ELEC5307:
Self-attention and Transformer

Background

• Transformer, a non-convolutional architecture, has been widely adopted in NLP, computer vision, and speech processing.

• Compared with CNN:

- Transformers powerfully model long-range interactions in a computationally-efficient manner.
- The learned representation of relationships is more general and robust than the local patterns from convolution modules

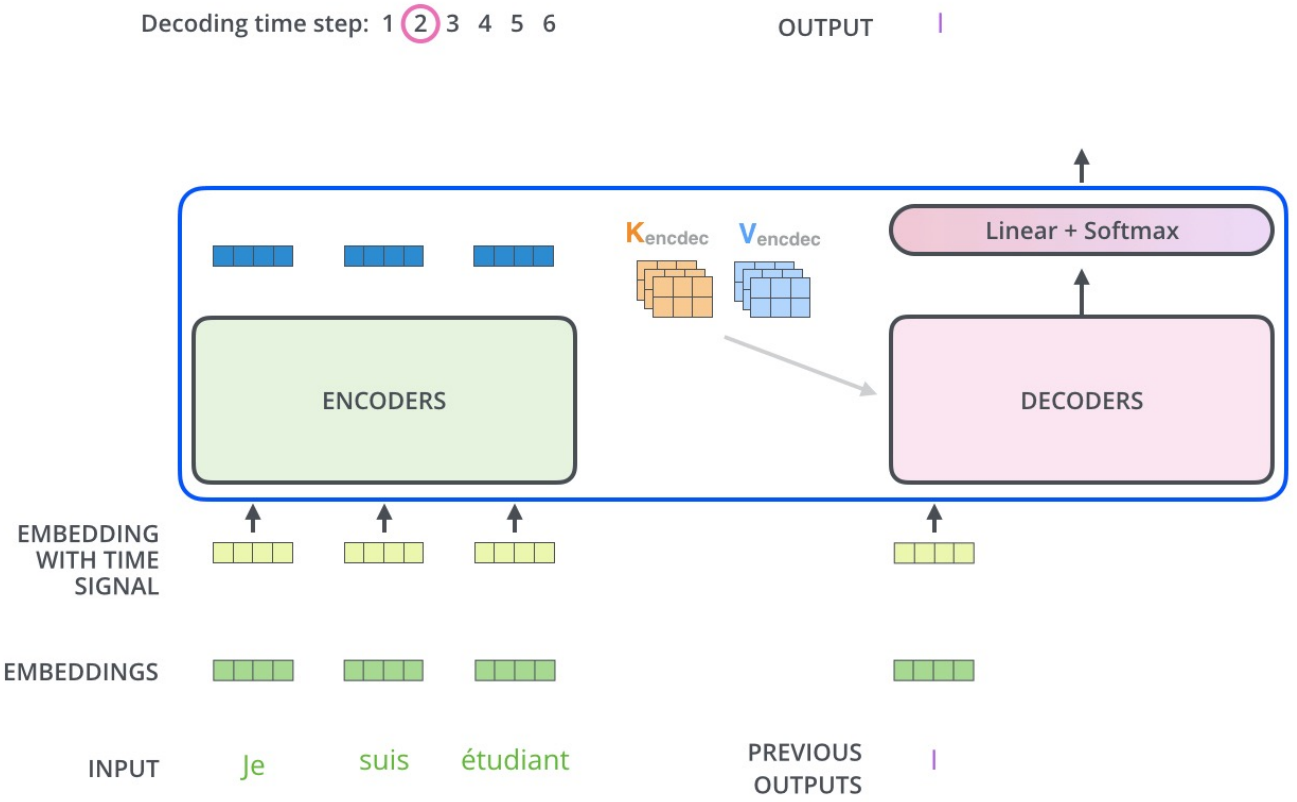
Outline

- Introduction about vanilla Transformer (including self-attention mechanism)

- Transformer used in visual recognition
  - Vision Transformer (ViT) for image classification
  - Swin Transformer for image classification
  - DETR for object detection
  - DETR for panoptic segmentation

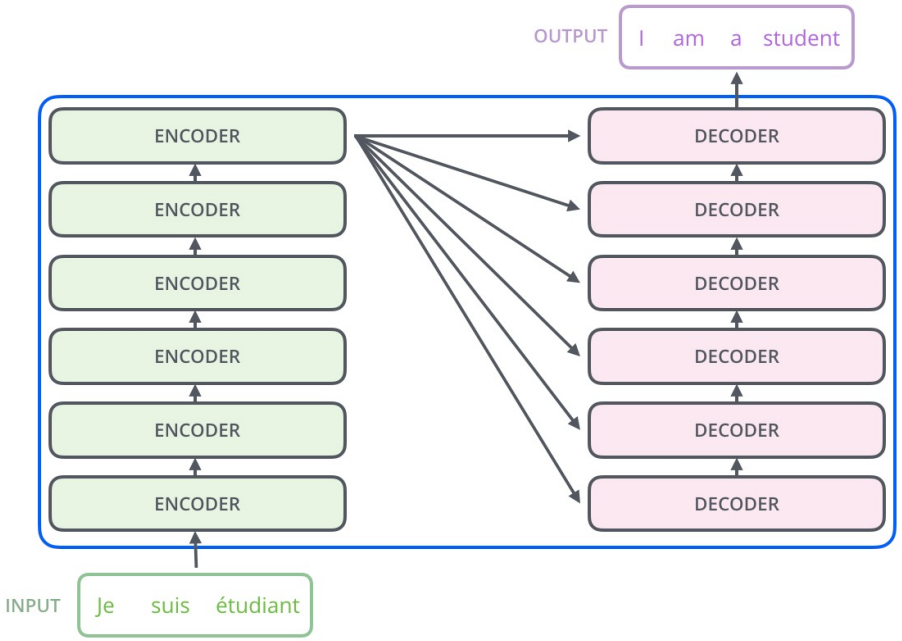
Transformer (Overview)

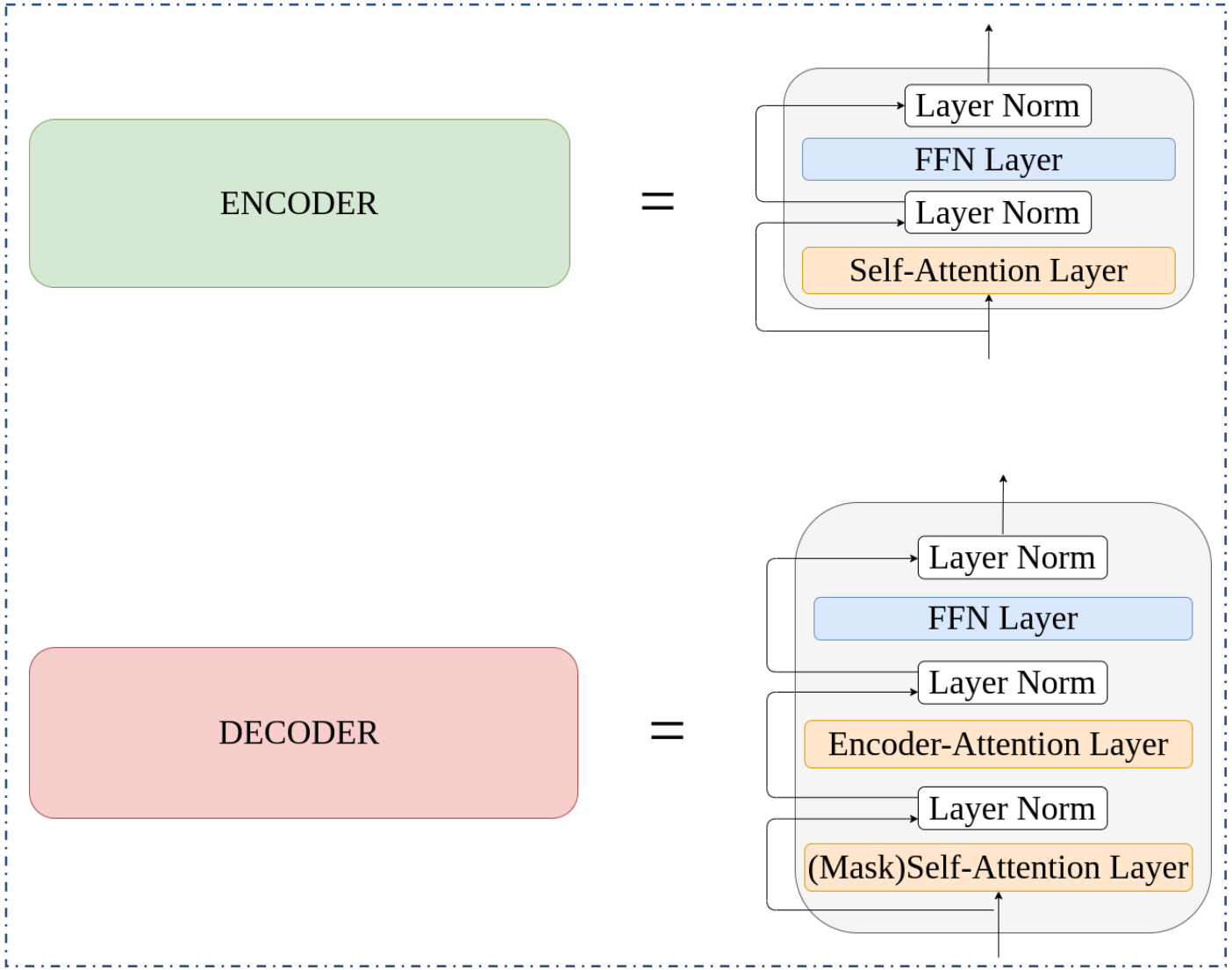
- First proposed by: Attention is all you need for machine translation task (seq2seq task)
- Published on NIPS 2017 by Google Research, Brain team
- Encoder - Decoder structure



Transformer (Overview)

- Multiple encoder and decoder are stacked together
- Components: **Self-attention layer,** Layer Norm, Feed Forward layer(linear layer)





Transformer (Self-Attention)

Sentence I:

“**The animal didn't cross the street because it was too tired**.”

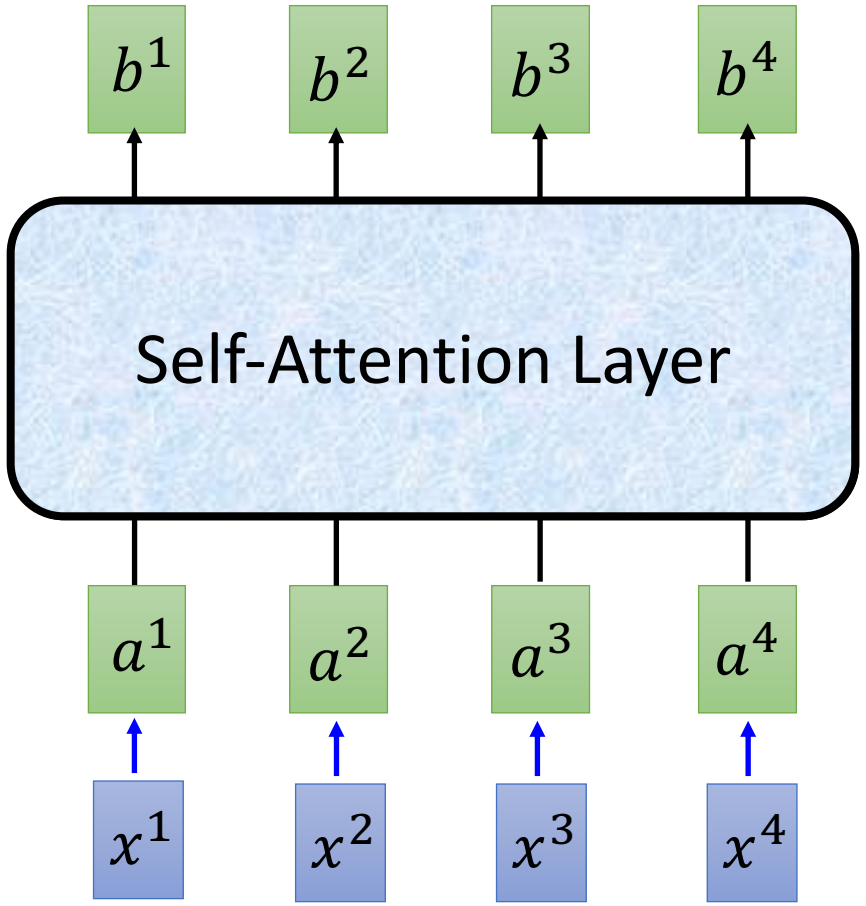
Sentence II:

“**The animal didn't cross the street because it was too wide**.”

