

A collage image featuring Transformers and Bert the Muppet. On the left, a large, dark grey Transformer is partially visible. In the center, a yellow and black Transformer is shown. On the right, the head of Bert the Muppet is shown with his characteristic black spiky hair and orange nose. The name 'BERT' is printed on his forehead. The background is a dark grey gradient.

Transformer

# Semi-supervised Sequence Learning

context2Vec

Pre-trained seq2seq



**ELMo**

**ULMFiT**

Multi-lingual

**MultiFiT**

Transformer

Bidirectional LM

**GPT**

Larger model  
More data

**GPT-2**

Defense



**Grover**



**BERT**

Cross-lingual

Multi-task

+ Generation

**XLNet**

**UDify**

**MT-DNN**

Knowledge distillation

**MT-DNN<sub>KD</sub>**

**MASS**

**UniLM**

Span prediction  
Remove NSP

Longer time  
Remove NSP  
More data

**SpanBERT**

**RoBERTa**

Permutation LM  
Transformer-XL  
More data

**XLNet**

+ Knowledge Graph



**ERNIE  
(Tsinghua)**

Neural entity linker

**KnowBERT**

Cross-modal

**VideoBERT**  
**CBT**  
**ViLBERT**  
**VisualBERT**  
**B2T2**  
**Unicoder-VL**  
**LXMERT**  
**VL-BERT**  
**UNITER**

Whole Word Masking



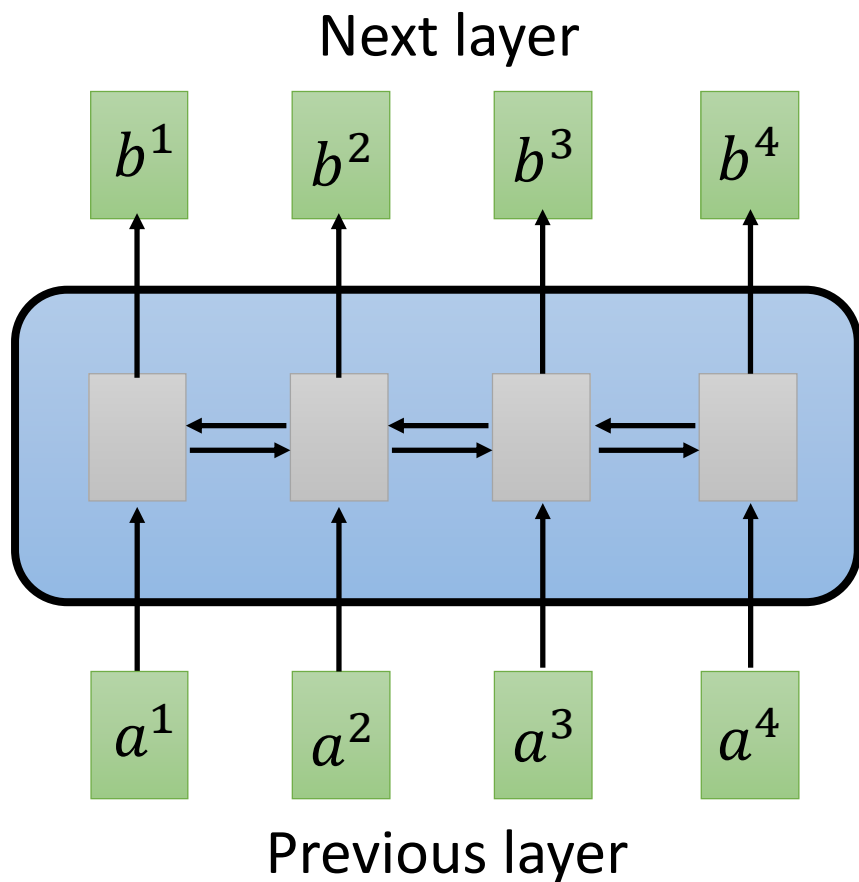
**ERNIE (Baidu)**  
**BERT-wwm**

The image features three Transformers characters from the G1 era against a black background. Optimus Prime is the central figure, shown from the waist up, with his right arm raised in a fist. He has a red and blue color scheme. To his right is Bumblebee, shown from the waist up, with a yellow and black color scheme. In the foreground, the lower legs and feet of Ironhide are visible, showing a blue and grey color scheme. The text 'Transformer' is overlaid in the center in a large, white, sans-serif font. Below it, the text 'Seq2seq model with "Self-attention"' is also overlaid in a smaller, white, sans-serif font.

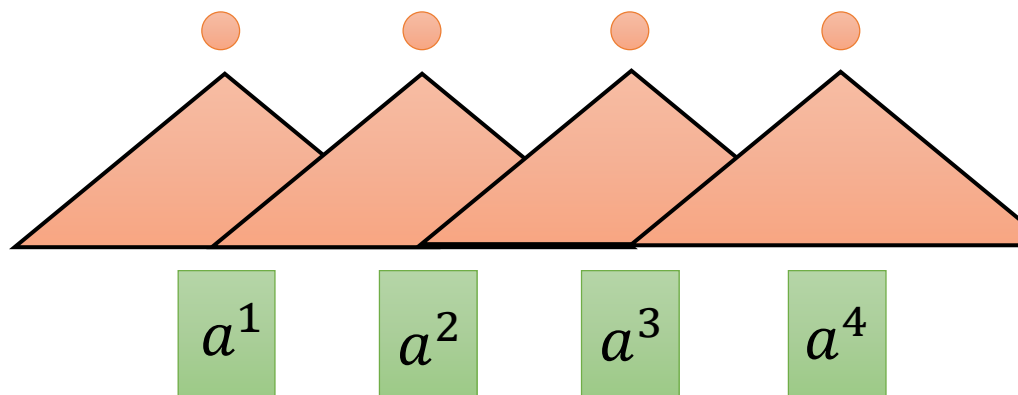
# Transformer

Seq2seq model with "Self-attention"

# Sequence

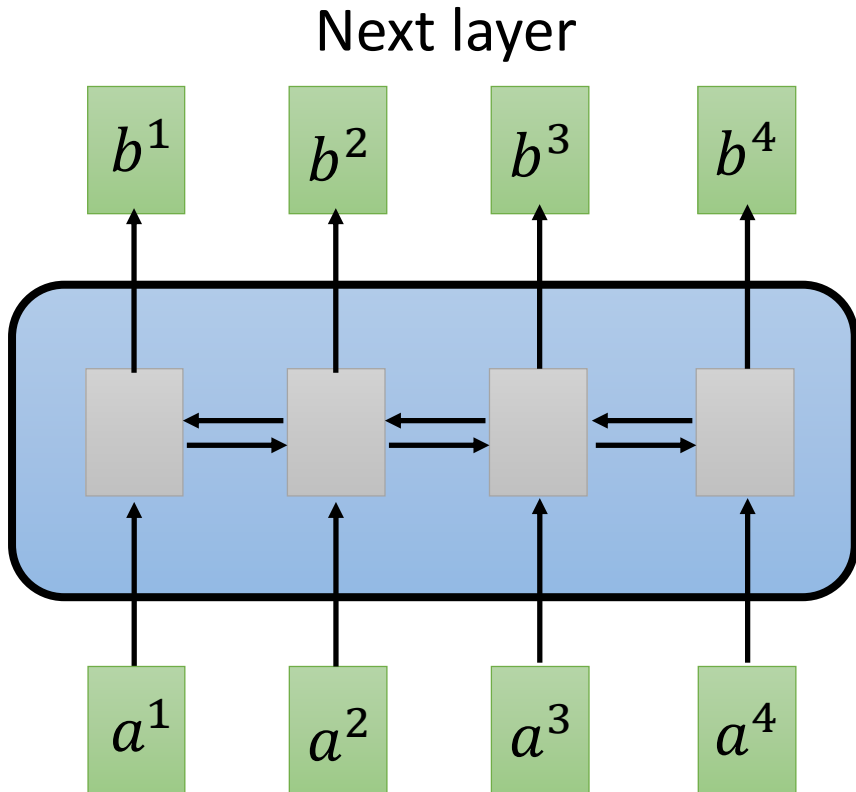


Hard to parallel !



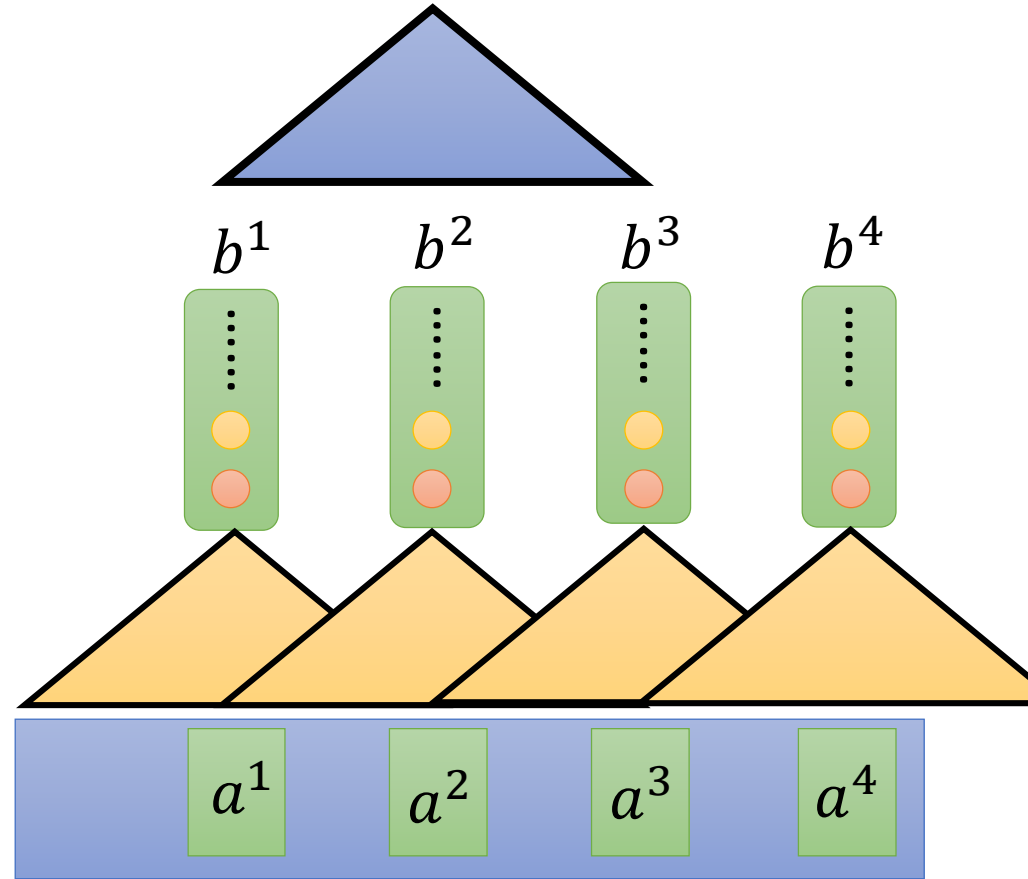
Using CNN to replace RNN

# Sequence



Hard to parallel

Filters in higher layer can consider longer sequence

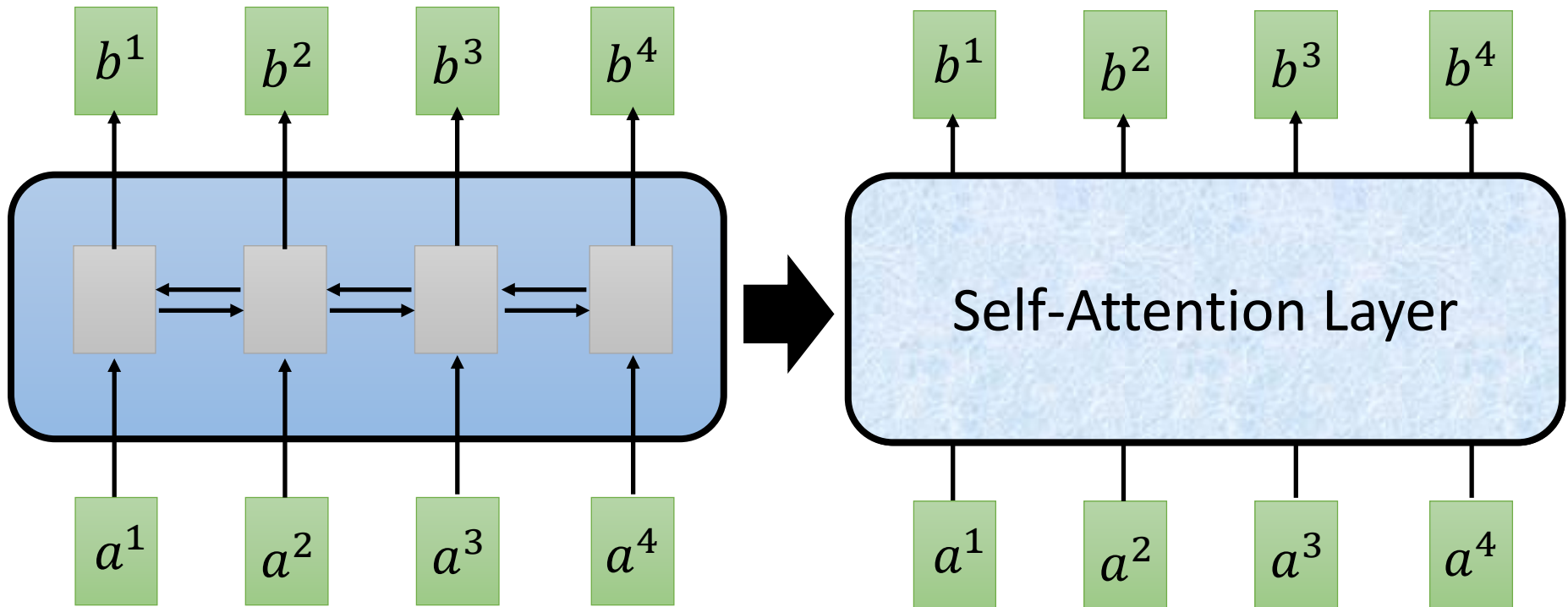


Using CNN to replace RNN  
(CNN can parallel)

# Self-Attention

$b^i$  is obtained based on the whole input sequence.

