

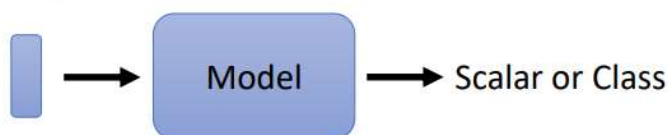
自注意力機制 (Self-attention)

- 自注意力機制 (Self-attention)
 - Self-attention 想解決的問題是什麼？
 - Self-attention 的運作
 - Self-attention 進階版本
 - Multi-head Self-attention
 - 位置資訊 (Positional Encoding 技術)
 - Self-attention v.s. CNN
 - Self-attention v.s. RNN
 - Self-attention (GNN)
- 上課資源：
 1. 【機器學習2021】自注意力機制 (Self-attention)_(上)(<https://www.youtube.com/watch?v=hYdO9CscNes>).
 2. 【機器學習2021】自注意力機制 (Self-attention)_(下)(<https://www.youtube.com/watch?v=gmsMY5kc-zw>).

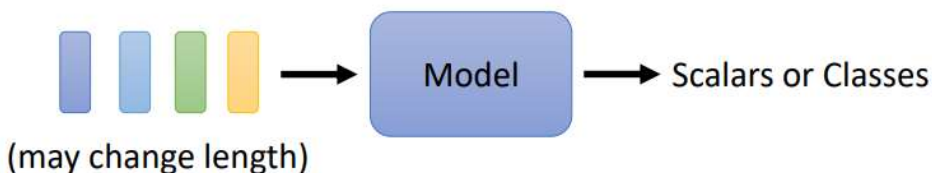
Self-attention 想解決的問題是什麼？

Sophisticated Input

- Input is a **vector**




- Input is a **set of vectors**



- 目前 network 的輸入都是一個 vector，輸出都是一個數值或是類別
- 如果輸入是一排 vector，而且向量的數目是會改變的，要怎麼處理？

Vector Set as Input

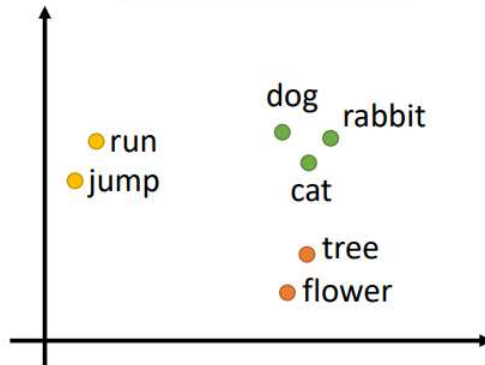
this is a cat



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

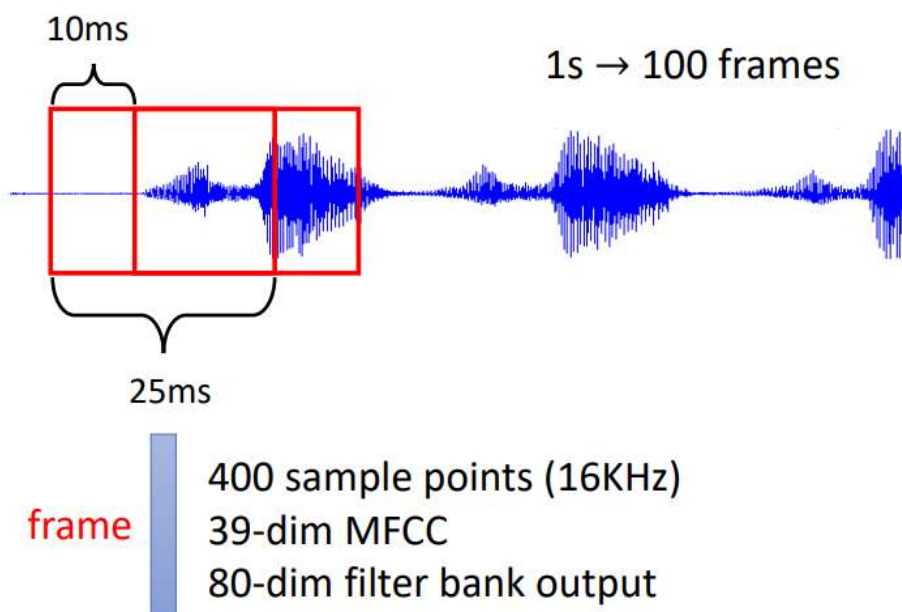


To learn more: <https://youtu.be/X7PH3NuYW0Q> (in Mandarin)

3

- 如何把詞彙表示成向量呢？
 - 用 One-hot Encoding，開一個很長的向量，長度跟世界上存在的詞彙一樣多
 - 用 Word Embedding，給每一個詞彙一個向量，這個向量是有語義的資訊，如果把 word embedding 畫出來，會發現相同類型的詞彙會在同一個區塊
- 延伸閱讀：Unsupervised Learning - Word Embedding (<https://www.youtube.com/watch?v=X7PH3NuYW0Q>).

Vector Set as Input

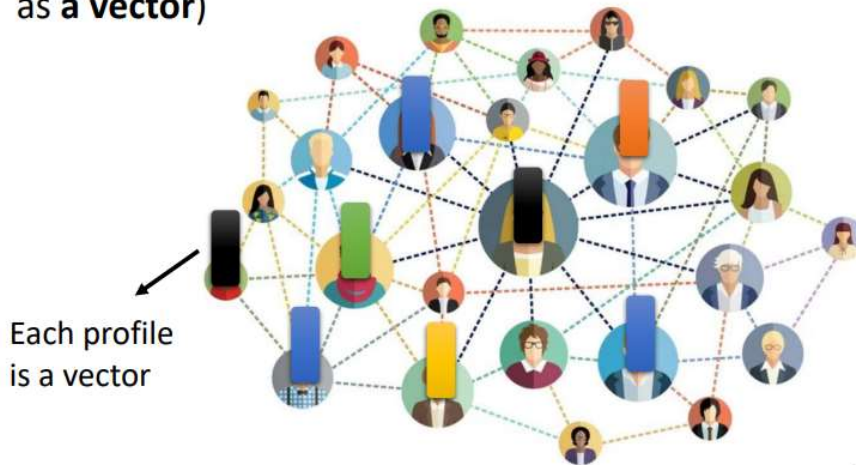


- 一段聲音訊號其實是一排向量，把一段聲音訊號取一個範圍，這個範圍叫做一個 window
- 把 window 裡面的資訊描述成一個向量，這個向量就叫做一個 Frame
- 在語音上會把一個向量叫做一個 Frame
- 通常一個 window 的長度為 25 ms
- 為了要描述整段的聲音訊號，會把 window 往右移 10 ms
- 一段聲音訊號用一段向量來表示，1s 的聲音訊號有 100 個 frames

<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**)



graph 可以看做是一堆向量所組成

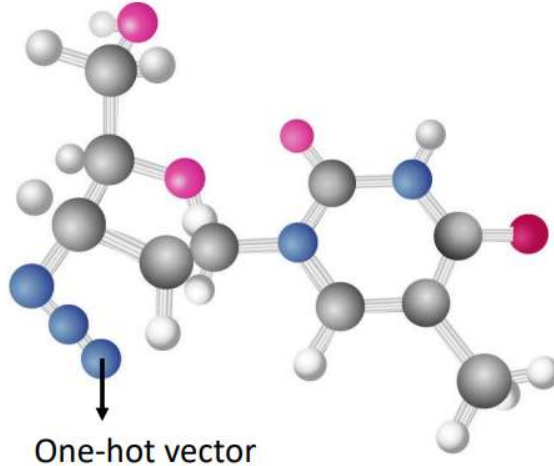
Vector Set as Input

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

$$H = [1 \ 0 \ 0 \ 0 \ 0 \ \dots]$$

$$C = [0 \ 1 \ 0 \ 0 \ 0 \ \dots]$$

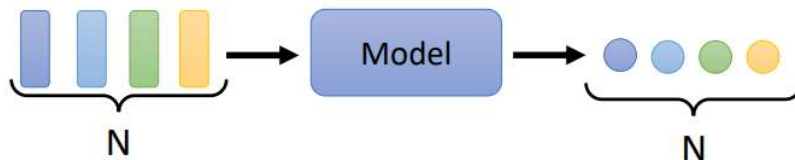
$$O = [0 \ 0 \ 1 \ 0 \ 0 \ \dots]$$

$$\vdots$$


- 把分子看做是一個模型的輸入，每一個分子可以當作是一個 graph，就是一堆向量
- 分子上的每一個球就是原子，就是一個向量
- 原子用一個向量來表示，可以用 One-Hot Vector

What is the output?

