Advance Deep Learning

Advance Deep Learning

Week 1: Attention

Week 2:

- Transfer Learning
- Multi-Task Learning

Week 3: Generative Adversarial Models

Week 4: Real World Applications

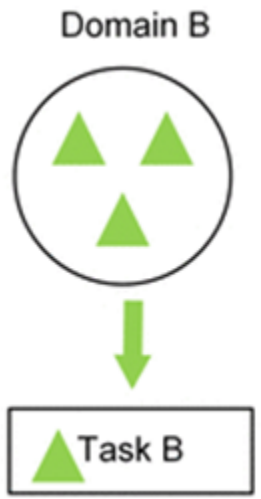
What is Attention?

What is Attention?

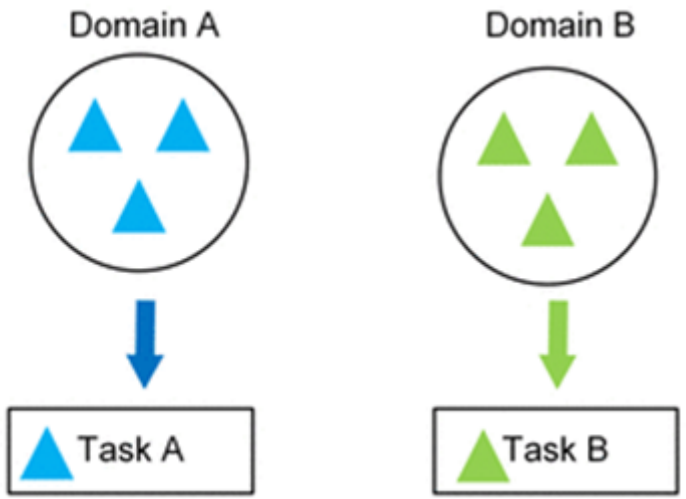
It is a nice movie.

What is Attention?

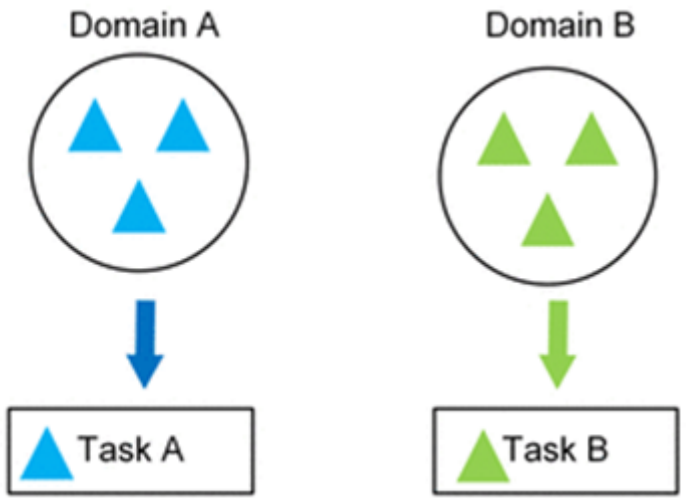
What is Transfer Learning?

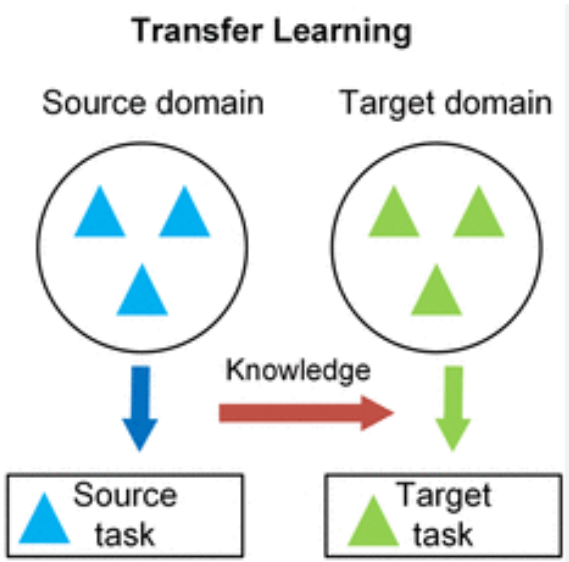


What is Transfer Learning?

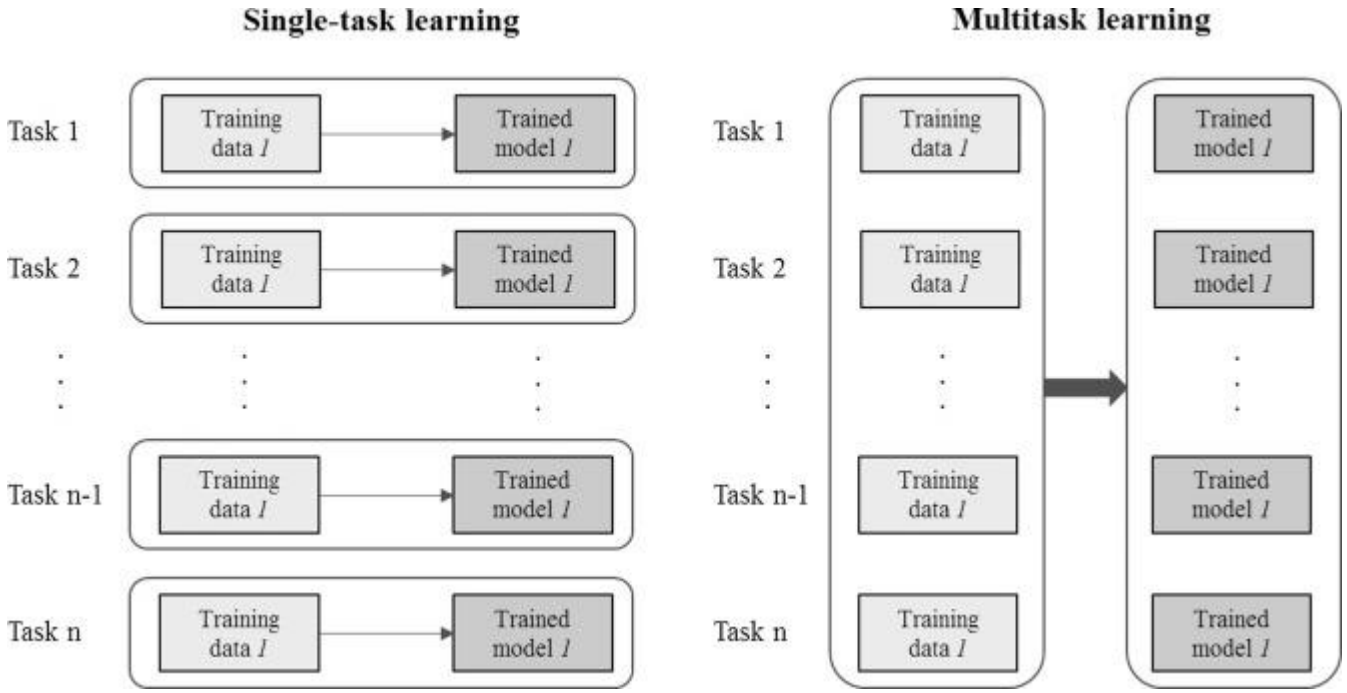


What is Transfer Learning?



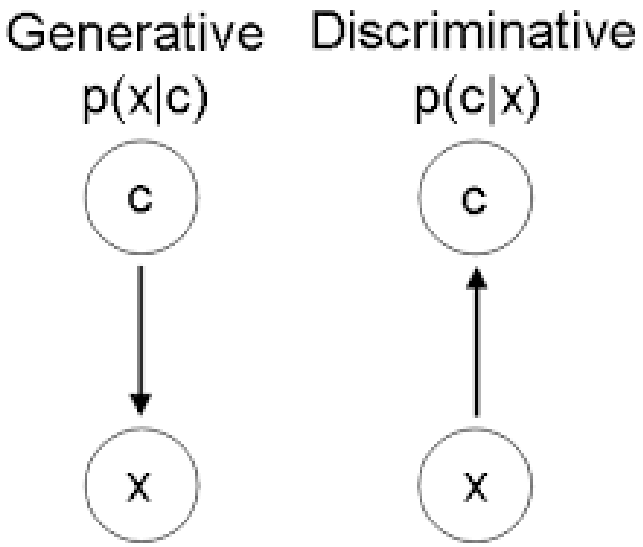


What is Multi-task Learning?

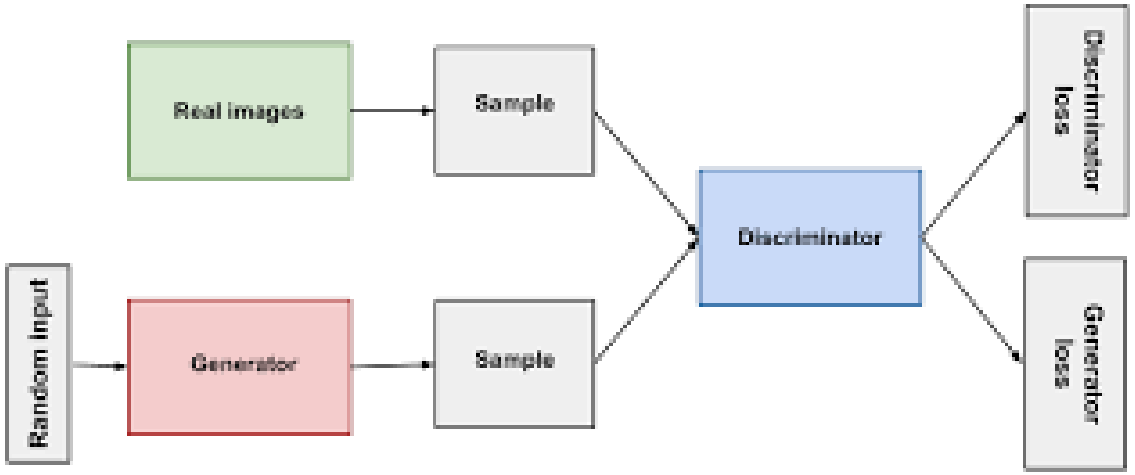


What is Generative Adversarial Network?

What are Generative and Discriminative Models?



Generative Adversarial Network?



Attention

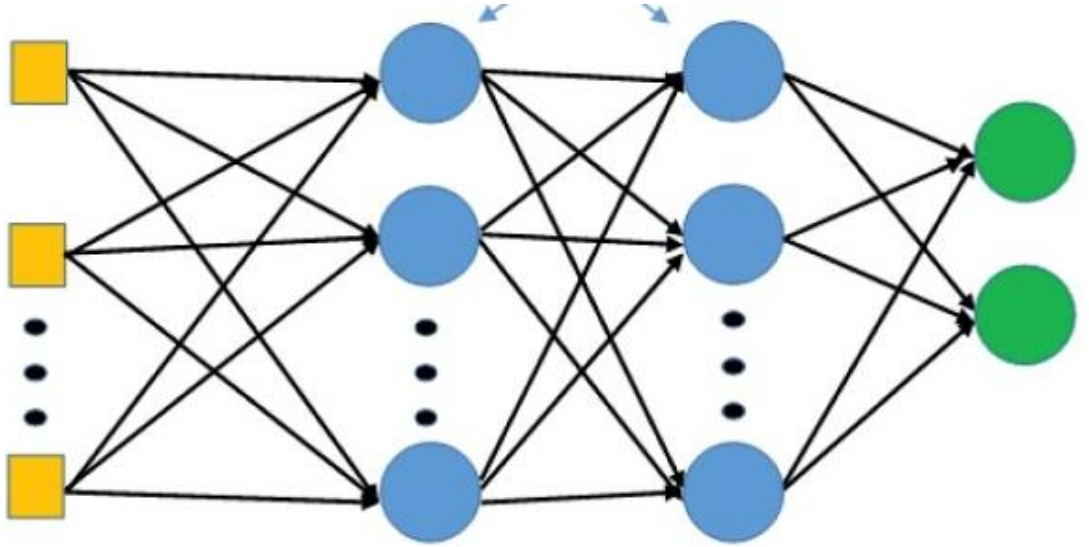
It is a nice movie

It is a nice movie

Word Embedding

It is a nice movie

Word Embedding

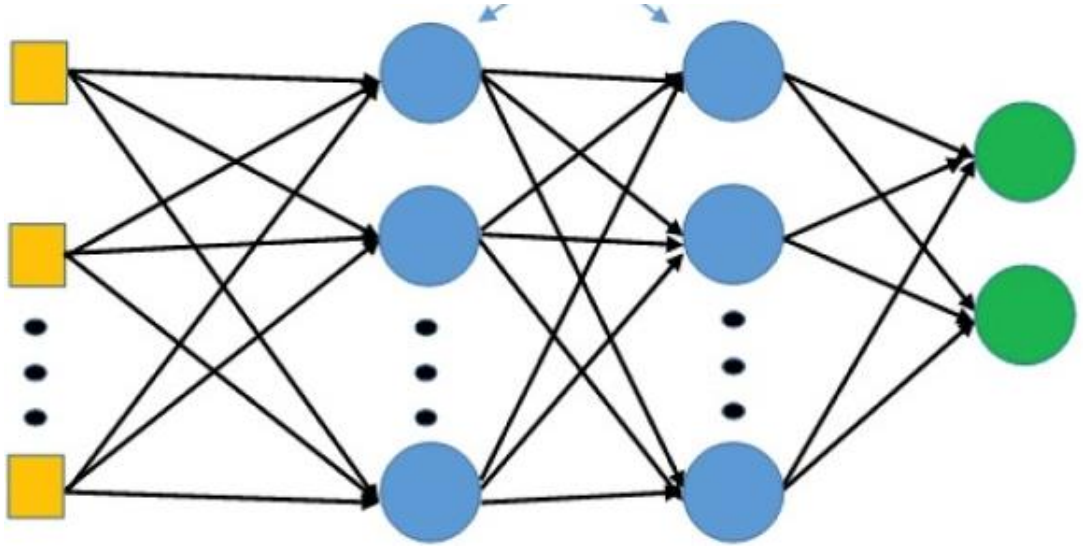


It is a nice movie

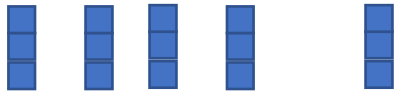
Word Embedding

Generate Embedding of the sentence

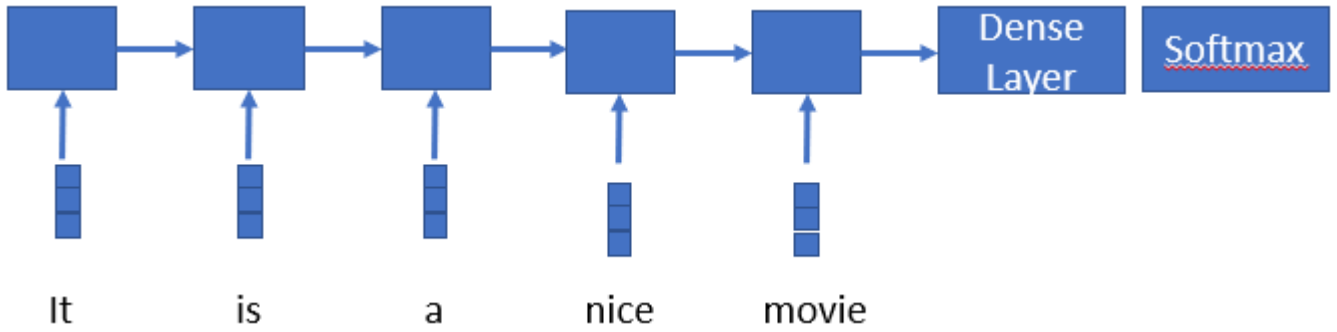
- **Sum of the embedding vector**
- **Average of the embedding vector**
- **Concatenation of the vectors**
- **Or, any sentence embedding**



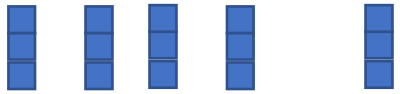
It is a nice movie



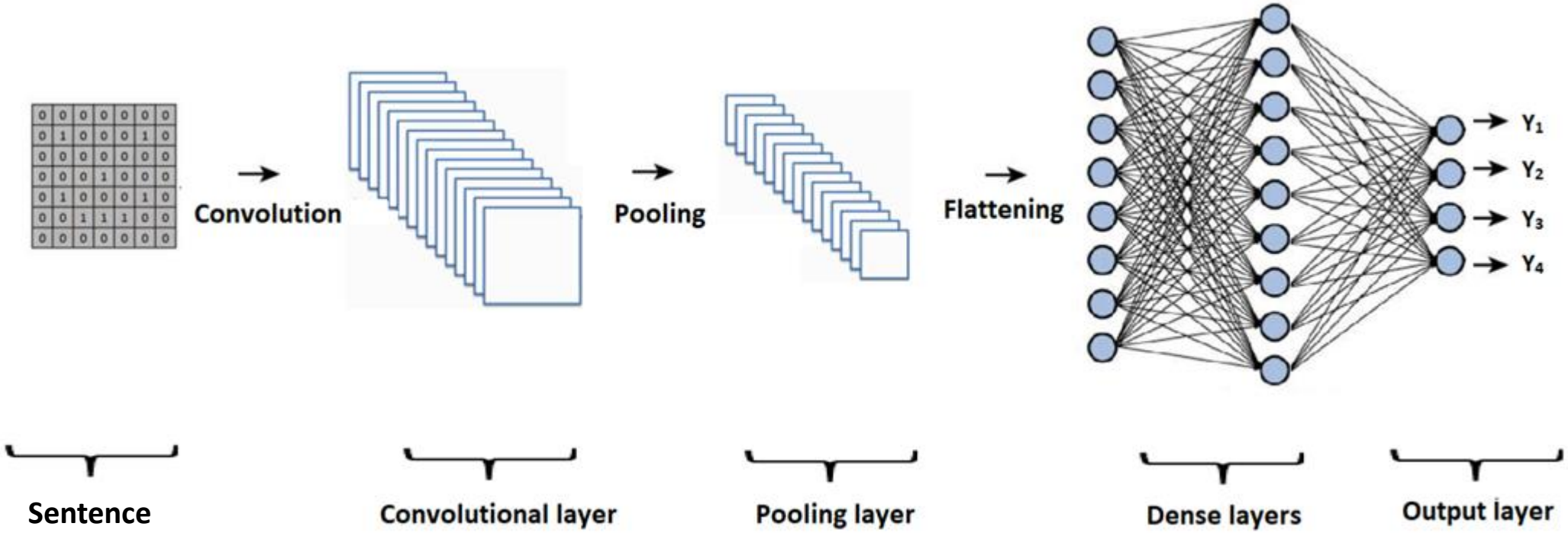
Word Embedding



It is a nice movie



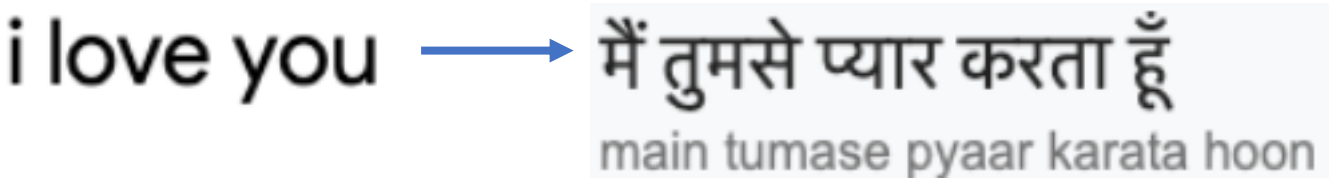
Word Embedding



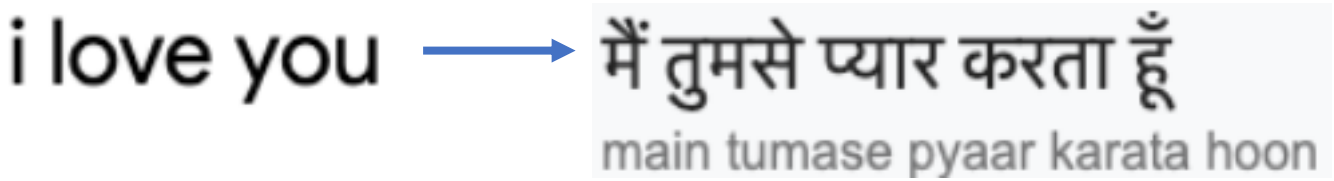
Sequence-to-Sequence Problem

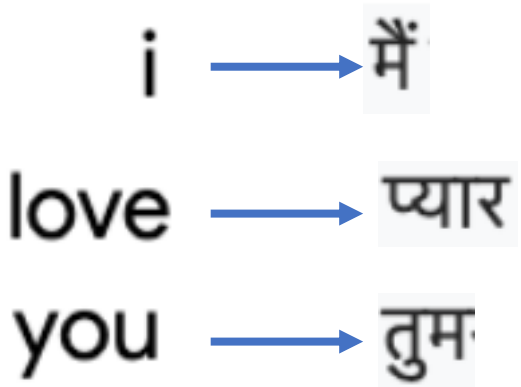
- POS
- NER
- Machine Translation
- Machine Transliteration
- so on

