Pattern Recognition

Transformers

林彥宇 教授
Yen-Yu Lin, Professor
國立陽明交通大學 資訊工程學系
Computer Science, National Yang Ming Chiao Tung University

Some slides are modified from Hong-yi Lee, Shusen Wang, Sy-Kai Chen, and Shao-Hao Lu

Outline

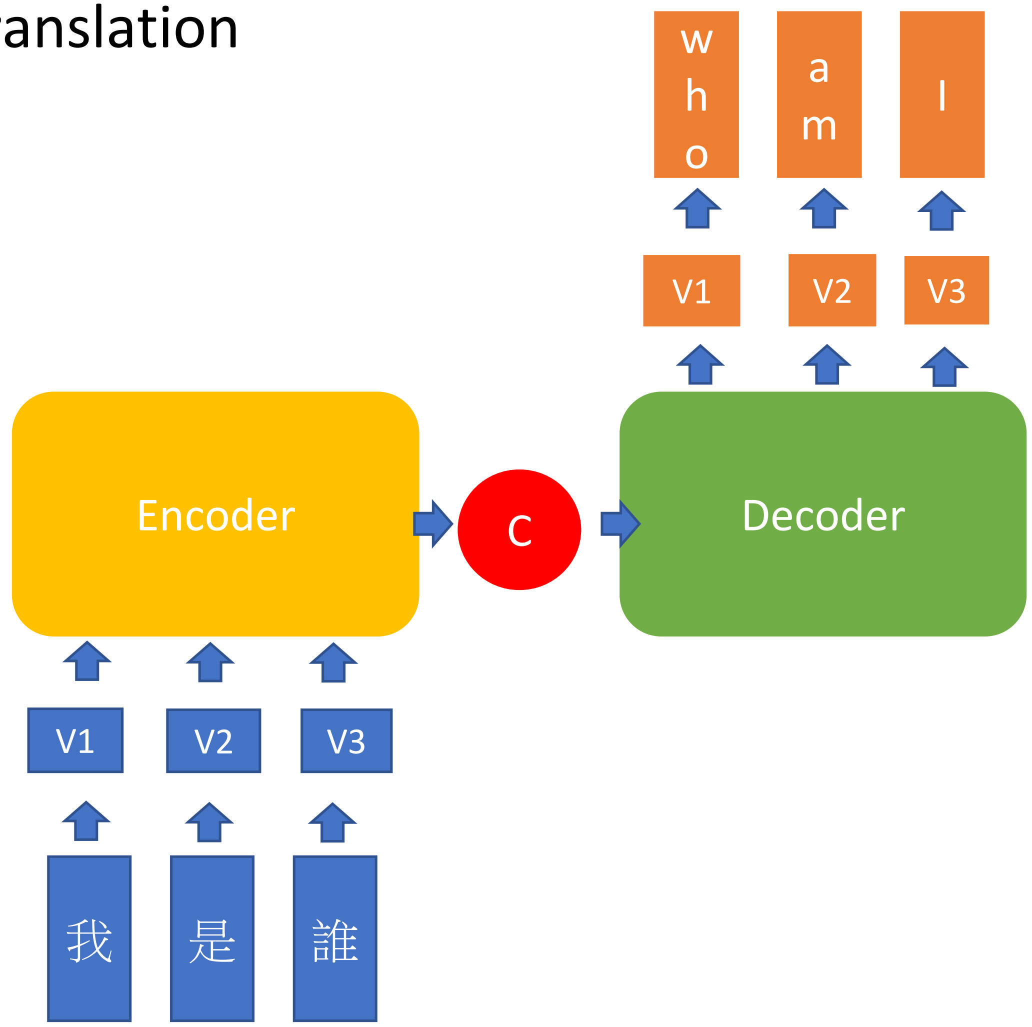
- Transformer
- ViT

Outline

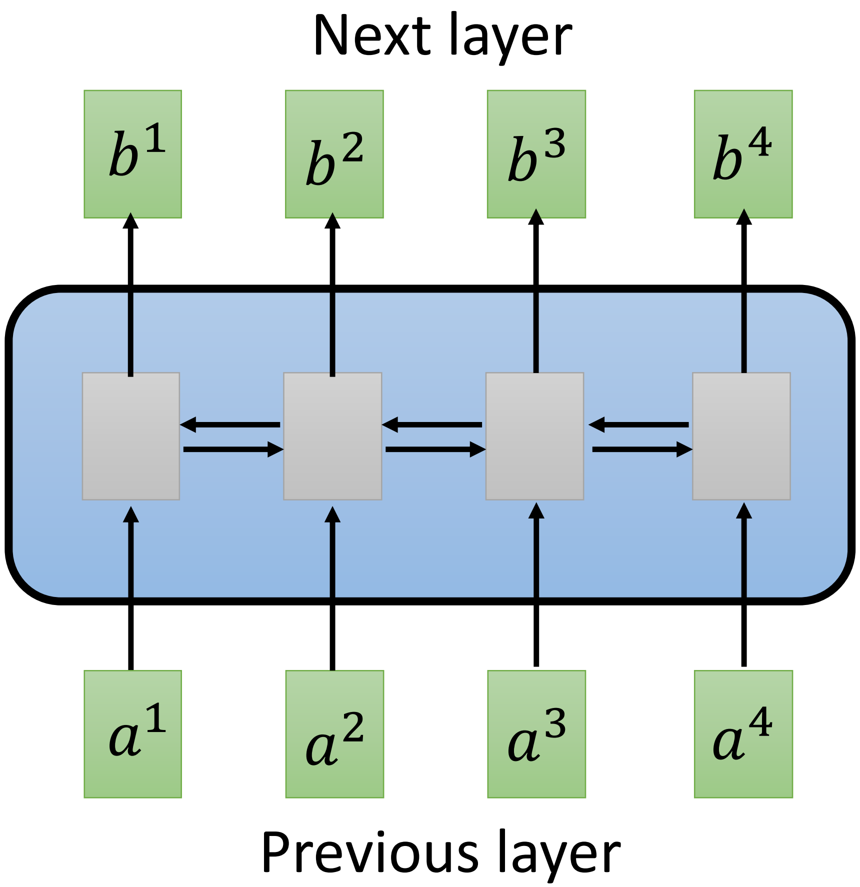
- Transformer
  - Most slides are (modified) from Prof. Hung-yi Lee
- ViT

Seq2seq model

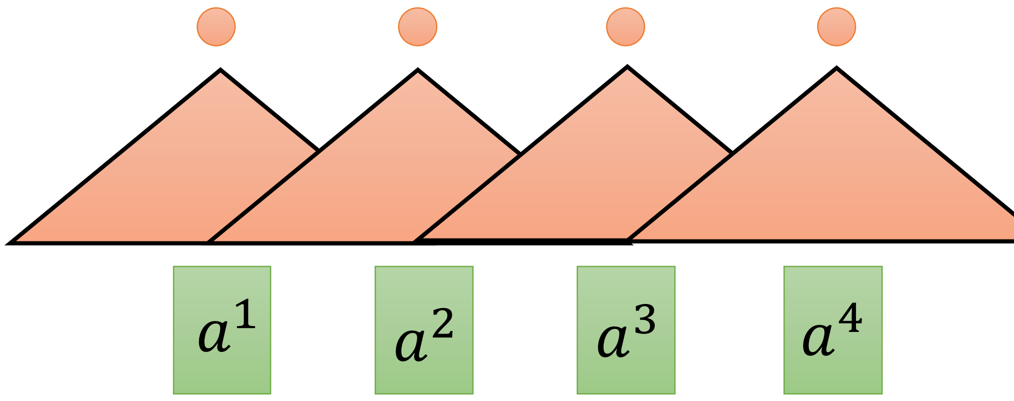
- Language translation



Sequence

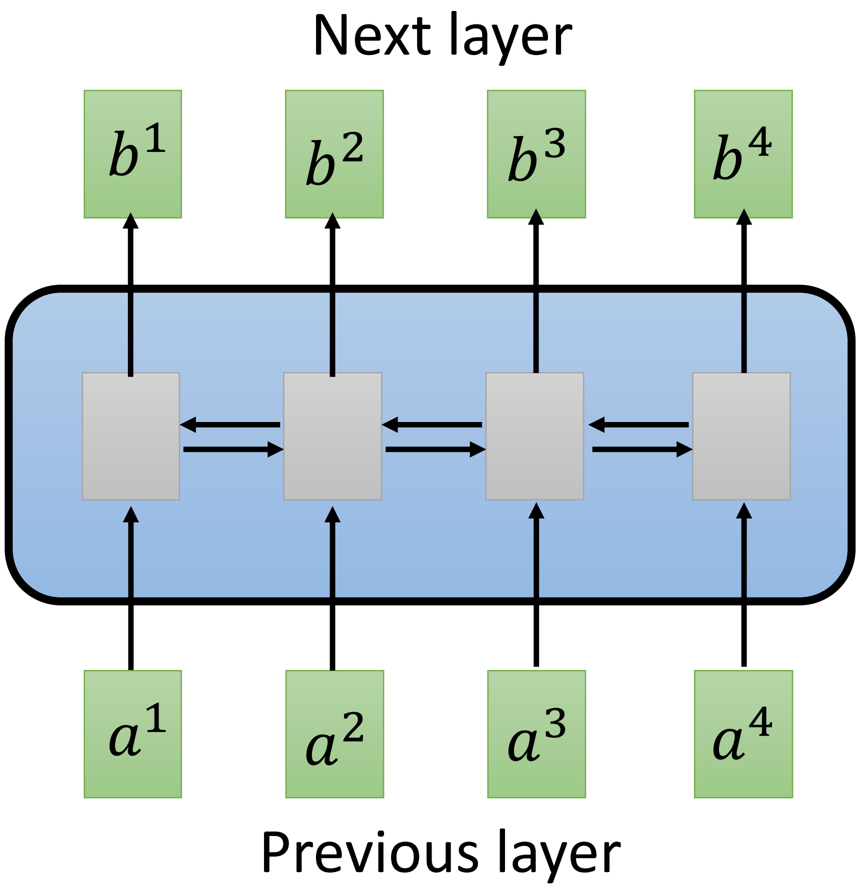


Hard to parallel !



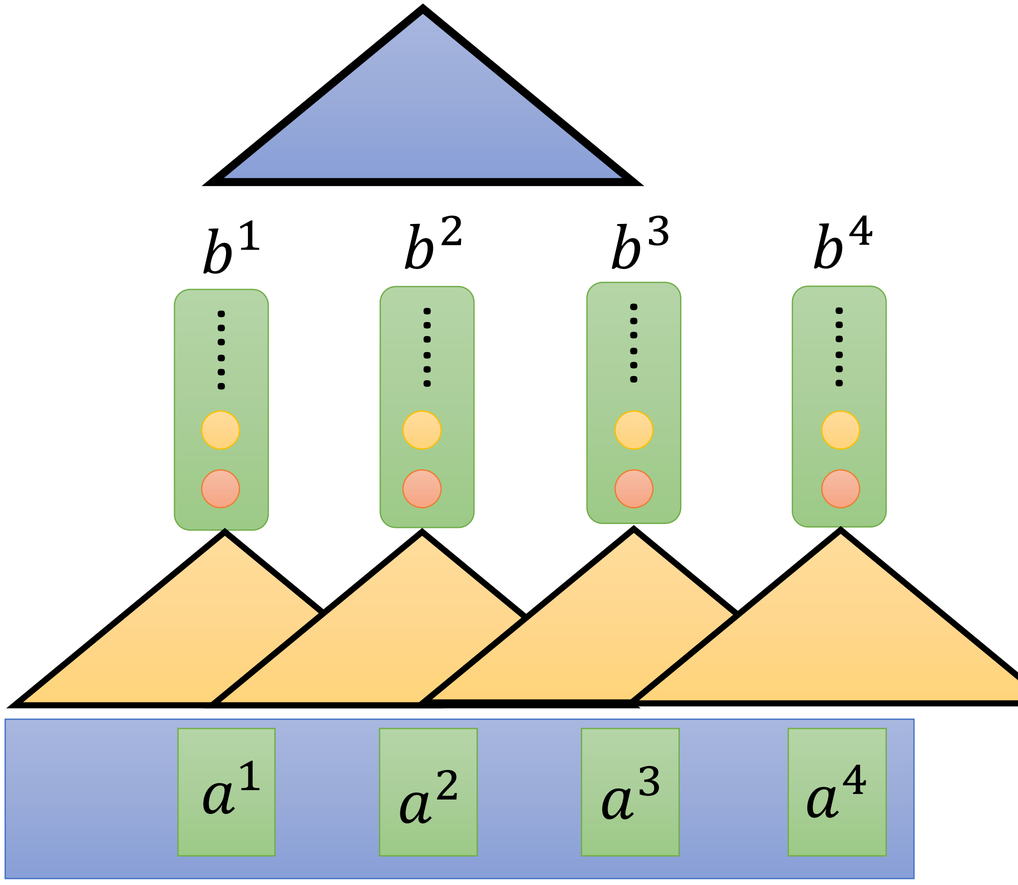
Using CNN to replace RNN

Sequence



Hard to parallel

Filters in higher layer can consider longer sequence



Using CNN to replace RNN (CNN can parallel)

Self-attention

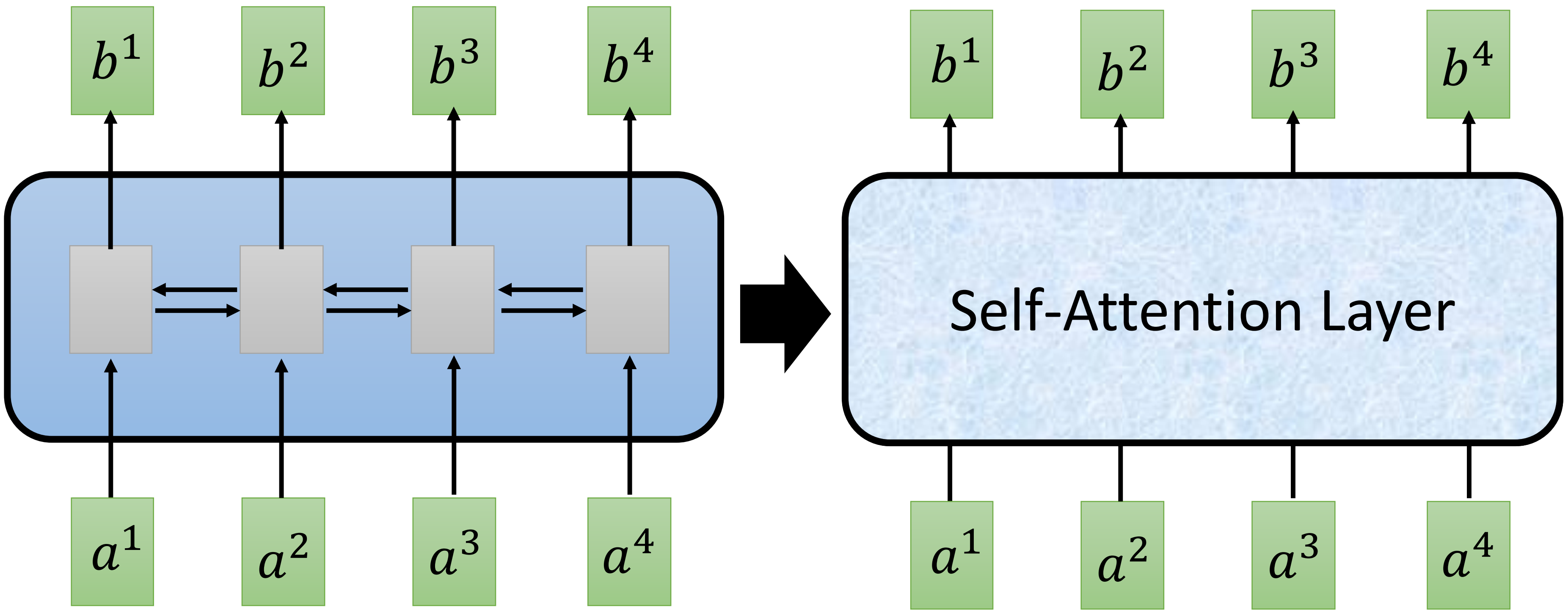
- What is self attention?
  - Self attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence

- Why self attention?
  - Solve the fundamental constraint in recurrent models

Self-Attention

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

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



Self-attention

https://arxiv.org/abs/1706.03762

q: query (to match others)

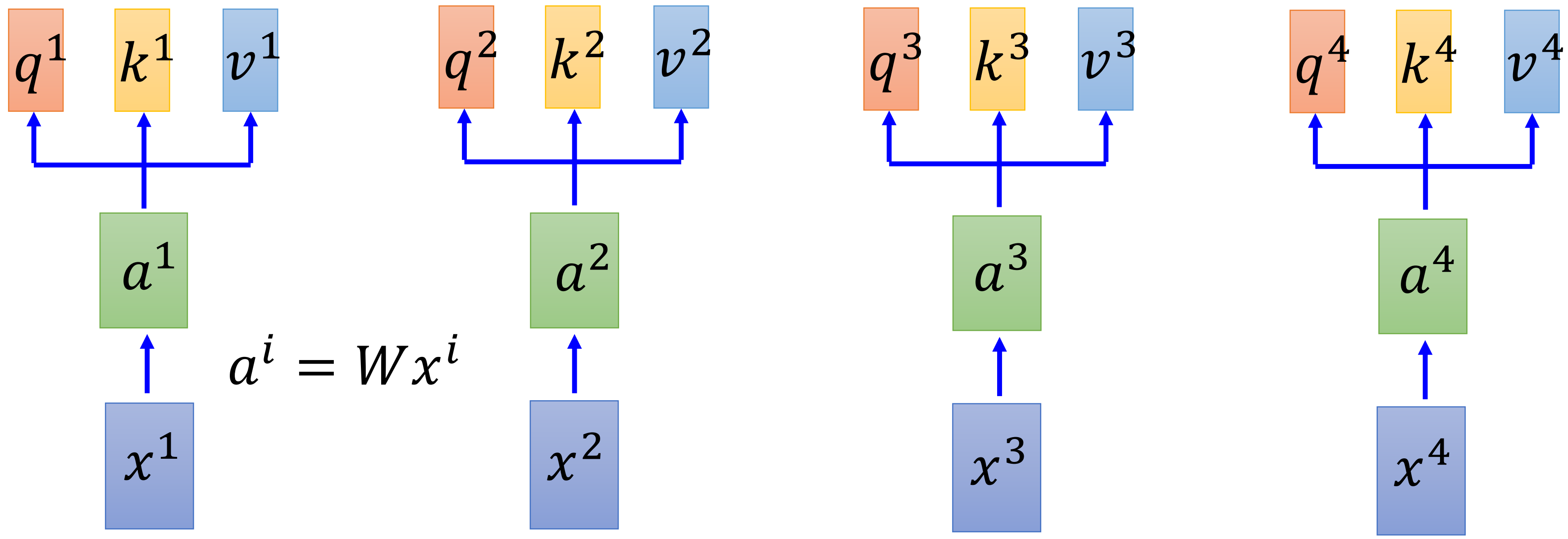
$$q^i = W^q a^i$$

k: key (to be matched)

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

v : value (information to be extracted)

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

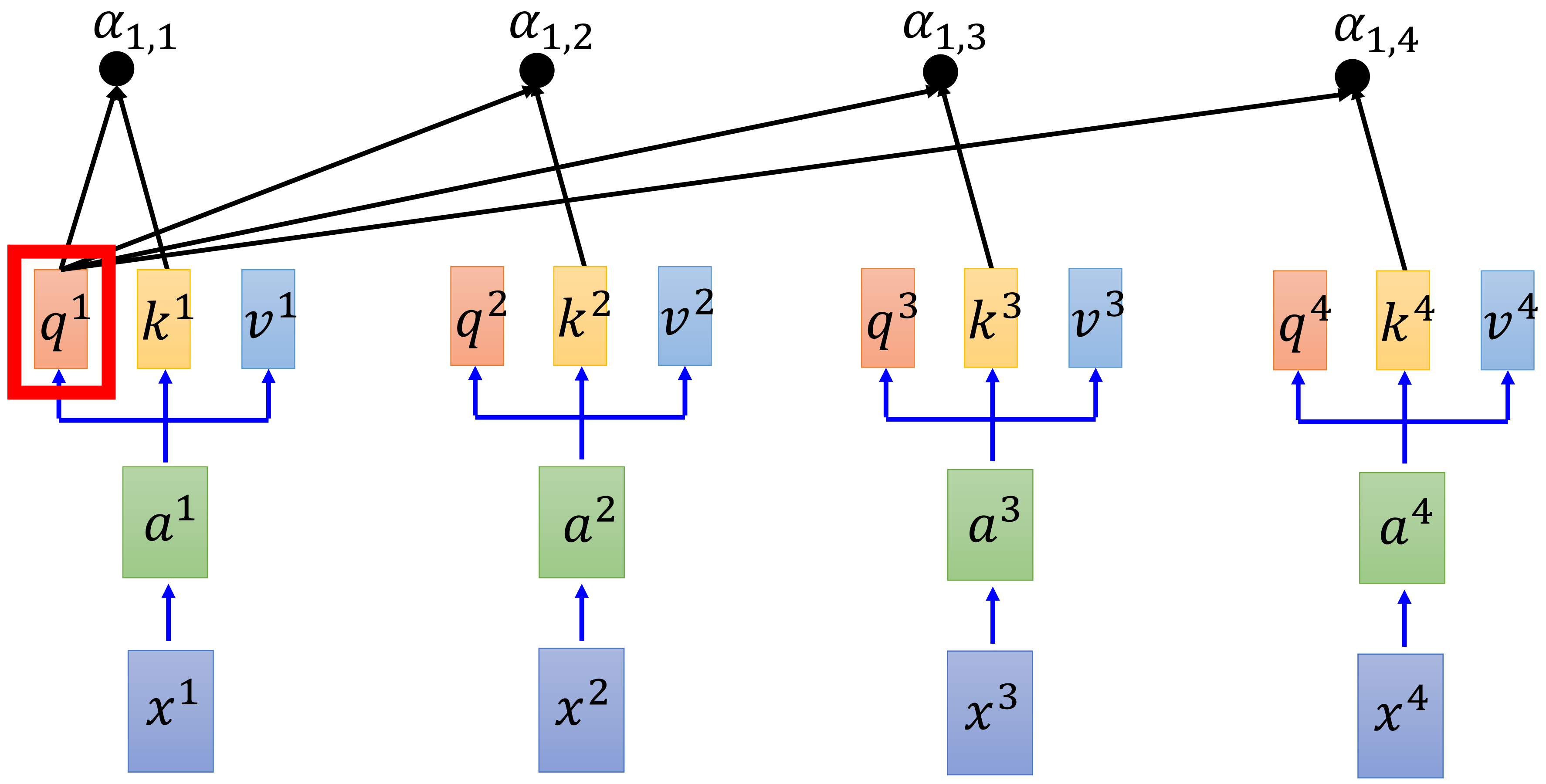


Self-attention

Calculate attention between each query and all keys

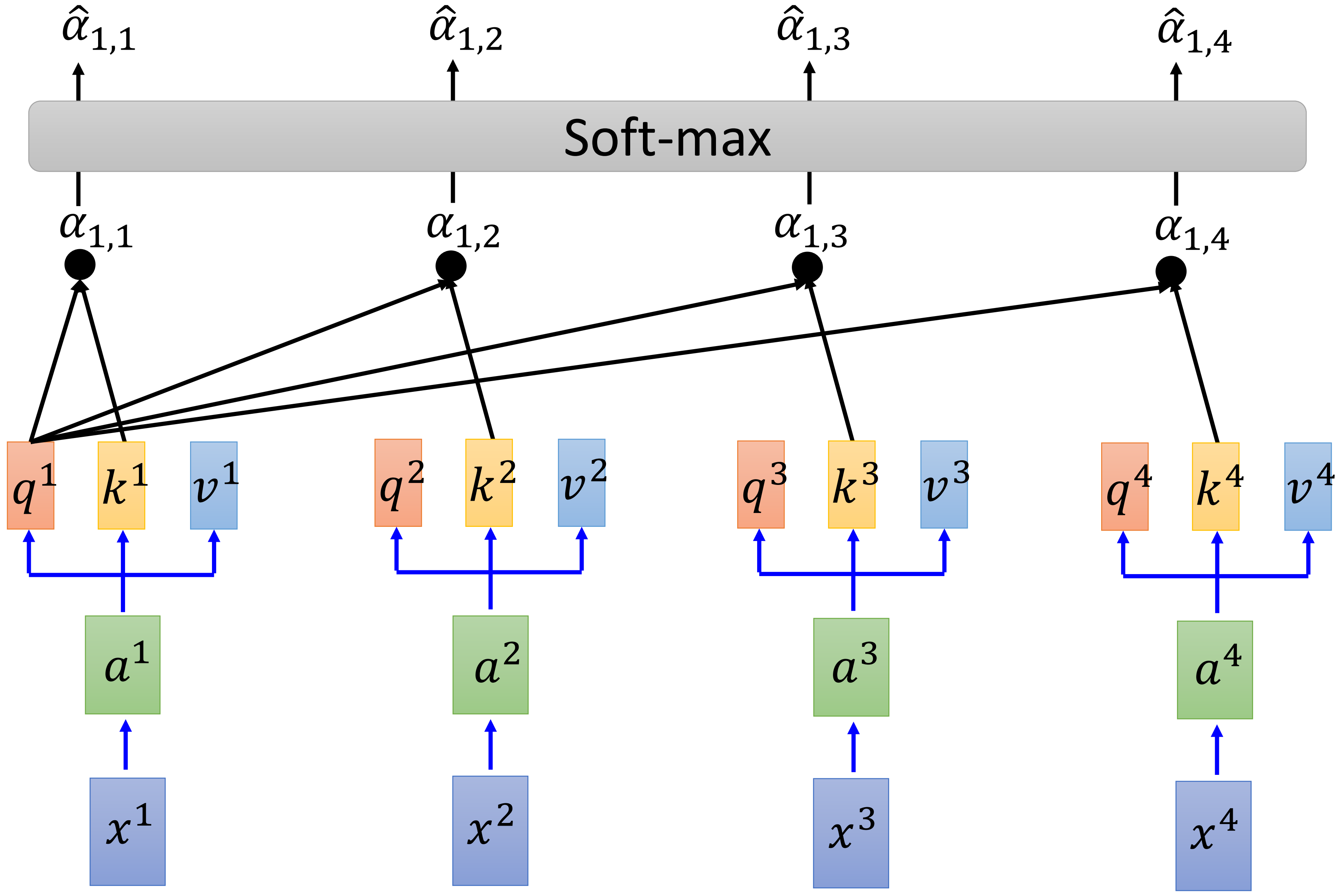
d is the dim of q and k

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



Self-attention

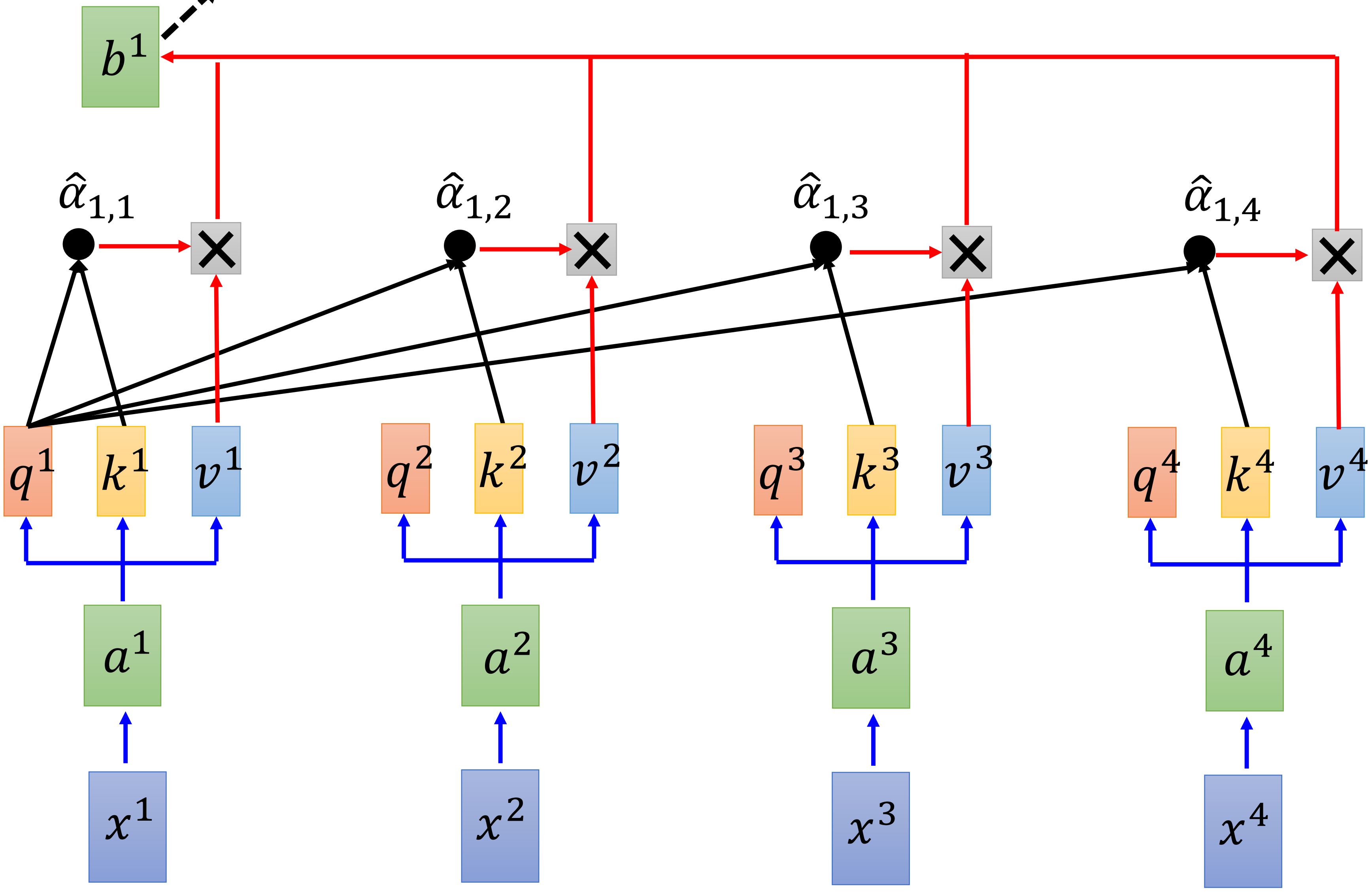
$$\hat{\alpha}_{1,i} = exp(\alpha_{1,i})/\sum_j exp(\alpha_{1,j})$$