- Each vector has a label.

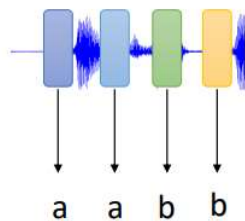


Example Applications

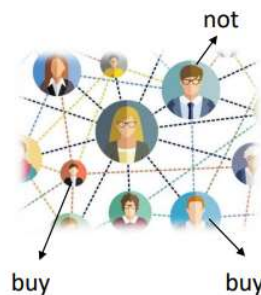
I saw a saw

↓ ↓ ↓ ↓
N V DET N

POS tagging



HW2

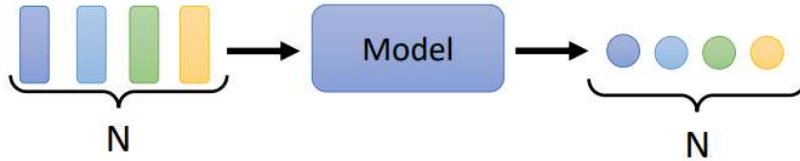


輸出是什麼？

- 每一個向量都有一個 label，輸入跟輸出的長度一樣

What is the output?

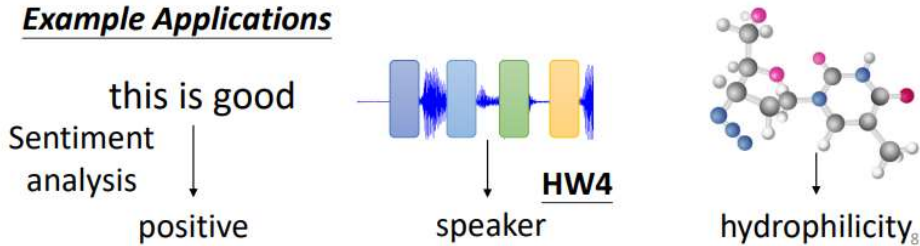
- Each vector has a label.



- The whole sequence has a label.



Example Applications



輸出是什麼？

2. 一整個 sequence，只需要輸出一個 label

What is the output?

- Each vector has a label. focus of this lecture



- The whole sequence has a label.



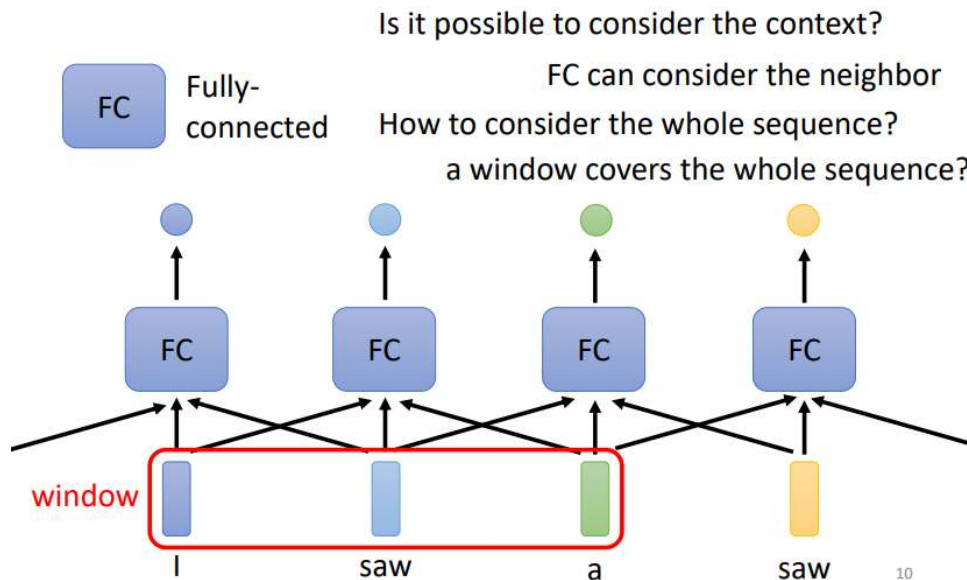
- Model decides the number of labels itself. seq2seq



輸出是什麼？

3. sequence to sequence 不知道應該輸出多少個 label

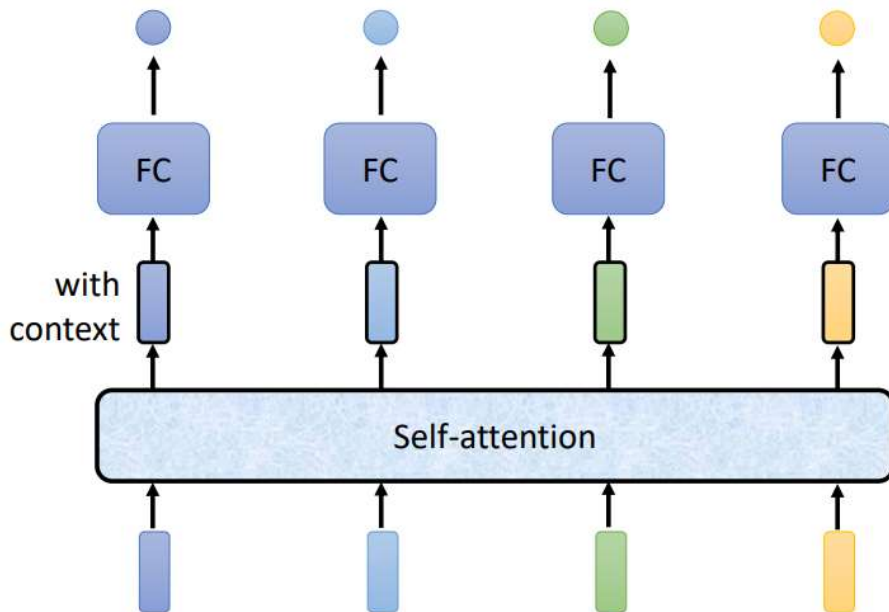
Sequence Labeling



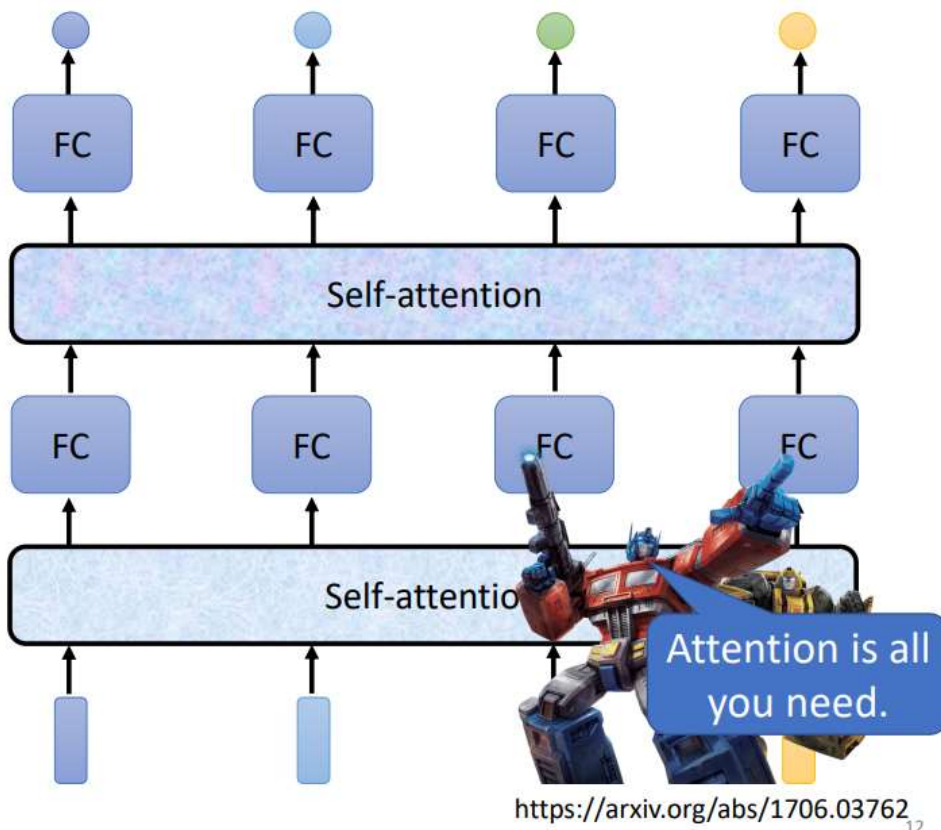
- 這堂課介紹第一個類型 (每一個向量輸出一個 label)
- 把每一個向量分別輸入到 fully-connected network 裡面
- 把前後向量一起丟到 fully-connected network
- 所以我們可以給 fully-connected network 一整個 window 的資訊，讓他可以考慮上下文，但這樣子的方法還是有極限
- 若今天有某個任務，不是考慮一個 window 就可以解決的，而是要考慮一整個 sequence 才能夠解決的話，要怎麼辦？
 - 把 window 開大一點，大到可以把整個 sequence 蓋住，要統計訓練資料裡面最長的 sequence 有多長，然後開一個 window 比最長的 sequence 還要長，才有可能把整個 sequence 蓋住
 - 但是開一個很大的 window，意味著你的 fully-connected 的 network 需要非常多的參數，不只運算量大，可能還容易 overfitting