Sequence-to-Sequence (Seq2Seq) Model

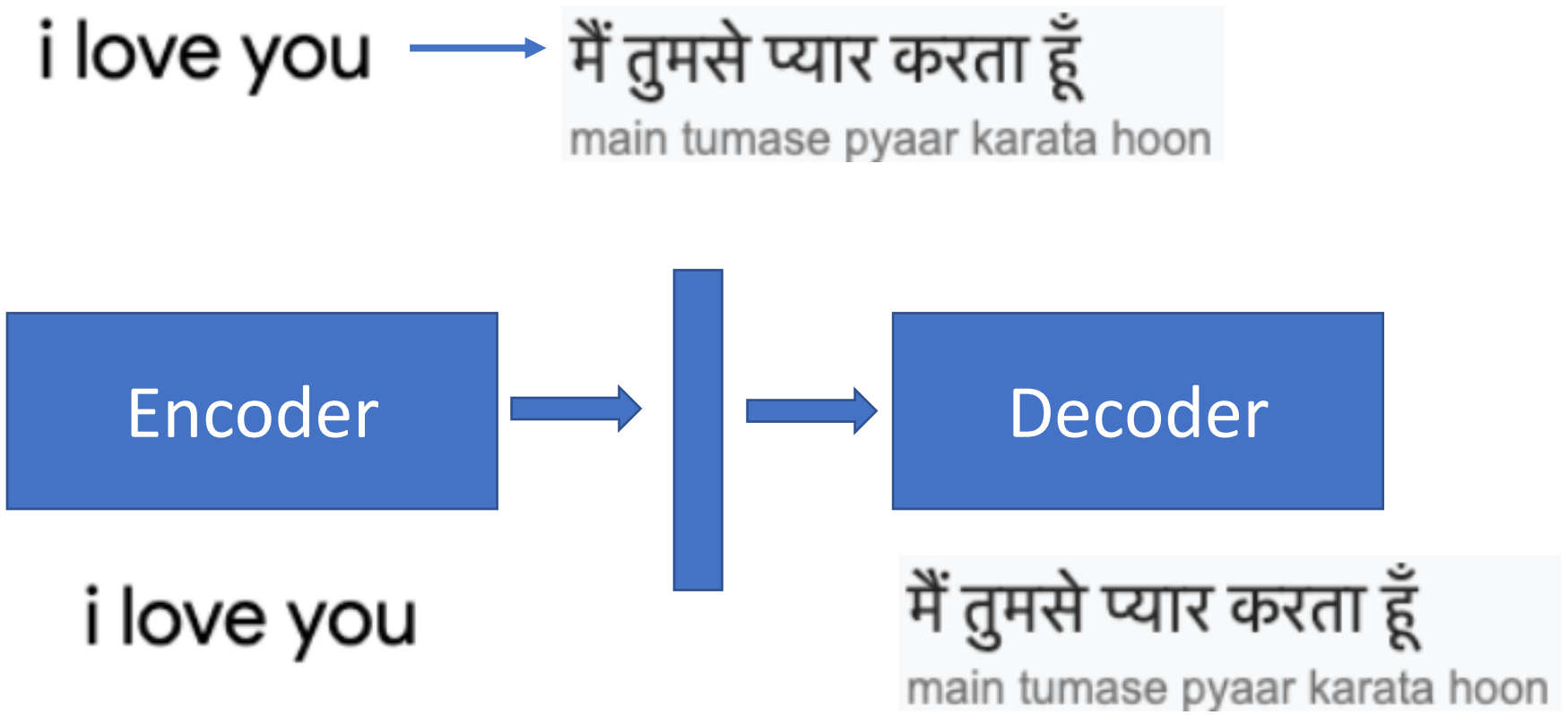


Sequence-to-Sequence (Seq2Seq) Model



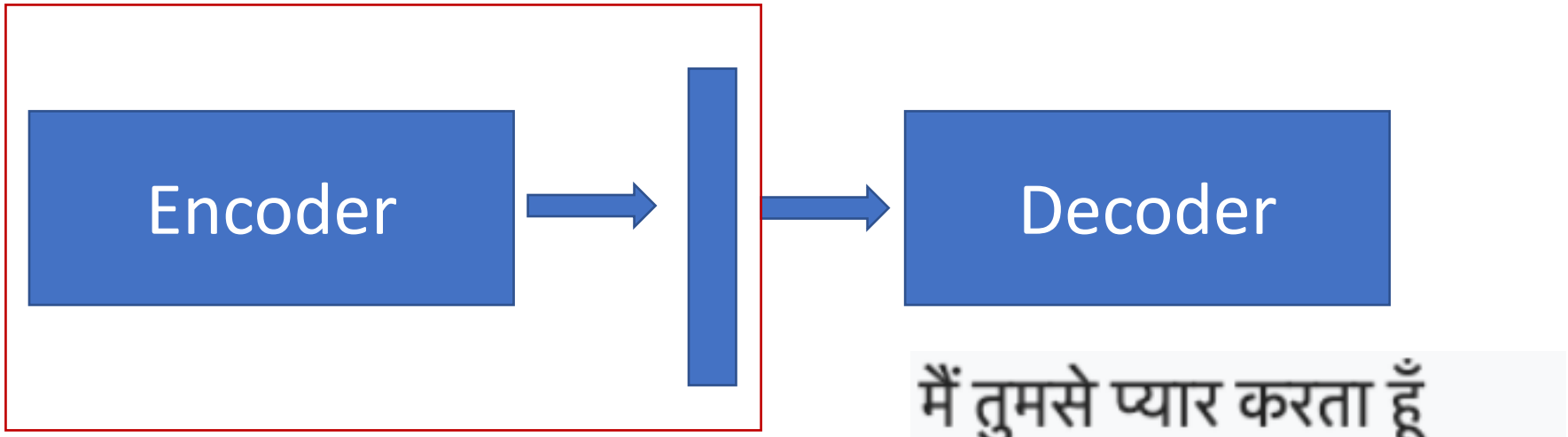


Sequence-to-Sequence (Seq2Seq) Model



Sequence-to-Sequence (Seq2Seq) Model

i love you → मैं तुमसे प्यार करता हूँ
main tumase pyaar karata hoon

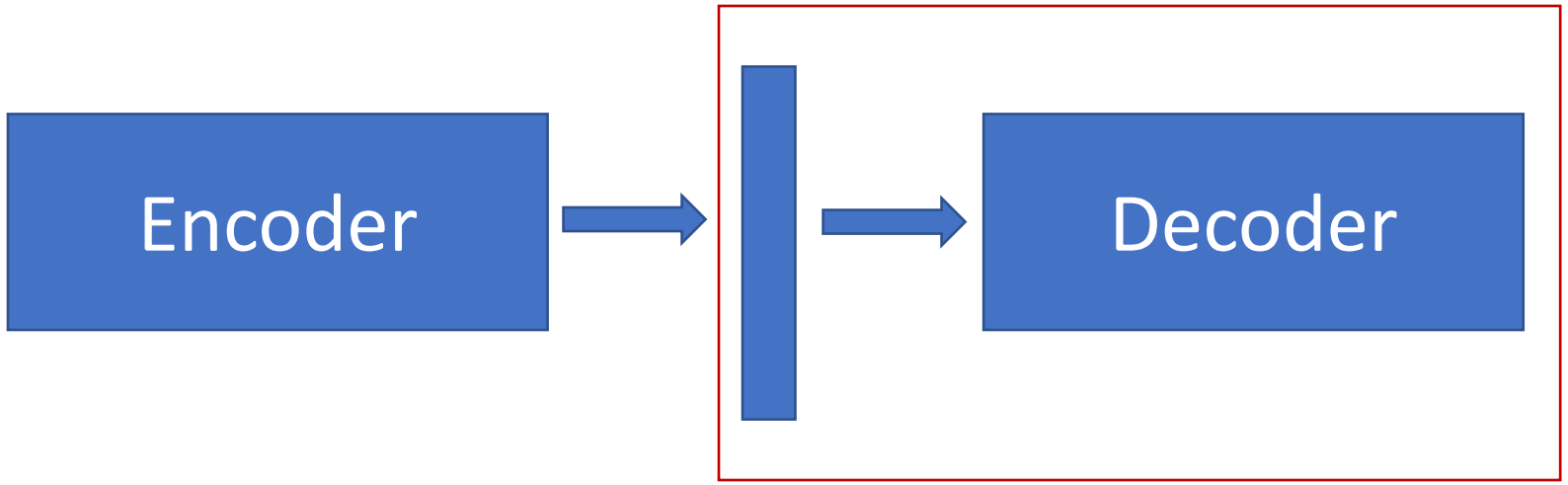


मैं तुमसे प्यार करता हूँ
main tumase pyaar karata hoon

i love you

Sequence-to-Sequence (Seq2Seq) Model

i love you → मैं तुमसे प्यार करता हूँ
main tumase pyaar karata hoon



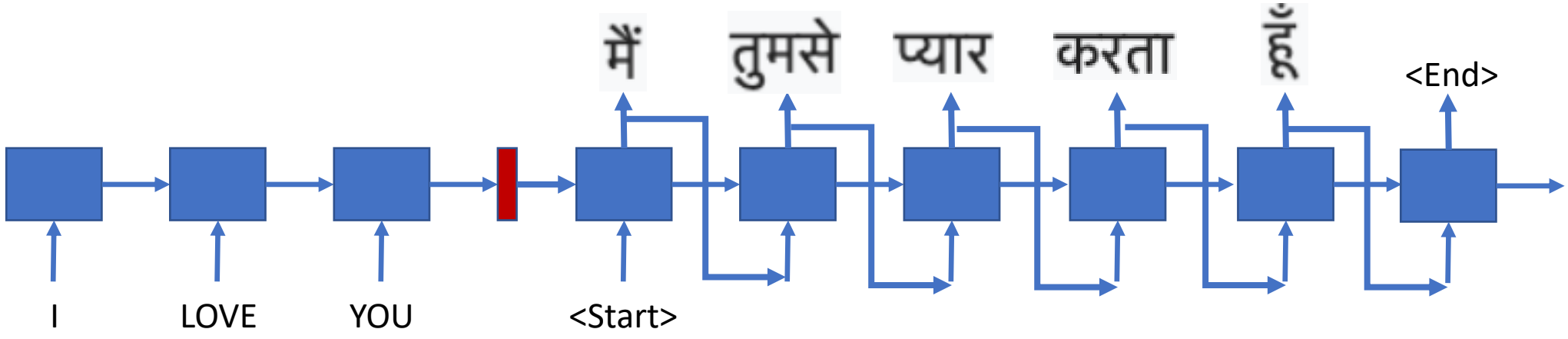
i love you

मैं तुमसे प्यार करता हूँ
main tumase pyaar karata hoon

Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

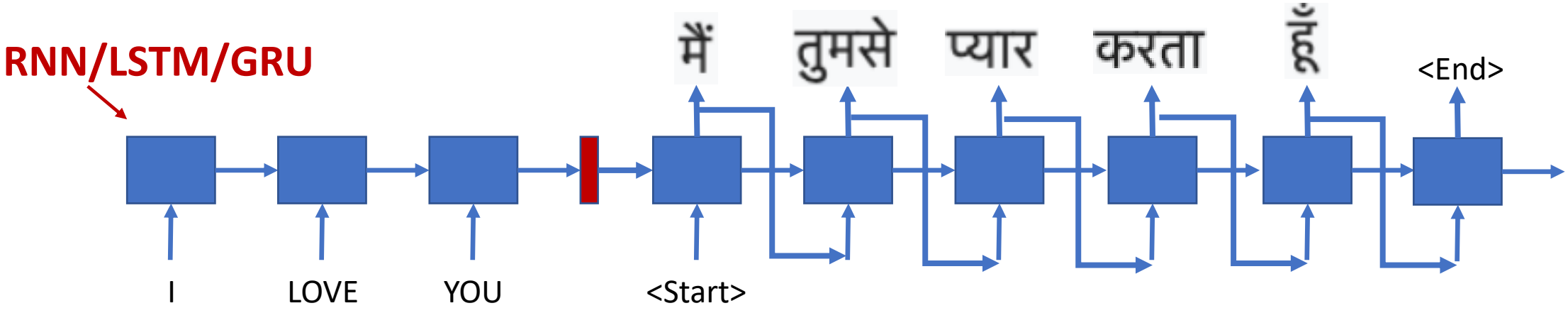
main tumase pyaar karata hoon



Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

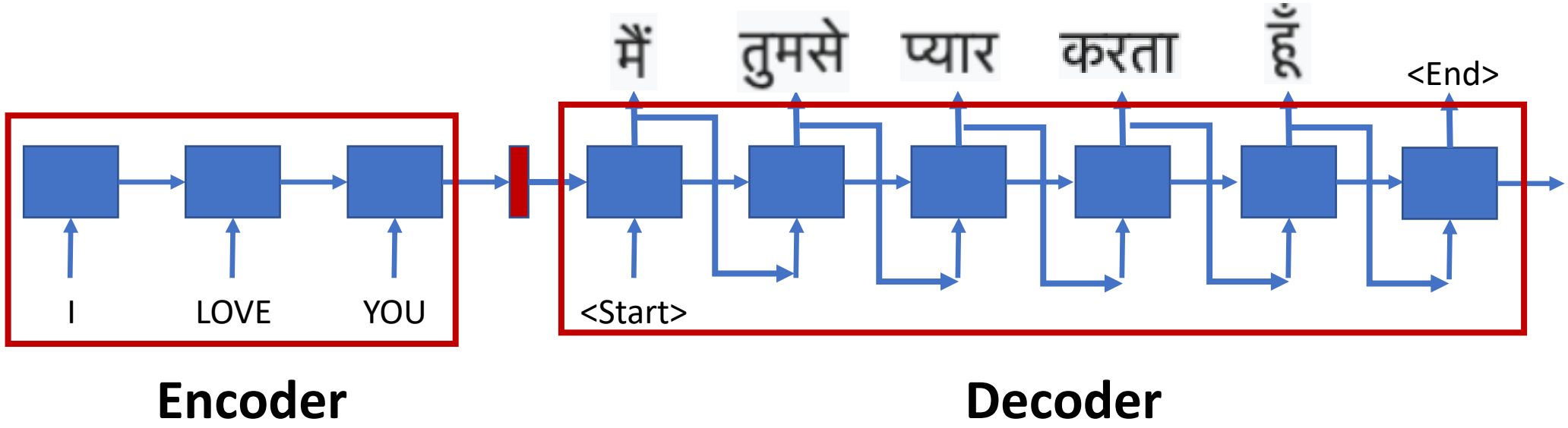
main tumase pyaar karata hoon



Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

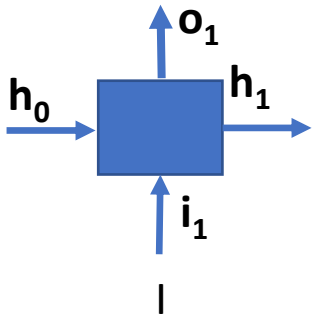
main tumase pyaar karata hoon



Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

main tumase pyaar karata hoon

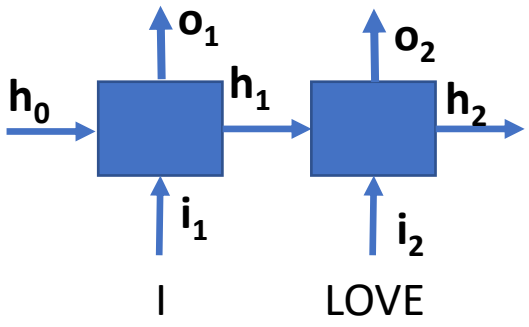


Encoder

Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

main tumase pyaar karata hoon

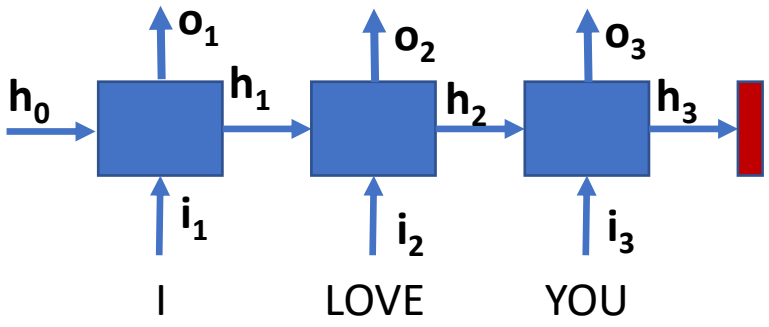


Encoder

Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

main tumase pyaar karata hoon

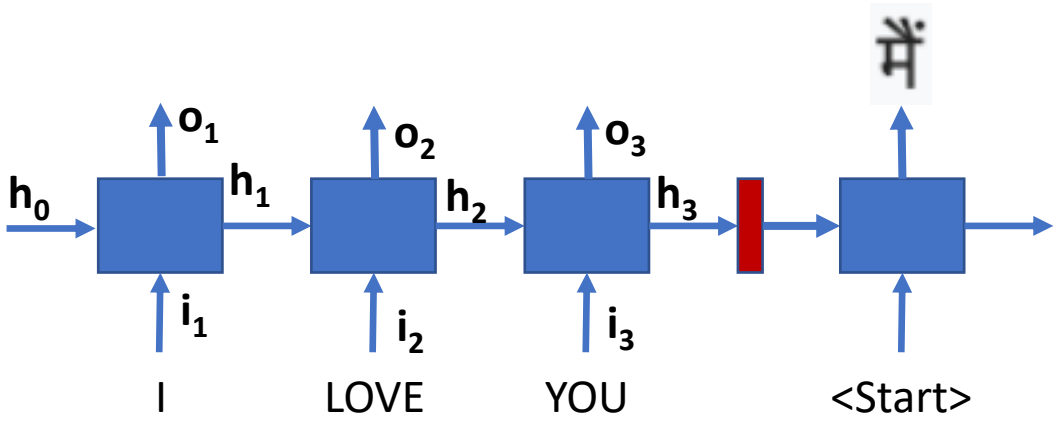


Encoder

Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

main tumase pyaar karata hoon

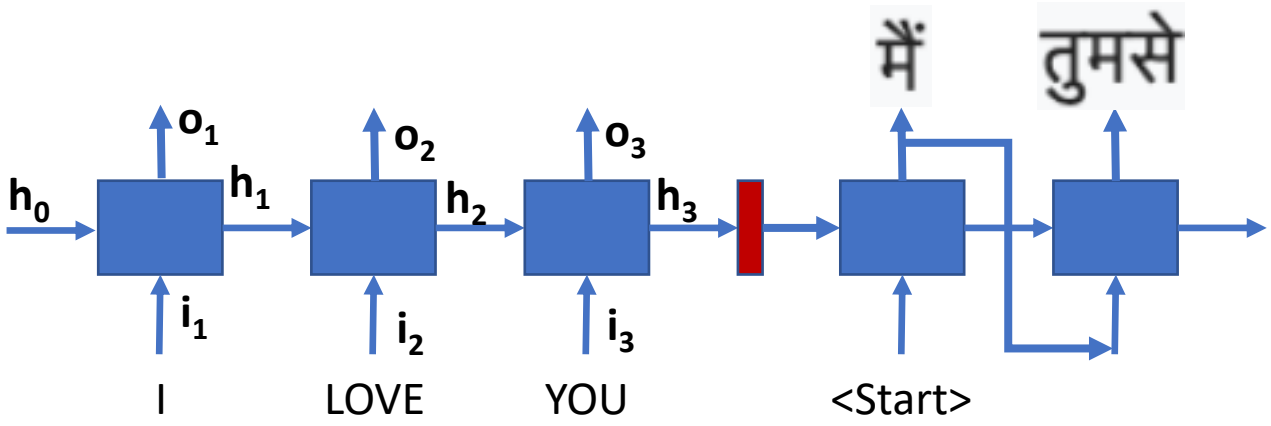


Encoder

Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

main tumase pyaar karata hoon

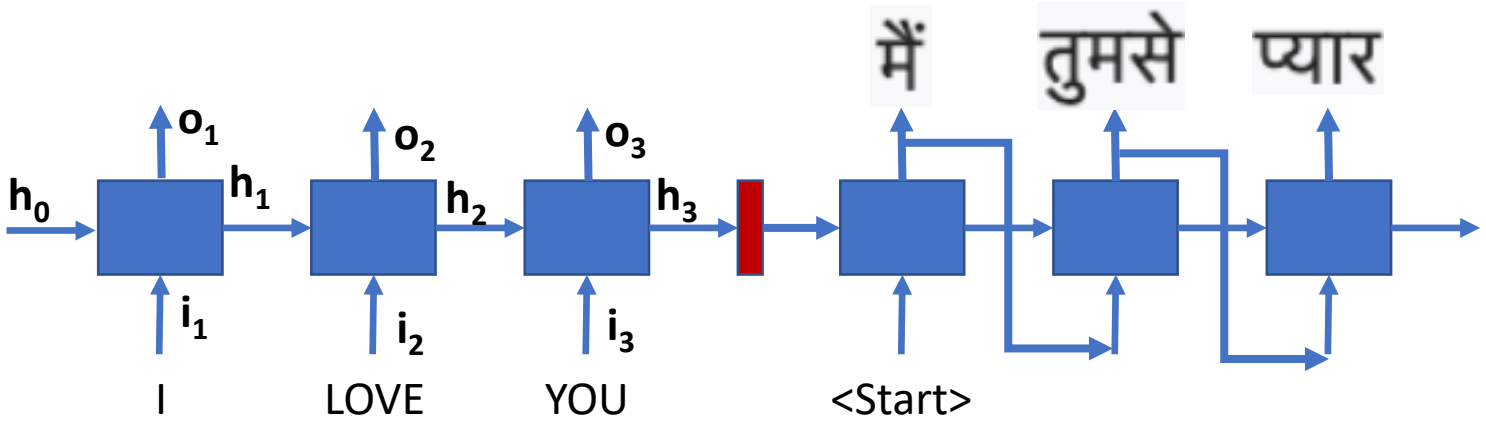


Encoder

Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

main tumase pyaar karata hoon

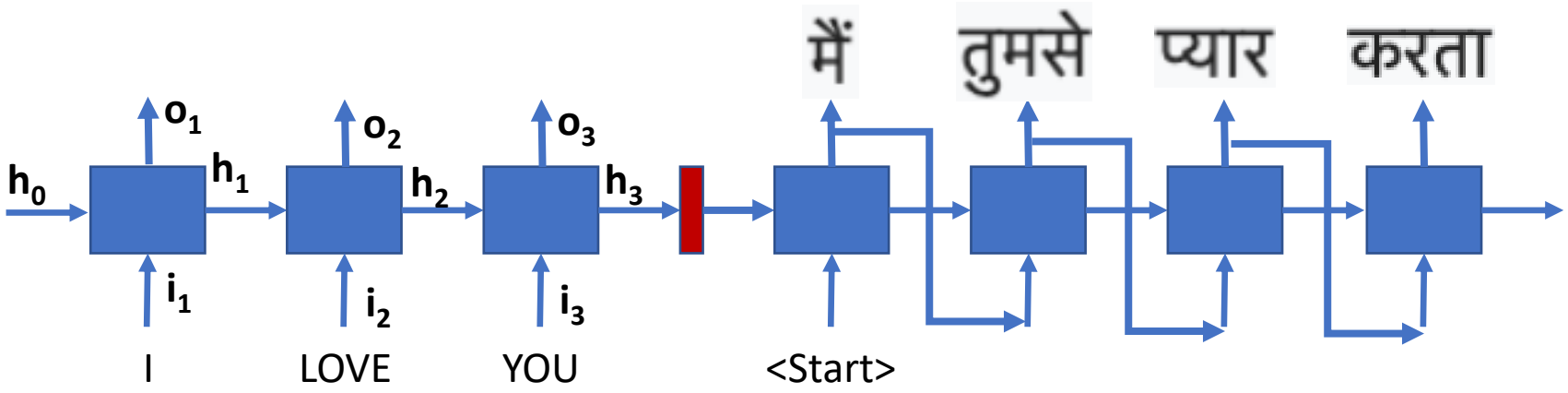


Encoder **Decoder**

Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

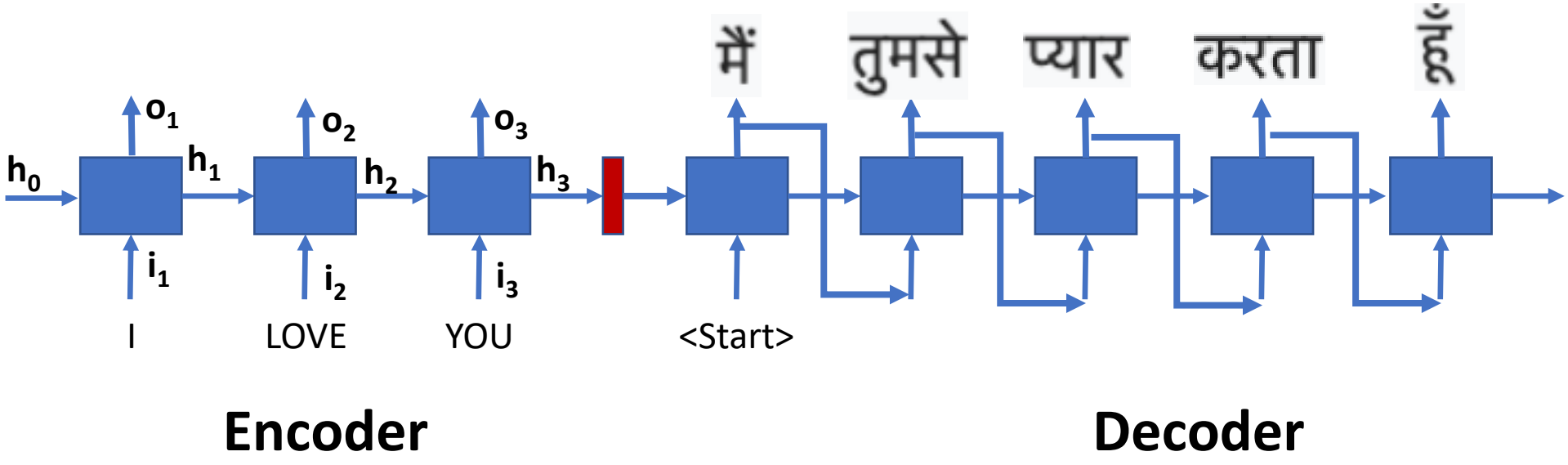
main tumase pyaar karata hoon



Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

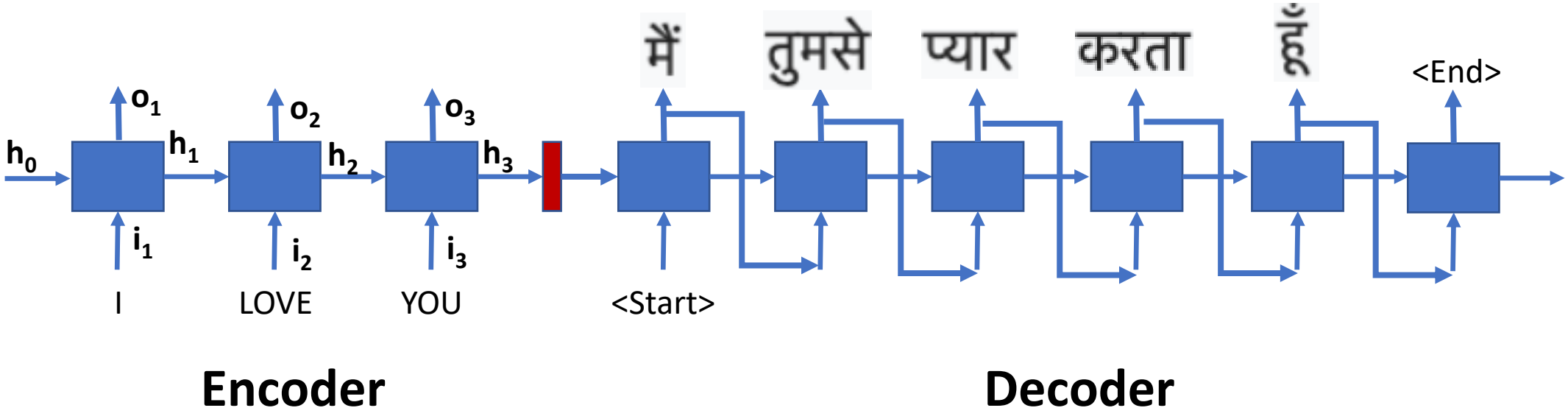
main tumase pyaar karata hoon



Sequence-to-Sequence (Seq2Seq)

i love you → मैं तुमसे प्यार करता हूँ

main tumase pyaar karata hoon



Attention

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

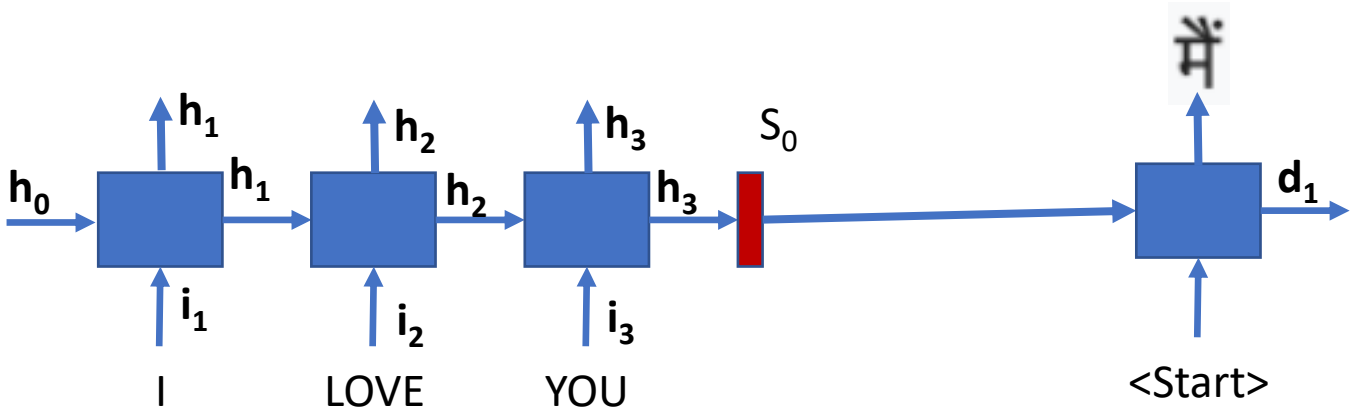
Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

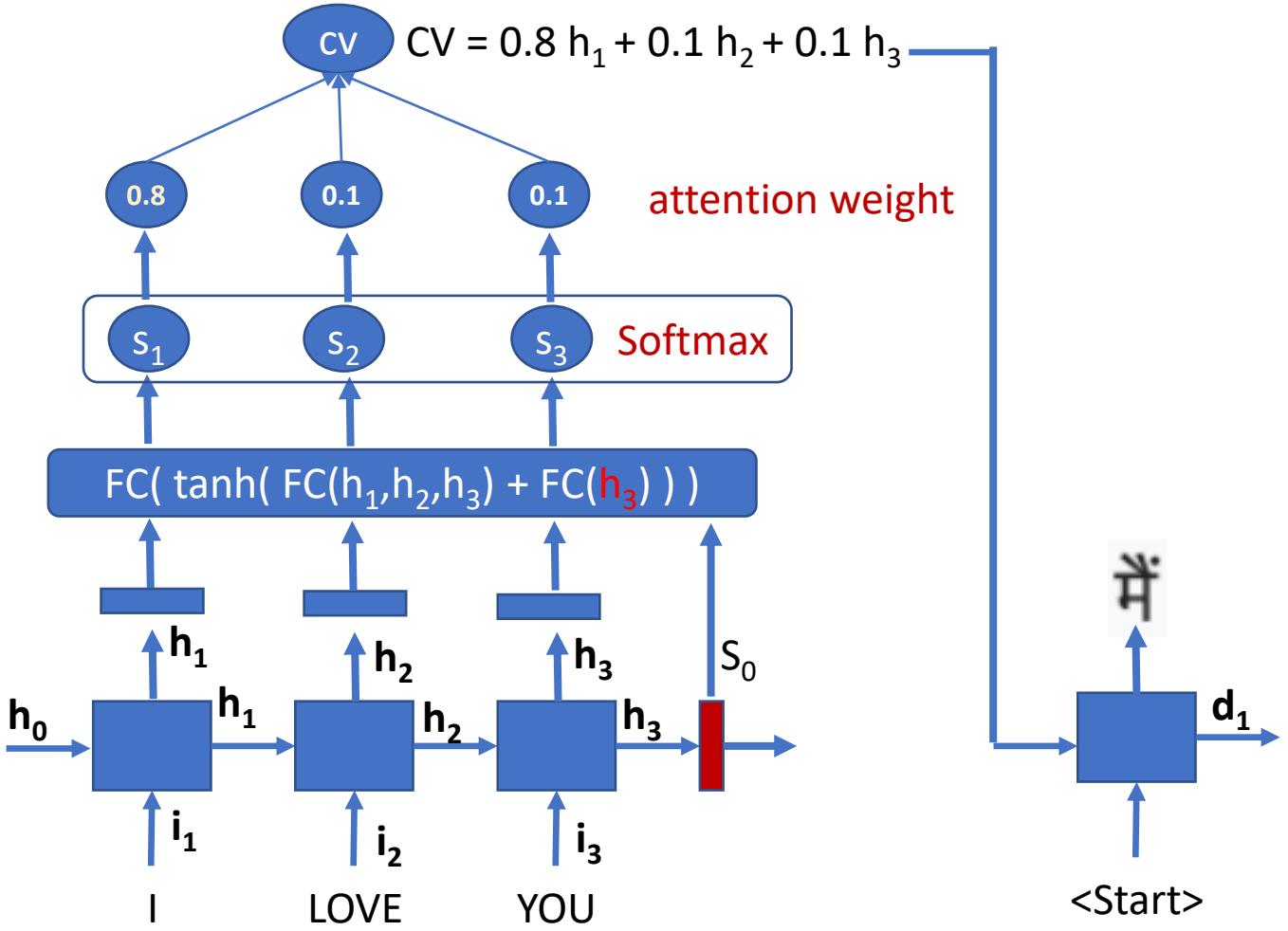
ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

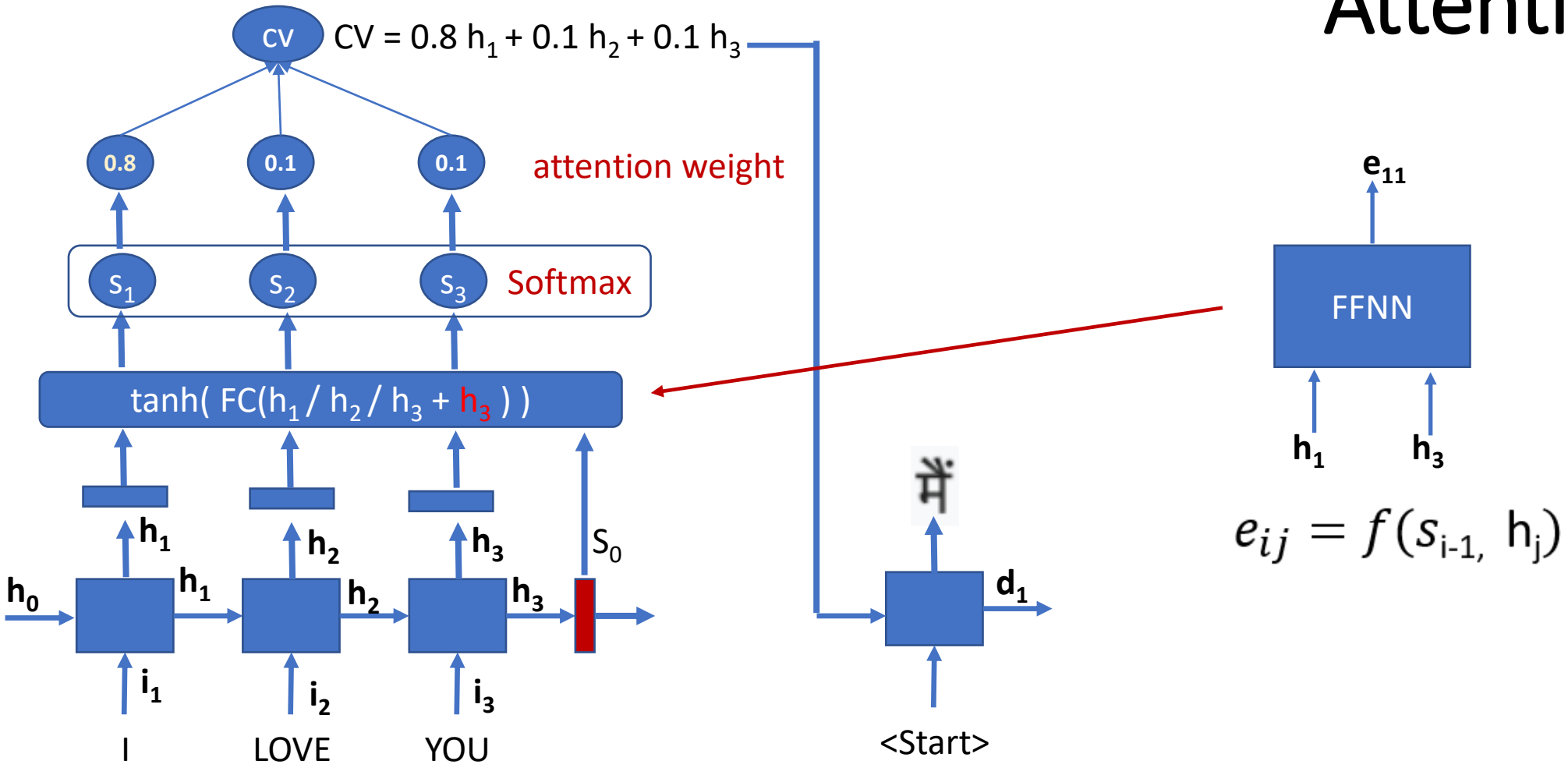
Attention



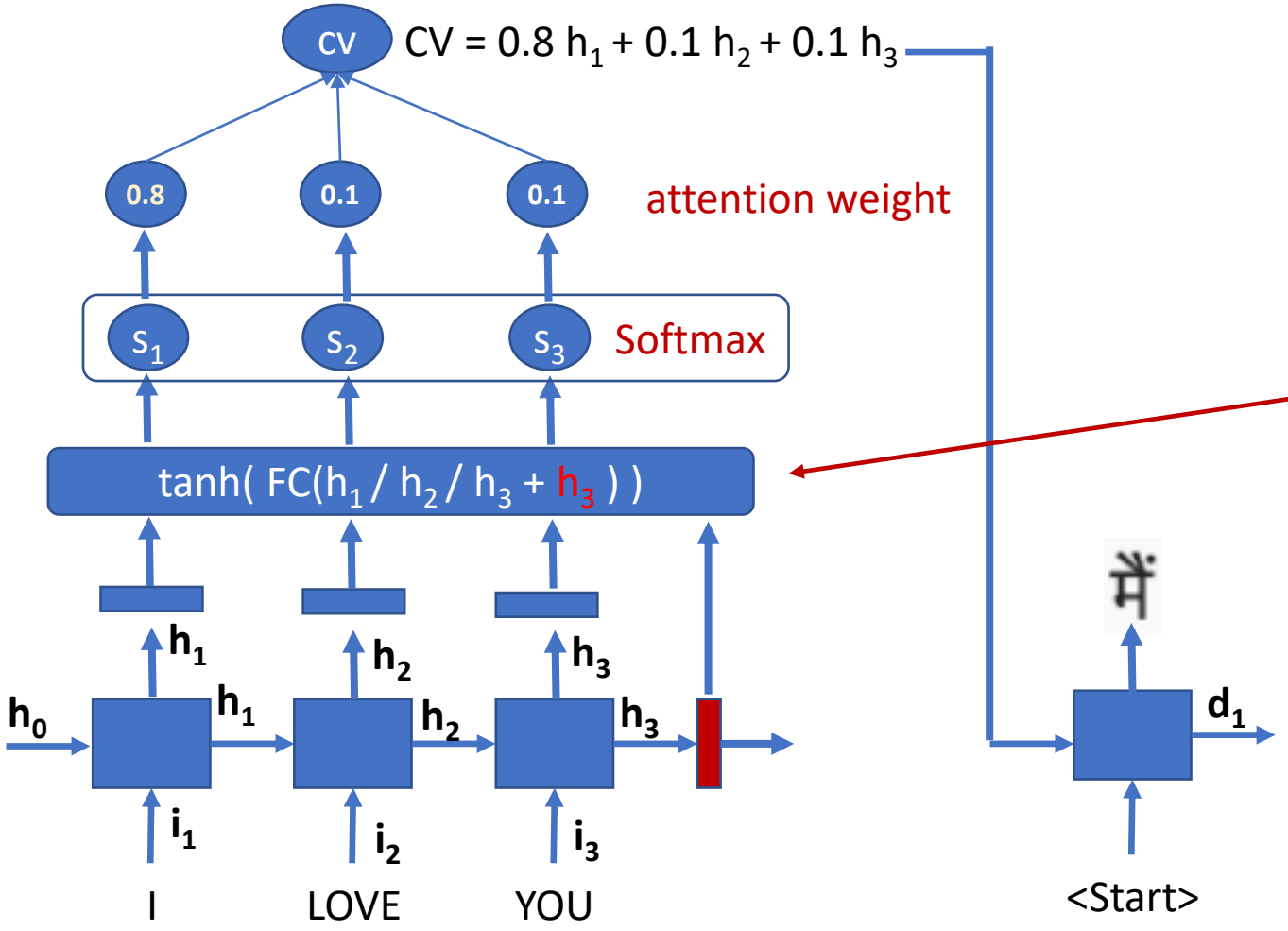
Attention

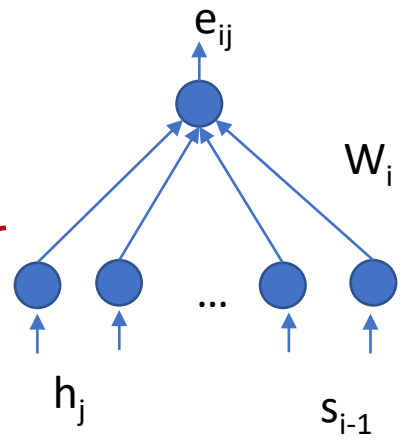

CV
CV = 0.8 h1 + 0.1 h2 + 0.1 h3
0.8
0.1
0.1
attention weight
S1
S2
S3
Softmax
FC(tanh(FC(h1,h2,h3) + FC(h3)))
h0
h1
h2
h3
S0
i1
i2
i3
I
LOVE
YOU
d1
<Start>

Attention


CV
CV = 0.8 h1 + 0.1 h2 + 0.1 h3
0.8
0.1
0.1
attention weight
S1
S2
S3
Softmax
tanh(FC(h1 / h2 / h3 + h3))
h1
h2
h3
S0
h0
h1
h2
h3
i1
i2
i3
I
LOVE
YOU
d1
<Start>
e11
FFNN
h1
h3
$e_{ij} = f(s_{i-1}, h_j)$

Attention

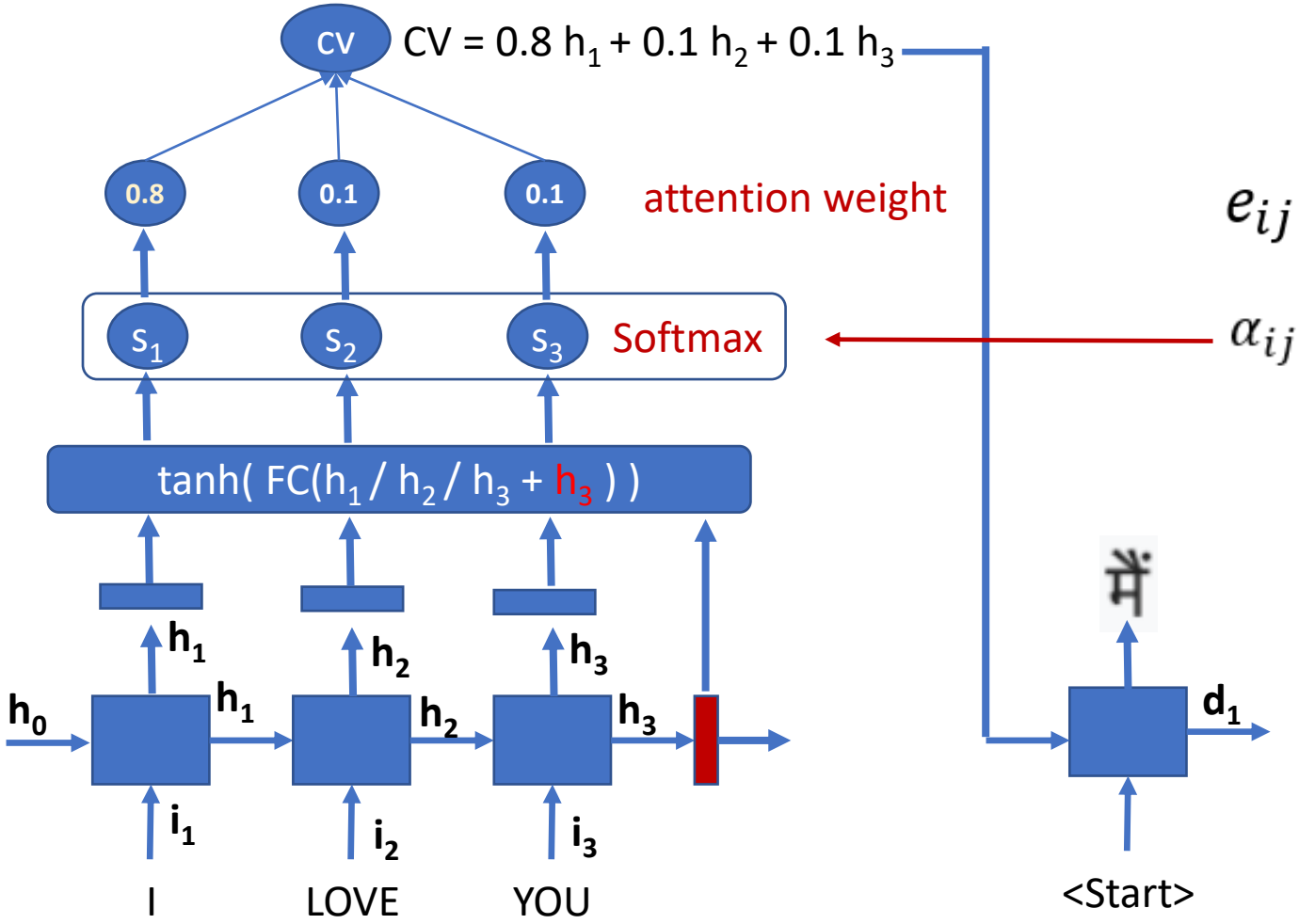




$$e_{ij} = f(s_{i-1}, h_j)$$

e_{ij} = tanh($W^T x$)
x = concatenation of S_{i-1} and h_j

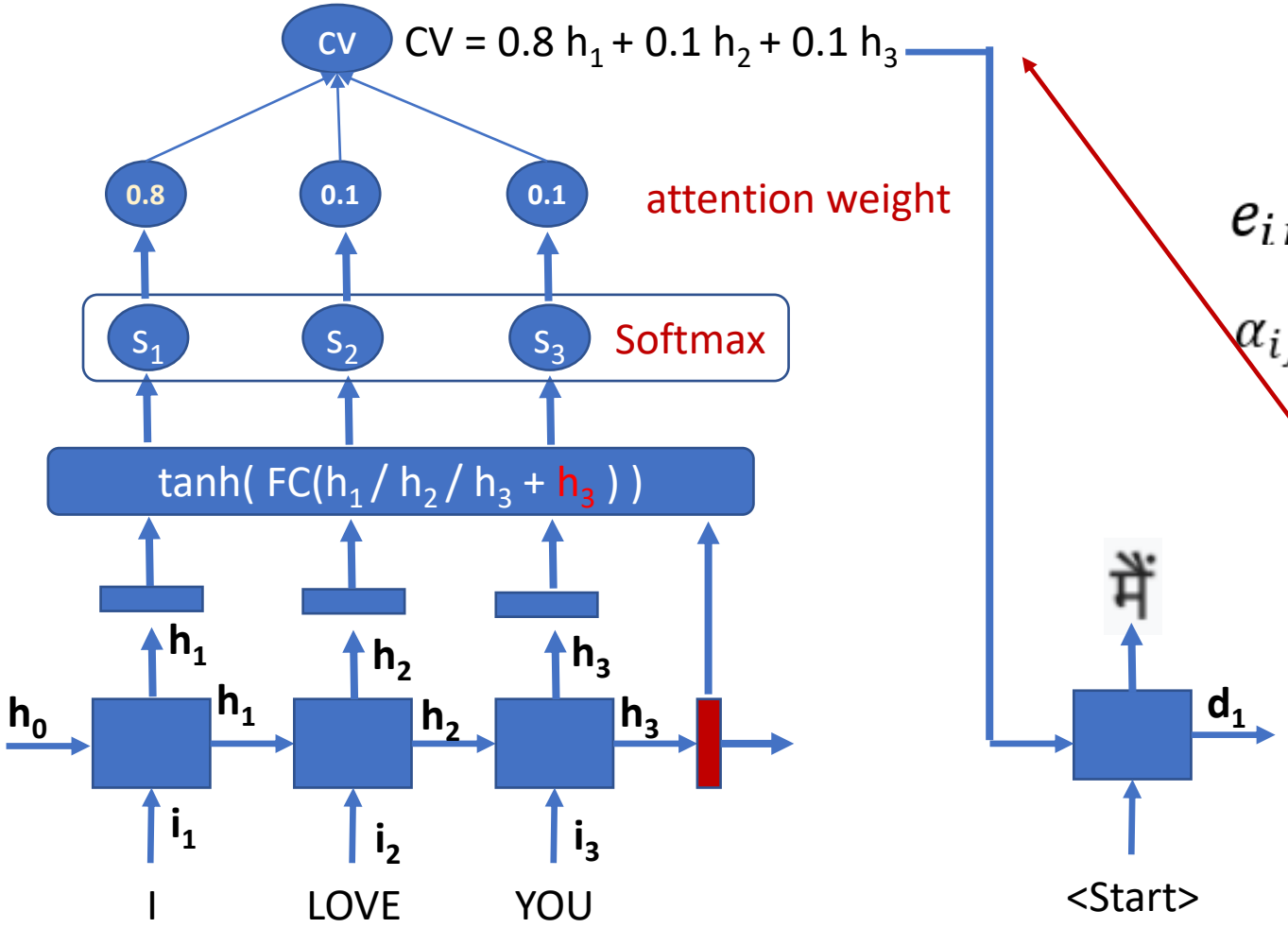
Attention



$$e_{ij} = f(s_{i-1}, h_j)$$

$$\alpha_{ij} = \mathrm{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

Attention



$$e_{ij} = f(s_{i-1}, h_i)$$

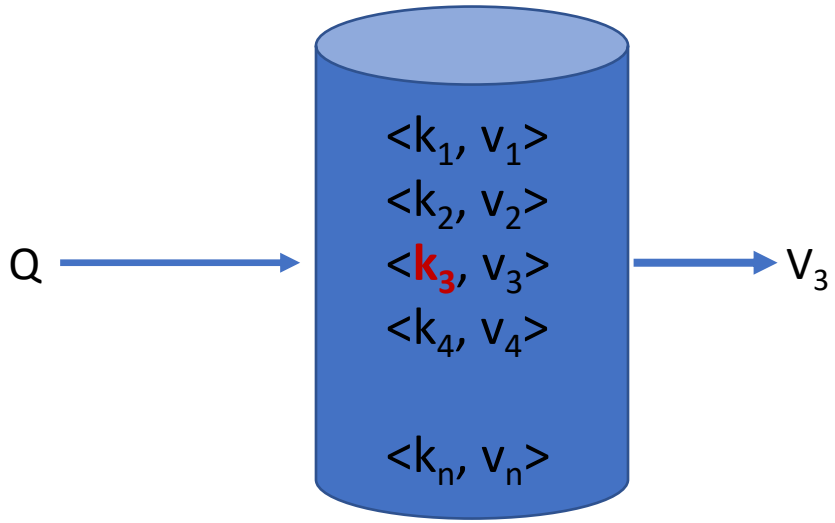
$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

