

深度學習

- RNN, LSTM, Transformer(Self-Attention), ViT

黃志勝 (Tommy Huang)

義隆電子 人工智慧研發部

國立陽明交通大學 AI學院 合聘助理教授

國立台北科技大學 電資學院合聘助理教授

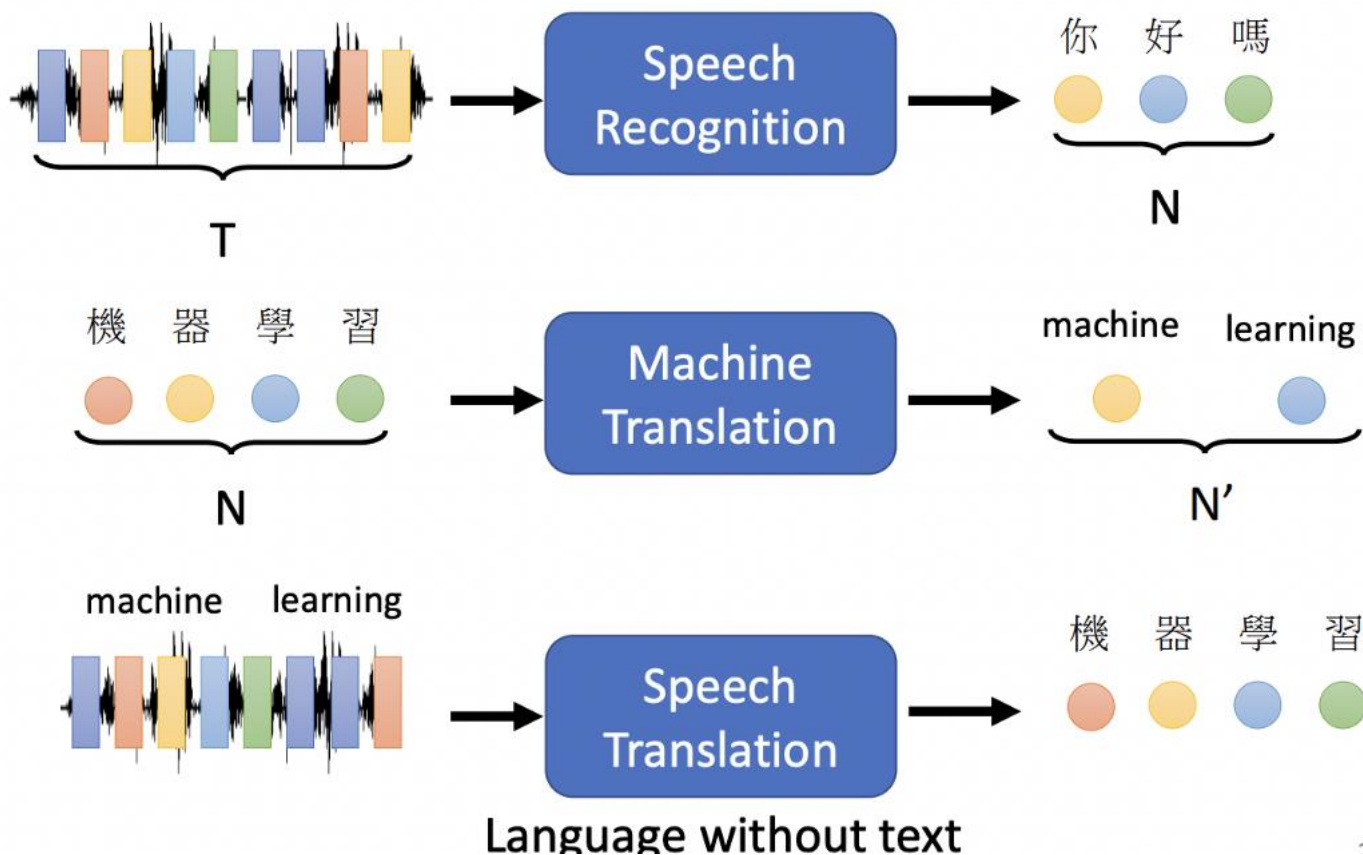


Sequence-to-sequence

Sequence-to-sequence (Seq2seq)

Input a sequence, output a sequence

The output length is determined by model.

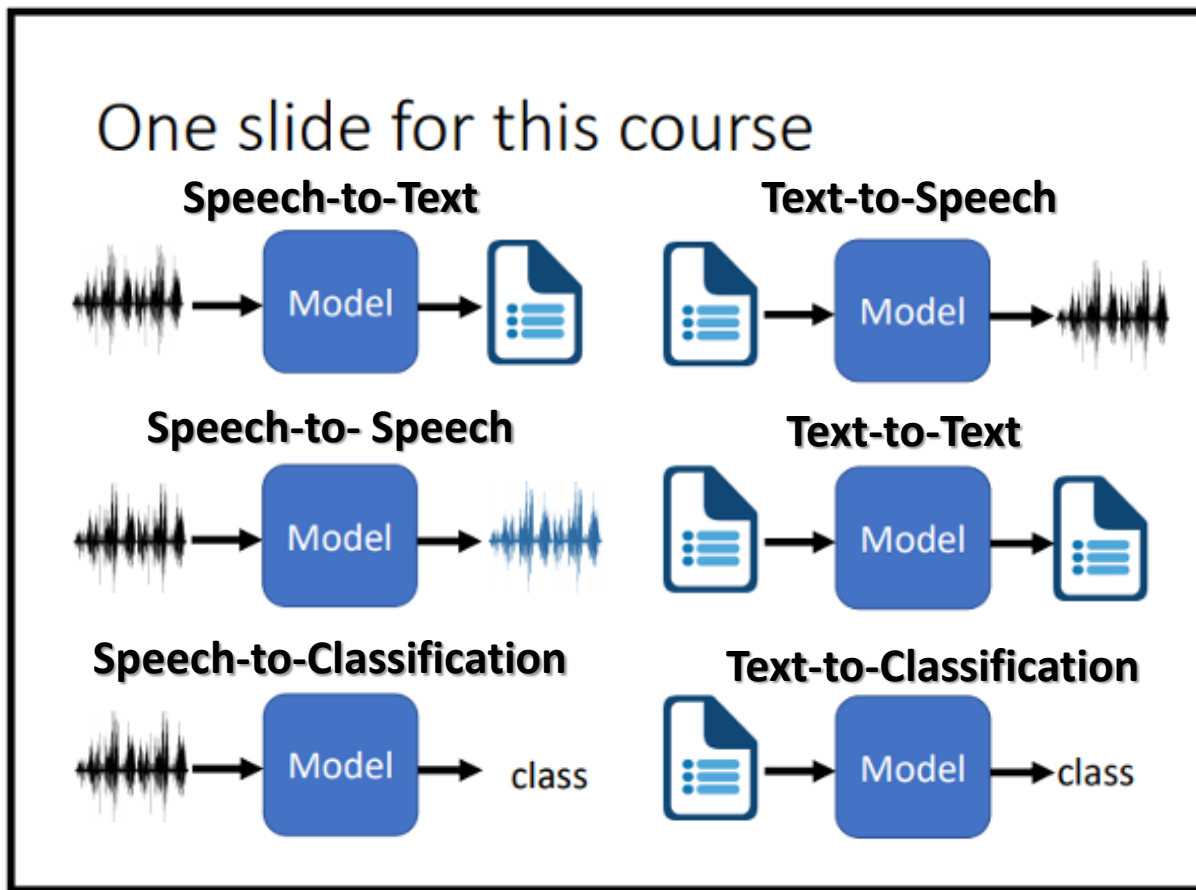


圖片: 李宏毅老師



Deep Learning for Human Language Processing

深度學習與人類語言處理

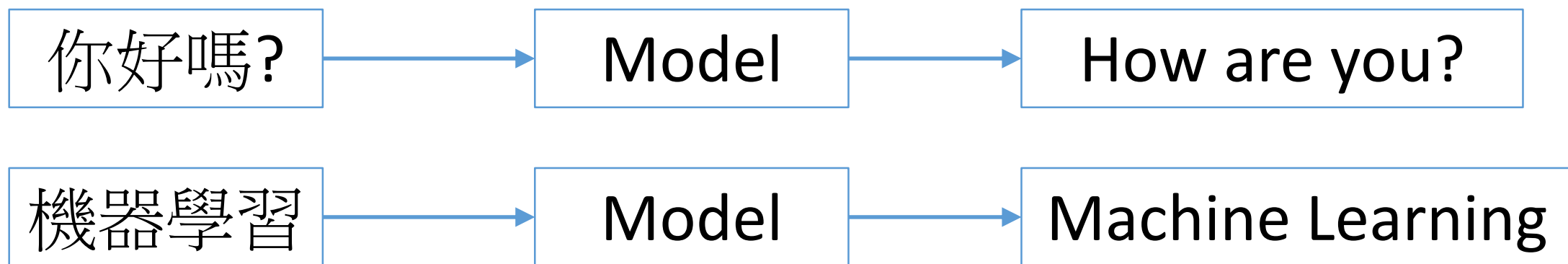


Source webpage: <https://speech.ee.ntu.edu.tw/~hylee/dlhlp/2020-spring.html>



Sequence-to-sequence

Text-to-Text: 中英文翻譯



電腦怎麼知道

機器學習 → Machine Learning

你好嗎? → How are you?



Sequence-to-sequence

Text-to-Text: 中英文翻譯

電腦怎麼知道

機器學習 → Machine Learning

你好嗎? → How are you?

給電腦一堆配對好的資料，讓模型去學中文和英文之間的關聯性。

Example/dataset.xlsx

	A	B
1	english	chinese
2	Hi.	嗨。
3	Hi.	你好。
4	Run.	你用跑的。
5	Wait!	等等！
6	Wait!	等一下！
7	Begin.	开始！
8	Begin	开始
9	Hello!	你好。
10	I try.	我试试。
11	I won!	我赢了。
12	Oh no!	不会吧。
13	Cheers!	乾杯!
14	Got it?	你懂了吗？
15	He ran.	他跑了。
16	Hop in.	跳进来。



中英文字-數字化

Example/dataset.xlsx

	A	B
1	english	chinese
2	Hi.	嗨。
3	Hi.	你好。
4	Run.	你用跑的。
5	Wait!	等等！
6	Wait!	等一下！
7	Begin.	开始！
8	Begin	开始
9	Hello!	你好。
10	I try.	我试试。
11	I won!	我赢了。
12	Oh no!	不会吧。
13	Cheers!	乾杯！
14	Got it?	你懂了吗？
15	He ran.	他跑了。
16	Hop in.	跳进来。

Word Embedding:

將『字詞/句子/文件』轉換成『向量』形式

Hi → [30, 7]

你好 → [7, 36]

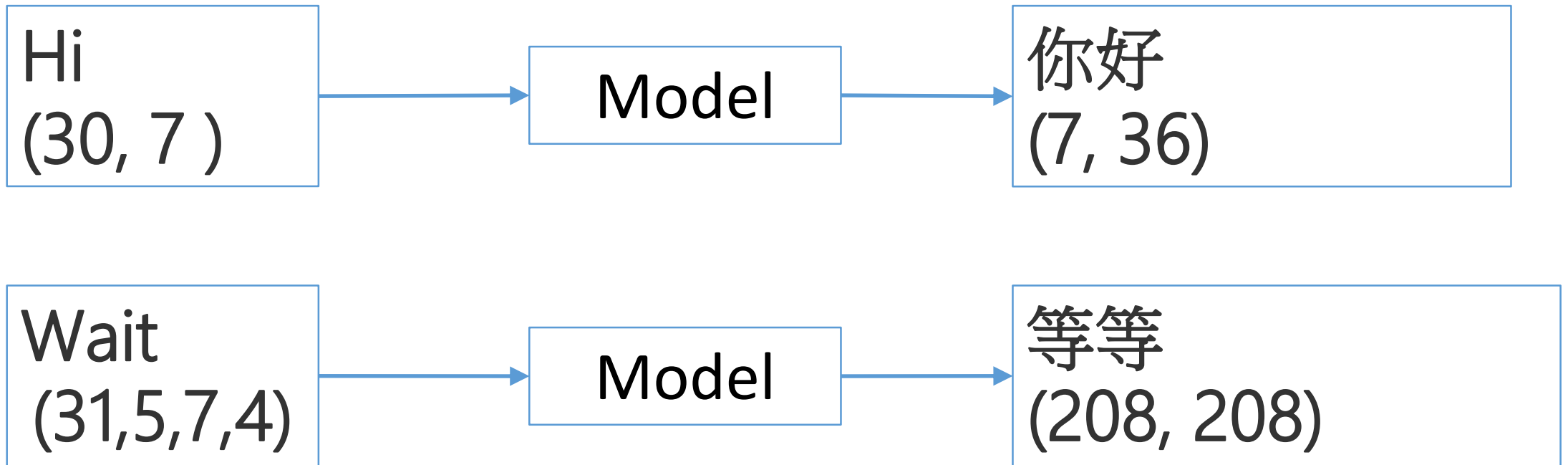
Wait → [31, 5, 7, 4]

等等 → [208, 208]

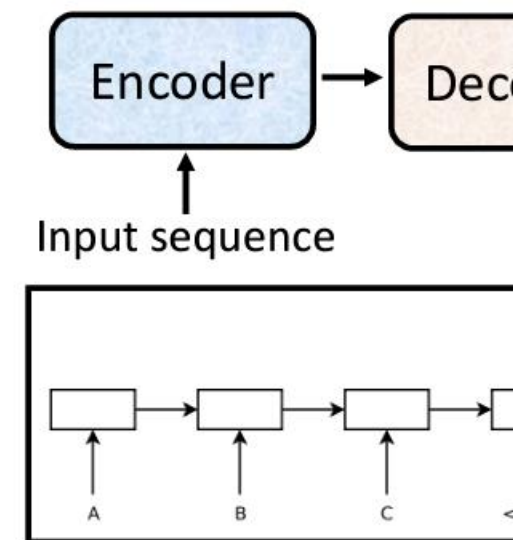
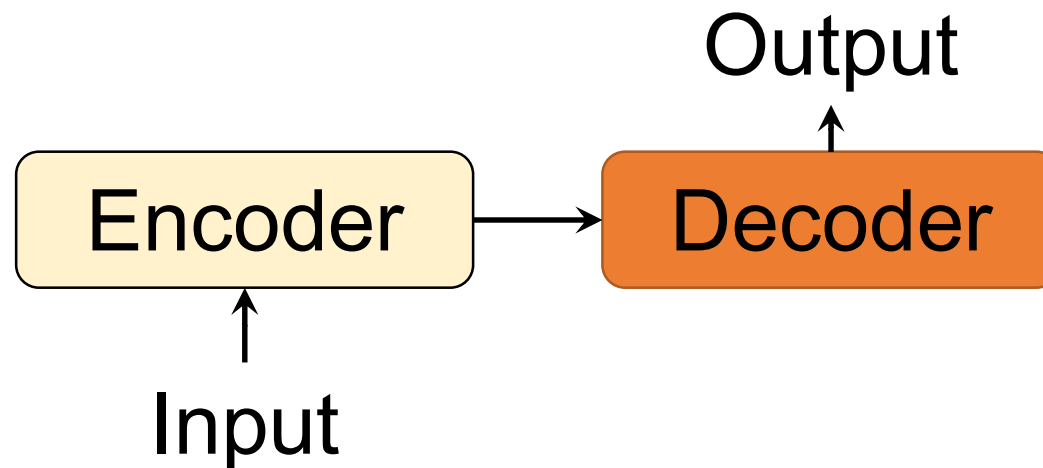
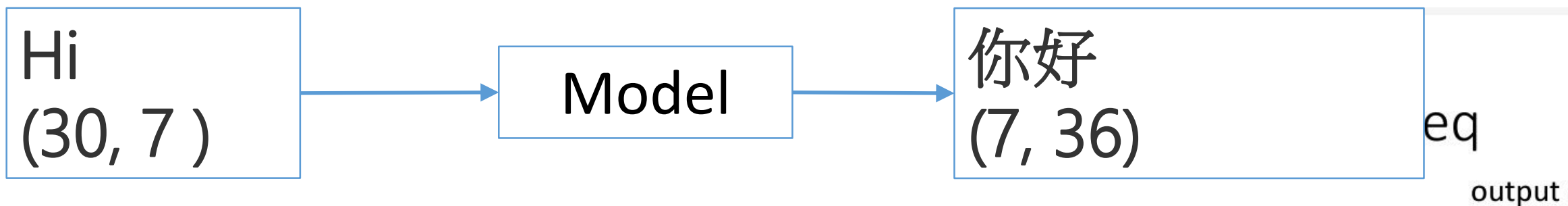


