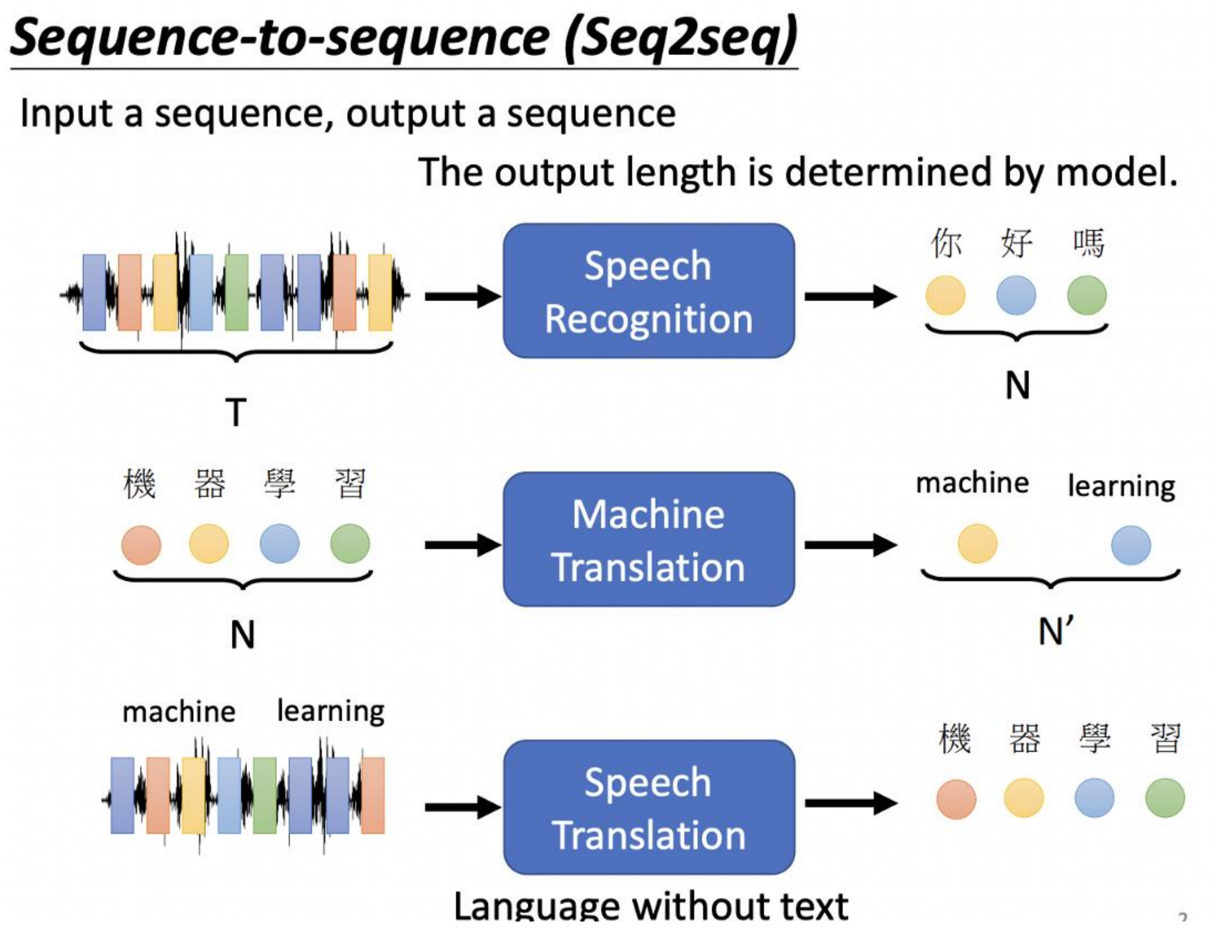
深度學習
- RNN, LSTM, Transformer(Self-Attention), ViT

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

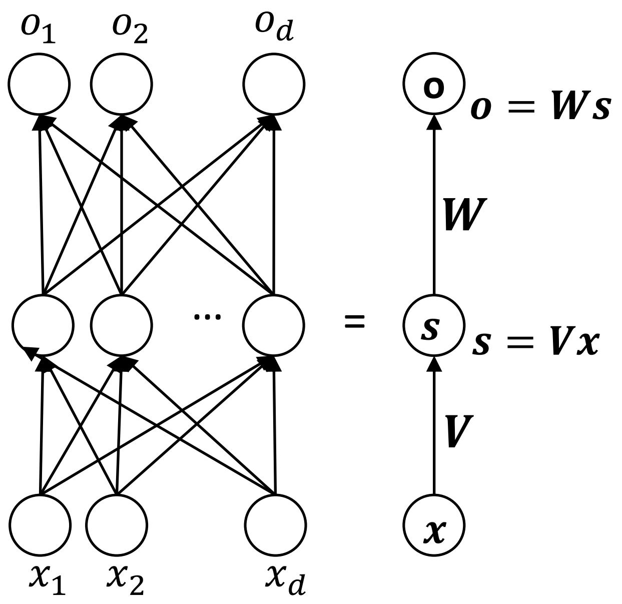


圖片:李宏毅老師

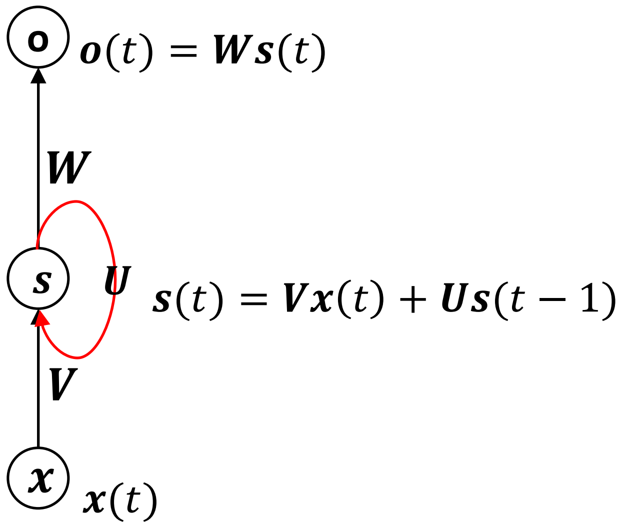
Recurrent Neural Network (RNN)

- RNN和一般MLP的差異就是引進了具有記憶功能的「狀態(State)」

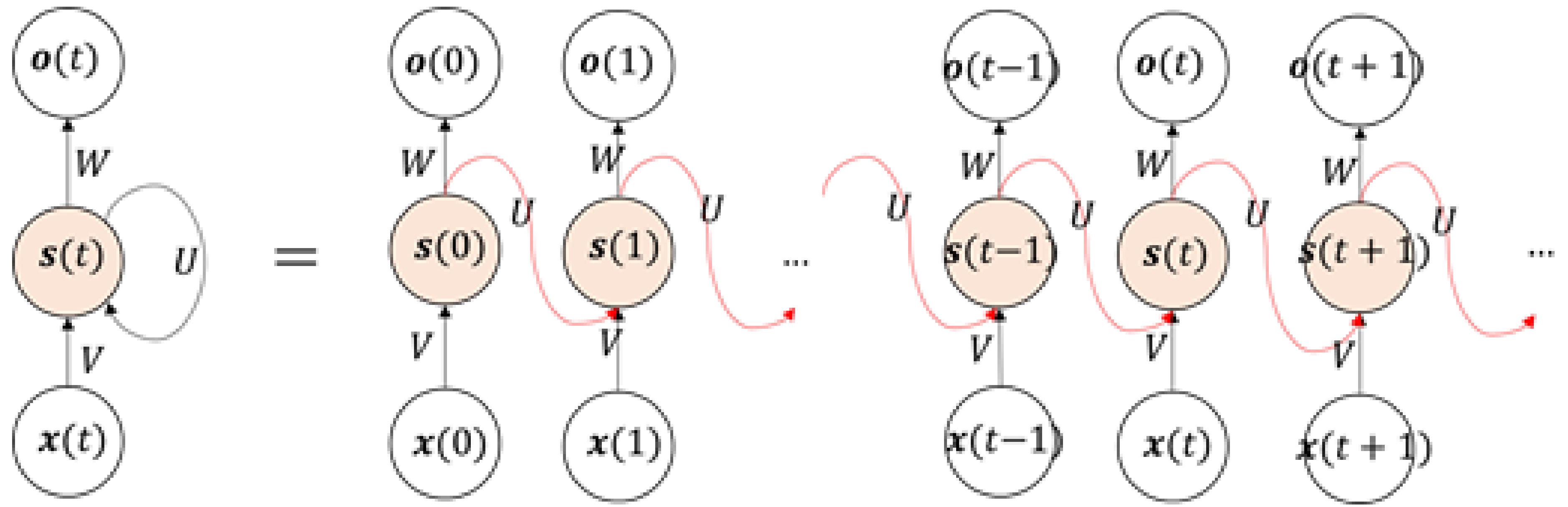
MLP



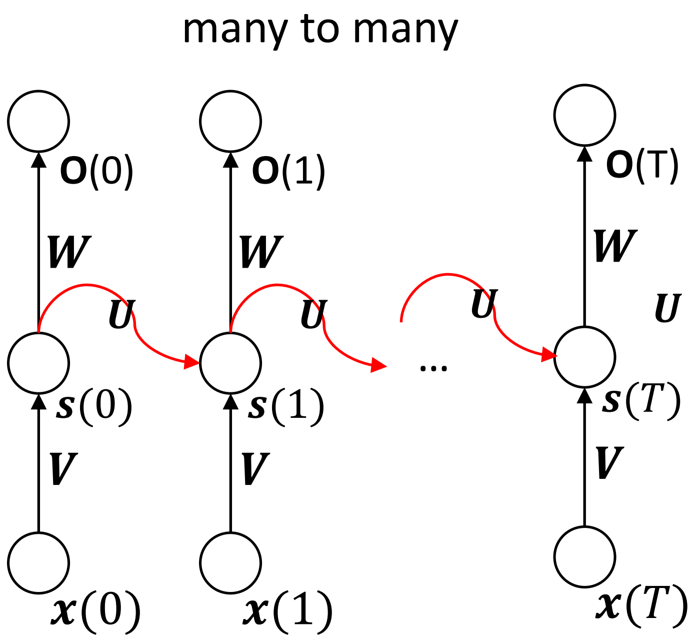
RNN

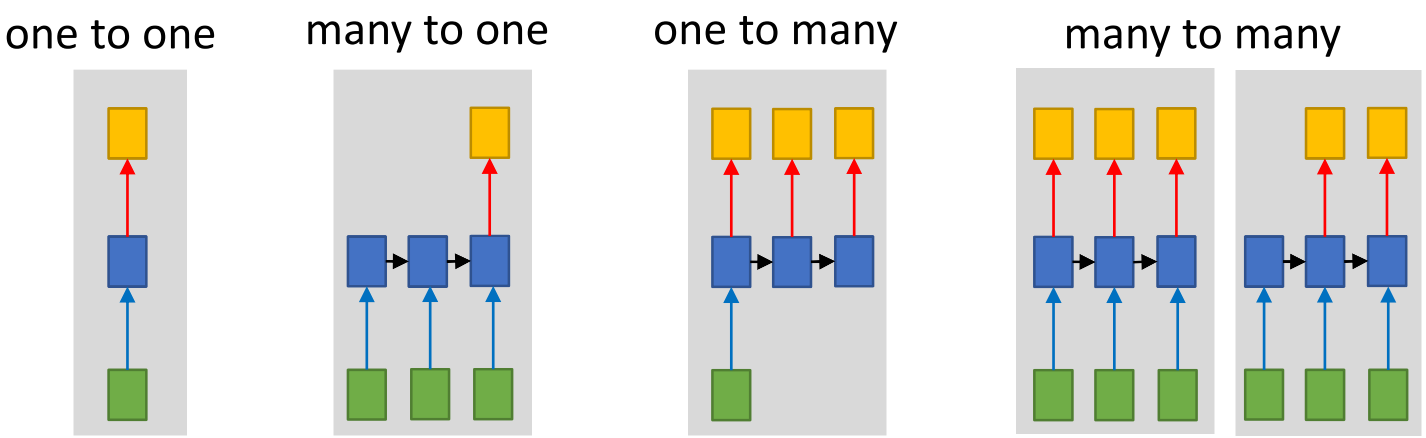


Recurrent Neural Network (RNN)



RNN

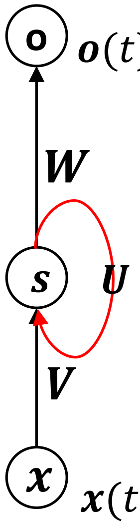




在RNN訓練期間，因為梯度會傳遞到最前一層做乘積，在前面課程提過層數一多可能會發生梯度消失和爆炸的問題，在RNN也是，傳遞不但跟層數相關和時間也先關，可以想像時間等於層數的概念，因此在RNN是很可能發生梯度消失/爆炸。

Long Short Term Memory

RNN

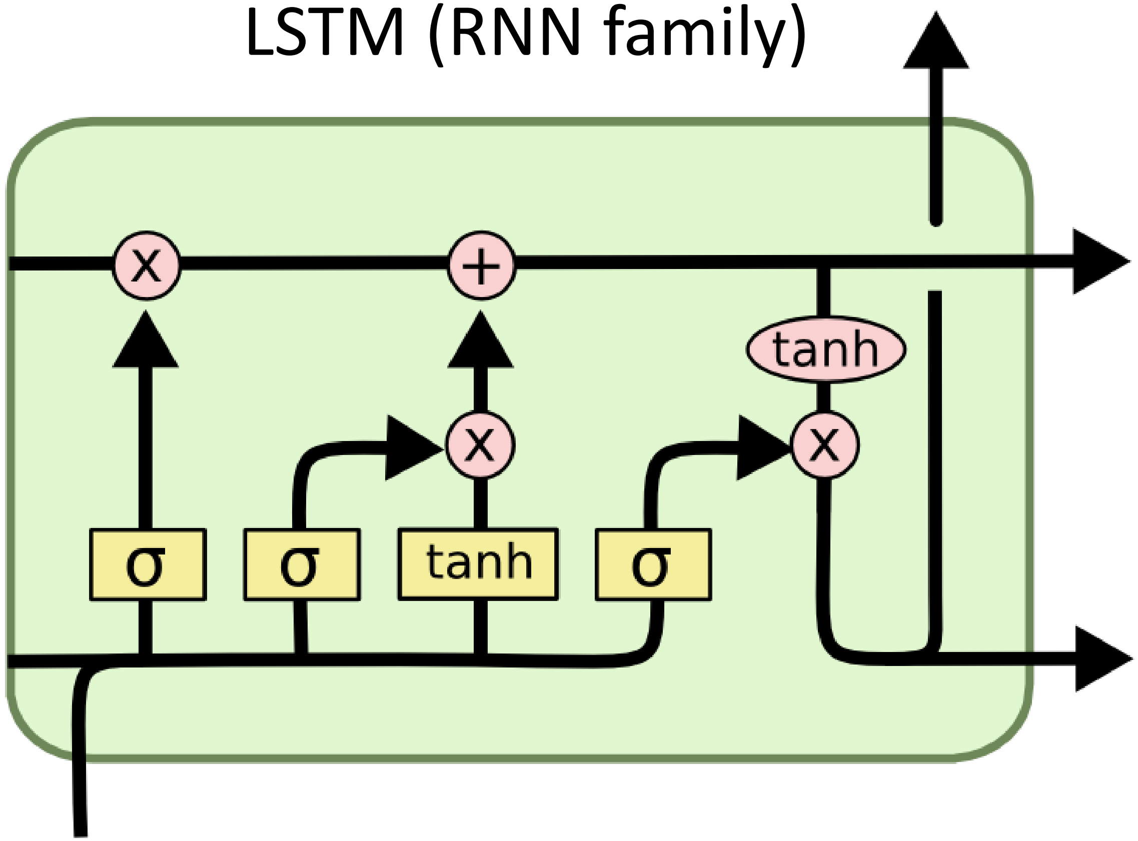


$$o(t) = \boldsymbol{W}s(t)$$

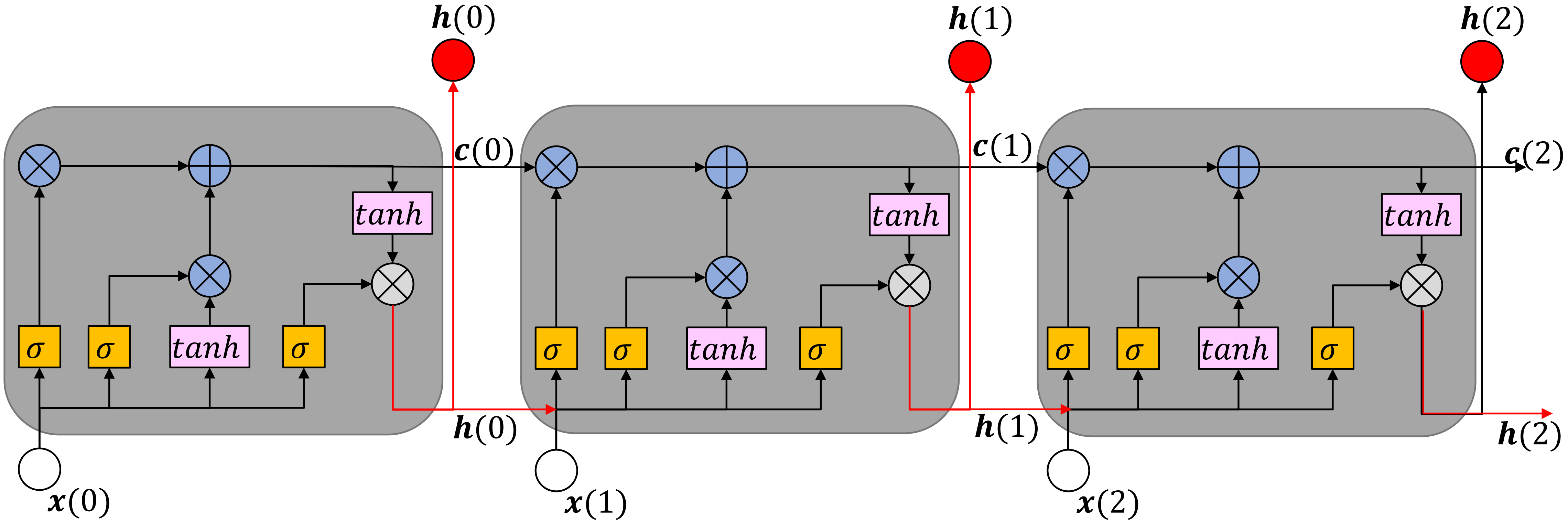
$$s(t) = \boldsymbol{V}x(t) + \boldsymbol{U}s(t-1)$$