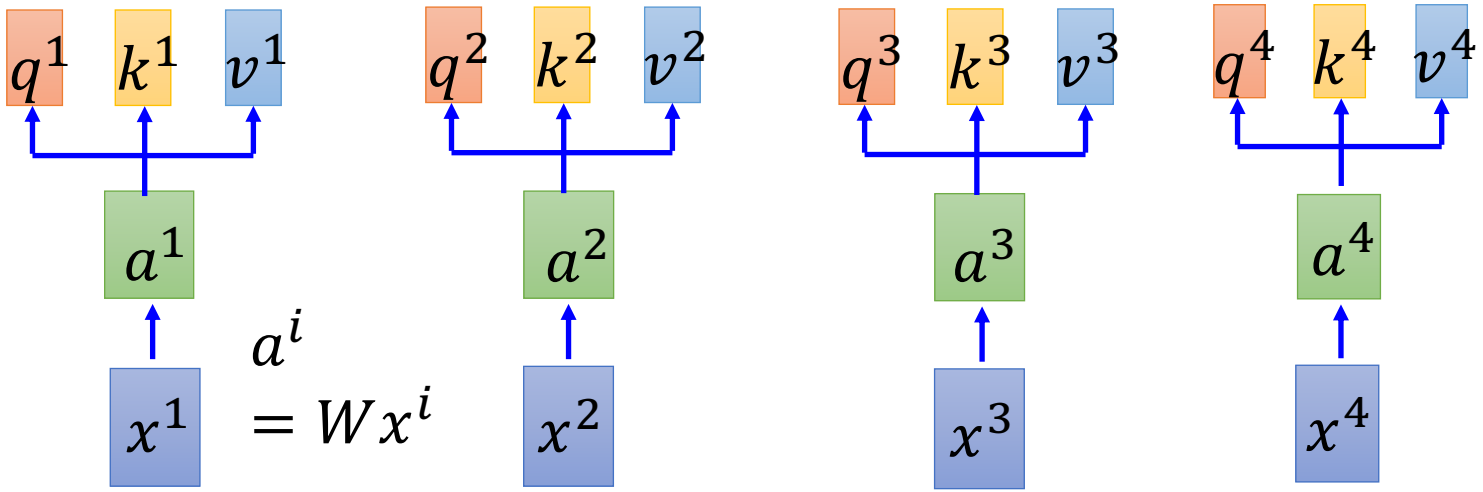
What does the word “it” refer to in these two sentences?

Self-attention is an attention mechanism relating different positions of a single sequence to compute a representation of the sequence.

Transformer (Self-Attention)



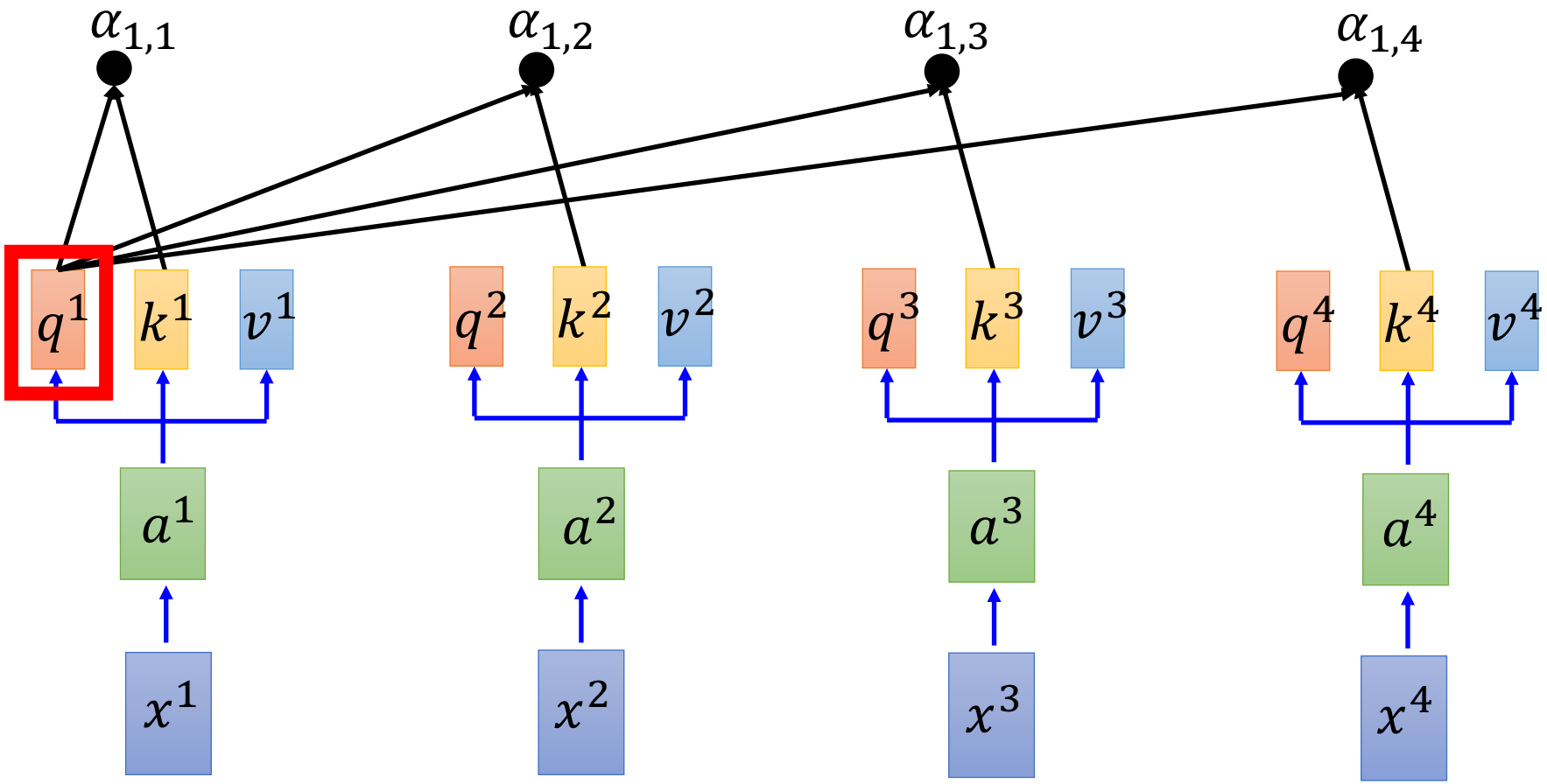
- q: query (to match others) $q^i = W^q a^i$
- k: key (to be matched) $k^i = W^k a^i$
- v: information to be extracted $v^i = W^v a^i$



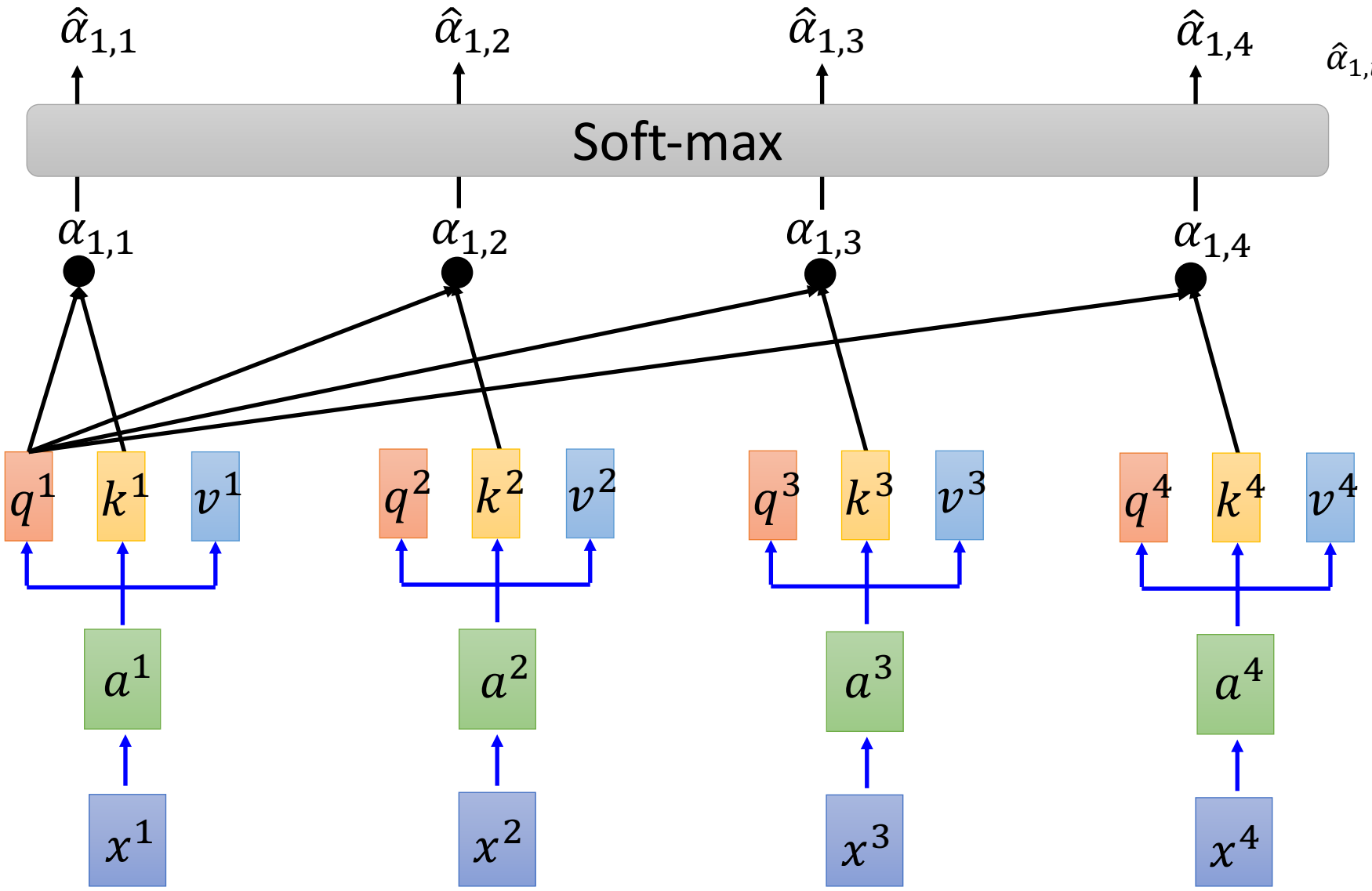
Transformer (Self-Attention)

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

d is the dim of q and k



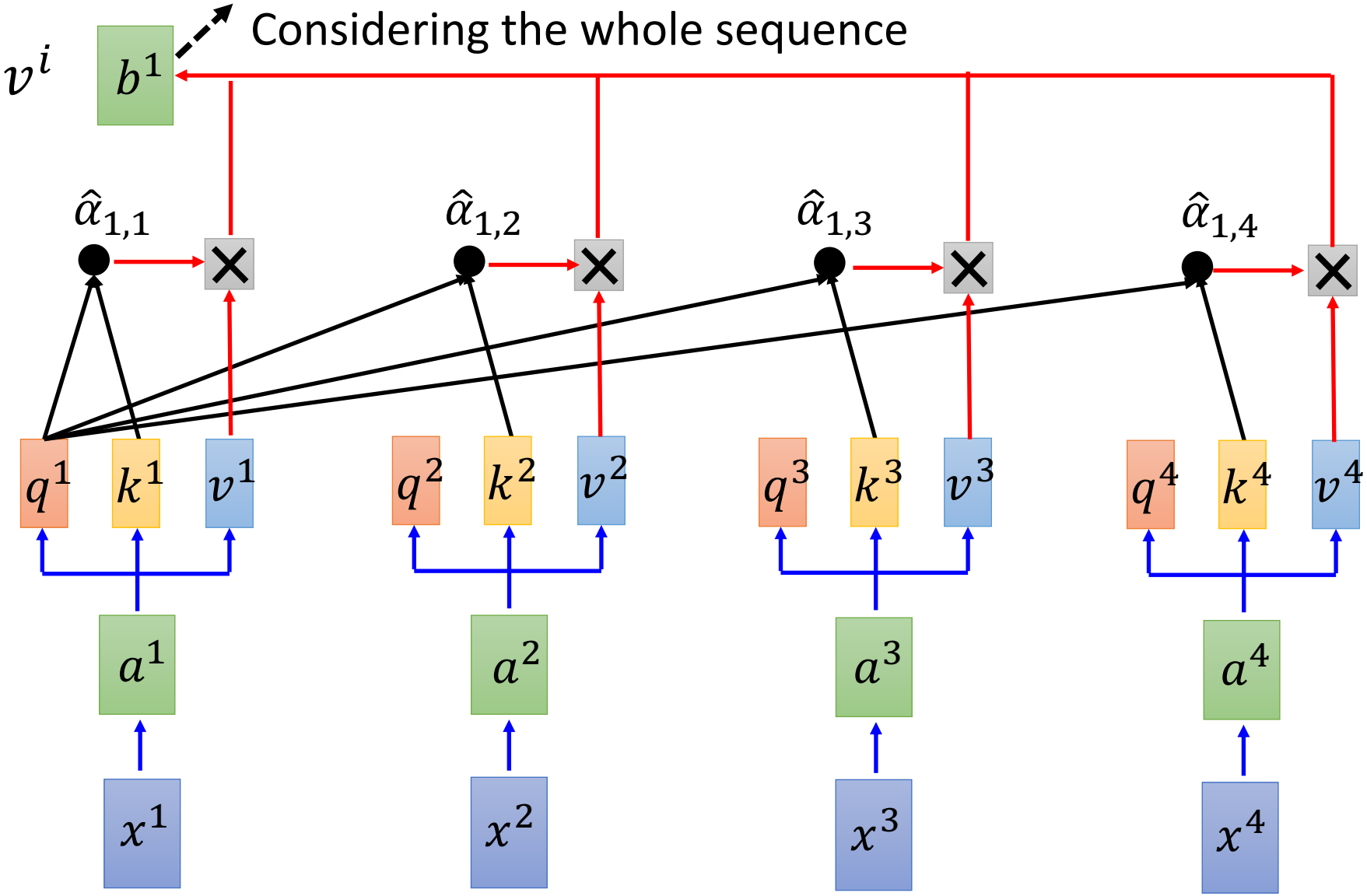
Transformer (Self-Attention)