Self-attention

Considering the whole sequence

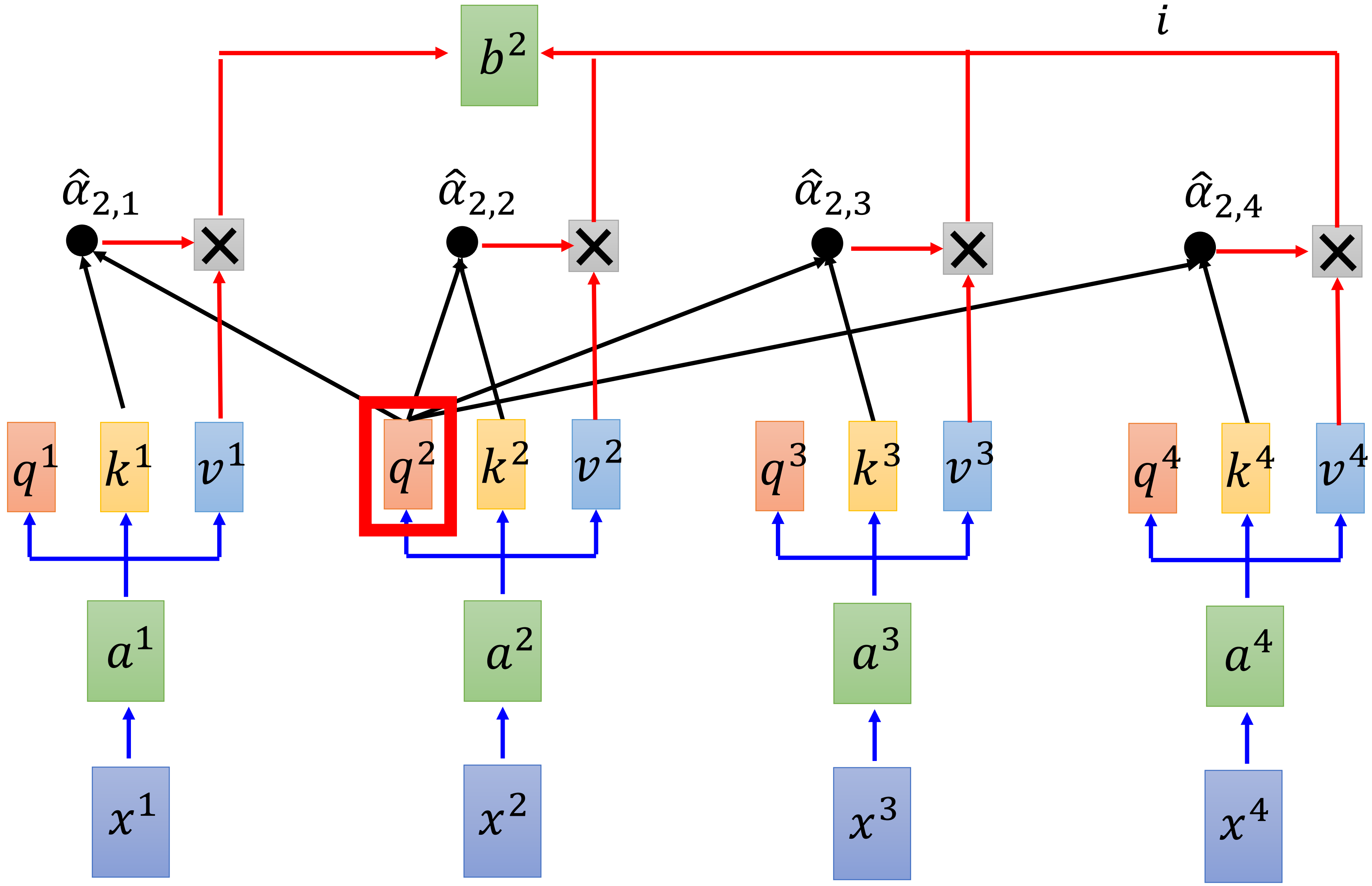
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$



Self-attention

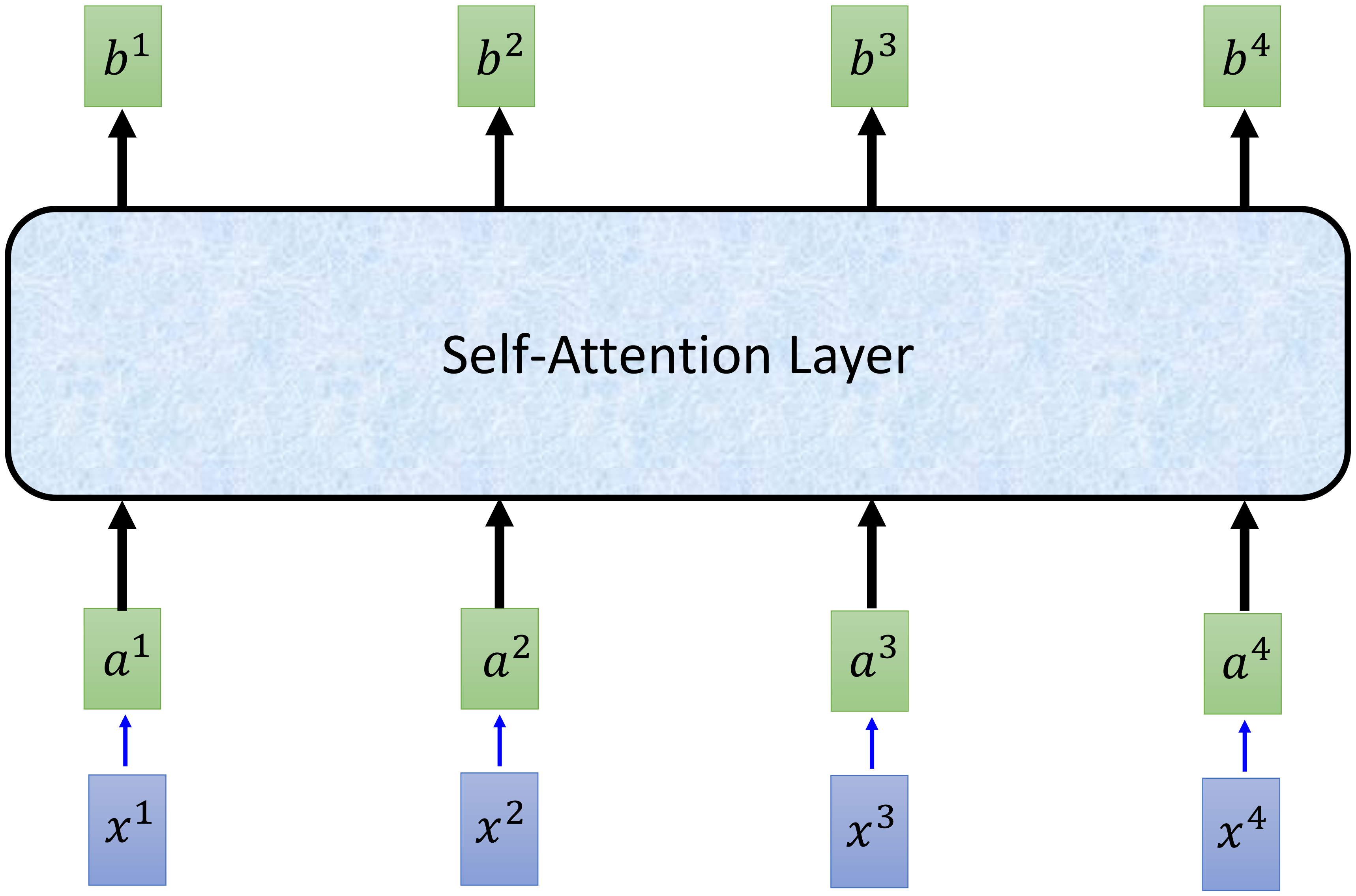
Calculate attention between each query and all keys

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



Self-attention

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



Self-attention

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

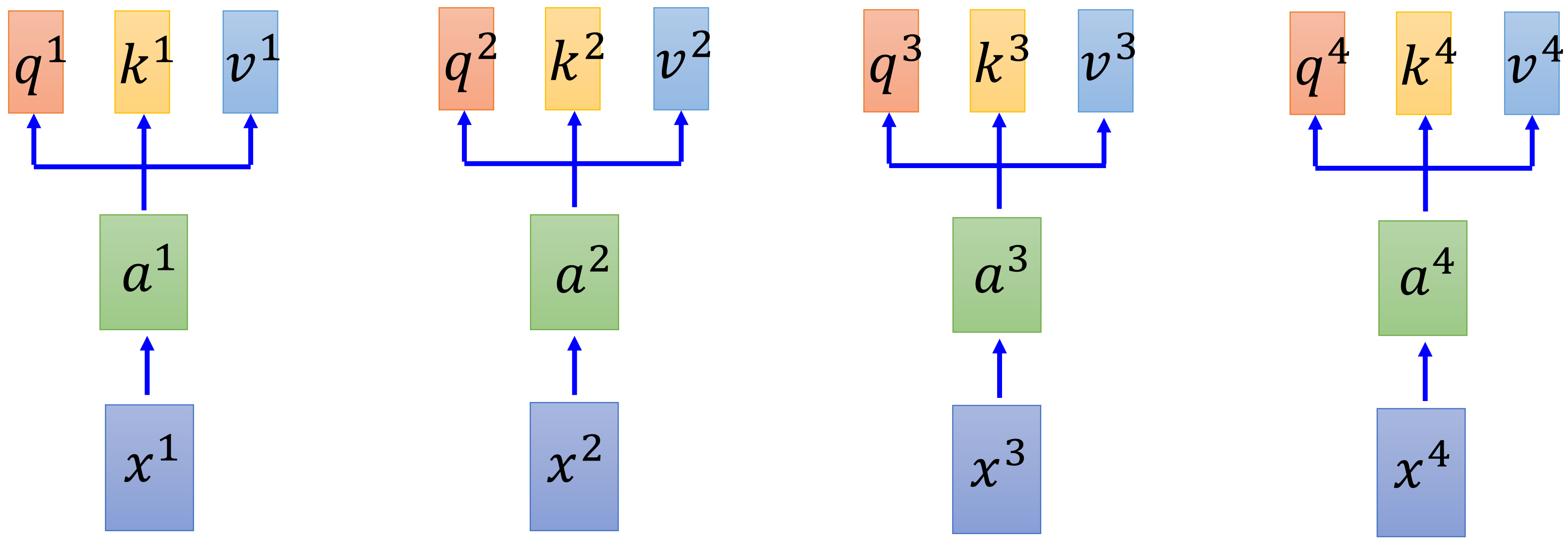
$$k^i = W^k a^i$$

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

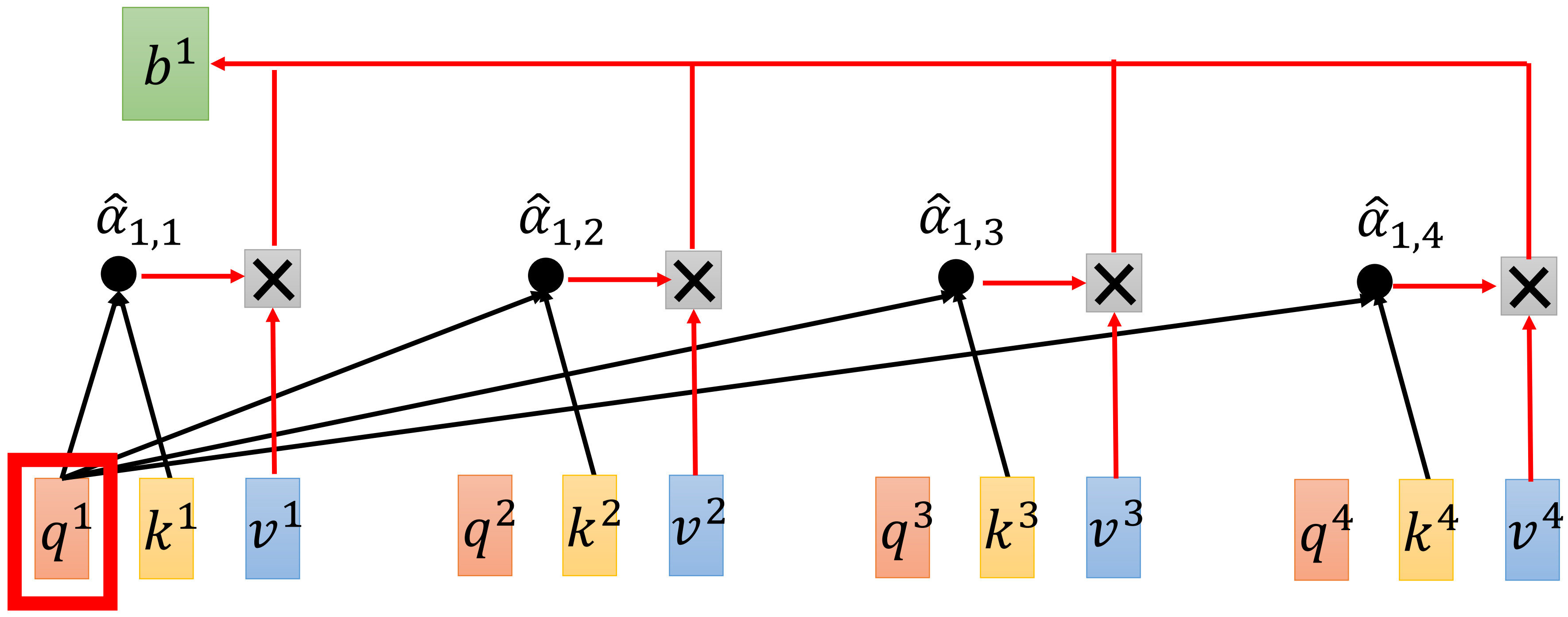
$$\underbrace{\begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix}}_{Q} = W^q \underbrace{\begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix}}_{\mathrm{I}}$$

$$\underbrace{\begin{bmatrix} k^1 & k^2 & k^3 & k^4 \end{bmatrix}}_{K} = W^k \underbrace{\begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix}}_{\mathrm{I}}$$

$$\underbrace{\begin{bmatrix} v^1 & v^2 & v^3 & v^4 \end{bmatrix}}_{V} = W^v \underbrace{\begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix}}_{\mathrm{I}}$$