Self-attention 的運作

Self-attention

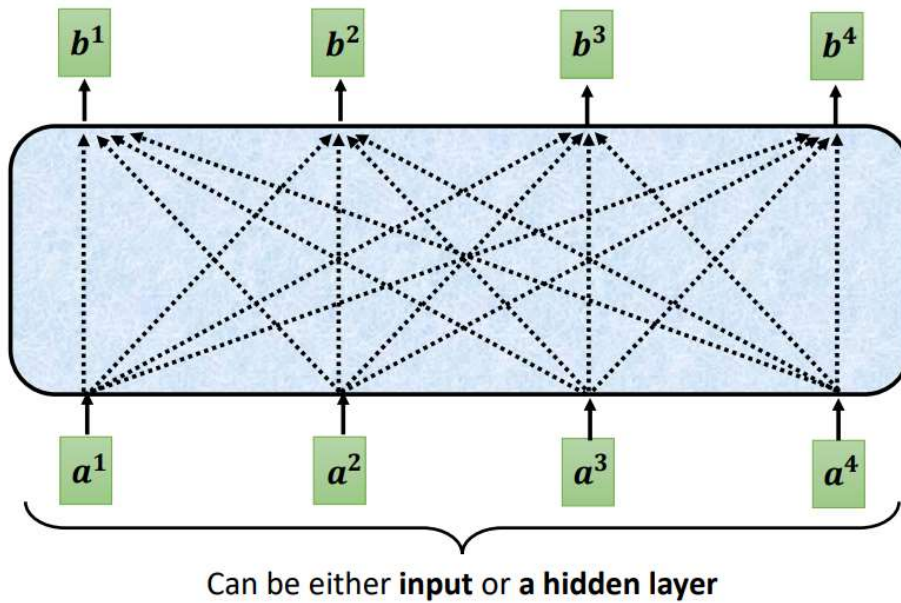


- self-attention 會吃一整個 sequence 的資訊，然後 input 幾個 vector，就會輸出幾個 vector
- 這四個 vector 都是考慮一整個 sequence 以後才得到的



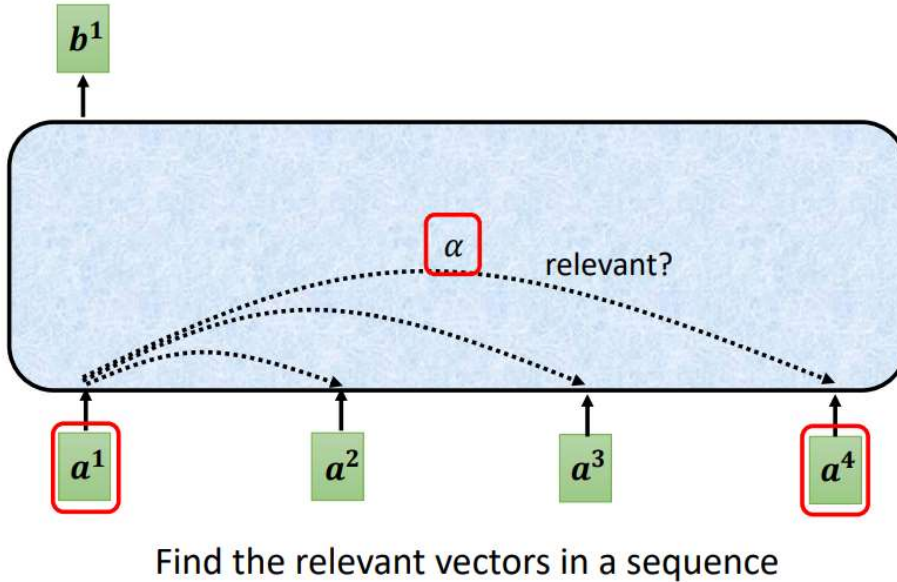
- 可以不只用一次，可以疊加很多次
- 可以把 fully-connected 的 network 跟 self-attention 交替使用

Self-attention



- self attention 的 input 就是一串的 vector
- 這個 vector 可能是整個 network 的 input，也可能是某個 hidden layer 的 output

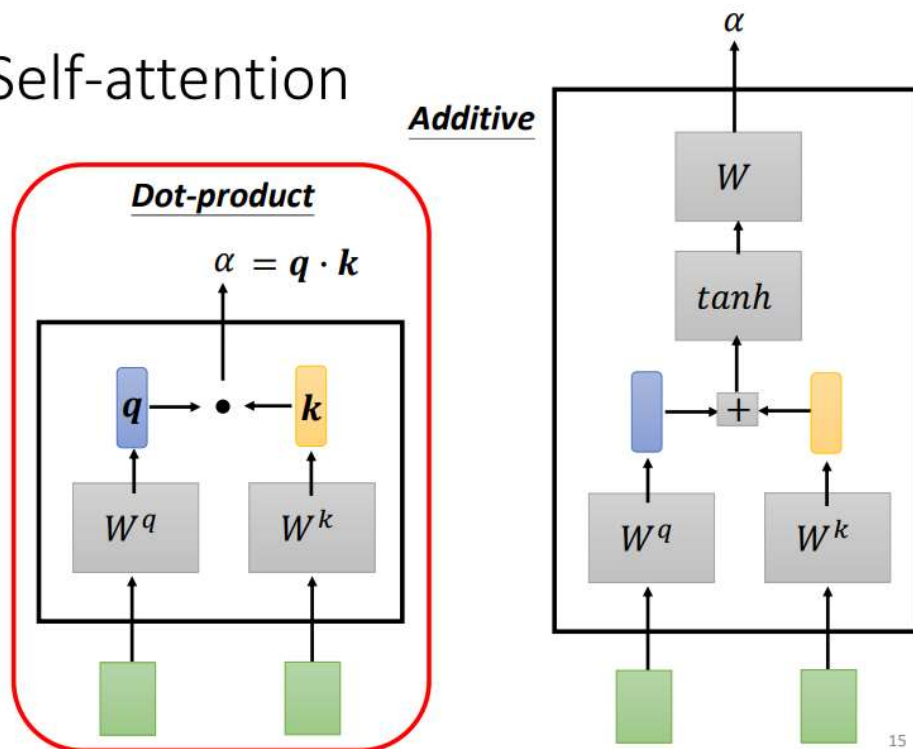
Self-attention



怎麼產生 b^1 這個向量？

- Step 1. 根據 a^1 找出這個 sequence 裡面跟 a^1 相關的其他向量
 - 今天要做 self-attention，目的就是為了要考慮整個 sequence，但是我們不希望把整個 sequence 所有的資訊包在一個 window 裡面，所以有個特別的機制
 - 根據 a^1 找出這個 **sequence** 裡面哪些部分是重要的，哪些部份跟判斷 a^1 是哪個 label、class、regression 數值所需要用到的資訊有關係
 - 每一個向量跟 a^1 的關聯程度用 α 表示
 - self-attention 的 module 怎麼自動決定兩個向量之間的關聯性？

