學號：B04902089 系級： 資工三 姓名：林政豪

1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.

   以下各題若無著名 laten dimension 皆為 20。

   normalize:

   Training 是使用減掉平均除以標準差:$z = \frac{x-\mu}{\sigma}$，且 rmse 乘上標準差。

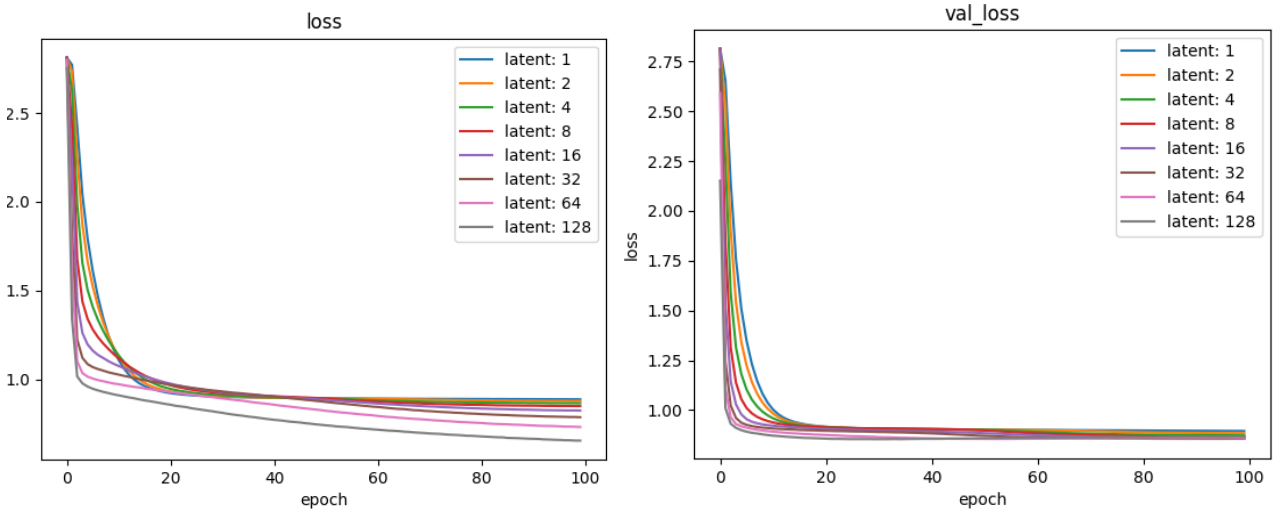
   Predict: 乘上標準差加上平均:$x = (z * \sigma) + \mu$再輸出到 ans.csv。

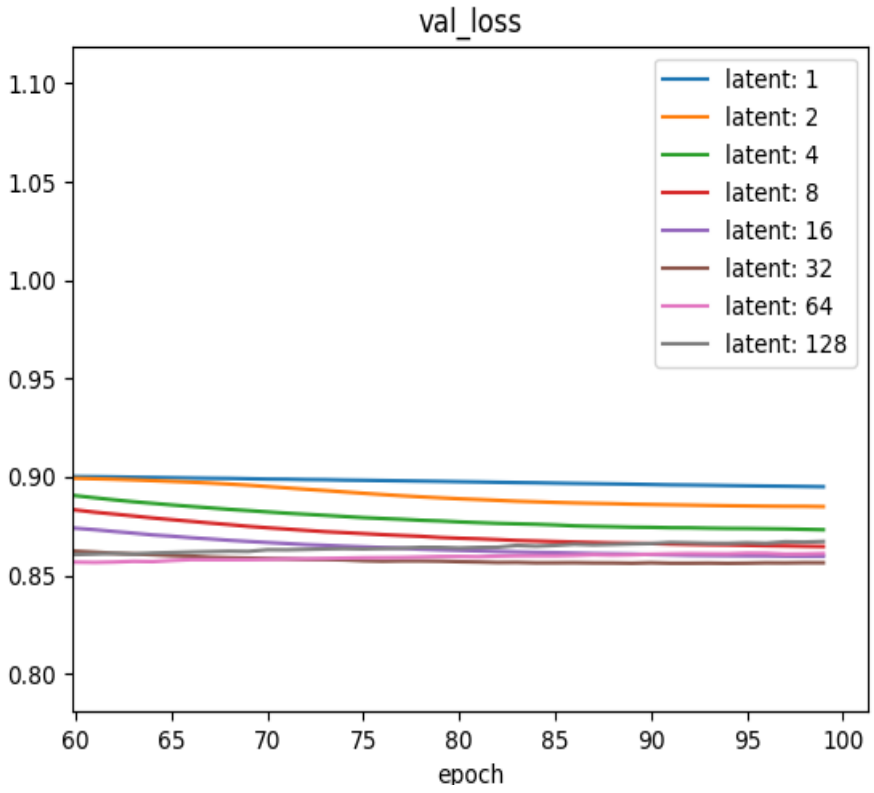
	Normalize	Not Normalized
Training loss	0.8144	0.8170
Validation loss	0.8595	0.8606
Kaggle public score	0.8556	0.8619