Self-attention



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

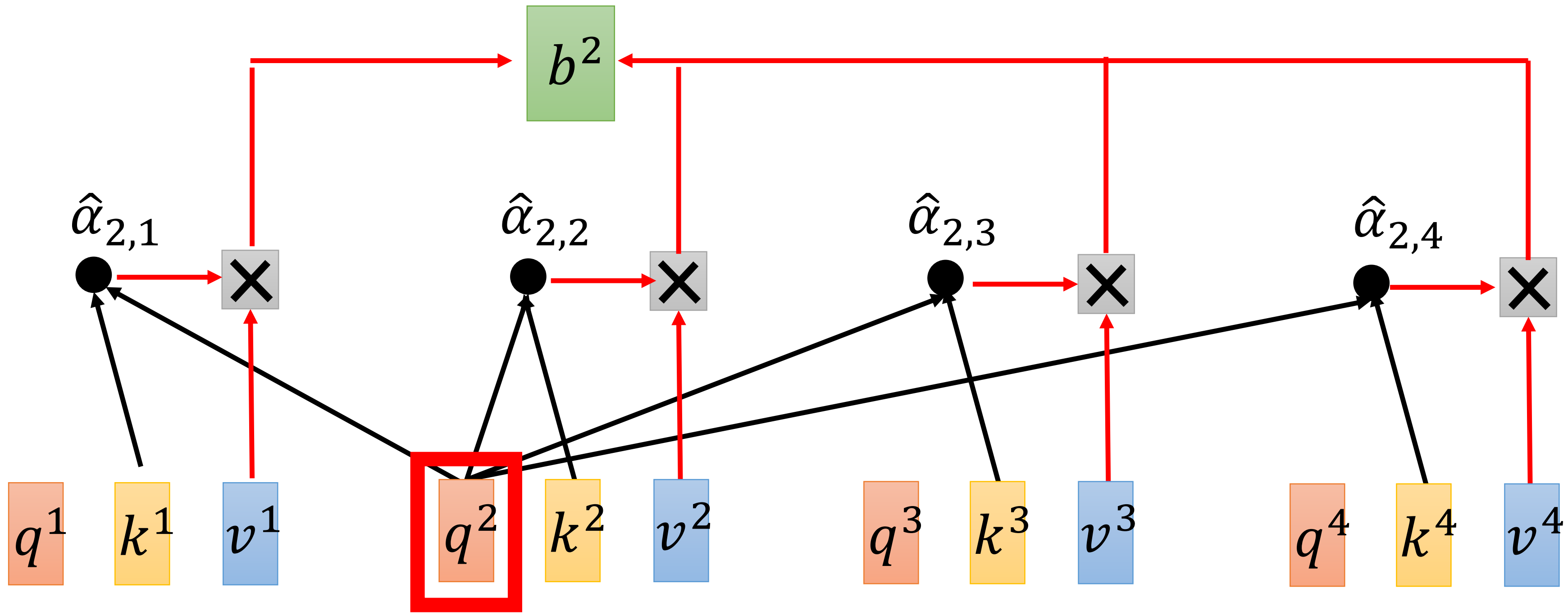
$\alpha_{1,3} = k^3 \cdot q^1$ $\quad \alpha_{1,4} = k^4 \cdot q^1$

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

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

Self-attention

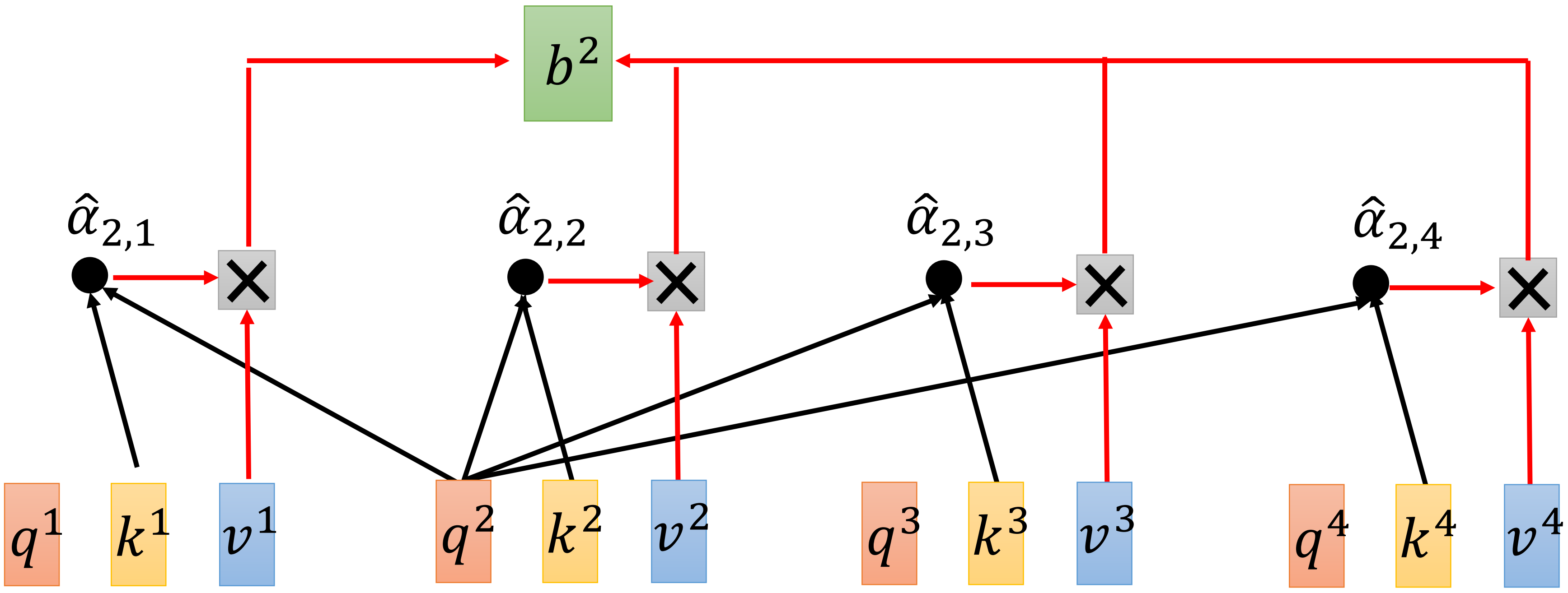
$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



$$\underbrace{\begin{matrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{matrix}}_{\hat{A}} \Leftarrow \underbrace{\begin{matrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{matrix}}_{A} = \underbrace{\begin{matrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{matrix}}_{K^T} \underbrace{\begin{matrix} q^1 & q^2 & q^3 & q^4 \end{matrix}}_{Q}$$

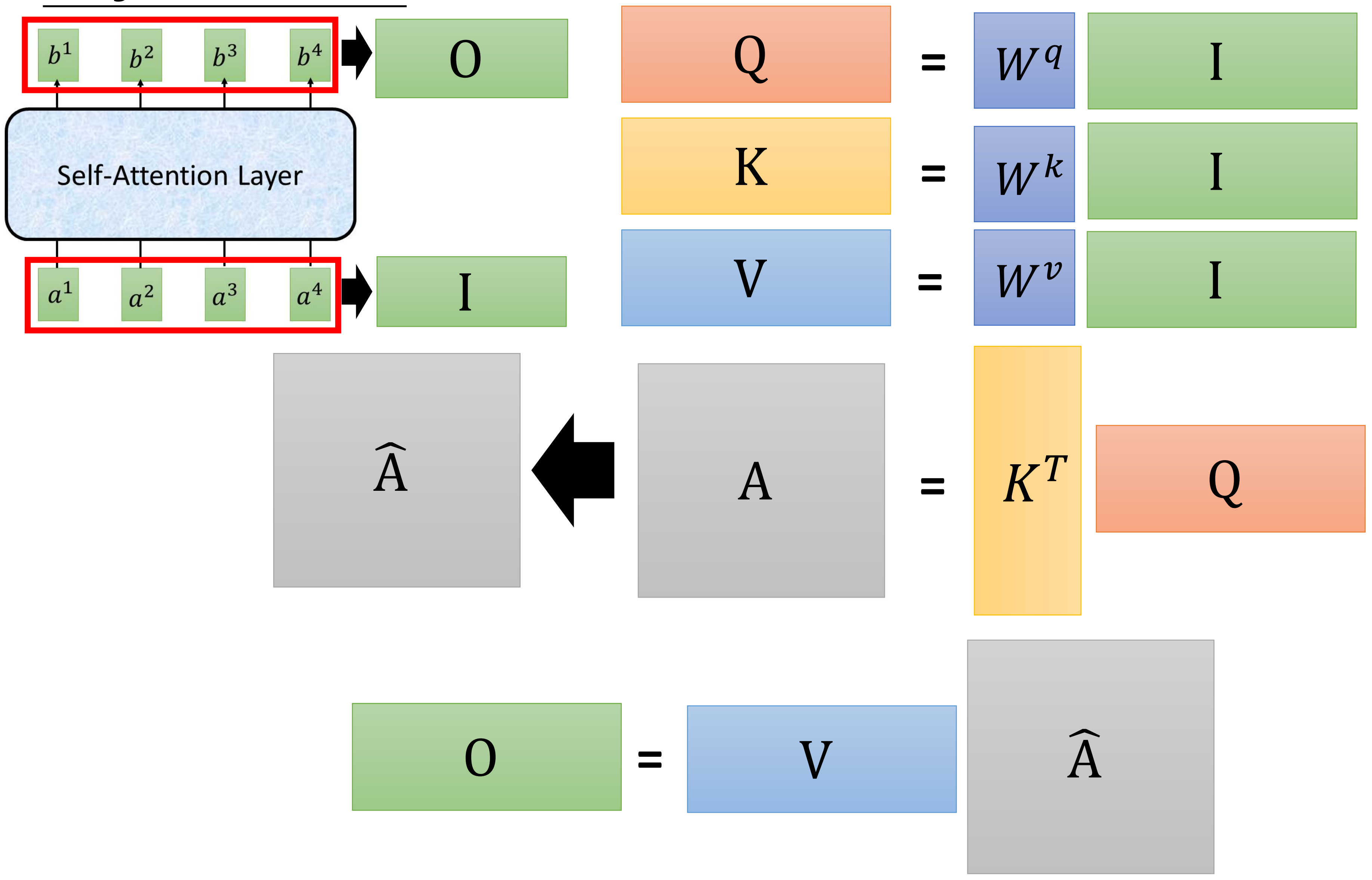
Self-attention

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



$$\underset{O}{\begin{bmatrix} b^1 & b^2 & b^3 & b^4 \end{bmatrix}} = \underset{V}{\begin{bmatrix} v^1 & v^2 & v^3 & v^4 \end{bmatrix}} \underset{\hat{A}}{\begin{bmatrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{bmatrix}}$$

Self-attention

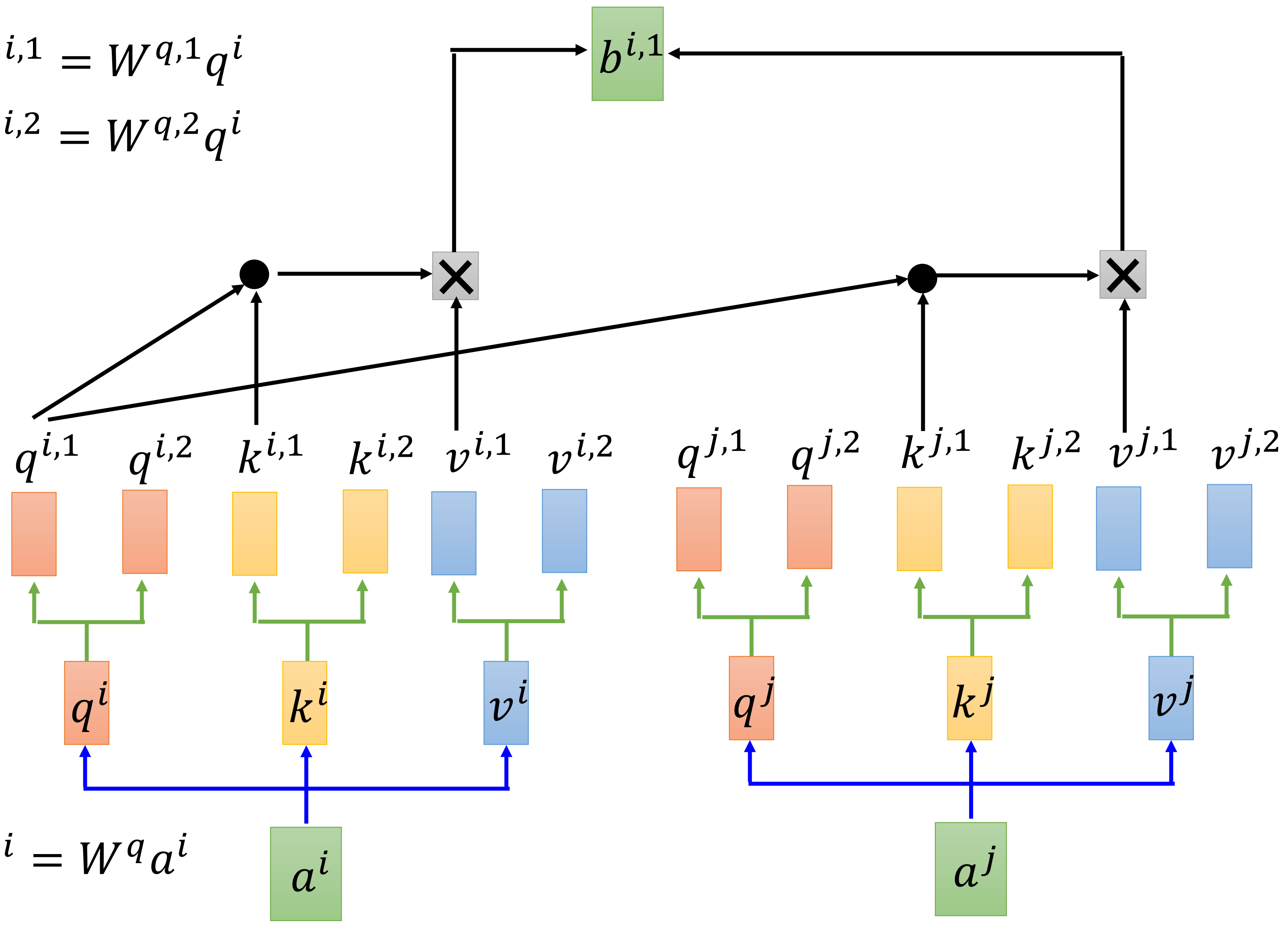


GPU can accelerate matrix multiplications

Multi-head Self-attention (2 heads as example)