Self-attention

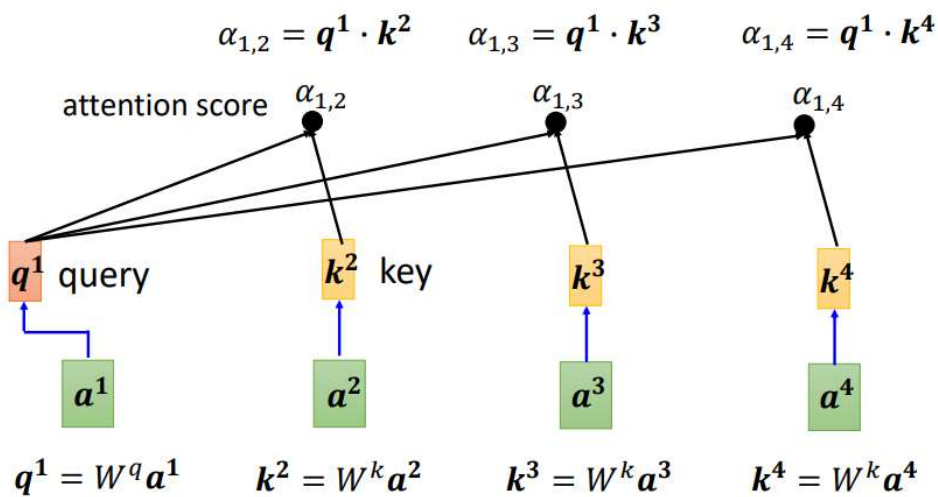


需要一個計算 attention 的 module，拿兩個向量作為輸入，然後直接輸出 α 那個數值，當作兩個向量之間的關聯度

如何計算這個 α 數值呢？

- **Method 1 : Dot-product (常用)**
 - 把輸入的兩個向量分別乘上兩個不同的矩陣，得到 q, k 兩個向量，把 q, k 做 dot product (做 element-wise 相乘再全部加起來得到)後得到 α
- **Method 2 : Additive**
 - 把輸入的兩個向量分別乘上兩個不同的矩陣，得到 q, k 兩個向量，把 q, k 串起來，再通過 \tanh function，最後再通過一個 Transform 後得到 α

Self-attention

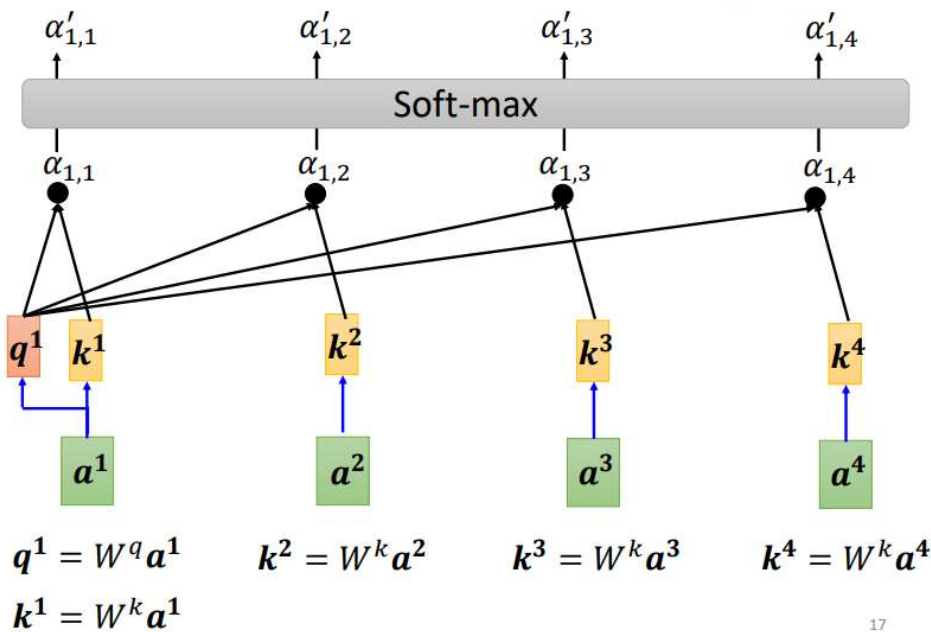


怎麼把 α 套用在 **self-attention** 裡面？

- 把 a^1 跟 a^2 、 a^3 、 a^4 分別去計算他們之間的關聯性

Self-attention

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



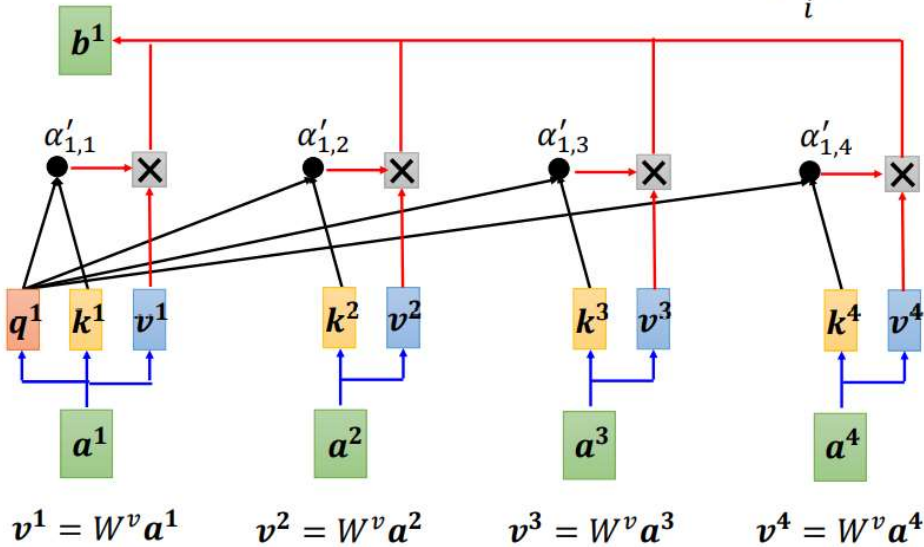
17

- q^1 也會跟自己算關聯性，所以也會把 a^1 乘上 W^k 得到 k^1
- 計算完跟每個向量的關聯性後，會做 soft-max (最常見)

Self-attention

Extract information based on attention scores

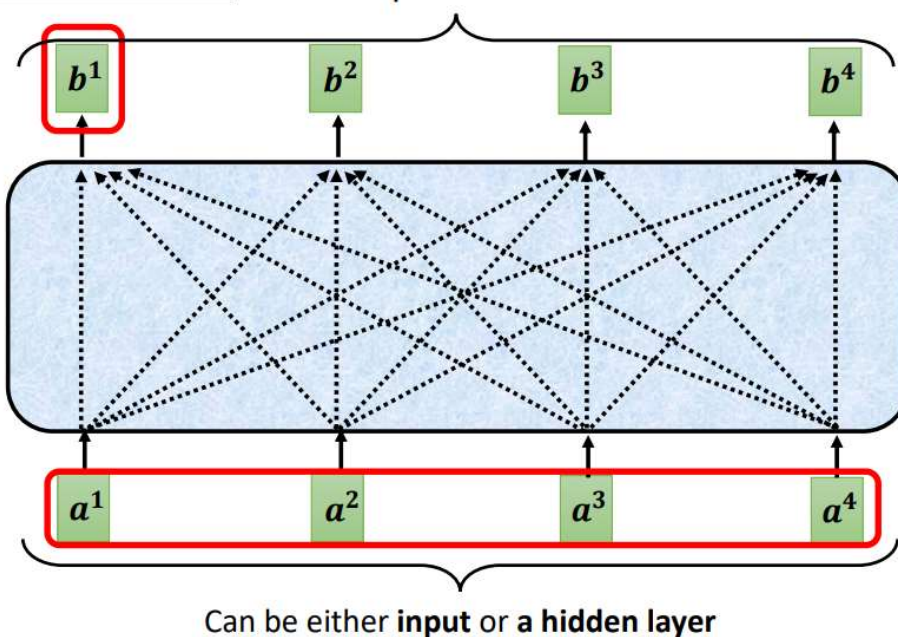
$$b^1 = \sum_i \alpha'_{1,i} v^i$$



- 根據 $\alpha'_{1,1}$ 去抽取出這個 sequence 裡面重要的資訊
- 根據 α 知道哪些向量跟 a^1 最有關係
- 把 a^1, a^2, a^3, a^4 個別乘上 W^v 得到新的向量，用 v^1, v^2, v^3, v^4 表示，接著將 v^1, v^2, v^3, v^4 都乘上 α' ，最後再加起來得到 b^1
- 如果某個向量得到的分數越高，假如 a^1 跟 a^2 的關聯性很強， a^1 得到的值很大，那在做 weighted sum 得到的 b^1 可能會比較接近 v^2