$b^1, b^2, b^3, b^4$  can be parallelly computed.



You can try to replace any thing that has been done by RNN with self-attention.

# Self-attention

<https://arxiv.org/abs/1706.03762>



$q$ : query (to match others)

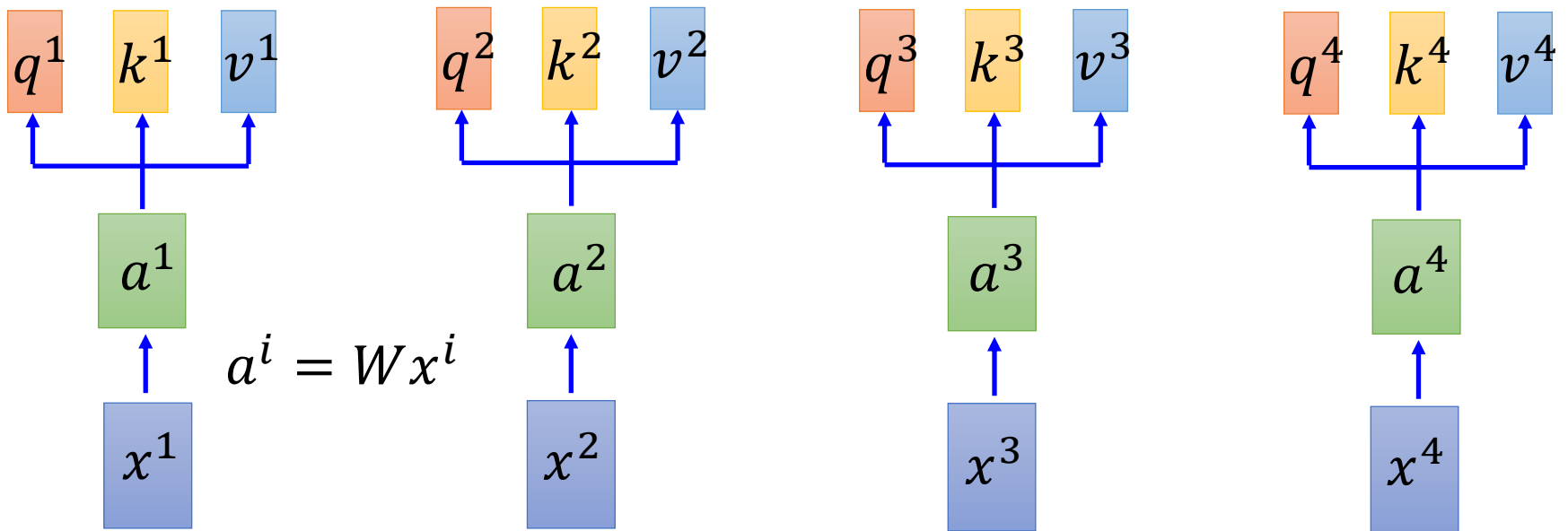
$$q^i = W^q a^i$$

$k$ : key (to be matched)

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

$v$ : information to be extracted

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



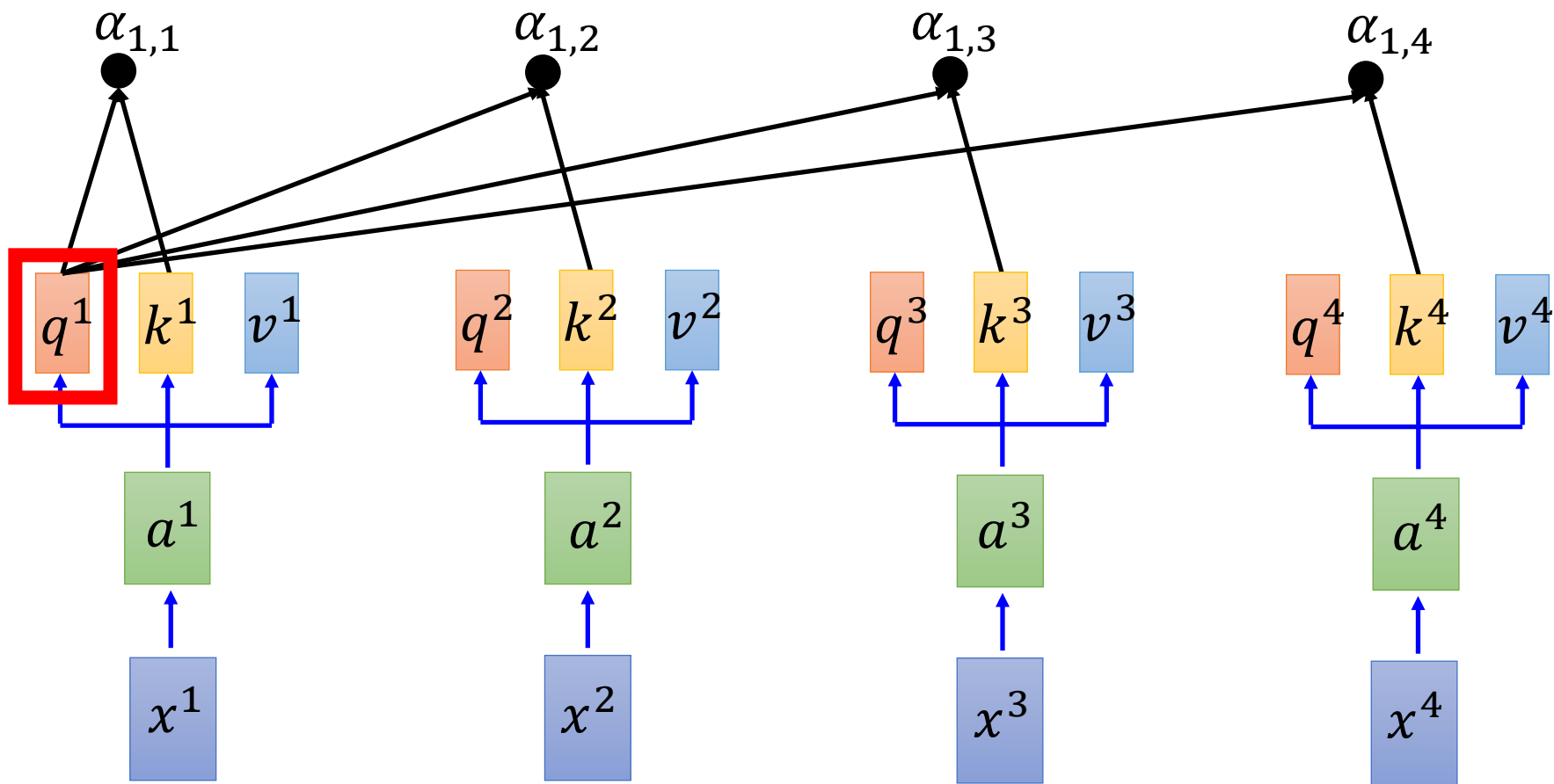
# Self-attention

拿每个 query  $q$  去对每个 key  $k$  做 attention

$d$  is the dim of  $q$  and  $k$

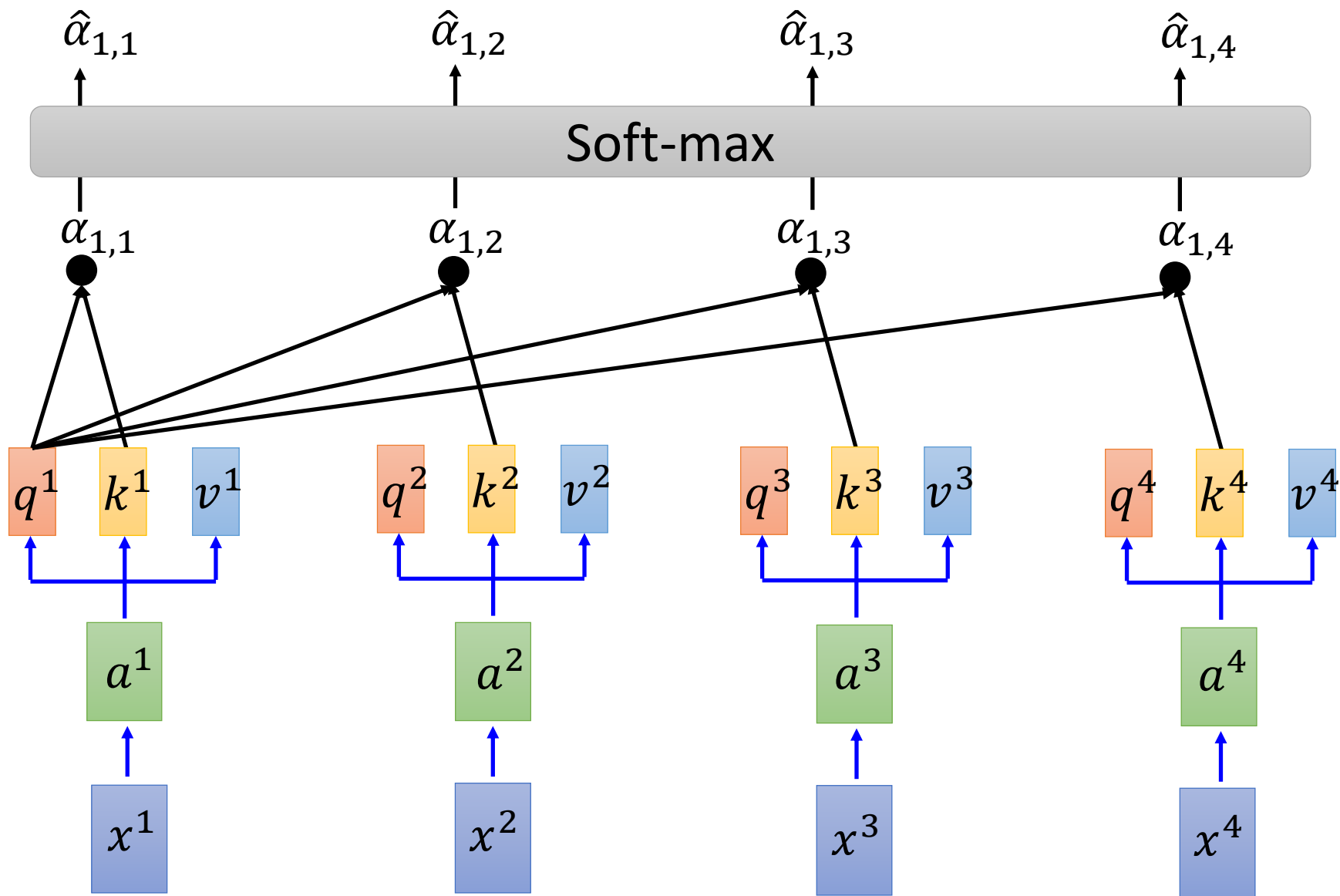
Scaled Dot-Product Attention:  $\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\text{dot product}} / \sqrt{d}$

