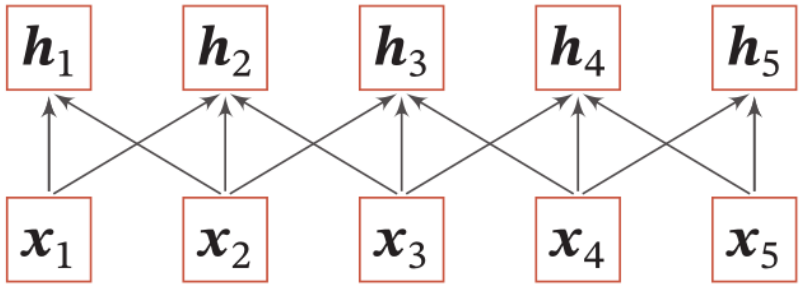
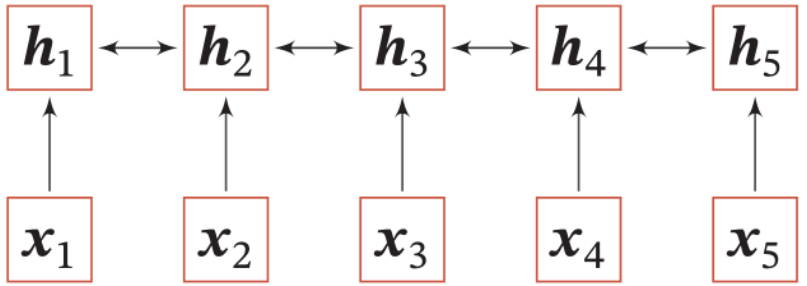
自注意力模型

自注意力模型

- 当使用神经网络来处理一个变长的向量序列时，我们通常可以使用卷积网络或循环网络进行编码来得到一个相同长度的输出向量序列。



(a) 卷积网络

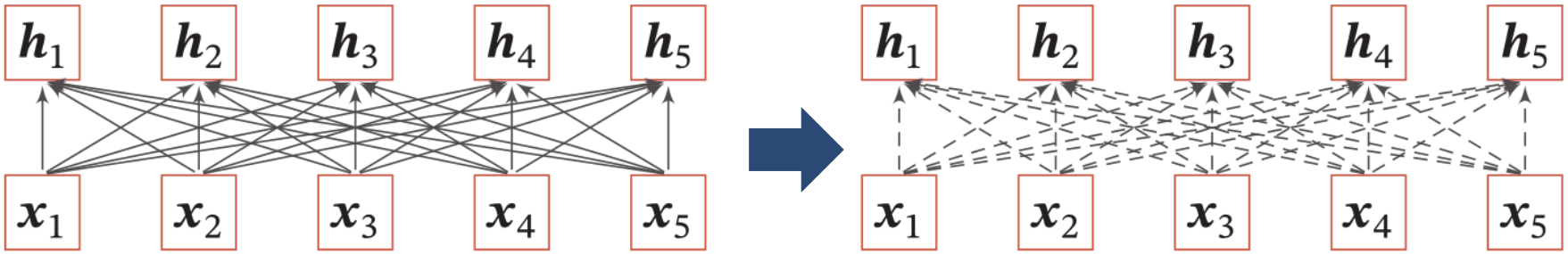


(b) 双向循环网络

只建模了输入信息的局部依赖关系

自注意力模型

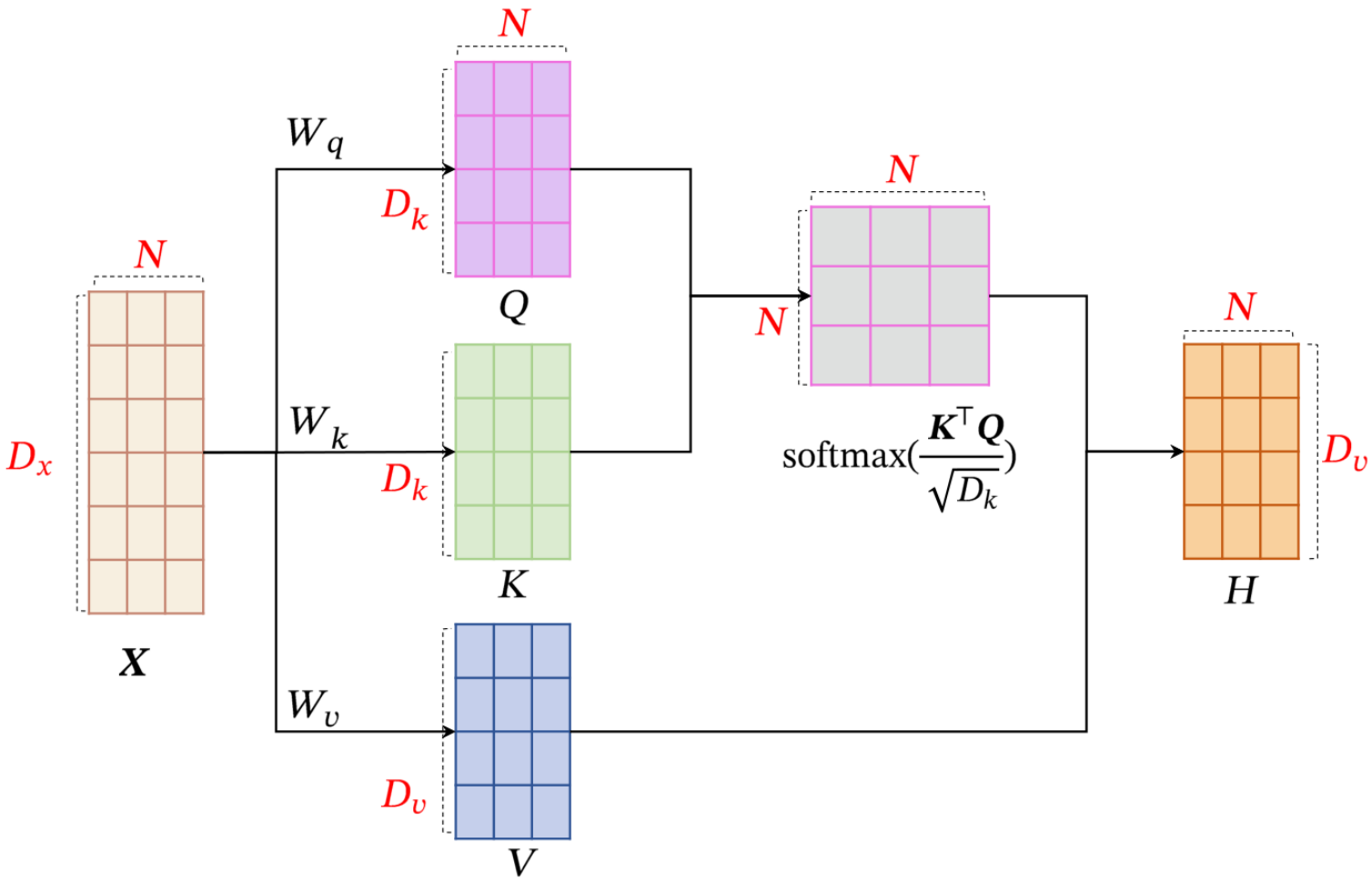
- 如何建立非局部（Non-local）的依赖关系
  - 全连接?