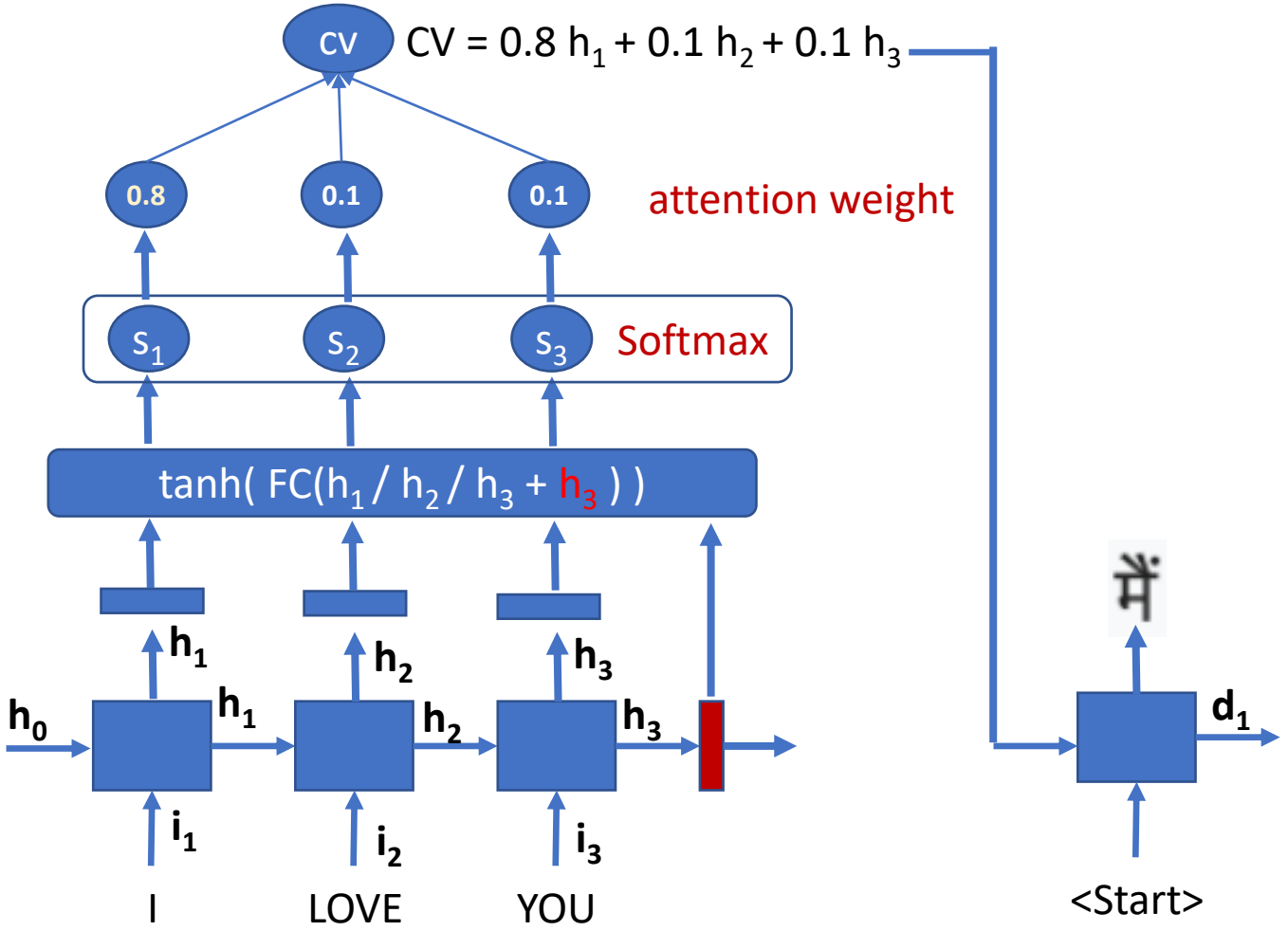
Attention – general concept

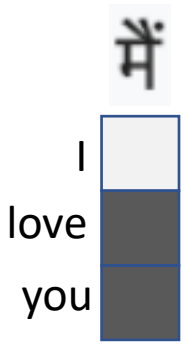
Q : Query
K : Key
V : Value

$$A(q, K, V) = \sum_i \frac{\exp(e_{qk_i})}{\sum_j \exp(e_{qk_j})} v_i$$

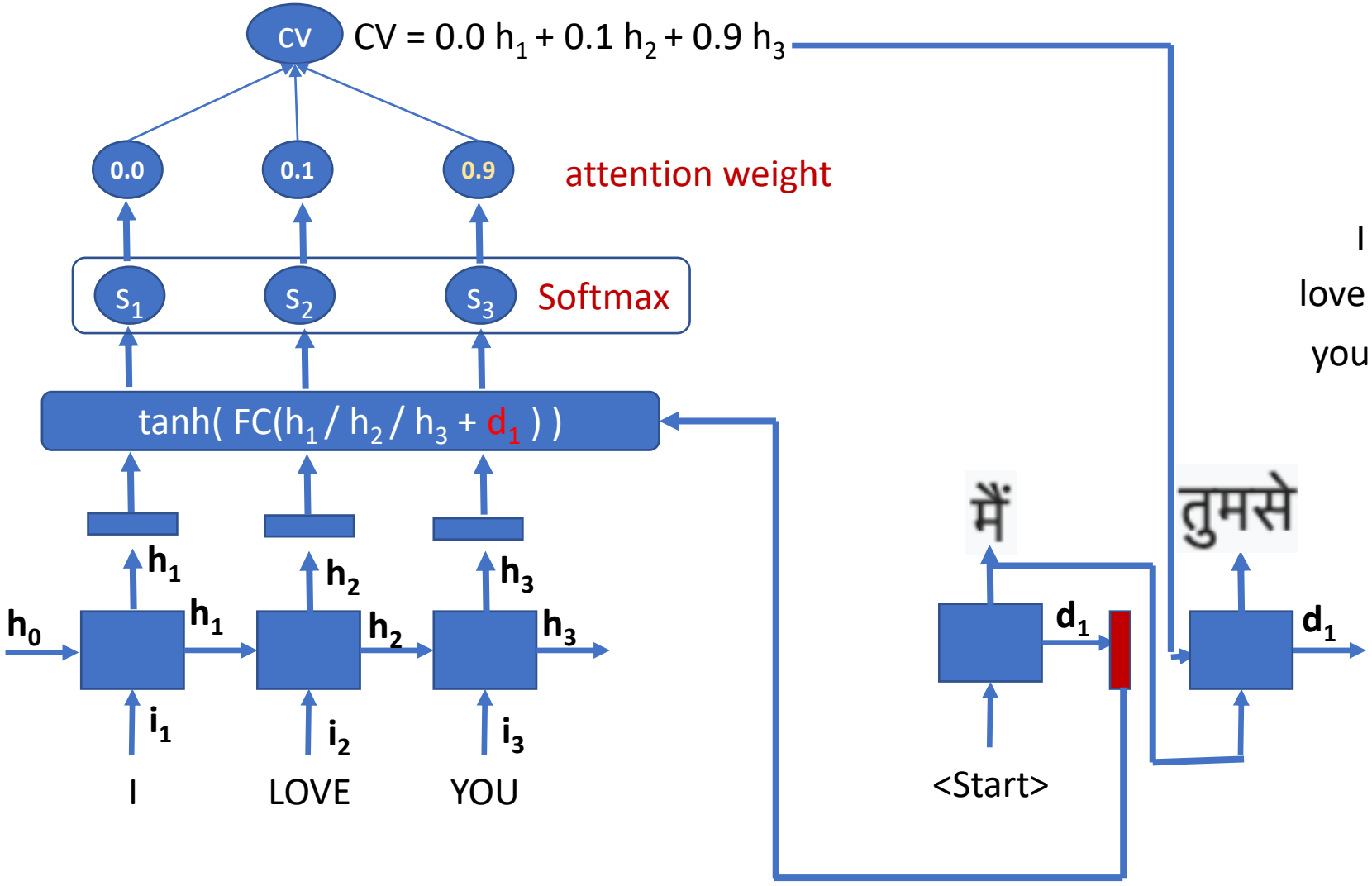


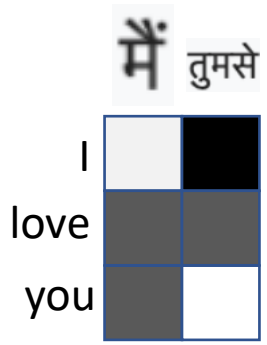
Attention


CV
CV = 0.8 h1 + 0.1 h2 + 0.1 h3
0.8
0.1
0.1
attention weight
S1
S2
S3
Softmax
tanh(FC(h1 / h2 / h3 + h3))
h0
h1
h2
h3
i1
i2
i3
I
LOVE
YOU
<Start>
d1

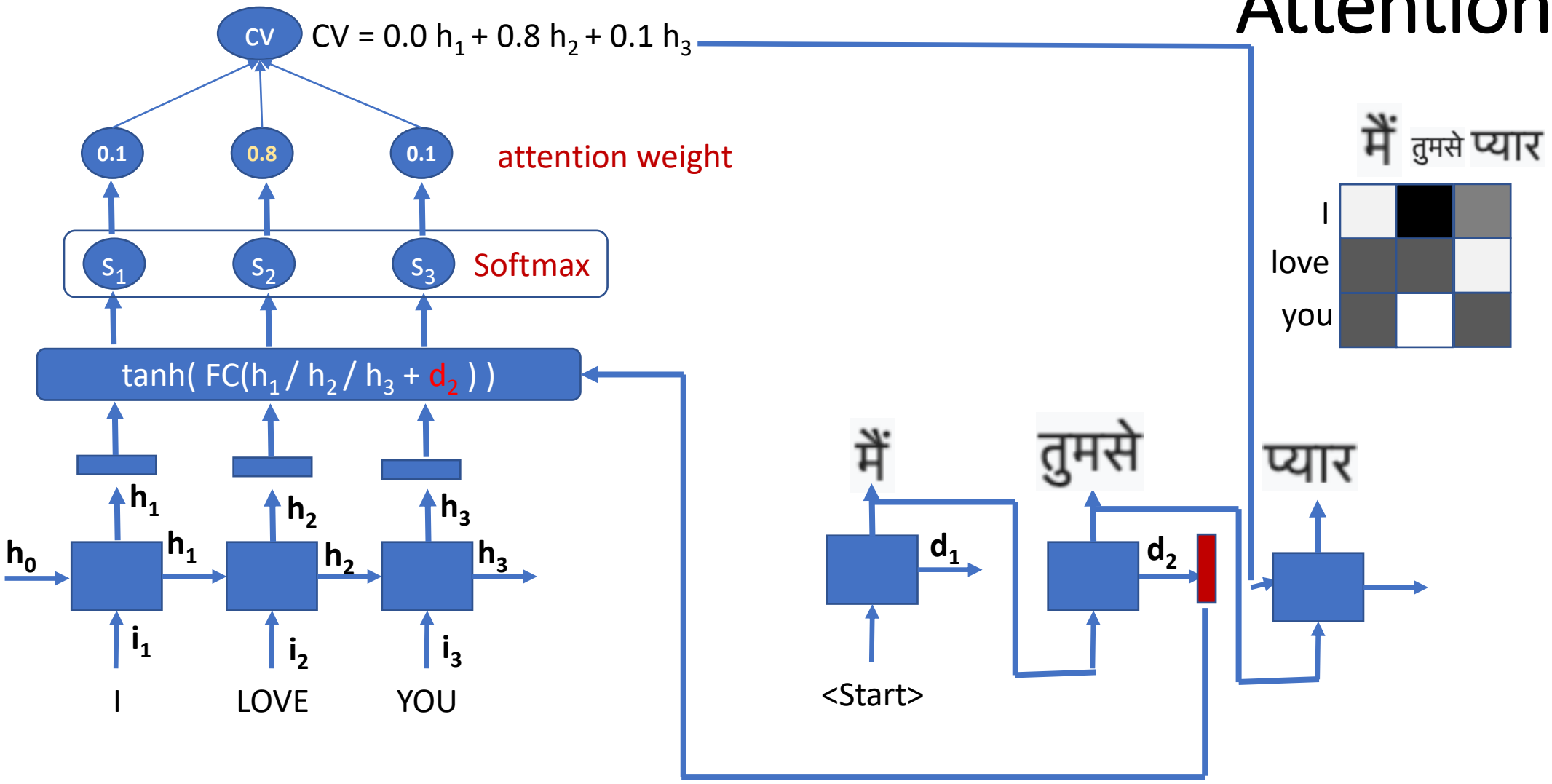

I
love
you

Attention


CV
CV = 0.0 h1 + 0.1 h2 + 0.9 h3
0.0
0.1
0.9
attention weight
S1
S2
S3
Softmax
tanh(FC(h1 / h2 / h3 + d1))
h0
h1
h2
h3
i1
i2
i3
I
LOVE
YOU
मैं
तुमसे
d1
<Start>


मैं
तुमसे
I
love
you

Attention


CV
CV = 0.0 h1 + 0.8 h2 + 0.1 h3
0.1
0.8
0.1
attention weight
S1
S2
S3
Softmax
tanh(FC(h1 / h2 / h3 + d2))
h0
h1
h2
h3
i1
i2
i3
I
LOVE
YOU
मैं
तुमसे
प्यार
d1
d2
<Start>
मैं तुमसे प्यार
I
love
you

Attention Types

Computing $e_1, \ldots, e_n \in \mathbb{R}$ from $k_1, \ldots, k_n \in \mathbb{R}^{d_1}$ and q $\in \mathbb{R}^{d_2}$

- Basic dot-product attention:

$$e_i = q^T k_i \in \mathbb{R}$$

  where $d_1 = d_2$

- Multiplicative attention:

$$e_i = q^T W k_i \in \mathbb{R}$$

  where $W \in \mathbb{R}^{d_2 x d_1}$ is a weight matrix

- Additive attention:

$$e_i = {W_3}^T \tanh(W_1 k_i + W_2 q) \in \mathbb{R}$$

  where $W_1 \in \mathbb{R}^{d_3 x d_1}$, $W_2 \in \mathbb{R}^{d_3 x d_2}$ are weight matrices and $W_3 \in \mathbb{R}^{d_3}$ is a weight vector.

Transformer

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

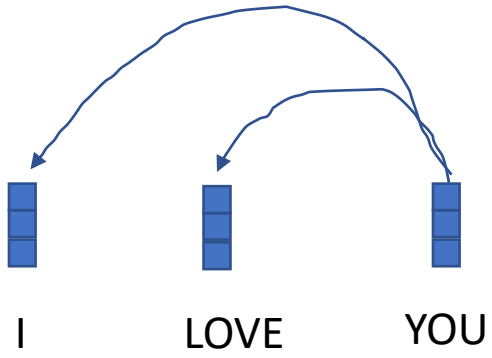
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

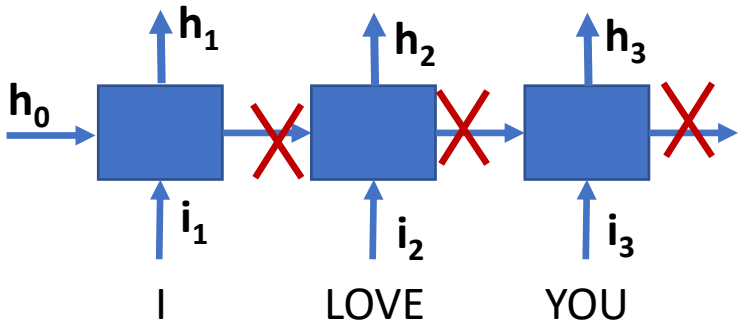
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

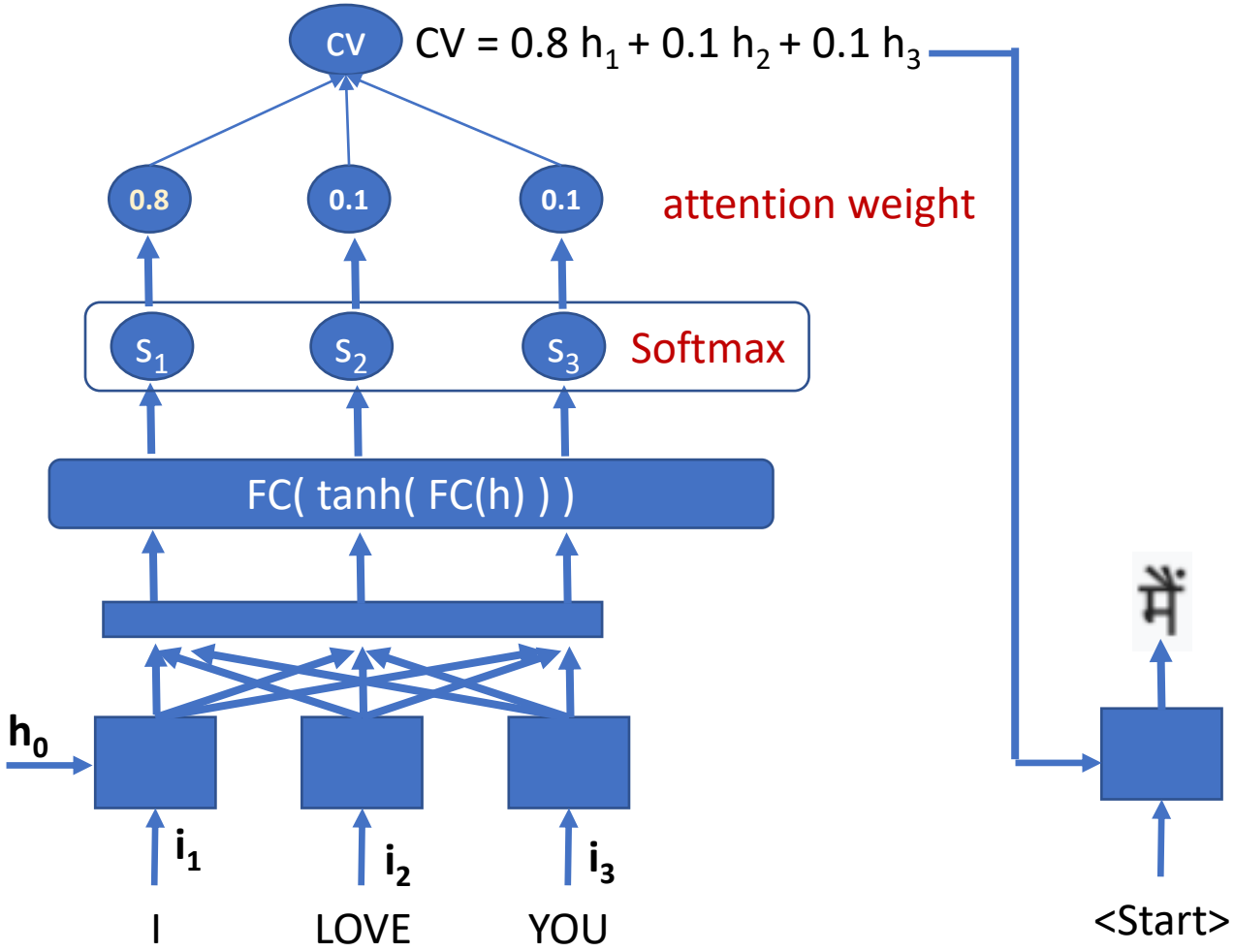
Self Attention

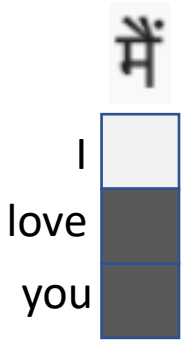


Transformer

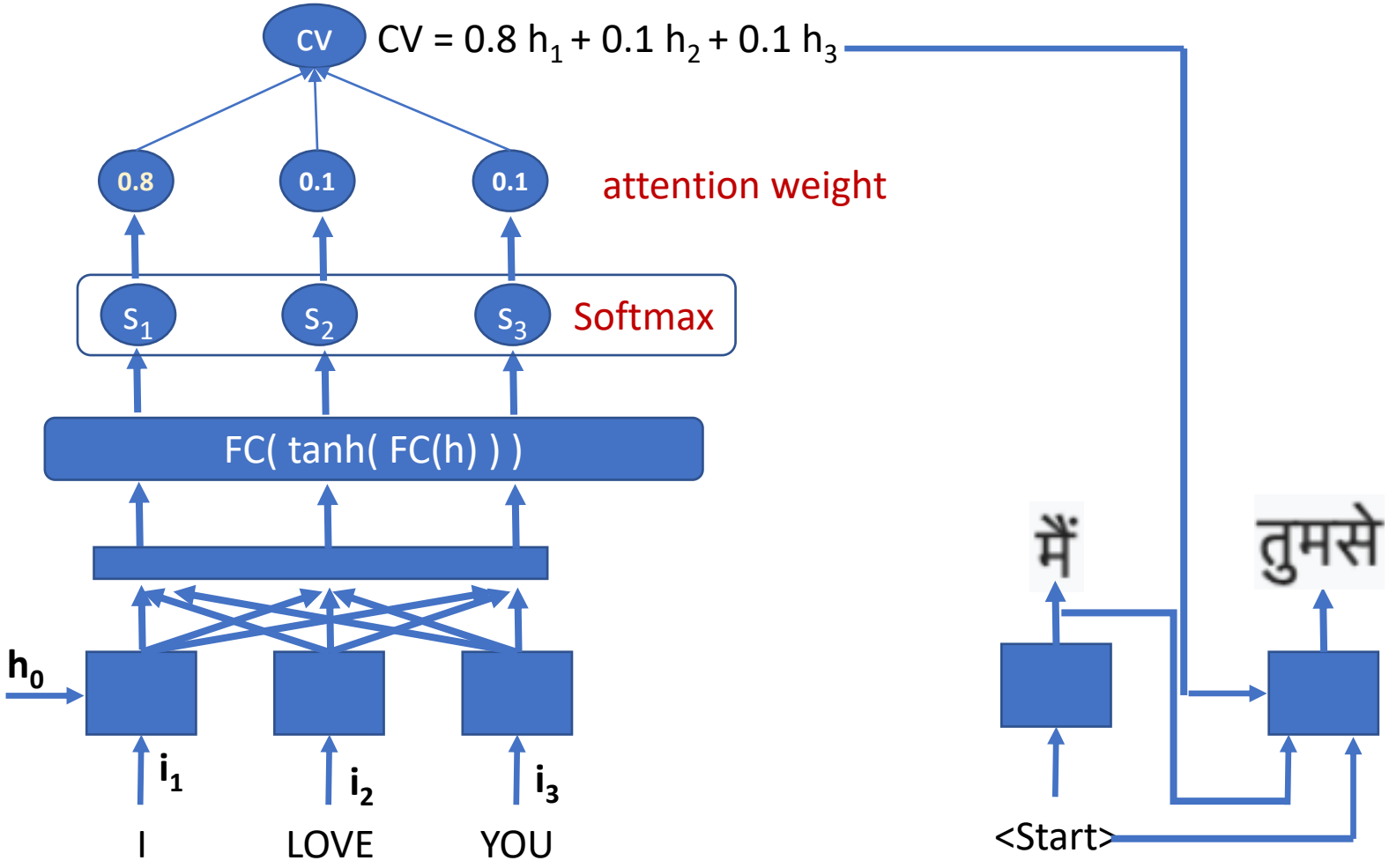


Attention

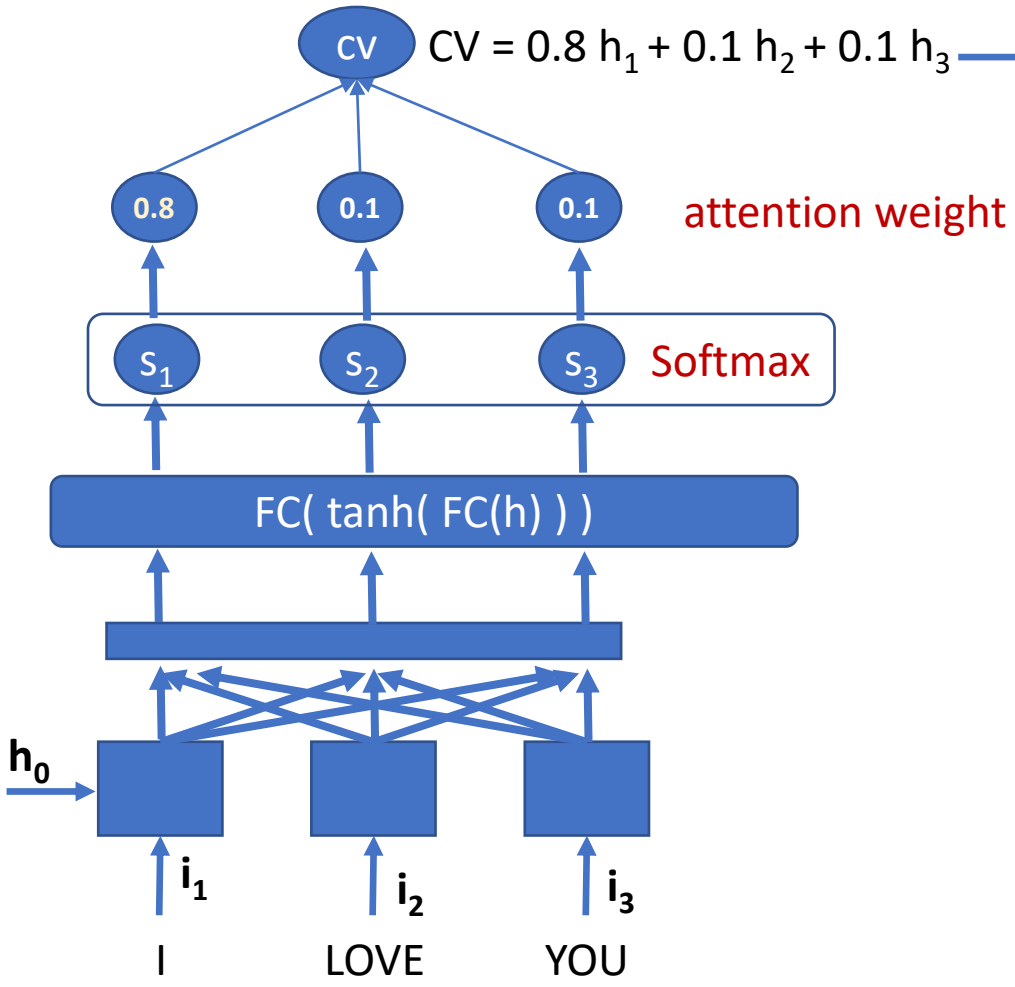

CV
CV = 0.8 h1 + 0.1 h2 + 0.1 h3
0.8
0.1
0.1
attention weight
S1
S2
S3
Softmax
FC(tanh(FC(h)))
h0
i1
i2
i3
I
LOVE
YOU
मैं
<Start>

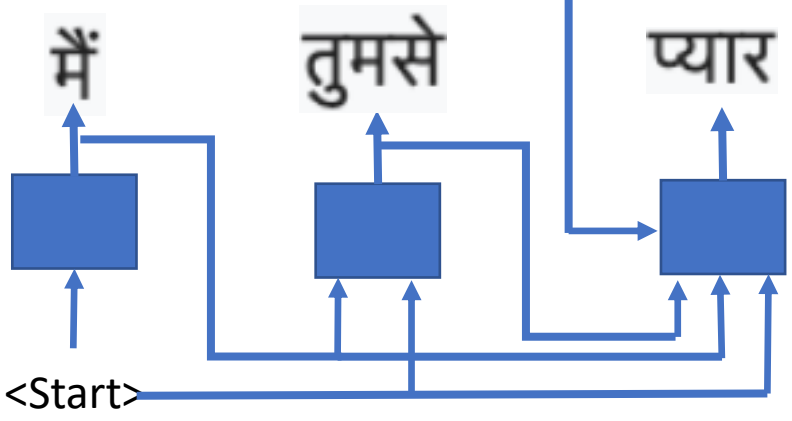

मैं
I
love
you

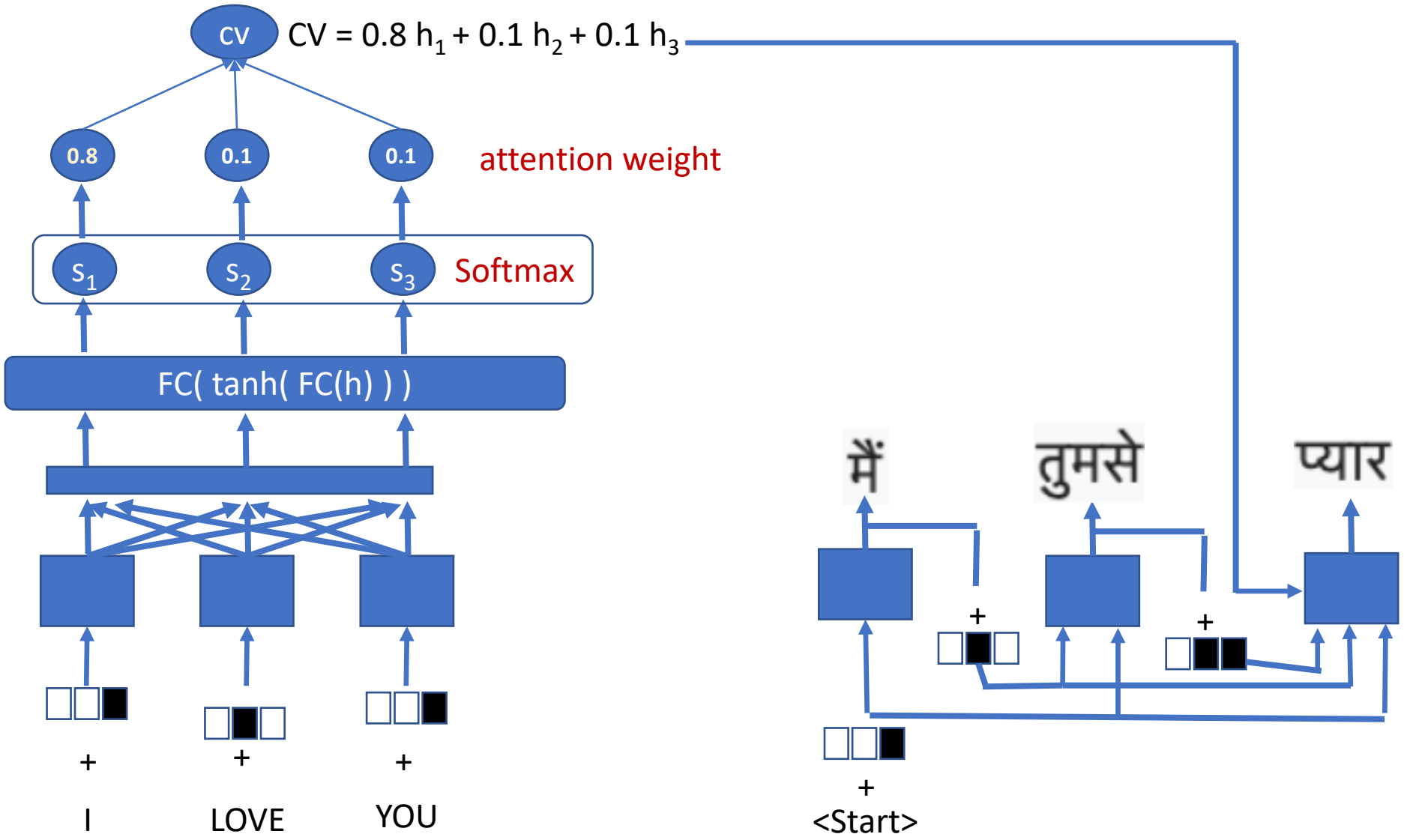
Attention


CV
CV = 0.8 h1 + 0.1 h2 + 0.1 h3
0.8
0.1
0.1
attention weight
S1
S2
S3
Softmax
FC(tanh(FC(h)))
h0
i1
i2
i3
I
LOVE
YOU
मैं
तुमसे
<Start>