$$x(t)$$

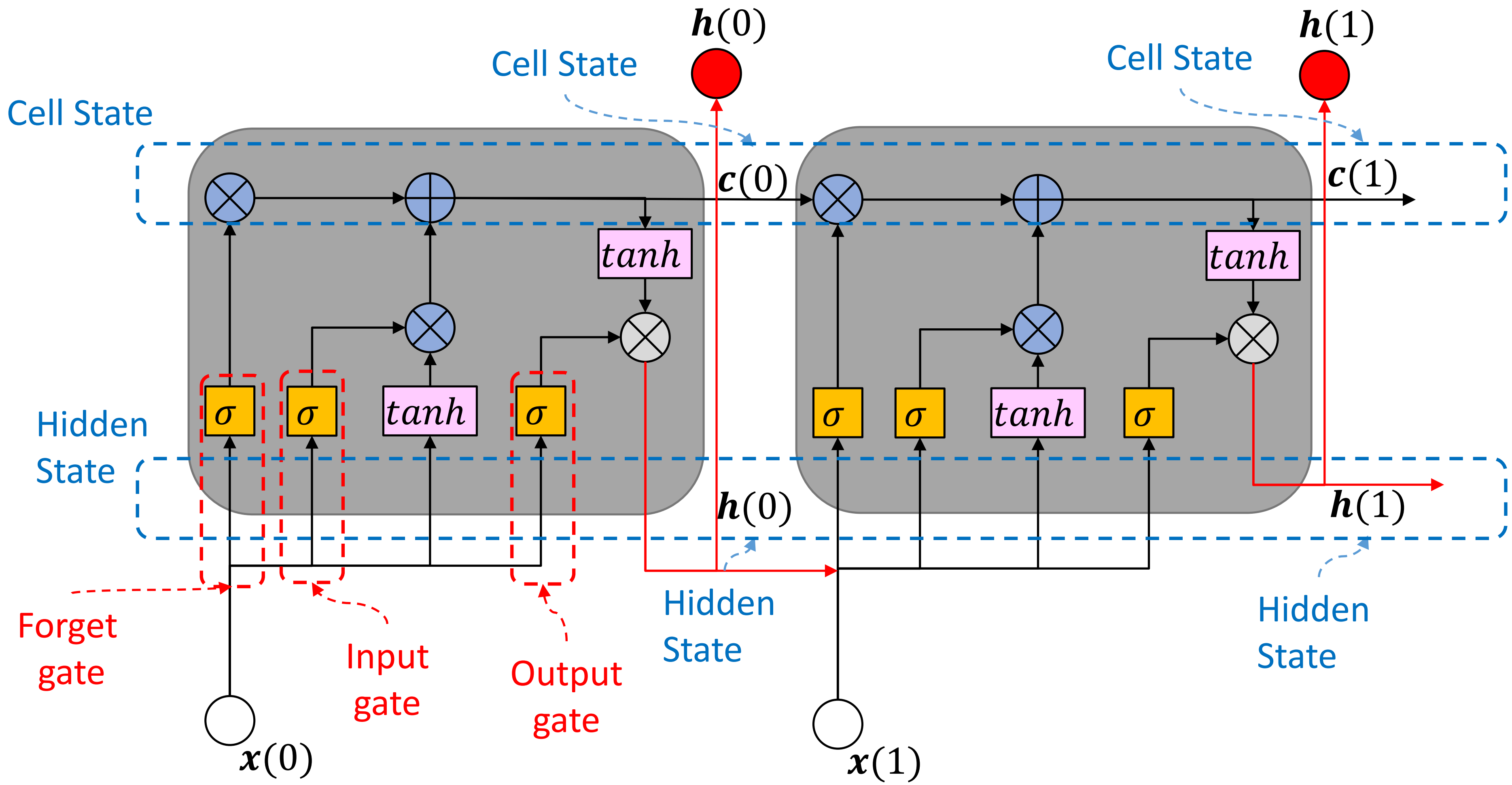
LSTM (RNN family)



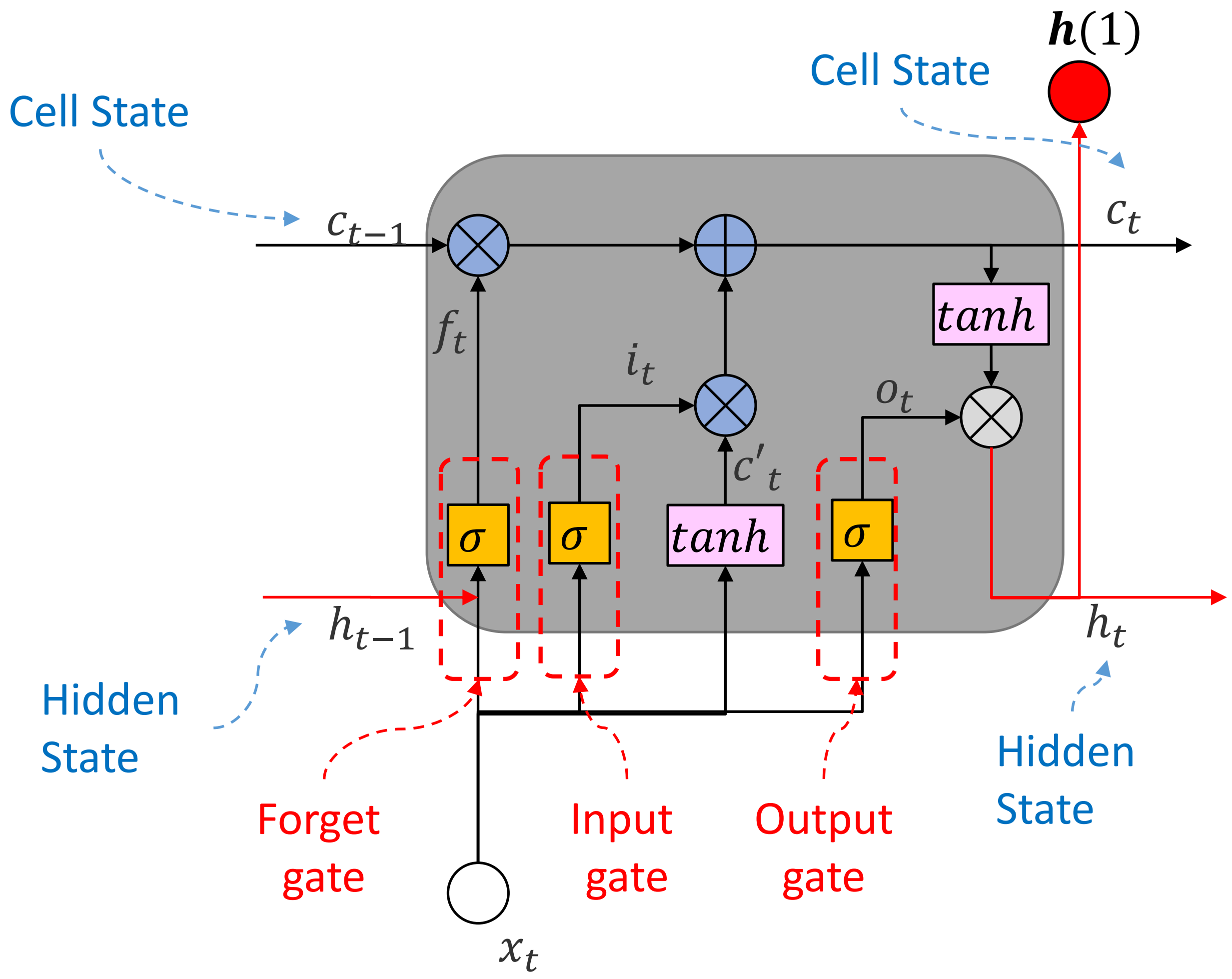
Long Short Term Memory



Long Short Term Memory



Long Short Term Memory



forget gate

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

Input gate

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

Cell State

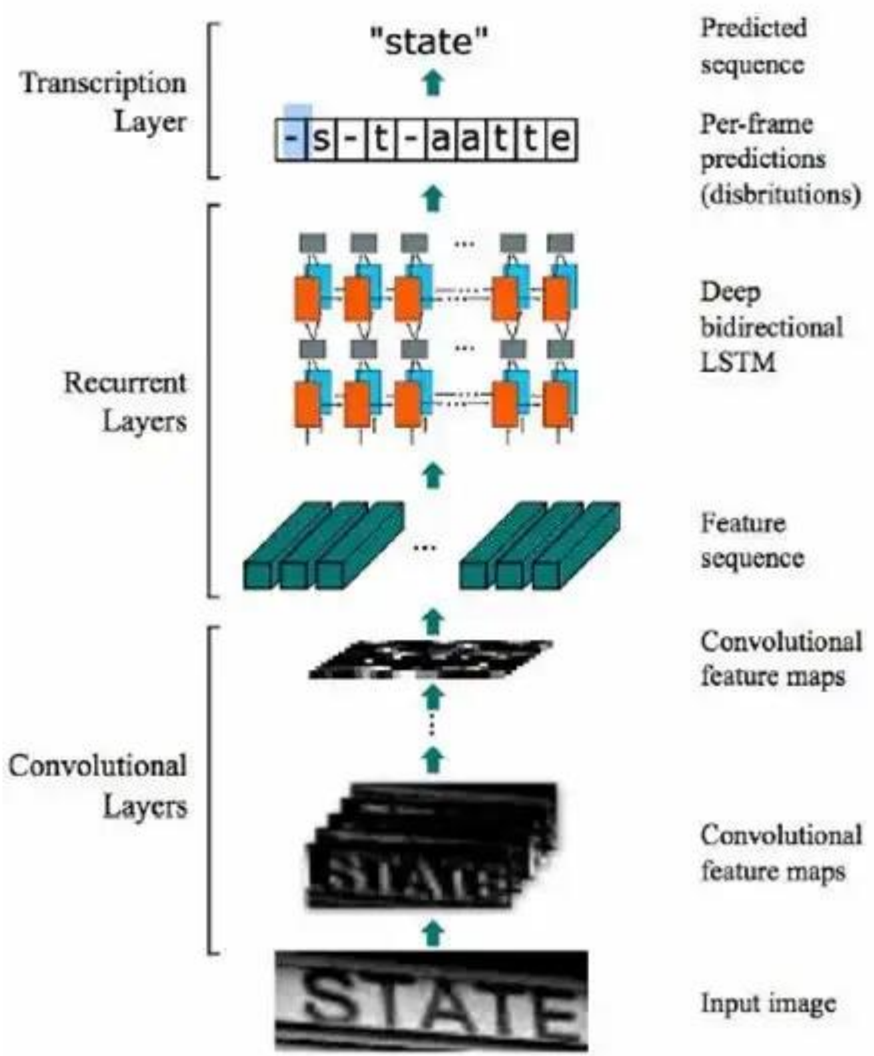
$$c'_t = tanh(W_c x_t + U_c h_{t-1})$$
$$c_t = (f_t c_{t-1} + i_t c'_t)$$

Output gate

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

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

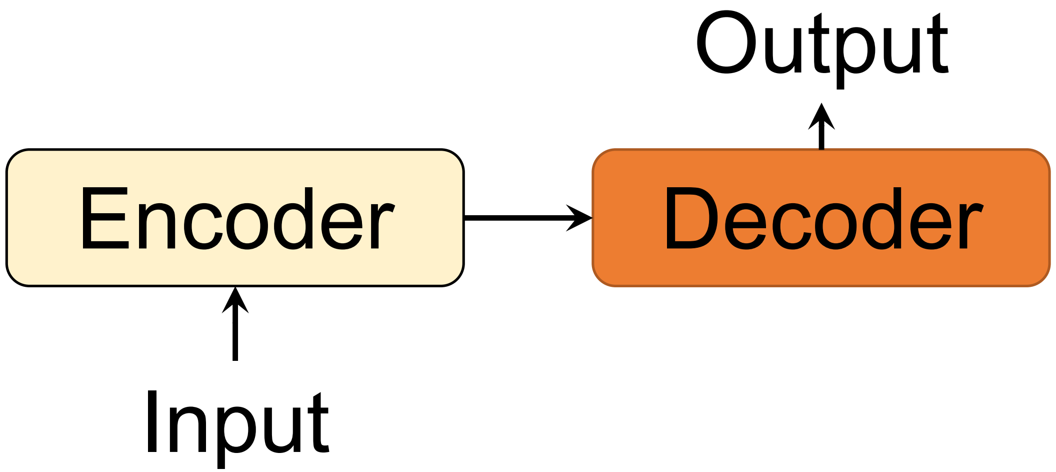
Convolutional Recurrent Neural Network)



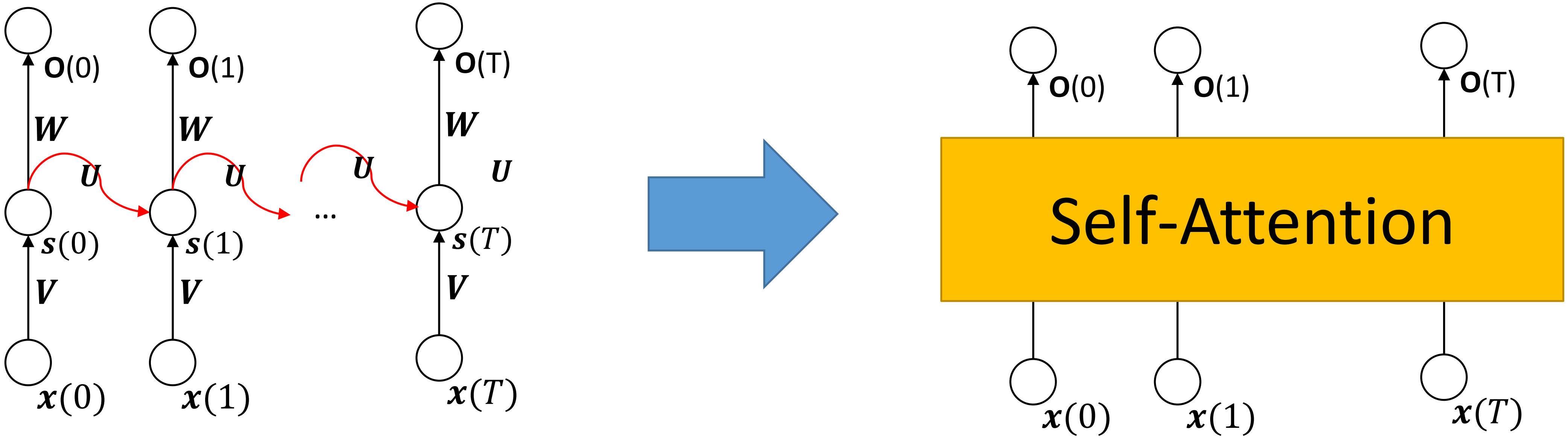
來源: https://github.com/qjadud1994/CRNN-Keras

Transformer (Seq2Seq)

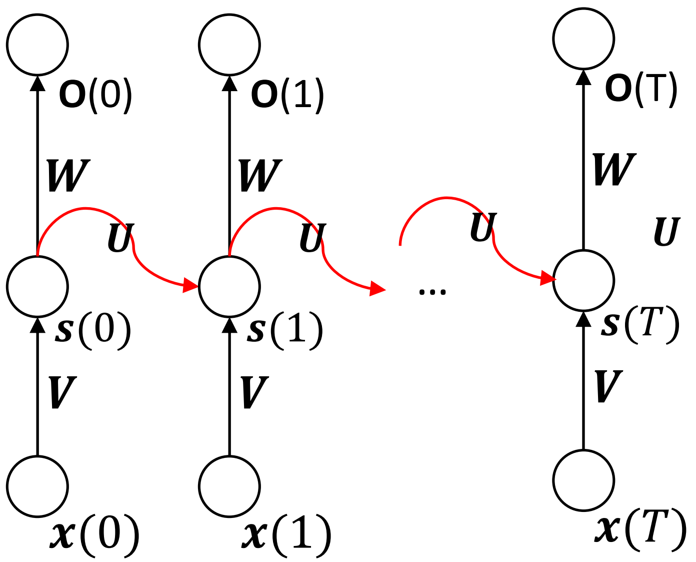
- Sequence-to-sequence: Seq2Seq
- Encoder: takes the input sequence and maps it into a higher dimensional space (n-dimensional vector).
- That abstract vector is fed into the Decoder which turns it into an output sequence.