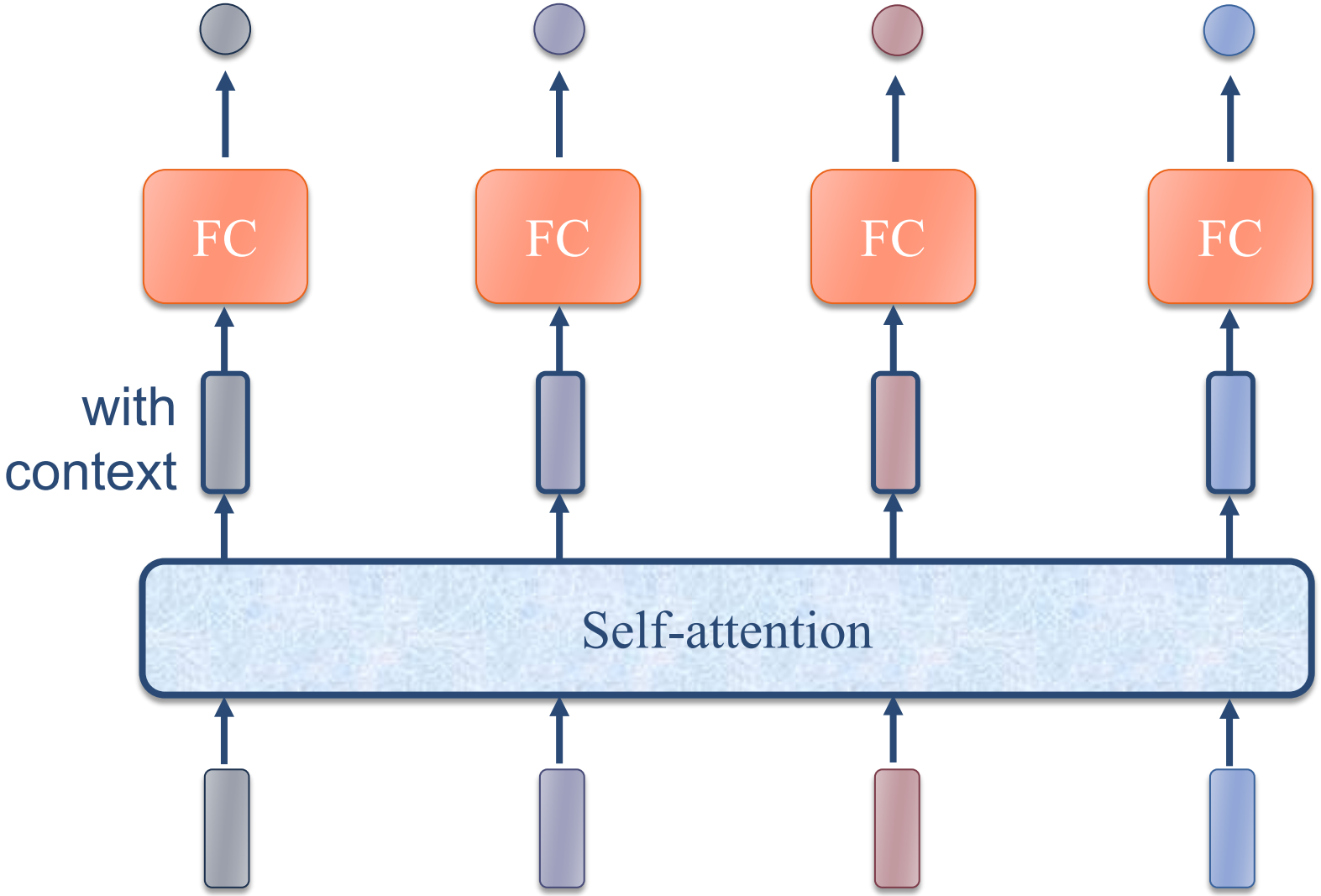
(a) 全连接模型　　(b) 自注意力模型

无法处理变长问题　　连接权重α_{ij} 由注意力机制动态生成

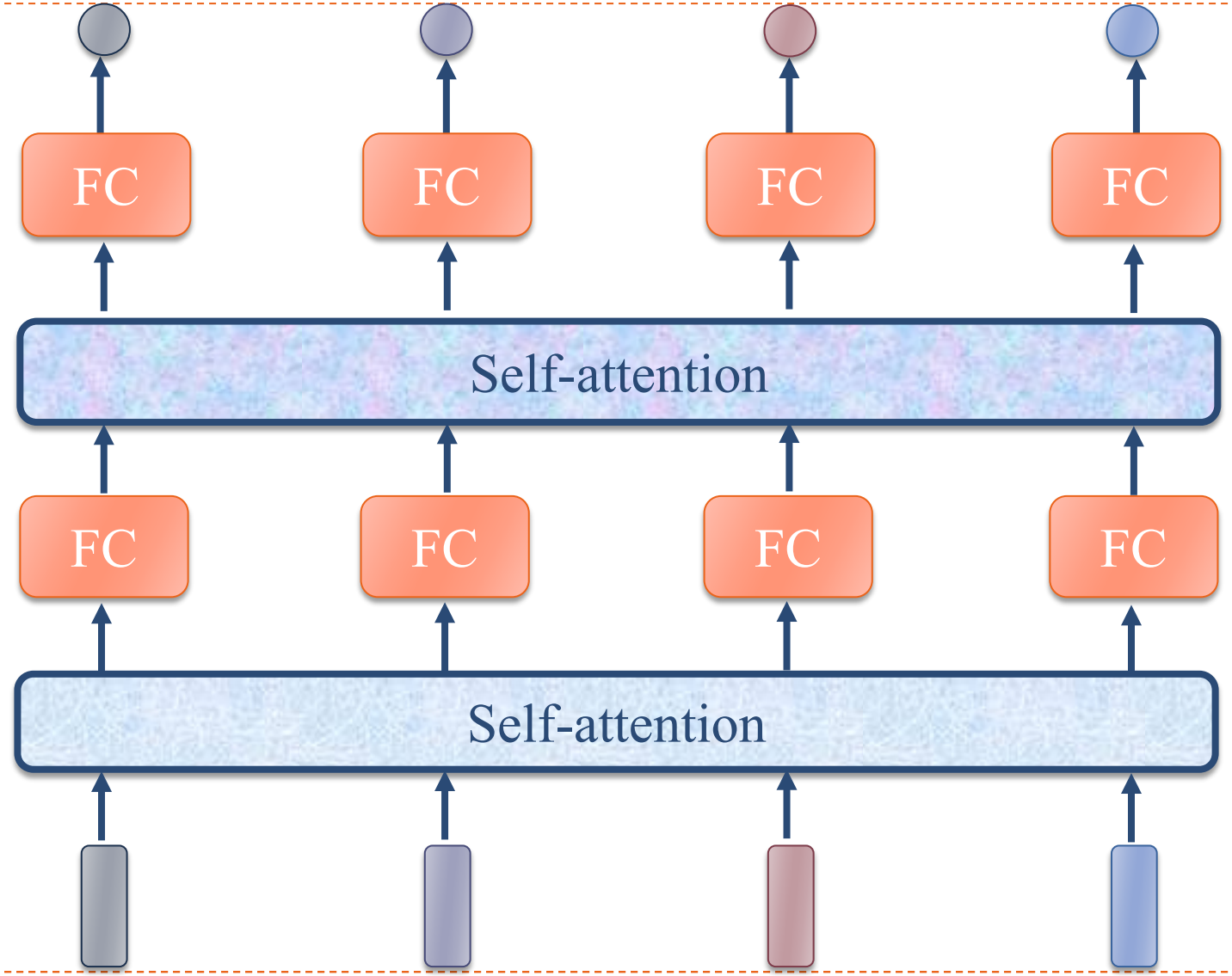
QKV模式（Query-Key-Value）



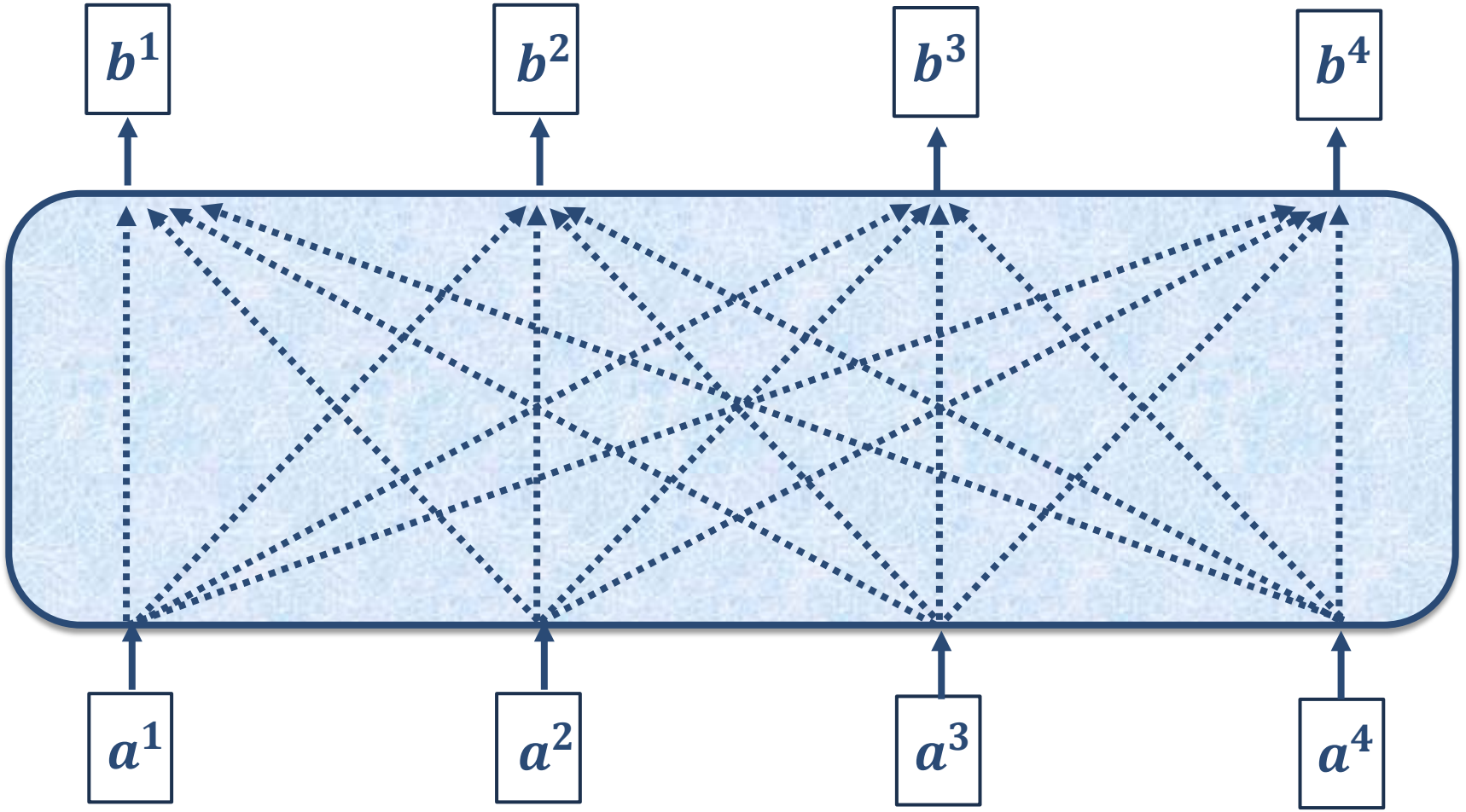
自注意力模型



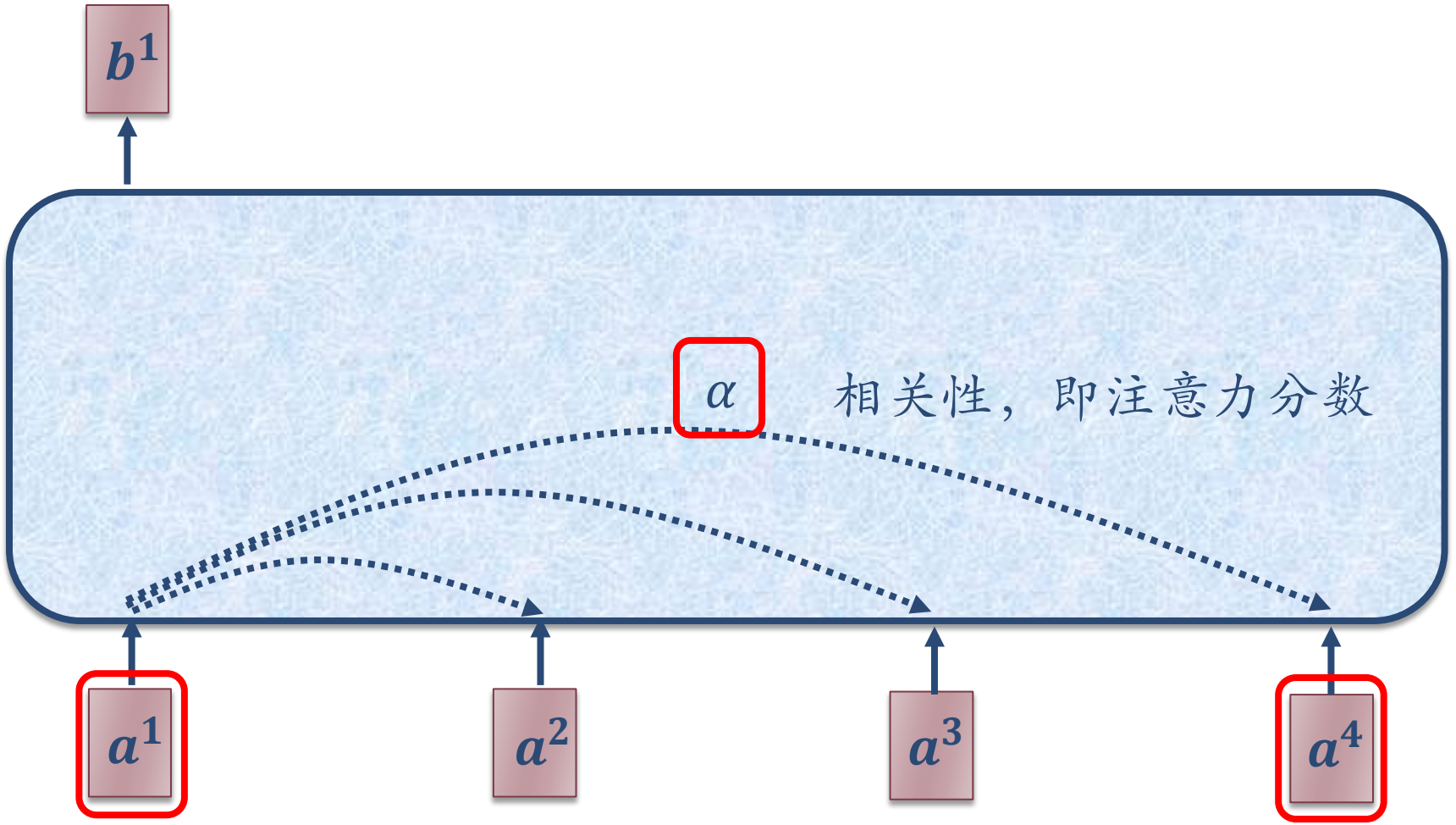
自注意力模型



自注意力模型

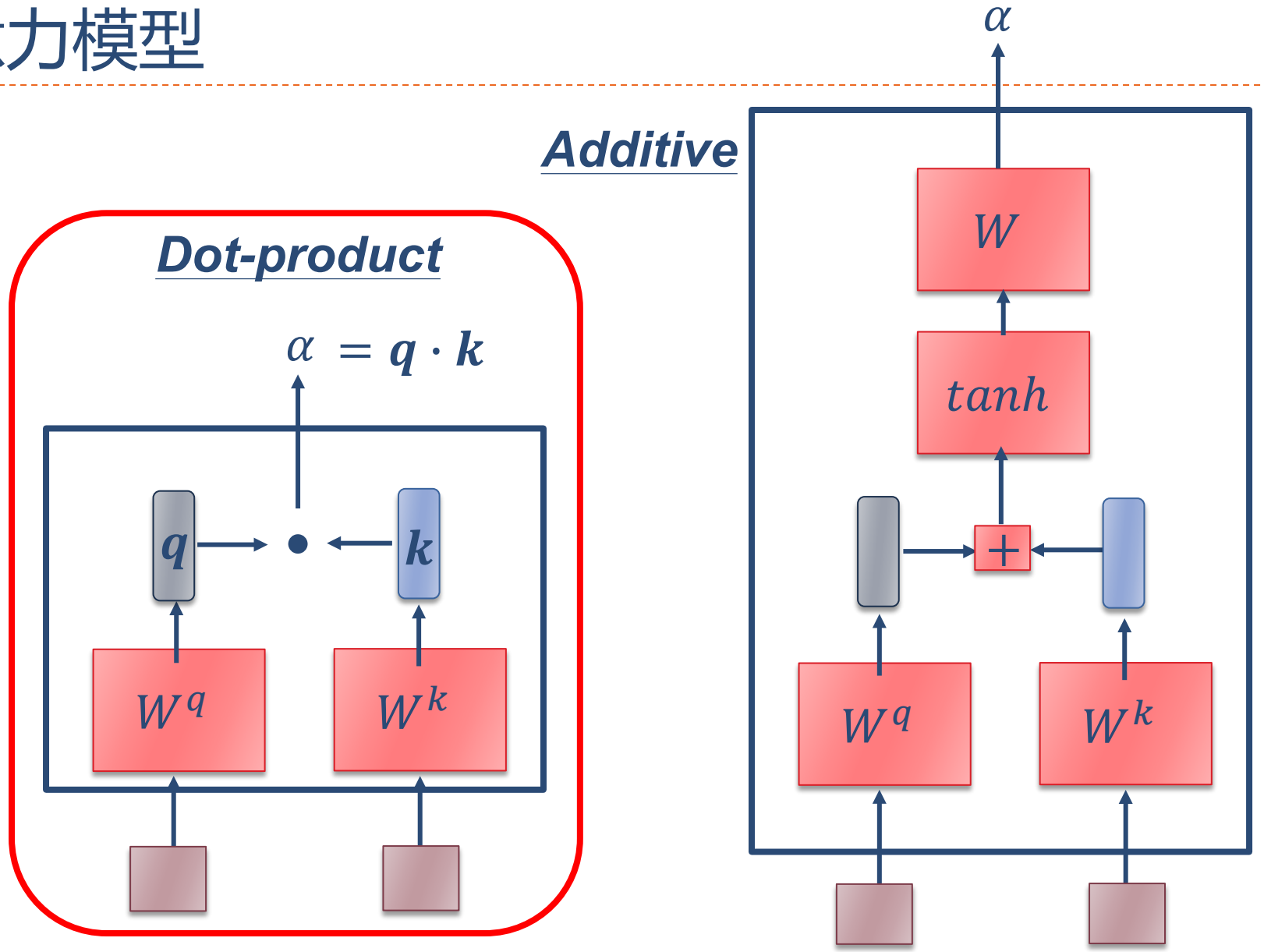


自注意力模型



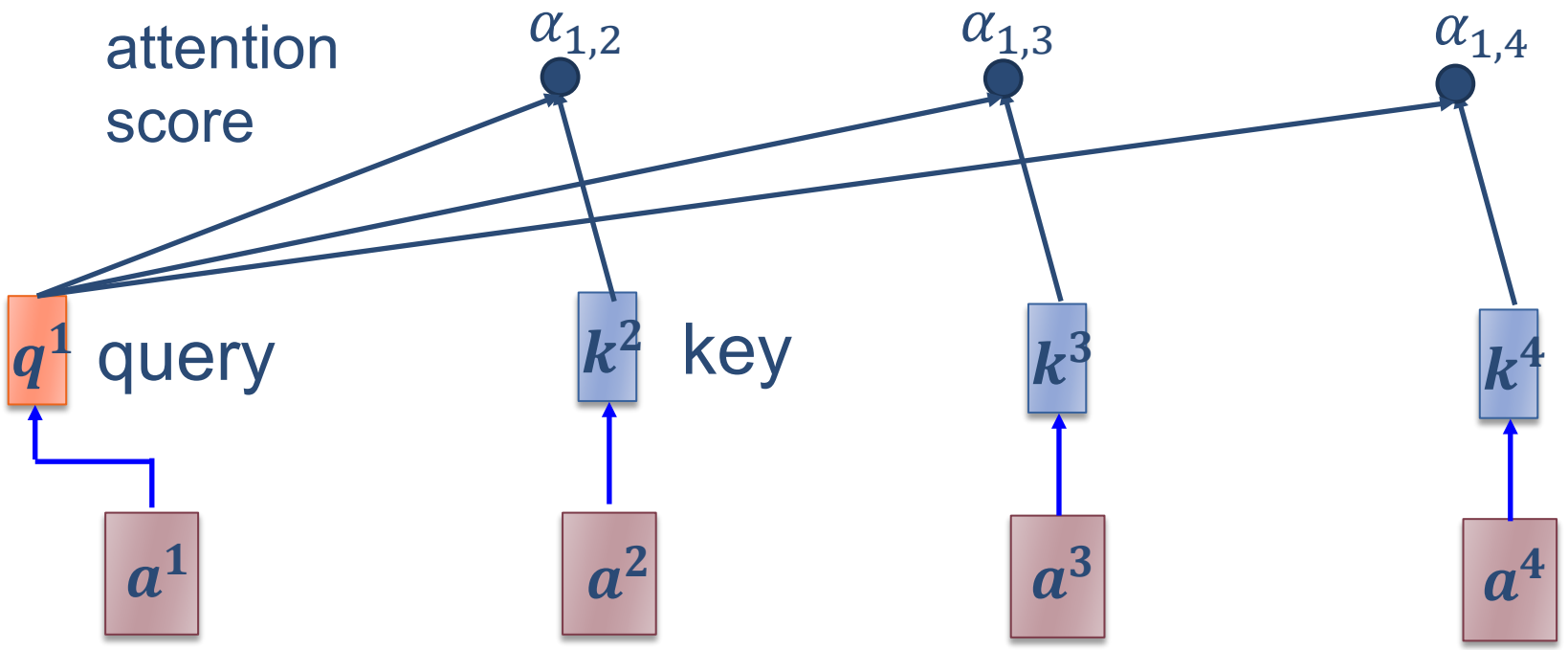
找出序列中相关的向量

自注意力模型



自注意力模型

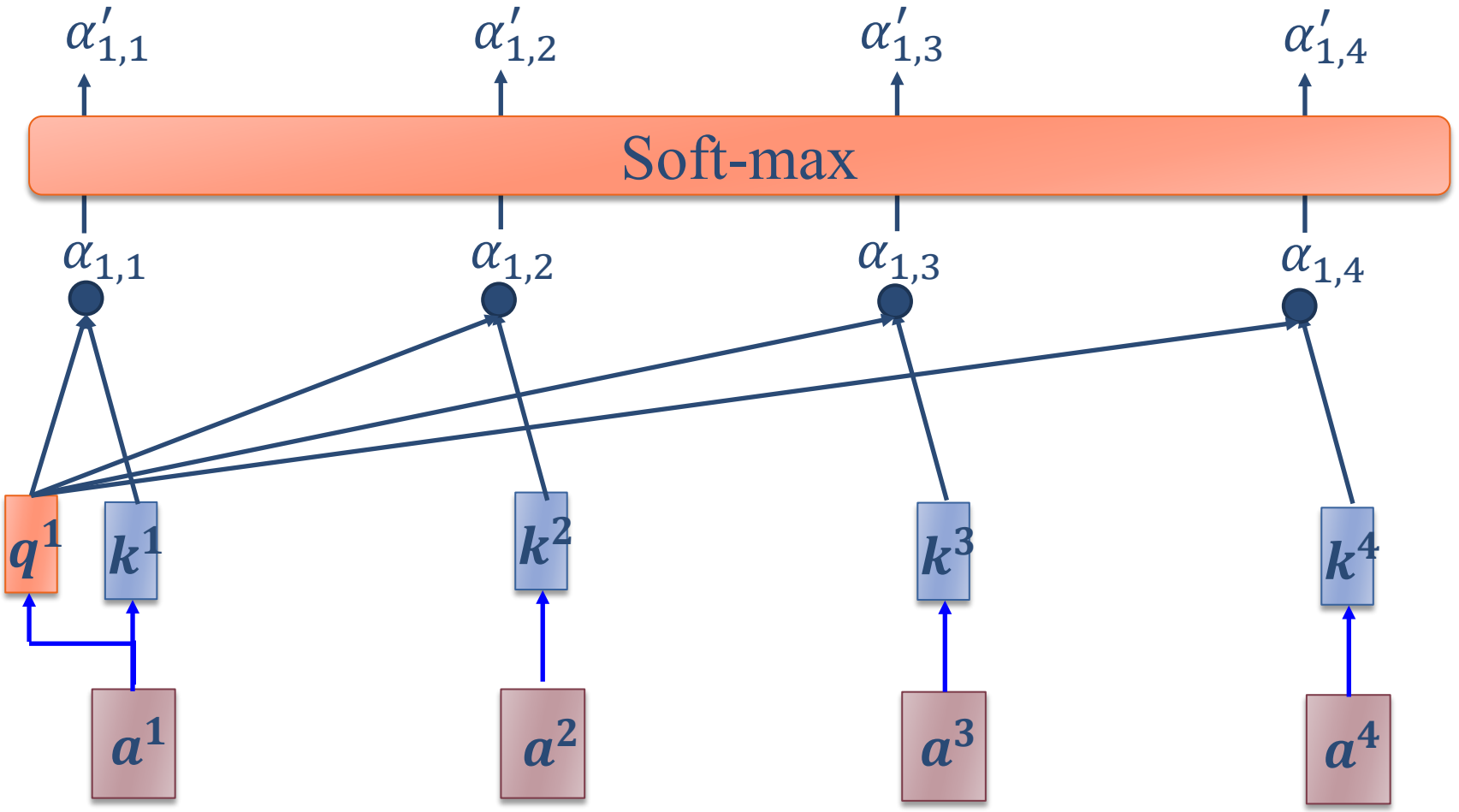
$$\alpha_{1,2} = \boldsymbol{q^1} \cdot \boldsymbol{k^2} \qquad \alpha_{1,3} = \boldsymbol{q^1} \cdot \boldsymbol{k^3} \qquad \alpha_{1,4} = \boldsymbol{q^1} \cdot \boldsymbol{k^4}$$



$$\boldsymbol{q^1} = W^q \boldsymbol{a^1} \qquad \boldsymbol{k^2} = W^k \boldsymbol{a^2} \qquad \boldsymbol{k^3} = W^k \boldsymbol{a^3} \qquad \boldsymbol{k^4} = W^k \boldsymbol{a^4}$$

自注意力模型

$$\alpha'_{1,i} = exp(\alpha_{1,i}) / \sum_j exp(\alpha_{1,j})$$

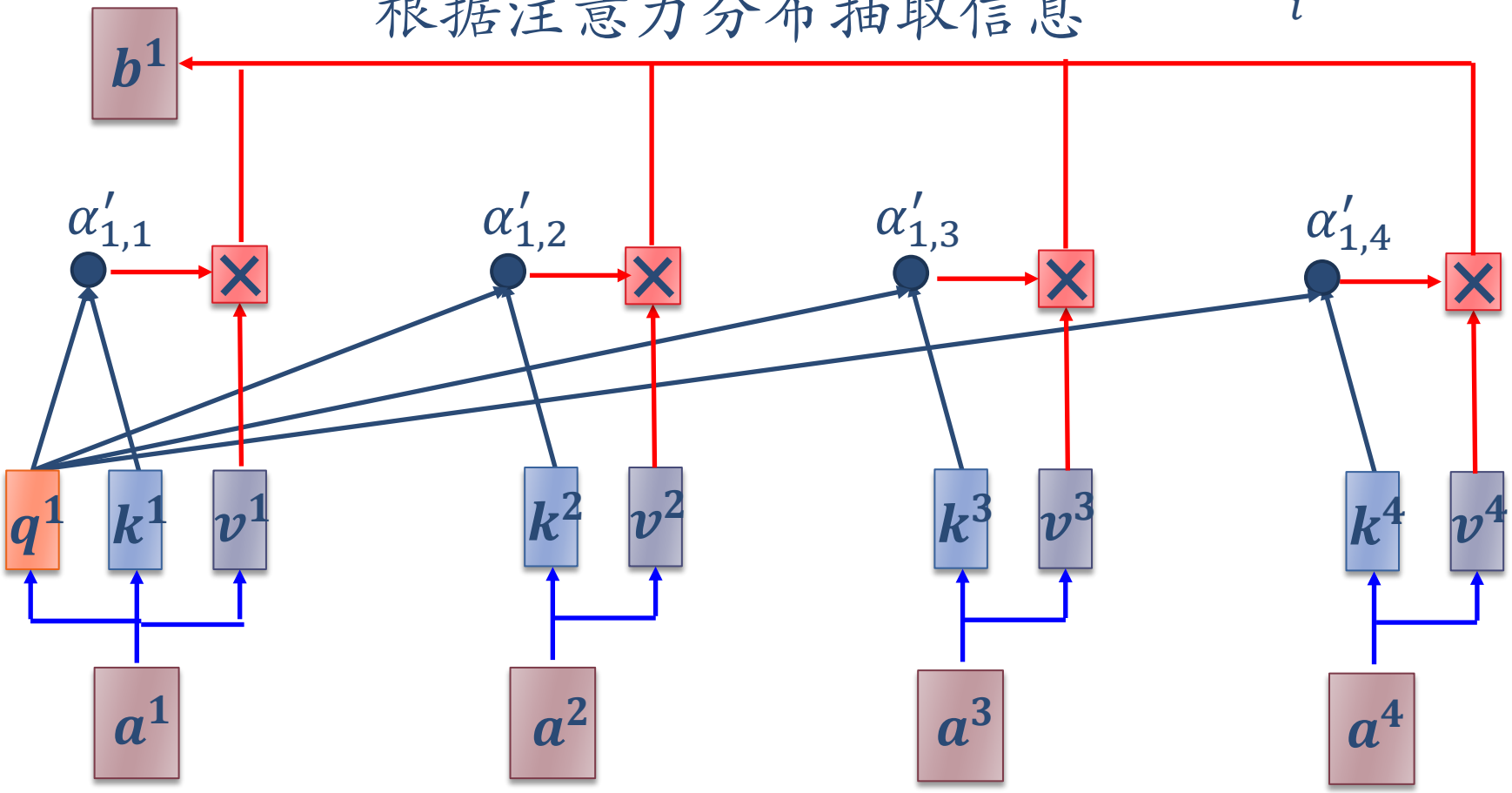


$$q^1 = W^q a^1$$
$$k^1 = W^k a^1$$
$$k^2 = W^k a^2$$
$$k^3 = W^k a^3$$
$$k^4 = W^k a^4$$

自注意力模型

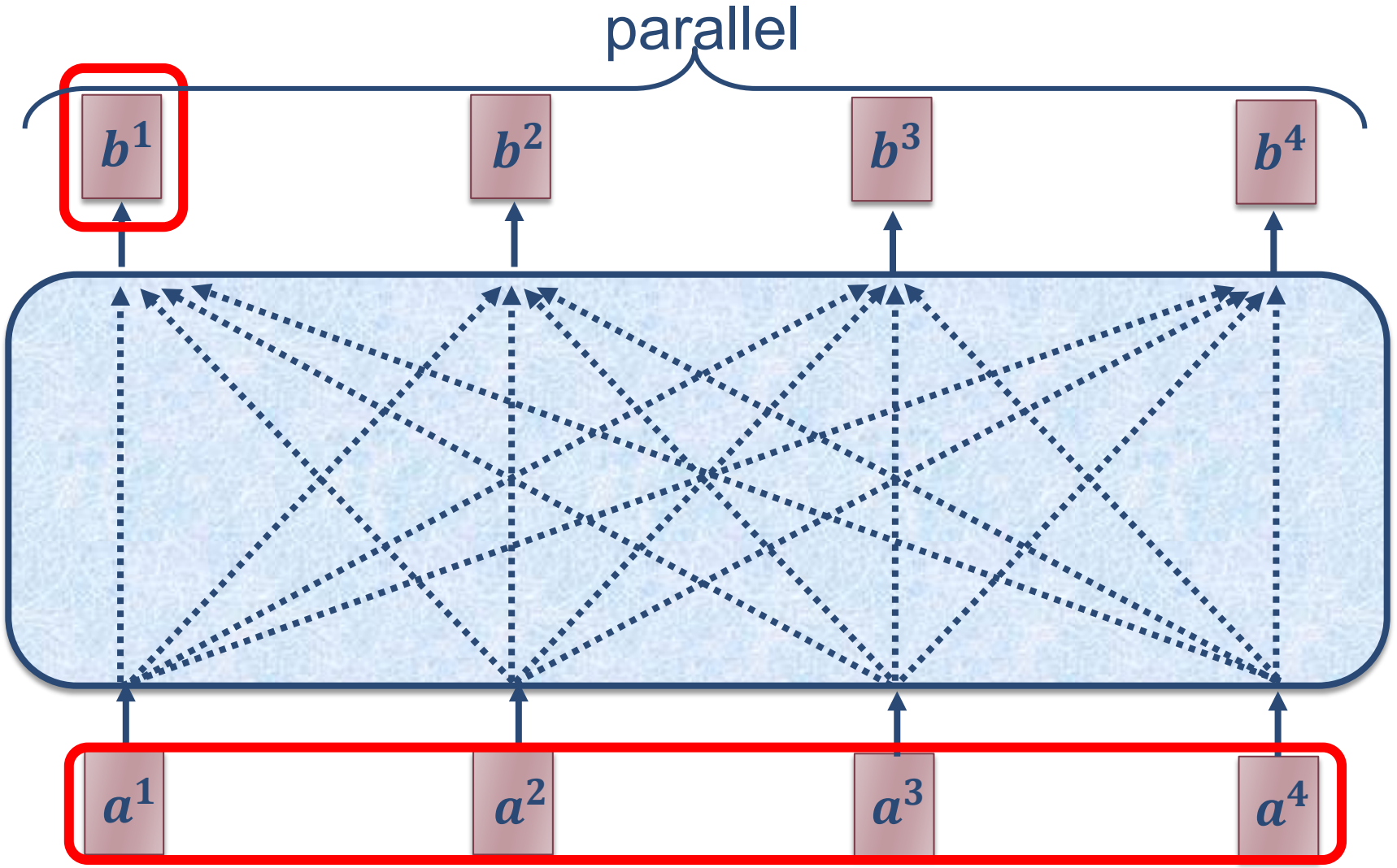
$$b^1 = \sum_i \alpha'_{1,i} v^i$$

根据注意力分布抽取信息



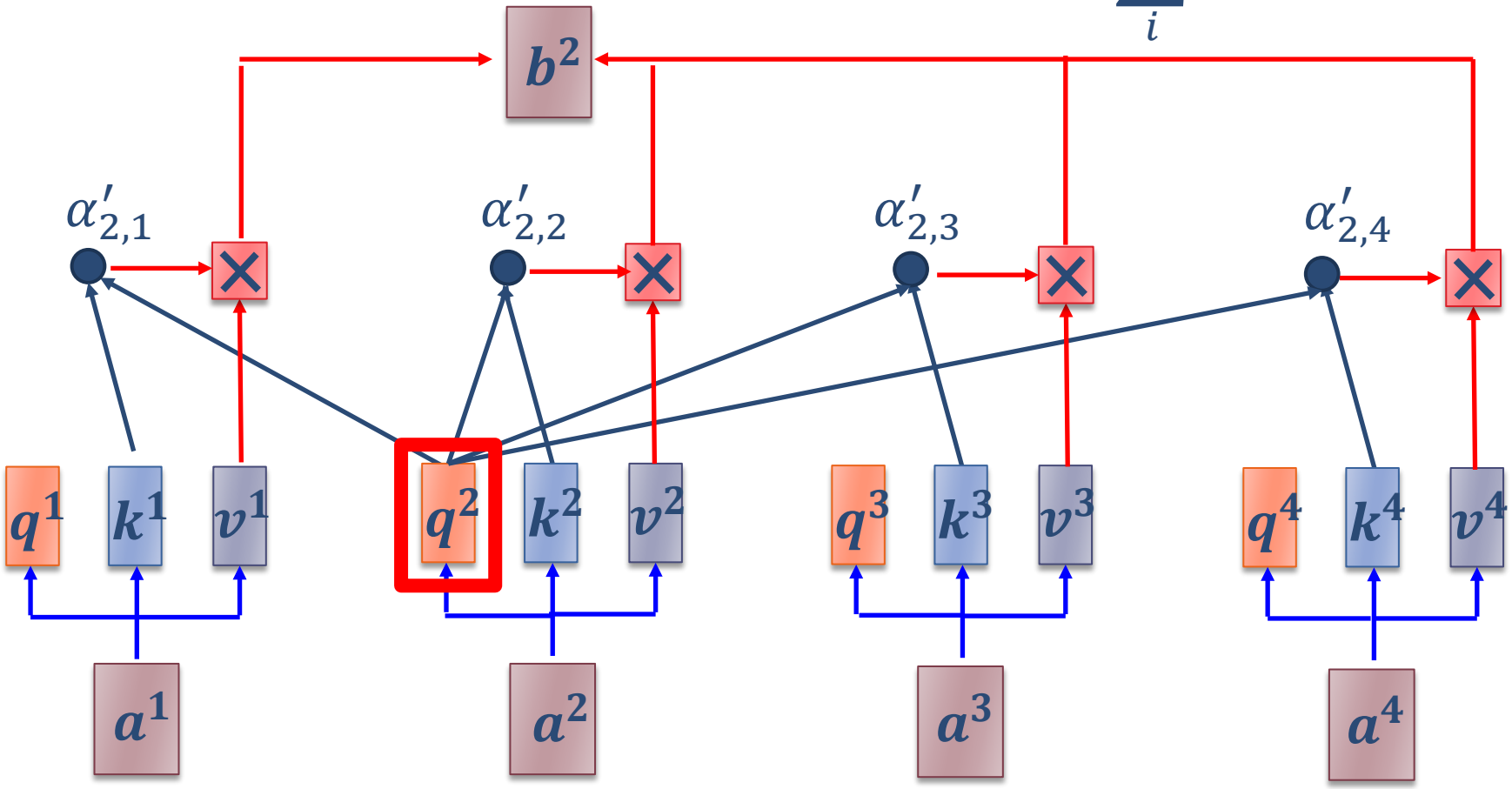
$$v^1 = W^v a^1 \qquad v^2 = W^v a^2 \qquad v^3 = W^v a^3 \qquad v^4 = W^v a^4$$

自注意力模型



自注意力模型

$$b^2 = \sum_i \alpha'_{2,i} v^i$$



自注意力模型

$$q^i = W^q a^i$$

$$q^1 q^2 q^3 q^4 = W^q a^1 a^2 a^3 a^4$$

Q I

$$k^i = W^k a^i$$

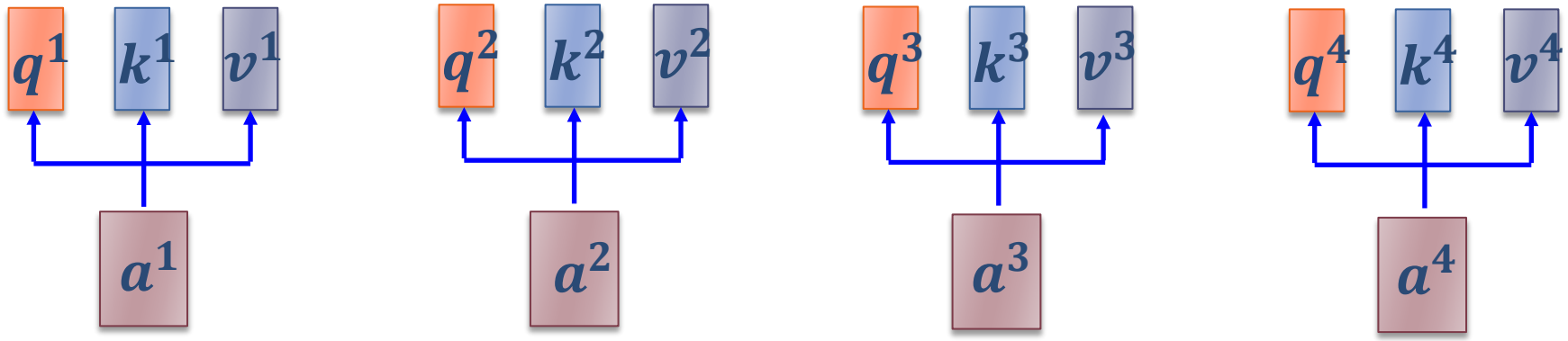
$$k^1 k^2 k^3 k^4 = W^k a^1 a^2 a^3 a^4$$

K I

$$v^i = W^v a^i$$

$$v^1 v^2 v^3 v^4 = W^v a^1 a^2 a^3 a^4$$

V I

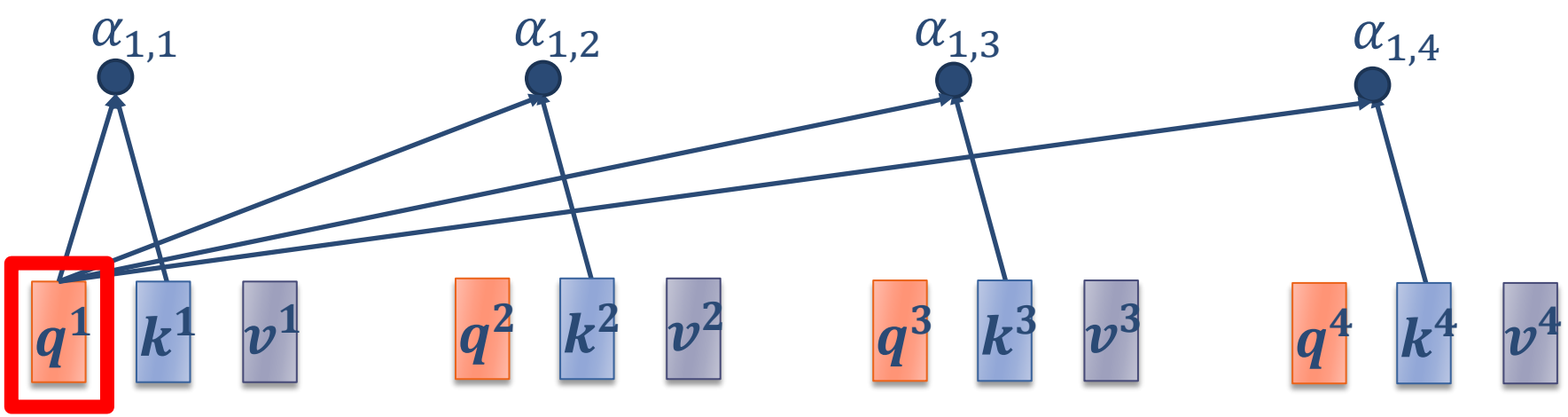


自注意力模型

$$\alpha_{1,1} = k^1 q^1 \quad \alpha_{1,2} = k^2 q^1$$

$$\alpha_{1,3} = k^3 q^1 \quad \alpha_{1,4} = k^4 q^1$$

$$\begin{matrix}\alpha_{1,1}\\ \alpha_{1,2}\\ \alpha_{1,3}\\ \alpha_{1,4}\end{matrix} = \begin{matrix}k^1\\ k^2\\ k^3\\ k^4\end{matrix} q^1$$

