



Transformer

李宏毅

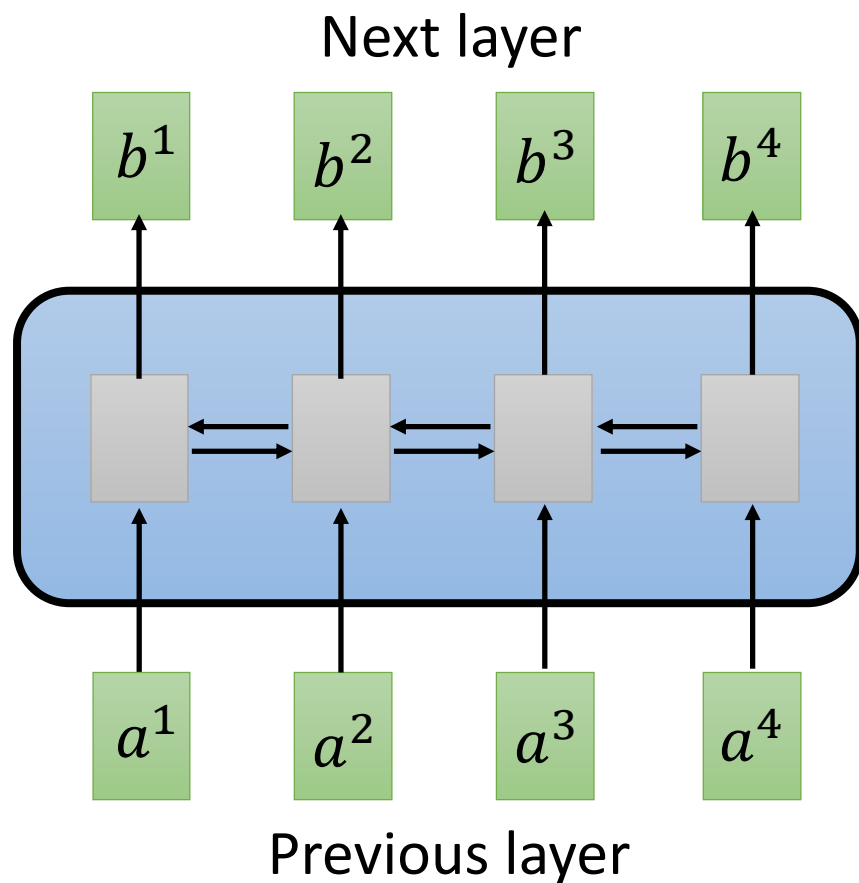
Hung-yi Lee

A detailed illustration of two Transformers characters, Optimus Prime and Bumblebee, set against a black background. Optimus Prime, on the left, is a large red and blue robot with a prominent chest screen and a blue visor. He is holding a long, grey, segmented cannon in his right hand. Bumblebee, on the right, is a smaller yellow and black robot with a red Autobot symbol on his chest. He is holding a yellow and black weapon. The text "Transformer" is overlaid in the center in a large, white, sans-serif font.

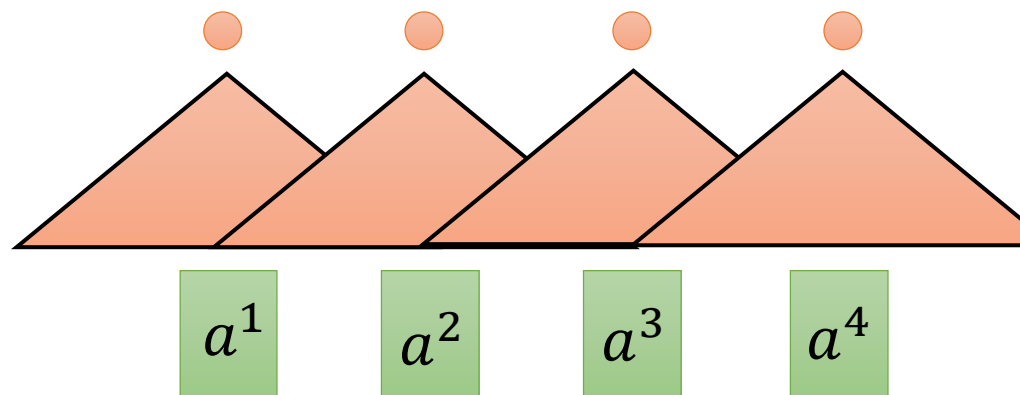
Transformer

Seq2seq model with “Self-attention”

Sequence

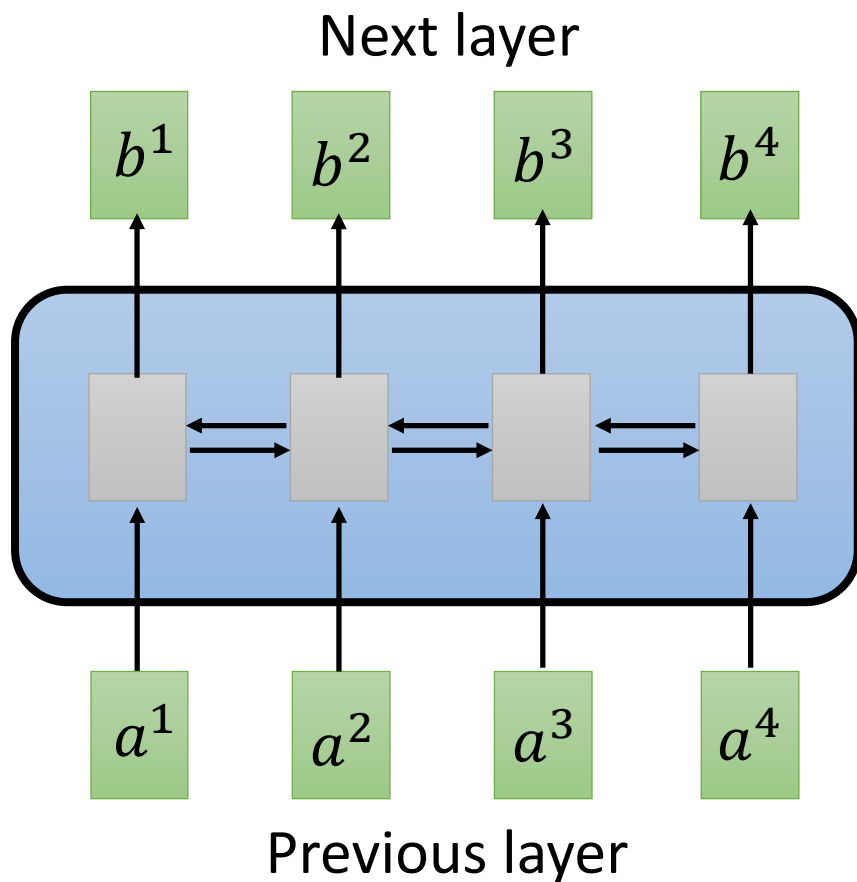


Hard to parallel !



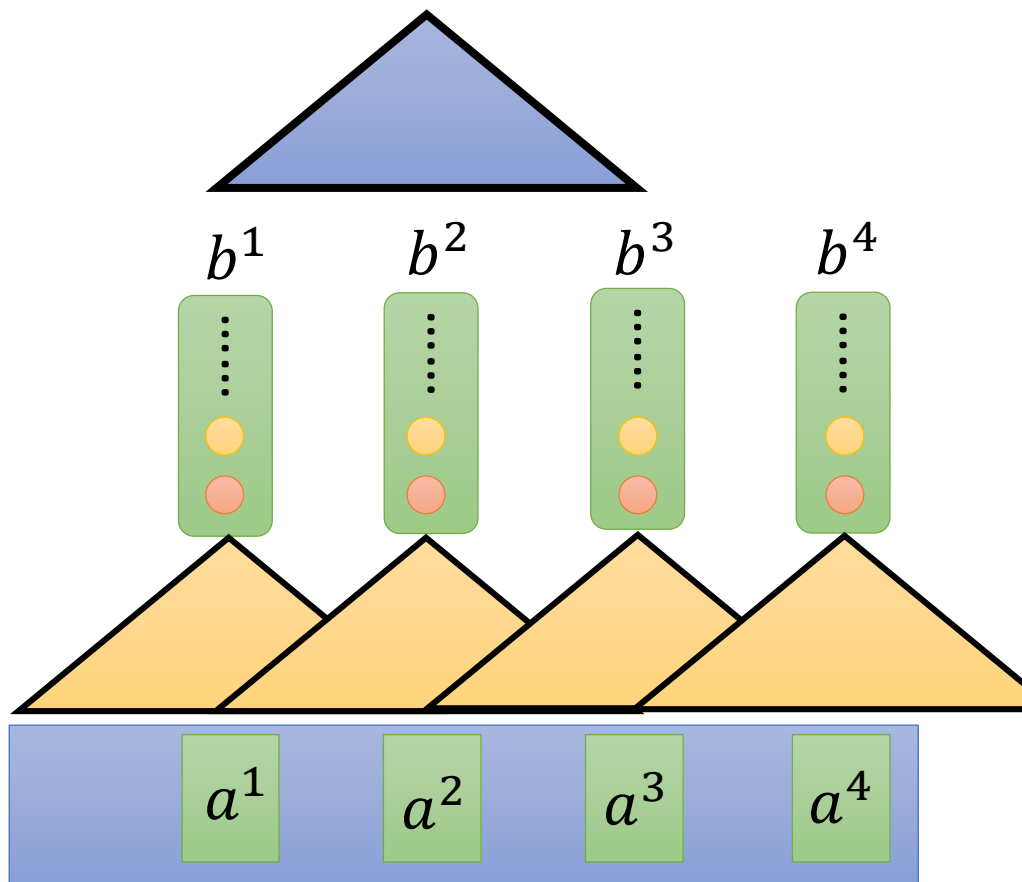
Using CNN to replace RNN

Sequence



Hard to parallel

Filters in higher layer can consider longer sequence

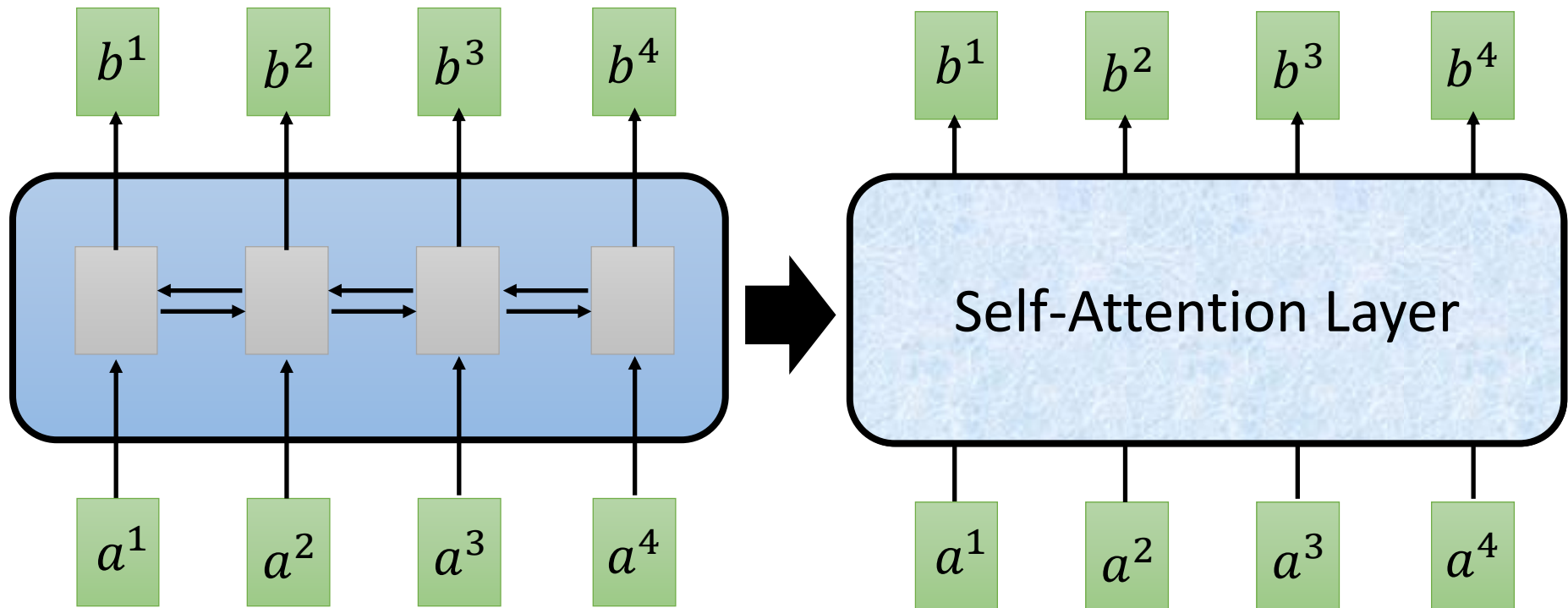


Using CNN to replace RNN
(CNN can parallel)

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)

