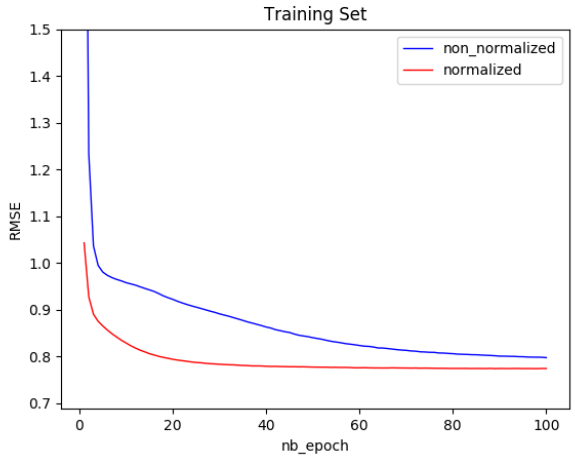
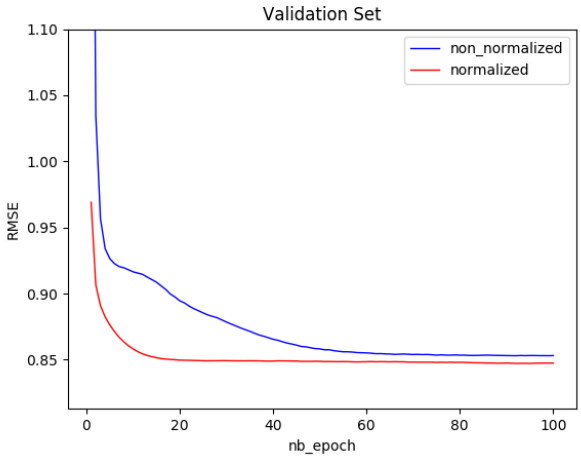
1. **(1%) 請比較有無 normalize (rating) 的差別。並說明如何 normalize.**

Model 為作業指定的 MF，latent dimension 大小選擇 16，有使用 embedding_regularizer 與 dropout，且有將 UserID, MovieID 分別 embedding 成單一數字來作為 User 和 Movie 的 Bias。訓練時，batch size 設為 1024，並使用訓練資料的 10% 作為 validation 之用。

在 Normalize 的實作上，我直接在原有 Model 的最後面用兩個 Lambda Layer 將原先的 output 乘上 Rating 的標準差，再加上其平均數，即可將原本的訓練目標轉換為標準化後的 Rating。

由下圖可以看出，不論是在 training 還是 validation 上，有 Normalize 在訓練一開始的 RMSE 就已經比較小了，且在收斂速度上也是有 Normalize 的較快，最後結果在 validation 上有 Normalize 較 non-Normalize 好，RMSE 約低 0.006。而從各項（training, validation, public, private）RMSE 可以看到有 Normalize 皆比沒有來得有更好的表現。



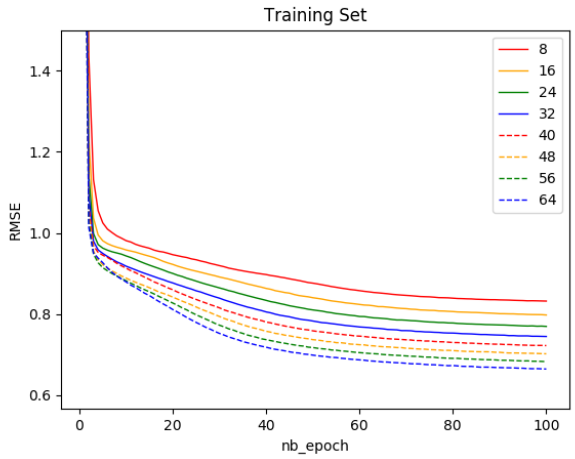


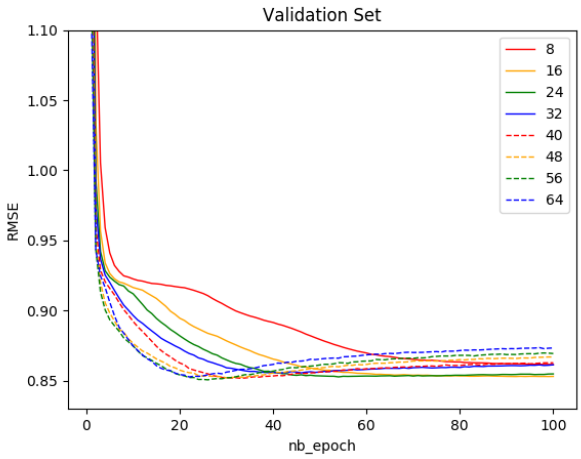
	Training	Validation	Public	Private
non-Normalized	0.79770	0.85315	0.85411	0.85616
Normalized	**0.77441**	**0.84755**	**0.85053**	**0.84946**