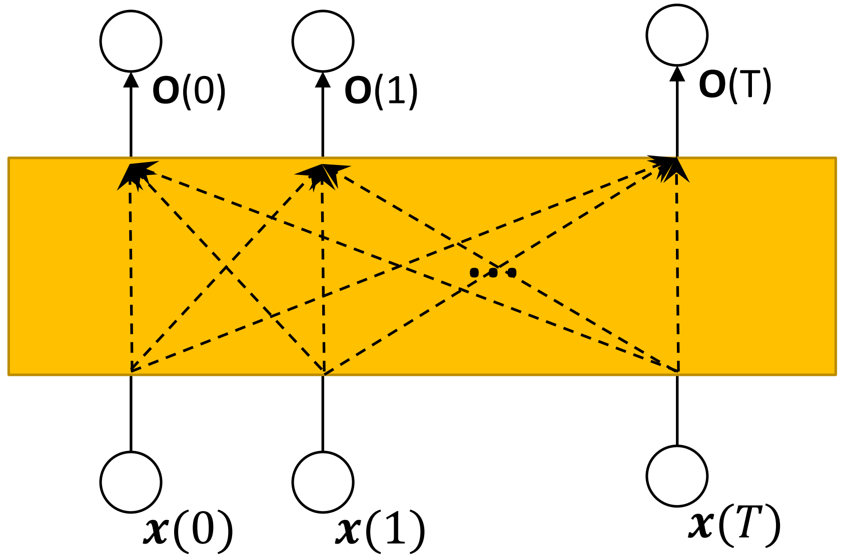
Transformer: Self-Attention



Transformer: Self-Attention

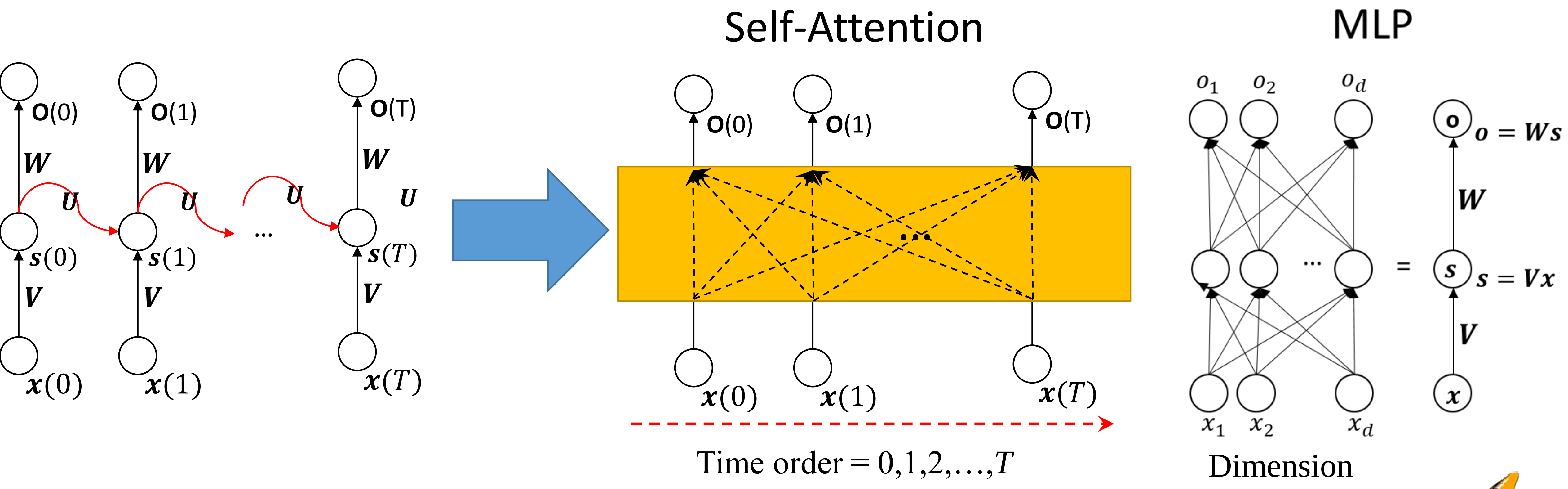


Self-Attention

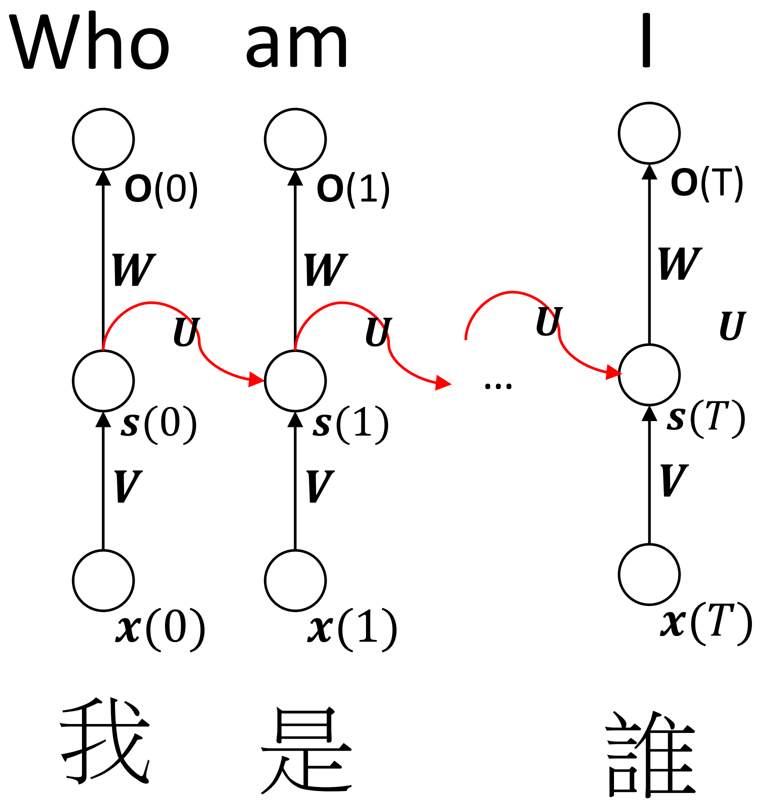


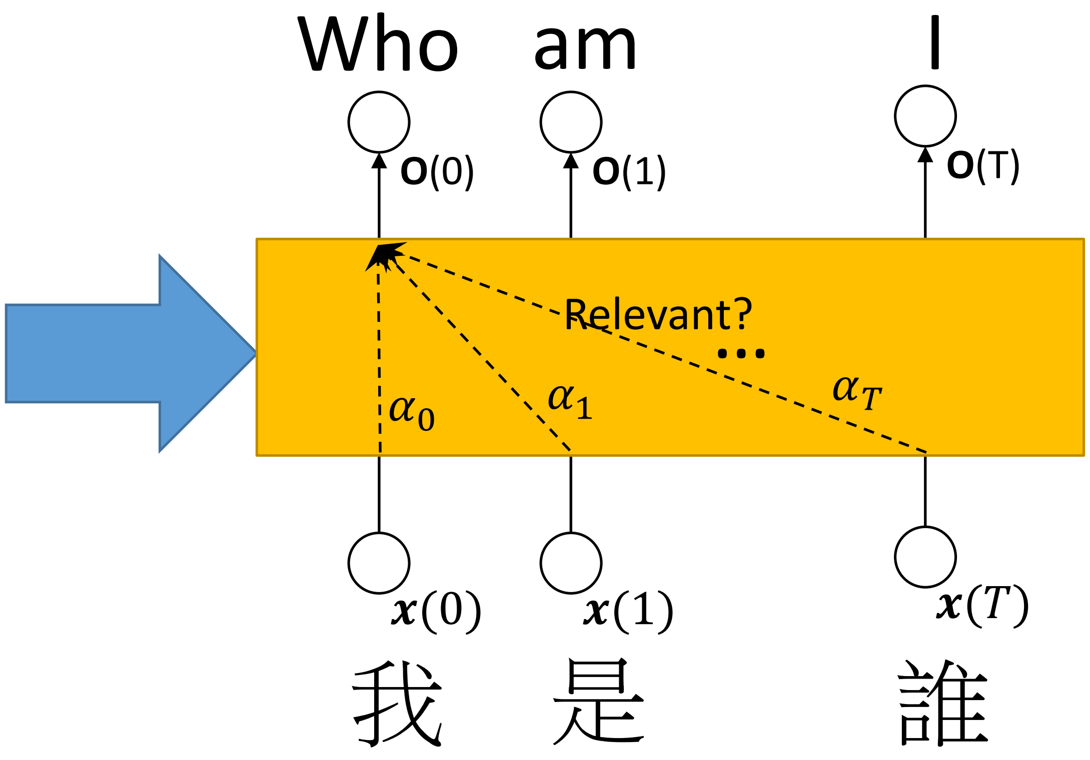
MLP?

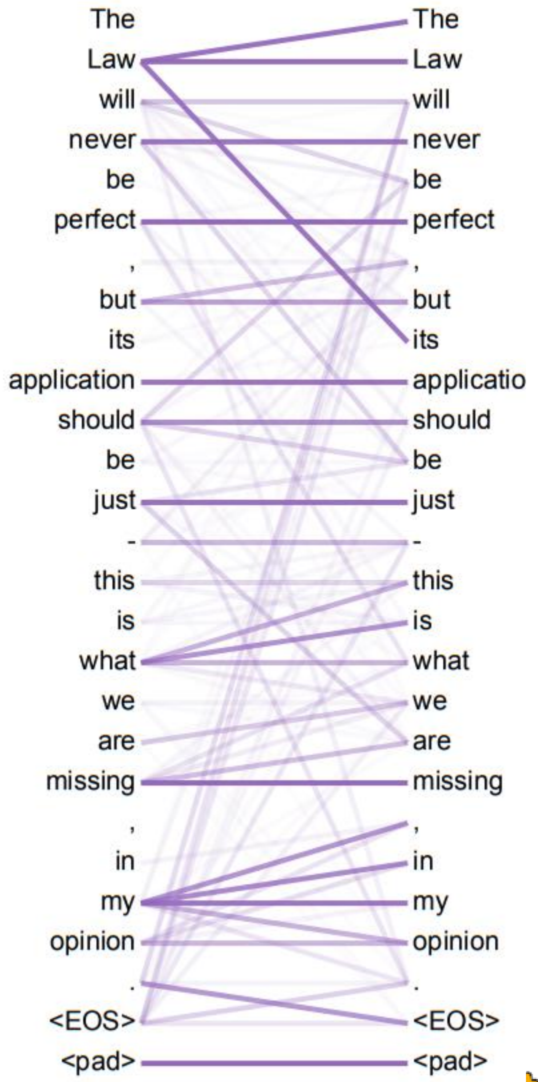
Transformer: Self-Attention



Transformer: Self-Attention



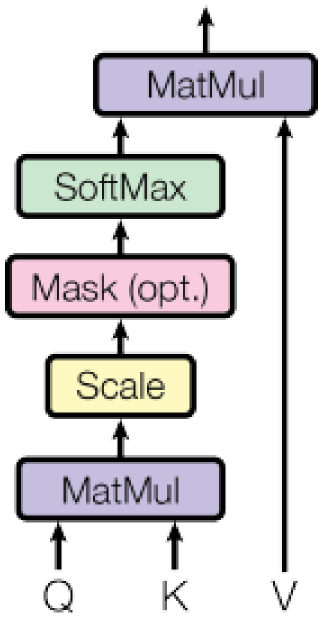




Transformer: Self-Attention

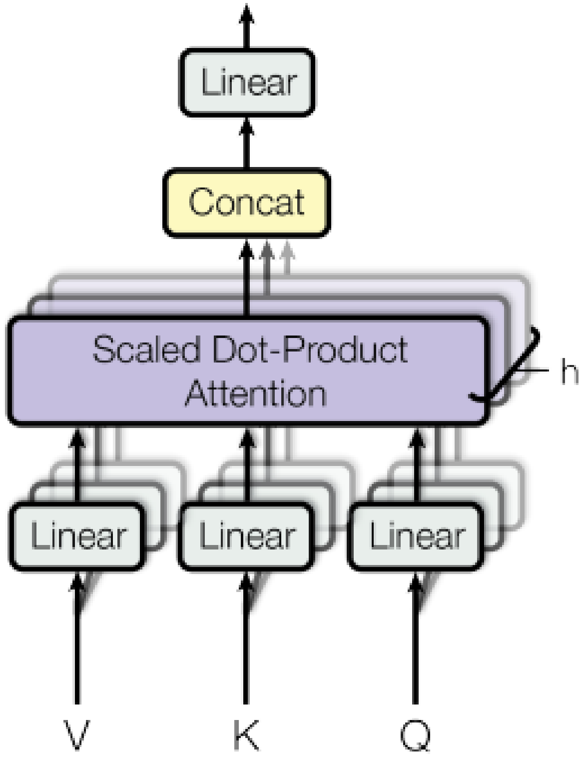
Single-Head Attention

Scaled Dot-Product Attention



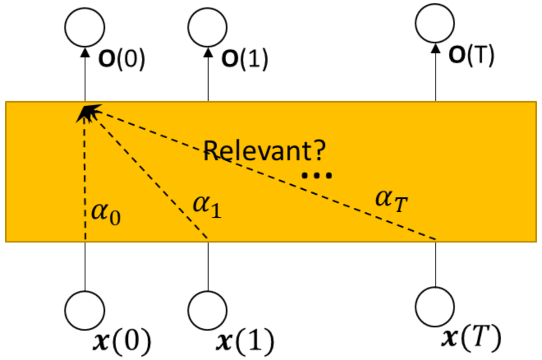
Multi-Head Attention

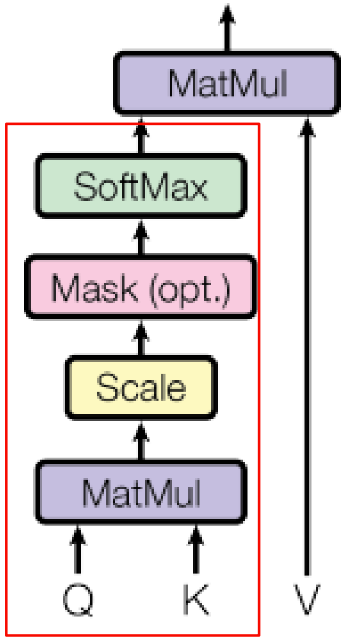
Multi-Head Attention



Reference: Attention Is All You Need (2017)