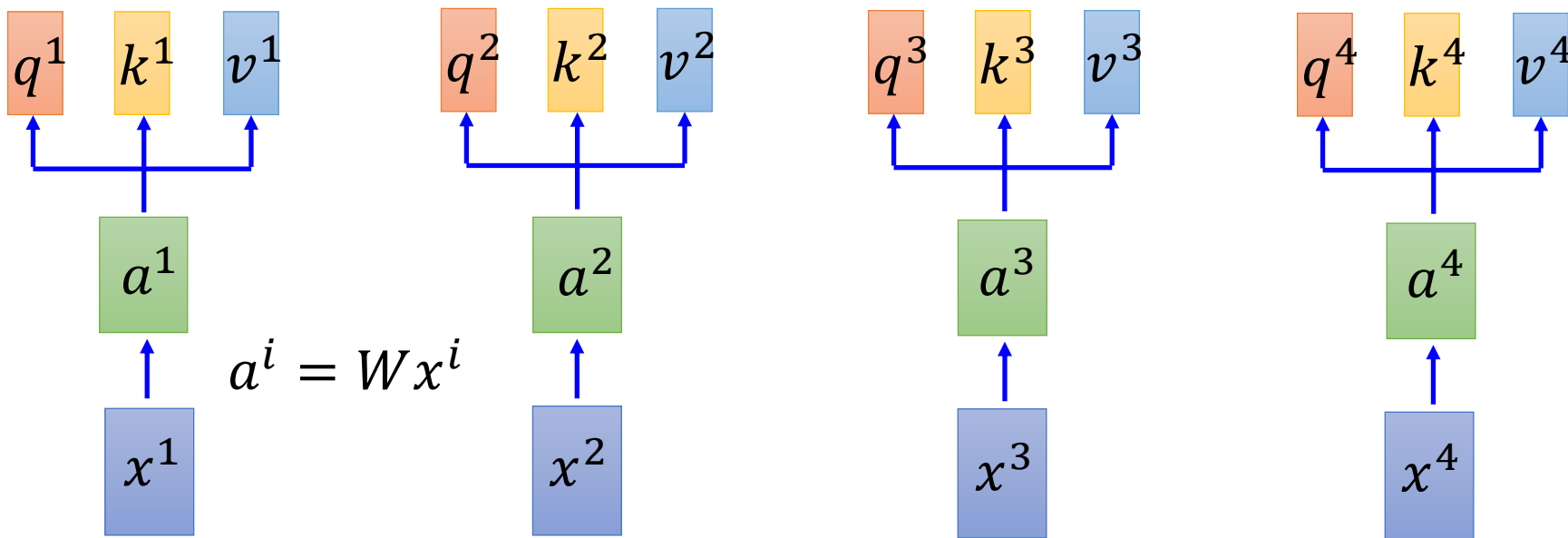
$$q^i = W^q a^i$$

k : key (to be matched)

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

v : information to be extracted

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



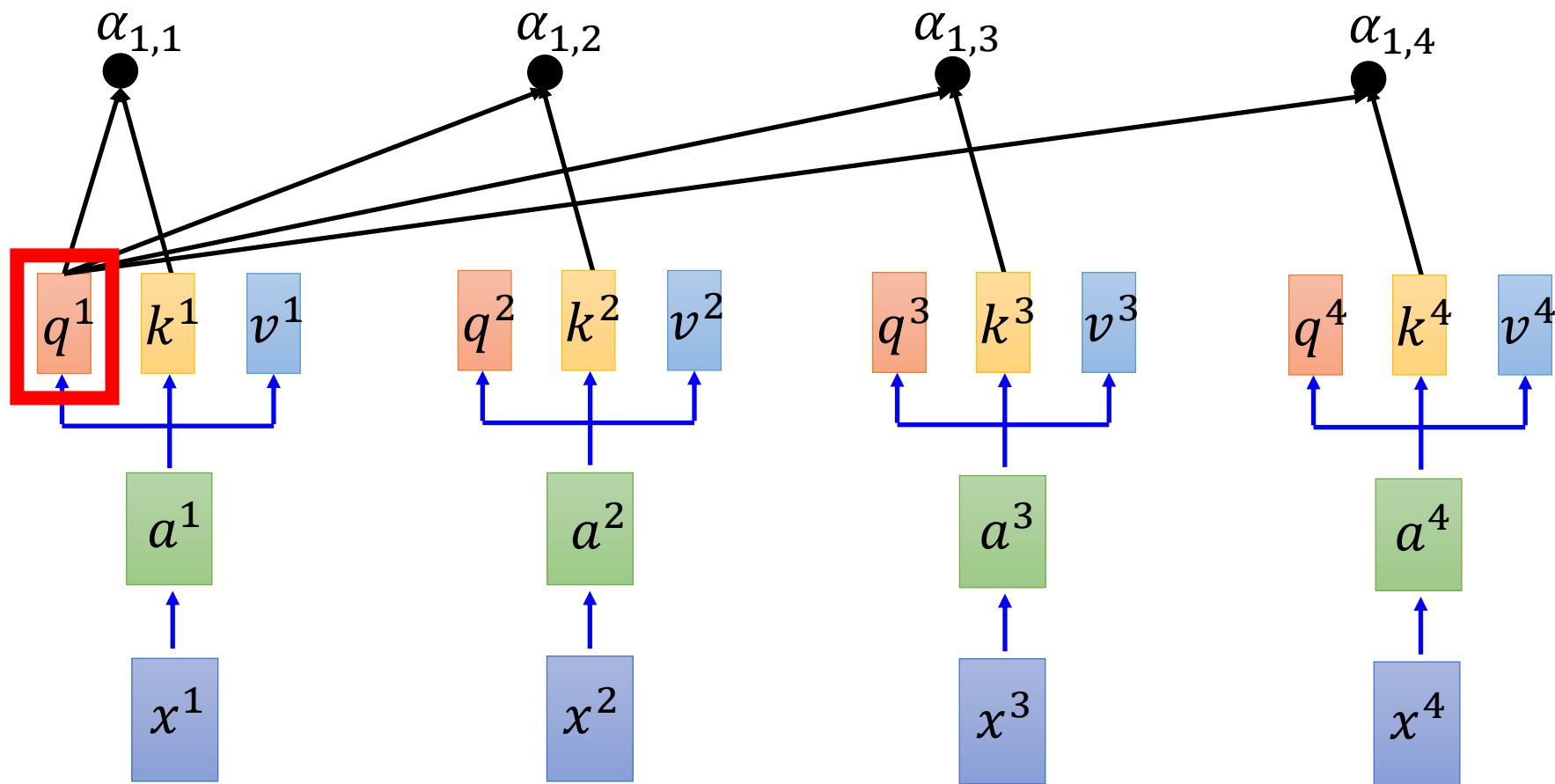
Self-attention

拿每個 query q 去對每個 key k 做 attention

d is the dim of q and k

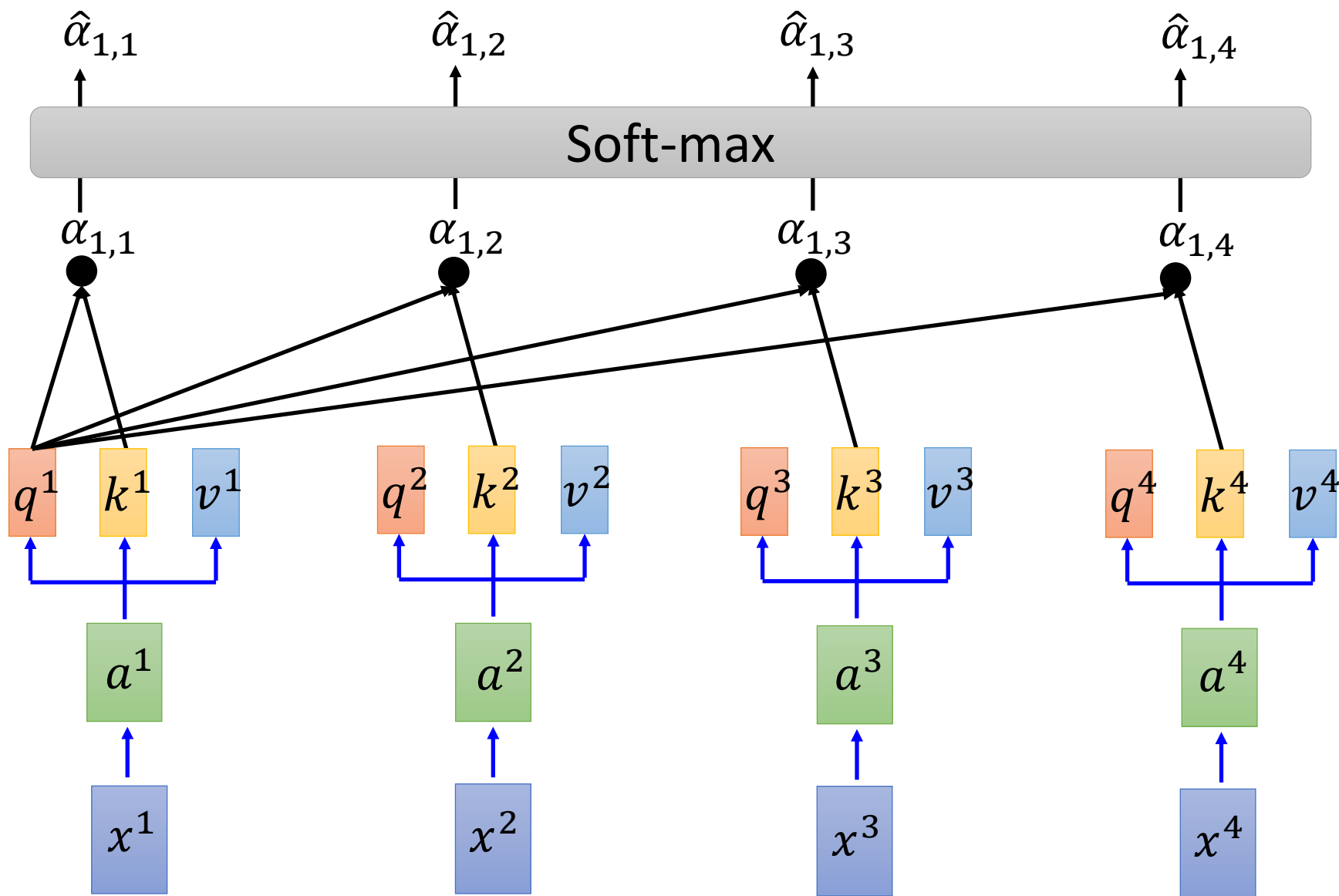
Scaled Dot-Product Attention: $\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\text{dot product}} / \sqrt{d}$

dot product



Self-attention

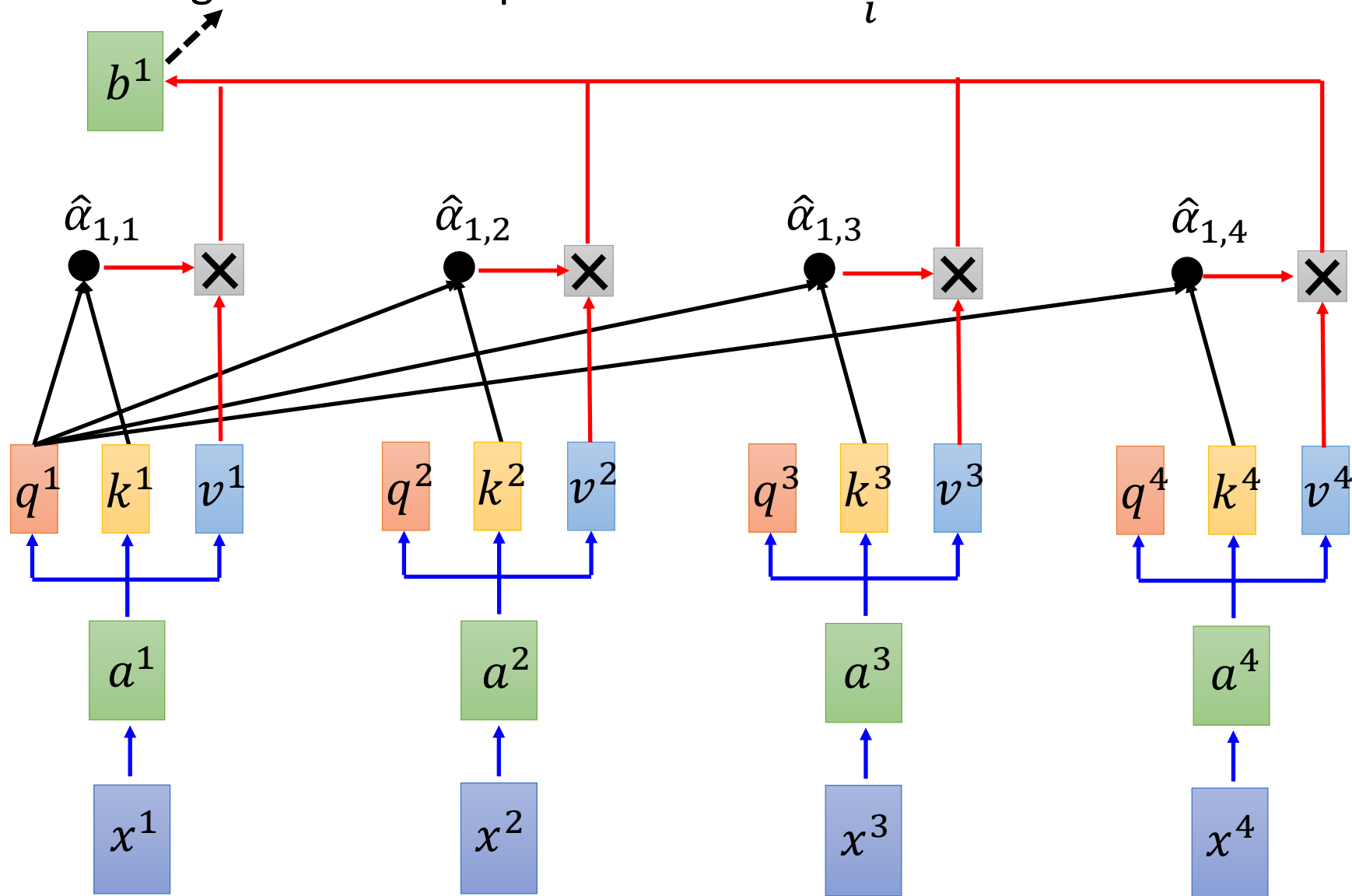
$$\hat{\alpha}_{1,i} = \exp(\alpha_{1,i}) / \sum_j \exp(\alpha_{1,j})$$



