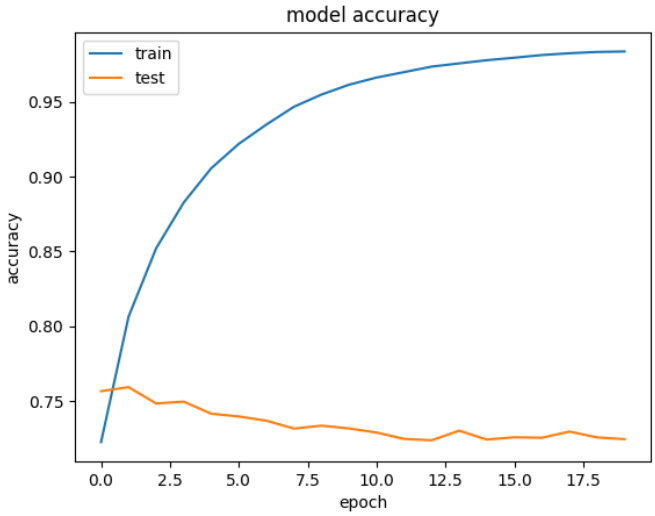
Machine Learning HW6 Report

學號：B06502149　系級：資工二　姓名：張琦琛

1. (1%) 請說明你實作之 RNN 模型架構及使用的 word embedding 方法，回報模型的正確率並繪出訓練曲線*

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 200, 256)	30154496
lstm_1 (LSTM)	(None, 96)	135552
dense_1 (Dense)	(None, 1)	97
activation_1 (Activation)	(None, 1)	0

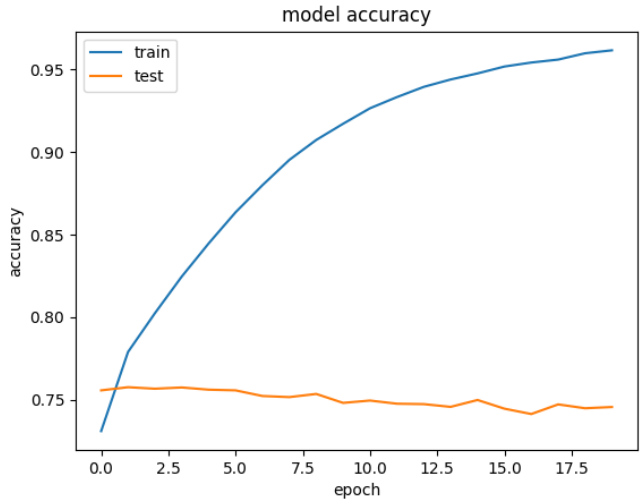
Total params: 30,290,145
Trainable params: 30,290,145
Non-trainable params: 0



對於模型架構的部分，我是使用 word2vec 的方法，用 gensim Word2Vec pretrained 好的 model 給 Embedding layer，並將每個詞的向量設為 256 維，每一句最多有 200 個詞，之後再接上一層的 LSTM ，dropout rate 為 0.2，最後用 sigmoid function 激活，而 loss function 使用 binary_crossentropy，optimizer 用 Adam。我有嘗試提高參數量或是換成 GRU 和 Bidirection_LSTM，但是結果都沒有比一層的 LSTM 佳。但不論是哪種 Model，在 Validation set 上最好的結果都是出現在第 2 個 epoch。而最後在 Kaggle 上的分數為 Public : 0.75850 Private : 0.75450。