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

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

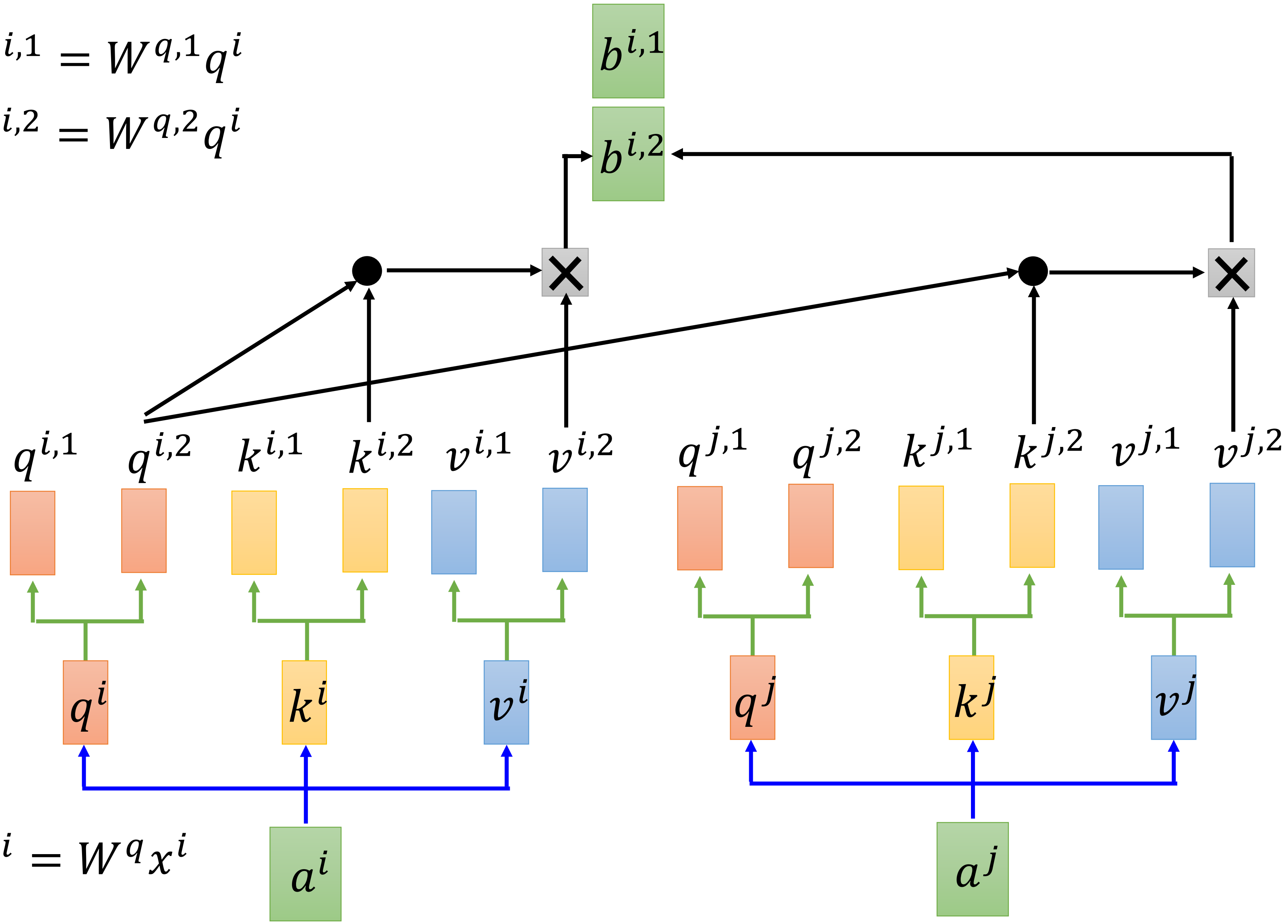


$q^i = W^q a^i$

Multi-head Self-attention (2 heads as example)

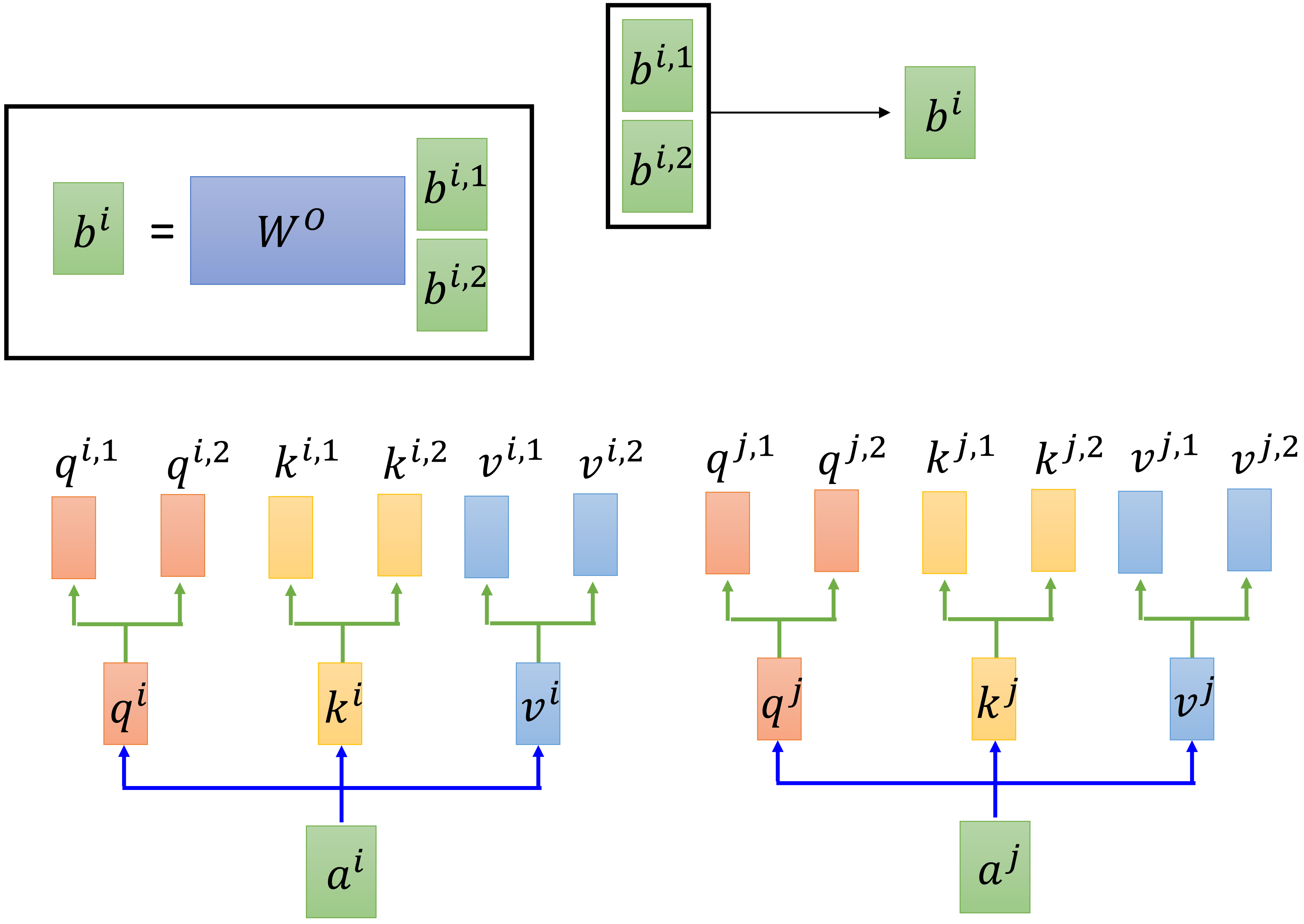
$q^{i,1} = W^{q,1} q^i$

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



$q^i = W^q x^i$

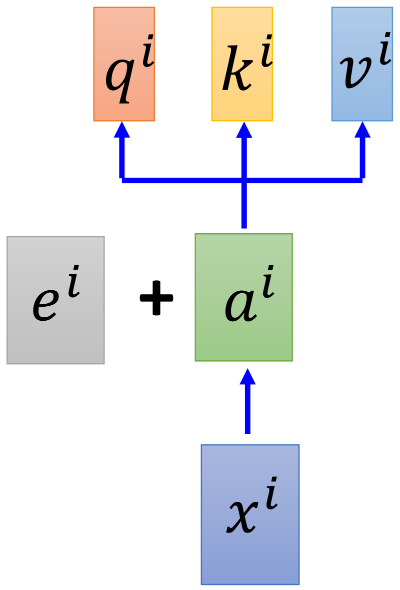
Multi-head Self-attention (2 heads as example)



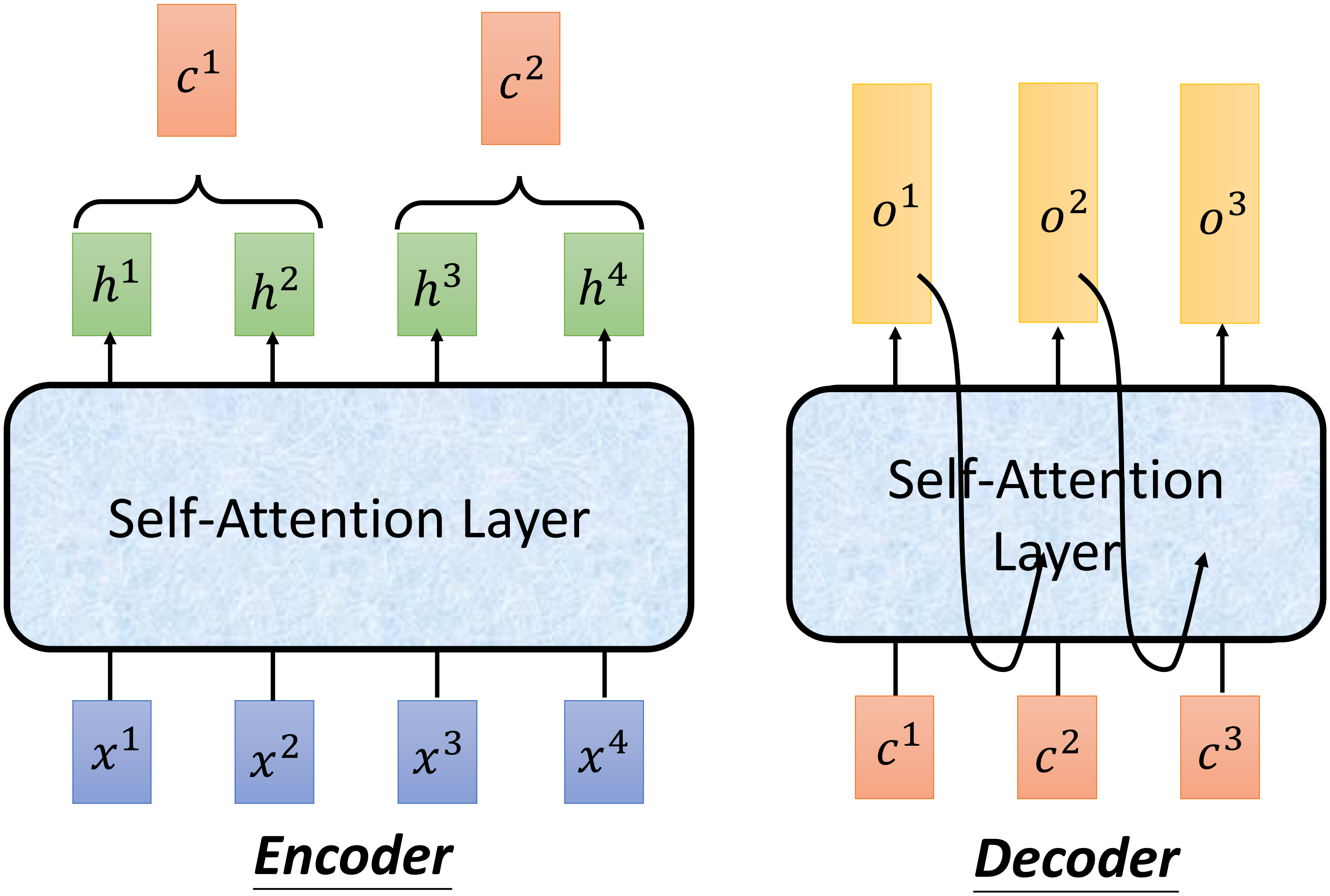
$b^i = W^O \begin{bmatrix} b^{i,1} \\ b^{i,2} \end{bmatrix}$
$b^{i,1}$
$b^{i,2}$
b^i
$q^{i,1}$ $q^{i,2}$ $k^{i,1}$ $k^{i,2}$ $v^{i,1}$ $v^{i,2}$
$q^{j,1}$ $q^{j,2}$ $k^{j,1}$ $k^{j,2}$ $v^{j,1}$ $v^{j,2}$
q^i k^i v^i
q^j k^j v^j
a^i
a^j

