模型種類

我們深度學習根據 inputs 和 outputs 種類大概可以分成四種

inputs

單輸入

一張圖

一個詞

多輸入

一篇文章(多詞)

一部影片(多圖)

outputs

單輸出

一個判斷

一個詞

多輸出

一篇文章(多詞)

一部影片(多圖)

什麼時候該加入順序到模型裡

影片，文章，音樂。。。這種有序資料你都必須考慮要不要加入順序類型的模型

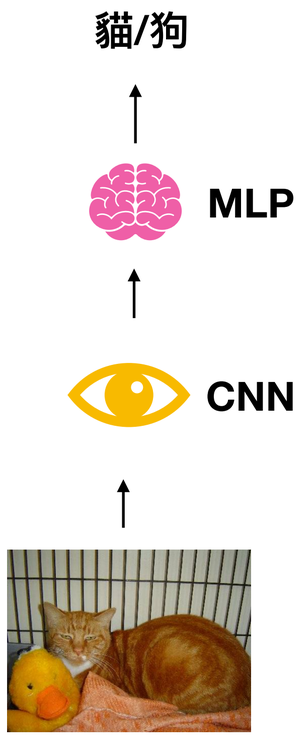
一對一

一對一

1. 輸入：單張圖片，輸出：單個預測，所以叫做一對一

順序的重要性： 無

1. 輸入和輸出都沒有順序，所以根本不需要順序型的模型



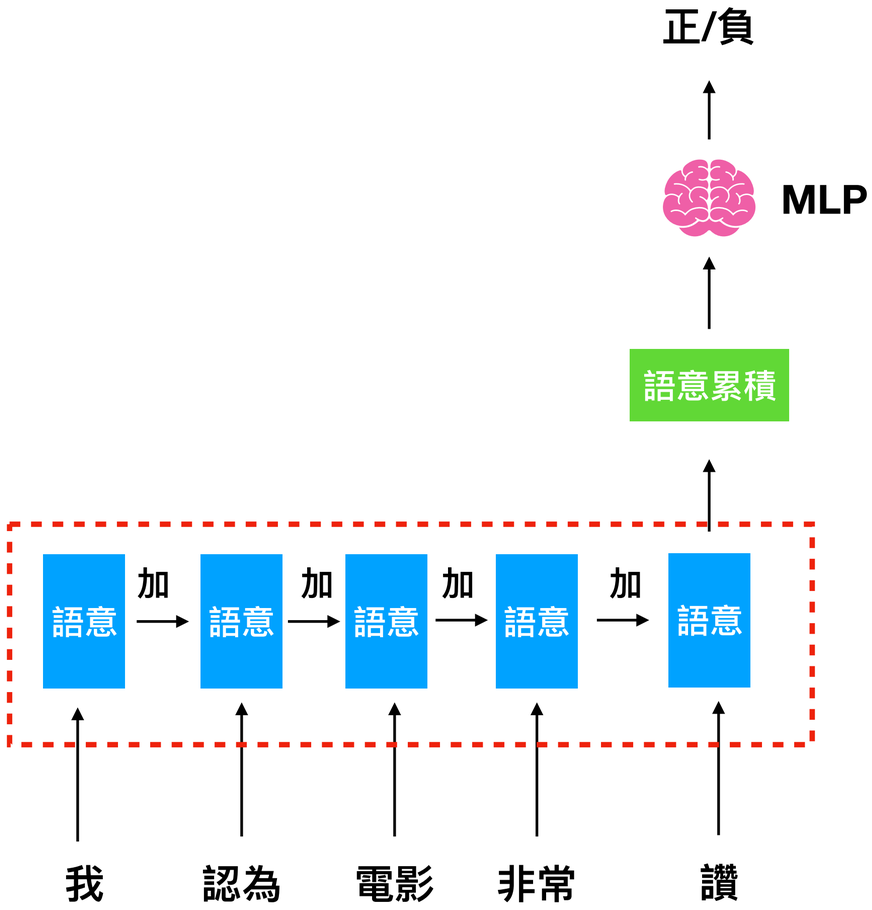
多對一

多對一(many-to-one)

1. 輸入：多個詞語，輸出：單個預測，所以叫做多對一
2. 例子: 文章 -> 情緒，影片 -> 動作

順序的重要性：★

1. 如果是文章情緒判別，那順序還好，畢竟判別情緒主要是靠出現了什麼詞語，對於順序沒這麼重要
2. 但如果是影片判別動作，那順序就蠻重要的了！



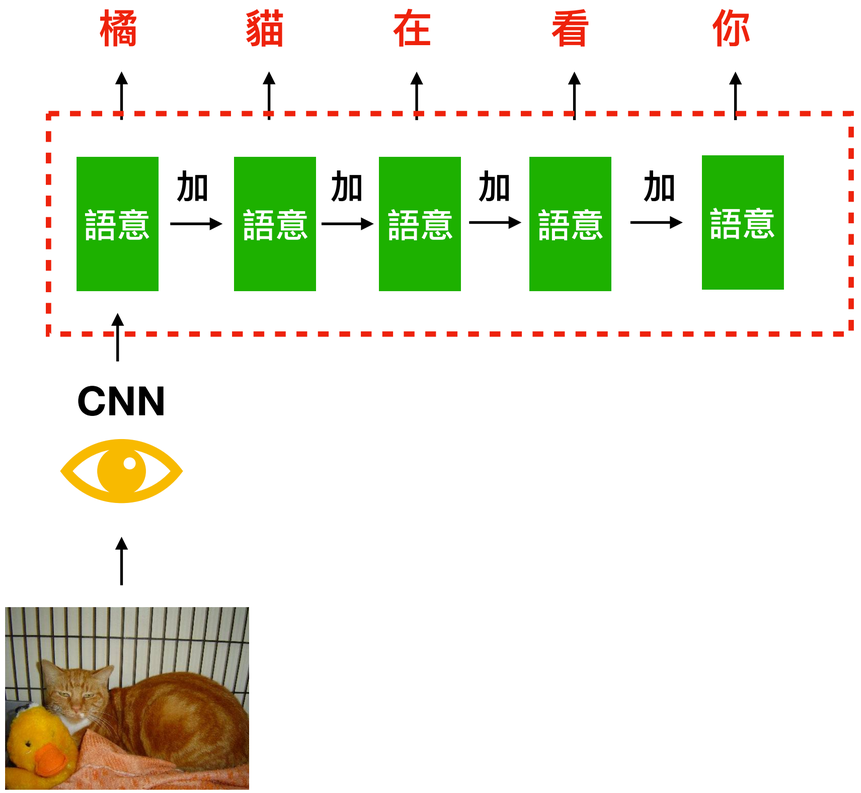
一對多

一對多(one-to-many)