dot product



# Self-attention

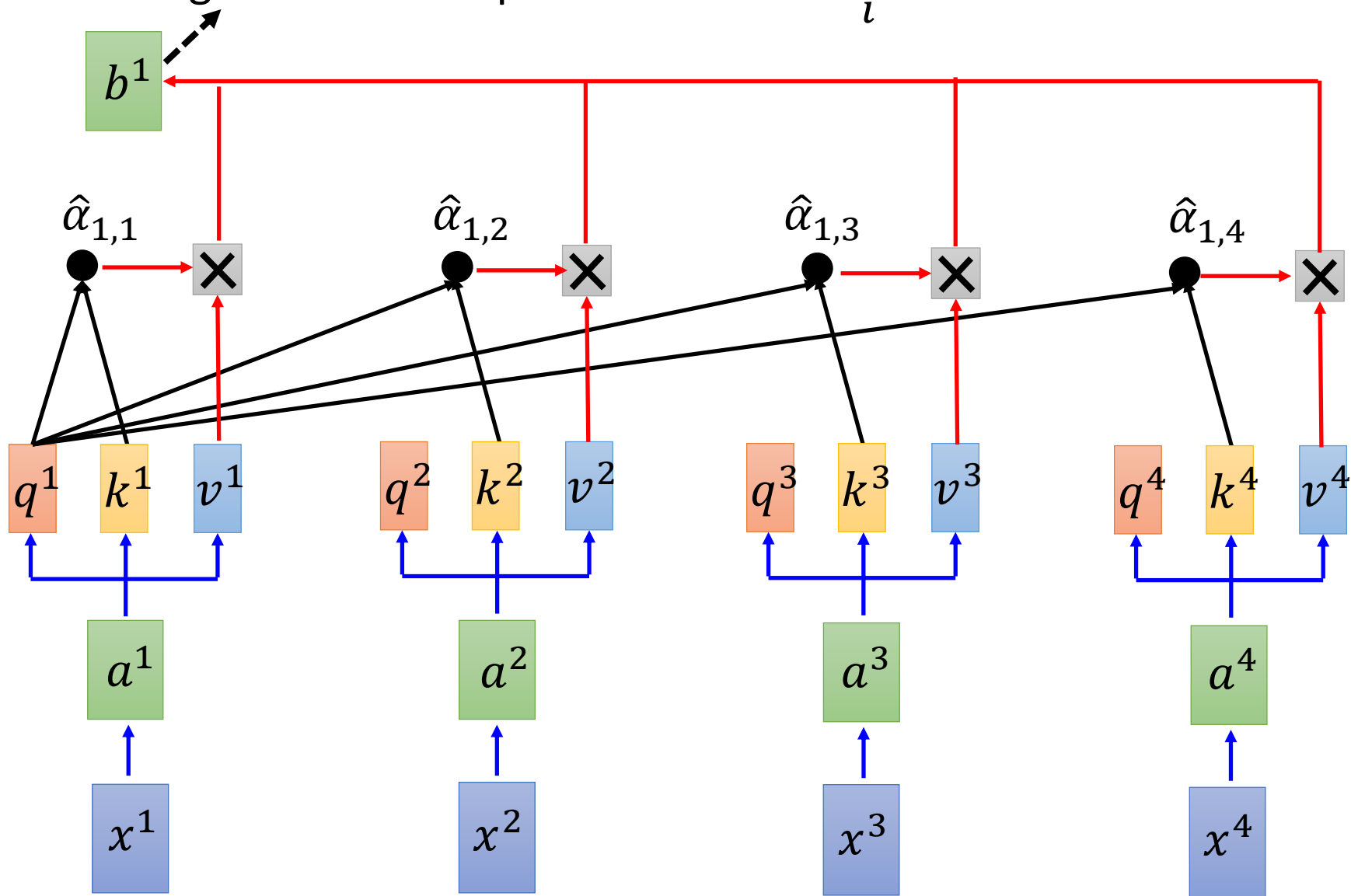
$$\hat{\alpha}_{1,i} = \exp(\alpha_{1,i}) / \sum_j \exp(\alpha_{1,j})$$



# Self-attention

Considering the whole sequence

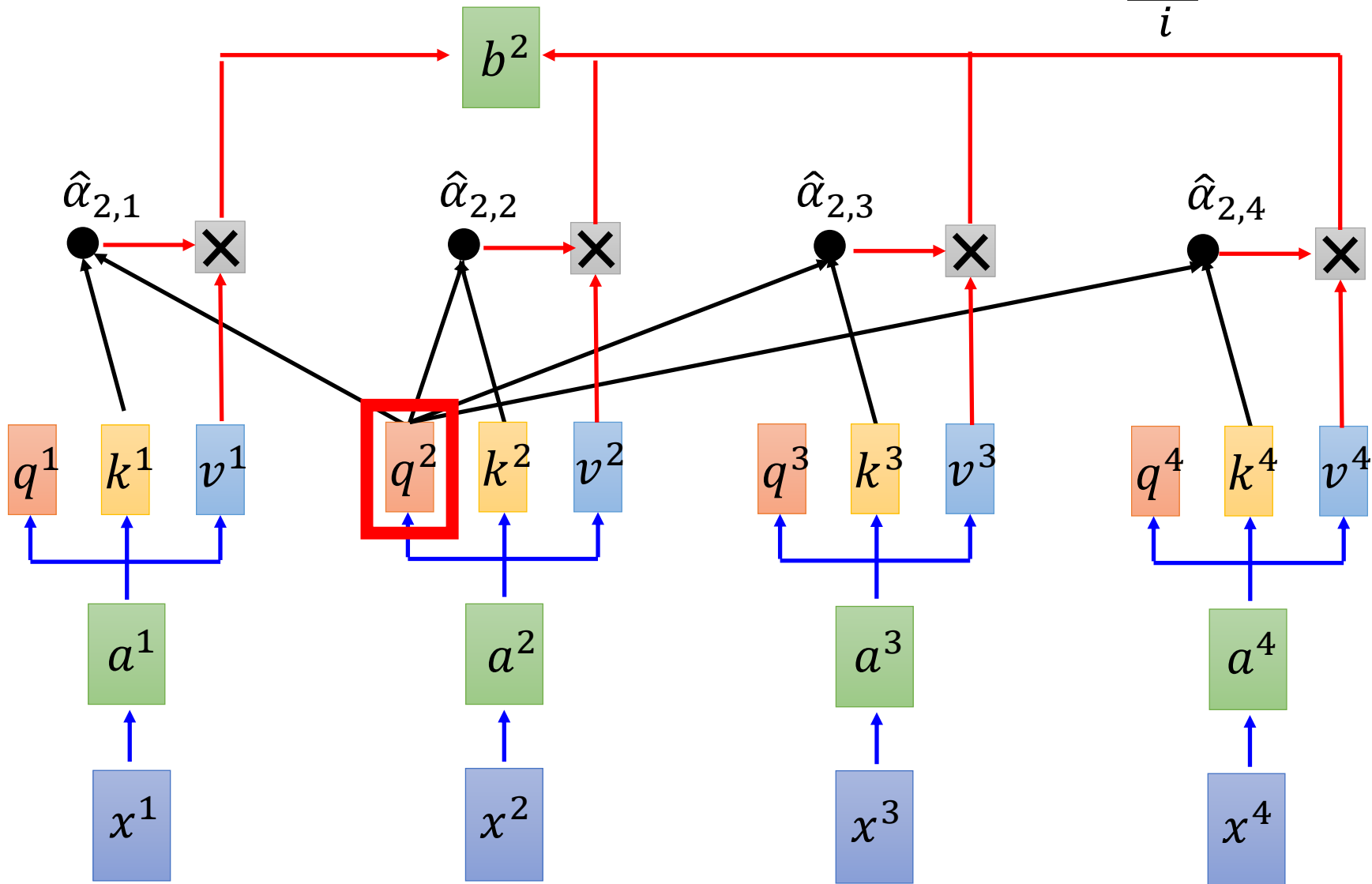
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$



# Self-attention

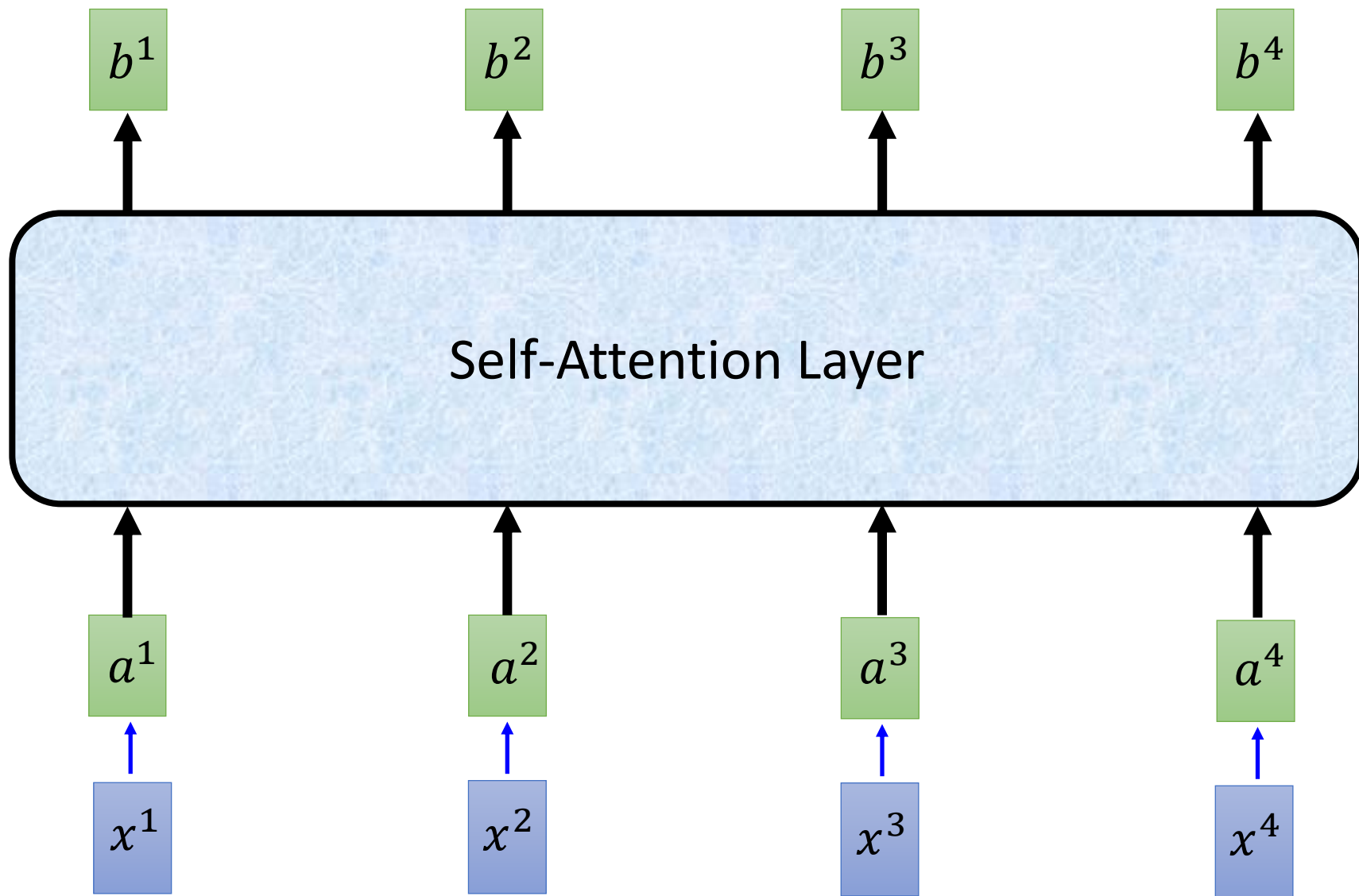
拿每个 query q 去对每个 key k 做 attention

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



# Self-attention

$b^1, b^2, b^3, b^4$  can be parallelly computed.



