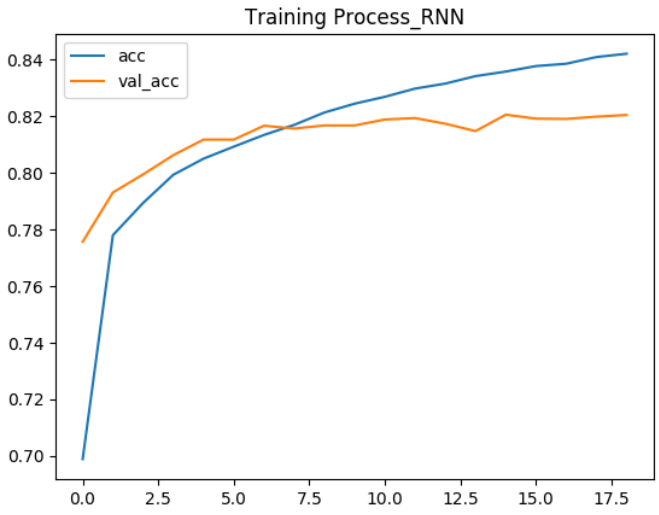
學號：B04901138 系級：電機三 姓名：張景程

1. (1%)請說明你實作的 RNN model，其模型架構、訓練過程和準確率為何？
   (Collaborators: 無)

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 20, 200)	24902000
gru_1 (GRU)	(None, 20, 512)	1095168
batch_normalization_1 (Batch	(None, 20, 512)	2048
gru_2 (GRU)	(None, 20, 256)	590592
batch_normalization_2 (Batch	(None, 20, 256)	1024
gru_3 (GRU)	(None, 128)	147840
batch_normalization_3 (Batch	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33024
batch_normalization_4 (Batch	(None, 256)	1024
dropout_2 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
batch_normalization_5 (Batch	(None, 128)	512
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Total params: 26,806,769
Trainable params: 1,902,209
Non-trainable params: 24,904,560



我使用了 gensim 的 Word2Vec，其參數 size 設為 200，min_count = 10。
也就是將 label 和 nolabel 的句子中，出現超過十次以上的單字才會保留起來做訓練，並將每個單字轉為長度 200 的 vector。
optimizer = Adam
Loss function = binary_crossentropy
max_length = 20
epoch = 50 (with Earlystopping)
最後一層 Dense(1,activation='sigmoid')，其他層 activation function 皆為 relu

accuracy on kaggle :
public : 0.81917 private : 0.81836

2. (1%)請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？
(Collaborators: 無)

