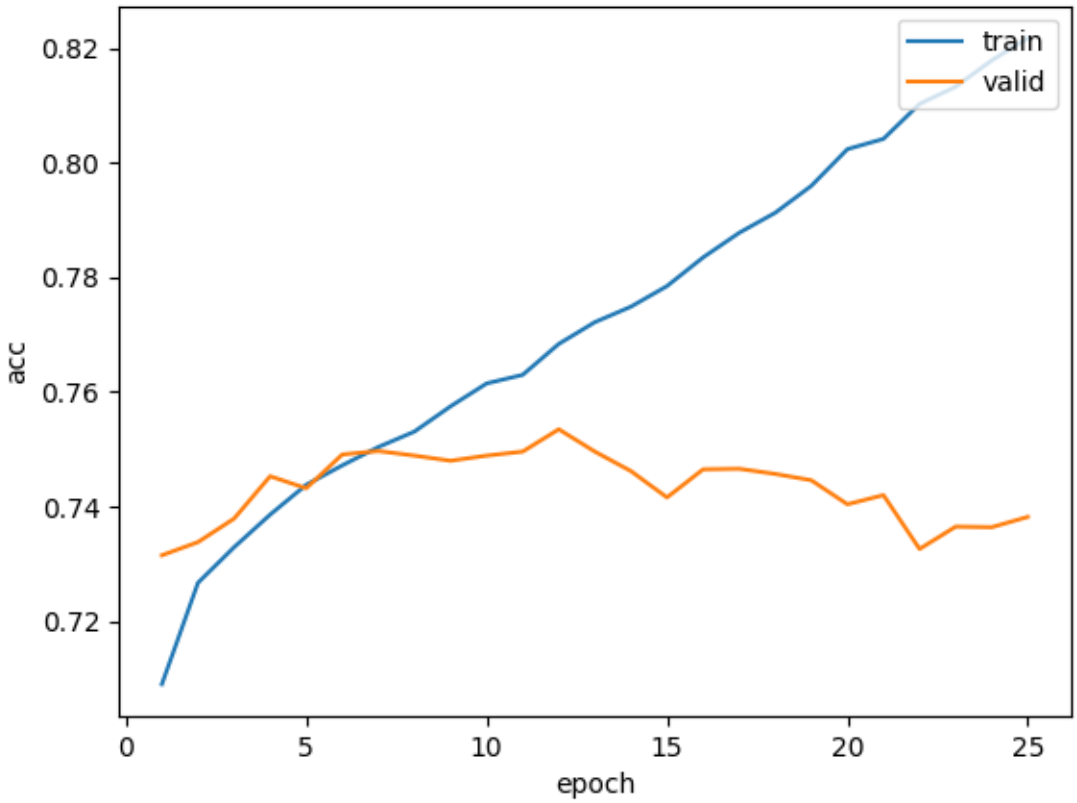
Machine Learning HW6 Report

學號：B05902128　系級：資工三　姓名：鄭百凱

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*

我用兩層雙向的 lstm，sequence size 是 60， lstm 的 output 是 300 維。然後我取在 lstm 中所有時間點的 output，對每一維做 mean 跟 max，當作 DNN 的 input。如此共有 300 * 2(雙向)*2(mean max) = 1200 維
然後 DNN 有三層，大小分別是 1200 -> 1000 -> 100 -> 2。Word embedding 取在 train 跟 test data 出現超過五次的詞彙用 gensim 去 train，每一個詞會對應到一個 100 維的向量。
RNN 在 trainacc 的上升幅度較 DNN 慢，valid acc 大約在 10-15 個 epoch 會達到最大值。丟到 kaggle 上的 model 是取最高 valid acc 的那次 epoch，結果如下。
Private score：0.74090
Public score：0.74680