1. 輸入：單張圖片，輸出：多個詞語
2. 上面這個例子我們叫做 Image Captioning (圖片標注)，簡單來說就是說明圖片

順序的重要性：★★★

1. 你輸出的東西在大部分情況下都跟時序非常非常的有關係，所以你必須在輸出的地方有個順序類型的模型！



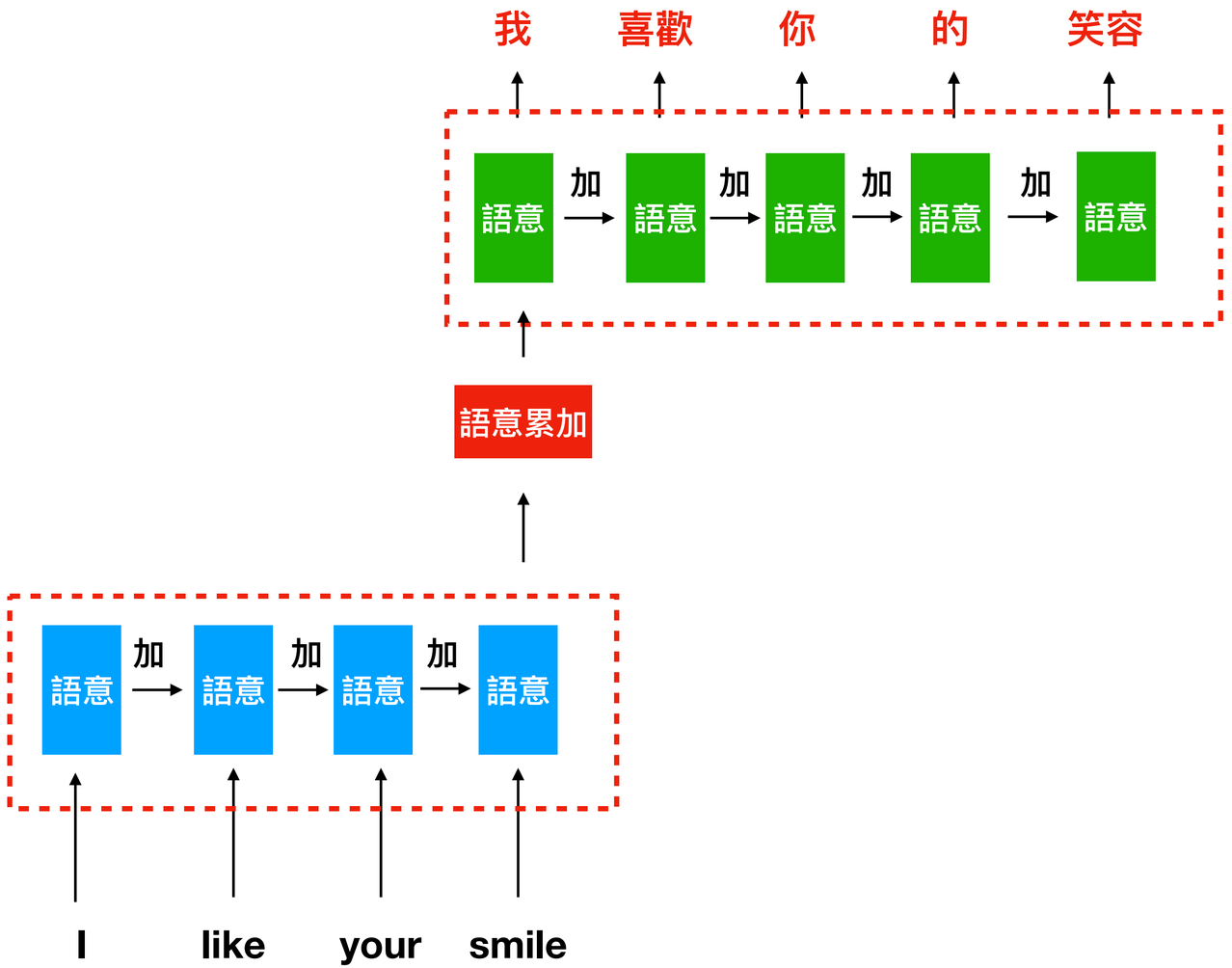
多對多

多對多(many-to-many)