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

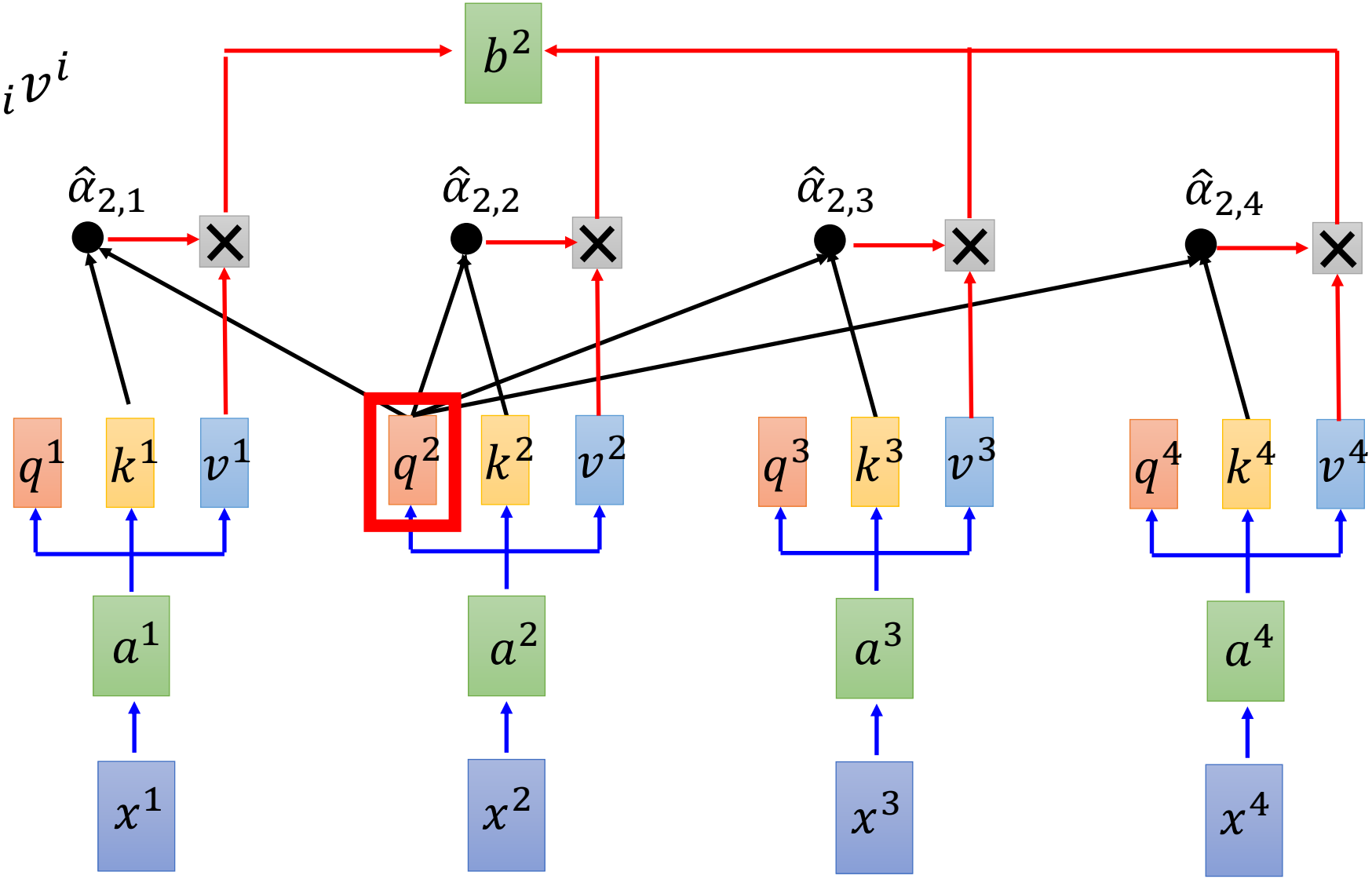
Transformer (Self-Attention)

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



Transformer (Self-Attention)

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



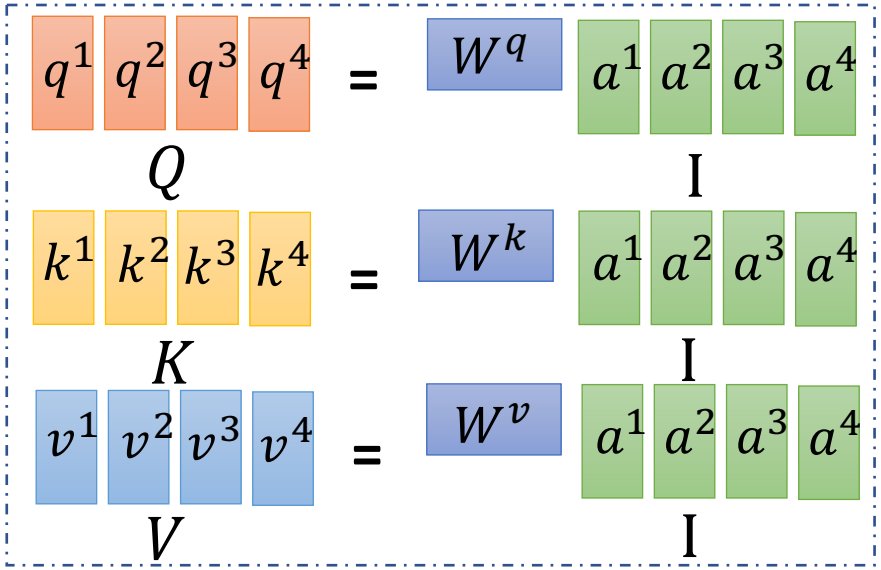
Transformer (Self-Attention)

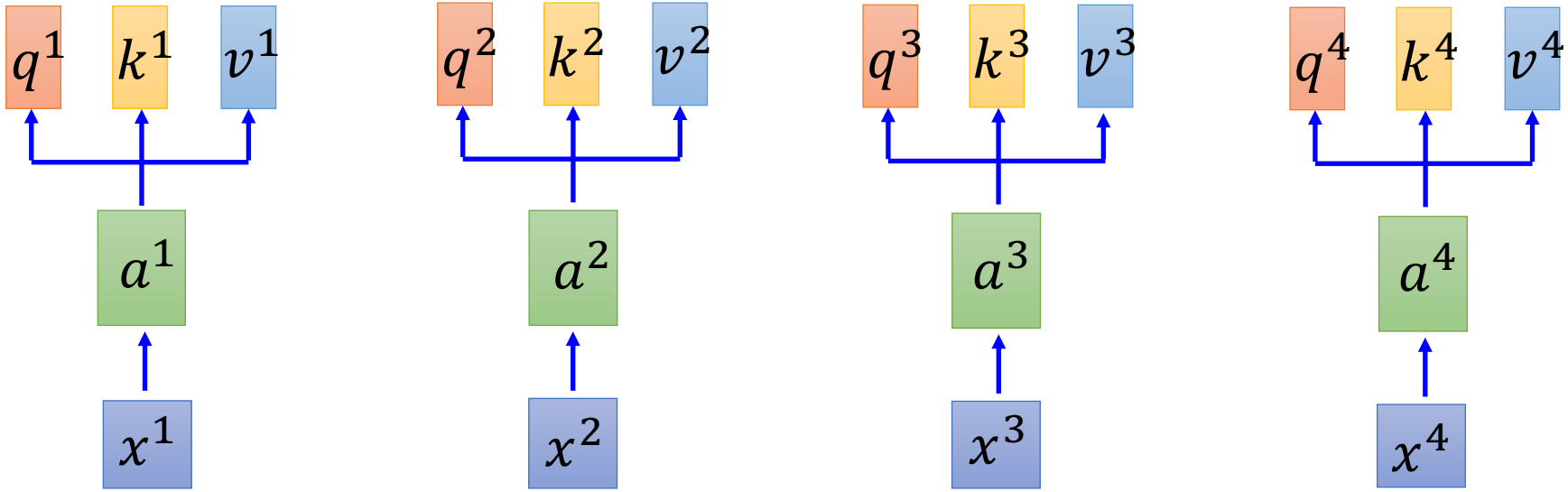
$$q^i = W^q a^i$$

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

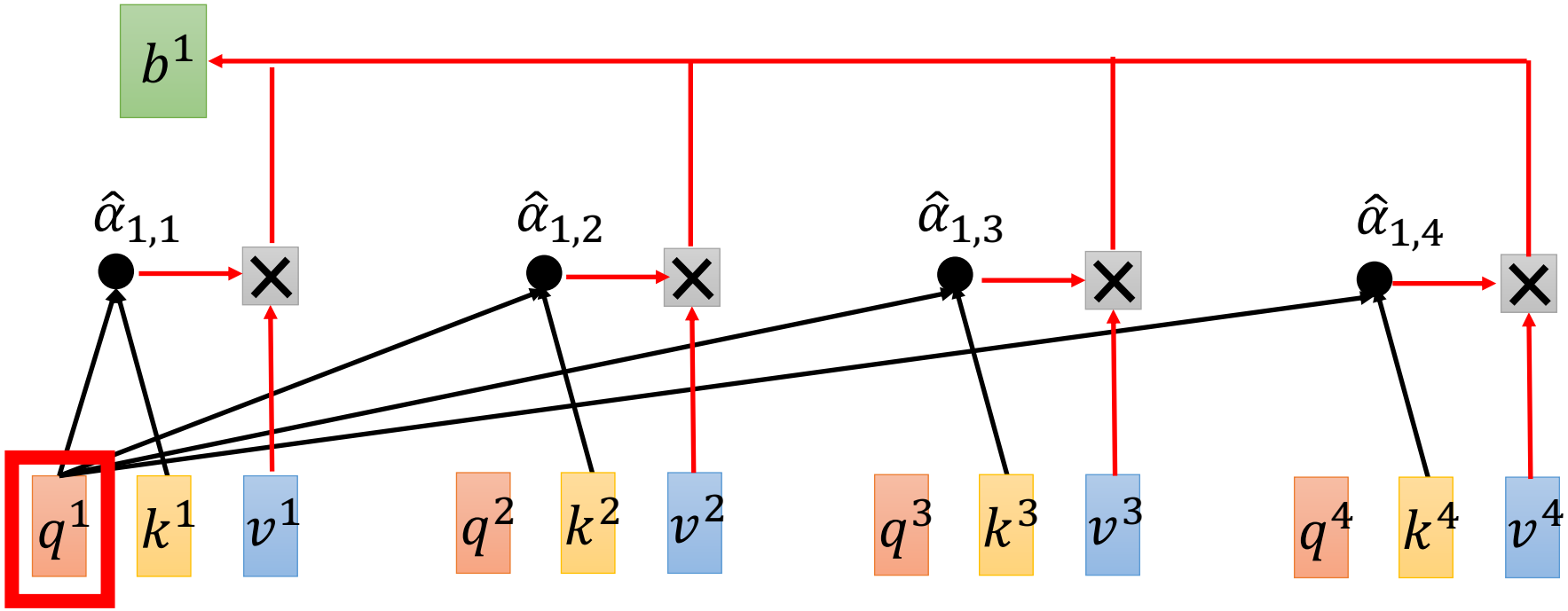
$$v^i = W^v a^i$$

parallelly compute





Transformer (Self-Attention)