# Self-attention

$$\begin{matrix} q^1 & q^2 & q^3 & q^4 \\ \hline Q \end{matrix} = \begin{matrix} W^q & \\ \hline \end{matrix} \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ \hline I \end{matrix}$$

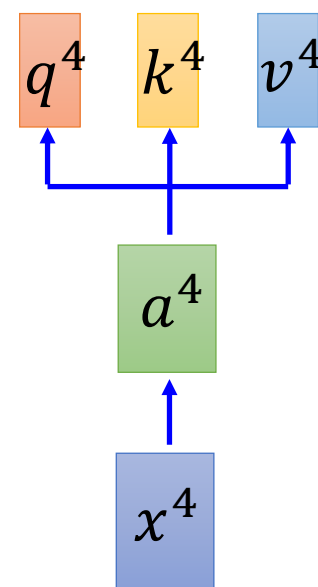
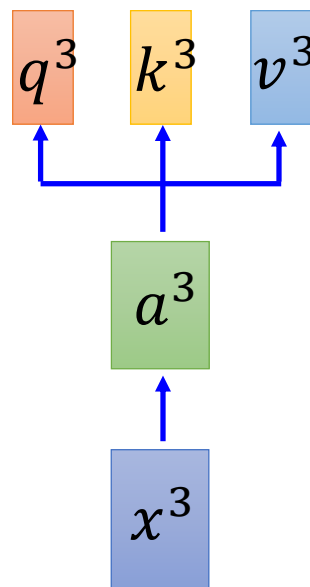
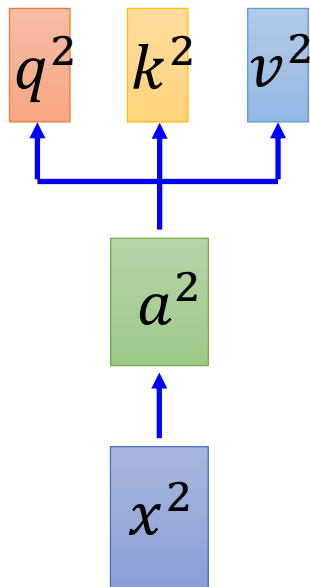
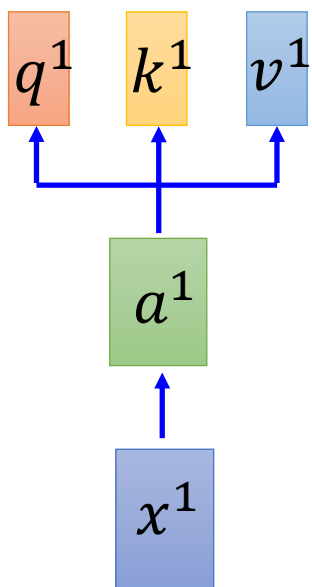
$$\begin{matrix} k^1 & k^2 & k^3 & k^4 \\ \hline K \end{matrix} = \begin{matrix} W^k & \\ \hline \end{matrix} \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ \hline I \end{matrix}$$

$$\begin{matrix} v^1 & v^2 & v^3 & v^4 \\ \hline V \end{matrix} = \begin{matrix} W^v & \\ \hline \end{matrix} \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ \hline I \end{matrix}$$

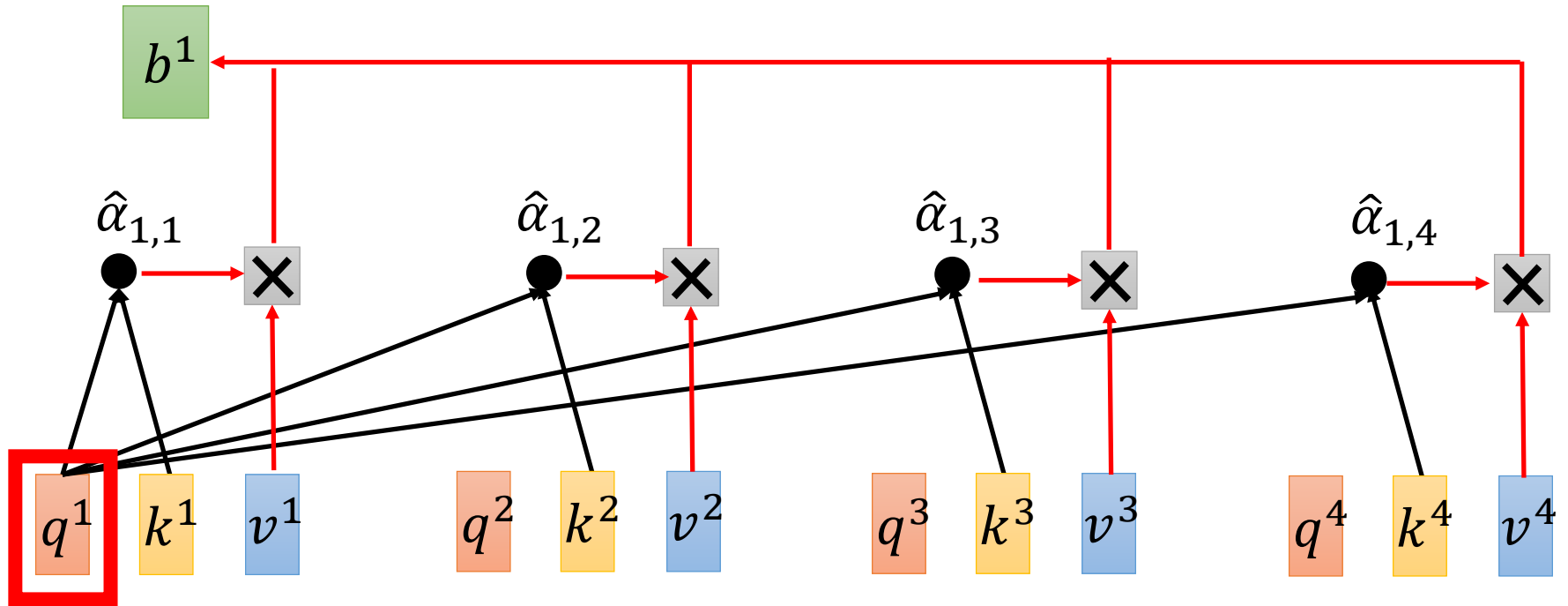
$$q^i = W^q a^i$$

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

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



# Self-attention



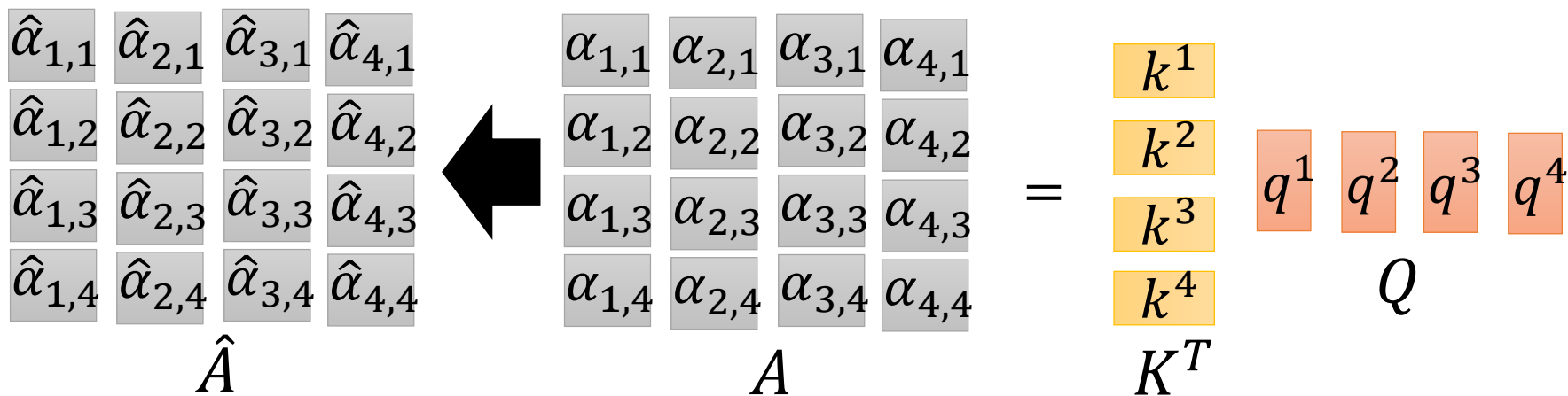
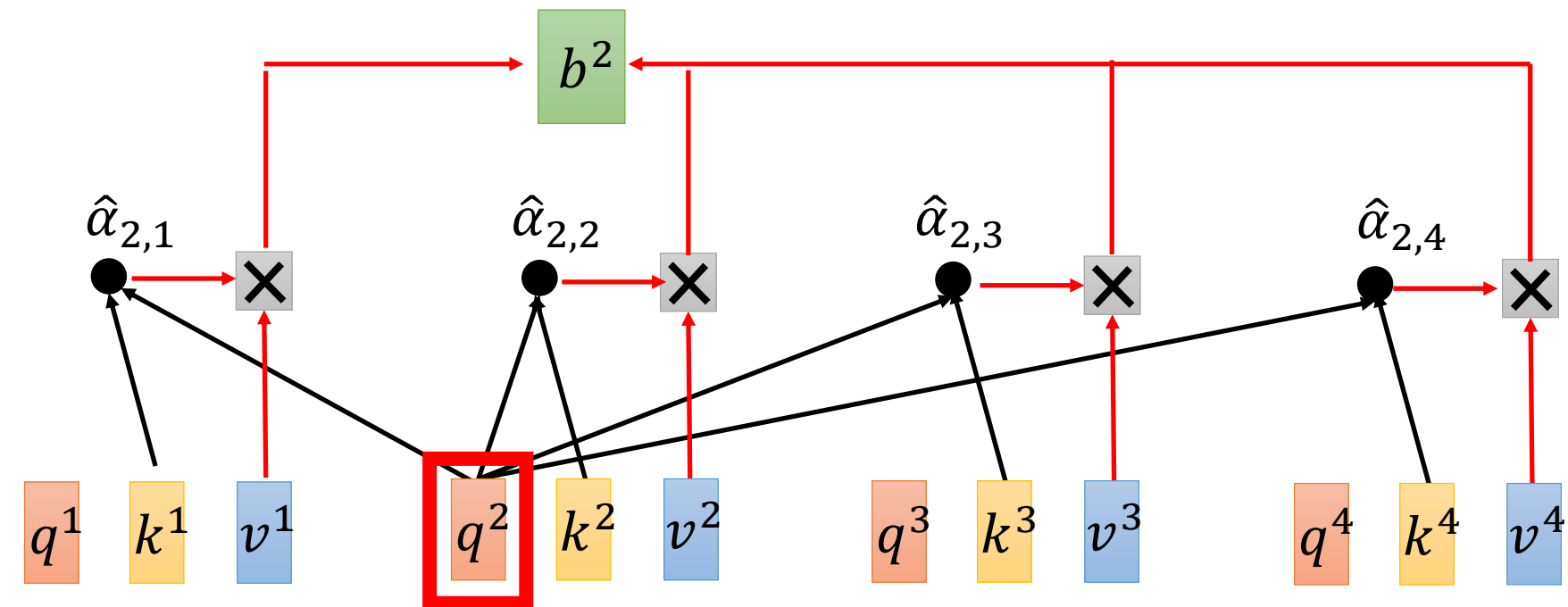
$$\alpha_{1,1} = k^1 q^1 \quad \alpha_{1,2} = k^2 q^1$$
$$\alpha_{1,3} = k^3 q^1 \quad \alpha_{1,4} = k^4 q^1$$

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

(ignore  $\sqrt{d}$  for simplicity)