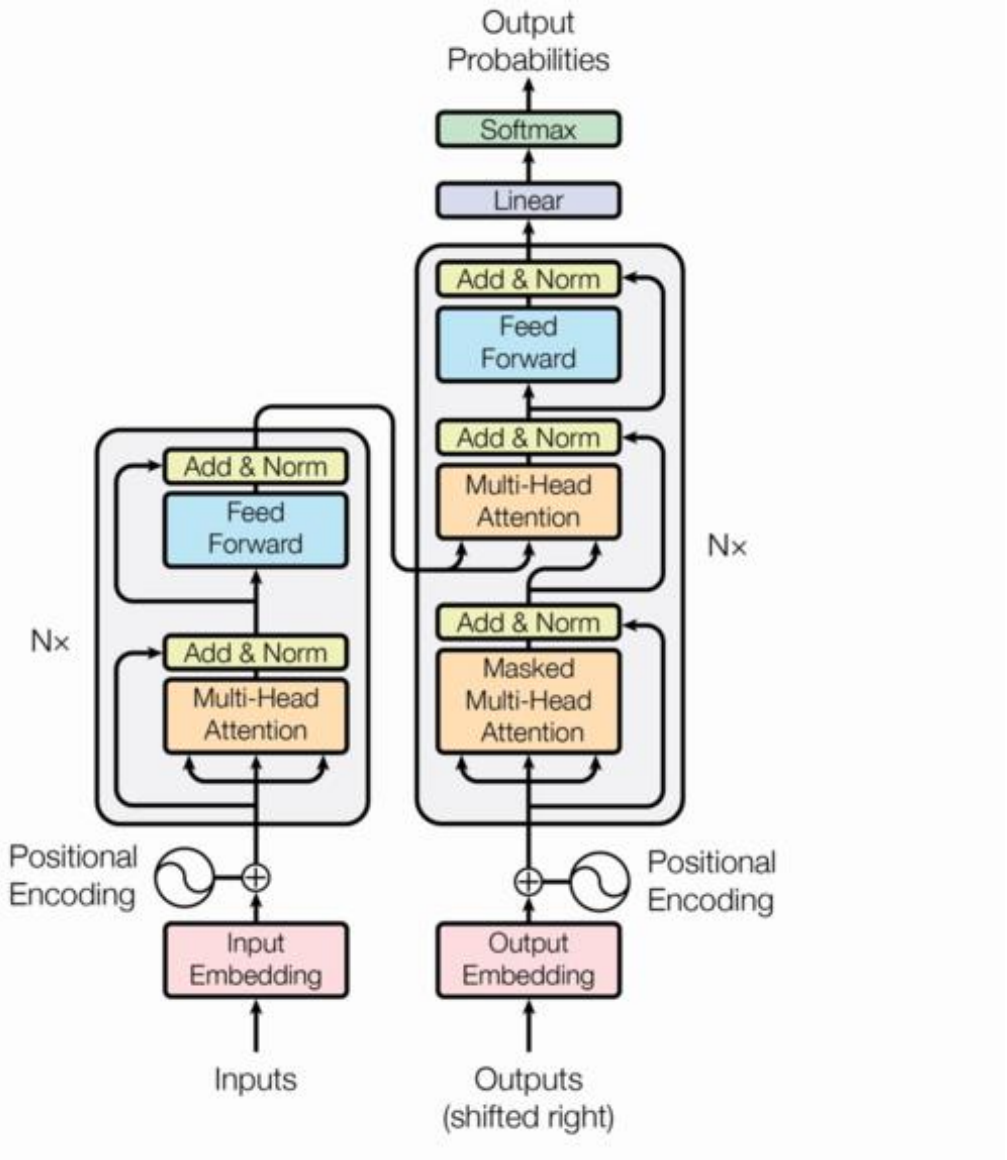
Attention


CV
CV = 0.8 h1 + 0.1 h2 + 0.1 h3
0.8
0.1
0.1
attention weight
S1
S2
S3
Softmax
FC(tanh(FC(h)))
h0
i1
i2
i3
I
LOVE
YOU

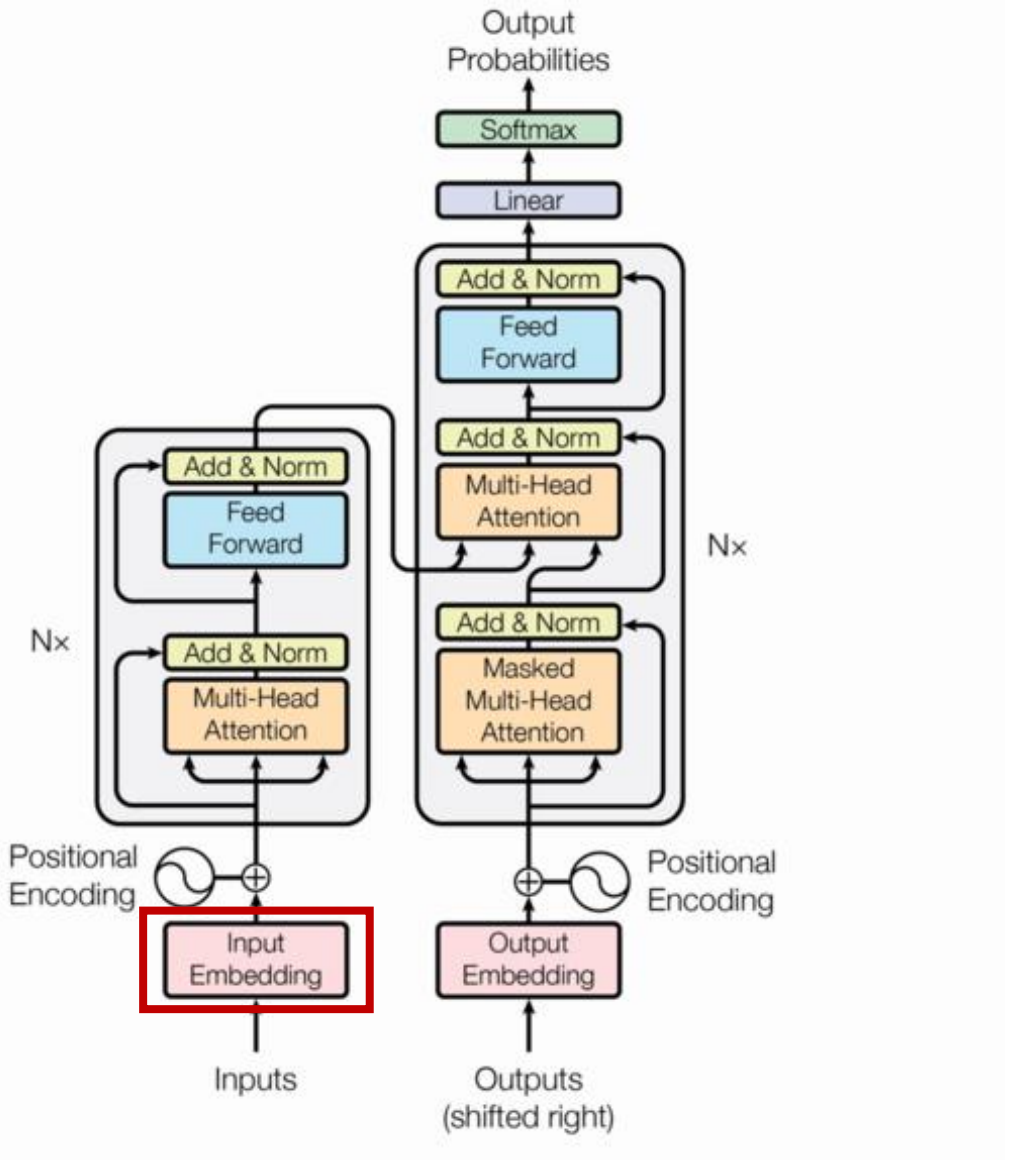

मैं
तुमसे
प्यार
<Start>


CV
CV = 0.8 h1 + 0.1 h2 + 0.1 h3
0.8
0.1
0.1
attention weight
S1
S2
S3
Softmax
FC(tanh(FC(h)))
+
+
+
I
LOVE
YOU
मैं
तुमसे
प्यार
+
+
+
<Start>

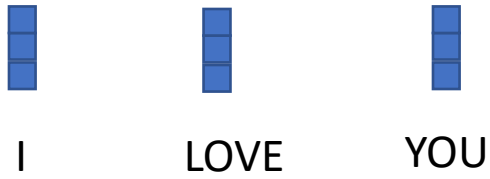
Transformer: Attention Is All You Need



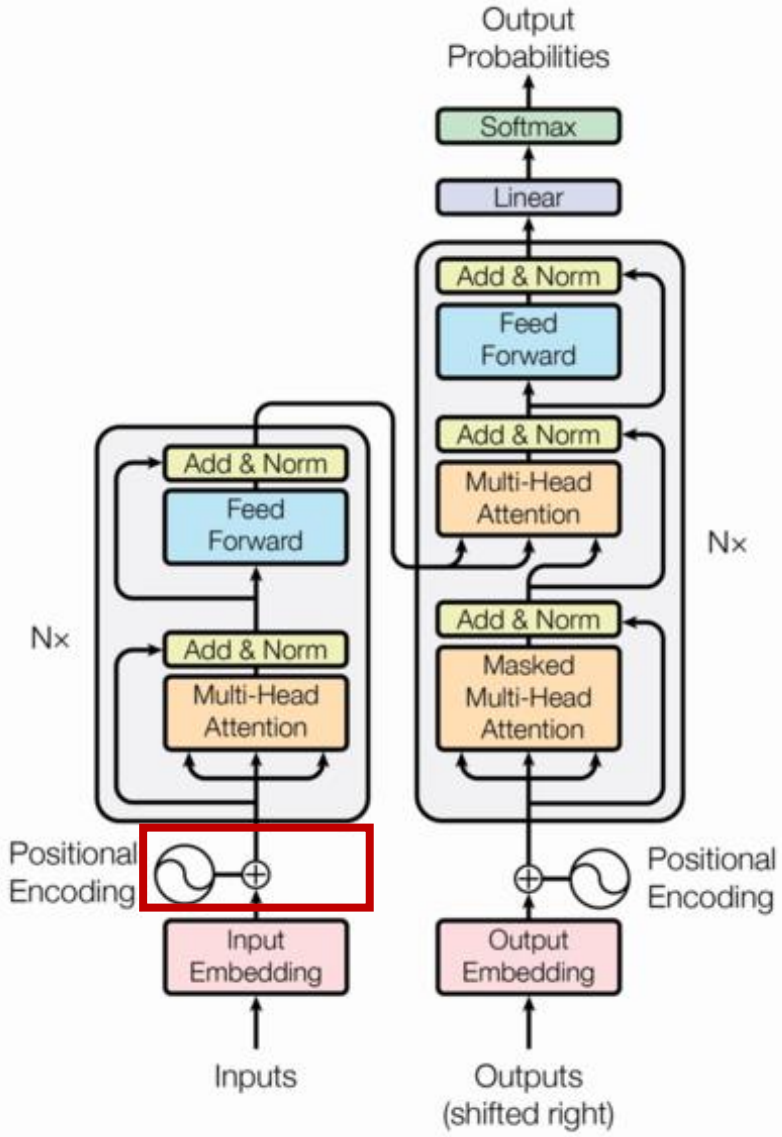
Transformer: Attention Is All You Need

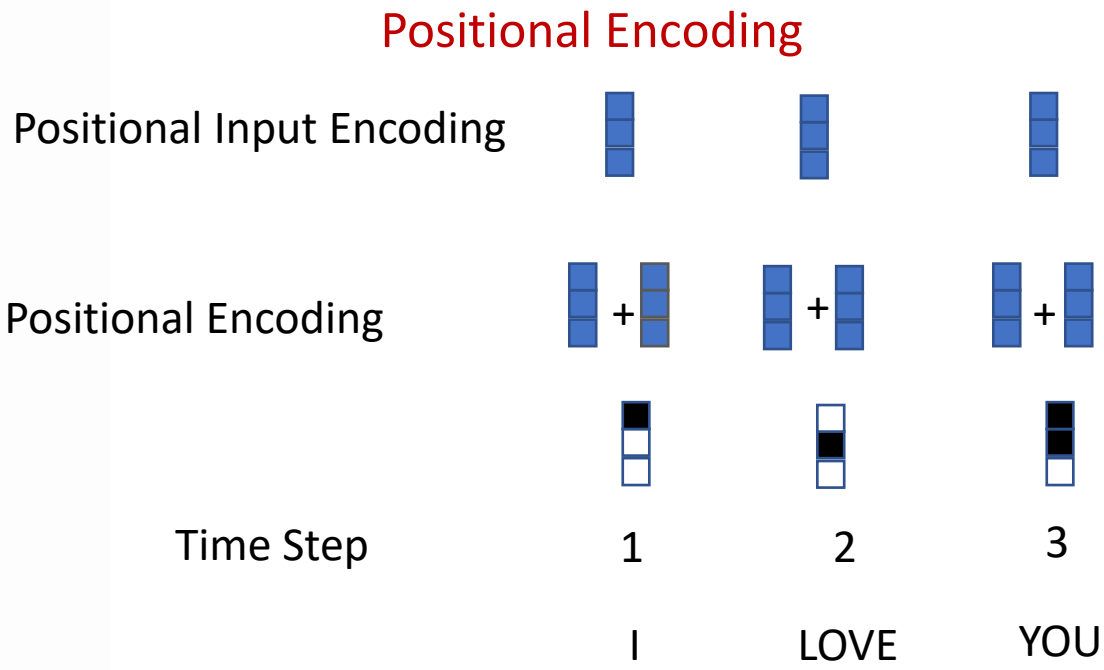


Word Embedding layer



Transformer: Attention Is All You Need

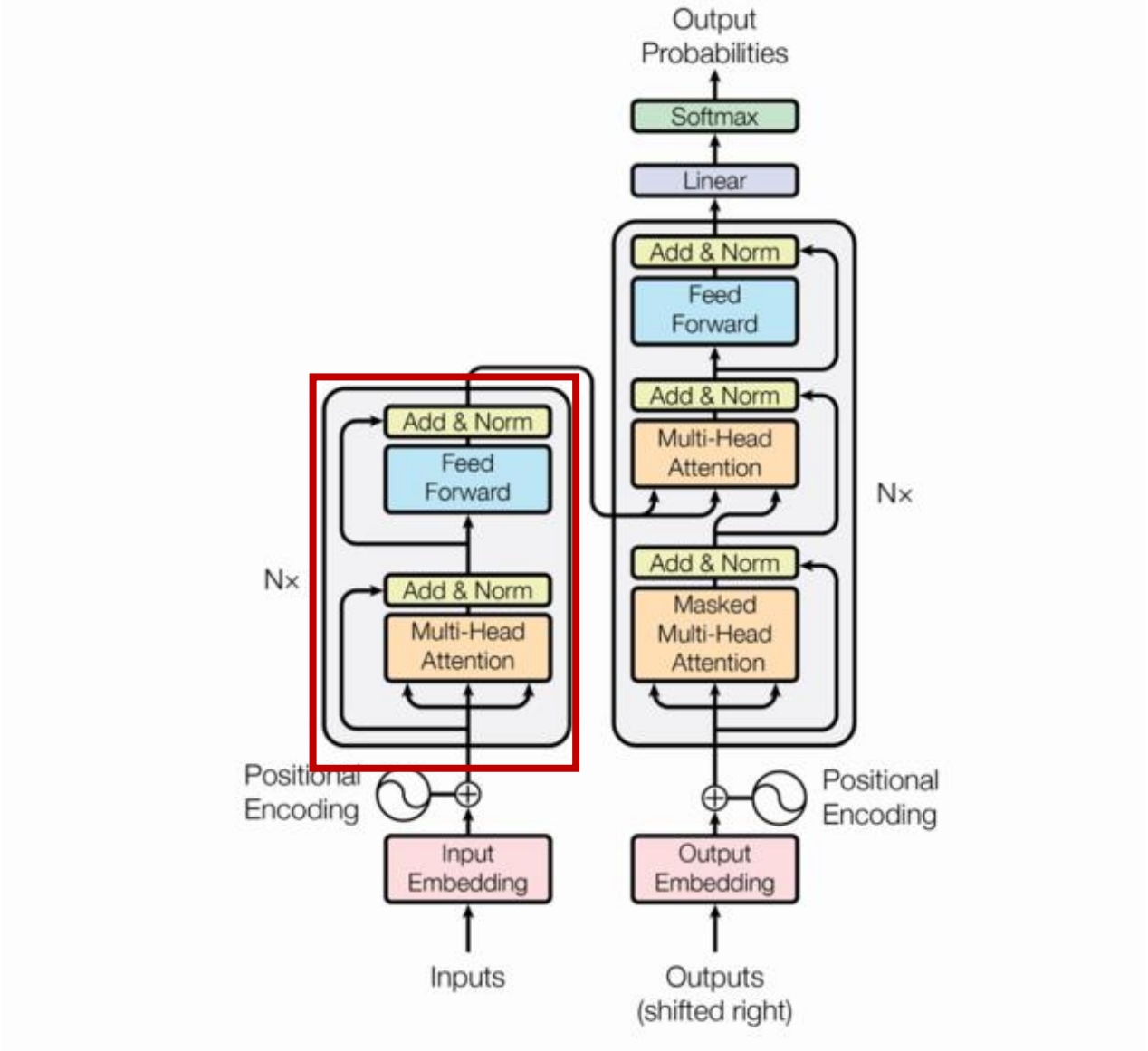


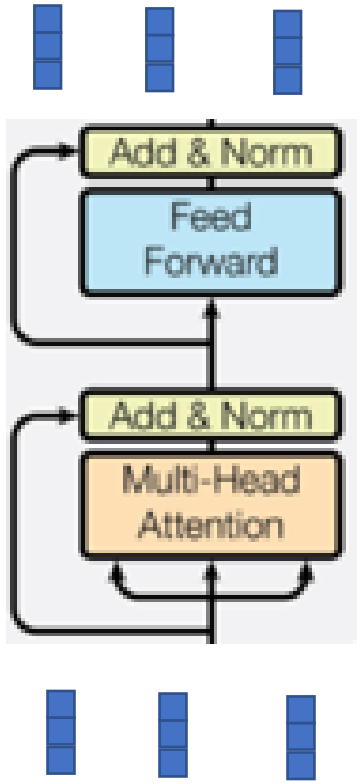


$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{2i/dmodel}})$$

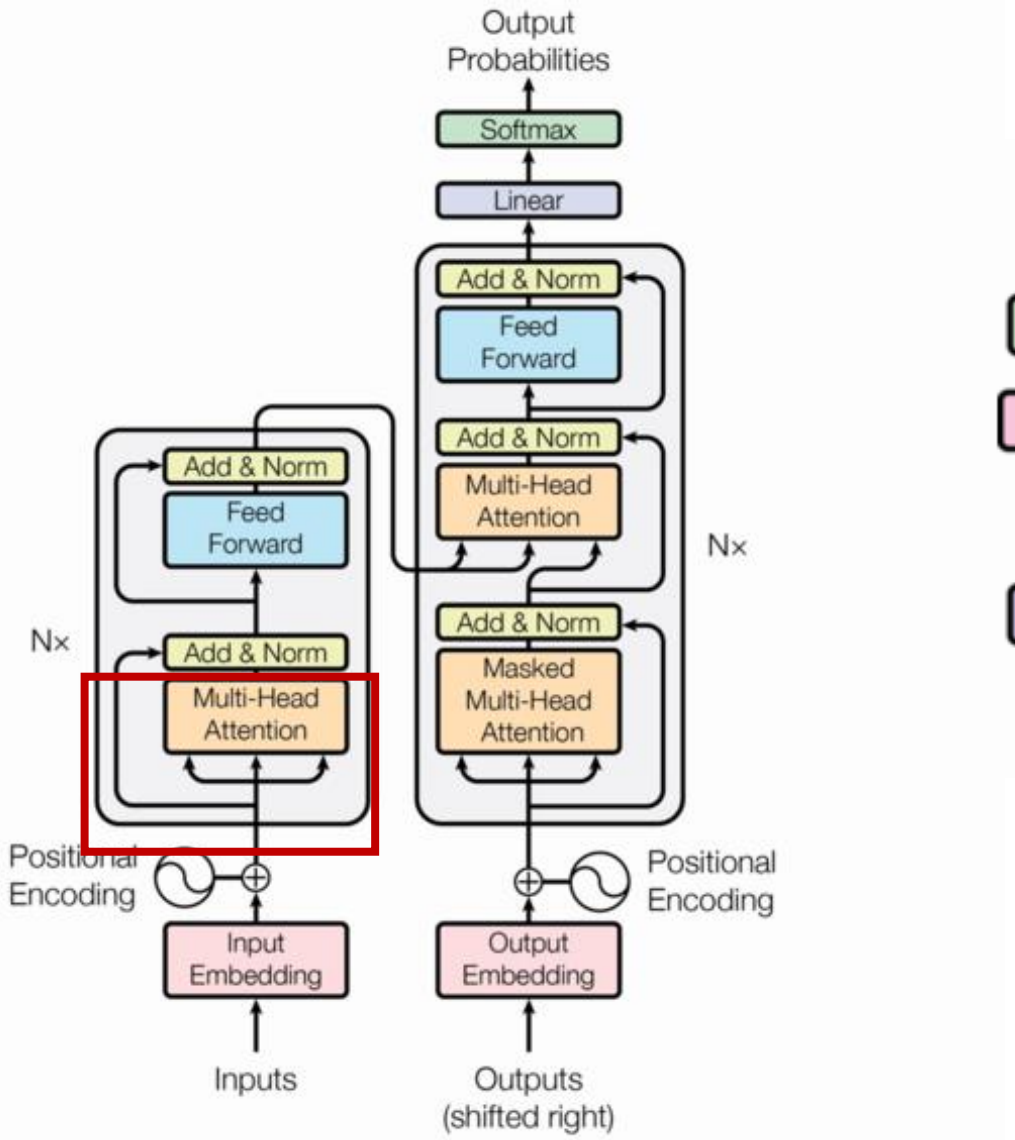
$$PE(pos, 2i) = sin(\frac{pos}{10000^{2i/dmodel}})$$

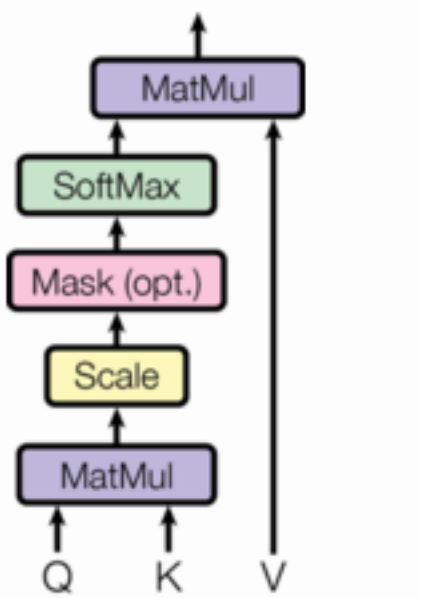
Transformer: Attention Is All You Need



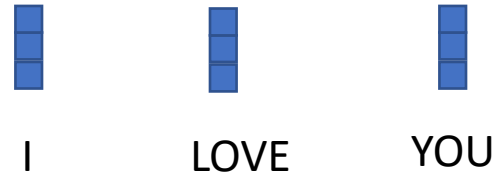


Transformer: Attention Is All You Need

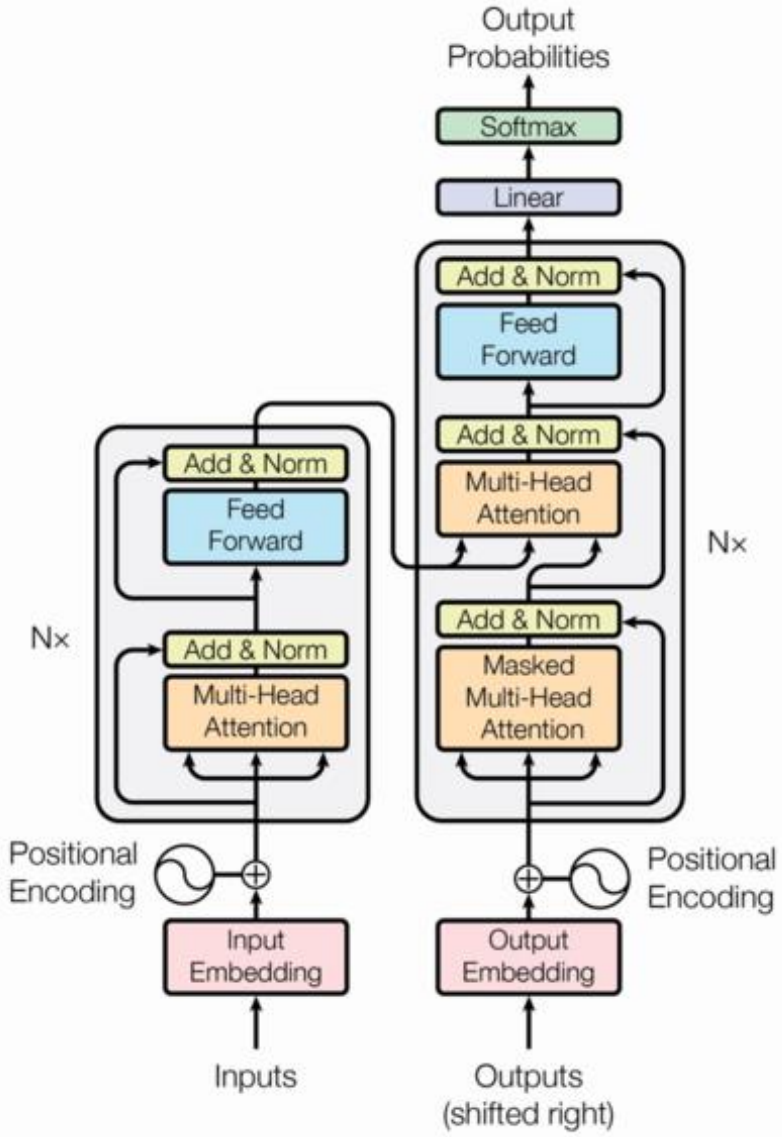


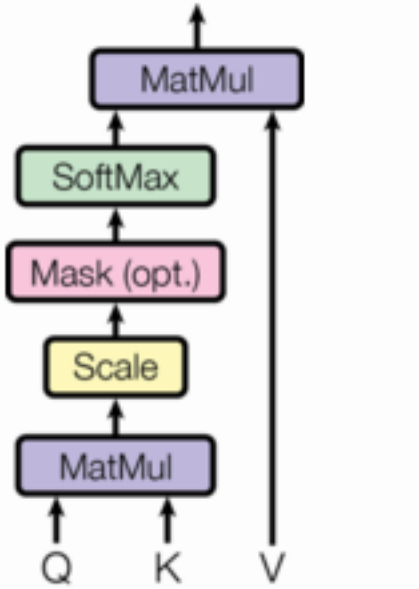


Self Attention

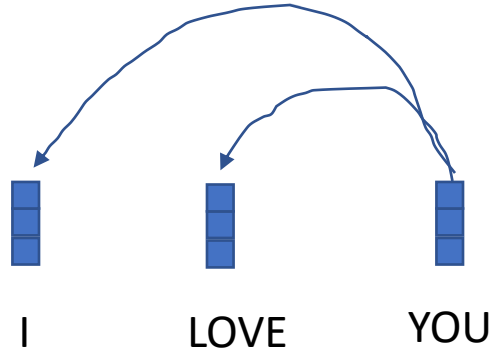


Transformer: Attention Is All You Need



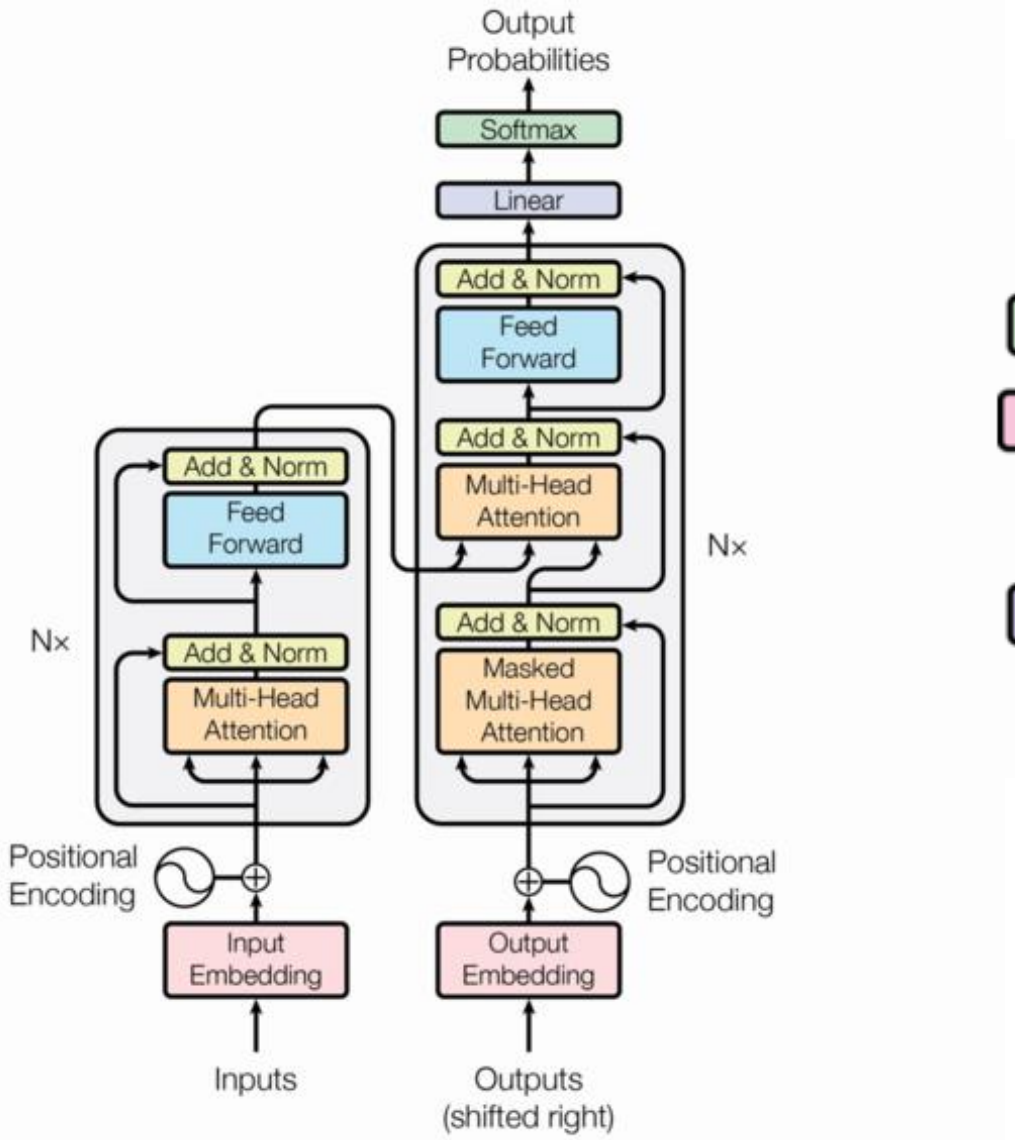


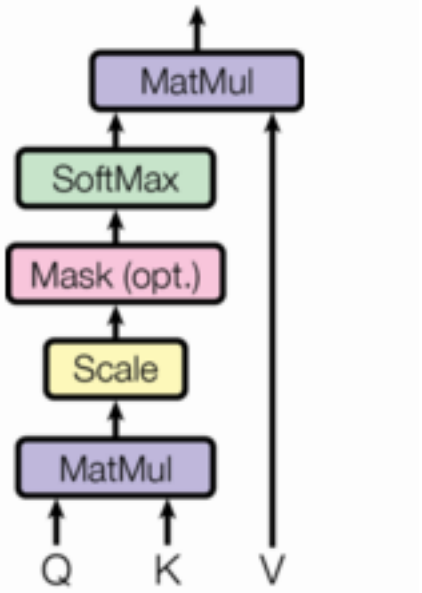
Self Attention



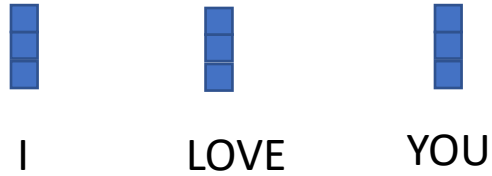
Softmax(Q . K)