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

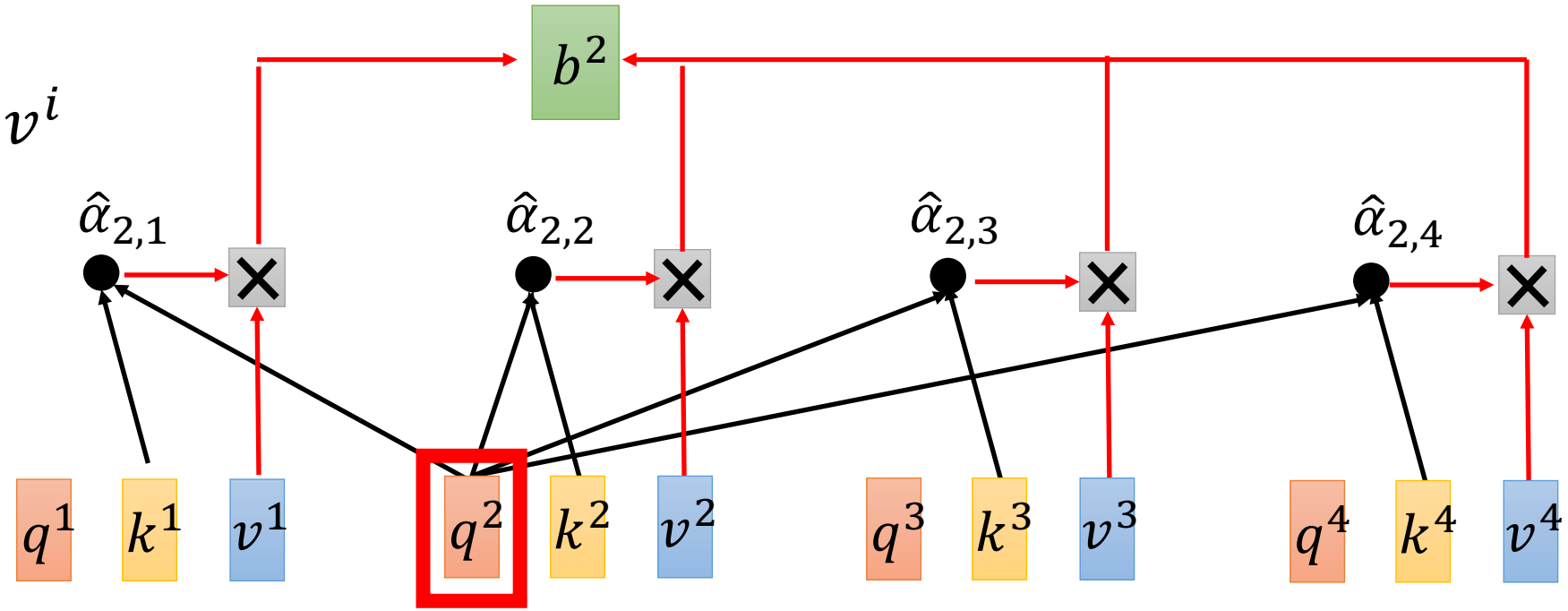
$$\alpha_{1,3} = k^3 q^1 \qquad \alpha_{1,4} = k^4 q^1$$

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

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

Transformer (Self-Attention)

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

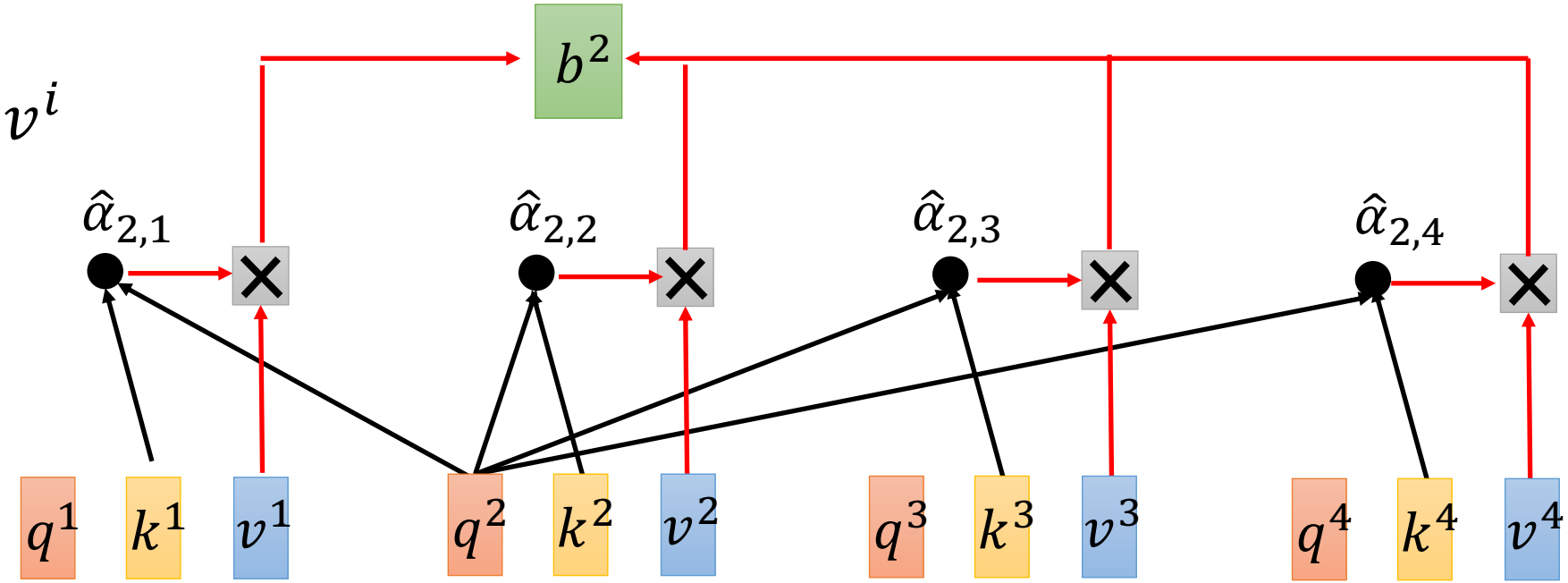


$$\begin{bmatrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{bmatrix} \leftarrow \begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix} = \begin{bmatrix} k^{1T} \\ k^{2T} \\ k^{3T} \\ k^{4T} \end{bmatrix} \begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix}$$

$\hat{A}$ A K^T Q

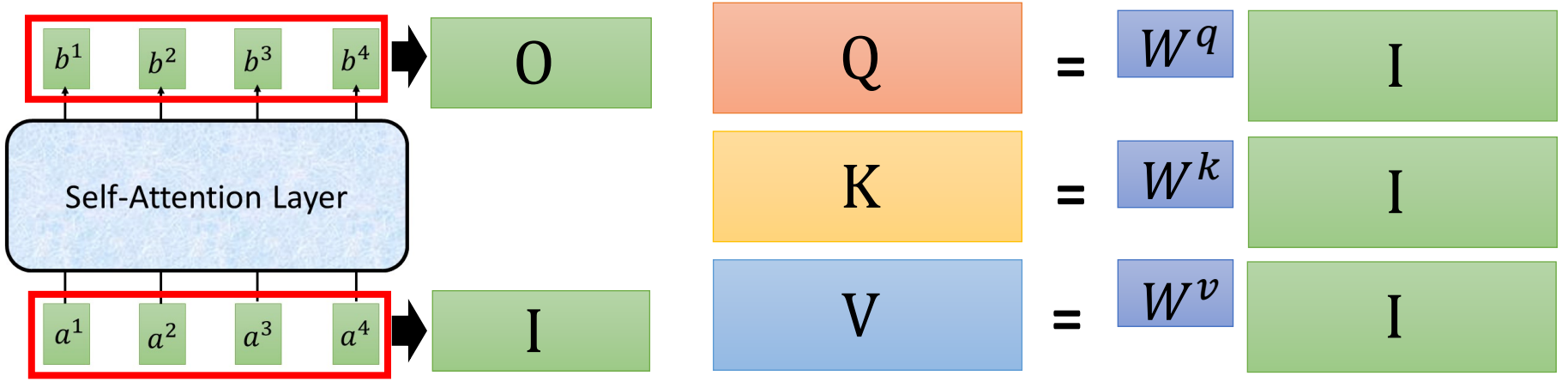
Transformer (Self-Attention)

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



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

Transformer (Self-Attention)



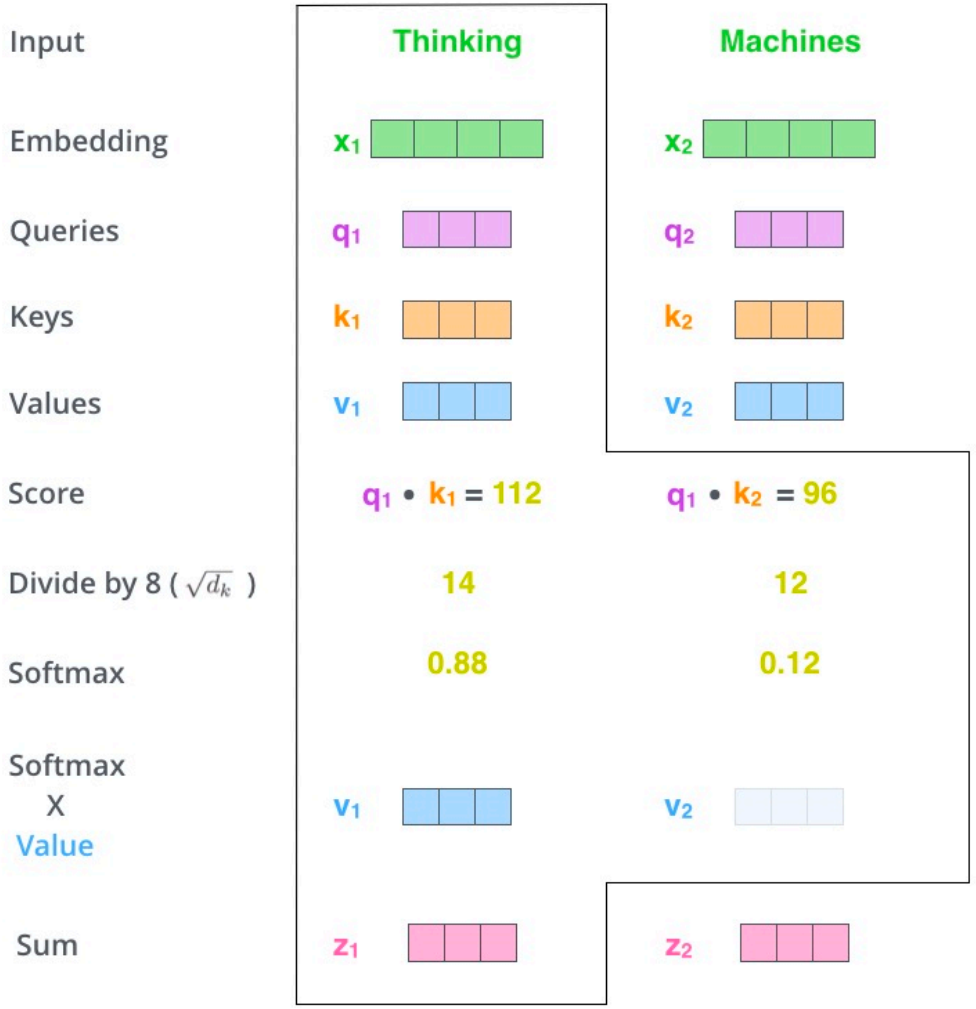
$$\hat{A} \leftarrow A = K^T Q$$

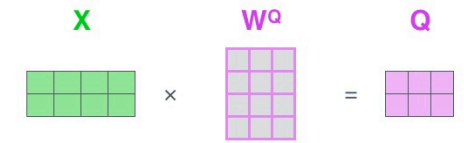
$$O = V \hat{A}$$

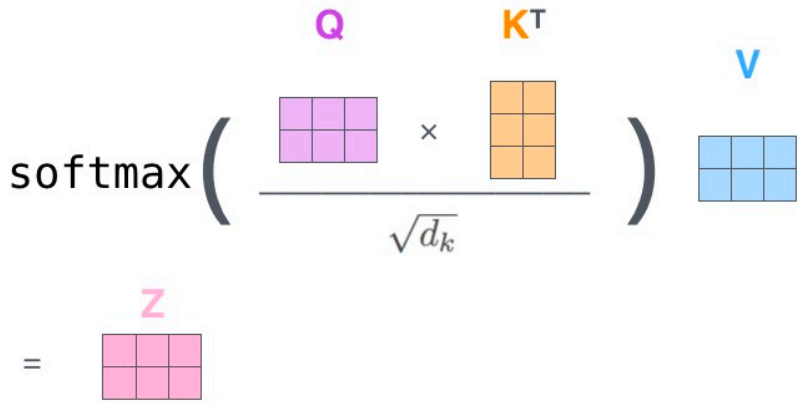
Matrix multiplication: can be accelerated with GPU

Transformer (Self-Attention)

