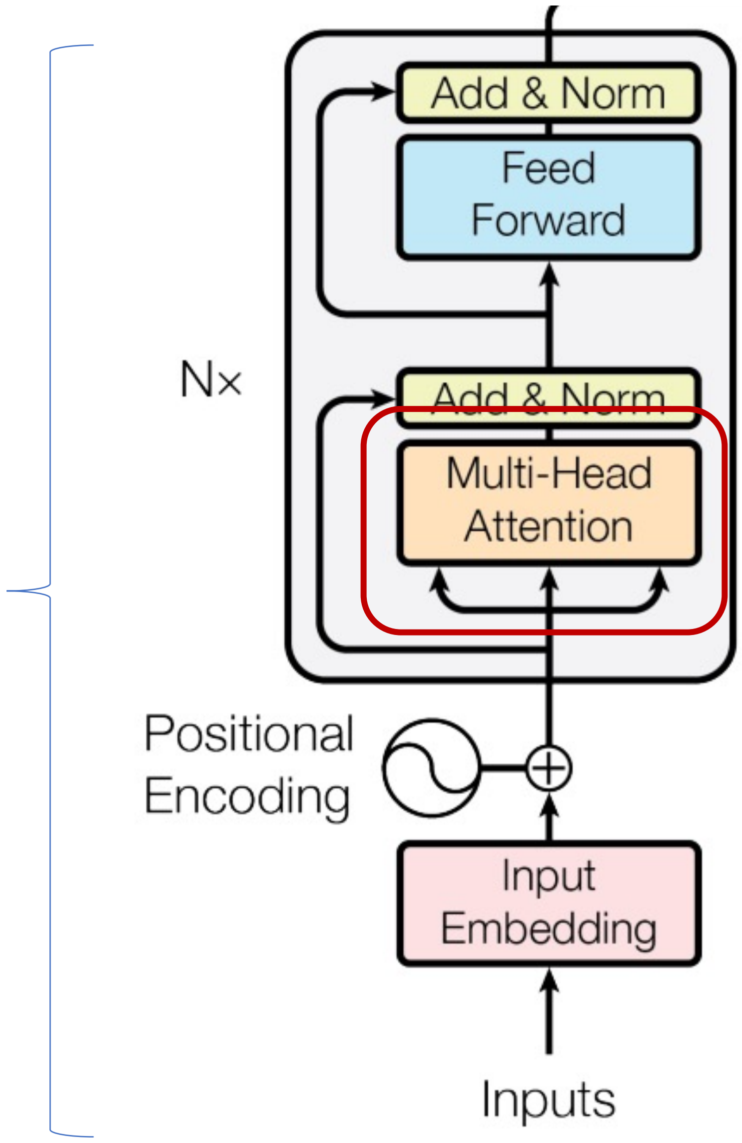
The positional encoding layer in the transformer