Sequence-to-sequence

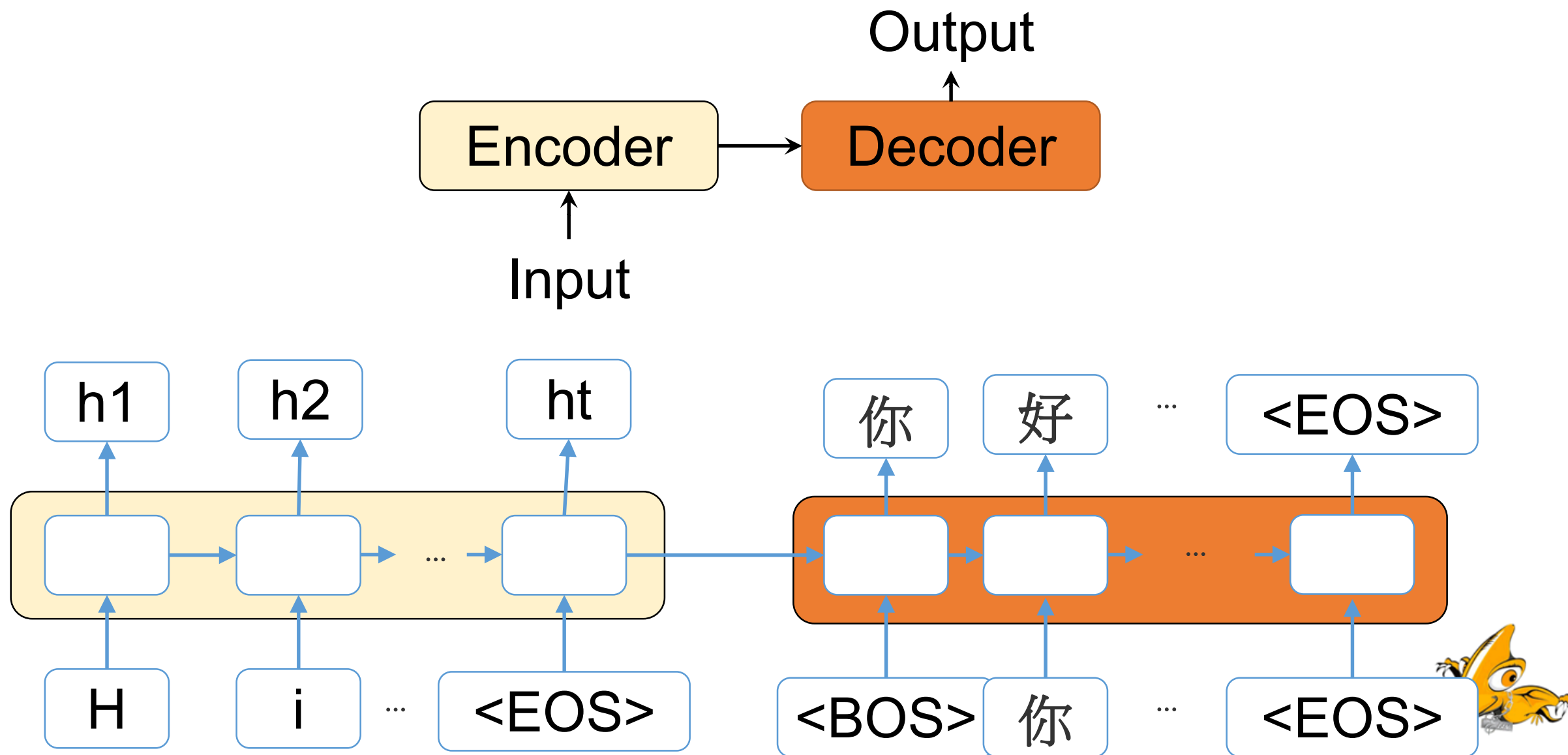
Text-to-Text: 中英文翻譯



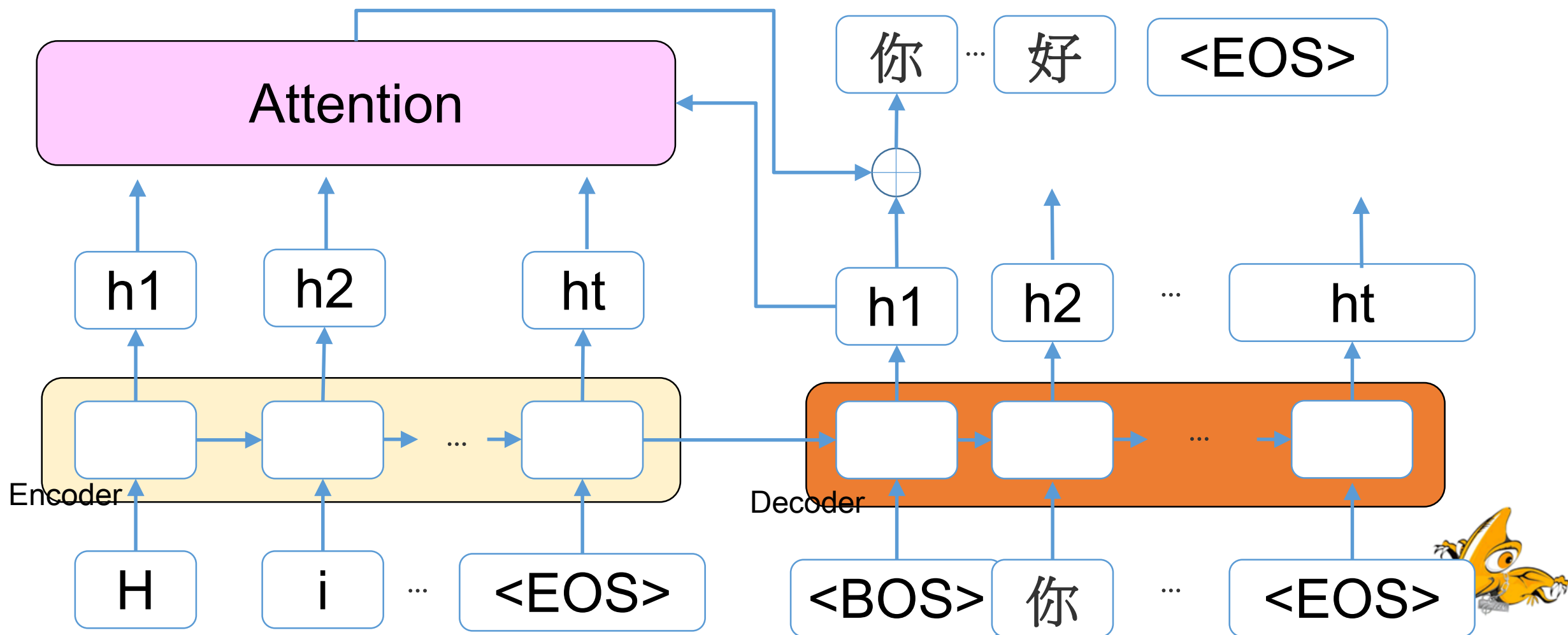
Sequence-to-sequence



Sequence-to-sequence



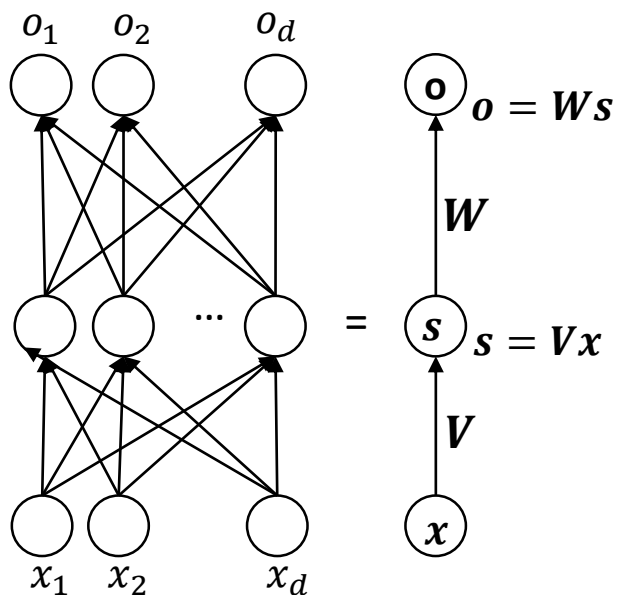
Sequence-to-sequence



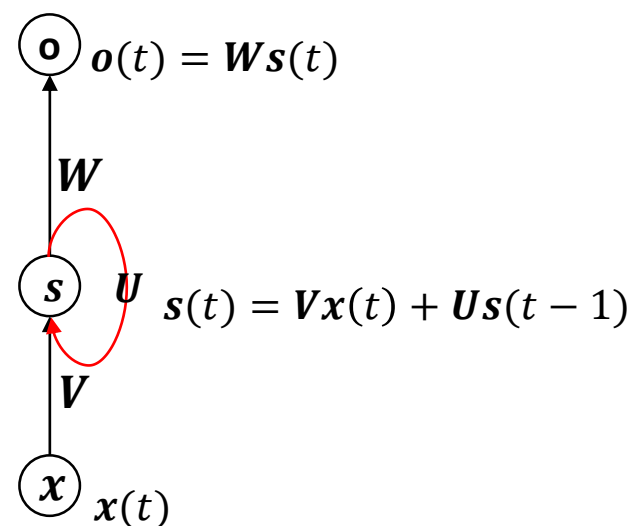
Recurrent Neural Network (RNN)

- RNN和一般MLP的差異就是引進了具有記憶功能的「狀態(State)」

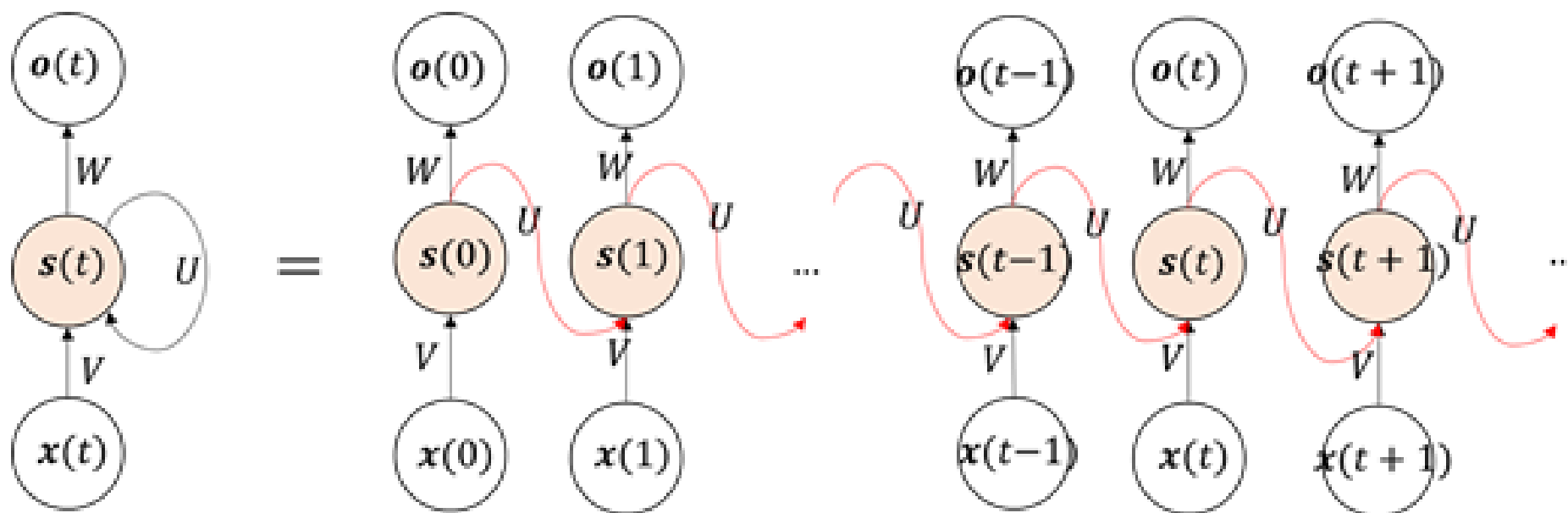
MLP



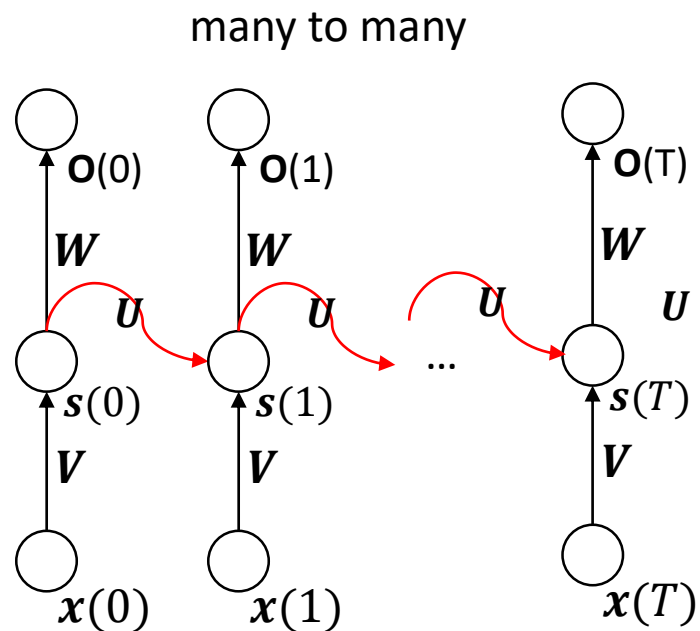
RNN



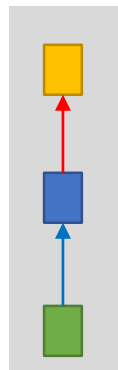
Recurrent Neural Network (RNN)



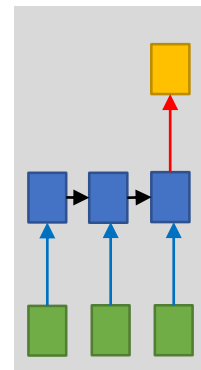
RNN



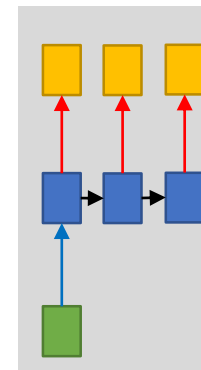
one to one



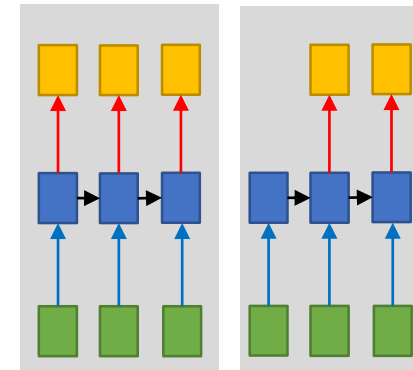
many to one



one to many



many to many

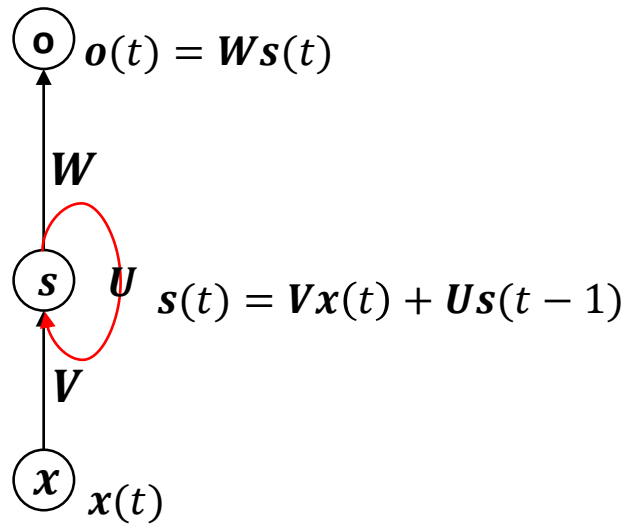


在RNN訓練期間，因為梯度會傳遞到最前一層做乘積，在前面課程提過層數一多可能會發生梯度消失和爆炸的問題，在RNN也是，傳遞不但跟層數相關和時間也相關，可以想像時間等於層數的概念，因此在RNN是很可能發生梯度消失/爆炸。

