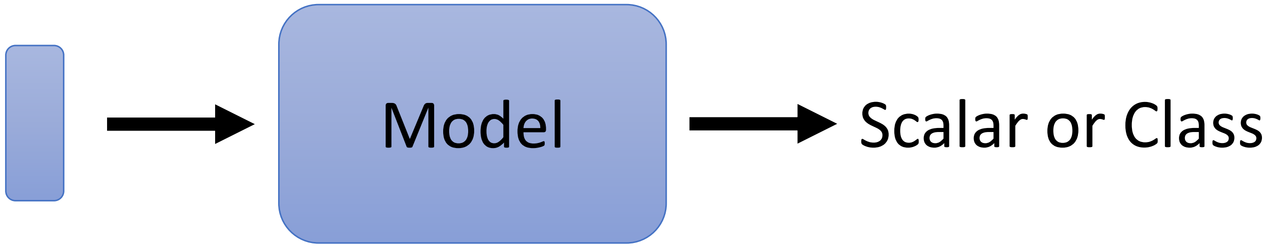
Self-attention

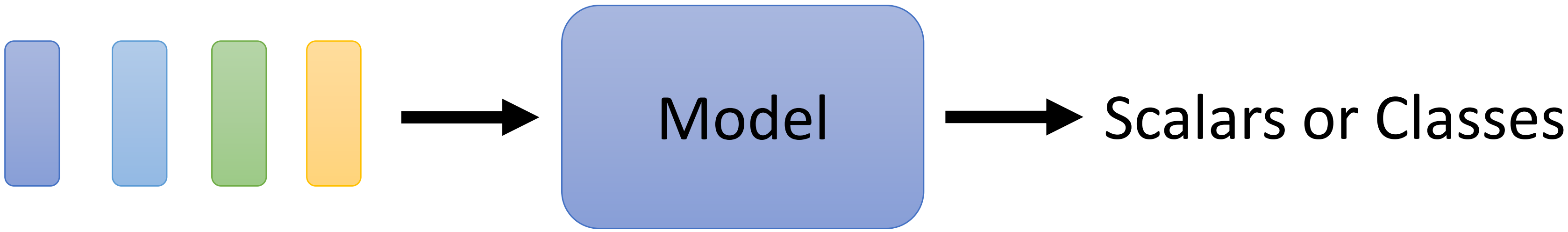
Hung-yi Lee
李宏毅

Sophisticated Input

- Input is **a vector**



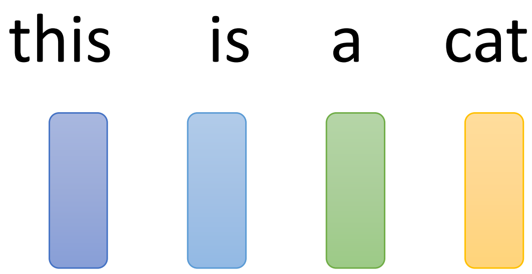
- Input is **a set of vectors**



(may change length)

输入不仅仅是一排向量，且输入向量的数目是会改变的

Vector Set as Input



最简单的做法，开一个很长的向量
每一个维度对应一个词汇
问题在于：假设所有词汇彼此之间是没有关系的

One-hot Encoding

apple = [1 0 0 0 0 ......]

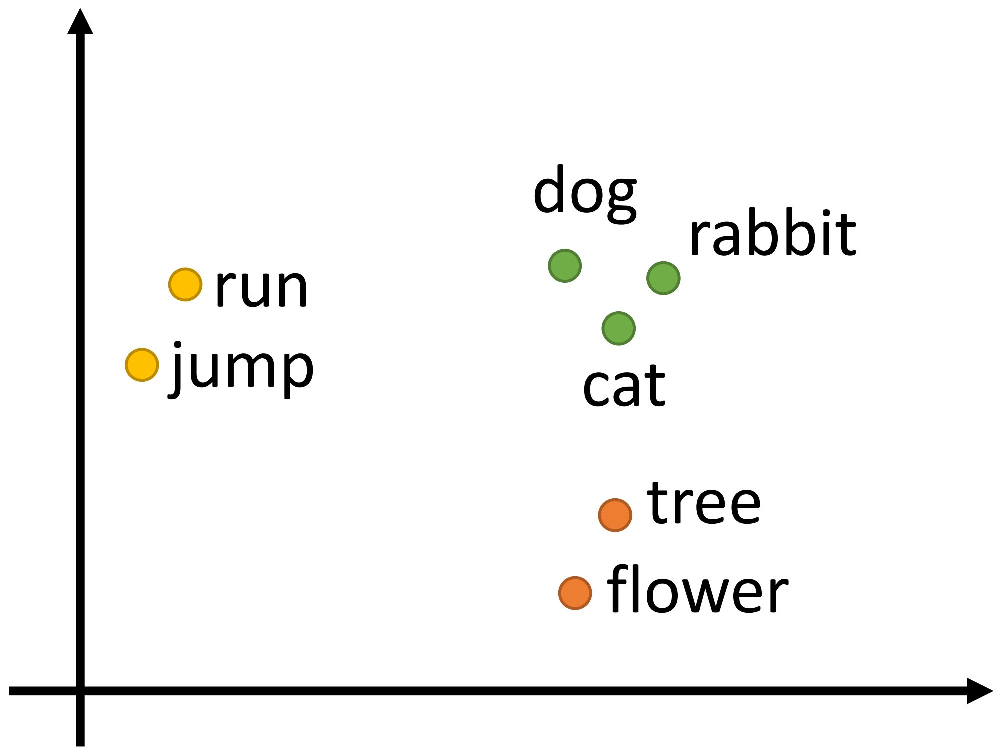
bag = [0 1 0 0 0 ......]

cat = [0 0 1 0 0 ......]

dog = [0 0 0 1 0 ......]

elephant = [0 0 0 0 1 ......]

Word Embedding

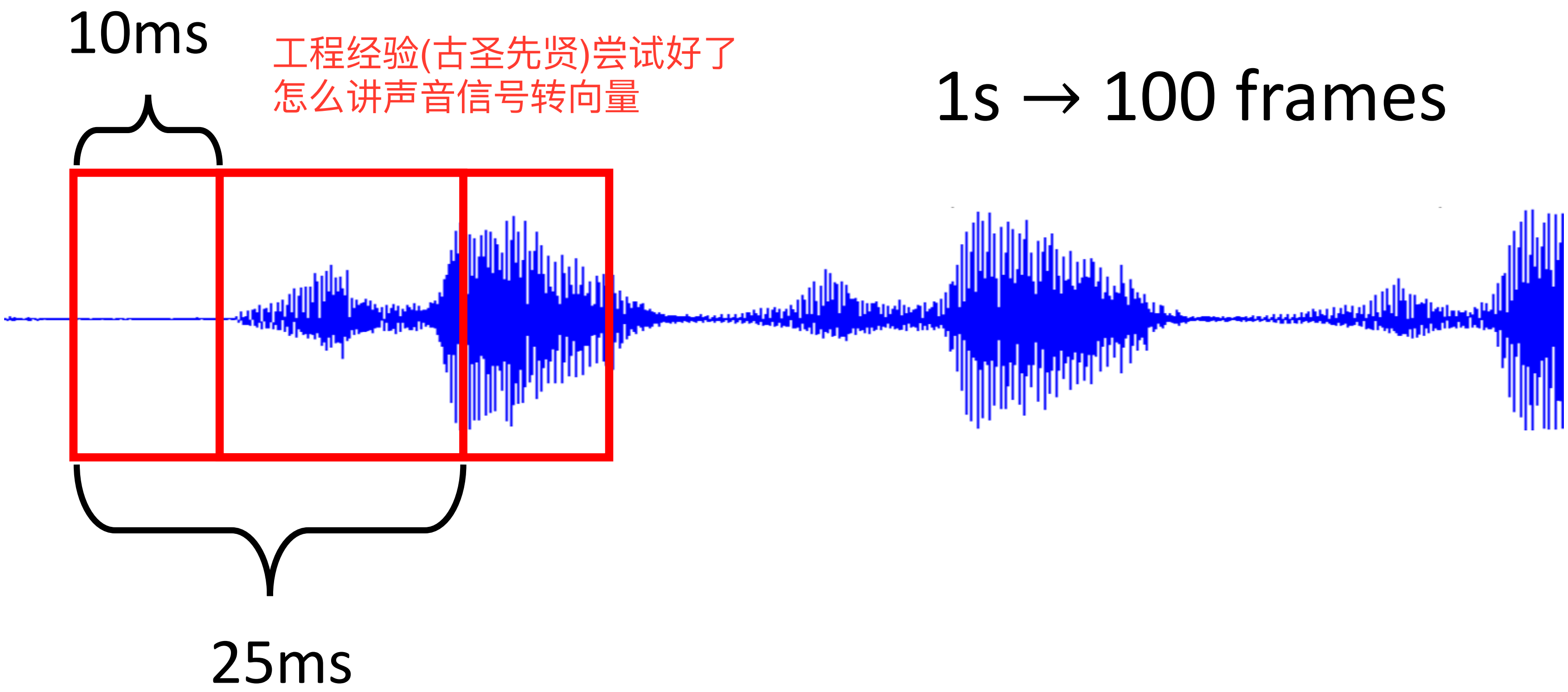


To learn more: https://youtu.be/X7PH3NuYW0Q (in Mandarin)

Vector Set as Input

工程经验(古圣先贤)尝试好了
怎么讲声音信号转向量

1s → 100 frames



frame

400 sample points (16KHz)
39-dim MFCC
80-dim filter bank output

https://medium.com/analytics-vidhya/social-network-analytics-f082f4e21b16

Vector Set as Input

- Graph is also a set of vectors (consider each **node** as **a vector**)

每个节点可以看成一个向量

Each profile is a vector

http://www.twword.com/wiki/%E5%88%86%E5%AD%90

Vector Set as Input

- Graph is also a set of vectors (consider each **node** as **a vector**)

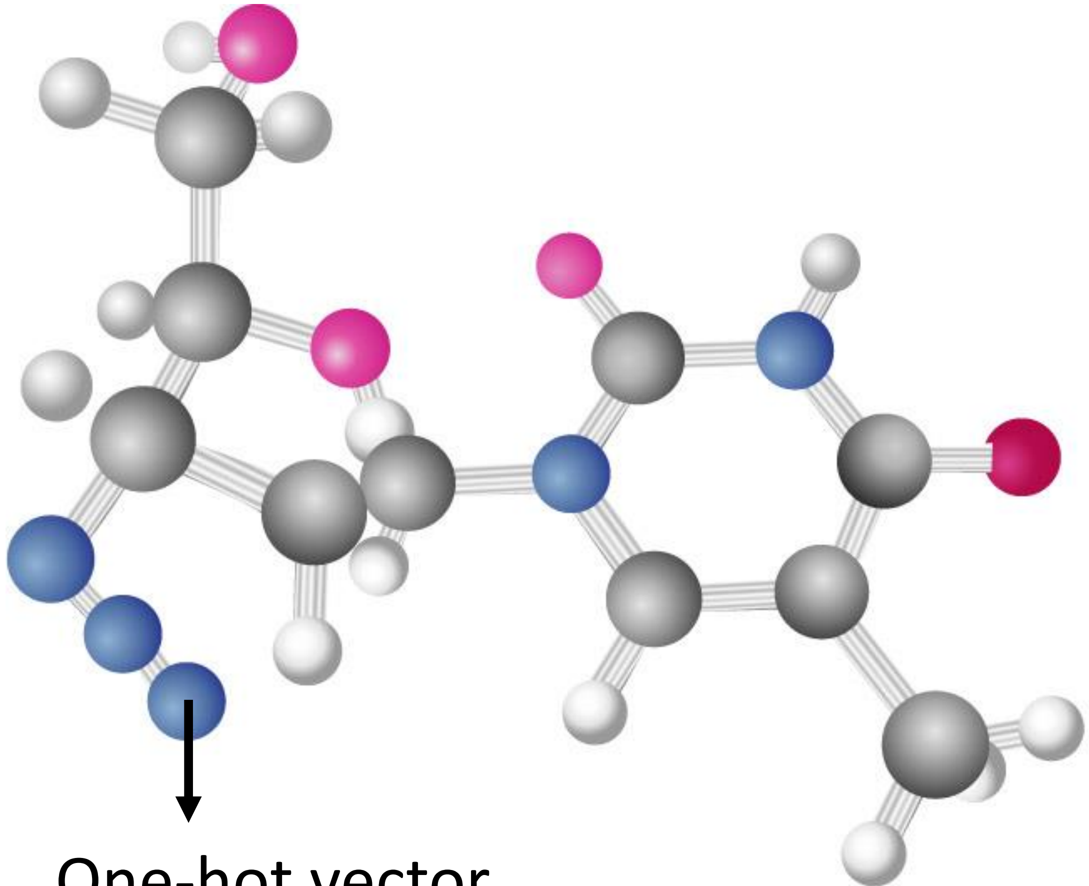
也可以按照 one-hot 来进行标注

H = [1 0 0 0 0 ......]

C = [0 1 0 0 0 ......]

O = [0 0 1 0 0 ......]

⋮



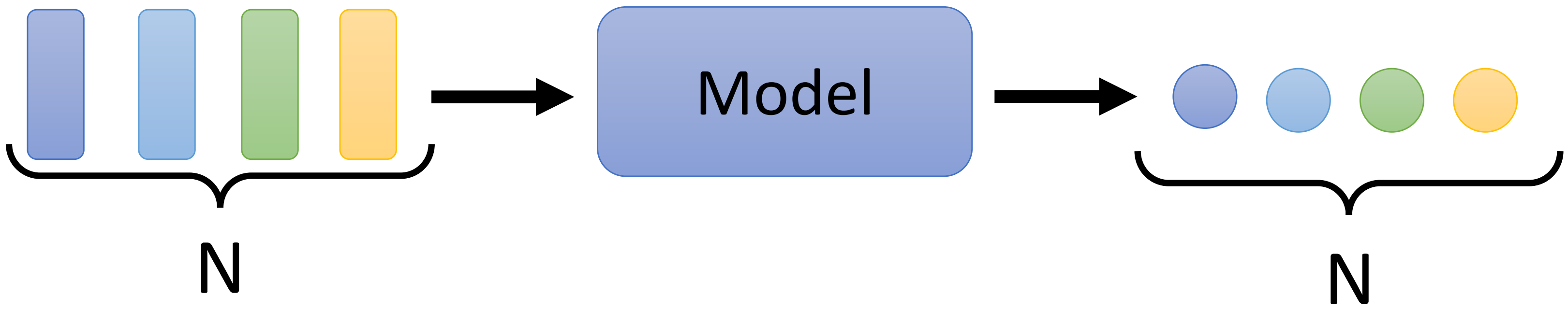
One-hot vector

分子上每个原子就是一个向量

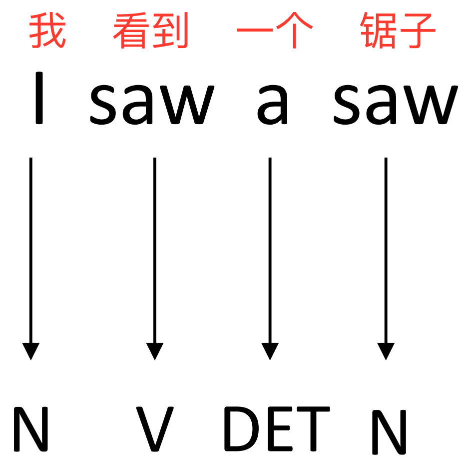
What is the output?

- Each vector has a label.

每个向量都有一个label，输入输出长度是一样的，模型不用操行

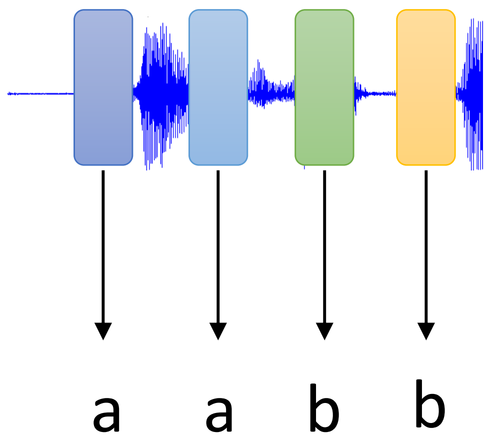


Example Applications



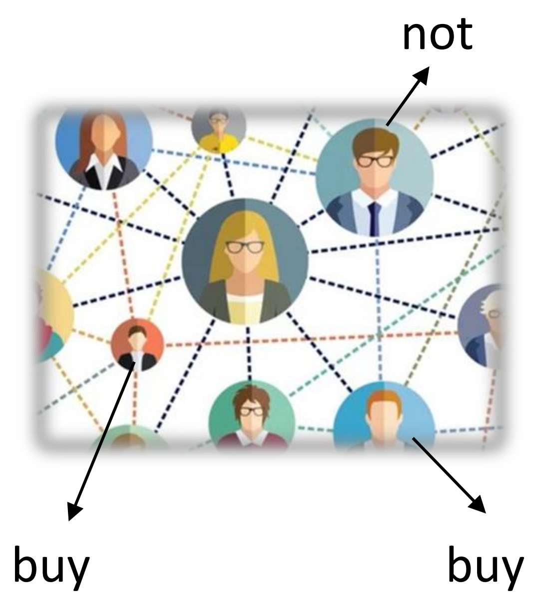
POS tagging

文字处理中的词性标注



语音识别

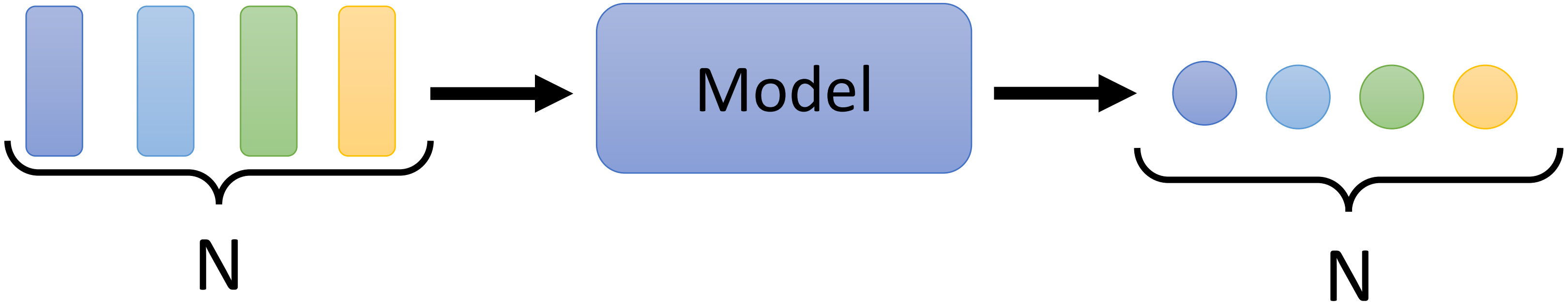
HW2



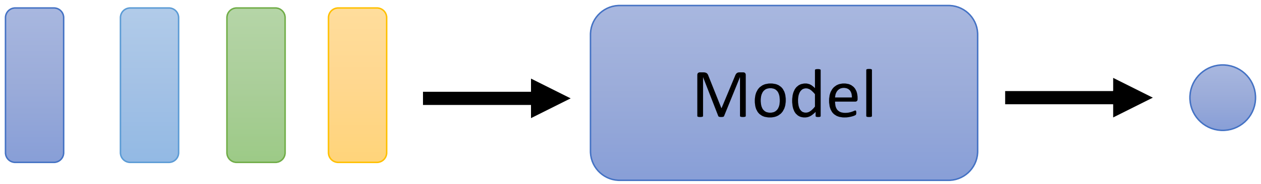
社交网络决定每个节点的特性

What is the output?