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Encoder

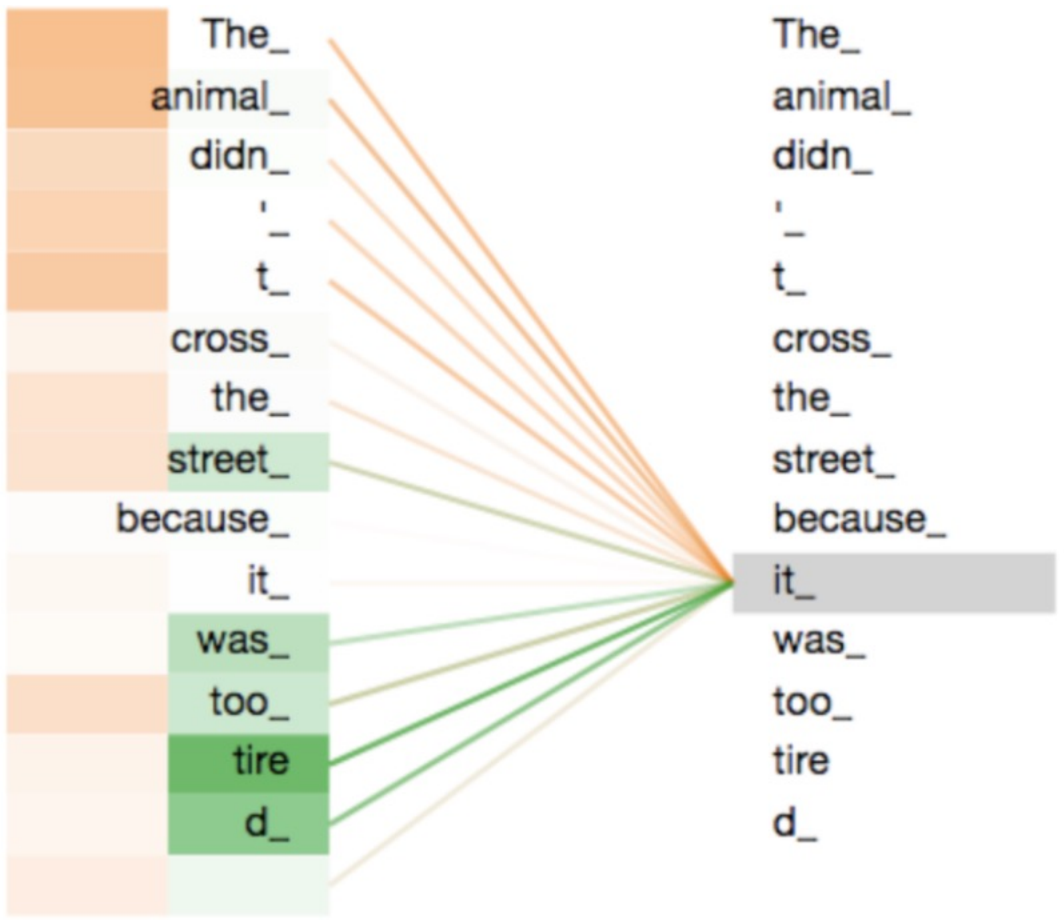


h_1 h_2 h_3 h_4
Encoder
x_1 x_2 x_3 x_4

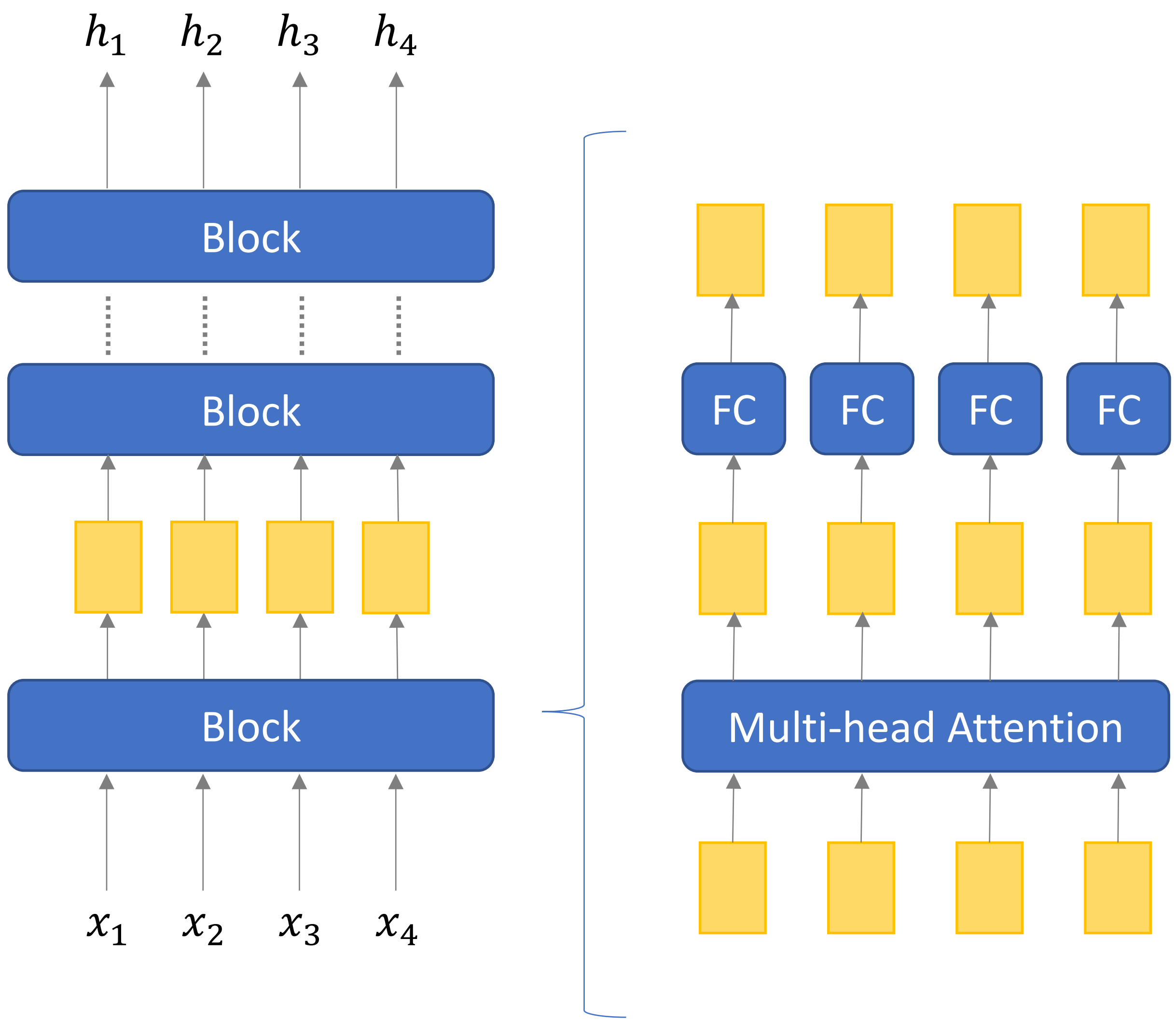


Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