Self-attention

Considering the whole sequence

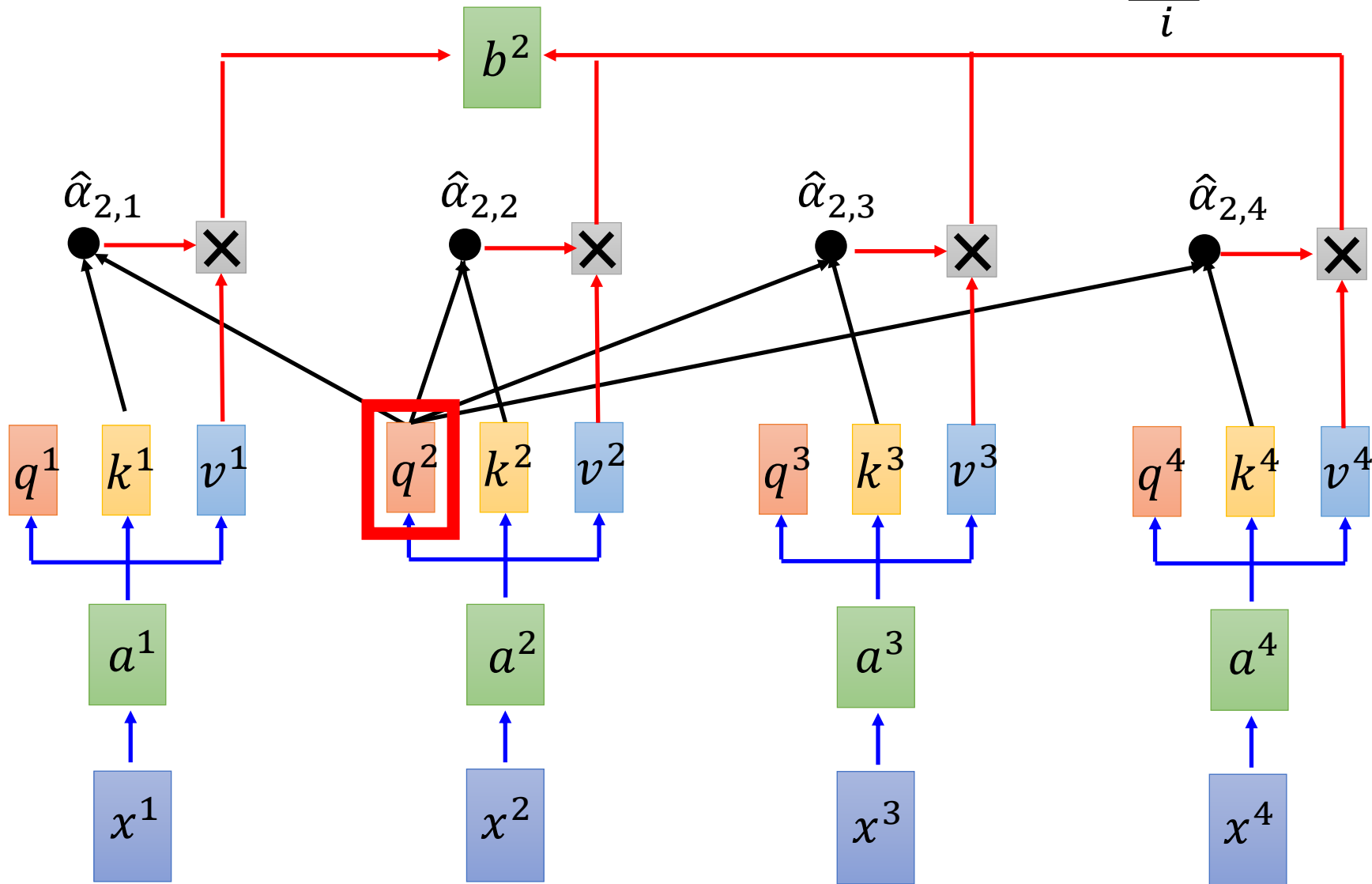
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$



Self-attention

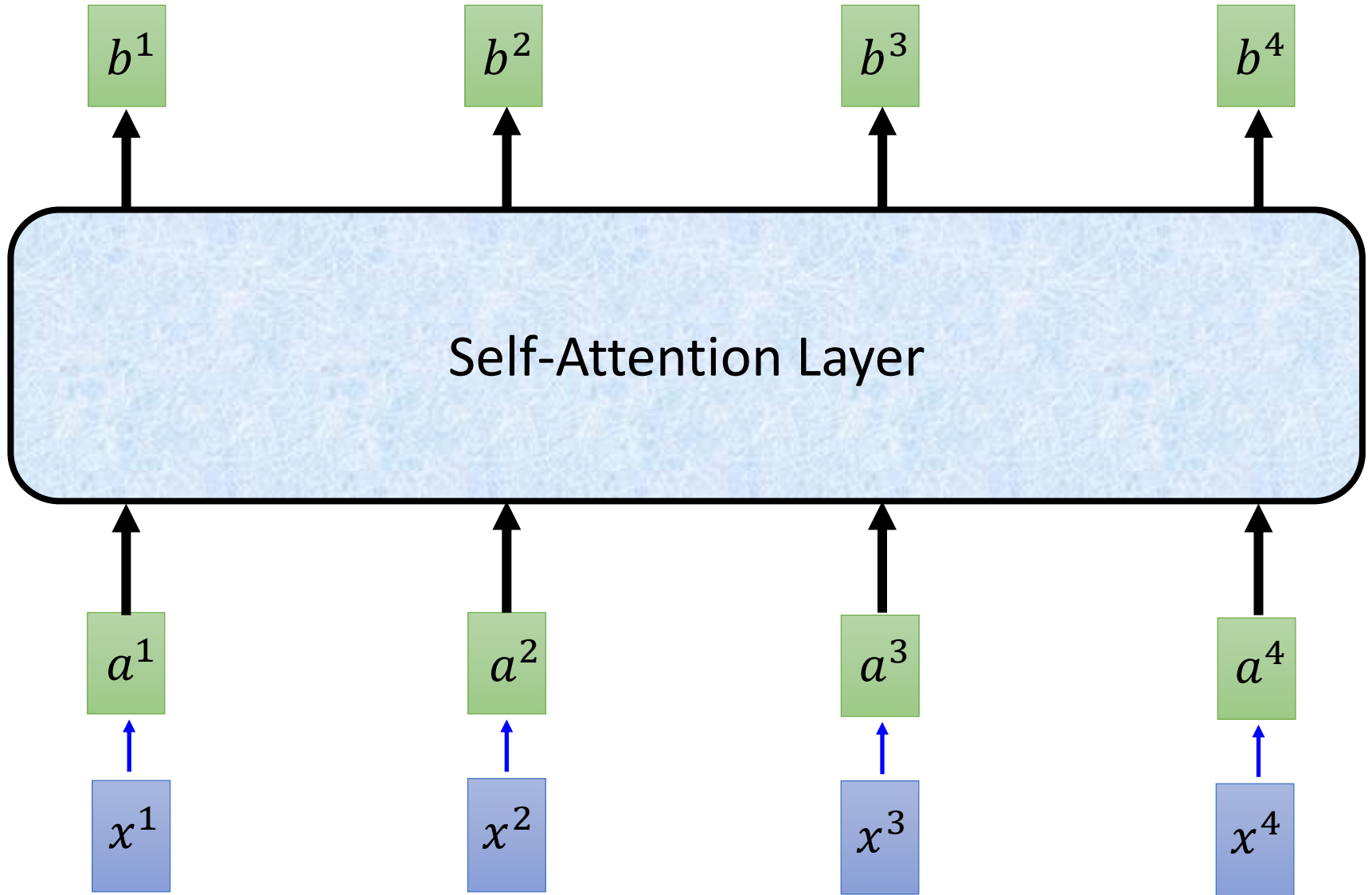
拿每個 query q 去對每個 key k 做 attention

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



Self-attention

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



Self-attention

$$\begin{array}{c} q^1 \ q^2 \ q^3 \ q^4 \\ Q \end{array} = \begin{array}{c} W^q \\ \text{I} \end{array} \begin{array}{c} a^1 \ a^2 \ a^3 \ a^4 \\ \text{I} \end{array}$$

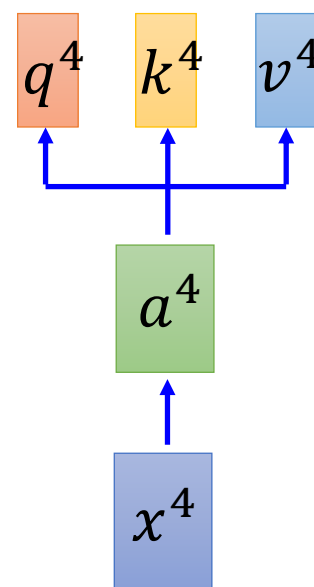
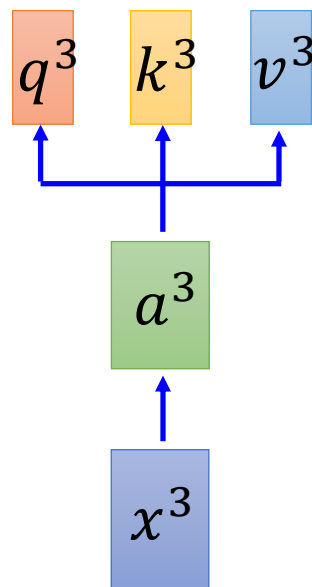
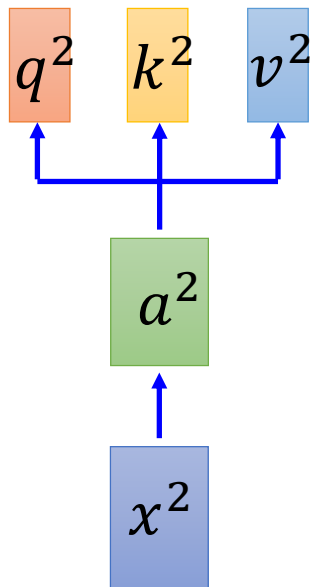
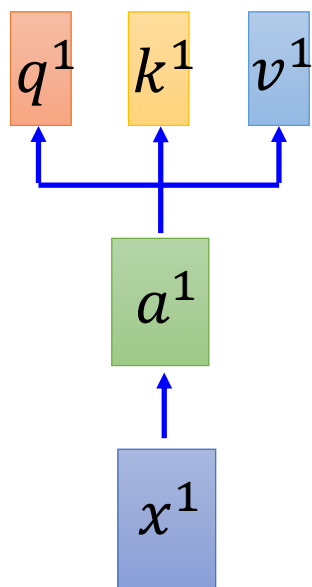
$$\begin{array}{c} k^1 \ k^2 \ k^3 \ k^4 \\ K \end{array} = \begin{array}{c} W^k \\ \text{I} \end{array} \begin{array}{c} a^1 \ a^2 \ a^3 \ a^4 \\ \text{I} \end{array}$$

$$\begin{array}{c} v^1 \ v^2 \ v^3 \ v^4 \\ V \end{array} = \begin{array}{c} W^v \\ \text{I} \end{array} \begin{array}{c} a^1 \ a^2 \ a^3 \ a^4 \\ \text{I} \end{array}$$