- Each vector has a label.



- The whole sequence has a label. 一整个序列输出一个 label

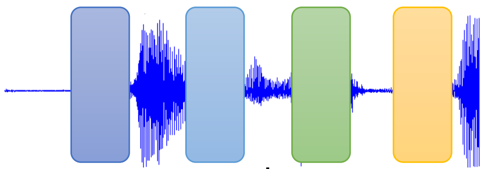


Example Applications

this is good

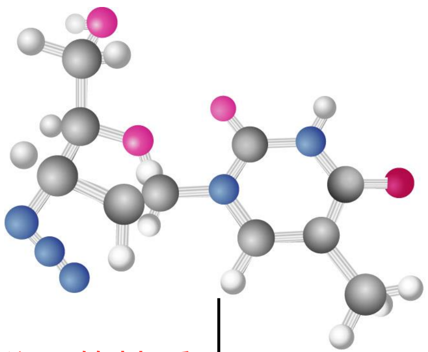
Sentiment analysis ↓ 情感分析

positive



语者辨认 ↓ **HW4**

speaker



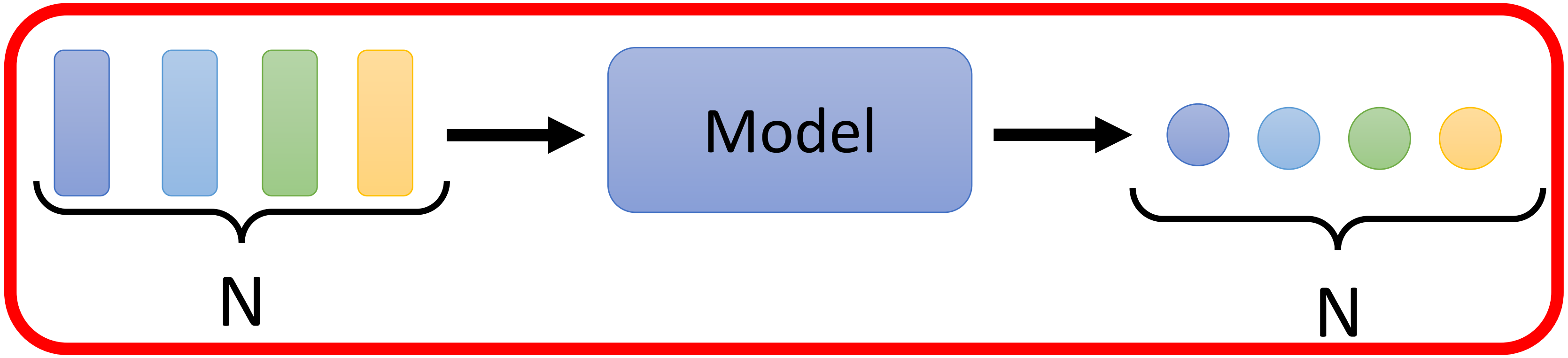
预测分子的性质 ↓

hydrophilicity

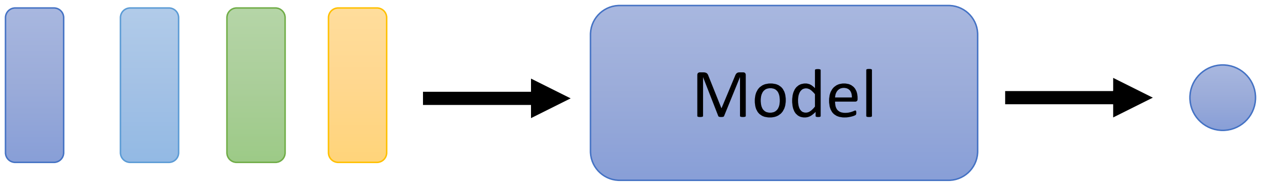
What is the output?

- Each vector has a label.

focus of this lecture



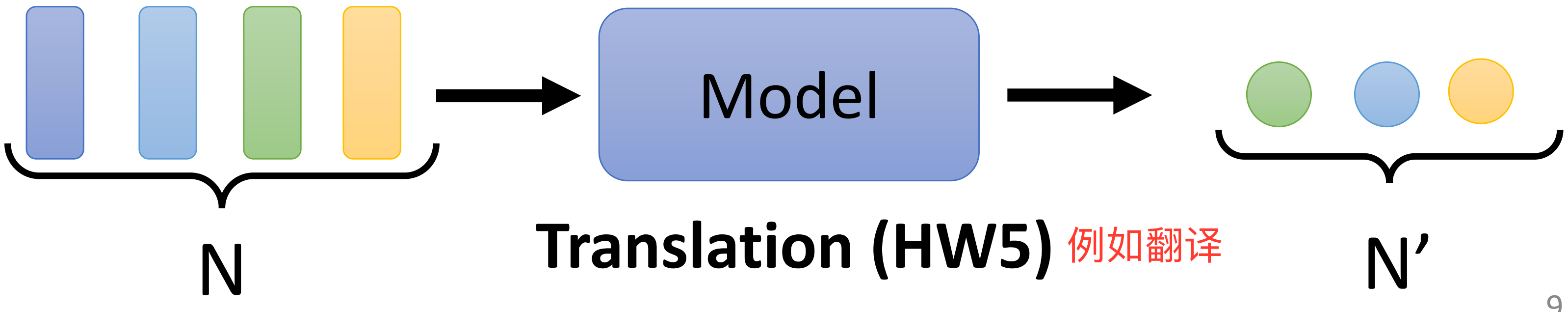
- The whole sequence has a label.



- Model decides the number of labels itself.

seq2seq

机器自己决定应该输出多少个 label



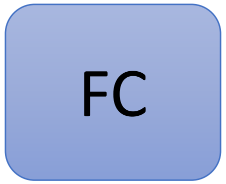
直接想法：各个击破，单个东西输入给网络获得单个输出
但是对于 FC 而言输入两个 saw 没有理由输出不同的东西
自然想法：能不能考虑上下文(window)？也就是输入连续几个来决定中间者是哪个

输入输出一样的情况

Sequence Labeling

Is it possible to consider the context?

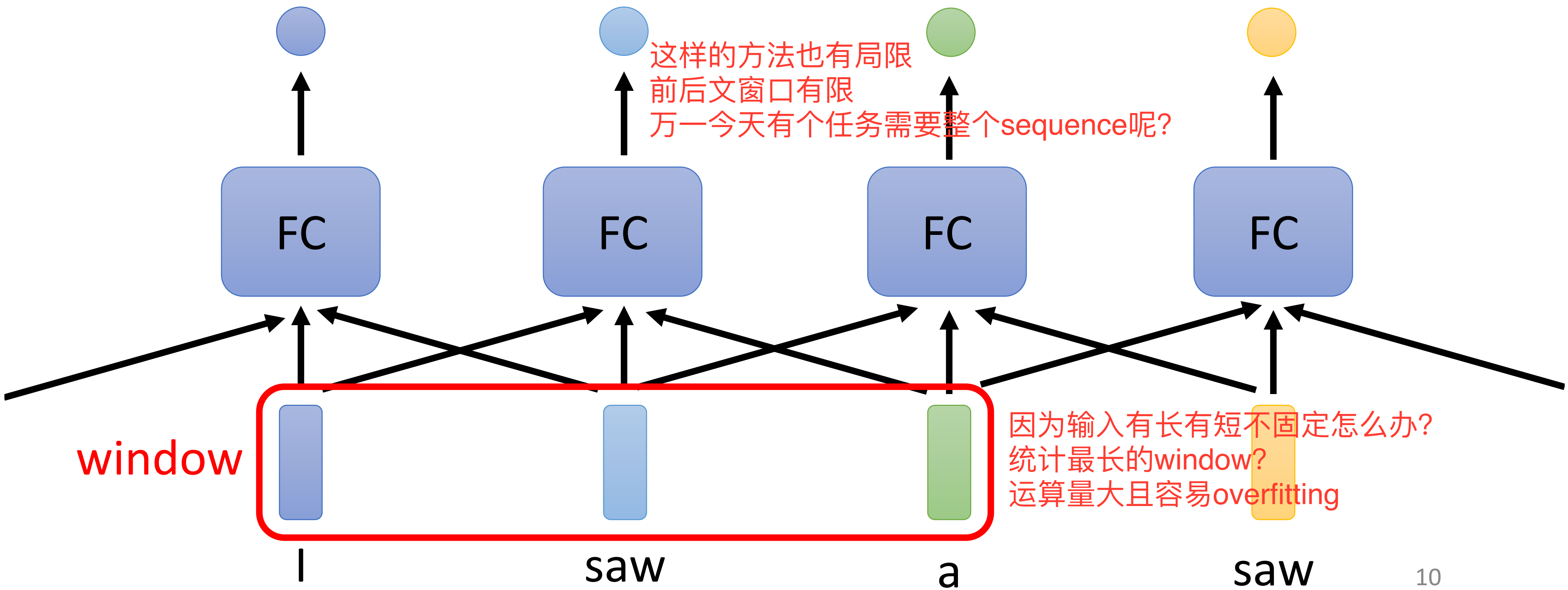
FC can consider the neighbor



Fully-connected

How to consider the whole sequence?

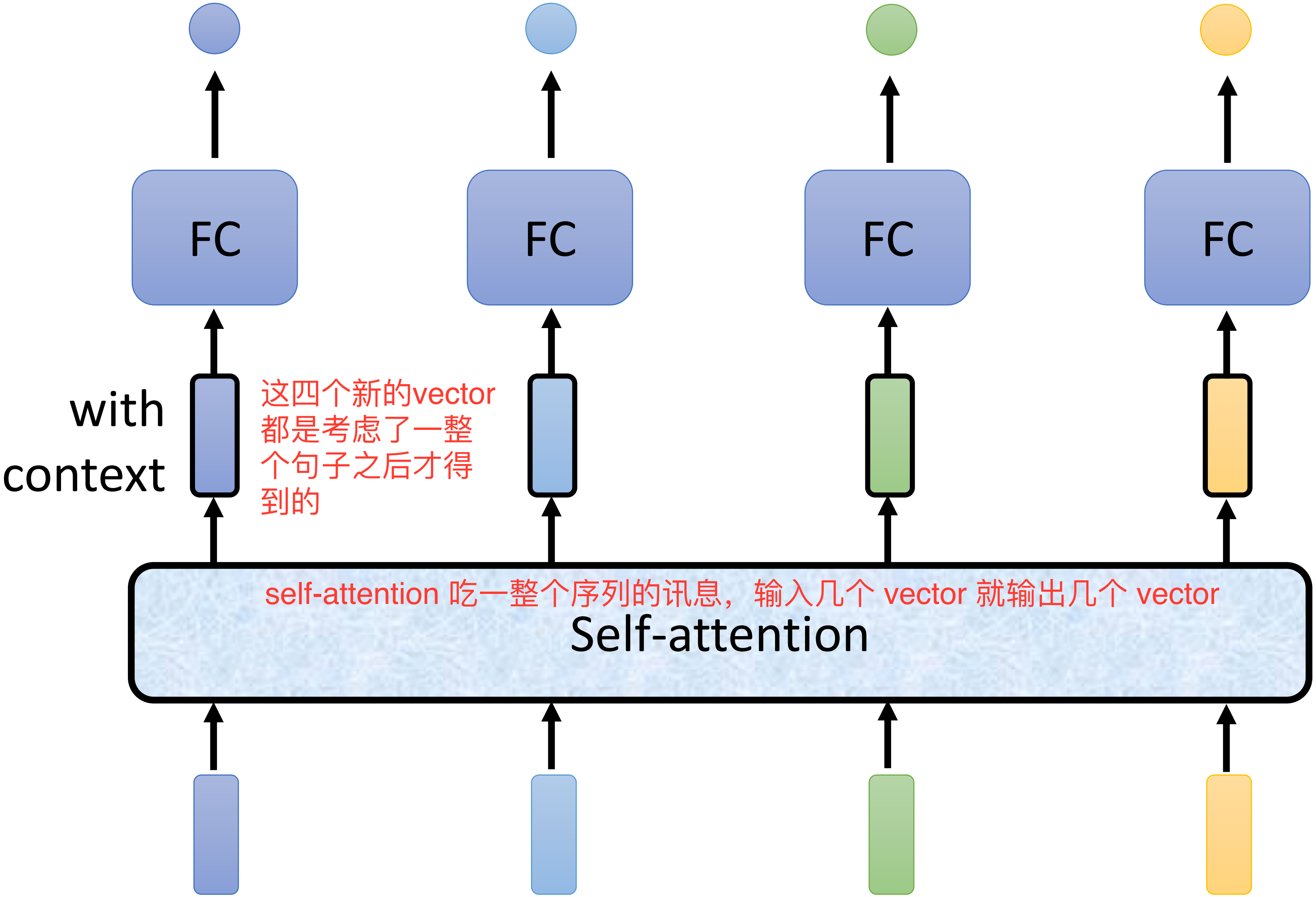
a window covers the whole sequence?



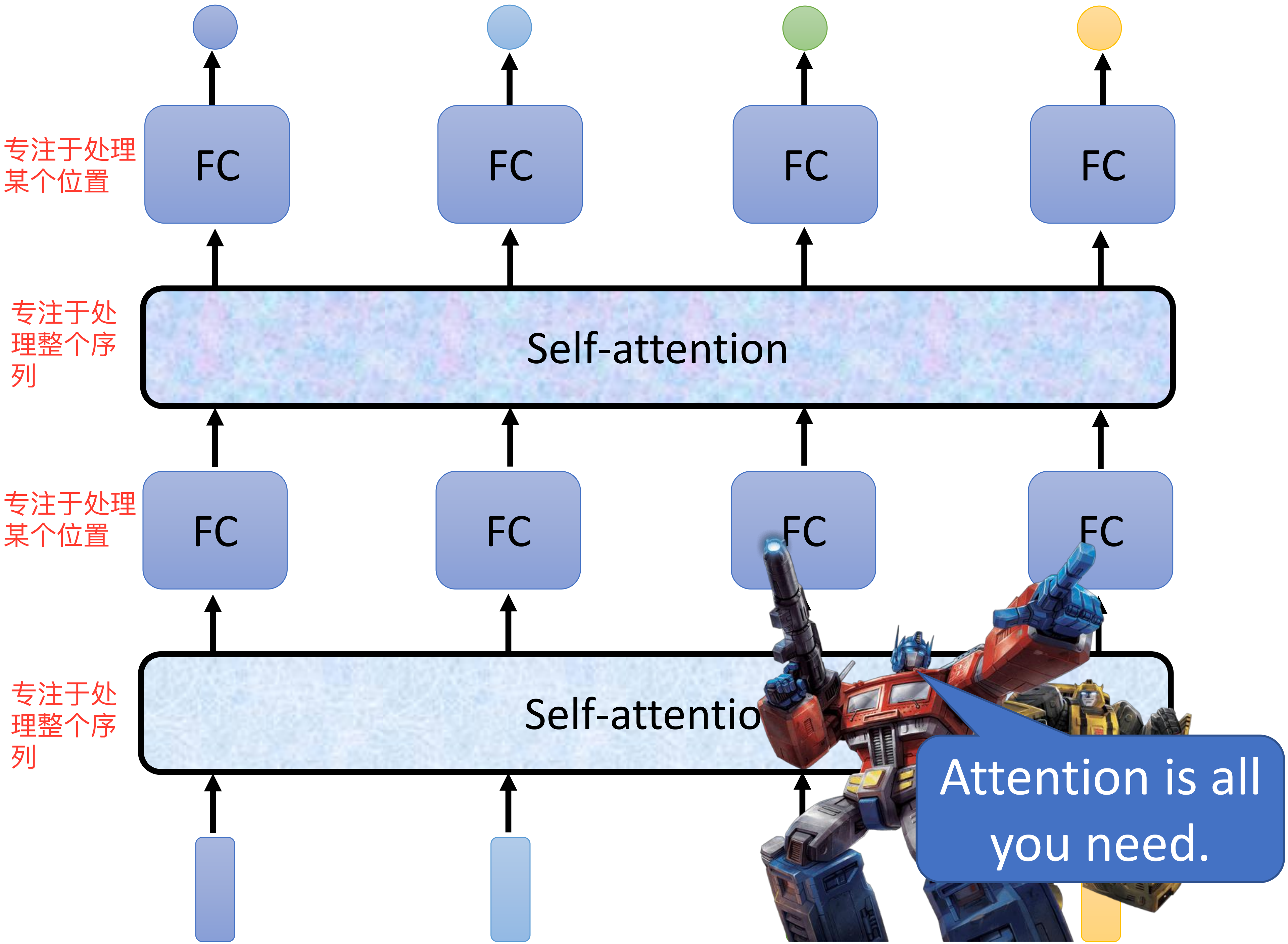
这样的方法也有局限
前后文窗口有限
万一今天有个任务需要整个sequence呢?