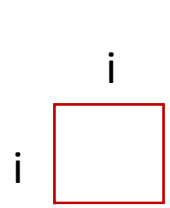
Transformer: Attention Is All You Need





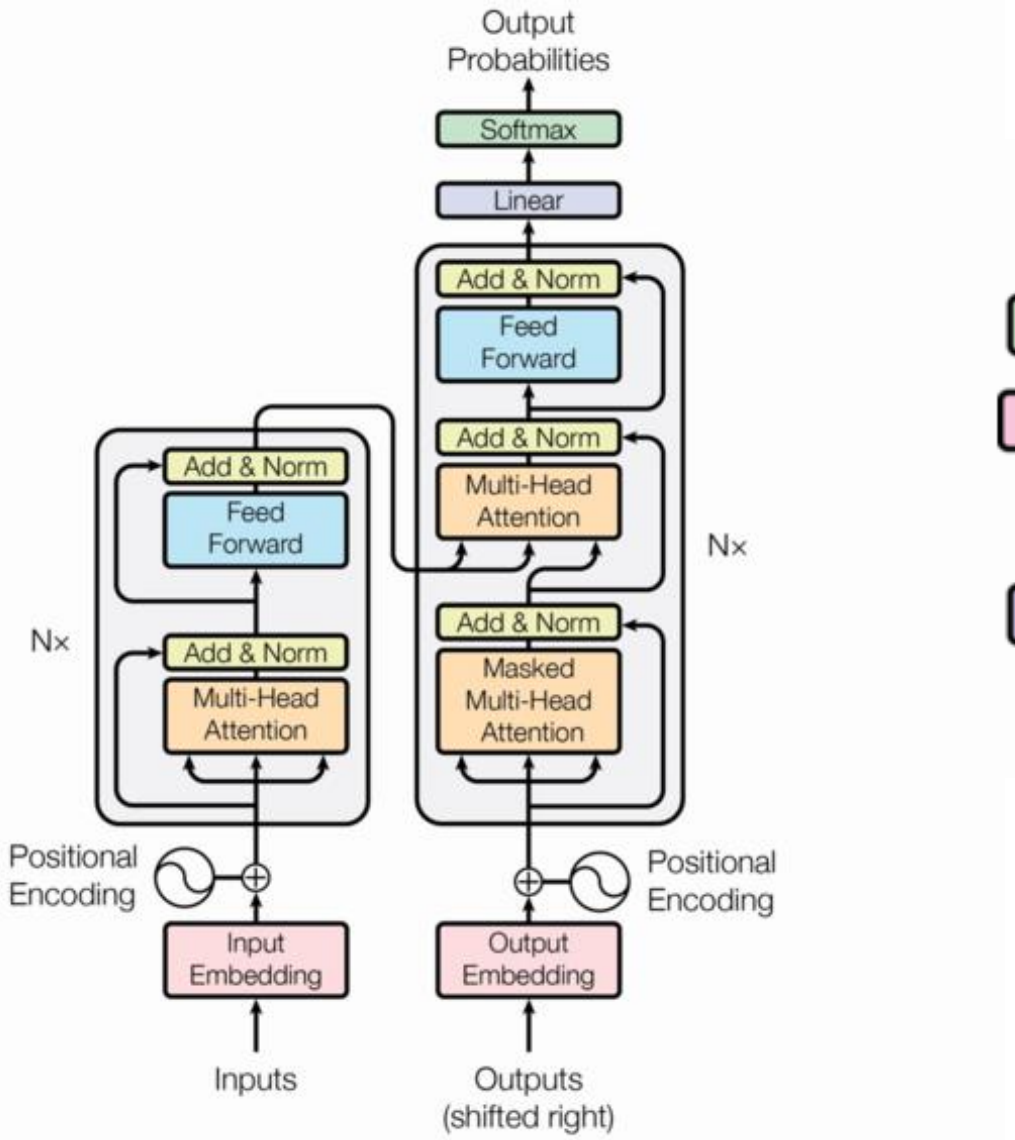
Self Attention

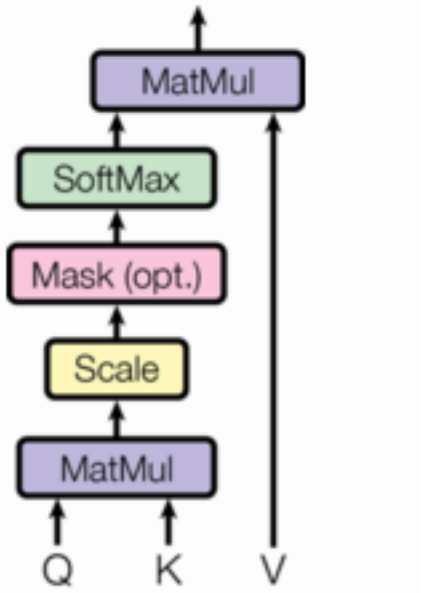




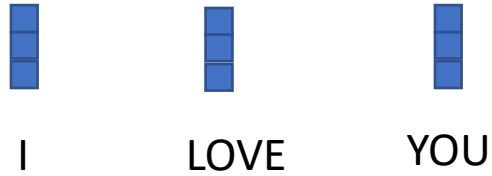
Softmax(I . I)

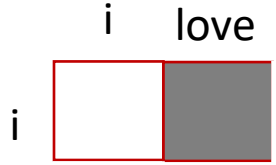
Transformer: Attention Is All You Need





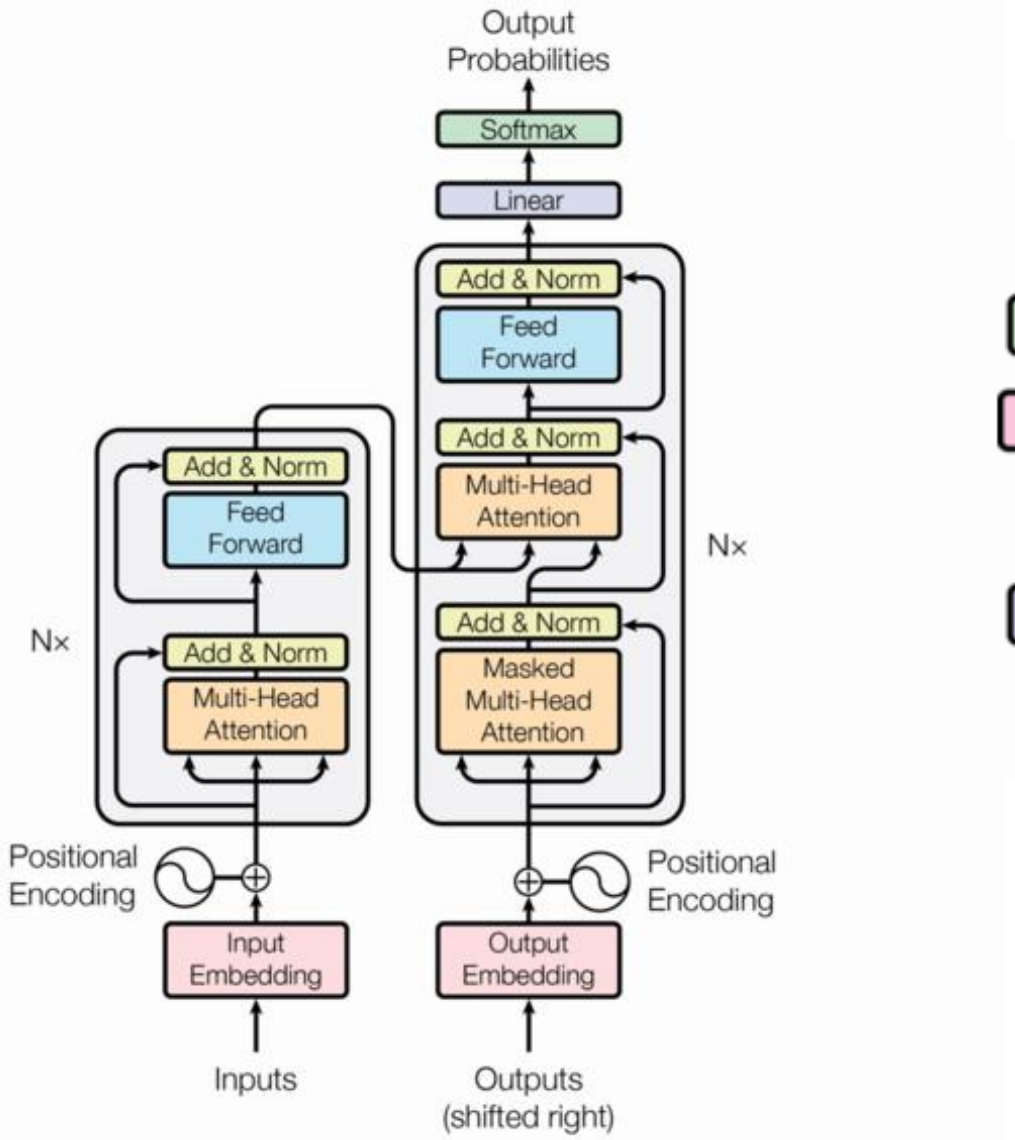
Self Attention

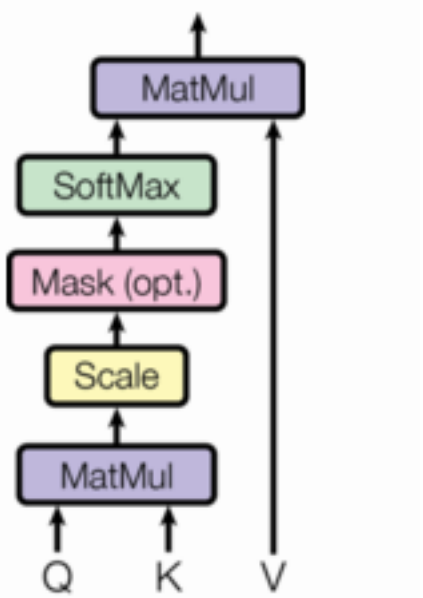




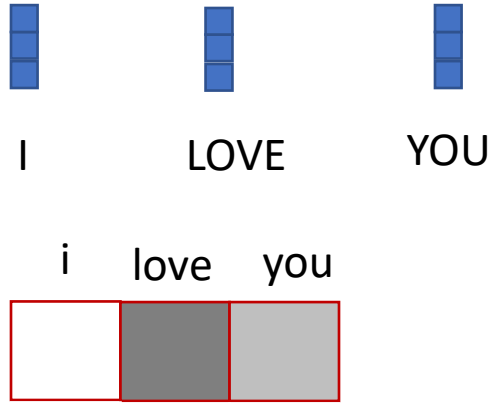
Softmax(I . LOVE)

Transformer: Attention Is All You Need



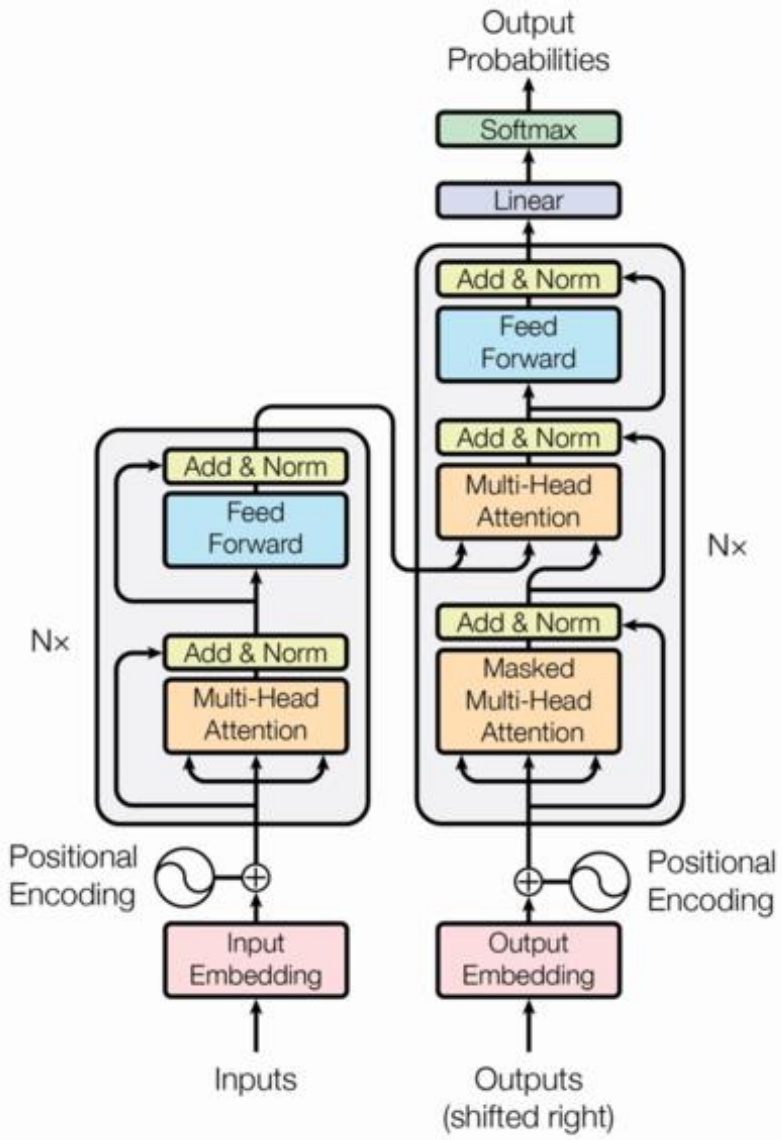


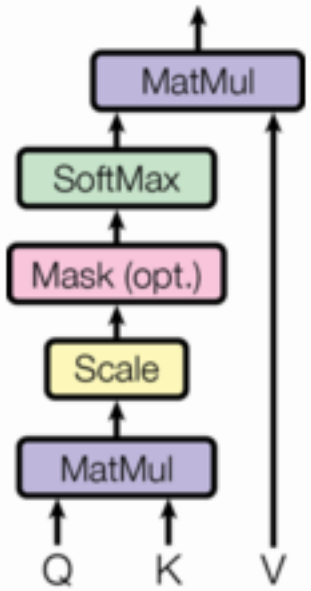
Self Attention



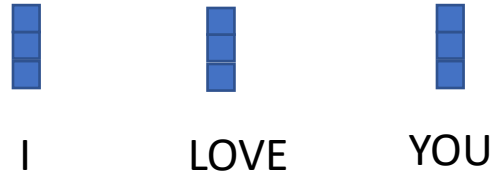
Softmax(I . YOU)

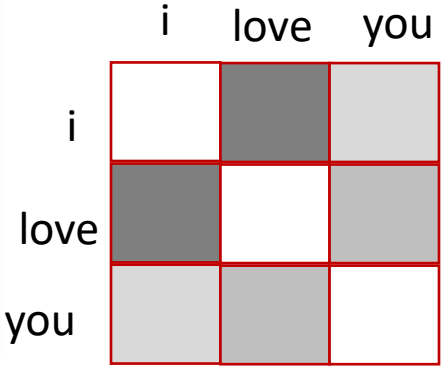
Transformer: Attention Is All You Need



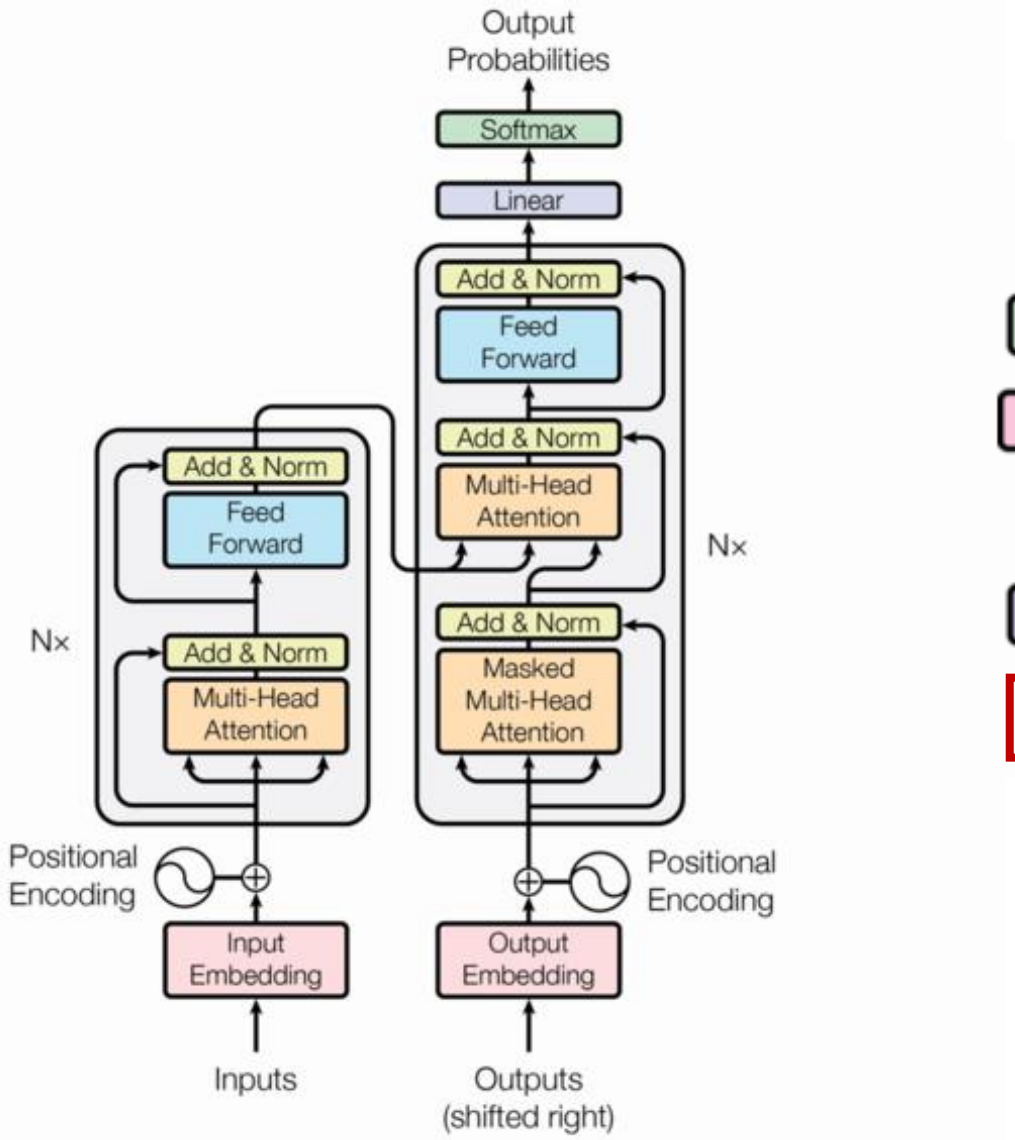


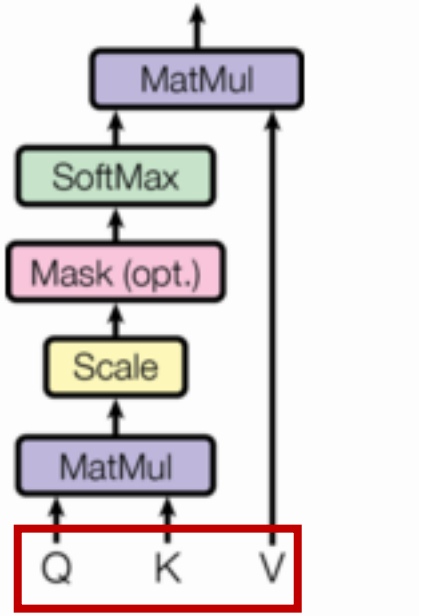
Self Attention



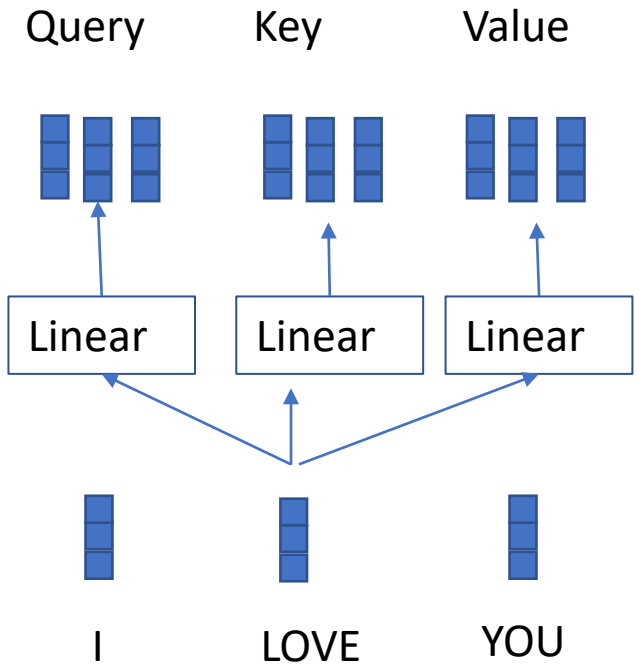


Transformer: Attention Is All You Need

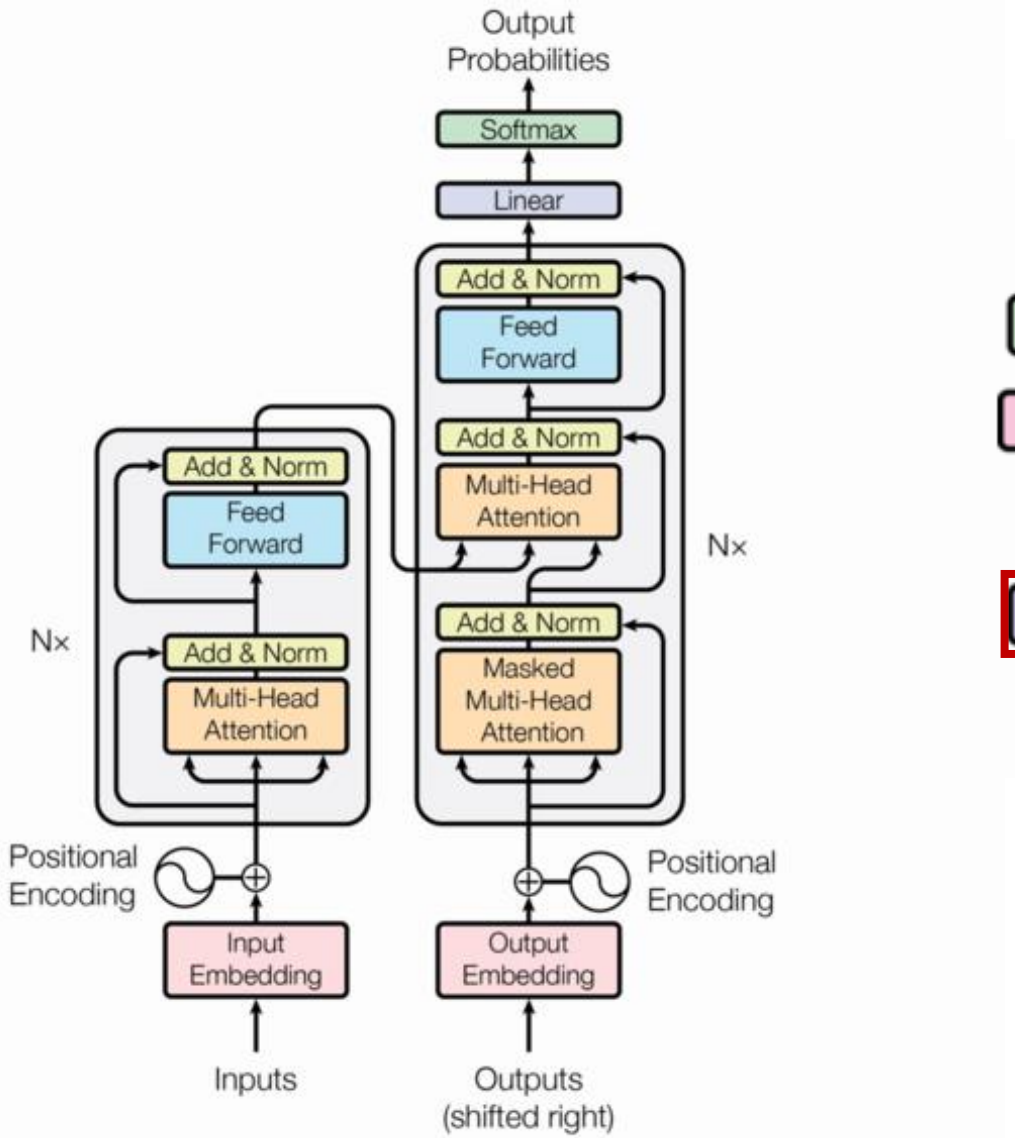


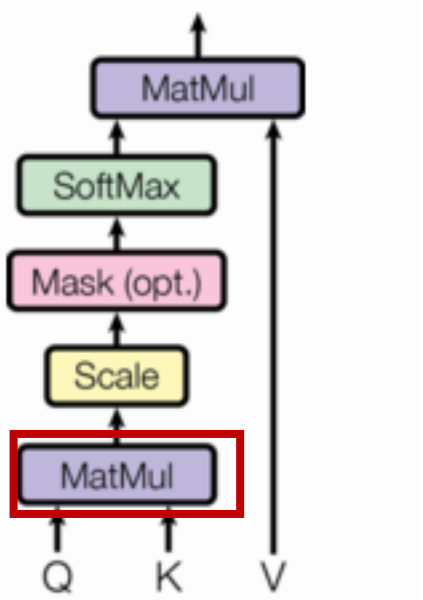


Self Attention



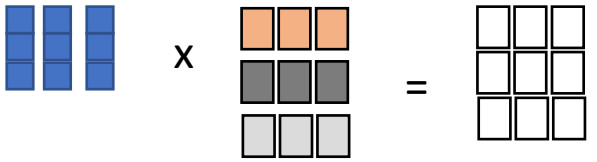
Transformer: Attention Is All You Need



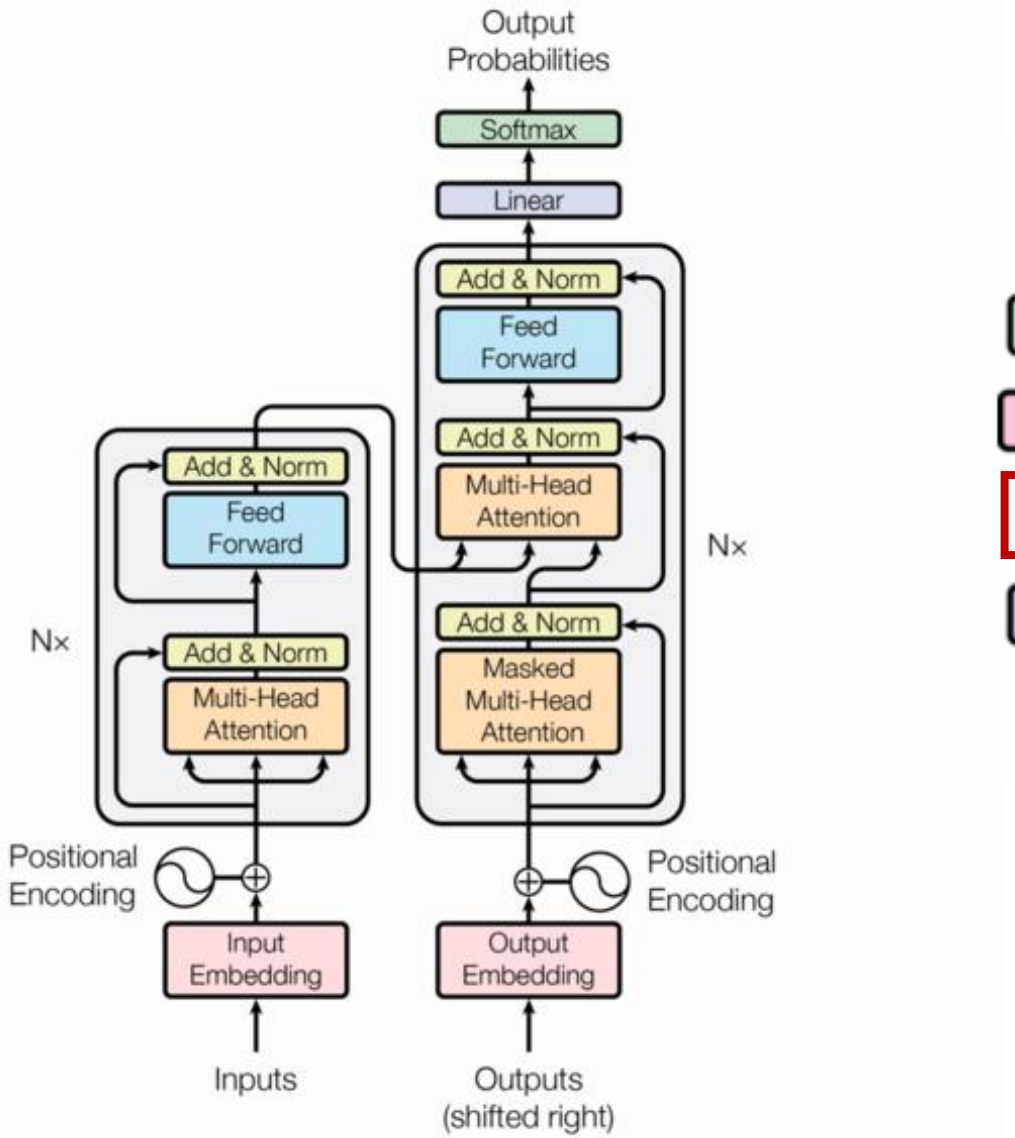


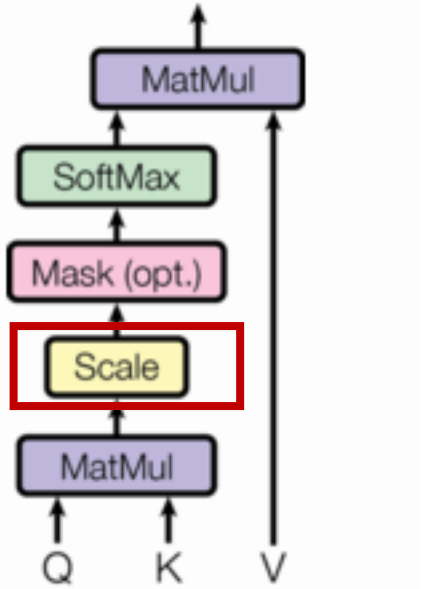
Self Attention

Query Key Score



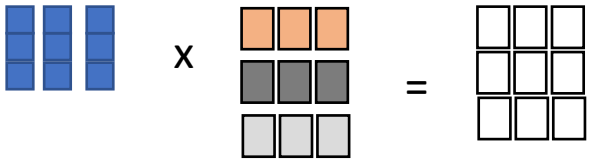
Transformer: Attention Is All You Need



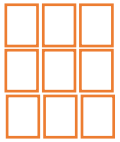


Self Attention

Query Key Score

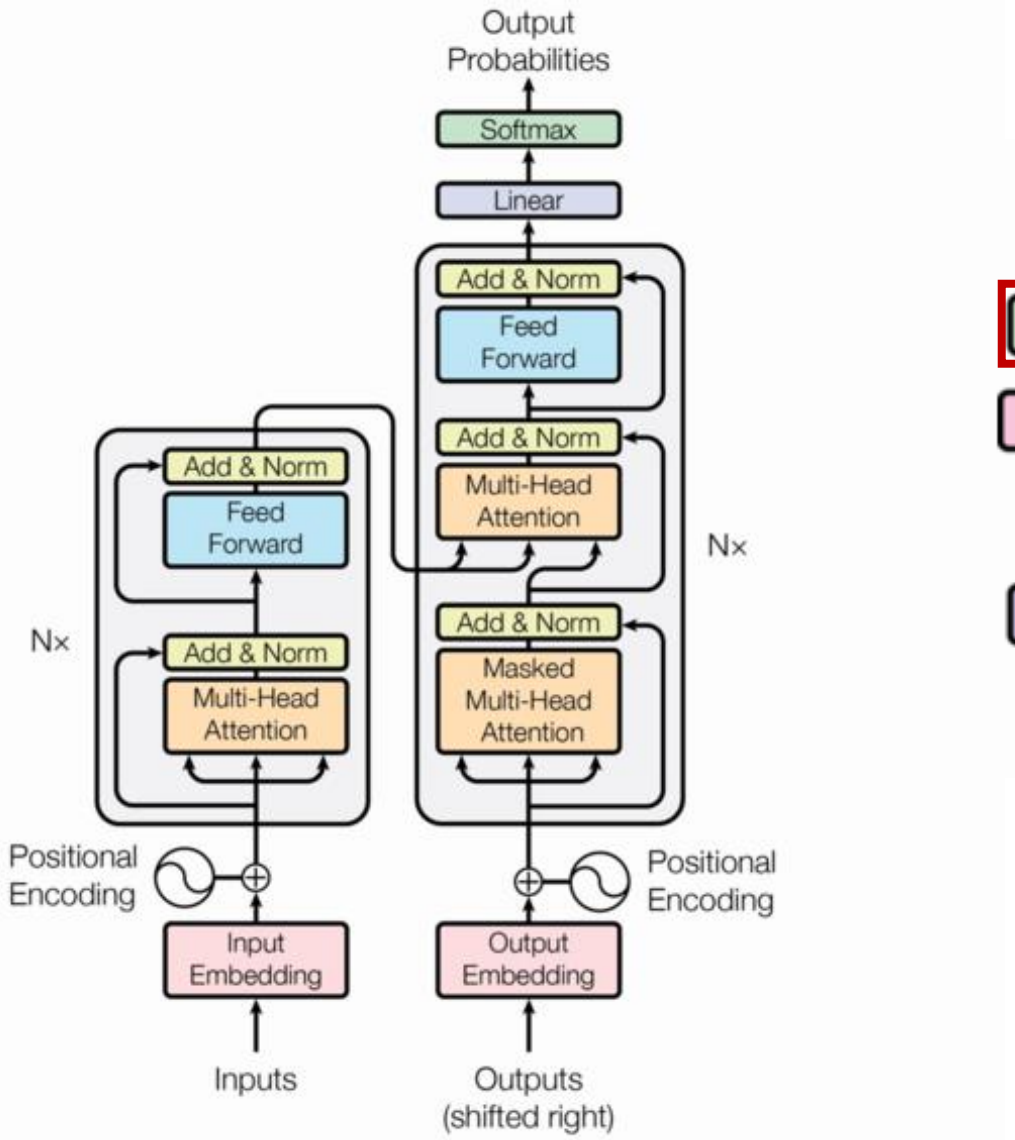


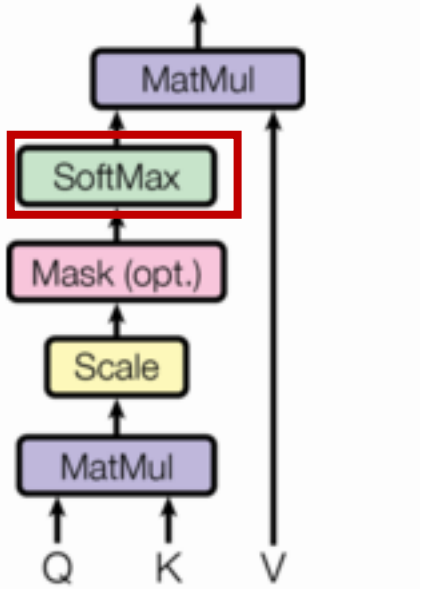
$$\frac{\square\square\square}{\square\square\square} = \frac{}{\sqrt{d_k}}$$



Scale

Transformer: Attention Is All You Need





Self Attention

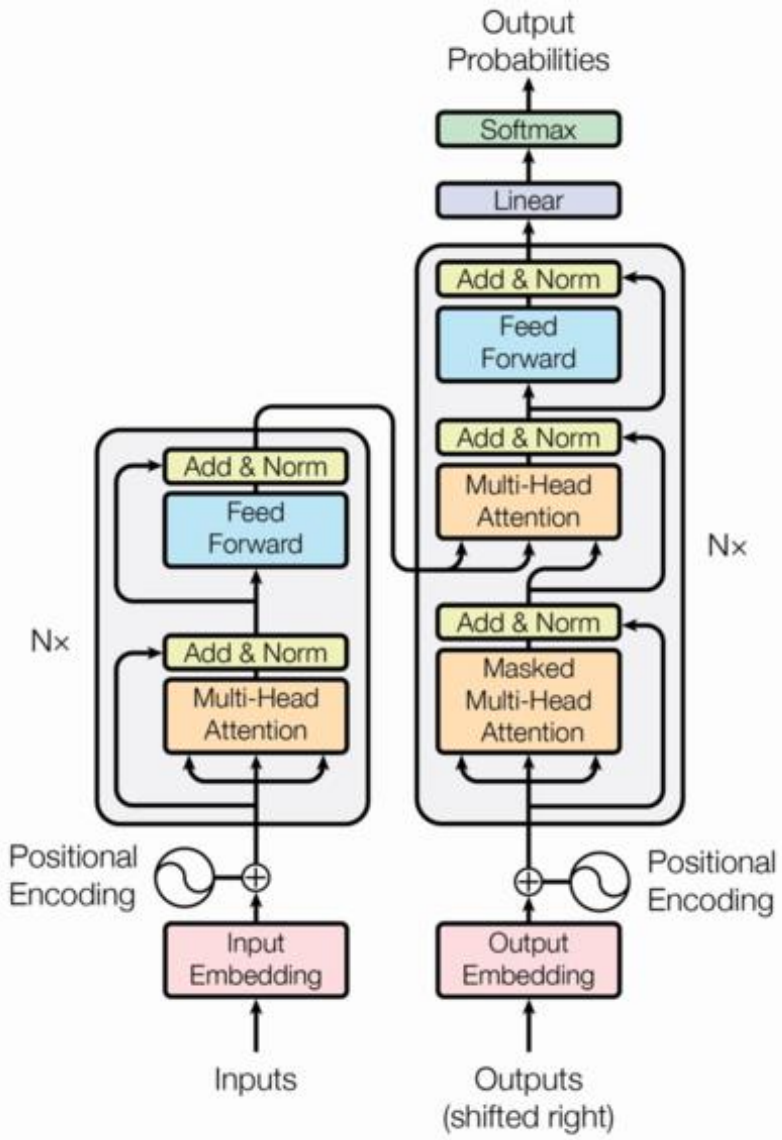
$$\frac{\square\square\square}{\sqrt{d_k}} = \square\square\square$$