Single-Head Self-Attention

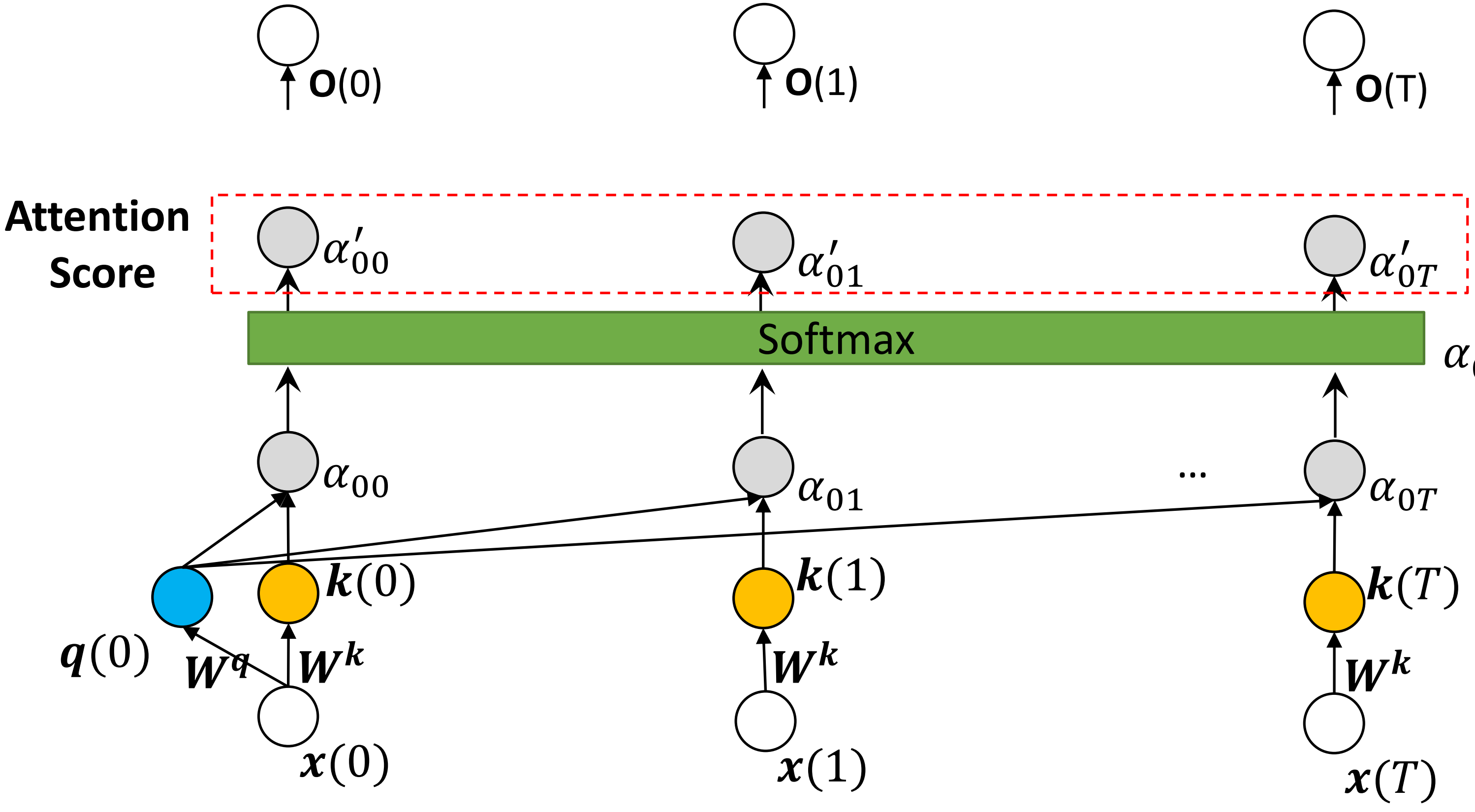




Q: Query

K: Key

V: Value



$$\alpha'_{0i} = \frac{\exp(\alpha_{0i})}{\sum_{j=0}^{T}\exp(\alpha_{0j})}$$

$$\forall i = 0,1,\ldots,T$$

$\boldsymbol{q}(0)$=$\boldsymbol{W}^q\boldsymbol{x}(0)$

$\boldsymbol{k}(0)$=$\boldsymbol{W}^k\boldsymbol{x}(0)$

$\alpha_{00} =< \boldsymbol{q}(0), \boldsymbol{k}(0) >$

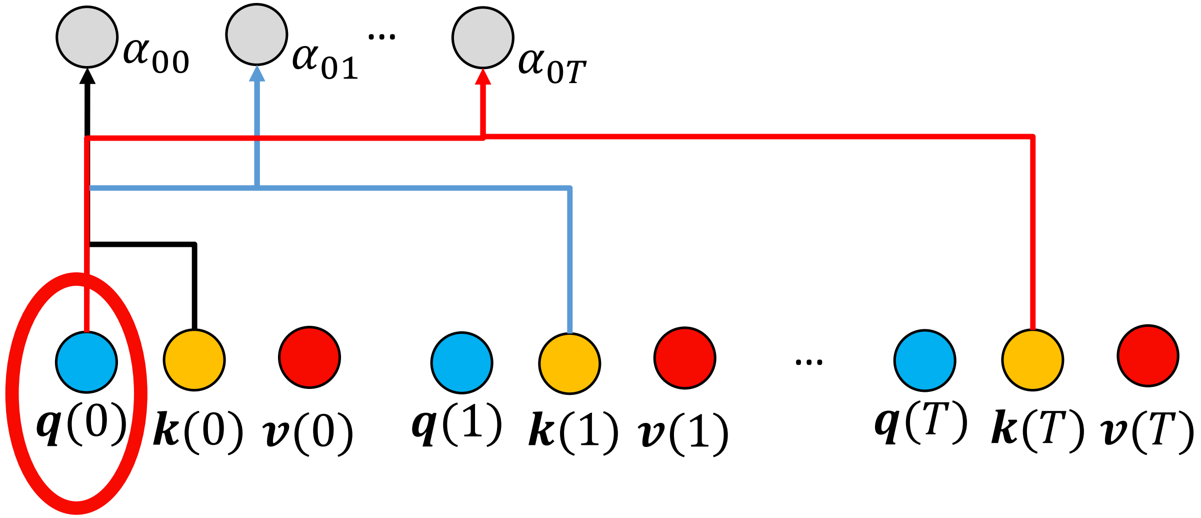
$\boldsymbol{k}(1)$=$\boldsymbol{W}^k\boldsymbol{x}(1)$

$\alpha_{01} =< \boldsymbol{q}(0), \boldsymbol{k}(1) >$

$\boldsymbol{k}(T)$=$\boldsymbol{W}^k\boldsymbol{x}(T)$

$\alpha_{0T} =< \boldsymbol{q}(0), \boldsymbol{k}(T) >$

Attention Score



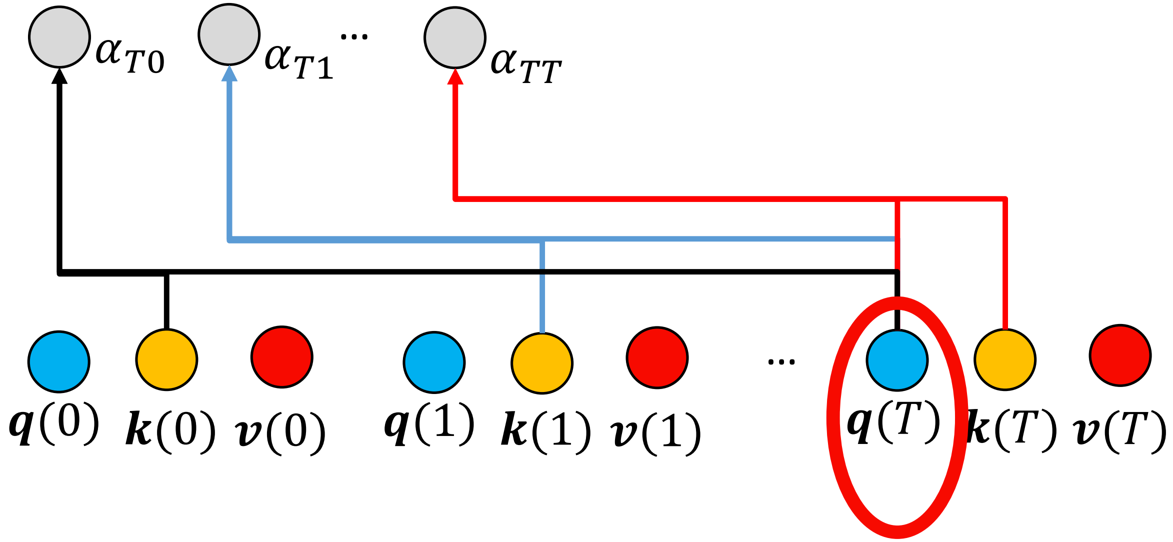
$\alpha_{00} =< \boldsymbol{q}(0), \boldsymbol{k}(0) >$

$\alpha_{01} =< \boldsymbol{q}(0), \boldsymbol{k}(1) >$

…

$\alpha_{0T} =< \boldsymbol{q}(0), \boldsymbol{k}(T) >$

Attention Score at $q(0)$



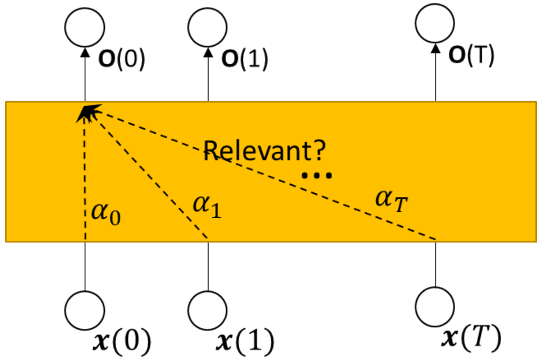
$\alpha_{T0} =< \boldsymbol{q}(T), \boldsymbol{k}(0) >$

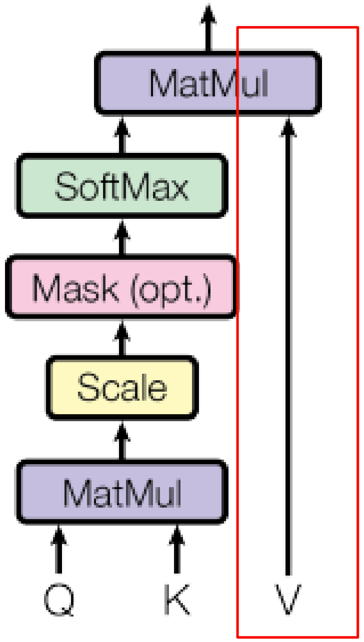
$\alpha_{T1} =< \boldsymbol{q}(T), \boldsymbol{k}(1) >$

…

$\alpha_{TT} =< \boldsymbol{q}(T), \boldsymbol{k}(T) >$

Attention Score at $q(T)$



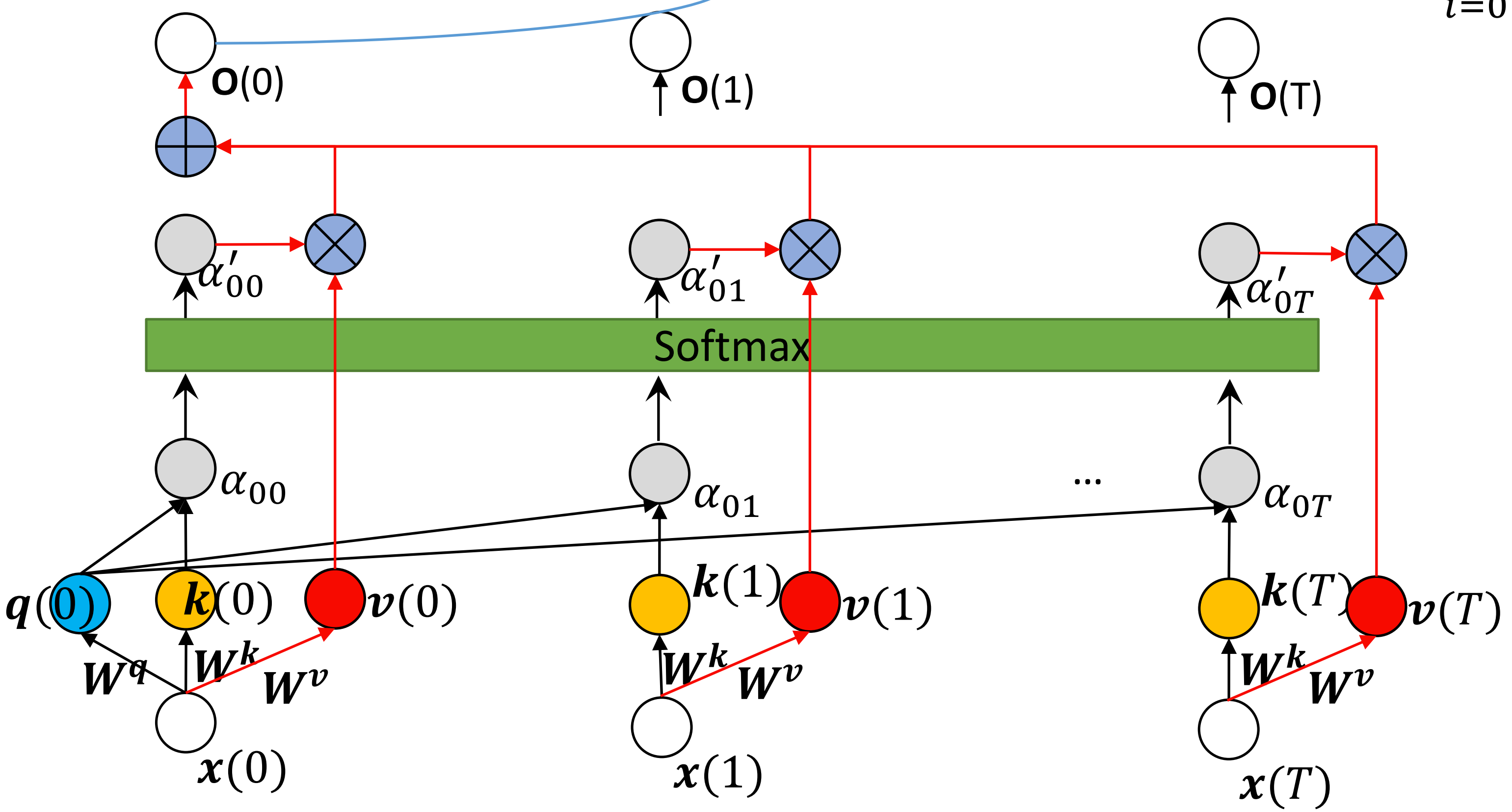


Q: Query

K: Key

V: Value

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



$\boldsymbol{q}(0) = \boldsymbol{W}^q \boldsymbol{x}(0)$

$\boldsymbol{k}(0) = \boldsymbol{W}^k \boldsymbol{x}(0)$

$\boldsymbol{v}(0) = \boldsymbol{W}^v \boldsymbol{x}(0)$

$\alpha_{00} = < \boldsymbol{q}(0), \boldsymbol{k}(0) >$

$\boldsymbol{k}(1) = \boldsymbol{W}^k \boldsymbol{x}(1)$

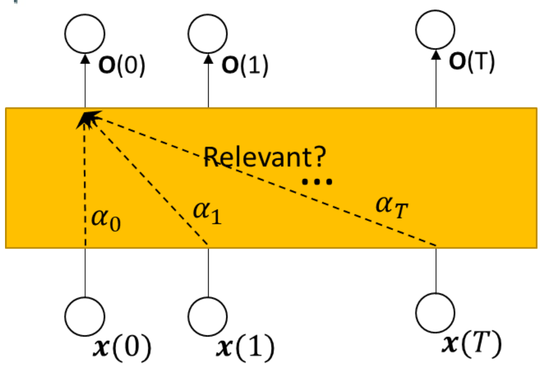
$\boldsymbol{v}(1) = \boldsymbol{W}^v \boldsymbol{x}(1)$

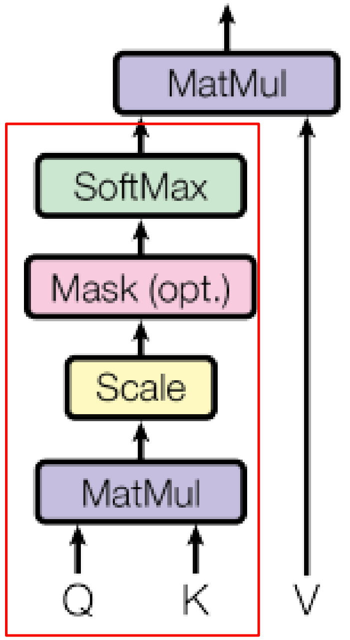
$\alpha_{01} = < \boldsymbol{q}(0), \boldsymbol{k}(1) >$

$\boldsymbol{k}(T) = \boldsymbol{W}^k \boldsymbol{x}(T)$

$\boldsymbol{v}(T) = \boldsymbol{W}^v \boldsymbol{x}(T)$

$\alpha_{0T} = < \boldsymbol{q}(0), \boldsymbol{k}(T) >$

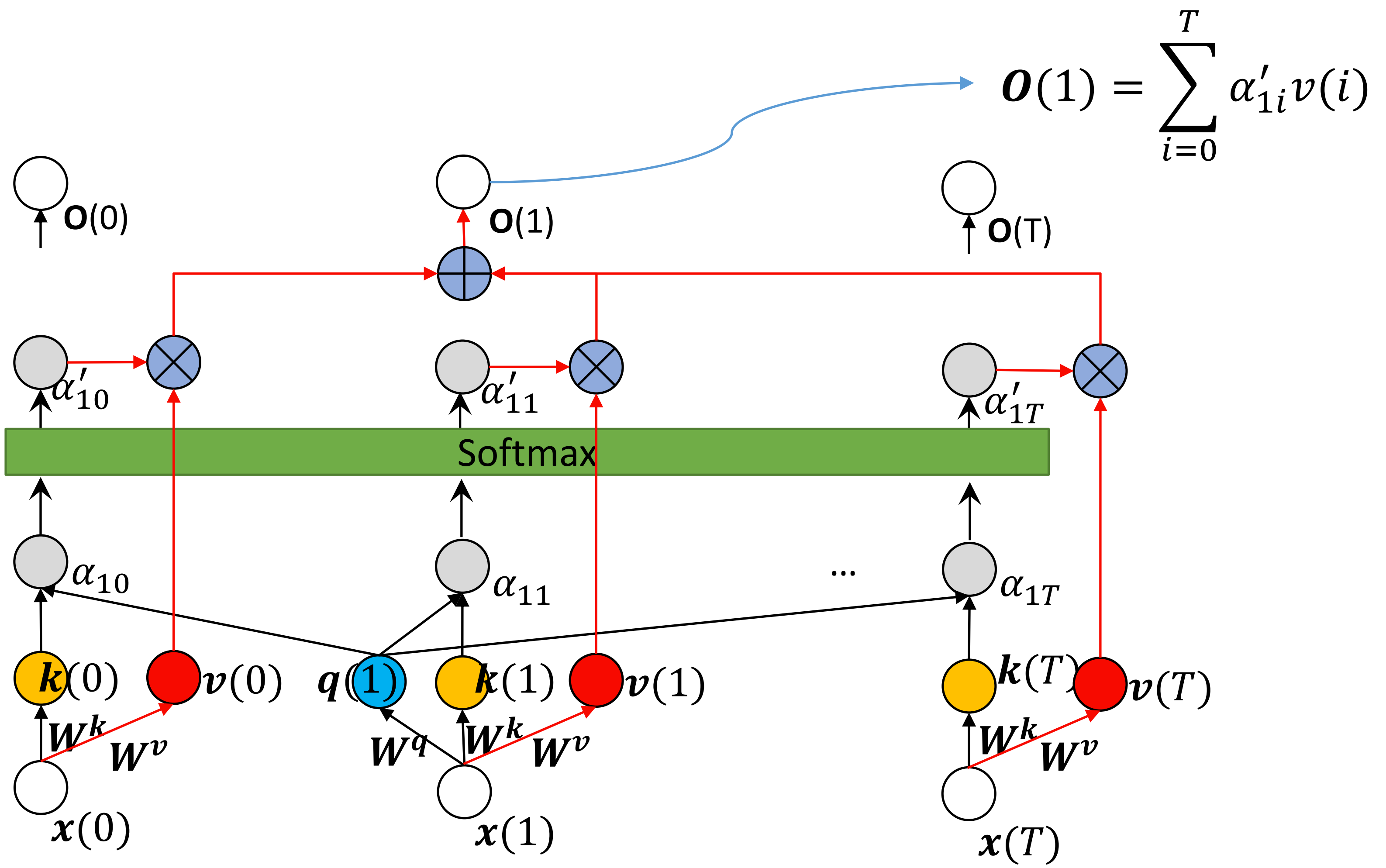




Q: Query

K: Key

V: Value

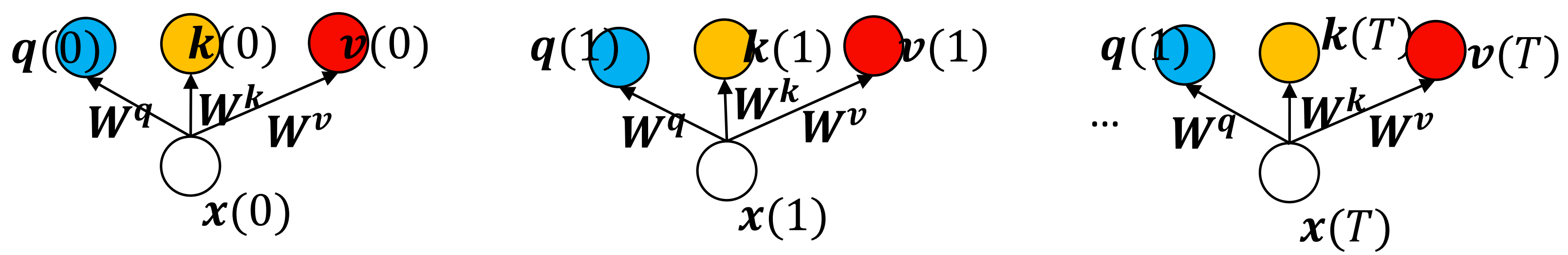


$\boldsymbol{k}(0)=\boldsymbol{W}^k\boldsymbol{x}(0)$
$\boldsymbol{v}(0)=\boldsymbol{W}^v\boldsymbol{x}(0)$
$\alpha_{10} =< \boldsymbol{q}(1), \boldsymbol{k}(0) >$