Multi-head attention

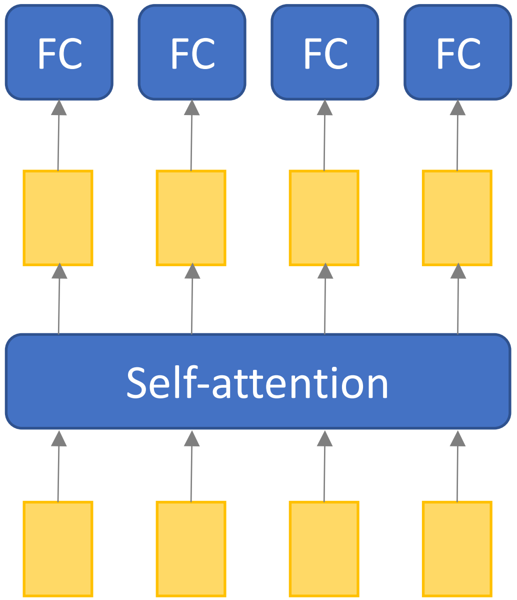


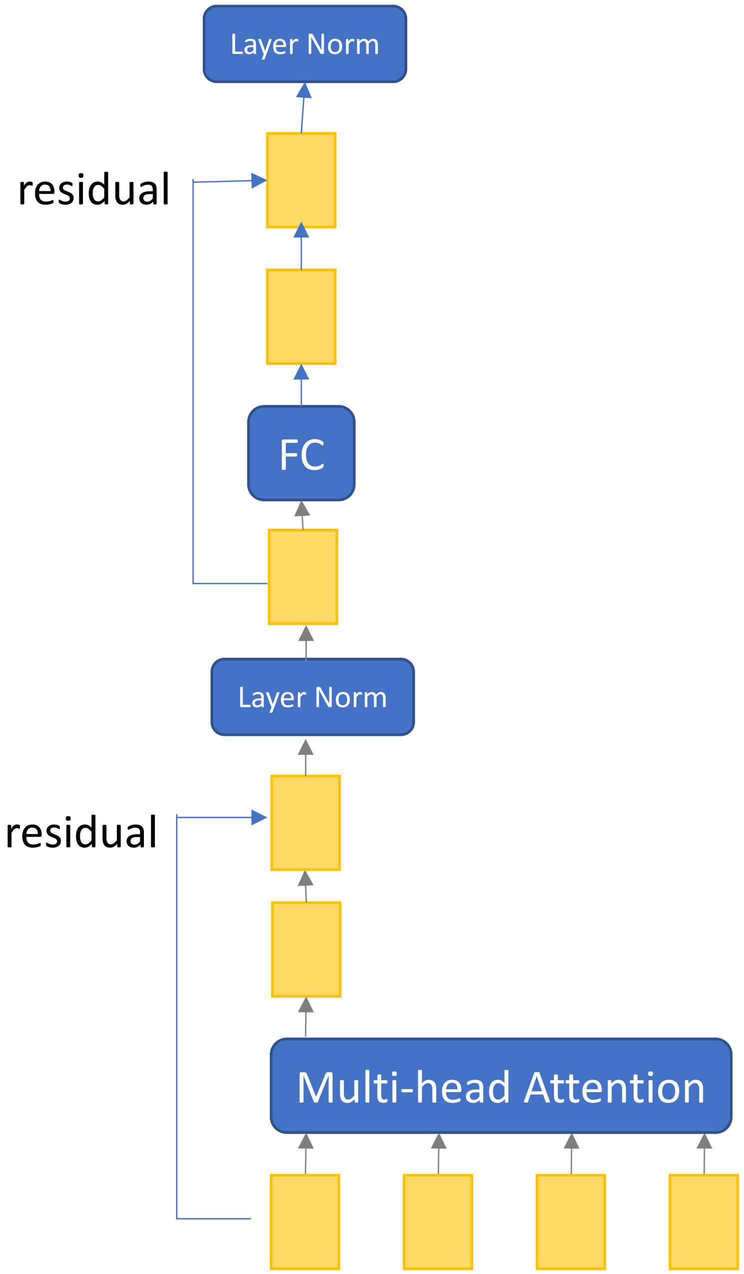
Encoder



h_1 h_2 h_3 h_4
Block
Block
Block
x_1 x_2 x_3 x_4
FC
FC
FC
FC
Multi-head Attention