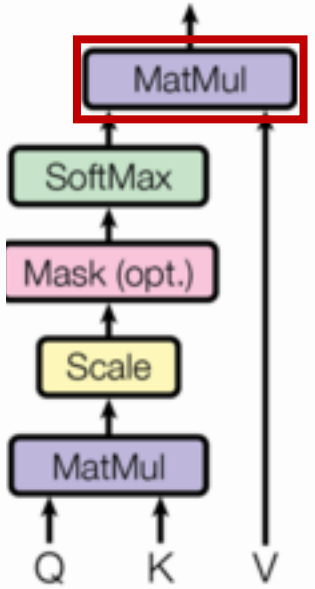
Scale

$$\text{Softmax}\left(\frac{\square\square\square}{\text{Scale}}\right) = \square\square\square$$

Attention Matrix

Transformer: Attention Is All You Need





Self Attention

Softmax ( ) =

Scale

Attention Matrix

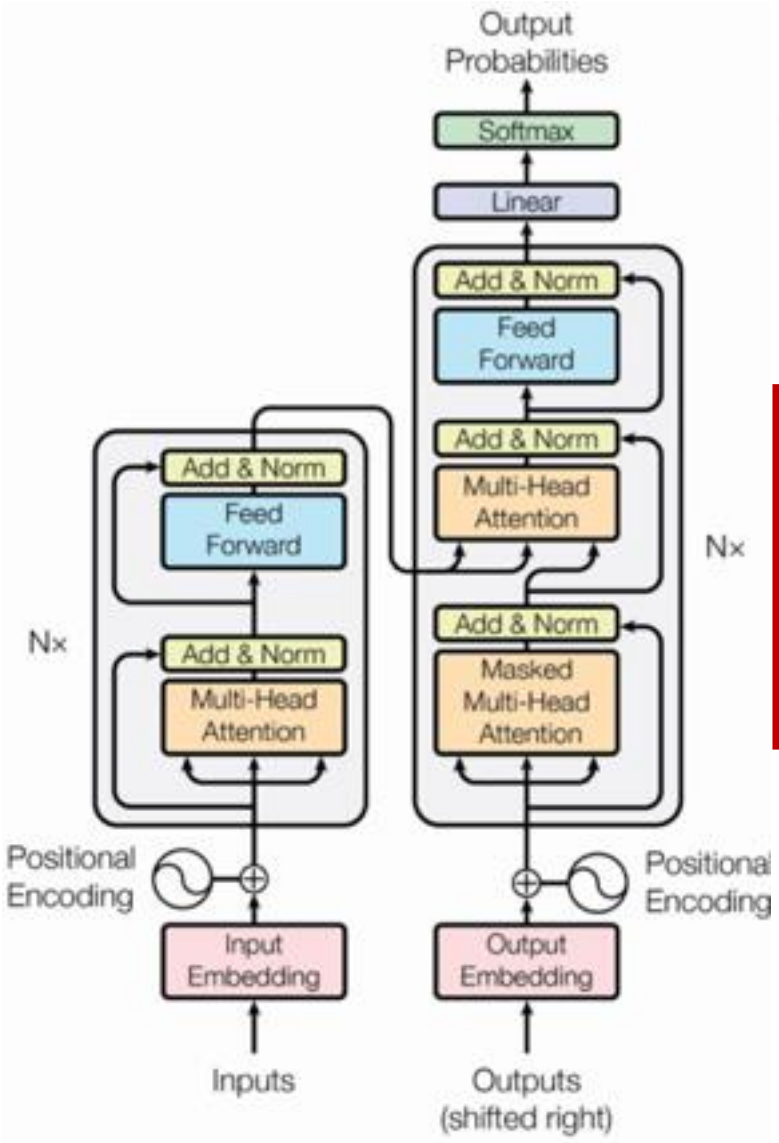
x =

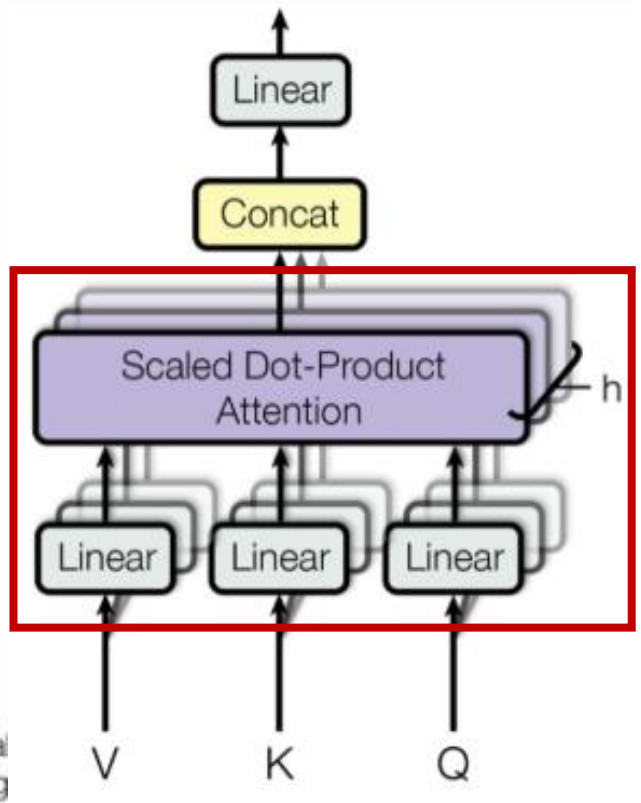
Attention Matrix

Value

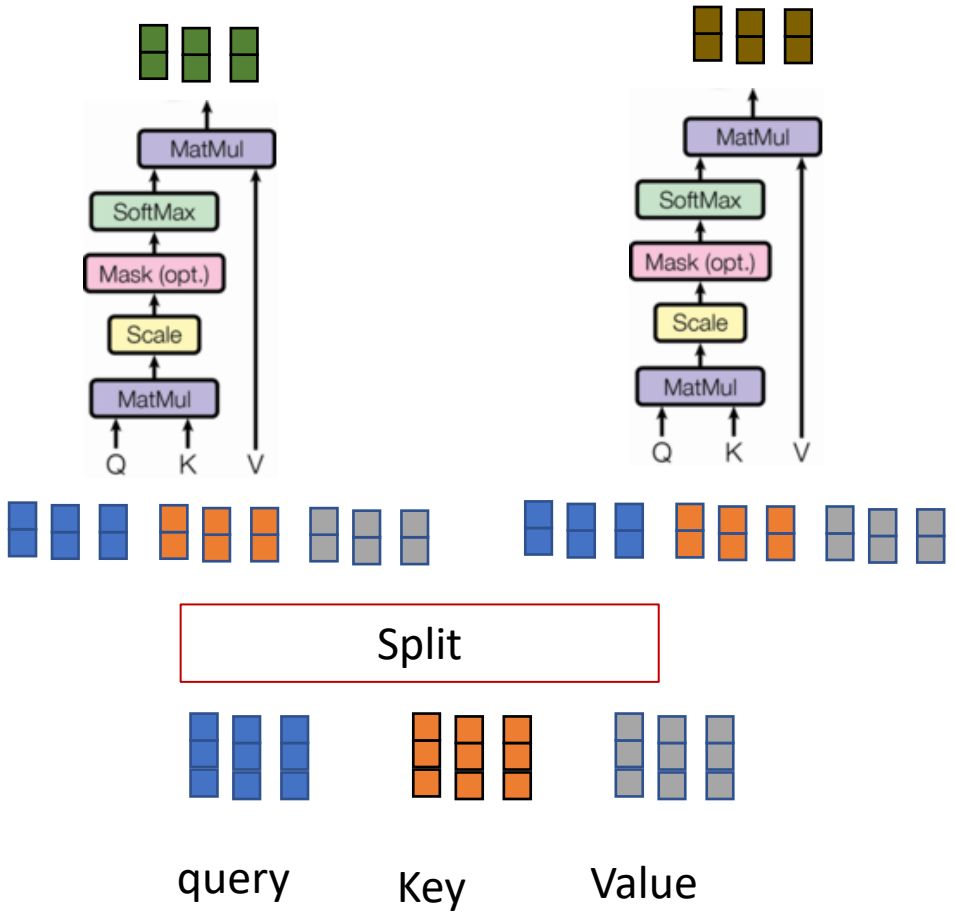
Transformed Input

Transformer: Attention Is All You Need

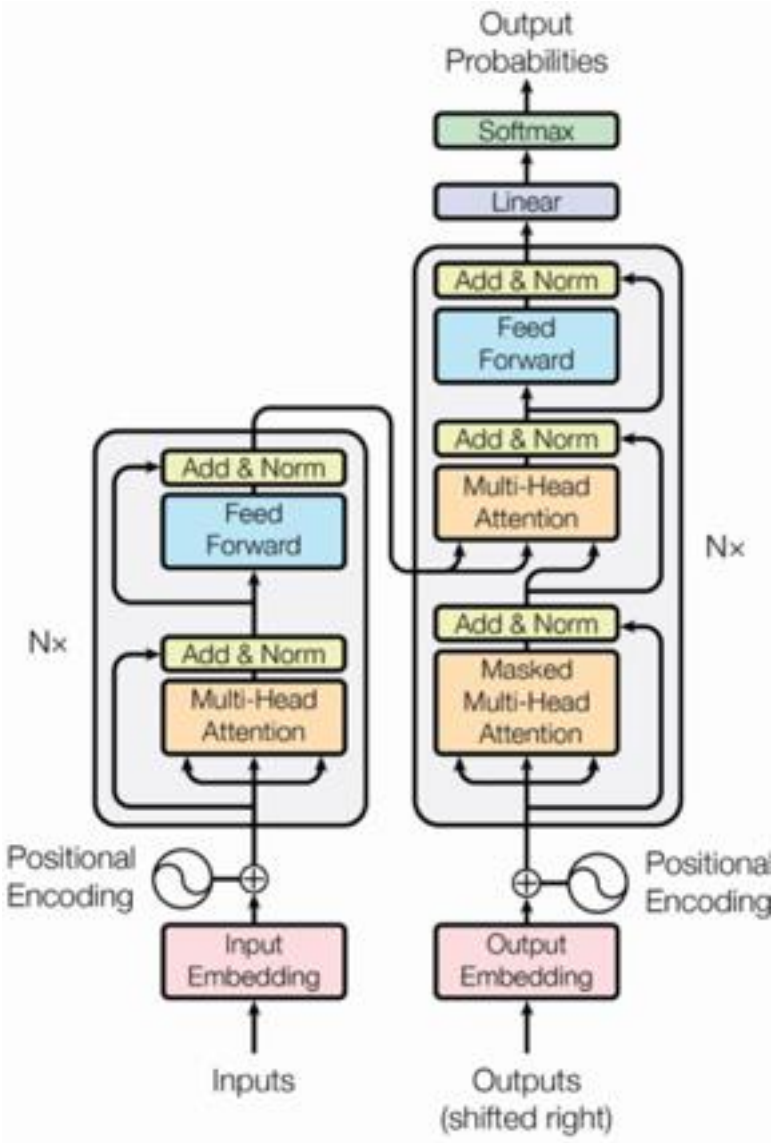


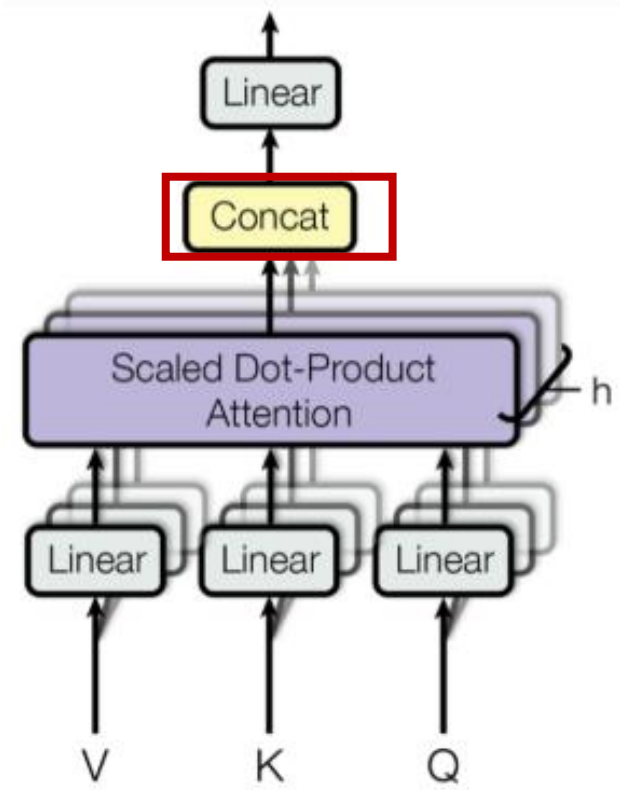


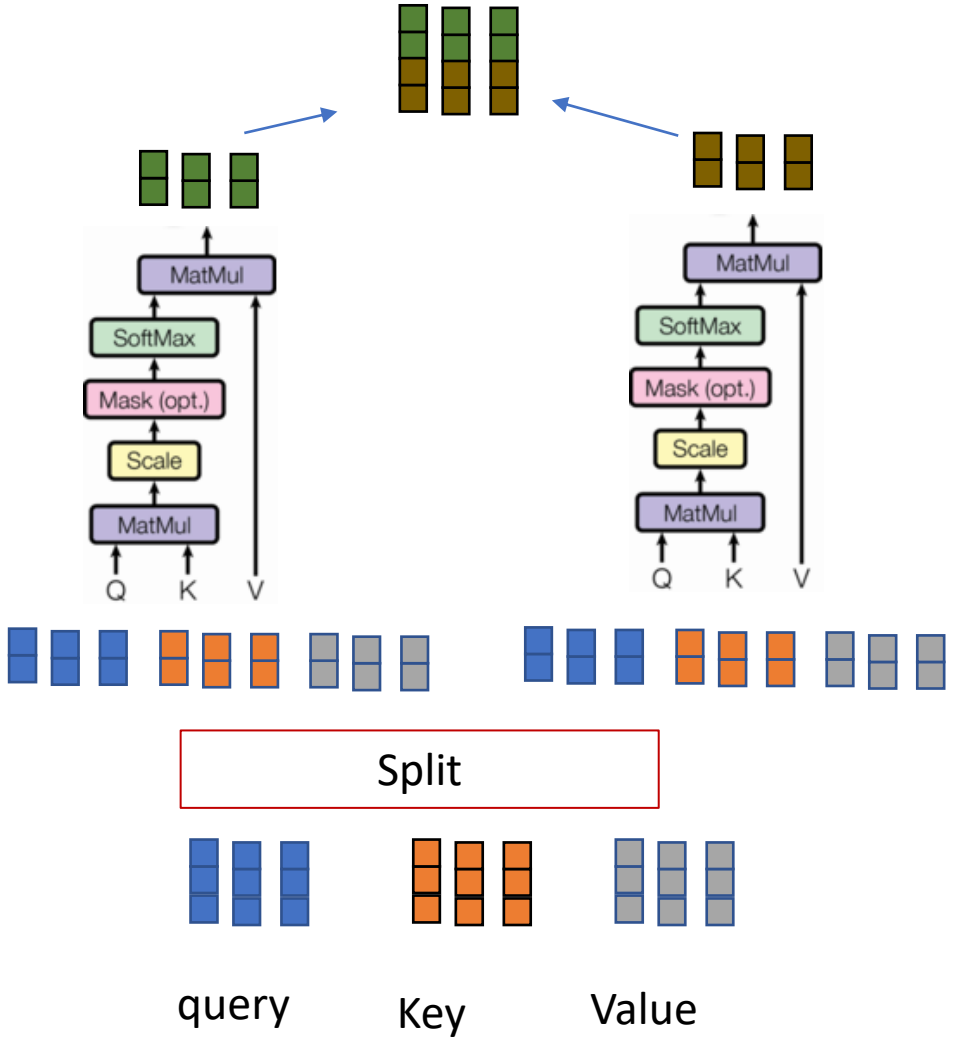
Multi Head Attention

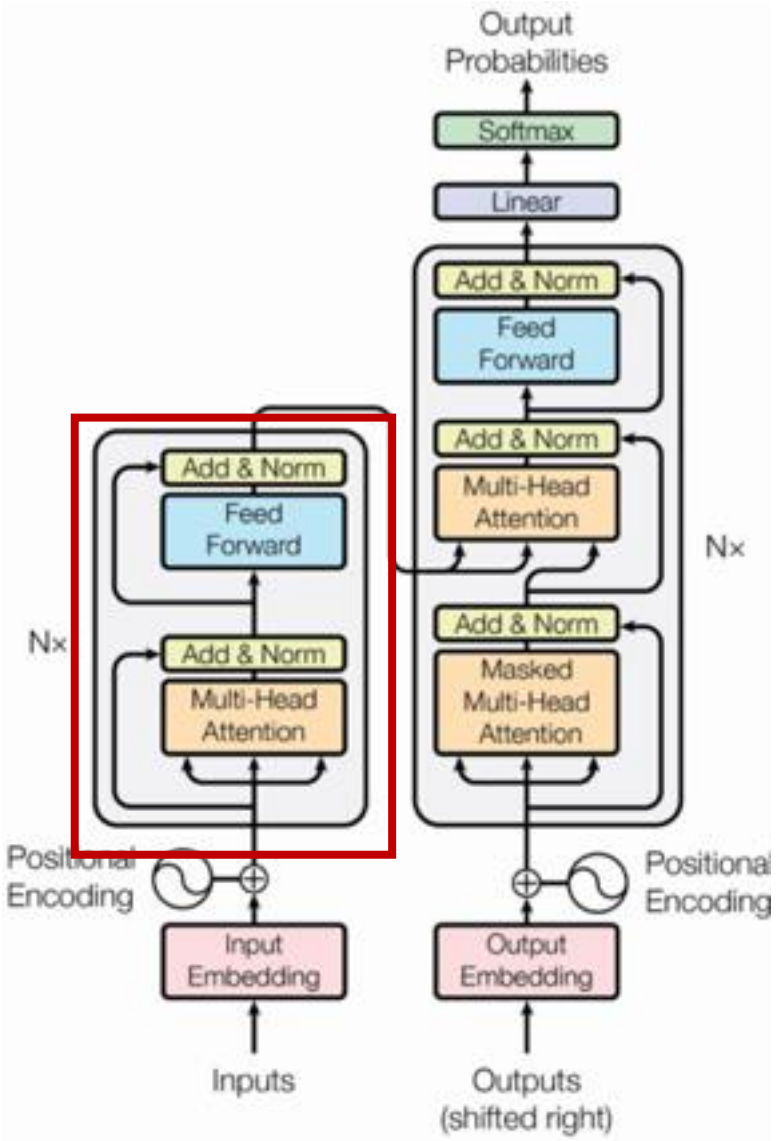


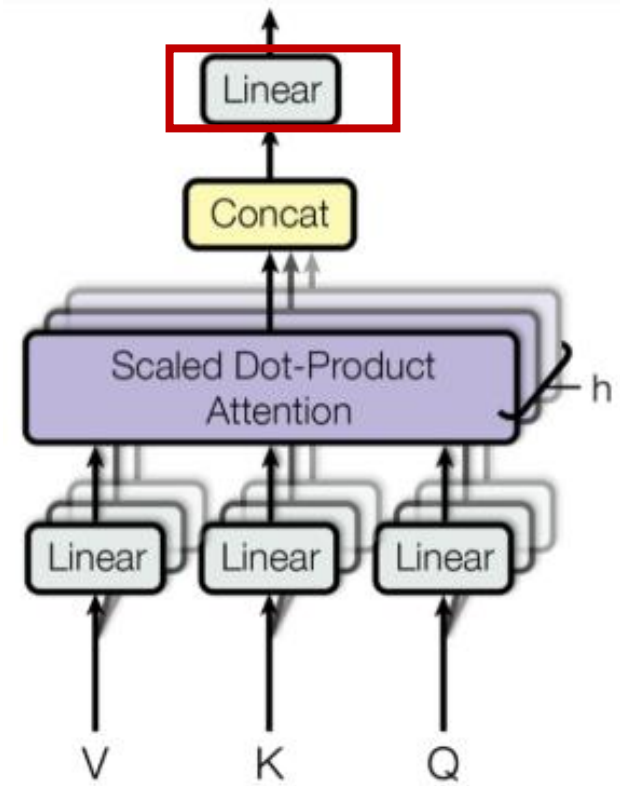
Transformer: Attention Is All You Need

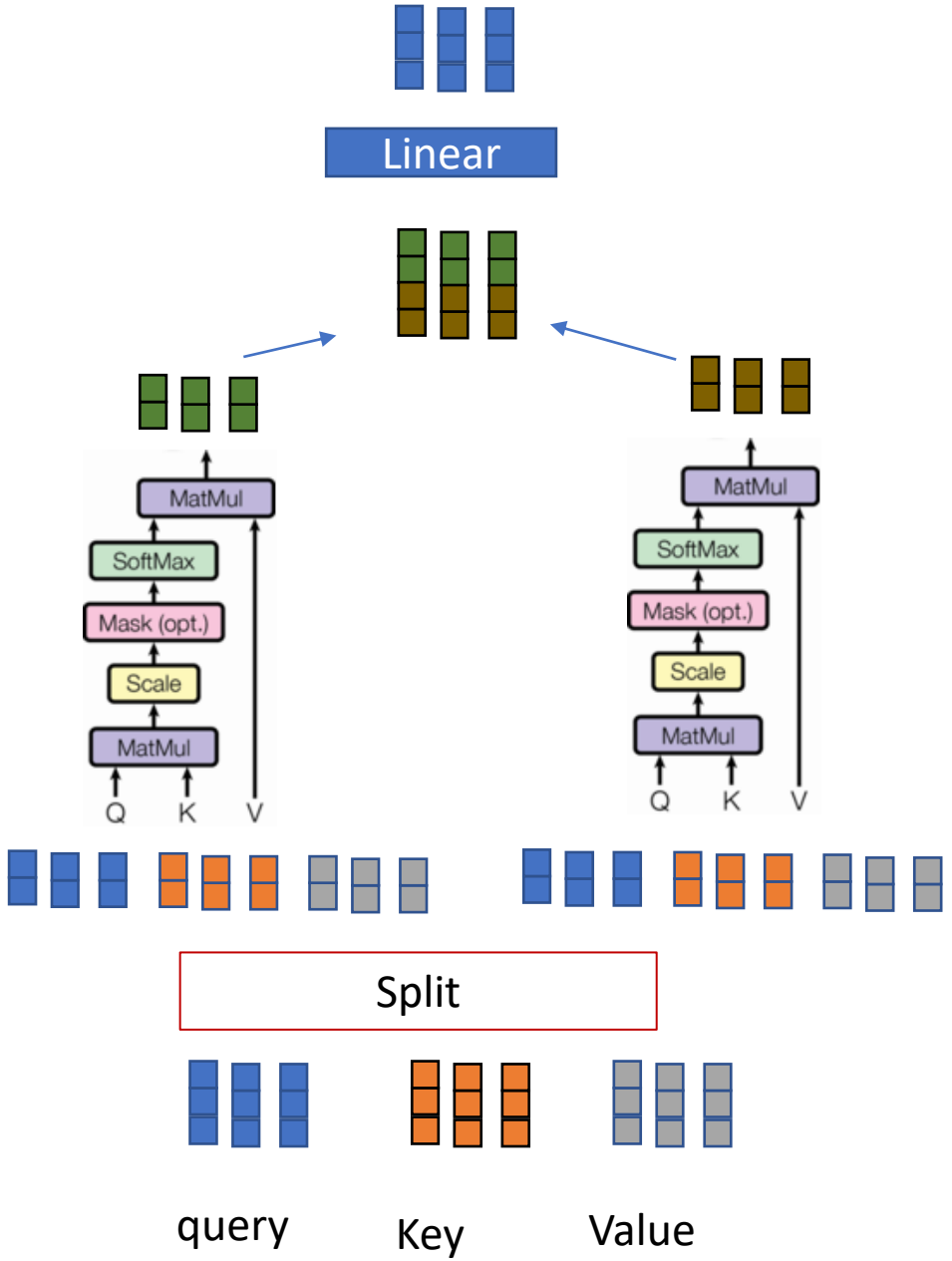




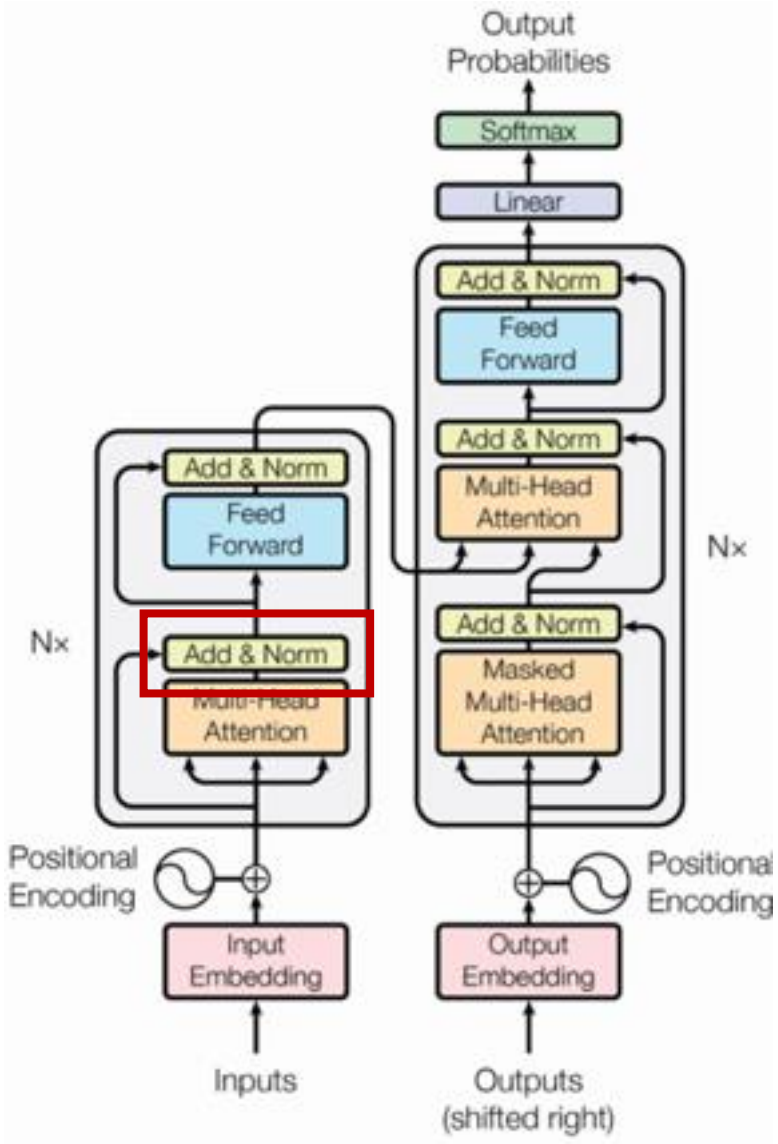


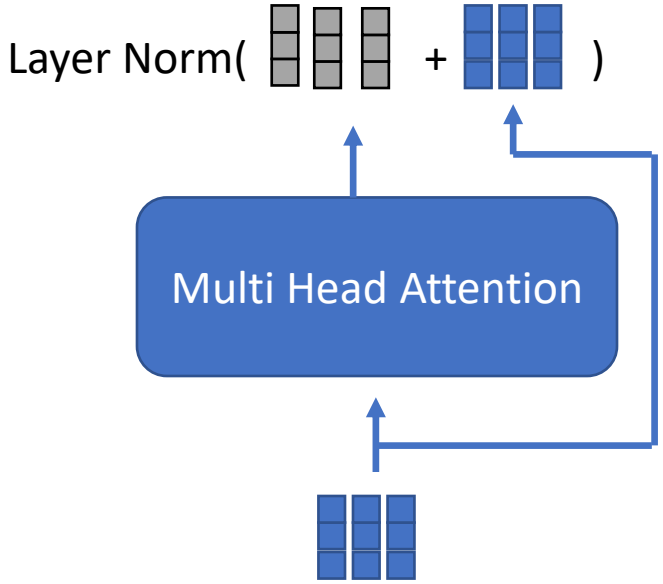

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)


Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

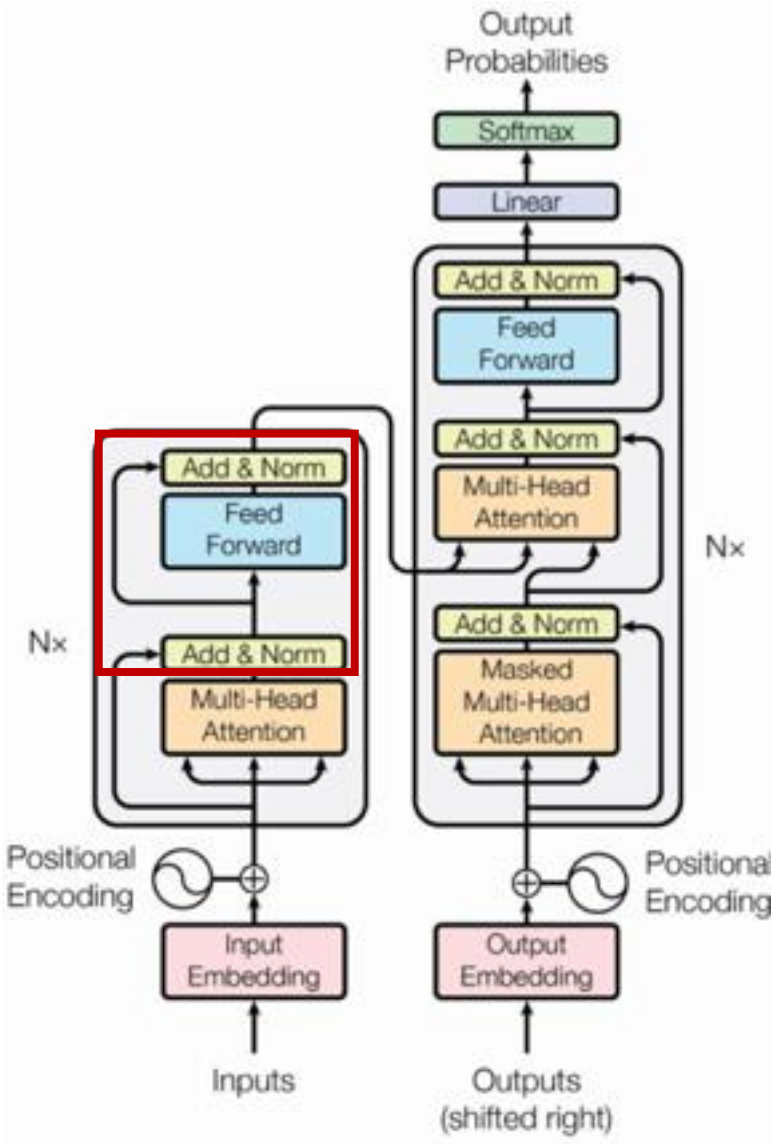

Linear
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Split
query
Key
Value