$\boldsymbol{q}(1)=\boldsymbol{W}^q\boldsymbol{x}(1)$
$\boldsymbol{k}(1)=\boldsymbol{W}^k\boldsymbol{x}(1)$
$\boldsymbol{v}(1)=\boldsymbol{W}^v\boldsymbol{x}(1)$
$\alpha_{11} =< \boldsymbol{q}(1), \boldsymbol{k}(1) >$

$\boldsymbol{k}(T)=\boldsymbol{W}^k\boldsymbol{x}(T)$
$\boldsymbol{v}(T)=\boldsymbol{W}^v\boldsymbol{x}(T)$
$\alpha_{1T} =< \boldsymbol{q}(1), \boldsymbol{k}(T) >$

用大矩陣直接運算QKV



$$\boldsymbol{q}(0)=\boldsymbol{W}^q\boldsymbol{x}(0)$$
$$\boldsymbol{k}(0)=\boldsymbol{W}^k\boldsymbol{x}(0)$$
$$\boldsymbol{v}(0)=\boldsymbol{W}^v\boldsymbol{x}(0)$$

$$\boldsymbol{q}(1)=\boldsymbol{W}^q\boldsymbol{x}(1)$$
$$\boldsymbol{k}(1)=\boldsymbol{W}^k\boldsymbol{x}(1)$$
$$\boldsymbol{v}(1)=\boldsymbol{W}^v\boldsymbol{x}(1)$$

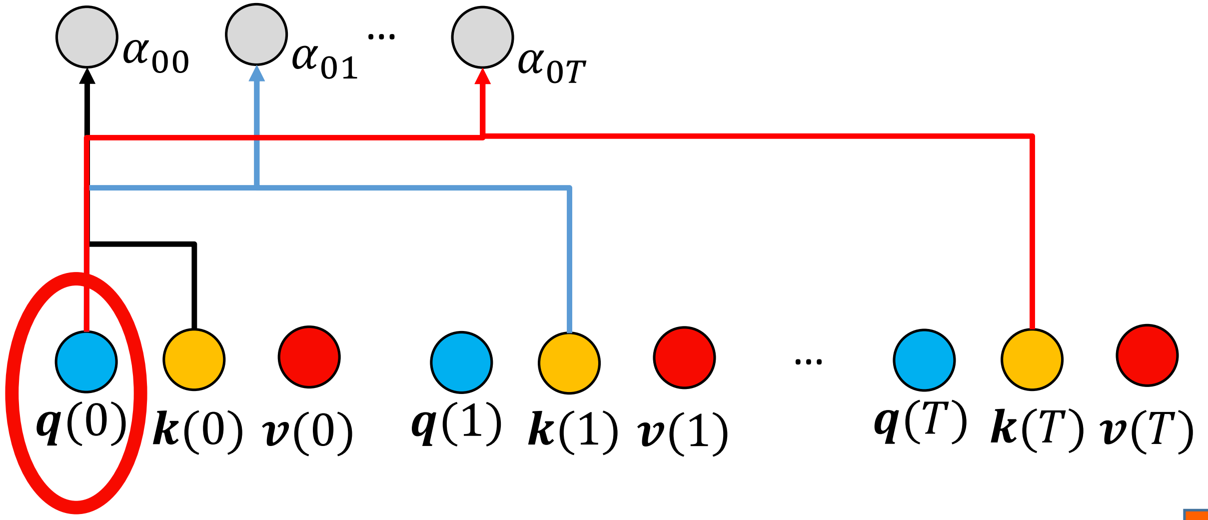
$$\boldsymbol{q}(T)=\boldsymbol{W}^q\boldsymbol{x}(T)$$
$$\boldsymbol{k}(T)=\boldsymbol{W}^k\boldsymbol{x}(T)$$
$$\boldsymbol{v}(T)=\boldsymbol{W}^v\boldsymbol{x}(T)$$

$$\underset{d\times T}{\mathbf{Q}} = \underset{d\times T}{\left[\underset{d\times 1}{\boxed{\boldsymbol{q}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{q}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{q}(T)}}\right]} = \underset{d\times d}{\boxed{\boldsymbol{W}^q}} \underset{d\times T}{\left[\underset{d\times 1}{\boxed{\boldsymbol{x}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{x}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{x}(T)}}\right]}$$

$$\underset{d\times T}{\mathbf{K}} = \underset{d\times T}{\left[\underset{d\times 1}{\boxed{\boldsymbol{k}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{k}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{k}(T)}}\right]} = \underset{d\times d}{\boxed{\boldsymbol{W}^k}} \underset{d\times T}{\left[\underset{d\times 1}{\boxed{\boldsymbol{x}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{x}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{x}(T)}}\right]}$$

$$\underset{d\times T}{\mathbf{V}} = \underset{d\times T}{\left[\underset{d\times 1}{\boxed{\boldsymbol{v}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{v}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{v}(T)}}\right]} = \underset{d\times d}{\boxed{\boldsymbol{W}^v}} \underset{d\times T}{\left[\underset{d\times 1}{\boxed{\boldsymbol{x}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{x}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{x}(T)}}\right]}$$

用大矩陣直接運算A



$$\alpha_{00} = <\boldsymbol{q}(0), \boldsymbol{k}(0)>$$
$$\alpha_{01} = <\boldsymbol{q}(0), \boldsymbol{k}(1)>$$
$$\dots$$
$$\alpha_{0T} = <\boldsymbol{q}(0), \boldsymbol{k}(T)>$$

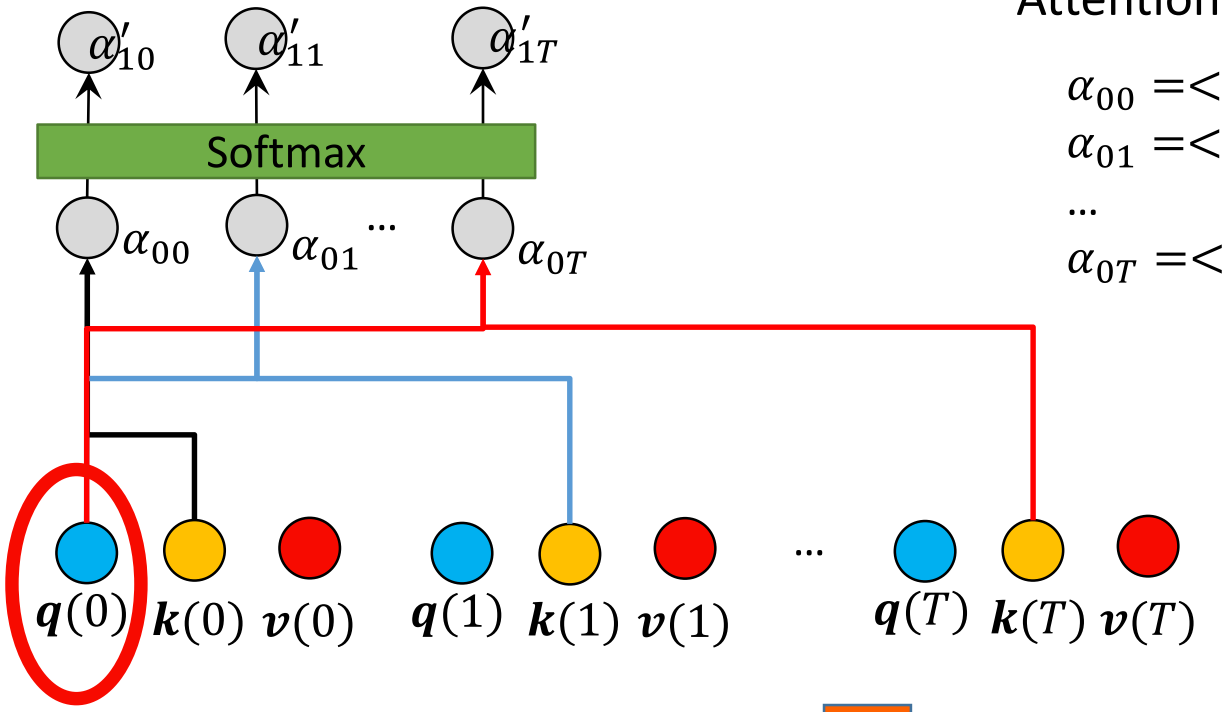
Attention Score at $q(0)$

Attention Score Matrix

$$\boxed{\begin{matrix}\alpha_{00} \\ \alpha_{01} \\ \vdots \\ \alpha_{0T}\end{matrix}}\begin{bmatrix} \alpha_{10} & \cdots & \alpha_{T0} \\ \alpha_{11} & \cdots & \alpha_{T1} \\ \vdots & \ddots & \vdots \\ \alpha_{1T} & \cdots & \alpha_{TT} \end{bmatrix} = \underset{T \times T}{\boldsymbol{A}} = <\boldsymbol{Q}, \boldsymbol{K}> = \underset{T \times d}{\boldsymbol{K}^T} \underset{d \times T}{\boldsymbol{Q}} = \underset{T \times d}{\begin{bmatrix} \boldsymbol{k}(0)_{d \times 1} \\ \boldsymbol{k}(1)_{d \times 1} \\ \\ \boldsymbol{k}(T)_{d \times 1} \end{bmatrix}} \underset{d \times T}{\begin{bmatrix} \underset{d \times 1}{\boldsymbol{q}(0)} & \underset{d \times 1}{\boldsymbol{q}(1)} & \cdots & \underset{d \times 1}{\boldsymbol{q}(T)} \end{bmatrix}}$$

Attention Score at $q(0)$

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



Attention Score at $q(0)$

$$\begin{aligned}\alpha_{00} &= \langle \boldsymbol{q}(0), \boldsymbol{k}(0) \rangle \\ \alpha_{01} &= \langle \boldsymbol{q}(0), \boldsymbol{k}(1) \rangle \\ &\dots \\ \alpha_{0T} &= \langle \boldsymbol{q}(0), \boldsymbol{k}(T) \rangle\end{aligned}$$

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

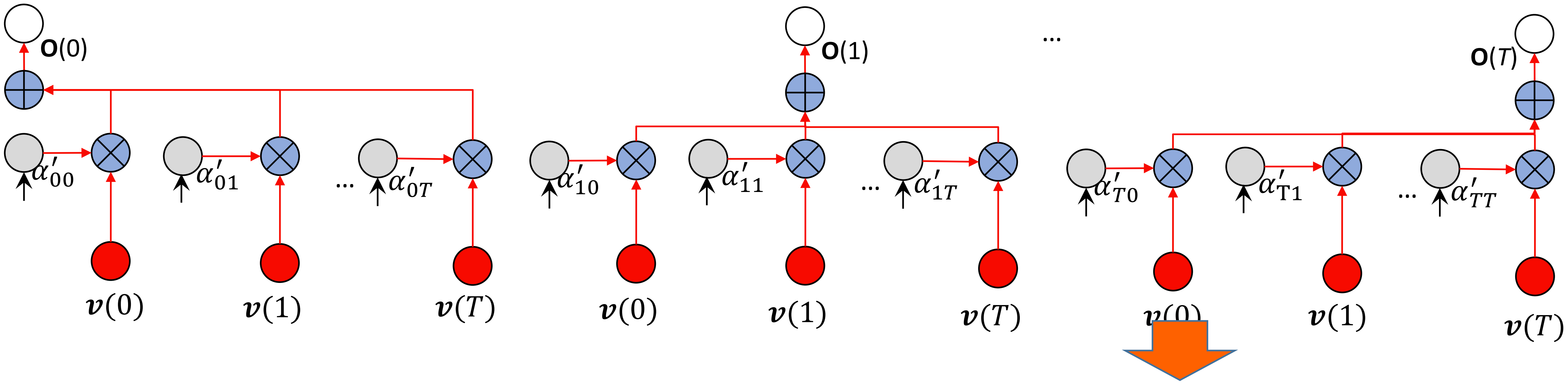
Attention Score Matrix

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

Attention Score at $q(0)$

$$\hat{\boldsymbol{A}} = \boldsymbol{softmax}(\boldsymbol{A})$$

用大矩陣直接運算O



$$\underset{d\times T}{\boldsymbol{O}} = \left[\underset{d\times 1}{\boxed{\boldsymbol{o}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{o}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{o}(T)}}\right]_{d\times T} = \underset{d\times T}{\boldsymbol{V}}\,\underset{T\times T}{\widehat{\boldsymbol{A}}} = \left[\underset{d\times 1}{\boxed{\boldsymbol{v}(0)}}\ \underset{d\times 1}{\boxed{\boldsymbol{v}(1)}} \cdots \underset{d\times 1}{\boxed{\boldsymbol{v}(T)}}\right]_{d\times T} \begin{bmatrix} \alpha'_{00} & \alpha'_{10} & \cdots & \alpha'_{T0} \\ \alpha'_{01} & \alpha'_{11} & \cdots & \alpha'_{T1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha'_{0T} & \alpha'_{1T} & \cdots & \alpha'_{TT} \end{bmatrix}_{T\times T}$$

矩陣直接運算Self-Attention

$$\underset{d\times T}{\mathbf{Q}} = \underset{d\times T}{\begin{bmatrix} \underset{d\times 1}{\boldsymbol{q}(0)} & \underset{d\times 1}{\boldsymbol{q}(1)} & \cdots & \underset{d\times 1}{\boldsymbol{q}(T)} \end{bmatrix}} = \underset{d\times d}{\boldsymbol{W}^{\boldsymbol{q}}} \underset{d\times T}{\begin{bmatrix} \underset{d\times 1}{\boldsymbol{x}(0)} & \underset{d\times 1}{\boldsymbol{x}(1)} & \cdots & \underset{d\times 1}{\boldsymbol{x}(T)} \end{bmatrix}}$$

$$\underset{d\times T}{\mathbf{K}} = \underset{d\times T}{\begin{bmatrix} \underset{d\times 1}{\boldsymbol{k}(0)} & \underset{d\times 1}{\boldsymbol{k}(1)} & \cdots & \underset{d\times 1}{\boldsymbol{k}(T)} \end{bmatrix}} = \underset{d\times d}{\boldsymbol{W}^{\boldsymbol{k}}} \underset{d\times T}{\begin{bmatrix} \underset{d\times 1}{\boldsymbol{x}(0)} & \underset{d\times 1}{\boldsymbol{x}(1)} & \cdots & \underset{d\times 1}{\boldsymbol{x}(T)} \end{bmatrix}}$$

$$\underset{d\times T}{\mathbf{V}} = \underset{d\times T}{\begin{bmatrix} \underset{d\times 1}{\boldsymbol{v}(0)} & \underset{d\times 1}{\boldsymbol{v}(1)} & \cdots & \underset{d\times 1}{\boldsymbol{v}(T)} \end{bmatrix}} = \underset{d\times d}{\boldsymbol{W}^{\boldsymbol{v}}} \underset{d\times T}{\begin{bmatrix} \underset{d\times 1}{\boldsymbol{x}(0)} & \underset{d\times 1}{\boldsymbol{x}(1)} & \cdots & \underset{d\times 1}{\boldsymbol{x}(T)} \end{bmatrix}}$$

$$\underset{T\times T}{\boldsymbol{A}} = <\boldsymbol{Q}, \boldsymbol{K}> = \underset{T\times d}{\boldsymbol{K}^T} \underset{d\times T}{\boldsymbol{Q}}$$

$$\hat{\boldsymbol{A}} = \boldsymbol{softmax}(\boldsymbol{A})$$

$$\underset{d\times T}{\boldsymbol{O}} = \boldsymbol{V}\hat{\boldsymbol{A}}$$

論文寫法

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

$$\boxed{\boldsymbol{O} = \boldsymbol{V} \times \boldsymbol{softmx}(\boldsymbol{K}^T\boldsymbol{Q})}$$