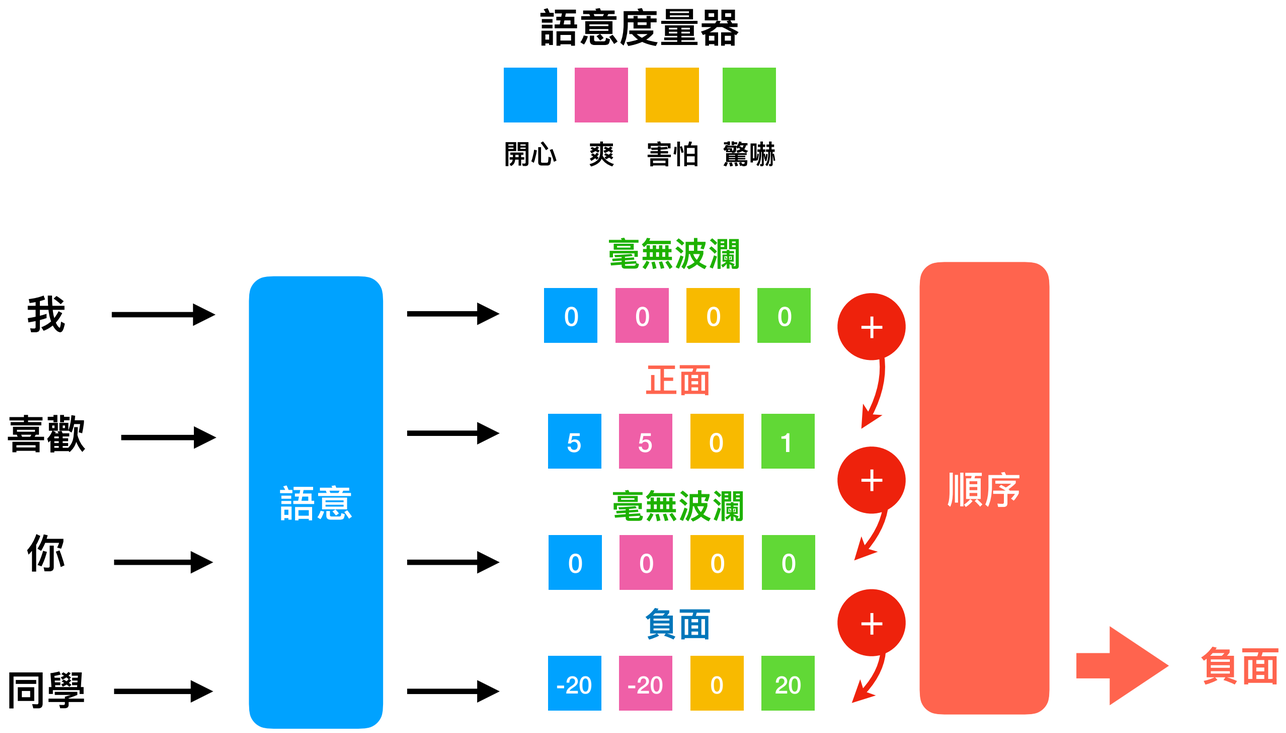
1. 輸入：多個詞語，輸出：多個詞語
2. 機器翻譯就是這種模型極好的例子

順序的重要性：★★★★★

1. 通常輸入端和輸出端都是跟順序有關的輸入，所以輸入和輸出端都必須加上順序類型的模型

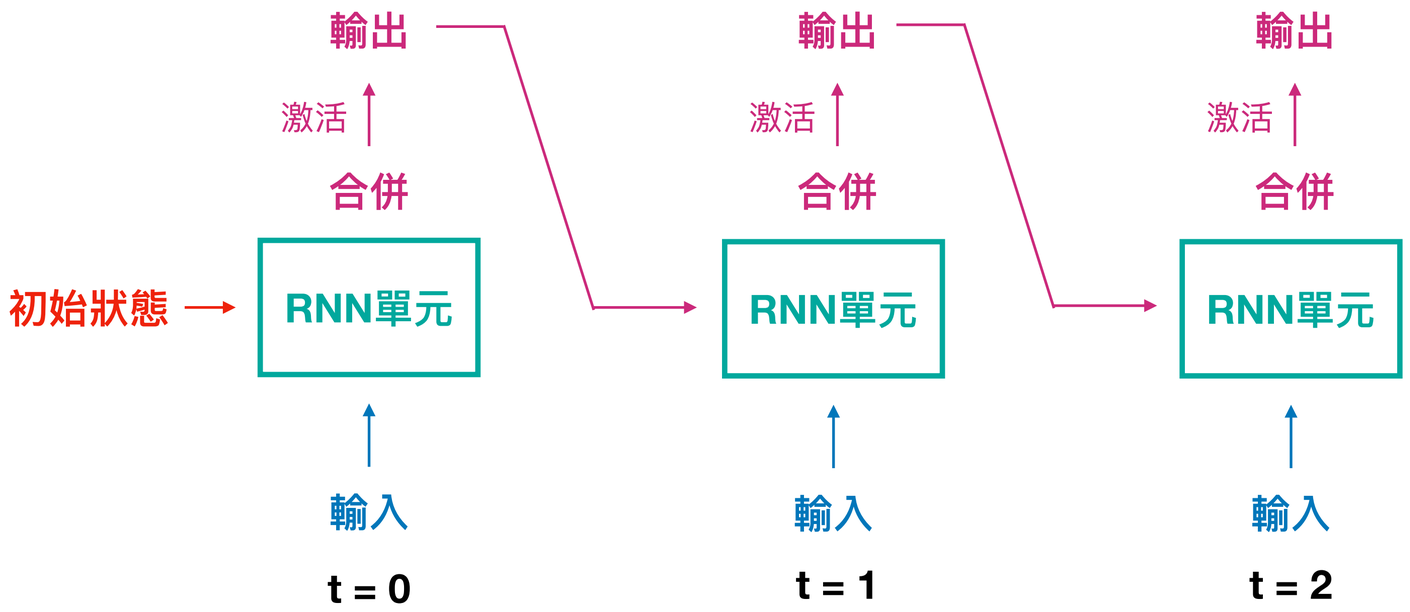


順序

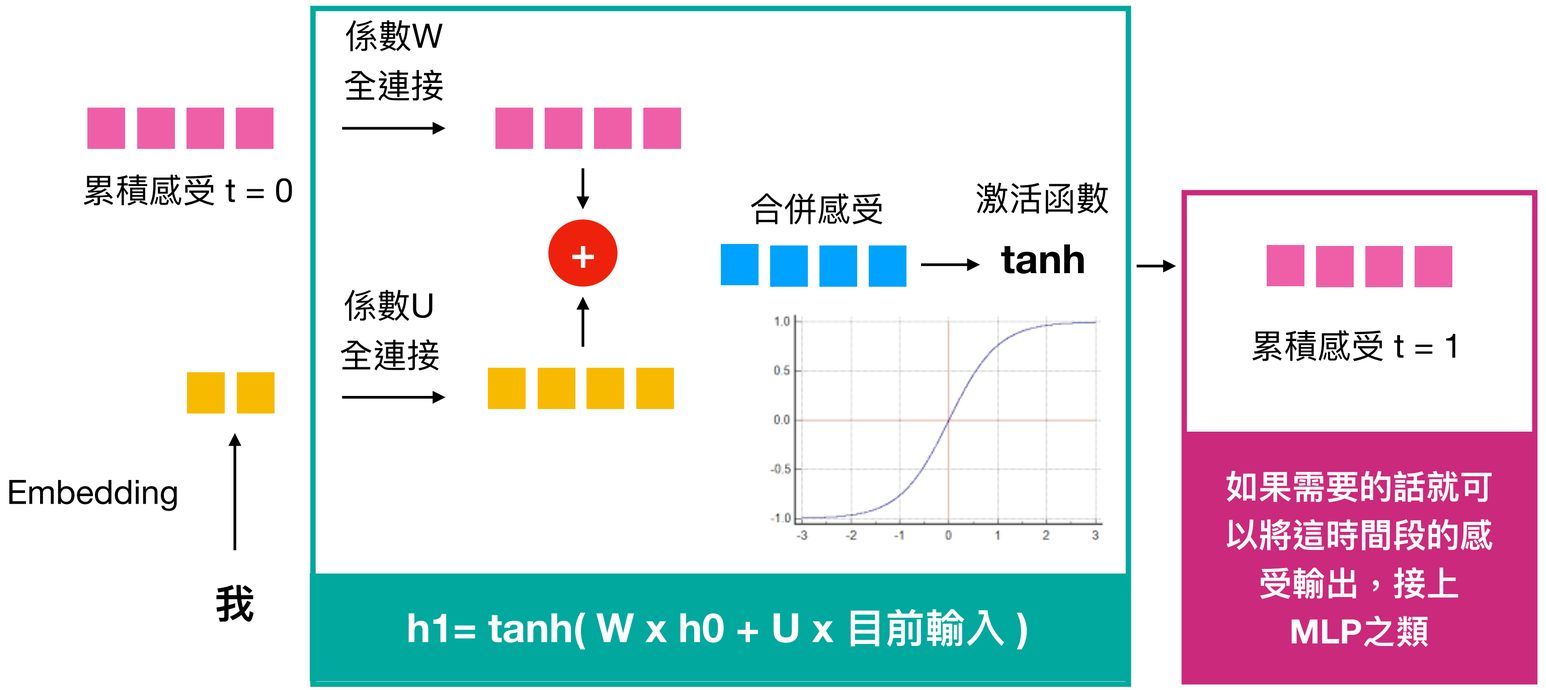

語意度量器
開心
爽
害怕
驚嚇
我
喜歡
你
同學
語意
毫無波瀾
0 0 0 0
正面
5 5 0 1
毫無波瀾
0 0 0 0
負面
-20 -20 0 20
+
+
+
順序
負面