Positional Encoding

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

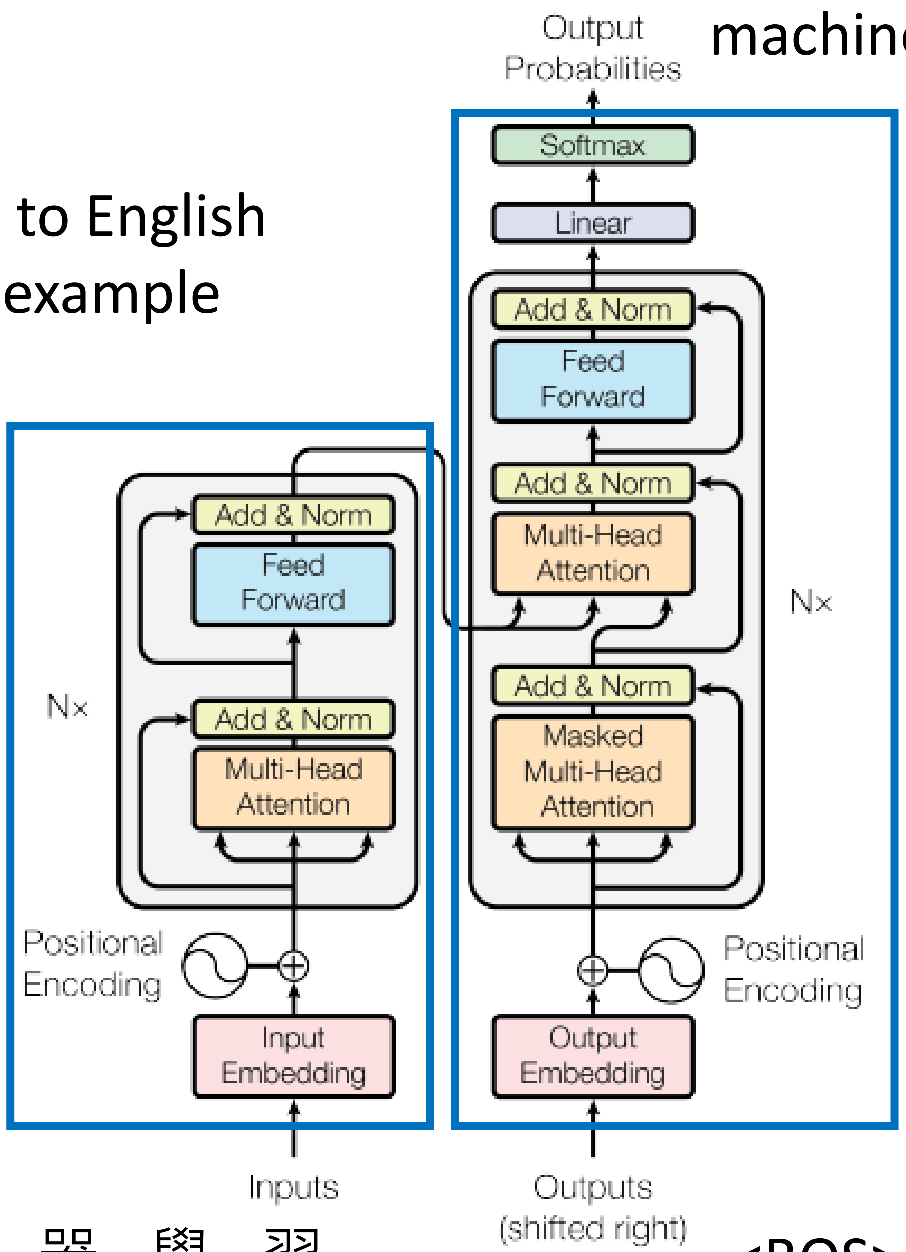


Seq2seq with Attention



Transformer

Using Chinese to English translation as example



Encoder

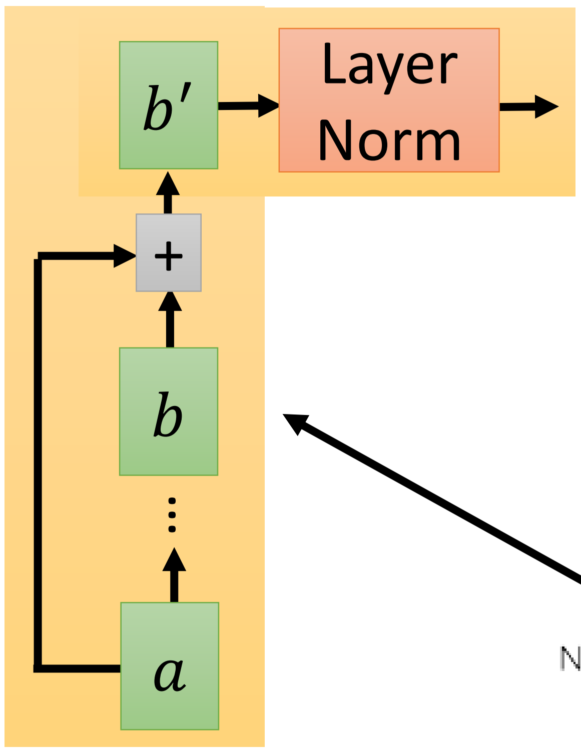
Decoder

machine learning

機 器 學 習

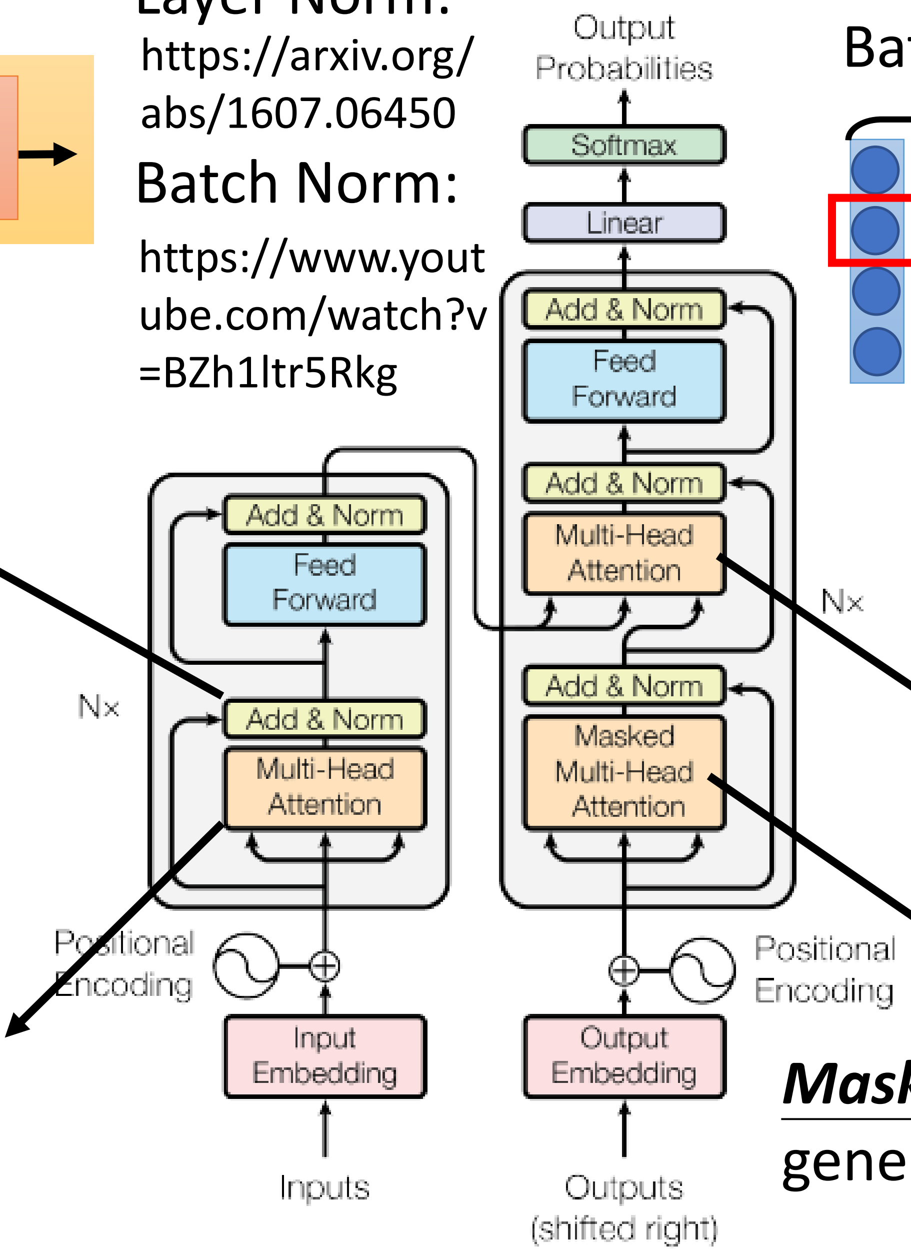
<BOS> machine

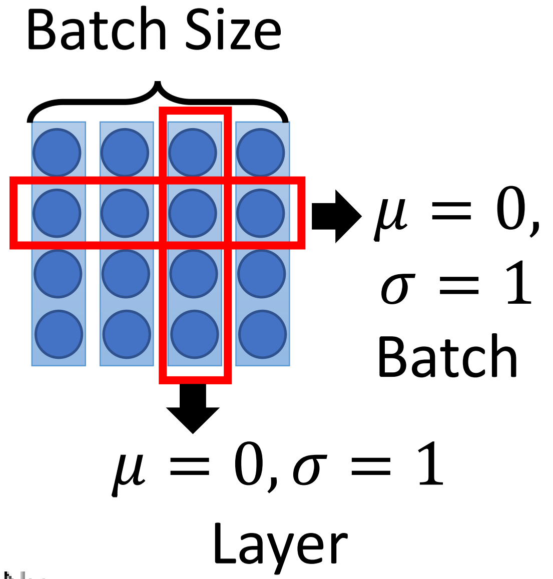
Transformer

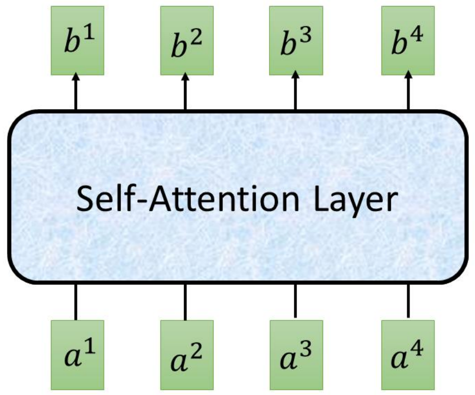


Layer Norm: https://arxiv.org/abs/1607.06450

Batch Norm: https://www.youtube.com/watch?v=BZh1ltr5Rkg







attend on the input sequence

Masked: attend on the generated sequence

Outline

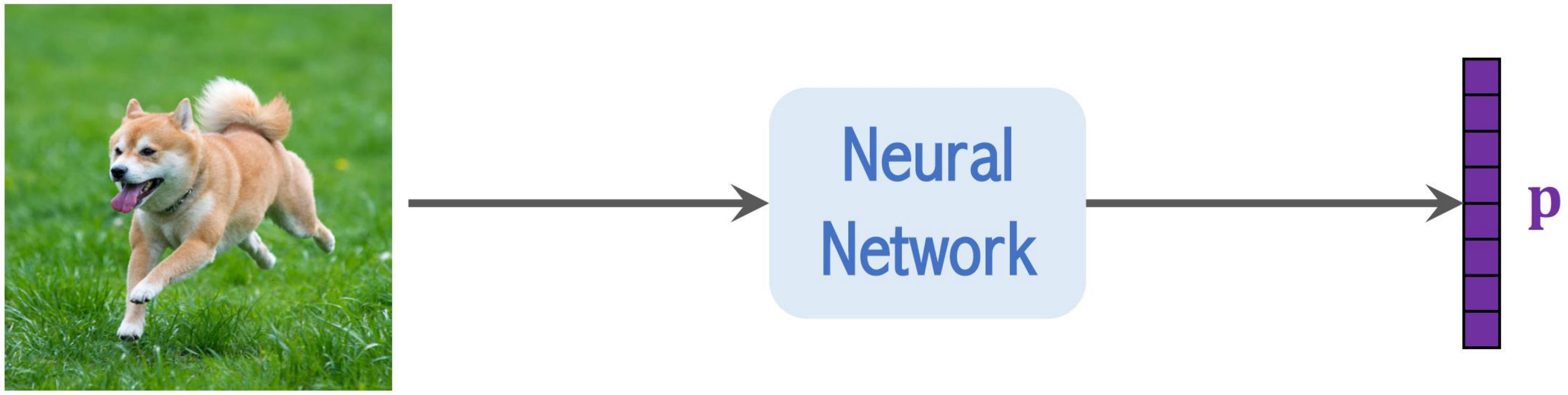
- Transformer
- ViT

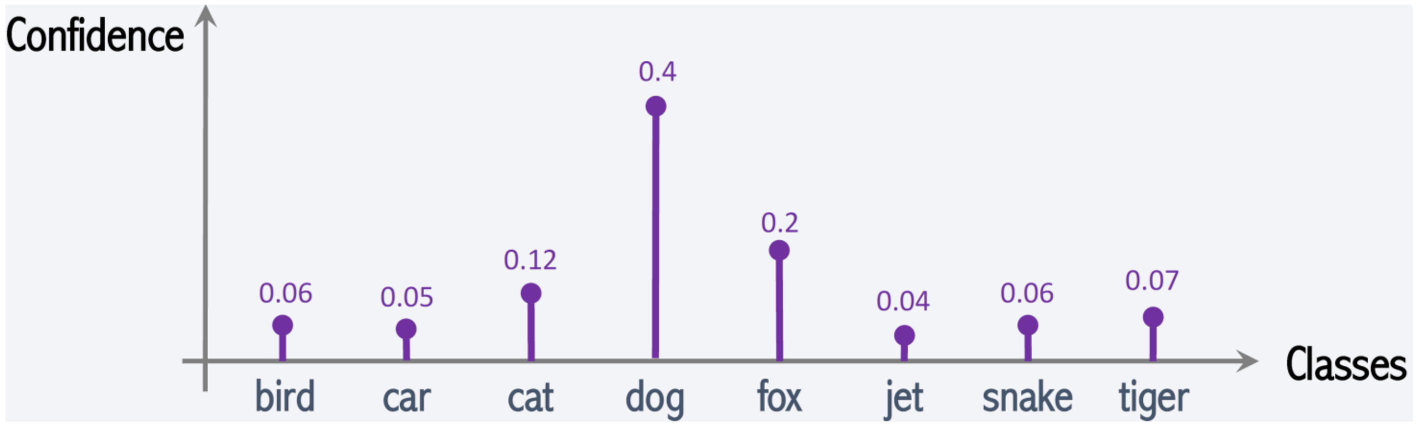
Vision Transformer (ViT)

[Dosovitskiy et al., ICLR'21]

- Self-attention-based architectures, e.g., Transformers, have become the model of choice in natural language processing
- CNNs, e.g., ResNet, were the state-of-the-art models for object recognition
- Vision transformer (ViT) beats CNNs by a small margin, if the dataset for pre-training is sufficiently large
- ViT is developed mainly based on Transformer
  - Encoder: linear embedding, multi-head attention, layer normalization, …
  - Position encoding