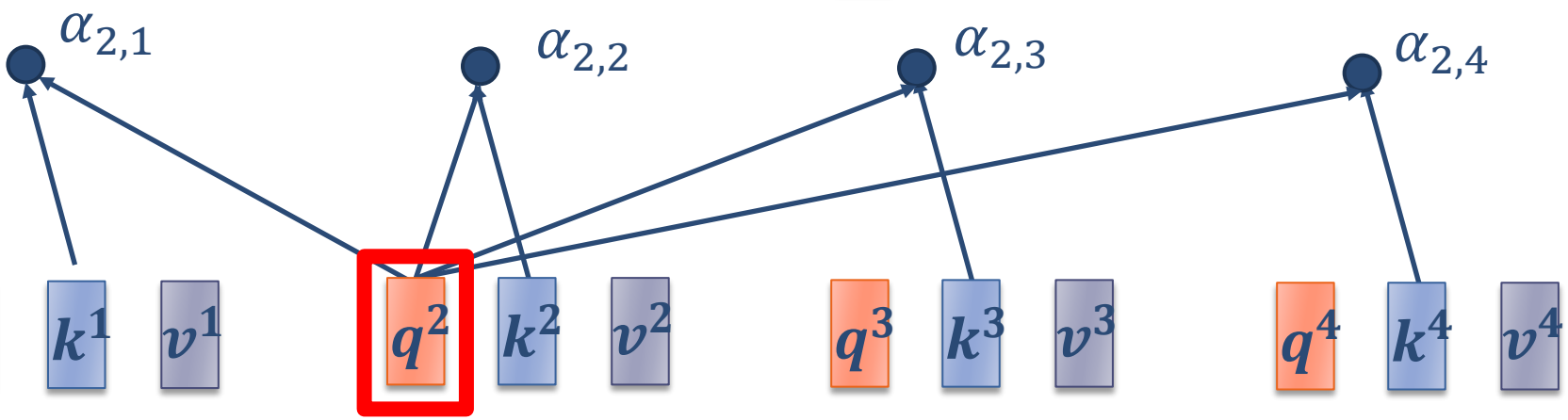


自注意力模型

$$\alpha_{1,1} = k^1 q^1 \quad \alpha_{1,2} = k^2 q^1$$

$$\alpha_{1,3} = k^3 q^1 \quad \alpha_{1,4} = k^4 q^1$$

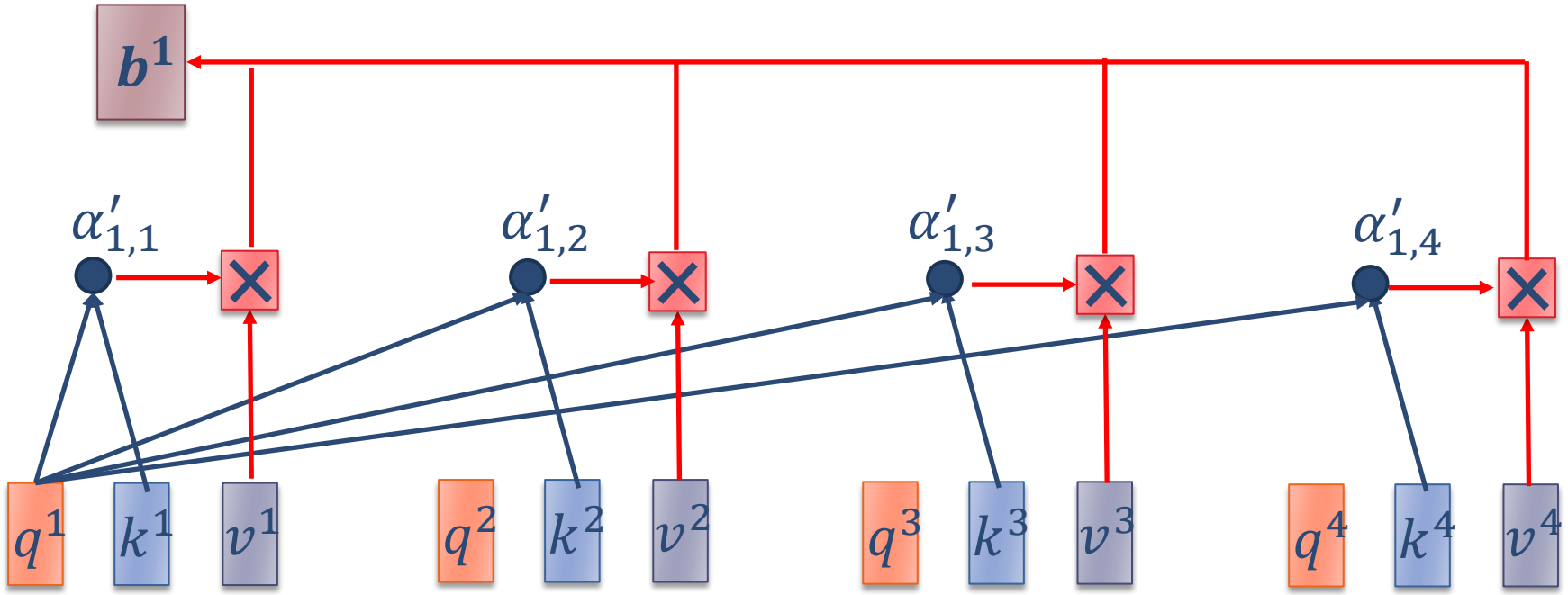
$$\begin{bmatrix} \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{bmatrix} = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} q^1$$



$$\begin{bmatrix} \alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1} \\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2} \\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3} \\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4} \end{bmatrix} = A' \quad \text{softmax} \quad \begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix} = A$$

$$= \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} = K^T \qquad \begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix} = Q$$

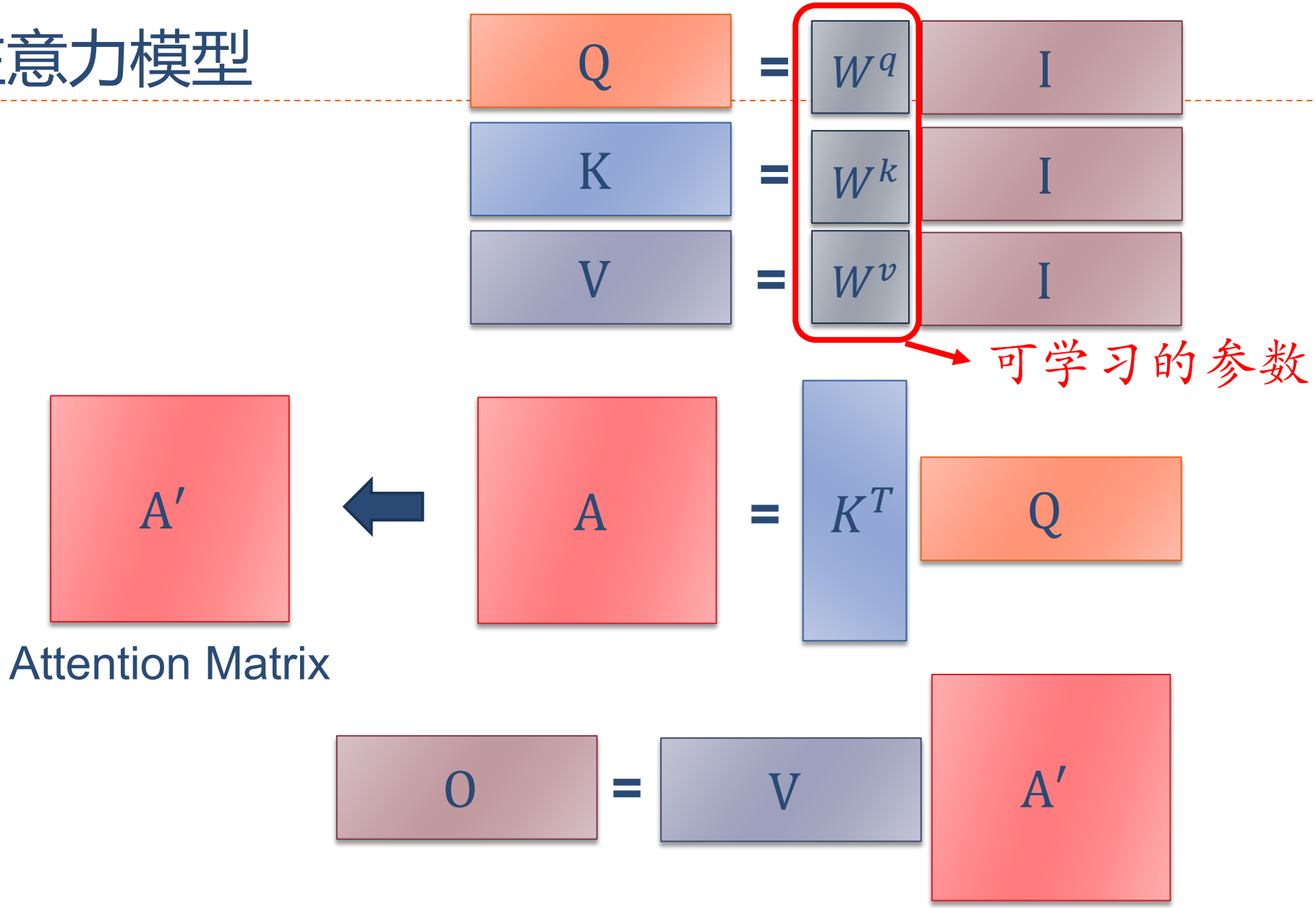
自注意力模型



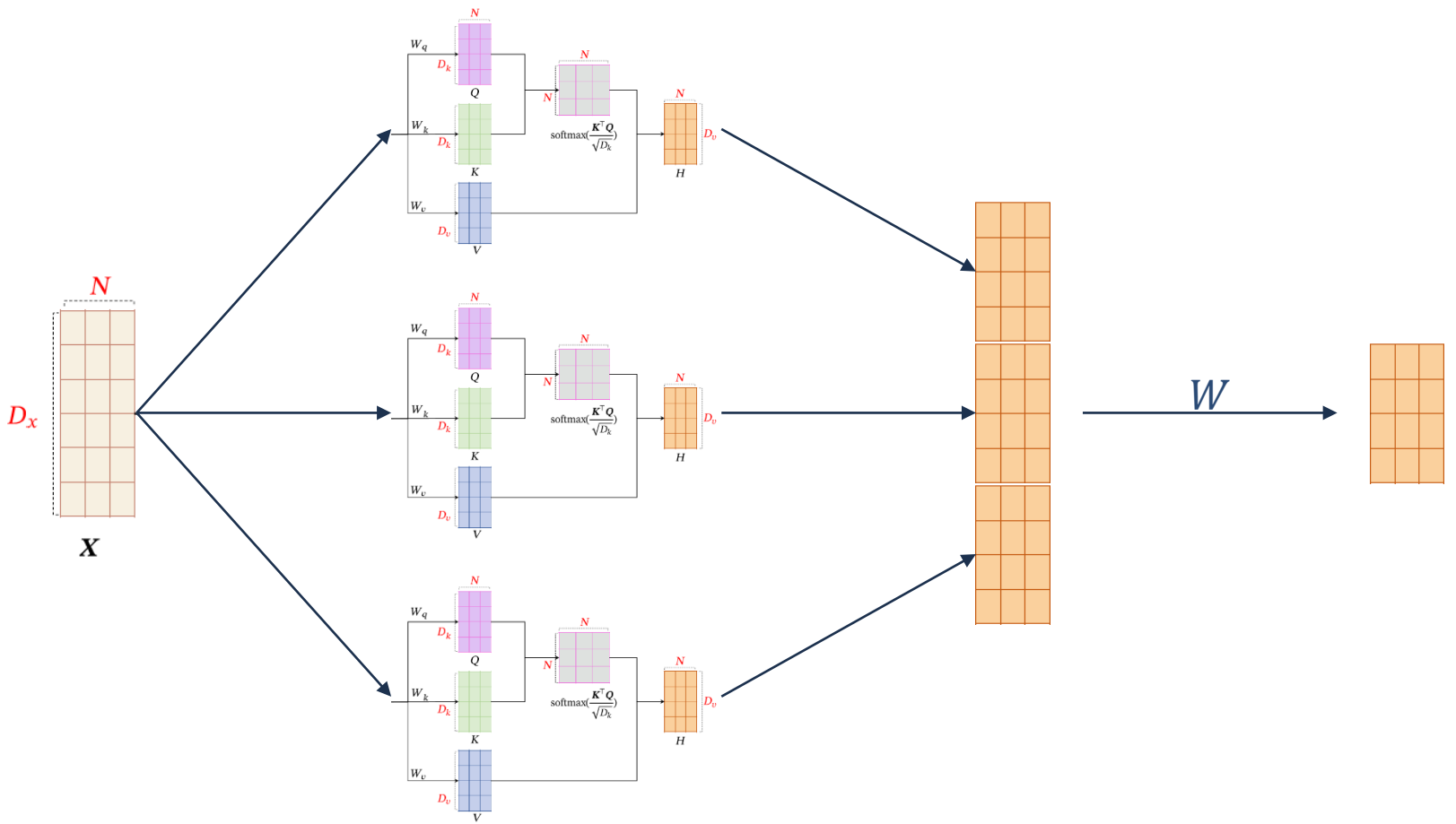
$$\underbrace{b^1\,b^2\,b^3\,b^4}_{O} = \underbrace{v^1\,v^2\,v^3\,v^4}_{V}$$

$$A' = \begin{bmatrix} \alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1} \\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2} \\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3} \\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4} \end{bmatrix}$$

自注意力模型



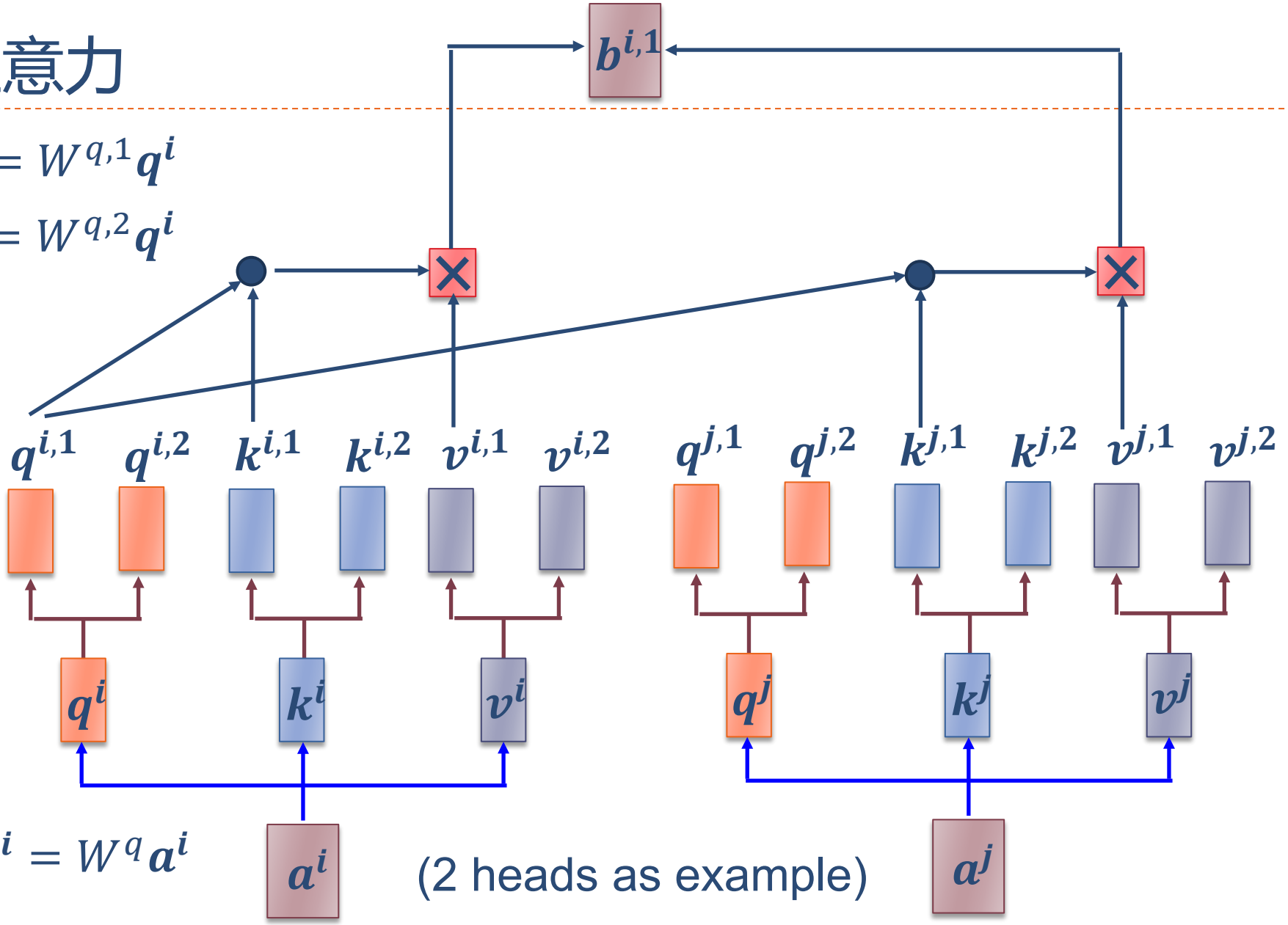
多头（multi-head）自注意力模型



多头自注意力

$$q^{i,1} = W^{q,1} q^i$$

$$q^{i,2} = W^{q,2} q^i$$



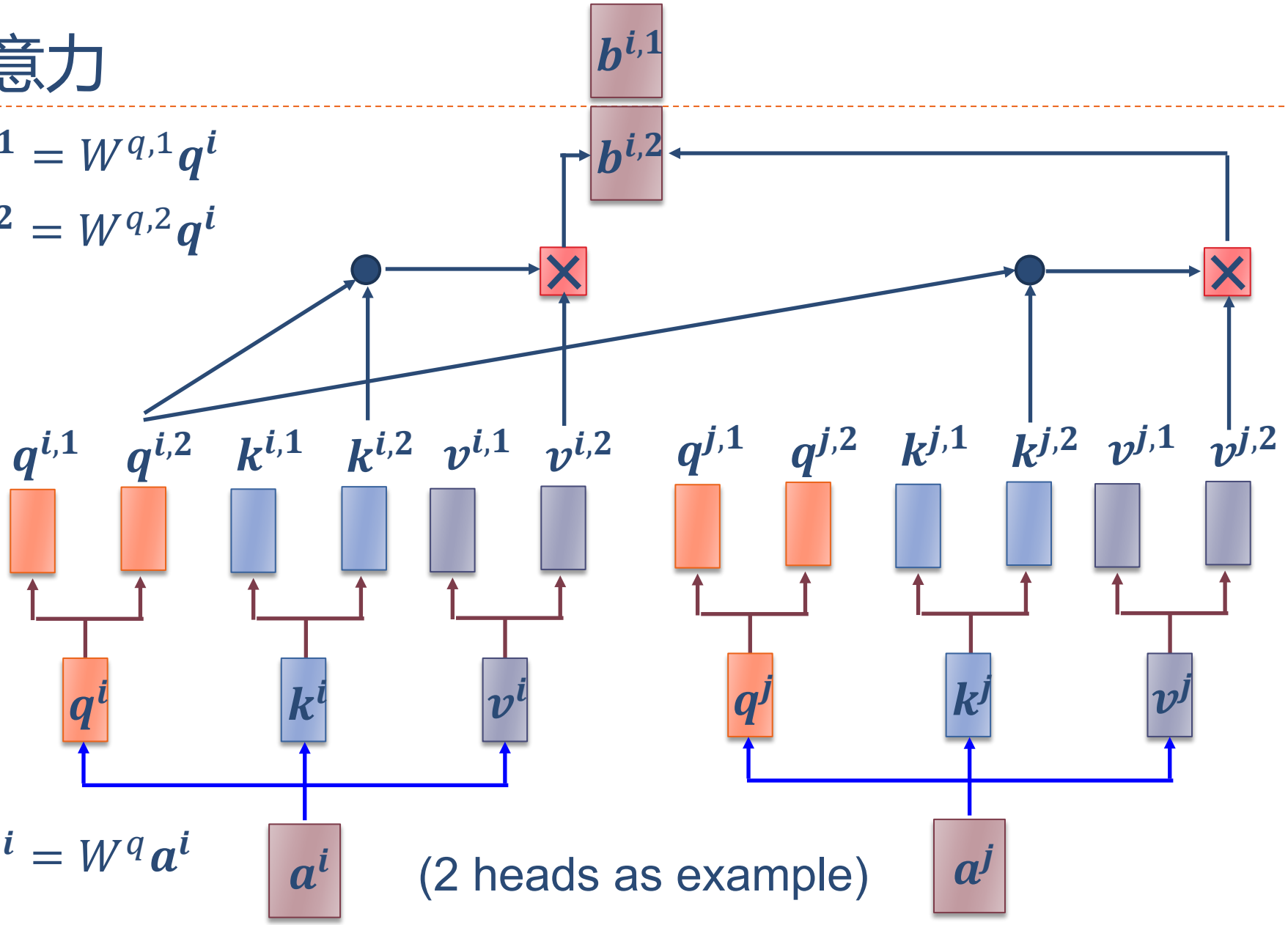
$$q^i = W^q a^i$$

(2 heads as example)

多头自注意力

$$q^{i,1} = W^{q,1} q^i$$

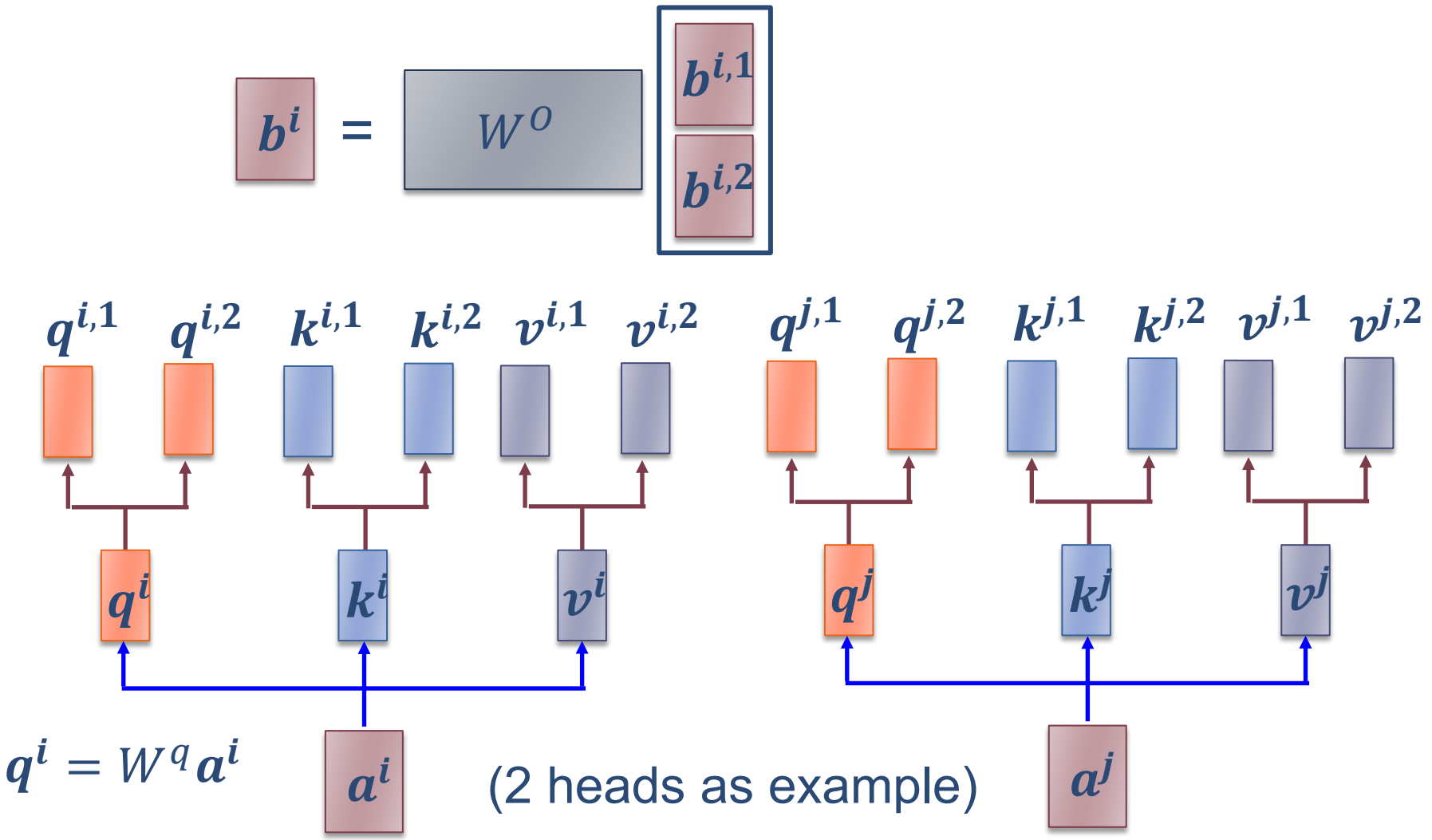
$$q^{i,2} = W^{q,2} q^i$$



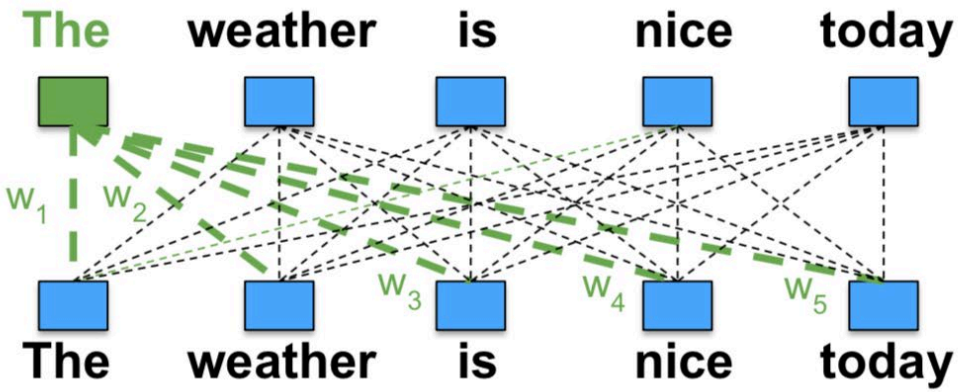
$$q^i = W^q a^i$$

(2 heads as example)

