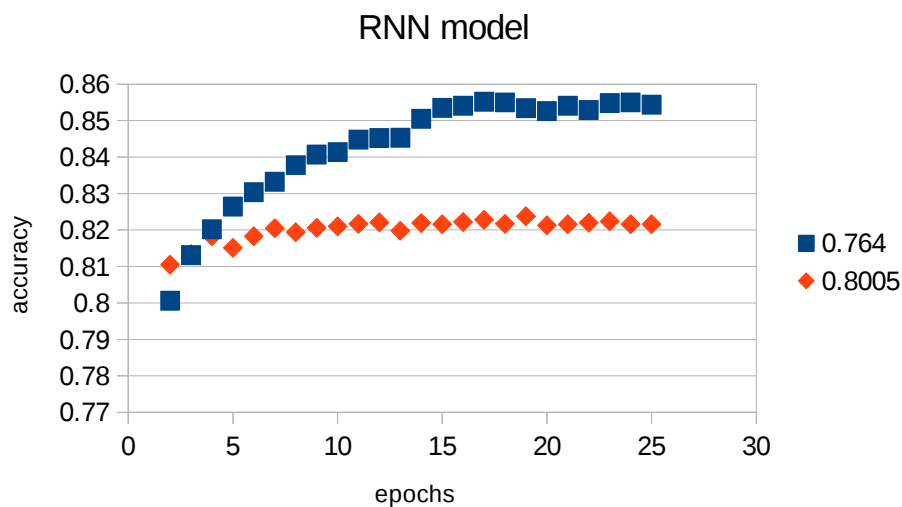


P1: 請說明你實作的 RNN model，其模型架構、訓練過程和準確率為何？ (1%)

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 32, 128)	949120
gru_1 (GRU)	(None, 32, 260)	303420
gru_2 (GRU)	(None, 32, 260)	406380
gru_3 (GRU)	(None, 32, 250)	383250
gru_4 (GRU)	(None, 250)	375750
dense_1 (Dense)	(None, 128)	32128
dense_2 (Dense)	(None, 1)	129
Total params: 2,450,177		
Trainable params: 2,450,177		
Non-trainable params: 0		
Train on 160000 samples, validate on 40000 samples		

(這個模型沒有做 semi-supervised learning，所以分數比 best 低，此處只說明對 label data 的設計)
RNN 如圖設計，embedding 之後過四層 GRU，再過一層 dense，每一層都用 selu 做 activation，並因此把 kernel 用 lecun_normal 初始化符合其理論假設，最後用 sigmoid 輸出
訓練過程：

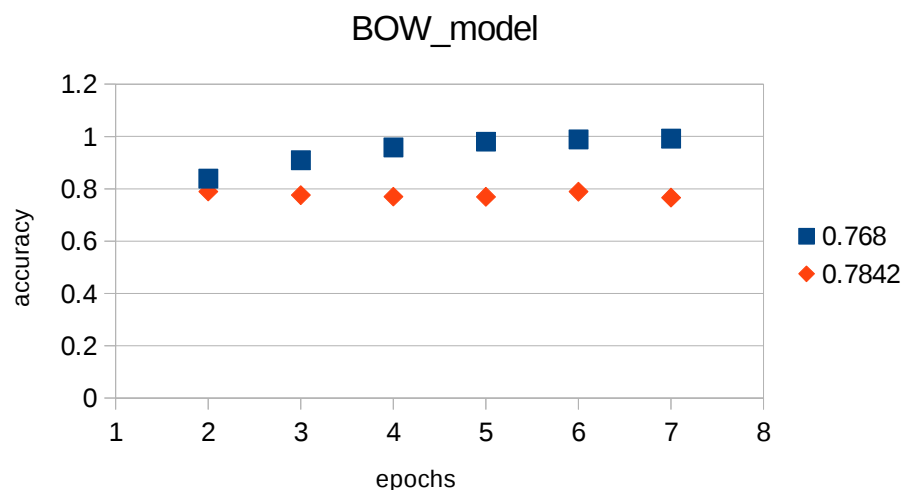


(深藍色為 training set 上，橘色為 validation set)
optimizer 用 Adamax，loss function 用 binary_category，epoch 在 25 因為 validation 無上升停止，validation 做到 82% 左右，實際在 public 分數為 0.82773，private 分數為 0.82665

P2: 請說明你實作的 BOW model，其模型架構、訓練過程和準確率為何？ (1%)

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	4096512
dense_2 (Dense)	(None, 512)	262656
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 512)	262656
dropout_2 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 512)	262656
dropout_3 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 1)	513
Total params: 4,884,993		
Trainable params: 4,884,993		
Non-trainable params: 0		

BOW 如圖設計，每一層用 relu(selu 分數差不多但是比較慢)
訓練過程：



optimizer 用 Adamax, loss function 用 binary_category, epoch 為 5, validation 做到 77%, 實際在 testing set 上差不多, public 分數為 0.77547, private 分數為 0.77618

P3: 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。 (1%)

	RNN	BOW
today is a good day, but it is hot	0.21903421	0.99943191
today is hot, but it is a good day	0.96815902	0.99943191

主觀看來第 2 句會判斷為 1, 第 1 句會判斷為 0, 在 RNN 的判斷上可以看出差別, 但對 BOW 而言兩者是一樣的, 所以同樣做出了正面的判斷

P4: 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。(1%)

這裡直接用 `tokenizer` 的 `filter` 去改且沒有用 `semi-supervised`，所以 `accuracy` 會比用 `Word2vec` 低

	public	private
filter 包含標點符號	0.79955	0.79725
filter 略過標點符號	0.80407	0.79982

略過標點符號的表現稍微好，但是看 `private` 發現差距並不是穩定明顯，主觀來說覺得大部分的標點符號對語氣影響不大，尤其是逗號句號一類，所以認為這個結果還算合理，實作上 0.8 左右大概也是 `tokenizer` 的

P5: 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。(1%)

每結束大約 5 個 `epoch` 的 `training` 之後，取 `unlabel data` 中 `shuffle` 過其中 200000 筆，`treshold` 取 0.1 以下和 0.9 以上加入 `training`，考慮的是 `training` 速度和準確度不要差太多，速度的關係我大概取到 500000 筆左右在 `train`

	public	private
無 <code>semi-supervised training</code> :	0.82234	0.82269
有 <code>semi-supervised training</code> :	0.82750	0.82721

加入 `semi-supervised training` 效果可以看出穩定上升 0.5% 以上，尤其在 `ensemble` 的時候放愈多 `semi-supervised` 的進來效果明顯好