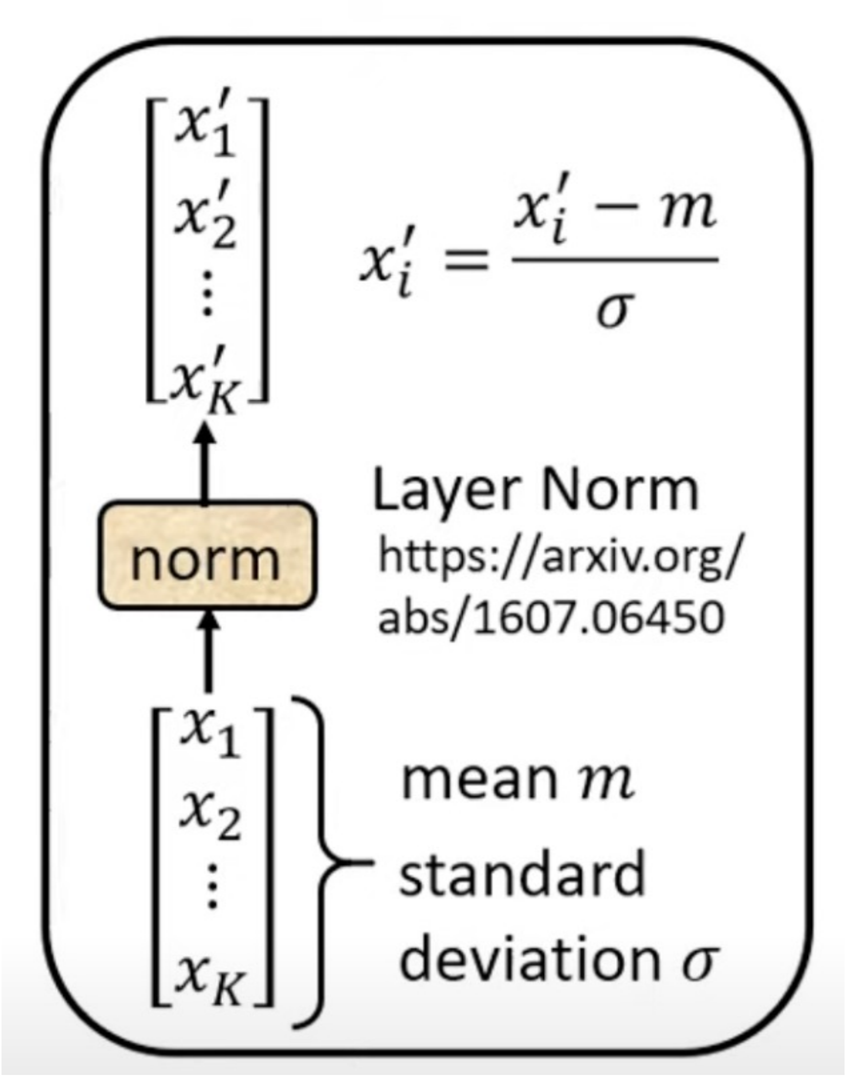
Encoder


FC
FC
FC
FC
Self-attention


Layer Norm
residual
FC
Layer Norm
residual
Multi-head Attention

Layer Norm

Name	Age	Height
Alice	19	158
Bob	21	172
Claire	22	163
David	20	166



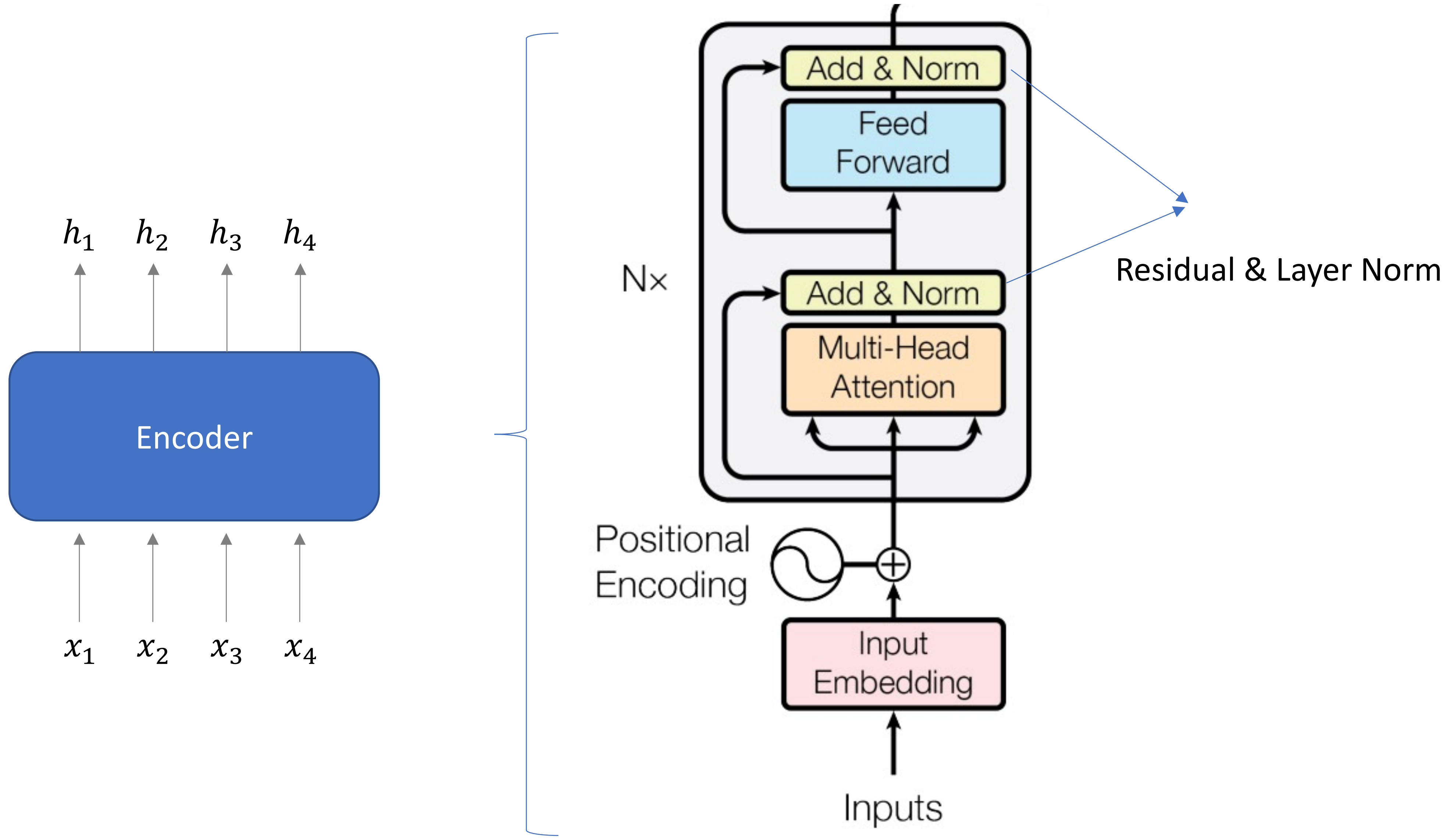
Batch Norm

- Each feature, normalize across all data points in the batch
- For example, for Age, normalize across 19, 21, 22, 20

