學號：R05922130 系級： 資工所 姓名：王瀚磊

1. **(1%)請問 softmax 適不適合作為本次作業的 output layer? 寫出你最後選擇的 output layer 並說明理由。**

Softmax 如下所示，雖然將 output 都壓縮到 0 與 1 之間，但其多了一個 constrain: 全部 output 加總為 1。在本題的 multi-class classification 中，有可能有超過一個以上的 label 機率很高（同一筆資料有多個 label），因此兩 softmax 相衝突。最後將 softmax 換成 sigmoid 後，可同時保留其壓縮至 0 與 1 之間並捨棄加總為 1 之 constrain，在訓練上表現也進步許多。

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Fig 1. Softmax Function

$$S(x) = \frac{1}{1 + e^{-x}}.$$

Fig 2. Sigmoid Function

2. **(1%)請設計實驗驗證上述推論。**

在同樣的 model structure 下，實際跑了 Sigmoid Function 以及 Softmax Function 得到結果如下表所示，從 validation f1 score 上表現來看可以發現 Sigmoid Fuunction 效果明顯較好。

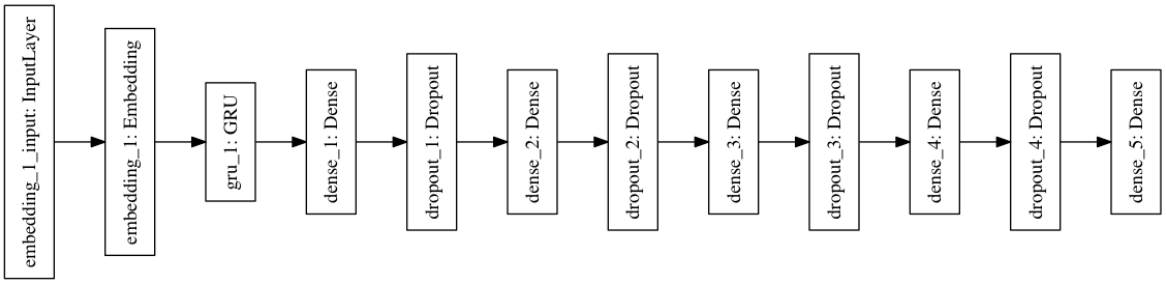

Fig 3. Model Structure

	Sigmoid	Softmax
Val_f1_score	0.502 (early stop: 64)	0.256 (early stop: 53)