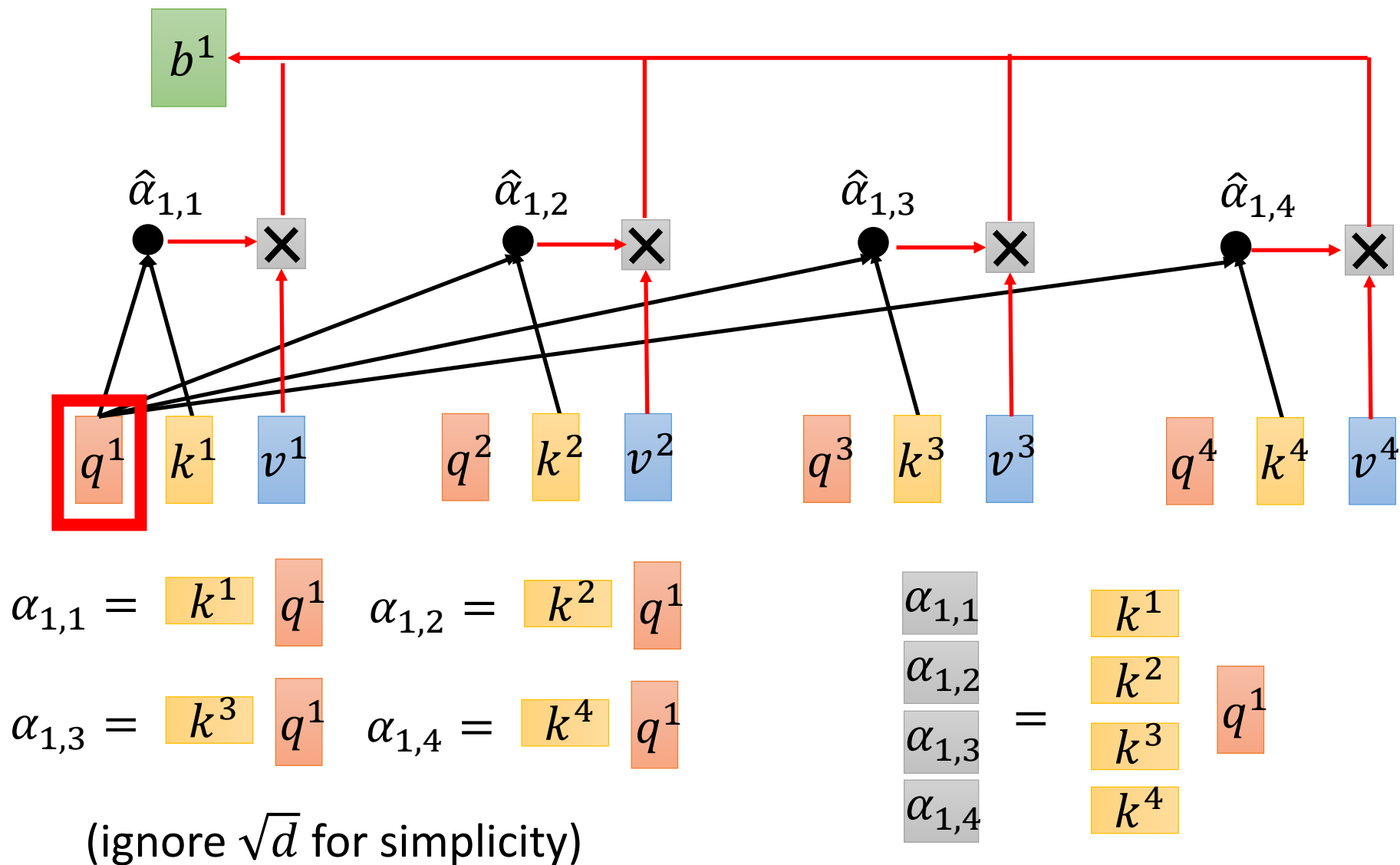
$$q^i = W^q a^i$$

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

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

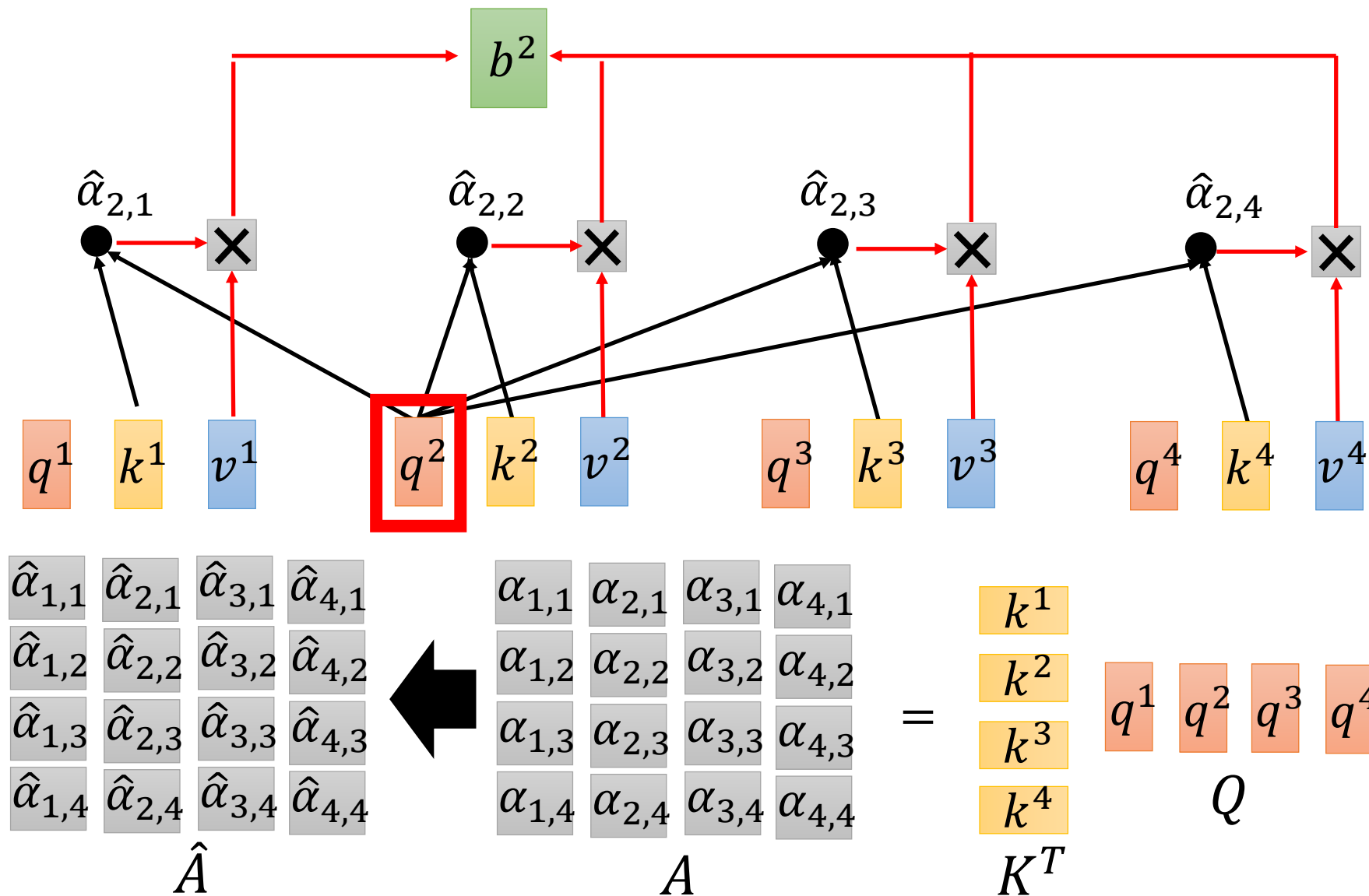


Self-attention



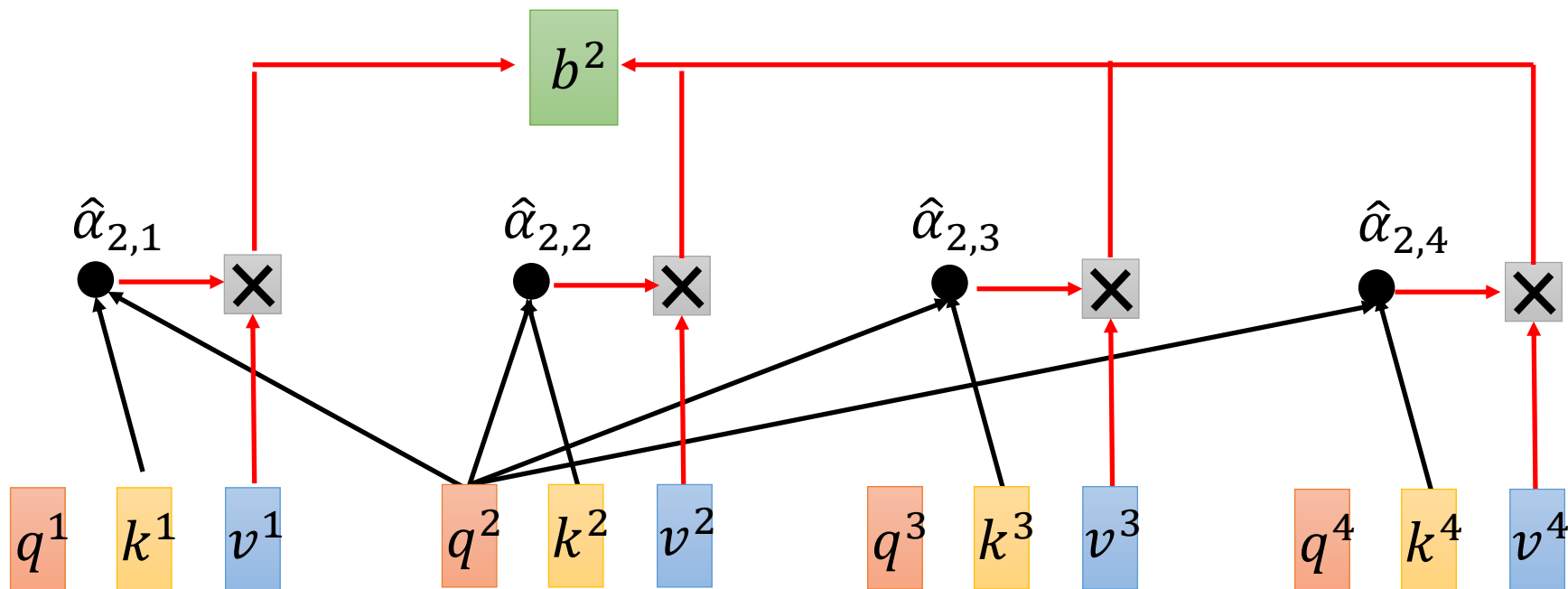
Self-attention

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



Self-attention

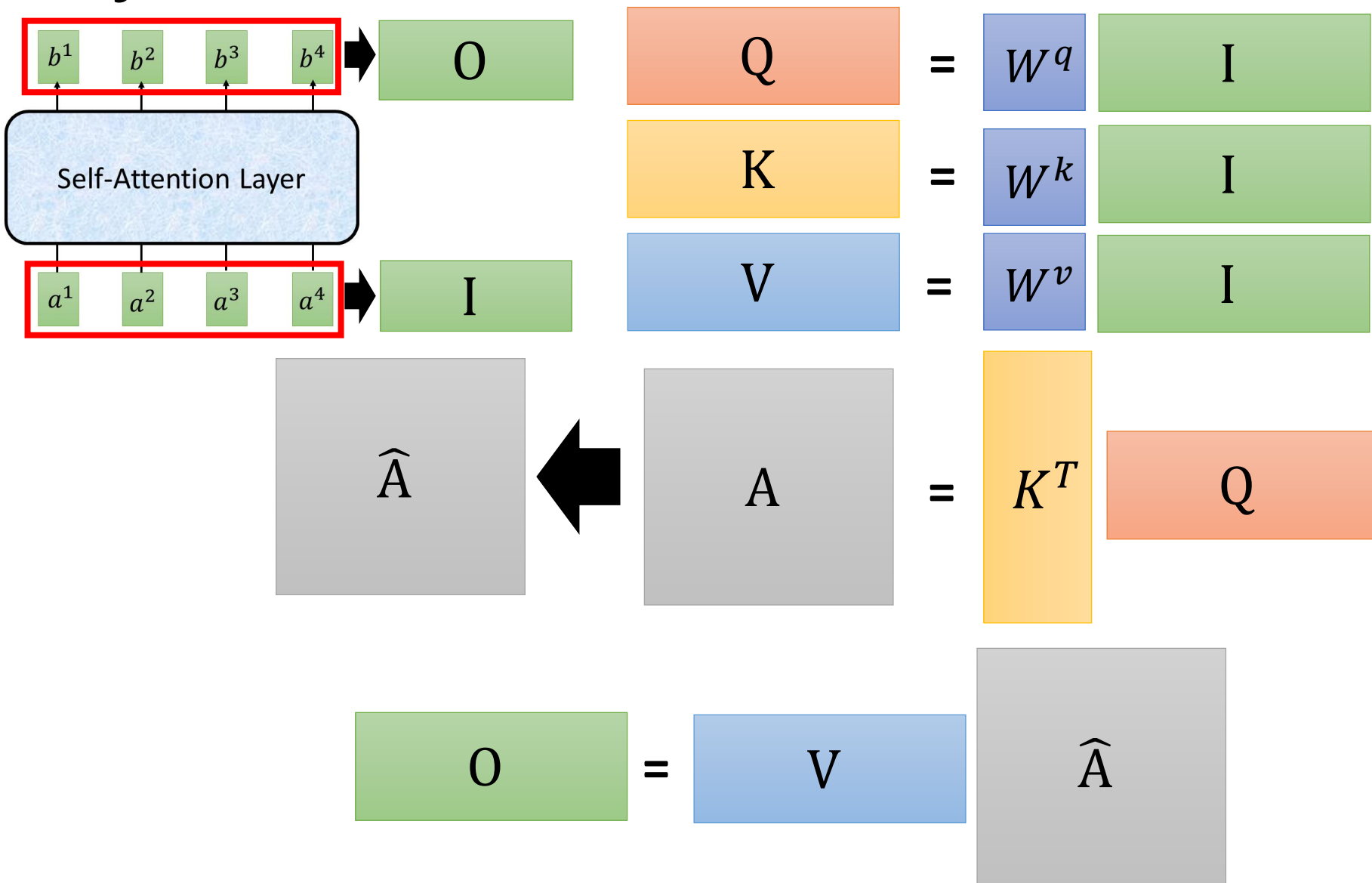
$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