Layer Norm

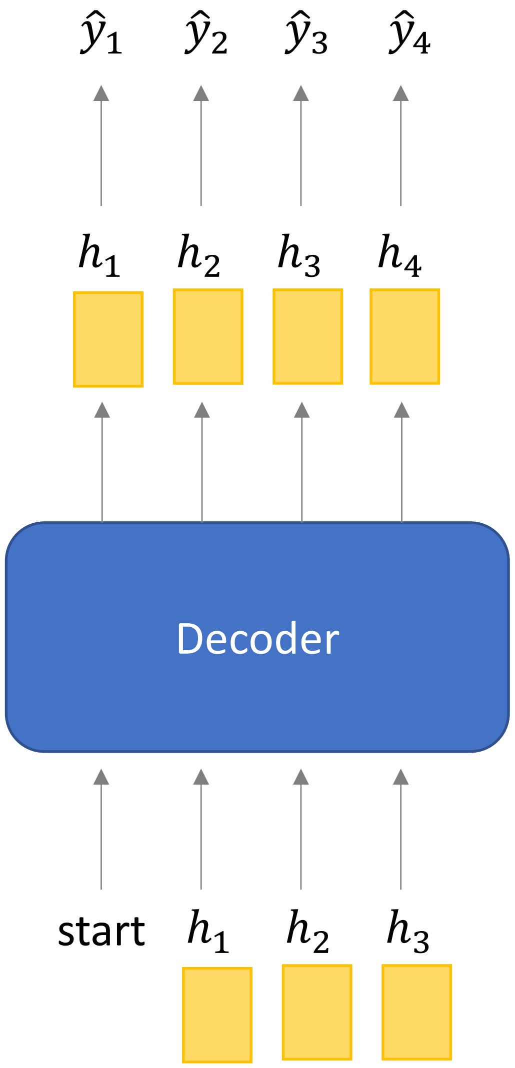
- Each data point, normalize across all features
- For example, for Alice, normalize across 19, 158

Encoder

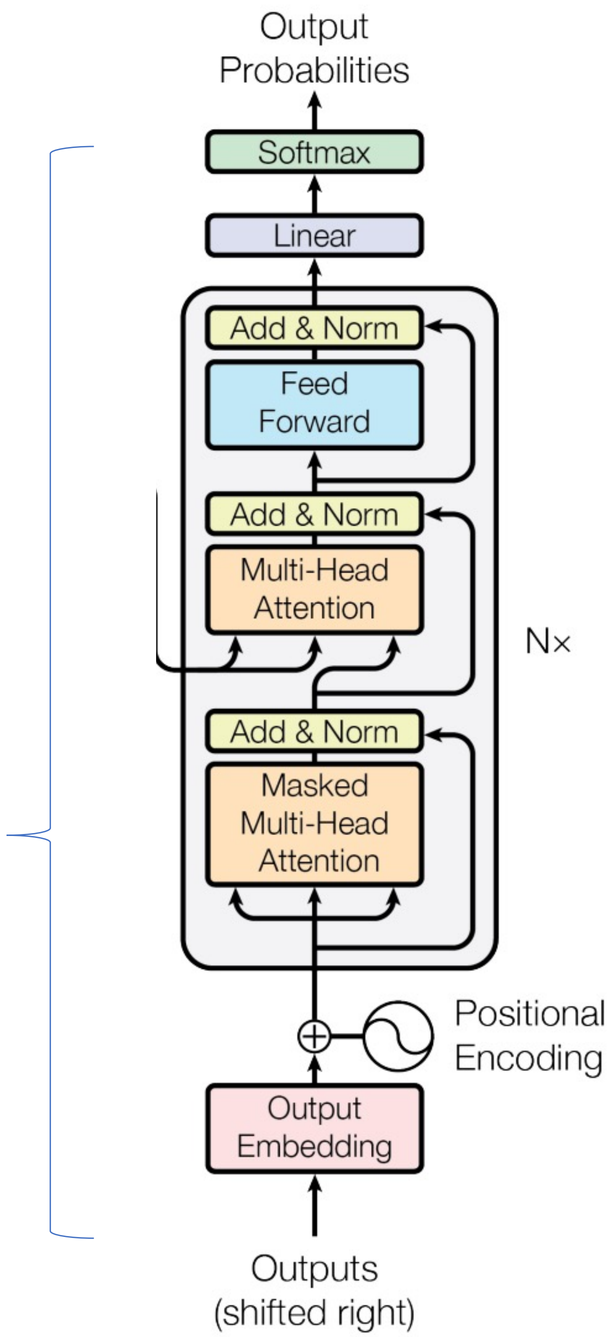


h_1 h_2 h_3 h_4
Encoder
x_1 x_2 x_3 x_4
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs
Residual & Layer Norm