RNN基礎概念

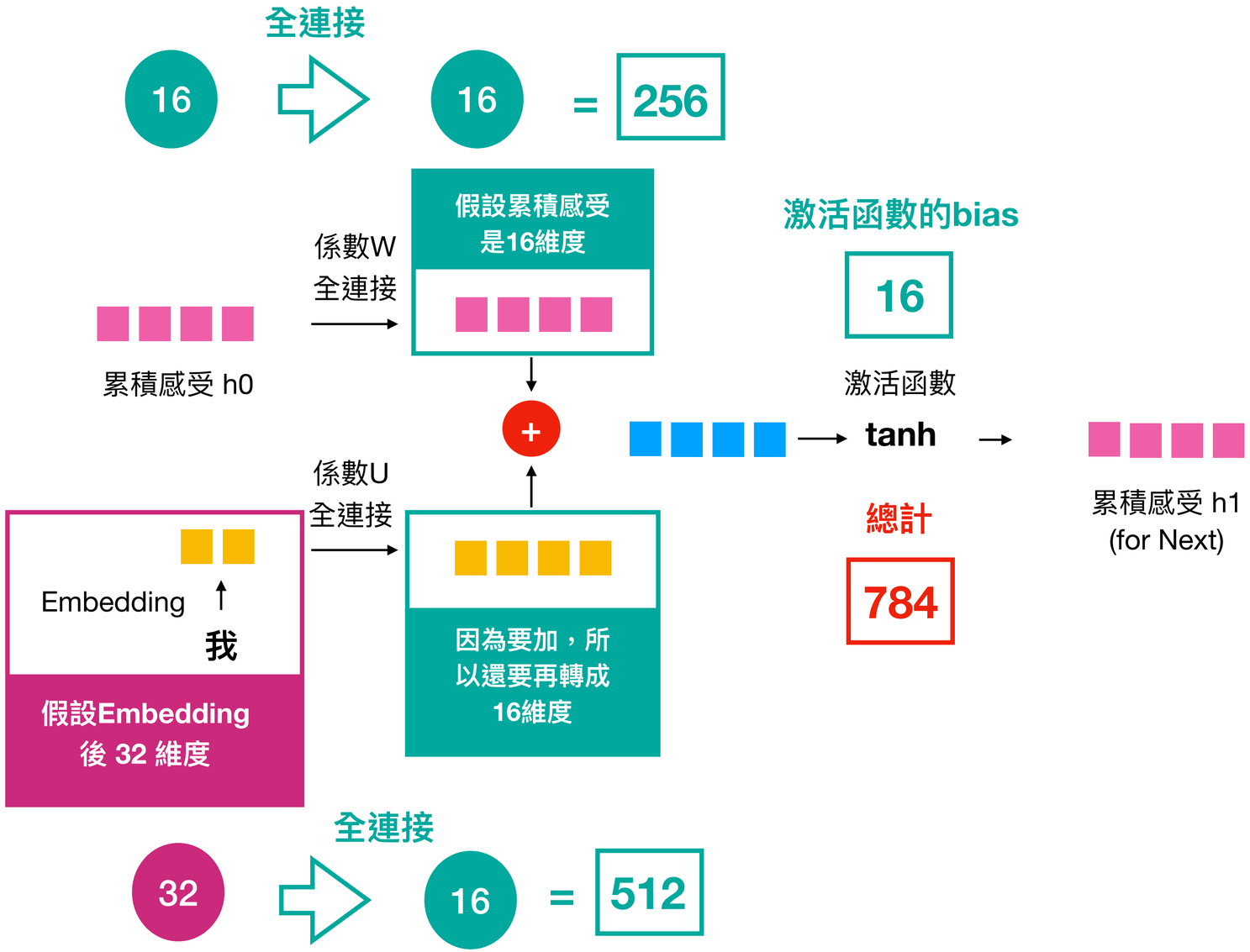
recurrent neural networks



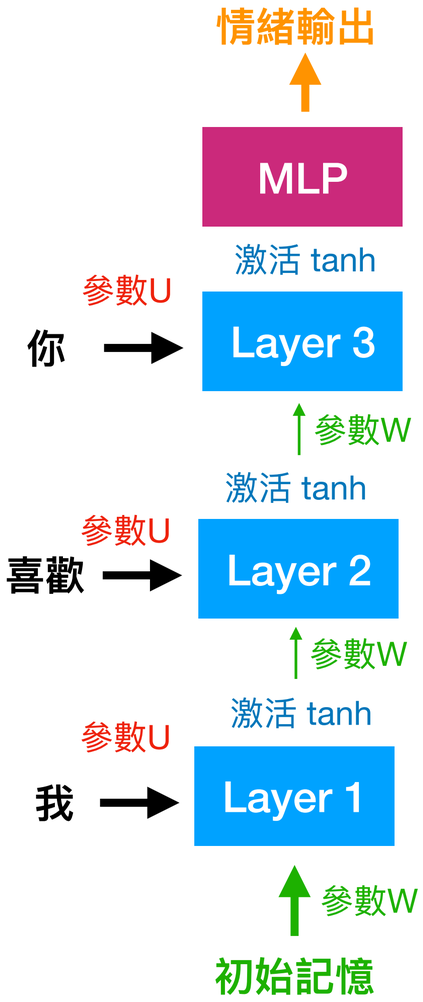
RNN


累積感受 t = 0
係數W
全連接
係數U
全連接
+
合併感受
激活函數
tanh
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3
Embedding
我
h1= tanh(W x h0 + U x 目前輸入)
累積感受 t = 1
如果需要的話就可以將這時間段的感受輸出，接上MLP之類

參數個數


全連接
16
16
=
256
假設累積感受
是16維度
係數W
全連接
累積感受 h0
激活函數的bias
16
激活函數
tanh
累積感受 h1
(for Next)
+
係數U
全連接
Embedding
我
假設Embedding
後 32 維度
因為要加，所
以還要再轉成
16維度
總計
784
全連接
32
16
=
512

RNN問題



翻過來變成直的你就懂問題在哪裡了，我們的『記憶輸入』不斷經過每一層的組合，而且這個層數一定是超級可怕，我們有時候一次就帶入 64 個詞作訓練

問題1: 反向傳播(梯度下降)

1. 回憶一下深度學習的精髓：負梯度(斜率)是我們更新參數的方向！