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

$\hat{\alpha}_{1,1}$	$\hat{\alpha}_{2,1}$	$\hat{\alpha}_{3,1}$	$\hat{\alpha}_{4,1}$
$\hat{\alpha}_{1,2}$	$\hat{\alpha}_{2,2}$	$\hat{\alpha}_{3,2}$	$\hat{\alpha}_{4,2}$
$\hat{\alpha}_{1,3}$	$\hat{\alpha}_{2,3}$	$\hat{\alpha}_{3,3}$	$\hat{\alpha}_{4,3}$
$\hat{\alpha}_{1,4}$	$\hat{\alpha}_{2,4}$	$\hat{\alpha}_{3,4}$	$\hat{\alpha}_{4,4}$

Self-attention



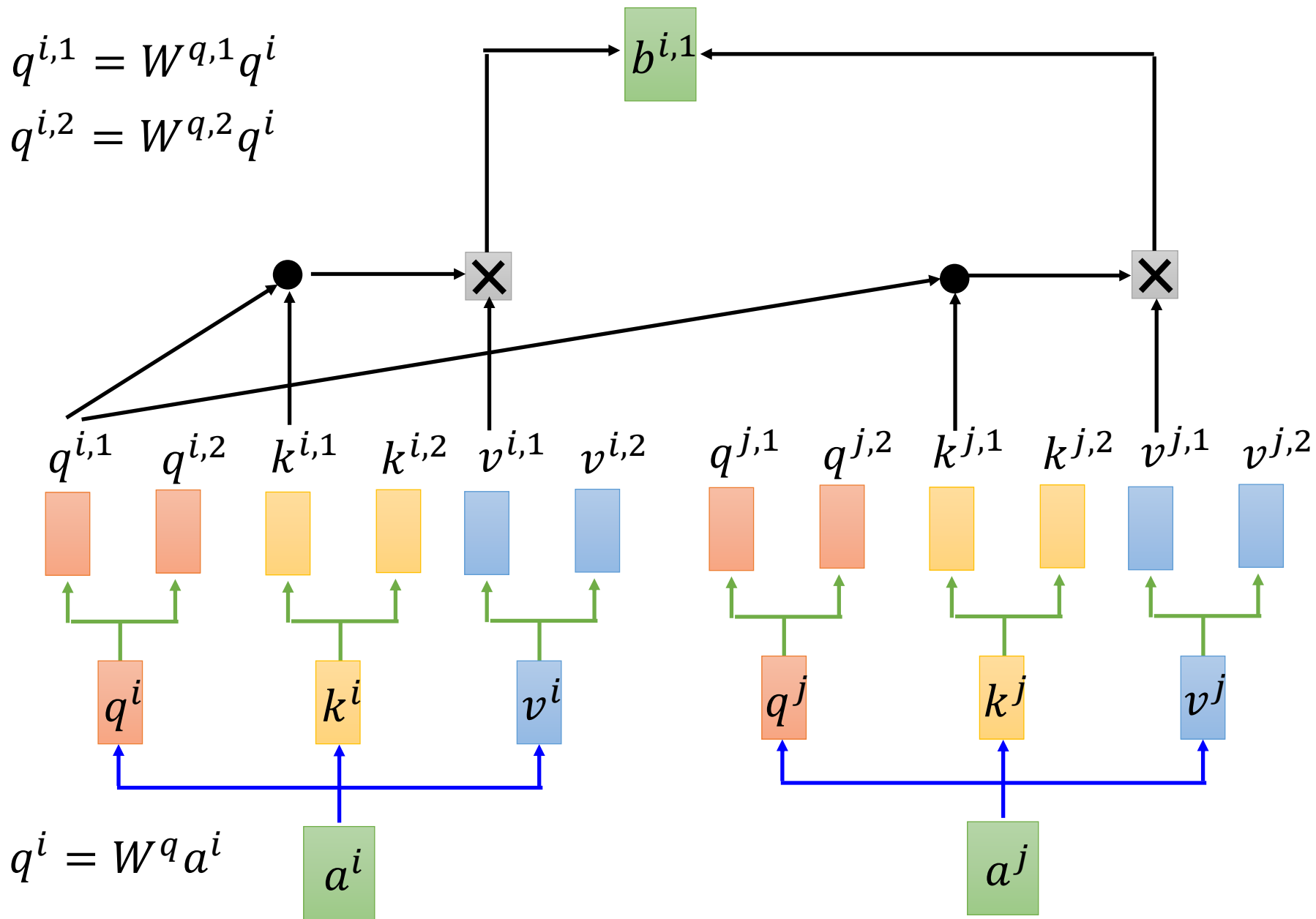
反正就是一堆矩陣乘法，用 GPU 可以加速

Multi-head Self-attention

(2 heads as example)

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

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

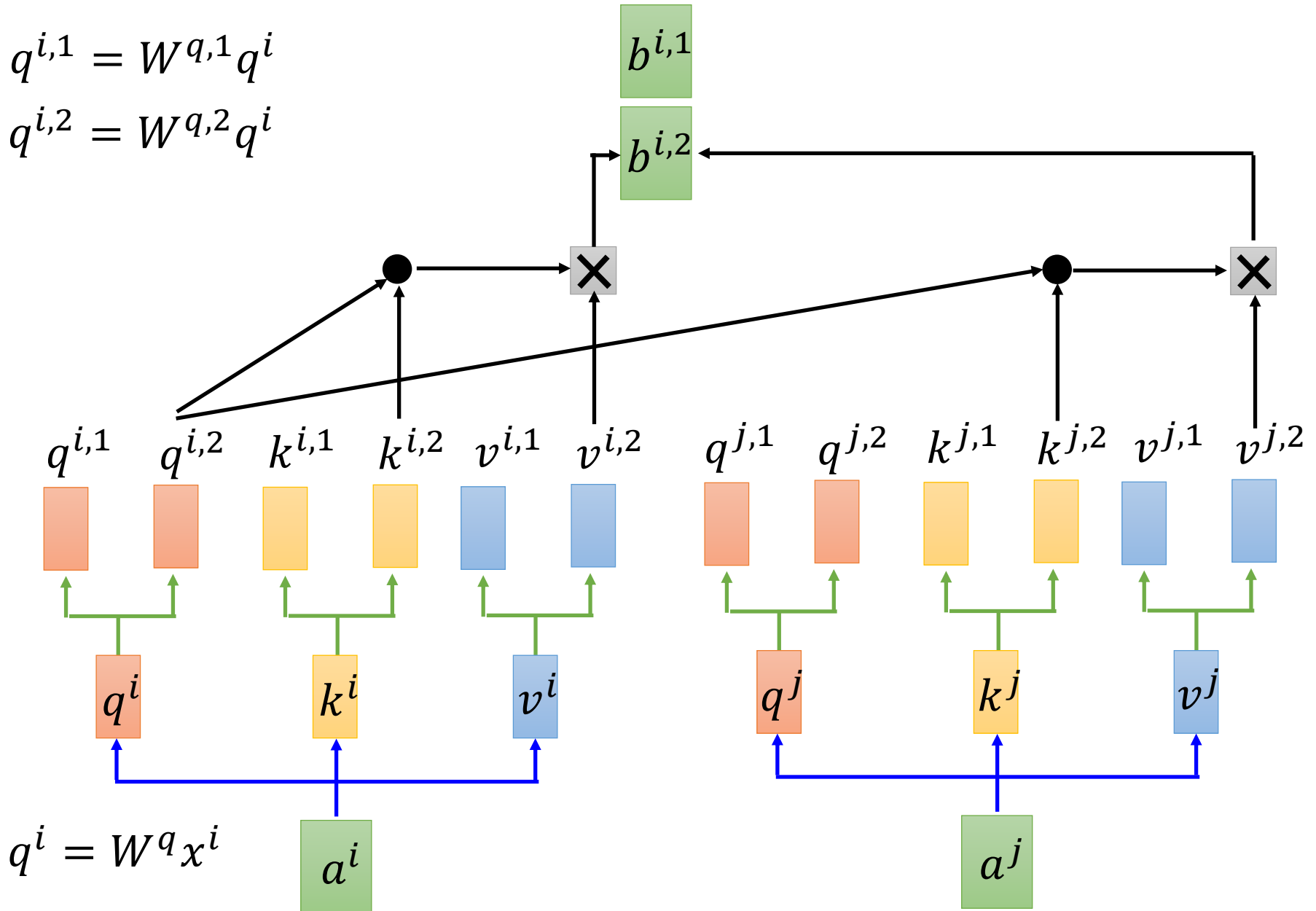


Multi-head Self-attention

(2 heads as example)

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

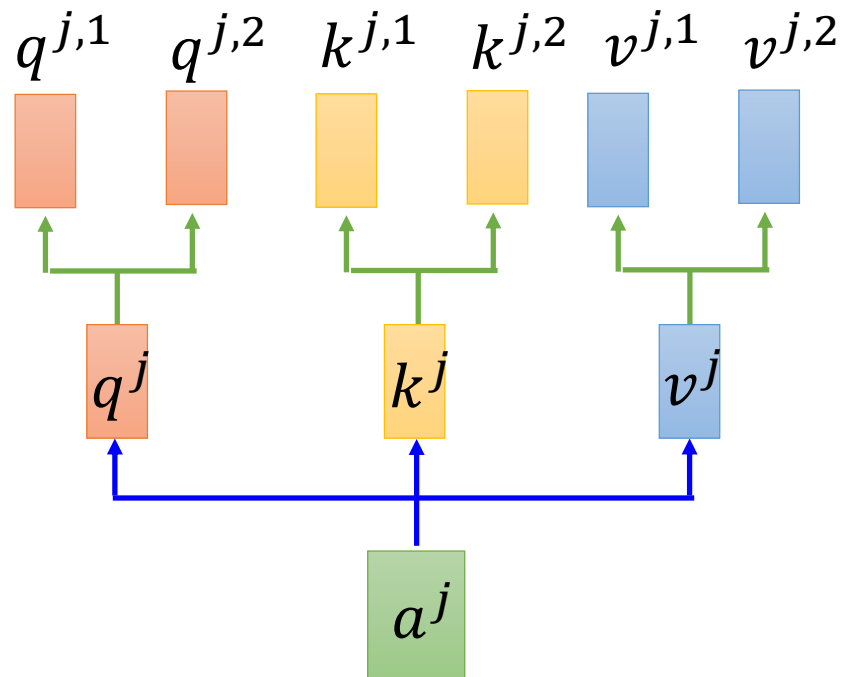
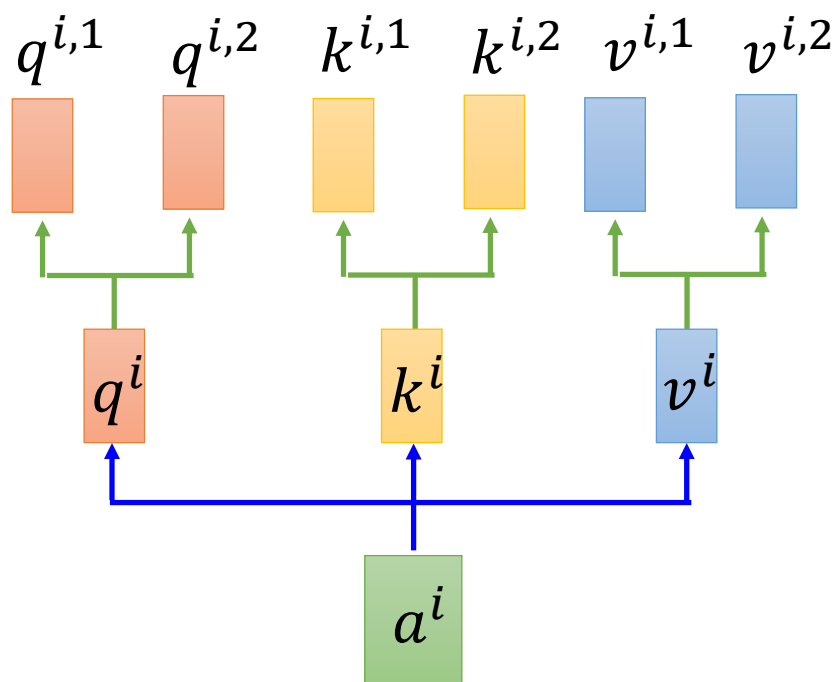
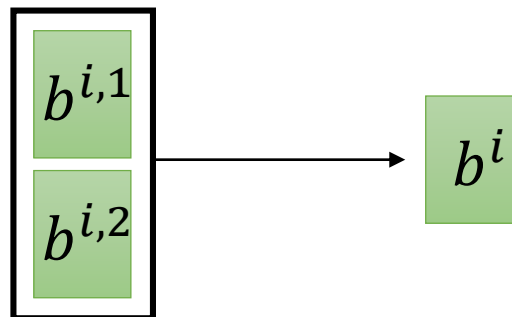
$$q^{i,2} = W^{q,2} q^i$$



