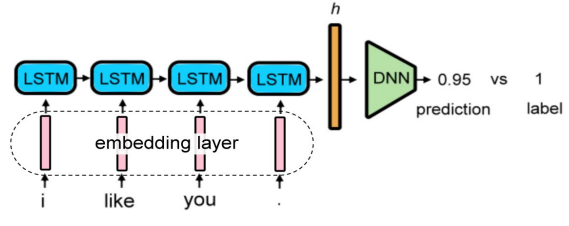


# Machine Learning HW5

學號：r06921081 系級：電機碩一 姓名：張邵瑀

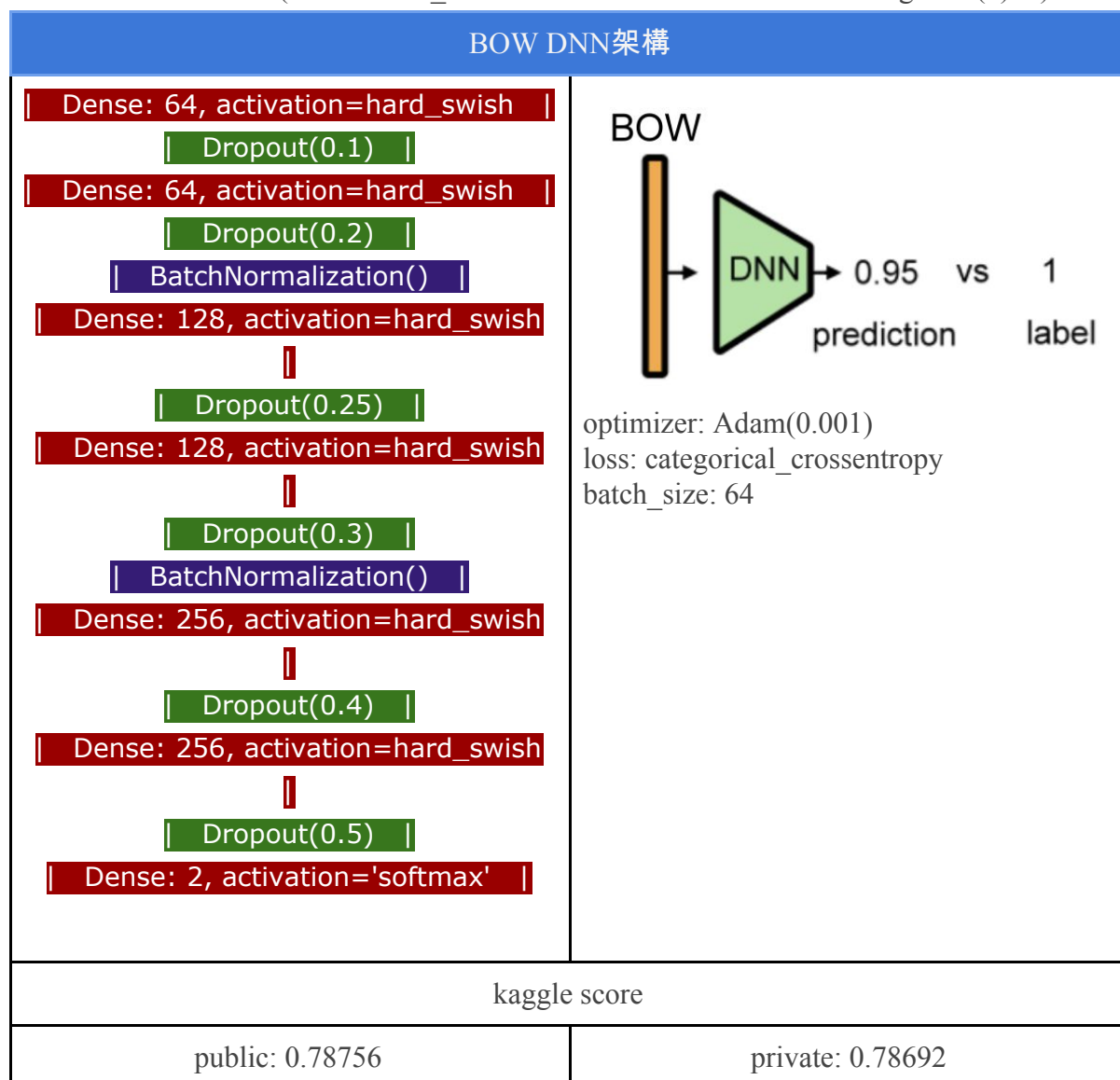
1. (1%) 請說明你實作的 RNN model，其模型架構、訓練過程和準確率為何？

答：這個模型架構是參考網路上一些簡單的RNN example的典型用法，並經過更換掉一些參數，如activation，加上recurrent\_dropout等等的數值，因為之前使用過leakyRelu跟softplus後想要找看看有沒有結合兩者優點的activation，所以參考在keras中自定義swish用sigmoid(x)\*x實作hard\_swish: hard\_sigmoid(x)\*x，我的構想是此問題要分成兩類，那就用分割更明確的hard\_sigmoid取代sigmoid，並將RNN層的activation也改為hard\_sigmoid，並藉由semi-supervise的資料(約5萬筆，提升約0.5%準確率)達到單個model 0.83465的準確率，而訓練過程則是用K-fold的方式把資料切成12等份，輪流當validation set訓練12種model再取出validation accuracy最高的model，接著再試著對準確率較高的model做ensemble可達到0.83616的準確率。