因为输入有长有短不固定怎么办?
统计最长的window?
运算量大且容易overfitting

Self-attention



self-attention 可以叠加很多次很多次很多次...

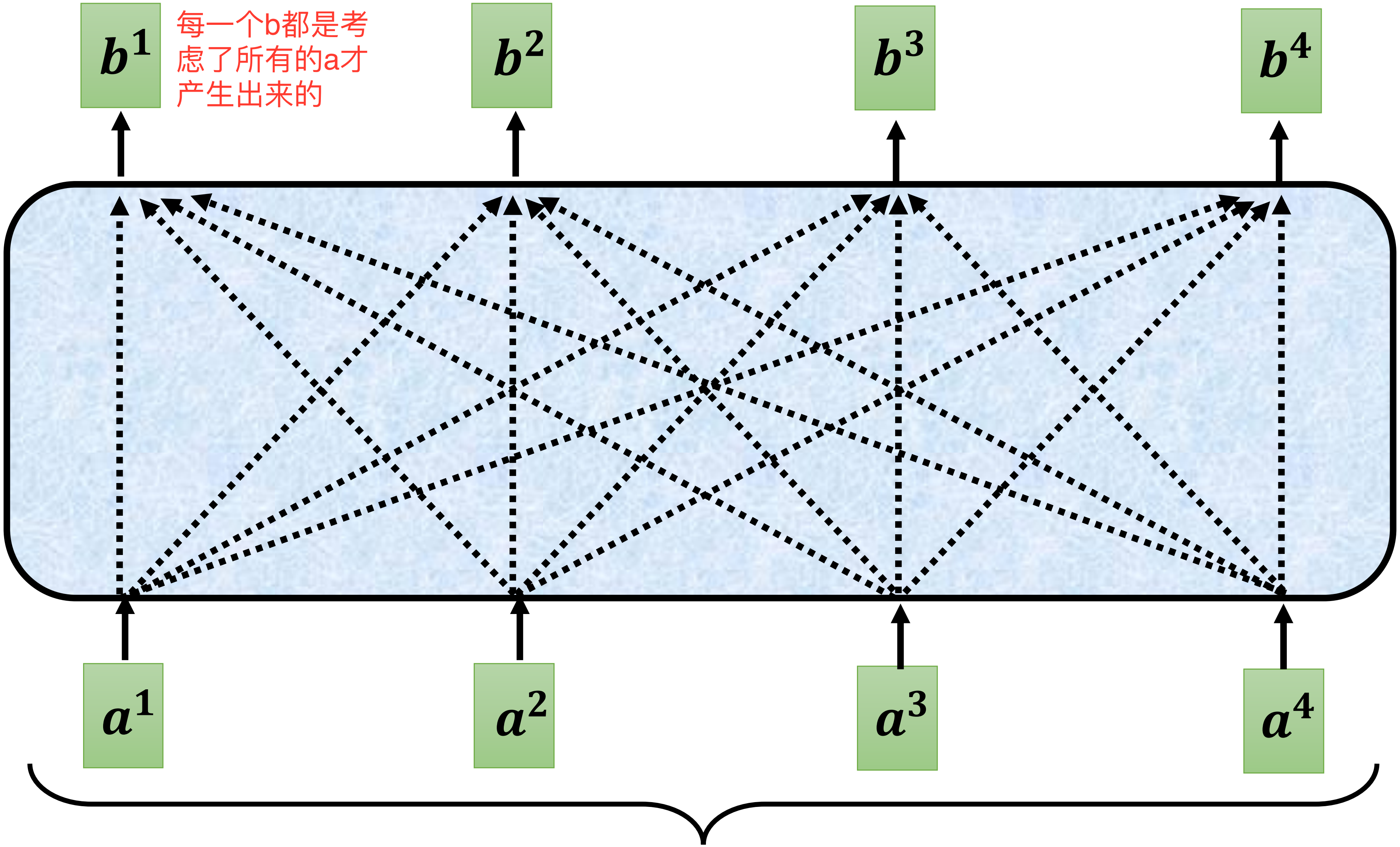


https://arxiv.org/abs/1706.03762

Self-attention

Transformer 最重要的地方就是 self-attention
其实更早的paper就提出来了这个思想，
Transformer 把它发扬光大

每一个b都是考虑了所有的a才产生出来的



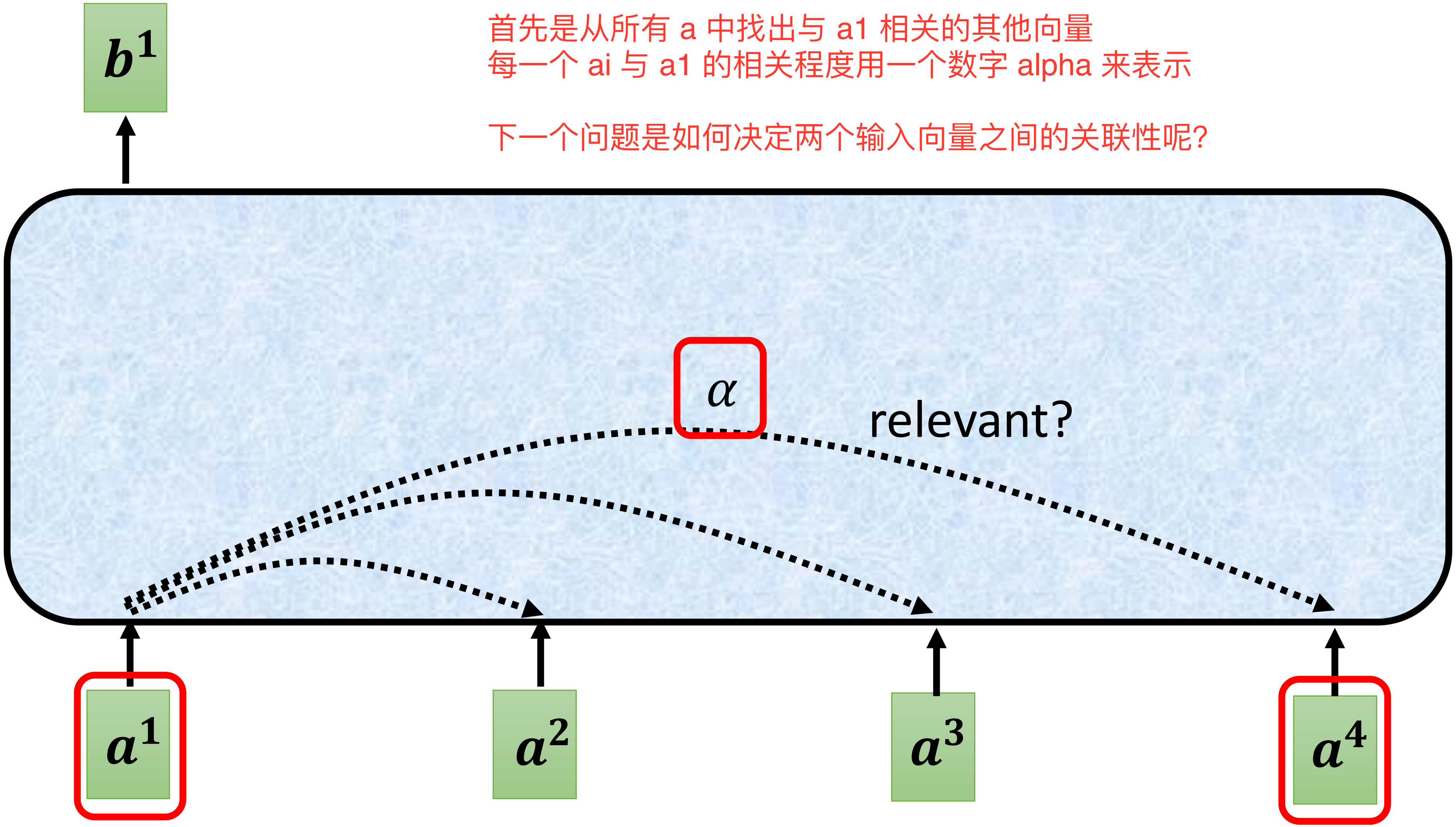
Can be either **input** or **a hidden layer**

Self-attention

关键在于怎么产生 b1?

首先是从所有 a 中找出与 a1 相关的其他向量
每一个 ai 与 a1 的相关程度用一个数字 alpha 来表示

下一个问题是如何决定两个输入向量之间的关联性呢?

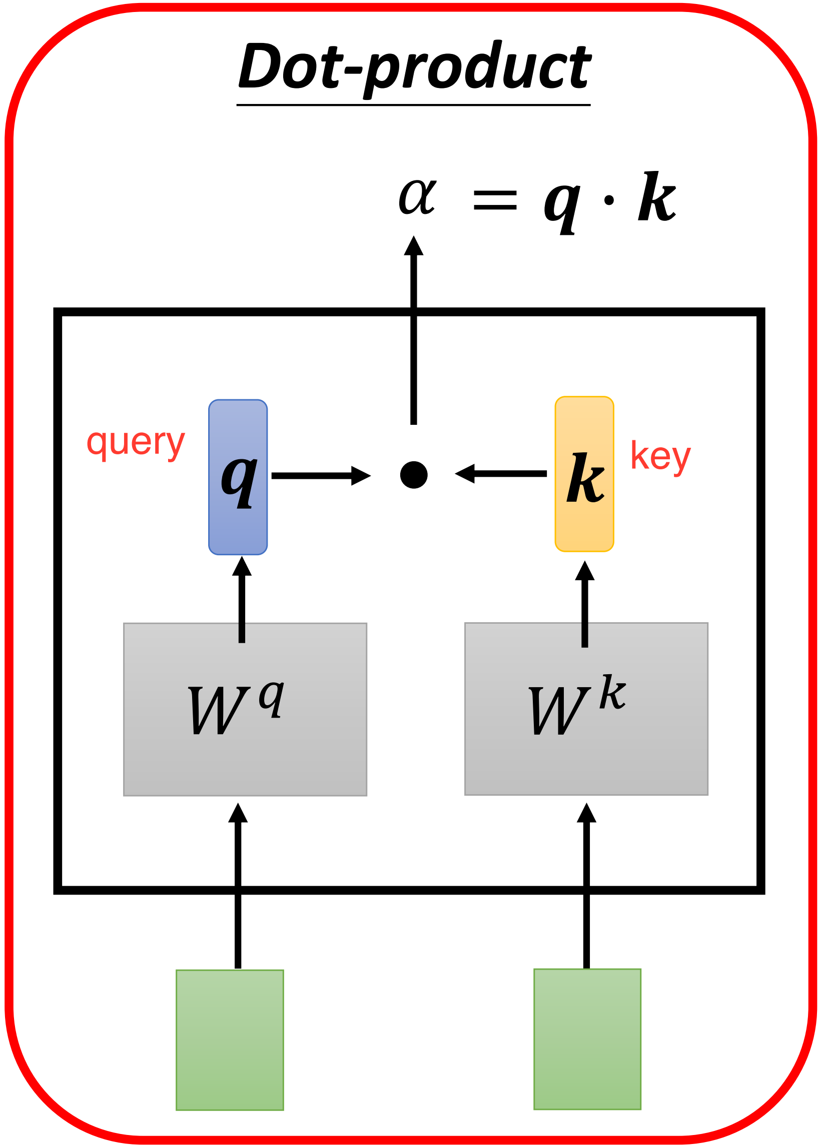


Find the relevant vectors in a sequence

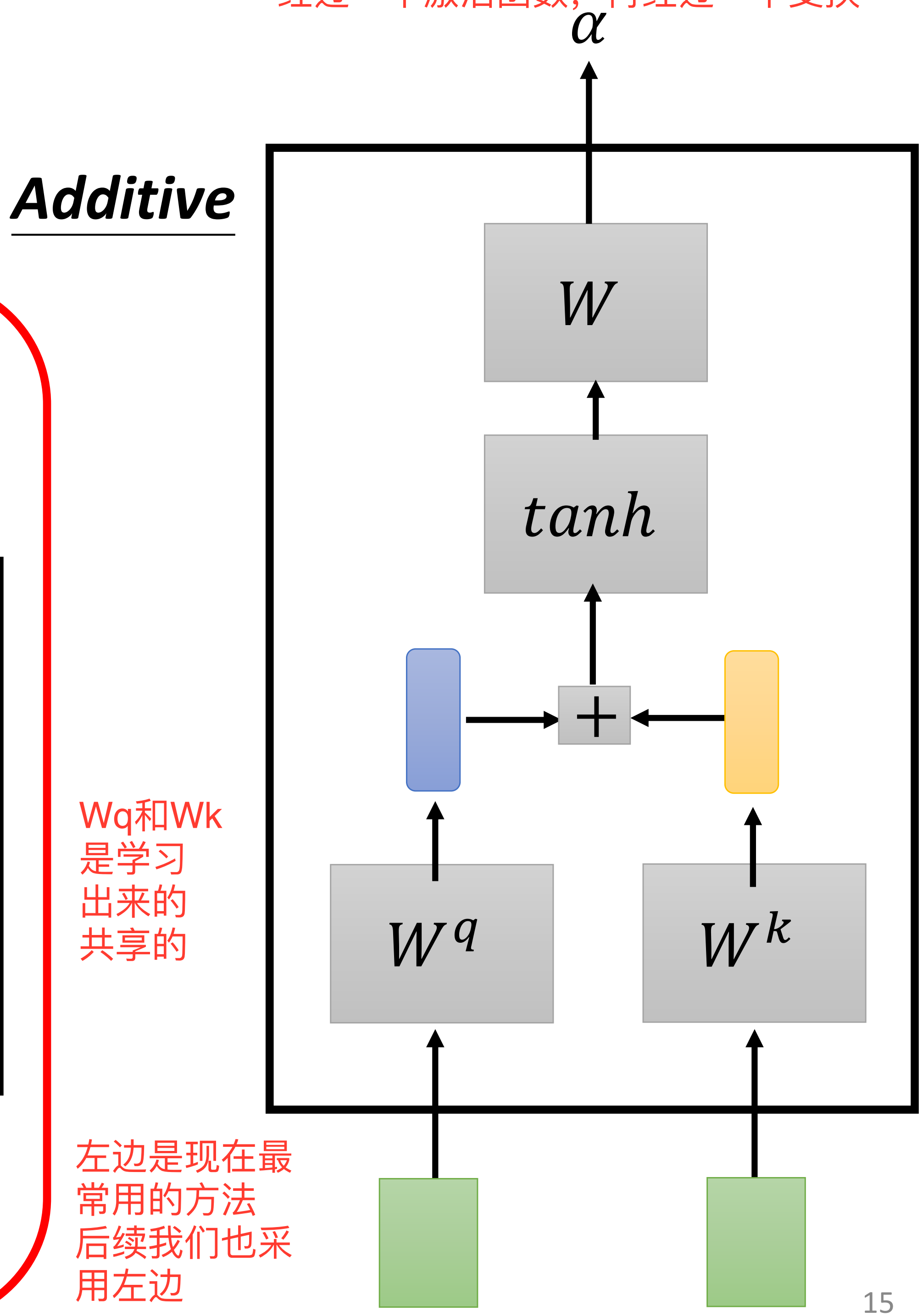
需要一个计算 attention 的模组
输入两个向量输出 alpha

Self-attention

其中一种做法就是点乘(元素乘再加)



Additive的做法就是把query和key加起来
经过一个激活函数，再经过一个变换



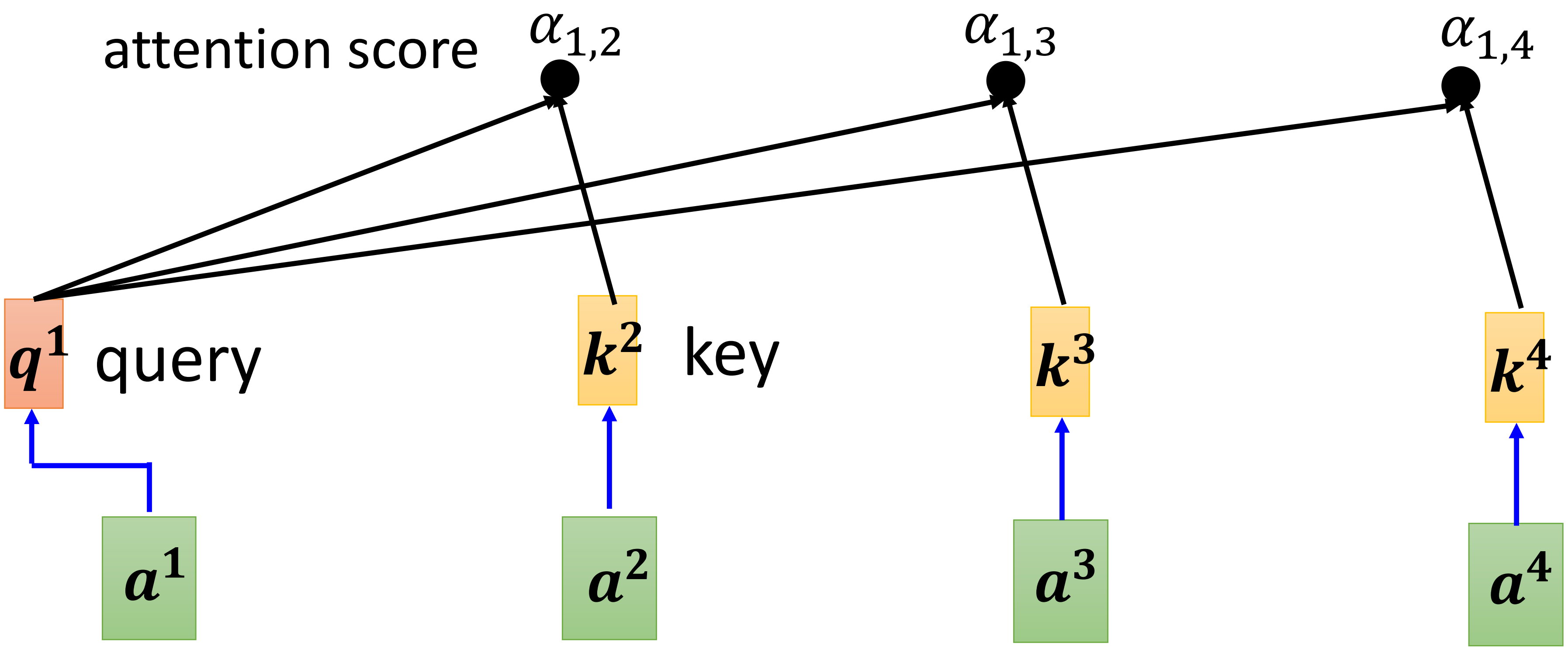
Wq和Wk
是学习
出来的
共享的

左边是现在最
常用的方法
后续我们也采
用左边

Self-attention

我们要分别计算a1与a2，a3，a4之间的关联性
把代表a1的query 乘上 a2，a3，a4的key
从名字我们可以看出，query相当于查询内容，key相当于关键词keyword，点积就是他们之间的相关性
直观理解例如 query 是 我爱你中国，key 有 中国，美国，巴基斯坦....