2. 總梯度 = 單個梯度相乘，而中間你乘上激活函數 tanh 的微分的時候，自然又被打折！打折！再打折！
3. 雖然你會想，我們的記憶傳導的路徑是用 W，這是每一層共享的，但你就會發現，離你越遙遠的記憶對於參數更新的貢獻就會越小

問題2: 正向傳播

1. 你可以想像一點，我們的記憶在每一層傳遞的時候都會乘上一個 W 因子，如果這個因子是『衰減(小於1)』的，那我們的記憶一定會隨著時間段慢慢消失
2. 反向傳播問題我們還可以使用 relu 激活來解決，但正向傳播的問題我們完全無法解決
3. 所以你會發現，我們的記憶都只有短期記憶，而沒有長期記憶

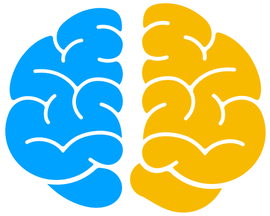
這就是著名的 RNN 記憶喪失問題，你也可以說 RNN 是一個『金魚腦』

RNN問題

你好像似懂非懂，只知道，從傳播的路徑上看來，我們好像不能記住長期的詞彙！那有沒有辦法用一個更簡單的說法來說明呢？

ＲＮＮ記憶

前面的感受　現在的輸入



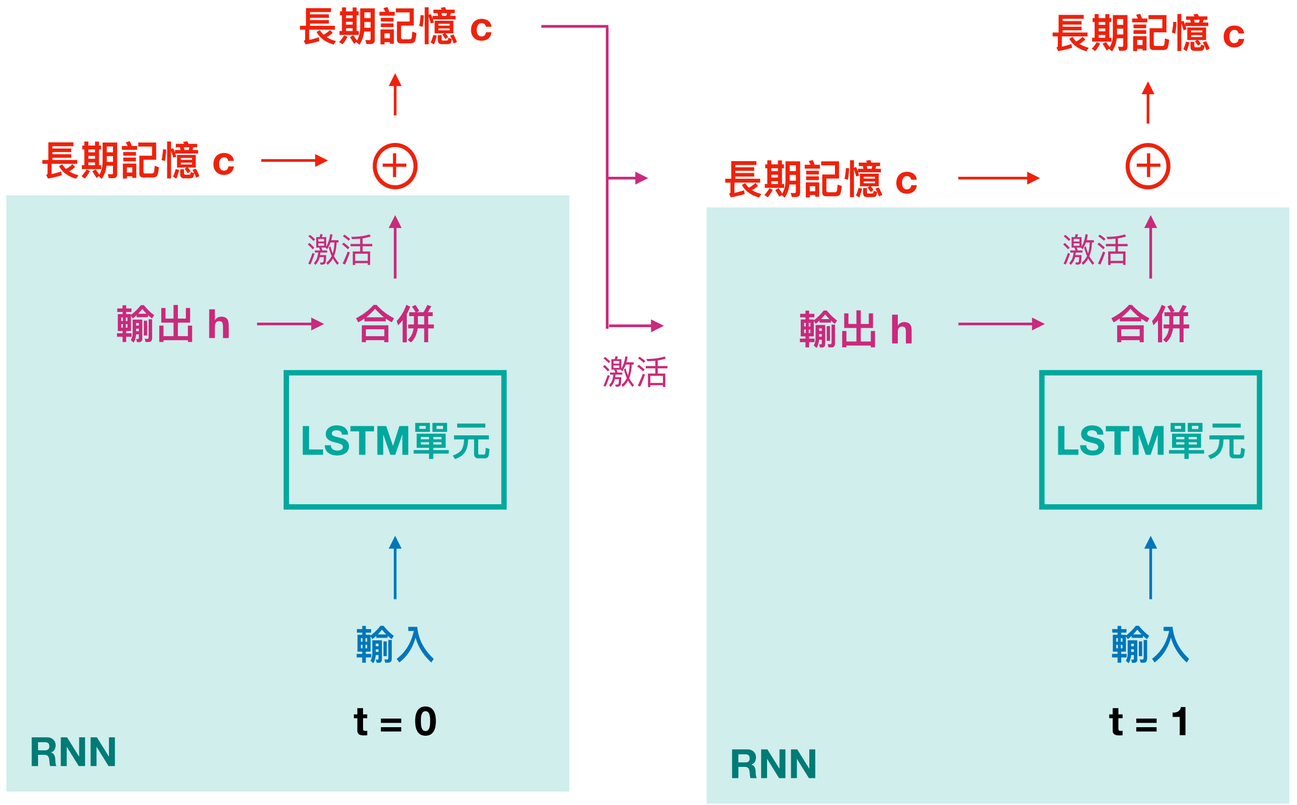
記住W這麼多　記住U這麼多

如果用一個大腦來形容ＲＮＮ，ＲＮＮ認為『每個時間段的輸入』一定對於後面會有影響，所以一定會把現在的輸入加進來考慮，但你會發現，很多輸入可能根本『不需記憶』，譬如說『是』，『在』...之類的詞，而你的ＲＮＮ還是會考慮這些東西，就導致你的記憶被這些東西佔掉位置