   由上表可見 Normalize 會使結果變好。

2. (1%)比較不同的 latent dimension 的結果。

   此題我分別使用 latent dimension = 1,2,4,8,16,32,64,128。





(左圖為 validation 放大圖)

從 loss 的圖來看 latent dimension = 128 時 loss 最小，推測是因為參數最多，最能 fit training data，但從 validation loss 來看 latent dimension 在 latent dimension = 32 時有最小值，推測原本資料的維度與他最相近，其他維度不是會 overfit 就是參數不足無法 fit，才會有如此結果。

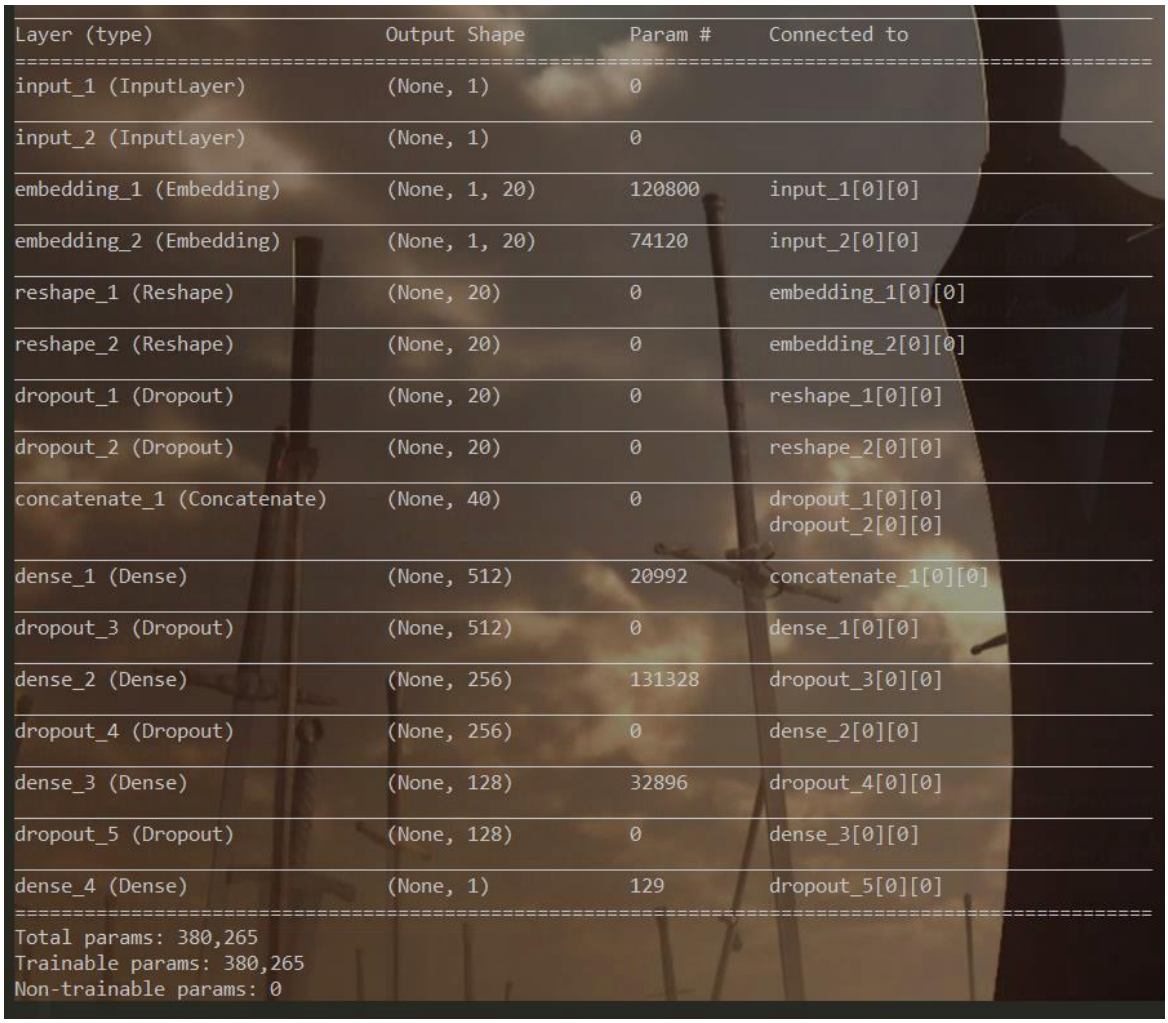
3. (1%)比較有無 bias 的結果。

	No bias	With bias
Training loss	0.8176	0.8170
Validation loss	0.8606	0.8606
Kaggle public score	0.8614	0.8619

兩者之間沒有明顯差異，推測幾乎沒有 bias 的需求。

4. (1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。

DNN embedding 的 drop rate 為 0.1，dense 之間的 drop date 為 0.2。



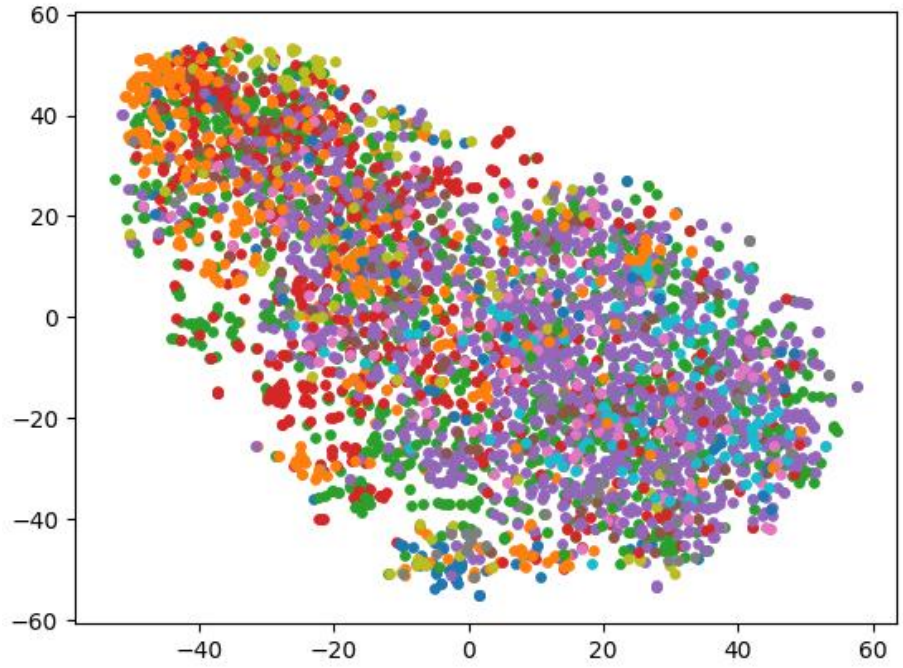
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1)	0	
input_2 (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 20)	120800	input_1[0][0]
embedding_2 (Embedding)	(None, 1, 20)	74120	input_2[0][0]
reshape_1 (Reshape)	(None, 20)	0	embedding_1[0][0]
reshape_2 (Reshape)	(None, 20)	0	embedding_2[0][0]
dropout_1 (Dropout)	(None, 20)	0	reshape_1[0][0]
dropout_2 (Dropout)	(None, 20)	0	reshape_2[0][0]
concatenate_1 (Concatenate)	(None, 40)	0	dropout_1[0][0] dropout_2[0][0]
dense_1 (Dense)	(None, 512)	20992	concatenate_1[0][0]
dropout_3 (Dropout)	(None, 512)	0	dense_1[0][0]
dense_2 (Dense)	(None, 256)	131328	dropout_3[0][0]
dropout_4 (Dropout)	(None, 256)	0	dense_2[0][0]
dense_3 (Dense)	(None, 128)	32896	dropout_4[0][0]
dropout_5 (Dropout)	(None, 128)	0	dense_3[0][0]
dense_4 (Dense)	(None, 1)	129	dropout_5[0][0]