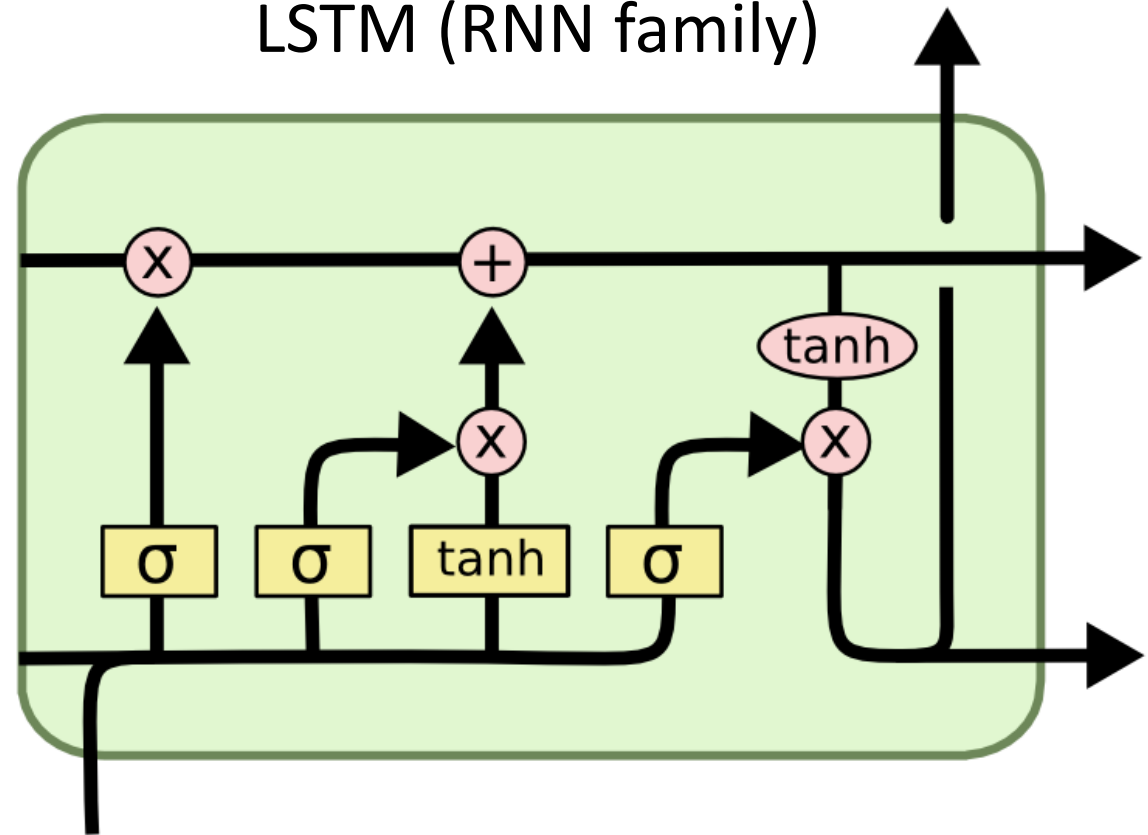


Long Short Term Memory

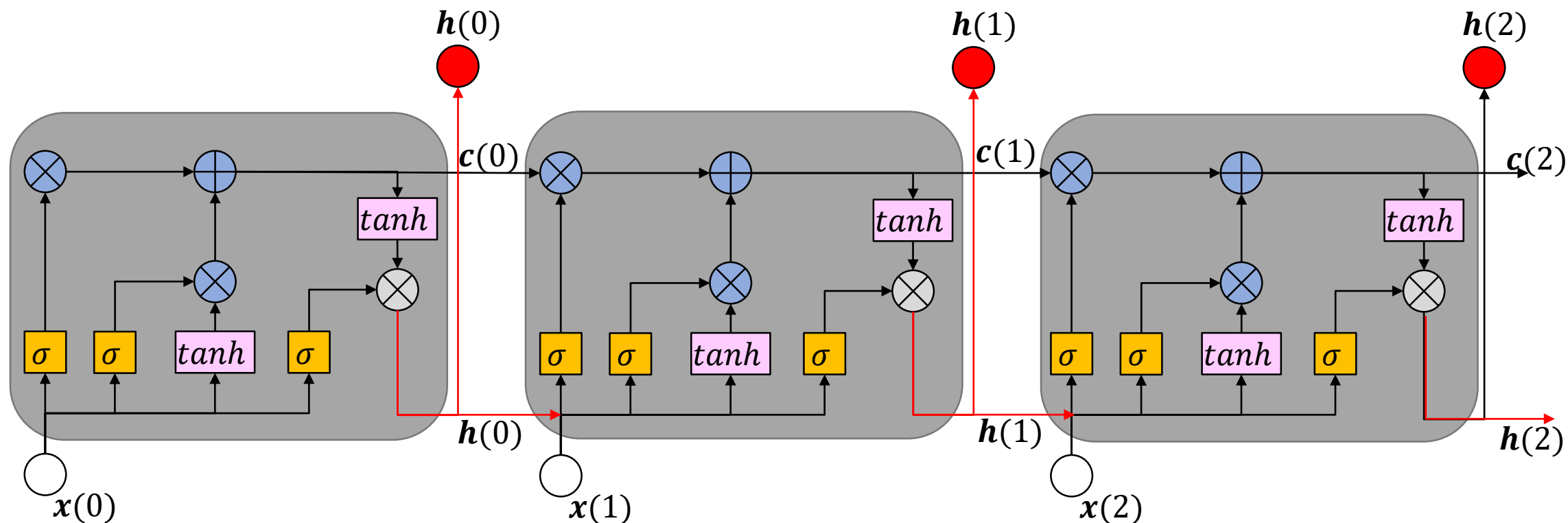
RNN



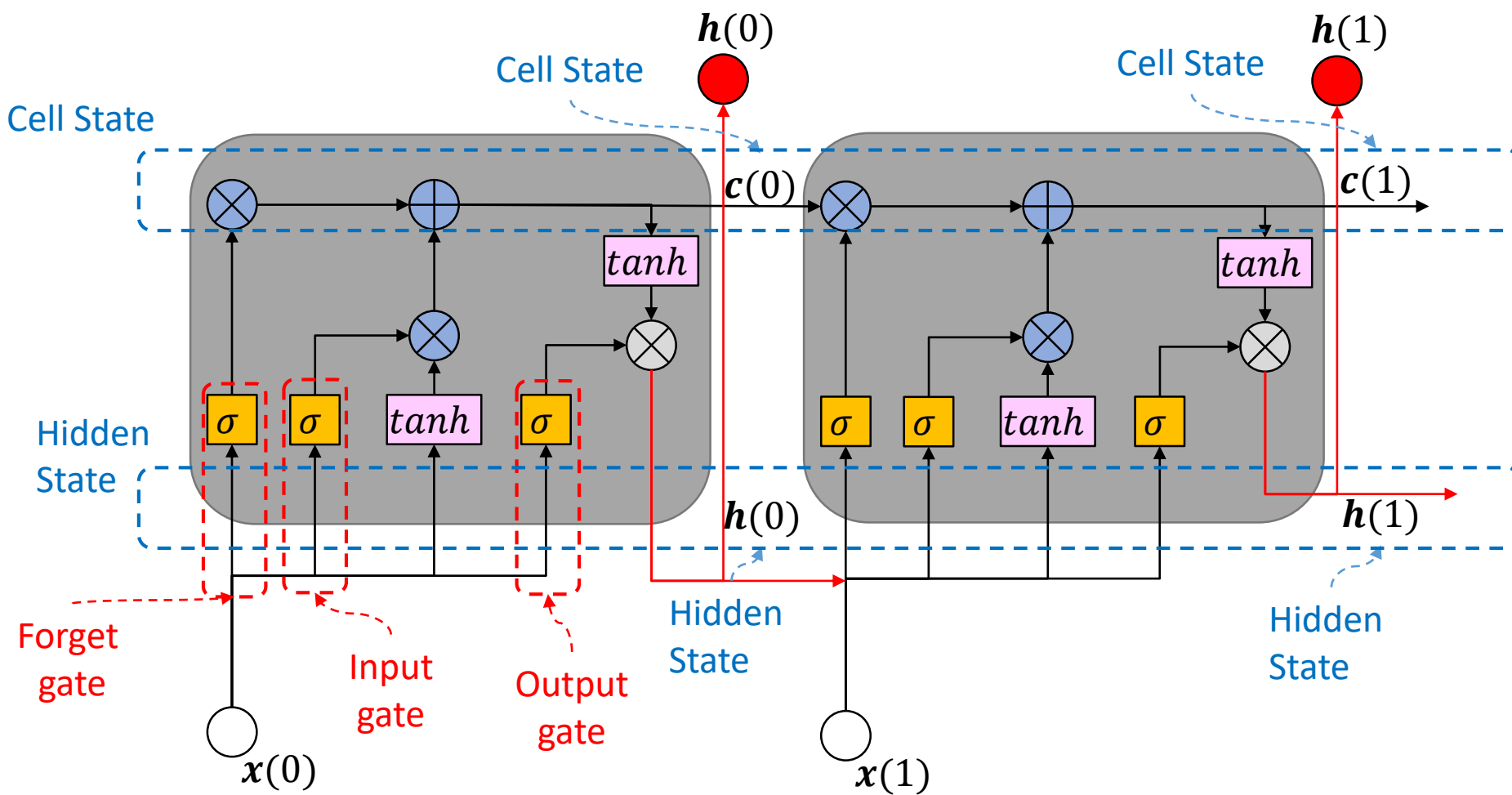
LSTM (RNN family)



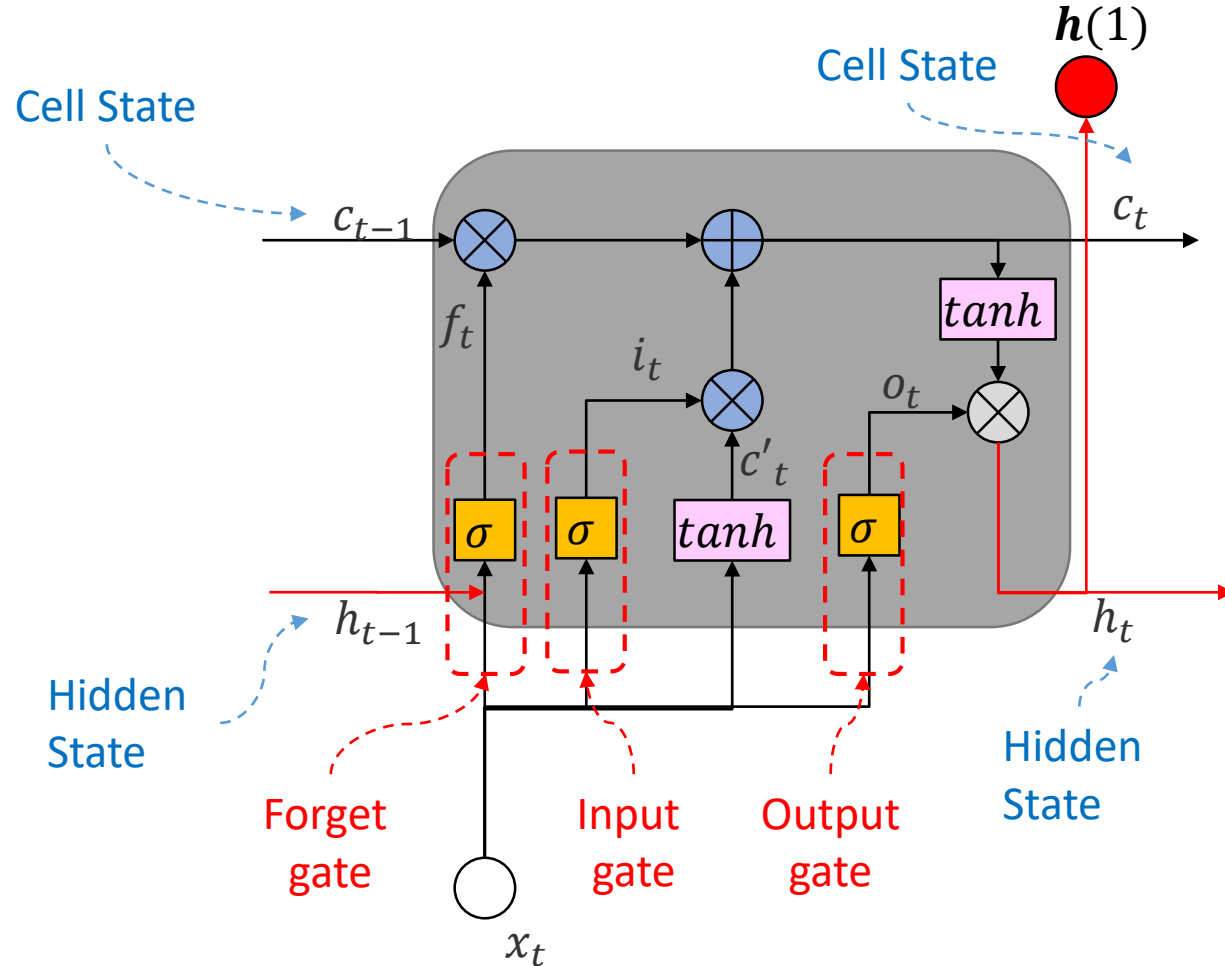
Long Short Term Memory



Long Short Term Memory



Long Short Term Memory



forget gate

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Input gate

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

Cell State

$$c'_t = \tanh(W_c x_t + U_c h_{t-1})$$

$$c_t = (f_t c_{t-1} + i_t c'_t)$$

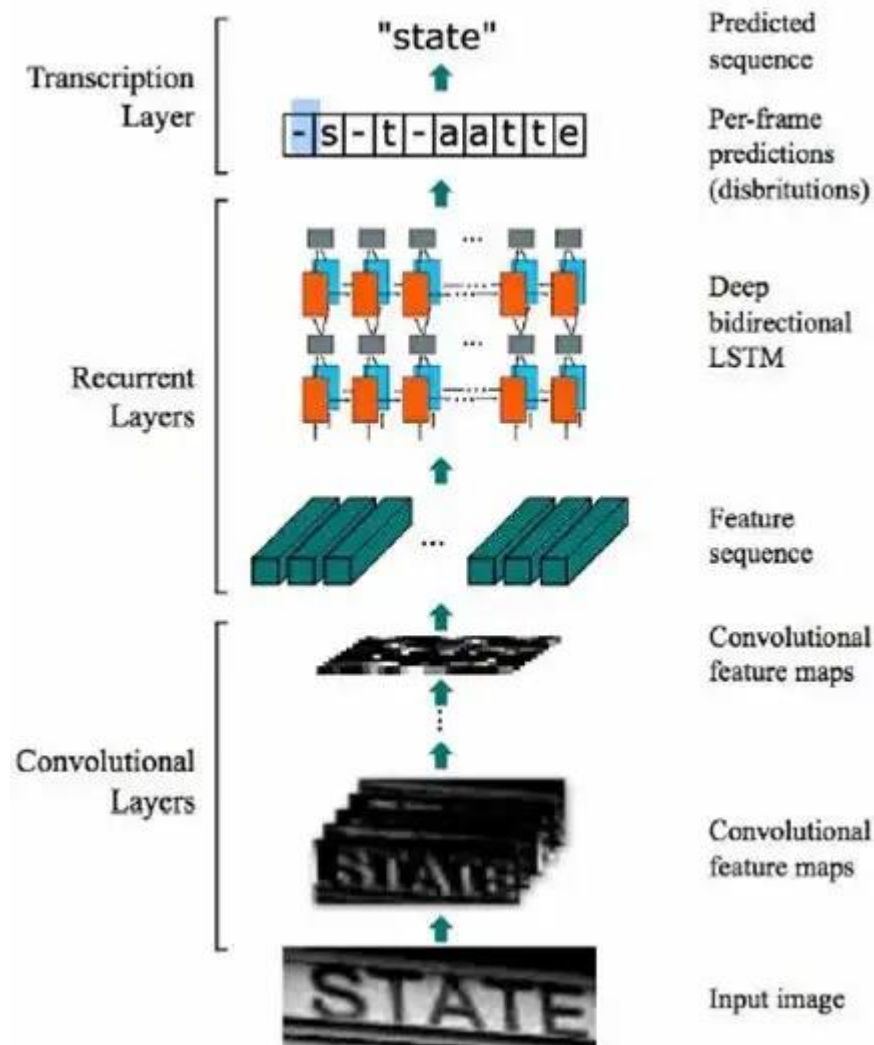
Output gate

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \tanh(c_t)$$

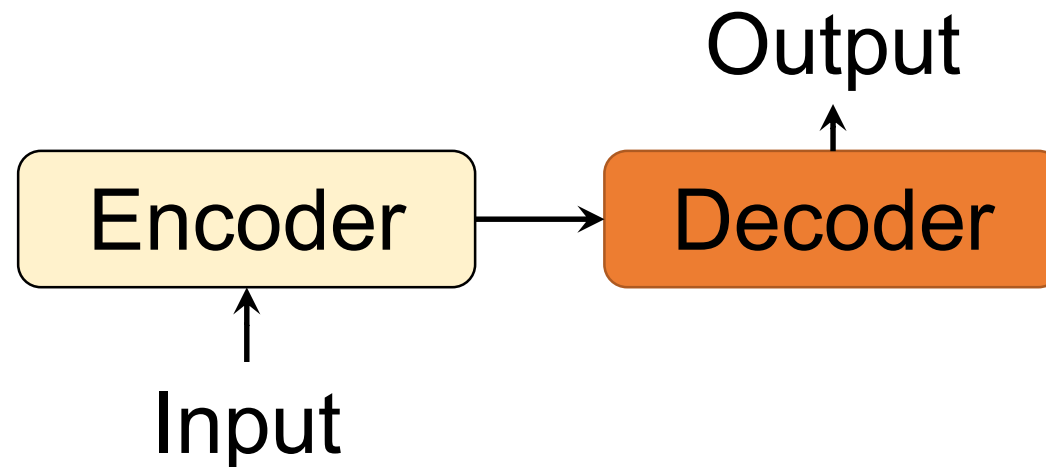


Convolutional Recurrent Neural Network

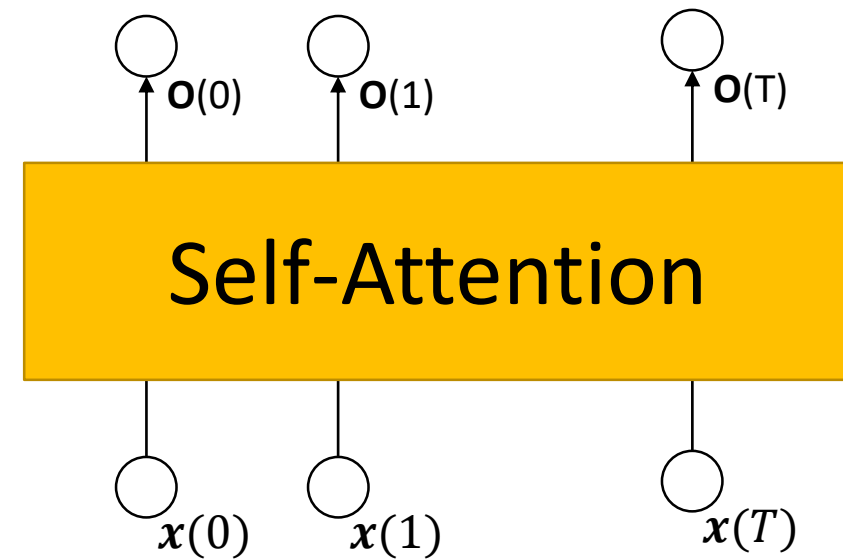
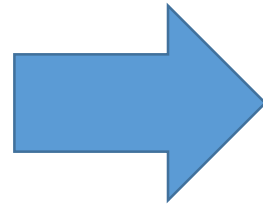
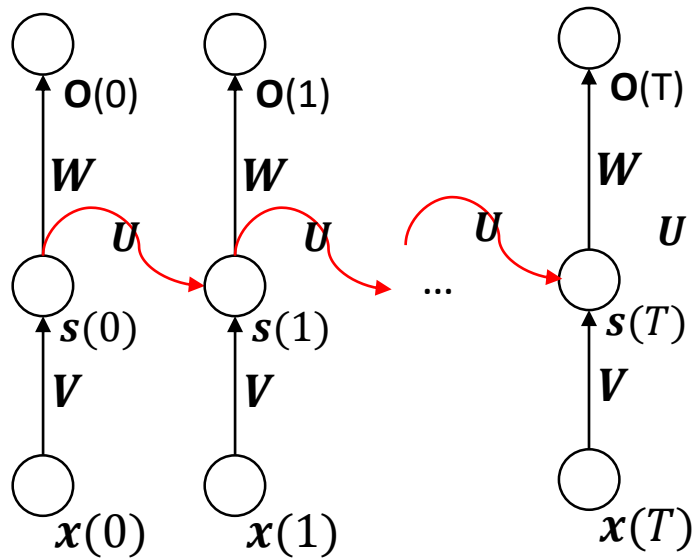


Transformer (Seq2Seq)

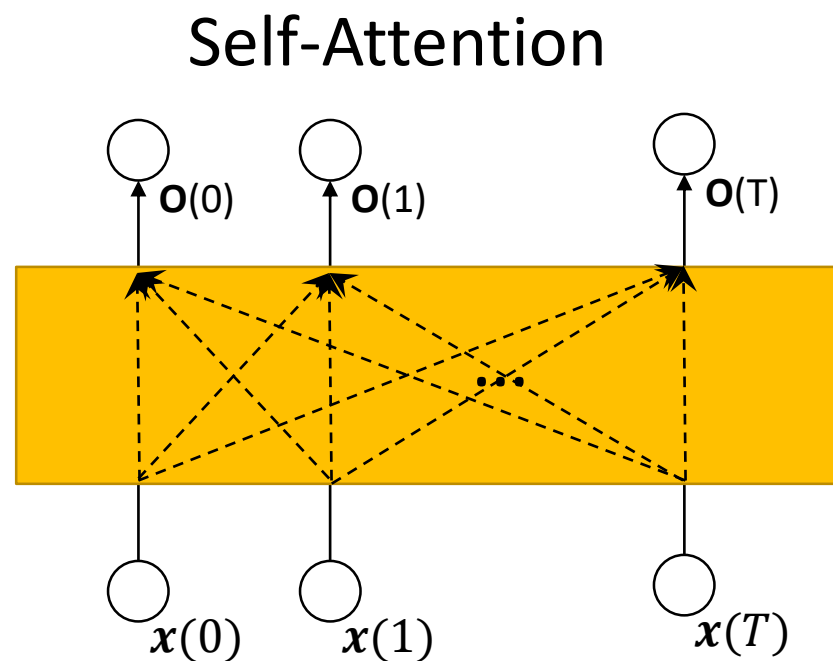
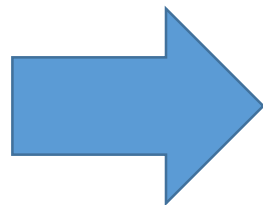
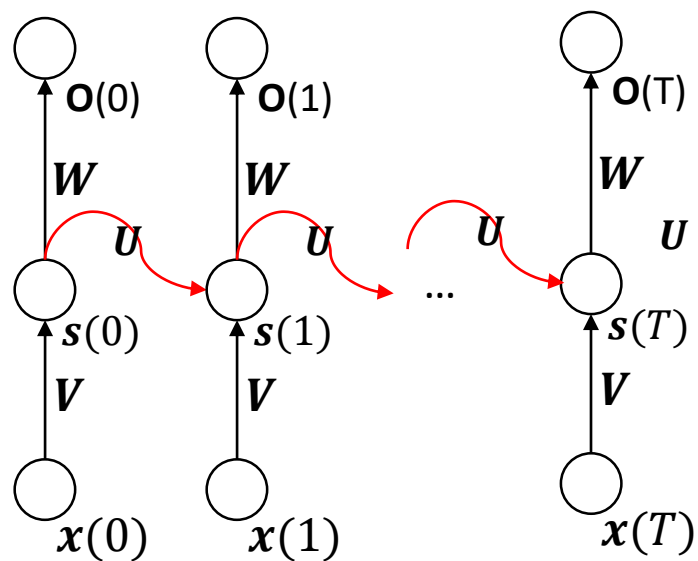
- Sequence-to-sequence: Seq2Seq
- Encoder: takes the input sequence and maps it into a higher dimensional space (n-dimensional vector).
- That abstract vector is fed into the Decoder which turns it into an output sequence.