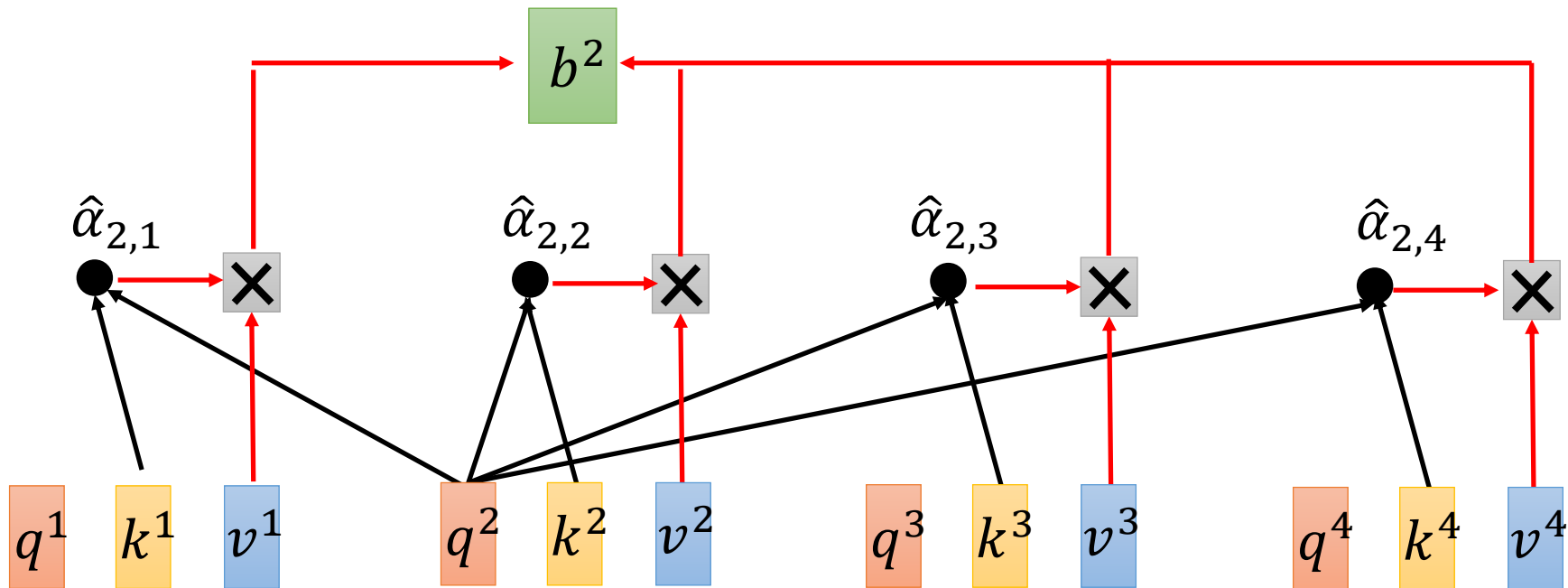
# Self-attention

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



# Self-attention

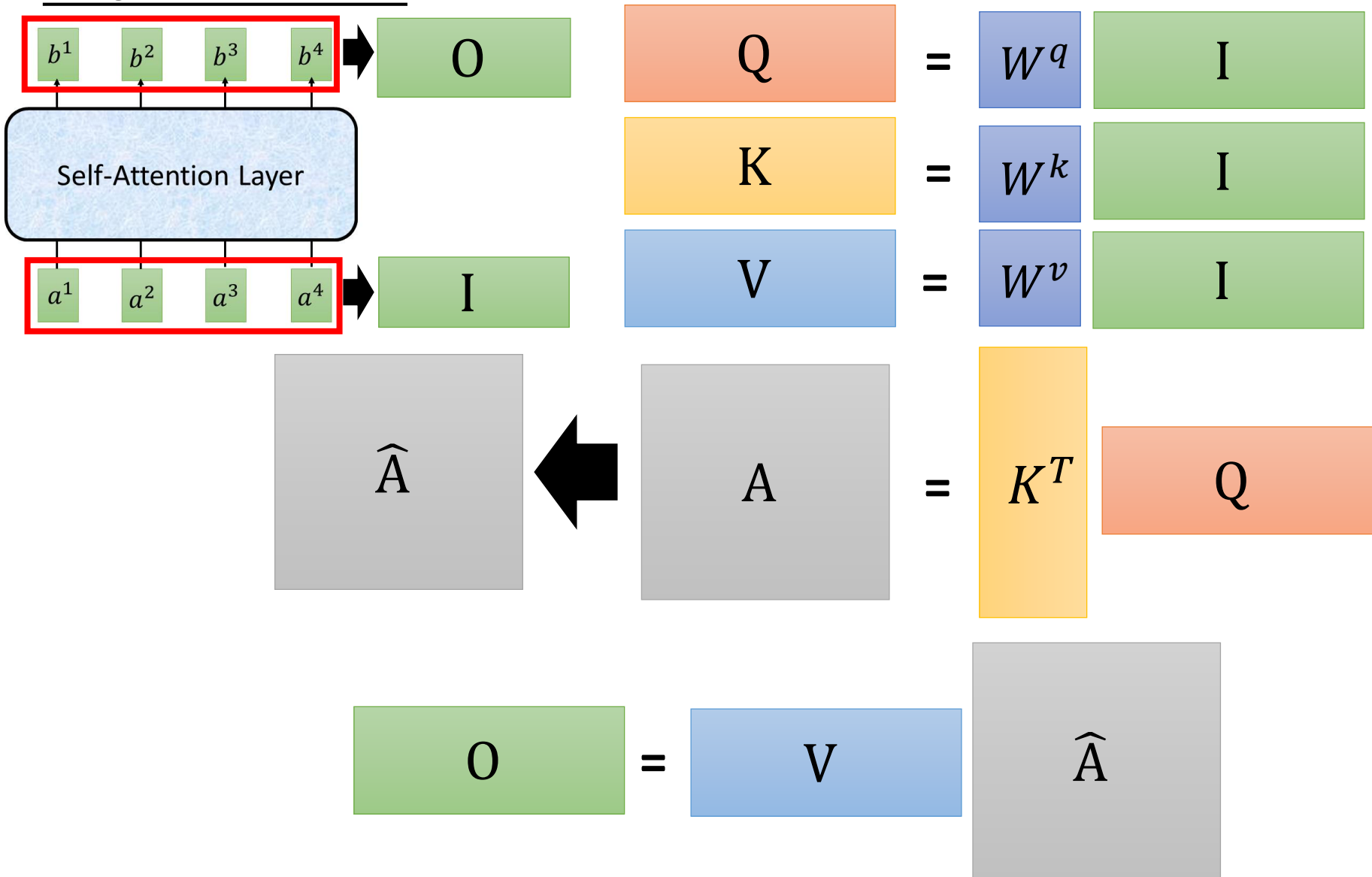
$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



$$\begin{matrix} b^1 & b^2 & b^3 & b^4 \\ \hline \end{matrix} \quad 0 = \begin{matrix} v^1 & v^2 & v^3 & v^4 \\ \hline \end{matrix} \quad \hat{A} \begin{matrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{matrix}$$

The diagram shows the calculation of  $b^2$  as a dot product of the  $v^2$  row from the matrix  $V$  and the  $\hat{\alpha}_{2,i}$  column from the matrix  $\hat{A}$ . The  $\hat{\alpha}_{2,i}$  elements are highlighted with red boxes in the matrix  $\hat{A}$ .

# Self-attention



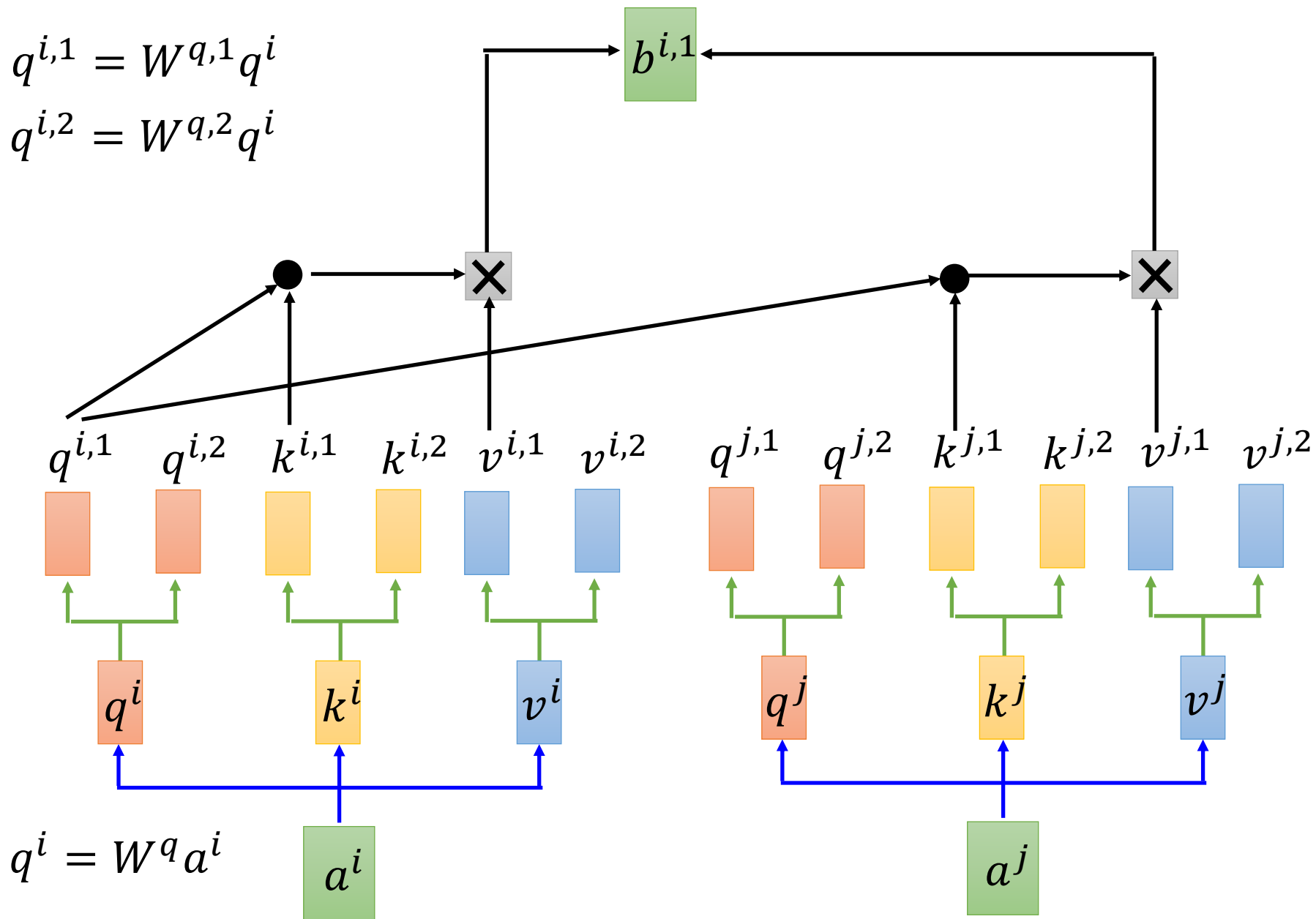
反正就是一堆矩阵乘法，用 GPU 可以加速

# Multi-head Self-attention

(2 heads as example)

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

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

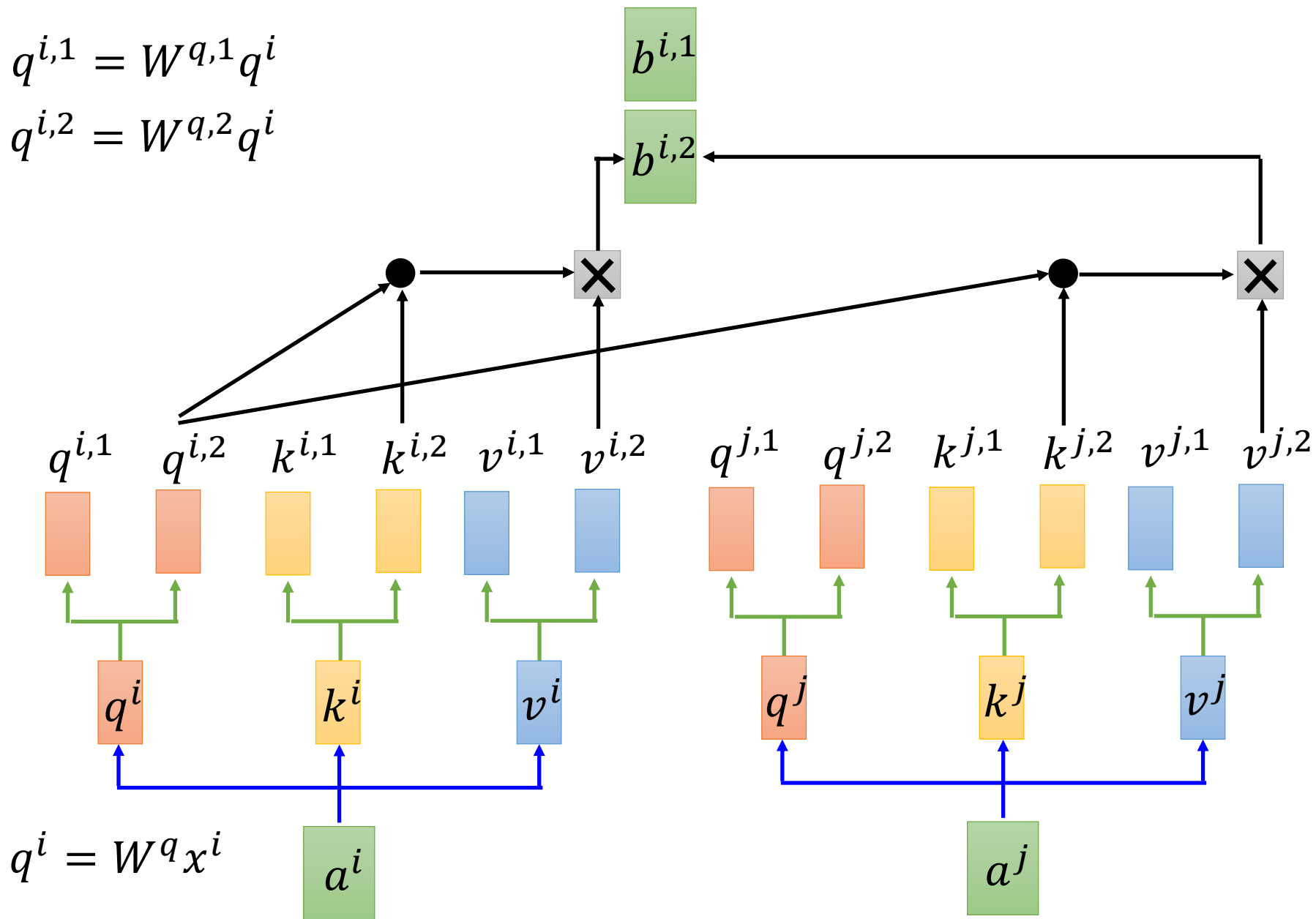


# Multi-head Self-attention

(2 heads as example)