多头自注意力


$b^i = W^O \begin{bmatrix} b^{i,1} \\ b^{i,2} \end{bmatrix}$
$q^{i,1}$ $q^{i,2}$ $k^{i,1}$ $k^{i,2}$ $v^{i,1}$ $v^{i,2}$
$q^{j,1}$ $q^{j,2}$ $k^{j,1}$ $k^{j,2}$ $v^{j,1}$ $v^{j,2}$
q^i k^i v^i
q^j k^j v^j
$q^i = W^q a^i$
a^i
(2 heads as example)
a^j

自注意力示例



$$w_1, w_2, w_3, w_4, w_5 = \text{softmax}\left(\begin{bmatrix} 0.6 & 0.2 & 0.8 \end{bmatrix}_{\text{The}} \times \begin{bmatrix} 0.6 & 0.2 & 0.9 & 0.4 & 0.4 \\ 0.2 & 0.3 & 0.1 & 0.1 & 0.1 \\ 0.8 & 0.1 & 0.8 & 0.4 & 0.6 \end{bmatrix}_{\text{The weather is nice today}} \right)$$

$$\begin{bmatrix} 1.8 \\ 2.3 \\ 0.4 \end{bmatrix}_{\text{The}} = w_1 \times \begin{bmatrix} 0.6 \\ 0.2 \\ 0.8 \end{bmatrix}_{\text{The}} + w_2 \times \begin{bmatrix} 0.2 \\ 0.3 \\ 0.1 \end{bmatrix}_{\text{weather}} + w_3 \times \begin{bmatrix} 0.9 \\ 0.1 \\ 0.8 \end{bmatrix}_{\text{is}} + w_4 \times \begin{bmatrix} 0.4 \\ 0.1 \\ 0.4 \end{bmatrix}_{\text{nice}} + w_5 \times \begin{bmatrix} 0.4 \\ 0.1 \\ 0.6 \end{bmatrix}_{\text{today}}$$

图片来源：http://fuyw.top/NLP_02_QANet/

自注意力模型的形式化表示

- 输入序列为$X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in R^{D_x \times N}$
- 首先生成三个向量序列

$$\boldsymbol{Q} = \boldsymbol{W}_q \boldsymbol{X} \in \mathbb{R}^{D_k \times N},$$

$$\boldsymbol{K} = \boldsymbol{W}_k \boldsymbol{X} \in \mathbb{R}^{D_k \times N},$$

$$\boldsymbol{V} = \boldsymbol{W}_v \boldsymbol{X} \in \mathbb{R}^{D_v \times N},$$

- 计算$\boldsymbol{h}_n$

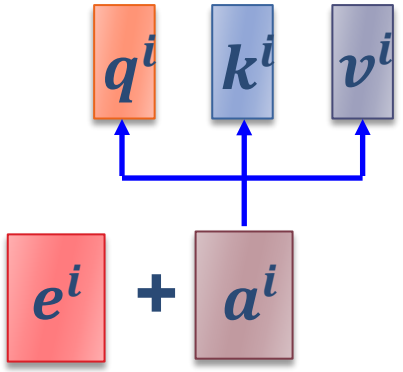
$$\boldsymbol{h}_n = \text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}_n\Big)$$

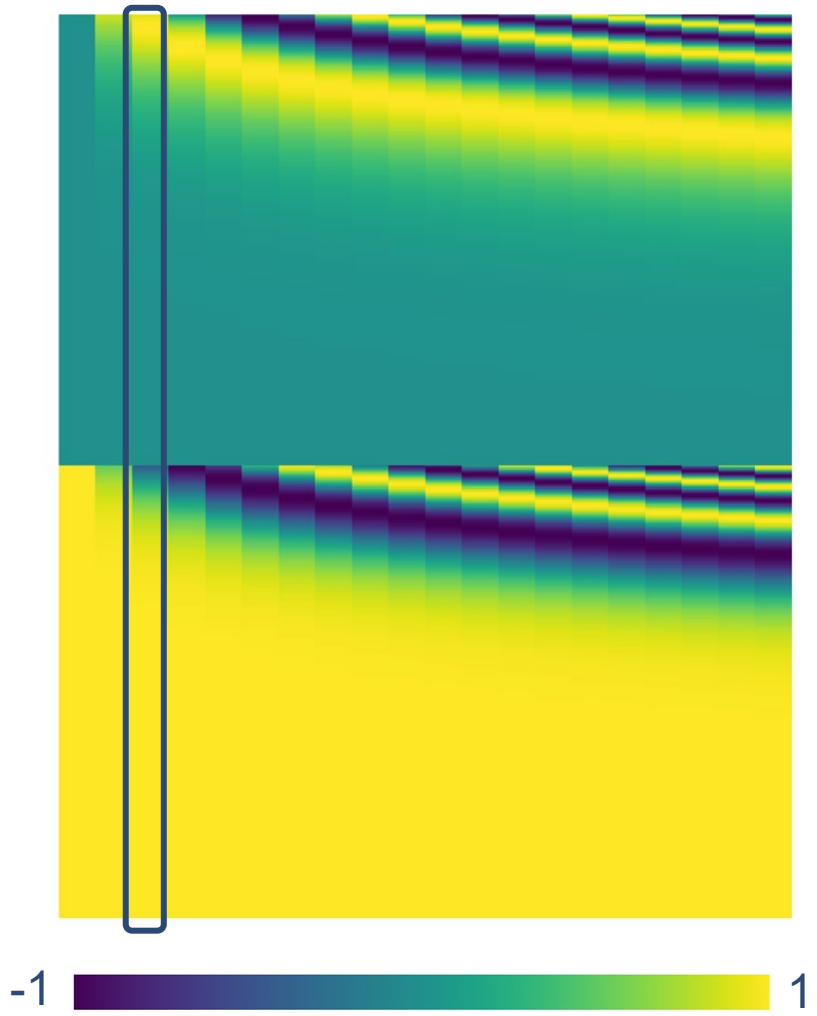
- 如果使用缩放点积来作为注意力打分函数，输出向量序列可以简写为

$$\boldsymbol{H} = \boldsymbol{V}\,\text{softmax}\Big(\frac{\boldsymbol{K}^{\top}\boldsymbol{Q}}{\sqrt{D_k}}\Big),$$

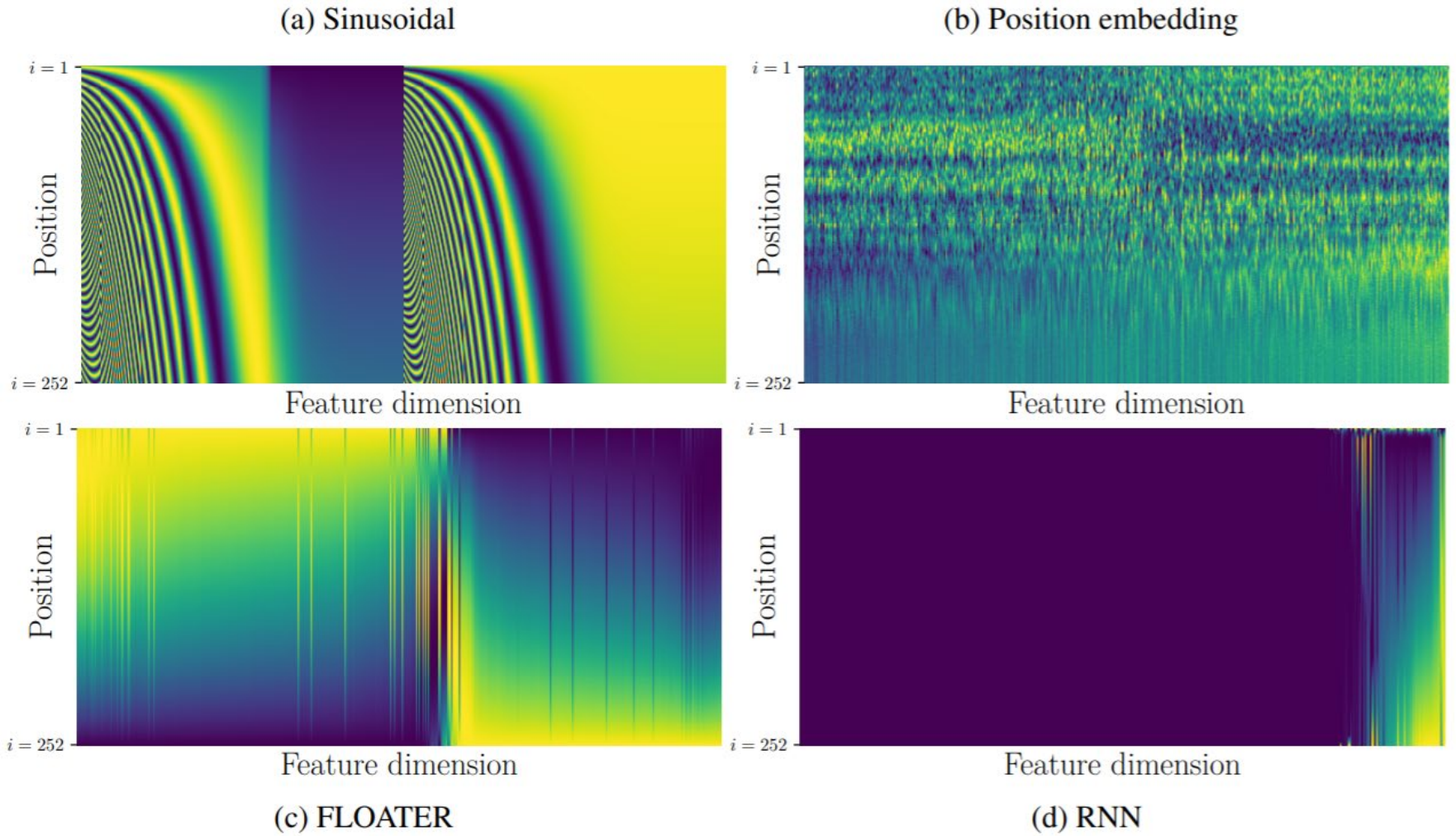
位置编码

- 自注意力机制没有位置信息
- 解决办法：为每个位置设置一个位置向量e^i
- 把位置向量跟原始向量相加
- 如何设计位置编码?
  - 人工设计
  - 从数据中学习





位置编码


(a) Sinusoidal
(b) Position embedding
$i = 1$
$i = 252$
Position
Feature dimension
(c) FLOATER
(d) RNN

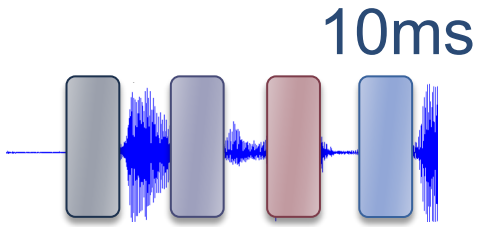
自注意力的应用

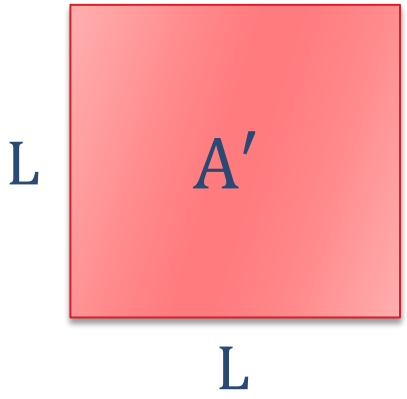
Transformer

BERT

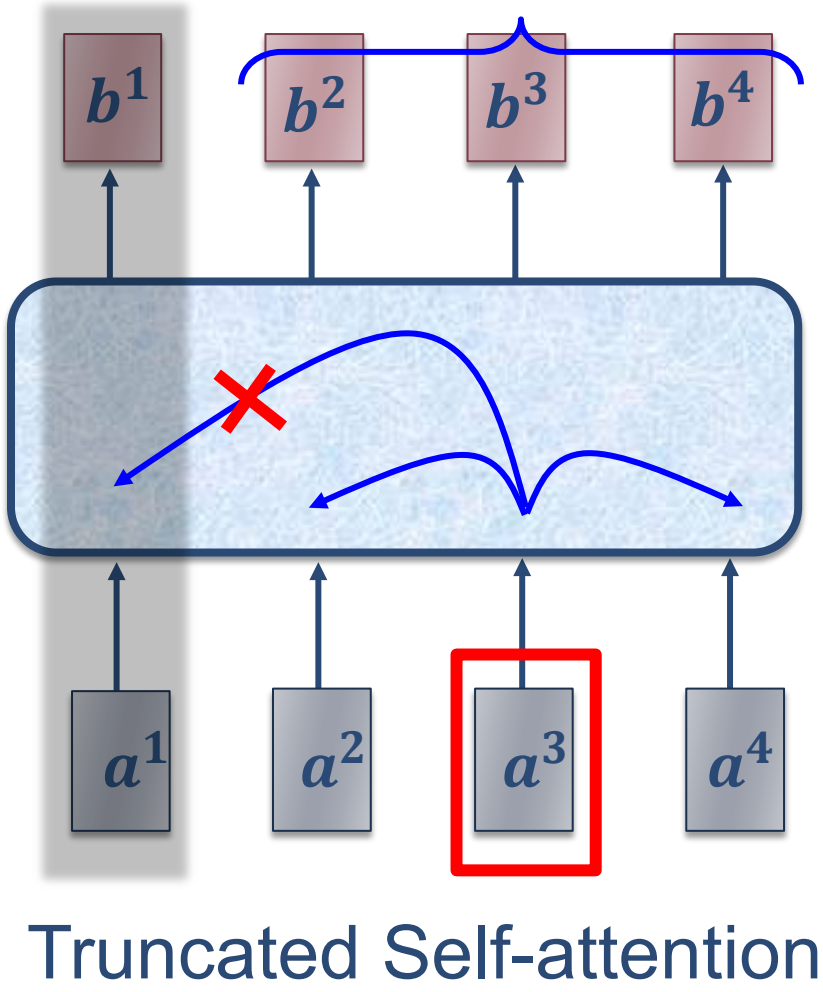
自注意力的应用

语音序列非常长





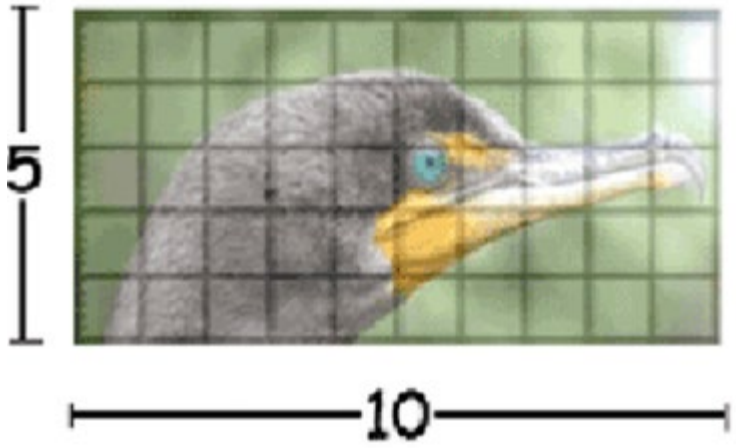
取一定范围的注意力

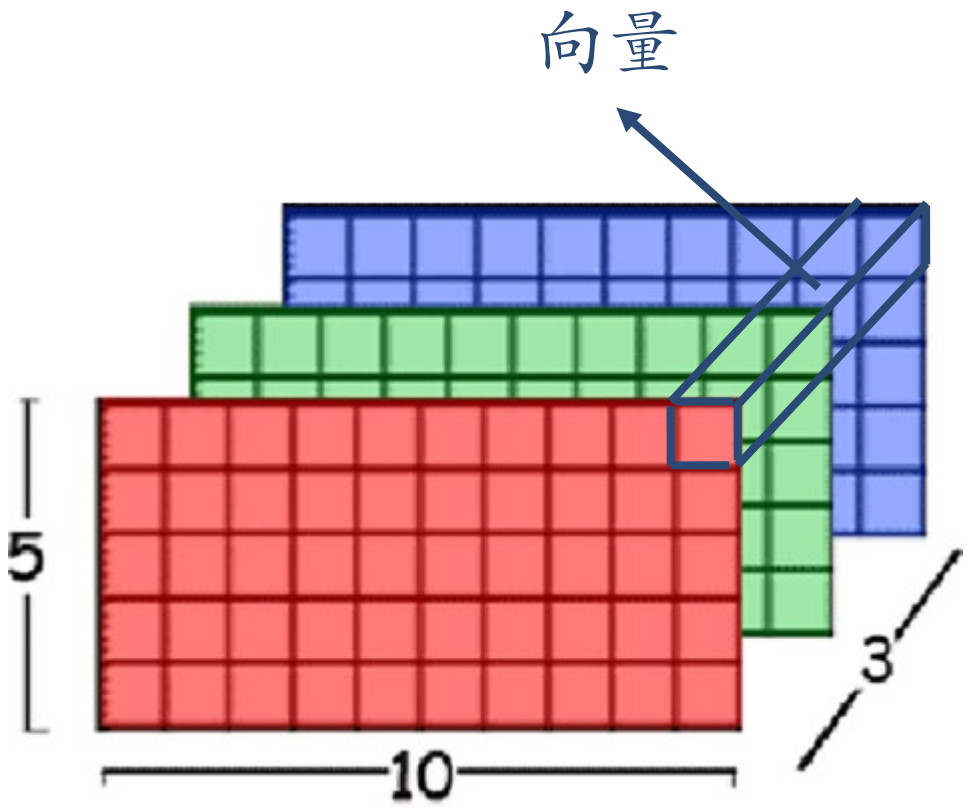


Truncated Self-attention

自注意力的应用

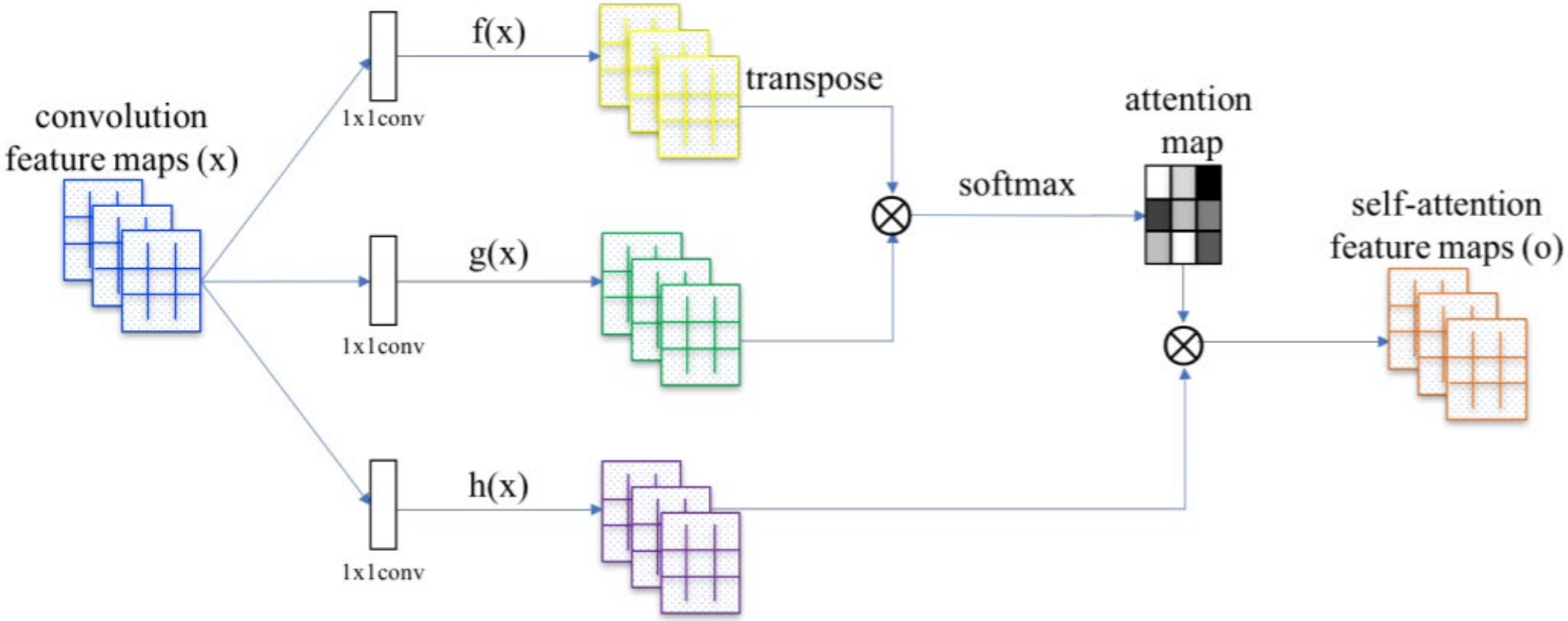
把图片看成向量的集合





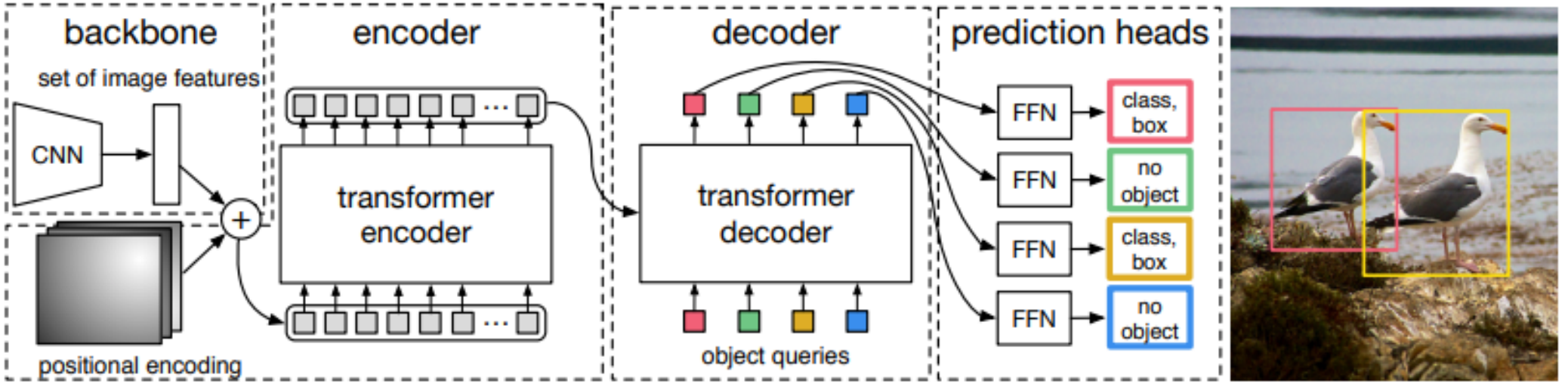
自注意力的应用

Self-Attention GAN

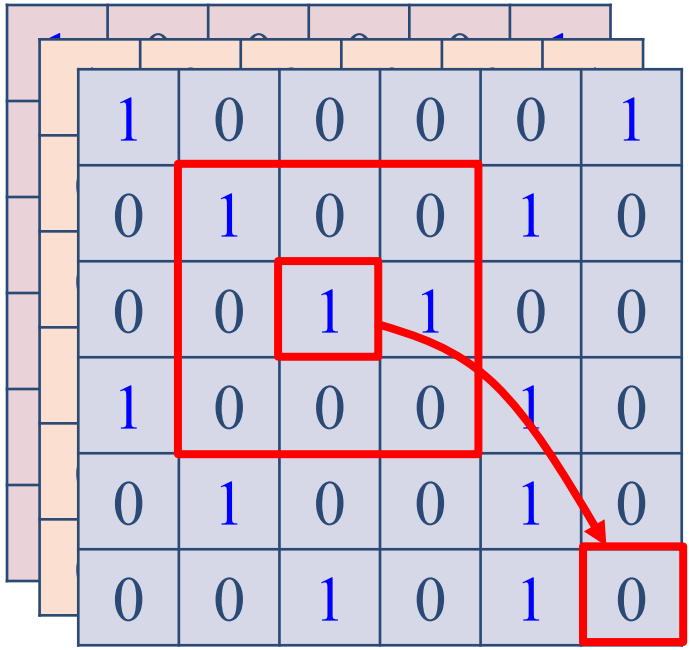


自注意力的应用

DEtection Transformer (DETR)



