Machine Learning HW6 Report

學號：R07943107　系級：電子所碩一　姓名：徐晨皓

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*。**

A. RNN 模型架構及 word embedding 方法

Word embedding 使用 gensim 的 Word2Vec，細節在問題 3 解釋。RNN 模型使用三層 Bidirectional GRU layers 再接著兩層 dense layers。激活函數方面，GRU 使用 tanh，dense 使用 relu，最後的 output layer 使用 sigmoid。Dropout 為 0.3；epochs 為 12。Optimizer 使用 adam。

Layer (type)	Output Shape	Param #
bidirectional_1 (Bidirection	(None, 40, 512)	701952
bidirectional_2 (Bidirection	(None, 40, 512)	1181184
bidirectional_3 (Bidirection	(None, 512)	1181184
dense_1 (Dense)	(None, 256)	131328
batch_normalization_1 (Batch	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 2)	514

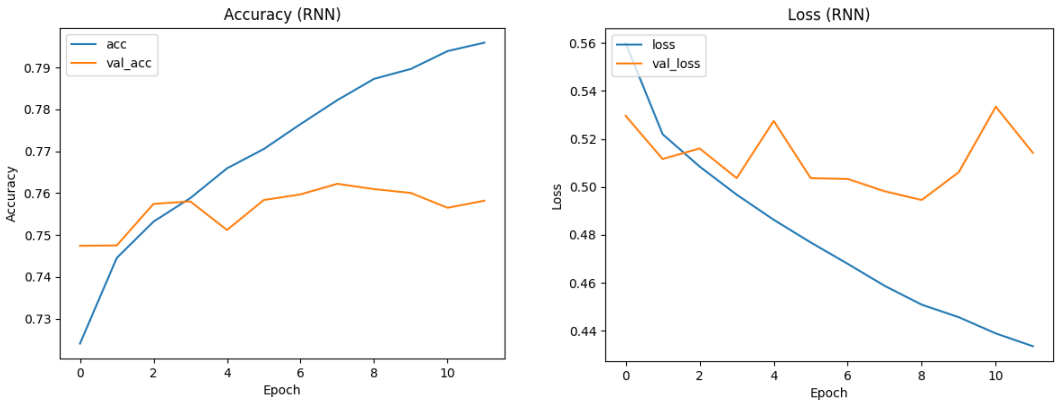
Total params: 3,197,186
Trainable params: 3,196,674
Non-trainable params: 512

B. 模型正確率

(1) Private score: 0.75980

(2) Public score: 0.76450

C. 訓練曲線



（取 10%的 training data 當作 validation set）

由上二圖可以觀察到在訓練過程中：

（1）RNN model 的 training accuracy 可以達到約 0.81。但 validation accuracy 只落在 0.75 和 0.76 之間。

（2）RNN model 的 training loss 下降至約 0.43。validation loss 只落在 0.50 和 0.54 之間。

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*。**

A. BOW+DNN 模型架構

BOW 的字典是使用 word2vec 蒐集而成。為避免記憶體不足的問題，將 min_count 設為 20，最後 word dictionary 共蒐集 10283 個字詞。

DNN 結構為三層 dense layers，每層的 cell 數量為 128。Iteration 數為 20；dropout 為 0.2。Epochs 設為 20。

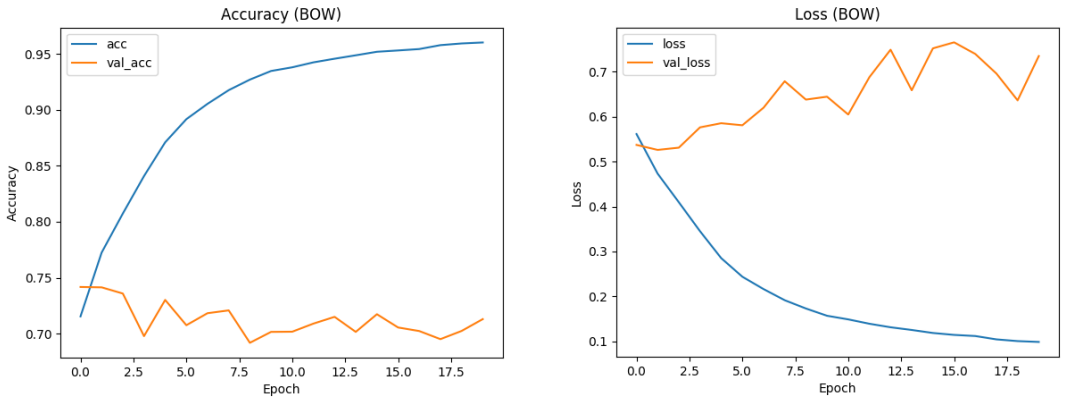
```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               1316352
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 258
=================================================================
Total params: 1,351,170
Trainable params: 1,350,402
Non-trainable params: 768
_________________________________________________________________
```