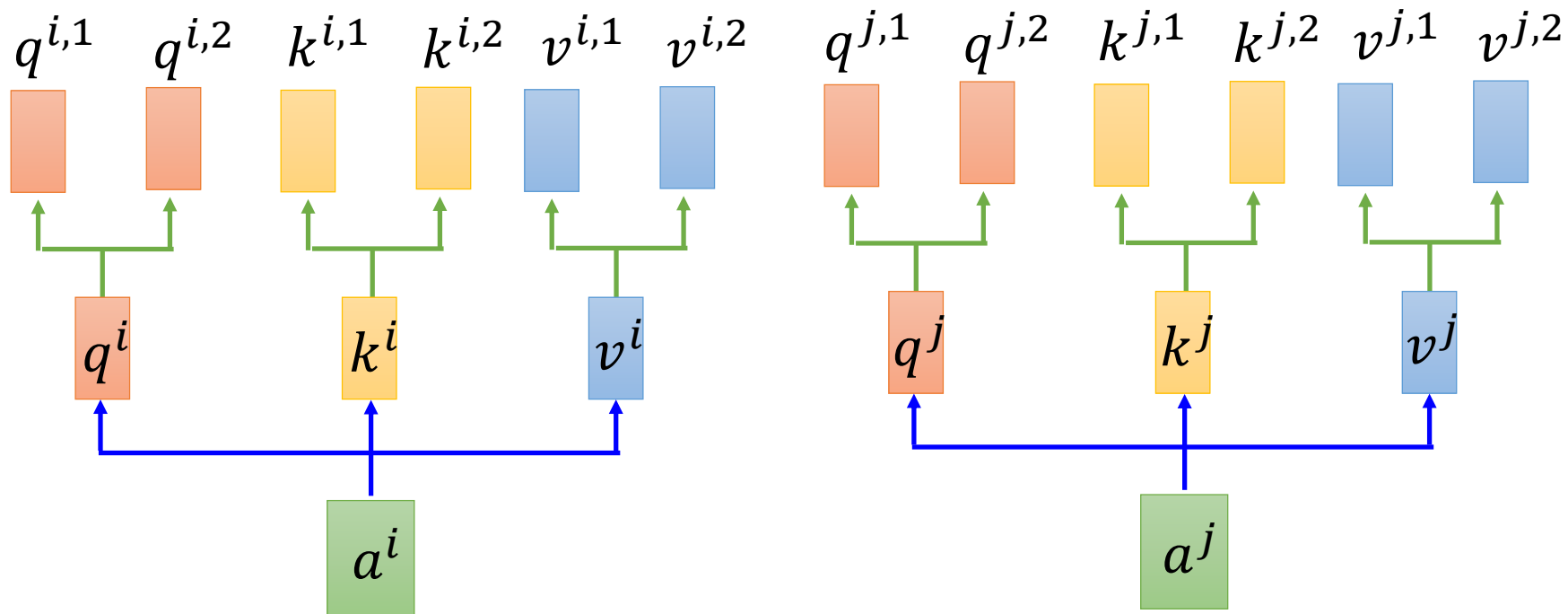
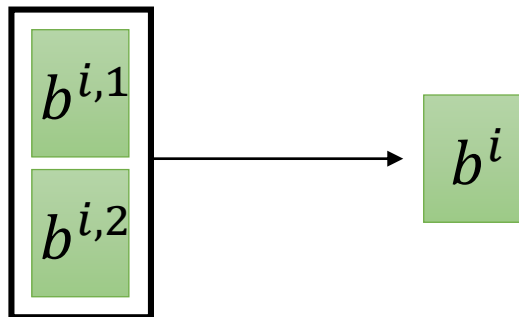
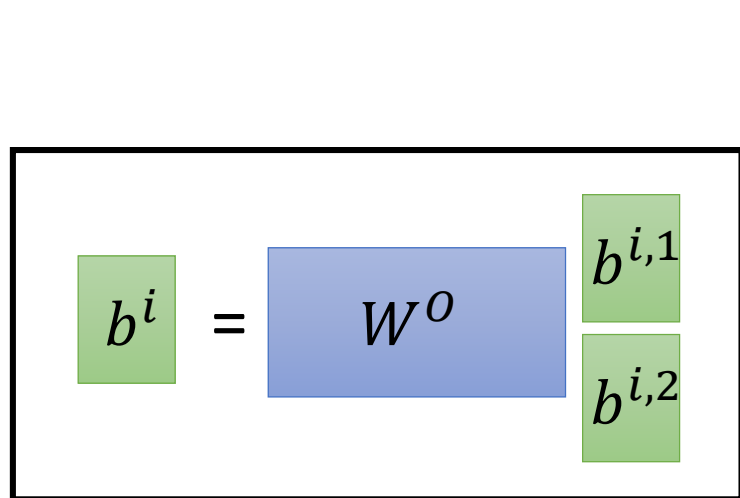
$$q^{i,1} = W^{q,1} q^i$$

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



# Multi-head Self-attention

(2 heads as example)

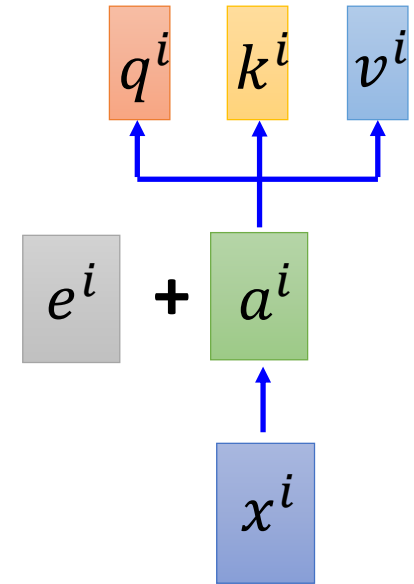


# Positional Encoding

- No position information in self-attention.
- Original paper: each position has a unique positional vector  $e^i$  (not learned from data)

$$\vec{p}_t^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$



$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1}$$

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

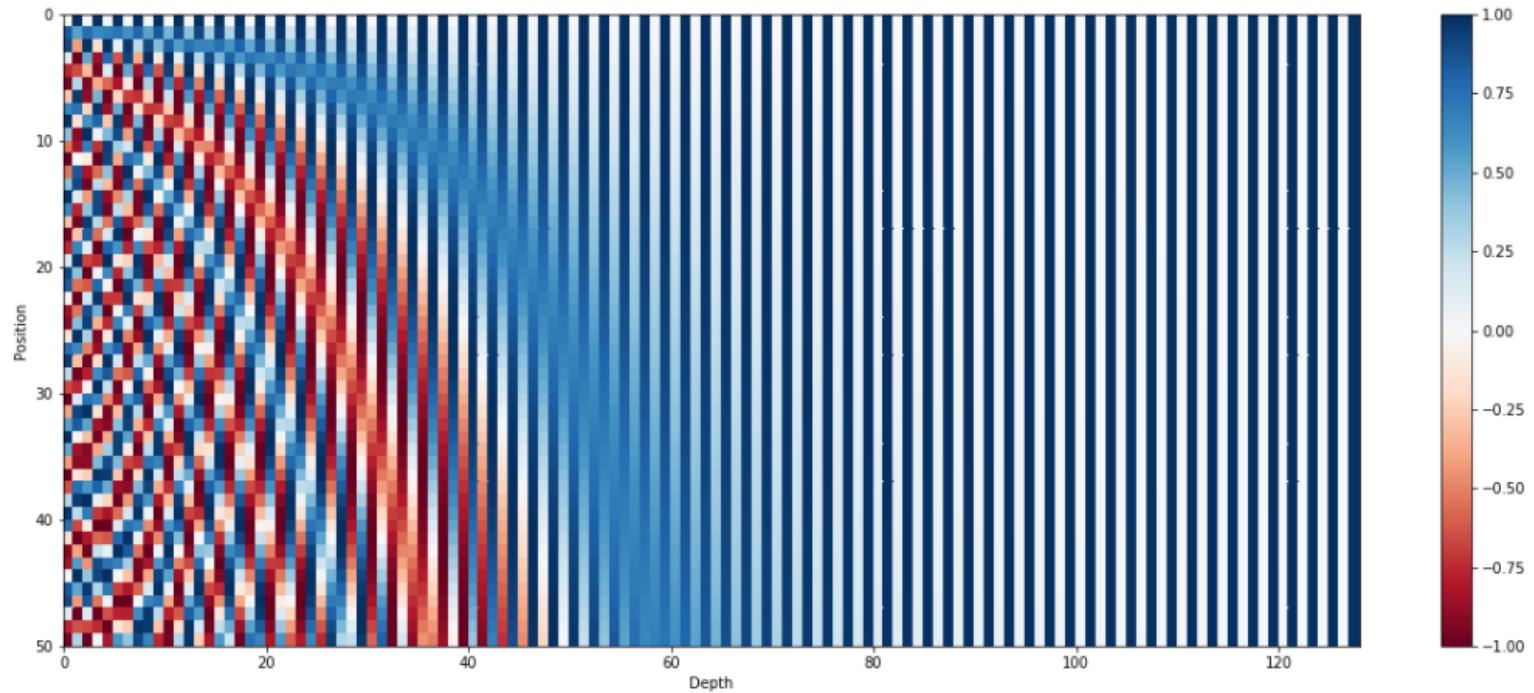
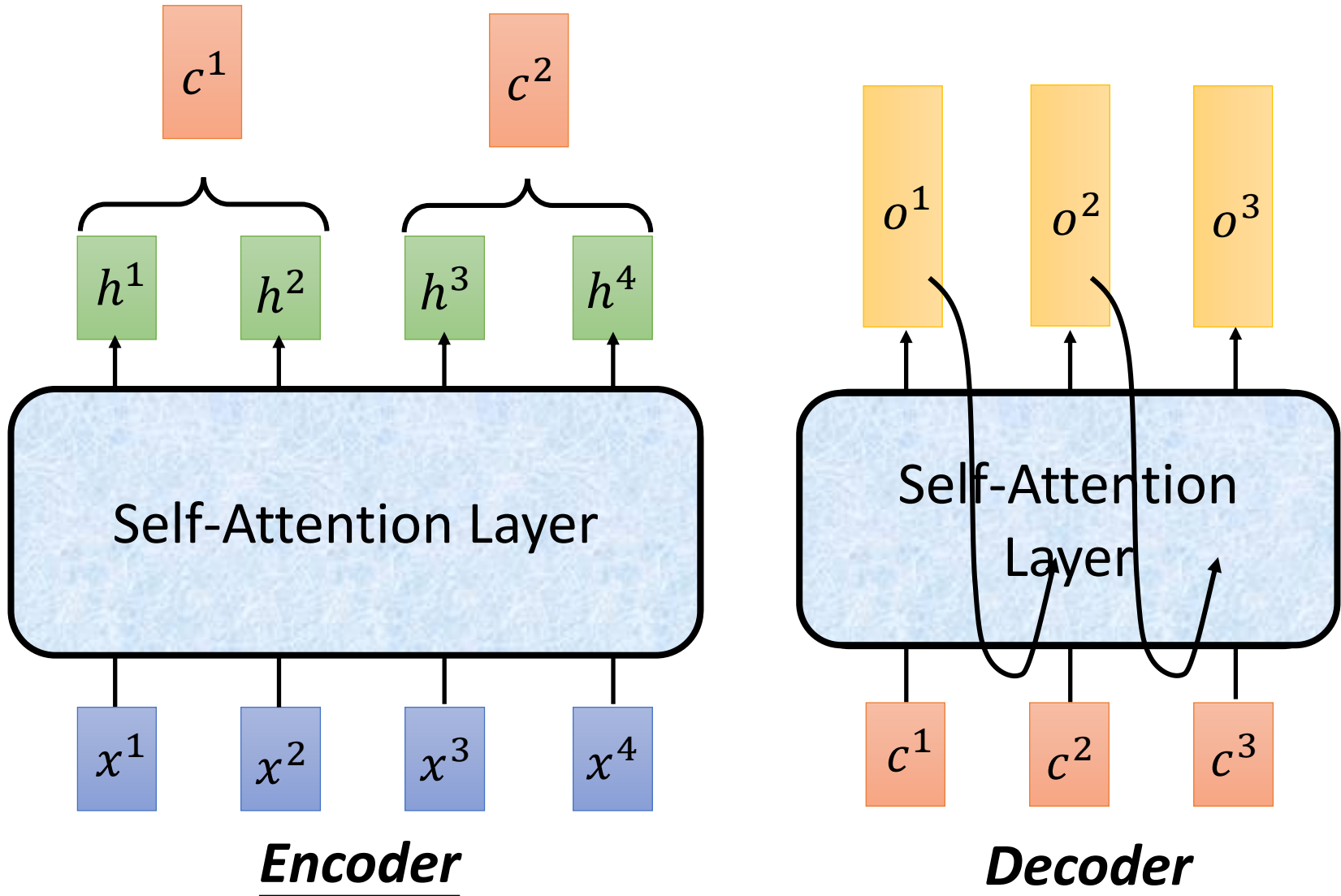