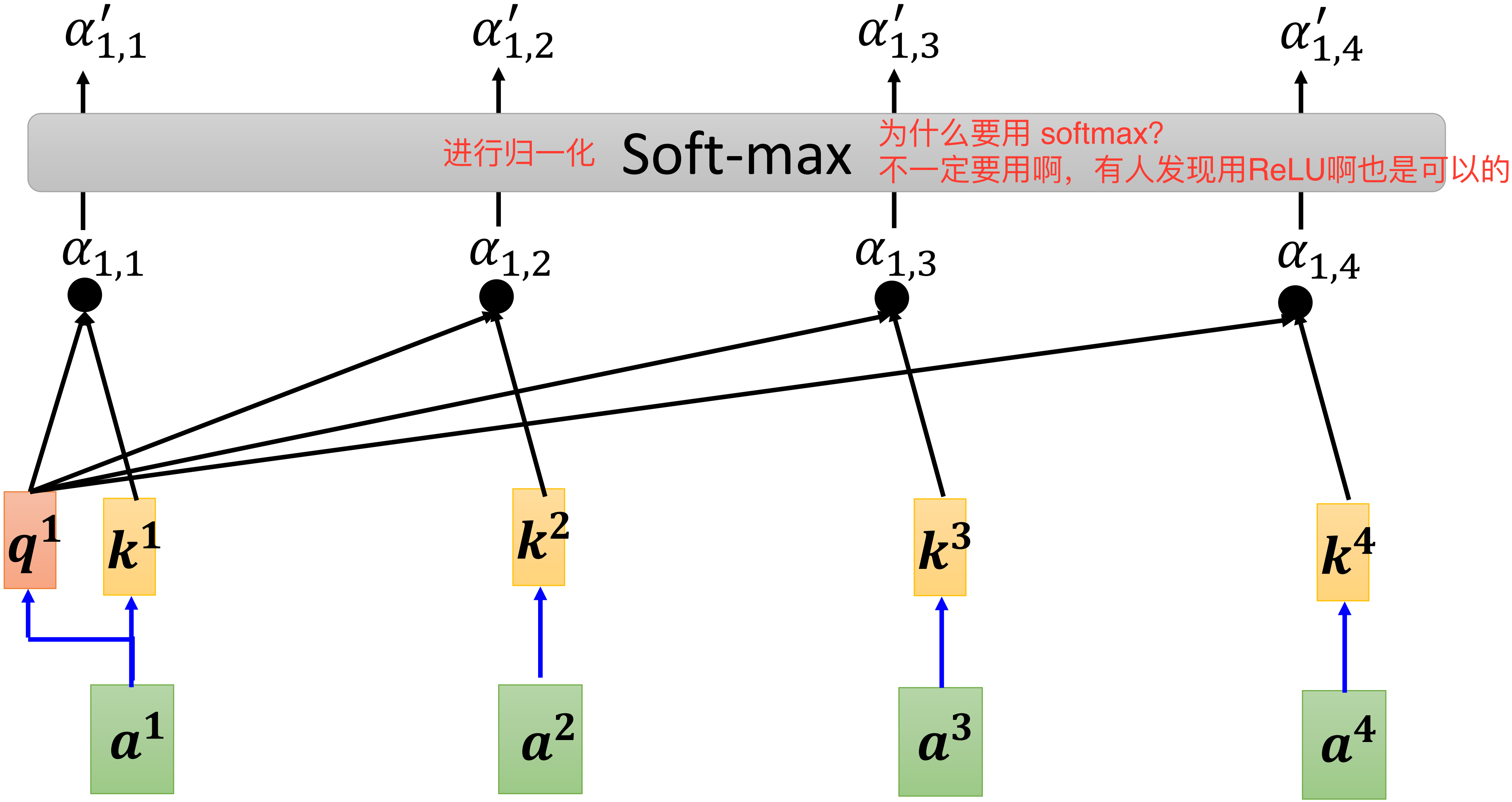
$$\alpha_{1,2} = \boldsymbol{q}^1 \cdot \boldsymbol{k}^2 \qquad \alpha_{1,3} = \boldsymbol{q}^1 \cdot \boldsymbol{k}^3 \qquad \alpha_{1,4} = \boldsymbol{q}^1 \cdot \boldsymbol{k}^4$$



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

Self-attention

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



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

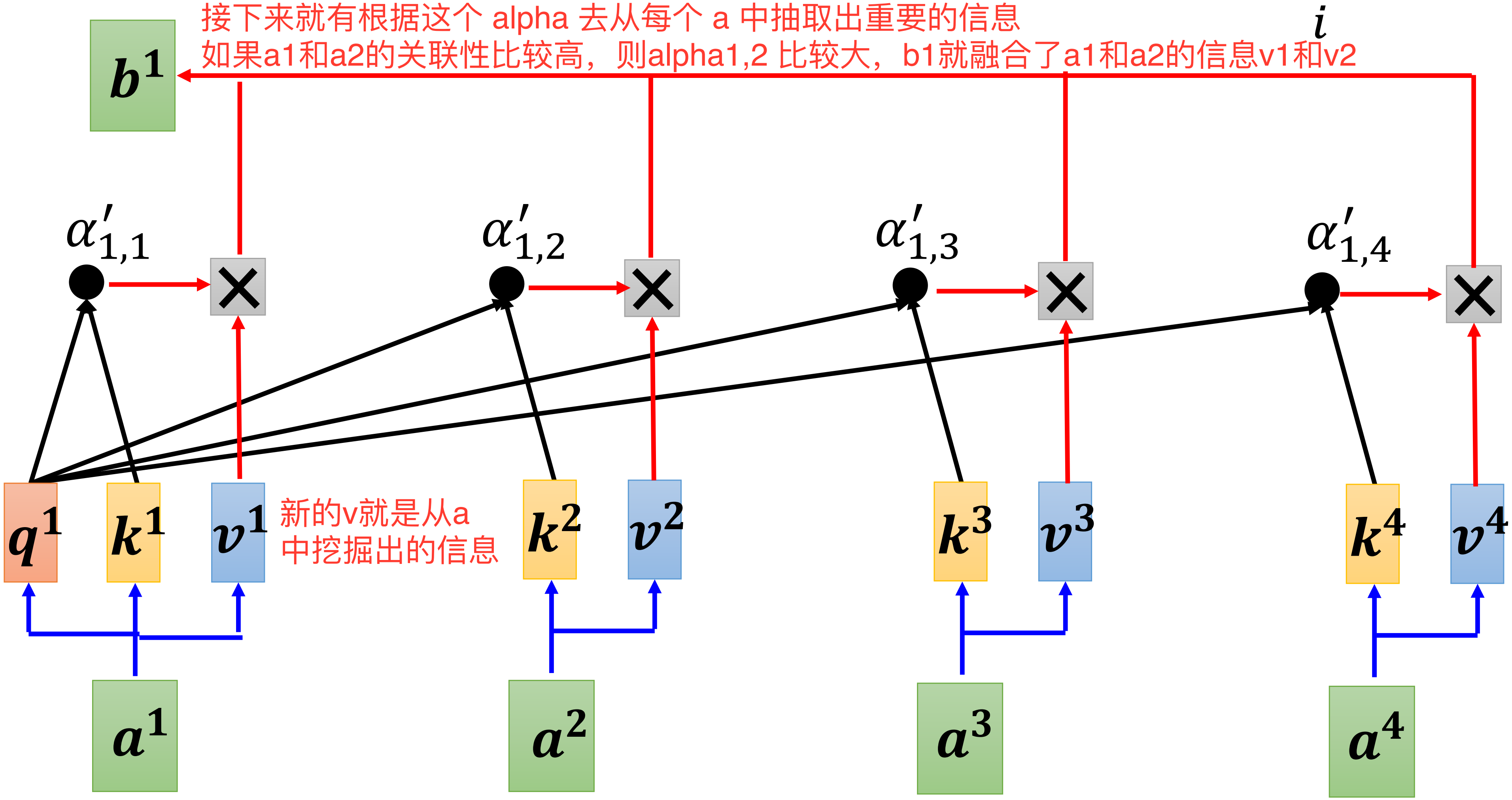
$k^1 = W^k a^1$

实际做的时候也会算自己和自己之间的关联性
我的理解是，指导自己的key要和自己的query关联
避免自己的key和query表意不一

Self-attention

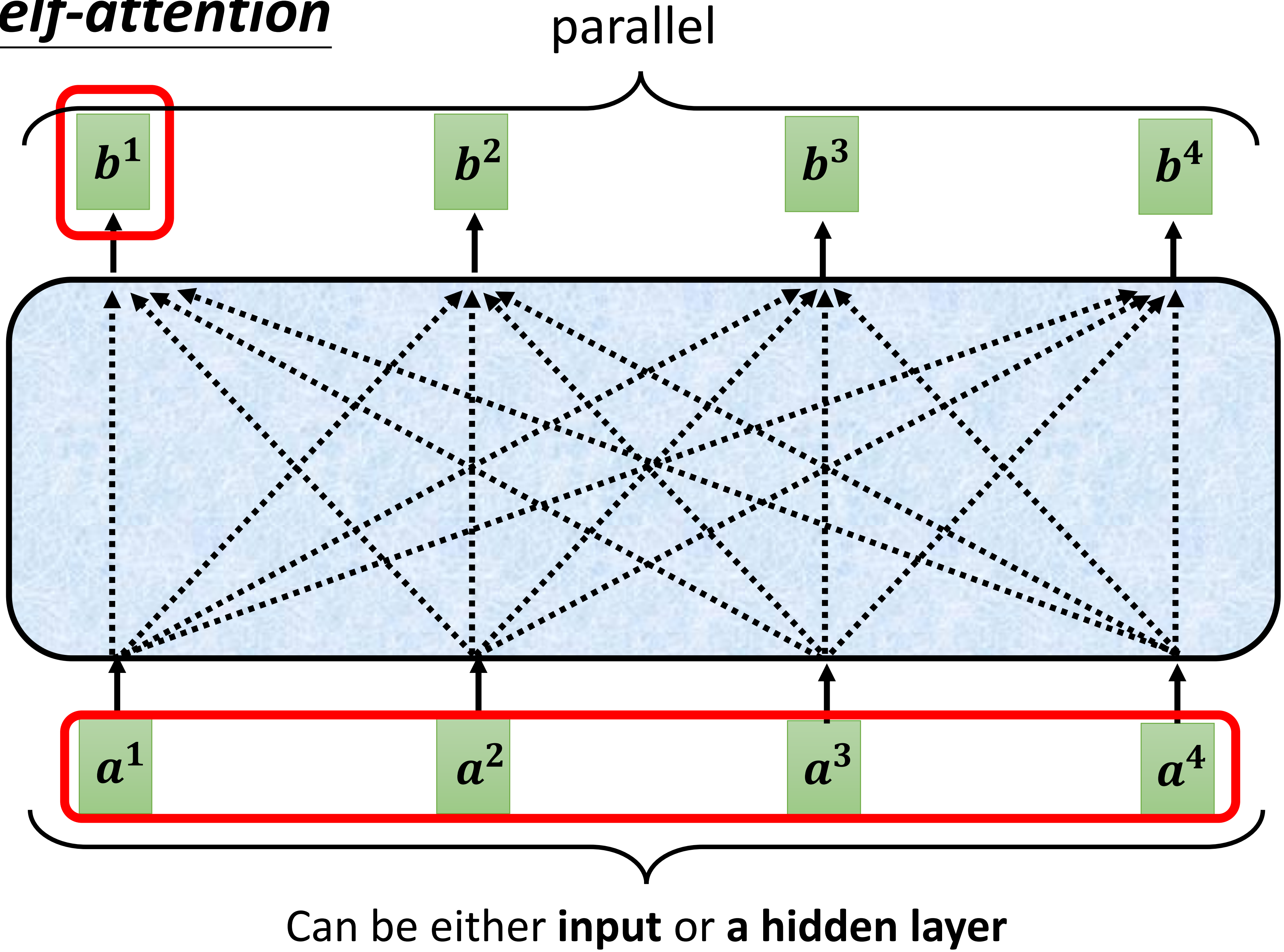
Extract information based on attention scores

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



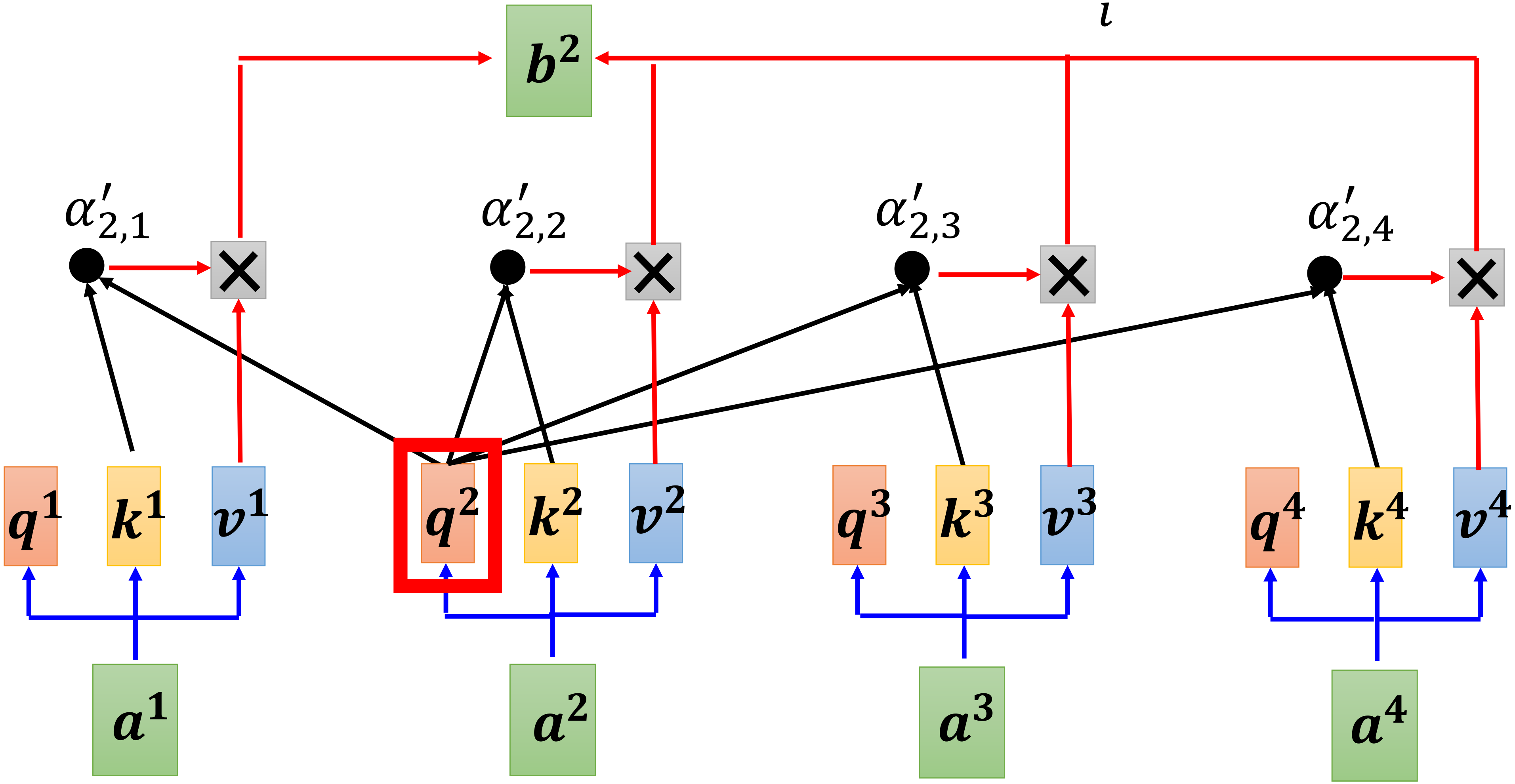
$v^1 = W^v a^1$ $\quad v^2 = W^v a^2$ $\quad v^3 = W^v a^3$ $\quad v^4 = W^v a^4$