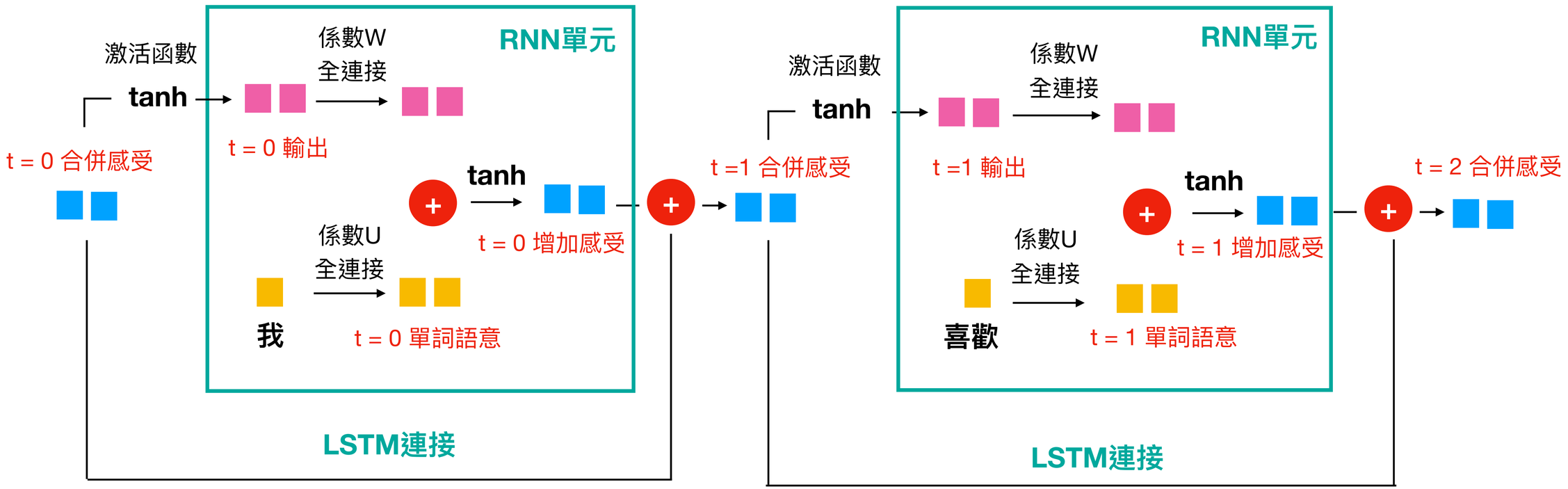
如果讓你翻一本書，你可能記得前五個詞有出現過『是』，但前十頁有沒有出現『是』你肯定不記得！白話來說，你的記憶應該分成『長期記憶』和『短期記憶』，『長期記憶』會記得比較關於『大綱』『感受』這樣的事情，而『短期記憶』比較記得『出現詞』這類的事情，我們的ＲＮＮ對於每個輸入都會考慮並加入記憶的這種做法，其實比較像是我們大腦中的『短期記憶』

LSTM基礎概念

Long Short-Term Memory

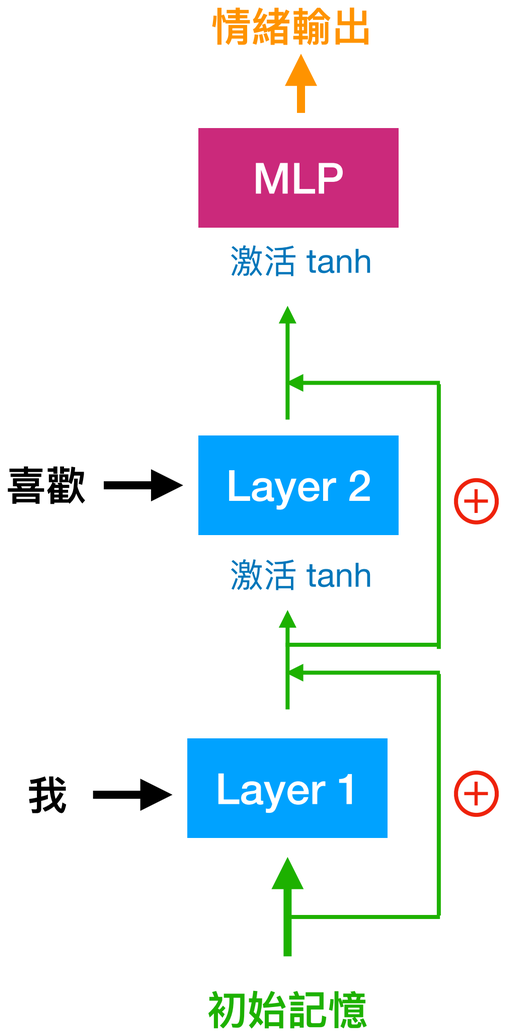


LSTM



如果要用一句話形容 LSTM:
累積感受[t] = 累積感受[t-1] + 增加感受[t]

LSTM解法



LSTM的想法其實就是我們在卷積網路 resnet 的想法，利用把輸入加到輸出，這樣我們只要學習輸入和輸出的『殘差』就可以了

影響 1: 對於參數的學習

影響 2: 對於記憶的影響

Layer函式 Layer(輸出, 輸入) = W * 輸出 + U * 輸入

單純RNN

輸出[1] = Layer(輸出[0], 輸入)

LSTM

輸出[1] = 輸出[0] + Layer(輸出[0], 輸入)