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

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

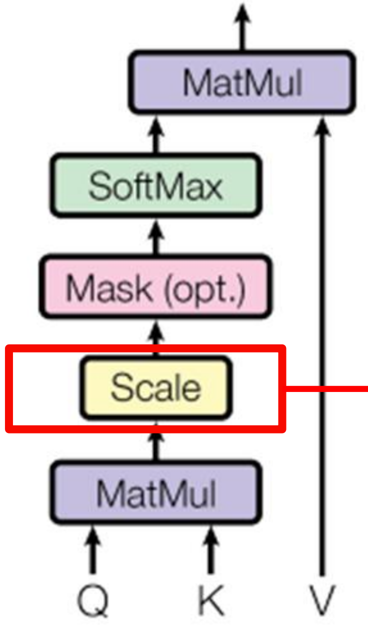
$$\boldsymbol{O} = \boldsymbol{V} \times \boldsymbol{softmx}(\boldsymbol{K}^T\boldsymbol{Q})$$

差異在矩陣擺法，和一個$\boxed{\sqrt{d_k}}$

$$\underbrace{\begin{bmatrix} \boldsymbol{k}(0)_{d\times 1} \\ \boldsymbol{k}(1)_{d\times 1} \\ \boldsymbol{k}(T)_{d\times 1} \end{bmatrix}}_{T\times d} \underbrace{\begin{bmatrix} \underbrace{\boldsymbol{q}(0)}_{d\times 1} & \underbrace{\boldsymbol{q}(1)}_{d\times 1} & \cdots & \underbrace{\boldsymbol{q}(T)}_{d\times 1} \end{bmatrix}}_{d\times T}$$

$\mathbf{K}^T$ $T\times d$

$\mathbf{Q}$ $d\times T$



K和Q在不同時間點上
d個維度相乘後的總和。

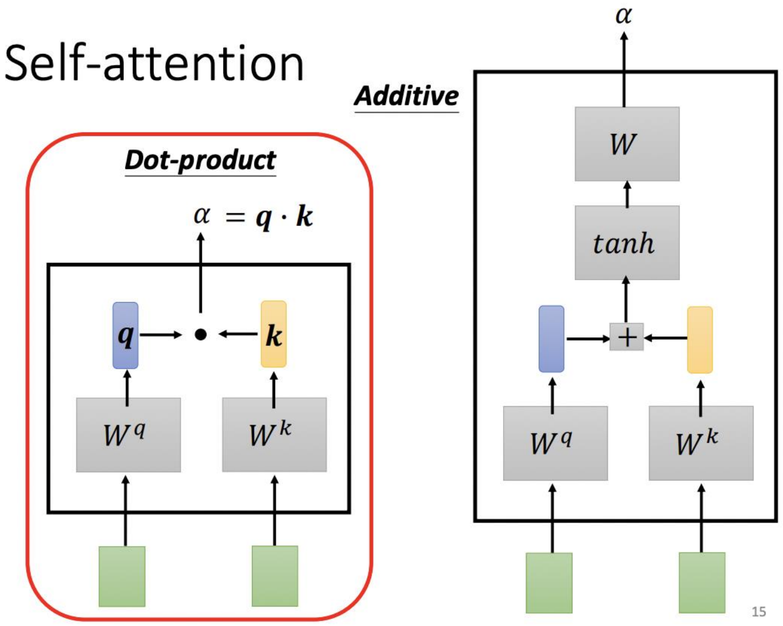
所以維度越大值越大，因此需要正規化(在右圖的Scale)，將每一個值域拉到一致。

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

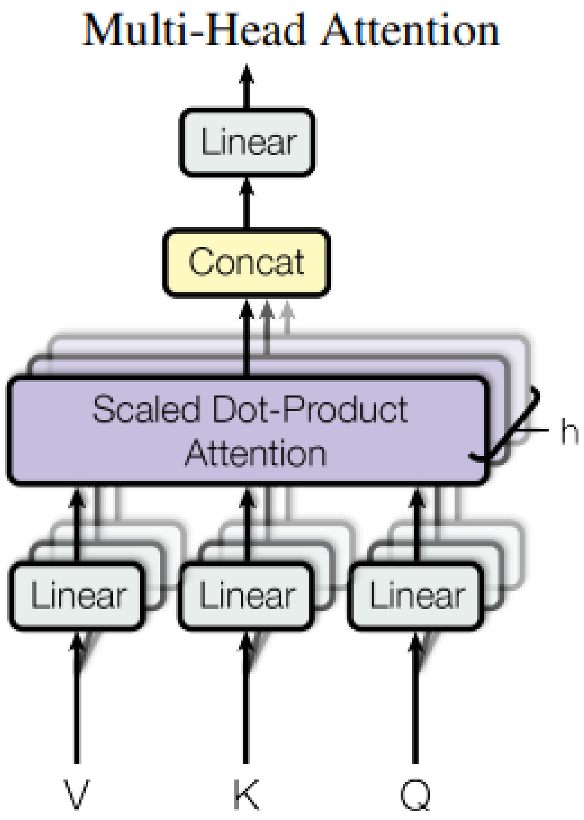
The two most commonly used attention functions are additive attention [2], and dot-product (multi-plicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward 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.

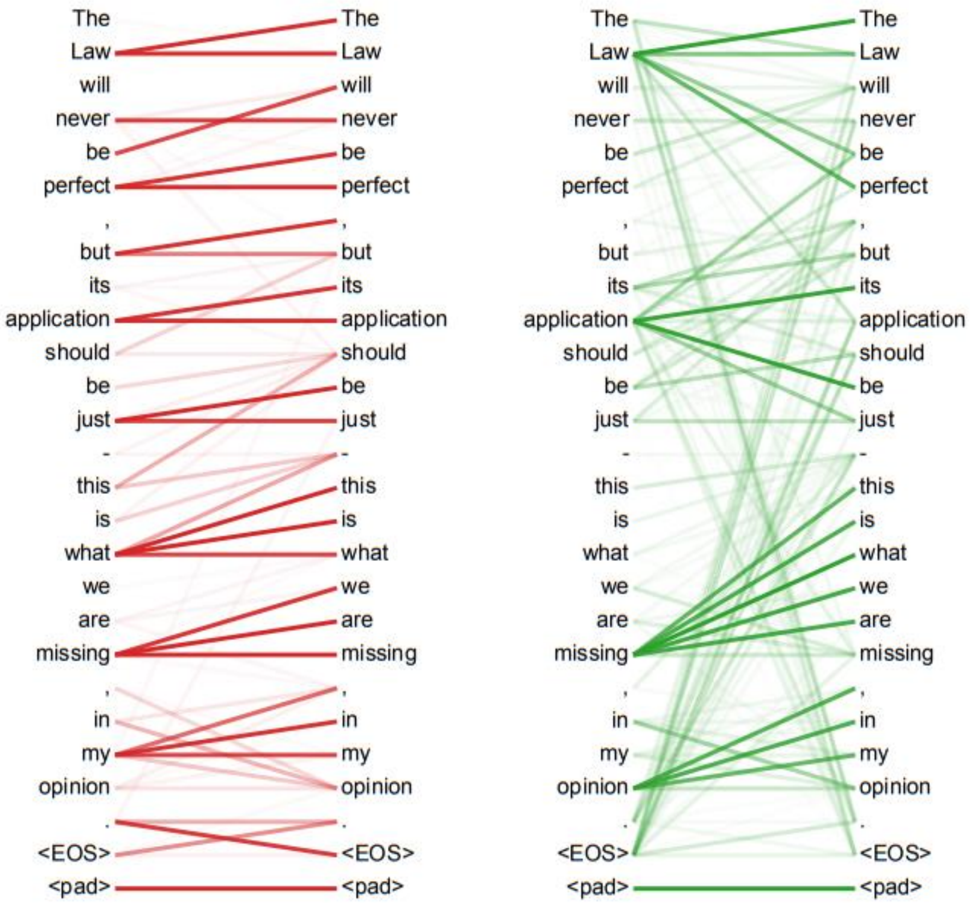


Reference: 李宏毅老師

Multi-head Self-attention

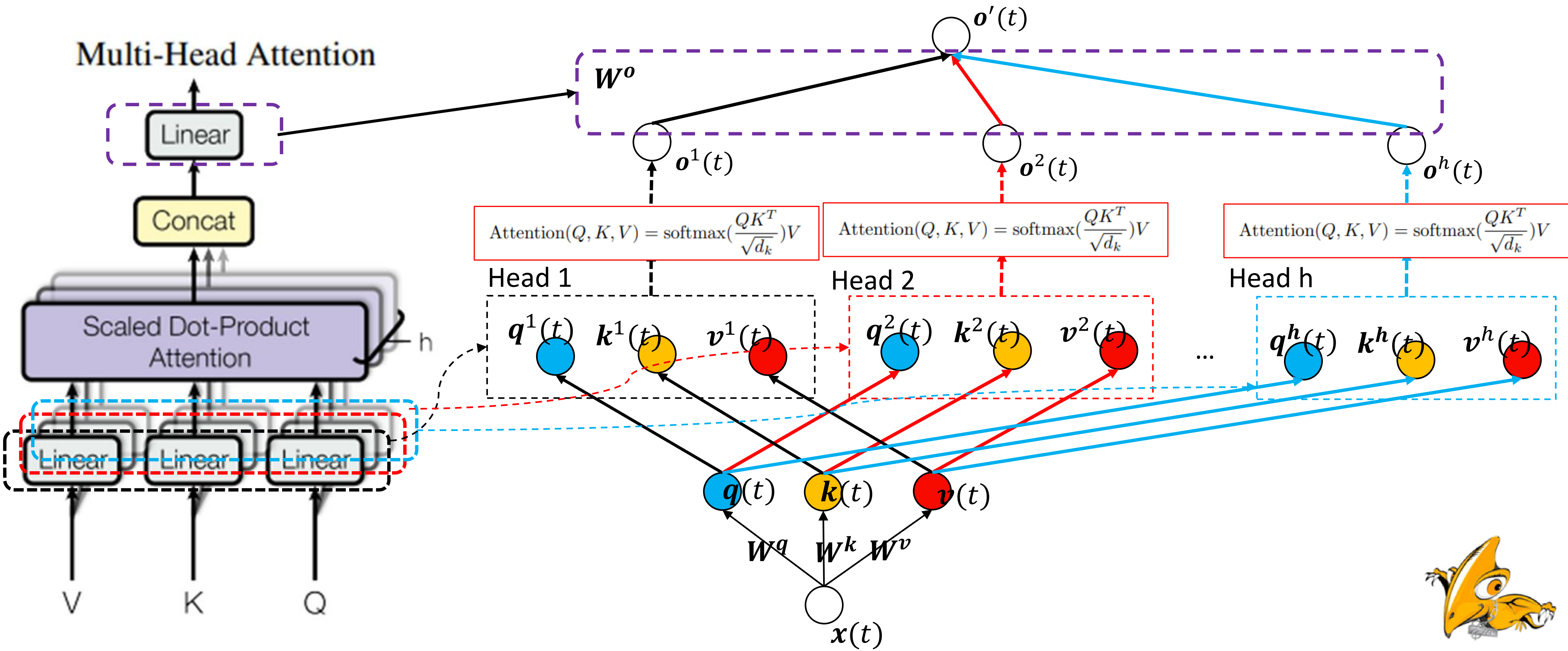
- 相關性在不同的位置可能有不同類別，所以需要多Multi-head來讓模型可以得到更多不同位置的表示。





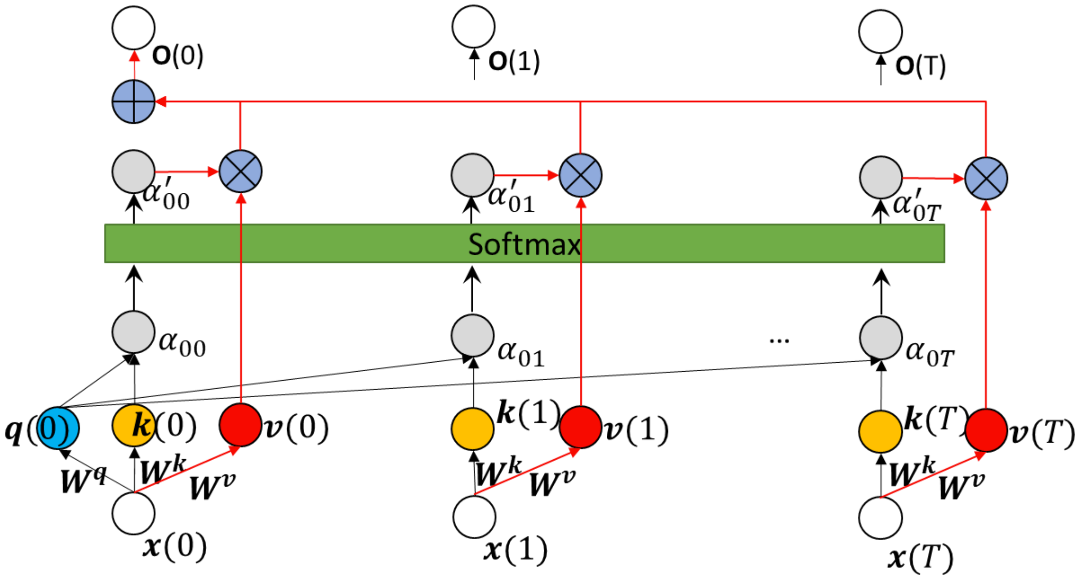
Many of the attention heads exhibit behaviour that seems related to the structure of the sentence.

Multi-head Self-attention



Positional Encoding

輸入是有訊續的$\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(T)$，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($\boldsymbol{e}_i, \forall i = 0,1, \ldots, T$)，整個過程稱為position encoding。

Positional Encoding

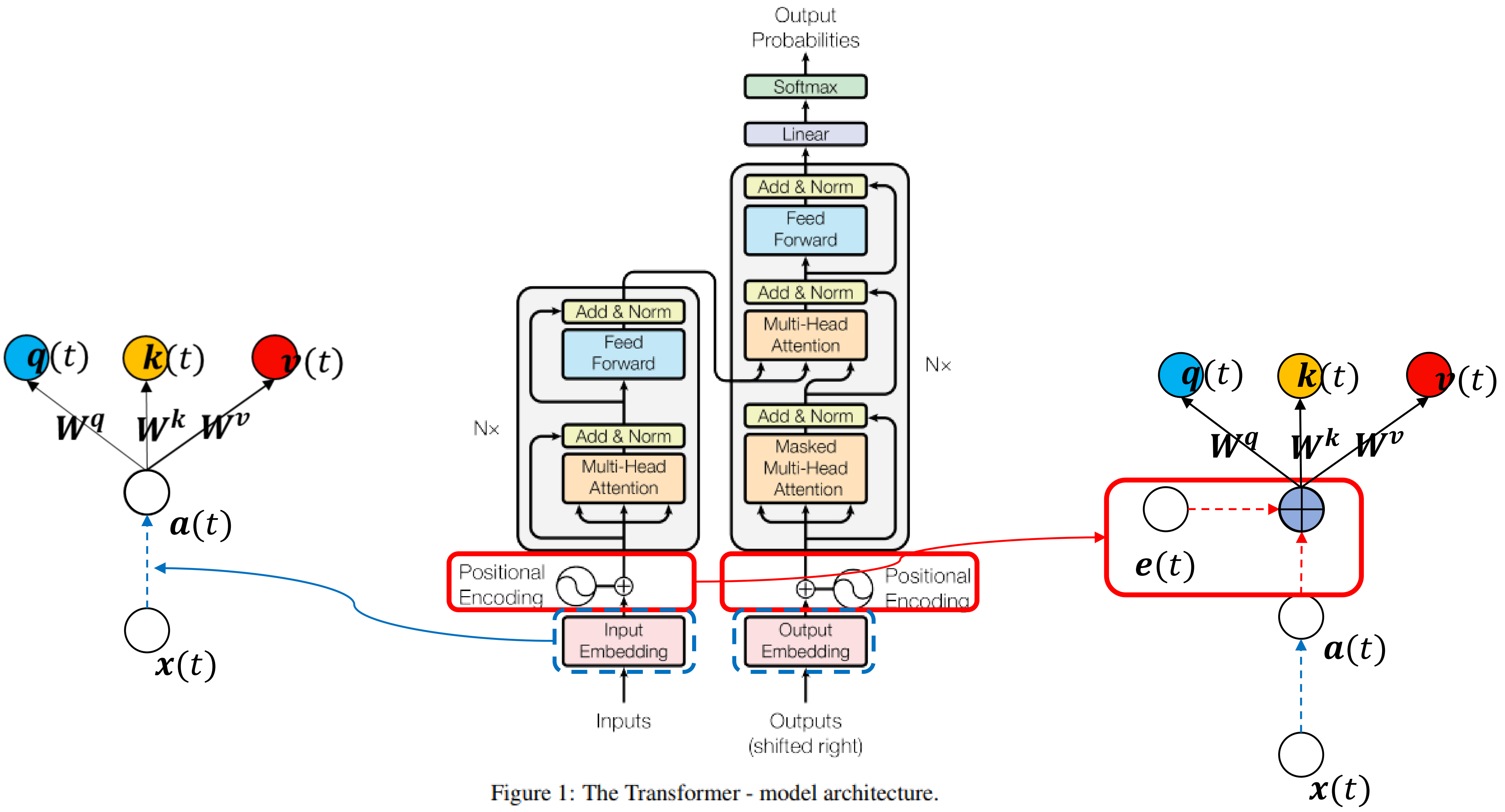
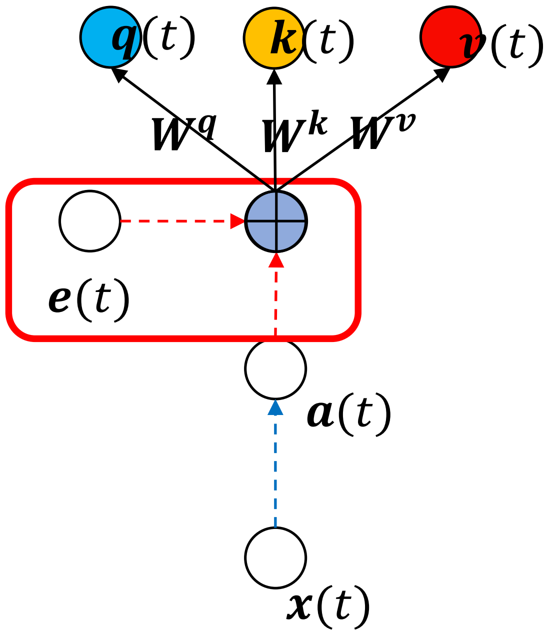


Figure 1: The Transformer - model architecture.