Transformer: Self-Attention



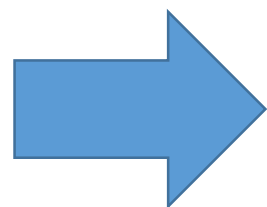
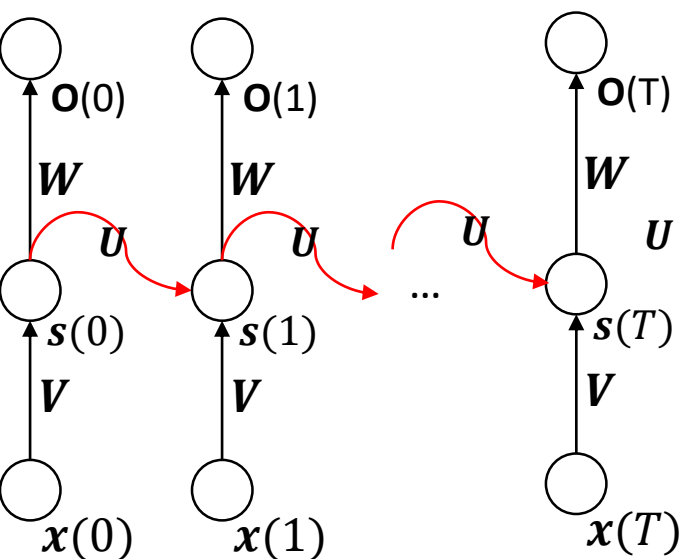
Transformer: Self-Attention



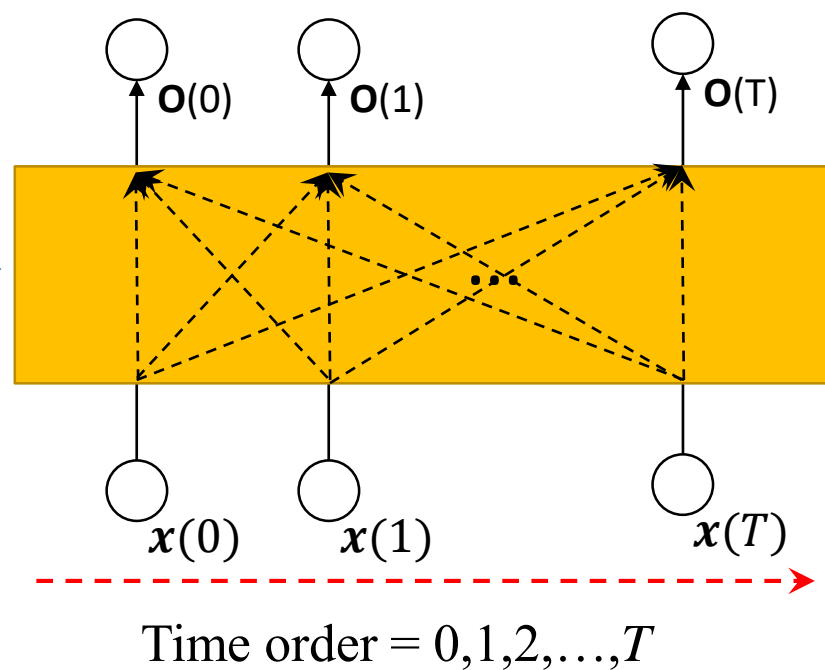
MLP?



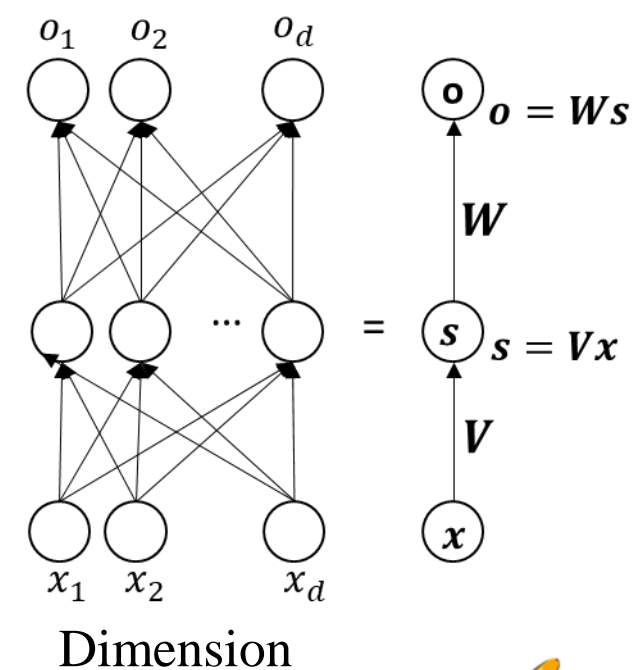
Transformer: Self-Attention



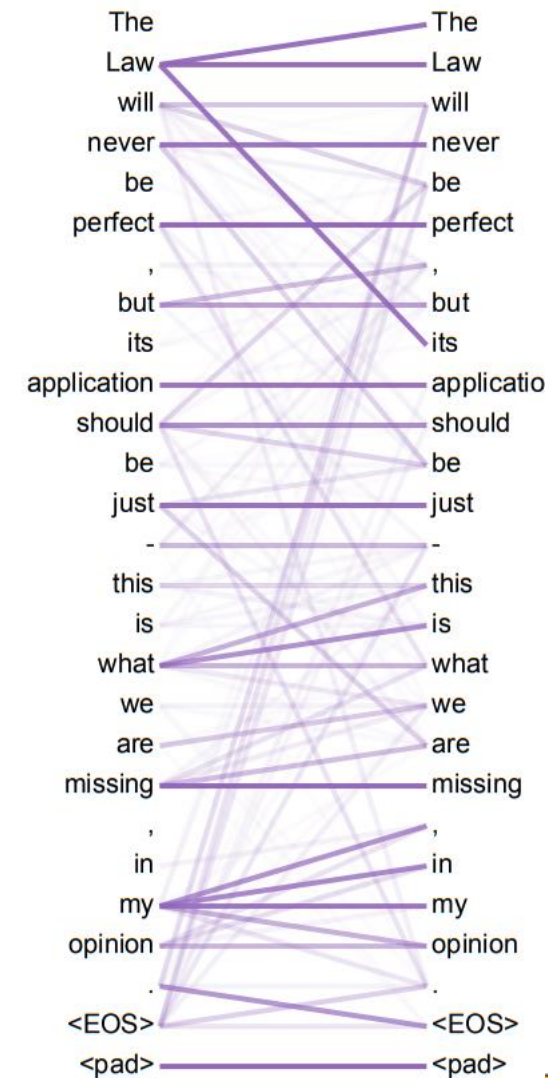
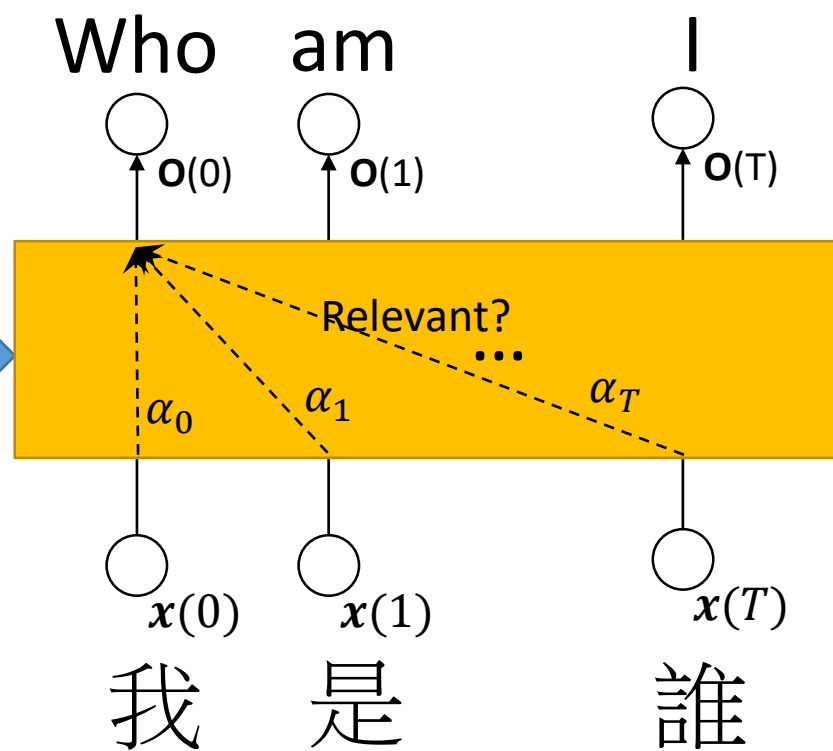
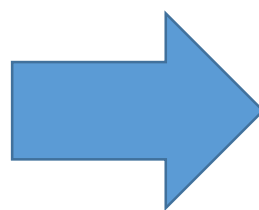
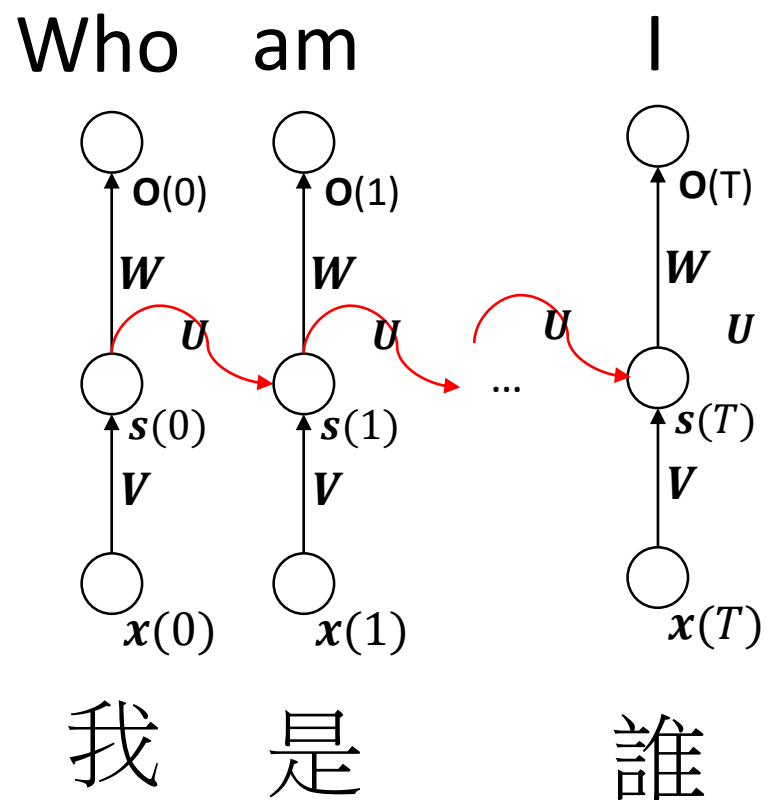
Self-Attention



MLP



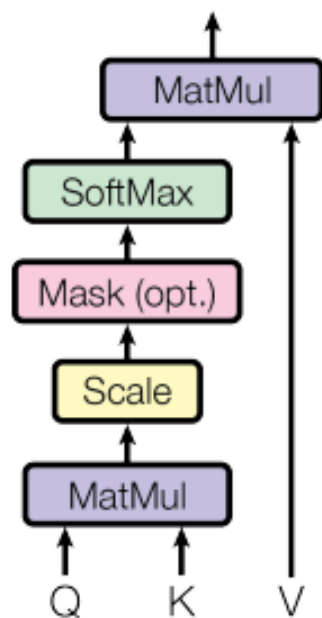
Transformer: Self-Attention



Transformer: Self-Attention

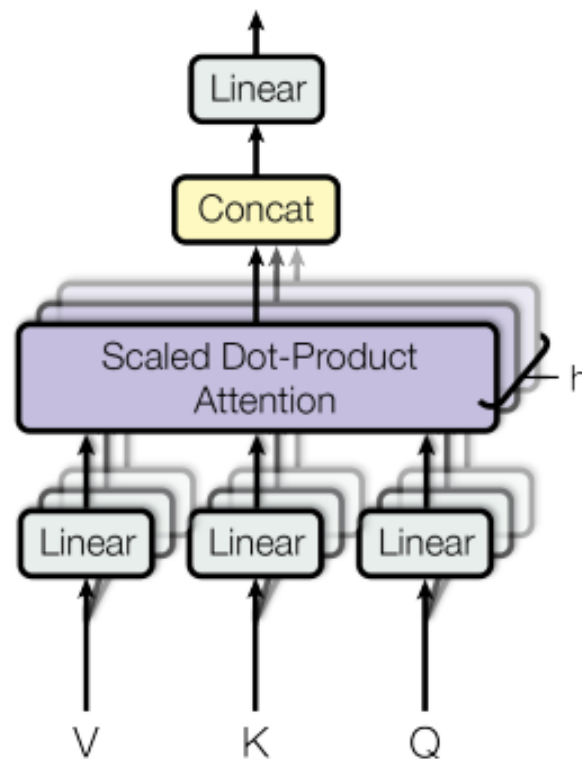
Single-Head Attention

Scaled Dot-Product Attention

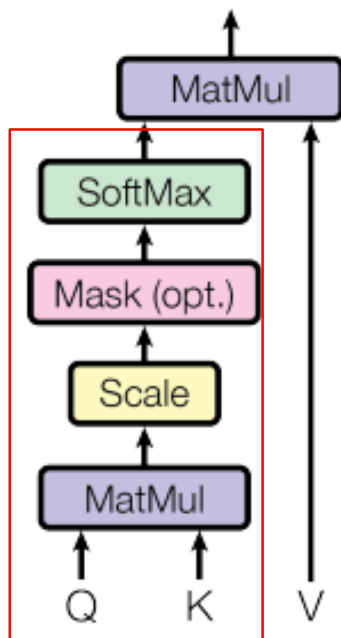
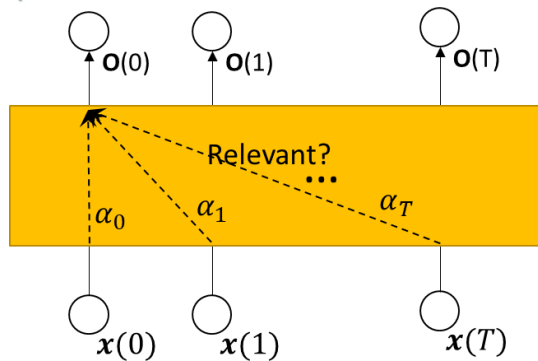


Multi-Head Attention

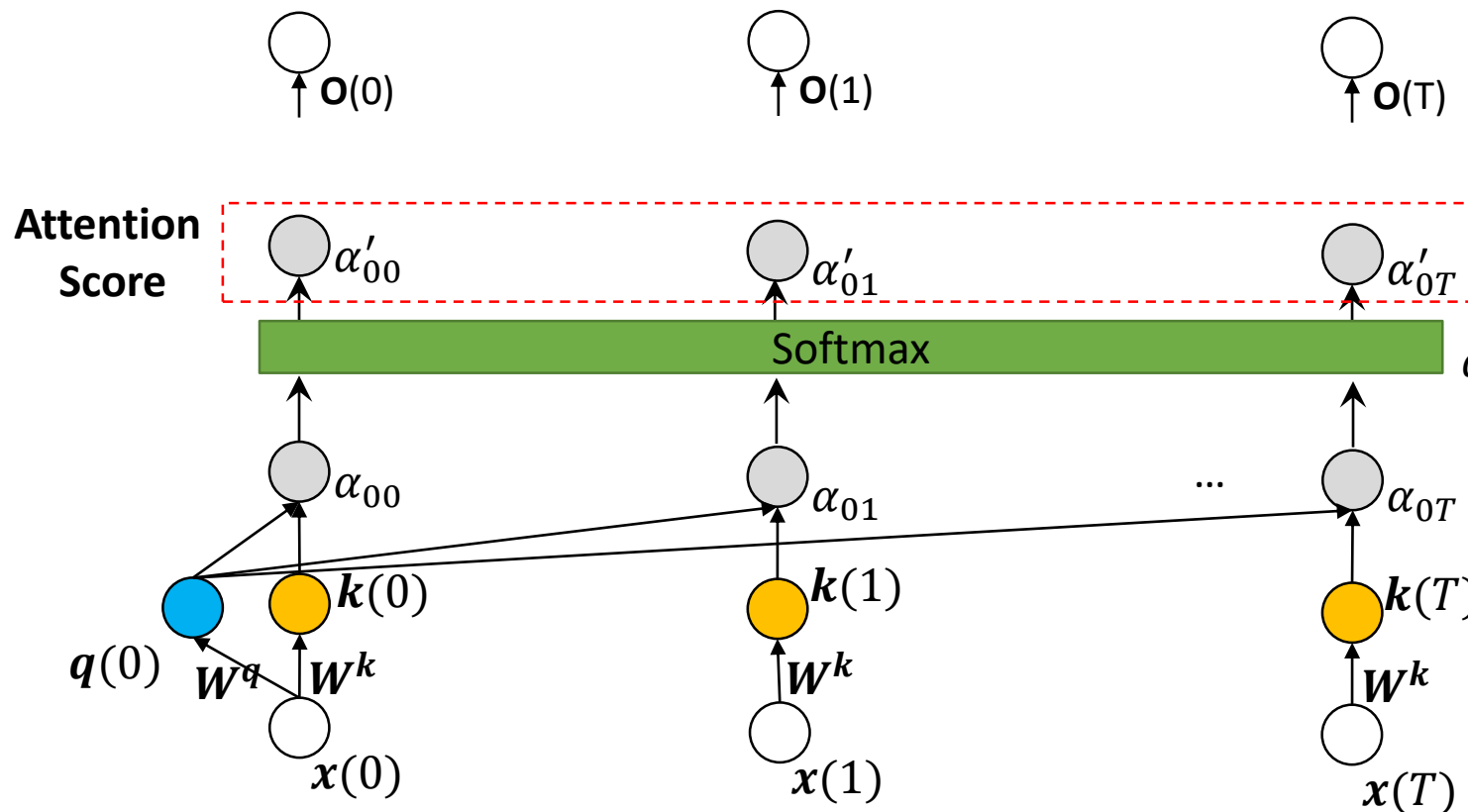
Multi-Head Attention



Single-Head Self-Attention



Q: Query
K: Key
V: Value



$$\begin{aligned} q(0) &= W^q x(0) \\ k(0) &= W^k x(0) \\ \alpha_{00} &= \langle q(0), k(0) \rangle \end{aligned}$$