2. **(1%) 比較不同的 latent dimension 的結果。**

Model 除了 latent dimension 的大小之外，其餘架構及訓練過程皆與第一小題同，且沒有將 Rating 做 Normalize。測試 8, 16, 24, 32, 40, 48, 56, 64 八種不同的 latent dimension 大小。

由下頁圖可以看出，在 training set 上，很直覺的發現當我們所設定的 latent dimension 越大時，最後收斂到的 RMSE 越低；但是在 validation 上並不是 latent dimension 越大越好，可以發現最後收斂時的 RMSE 中，**最好的是 16**、其次是 24，且某種程度上當我們設定的 latent dimension 越大，最後收斂的 RMSE 也越差，可猜測是參數量太大所造成的 over fitting。

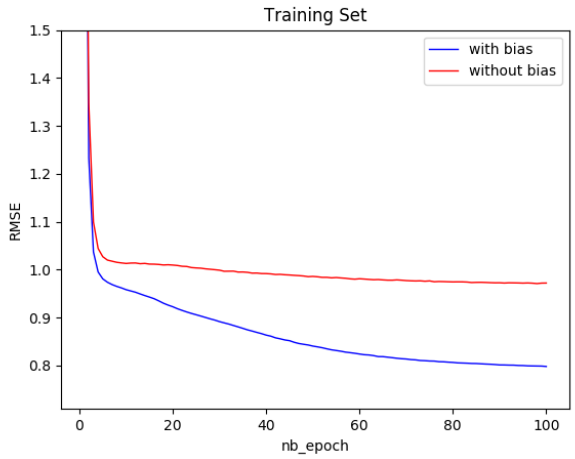


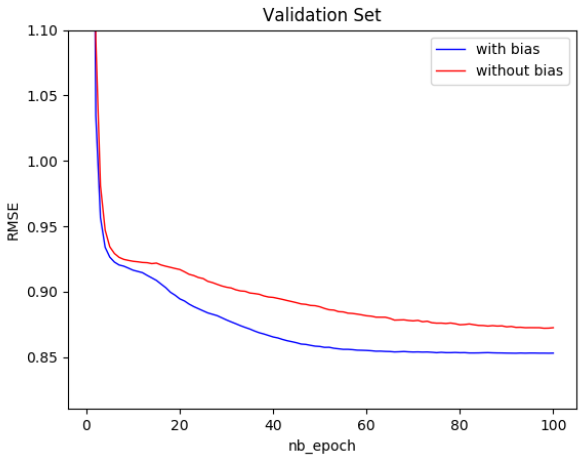


3. **(1%) 比較有無 bias 的結果。**

   Model 除了 bias 的有無之外，其餘架構及訓練過程皆與第一小題同，且沒有將 Rating 做 Normalize。

   由下圖可以看出，不論是在 training 還是 validation 上，有 bias 皆比沒有能達到更小的 RMSE，推估其原因為 bias 能夠表現每個 User 和 Movie 各有的 Rating 偏移，因此有加入 bias 的 Model 更可以準確預測到 Rating。由各項 RMSE 也可以看出有 bias 確實有更好的表現。





	Training	Validation	Public	Private
with bias	**0.79770**	**0.85315**	**0.85411**	**0.85616**
without bias	0.97186	0.87256	0.87559	0.87745

