Machine Learning HW6 Report

學號：B04505028 系級：工海四 姓名：林秀銓

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*

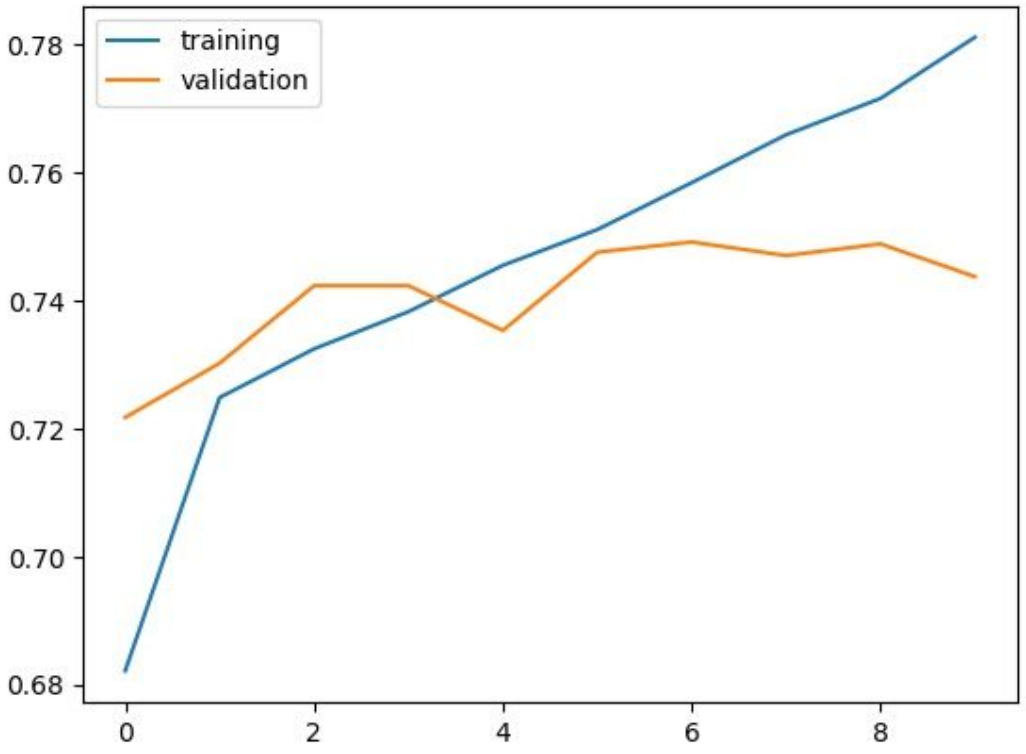
model:

```
def build_model():
    model = Sequential()
    model.add(Bidirectional(LSTM(256, recurrent_dropout=0.2, dropout=0.2, return_sequences=True), input_shape=(40, 250)))
    model.add(Bidirectional(LSTM(256, recurrent_dropout=0.2, dropout=0.2, return_sequences=True), input_shape=(40, 250)))
    # model.add(Bidirectional(LSTM(256, recurrent_dropout=0.2, dropout=0.2, return_sequences=True), input_shape=(40, 250)))
    # model.add(Bidirectional(LSTM(512, recurrent_dropout=0.2, dropout=0.2, return_sequences=True), input_shape=(40, 250)))
    model.add(Bidirectional(LSTM(256, recurrent_dropout=0.2, dropout=0.2, return_sequences=False), input_shape=(40, 250)))
    model.add(BatchNormalization())

    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(32, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))

    model.add(Dense(2, activation='softmax'))
    model.summary()
    return model
```

word embedding 的方法是先將train_x.csv和test_x.csv中的data用jieba分割好，將他分一起丟進gensim套件中的Word2Vec訓練，然後再用這個訓練好的model將training data轉換成vector。

模型正確率：
public : 0.75280, private : 0.74460

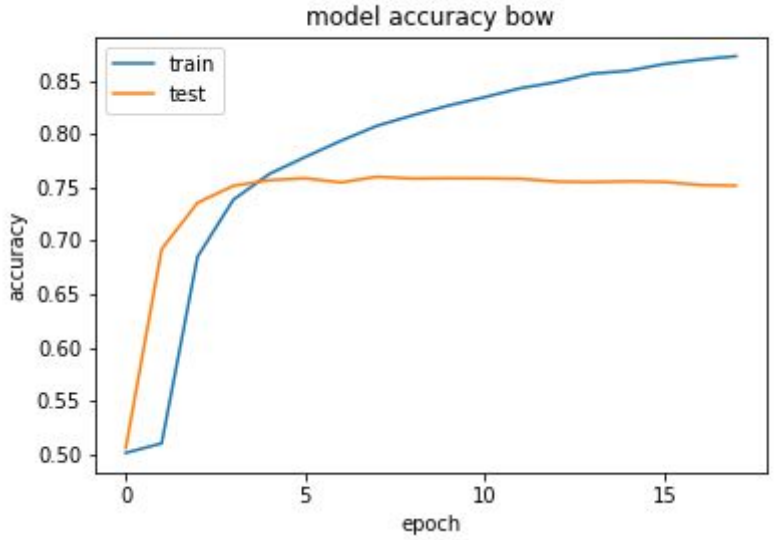
訓練曲線：



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*。
   正確率：
   public: 0.75250, private:0.74970

   先將training data和testing data利用jieba斷詞，再建立一個list儲存所有出現過的詞(不重複)，再利用這個list將training data轉換成vector，vector即為每個詞出現的次數，然後在丟進DNN的model裡訓練。

```
model = Sequential()
model.add(Dense(256, activation = 'relu', input_shape=(len(d))))
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(16, activation='relu'))
model.add(Dropout(0.5))


model.add(Dense(2))
model.add(Activation('softmax'))

model.summary()
```



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。

我improve performance的方法是用7個model去做ensemble，將training data分成7份，每個model都用不同的一份做validation data，剩下的拿去train，做後在將結果做voting。ensemble可以使模型進步是因為每一個 model 獨自拿出來看 variance 都很大，但是把不同的 variance 很大的 model 集合起來以後，他

的 variance 就不會這麼大，且他的 bias 會是小的。

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：private : 0.72720, public : 0.72820
   有做斷詞：private : 0.74170, public : 0.74740
   使用同樣的模型即訓練次數下，有做斷詞的情況下的預測結果比沒有做斷詞的情況要好一點，可能是因為一個詞的意思和每個字分開的意思會有所不同，所以若以字為單位，會無法精確表達語句的意思，造成結果略差於有做斷詞的情況。

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:
   "在說別人白痴之前，先想想自己" : [0.5453909, 0.4546091]，判斷結果為非惡意留言
   "在說別人之前先想想自己，白痴" : [0.43762892, 0.56237113]，判斷結果為惡意留言

   BOW: 兩句output皆為[0.31005, 0.68995]，判斷結果為惡意留言

   因為BOW只考慮句子中有哪些詞，並不考慮其排列情形，所以經過斷詞後，兩句所包含的詞都相同且數量一樣，造成兩句話有一樣的結果，而RNN則會考慮每個詞的排列情形，所以會有不同的結果。