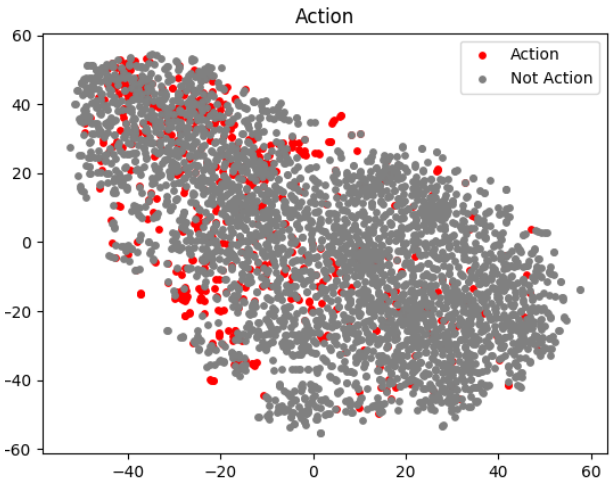
Total params: 380,265
Trainable params: 380,265
Non-trainable params: 0

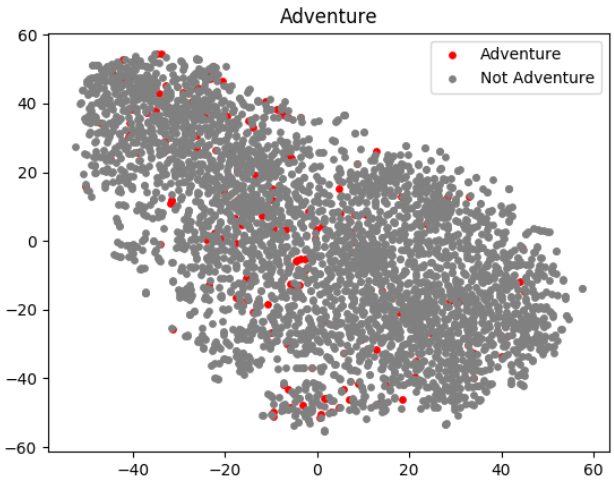
	DNN	MF
Training loss	0.8850	0.8170
Validation loss	0.8911	0.8606
Kaggle public score	0.8945	0.8619

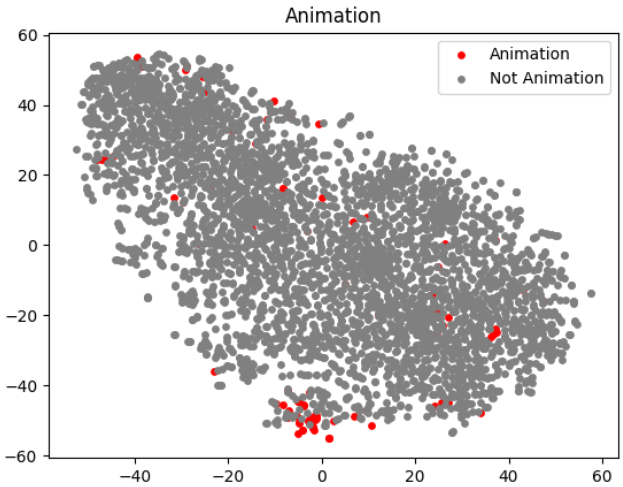
在參數量相差不大時，DNN 和 MF 相比遜色許多，且 overfit 的速度比 MF 快上不少，不到五個 epoch validation loss 就超越 loss，十個 epoch 之後 validation loss 就會開始變高。