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

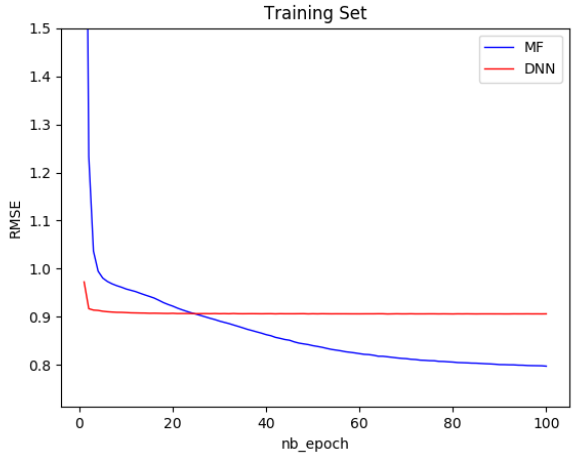
   (collaborator: B04902008 曾千育)

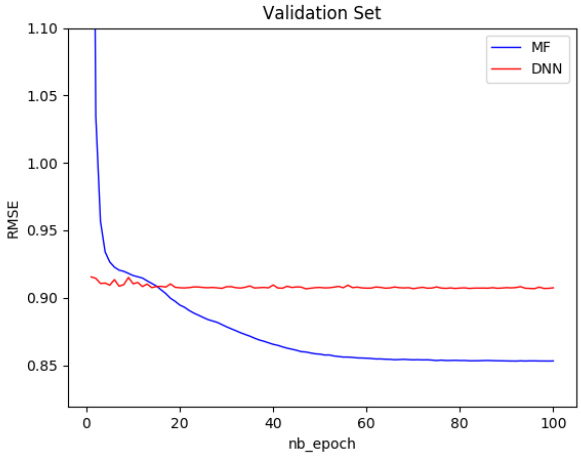
   MF Model 與第一小題同，且沒有對 Rating 做 Normalize。而 DNN Model 的部分則是將 MF Model 中 UserID, MovieID 分別 embedding 成的兩個 16 維向量用 keras.layers.Concatenate 併成一個 32 維向量再丟進 unit 依序為 256, 128, 64, 32, 1 的 Dense Layer，最後再加上用 UserID,

MovieID embedding 出來的兩個對應的 bias 後獲得 output，等於是當成一個 regression 問題來解決 。

由下圖可以看出，DNN Model 不論是在 training 或是 validation 上都很快就收斂了，且收斂的結果並沒有 MF 來得好，一開始想說可能是因為 embedding 出來的維度共只有 32 維，可能不夠讓 DNN 萃取出足夠的資訊，但有另外試驗 embedding 出共 128 維的版本畫出來的走勢也跟下面差不多，目前將這樣的結果差異歸咎在 DNN Model 的參數並沒有調整太好，所以造成效果不好；或是在這種情境下，使用 MF 作為訓練模型就是能夠得到比較好的效果（例如透過將兩個 embedding 出來的向量內積，比起用 DNN 做類似 regression 得到的值更能反映透過 embedding 出來的特性，因此有 MF 才有較好的表現）。

另外從各項 RMSE 也都可以看出 MF 較 DNN 表現來得好。

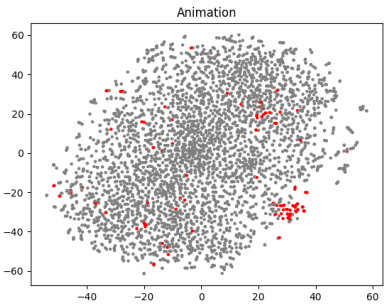


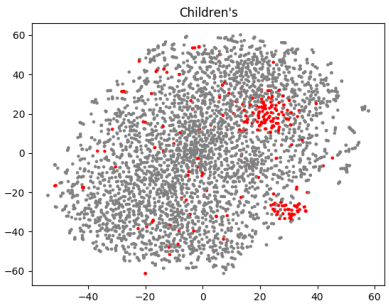


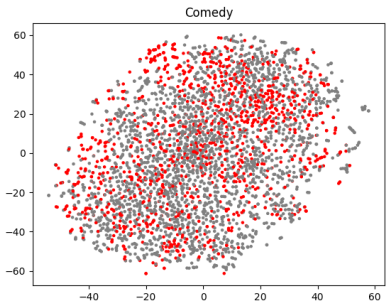
	Training	Validation	Public	Private
MF	0.79770	0.85315	0.85411	0.85616
DNN (32)	0.90228	0.90921	0.91048	0.90720
DNN (128)	0.90266	0.90539	0.91047	0.90684