RNN架構	
<pre>  Embedding   Bidirectional: GRU: 256 activation: hard_sigmoid return_sequences = True dropout = 0.2 recurrent_dropout = 0.4 kernel_initializer: he_uniform  Bidirectional: GRU: 256 activation: hard_sigmoid return_sequences = True dropout = 0.25 recurrent_dropout = 0.5 kernel_initializer: he_uniform    BatchNormalization    Dense(256,activation=hard_swish)    Dropout(0.5)    Dense(256,activation=hard_swish)    Dropout(0.5)    Dense(2, activation='softmax') </pre>	 <p>word2vec: 使用gensim套件實作，把每個詞轉成200維的向量，window取10 tokenize: 使用keras內建的tokenizer</p> <p>optimizer: Adam(0.003) loss: categorical_crossentropy batch_size: 1300</p>
kaggle score	
public: 0.83465	private: 0.83272

2. (1%) 請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？

答：我使用keras中內建的texts\_to\_matrix把句子轉成向量，但因為記憶體上的限制所以我使用10000維的vector(使用Tokenizer(num\_words=10000,filters='\t'))使用DNN訓練，我使用的model如下圖，雖然我設定跑1000個epoch，但其實在前幾個epoch就已經收斂，後面愈訓練loss不但不會下降，準確率也會下降，而且準確率大概在78~79%這附近就是極限了。(備註：hard\_swish為自定義的activation 為hard sigmoid(x)\*x)：



3. (1%) 請比較bag of word與RNN兩種不同model對於"today is a good day, but it is hot"與"today is hot, but it is a good day"這兩句的情緒分數，並討論造成差異的原因。

答：在BOW裡並沒有前後文的概念，單純計算每個文字的頻率與其情緒分數，又因為兩句話的向量一樣，所以分數一模一樣，但RNN模型應該是觀察文字最後幾個字的意涵做出的預測。

	RNN	bag of word
today is a good day, but it is hot	0.96555728(負面)	0.66426373(負面)

today is hot, but it is a good day	0.97569917(正面)	0.66426373(負面)
------------------------------------	----------------	----------------

4. (1%) 請比較"有無"包含標點符號兩種不同tokenize的方式，並討論兩者對準確率的影響。

準確率	有標點符號	無標點符號
public	0.78462	0.80150
private	0.78524	0.80038

答：在這裡我濾掉的是 '~', '!', '@', '#', '\$', '%', '^', '&', '\*', '(', ')', '\_', '+', '-', '=', '[', ']', '{', '}', '\\', '|', ';', ':', '\'', '"', '\", '\'', '/', '<', '>', '?', 可以看到，沒有使用標點符號的句子準確率明顯下降很多，甚至過不了 simple，因為有些句子的段落跟情緒表達很有關係，並且有些如 '!' 或是 '?' 等等可以表達情緒的符號也被一並濾掉，在前後句判斷上可能出現更模糊的地帶，並且在 predict 時發生失去斷句的依據。（此處使用的 model 為較早期的版本所以準確率較低）

5. (1%) 請描述在你的 semi-supervised 方法是如何標記 label，並比較有無 semi-supervised training 對準確率的影響。

答：我的作法是用正確率 83.415% 的模型，對沒有 label 的 Data 做預測，取 0.7~0.935 準確率左右(主要目的是取和 label data 差不多數量的資料約 20 萬筆)的正面情緒或負面情緒預測分數來當作資料，由老師上課提到的非黑即白可以提高一定程度的準確率但相差的幅度並不會有飛躍性的成長，大概在 0.5% 以內，但是用此方法提高準確率的 model 再去 predict no-label data 再取該區間 data 來重複做可以有效的提高準確率，以下是單個 model 的結果，更進一步想我把訓練出來的 model 做 ensemble 提高準確度之後再做重複的訓練提高精準度，但我發現我用 hard\_swish 跟 sigmoid 做出來的結果 softmax 的評分大都會集中在 0.95~1 (如第 3 題結果)，使得我再度改變架構把 hard\_swish 改成 swish，hard\_sigmoid 改成 tanh 然後把 range 加寬到取 softmax 0.65~0.88 分的資料來使用，因為訓練資料變多了，所以可以把更多 labeled 資料切成 validation set 以提高評估的一般性，最後反覆訓練可以達到單個 model 83.950% 的準確率，最後再用 ensemble 把 12 個 model 一起預測，但僅僅提升到 0.84030 也許是我的區間抓太鬆，或是已經做到這個 model 的極限了。

單個 model	supervised	semi-supervised(one round)	loopyly-semi
public	0.82979	0.83295	0.83950
private	0.82810	0.83100	0.83704

reference:

<https://machinelearningmastery.com/develop-word-embeddings-python-gensim/>

<https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/>

<https://machinelearningmastery.com/deep-learning-bag-of-words-model-sentiment-analysis/>

<https://github.com/keras-team/keras/issues/8716>