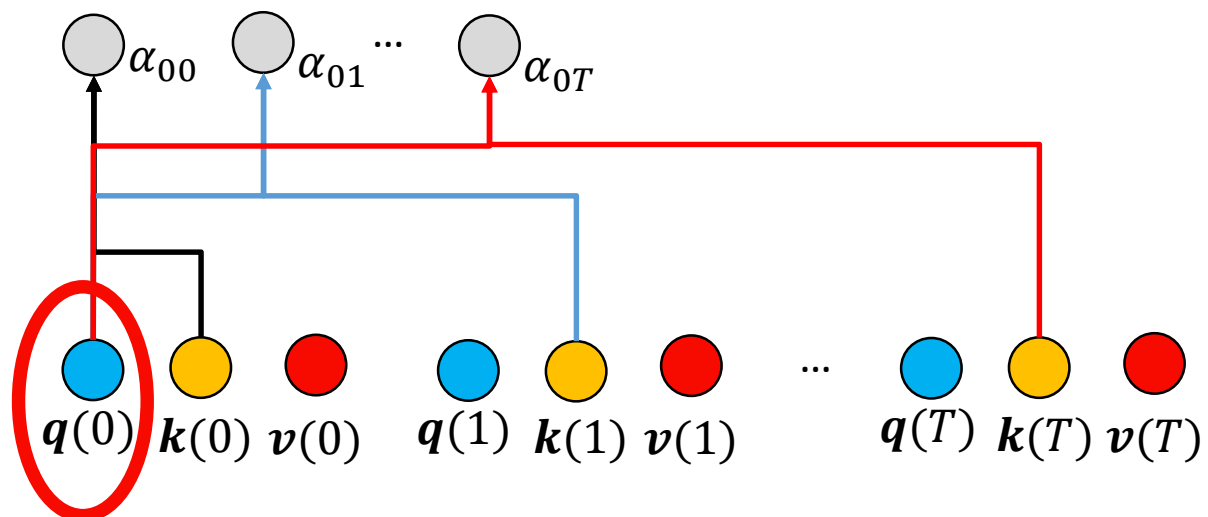
$$\begin{aligned} k(1) &= W^k x(1) \\ \alpha_{01} &= \langle q(0), k(1) \rangle \end{aligned}$$

$$\begin{aligned} k(T) &= W^k x(T) \\ \alpha_{0T} &= \langle q(0), k(T) \rangle \end{aligned}$$

$$\alpha'_{0i} = \frac{\exp(\alpha_{0i})}{\sum_{j=0}^T \exp(\alpha_{0j})} \quad \forall i = 0, 1, \dots, T$$



Attention Score



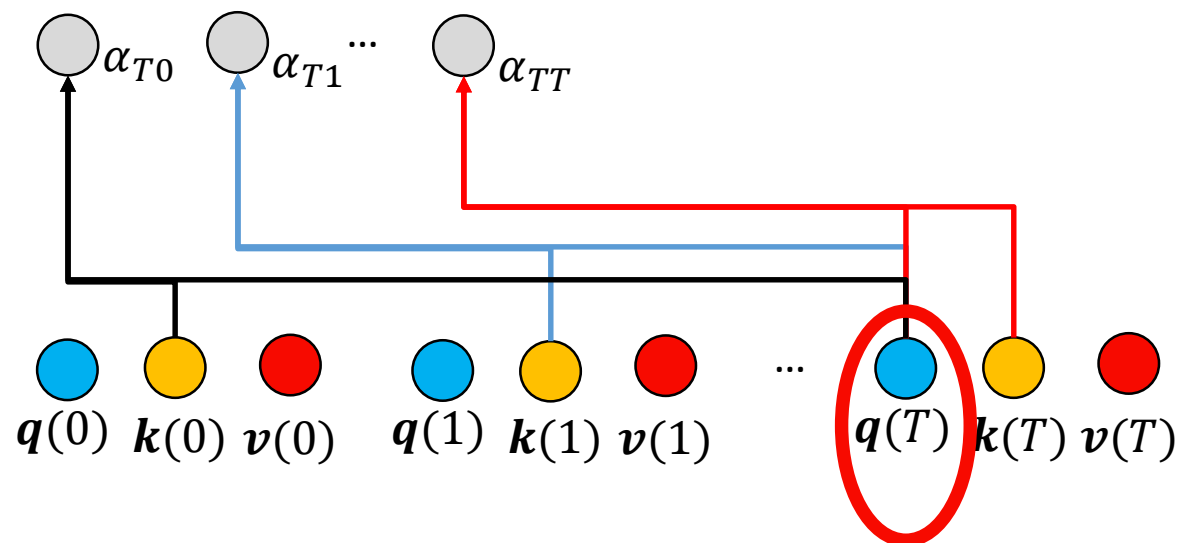
$$\alpha_{00} = \langle \mathbf{q}(0), \mathbf{k}(0) \rangle$$

$$\alpha_{01} = \langle \mathbf{q}(0), \mathbf{k}(1) \rangle$$

...

$$\alpha_{0T} = \langle \mathbf{q}(0), \mathbf{k}(T) \rangle$$

Attention Score at $\mathbf{q}(0)$



$$\alpha_{T0} = \langle \mathbf{q}(T), \mathbf{k}(0) \rangle$$

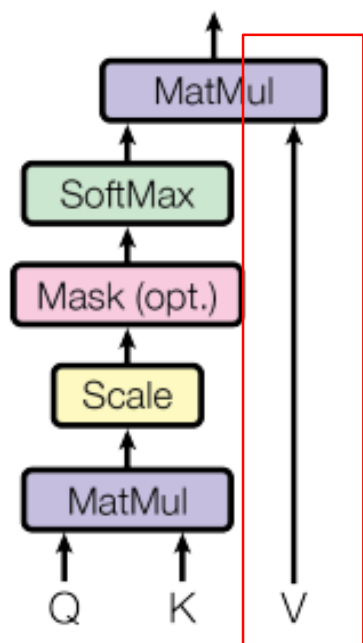
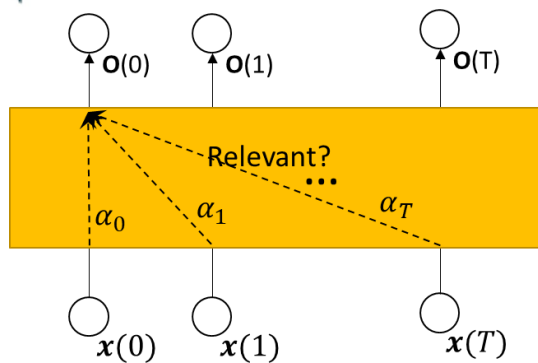
$$\alpha_{T1} = \langle \mathbf{q}(T), \mathbf{k}(1) \rangle$$

...

$$\alpha_{TT} = \langle \mathbf{q}(T), \mathbf{k}(T) \rangle$$

Attention Score at $\mathbf{q}(T)$

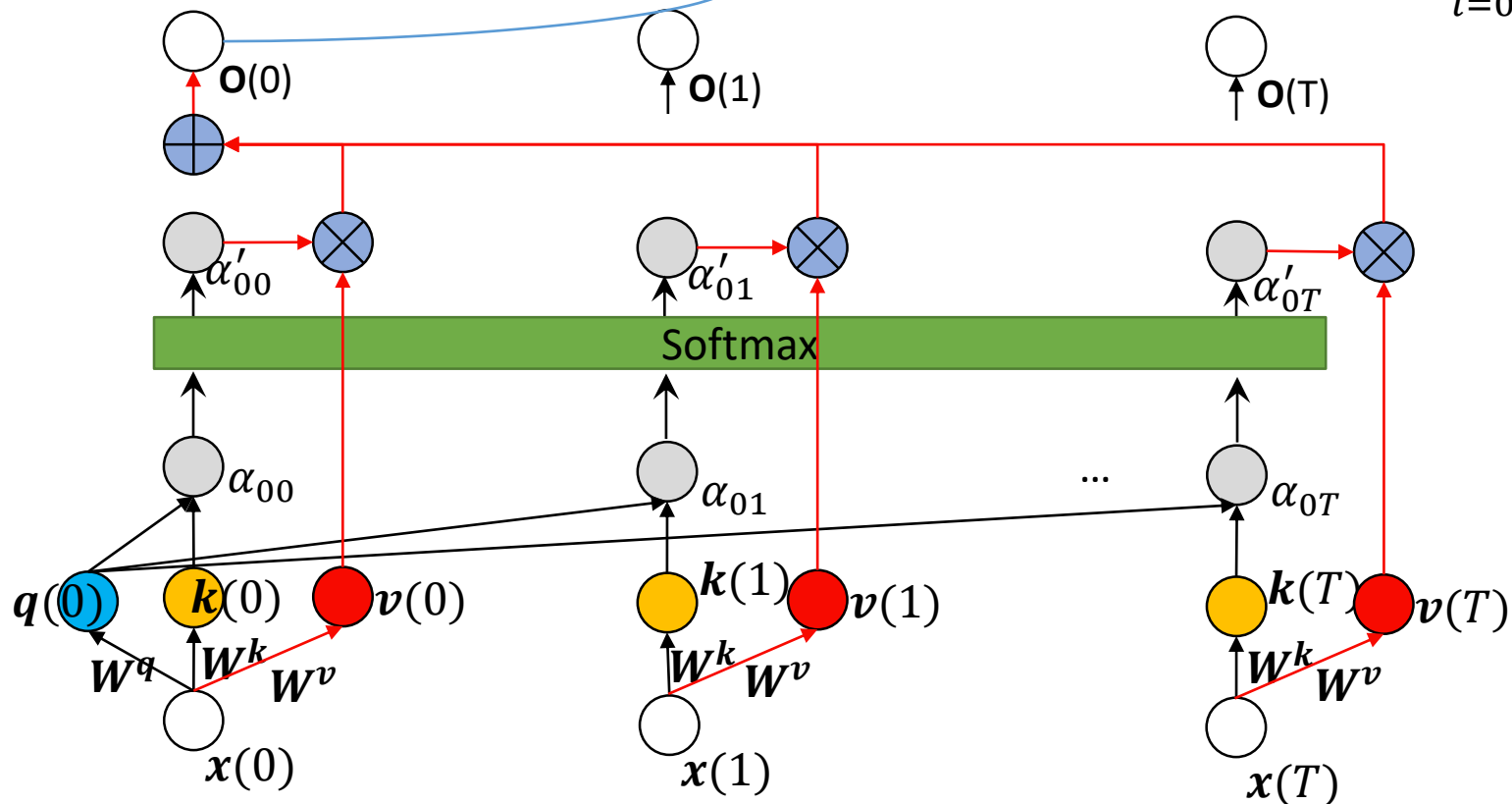




Q: Query

K: Key

V: Value



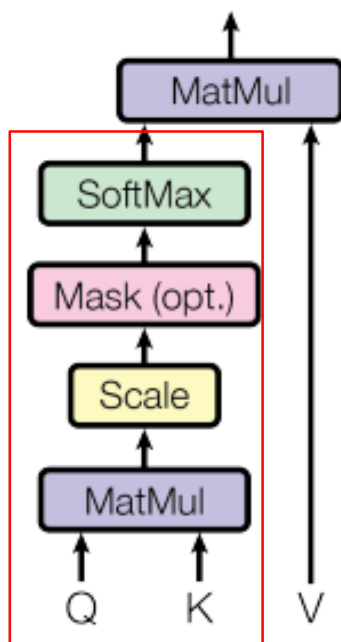
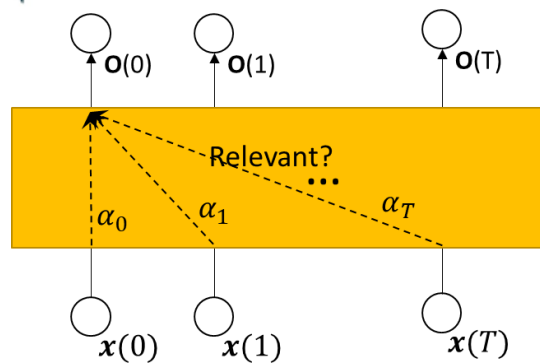
$$\mathbf{o}(0) = \sum_{i=0}^T \alpha'_{0i} v(i)$$

$$\begin{aligned} q(0) &= W^q x(0) \\ k(0) &= W^k x(0) \\ v(0) &= W^v x(0) \\ \alpha_{00} &= \langle q(0), k(0) \rangle \end{aligned}$$

$$\begin{aligned} k(1) &= W^k x(1) \\ v(1) &= W^v x(1) \\ \alpha_{01} &= \langle q(0), k(1) \rangle \end{aligned}$$

$$\begin{aligned} k(T) &= W^k x(T) \\ v(T) &= W^v x(T) \\ \alpha_{0T} &= \langle q(0), k(T) \rangle \end{aligned}$$

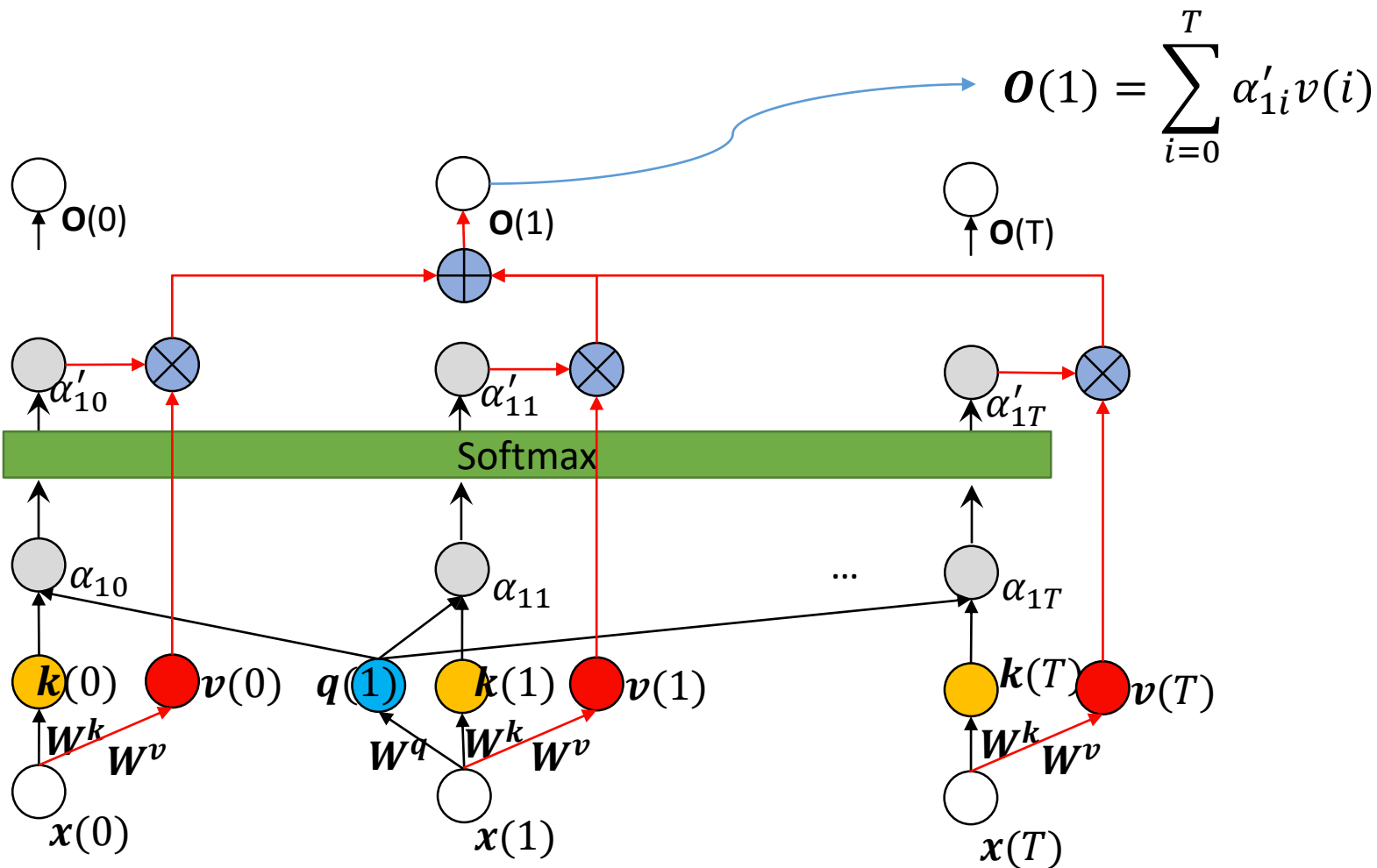




Q: Query

K: Key

V: Value



$$k(0) = W^k x(0)$$

$$v(0) = W^v x(0)$$

$$\alpha_{10} = \langle q(1), k(0) \rangle$$

$$q(1) = W^q x(1)$$

$$k(1) = W^k x(1)$$

$$v(1) = W^v x(1)$$

$$\alpha_{11} = \langle q(1), k(1) \rangle$$

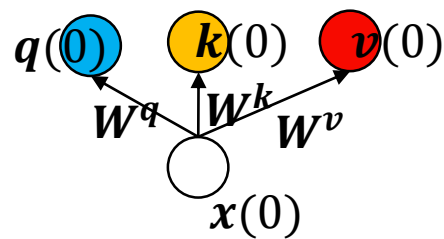
$$k(T) = W^k x(T)$$

$$v(T) = W^v x(T)$$

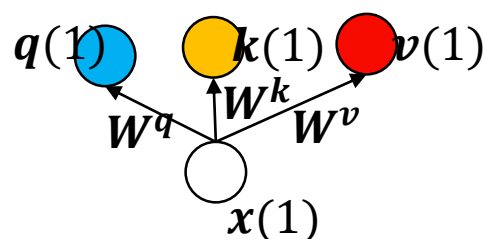
$$\alpha_{1T} = \langle q(1), k(T) \rangle$$