Self-attention

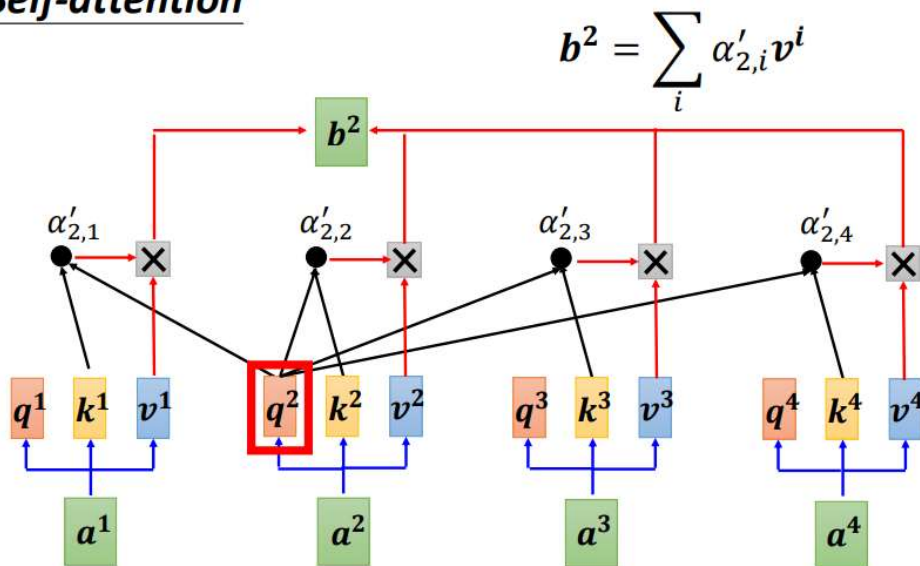
parallel



Can be either input or a hidden layer

- 說明如何從整個 sequence 得到 b^1

Self-attention



- b^2 跟 b^1 計算方式相同

Self-attention

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

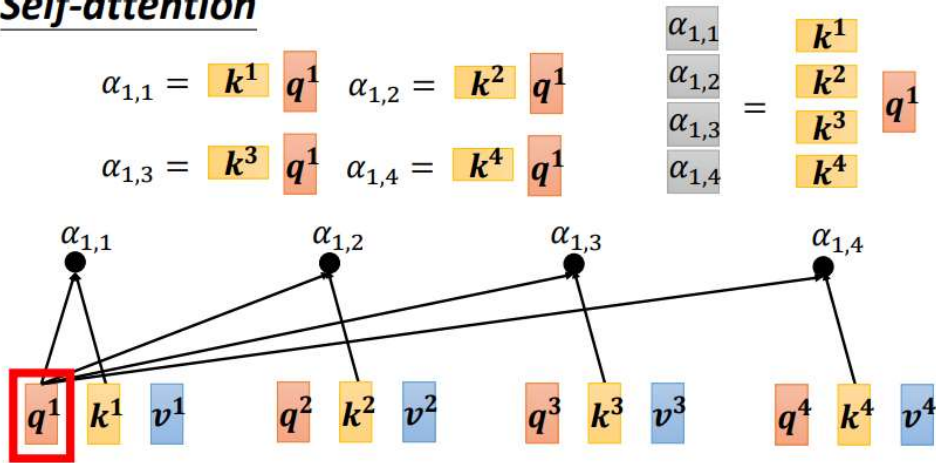
$$k^i = W^k a^i \quad \begin{matrix} k^1 & k^2 & k^3 & k^4 \\ K \end{matrix} = \begin{matrix} W^k & a^1 & a^2 & a^3 & a^4 \\ I \end{matrix}$$

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

The diagram shows the transformation of input vectors a^1, a^2, a^3, a^4 into query (q), key (k), and value (v) vectors using weight matrices $W^q, W^k,$ and W^v . The resulting vectors are grouped into matrices $Q, K,$ and V .

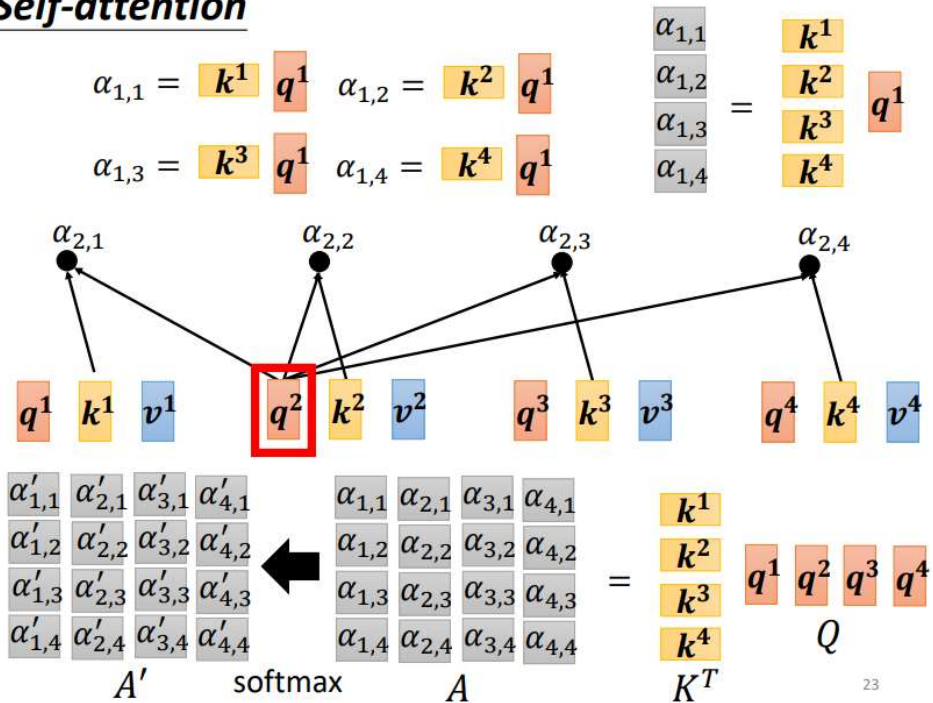
從 a 得到 q, k, v

Self-attention



從 q 得到 α (attention score)

Self-attention

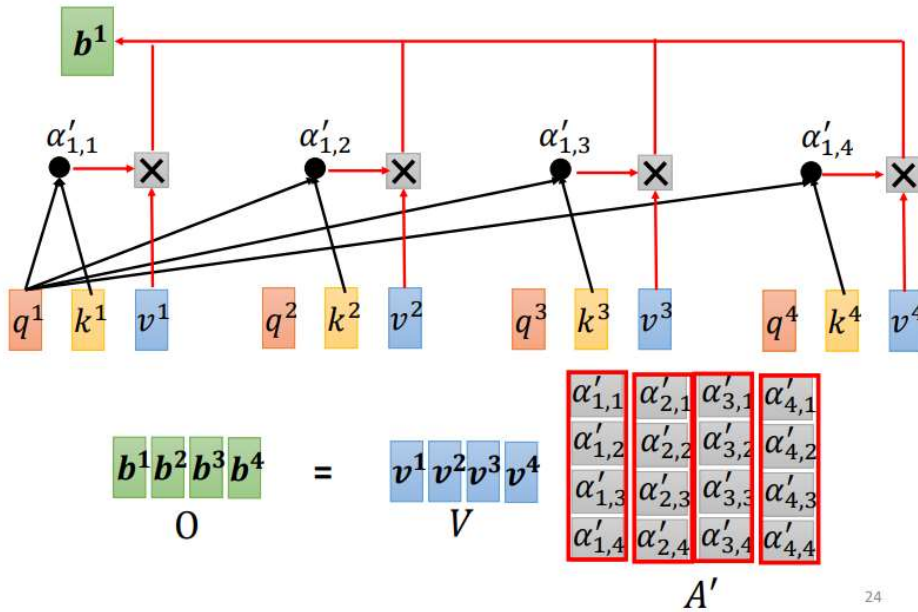


從每個 q 得到每個 α (attention score)

操作跟上一個投影片相同

最後再將 A 做 soft max 得到 A'