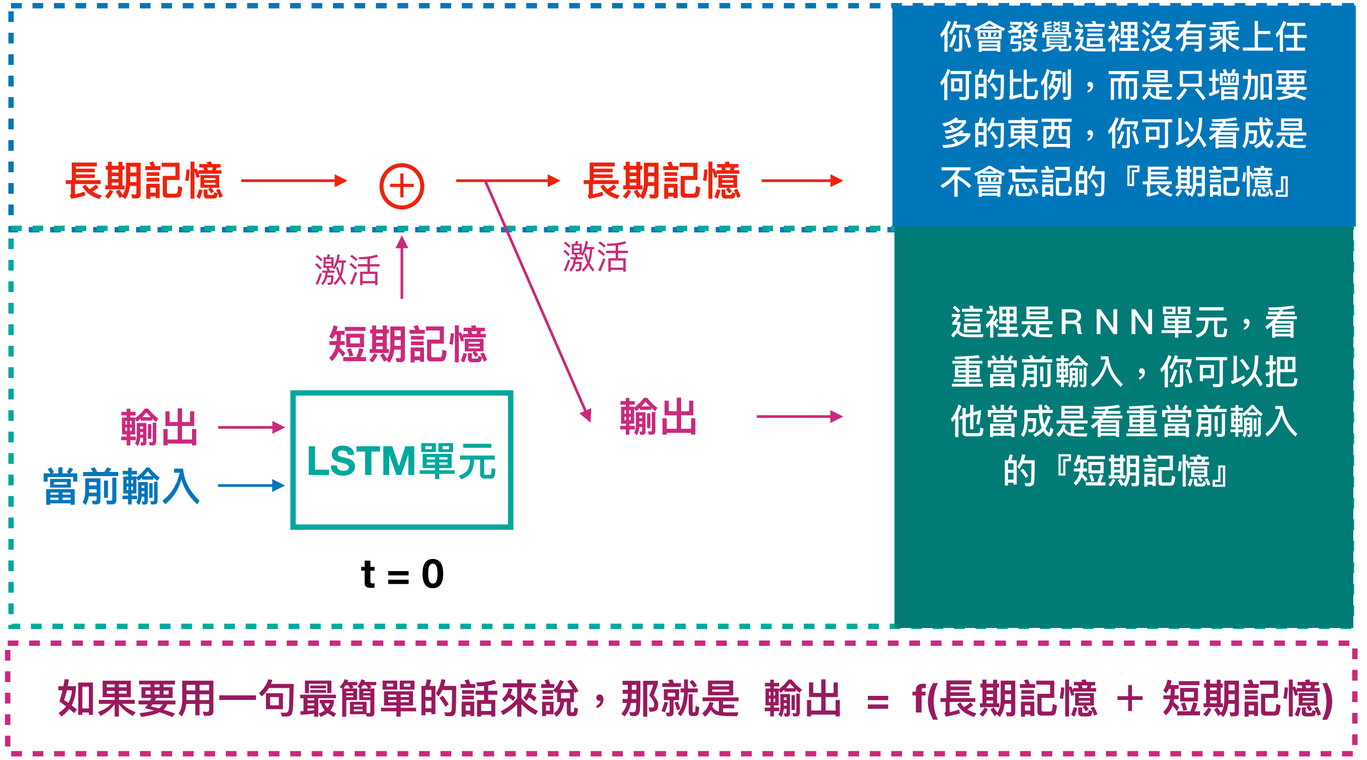
1. 有了殘差以後，其實就像是我先給你一個地基，先把上次的記憶完整拿過來，再做一個 fine-tuned 就好，要訓練的對象從完整函數變成殘差函數，要訓練一個參數漂亮的層就變簡單了