B. 模型正確率

(1) Private score: 0.71030

(2) Public score: 0.71270

C. 訓練曲線



（取 10%的 training data 當作 validation set）

由上二圖可以觀察到在訓練過程中：

（3）DNN model 的 training accuracy 可以達到約 0.96。但 validation accuracy 卻在 0.68 和 0.74 之間震盪。

（4）DNN model 的 training loss 下降至約 0.10。但 validation loss 卻逐漸上升並劇烈震盪，可能是為了讓 model 更 fit training data 造成 validation loss 增加。

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。**

A. Preprocess

在 preprocess 中，基本想法是將留言裡不影響判斷惡意性的字詞去除。由於留言裡有 Bxx 與 bxx 對於各樓的回覆標記，這些標記並不影響留言的惡意性，故將這些字詞移除。原以為表情符號可以被移除，但經過實驗發現表情符號可以幫助判斷留言的惡意性，例如有一則留言只有一個比中指的符號，若將表情符號移除就無法判斷此留言的惡意。

B. Embedding

使用 gensim 的 Word2Vec 實作 word embedding。參數為 size=200, window=5, min_count=3, iter=30。每則留言最大的 word vectors 數量為 40。每個 vector 都有做 normalization ($\frac{vec-mean(vec)}{std(vec)}$)，從實驗發現若不做 normalization，training accuracy 只能落在 0.7300~0.7400，但做完 normalization 後可以進步到 0.7500 以上。原因可能為未經 normalized 的 word vector 的數值差異可能很大，造成 model 不容易 fit training data。

C. Ensemble

此方法是幫助 performance 進步最大的方法。共使用 11 個 models，每個 model 皆隨機選取 90%的 data 為 training set，剩下的 10%為 validation set。這些 model 個別的 public score 基本上落在 0.7520~0.7580 之間。Ensemble 方式為全部 model 的 weight 都相同來進行投票，以獲得最多票的 label 作為 output label。以這樣多個 model ensemble 的方法可以避免單一模型的 overfitting，進而改善 performance，達到 0.76450 的 public score。

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。**

A. 有做斷詞

(1) Private score: 0.75300

(2) Public score: 0.75520

（只使用單一 RNN model）

B. 不做斷詞

(1) Private score: 0.74080

(2) Public score: 0.74830

C. 差異原因

由上面結果，可以發現不做斷詞的準確率較有做斷詞的低。可能是因為如果不做斷詞，雖然字的順序性還存在，但是一些詞的意義可能會變得模糊，使得 RNN 會判斷錯誤。

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。

A. RNN 分數

（1）"在說別人白痴之前，先想想自己"：[0.65654860, 0.53573583]

（2）"在說別人之前先想想自己，白痴"：[0.34529397, 0.78362780]

（只使用單一 RNN model）

B. BOW 分數

（1）"在說別人白痴之前，先想想自己"：[0.0263901, 0.7742242]

（2）"在說別人之前先想想自己，白痴"：[0.0263901, 0.7742242]

C. 差異原因

BOW 蒐集到的這兩句話的字詞為

['說','別人','之前','先','想想','自己',',','白癡']。

由於 BOW 並不會考慮字詞的先後順序，因此這兩則留言對於 BOW 來說，是完全一模一樣的 input，導致 BOW+DNN 對於這兩句話的 output 是一模一樣的。另外，可能是因為有‘白癡’這個字詞，因此 BOW+DNN model 判定此二則留言為惡意留言。

由於 RNN 會考慮字詞的先後順序，因此儘管這兩則留言所包含的字詞是一樣的，但因順序不同，RNN 會將這兩句話視為不同的 input。從 RNN output 結果，"在說別人白痴之前，先想想自己"應為非惡意留言，而"在說別人之前先想想自己，白痴"為惡意留言，應為正確的。