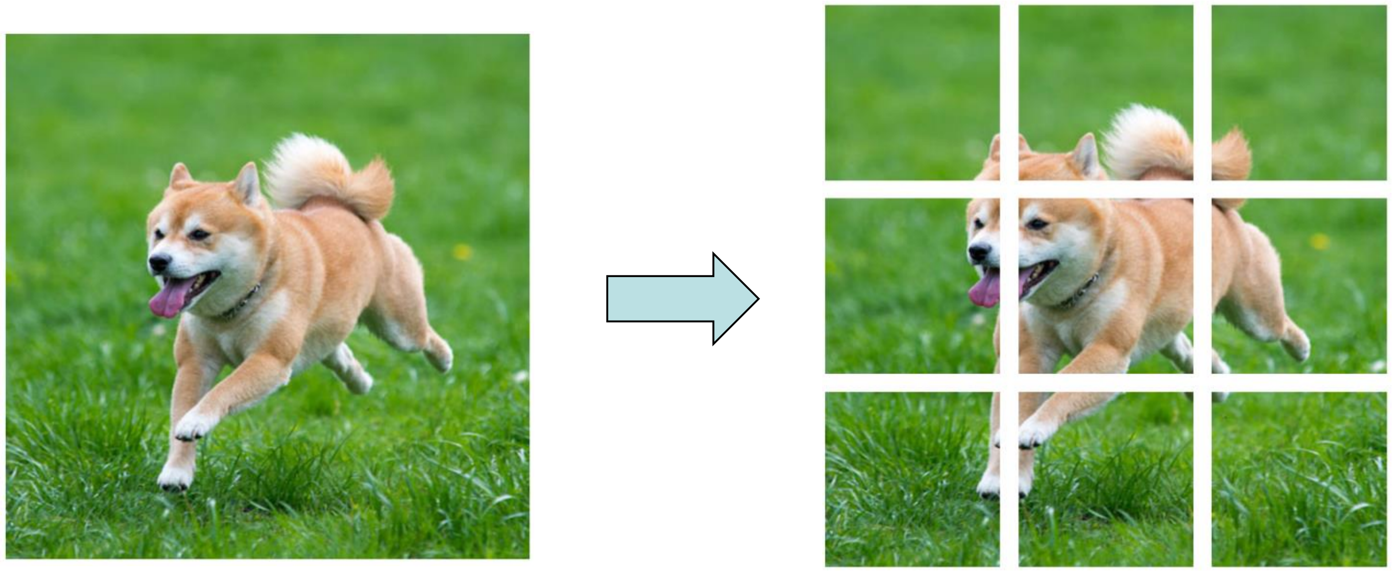
Object recognition





An image as a set of patches

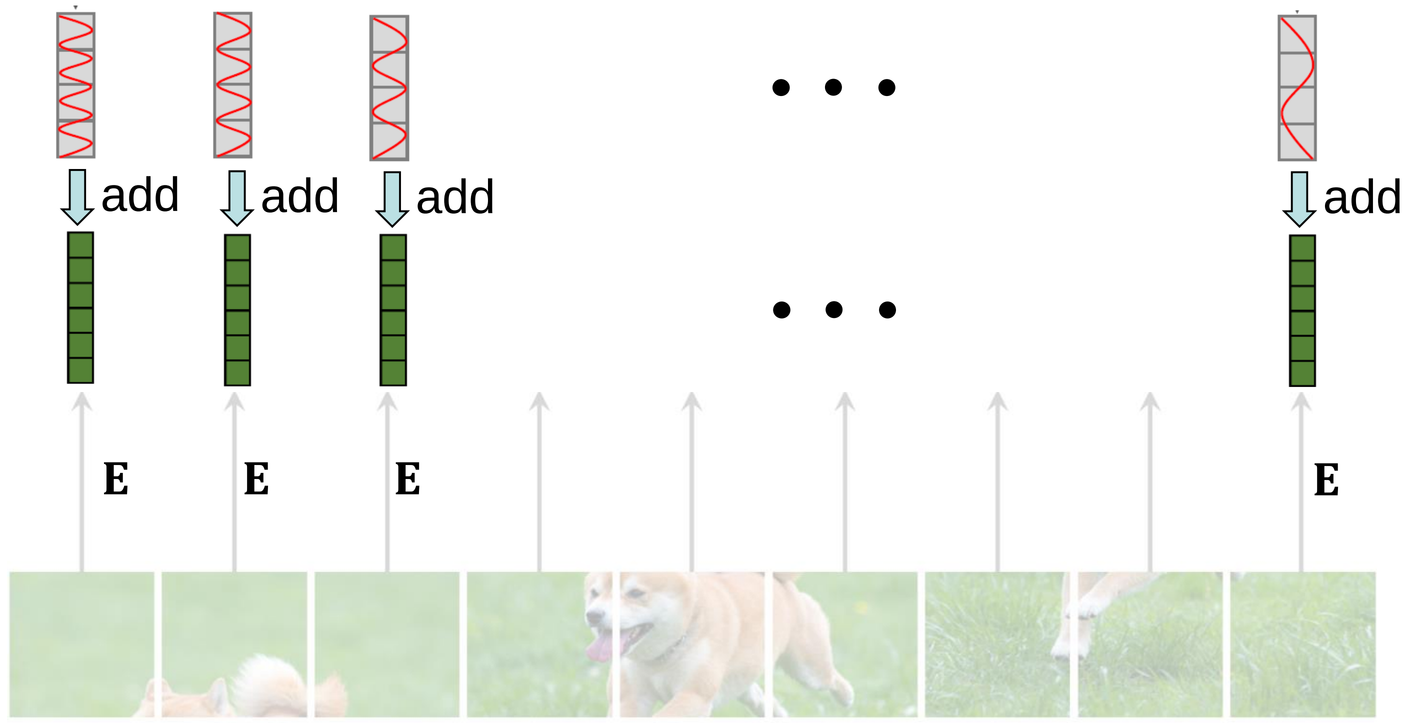
- To apply Transformer to object recognition with minor modifications, we split an image into patches
- Patches are specified by the user
  - Overlapping or non-overlapping
  - Patch size



國立交通大學
National Chiao Tung University

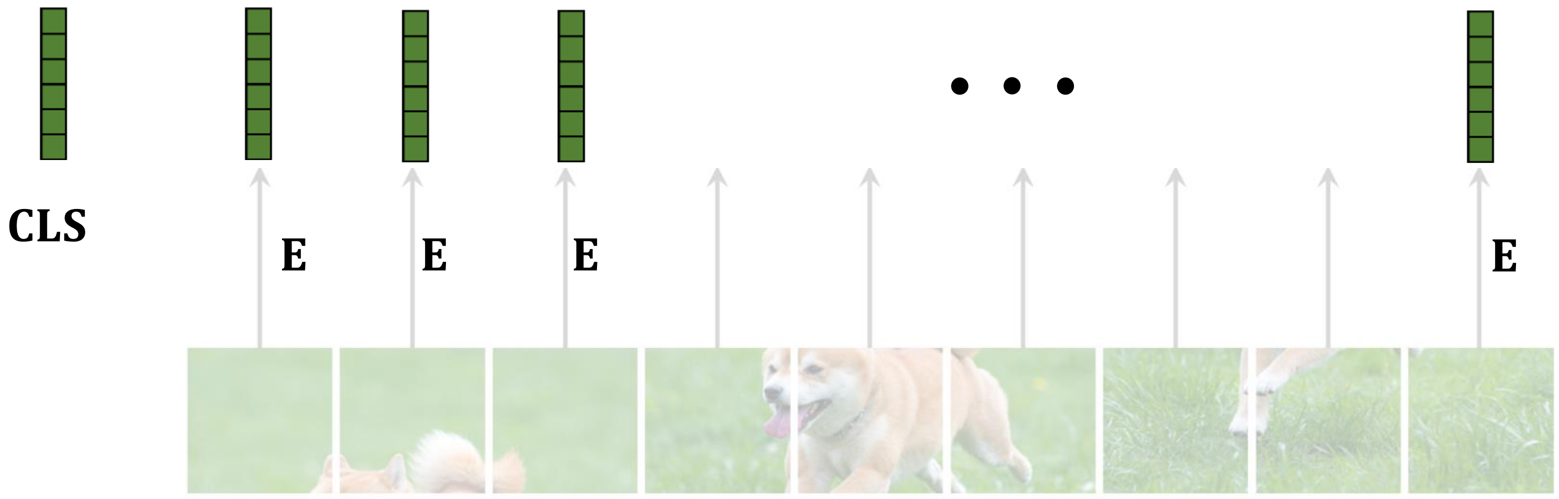
Linear embeddings and positional encoding of patches

- We embed each patch by linear projection $\mathbf{E}$
- 1D learnable positional encoding is used and added to linear embedding



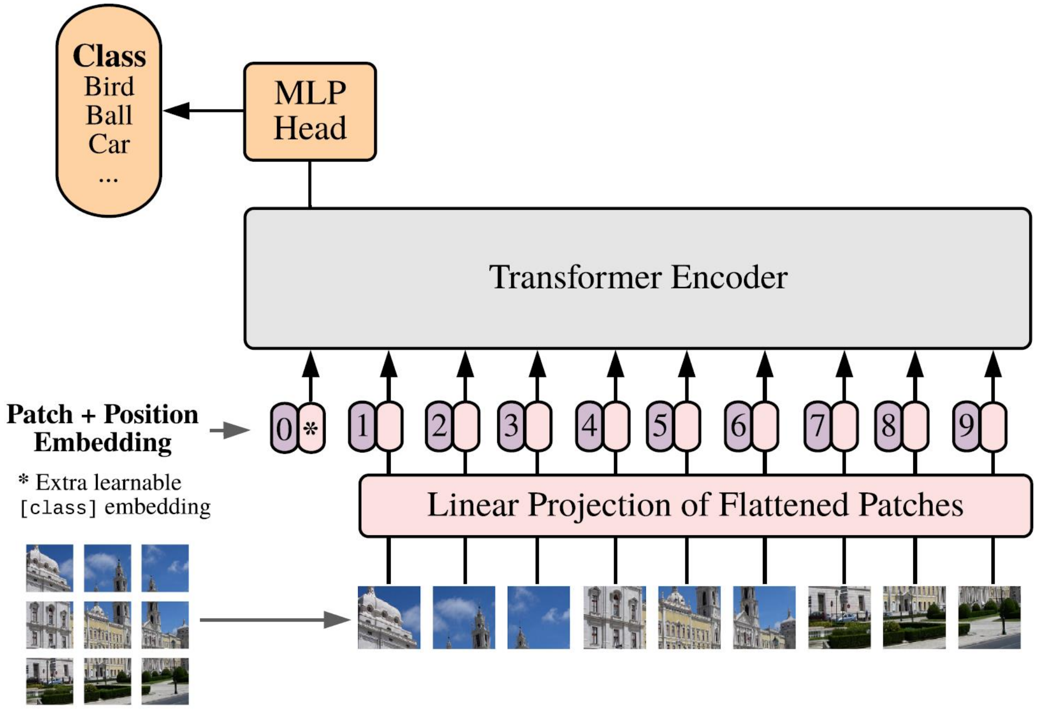
Class token

- A special token, called class token, is included
- Class token does not belong to any patch
- It is represented by a vector that is learnable
- During N blocks, it gathers information from other tokens, and its output vector is used to derive a classifier



Overview

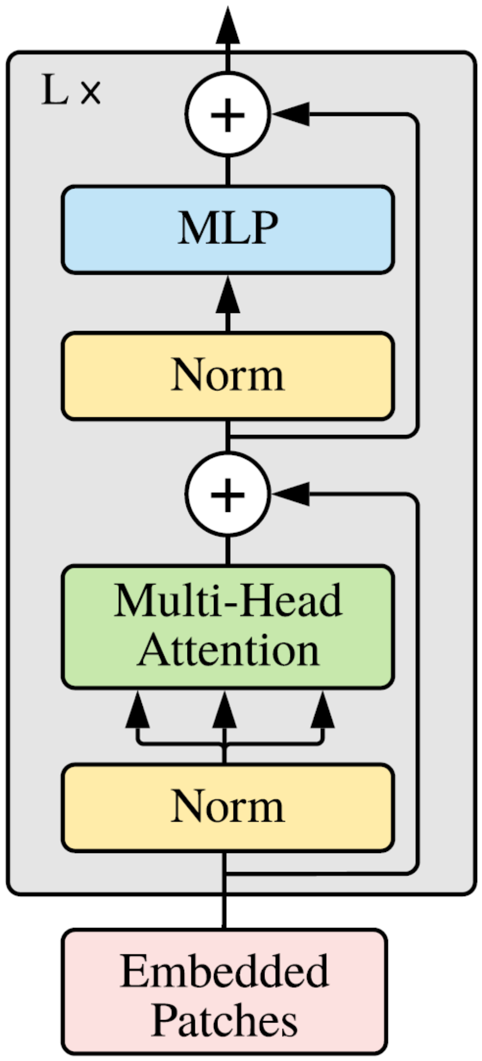
1. Split an image into fixed-size patches (16 x 16), linearly embed each of them, and then add CLS token (learnable)
2. Add positional encoding
3. Transformer encoder operation
4. Use [cls] token to do the classification



Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
0 * 1 2 3 4 5 6 7 8 9
* Extra learnable [class] embedding
Linear Projection of Flattened Patches

國立交通大學
National Chiao Tung University

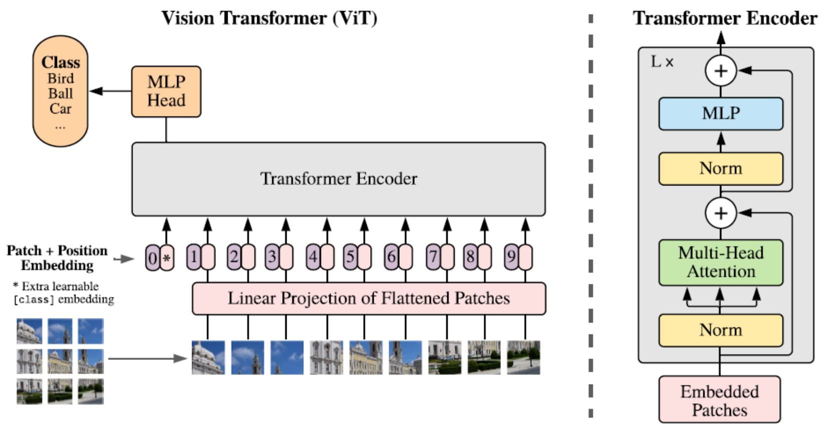
Transformer encoder

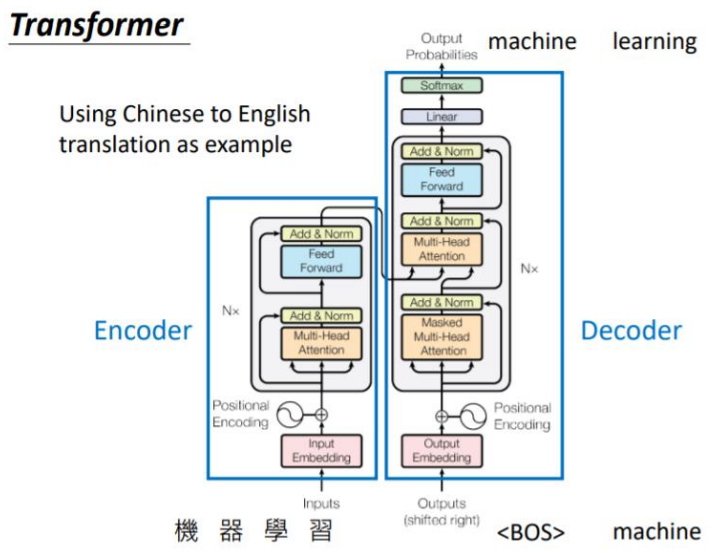


$$\mathbf{z}_0 = [\mathbf{x}_\text{class};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos},$$
$$\mathbf{z'}_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1},$$
$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell,$$
$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0)$$

ViT vs. Transformer

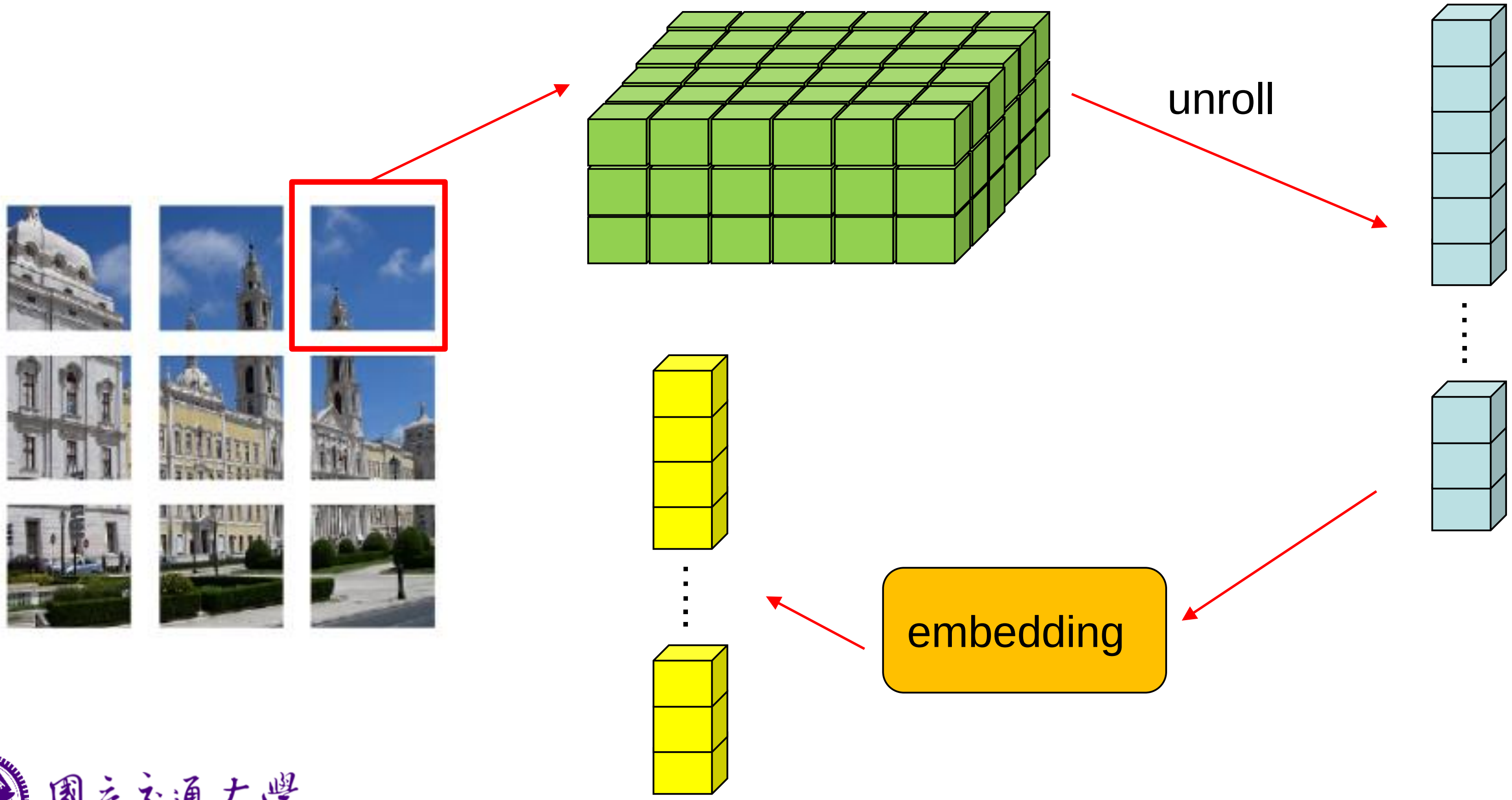
1. VIT only uses the encoder part
2. Split image into patches and embed them as tokens.
   (Word vectors as tokens in Transformer)
3. Add one additional learnable token: [CLS] token
   (It servers as conceptual summary about the whole image tokens)
4. Different positional encoding methods