Positional Encoding



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

$$\boldsymbol{e}(1) = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=1} \qquad \boldsymbol{a}(1) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=1}$$

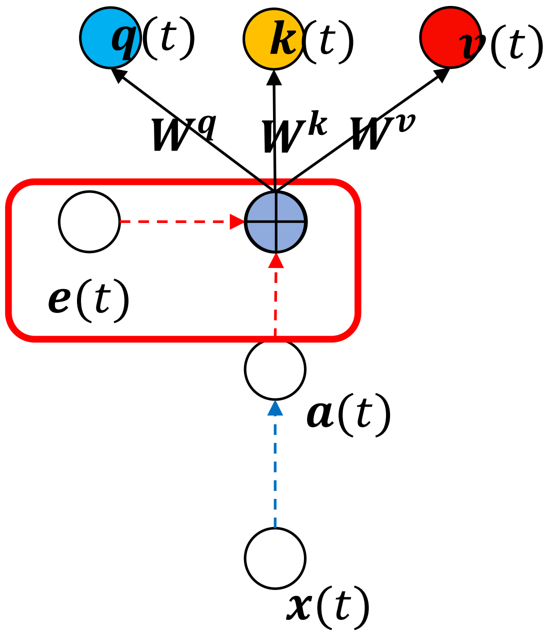
$$\boldsymbol{a}(2) + \boldsymbol{e}(2) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=2} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=2}$$

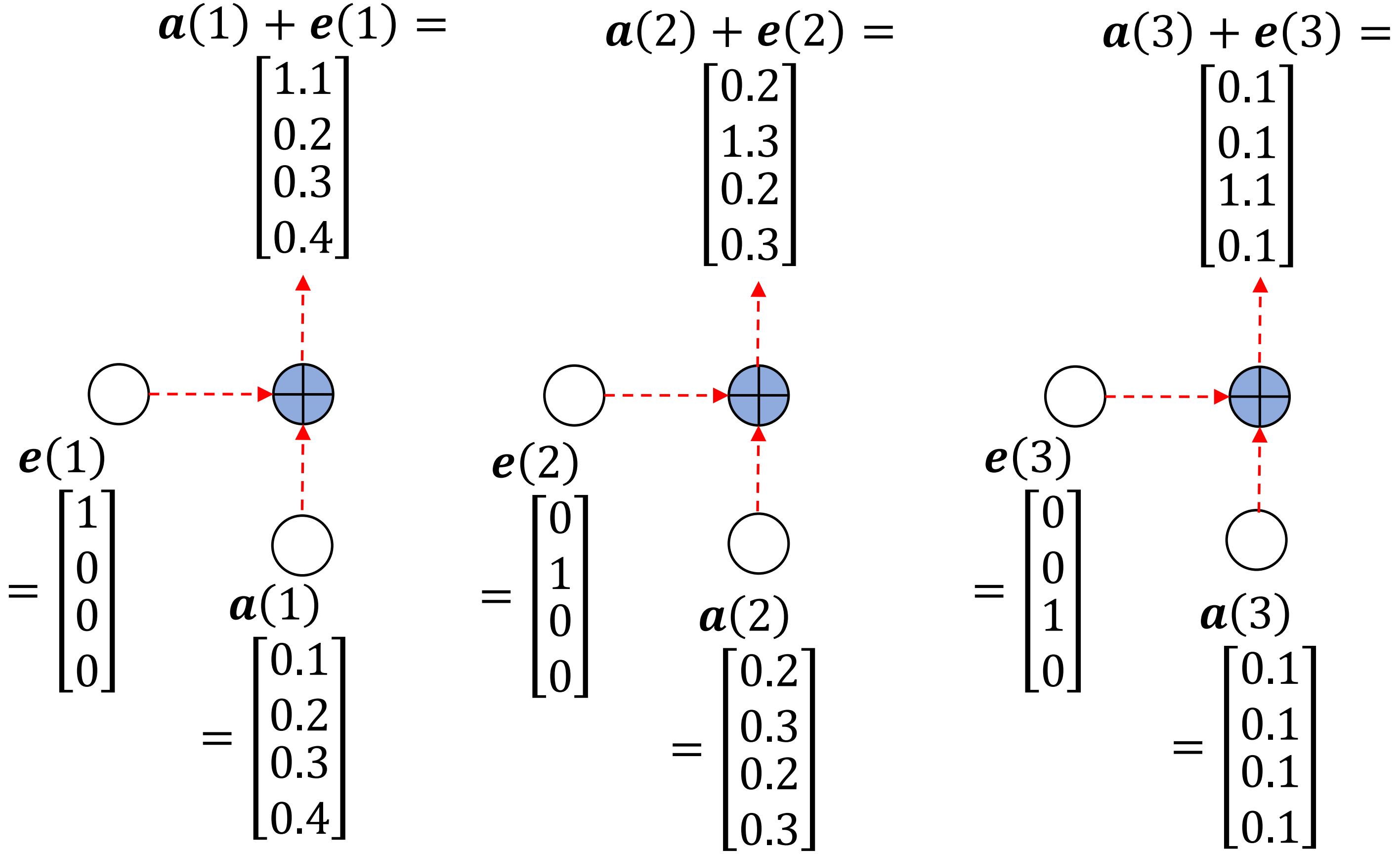
$$\boldsymbol{e}(2) = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=1} \qquad \boldsymbol{a}(2) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=2}$$

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

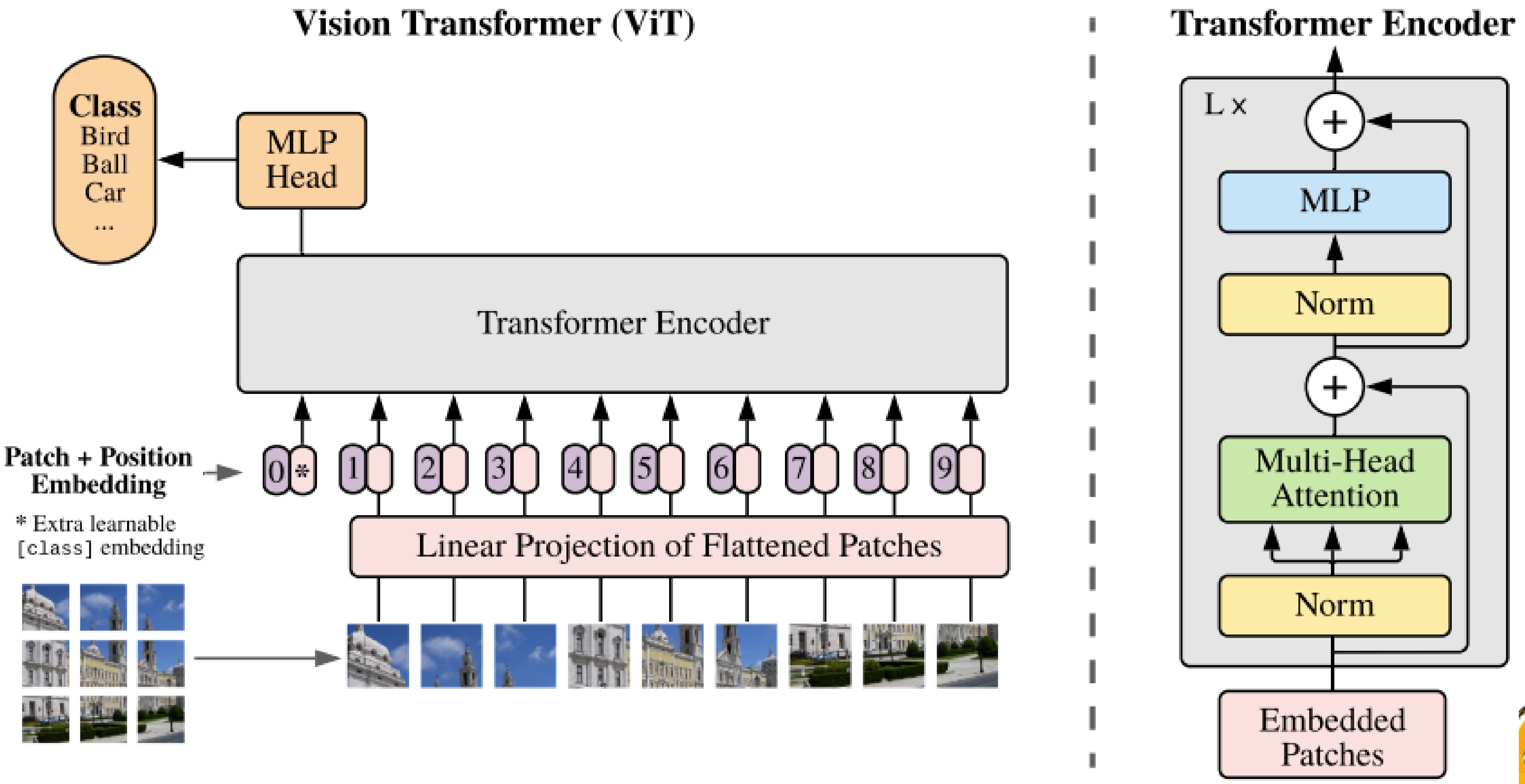
$$\boldsymbol{e}(3) = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_{t=1} \qquad \boldsymbol{a}(T=3) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}_{t=3}$$

Positional Encoding

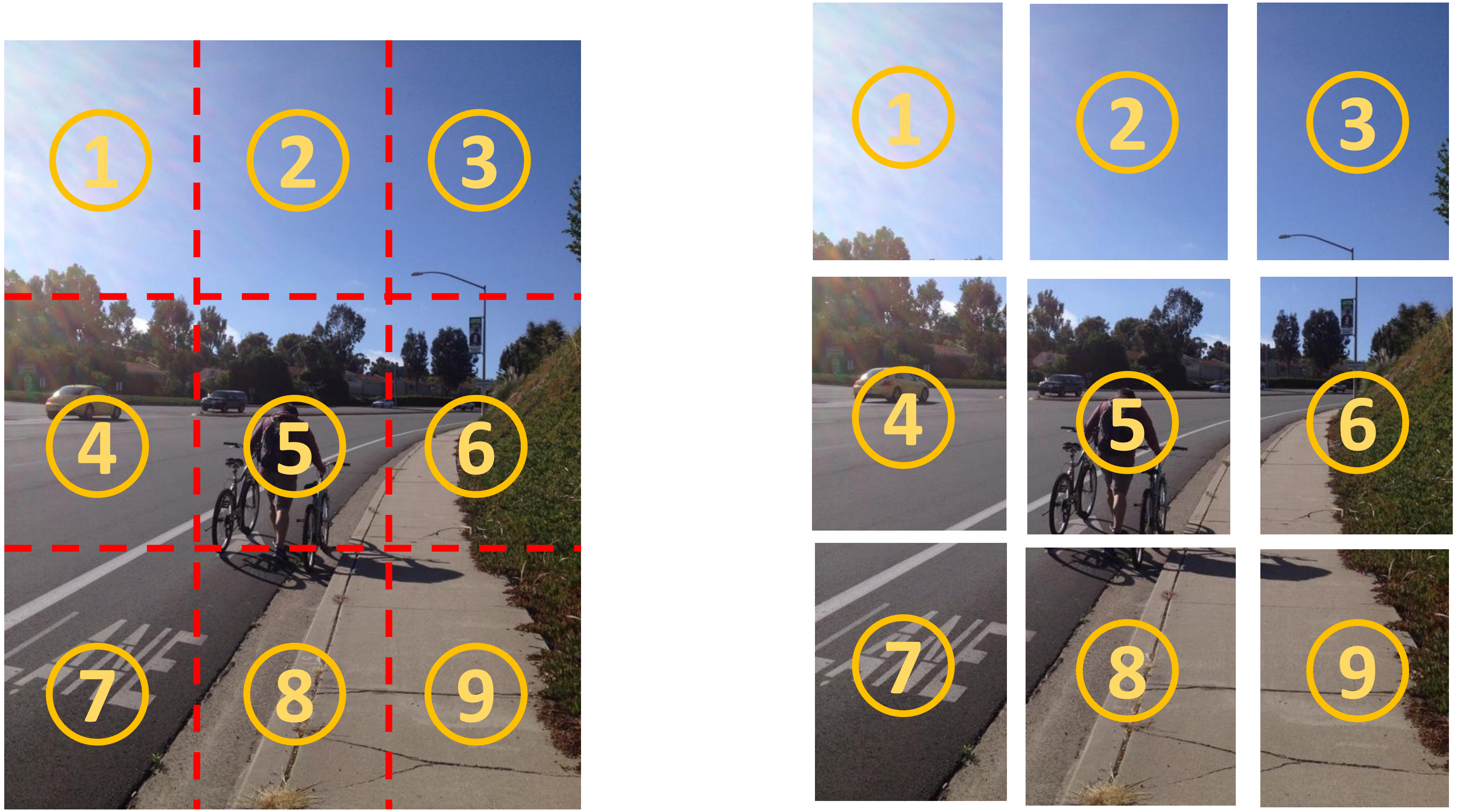




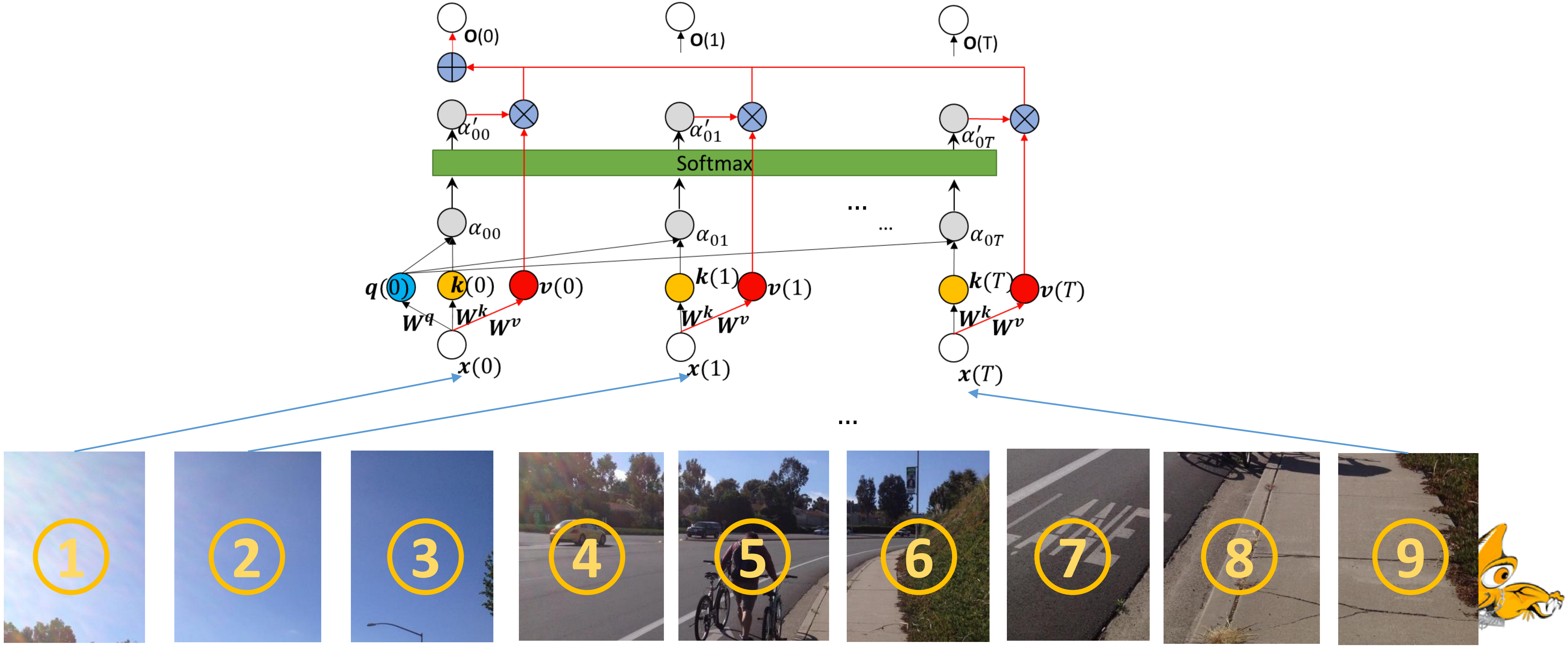
Vision Transformer(ViT)