Self-attention



Self-attention

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



Self-attention

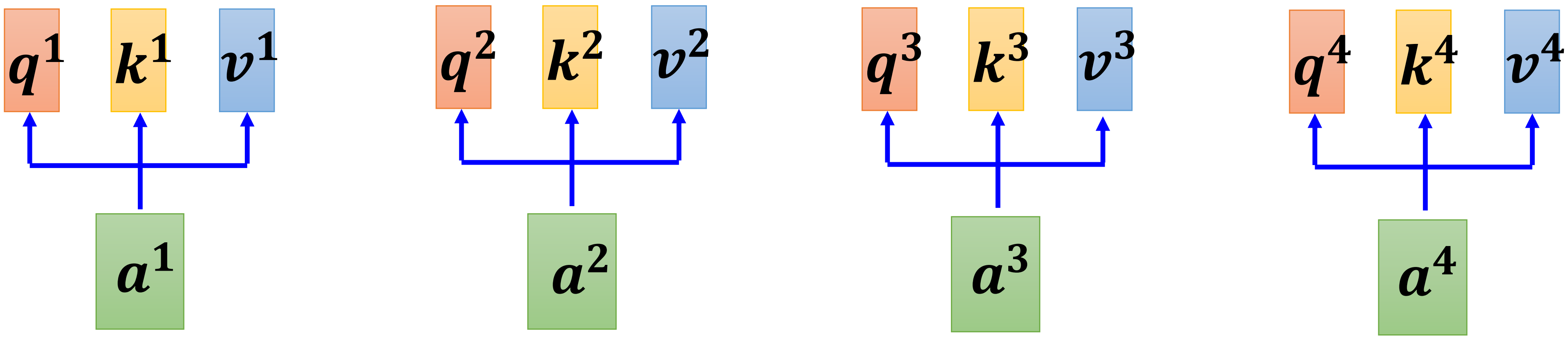
注意b1到b4不是依次计算的
而是一次性全部计算出来的

从矩阵乘法的角度来理解一下

$$q^i = W^q a^i \qquad \underbrace{q^1 \, q^2 \, q^3 \, q^4}_{Q} = W^q \, \underbrace{a^1 \, a^2 \, a^3 \, a^4}_{I}$$

$$k^i = W^k a^i \qquad \underbrace{k^1 \, k^2 \, k^3 \, k^4}_{K} = W^k \, \underbrace{a^1 \, a^2 \, a^3 \, a^4}_{I}$$

$$v^i = W^v a^i \qquad \underbrace{v^1 \, v^2 \, v^3 \, v^4}_{V} = W^v \, \underbrace{a^1 \, a^2 \, a^3 \, a^4}_{I}$$

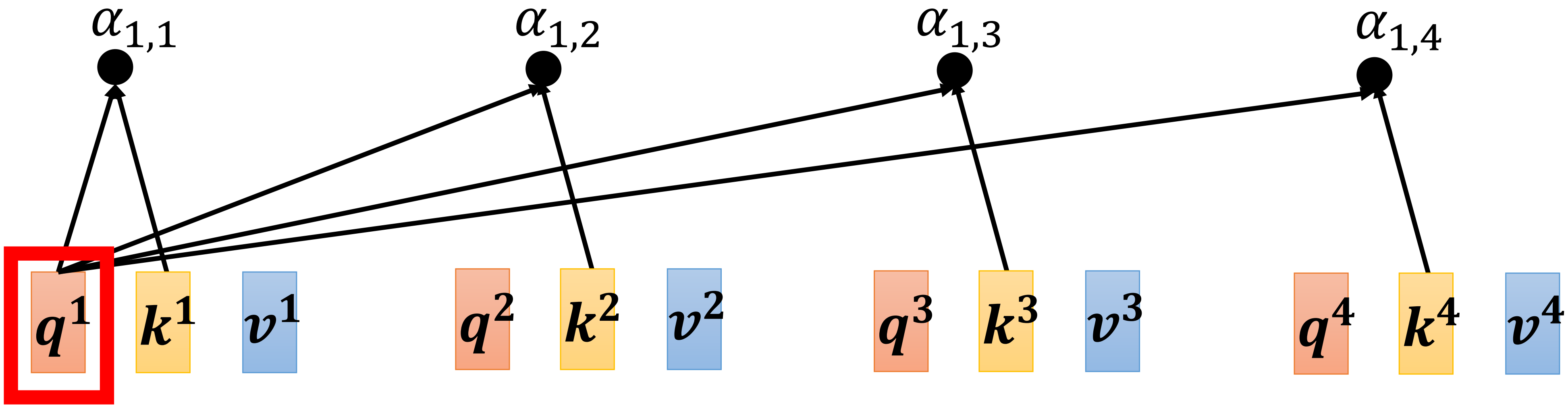


Self-attention

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

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

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

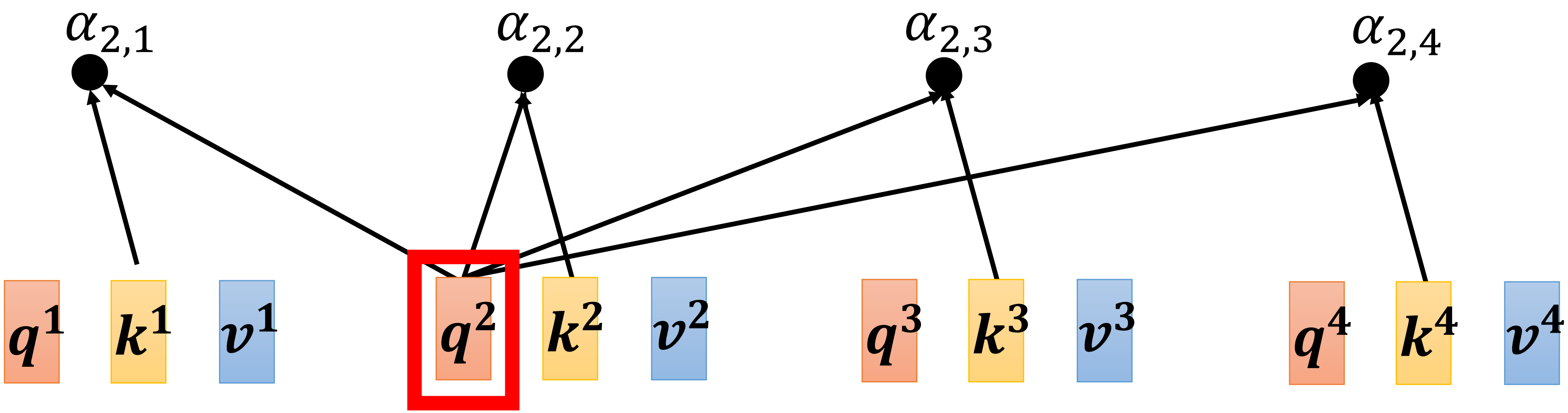


Self-attention

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

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

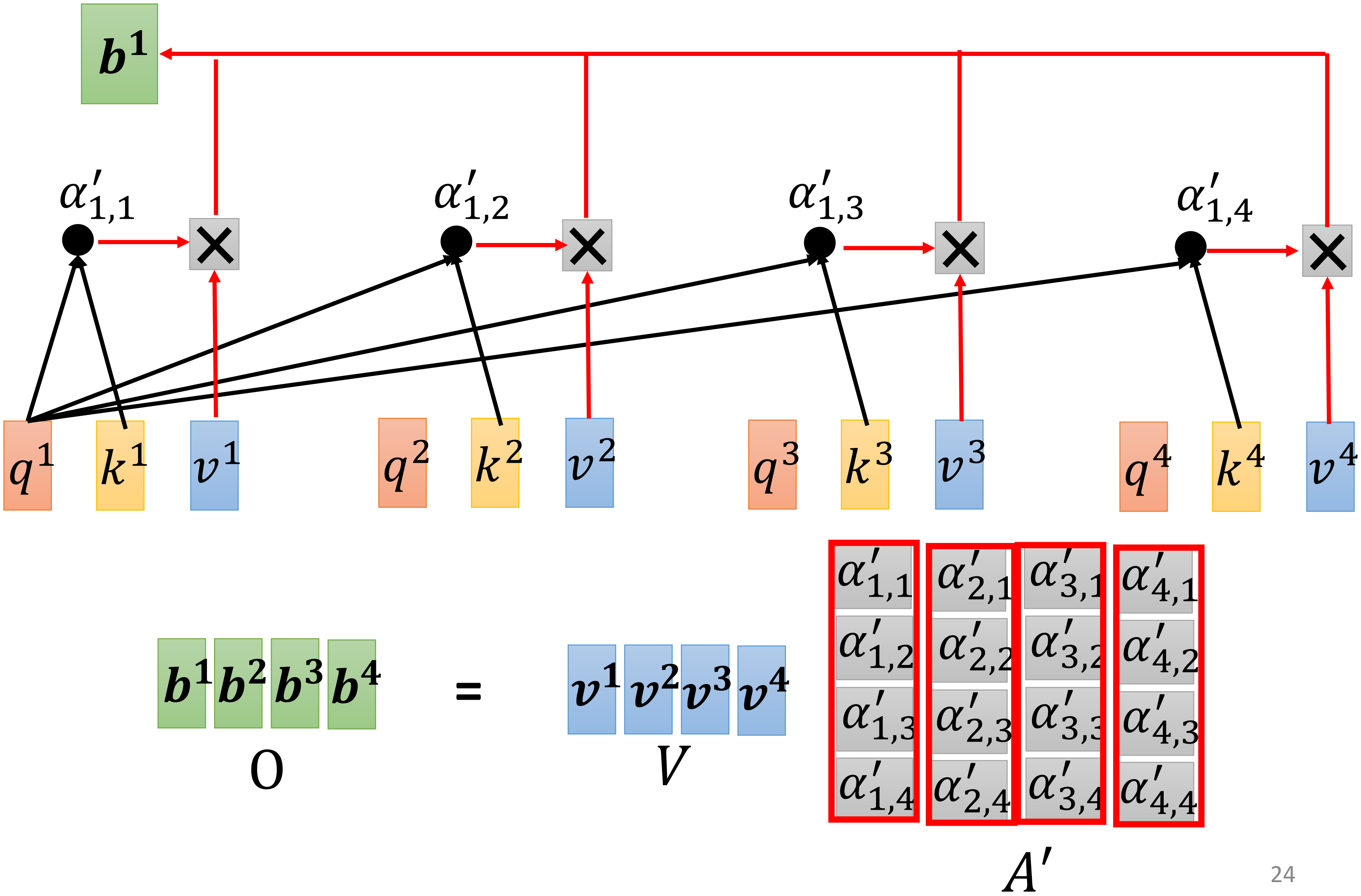
$$\begin{bmatrix} \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{bmatrix} = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} q^1$$



$$\underbrace{\begin{bmatrix} \alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1} \\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2} \\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3} \\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4} \end{bmatrix}}_{A'} \Leftarrow \underbrace{\begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix}}_{K^T} \underbrace{\begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix}}_{Q}$$

softmax按列做归一化

Self-attention



Self-attention

$$Q = W^q \, I$$

$$K = W^k \, I$$

$$V = W^v \, I$$

唯一需要学习的参数
其他的操作都是人为设置好的

Parameters to be learned

$$A' \leftarrow A = K^T \, Q$$

Attention Matrix

$$O = V \, A'$$

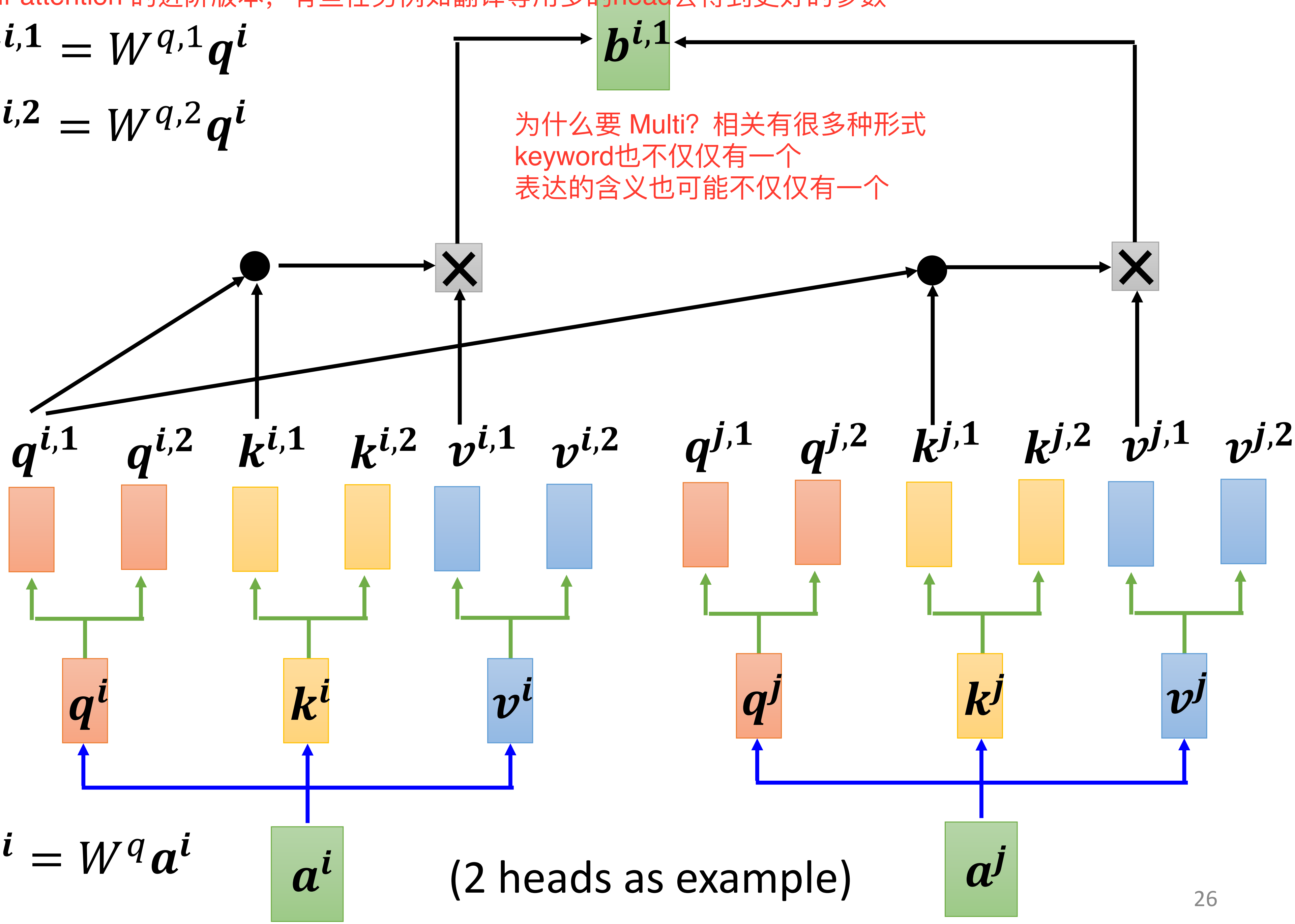
Multi-head Self-attention Different types of relevance

self-attention 的进阶版本，有些任务例如翻译等用多的head会得到更好的参数

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

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

为什么要 Multi？相关有很多种形式
keyword也不仅仅有一个
表达的含义也可能不仅仅有一个



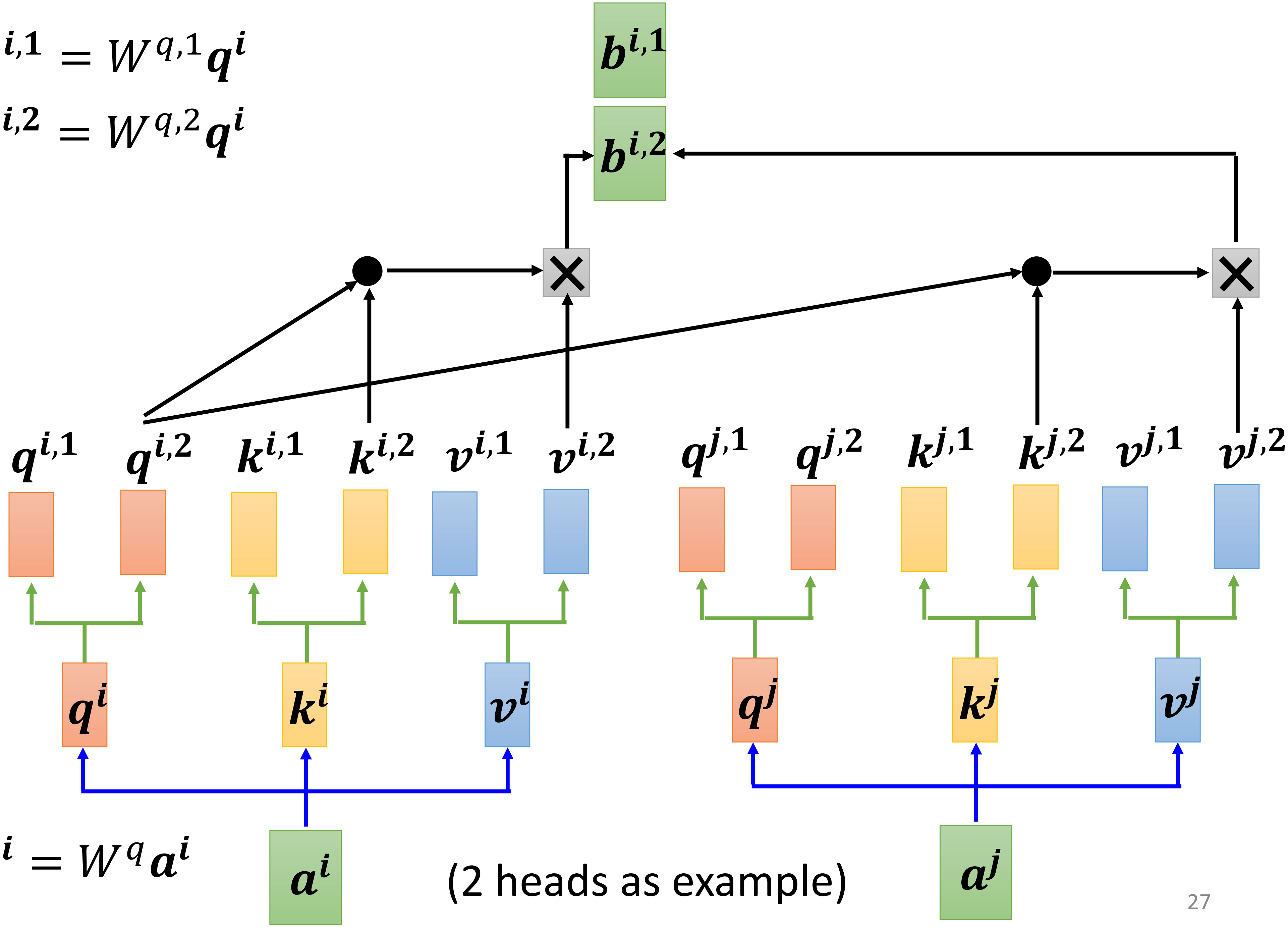
$$q^i = W^q a^i$$

(2 heads as example)