ViT vs. Transformer

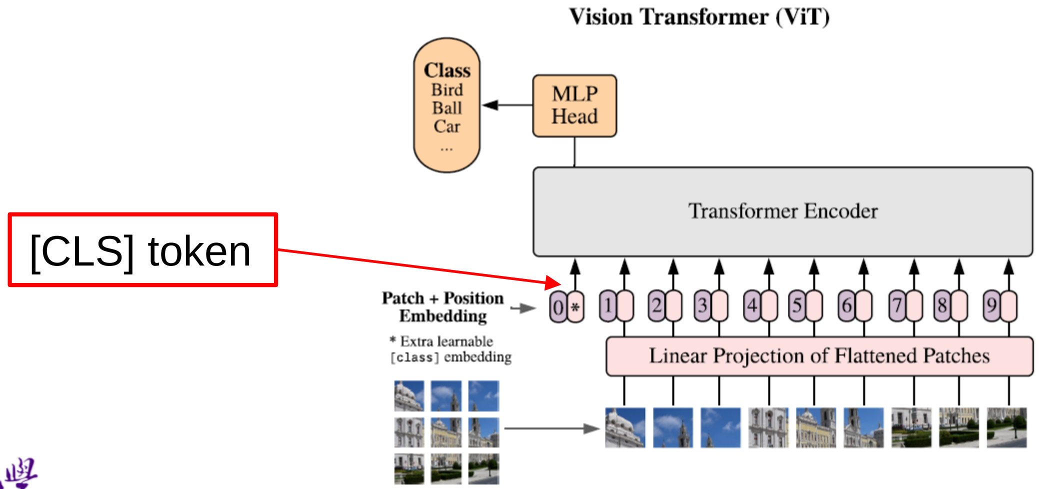
- 2. Split an image into patches and embed them as tokens



國立交通大學
National Chiao Tung University

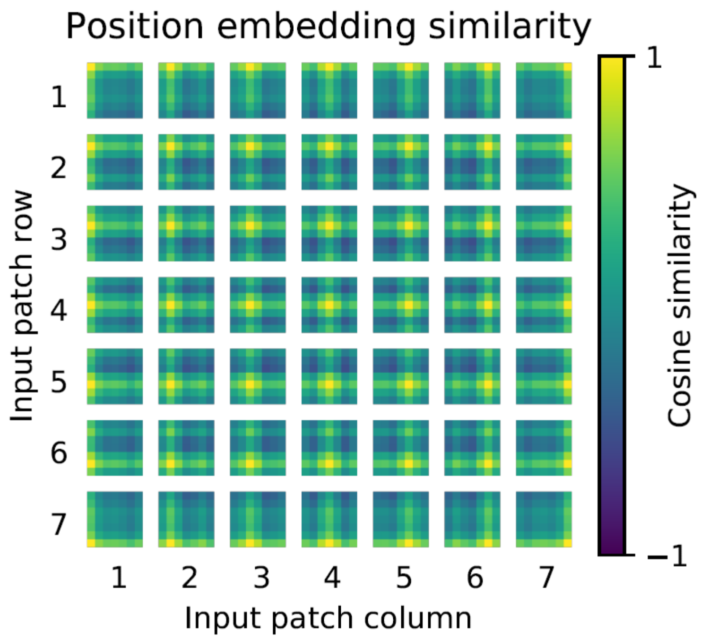
ViT vs. Transformer

- 3. Add one additional learnable token: [CLS] token
  - Original transformer is designed for NLP (sec2sec task), so its output is also a sequence (a series of word vectors).
  - ViT is used for classification task (only 1 output vector). It would be unfair to choose one particular token for image prediction. Hence, one additional learnable token is included as conceptual summary about the whole image tokens and for final prediction



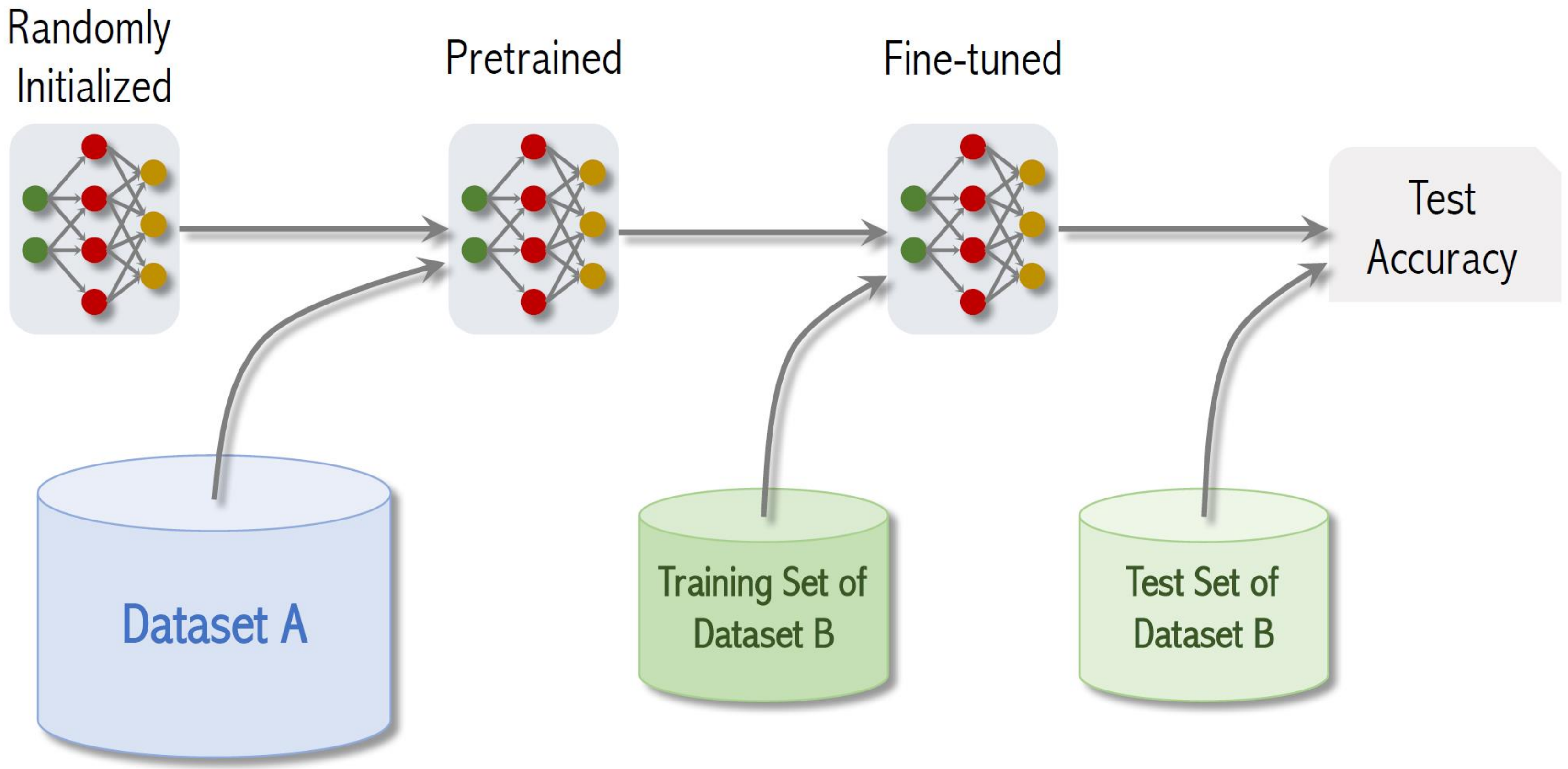
ViT vs. Transformer

- 4. Different positional encoding methods
  - Positional encoding in the original transformer is given by using sine and cosine functions
  - Positional encoding in ViT is learnable
    - Cosine similarity among the learned position encoding



国立交通大學
National Chiao Tung University

Pre-training, fine-tuning, and testing



Experiments: Network configurations

- Three variants of ViT of different sizes

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

  - ViT-L/16: the Large variant with 16 x 16 input patch size
  - ViT-H/14: the Huge variant with 14 x 14 input patch size

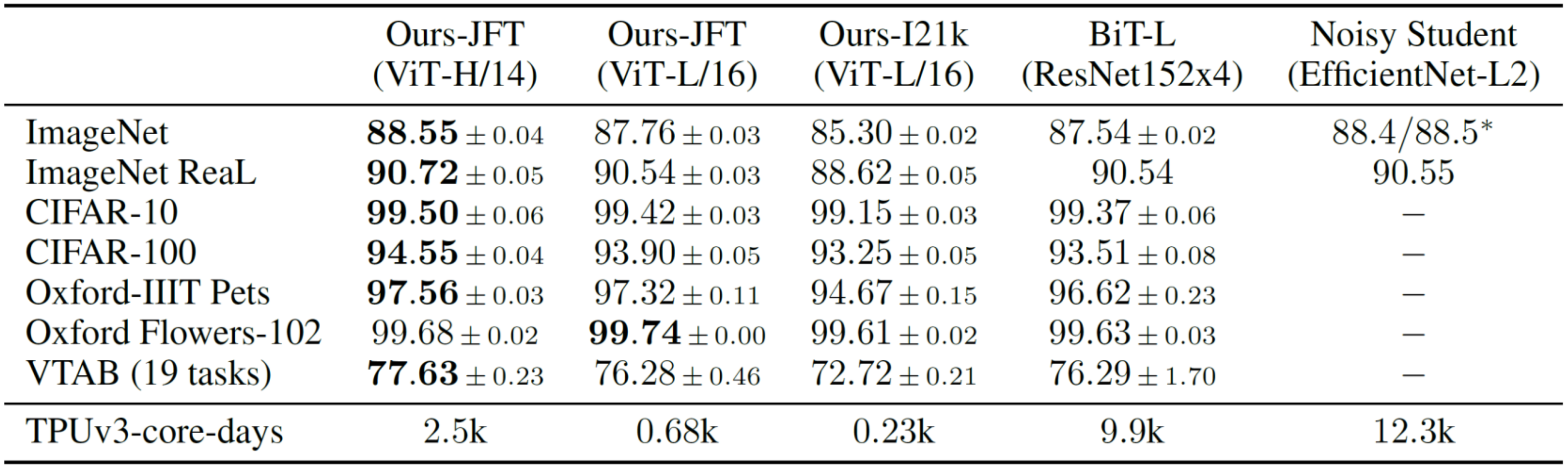
Competing methods and datasets

- Competing methods
  - BiT (Big Transfer): A variant of ResNet
  - Noisy Student: A variant of EfficientNet

- Datasets for pre-training
  - ImageNet: 1.3M images of 1K classes (small)
  - ImageNet-21K: 14M images of 21K classes (medium)
  - JFT: 300M images of 18K classes (large)

Experimental results: Comparison with SoTA

- With the model pre-trained on ImageNet-21K, ViT is comparable with (or slightly underperforms) ResNet
- With the model pre-trained on JFT, ViT slightly outperforms ResNet, while taking less computational resources for pre-training

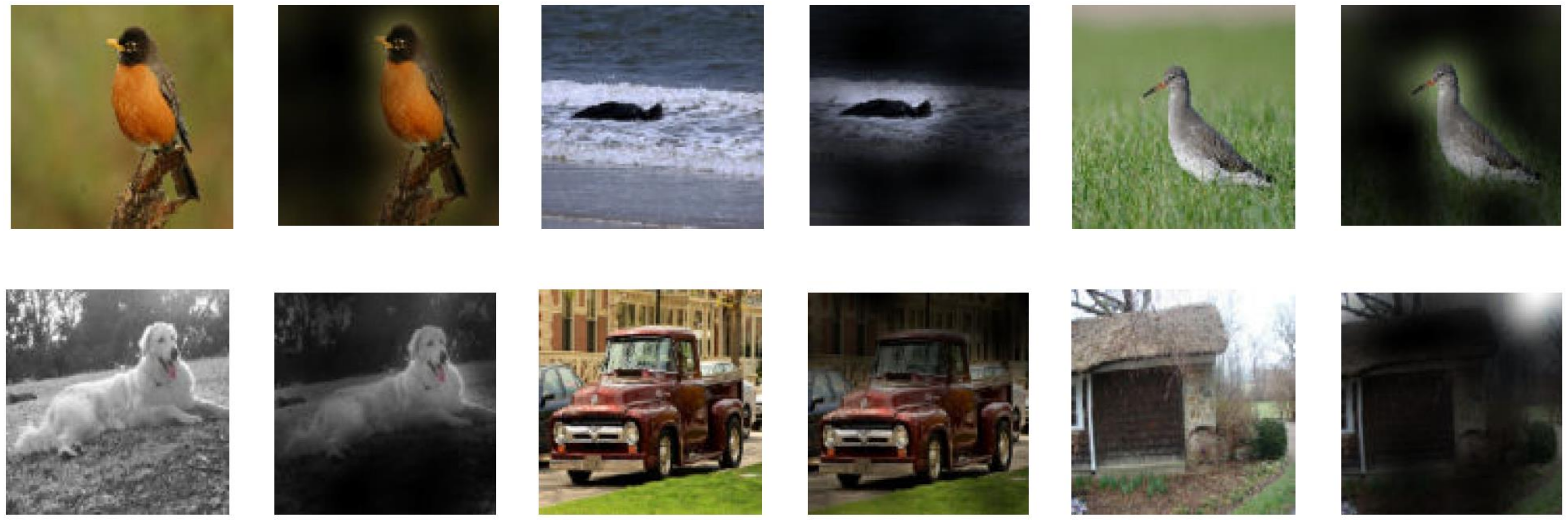


	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 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

國立交通大學
National Chiao Tung University

Attention derived by ViT

- Attention maps are inferred by using Attention Rollout [Abnar and Zuidema, ACL'20]
- ViT attends to regions semantically relevant to classification



國立交通大學
National Chiao Tung University

References

- Transformer
  - A. Vaswani et al.: Attention is All you Need, in NIPS 2017.
- ViT
  - A. Dosovitskiy et al.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, in ICLR 2021.

Thank You for Your Attention!

Yen-Yu Lin (林彥宇)

Email: lin@cs.nctu.edu.tw

URL: https://www.cs.nctu.edu.tw/members/detail/lin

國立交通大學
National Chiao Tung University