Summary of the self-attention calculation





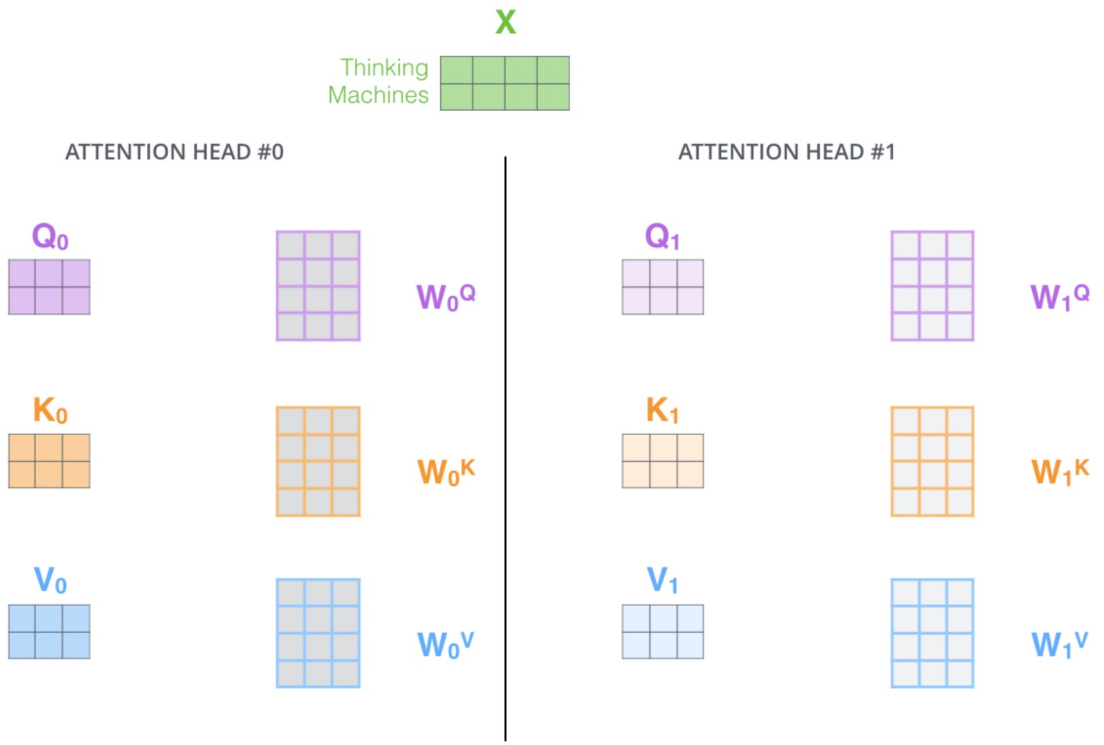


Transformer (Multi-head Self-Attention)

Multi-head self-attention:

- Expands the model's ability to focus on different positions.

- Gives the attention layer multiple "representation subspaces".

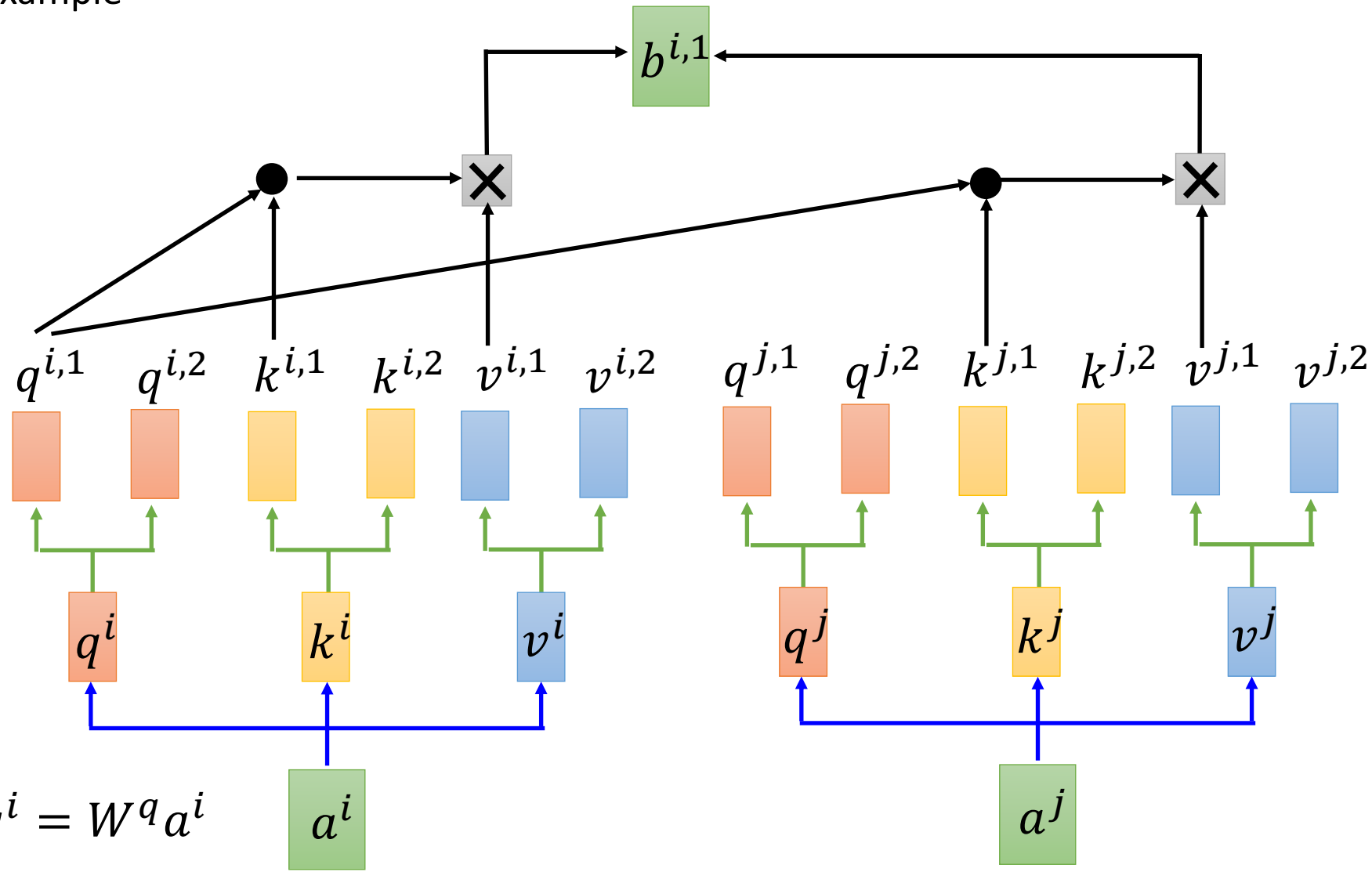


Transformer (Multi-head Self-Attention)

Two heads as an example

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

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



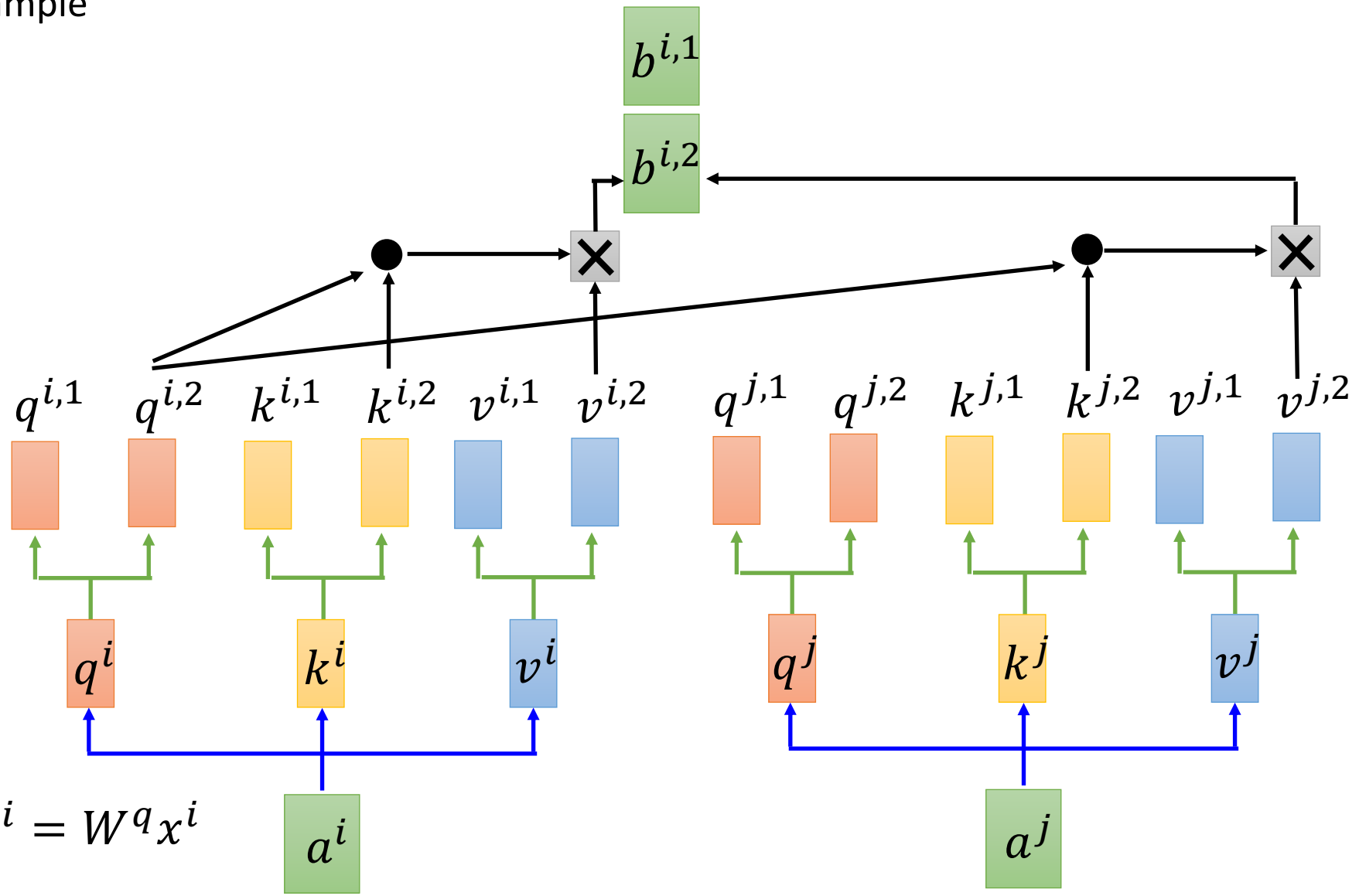
$$q^i = W^q a^i$$

Transformer (Multi-head Self-Attention)