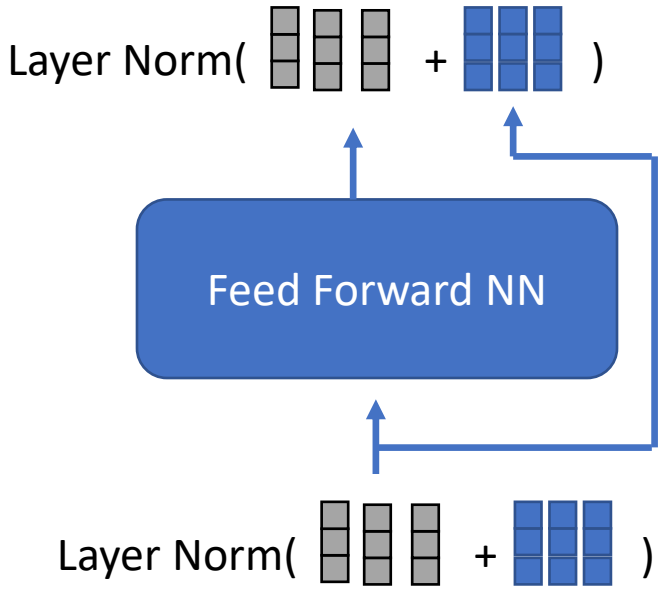
Transformer: Attention Is All You Need

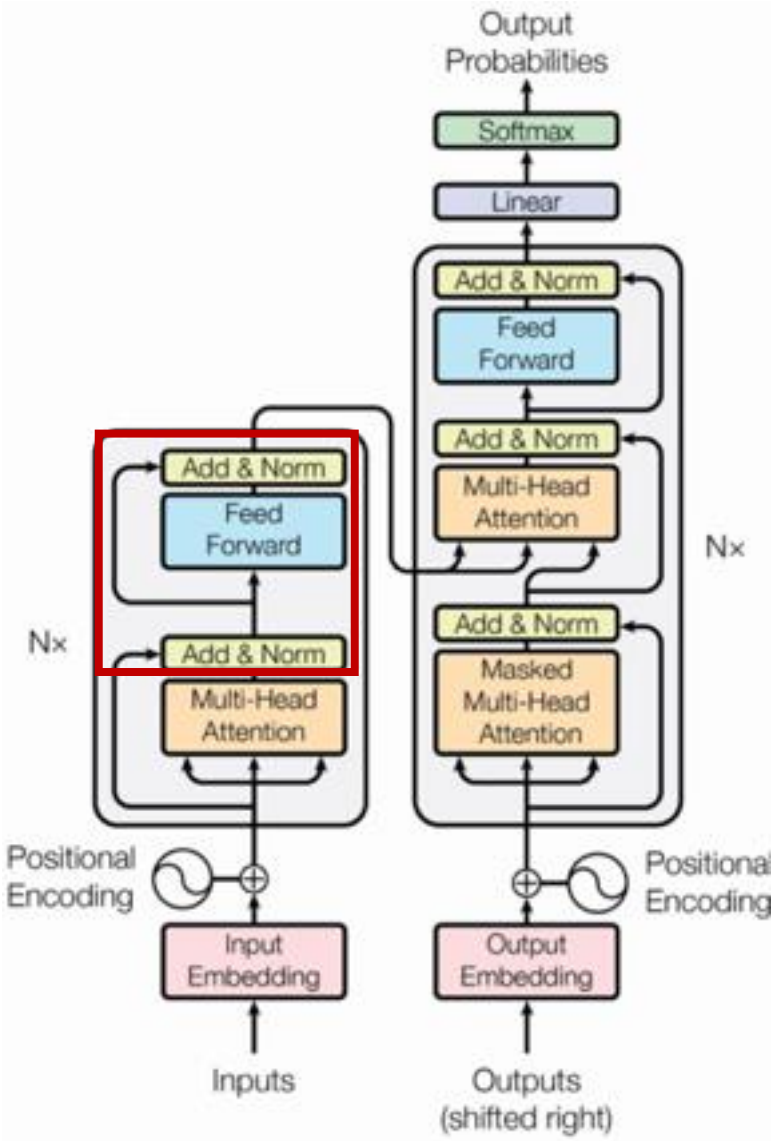




Transformer: Attention Is All You Need







Encoder Output

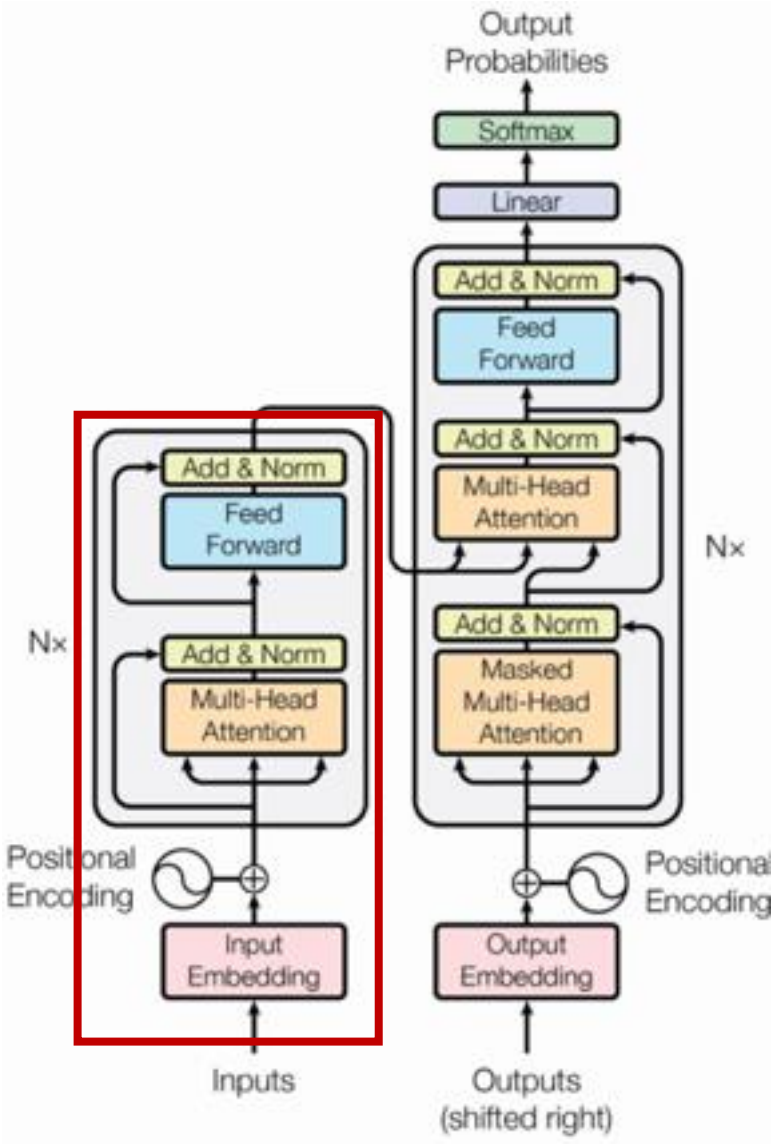
Layer Norm(+)

Linear

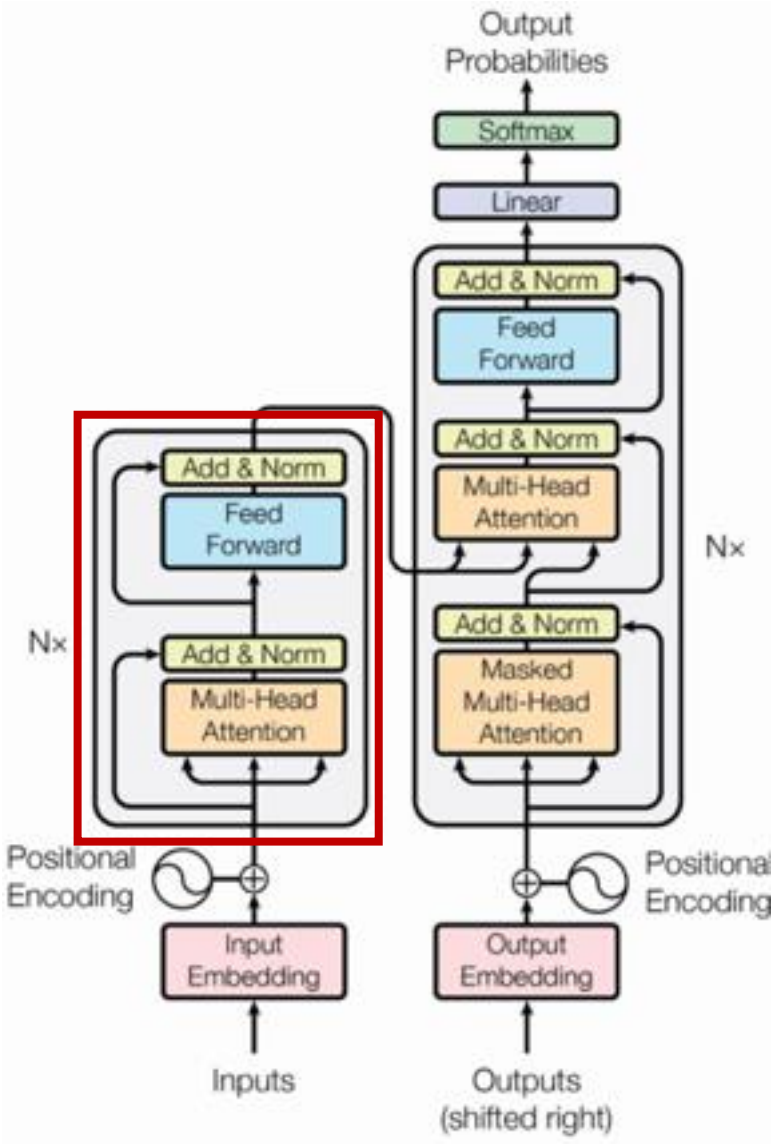
ReLU

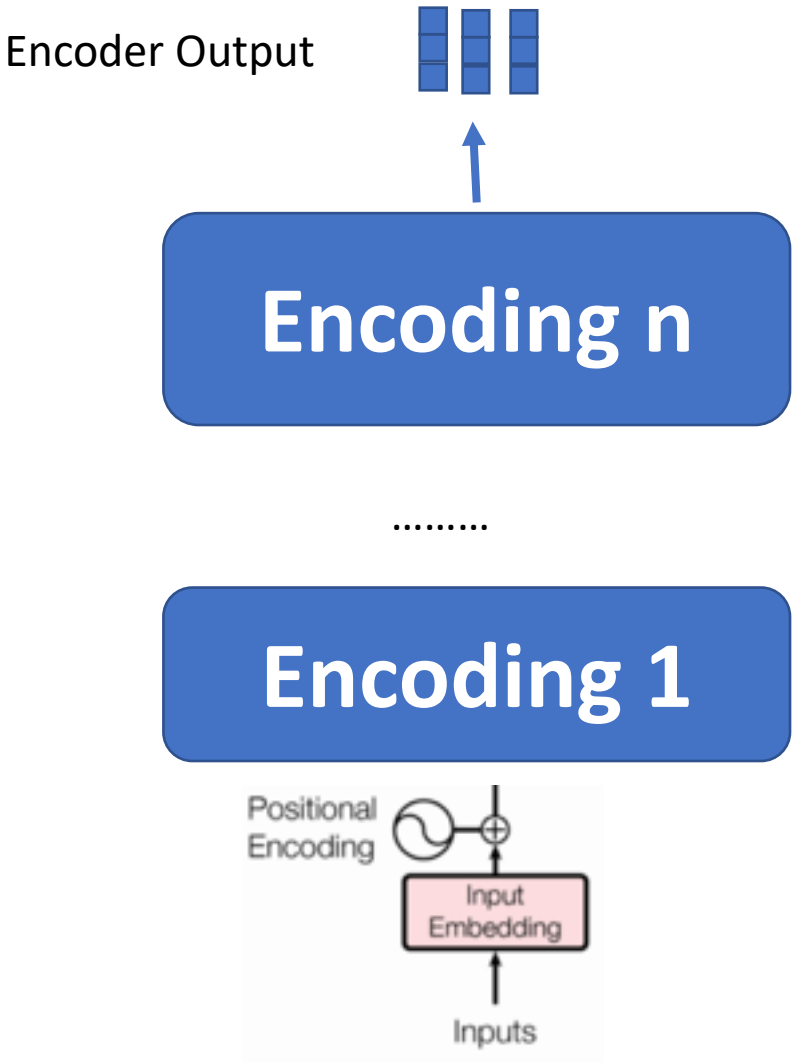
Linear

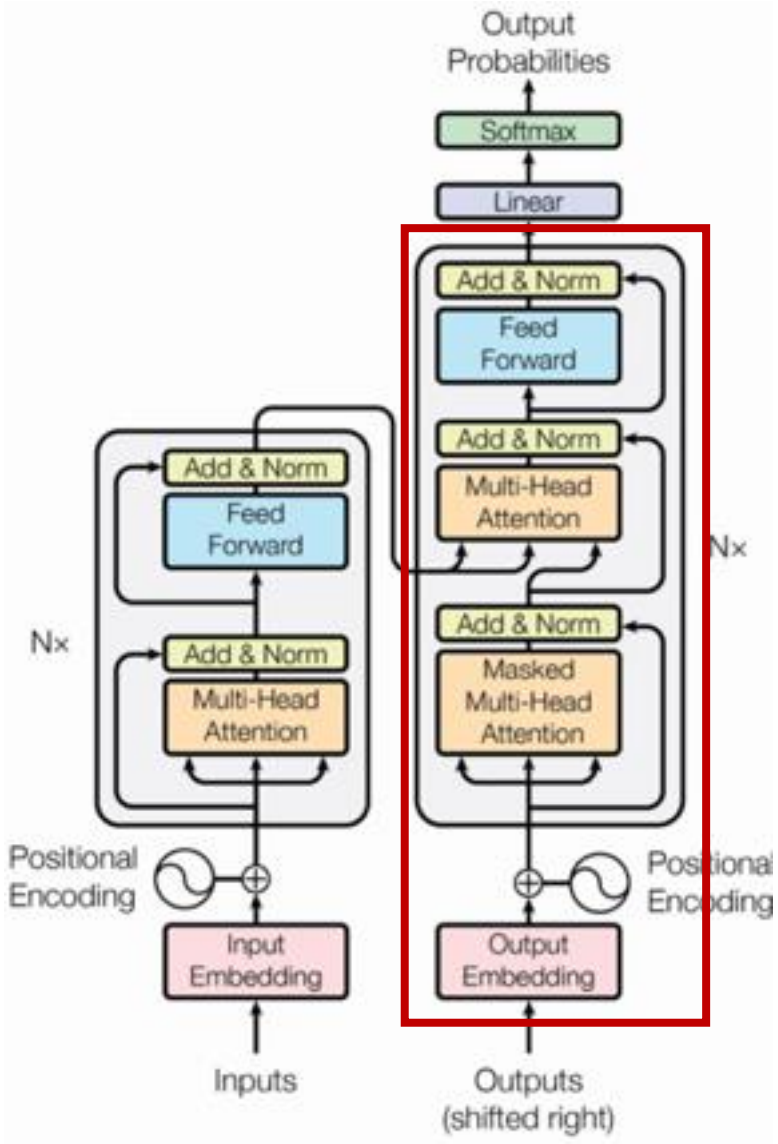
Layer Norm(+)


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
Positional Encoding
Output Embedding
Inputs
Outputs (shifted right)

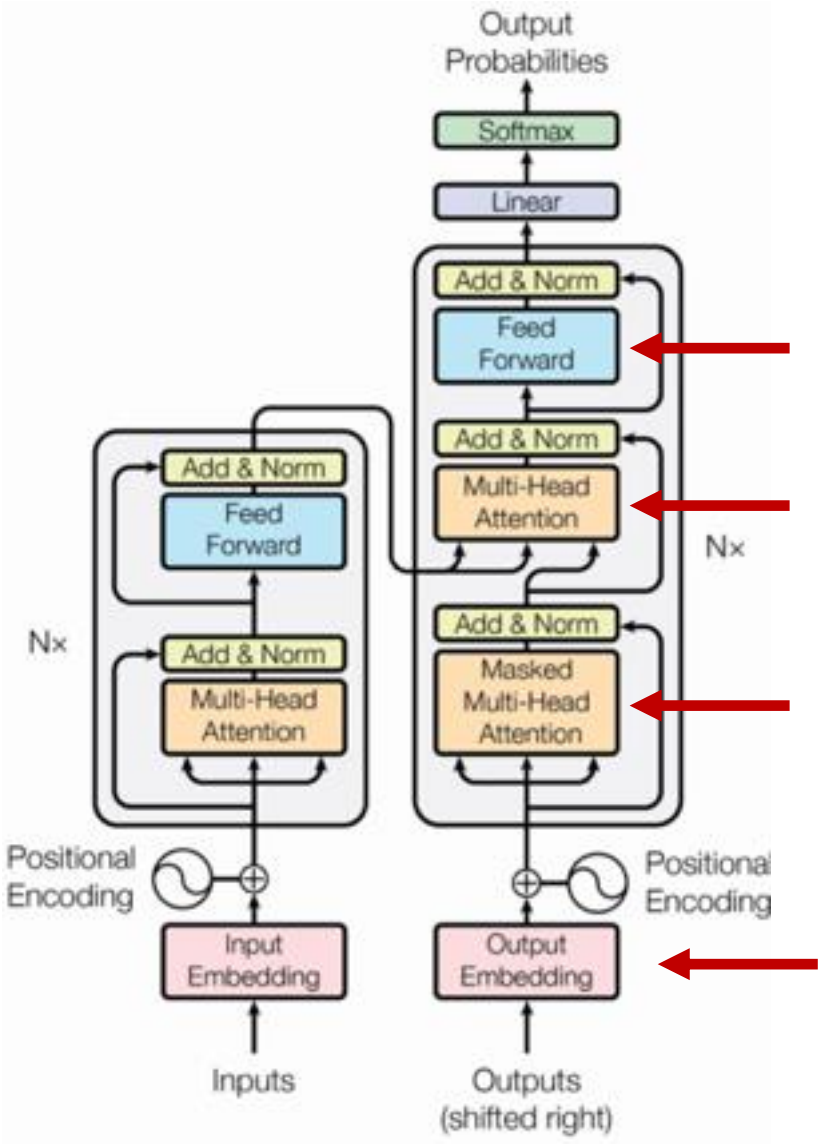
Encoding


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

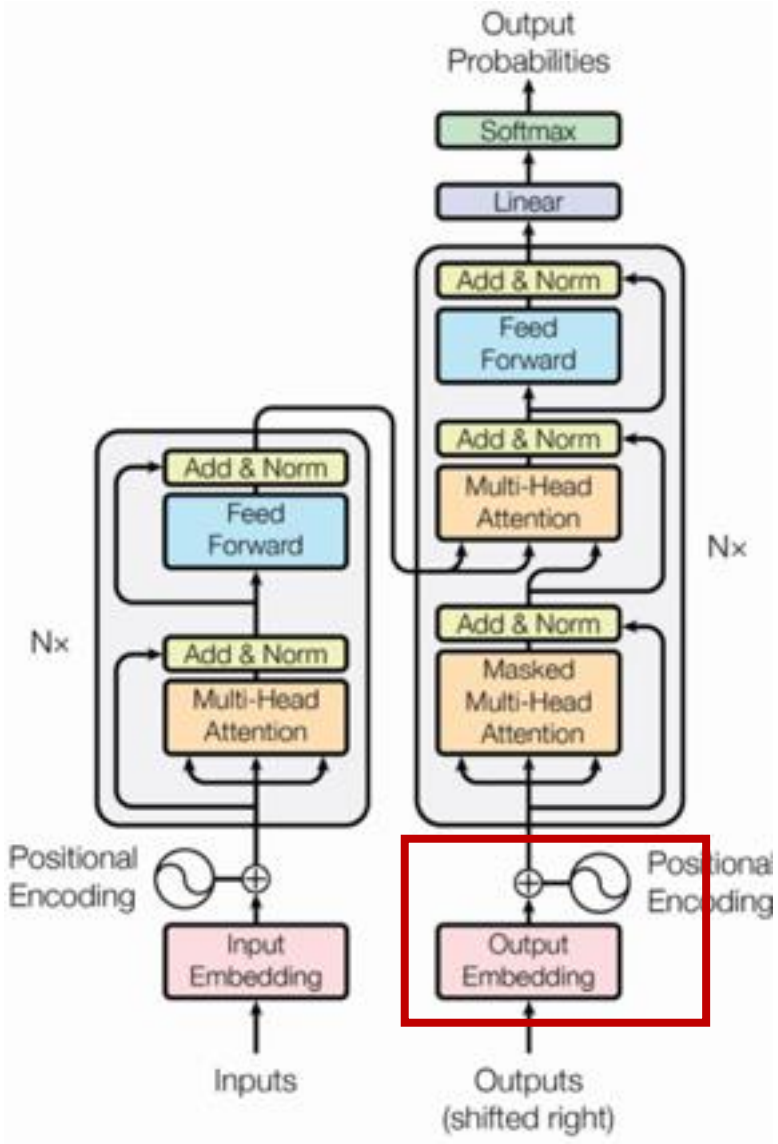

Encoder Output
Encoding n
.....
Encoding 1
Positional Encoding
Input Embedding
Inputs


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

Decoding


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

Decoding

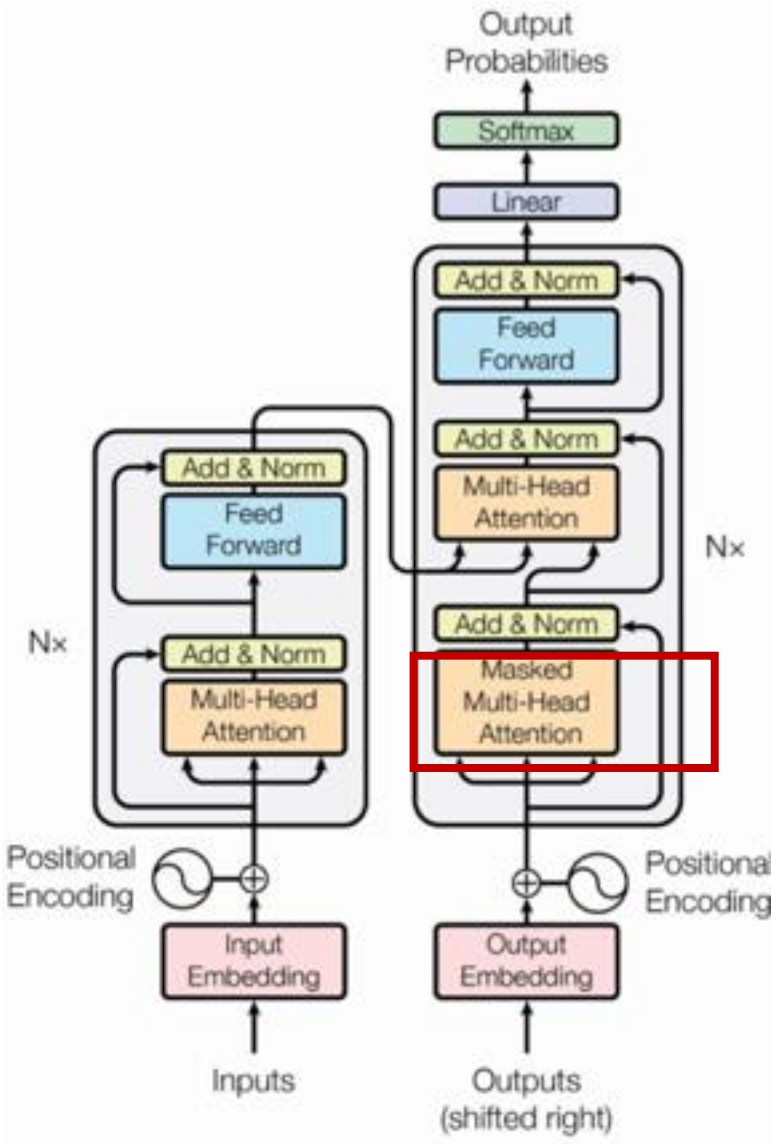


Word Embedding

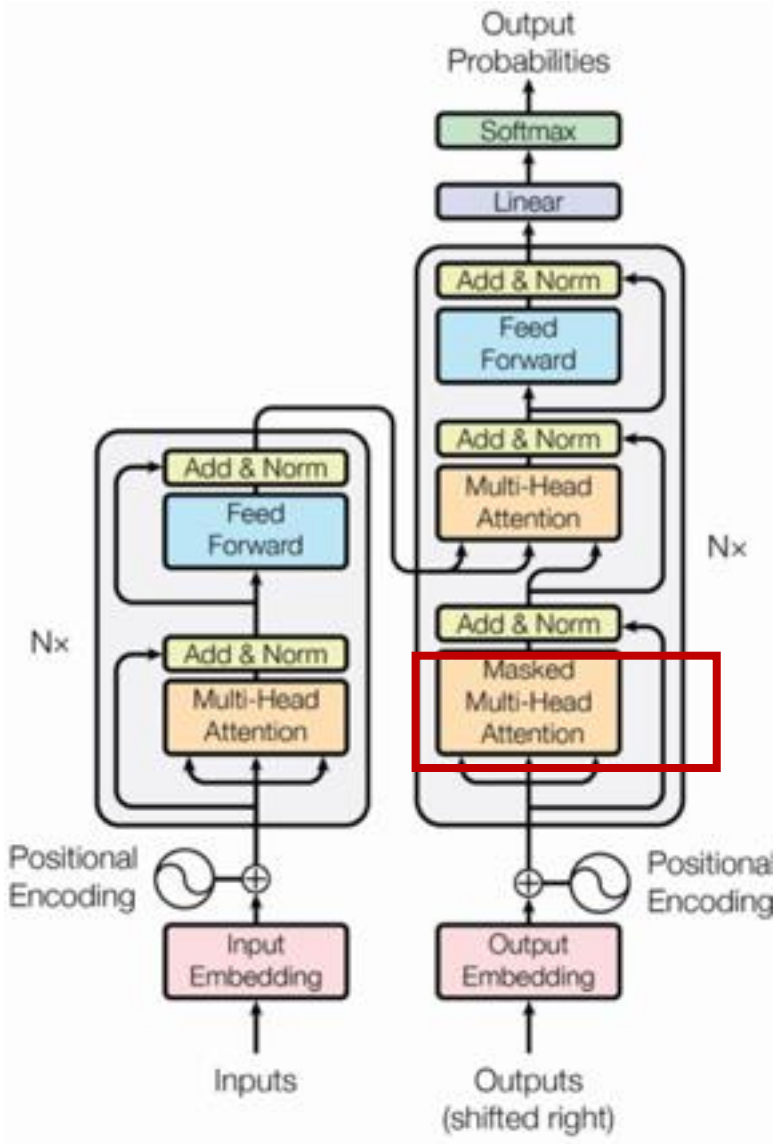
+

Positional Encoding

Masked Attention



i love you → मैं तुमसे प्यार करता हूँ



Masked Attention

Query Key Score

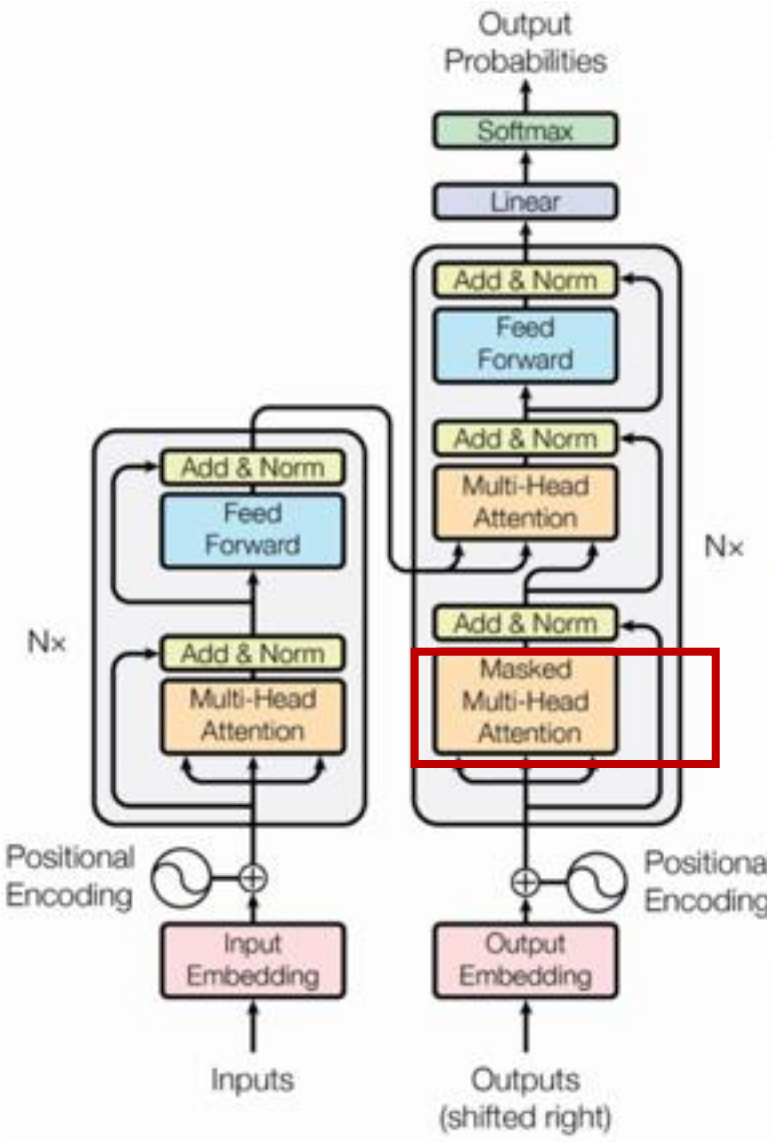
x =

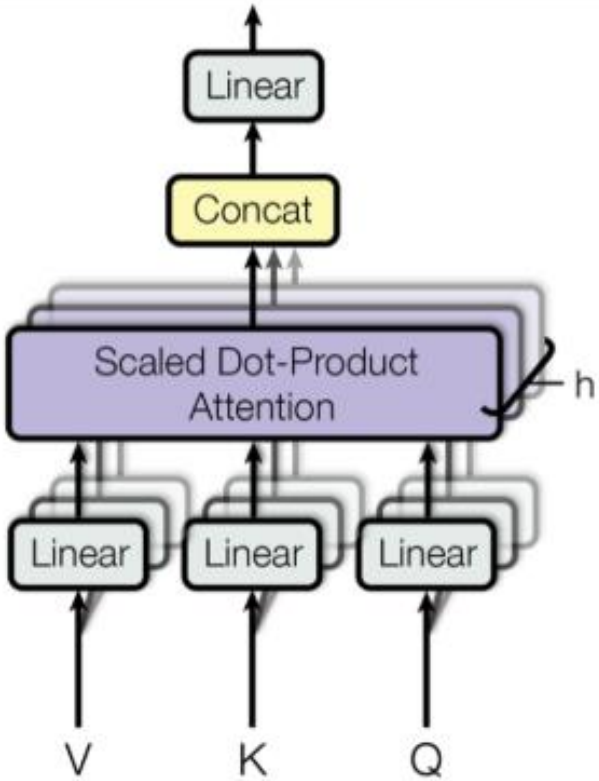
Softmax ( ) Scale

Attention Matrix

x =

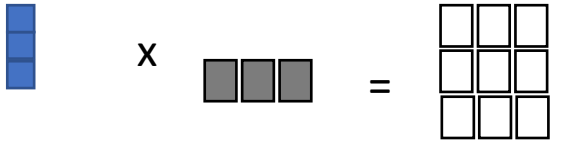
Element wise product





Masked Attention

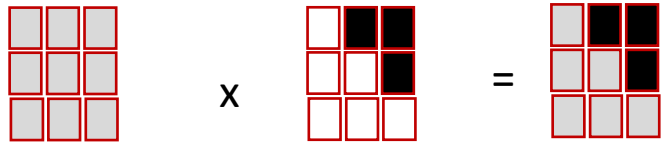
Query Key Score



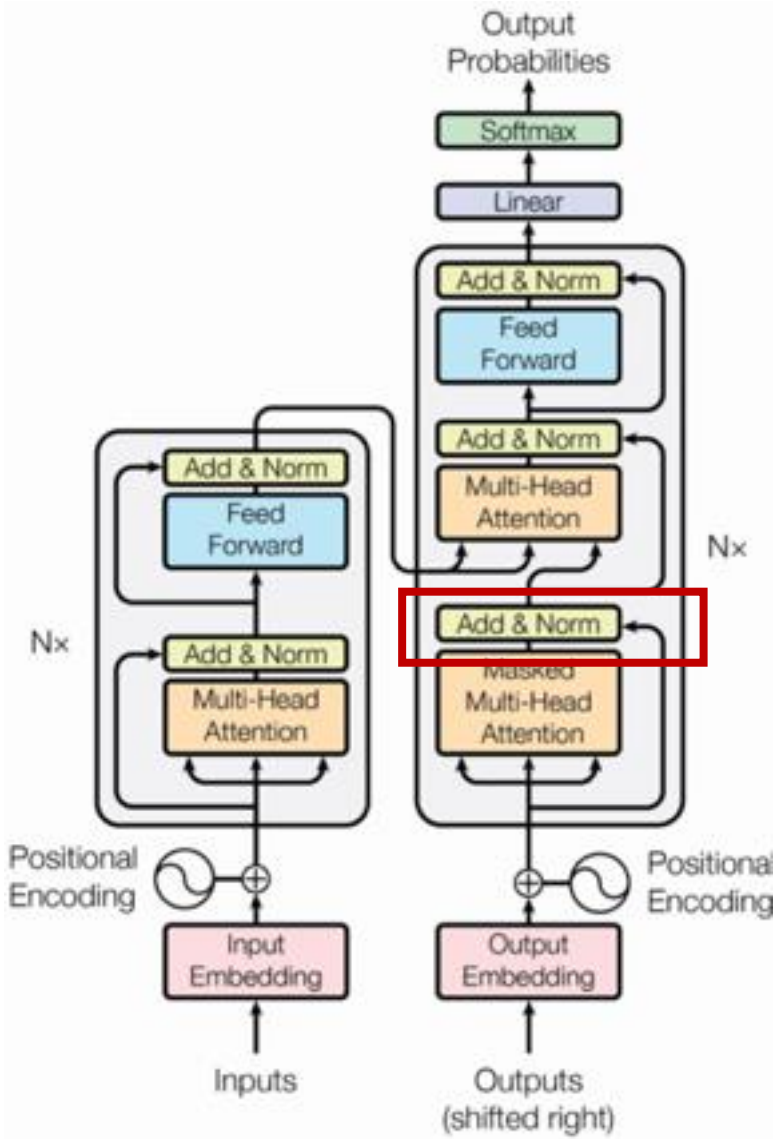
Softmax ( )