Multi-head Self-attention

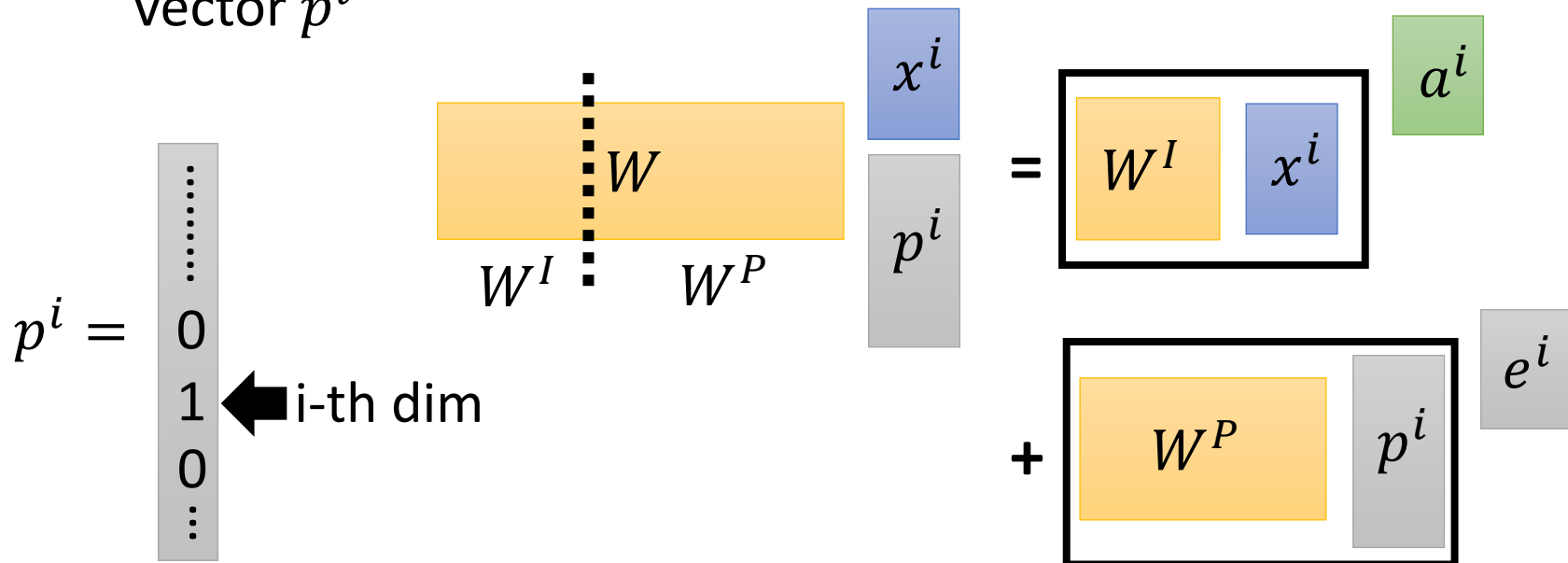
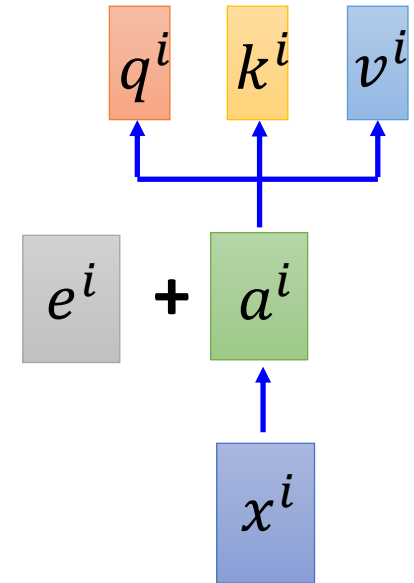
(2 heads as example)

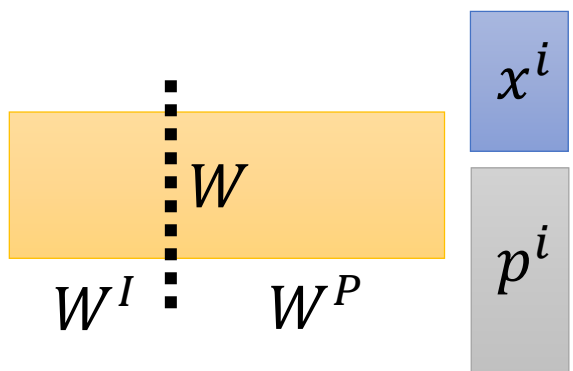
$$b^i = W^O \begin{bmatrix} b^{i,1} \\ b^{i,2} \end{bmatrix}$$



Positional Encoding

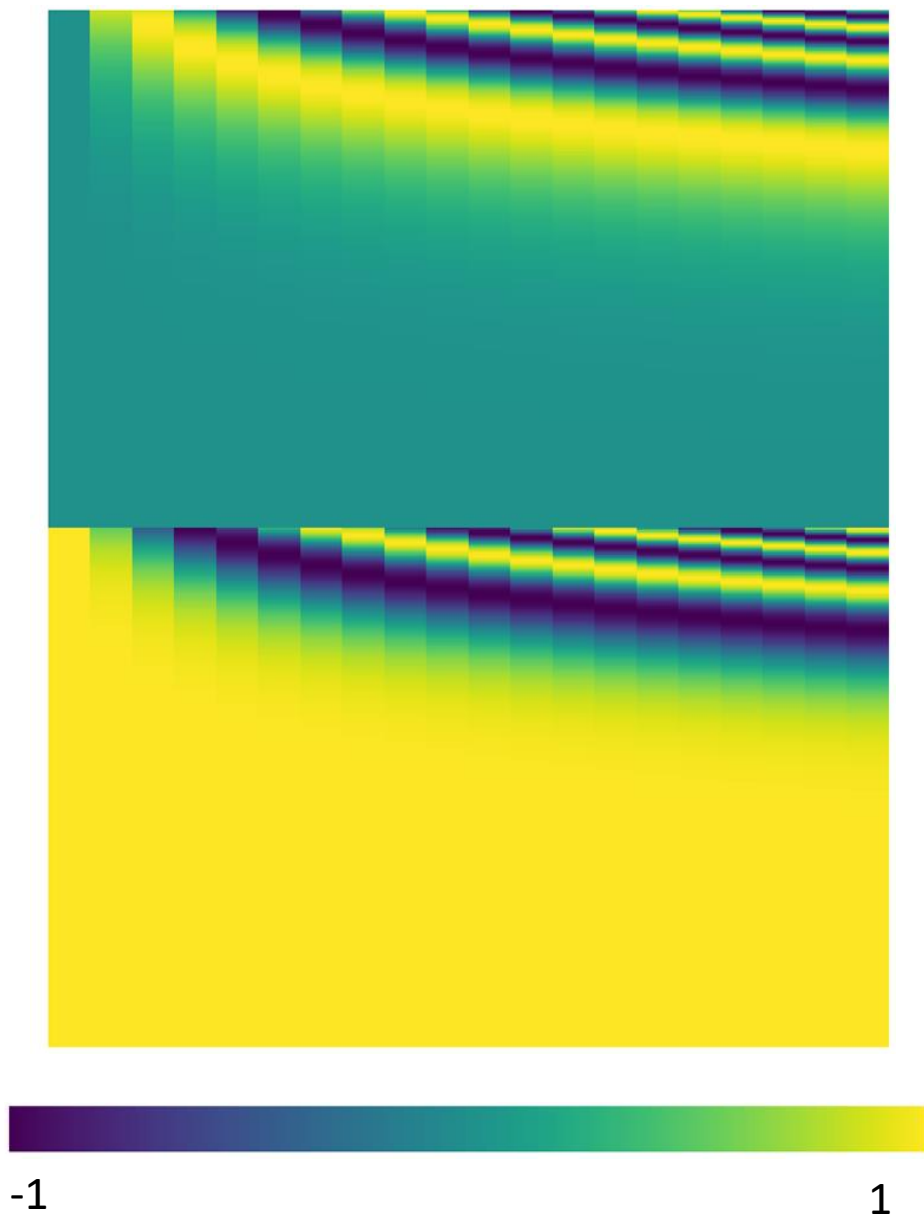
- No position information in self-attention.
- Original paper: each position has a unique positional vector e^i (not learned from data)
- In other words: each x^i appends a one-hot vector p^i





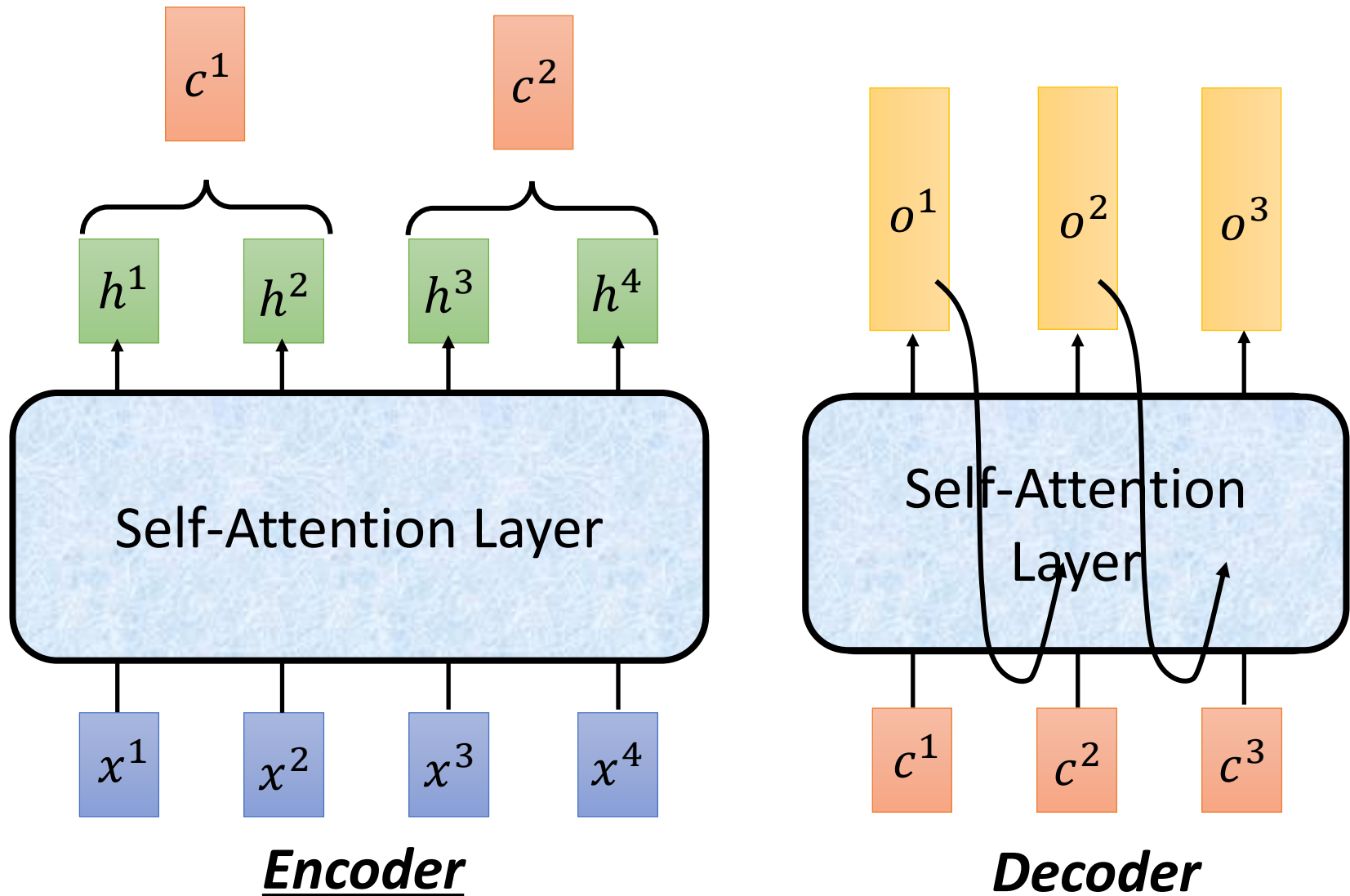
$$= \begin{bmatrix} W^I & x^i \end{bmatrix} a^i + \begin{bmatrix} W^P & p^i \end{bmatrix} e^i$$