比较：自注意力与卷积神经网络



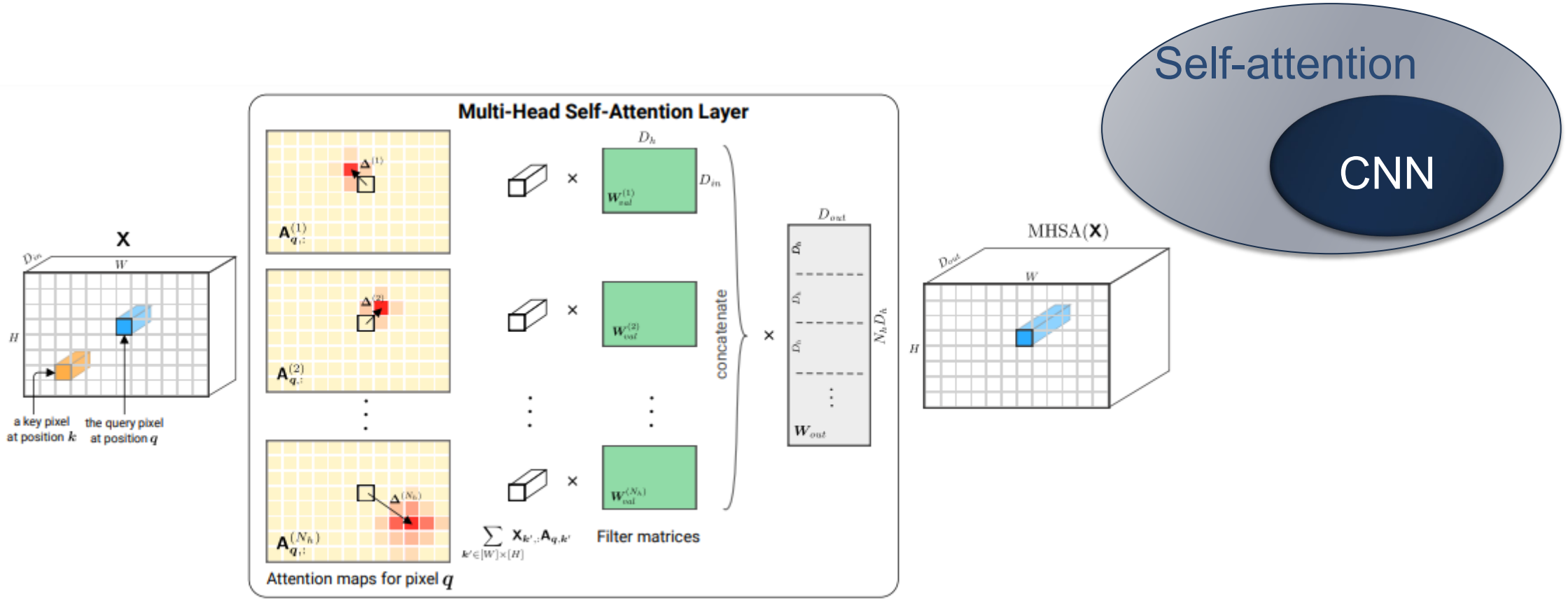
CNN: 可以看成一种限制在局部感受野的自注意力

- CNN是一种简化版的自注意力

自注意力: 有可学习感受野的CNN

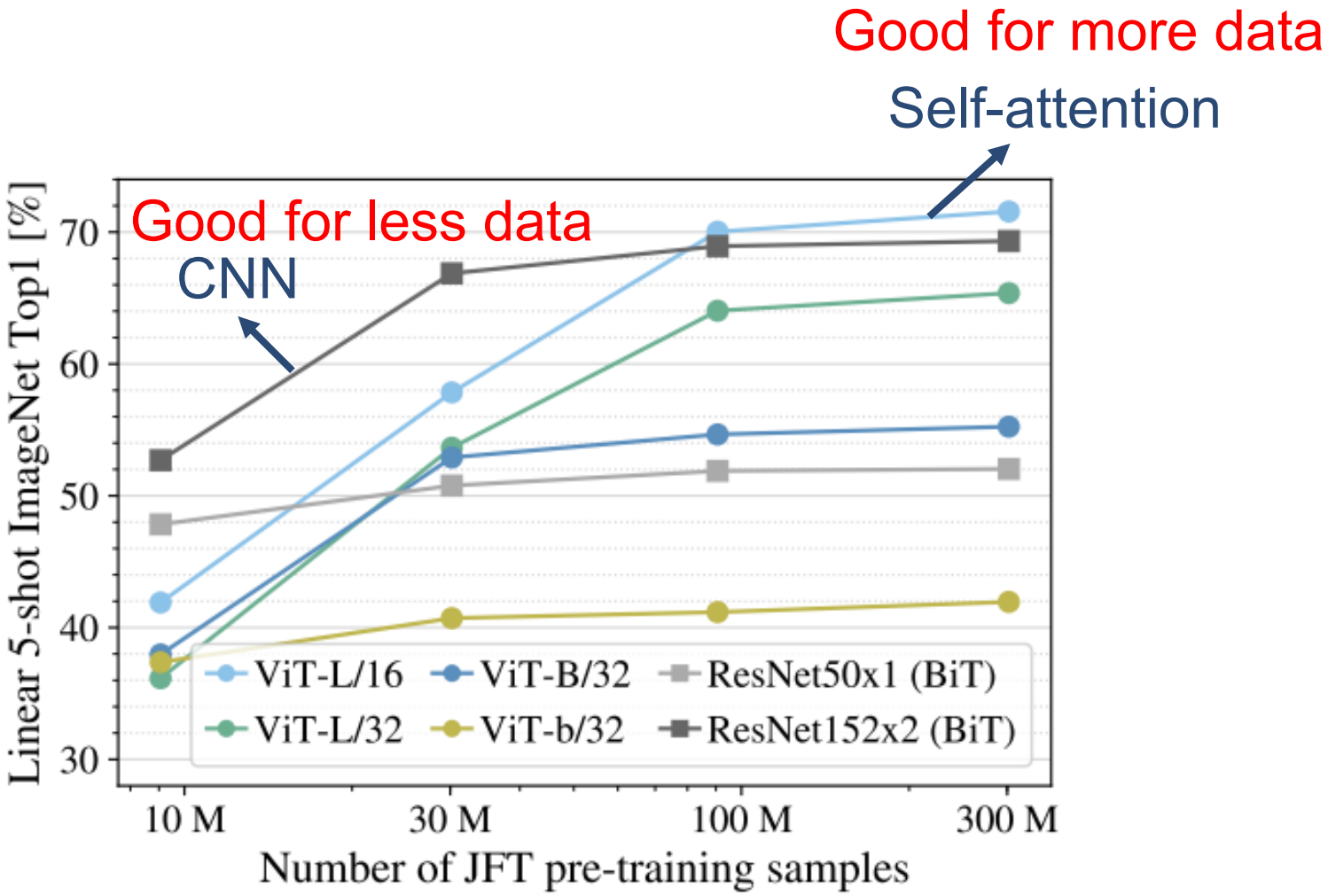
- 自注意力是一种复杂版本的CNN

比较：自注意力与卷积神经网络

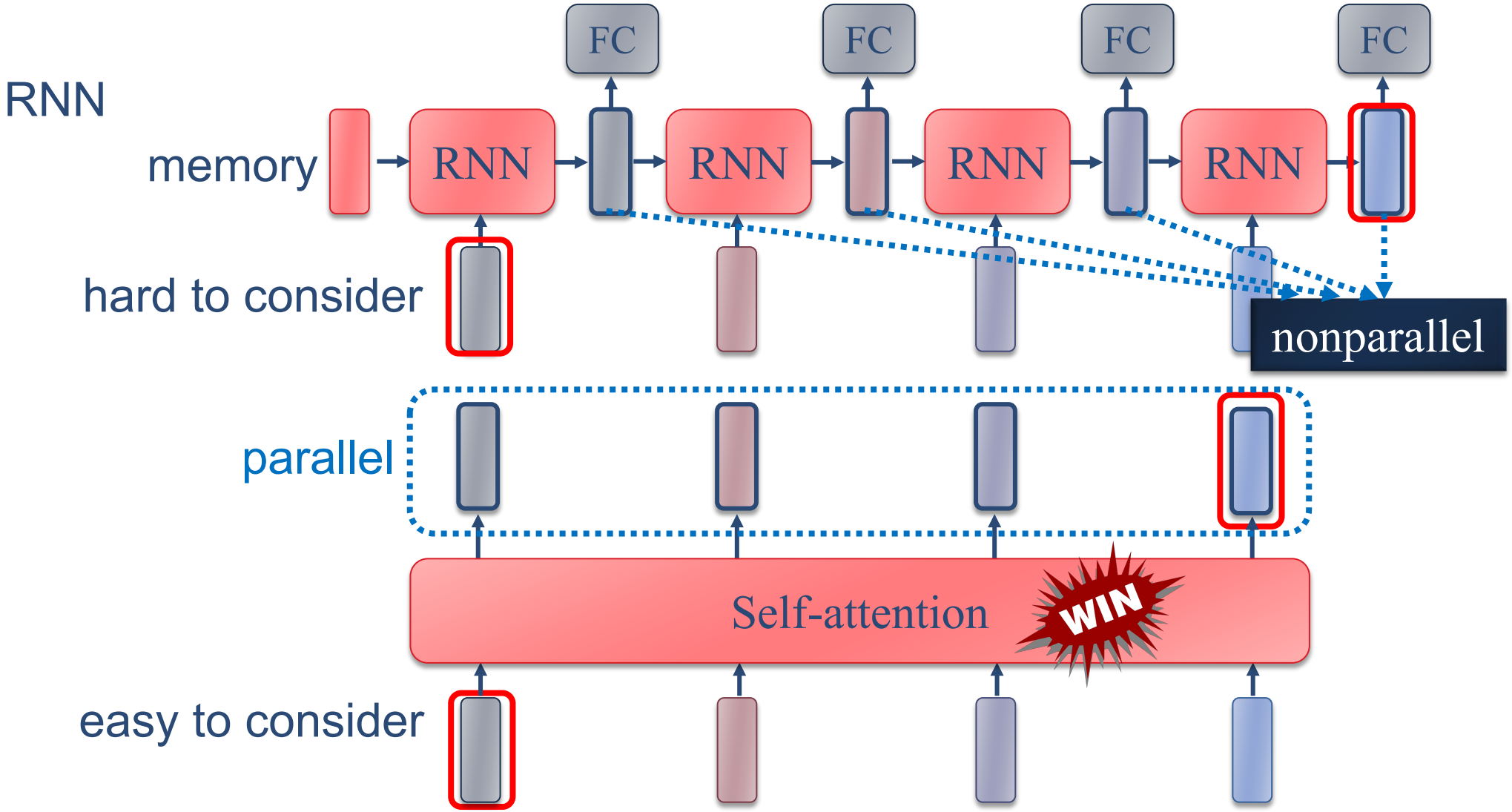


On the Relationship between Self-Attention and Convolutional Layers

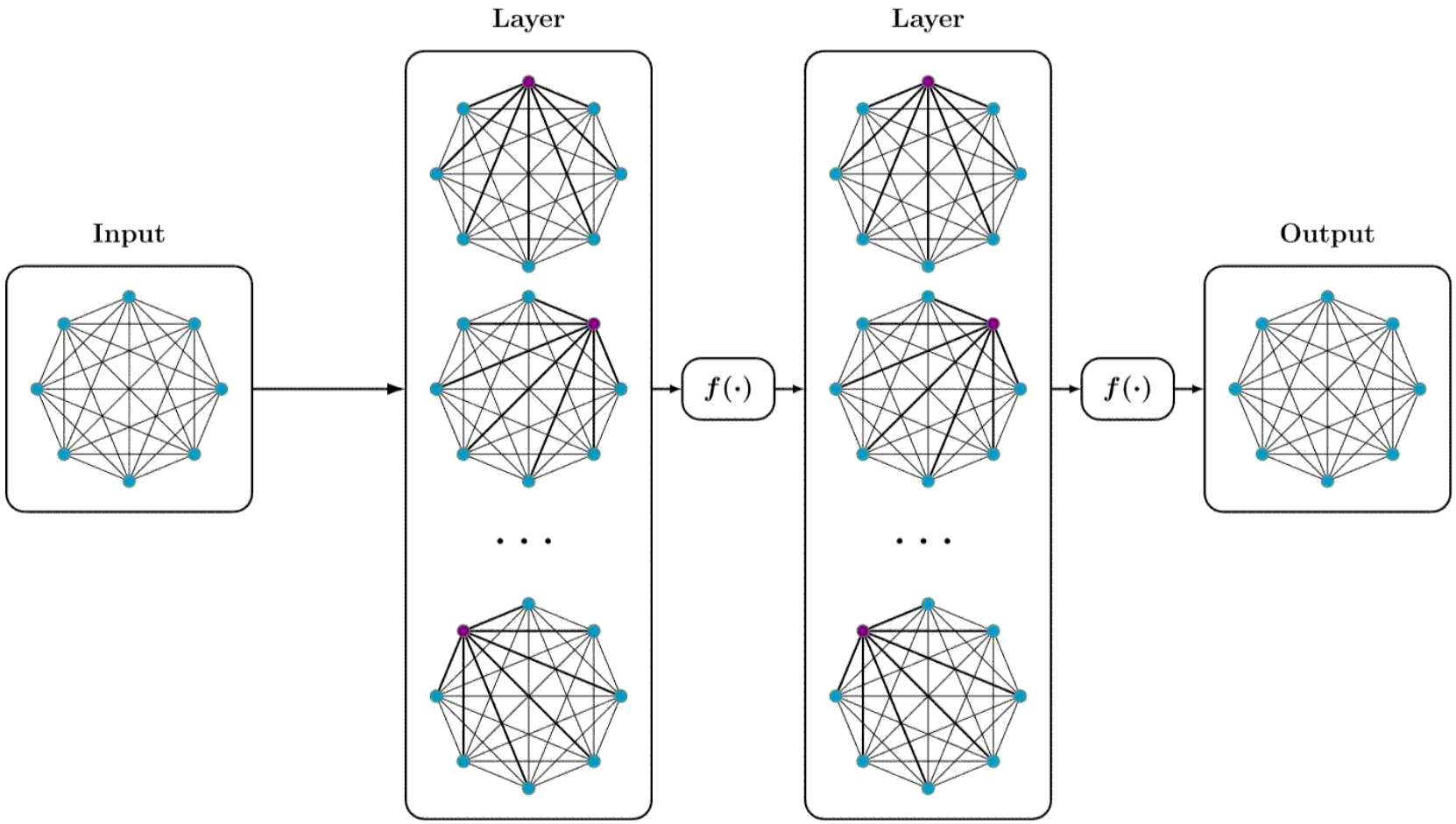
比较：自注意力与卷积神经网络



比较：自注意力与循环神经网络



比较：自注意力与循环神经网络



复杂度分析

模型	每层复杂度	序列操作数	最大路径长度
CNN	$O(kLd^2)$	$O(1)$	$O(\log_k(L))$
RNN	$O(Ld^2)$	$O(L)$	$O(L)$
自注意力	$O(L^2d)$	$O(1)$	$O(1)$

