Machine Learning HW6 Report

學號：b04104040　　系級：工海三　　姓名：解正安

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*

基本參數:

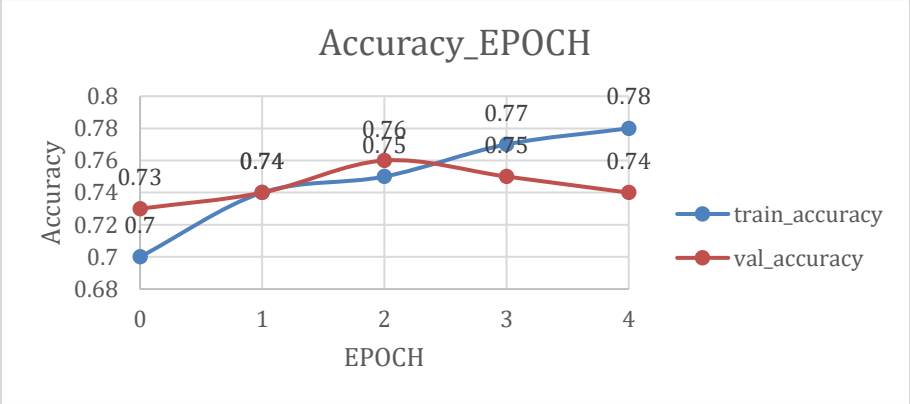
```
self.encoder = nn.LSTM(input_size=embed_size, hidden_size=self.num_hiddens,
                       num_layers=num_layers, bidirectional=self.bidirectional,
                       dropout=0.5)
if self.bidirectional:
    self.decoder = nn.Sequential(
                   #nn.Linear(num_hiddens * 10, 50), ->hidden
                   nn.Linear(num_hiddens * 4, 50), #->att
                   nn.ReLU(),
                   nn.Linear(50, labels),
                   nn.ReLU(),
                   )
```

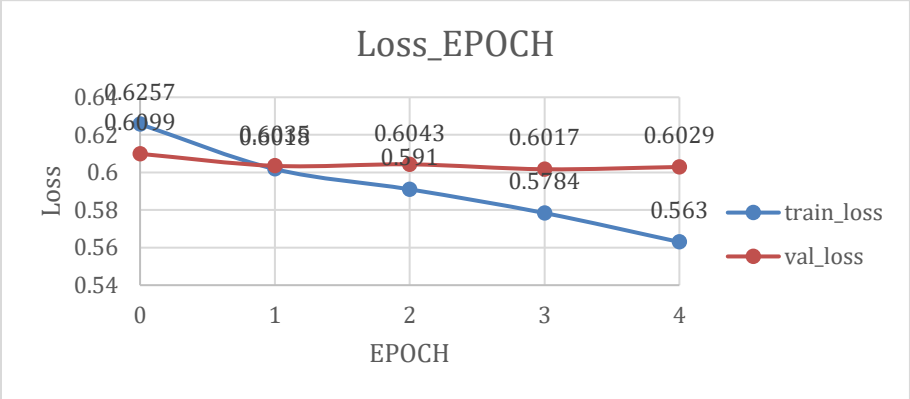
Epochs=5, Embed_size=300, Hiddens_size=300, Layers = 3, Bidirectional = True, BATCH_SIZE = 32, learning rate=0.001

模型採用 LSTM，為了避免 overfit，dropout 設為 0.5。而 LSTM 出來的 output，再由 average pooling 和 max pooling 去找適合的 hidden，最後經過兩層的 DNN 作為輸出。

Word embedding 採用 CBOW，依據上下文建立一個詞庫，並將 train 和 test 的 data 都進行訓練，window=5,iter=20。對於無法識別的字採用 sample code 的方式<UNK>放入。而最後的訓練正確率為: Private:0.75920 Public: 0.76390

曲線如下: (非最佳的那次訓練，因每次 data 都有 shuffle，結果並不穩定)





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*。

BOW

```
def make_bow(self,data,test):
    result=[]
    i=0
    for sentence in data:
        i+=1
        #print("\r==Make bow:"+str(i)+" sentenc
        vec = torch.zeros(len(self.word2index))
        for word in sentence:
            vec[self.word2index[word]] += 1
        result.append(vec)
    return result
```