1. 有了 LSTM 後，我們相當是直接把上次的記憶『拷貝』過來，再增加這次該新增的記憶，我們擁有了一條『強力的記憶線』

LSTM是一個超群的記憶大師：因為你是以記憶為主，增加這次要多的東西

長短期記憶

剛才我們從角度來考慮問題解決，那有沒有一個更簡單的想法解釋為何我們 LSTM 叫做『長短期記憶』呢？

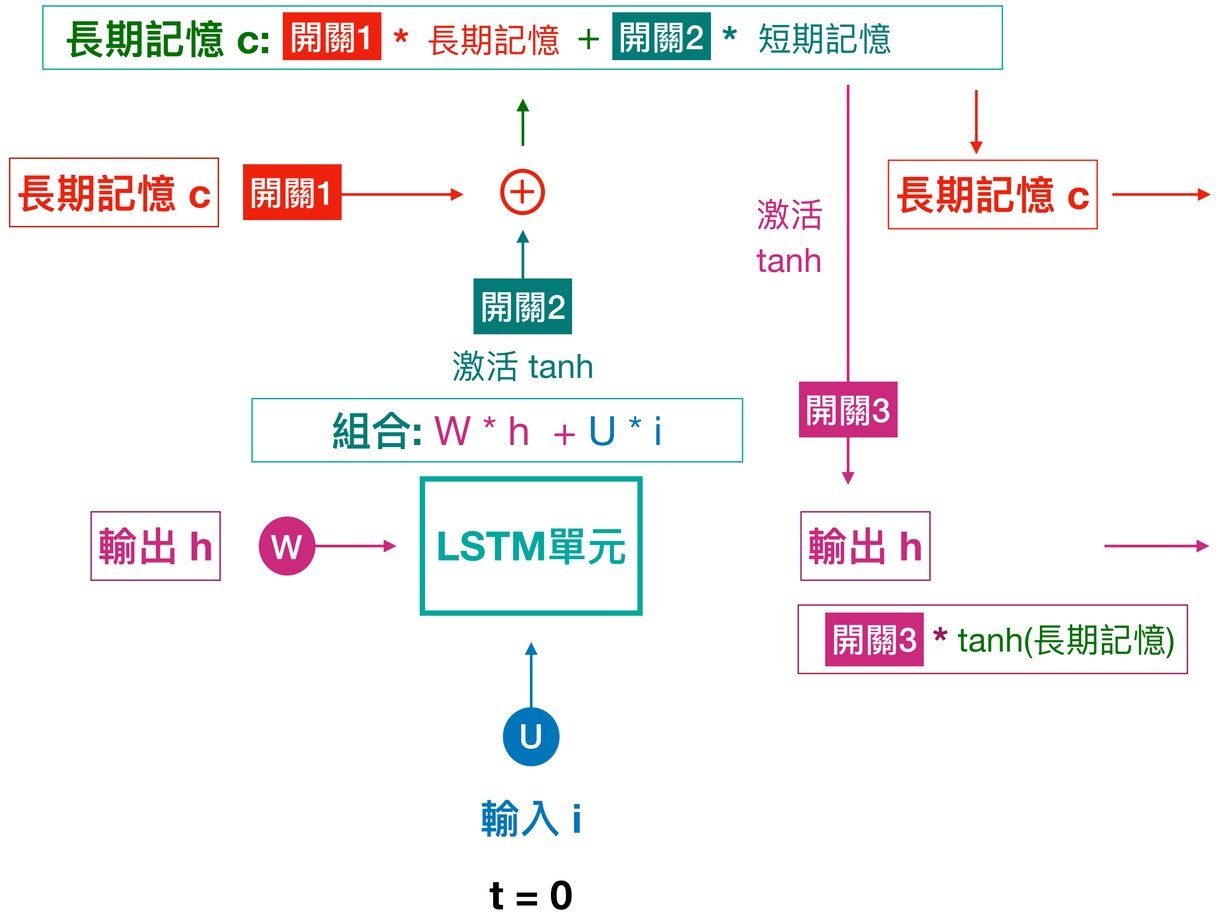


LSTM最後改良

之前我們的做法都是將『上次的記憶』完整的拿過來，這時候我們想到一個魔鬼的想法，我們可不可以多加上一個功能：控制我要『多少%的上次記憶』和『多少%的這次輸入』和『輸出多少%的記憶』呢？

加上開關是一個好想法

開關 (0~1)	記憶		記憶 (過濾)
0.8	2	=	1.6
0.0	1		0
0.2	3		0.6
1.0	5		5



開關製造

開關的意義

開關1 『遺忘閘』(forget gate)
決定多少『上次記憶』通過

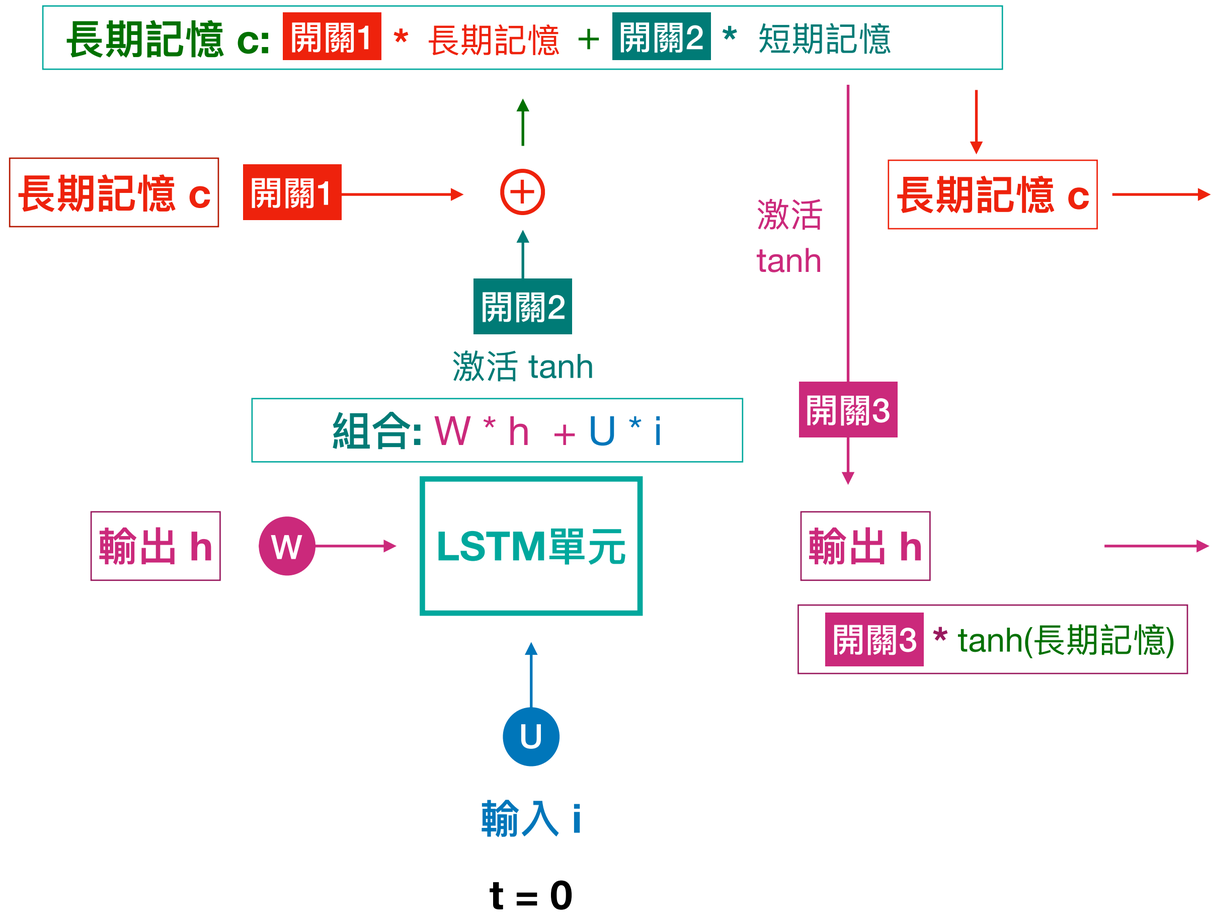
開關2 『輸入閘』(input gate)
決定多少『這次輸入』通過

開關3 『輸出閘』(output gate)
決定多少『這次輸出』通過

開關如何製造呢？

每次的開關狀態都不一樣，所以當然由這次輸入和上次記憶組合，並且使用 sigmoid 保證值落在 0 - 1 之間

開關 = sigmoid(參數 * h + 參數 * i)



參數個數

開關 = sigmoid(參數 * h + 參數 * i)

h: 16　　i: 32

開關1	U: 32 x 16 = 512
開關2	W: 16 x 16 = 256
開關3	
記憶增加	BIAS: 16

4 x (512 + 256 + 16) = 3136