k卷积核大小　L序列长度　d维度

Transformer

注意力机制

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

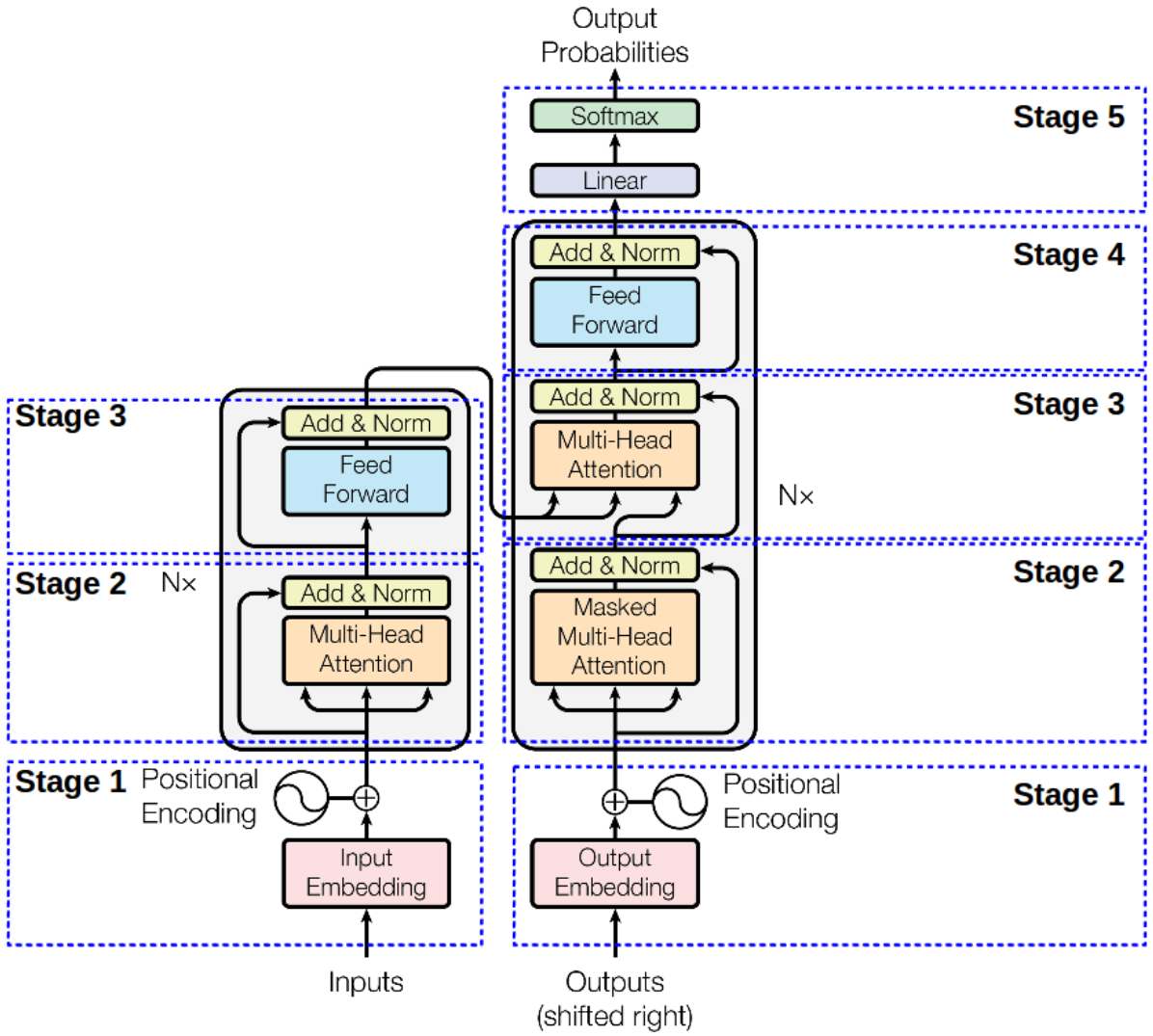
Abstract

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

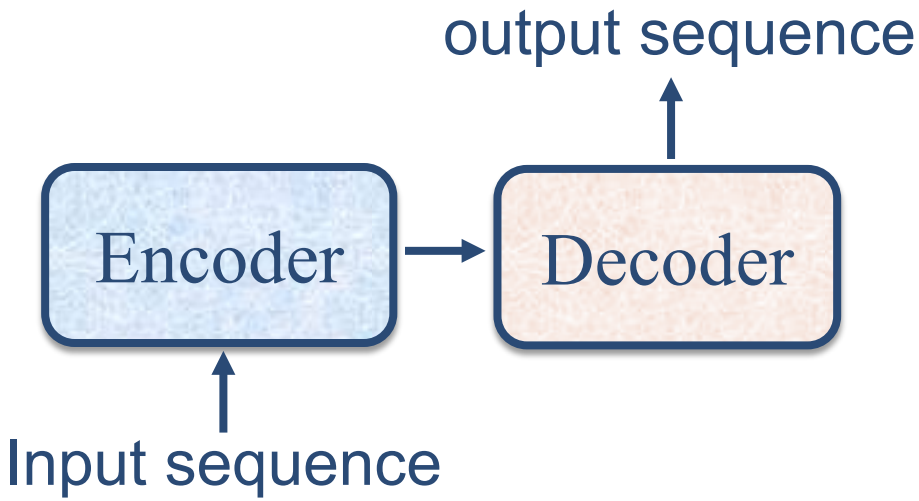

Transformer

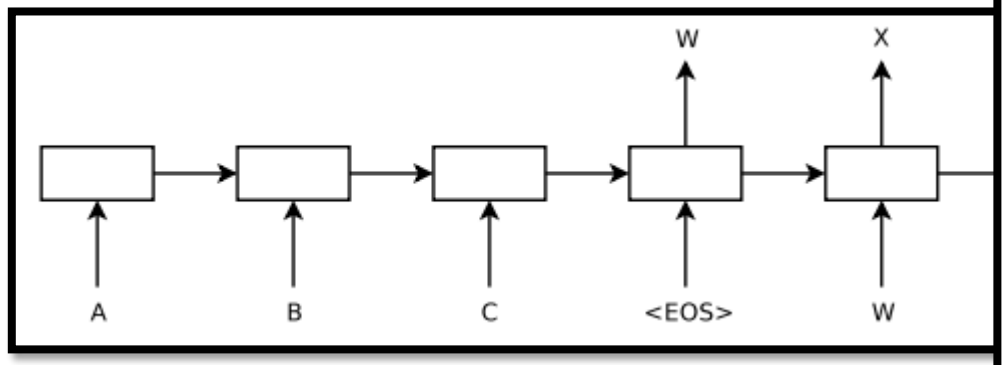
Transformer

https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/#.W90QB5Mzabg

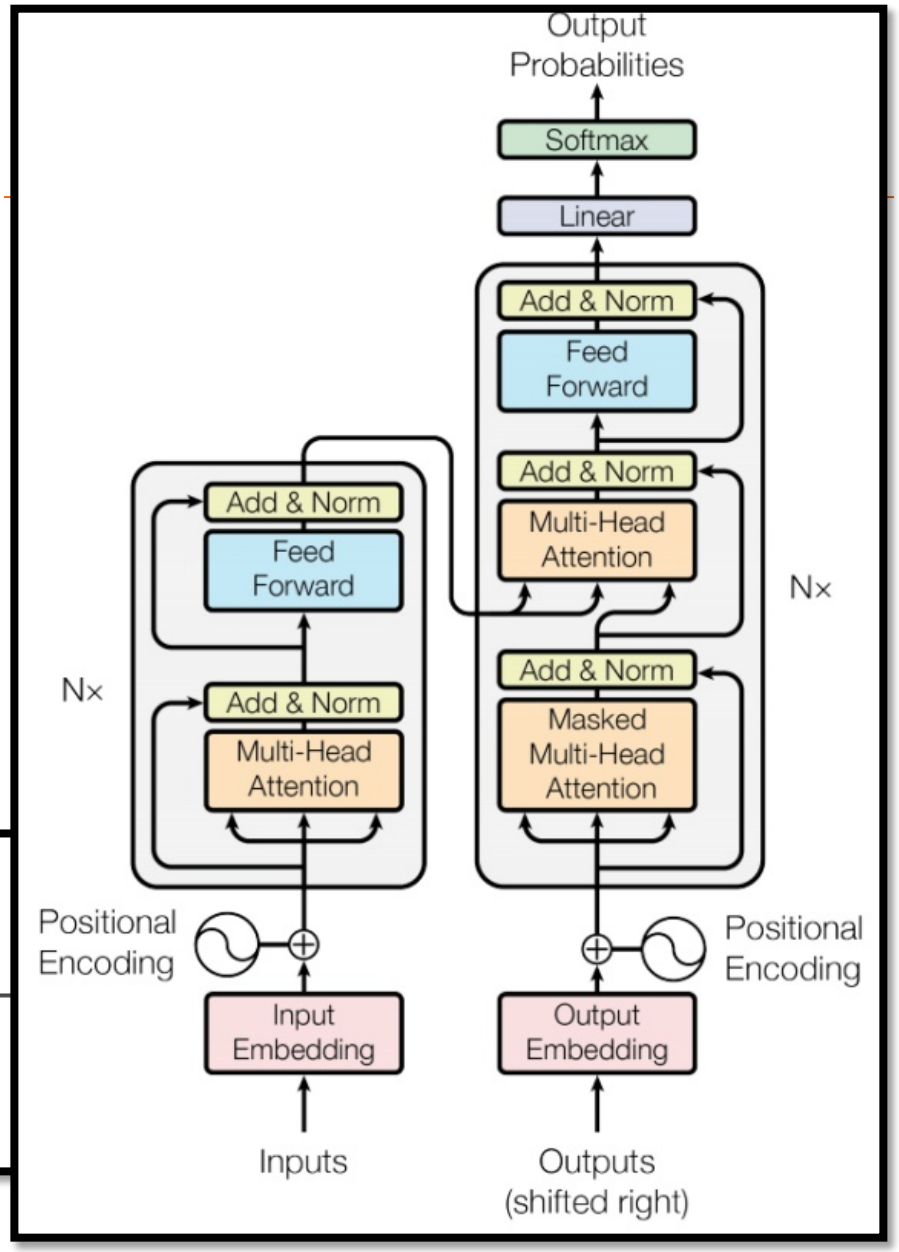


Seq2Seq



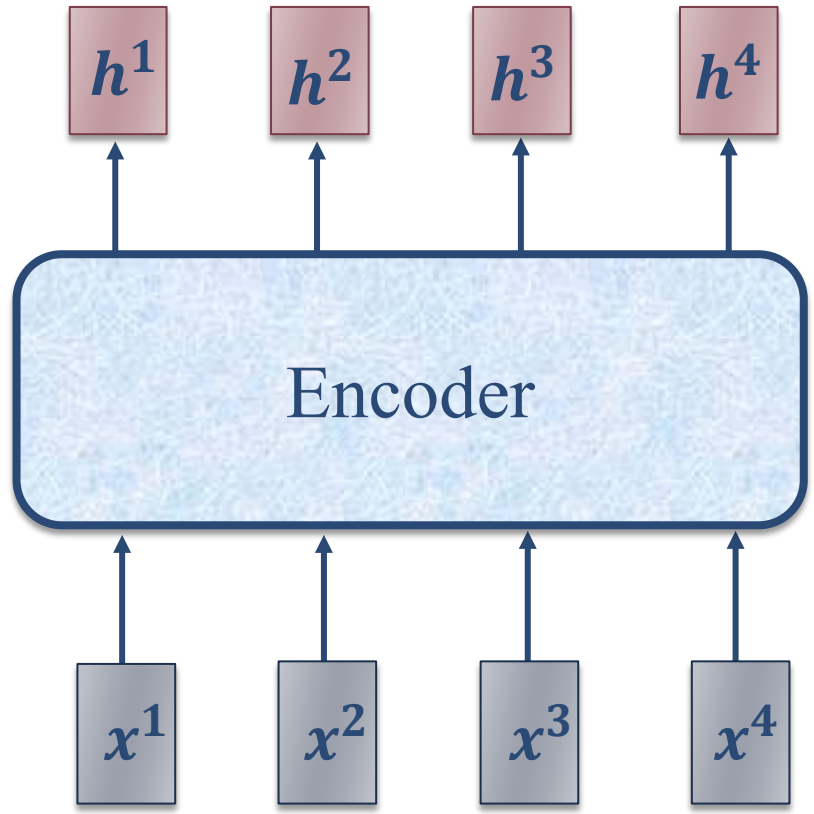


Sequence to Sequence Learning with Neural Networks

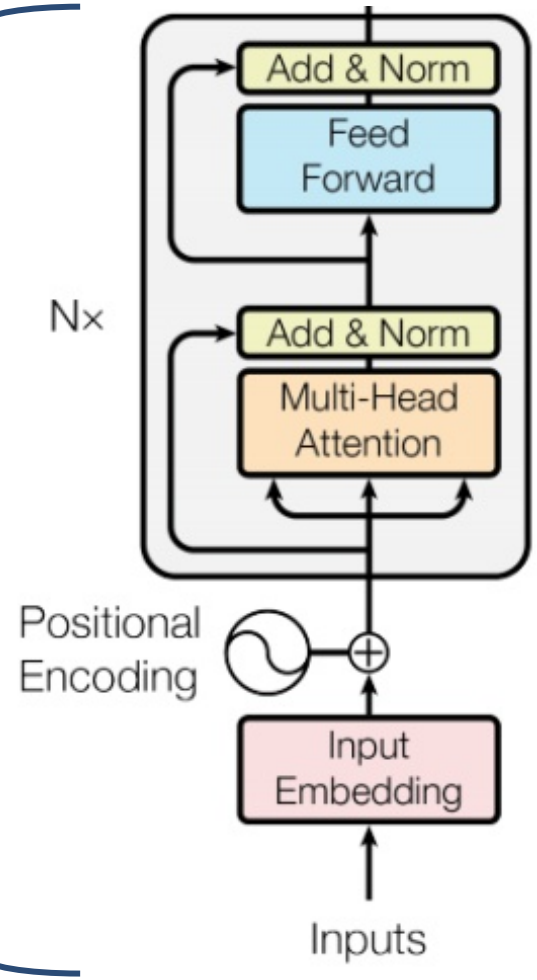


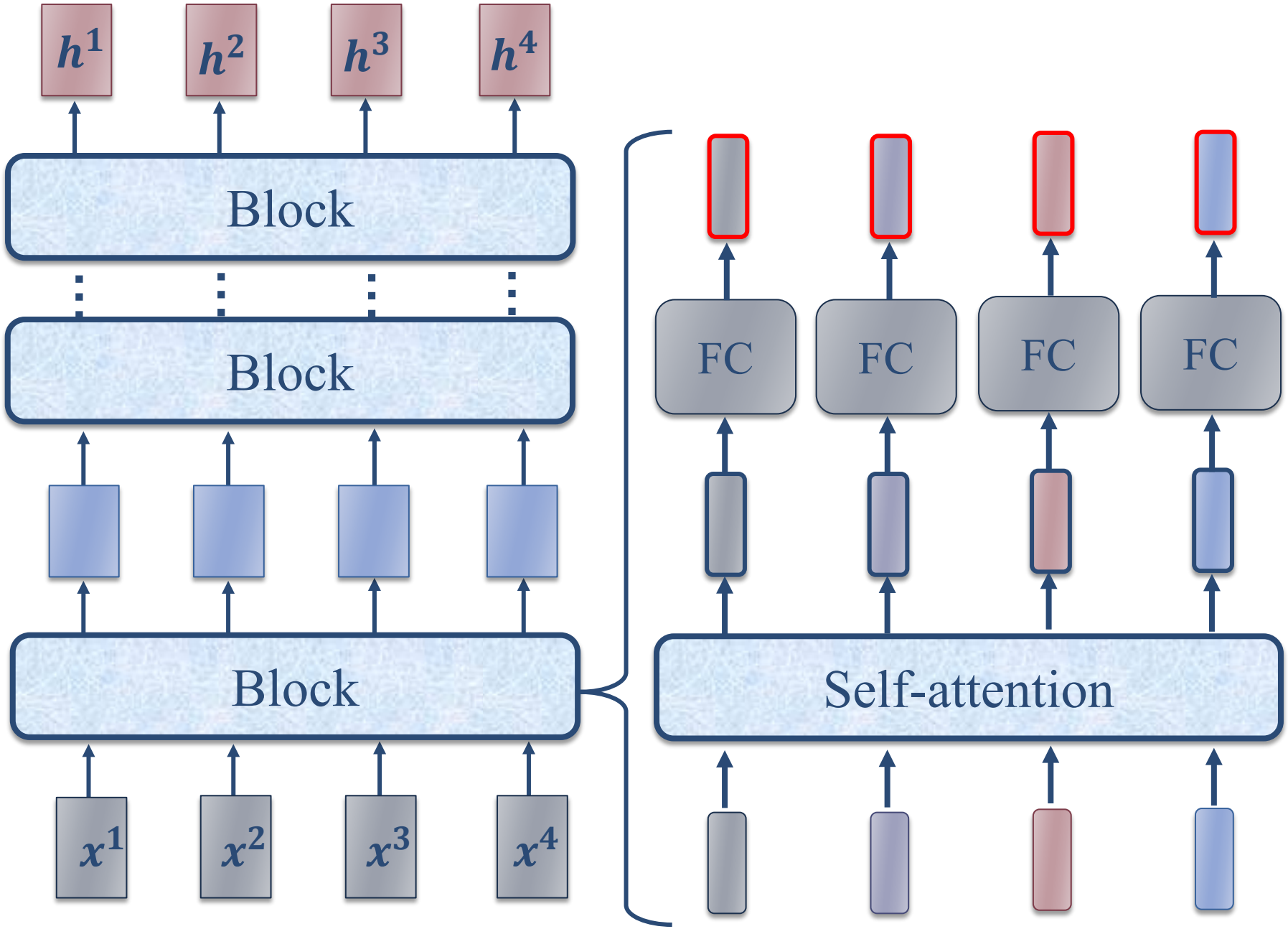
Transformer

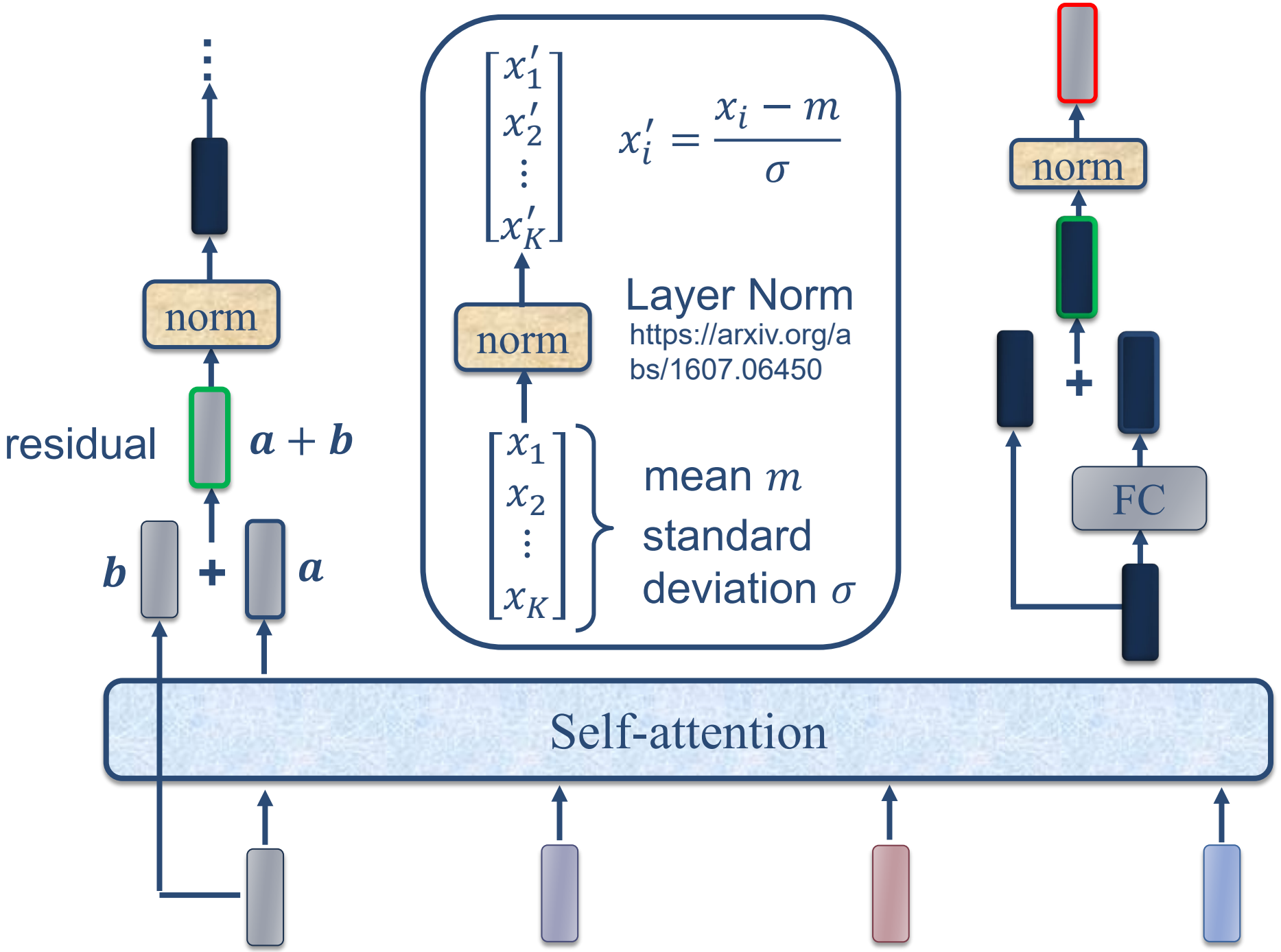
Transformer's Encoder



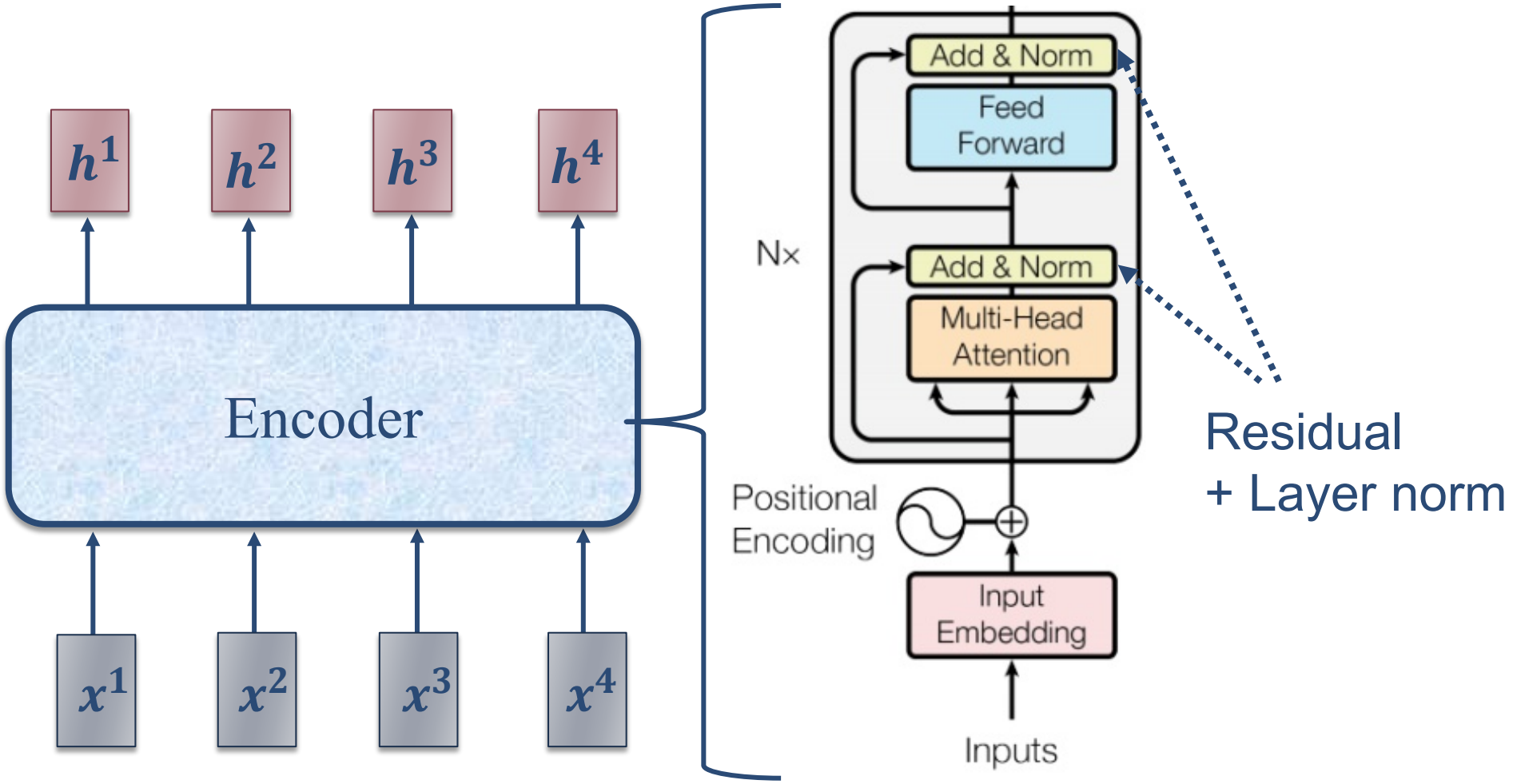
Transformer's Encoder



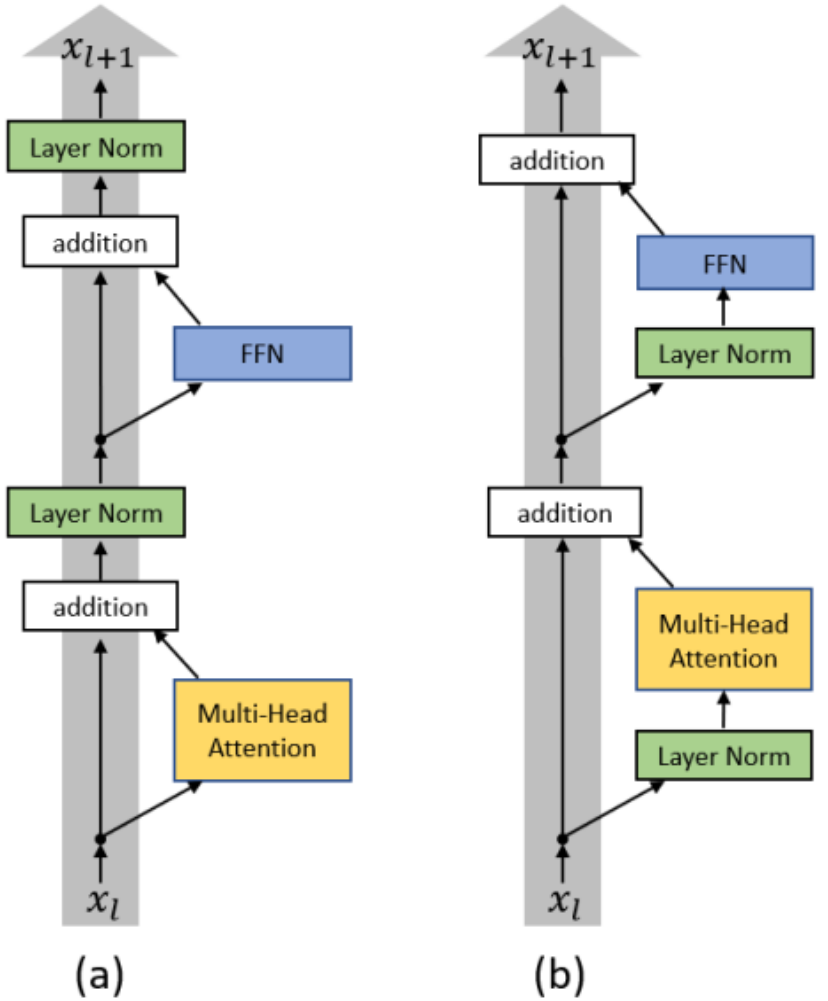

h^1
h^2
h^3
h^4
Block
Block
Block
x^1
x^2
x^3
x^4
FC
FC
FC
FC
Self-attention


norm
residual
$a+b$
b
+
a
Self-attention
$x_i' = \frac{x_i - m}{\sigma}$
x_1' x_2' x_K'
norm
Layer Norm
https://arxiv.org/abs/1607.06450
x_1 x_2 x_K
mean m
standard deviation σ
norm
+
FC

Transformer's Encoder



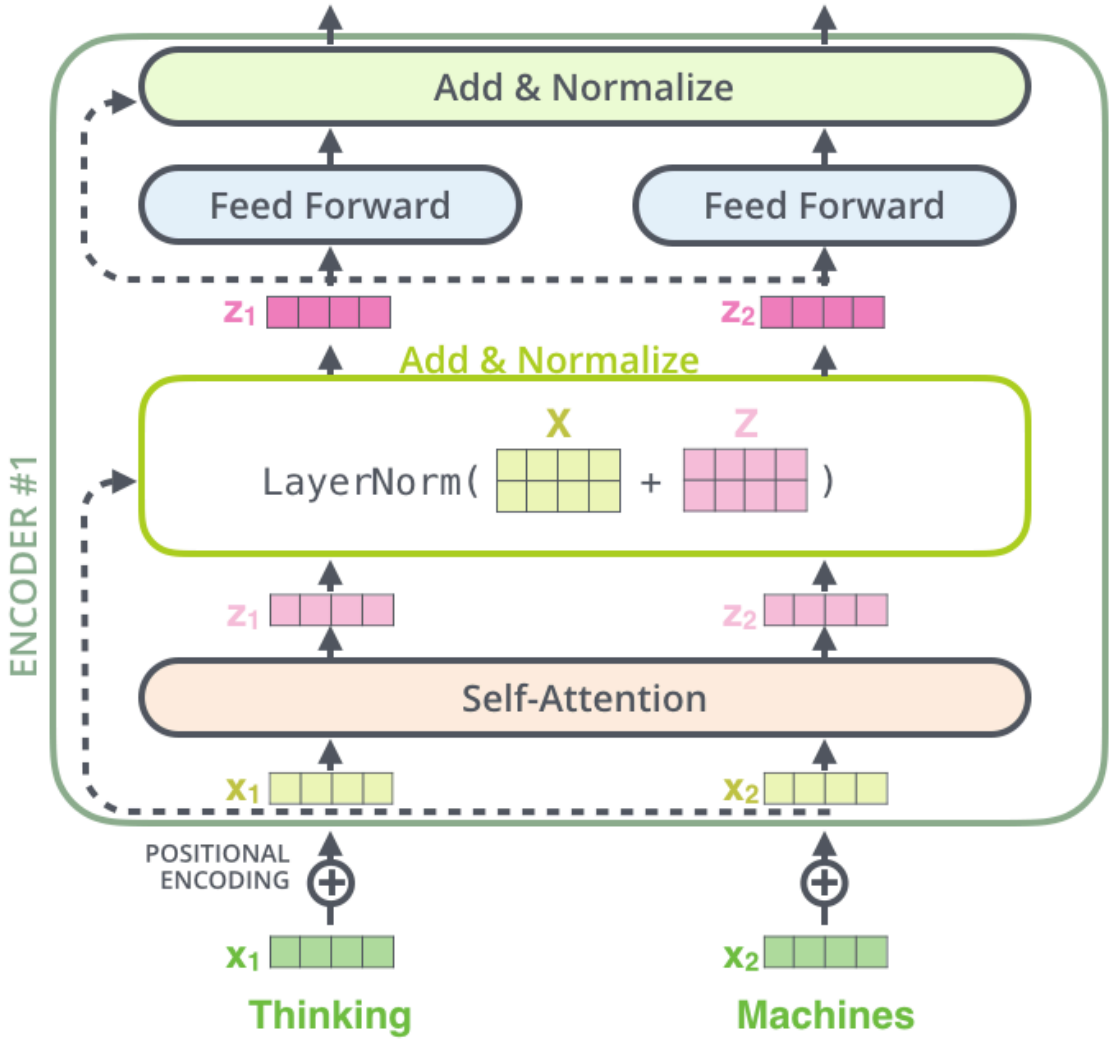
Transformer's Encoder



Transformer's Encoder

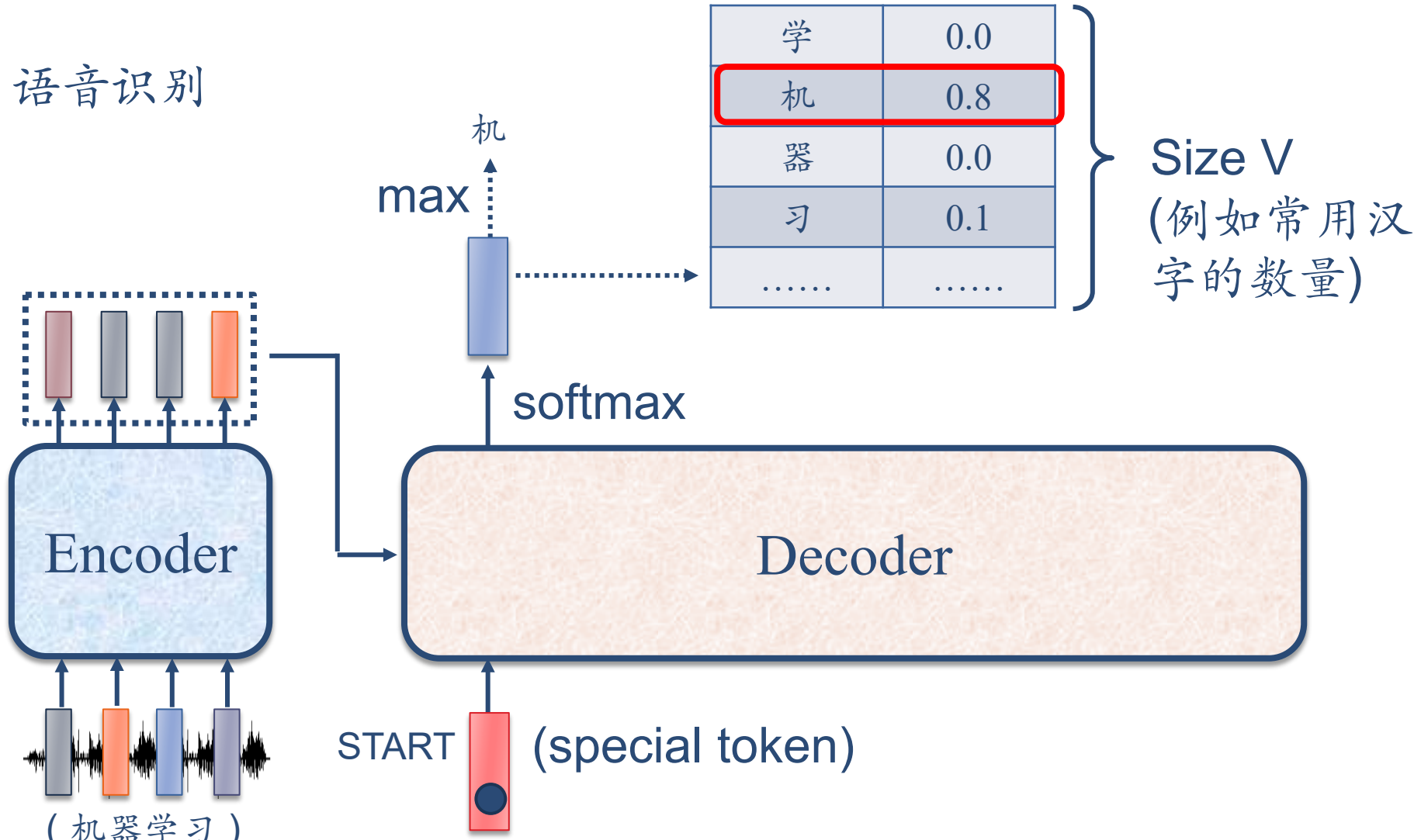
图片来源：http://jalammar.github.io/illustrated-transformer/