Decoder

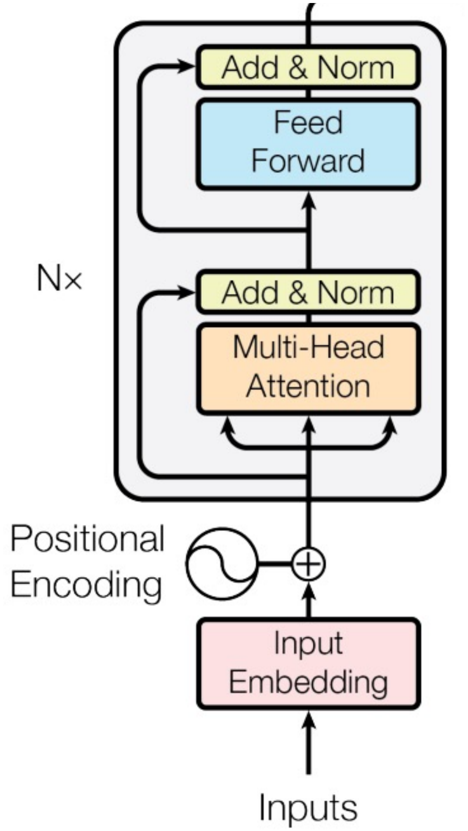


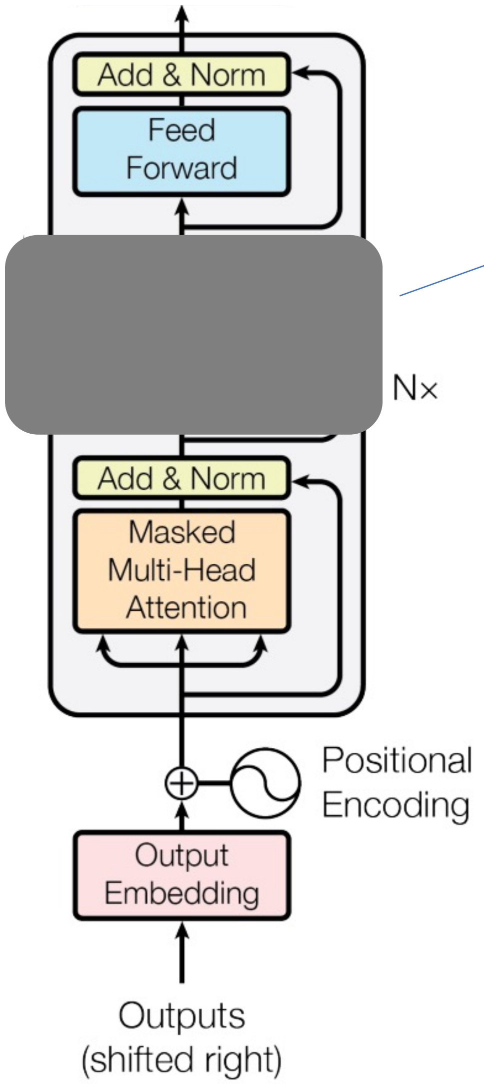
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$
h_1 h_2 h_3 h_4
Decoder
start h_1 h_2 h_3