Multi-head Self-attention Different types of relevance

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

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

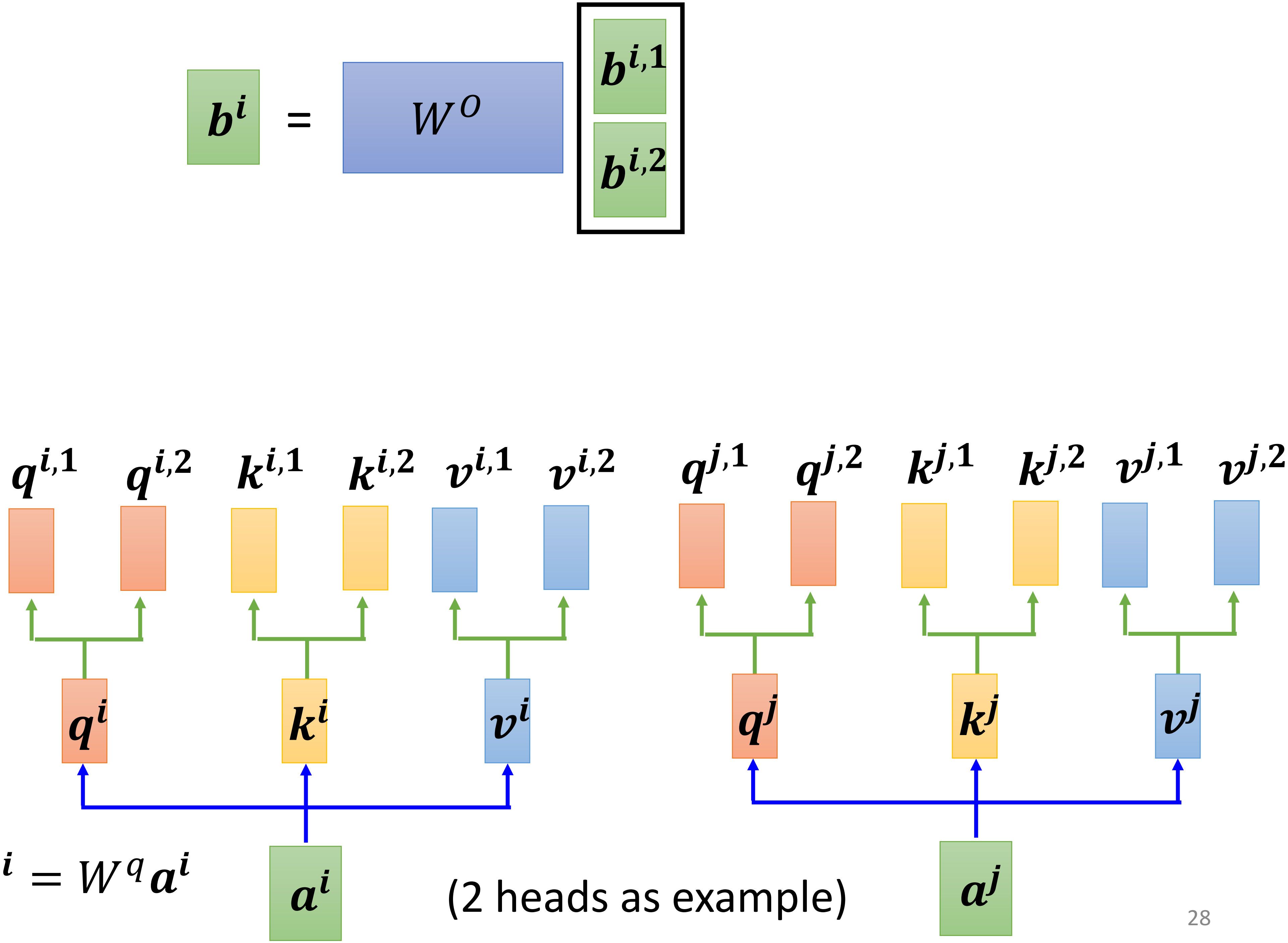


$$q^i = W^q a^i$$

(2 heads as example)

Multi-head Self-attention

Different types of relevance



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

(2 heads as example)

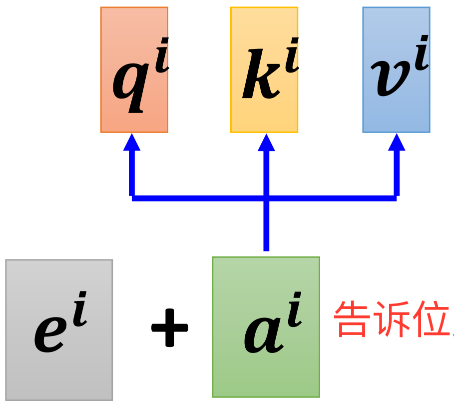
讲到目前为止，发现它少了一个很重要的信息，就是位置的信息。
对 self-attention 而言，每一个输入在第几个不影响生成的对应的 b ，这其实是有问题的
因为位置会是有差别的。对self-attention 而言，任何两个词的位置是一样的，天涯若比邻。

Positional Encoding

但是我们如果知道动词不容易出现在句首
那么可能效果会更好

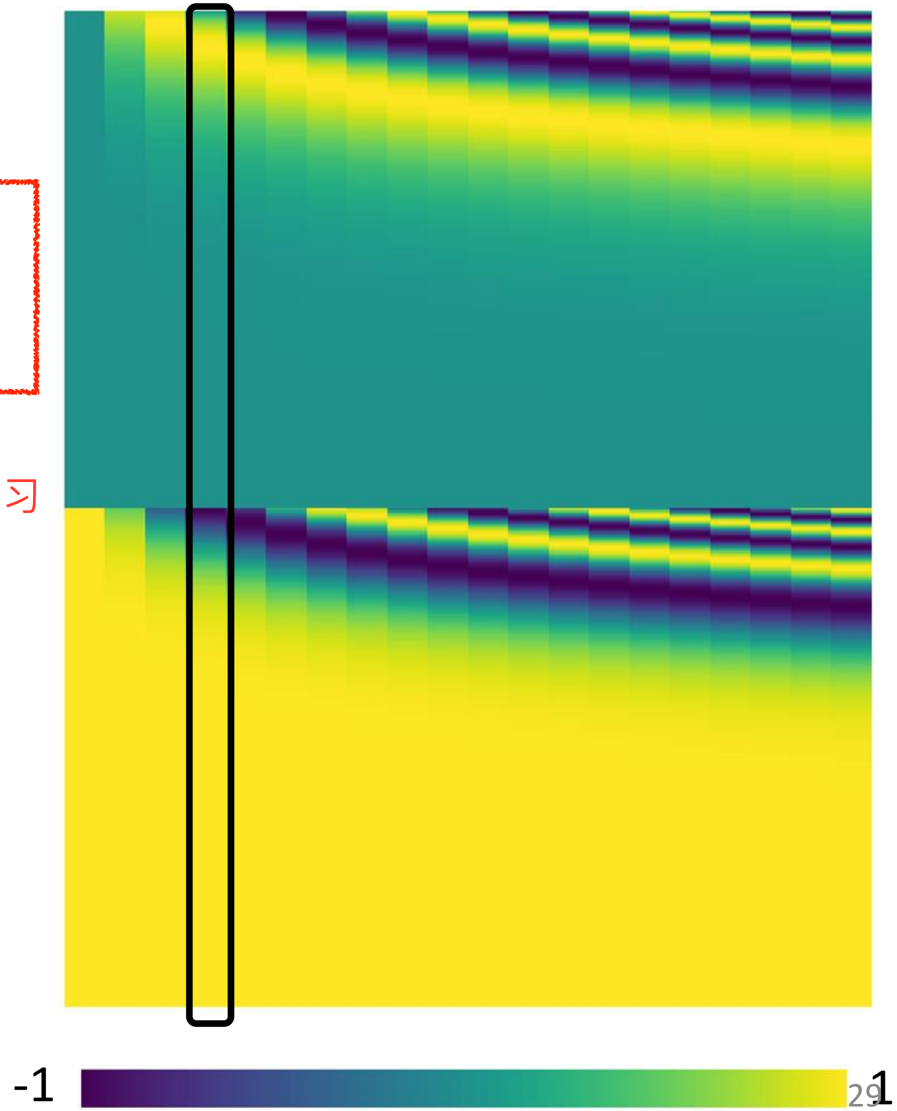
- No position information in self-attention.
- Each position has a unique positional vector e^i
  为每一个位置设置一个独特的专属的位置 vector
- **hand-crafted**
- **learned from data**

可以人工构造
也可以从数据中学习



告诉位置的讯息

Each column represents a positional vector e^i



Attention is all you need 论文使用的位置编码如图，把每个列向量加到不同的a上

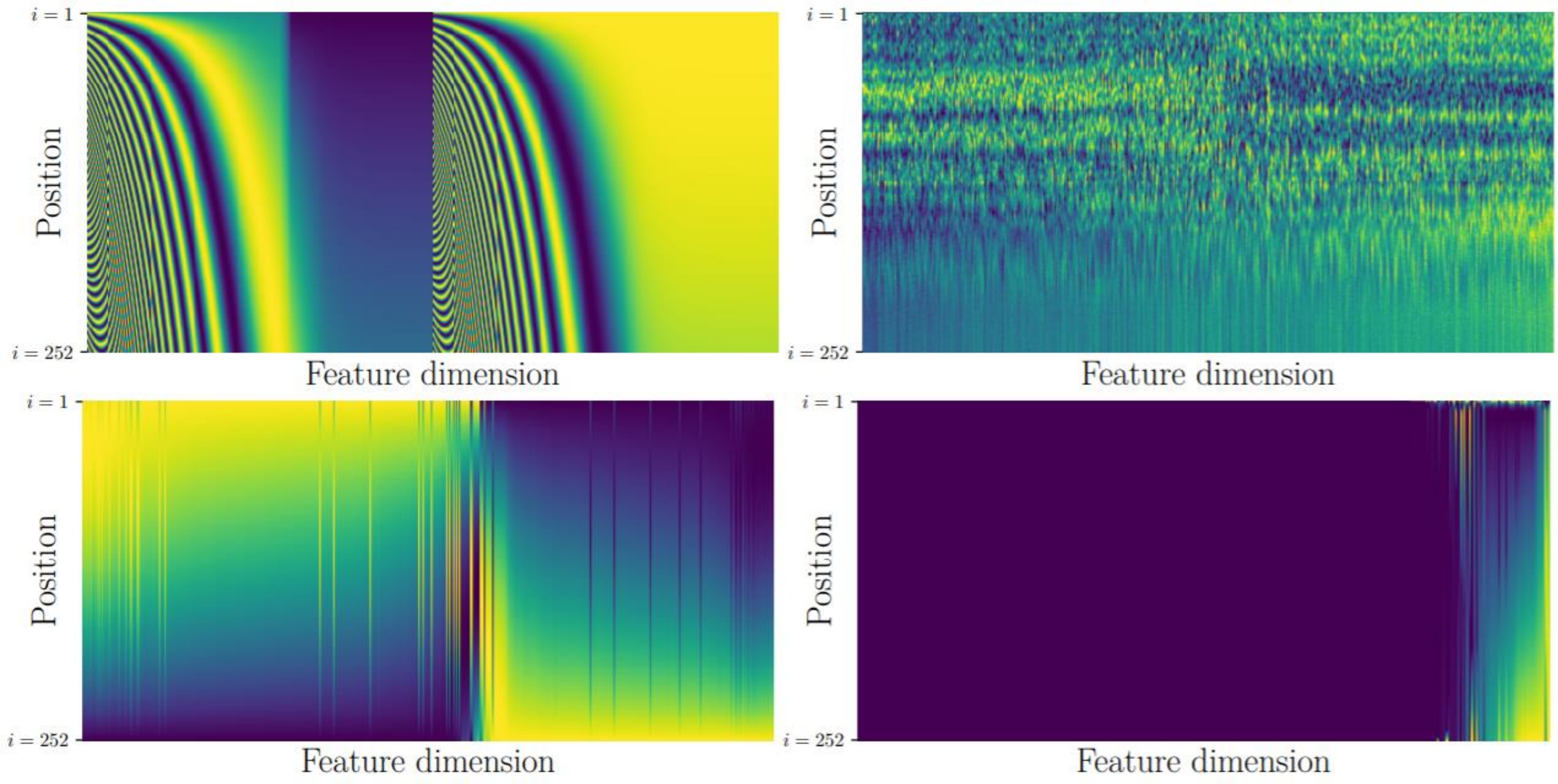
https://arxiv.org/abs/2003.09229

Table 1. Comparing position representation methods

Methods	Inductive	Data-Driven	Parameter Efficient
Sinusoidal (Vaswani et al., 2017)	✓	✗	✓
Embedding (Devlin et al., 2018)	✗	✓	✗
Relative (Shaw et al., 2018)	✗	✓	✓
This paper	✓	✓	✓

(a) Sinusoidal

(b) Position embedding



(c) FLOATER

(d) RNN

Many applications ...

Transformer

https://arxiv.org/abs/1706.03762

BERT

https://arxiv.org/abs/1810.04805

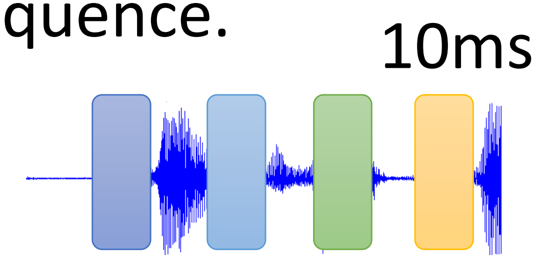
Widely used in Natural Langue Processing (NLP)!

https://arxiv.org/abs/1910.12977

Self-attention for Speech

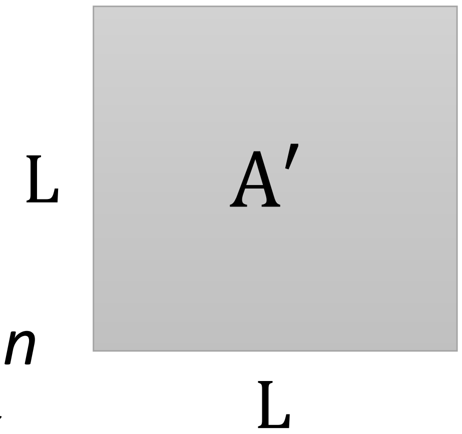
做语音的话，声音讯号的长度很客观

Speech is a very long vector sequence.