- 基于自注意力的模型
- 引入其它操作
  - 位置编码
  - 层归一化
  - 直连边
  - 逐位的FNN

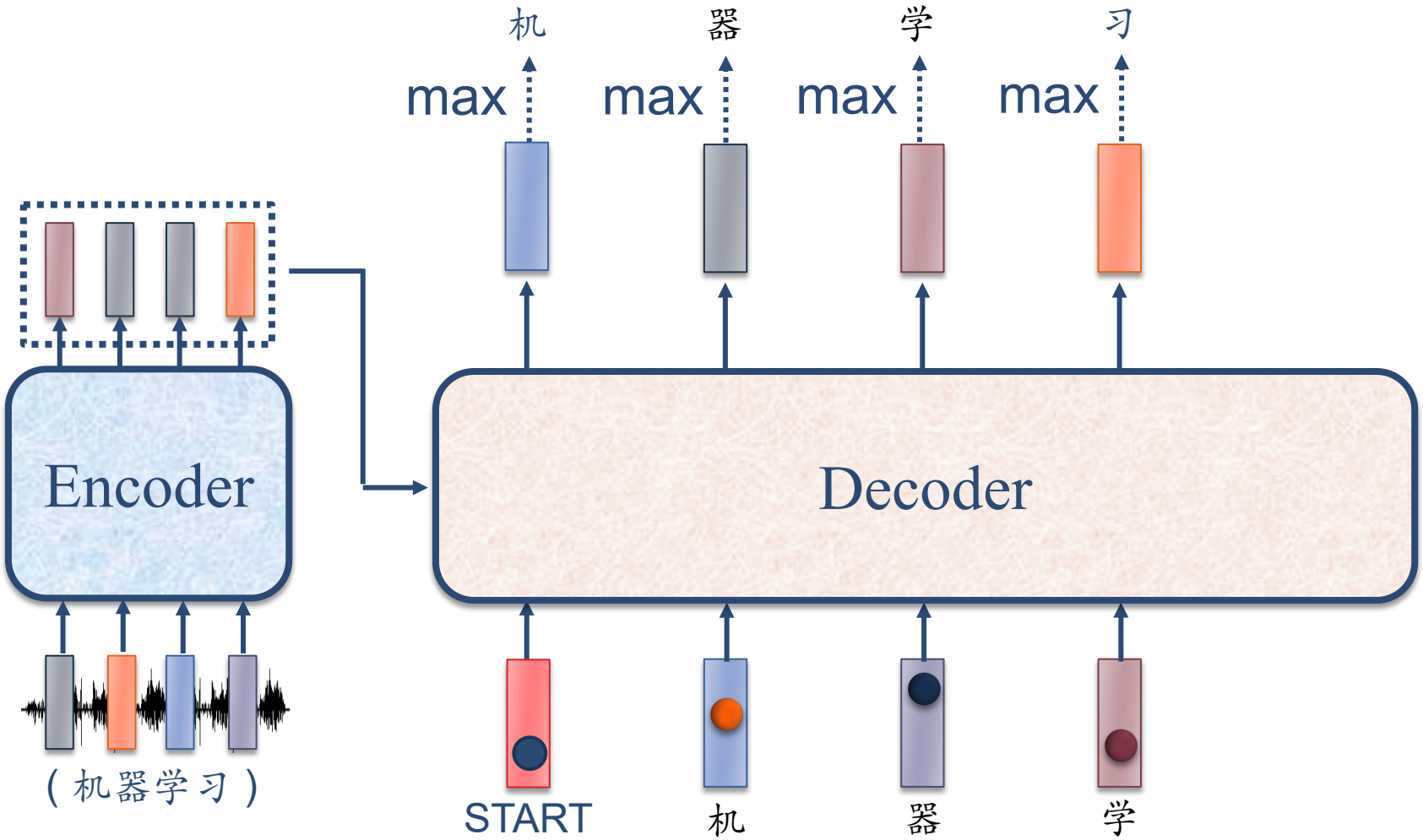


Transformer's Decoder——Autoregressive（AT）

案例：语音识别

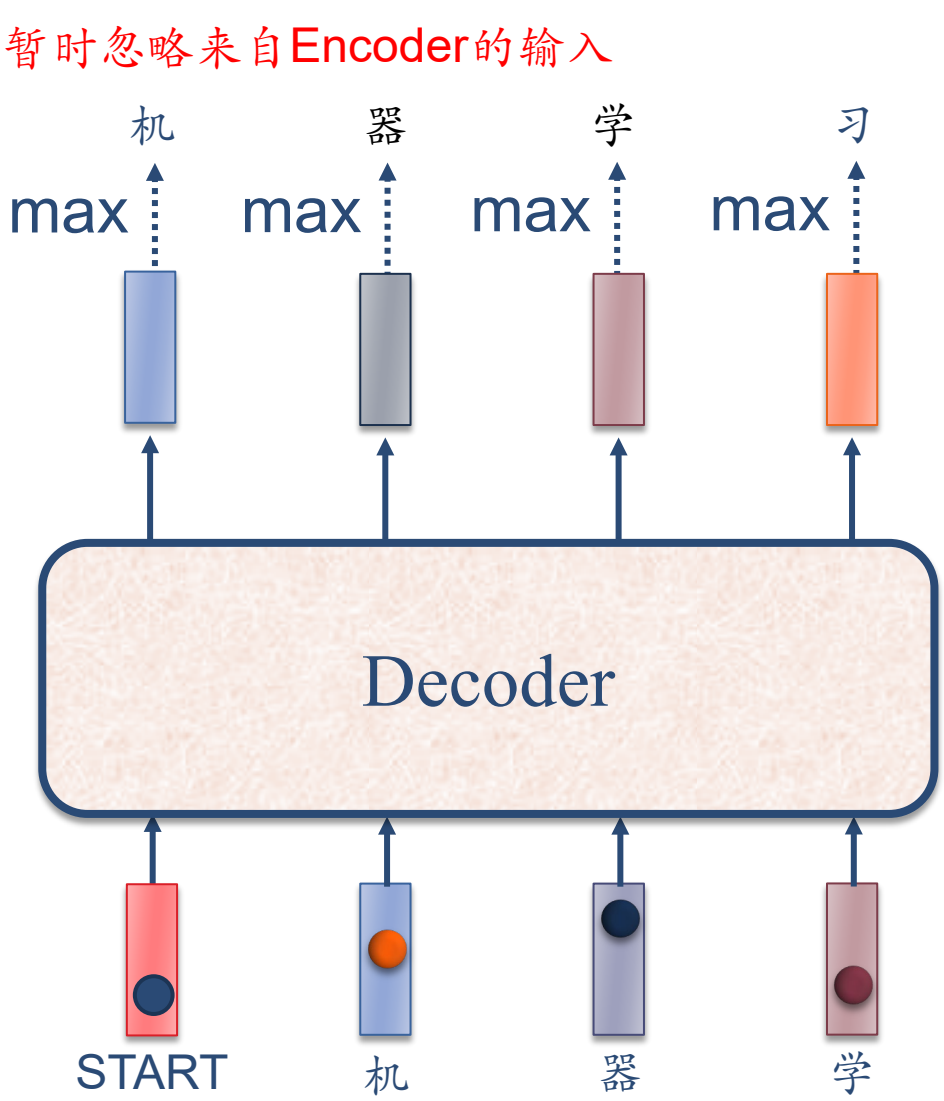


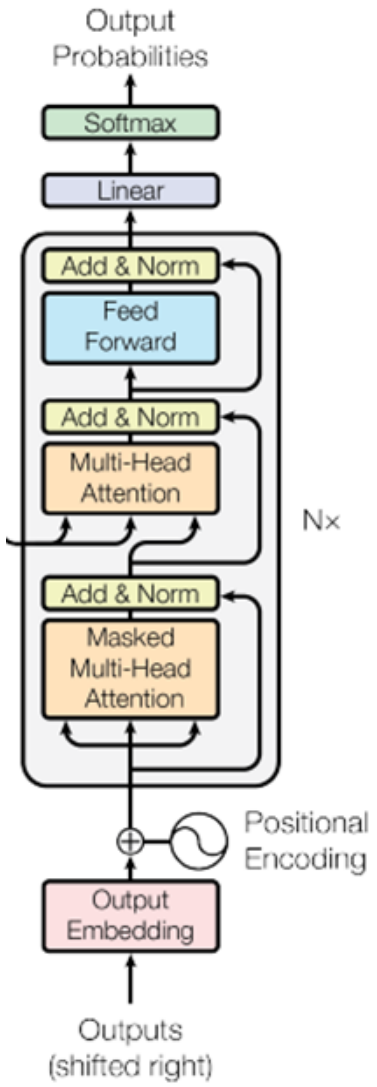
Transformer's Decoder——Autoregressive（AT）

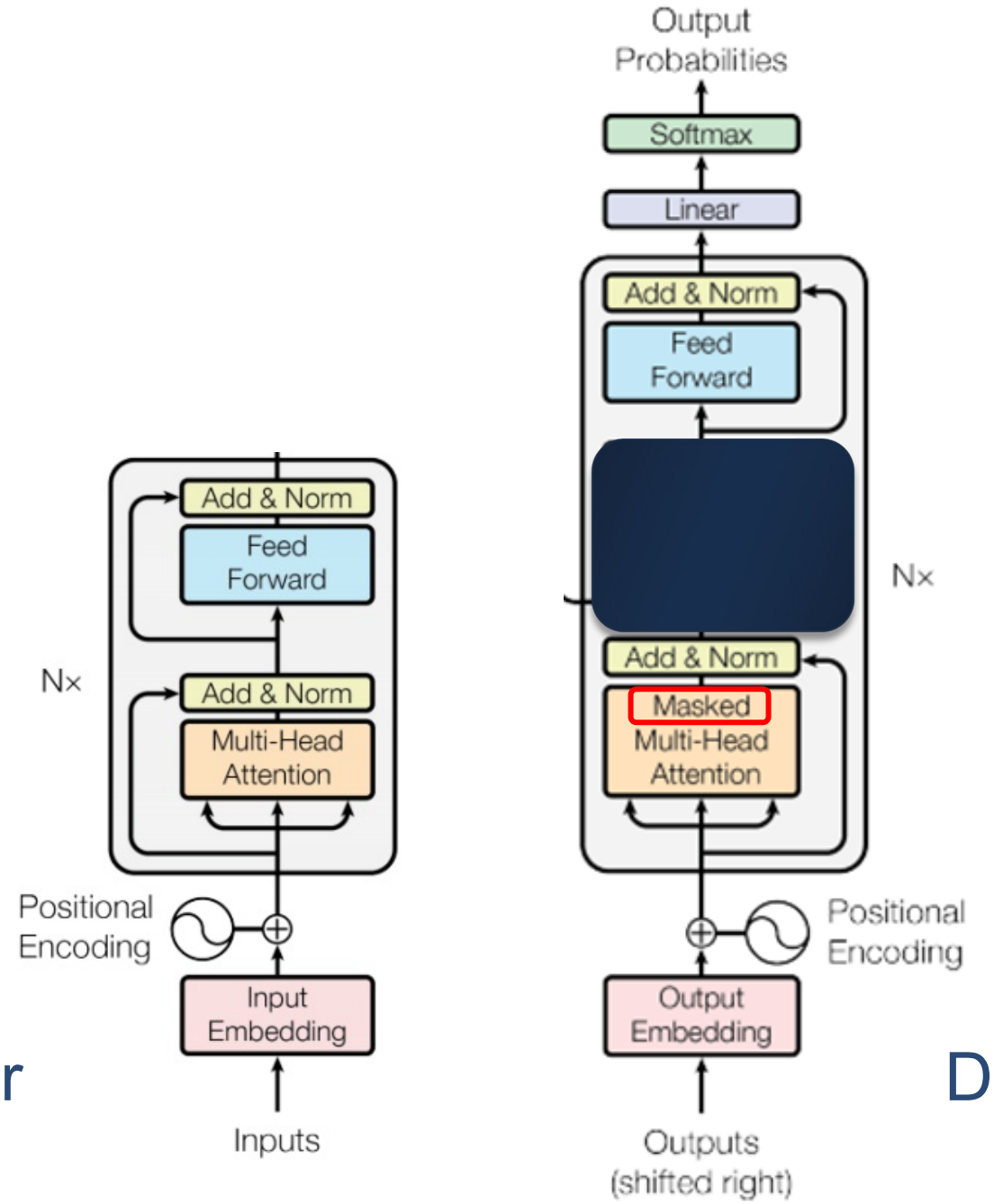


Transformer's Decoder——Autoregressive（AT）

暂时忽略来自Encoder的输入



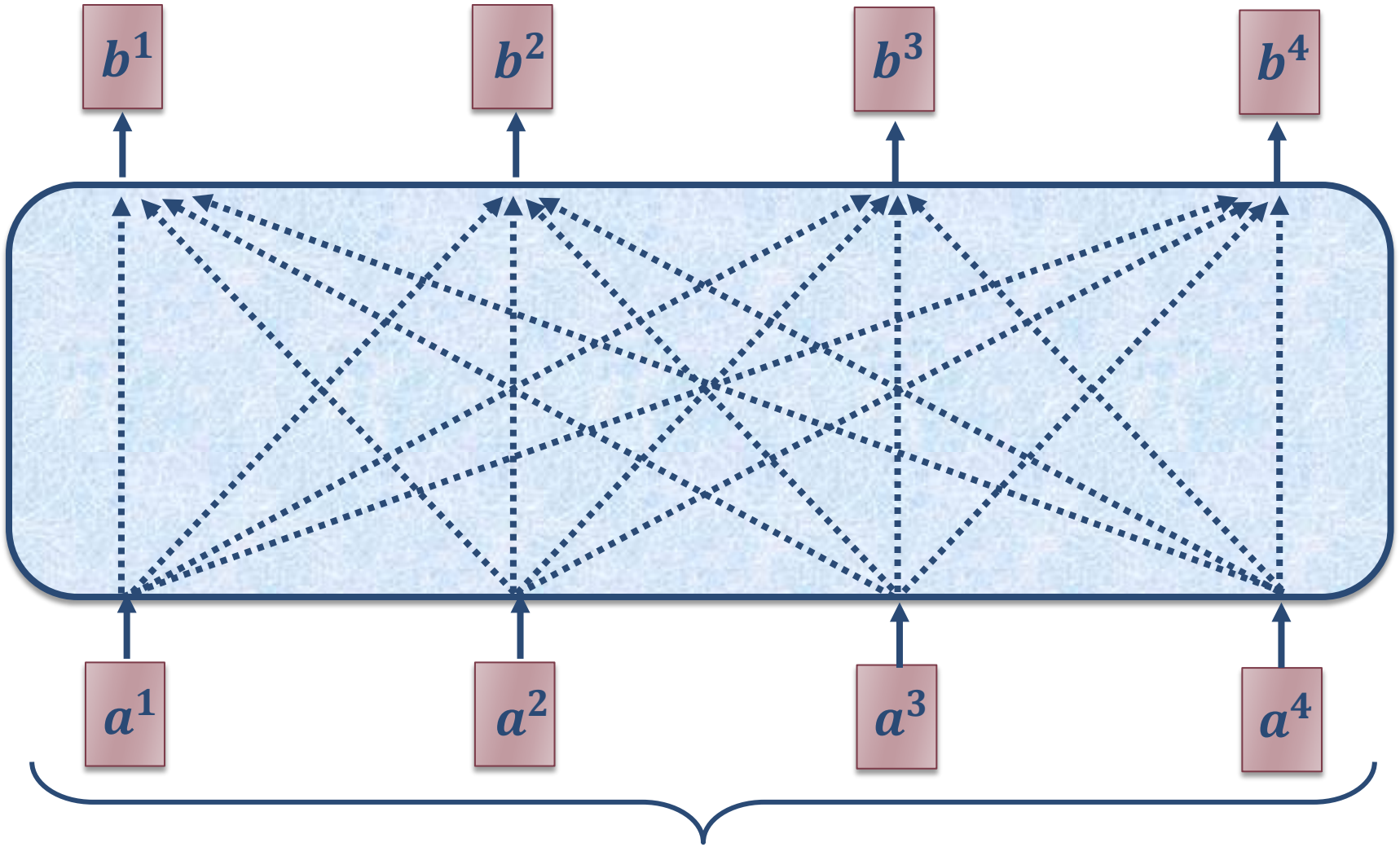




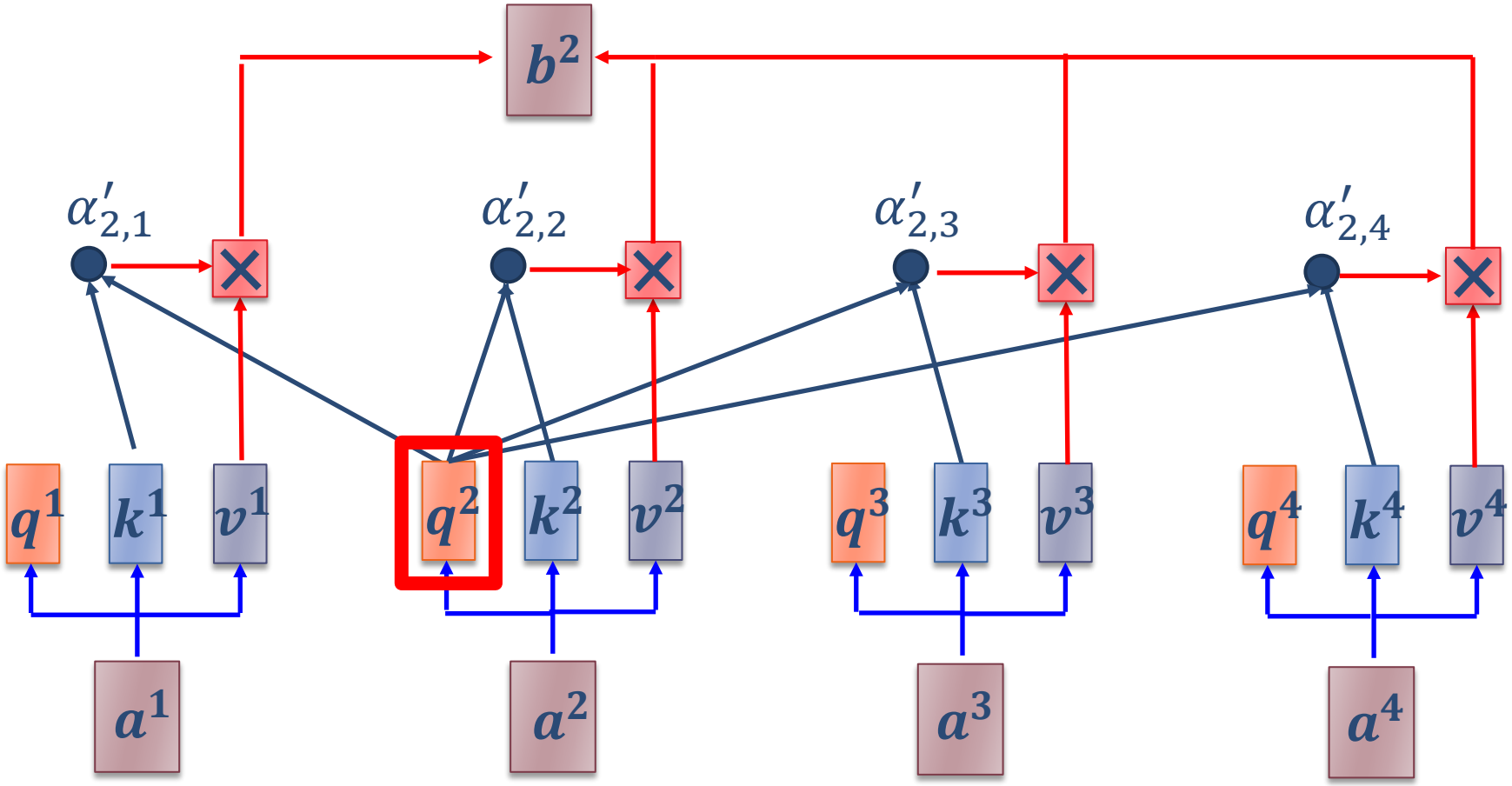
Encoder

Decoder

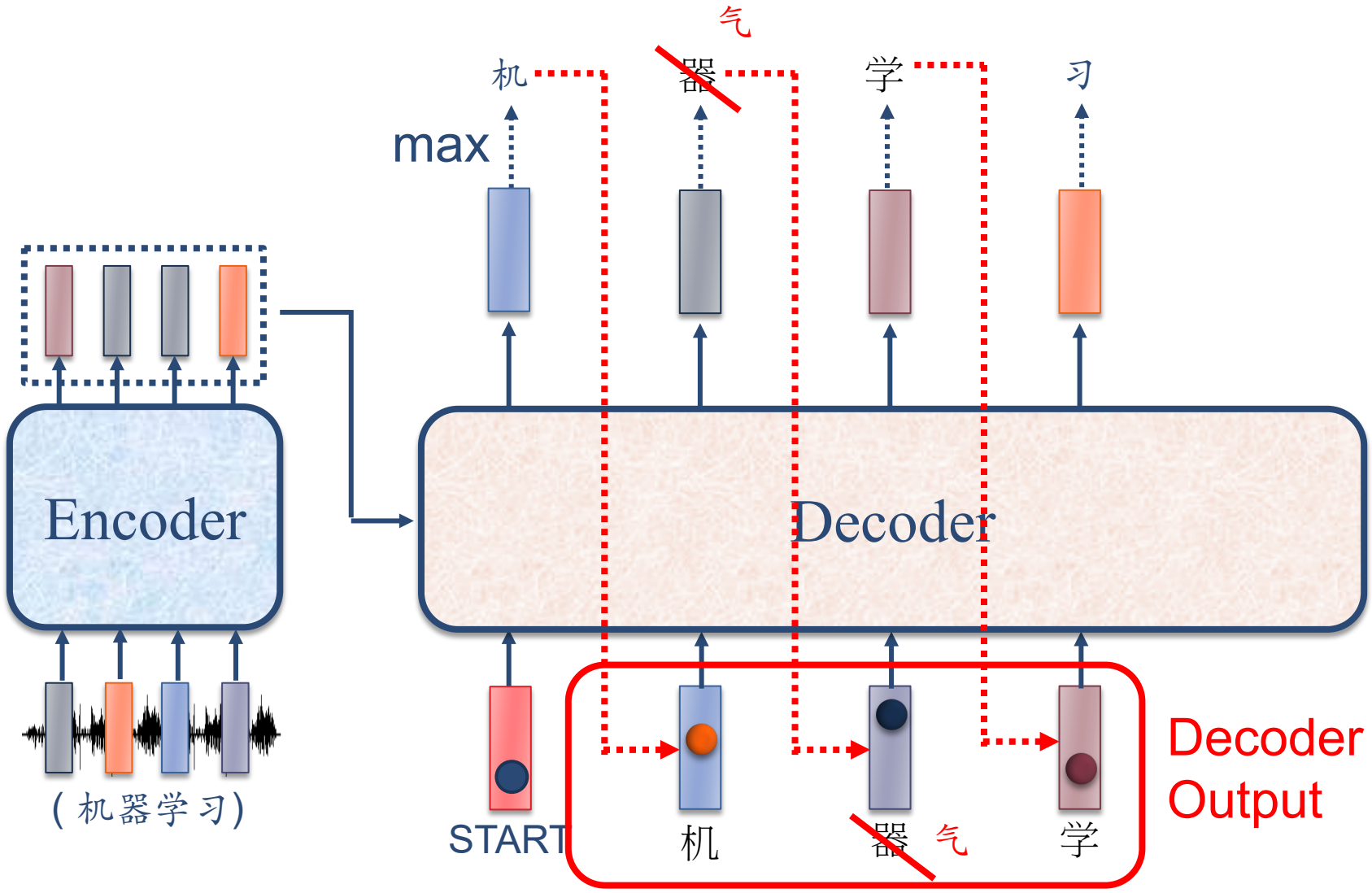
Self-attention → Masked Self-attention



Self-attention → Masked Self-attention



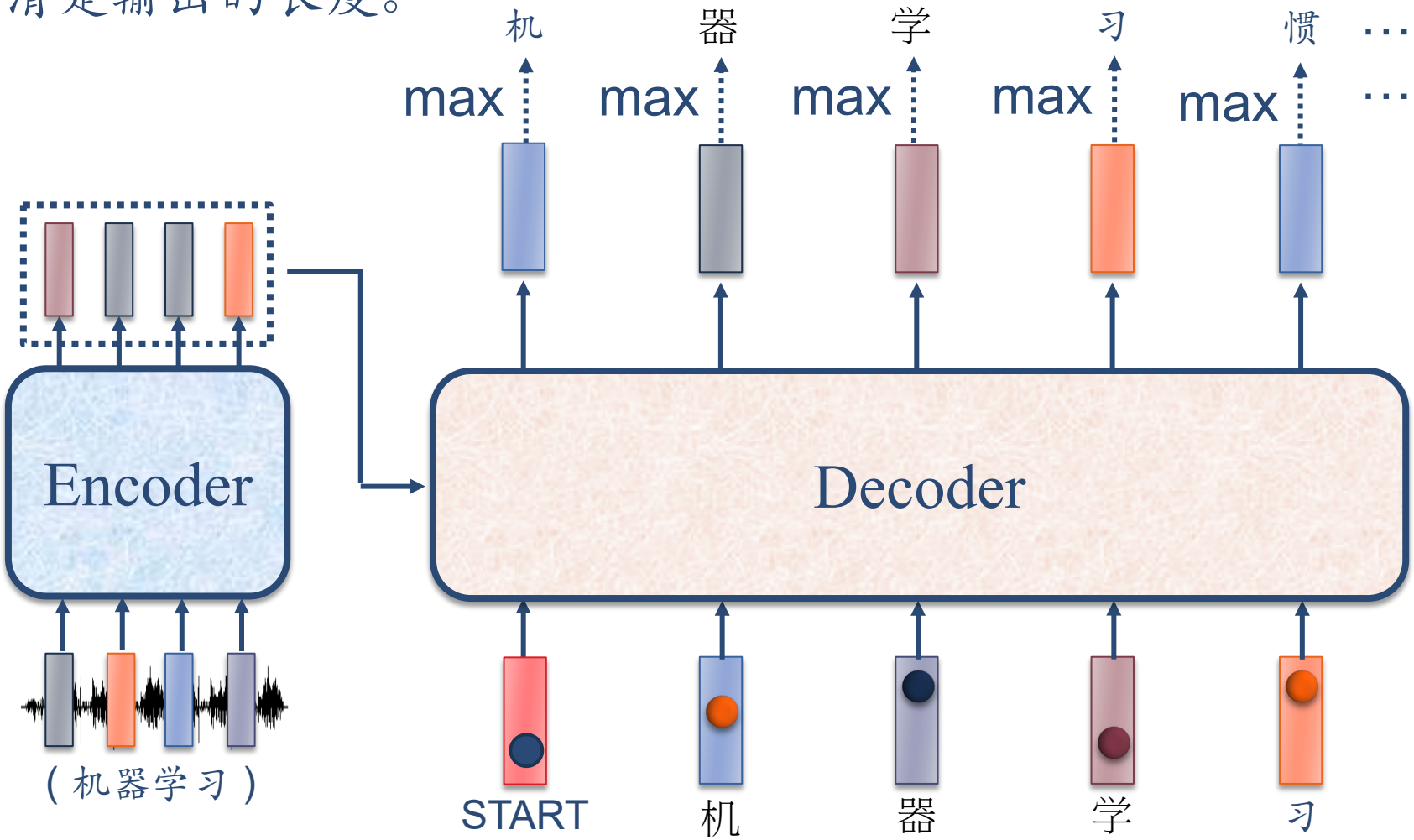
Transformer's Decoder——Autoregressive（AT）