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

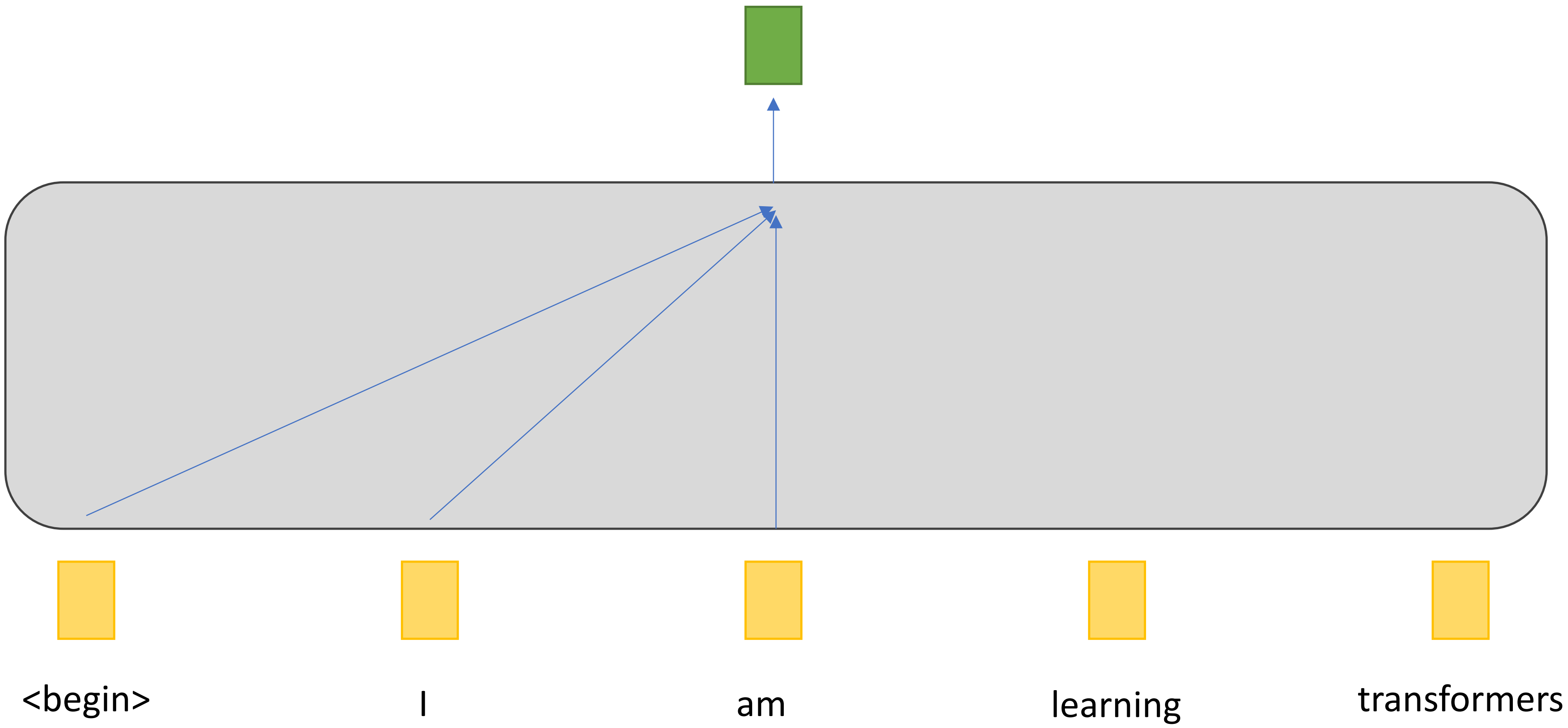
Encoder and Decoder



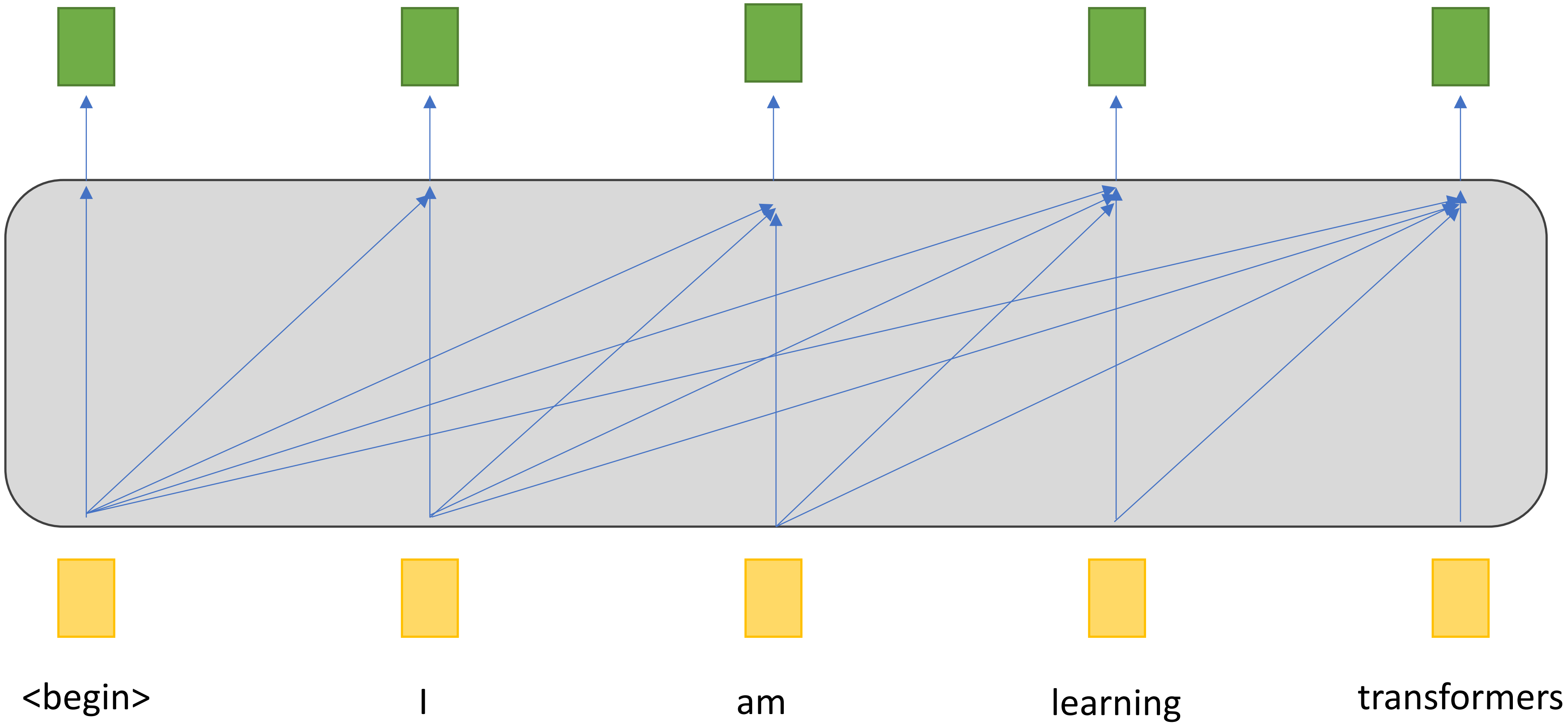


- Let's ignore this for now, will discuss later.
- The rest are very similar to encoder