2. (1%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正確率並繪出訓練曲線*。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                1280064
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 1,280,129
Trainable params: 1,280,129
Non-trainable params: 0
```

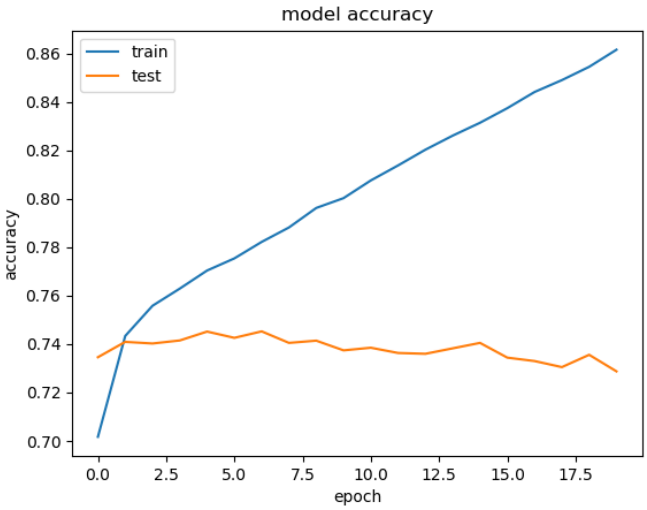


我只用了一層 Dense Layer，並加上 0.2 的 dropout rate，將一句話變成 30000 維的詞袋向量，也就是統計出現頻率最高的前 30000 個字放進字典。BOW + DNN 的方法在 Kaggle 上的分數為 Public : 0.72340 Private : 0.72410。我認為是因為 BOW 只是利用句子的結構將一句話轉成向量，這種方法的向量維度較 Word2Vec 低，且不能將意思相近的字詞轉成距離相近的向量，跟 RNN 相比，缺乏了一些語意的判斷。

3. (1%) 請敘述你如何 improve performance（preprocess, embedding, 架構等），並解釋為何這些做法可以使模型進步。
在 Preprocessing 的部分我將句子中一些重複出現的特殊符號縮減成一個，因為我發現在 Data 中常常會有一些很長的表情符號、問號、驚嘆號、波浪等等，這讓 jeiba 在做斷詞的時候效果降低，也就間接影響判斷，此外，我讓一個句子的長度最多有 200 個字詞，缺的部分補 0，多的部分去除，因為有些句子可以多達 10000 多的詞，但大部分的句子都只有 100 個左右的詞，若讓每句都補成 10000 多個

詞，將會嚴重失去語意。Word embedding 的部分，產生詞向量的時候會根據左右 5 個詞判斷，透過 word2vec 的 windows 參數控制，讓詞與詞之間的關聯性更準確，並讓詞向量維度為 256，設太小的話會無法區別詞的差異。至於最後我是將 5 個 model 做 Ensemble，提高 Kaggle 上的分數。

4. (1%) 請比較不做斷詞 (e.g., 以字為單位) 與有做斷詞，兩種方法實作出來的效果差異，並解釋為何有此差別。
   將沒有做斷詞直接以字為單位的資料，給 model 訓練，在 Kaggle 上的分數為 Public : 0.72660 Private : 0.73540，以下為其訓練曲線。



   在沒有做斷詞的情況下，準確率比有做斷詞來的低。我認為是因為在中文的字詞中，雖然有些字詞拆開來仍能表達原本的意思，但是大部分的字詞拆開後的意思會跟原本的有很大差異，例如:[低能] -> [低],[能]; [邊緣人]->[邊],[緣],[人]。如此一來，Word2Vec 的效果將變得不好，由於語意的改變會讓他將字詞轉向量時，可能發生儘管沒有相關性，兩者的向量卻會很接近，或者是應該高度相關的字詞，兩者的向量卻差很遠，讓 RNN 無法根據正確的語意判斷。

5. (1%) 請比較 RNN 與 BOW 兩種不同 model 對於 "在說別人白痴之前，先想想自己 "與"在說別人之前先想想自己，白痴" 這兩句話的分數（model output），並討論造成差異的原因。
   BOW 的 model output 皆為 0.7281，由於這兩句話所產生出來的詞袋向量非常相近，因此兩句話的判斷結果都相同，而判斷成惡意的原因可能是'白痴'這個強烈的關鍵字導致。而 RNN 也將兩句都判斷成 1，也就是兩句都是惡意留言，但是 model 預測第一句的機率是 0.51281, 第二句為 0.54878，都非常接近 0.5，雖然第一句判斷錯誤，但是我認為 model 還是有學到語意的判斷，畢竟通過 sigmoid function 後仍接近 0.5，可能是'白痴'這個詞太強烈導致機率向 1 偏移。