Two heads as an example

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

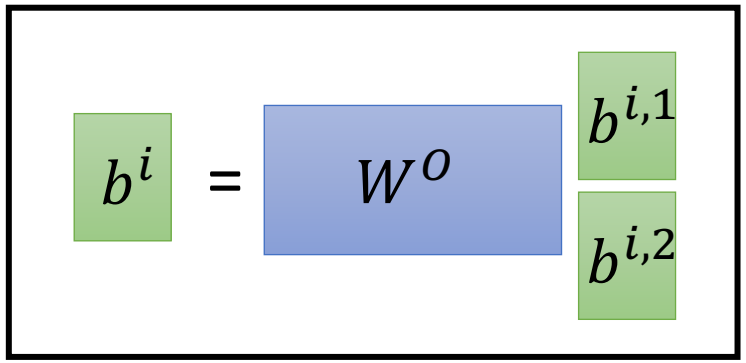
$$q^{i,2} = W^{q,2} q^i$$

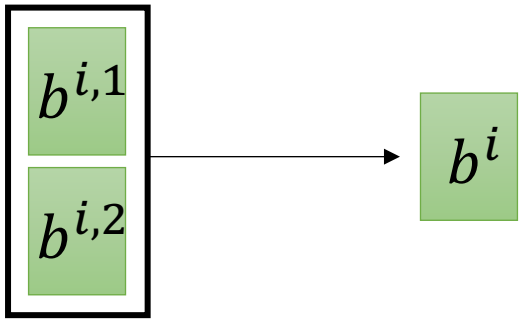


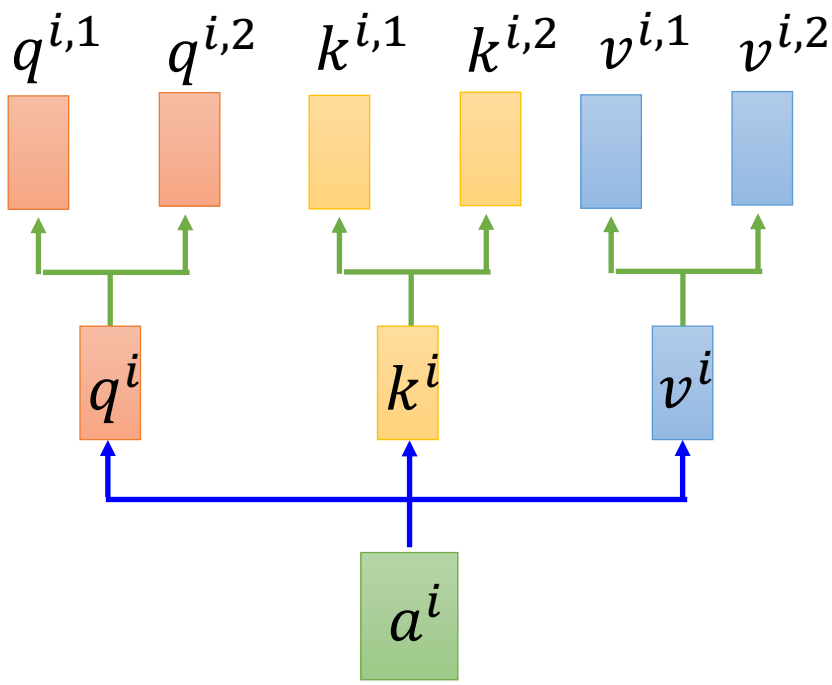
$$q^i = W^q x^i$$

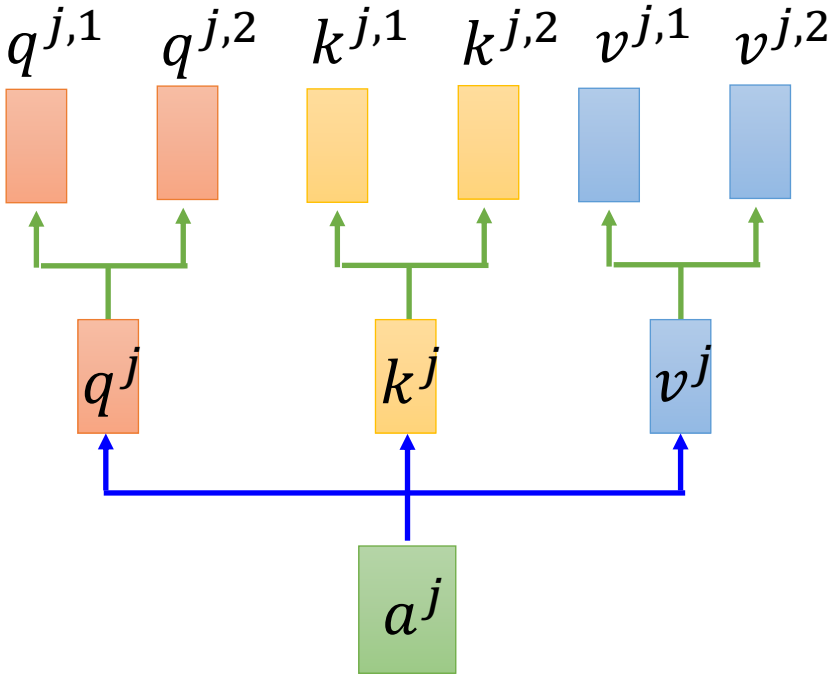
Transformer (Multi-head Self-Attention)

Two heads as an example





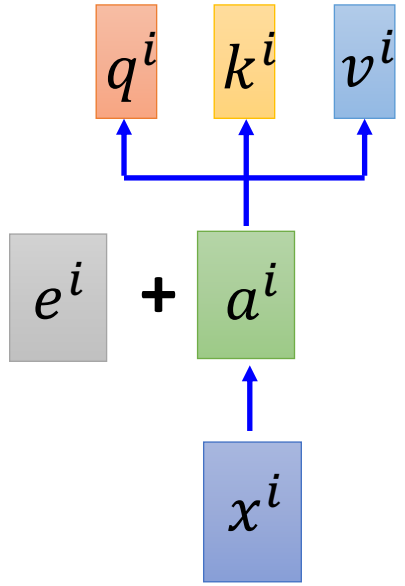


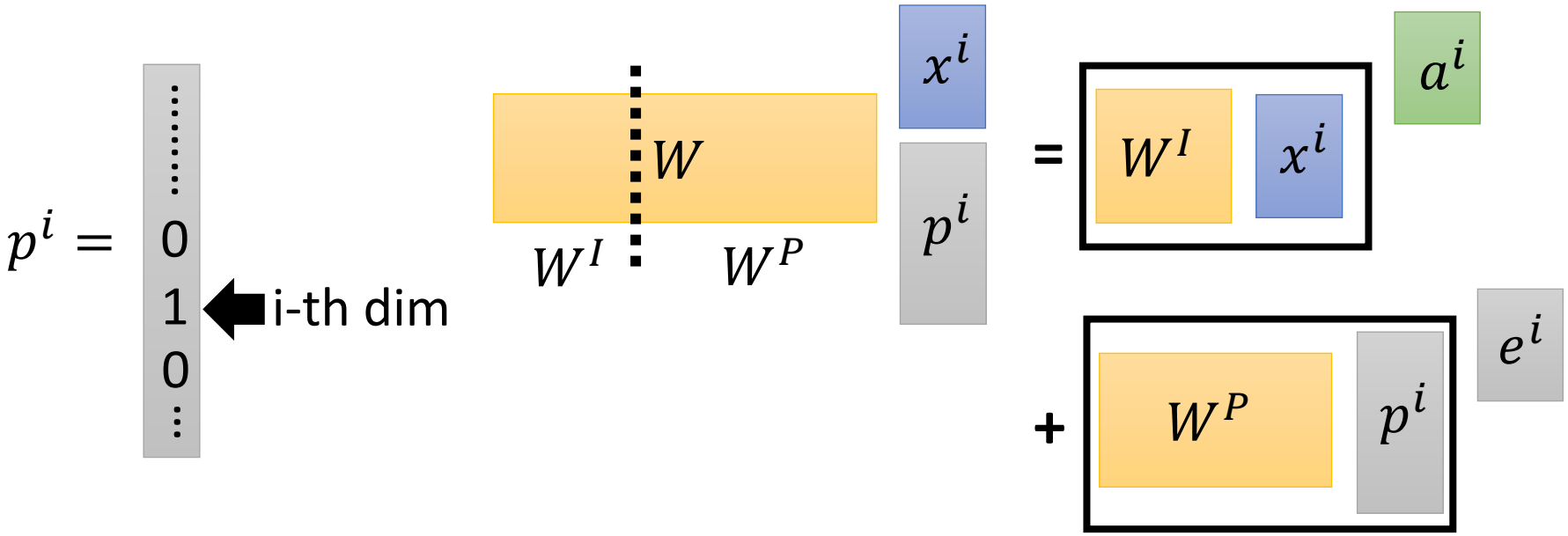


Transformer (Positional Embedding)

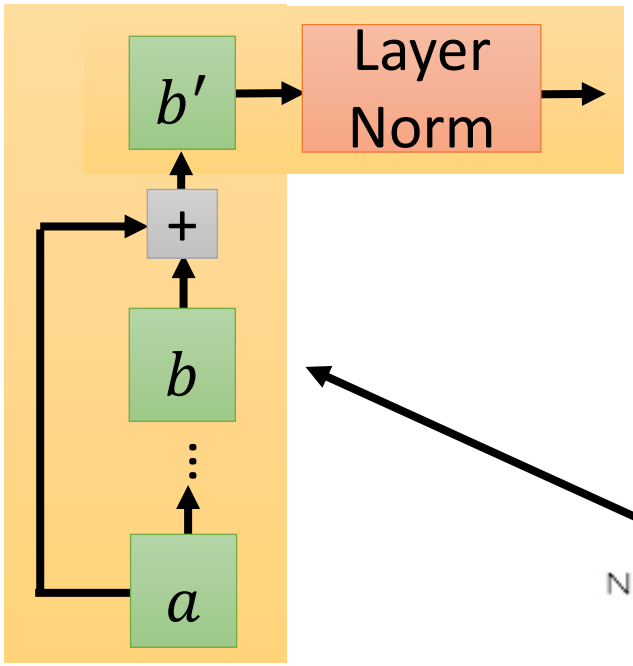
- No position information in self-attention.
- Original paper: each position has a unique positional vector e^i

  In other words: each x^i appends a one-hot vector p^i



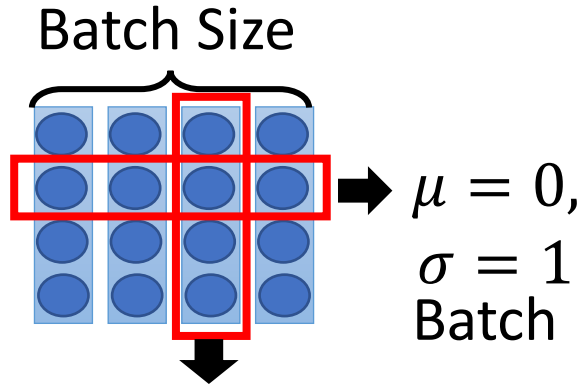


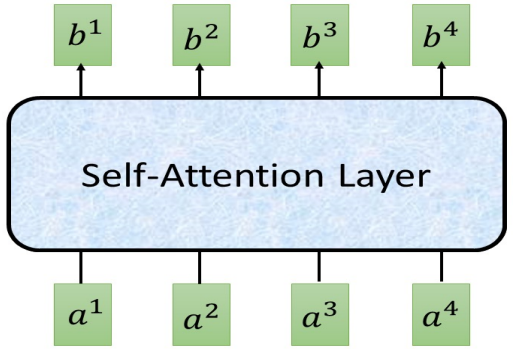
Transformer (Framework)

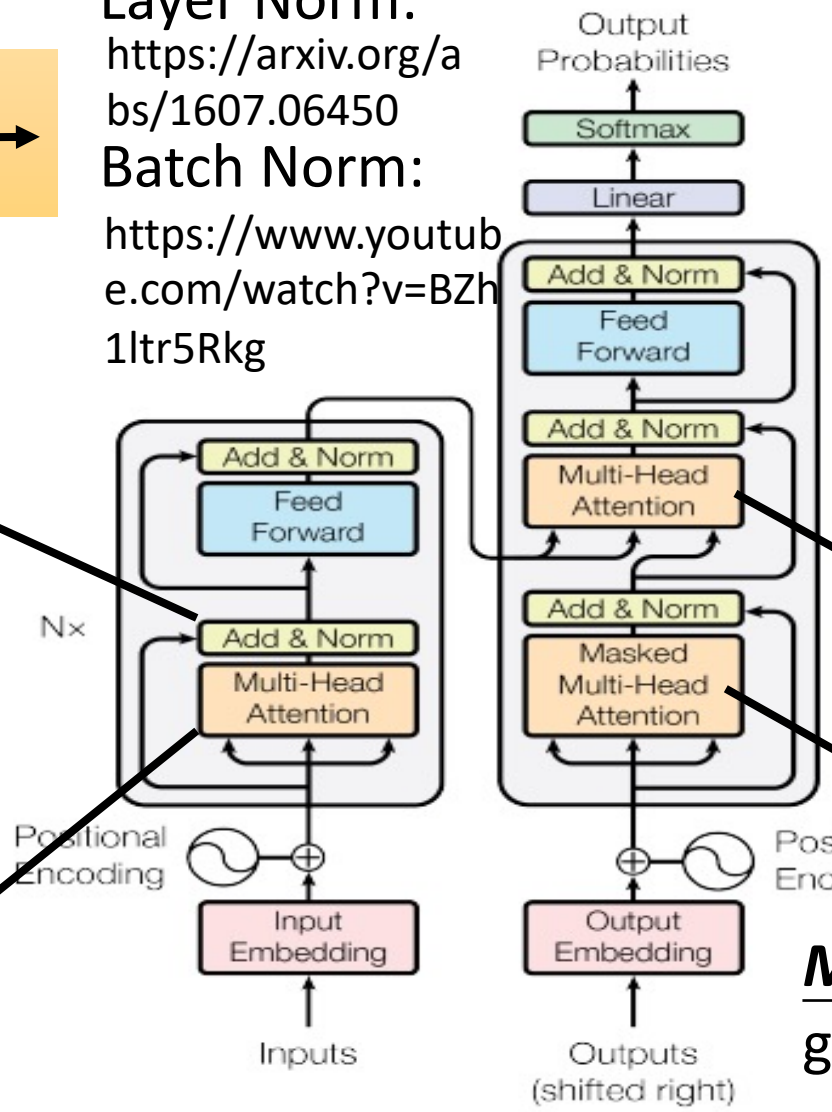


Layer Norm:
https://arxiv.org/abs/1607.06450

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







attend on the input sequence

Masked: attend on the generated sequence

Transformer (Framework)

Transformer uses self-attention in three ways:

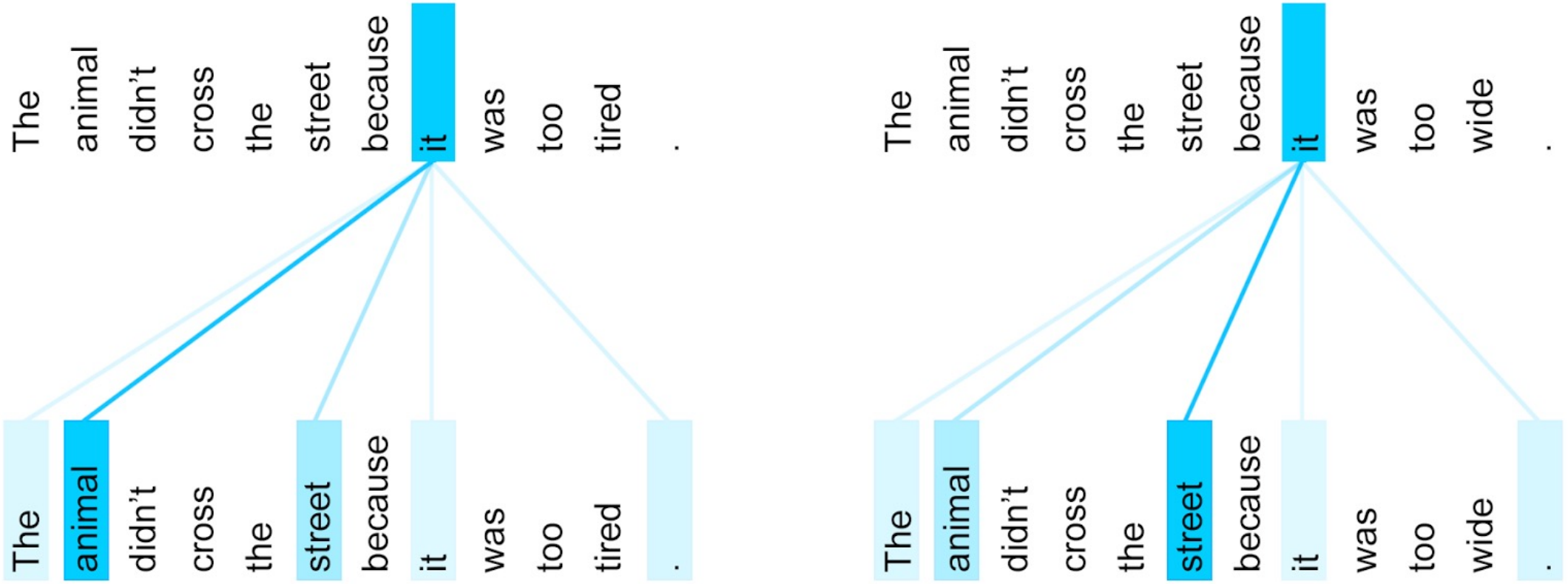
- Used in the encoder. In a self-attention layer all of the keys, values and queries come from the output of the previous layer. Each position in the encoder can attend to all positions in the previous layer.

- Used in the "encoder attention" layer of the decoder. The queries come from the previous decoder layer, and the keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence.

- Used in the “Masked self-attention” layer of the decoder. It allows each position in the decoder to attend to all positions in the decoder up to and including that position.

Encoding-Decoding Process

Transformer (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)

Transformer vs RNN

- Transformer processes the sentence all at once (no recursion is used)
  - RNN uses sequential processing: process the sentence word by word

- Transformer performs better in long-range dependencies due to the self-attention mechanism
  - RNN is not very efficient in handling long sequences.

- Transformer allows more parallelization than RNNs and reduces training times

Transformer (more reference)

- Self attention layer refers to Professor Li Hongyi's Presentation

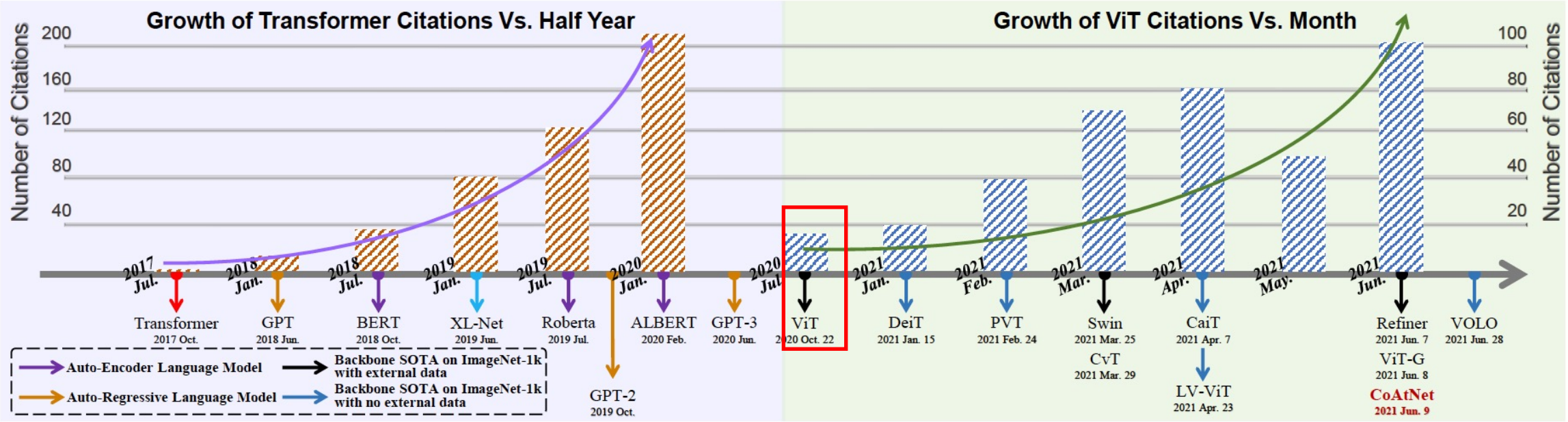
- Resources that greatly accelerates the learning

  1. Illustrated Transformer - Jay Alammar
  2. Transformers from Scratch - Peter Bloem
  3. The Annotated Transformer - Harvard NLP

Visual Transformers

- Transformer has been migrated to Computer Vision tasks: classification, detection, segmentation, tracking, generation, and enhancement.

- Visual Transformers have achieved SOTA performance over many benchmarks.

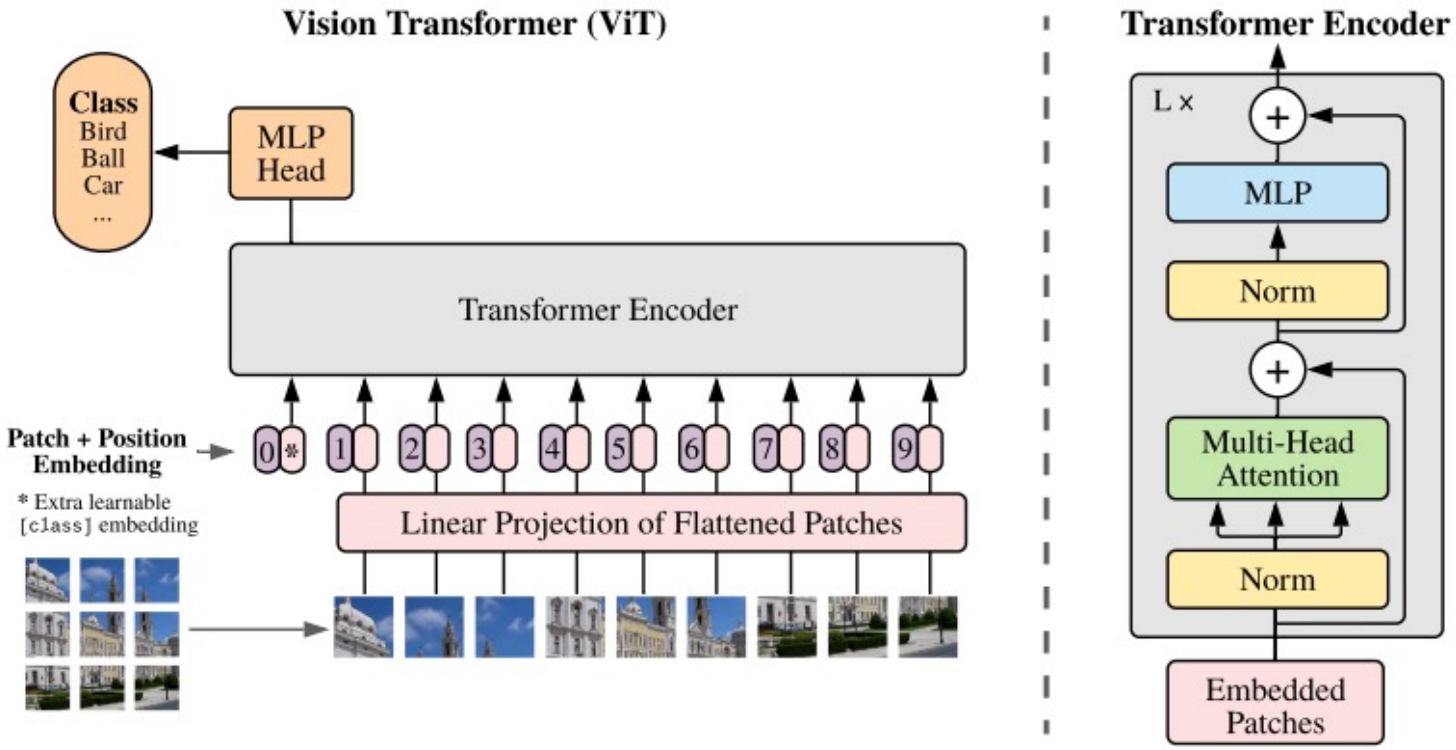


Vision Transformer (ViT)

- First proposed by: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale for image classification task

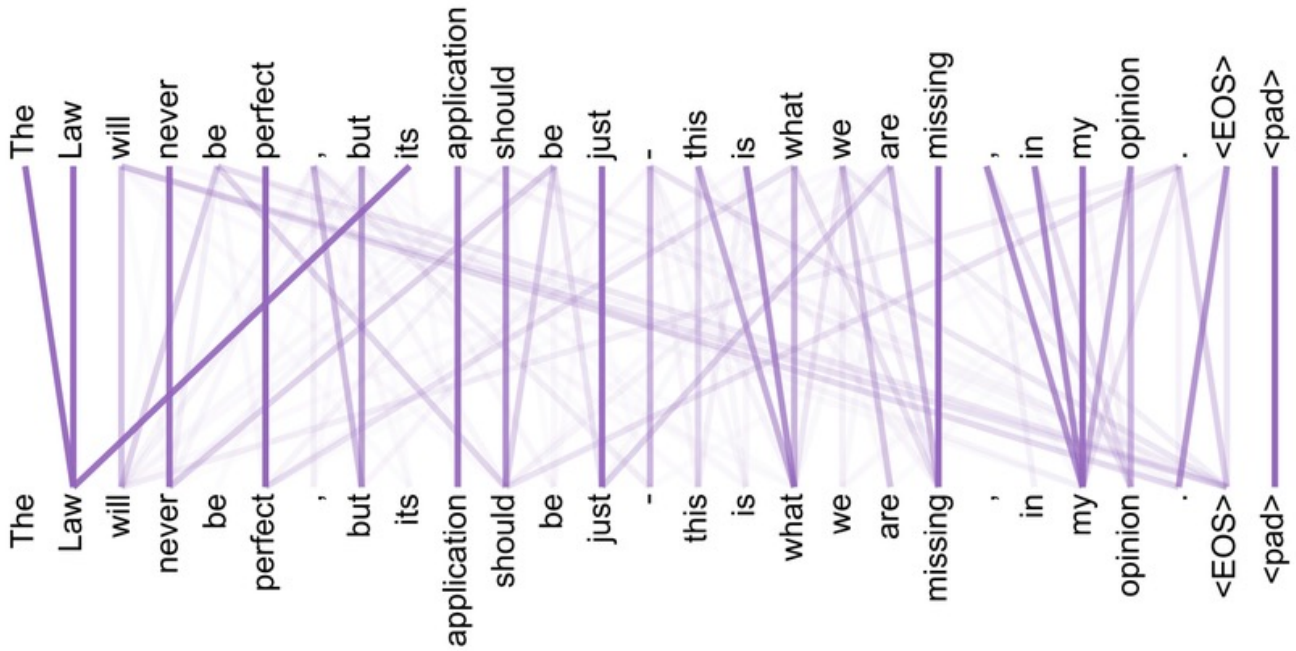
- Published on ICLR 2021 by Google Research, Brain team

Vision Transformer (ViT)



- Divide an input image into 196 (14x14) small images of size (16x16)
- Treat it as embedding in NLP
- Use it as an input for traditional transformer encoder
- Use 12 transformer layers (Norm, Multi-head attention, etc.)
- Take the last output, use it as input for Dense Layer with 1000 classes

ViT: Why don’t we use a full image for transformer?