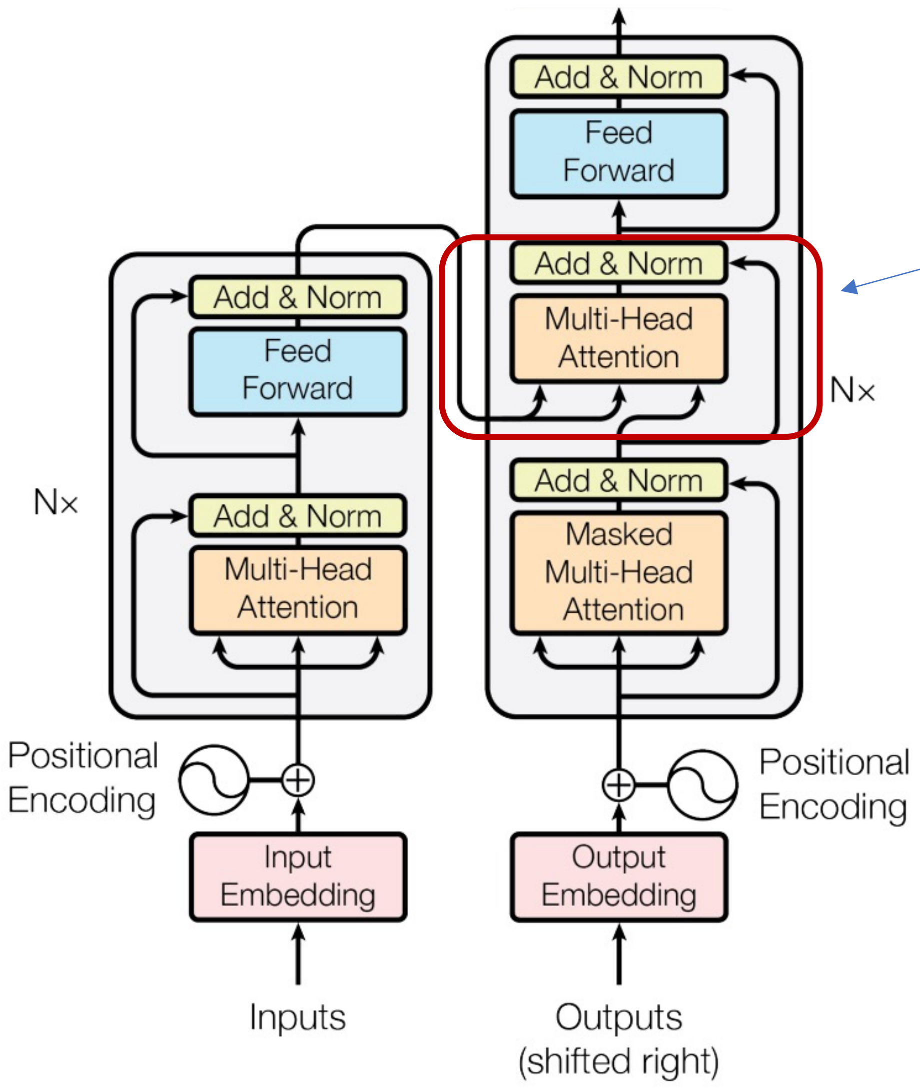
Masked Multi-head Attention



Masked Multi-head Attention

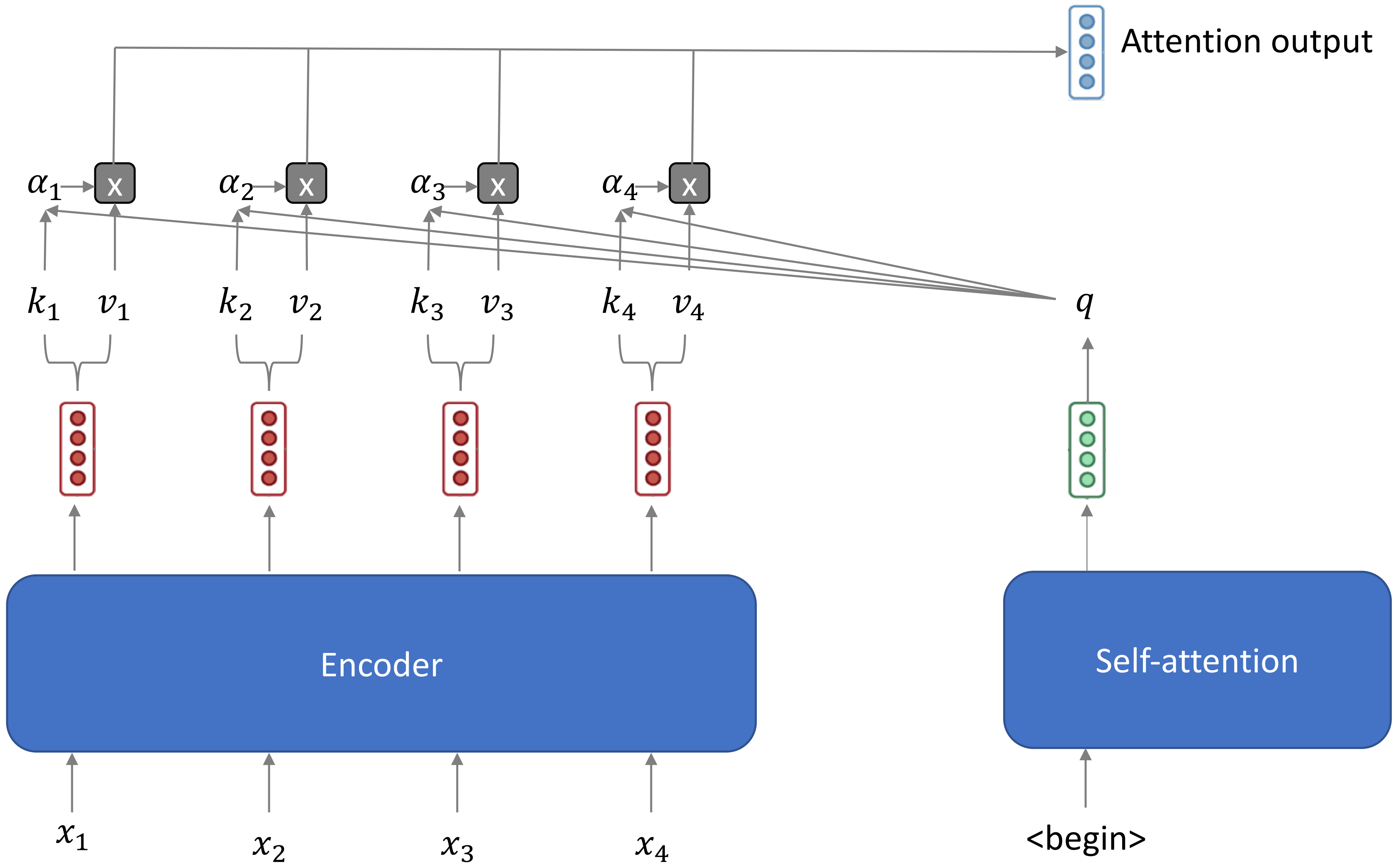


Cross Attention



Cross Attention

Cross Attention



Pigeonhole for Q&A

https://pigeonhole.at/EE7207WEEK9