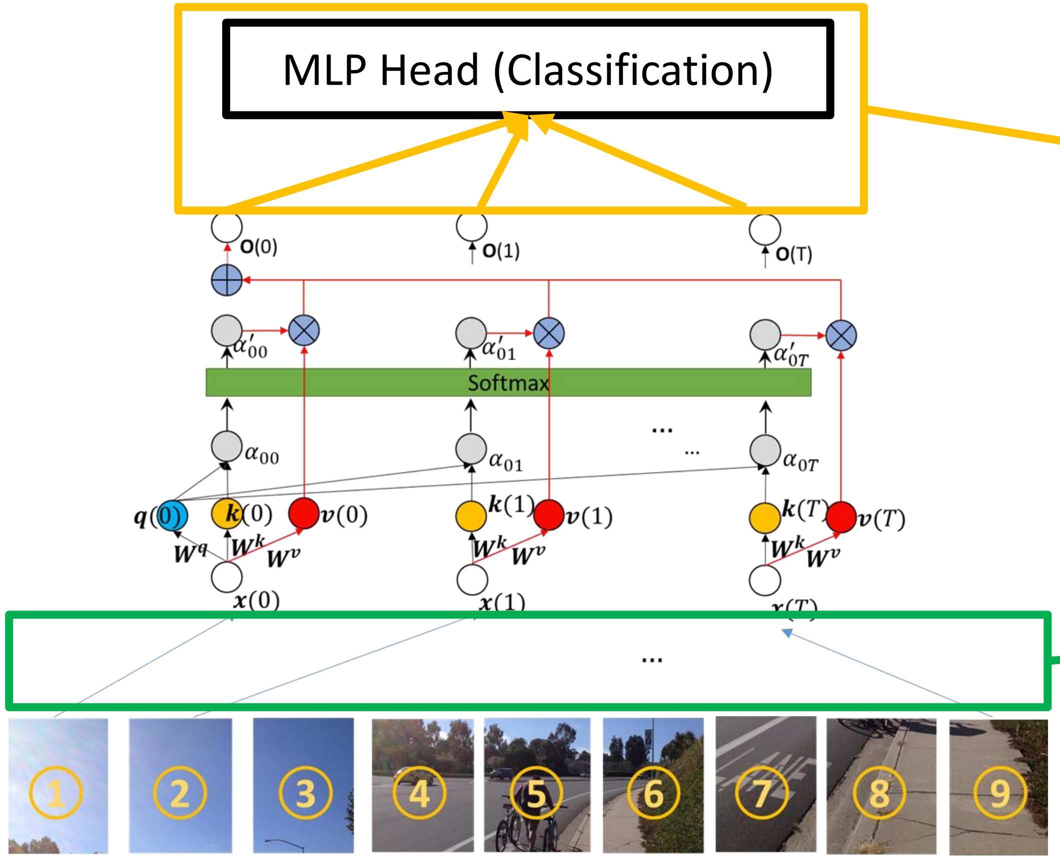
Vision Transformer(ViT)

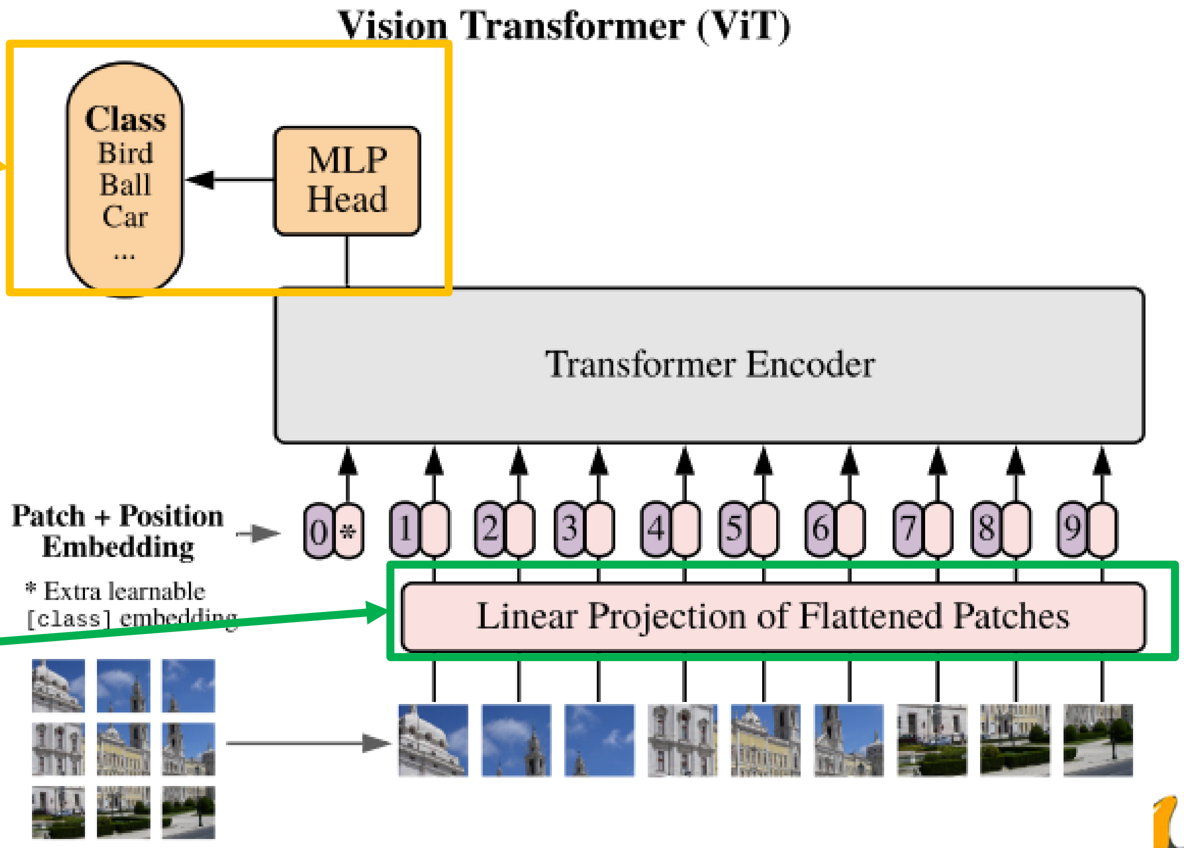


Vision Transformer(ViT)



Vision Transformer(ViT)



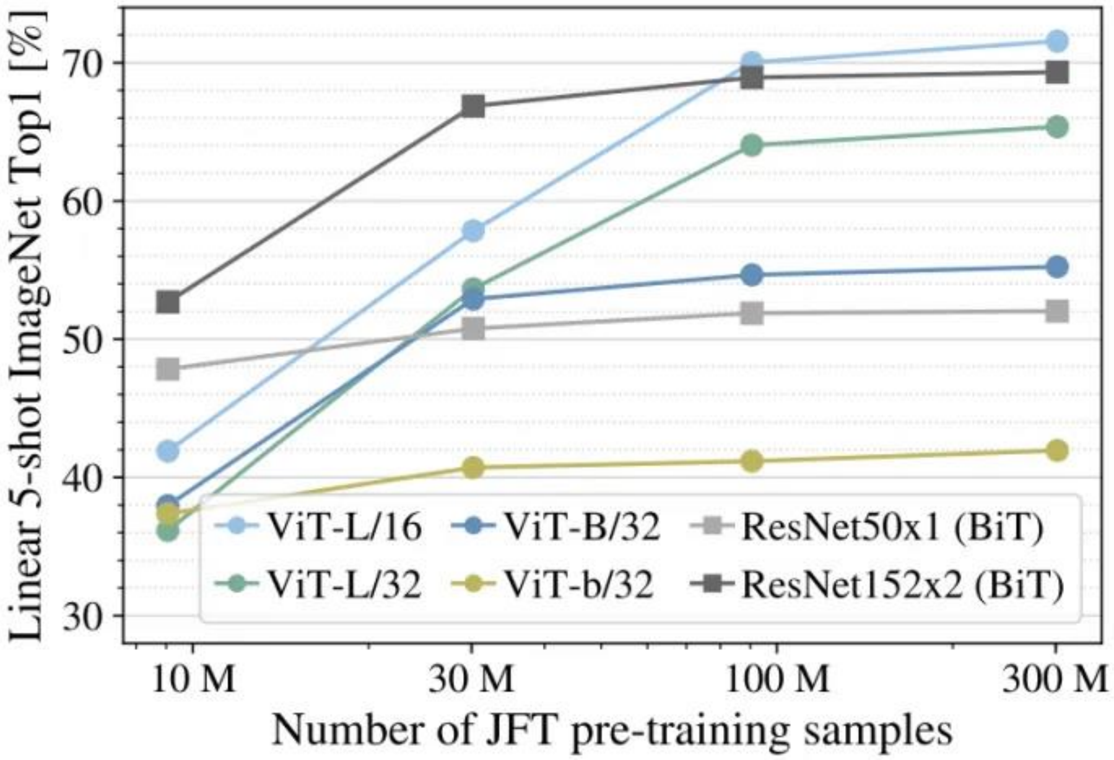


Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

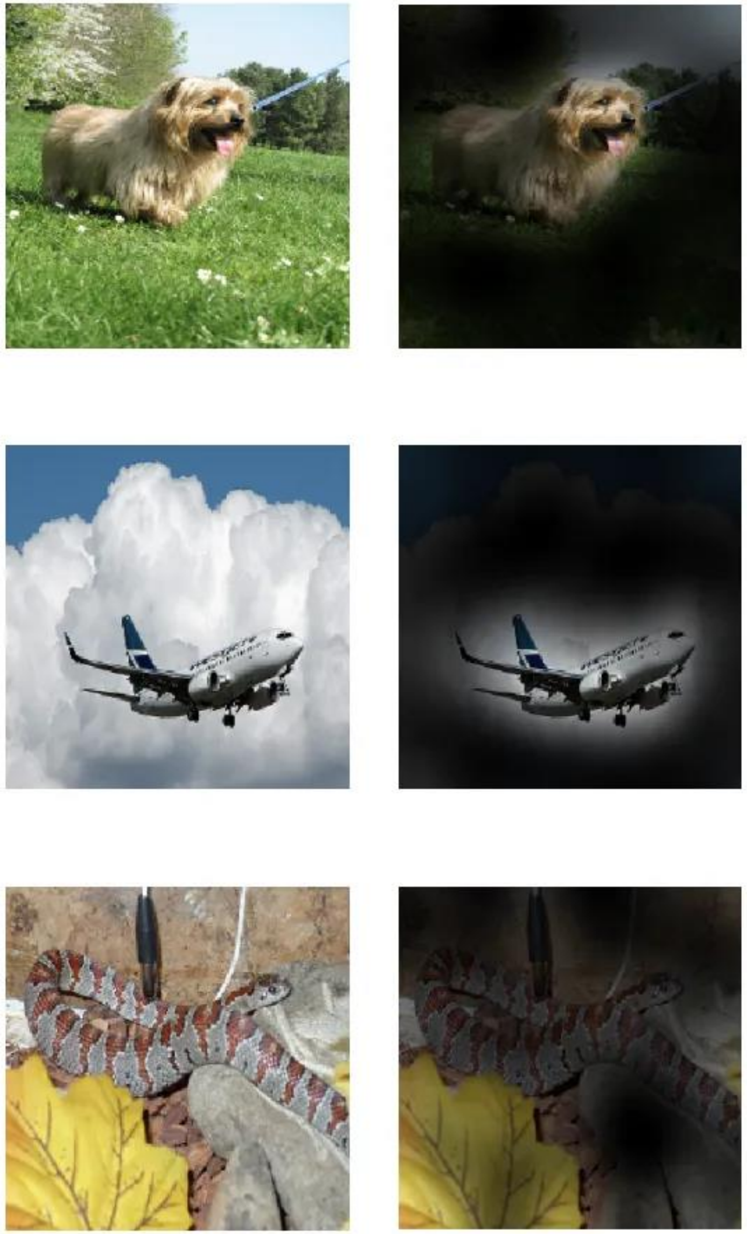
ViT-L/16表示的是large model使用的是16*16的patch

	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** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k



Big Transform: BiT

Input Attention



CNN and Self-Attention

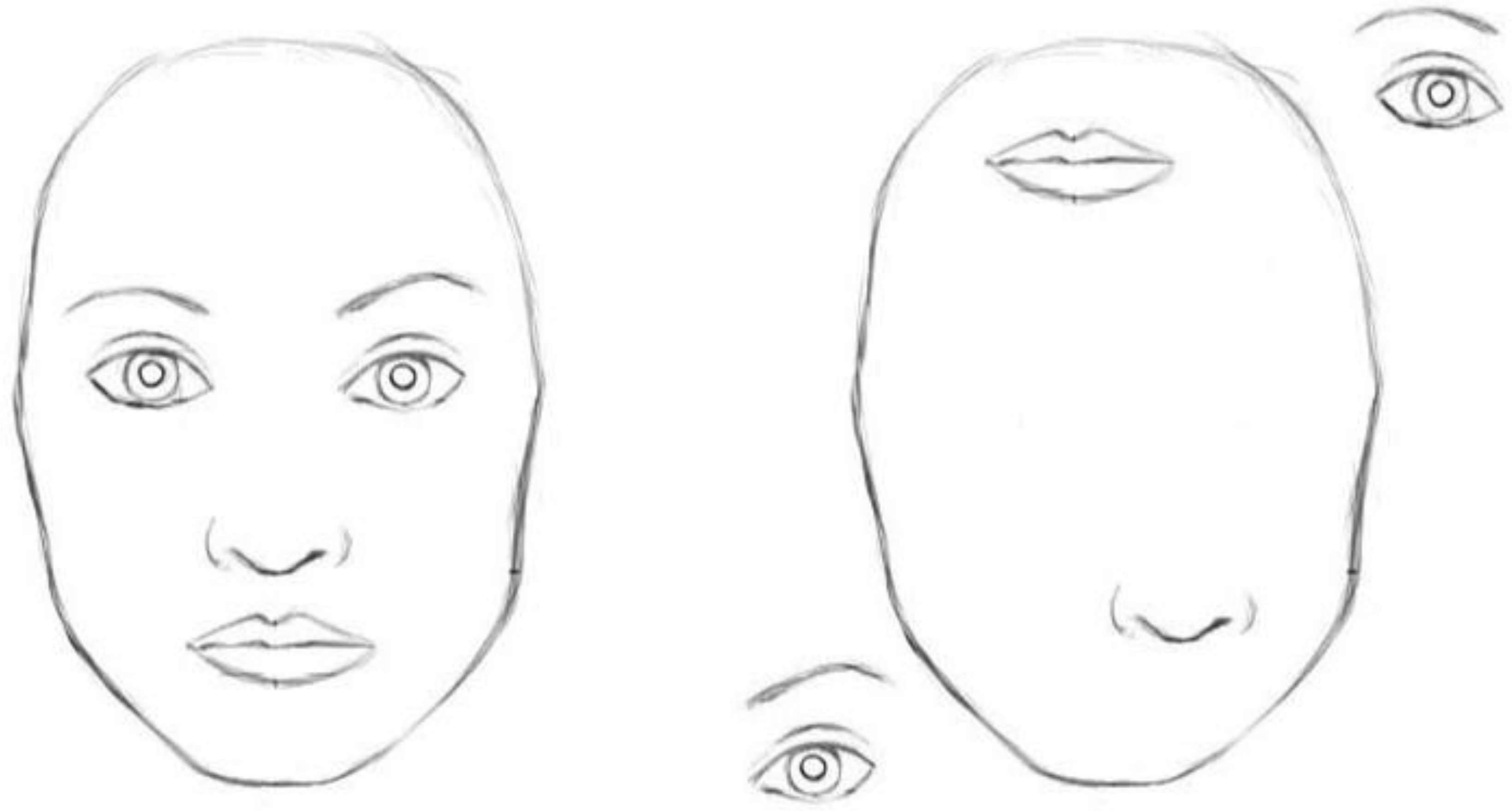
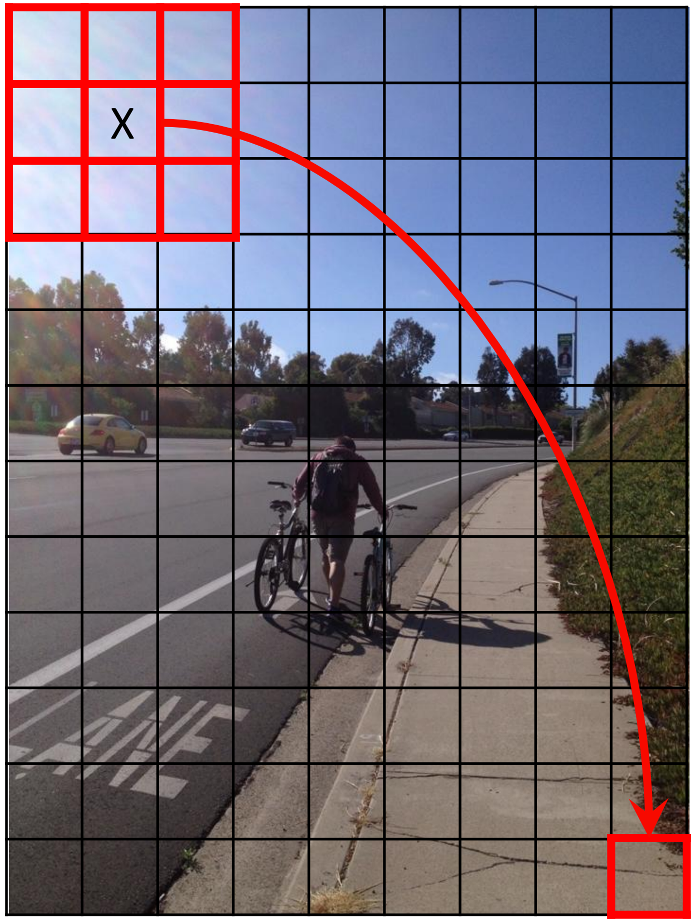


Fig 1: Both images are similar for a CNN as it does not consider the relative positioning of facial features

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

CNN and Self-Attention

CNN
Receptive field



Self - Attention