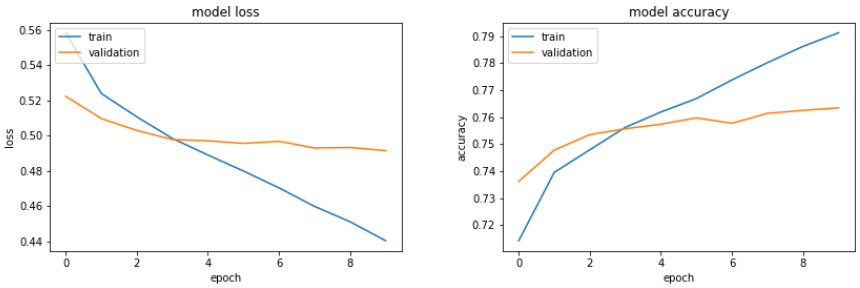
Machine Learning HW6 Report

學號：b05901025 系級：電機三　姓名：王鈺能

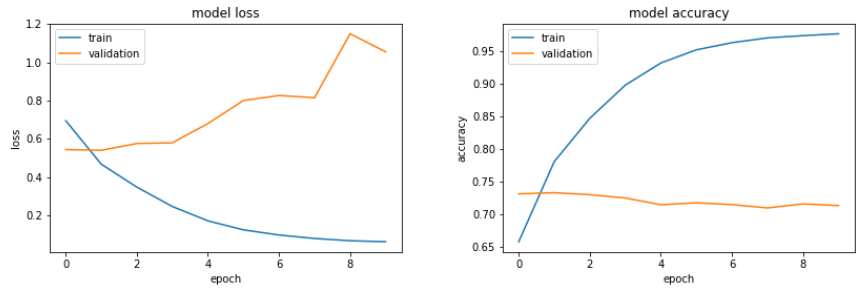
1. (1%) 請說明你實作之 RNN 模型架構及使用的 word embedding 方法，回報模型的正確率並繪出訓練曲線*

   我用 training 和 testing data 作為 corpus 訓練 word2vec，再將其接到 RNN 裡一起 train，RNN 架構為 word2vec embedding + bidirectional LSTM(512) + attention(100) + dense layer(256) + 一層 output layer(2)，此外層層間有些許 dropout，訓練正確率單一 model val_acc 最高為 0.7659，bagging 後在 private testing set 正確率達 0.77090。下圖為訓練曲線，可以看出 validation loss/acc 正常的下降/上升。



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*。

   我用 training 和 testing data 作為 corpus，將出現次數小於三的詞丟棄後將剩下詞編號，做成 BOW 後丟到 3 層 dense layer(256, 256, 128)，在街上 output layer(2)，訓練正確率 val_acc 最高為 0.7329。下圖為訓練曲線，在第二個 epoch 後其實就已經 overfit 了，validation loss/acc 上升/下降。



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。

   Preprocess ：

   原本我把所有詞(約 12 萬)都丟進 model 裡，每個都對應一個 embedding vector，後來我把出現次數小於 3 的詞都當作 unknown，如此只剩下約 5 萬的詞，訓練效果好很多，原因應該是原本的參數量太大而且其實出現次數少的做 embedding 也

比較不準，所以有很多不好的 data 跟參數，把這些 data 和參數拿掉之後，embedding vector 比較好而且也較不容易因為參數量過大而 overfit。

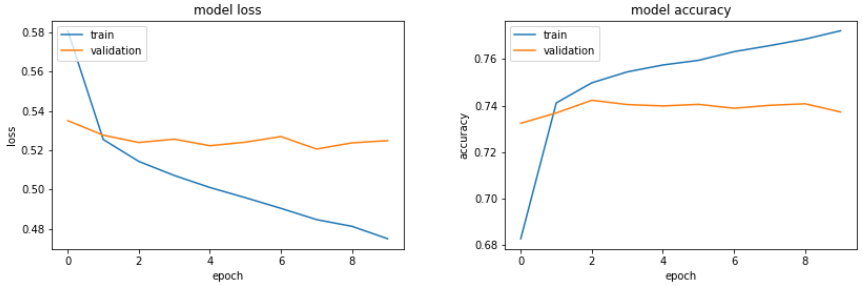
Embedding：

我試過(1)不用 word2vec 直接 train embedding(2)用 word2vec 後參數不能動+RNN(3)用 word2vec 後參數+RNN 一起 train，效果是(3)最好，這結果相當合理，因為除了 corpus 的語意關聯之外，有些 task-specific 的語意關聯可以在和 RNN 一起 train 時做修正，因此效果最好。

Attention：

由於一句話中每個字對於語意的重要性不同，加入 attention layer 理論上可以讓 model 專注於比較重要的詞使 performance 變好，但我做起來其實沒什麼感覺，好的幅度不明顯。

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。



我用同樣的模型架構 train 沒有斷詞的 corpus，訓練曲線如上圖，顯然不如有斷詞的 performance，我認為是因為本來句子意義的基本單位是「詞」而非「字」，因此用詞丟進 model 給的資訊是比較精確的，所以做斷詞後訓練的結果會比較好。

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。

[非惡意 惡意]	"在說別人白痴之前，先想想自己”	"在說別人之前先想想自己，白痴"
RNN	[0.54006422 0.45993581]	[0.34032866 0.65967137]
BOW	[0.0073674 0.99263257]	[0.0073674 0.99263257]

對 BOW 來說，兩句話斷詞後給的 input vector 是一模一樣的，所以 output 是一模一樣的，而且他可能看到一些關鍵字就當作是惡意(例如：白痴)，所以不管前後文順序就直接判定為惡意；而 RNN 會考慮前後文關係，因此同樣的詞但順序不同，就會得到不同的 output，在這裡也成功解讀出兩者語意的不同。