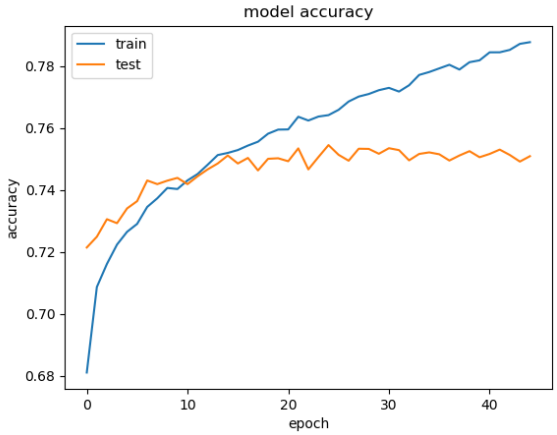
HW6 Report

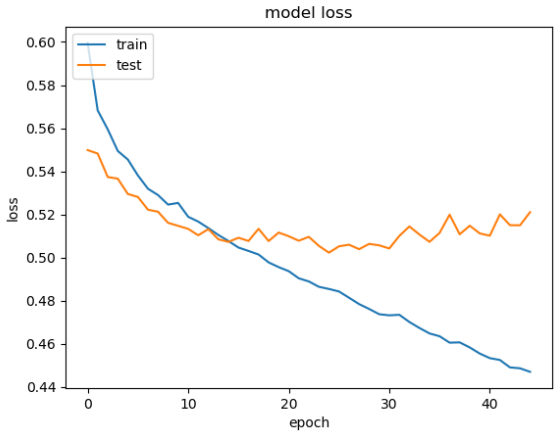
學號：B06901020 系級：電機二 姓名：張恆瑞

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.add(Embedding(input_dim=embedding_matrix.shape[0],
                    output_dim=embedding_matrix.shape[1],
                    weights=[embedding_matrix],
                    trainable=False))
model.add(LSTM(128,activation="tanh",dropout=0.3,return_sequences = True,
        kernel_initializer='Orthogonal'))
model.add(Bidirectional(LSTM(128,activation="tanh",dropout=0.3,return_sequences= False,
        kernel_initializer='Orthogonal')))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

我使用了 Jieba 來做斷詞，並利用 Gensim 的 Word2Vec model 來做 word embedding (每個 vector 為 128 維)，而且在訓練整個 RNN model 時固定 word embedding model。





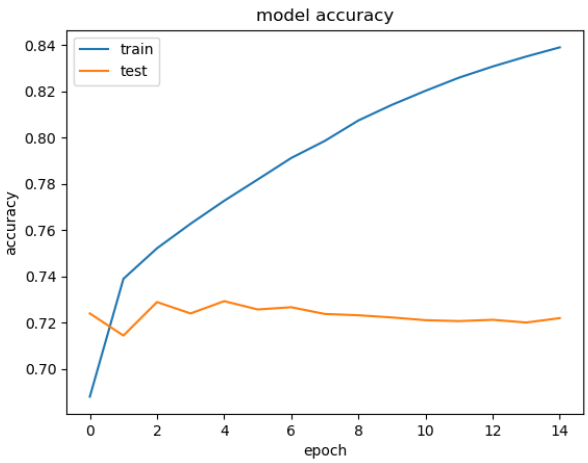
因為單一模型的準確率達不到 75.5%，因此使用了 ensemble 技巧把 5 個不同初始化的模型綜合在一起用(其中一個是較簡化版的模型，並且將 LSTM 改為 GRU)。

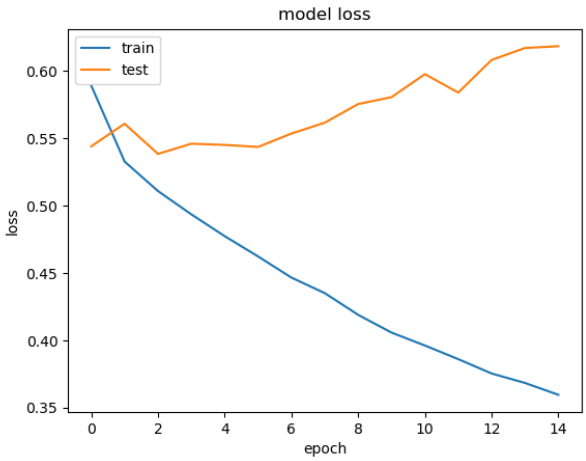
	Public	Private
Accuracy	0.75800	0.75490

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*。

```
model.add(Dense(64, activation='relu',
                input_shape=(v_size, )))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

同樣我也使用 Jieba 來做斷詞，我將 BOW 的向量設為 7375 維(只考慮出現過 33 次以上的詞)，因為這是我電腦跑得動的上限。





我怎樣調參數 model 都還是長得爛爛的，而且很容易 overfit training set。

	Public	Private
Accuracy	0.73300	0.73550

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。

一開始訓練時準確率始終上不到 70%，因此我只保留中英文字、注音、數字、常用標點符號及空白，同時將英文字母都轉成小寫，這樣可以把特殊的外文符號以及表情符號過濾掉，而這些往往不帶有什麼資訊或是會混淆視聽。再來，我還把句子中的 b0~b500 都去除掉，因為這些也沒有什麼特別的意義，只是用在 tag 某樓的時候用到。

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。

單一模型(非 ensemble)	Public	Private
Word level model	0.75350	0.75270
Character level model	0.73310	0.73460

很明顯做了斷詞後效果會好很多，因為很多時候在判斷語意時是要看「詞 (word)」而不只是看單一的「字 (character)」，例如：光看「低」和「能」以及看「低能」語意就完全不一樣。

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
在說別人白痴之前，先想想自己	0.392344	0.465604
在說別人之前先想想自己，白痴	0.474401	0.465604

這兩句話的主要差異在於「白痴」的位置顛倒了，用 RNN 的時候，可以看出「白痴」在句子後面時 RNN 會「記得比較清楚」，因此分數會比較高；反之，距離句子結尾愈遠的「白痴」影響力就愈小。BOW 則是因為 word vector 只記錄每個詞出現的次數，不論詞語在句子中的位置，所以這兩個詞語換位的句子會得到相同的分數。