The equation shows the matrix W being multiplied by a vector a^i (green box) and added to the product of W^P and p^i (gray box). The vectors x^i and p^i are shown as components of the first term.



source of image: <http://jalammar.github.io/illustrated-transformer/>

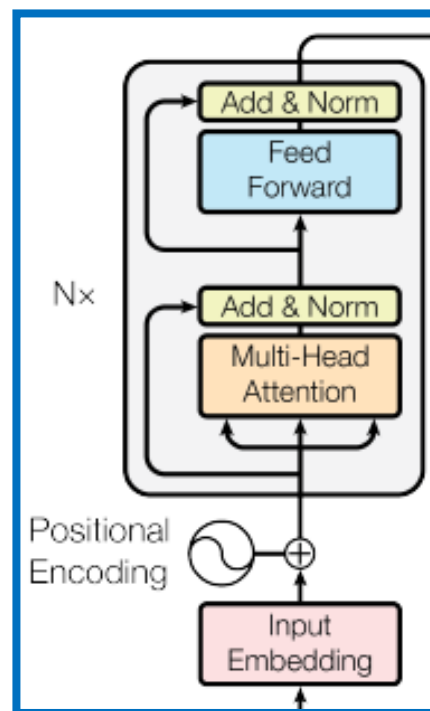
Seq2seq with Attention



Transformer

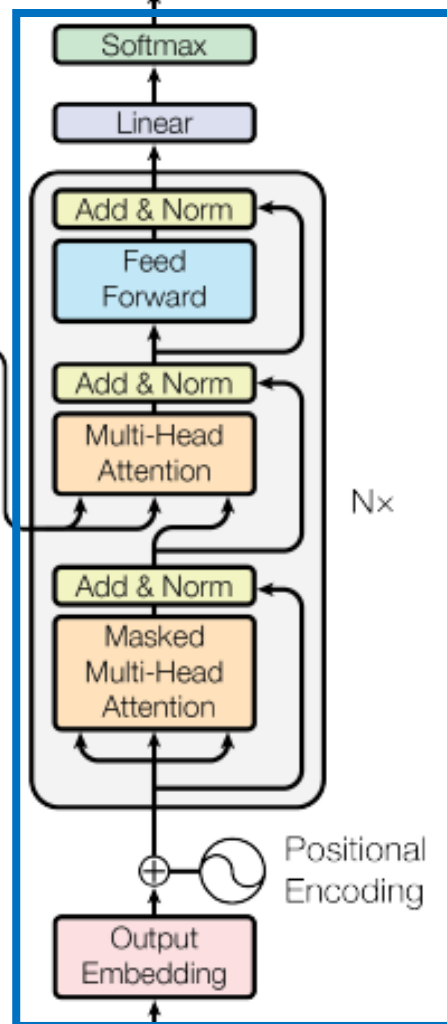
Using Chinese to English translation as example

Encoder



機器學習

machine learning

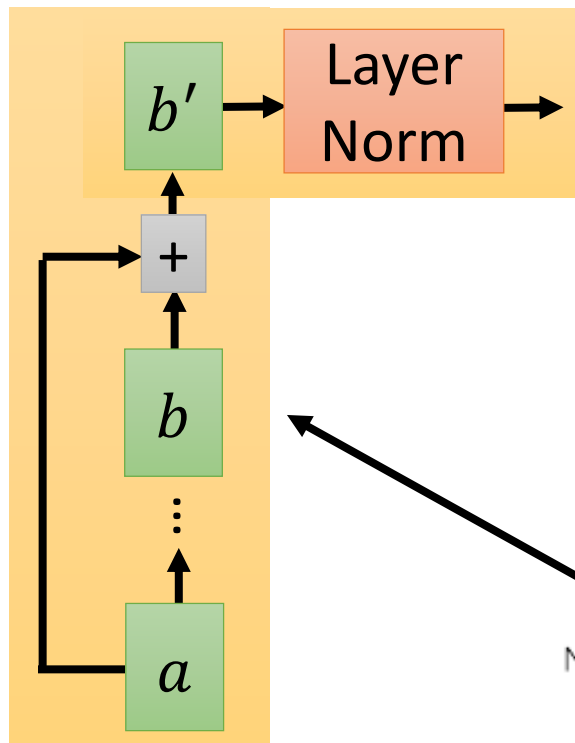


Decoder

<BOS>

machine

Transformer

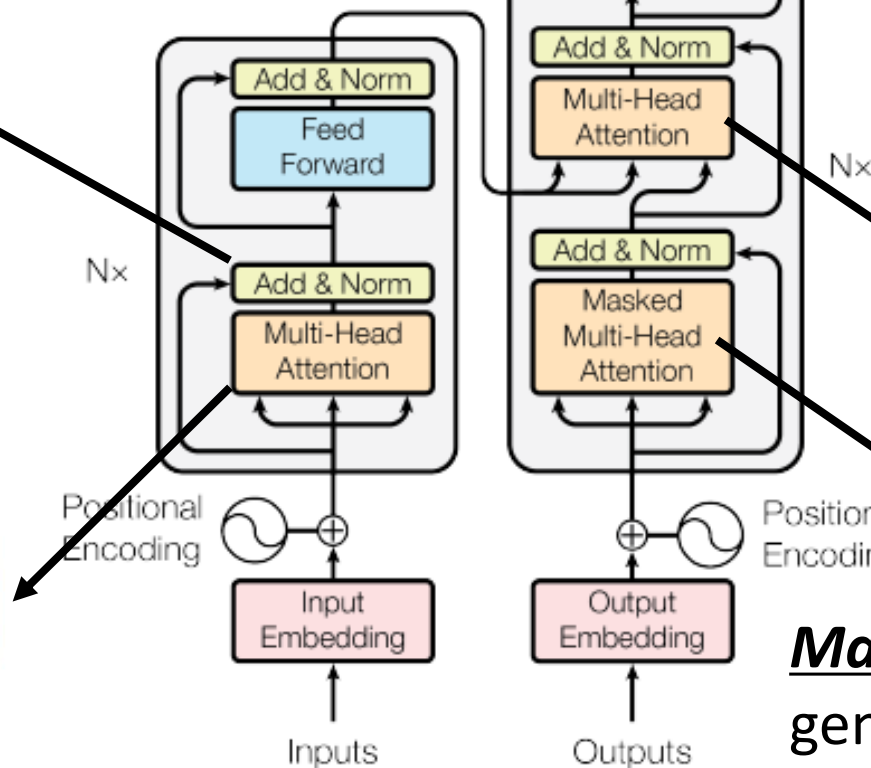
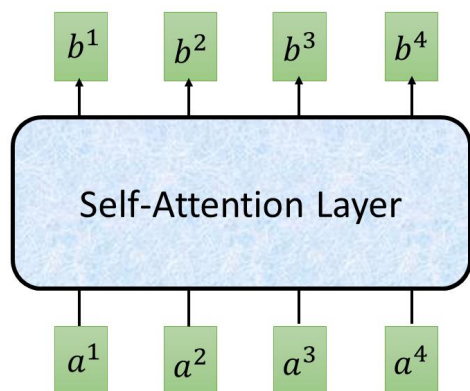


Layer Norm:

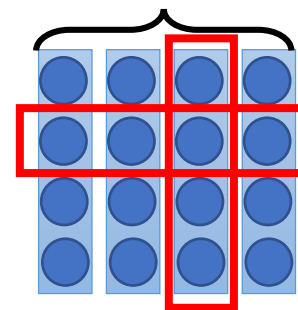
<https://arxiv.org/abs/1607.06450>

Batch Norm:

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



Batch Size



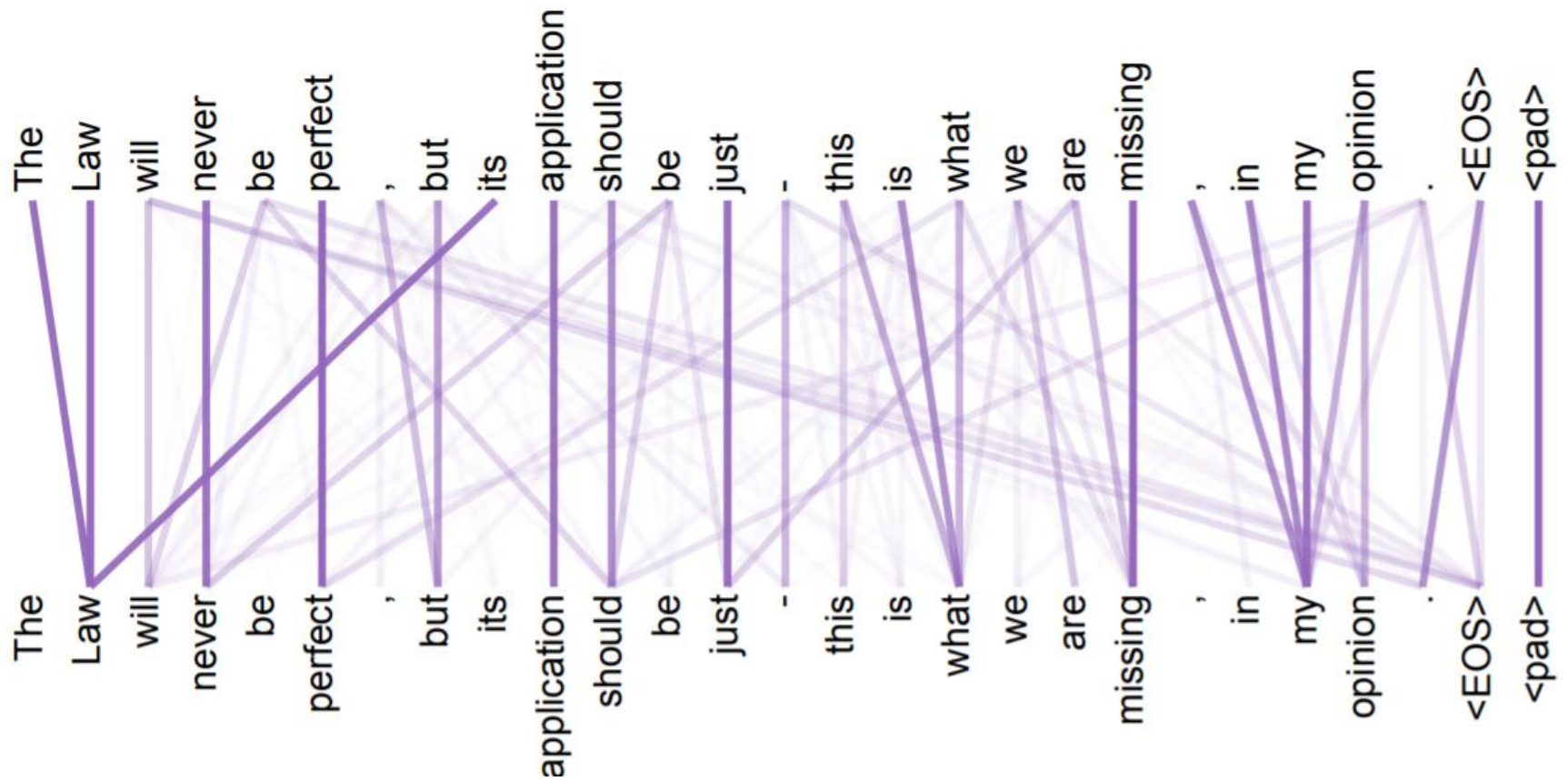
$\mu = 0,$
 $\sigma = 1$
Batch

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

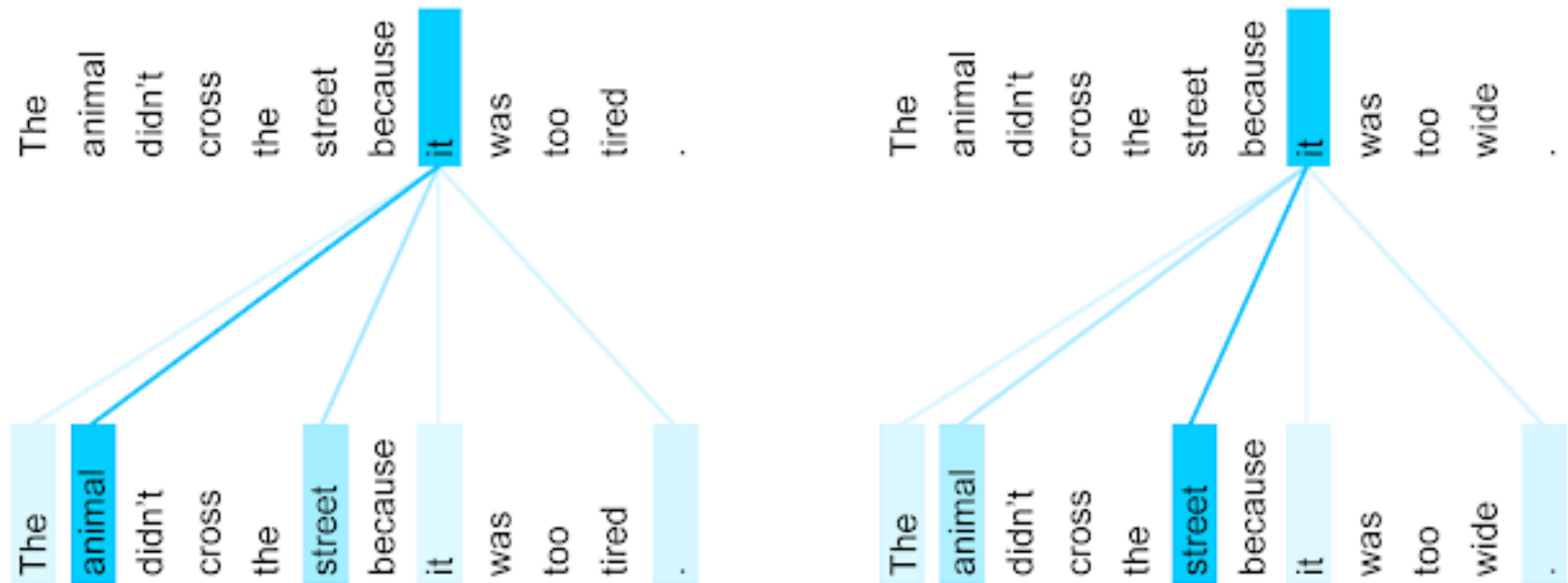
attend on the
input sequence

Masked: attend on the
generated sequence

Attention Visualization



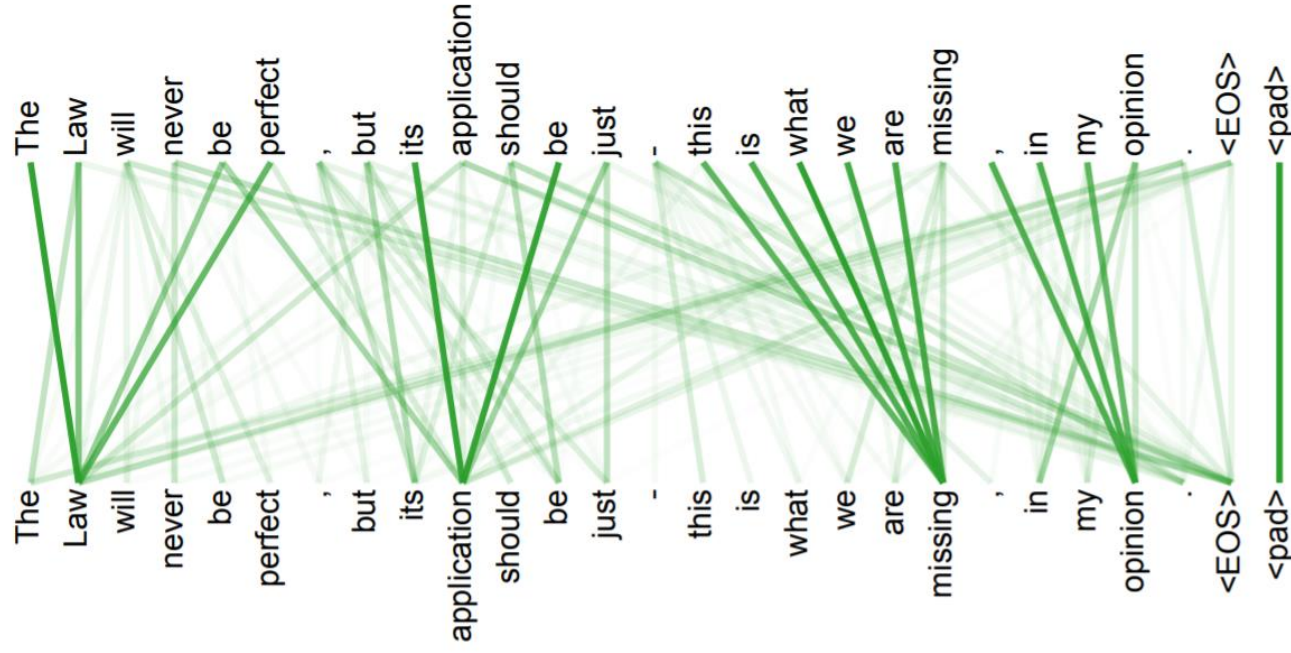
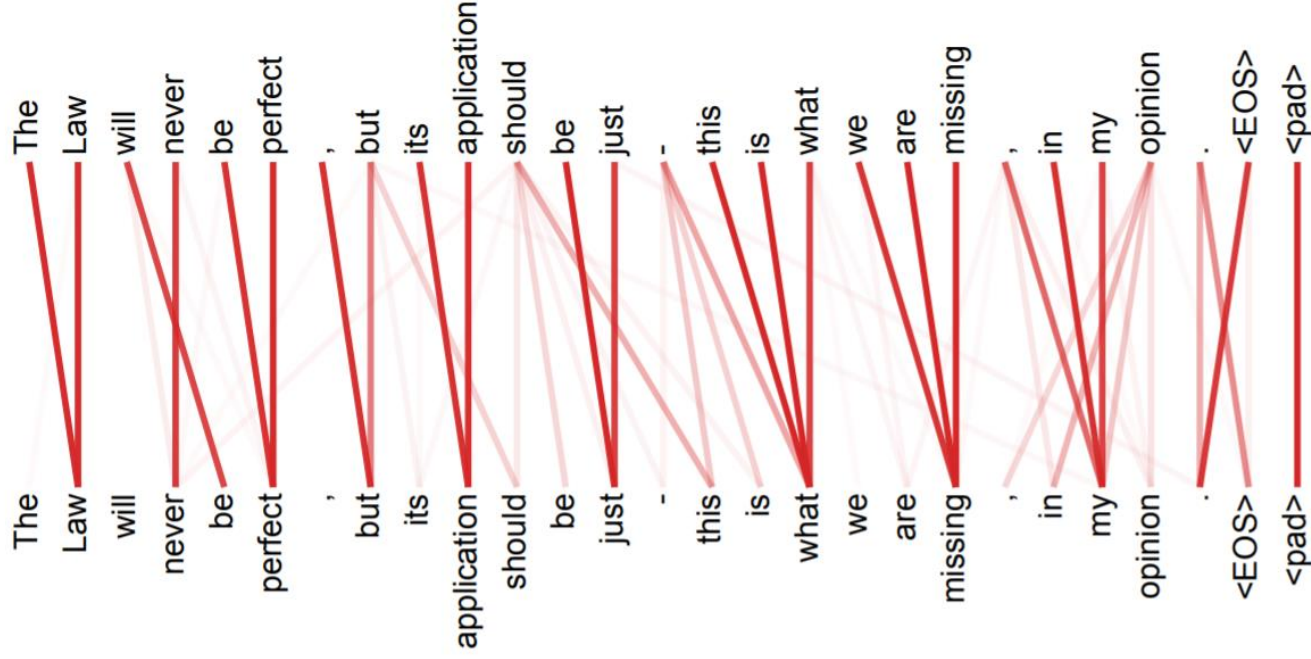
Attention Visualization



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).

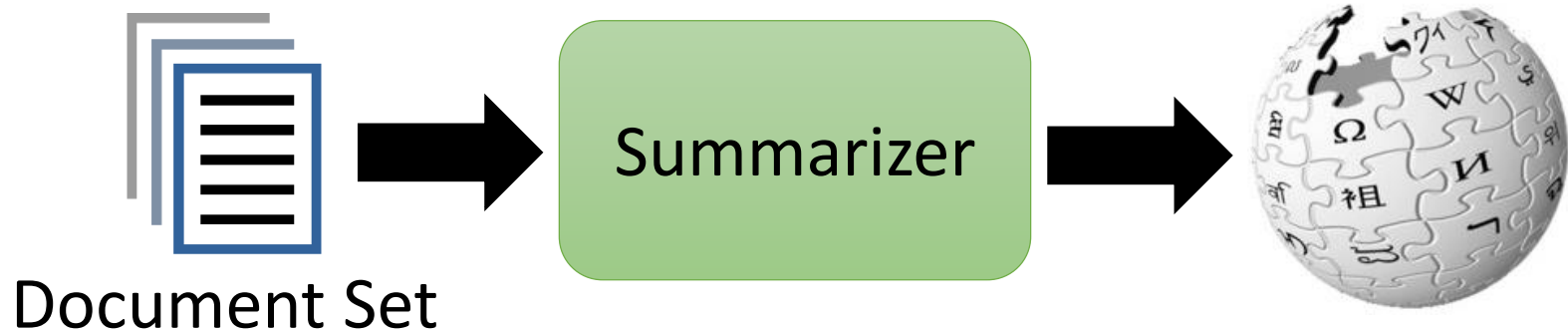
<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

Multi-head Attention



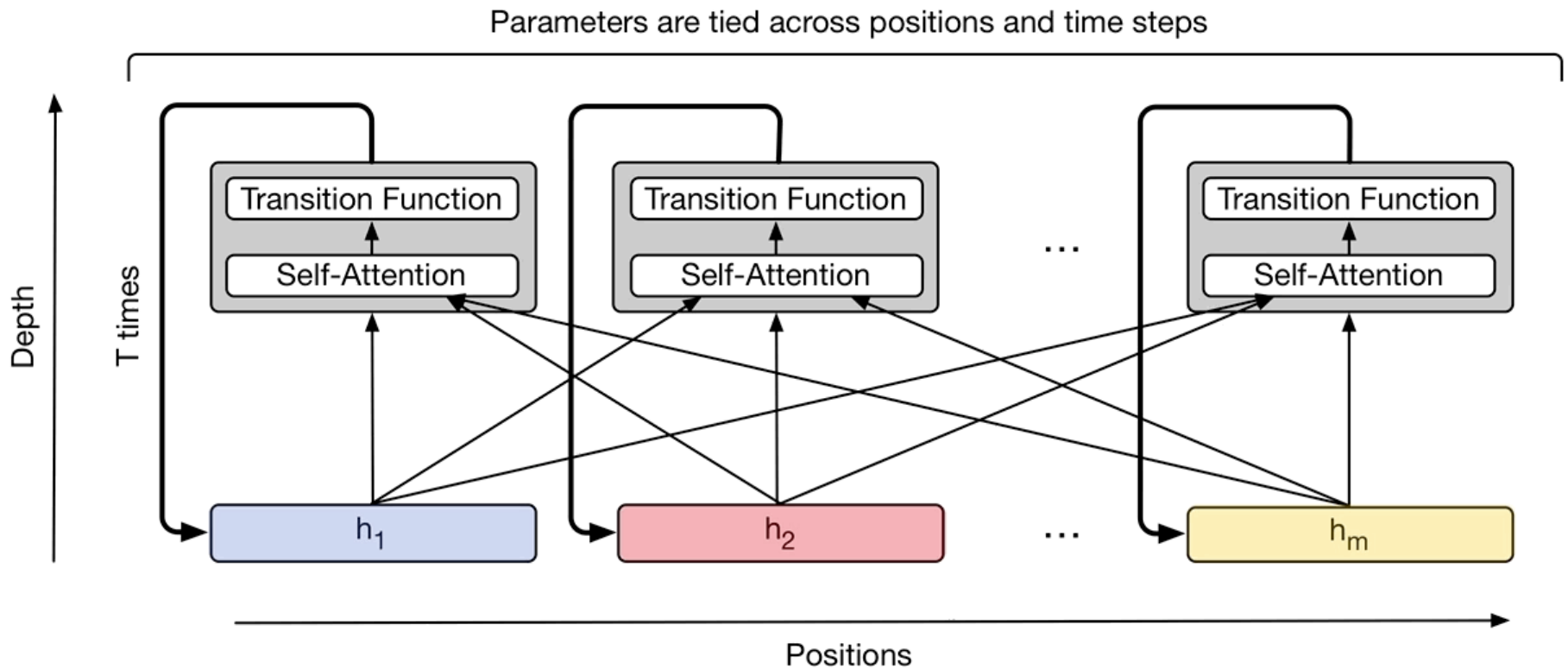
Example Application

- 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

Universal Transformer



<https://ai.googleblog.com/2018/08/moving-beyond-translation-with.html>

Self-Attention GAN