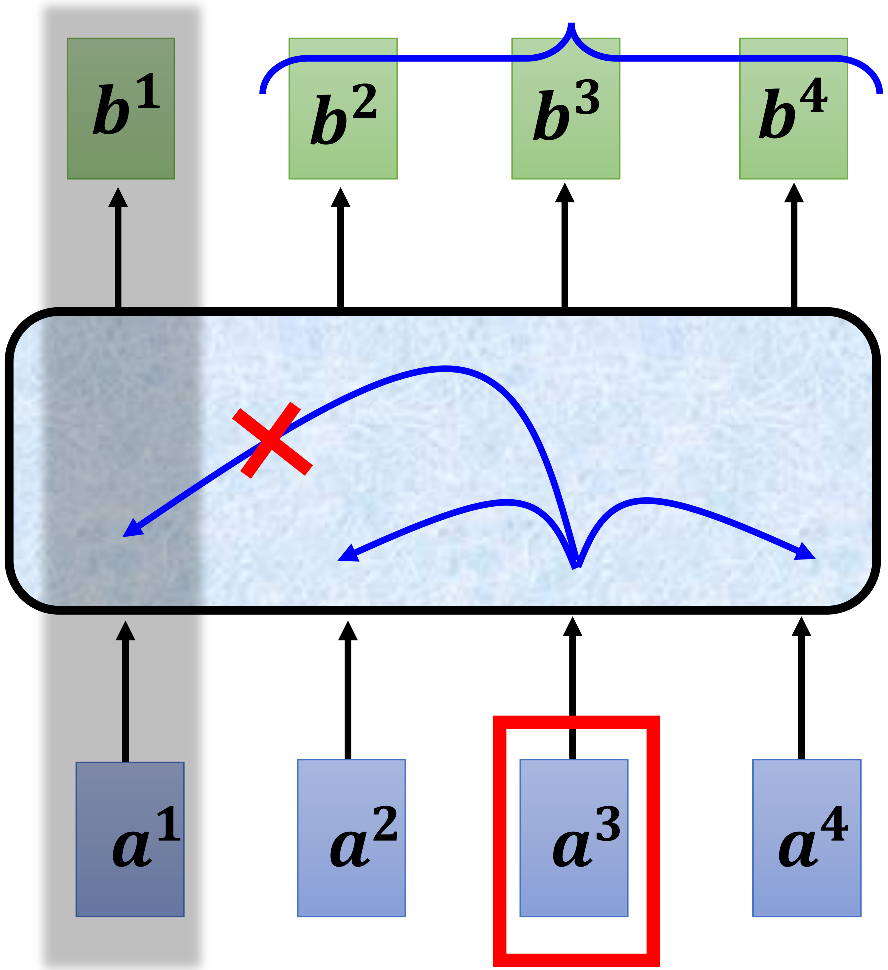
If input sequence is length L

计算复杂度太高
所以给一个窗口W



Attention Matrix

Attention in a range



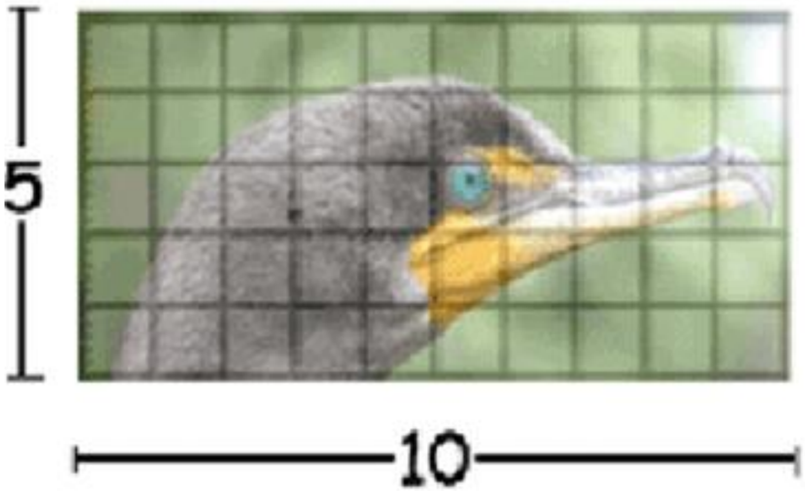
Truncated Self-attention

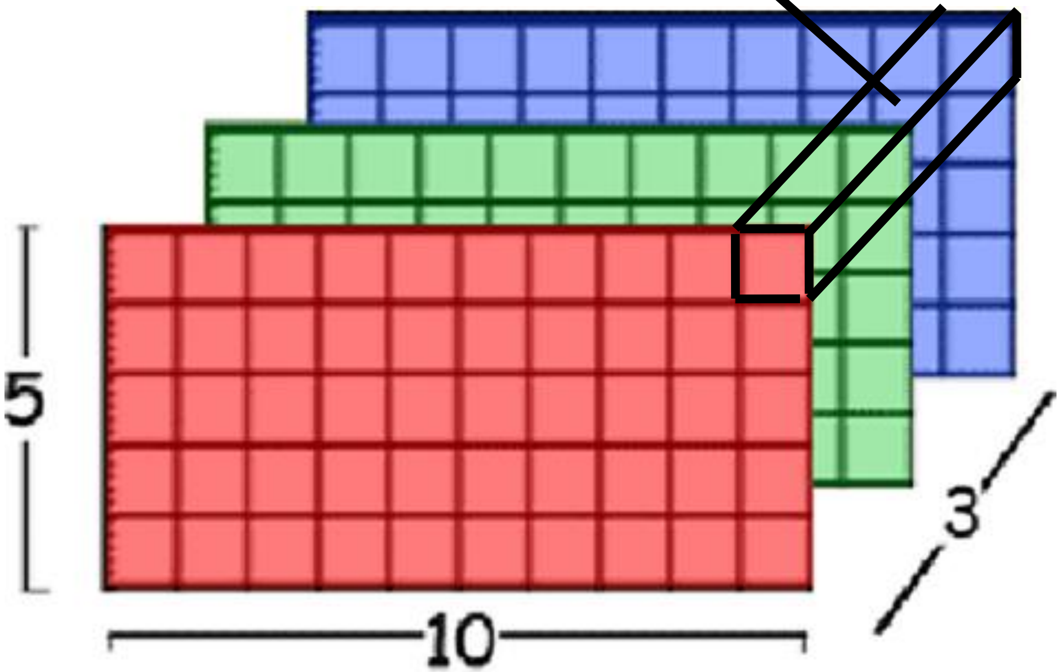
只看一个小窗口就好，人设定的

Self-attention for Image

An **image** can also be considered as a **vector set**.

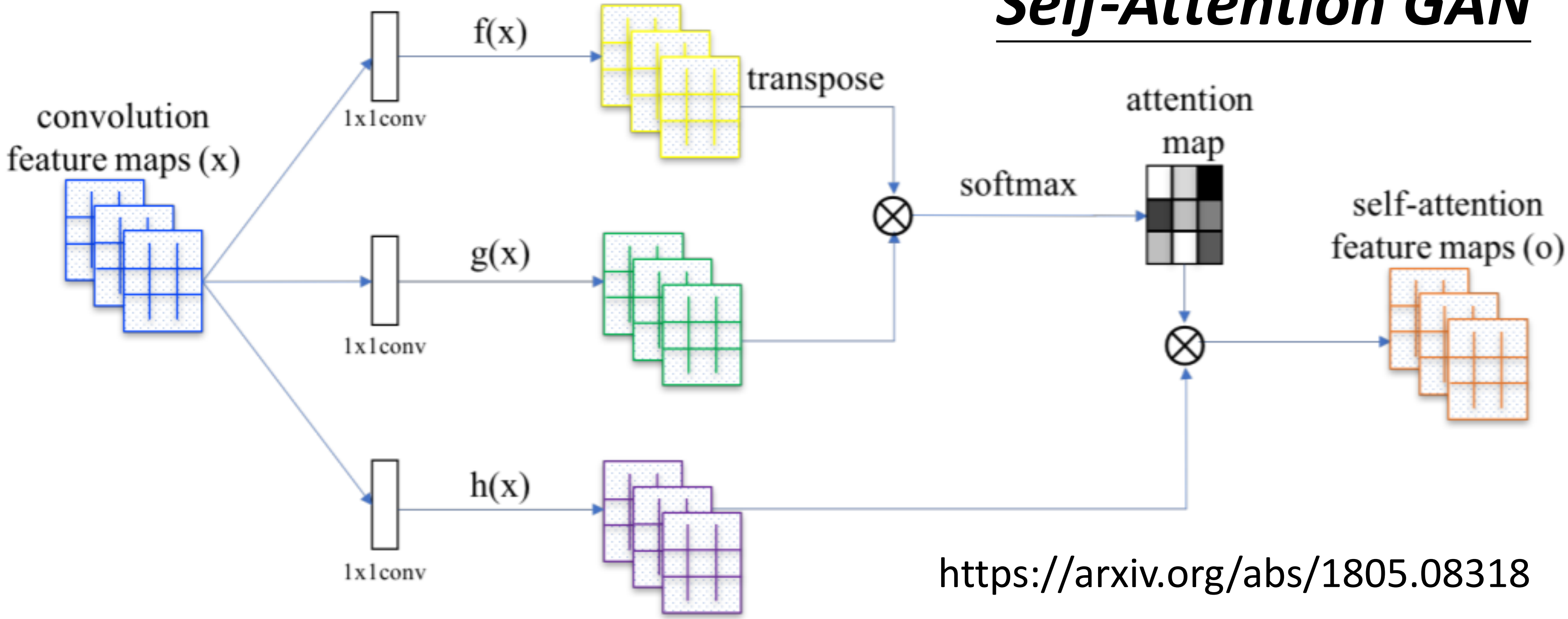
This is a vector.





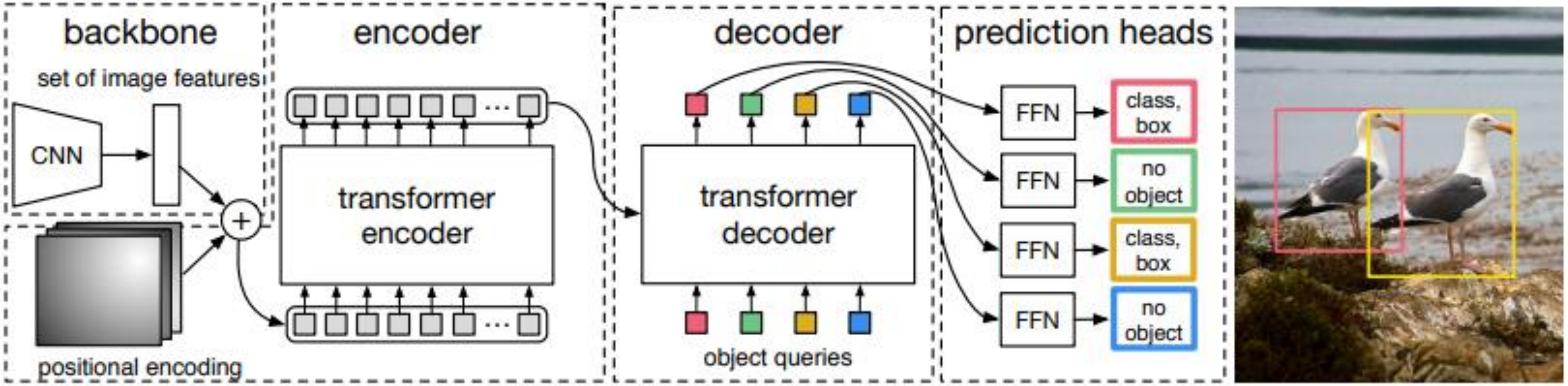
Source of image: https://www.researchgate.net/figure/Color-image-representation-and-RGB-matrix_fig15_282798184

Self-Attention GAN



https://arxiv.org/abs/1805.08318

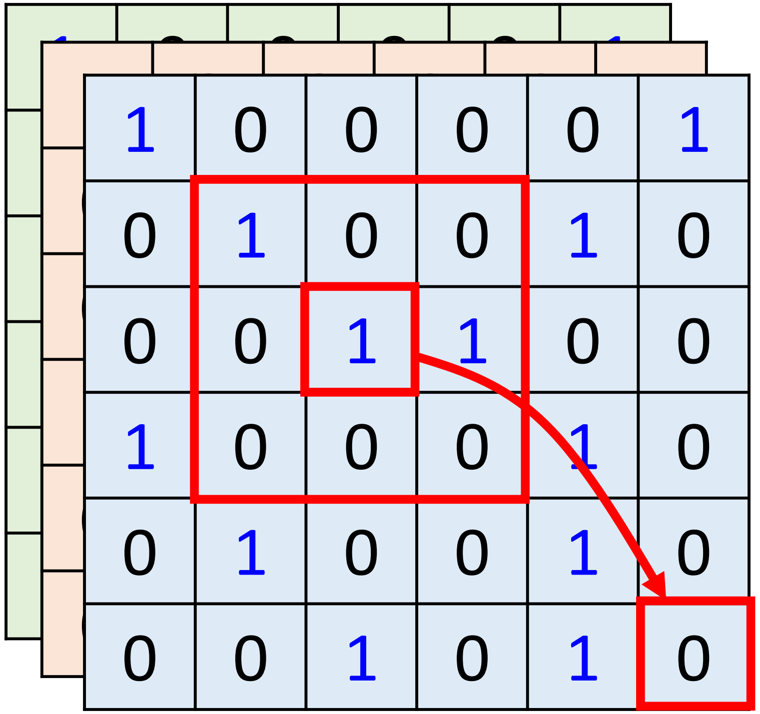
DEtection Transformer (DETR)



https://arxiv.org/abs/2005.12872

Self-attention v.s. CNN

这一张 ppt 很深刻



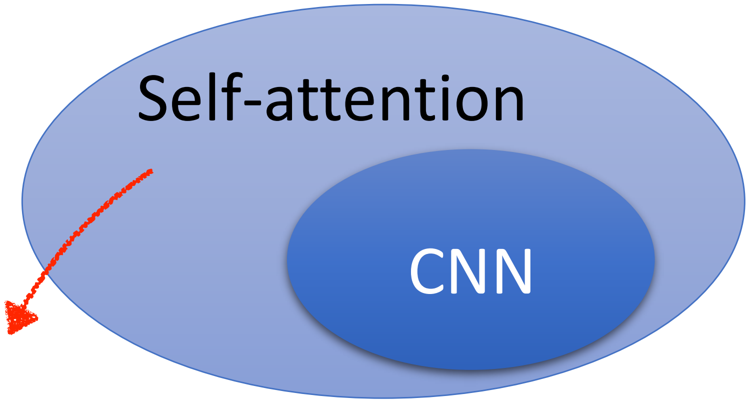
CNN: self-attention that can only attends in a receptive field

- ➢ CNN is simplified self-attention.

Self-attention: CNN with learnable receptive field 自己学receptive field长什么样子

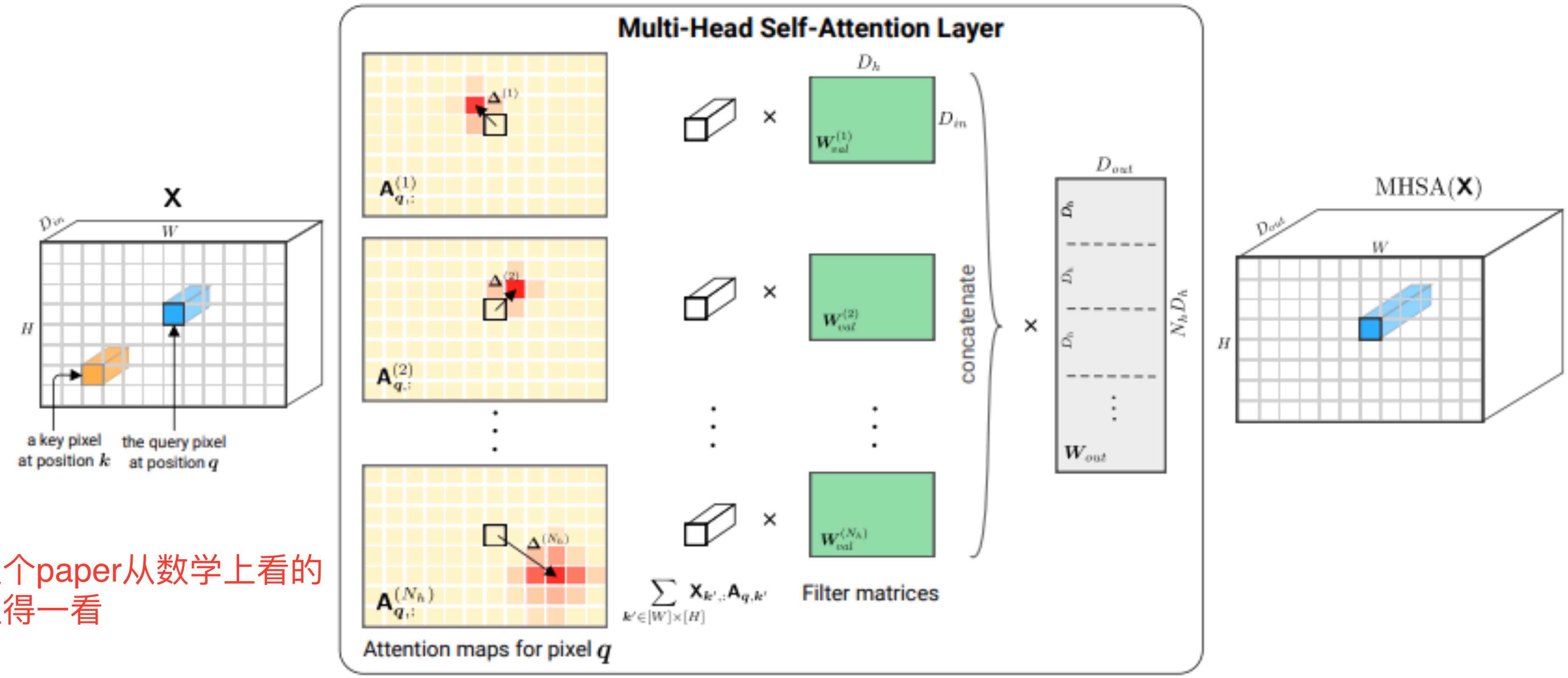
- ➢ Self-attention is the complex version of CNN.

Self-attention v.s. CNN



CNN 是 self-attention 的特例!