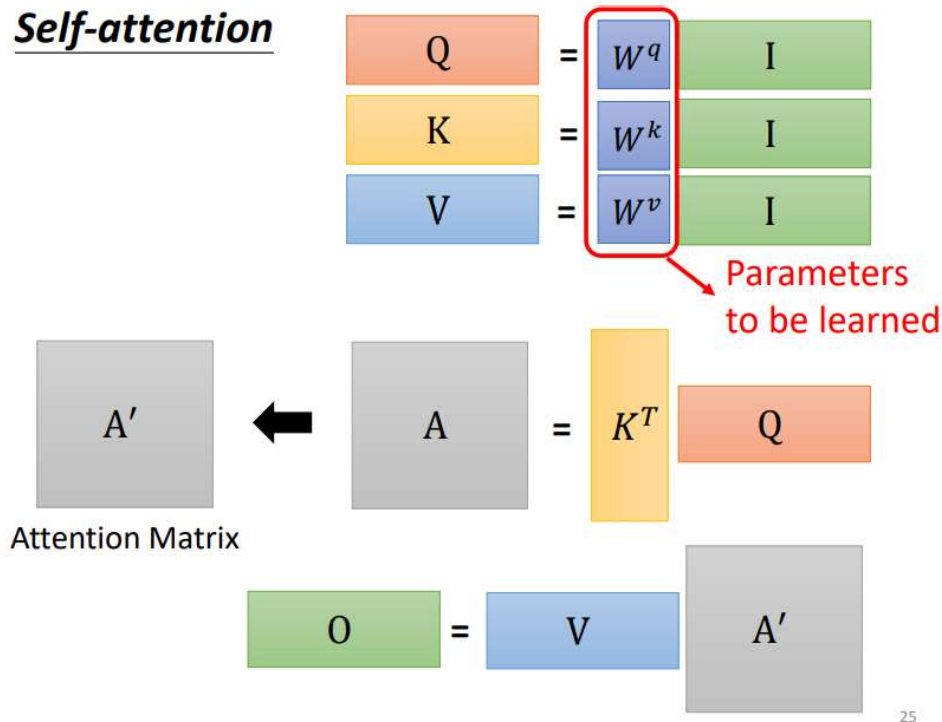
Self-attention



24

計算出 b^1, b^2, b^3, b^4

Self-attention



25

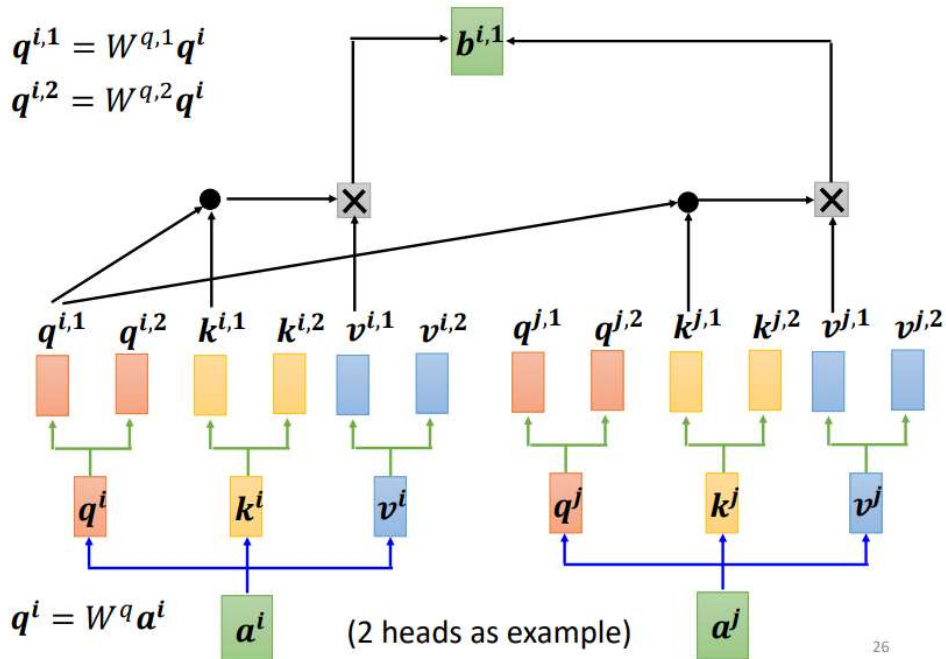
整合整個計算

整個過程只有 W^q, W^k, W^v 參數是未知的，需要用訓練資料找出

Self-attention 進階版本

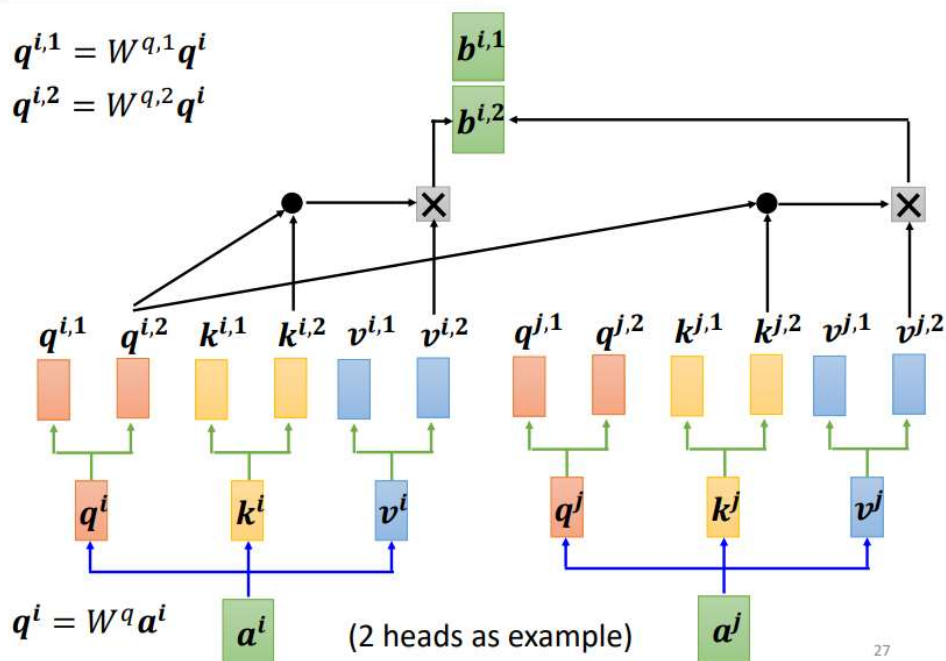
Multi-head Self-attention

Multi-head Self-attention Different types of relevance



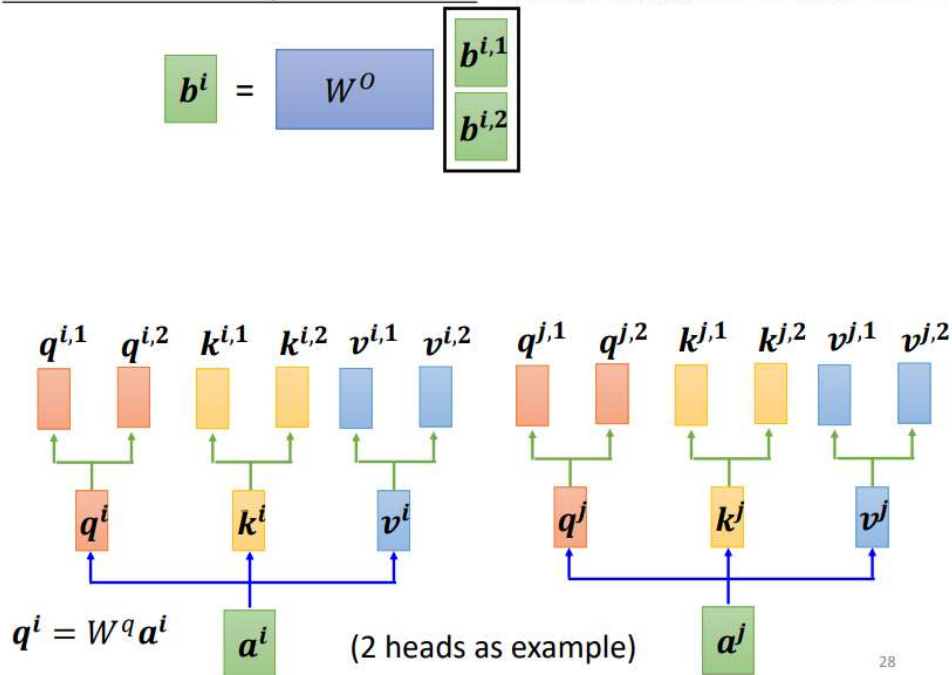
把 q^i, k^i, v^i 分別乘上兩個矩陣得到不同的 head
 方式跟之前一樣，得到 $b^{i,1}$