Transformer's Decoder——Autoregressive（AT）

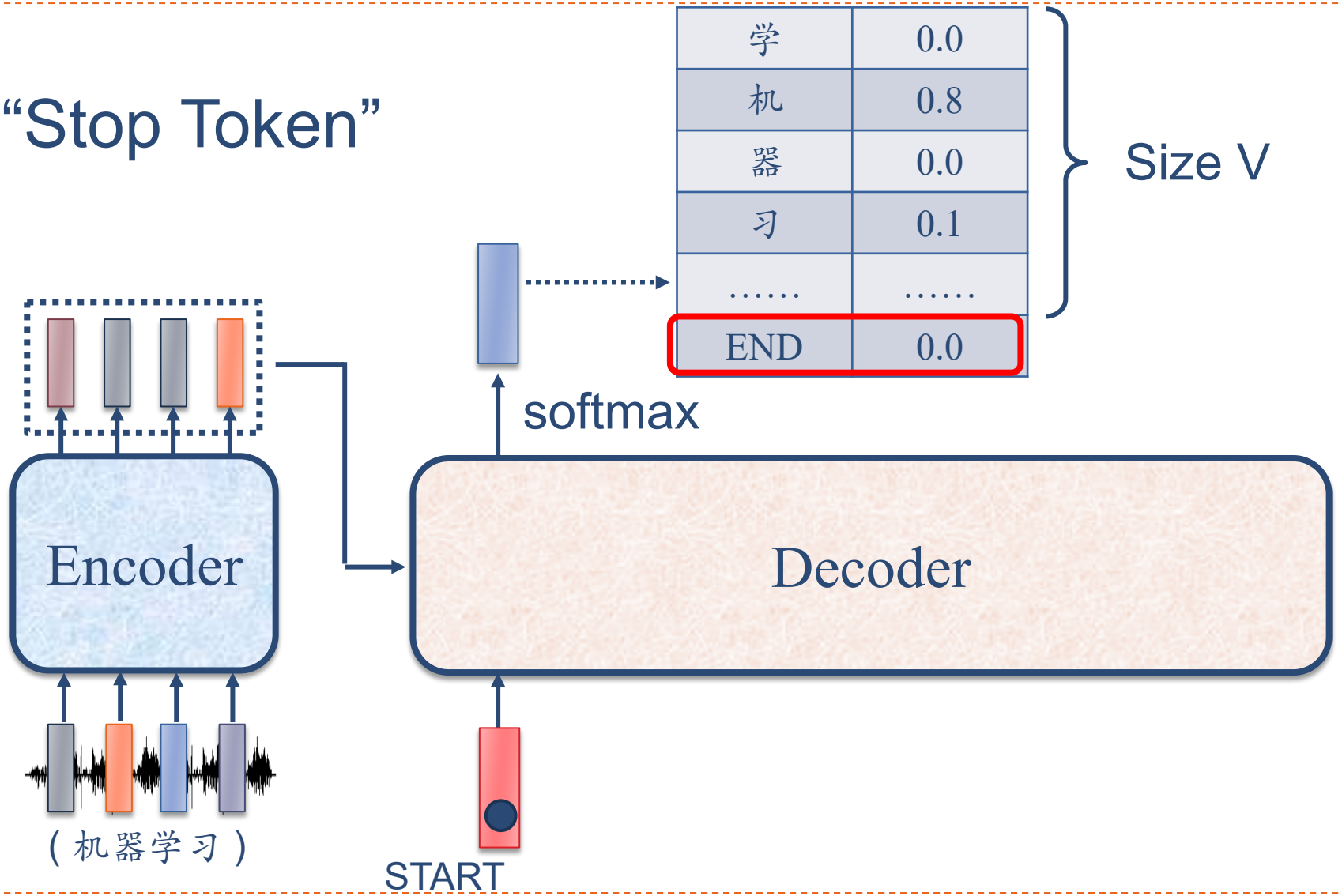
我们并不清楚输出的长度。

Never stop!

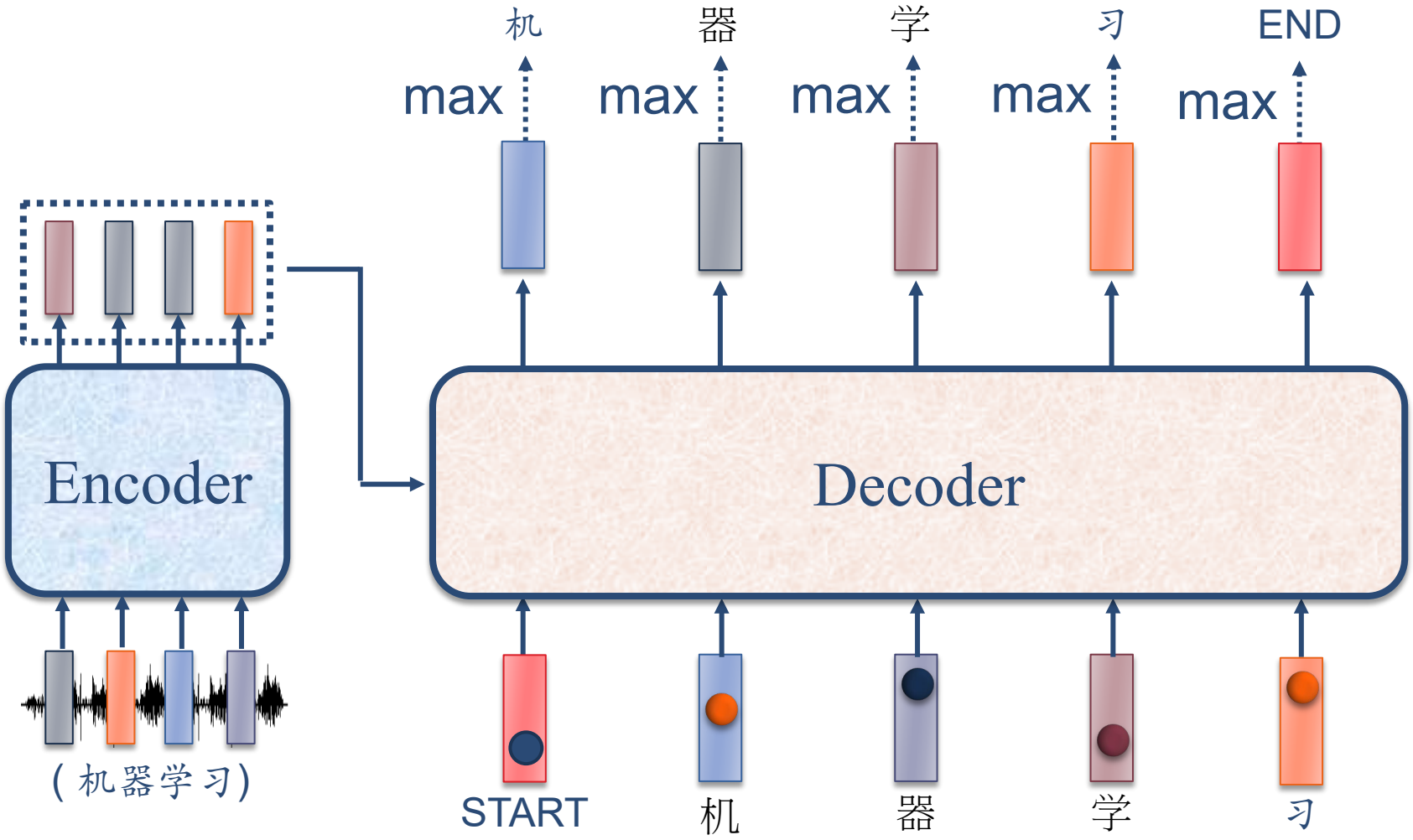


Transformer's Decoder——Autoregressive（AT）

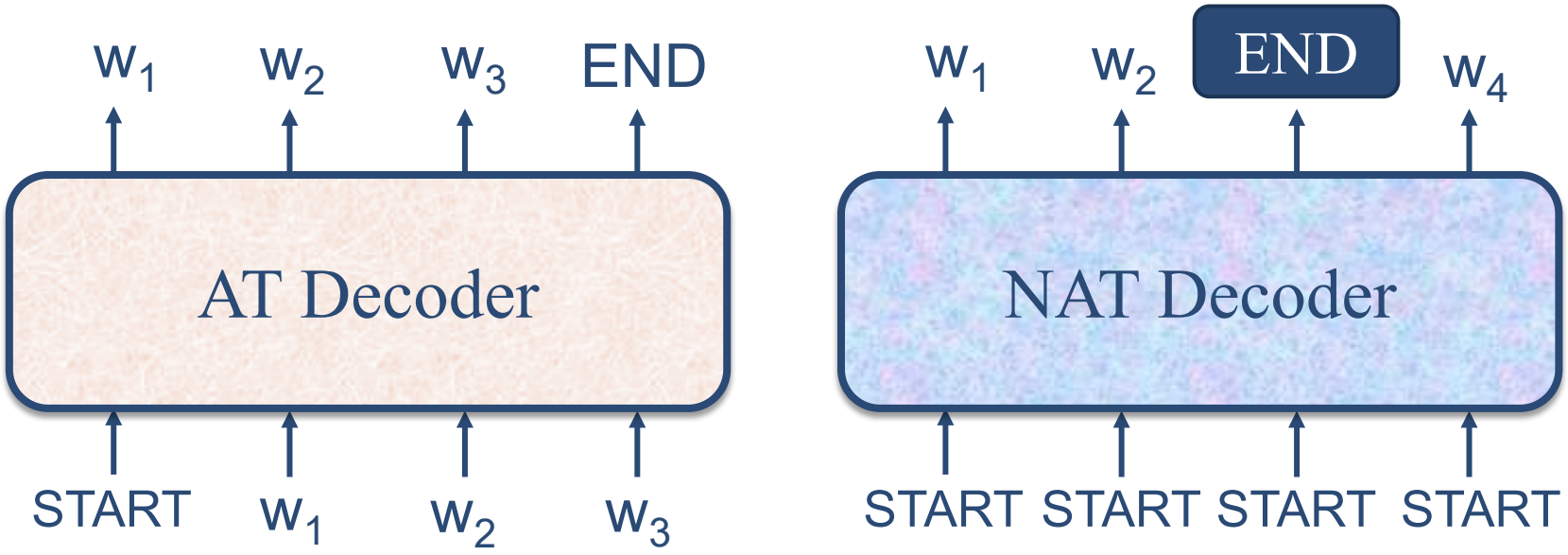
添加"Stop Token"



Transformer's Decoder——Autoregressive（AT）

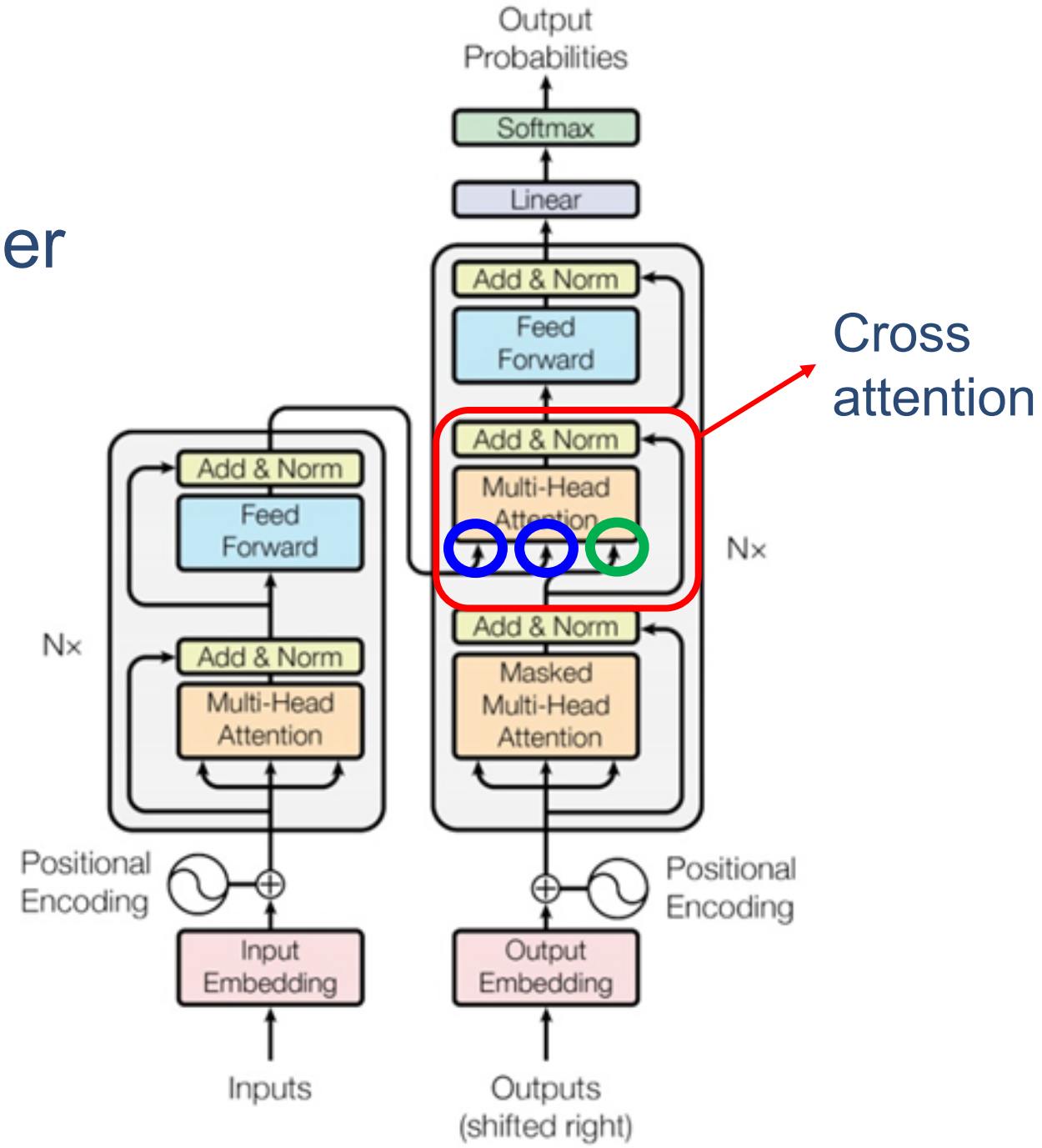


Transformer's Decoder——Non-Autoregressive（NAT）

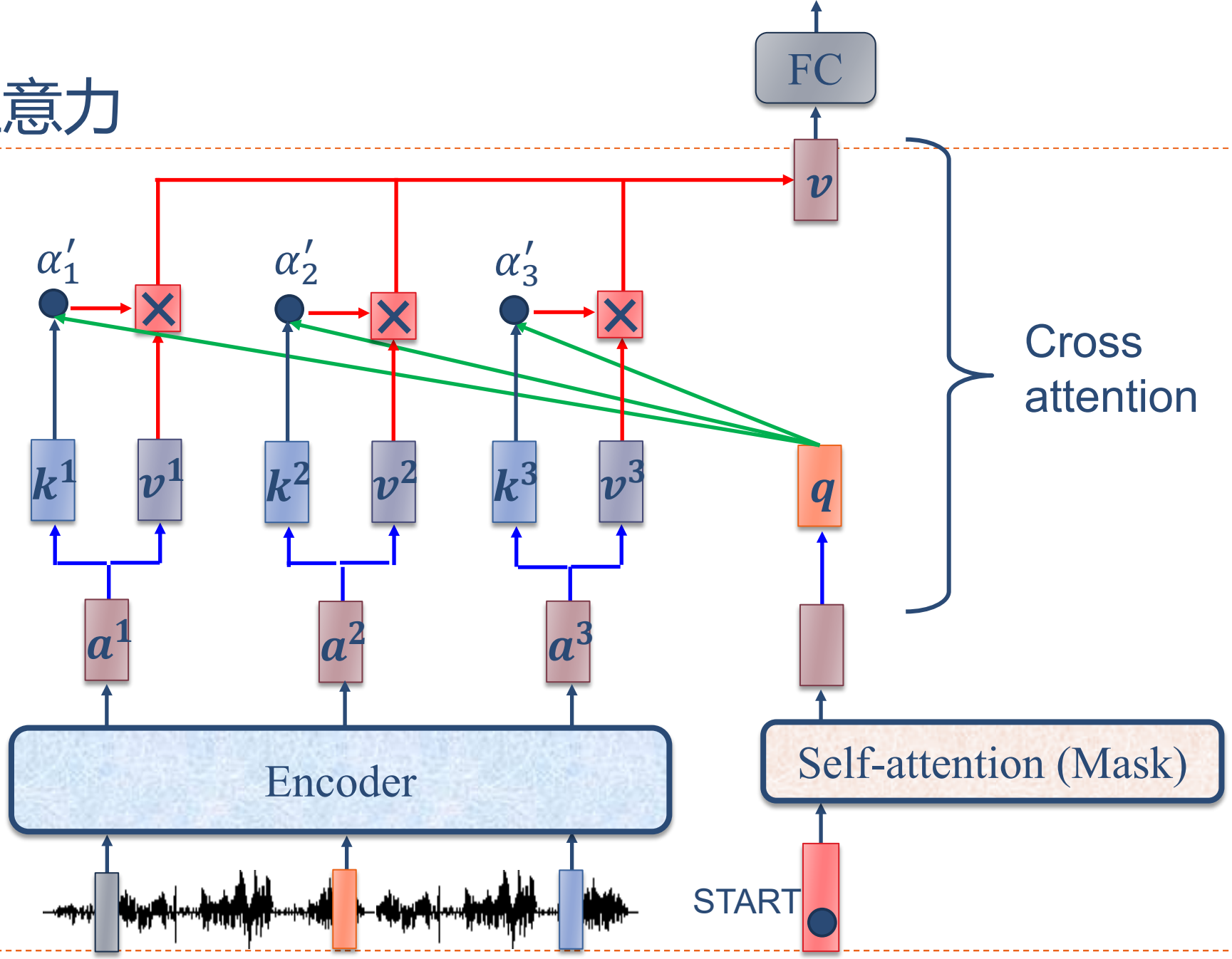


- NAT解码器一次生成整个序列；
- NAT怎样决定输出的长度？用另一个模型预测输出长度；或截取END之前的内容。
- NAT的优势：可并行（而AT只能逐个Token生成）；能够很好地控制输出长度；
- NAT的劣势：通常性能比AT差。

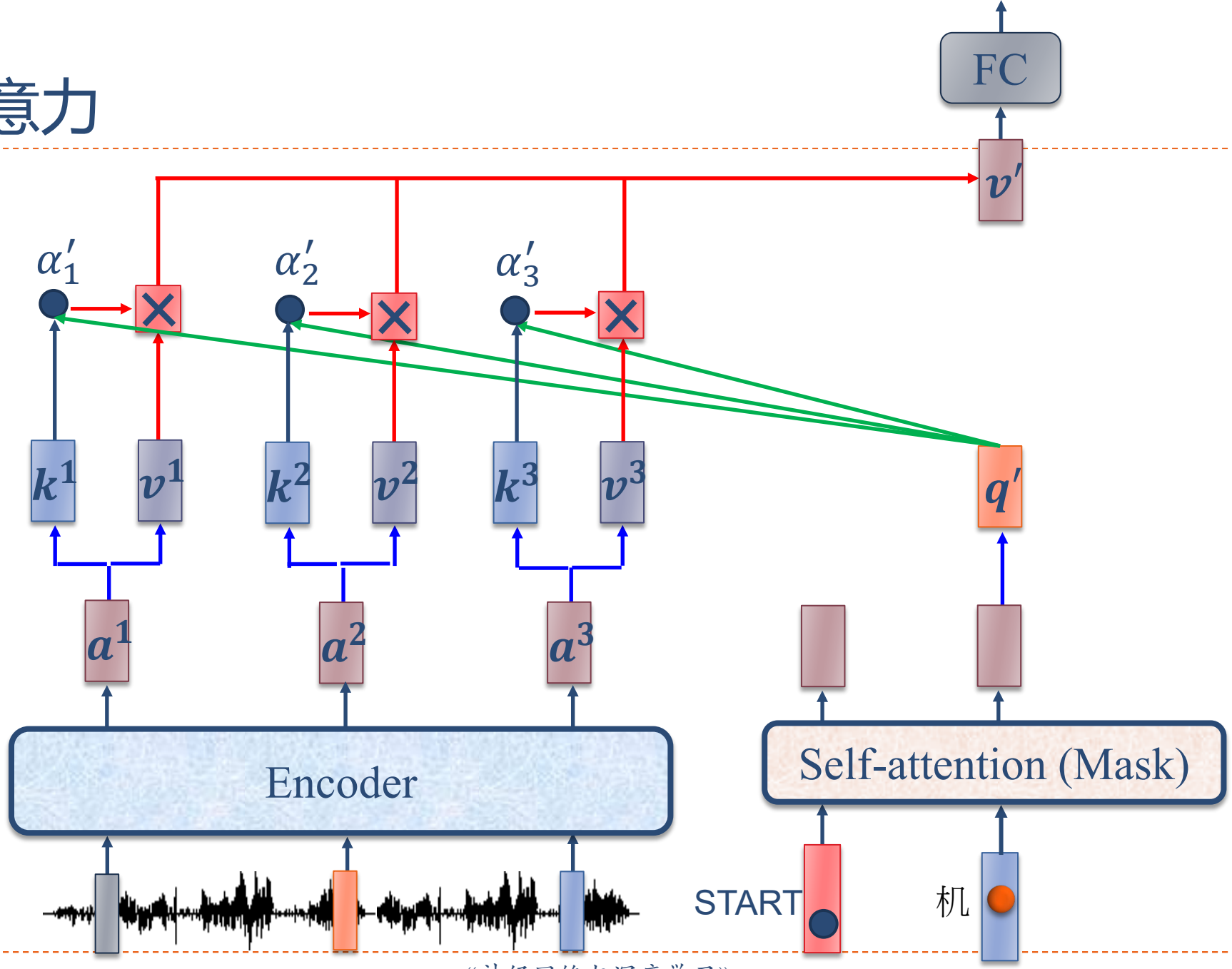
Encoder-Decoder



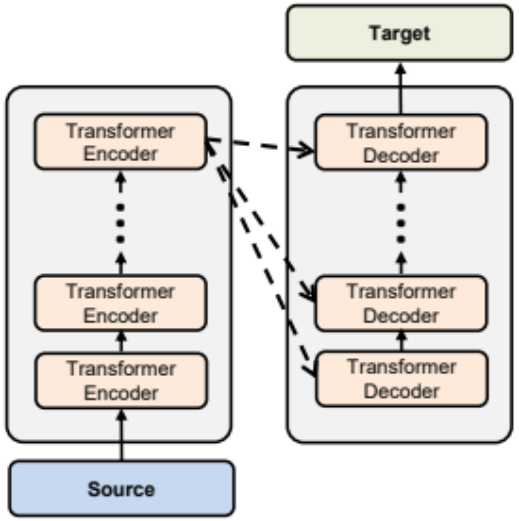
交叉注意力


FC
v
α'_1
α'_2
α'_3
k^1
v^1
k^2
v^2
k^3
v^3
q
Cross attention
a^1
a^2
a^3
Encoder
Self-attention (Mask)
START

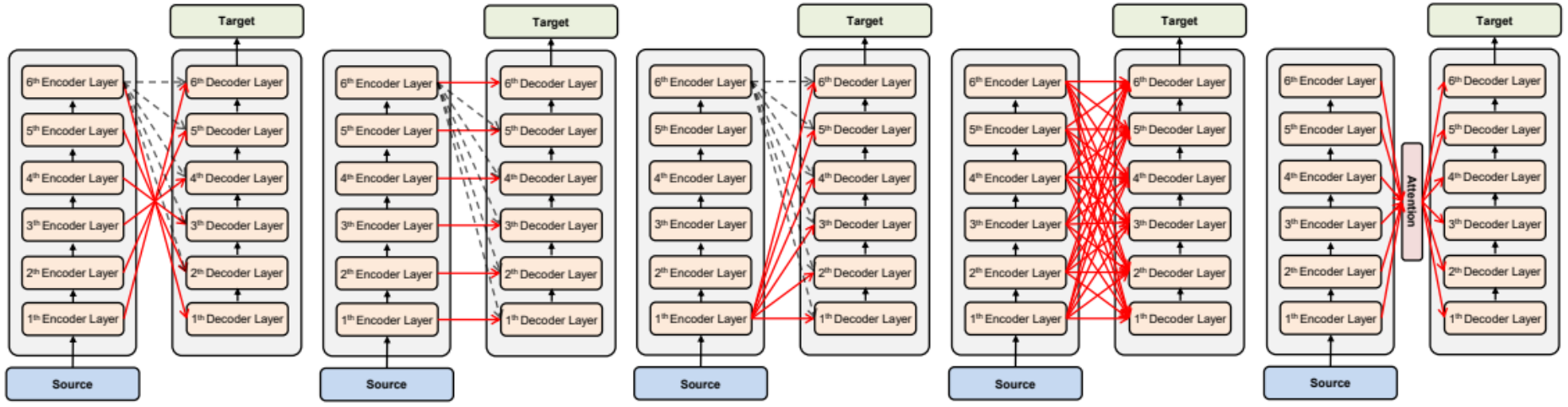
交叉注意力


FC
v'
α'_1
α'_2
α'_3
k^1
v^1
k^2
v^2
k^3
v^3
q'
a^1
a^2
a^3
Encoder
Self-attention (Mask)
START
机

交叉注意力



(a) Conventional Transformer



(a) Granularity Consistent Attention (b) Granularity Parallel Attention (c) Fine-Grained Attention (d) Full Matching Attention (e) Adaptive Matching Attention

Transformer的训练

Teacher Forcing: 用真实标签作为Decoder输入。