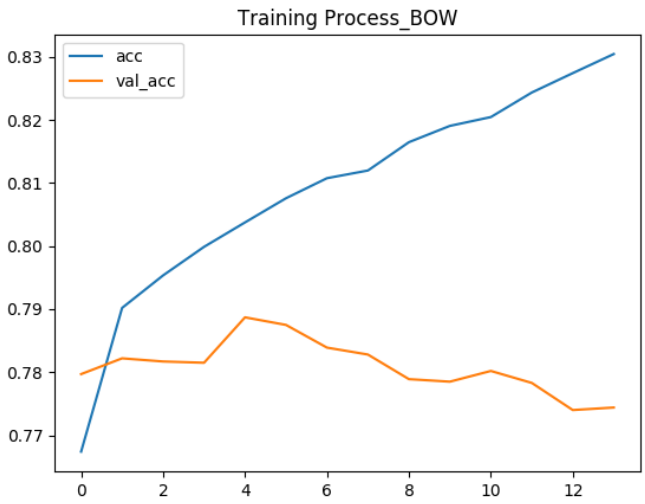
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               3072512
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257
=================================================================
Total params: 3,466,753
Trainable params: 3,466,753
Non-trainable params: 0
_________________________________________________________________
```

我用 keras 裡內建的 tokenizer 中的 Tokenizer.text_to_matrix()來實作 BOW model，其中 tokenizer 的參數 num_words 設為 6000，最後再接四層 DNN，units 分別為 512,512,256,1。

從 training_procedure 的圖來看，和 RNN 比起來，BOW 的準確率較低，大約在 0.78 左右，且大約從第三個 epoch 開始，就開始有 overfit 的現象。

accuracy on kaggle :

public : 0.78965　　private : 0.78927



3. (1%)請比較 bag of word 與 RNN 兩種不同 model 對於”today is a good day, but it is hot”與“today is hot, but it is a good day”這兩句的情緒分數，並討論造成差異的原因。

(Collaborators: 無)

	RNN	BOW
”today is a good day, but it is hot”	0.4354555	0.64216095
“today is hot, but it is a good day”	0.98816609	0.64216095

由 RNN 的觀察結果發現，“today is hot, but it is a good day”這句的分數相當高，是很 positive 的句子，推測是因為後半段”but it is a good day”，”but“這個轉折語氣讓整個句子帶有很正面的語氣；相反的，”today is a good day, but it is hot”，一樣是轉折語氣，但是機器會比較 focus 在”but 後面的部分“，也就是”it is hot”，導致這句話的分數比較低，但不會低得太誇張，因為前半部分的語意仍是正面的。

另一方面，BOW 因為不會考慮句子的前後順序，所以對 machine 來說，這兩個句子是相同的 input，因此會得到相同的分數。

4. (1%)請比較"有無"包含標點符號兩種不同 tokenize 的方式，並討論兩者對準確率的影響。

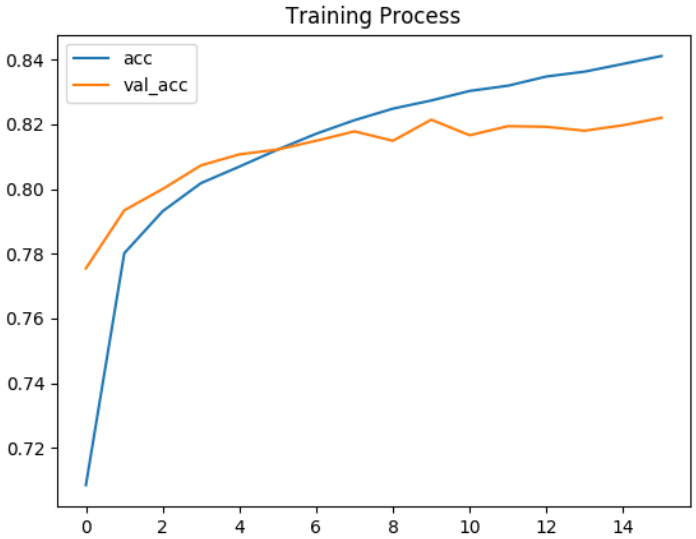
(Collaborators: 無)

沒包含標點符號：所有 model 同第一小題，tokenizer 預設為

filters = '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n'

有包含標點符號：改 tokenizer 內的 filters = (‘ ‘)

有包含標點符號的 accuracy 比較高，推測是因為一些標點符號會使語氣改變或是增強情緒的表現，像是”！“ ,” ? “等等符號



accuracy on kaggle :

public : 0.82386

private : 0.82201

5. (1%)請描述在你的 semi-supervised 方法是如何標記 label，並比較有無 semi-surpervised training 對準確率的影響。

   (Collaborators: 梁書哲 B04901019)

   答：

   我標記 label 的方式是將第一次 predict 出來的結果設 threshold=0.05，也就是取 predict 的值大於 0.95 為 1，小於 0.05 的標記為 0，將這些標記後的 data 加入 training data 中再繼續做 training，總共跑了七次 iteration，training process 如下：

Iteration	label data	non-labeldata	val_acc
0	0	1178614	
1	214294	964320	0.7847
2	464547	714067	0.7949
3	576373	602241	0.7979
4	685771	492843	0.8045
5	755489	423125	0.8100
6	804373	374241	0.8097
7	832540	346074	0.8115

accuracy on kaggle :

public : 0.81262 private : 0.81052

做 semi-supervised training 在 kaggle 上的準確率反而下降，但下降幅度並不大，大約 0.7%左右。