Multi-head Self-attention Different types of relevance



計算 $b^{i,2}$

Multi-head Self-attention Different types of relevance



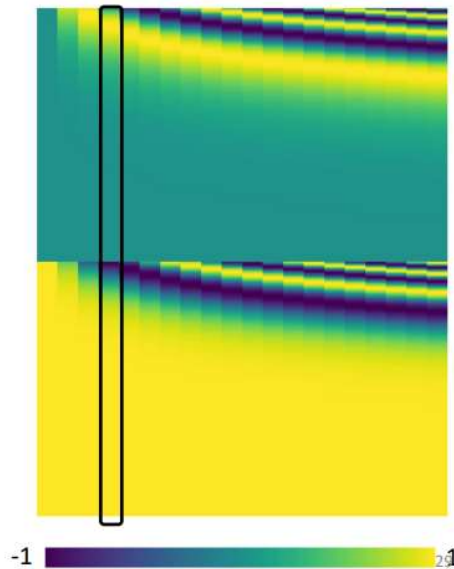
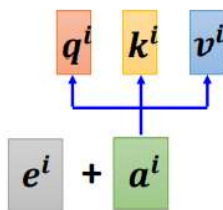
把 $b^{i,1}$ 跟 $b^{i,2}$ 接起來，再乘上一個矩陣 W^O ，得到 b^i

位置資訊 (Positional Encoding 技術)

Positional Encoding

Each column represents a positional vector e^i

- No position information in self-attention.
- Each position has a unique positional vector e^i
- **hand-crafted**
- **learned from data**

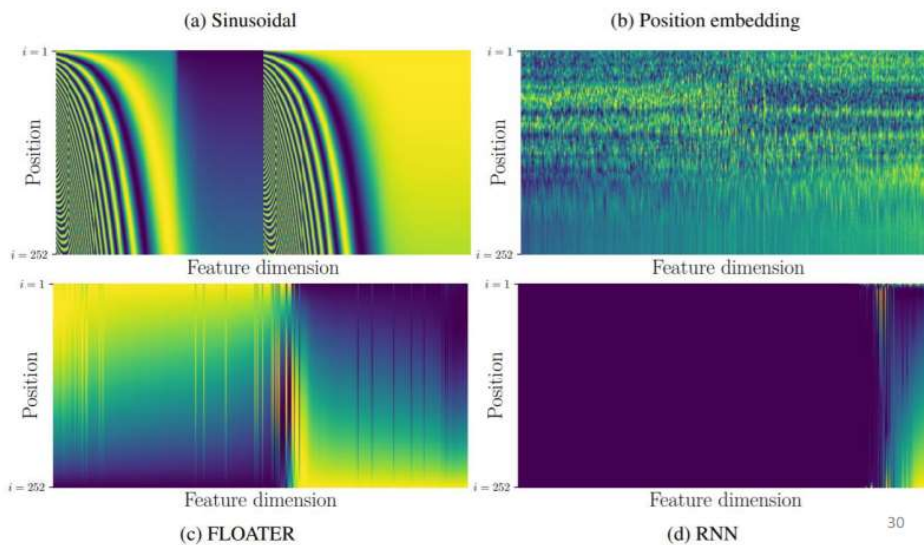


- self-attention 少了位置的資訊
- 需要用到 Positional Encoding 技術
 - 為每一個位置設定一個 vector 叫做 positional vector e^i
 - 把 e^i 加到 a^i 上
 - 是人工設定

<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	✓	✓	✓



各式各樣的方法產生 positional encoding

Many applications ...



Transformer

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



BERT

<https://arxiv.org/abs/1810.04805>

Widely used in Natural Language Processing (NLP)! 31

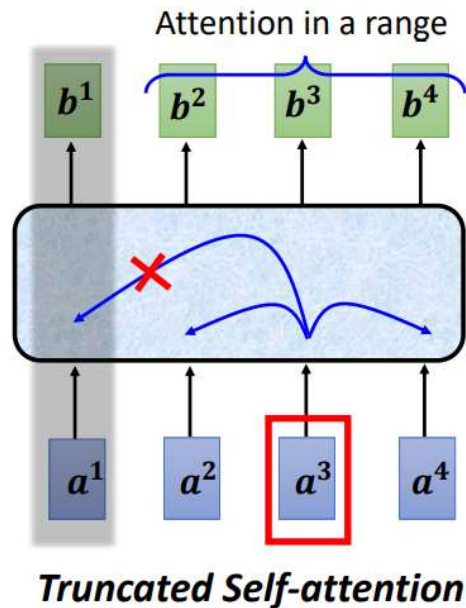
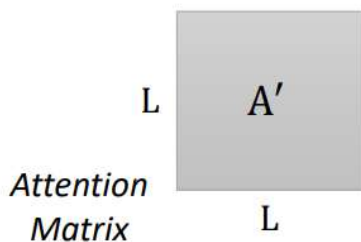
BERT、NLP 也有用到 self-attention

Self-attention for Speech

Speech is a very long vector sequence.



If input sequence is length L



32

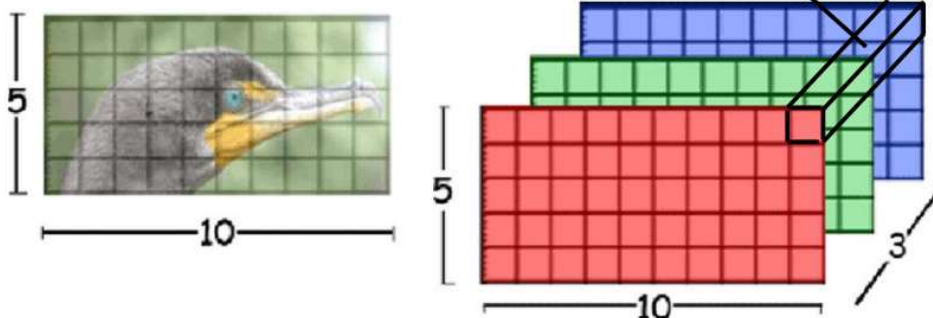
- 做語音的時候也可以用 self-attention
- 但是很容易向量的資訊量太大，不易處理
- 所以使用 Truncated self-attention

Truncated self-attention

- 不要看一整句話，只看一個小的範圍 (範圍由人設定)

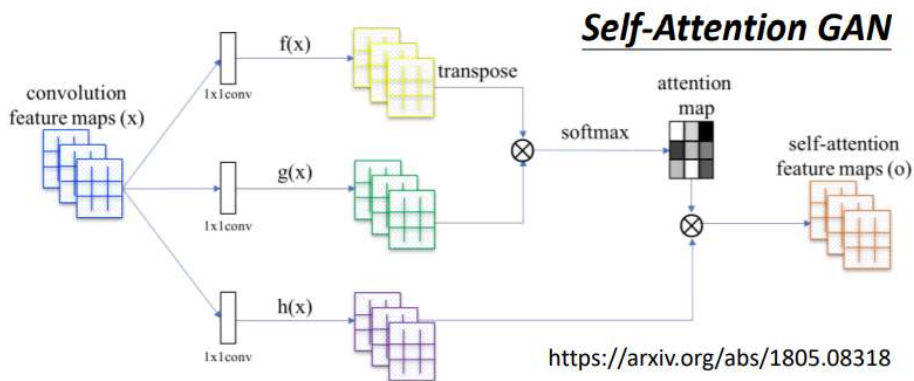
Self-attention for Image

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

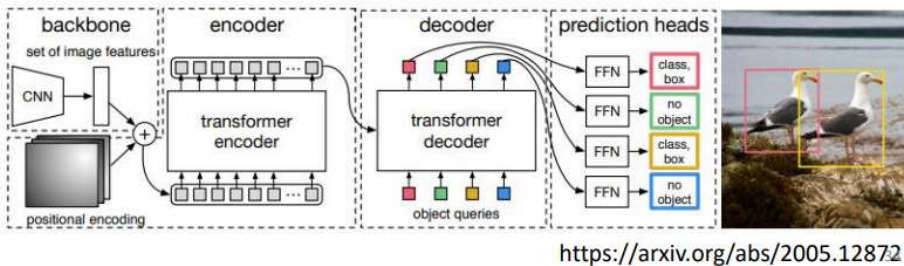


Source of image: https://www.researchgate.net/figure/Color-image-representation-and-RGB-matrix_fig15_282798184

- self-attention 可以被用在影像上
- 這張 5×10 的圖片，可以看做是一個 tensor 大小是 $5 \times 10 \times 3$
- 把每一個 pixel 看做是一個 3 維的向量