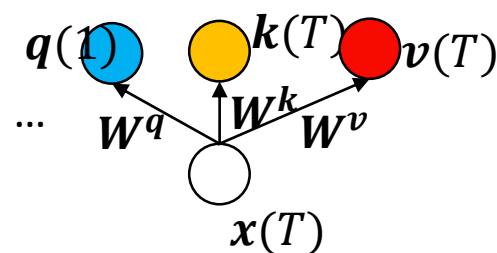
用大矩陣直接運算QKV



$$\begin{aligned} q(0) &= W^q x(0) \\ k(0) &= W^k x(0) \\ v(0) &= W^v x(0) \end{aligned}$$



$$\begin{aligned} q(1) &= W^q x(1) \\ k(1) &= W^k x(1) \\ v(1) &= W^v x(1) \end{aligned}$$



$$\begin{aligned} q(T) &= W^q x(T) \\ k(T) &= W^k x(T) \\ v(T) &= W^v x(T) \end{aligned}$$



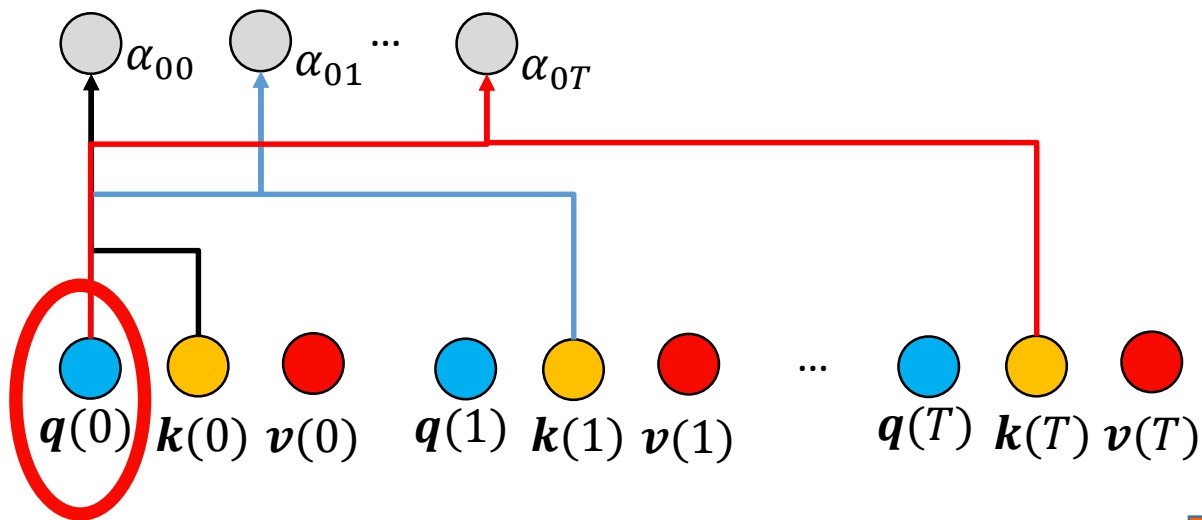
$$\mathbf{Q}_{d \times T} = \begin{bmatrix} q(0) & q(1) & \dots & q(T) \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T} = \mathbf{W}^q_{d \times d} \begin{bmatrix} x(0) & x(1) & \dots & x(T) \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T}$$

$$\mathbf{K}_{d \times T} = \begin{bmatrix} k(0) & k(1) & \dots & k(T) \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T} = \mathbf{W}^k_{d \times d} \begin{bmatrix} x(0) & x(1) & \dots & x(T) \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T}$$

$$\mathbf{V}_{d \times T} = \begin{bmatrix} v(0) & v(1) & \dots & v(T) \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T} = \mathbf{W}^v_{d \times d} \begin{bmatrix} x(0) & x(1) & \dots & x(T) \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T}$$



用大矩陣直接運算A



$$\alpha_{00} = \langle \mathbf{q}(0), \mathbf{k}(0) \rangle$$

$$\alpha_{01} = \langle \mathbf{q}(0), \mathbf{k}(1) \rangle$$

...

$$\alpha_{0T} = \langle \mathbf{q}(0), \mathbf{k}(T) \rangle$$

Attention Score at $\mathbf{q}(0)$



$$\begin{bmatrix} \alpha_{00} & \alpha_{10} & \dots & \alpha_{T0} \\ \alpha_{01} & \alpha_{11} & \dots & \alpha_{T1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{0T} & \alpha_{1T} & \dots & \alpha_{TT} \end{bmatrix} \quad \text{Attention Score Matrix}$$

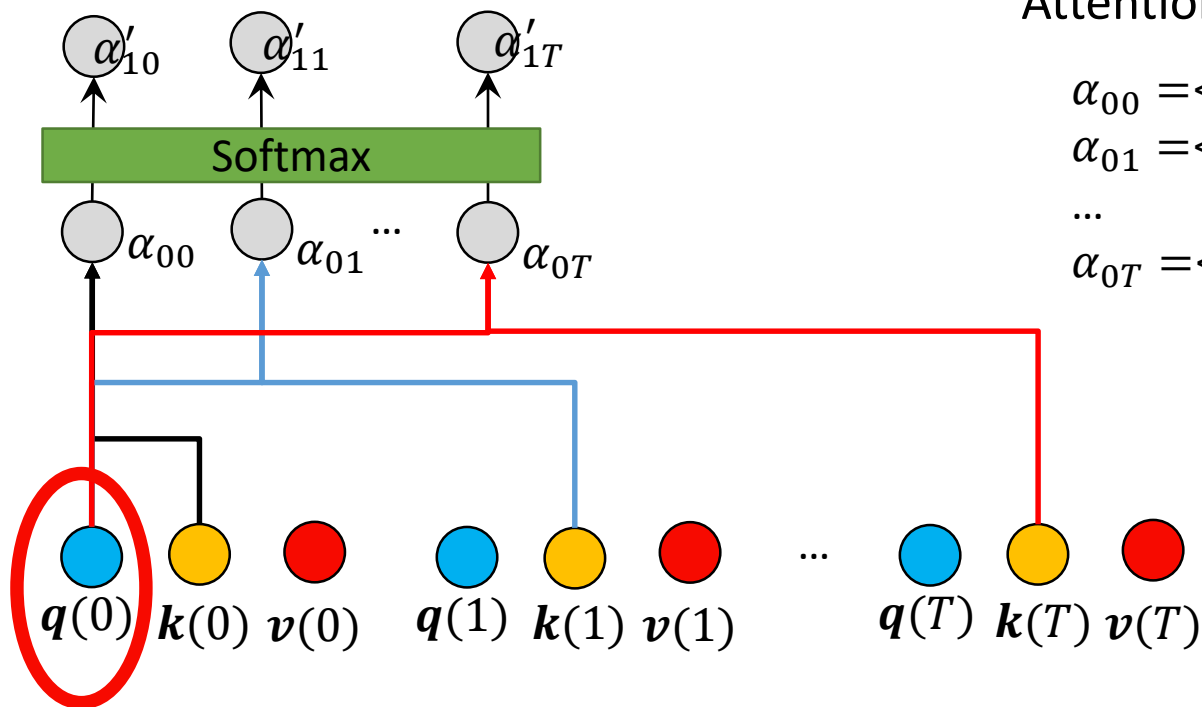
$$= \underset{T \times T}{\mathbf{A}} = \langle \underset{T \times d}{\mathbf{Q}}, \underset{d \times T}{\mathbf{K}} \rangle = \underset{T \times d}{\mathbf{K}^T} \underset{d \times T}{\mathbf{Q}} =$$

$$\begin{bmatrix} \mathbf{k}(0)_{d \times 1} \\ \mathbf{k}(1)_{d \times 1} \\ \vdots \\ \mathbf{k}(T)_{d \times 1} \end{bmatrix} \quad \begin{bmatrix} \mathbf{q}(0)_{d \times 1} & \mathbf{q}(1)_{d \times 1} & \dots & \mathbf{q}(T)_{d \times 1} \end{bmatrix}$$

$$\underset{T \times d}{\mathbf{K}^T} \quad \underset{d \times T}{\mathbf{Q}}$$

Attention Score at $\mathbf{q}(0)$





Attention Score at $q(0)$

$$\begin{aligned} \alpha_{00} &= \langle \mathbf{q}(0), \mathbf{k}(0) \rangle \\ \alpha_{01} &= \langle \mathbf{q}(0), \mathbf{k}(1) \rangle \\ &\vdots \\ \alpha_{0T} &= \langle \mathbf{q}(0), \mathbf{k}(T) \rangle \end{aligned} \xrightarrow{\text{Softmax}} \alpha'_{0i} = \frac{\exp(\alpha_{0i})}{\sum_{j=0}^T \exp(\alpha_{0j})} \quad \forall i = 0, 1, \dots, T$$

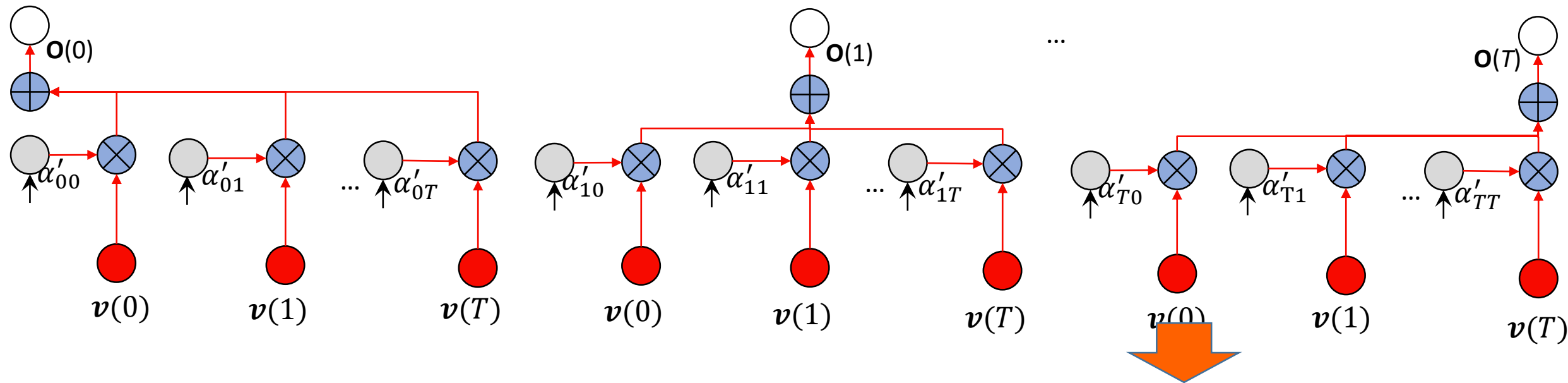
$$\begin{bmatrix} \alpha_{00} & \alpha_{10} & \dots & \alpha_{T0} \\ \alpha_{01} & \alpha_{11} & \dots & \alpha_{T1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{0T} & \alpha_{1T} & \dots & \alpha_{TT} \end{bmatrix} = \mathbf{A}_{T \times T} \xrightarrow[\text{For each column}]{\text{Softmax}} \hat{\mathbf{A}}_{T \times T} = \begin{bmatrix} \alpha'_{00} & \alpha'_{10} & \dots & \alpha'_{T0} \\ \alpha'_{01} & \alpha'_{11} & \dots & \alpha'_{T1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha'_{0T} & \alpha'_{1T} & \dots & \alpha'_{TT} \end{bmatrix}$$

$$\hat{\mathbf{A}} = \text{softmax}(\mathbf{A})$$

Attention Score at $q(0)$



用大矩陣直接運算O



$$\mathbf{O} = \begin{bmatrix} \boxed{o(0)} & \boxed{o(1)} & \cdots & \boxed{o(T)} \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T} = \mathbf{V} \hat{\mathbf{A}} = \begin{bmatrix} \boxed{v(0)} & \boxed{v(1)} & \cdots & \boxed{v(T)} \\ d \times 1 & d \times 1 & & d \times 1 \end{bmatrix}_{d \times T} \begin{bmatrix} \alpha'_{00} & \alpha'_{10} & \cdots & \alpha'_{T0} \\ \alpha'_{01} & \alpha'_{11} & \cdots & \alpha'_{T1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha'_{0T} & \alpha'_{1T} & \cdots & \alpha'_{TT} \end{bmatrix}_{T \times T}$$



矩陣直接運算Self-Attention

$$\begin{aligned}
 \mathbf{Q} &= \begin{bmatrix} \boxed{q(0)} & \boxed{q(1)} & \cdots & \boxed{q(T)} \end{bmatrix}_{d \times T} = \begin{bmatrix} W^q \end{bmatrix}_{d \times d} \begin{bmatrix} \boxed{x(0)} & \boxed{x(1)} & \cdots & \boxed{x(T)} \end{bmatrix}_{d \times T} \\
 \mathbf{K} &= \begin{bmatrix} \boxed{k(0)} & \boxed{k(1)} & \cdots & \boxed{k(T)} \end{bmatrix}_{d \times T} = \begin{bmatrix} W^k \end{bmatrix}_{d \times d} \begin{bmatrix} \boxed{x(0)} & \boxed{x(1)} & \cdots & \boxed{x(T)} \end{bmatrix}_{d \times T} \\
 \mathbf{V} &= \begin{bmatrix} \boxed{v(0)} & \boxed{v(1)} & \cdots & \boxed{v(T)} \end{bmatrix}_{d \times T} = \begin{bmatrix} W^v \end{bmatrix}_{d \times d} \begin{bmatrix} \boxed{x(0)} & \boxed{x(1)} & \cdots & \boxed{x(T)} \end{bmatrix}_{d \times T}
 \end{aligned}$$

$$\mathbf{A} = \langle \mathbf{Q}, \mathbf{K} \rangle = \mathbf{K}^T \mathbf{Q}$$

$T \times T$
 $T \times d$
 $d \times T$

$$\hat{\mathbf{A}} = \text{softmax}(\mathbf{A})$$

$$\mathbf{O} = \mathbf{V} \hat{\mathbf{A}}$$

$d \times T$

論文寫法

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

$$\mathbf{O} = \mathbf{V} \times \text{softmax}(\mathbf{K}^T \mathbf{Q})$$

差異在矩陣擺法，和一個 $\sqrt{d_k}$



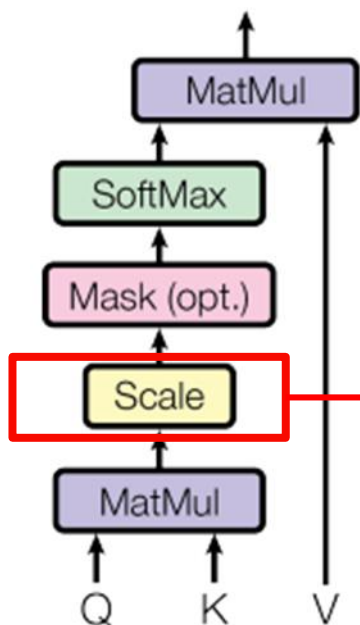
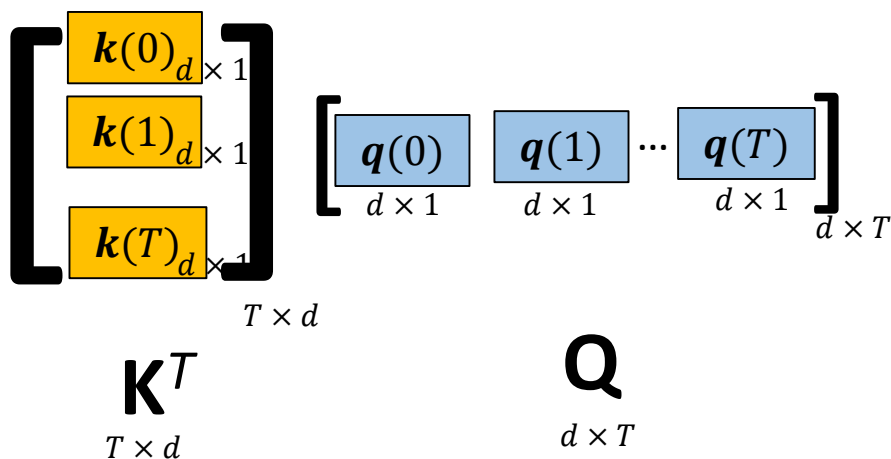
矩陣直接運算Self-Attention

