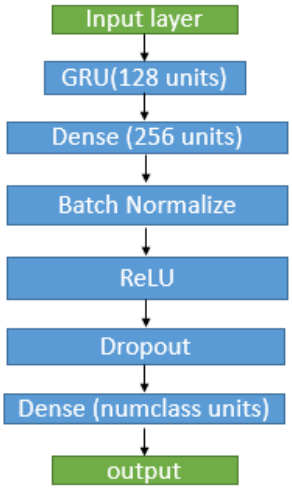
Machine Learning HW6 Report

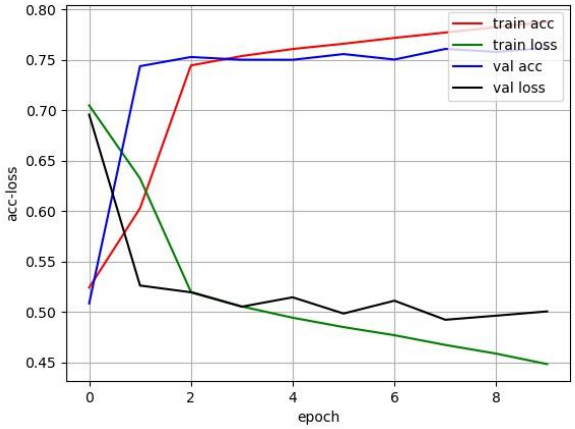
學號：r07942092　系級：電信碩一　　姓名：白佳灝

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*

   acc : 0.75650

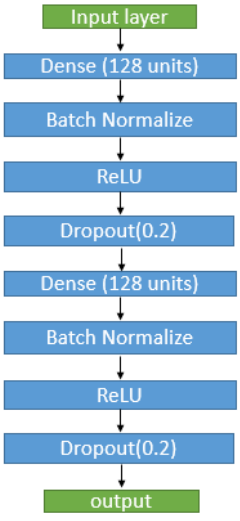
   RNN 模型如下圖，實作 word embedding 的方法為，先將每行文字經濾波器濾掉奇怪的字元 (filter: '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n。 \\/，、嗎阿ㄇㄅ')，再用 jieba 切成詞彙，再利用 word2vector 將每個詞彙以 200 dim 的 vector 去 encode，但詞彙須超過出現三次再作編碼，最後送進 RNN 裡面，並存取每次 epoch 訓練的 model。最後會人工選擇 val loss 最低的那個送進 kaggle。

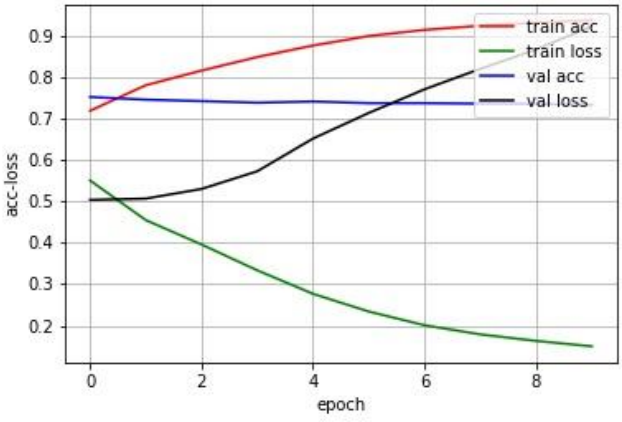




2. (1%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正確率並繪出訓練曲線*。

   acc:0.75000





前處理的部分跟 RNN 一樣，也會先濾掉奇怪的字元，再用 jieba 切字，並利用 tokenizer 協助我們將文字編碼成 BOW 之形式，送進 DNN 時，也會記錄每個 epoch 的 val loss，人工選擇 val loss 最低的那個送進 kaggle。

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。

(preprocess)
如果再將文句前半充滿了許多符號，而真正的文意在後半，如果 model 只看 40 個字，那它就都只看到符號而已，所以加上 filter 可以協助把不一些不相關的東西濾掉，結果相對也比較好。
(embedding)
一開始實作 model 時，只考慮將每個詞彙以 200 dim encode，且固定每次只讓 rnn 吃 40 的詞彙，不足 40 字的則補 0，那時的準確率大概是 75.4%，而改良版為將詞彙以 300 dim encode，並且提升 rnn 一次看的詞彙至 80 個，acc 提升至 75.8%，我認為會進步的原因是，給 rnn 看的資訊變多了，例如:有些較長的評論可能它的文章重點在後半段，如果使用開始的 model 就無法參考後半段的評論作決定是否為惡意留言。

(架構)
原本採用 DNN 的架構，但是我發現，它學習的效果並不會因為 epoch 的提升而變好，從 p2 的 val loss 可以看到，它是呈現一個發散的結果，而 RNN 的 val loss 則是收斂的，因此採用 RNN 可以讓模型進步。

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。

不做斷詞的結果會比較差(大約 71%)，原因可能是因為切到某些關鍵詞，使得沒辦法讓 RNN 學到這個詞是又惡意的，而是只看到單一的字，但其實有時可能斷字就能學到是惡意語句，不過斷詞能讓 RNN 對語句有更完整的了解，所以斷詞的準確度比較高。

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"	"在說別人之前先想想自己，白痴"
RNN	[0.65579015 0.34420988]	[0.5638017 0.43619826]
BOW	[0.4340943 0.5659057]	[0.4340943 0.5659057]

如果以人的角度來看的話，第一句話應該是沒有戴著惡意評論的，但第二句話卻有。首先我們看 RNN 的結果，它認為這兩句話都沒有惡意，但是相對的它給第二句話為惡意的機率卻提高了(認為第二句可能是惡意)。而 BOW 它給的惡意何不是惡意的分數都是一樣的(都認為是惡意)，RNN 判的比 BOW 好的原因是因為，他考慮了字詞前後的相關序列，但是 BOW 是以字出現的頻率來判斷，所以在這邊表現得比較不好。