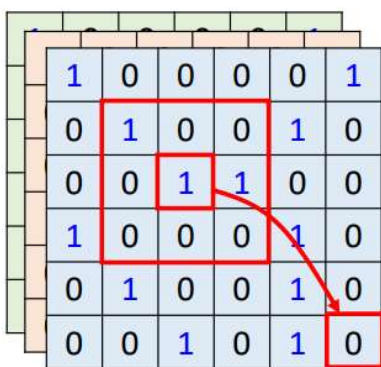
DEtection Transformer (DETR)



self-attention 用在圖片上的例子

Self-attention v.s. CNN

Self-attention v.s. CNN



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

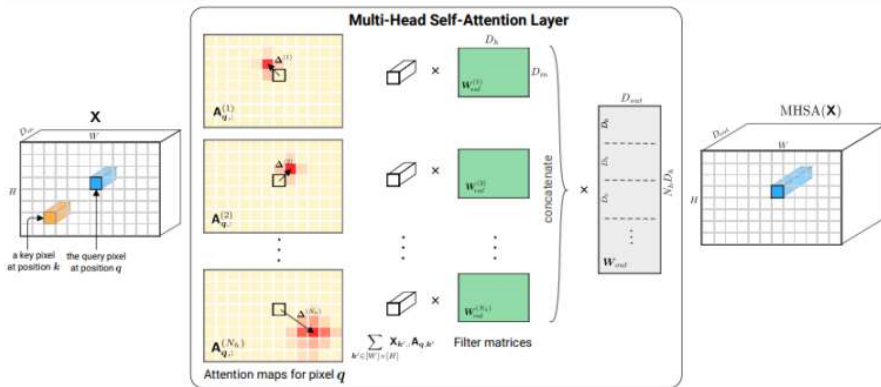
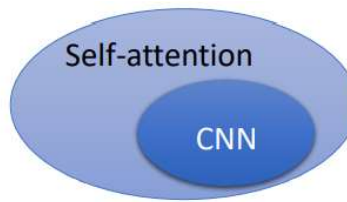
➤ CNN is simplified self-attention.

Self-attention: CNN with learnable receptive field

➤ Self-attention is the complex version of CNN.

- CNN 可以看做是一個簡化版的 self-attention，只考慮 receptive field 裡面的資訊
- self-attention 是考慮整張圖片的資訊

Self-attention v.s. CNN



On the Relationship between Self-Attention and Convolutional Layers

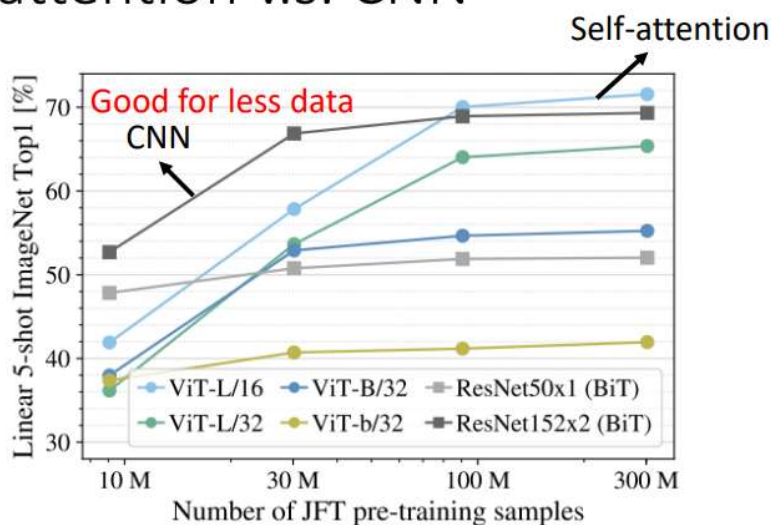
<https://arxiv.org/abs/1911.03584>

36

- 論文說明 CNN 就是 self-attention 的特例
- self-attention 只要設定合適的參數，可以做到跟 CNN 一模一樣的事情
- self-attention 只要通過某些限制，就可以變成 CNN
- 比較 flexible 的 model，比較需要更多的 data，如果 data 不夠就很有可能會 overfitting
- 比較有限制的 model，適合在 data 少的，少的時候可能比較不會 overfitting

Self-attention v.s. CNN

Good for more data



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

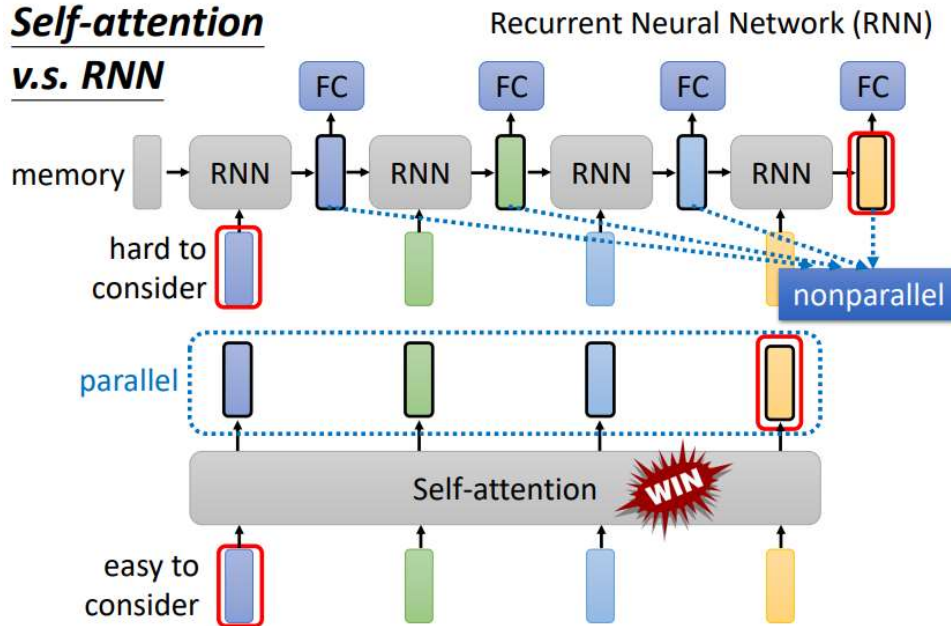
<https://arxiv.org/pdf/2010.11929.pdf>

用不同的 data 量去訓練 CNN 跟 self-attention

Self-attention v.s. RNN

Self-attention

v.s. RNN

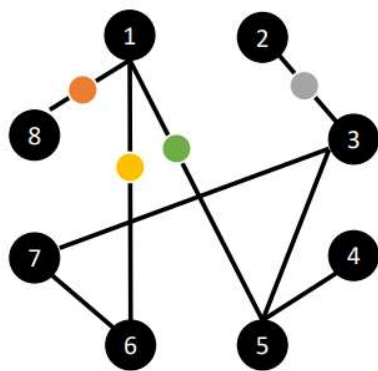


Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention
<https://arxiv.org/abs/2006.16236>

RNN 很大部分可以用 self-attention 取代

Self-attention (GNN)

Self-attention for Graph



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

Attention Matrix

	1	2	3	4	5	6	7	8
1					●	●		●
2			●					
3		●						
4								
5	●							
6	●							
7								
8	●						0	

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

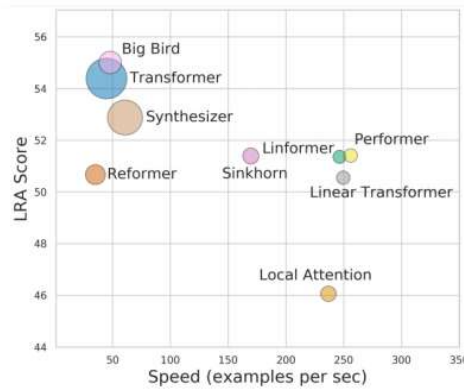
40

擁有很多限制的 self-attention，這個方式就是一種 GNN

To Learn More ...

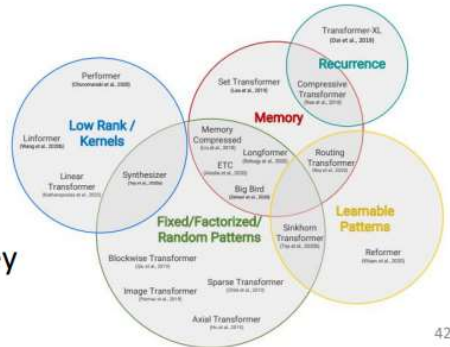
Long Range Arena: A Benchmark for Efficient Transformers

<https://arxiv.org/abs/2011.04006>



Efficient Transformers: A Survey

<https://arxiv.org/abs/2009.06732>



42

self-attention 各式各樣的變形

- 延伸閱讀：
 - Recurrent Neural Network (Part I) (<https://www.youtube.com/watch?v=xCGidAeyS4M>).
 - Graph Neural Network (1/2) (<https://www.youtube.com/watch?v=eybCCtNKwzA>).

課程網頁 (<https://speech.ee.ntu.edu.tw/~hylee/ml/2022-spring.php>).

tags: 2022 李宏毅_機器學習