Figure 2 - The 128-dimensional positional encoding for a sentence with the maximum length of 50.

Each row represents the embedding vector  $\vec{p}_t$

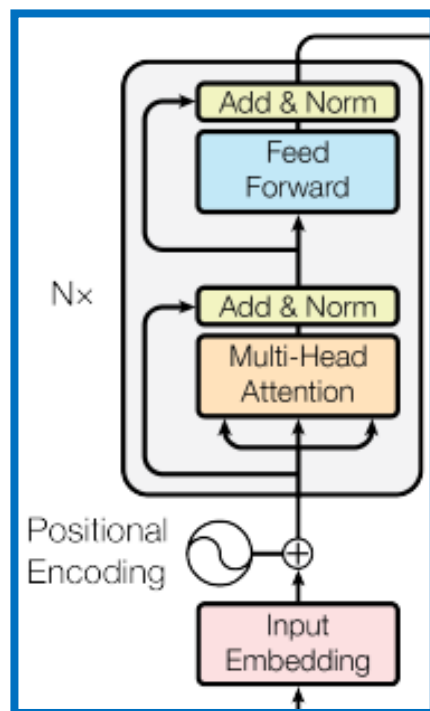
# Seq2seq with Attention



# Transformer

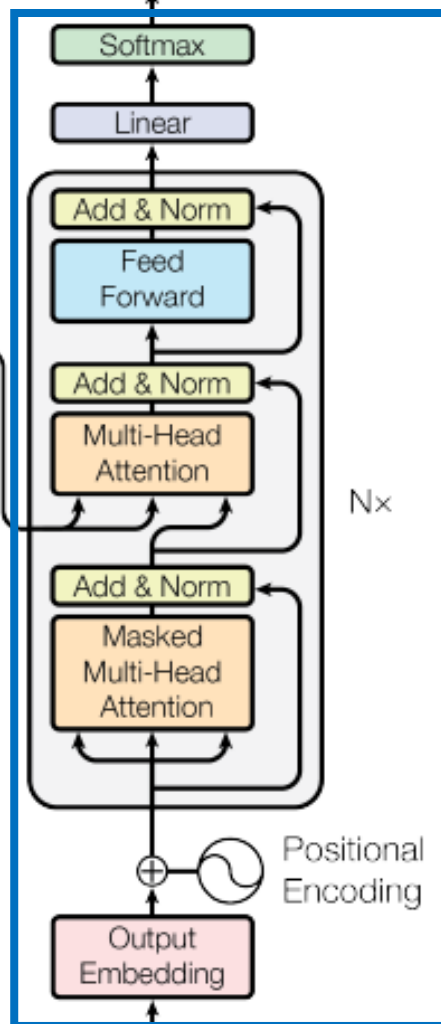
Using Chinese to English translation as example

Encoder



机器学习

machine learning  
Output Probabilities



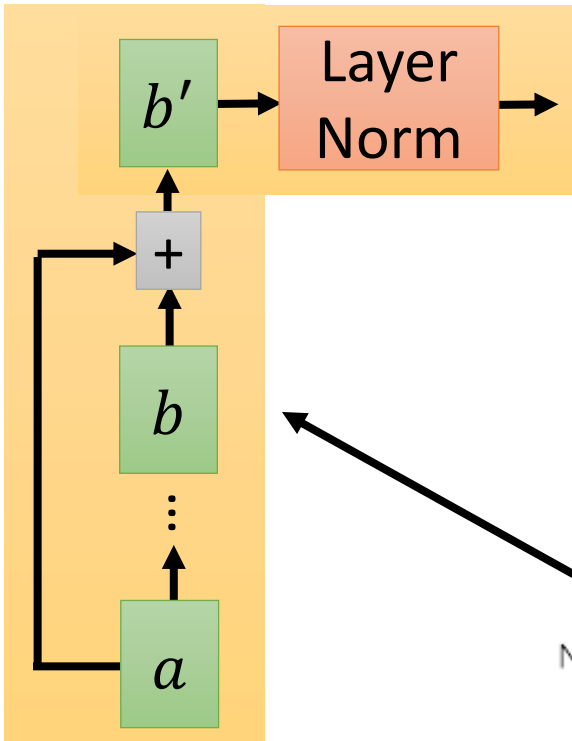
Decoder

Outputs  
(shifted right)

<BOS>

machine

# Transformer

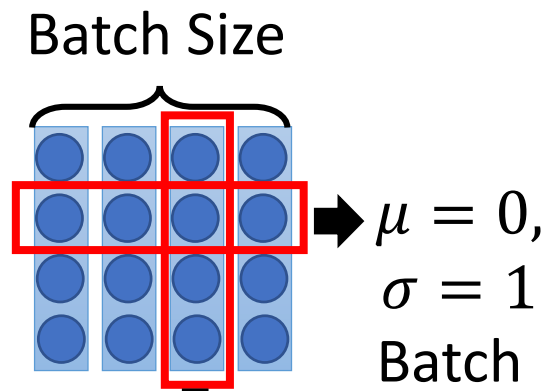
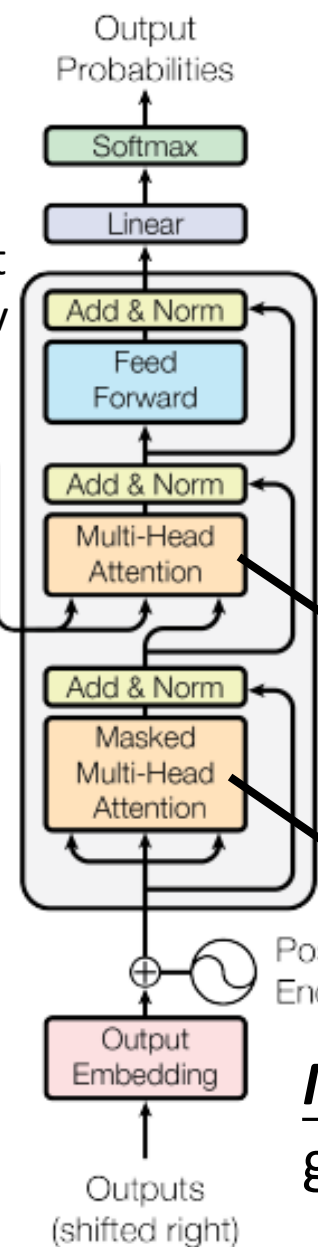
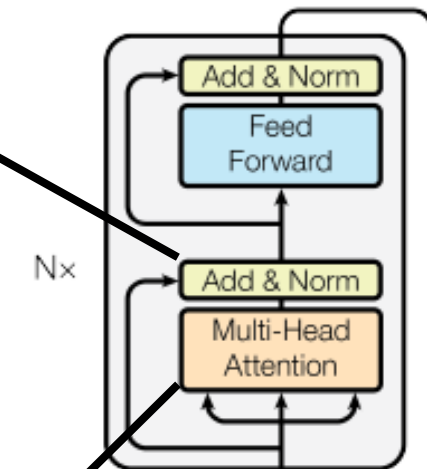
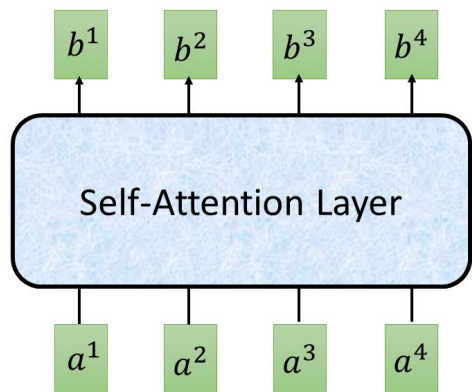


Layer Norm:

<https://arxiv.org/abs/1607.06450>

Batch Norm:

<https://www.youtube.com/watch?v=BZh1ltr5Rkg>



$\mu = 0, \sigma = 1$   
Layer

attend on the  
input sequence

**Masked**: attend on the  
generated sequence

Positional Encoding

Positional Encoding

Input Embedding

Output Embedding

Inputs

Outputs  
(shifted right)