DNN

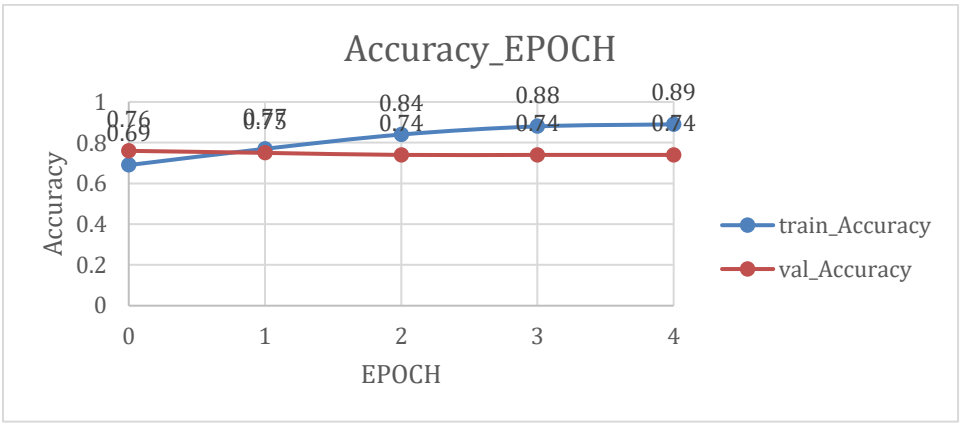
```
class BoW(nn.Module):
    def __init__(self, num_labels, vocab_size):
        super(BoWClassifier, self).__init__()
        self.linear = nn.Sequential(
                        nn.Linear(vocab_size, 2048),
                        nn.ReLU(),
                        nn.Linear(2048, 512),
                        nn.ReLU(),
                        nn.Dropout(p=0.5),
                        nn.Linear(512, 128),
                        nn.ReLU(),
                        nn.Dropout(p=0.5),
                        nn.Linear(128, num_labels),
                        nn.ReLU(),
                        nn.Dropout(p=0.5),
                        )
    def forward(self, bow_vec):
        outputs=self.linear(bow_vec)
        return outputs
```

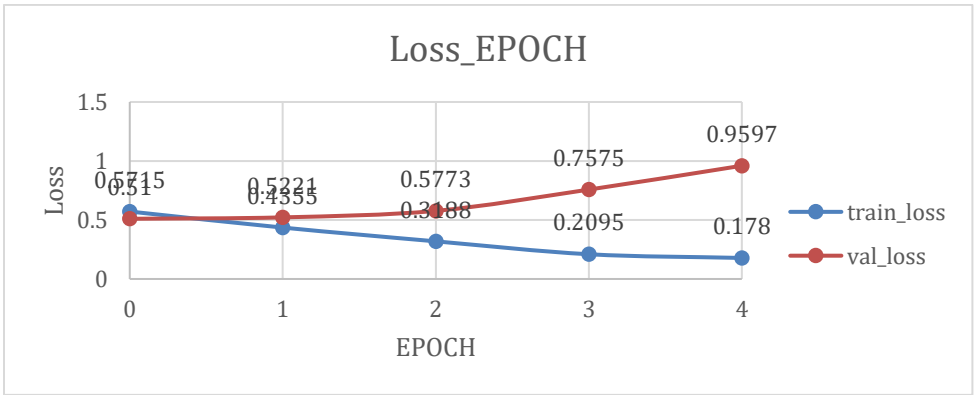
資料前處理同 RNN 在第三題有說明。而 BOW+DNN 參數如下:

Epochs=4, BATCH_SIZE = 128, lr = 0.001

Word Embedding 同 RNN。bow 的部分，在每次 BATCH SIZE 時，改寫 collate function，中文句子轉換成 INDEX 去訓練，減少全部轉換時的記憶體浪費。總單字數為 133768。DNN 部分則是 4 層，參數如圖所示，為了避免 overfit，加上 dropout=0.5 得到較好結果。最後模型訓練正確率:

Private: 0.75510 Public: 0.75150





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。

```
row = re.sub("8+[(+*)]+9", "廢物", row)
#row = re.sub("B+\d+", "", row)
#row = re.sub("b+\d+", "", row)
row = re.sub("ㄅㄔ", "白癡", row)
row = re.sub("ㄐㄎ", "靠夭", row)
row = re.sub("ㄌㄙ", "垃圾", row)
row = re.sub("巛ㄢˋㄋㄧㄣˊ", "幹你娘", row)
row = re.sub("森77", "生氣氣", row)
row = re.sub("80", "霸凌", row)
row = re.sub("87", "白癡", row)
row = re.sub("Hen", "很", row)
row = re.sub("沒關C", "沒關係", row)
row = re.sub("ㄋㄊㄇ", "笨蛋", row)
row = re.sub("ㄉ", "的", row)
row = re.sub("ㄏ", "呵", row)
row = re.sub("ㄅ", "吧", row)
row = re.sub("ㄚ", "啊", row)
row=re.sub(" ", "", row)
```

Data preprocessing: 利用 re.sub()將髒 data 濾除，像是常見的注音文轉換可略為提升準確率，因為ㄅㄔ或是ㄐㄎ不是所有人發文都會這樣打，如果轉成更常見的靠夭或是垃圾，機器更能學習。空格部分也完全刪除，避免原本的空格是人們聊天亂打的，交由 jieba 去處理分割。

Embedding: 利用 CBOW，將 test 和 train data 一起訓練，調參發現 window=5 效果較好，iter 設為 20。而 window 設為 5 較好應該是中文裡面涵蓋主詞+助詞+動詞+受詞和標點符號，符合中文最長句子大概的格式。

Trainging: 由於為了讓長度相同，加上 pad 對訓練結果影響很大，為了避免機器訓練時看到一堆的 pad，因此採用每個 batch 做 pad_sequence，且在丟入 lstm 時做 pack_padded_sequence，結束後做 pad_packed_sequence，確保訓練時 pad 不會被加進去訓練。值得一提的是，由於不少句子長度超過 300，為了方便機器訓練，全部都刪減成長度 200，小於 200 不處理。最後取 hidden 的部分，採用 pooling 的方式，取最大或是平均找較佳的 hidden 層。最後我並未做 ensemble，因為發現要 train 出一個好 model 需要切好的 train 跟 validation，但我每次做 shuffle 結果有高有低，不是所有 model 都很好的情況下，ensemble 反而達不到預期

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。

不做斷詞(EPOCH=2)		有做斷詞(EPOCH=2)	
Public	Private	Public	Private
0.49450	0.50300	0.76390	0.75920

以中文語言來說，因為不像英文字一個字就可以有一個完整的含意，必須做斷詞才能表現出詞語的意思。舉例來說，“像 一樓 那種 就是 窮肥宅 垃圾” 如果不做斷詞:“像 一 樓 那 種 就 是 窮 肥 宅 垃 圾”，每個字分開就會不確定字是跟前面還是後面的字有關，甚至無關，導致機器很難了解整句話，而如果是英文就會有 That kind of guy in 1F is fat nerd and garbage. 每個字都有分開來，且有自己的意思。我們在做 ml 會做 word embedding，就是希望從前後詞句，去判斷可以填入甚麼字詞(cbow)，以此減少參數。而做過斷詞，就能讓機器更方便去判斷詞句，對訓練有較好的結果。(不做斷詞總字數為 4854；後者為 33401)

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。

RNN				BOW			
句子 1		句子 2		句子 1		句子 2	
0	1	0	1	0	1	0	1
0.7777	0.2223	0.3111	0.6889	0.0427	0.9573	0.0427	0.9573

由人進行句子的判斷，句子前者分數屬於 0 後者屬於 1，而 RNN 在判斷上兩者正確，BOW 卻都是 1，是因為 RNN 有考慮前後語句的關係，中文在語意上必須考量先後。因此”說別人白癡之前”跟直接罵別人”白癡”語意不同。但 BOW 只考慮一個句子有甚麼字，兩句都有白癡這個字眼，導致 MODEL 認為兩個句子都相同，結果都是 1。