論文寫法

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

$$O = V \times \text{softmax}(K^T Q)$$

差異在矩陣擺法，和一個 $\sqrt{d_k}$



K和Q在不同時間點上

d個維度相乘後的總和。

所以維度越大值越大，因此需要正規化(在右圖的Scale)，將每一個值域拉到一致。

最簡單方式有d個維度就除上d

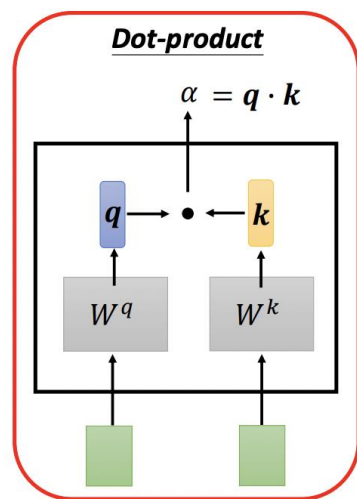
但論文第四頁內提到當維度很大的時候，預防梯度過小的問題，因此開根號。



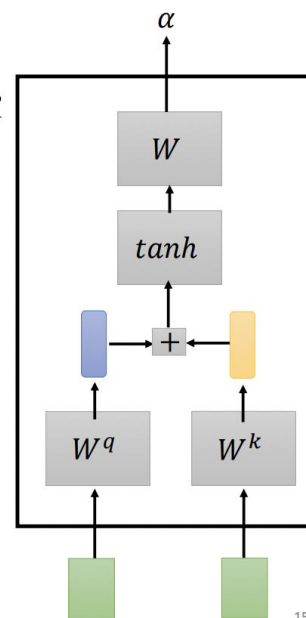
論文提到的Self-Attention

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

Self-attention



Additive

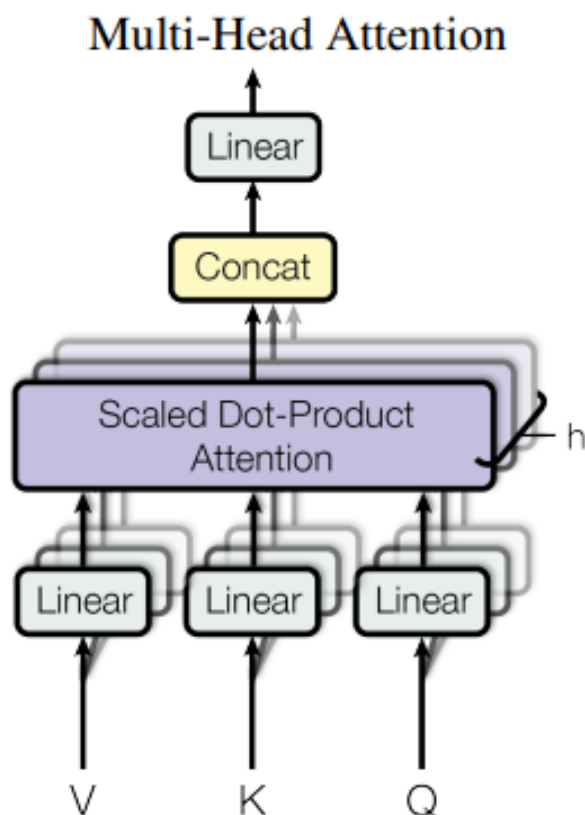


Reference: 李宏毅老師



Multi-head Self-attention

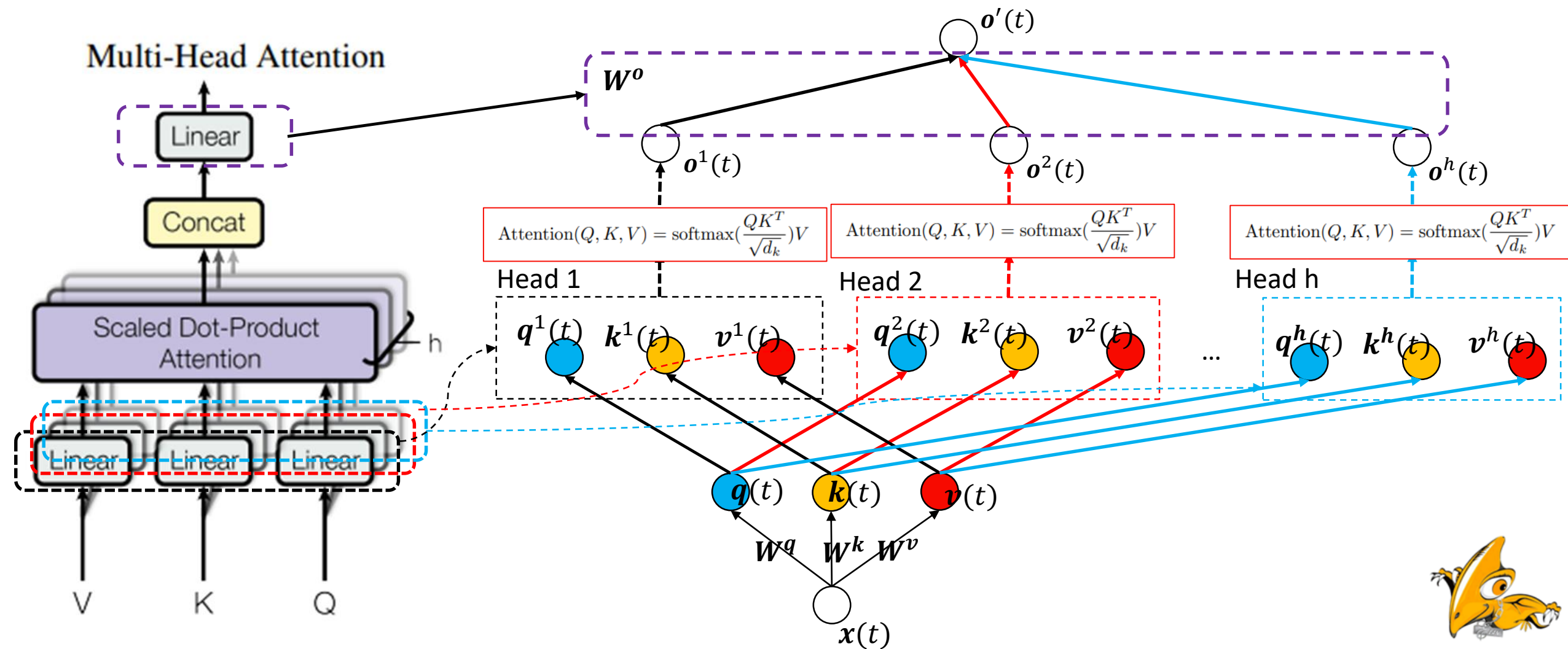
- 相關性在不同的位置可能有不同類別，所以需要多Multi-head來讓模型可以得到更多不同位置的表示。



Many of the attention heads exhibit behaviour that seems related to the structure of the sentence.

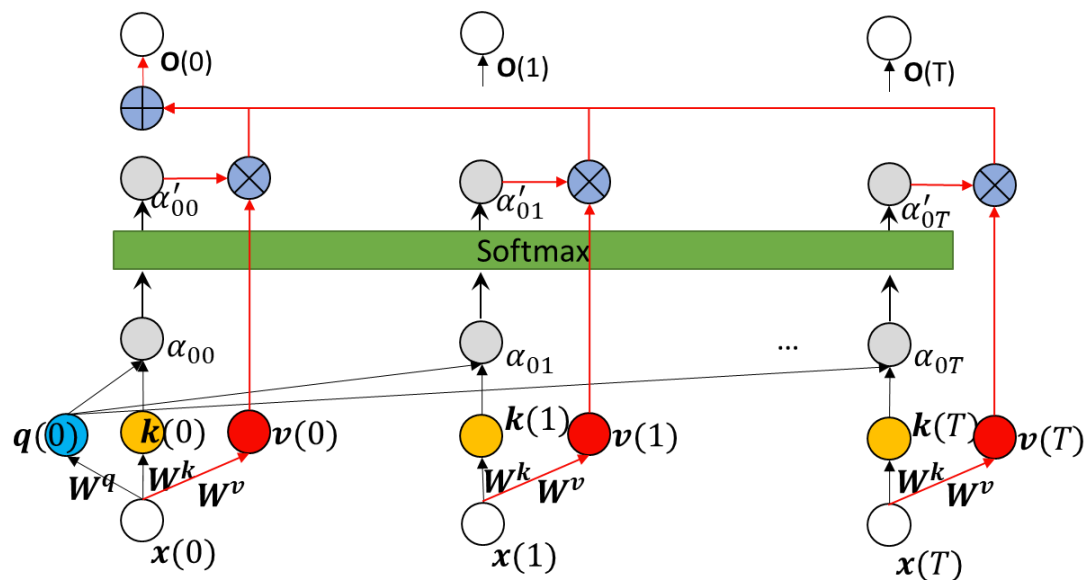


Multi-head Self-attention



Positional Encoding

輸入是有訊續的 $\mathbf{x}(0), \mathbf{x}(1), \dots, \mathbf{x}(T)$ ，但在前面介紹的Attention是沒有考慮順序性的。



因為架構沒有順序(位置)資料，所以在Transformer中每一個位置給予一個positional vector($\mathbf{e}_i, \forall i = 0, 1, \dots, T$)，整個過程稱為position encoding。



Positional Encoding

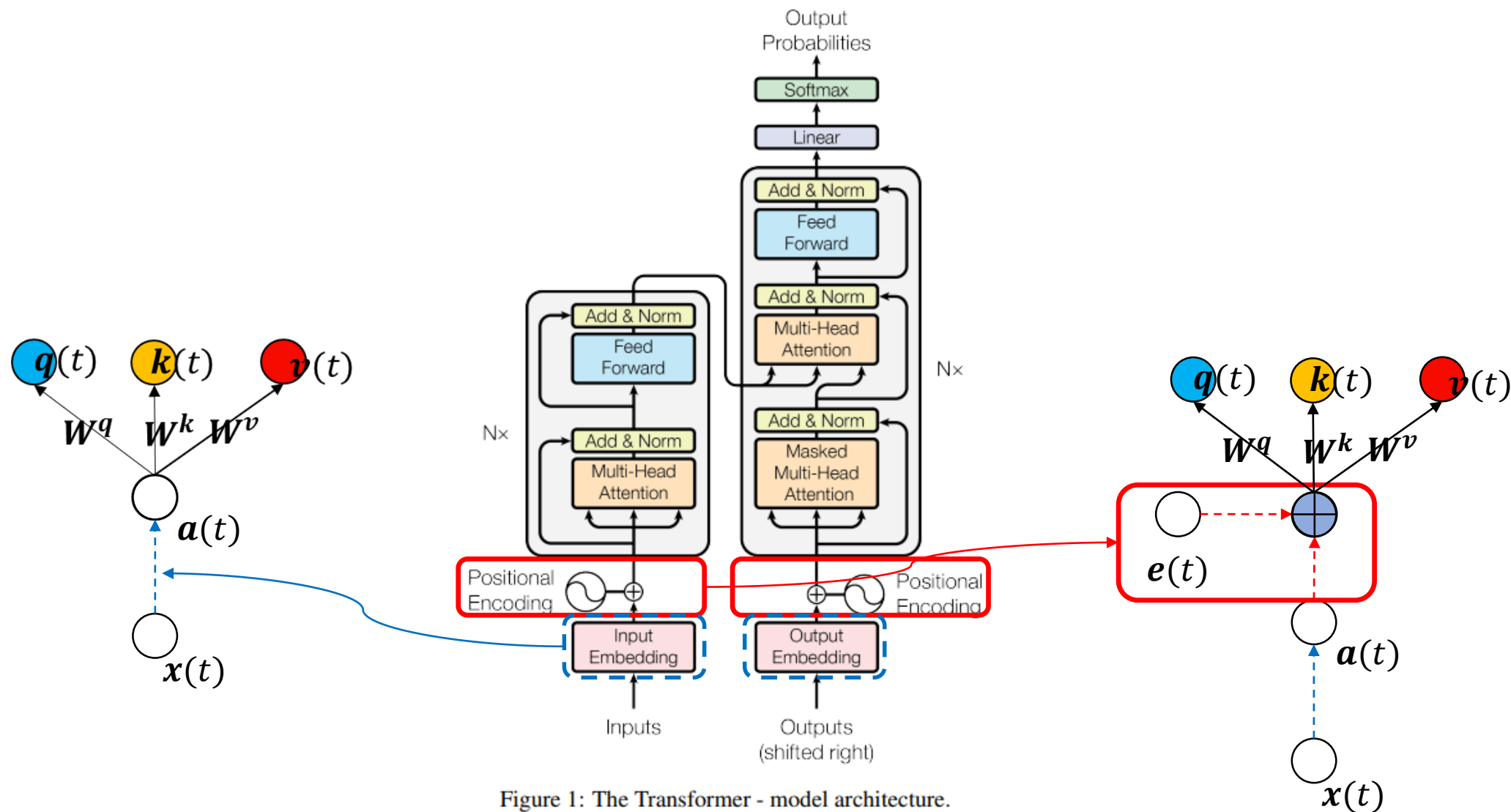
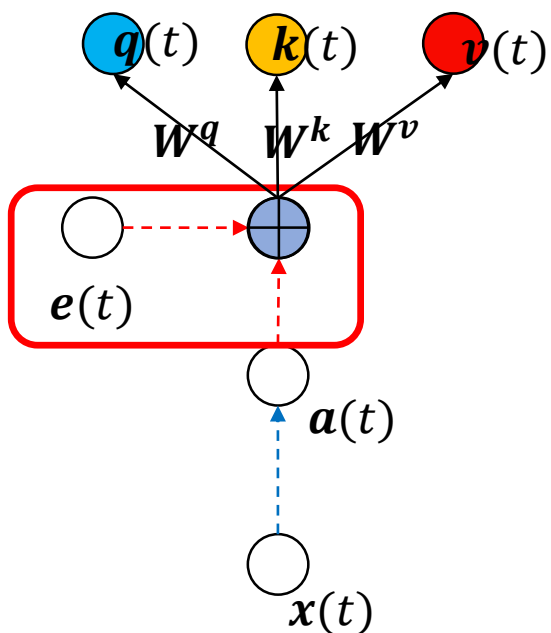


Figure 1: The Transformer - model architecture.



Positional Encoding

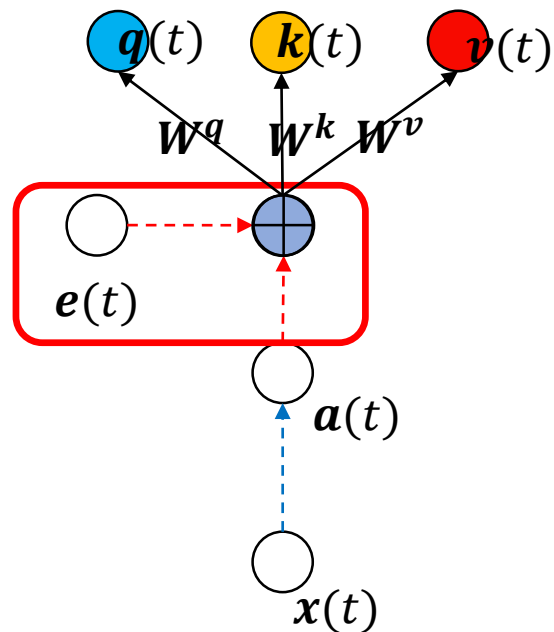


$$\begin{aligned}
 & \mathbf{a}(1) + \mathbf{e}(1) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=1} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=1} \\
 & \mathbf{a}(2) + \mathbf{e}(2) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=2} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=2} \\
 & \mathbf{a}(3) + \mathbf{e}(3) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=3} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=3}
 \end{aligned}$$

$$\begin{aligned}
 & = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=1} + \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=1} \\
 & = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=1} + \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=2} \\
 & = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=1} + \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=3}
 \end{aligned}$$



Positional Encoding



$$a(1) + e(1) = \begin{bmatrix} 1.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}$$

Diagram illustrating the calculation of $a(1) + e(1)$. The input $x(1)$ is transformed into $a(1)$. The positional encoding $e(1)$ is added to $a(1)$ to produce the query vector $q(1)$.

$$e(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$a(1) = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}$$

$$a(2) + e(2) = \begin{bmatrix} 0.2 \\ 1.3 \\ 0.2 \\ 0.3 \end{bmatrix}$$

Diagram illustrating the calculation of $a(2) + e(2)$. The input $x(2)$ is transformed into $a(2)$. The positional encoding $e(2)$ is added to $a(2)$ to produce the query vector $q(2)$.

$$e(2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$a(2) = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.2 \\ 0.3 \end{bmatrix}$$

$$a(3) + e(3) = \begin{bmatrix} 0.1 \\ 0.1 \\ 1.1 \\ 0.1 \end{bmatrix}$$

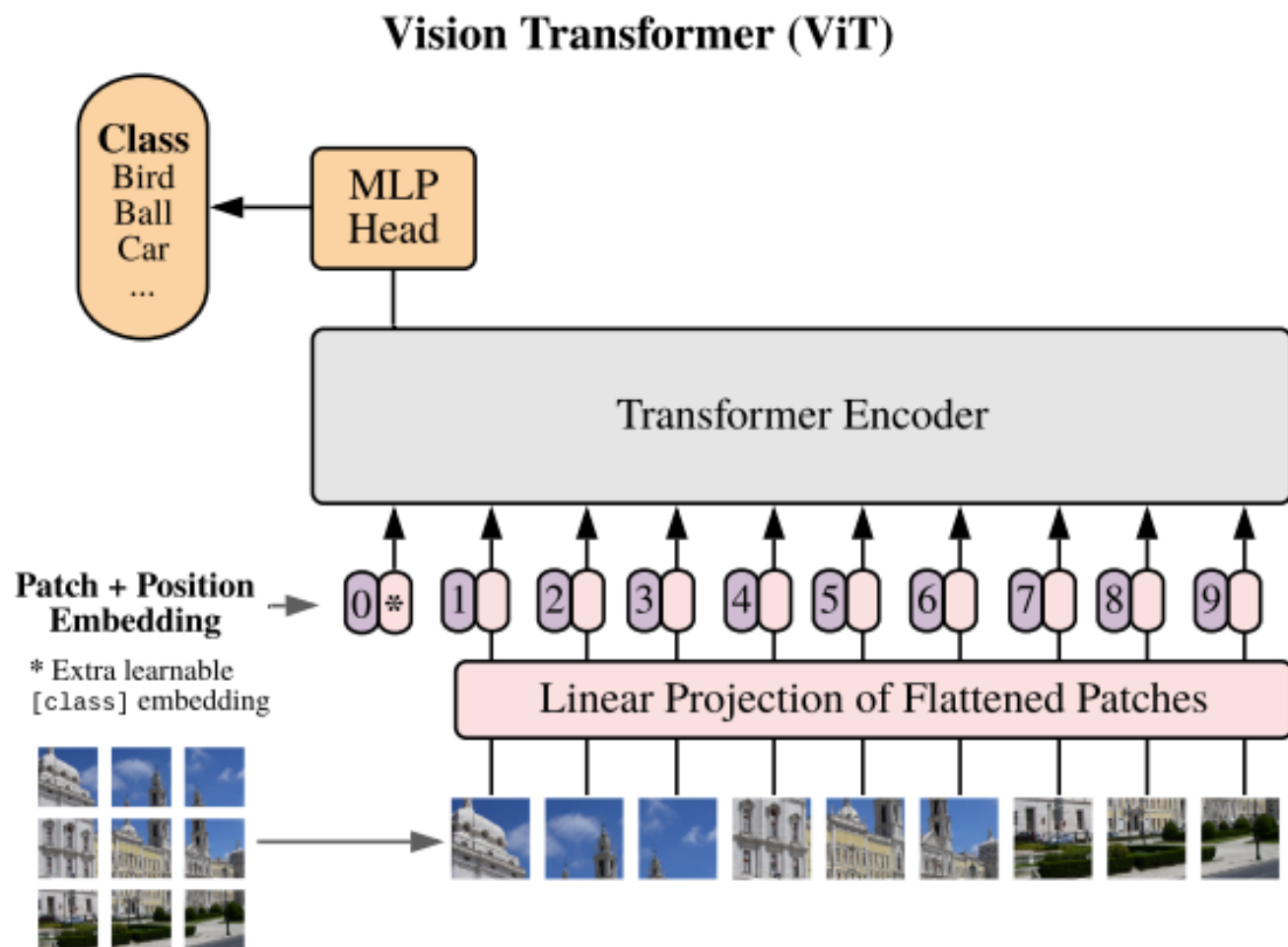
Diagram illustrating the calculation of $a(3) + e(3)$. The input $x(3)$ is transformed into $a(3)$. The positional encoding $e(3)$ is added to $a(3)$ to produce the query vector $q(3)$.