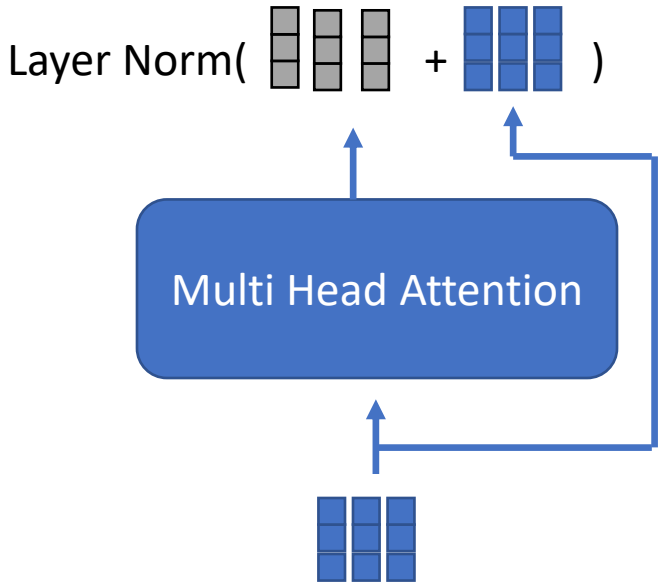
Scale

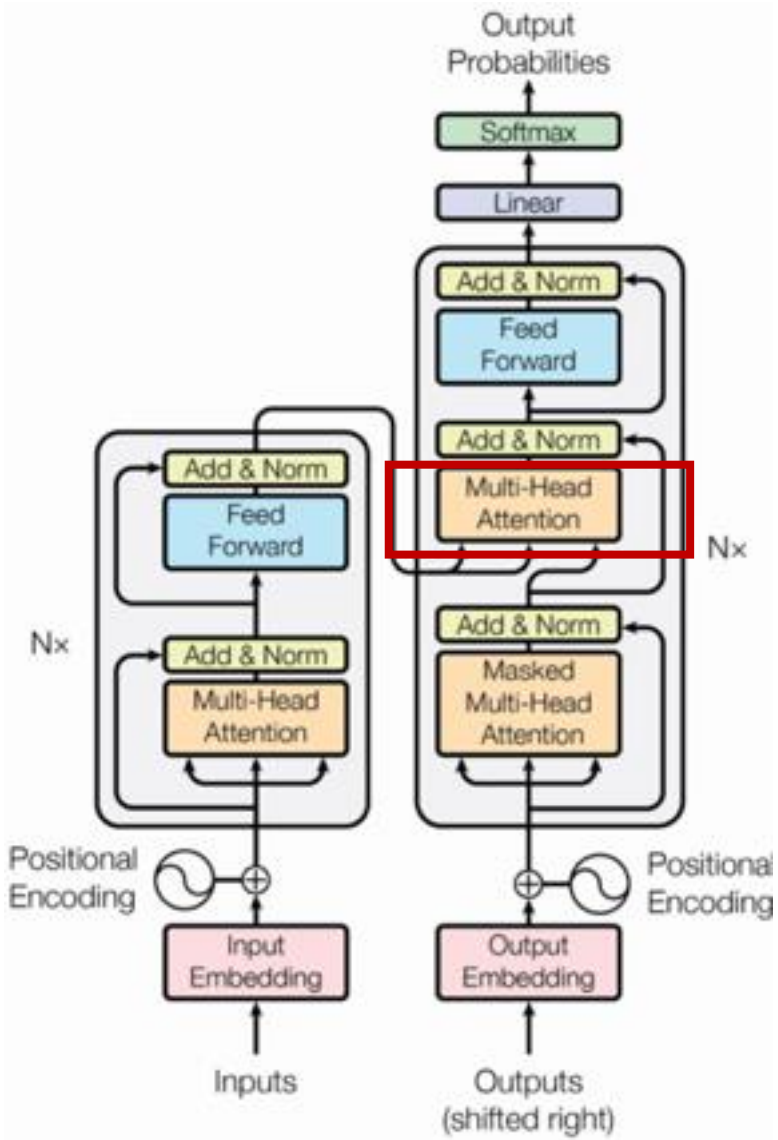
Attention Matrix



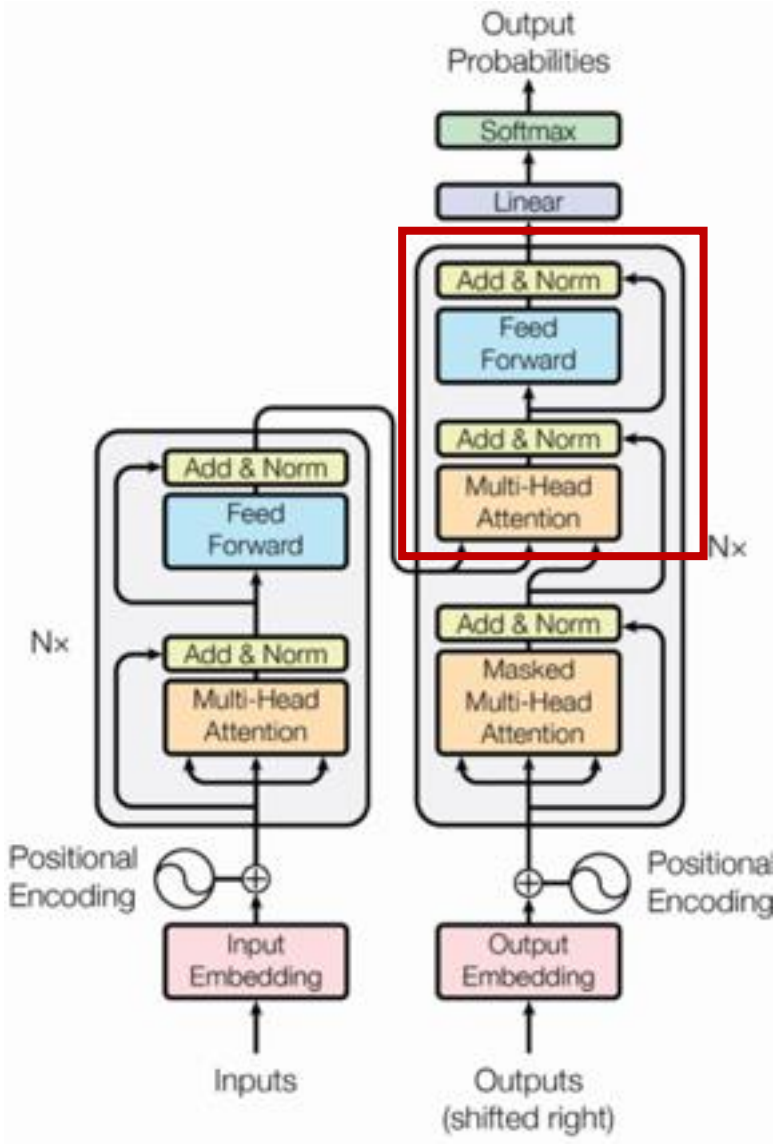
Element wise product


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)

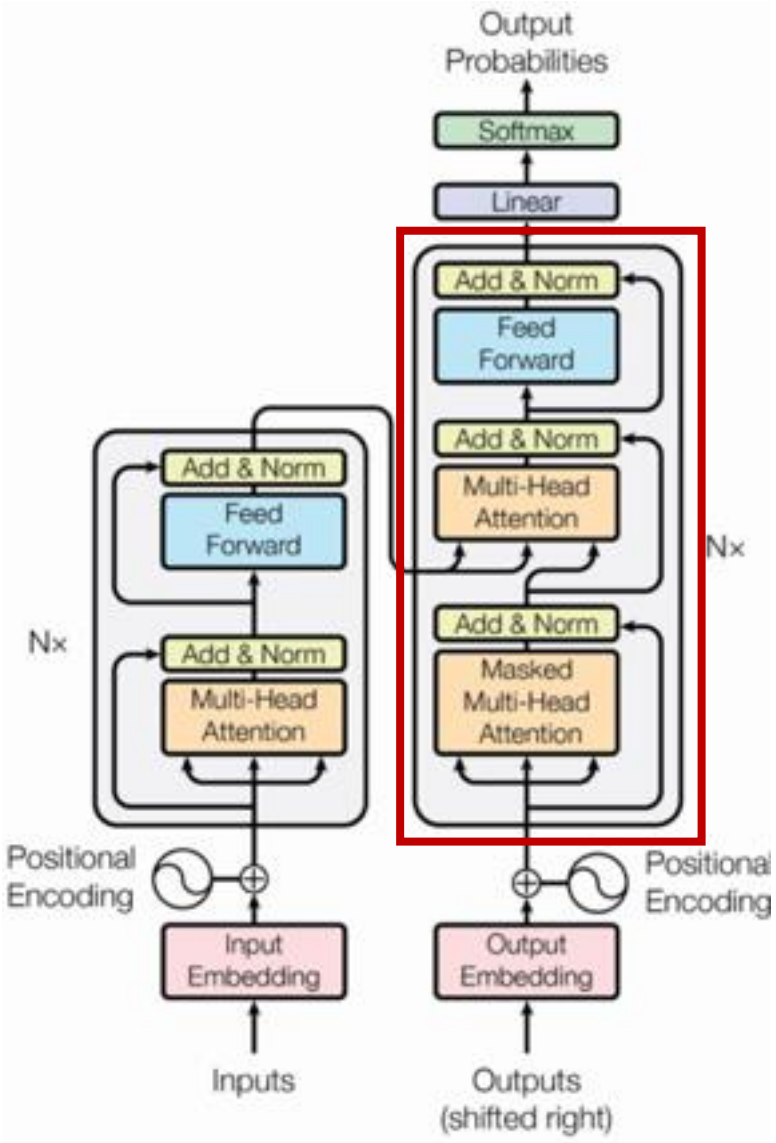

Layer Norm(+)
Multi Head Attention

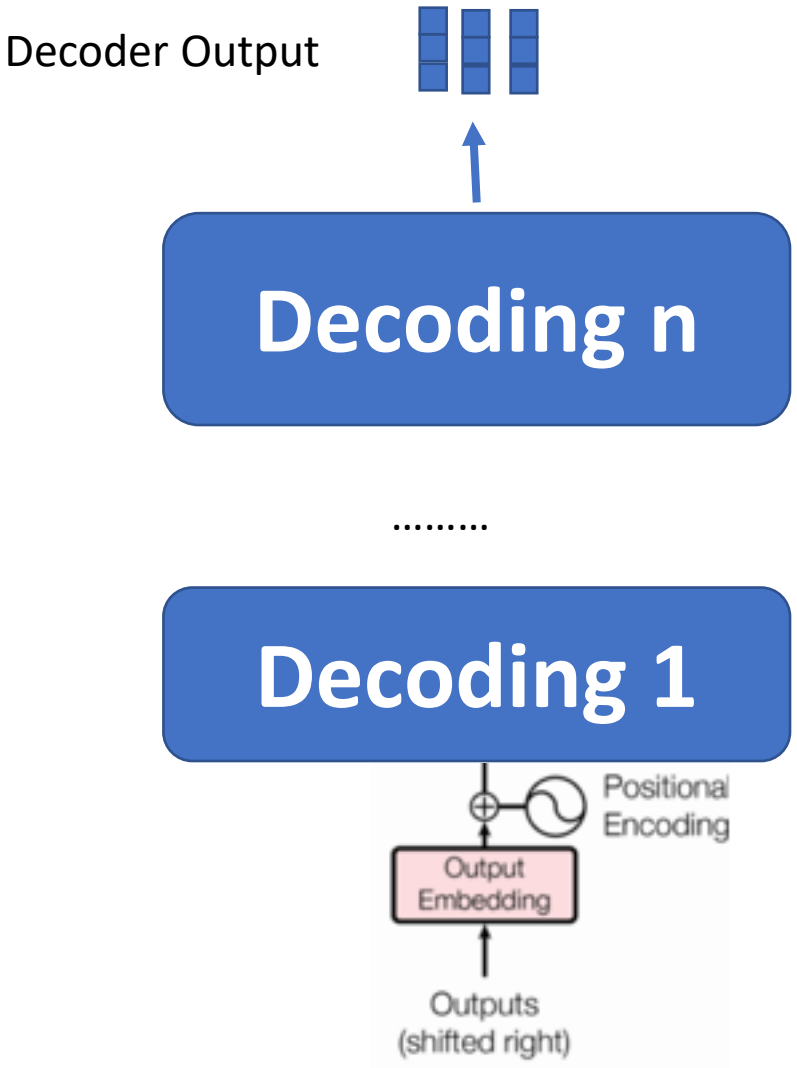


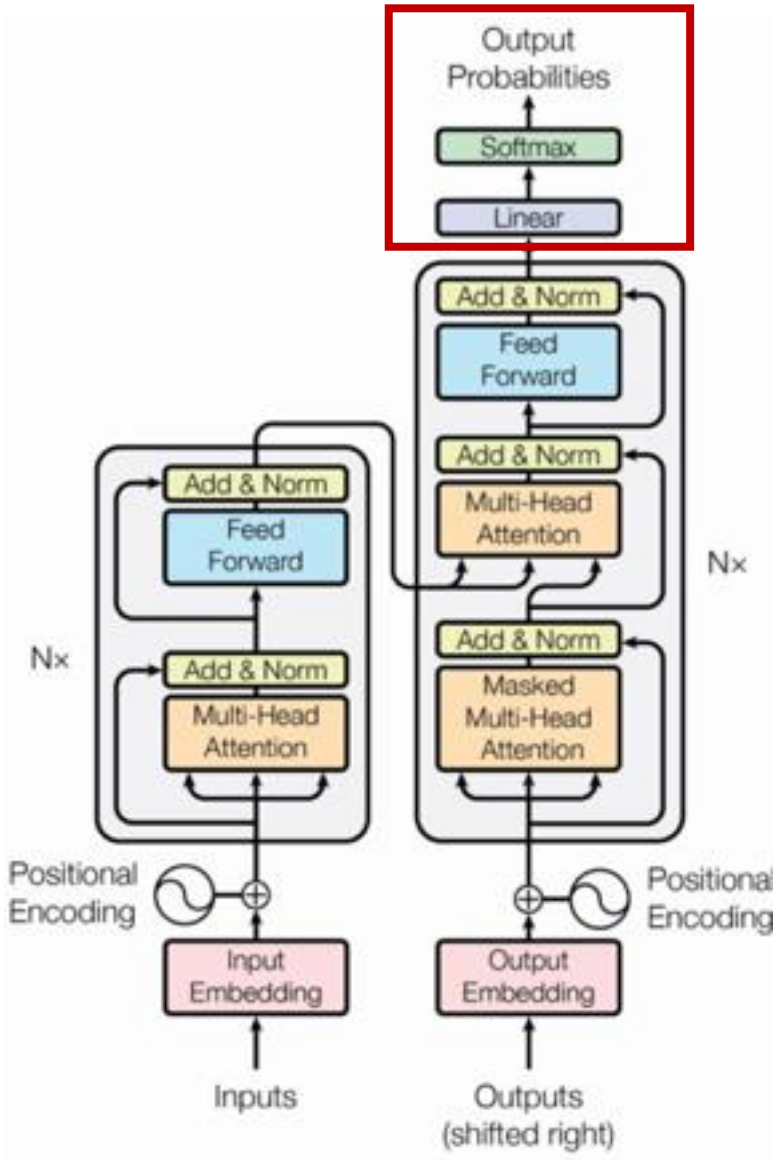
- K and V pairs from the encoder
- Q from the decoder



Other component of the decoder are same as that of the encoder


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


Decoder Output
Decoding n
.....
Decoding 1
Positional Encoding
Output Embedding
Outputs (shifted right)

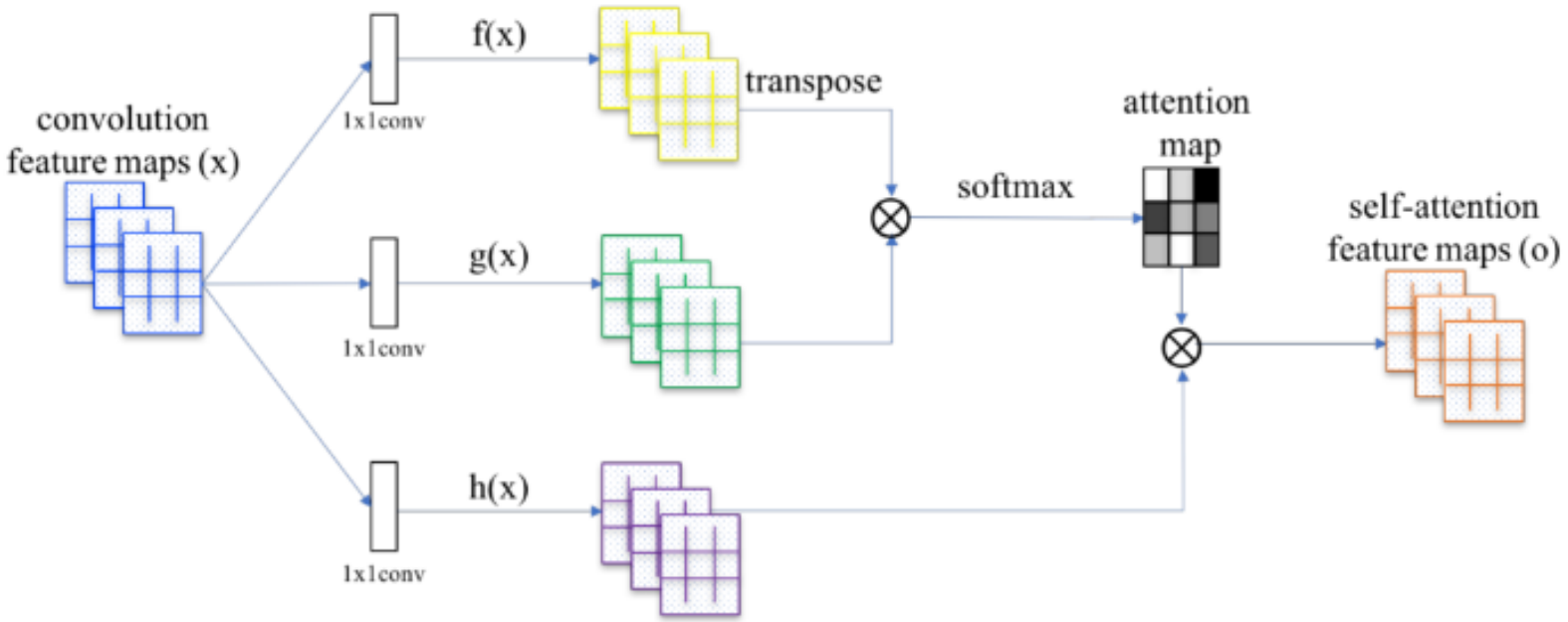

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)

Classifier

Attention in CNN

- **Selp Attention**
- **Squeeze and Excitation (SE)**
- **Convolutional Block Attention Module (CBAM)**

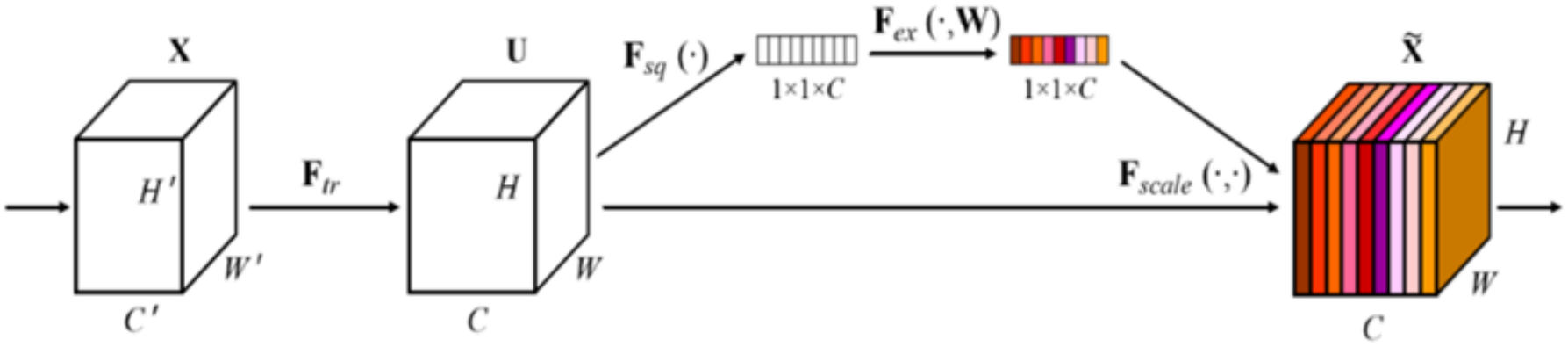
Self Attention Model in CNN



Squeeze and Excitation

Attention mechanism

Squeeze and Excitation Model

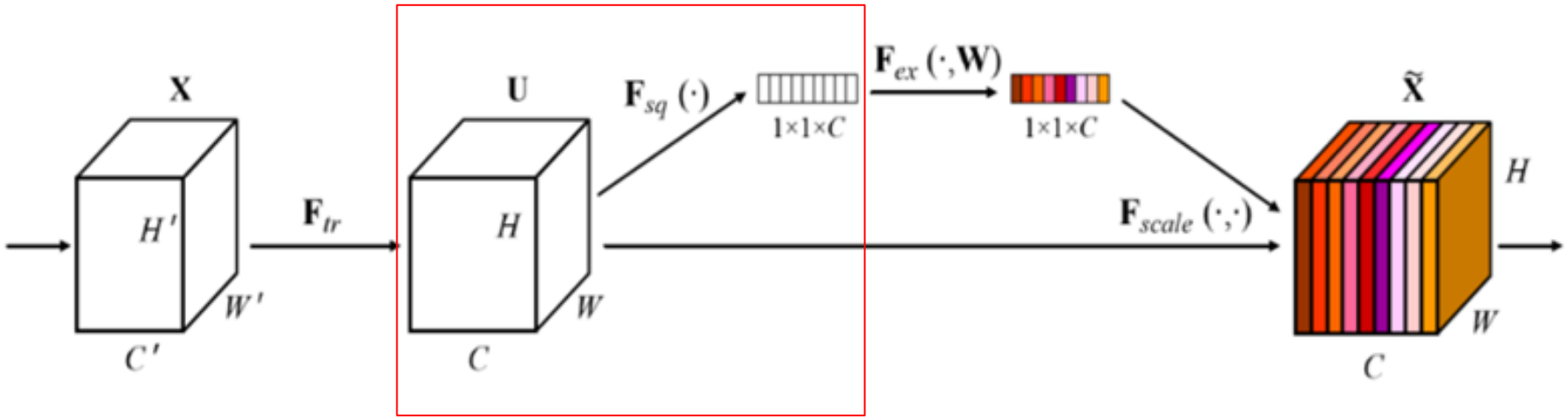


It is a channel attention

- **Squeeze operation:** Apply global average pooling to aggregate feature maps across their spatial dimensions H x W to produce a channel descriptor. It produces a **1x1xC** vector.

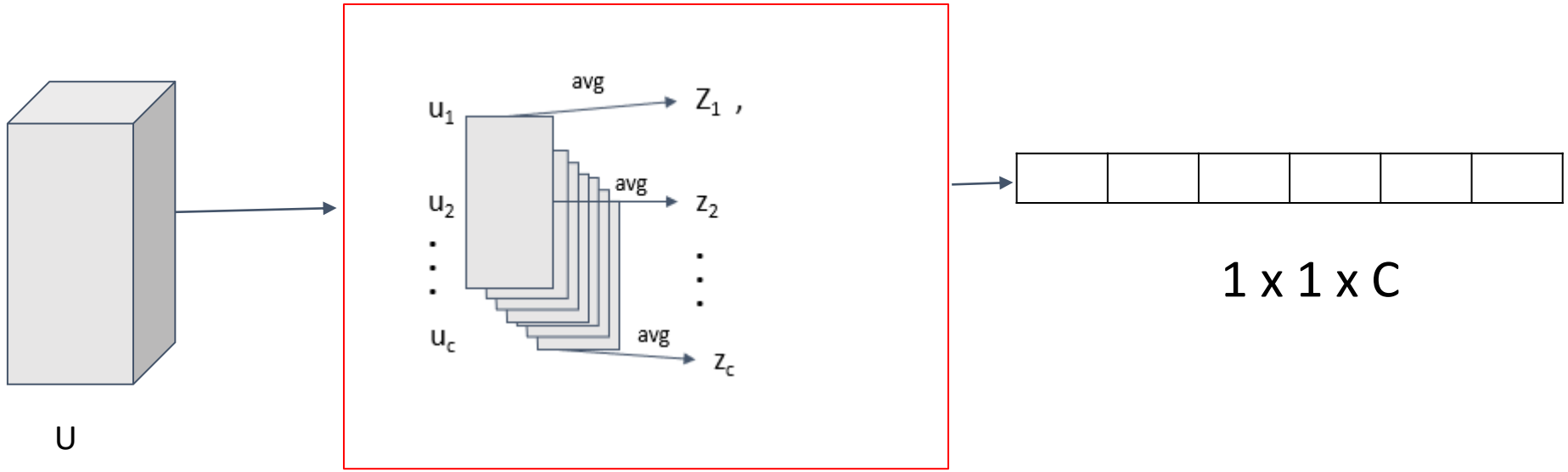
- **Excitation operation:** It applies fully-connected layers with one hidden layer using ReLU activation function on the **Squeeze vecto, and produces a channel weighted vector**

Squeeze operation



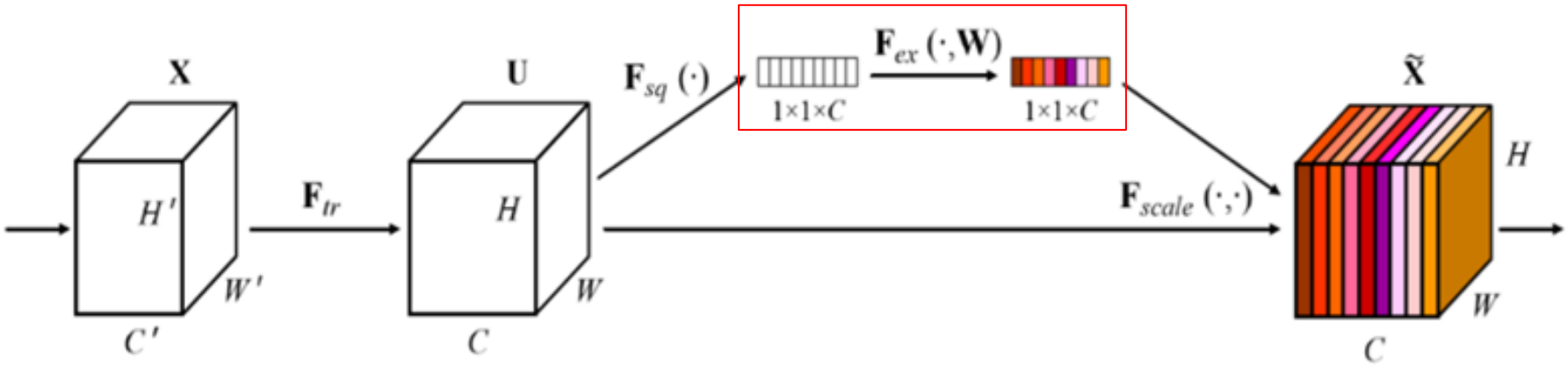
$$Z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)$$

Squeeze operation



$$Z_1 = F_{sq}(u_1) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_1(i,j)$$

Excitation operation



$$s = F_{ex}(z, W)$$

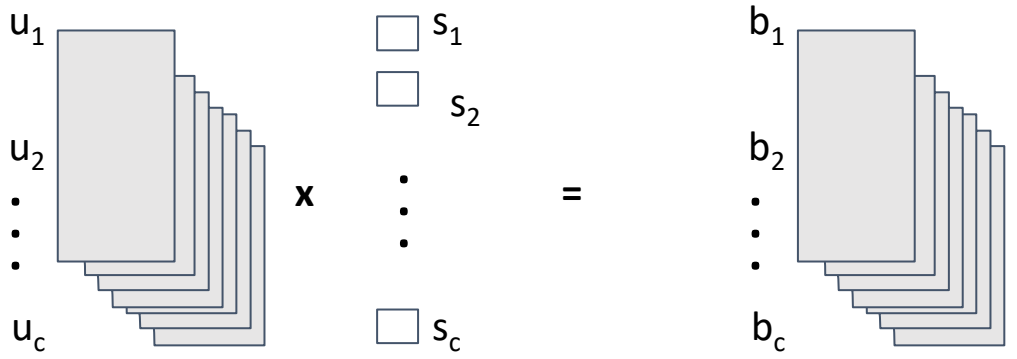
$$s = sigmoid(w_2(Relu(z, w_1)))$$

Output feature map

The final output feature map B is obtained by rescaling U with the activations s:

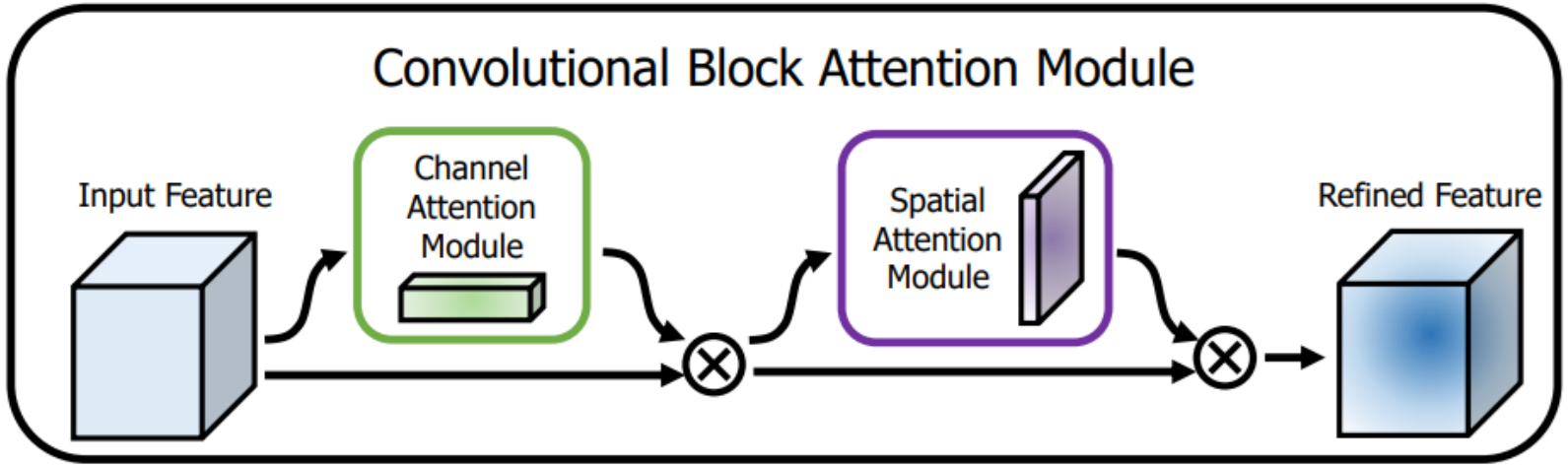
$$b_c = F_{scale}(u_c, s_c)$$

$F_{scale}(u_c, s_c)$ refers to channel-wise multiplication between the scalar $\mathbf{s}_c$ and the feature map $\mathbf{u}_c$



Convolutional Block Attention Module (CBAM)

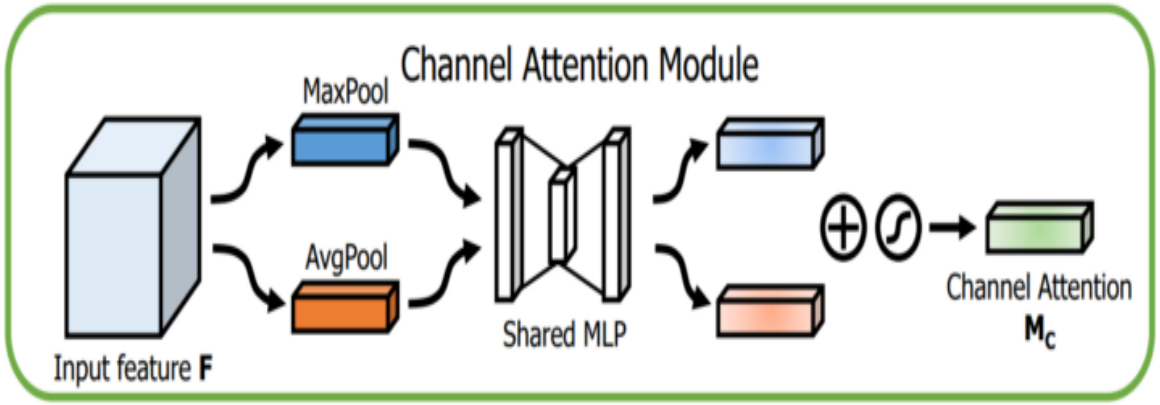
(Provide attention on both channel and spatial)



It has two components

- **Channel Attention module (CAM)**: Which channel is important
- **Spatial Attention module (SAM)**: Which region in the feature map is important

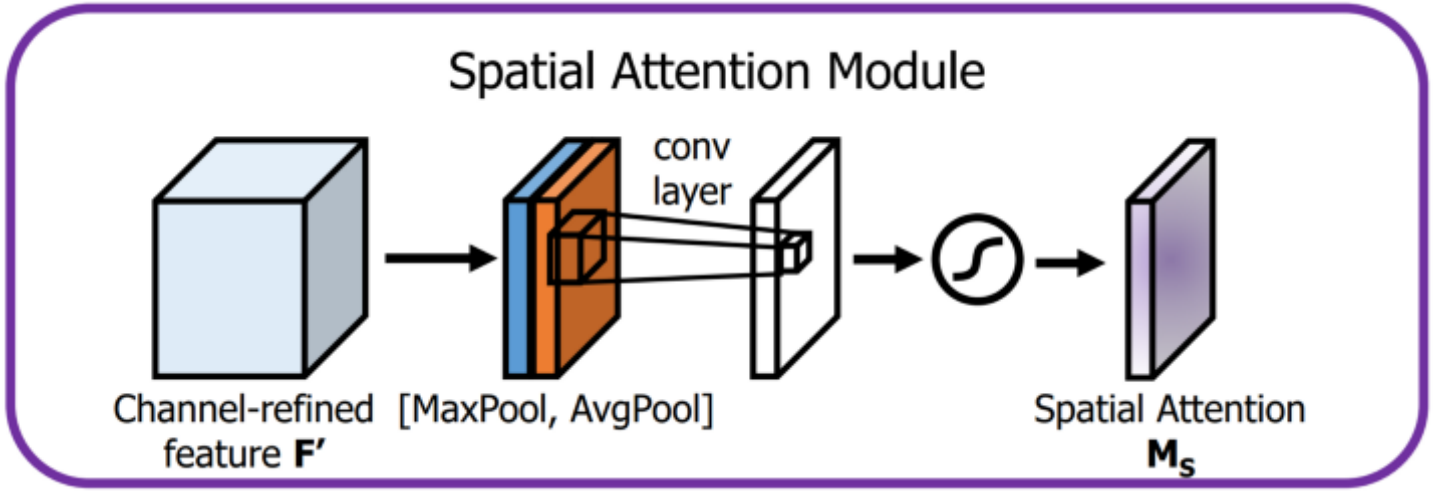
Channel Attention module (CAM)



It resembles Squeeze and Excite (SE) model.

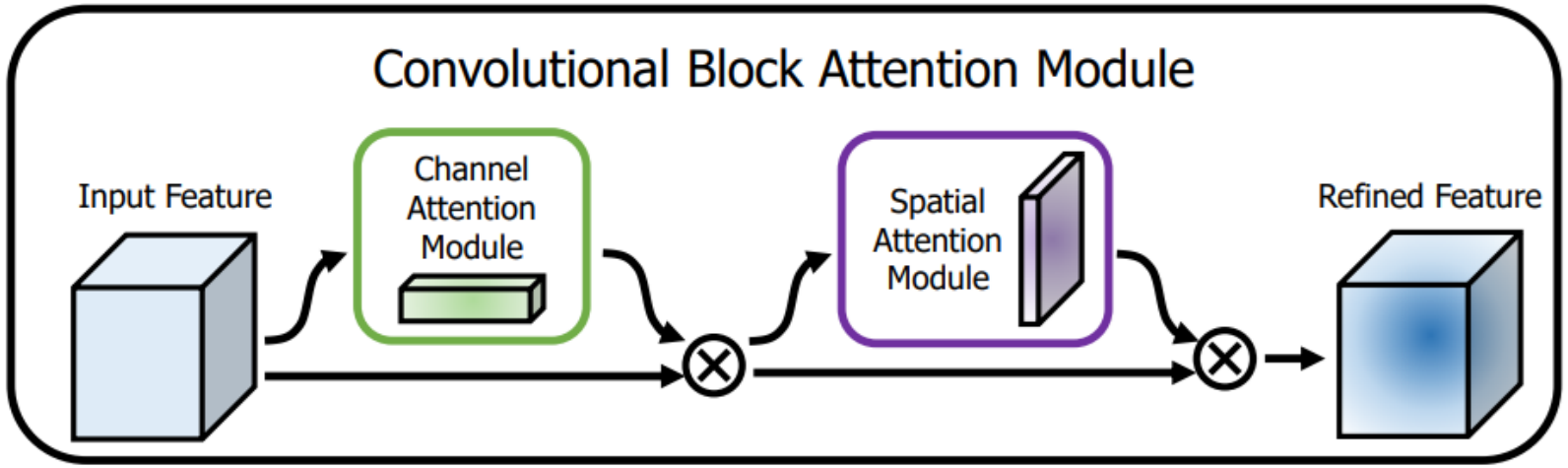
$$\mathbf{M_c}(\mathbf{F}) = \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F})))$$
$$= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),$$

Spatial Attention module (SAM)



- Apply **Global Average Pool** and **Global Max Pooling across channels.**
- Concatenate and pass it through a small convolutional block of 7x7 kernel size.

$$\begin{aligned}\mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),\end{aligned}$$



$$\mathbf{F}' = \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F},$$
$$\mathbf{F}'' = \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}',$$