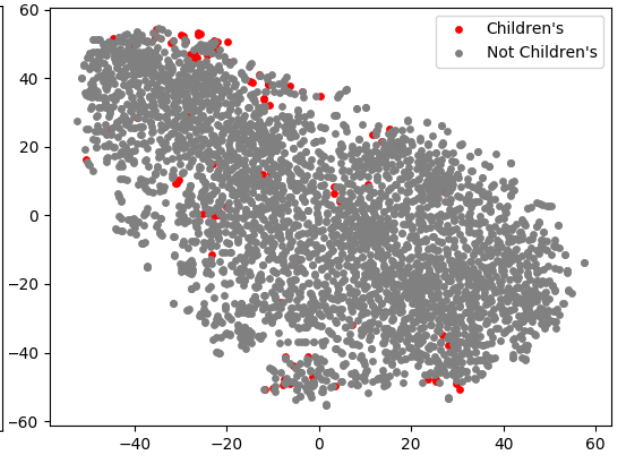
5. (1%)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。由於原圖不易觀看，接下來我使用二分法將每種電影的分布分別畫出。
(collaborator:b04902021 陳弘梵)

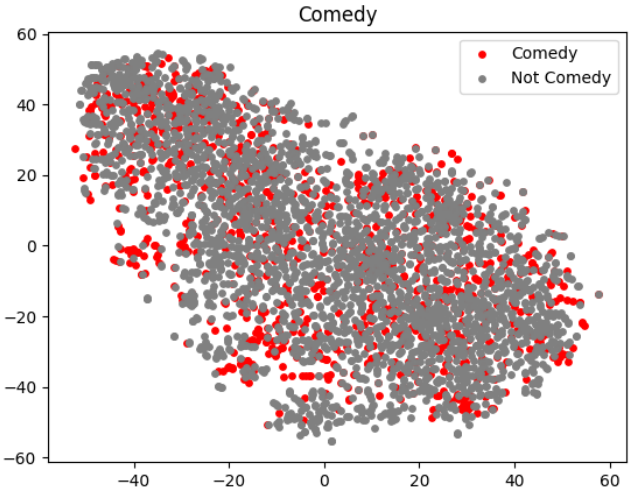


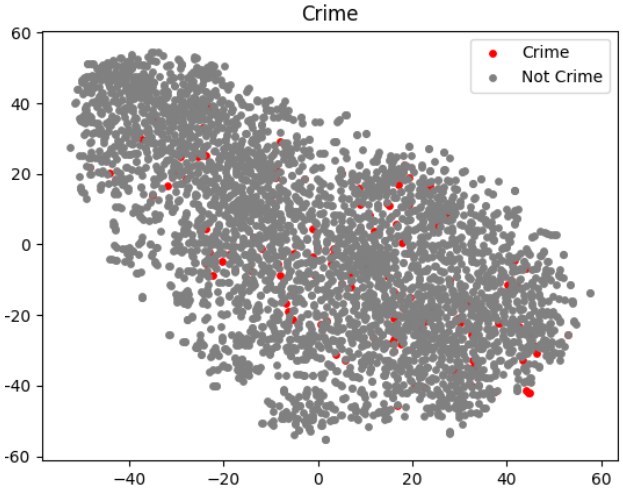


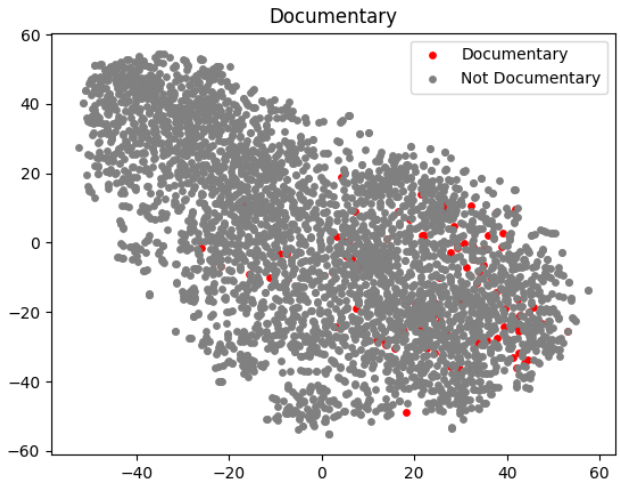


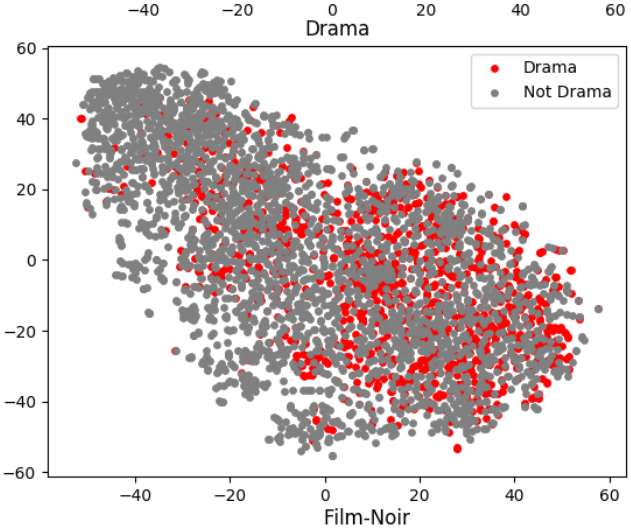


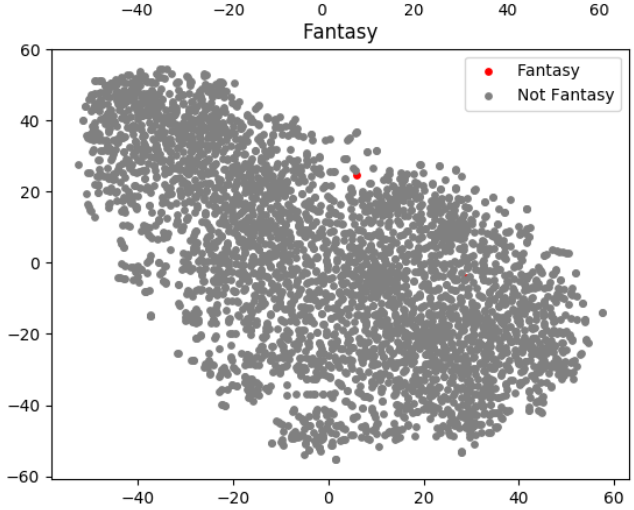


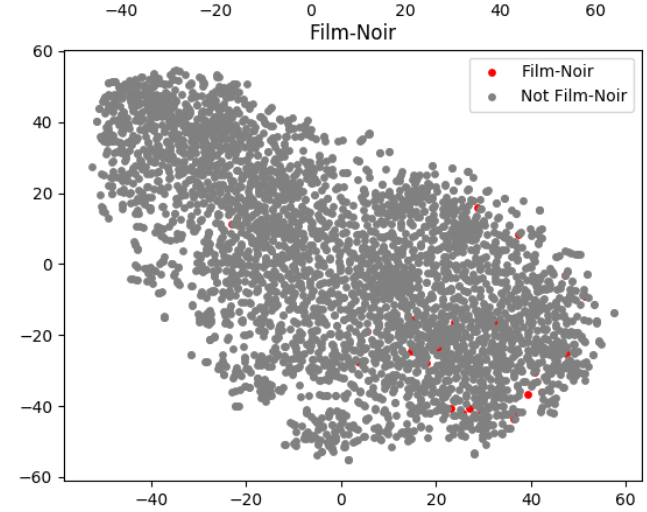

Comedy
Comedy
Not Comedy

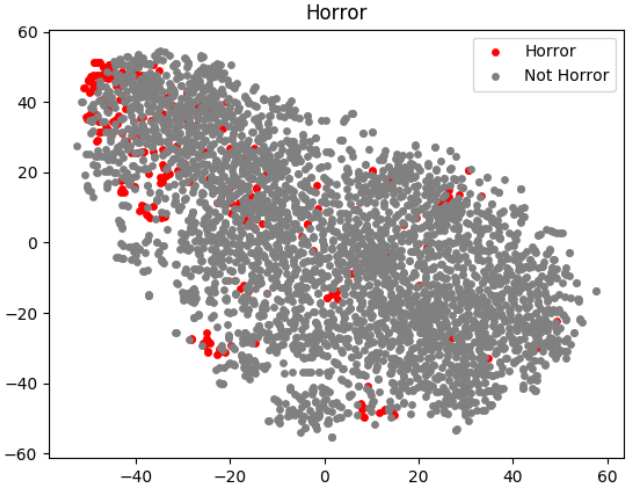

Crime
Crime
Not Crime

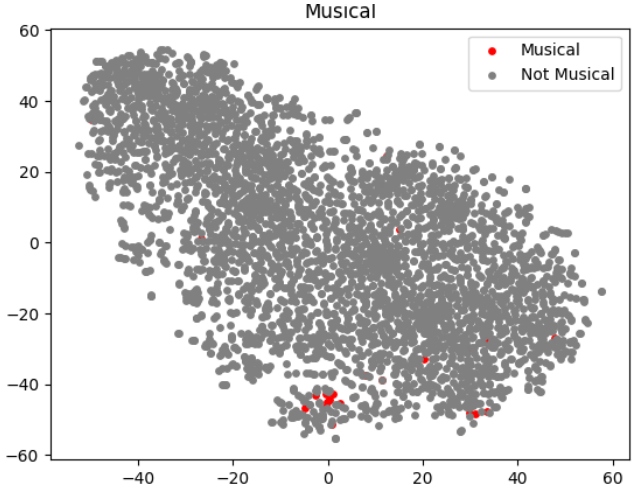

Documentary
Documentary
Not Documentary