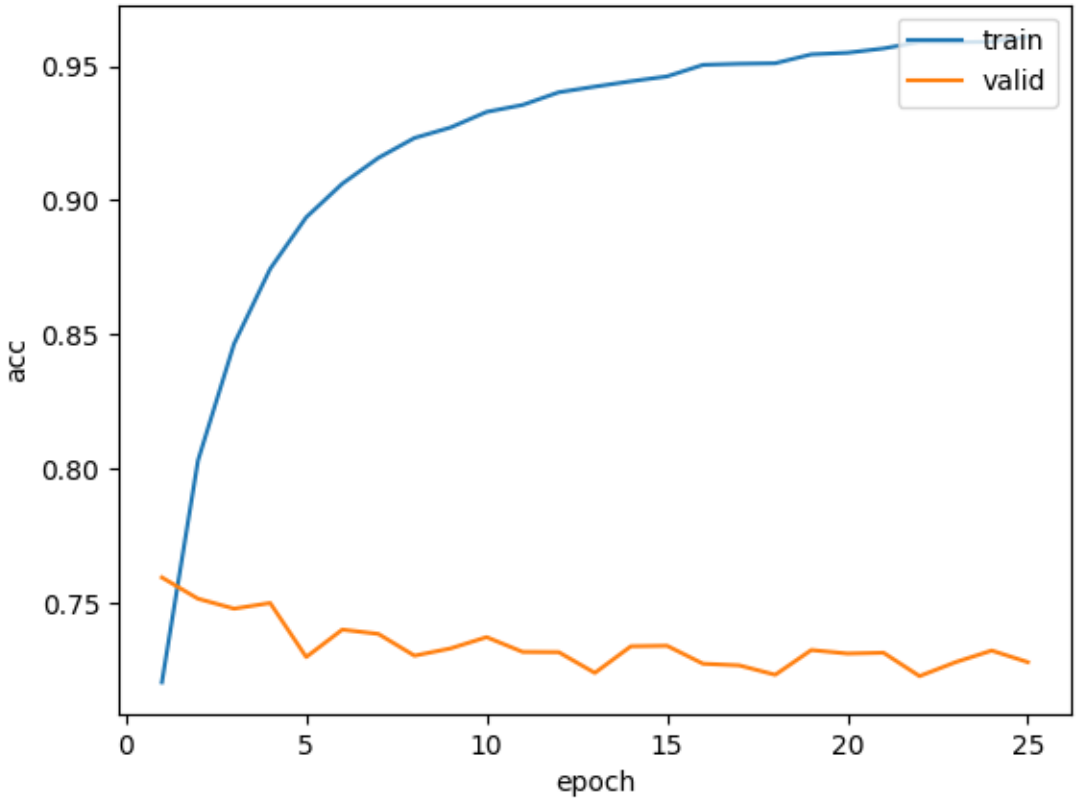
2. (1%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正確率並繪出訓練曲線*。

Train+test data 總共有約 120000 個詞彙，所以每一個句子會形成一個 120000 維的向量。然後丟進 DNN model 做訓練。Model 共有四層，每一層都有 0.5 的 dropout、batchnormalization，activate function 是 leakyrelu(0.2)，每一層的大小分別是 120000 -> 1000 -> 200 -> 50 -> 2，最後接一個 softmax。

相較於 RNN，DNN 比較容易 overfit，而我丟到 kaggle 上的 model 是所有 epoch 中在 validation set 上表現最好的 model，結果如下。

Private score：0.75570

Public score：0.75870



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。

(1). 將 gensim 的 wordvector mincount 改成 5，避免 gensim 訓練那些出現次數很少的詞彙。並將那些沒在 wordvector 當中出現的字隨機 random 一個向量來取代，

跟 padding 的 0 向量做區分。如此可以避免 model 將 unknown 的詞視作句尾，也能避免那些可能是打錯字或是沒人在用的詞彙來混淆 model。
(2)用兩層 lstm，增加 model 複雜度。
(3)對 lstm 用 dropout，避免 model overfit。

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。
做斷詞：
Private score：0.74090
Public score：0.74680
不做斷詞：
Private score：0.74050
Public score：0.74760

這次的 task 是找出惡意留言，通常惡意留言中的惡意詞彙都比較簡短，例如一些髒話跟罵人的話，而這些惡意詞彙當中出現的字其實也很多重複。像是看到「娘」基本上就有很高機率是惡意留言。所以這個 task 對 model 來說有沒有斷詞的差別並沒有很大。

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。

在說別人白痴之前，先想想自己
RNN： 惡意留言機率：0.3434
BOW： 惡意留言機率：0.5815
在說別人之前先想想自己，白痴
RNN： 惡意留言機率：0.5359
BOW： 惡意留言機率：0.5815

對 BOW 來說，這兩句話斷句之後是一樣的，估計是因為看到「白痴」這個詞，所以兩句話都被認定為是惡意留言。
RNN 則是會考慮到「白痴」這個詞出現的時間還有前後所接的詞，而有所差異。可能 model 看到「白痴」之後接的是「之前」，推測是反詰的語氣，因此變成非惡意留言。另一句則是在句尾看到「白痴」，因此推論是惡意留言。