Attention in NLP

Suppose we treat every pixel in the image(3*224*224) as a word token in NLP for transformer calculation

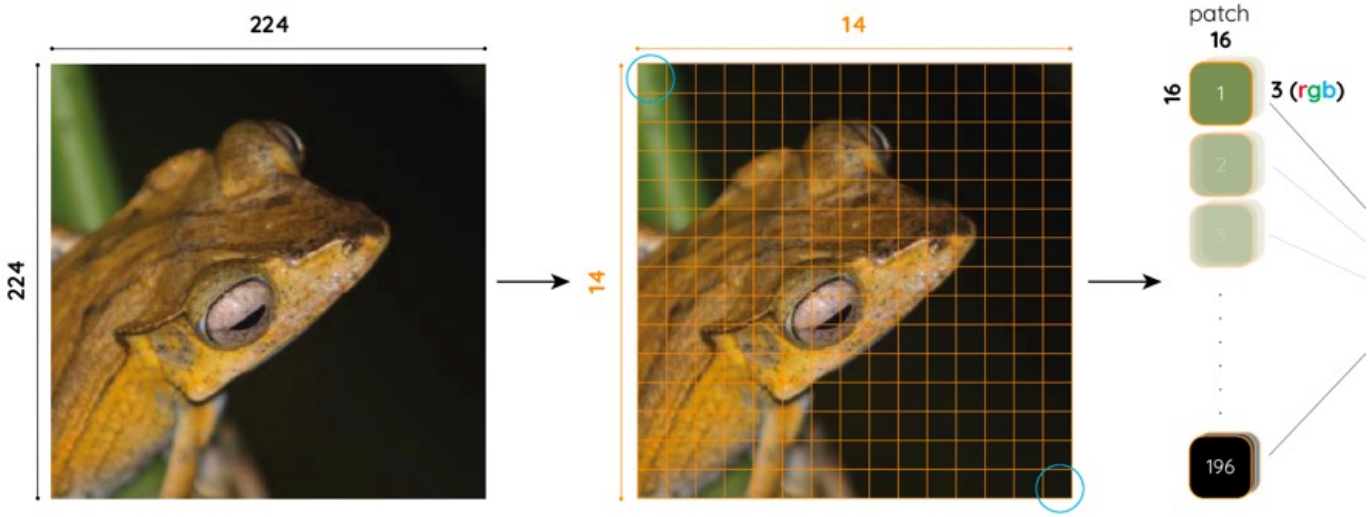
n^2 parameters need to be stored (n is the number of input token)

$(3*224*224)^2$ = 22 billion parameters

The complexity is too high

ViT: Split Image into Patches

- Instead of calculating self-attention in pixel-level, we divided the image into patches and calculate in patch-level.



Suppose we have an image of size (3*224*224)

↓

Divided into 196(14*14) patches of size (3*16*16)

↓

Where: 16 is the patch size (artificial parameter)
14 = 224 / 16

↓

Each patch is converted into a vector of size 3*16*16=768

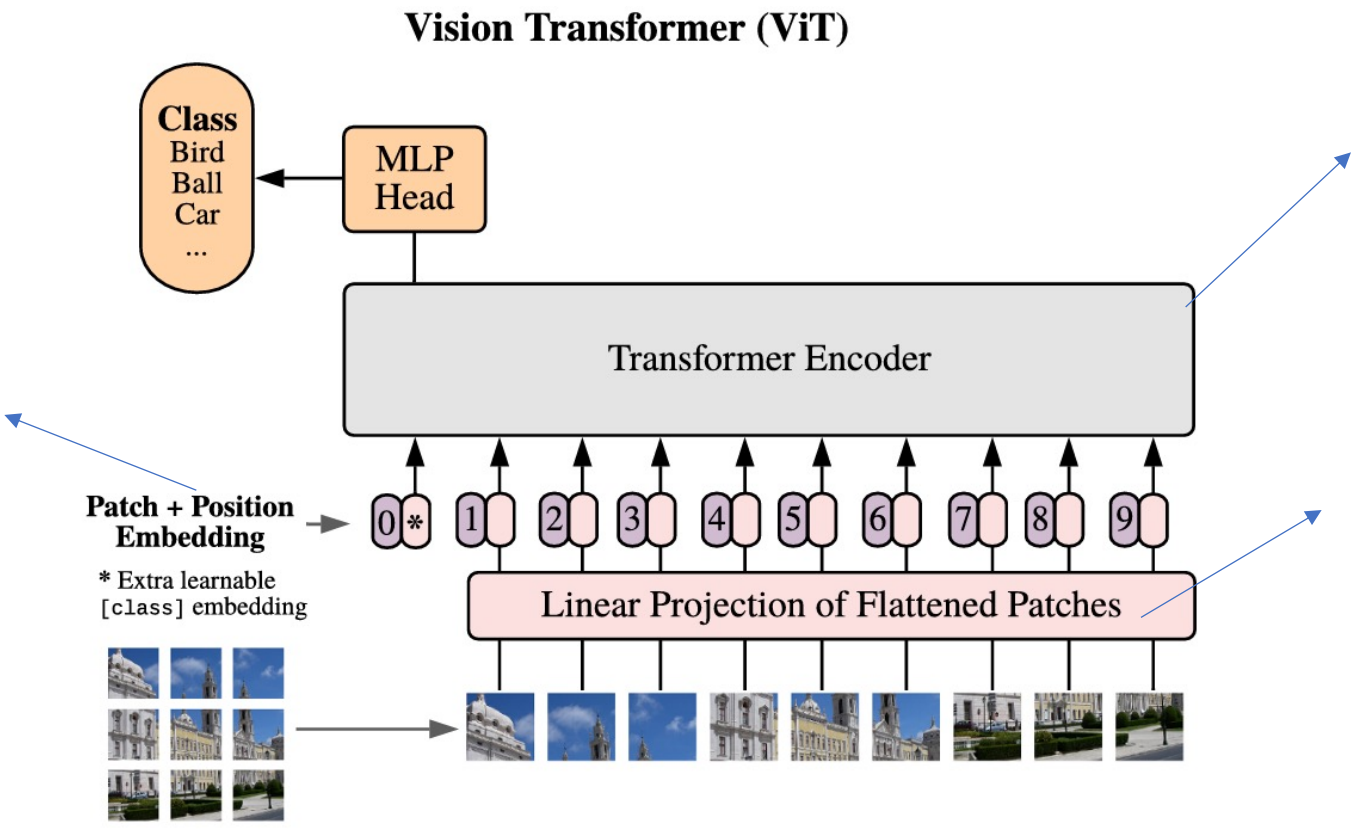
↓

We convert an image into 196 vectors of size 768, a matrix of size (196, 768)

Vision Transformer framework

position embedding: same as “attention is all you need” since self-attention has no position information

[class] embedding: represent a global embedding, further used for image classification

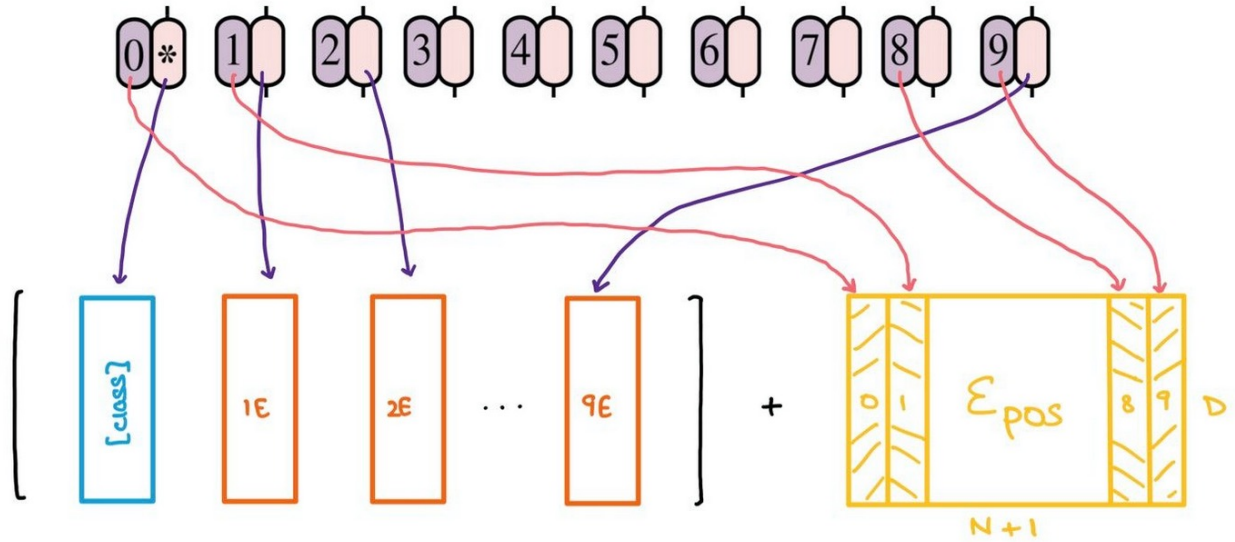


Same structure with the encoder in “attention is all you need”

Flattened Patches: (196, 768), further use a linear projection to map to a fix dimension (artificial parameter)

Vision Transformer framework

Join the patch embeddings and the positional embeddings



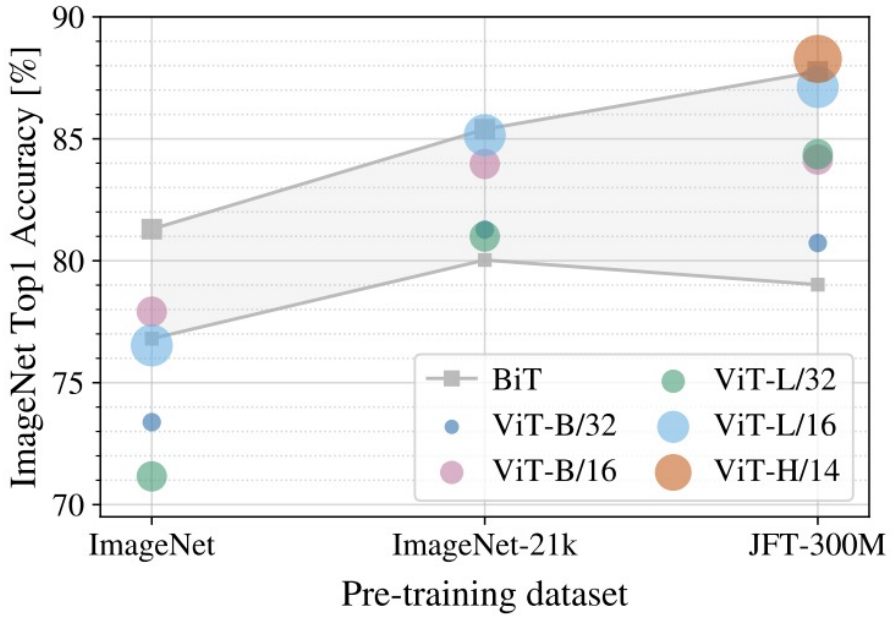
ViT: Results Comparison

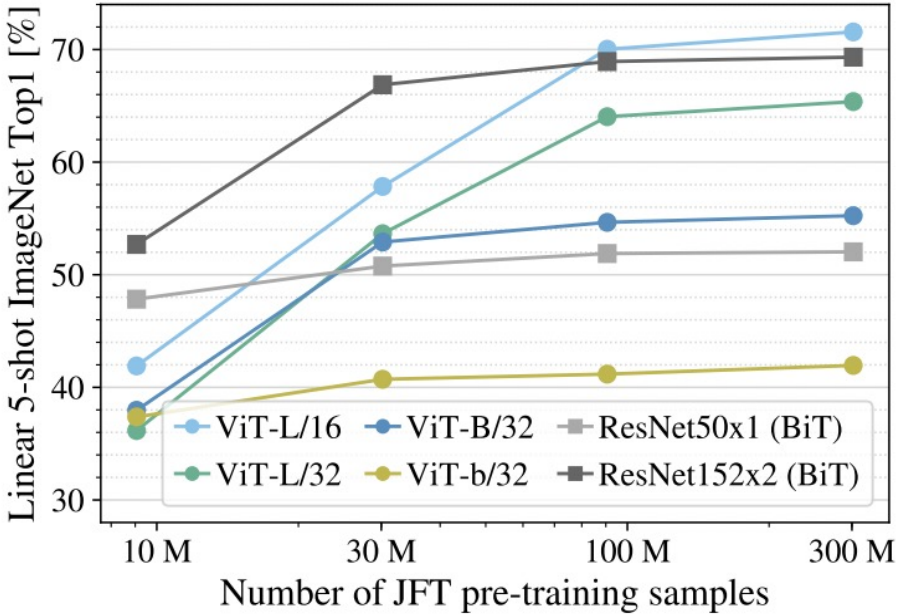
Comparison with state of the art on popular image classification benchmarks

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

- Worse than Resnet when trained just on ImageNet
- Performance improved when pre-trained on big dataset (JFT)

ViT: Results Comparison





- **B**: base model **L**: large model **H**: high resolution **number**: patch size
- larger ViT variants overtake smaller ones as the dataset grows.

- ResNets perform better with smaller pre-training datasets but plateau sooner than ViT, which performs better with larger pre-training.

ViT: Attention Visualization

Input image

Attention

Input image

Attention

Pseudocode of ViT

```
def ViT (input):
    patches = Create_Patches(input)
    patch_embed = Patch_Embedding(patches)
    sequence = Concat(class_token, patch_embed) + Position_embedding
    hidden states = Transformer(sequence)
    class_output = Classification_Head(hidden_states[0])
    return class_output
```

Development of Visual Transformers

A Survey of Visual Transformers by Y. Liu et al