需要更多的data
不然容易overfit



这个paper从数学上看的
值得一看

On the Relationship between Self-Attention and Convolutional Layers

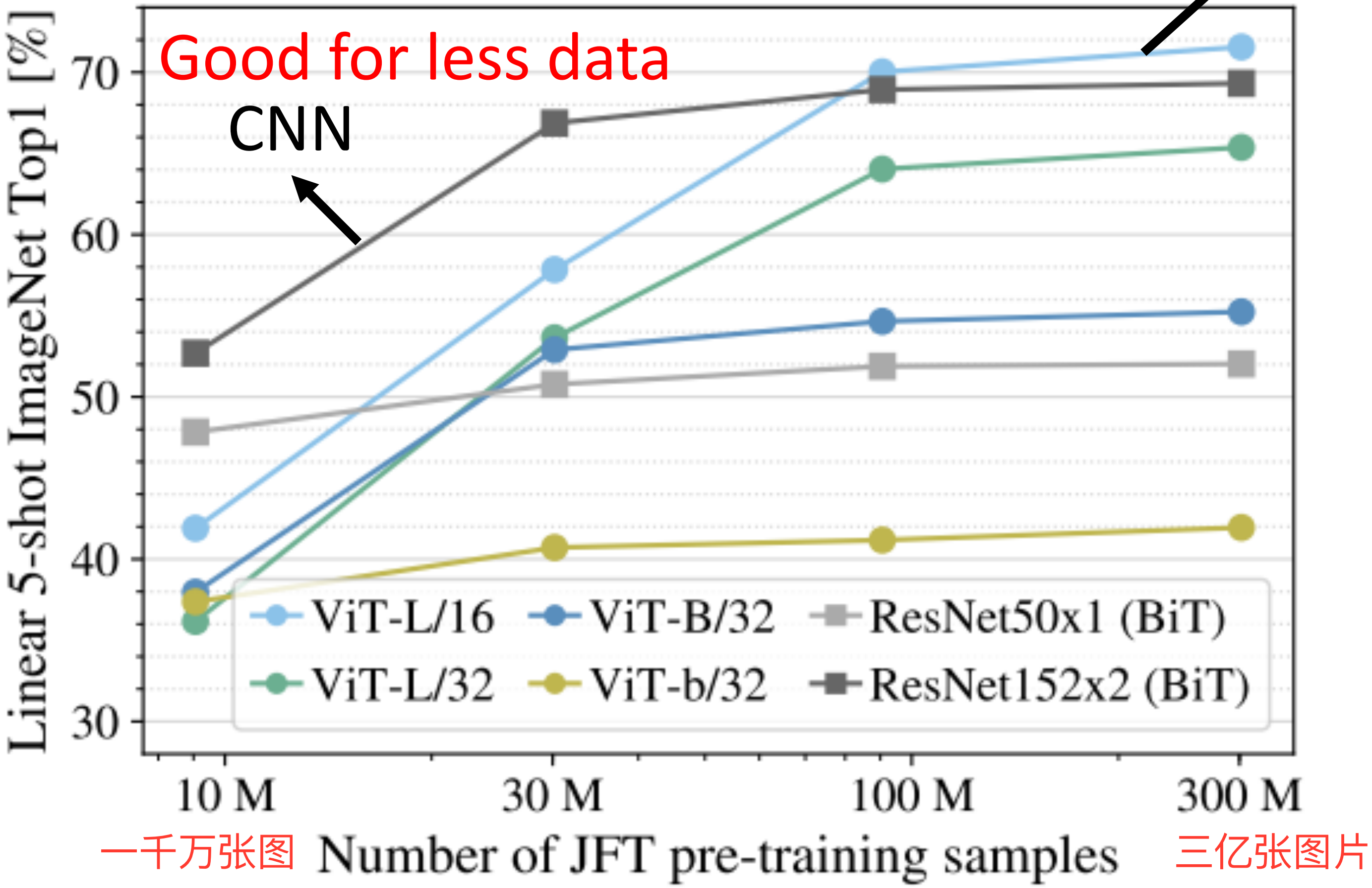
https://arxiv.org/abs/1911.03584

2019年11月的paper!

Self-attention v.s. CNN

Good for more data

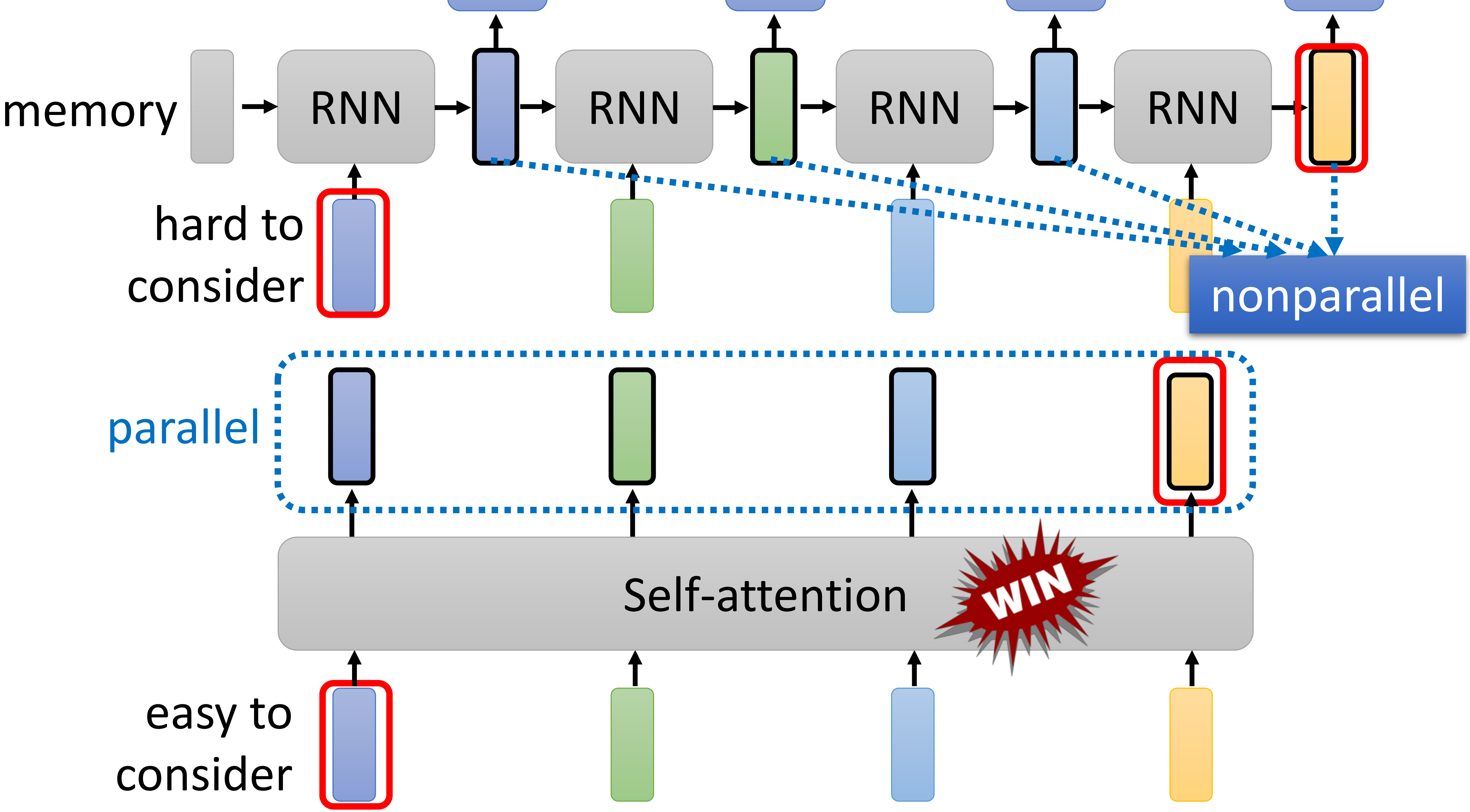
Self-attention



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

https://arxiv.org/pdf/2010.11929.pdf

Self-attention v.s. RNN



Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

https://arxiv.org/abs/2006.16236

To learn more about RNN ……

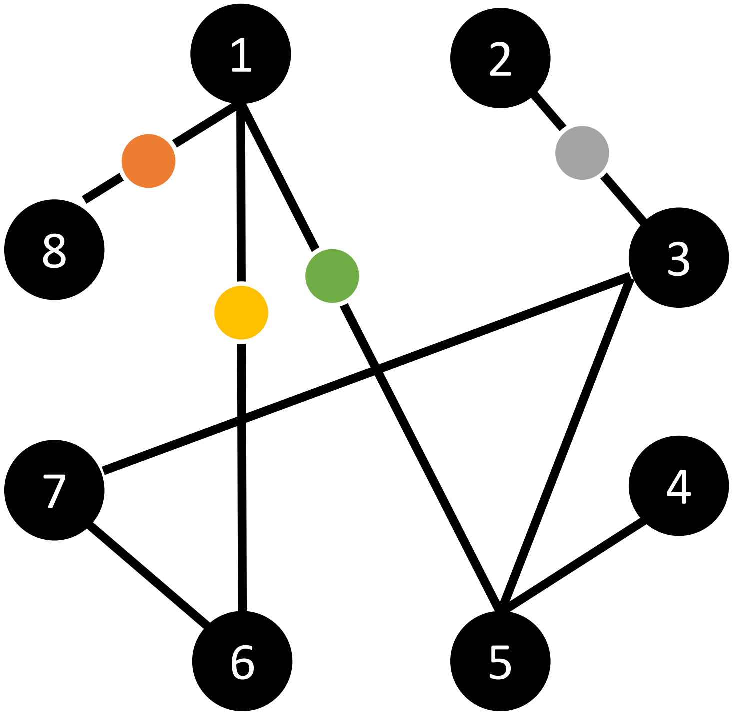
https://youtu.be/xCGidAeyS4M

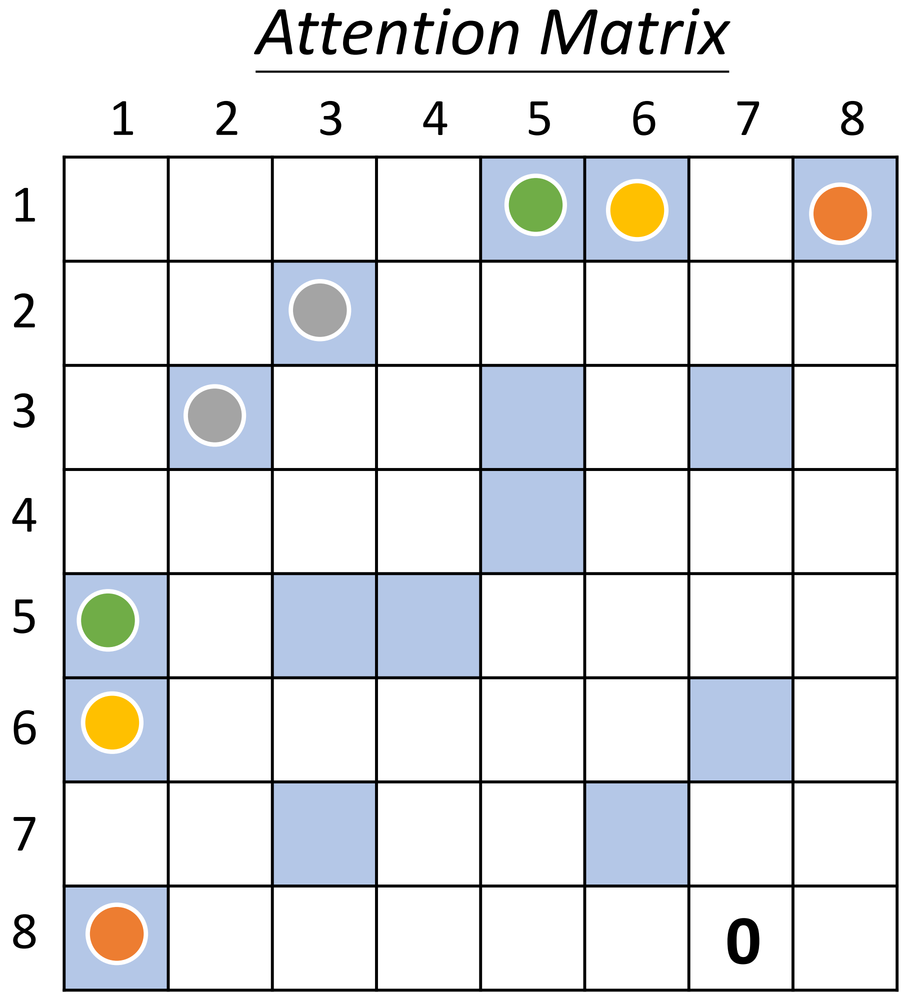
(in Mandarin)

https://youtu.be/Jjy6ER0bHv8

(in English)

Self-attention for Graph





Consider **edge**: only attention to connected nodes

This is one type of **Graph Neural Network (GNN)**.

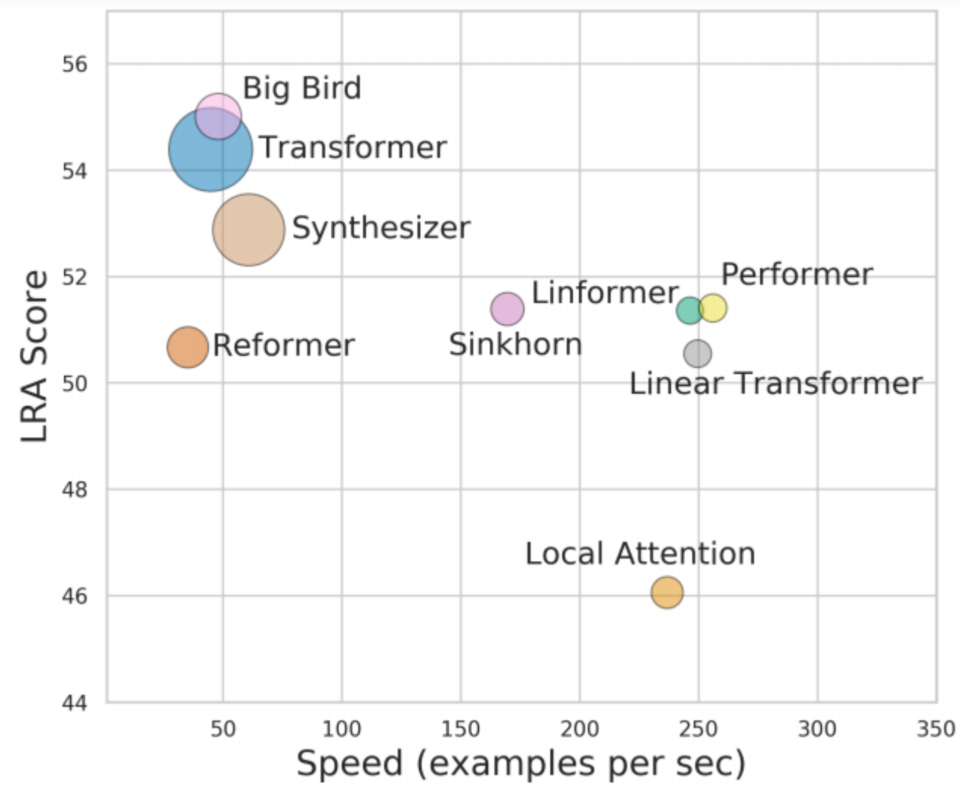
Self-attention for Graph

- To learn more about GNN …

https://youtu.be/eybCCtNKwzA
(in Mandarin)

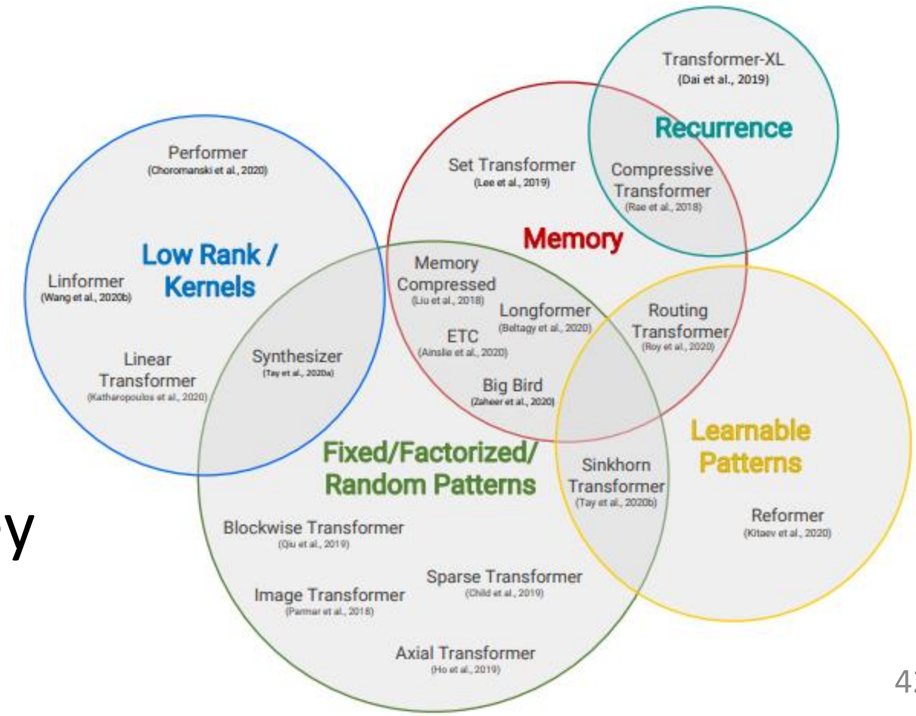
https://youtu.be/M9ht8vsVEw8
(in Mandarin)

To Learn More …



Long Range Arena: A Benchmark for Efficient Transformers

https://arxiv.org/abs/2011.04006



Efficient Transformers: A Survey

https://arxiv.org/abs/2009.06732


Q&A