$$e(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

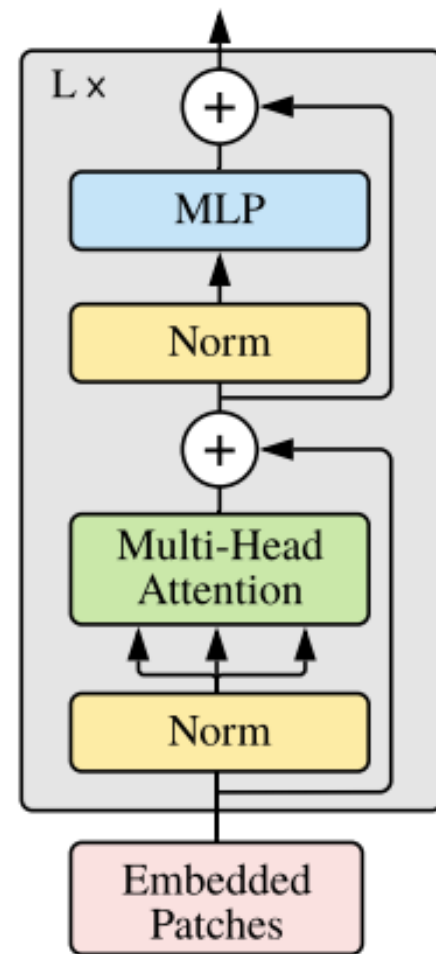
$$a(3) = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$



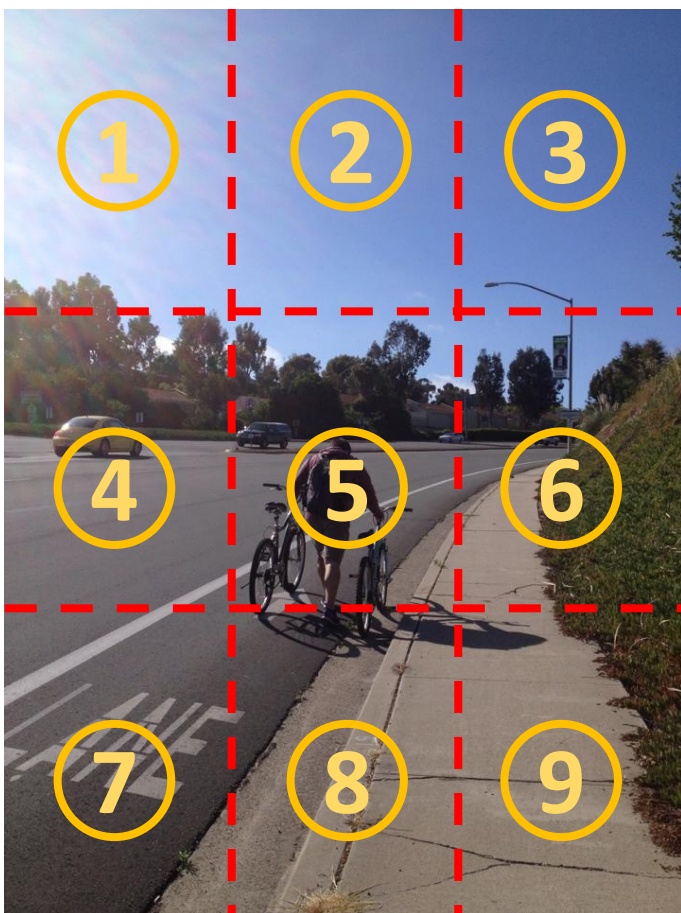
Vision Transformer(ViT)



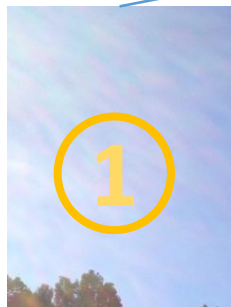
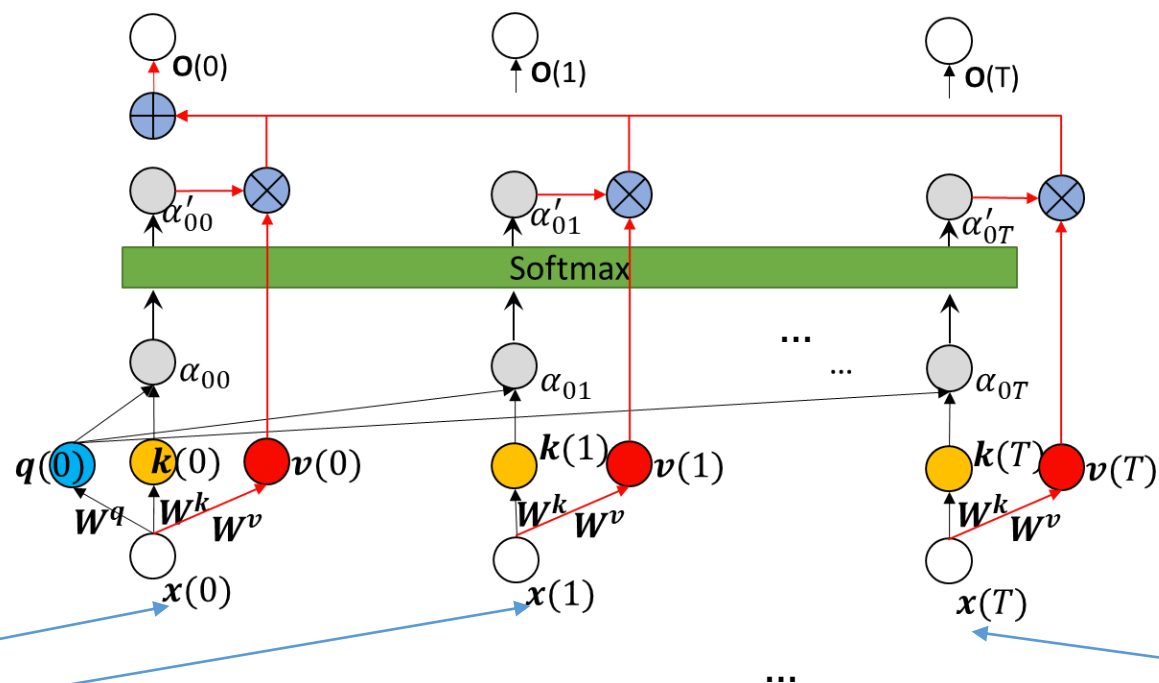
Transformer Encoder



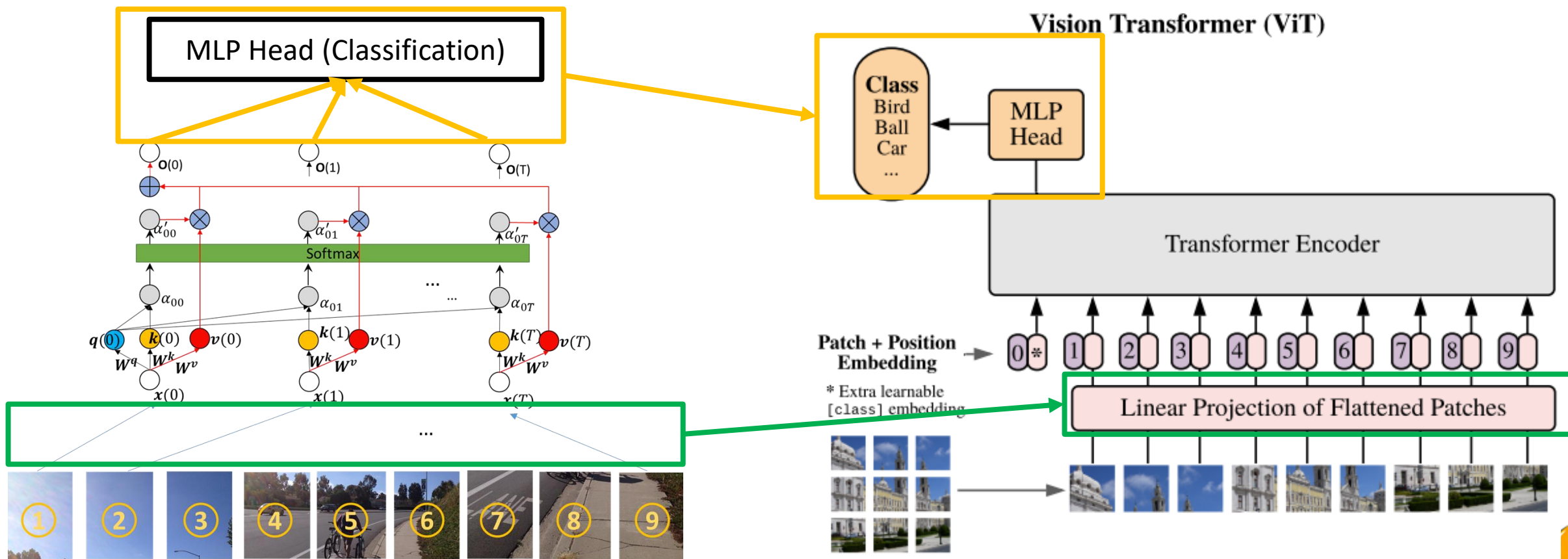
Vision Transformer(ViT)



Vision Transformer(ViT)



Vision Transformer(ViT)

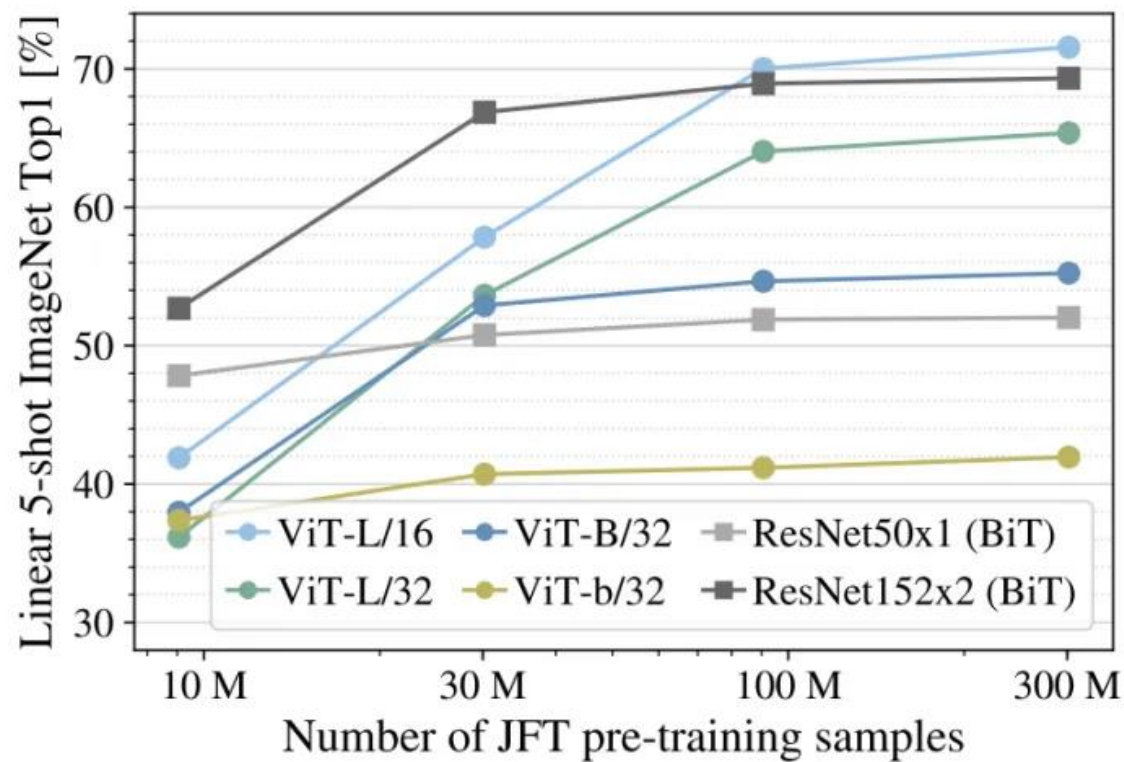


Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

ViT-L/16表示的是large model使用的是16*16的patch

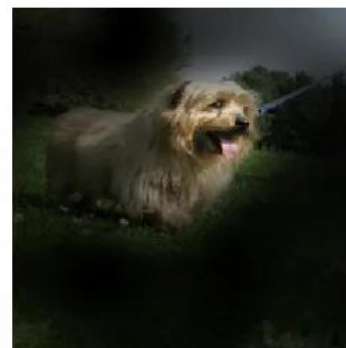
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21K (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	—
CIFAR-100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	—
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	—
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03	—
VTAB (19 tasks)	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	—
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k



Input



Attention



CNN and Self-Attention

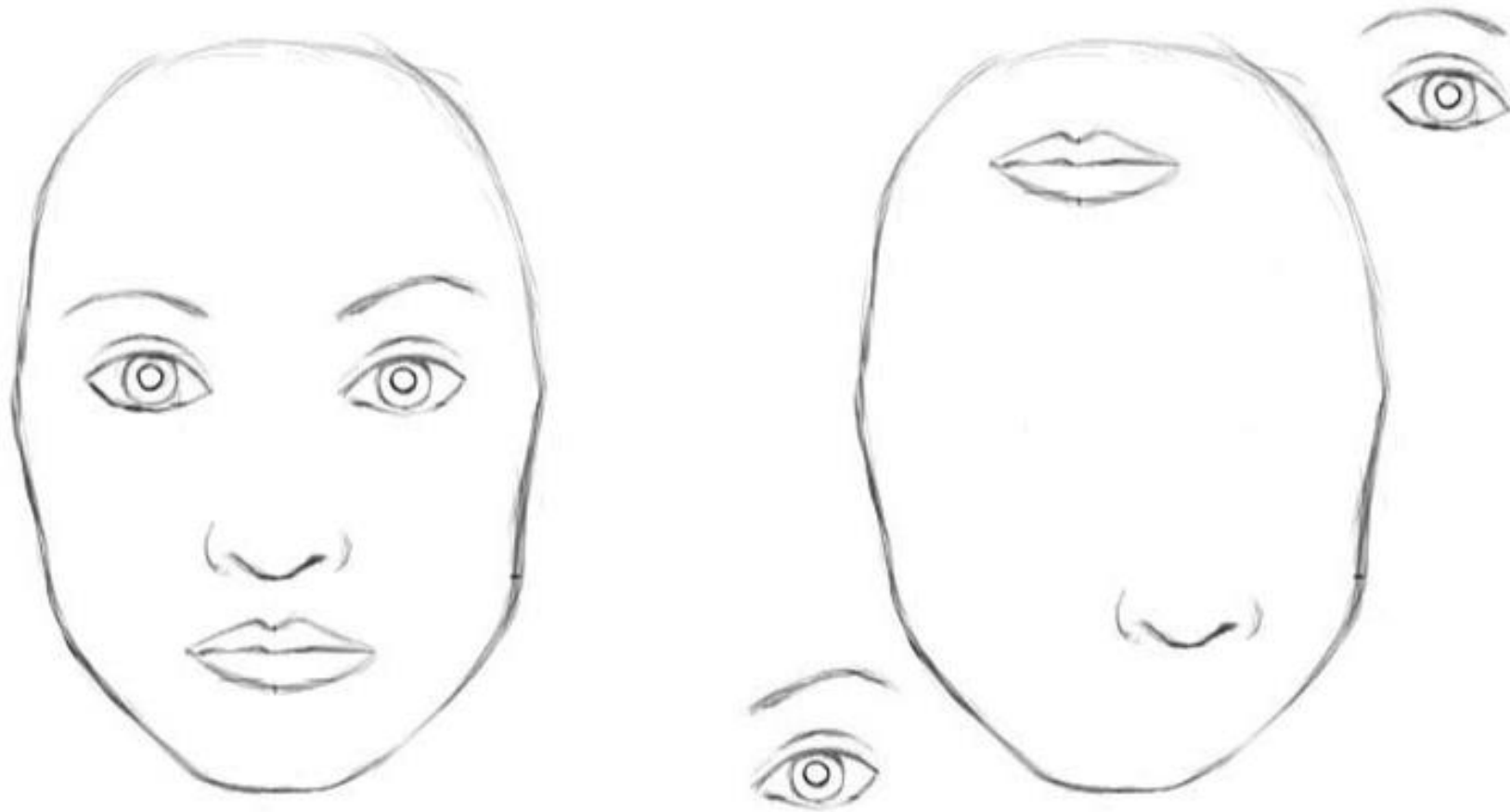


Fig 1: Both [images](#) are similar for a CNN as it does not consider the relative positioning of facial features

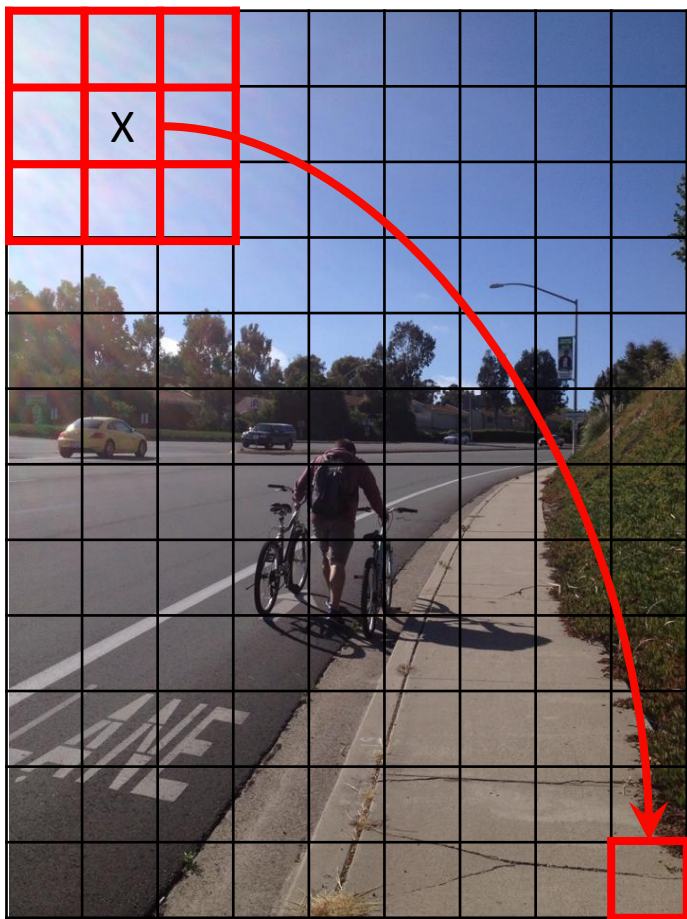
<https://towardsdatascience.com/is-this-the-end-for-convolutional-neural-networks-6f944dccc2e9>



CNN and Self-Attention

CNN

Receptive field



Self - Attention