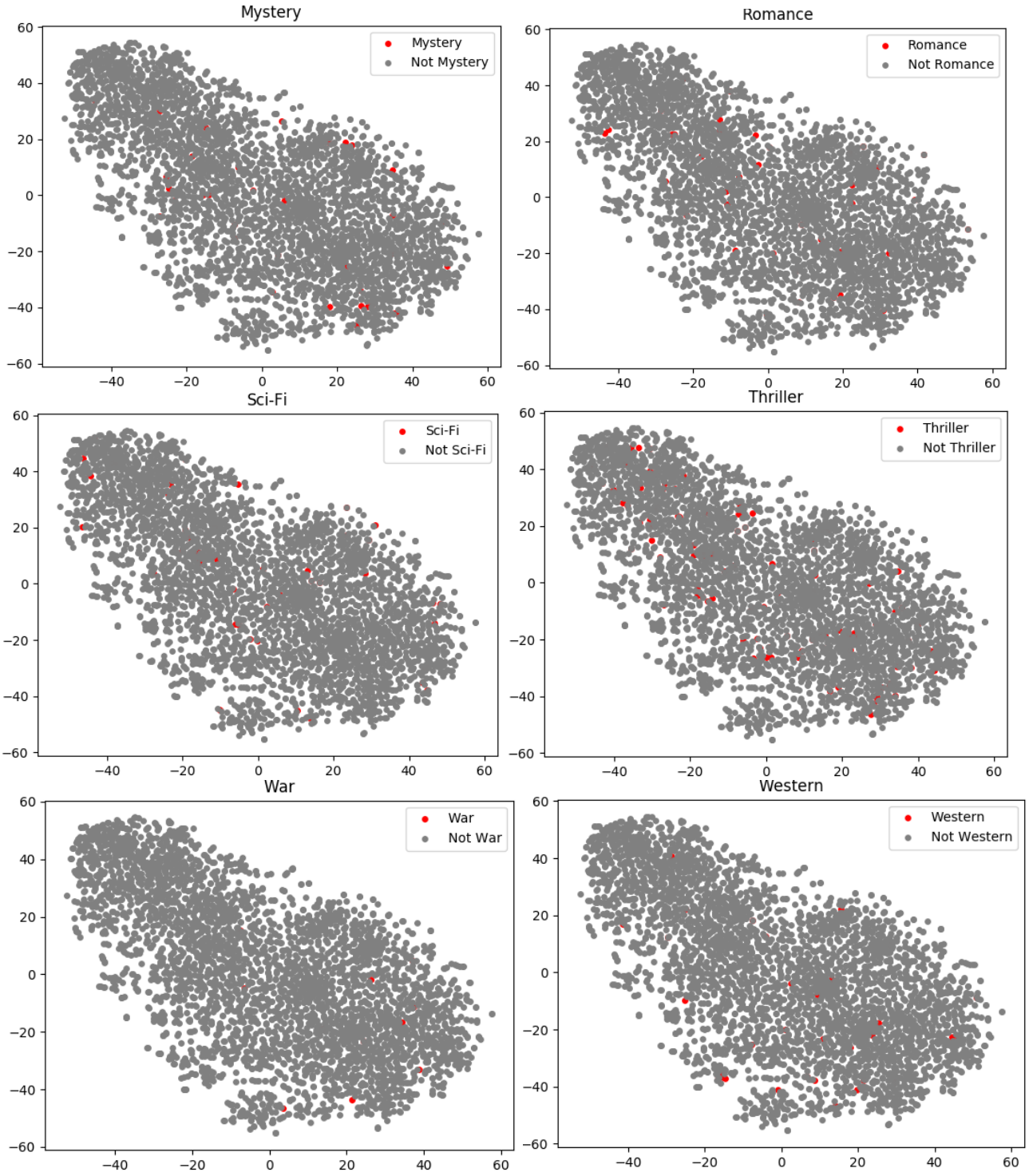

Drama
Drama
Not Drama


Fantasy
Fantasy
Not Fantasy


Film-Noir
Film-Noir
Not Film-Noir


Horror
Horror
Not Horror


Musical
Musical
Not Musical


Mystery
Mystery
Not Mystery
Romance
Romance
Not Romance
Sci-Fi
Sci-Fi
Not Sci-Fi
Thriller
Thriller
Not Thriller
War
War
Not War
Western
Western
Not Western

6. (BONUS)(1%)試著使用除了 rating 以外的 feature, 並說明你的作法和結果，結果好壞不會影響評分。

(collaborator:b04902021 陳弘梵)

我使用 movies.csv 和 users.csv 的資訊，training input 為 899873 筆的 UserId 和 MovieId，在 UserId 後加上性別(F→0,M→1)、年紀、職業， 在 MovieId 後加上電影的類別(以 len = 18 的 list 表示)，DNN layers 各項參數與第 4 題 DNN 相同。