Output  
Probabilities

Softmax

Linear

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

Add & Norm

Masked Multi-Head Attention

Add & Norm

Multi-Head Attention

Add & Norm

Feed Forward

Add & Norm

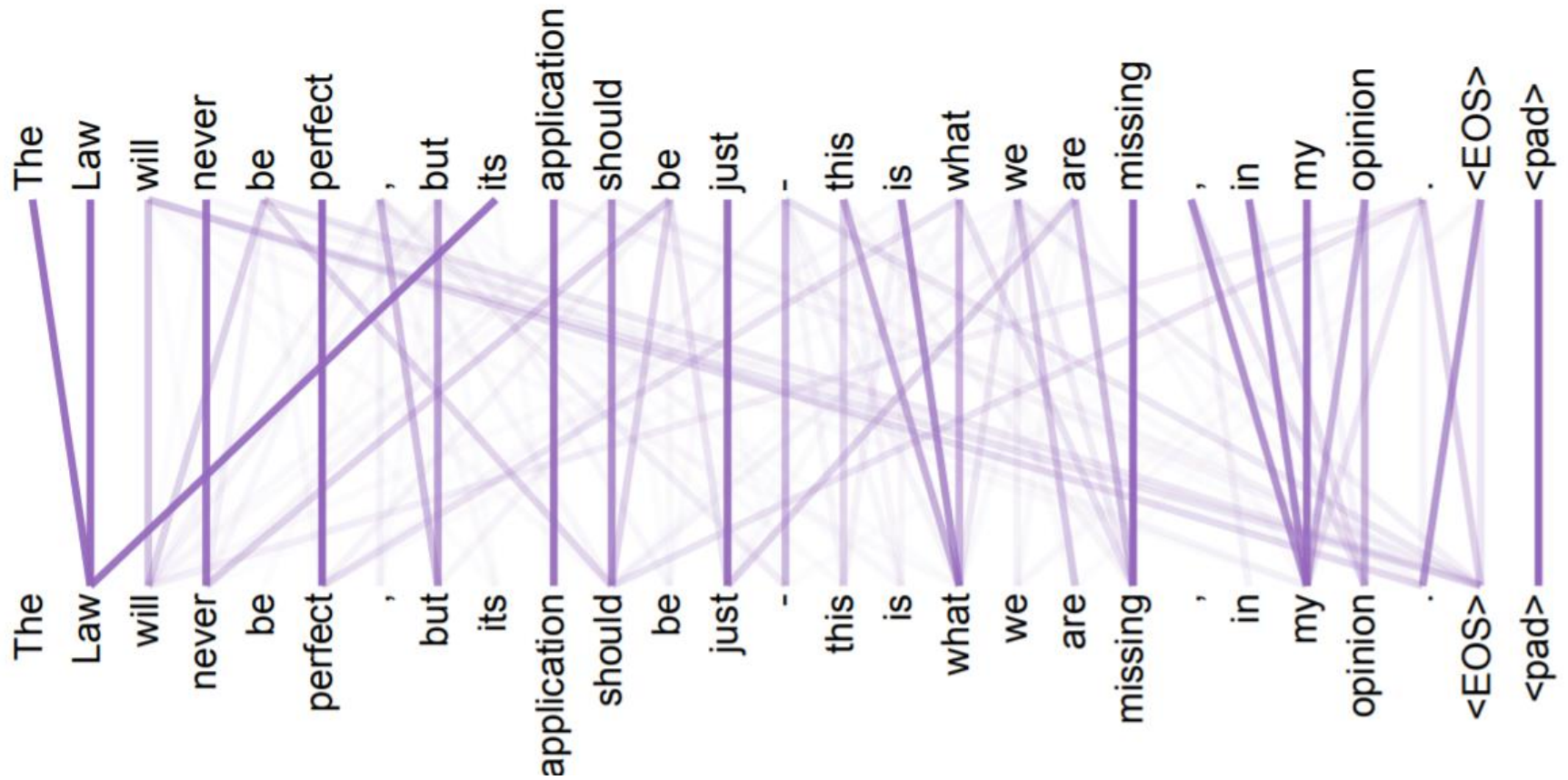
Multi-Head Attention

Add & Norm

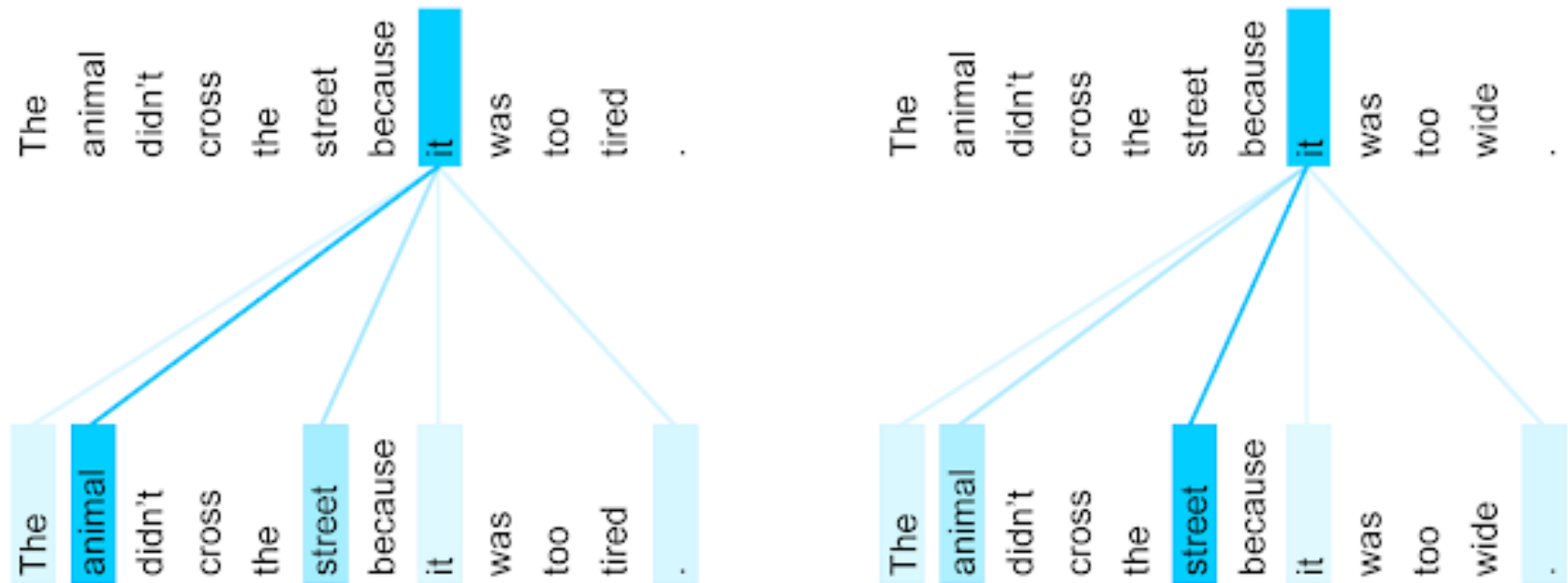
Feed Forward

Add & Norm

# Attention Visualization



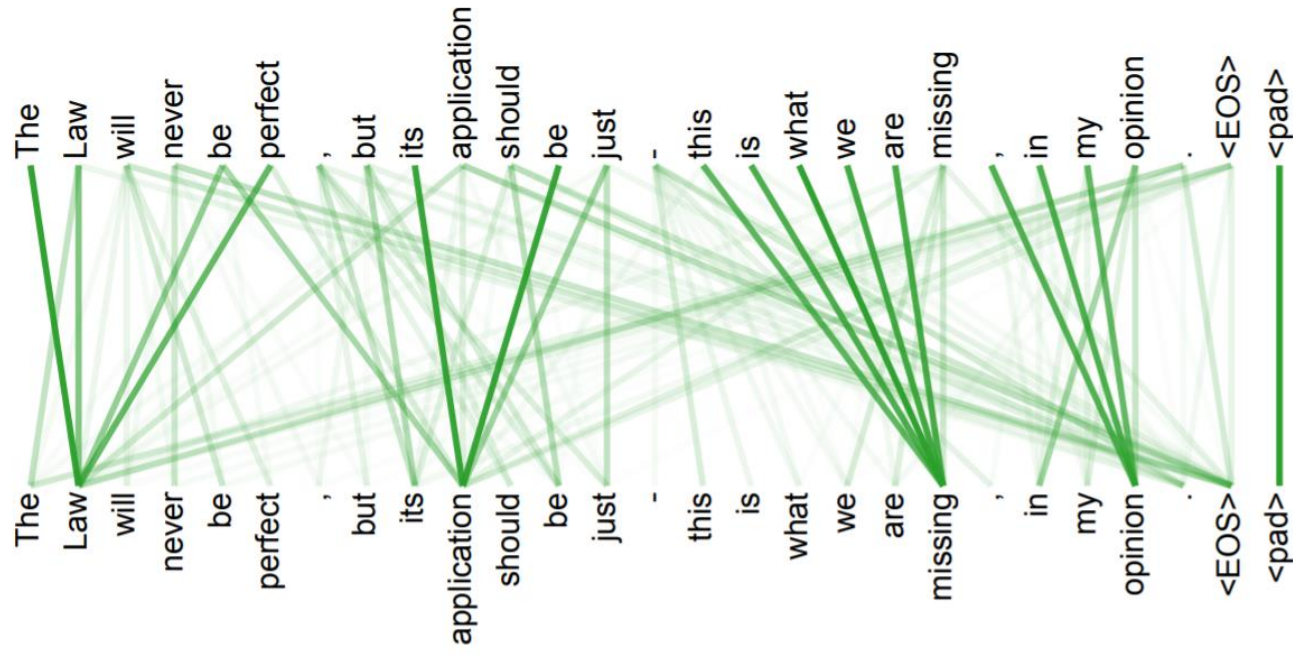
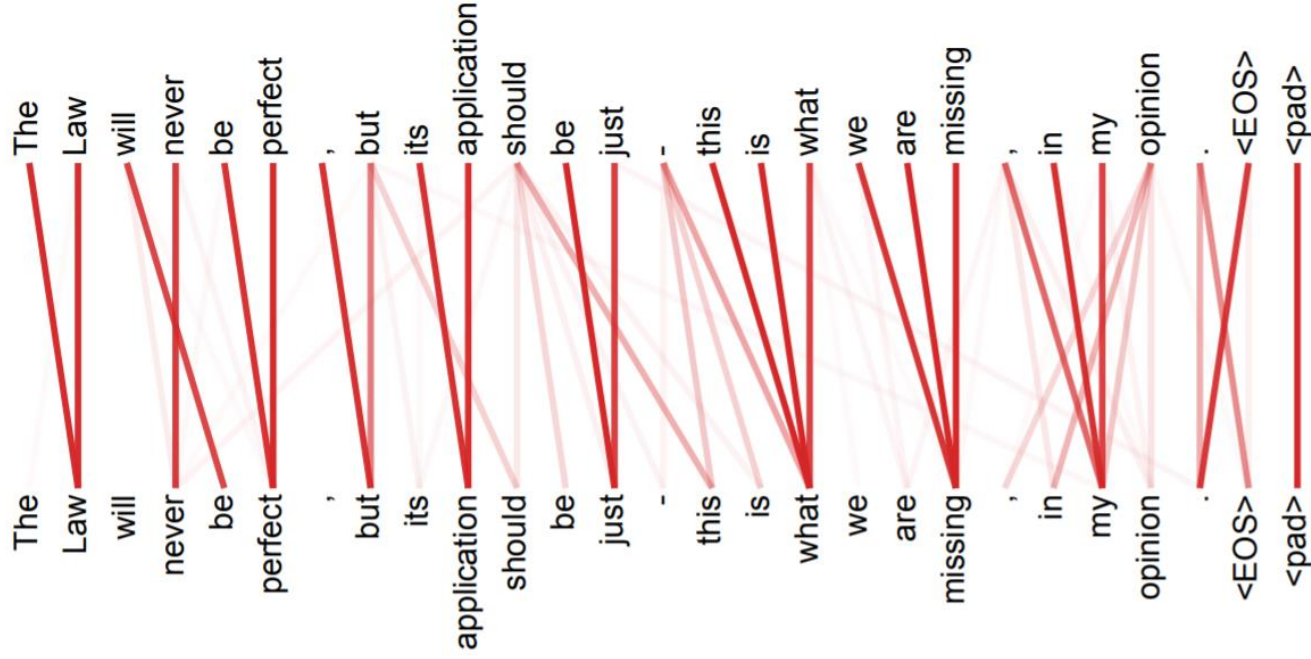
# Attention Visualization



The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads)

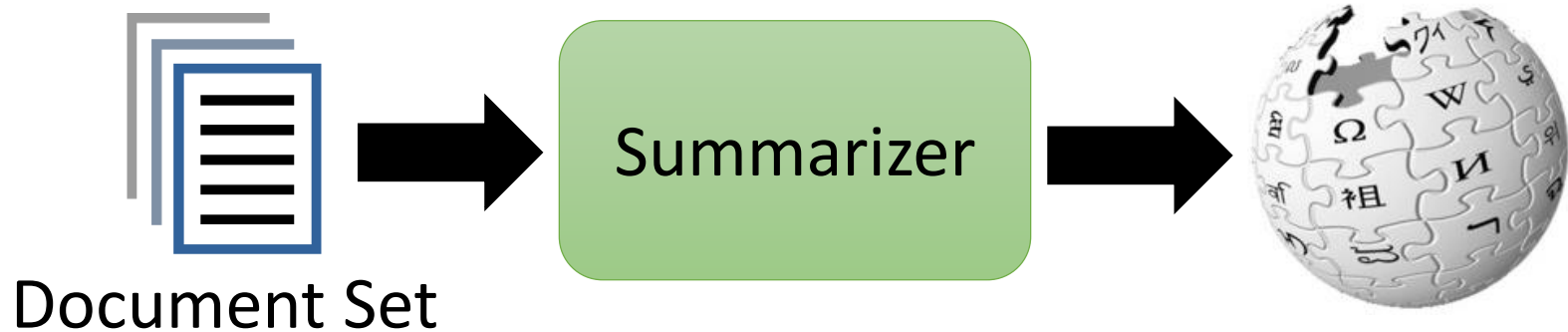
<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

# Multi-head Attention



# Example Application

- If you can use seq2seq, you can use transformer.



Dataset	Input	Output	# examples
Gigaword (Graff & Cieri, 2003)	$10^1$	$10^1$	$10^6$
CNN/DailyMail (Nallapati et al., 2016)	$10^2-10^3$	$10^1$	$10^5$
WikiSum (ours)	$10^2-10^6$	$10^1-10^3$	$10^6$