Table 1. Comparison between different Activation Function

3. (1%)請試著分析 tags 的分布情況(數量)。

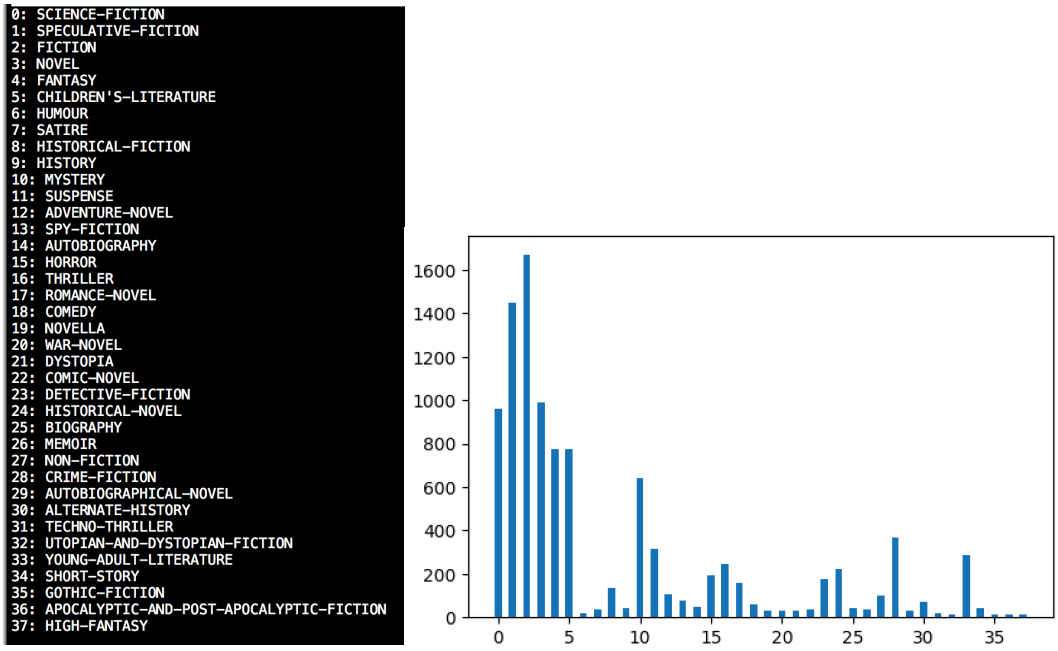

Fig 4. The distribution of training data

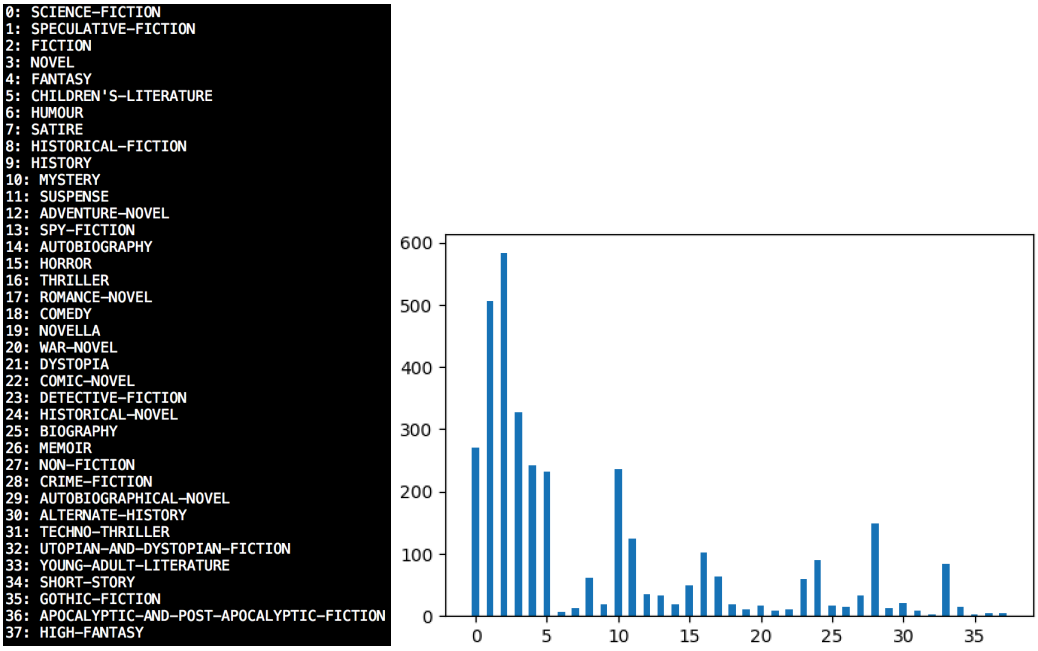

Fig 5. The distribution of predict data

我原本畫完圖以為我輸出錯東西，怎麼兩張圖走向這麼接近，後來檢查後發現真的是 training data 以及 predict data 的分佈竟然如出一徹，都是 FICTION 以及 SPECULATIVE-FICTION 最多，原來在 RNN（GRU,LSTM）底下對於 training data 的分佈是如此的 dependent，因此我認為要是 training data 能夠搜集夠多的每種 tag，最後 train 出來的效果一定不錯。我這次能夠過 simple baseline 也是因為盡量加大 training set，犧牲 validation set 後勉強過的，我想

在 data 較少的 tag 預測結果一定錯的亂七八糟，也難怪這次大家的正確率都在 0.5 左右，因為只有一半的 data 是分佈在 training data 夠多的 tag 上。

4. **(1%)本次作業中使用何種方式得到 word embedding?請簡單描述做法。**

我使用的是助教提供的 sample code，首先使用 keras.preprocessing 中的 Tokenizer 記錄 training data 以及 testing data 中的每個 word，接著參考網上下載的 glove.6B vector 即可得到一組 weight 能夠將 data 中的 word 轉換成一個 vector，接著將這組 weight 丟進 model 中第一層的 embedding layer，即可在每次 train 的時候將 input 轉換成 vector 在丟進 GRU 裡 train。

5. **(1%)試比較 bag of word 和 RNN 何者在本次作業中效果較好。**

在實作之前時常耳聞其他同學說 bag of word 很好過 simple baseline，然而在我實作之後發現 bag of word 很容易在 training set 上 overfitting，而 val_f1_score 仍然停留在 0.45 左右，無法輕易過 baseline，之後我嘗試將 Tokenizer 中加入些 constrain：只考慮出現頻率最高的 25000 個 word，val_f1_score 只有上升到 0.48，同樣無法過 baseline。而 RNN 而言由於 Tokenizer 每次的結果都不一樣，而且在切 validation set 時候又有 random，考量到 early stop 以及 checkpoint 都是 depend on validation set，因此每次 train 玩的結果起伏很大。同樣的 model structure 以及參數，train 了不同次其 Kaggle 結果從 0.498~0.46 都有，因此也不是很好 train。然而就結果而言 RNN 較難 overfitting（在 training set 中都沒辦法 train 好ＸＤ），且最後的 val_f1_score 表現也稍微較好，因此我認為這次作業 RNN 稍微優於 bag of word。

	Bag of word	RNN
val_f1_score	0.486 (early stop: 39)	0.502 (early stop: 64)

Table 2. Comparison between Bag of Word and RNN