在與陳弘梵同學討論後發現 users.csv 有不少奇怪的資訊，例如一歲的 user 就有超過兩萬筆，所以嘗試把 user 資訊排除，只使用 movie 資訊，效果有好一些些，但不明顯。

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 4)	0	
input_2 (InputLayer)	(None, 19)	0	
embedding_1 (Embedding)	(None, 4, 20)	120800	input_1[0][0]
embedding_2 (Embedding)	(None, 19, 20)	74120	input_2[0][0]
reshape_1 (Reshape)	(None, 80)	0	embedding_1[0][0]
reshape_2 (Reshape)	(None, 380)	0	embedding_2[0][0]
dropout_1 (Dropout)	(None, 80)	0	reshape_1[0][0]
dropout_2 (Dropout)	(None, 380)	0	reshape_2[0][0]
concatenate_1 (Concatenate)	(None, 460)	0	dropout_1[0][0] dropout_2[0][0]
dense_1 (Dense)	(None, 512)	236032	concatenate_1[0][0]
dropout_3 (Dropout)	(None, 512)	0	dense_1[0][0]
dense_2 (Dense)	(None, 256)	131328	dropout_3[0][0]
dropout_4 (Dropout)	(None, 256)	0	dense_2[0][0]
dense_3 (Dense)	(None, 128)	32896	dropout_4[0][0]
dropout_5 (Dropout)	(None, 128)	0	dense_3[0][0]
dense_4 (Dense)	(None, 64)	8256	dropout_5[0][0]
dropout_6 (Dropout)	(None, 64)	0	dense_4[0][0]
dense_5 (Dense)	(None, 1)	65	dropout_6[0][0]

Total params: 603,497
Trainable params: 603,497

	Bonus without user data	Bonus	MF
Training Loss	0.8700	0.8716	0.8170
Validation Loss	0.8750	0.8761	0.8606
Kaggle public score	0.8770	0.8792	0.8619