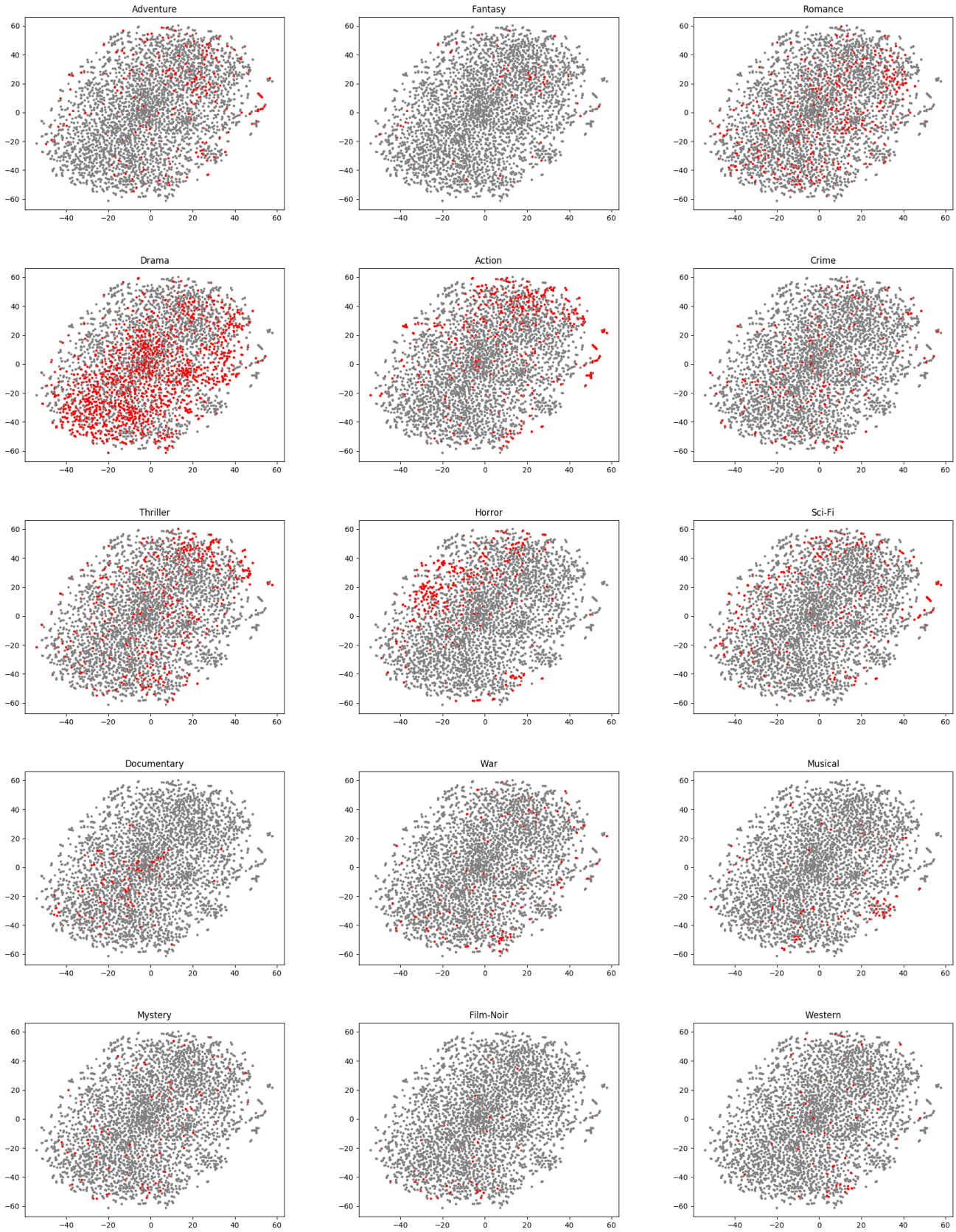
5. **(1%) 請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。**

將第一小題沒有 Normalize 的 Model 中將 MovieID 的 embedding layer 抽取 weight 出來，使用 sklearn.manifold.TSNE 將其降維成 2 維，並根據 18 種電影種類畫出 18 個二分的散佈圖如下（紅色的點即為對應其標題的類別）：

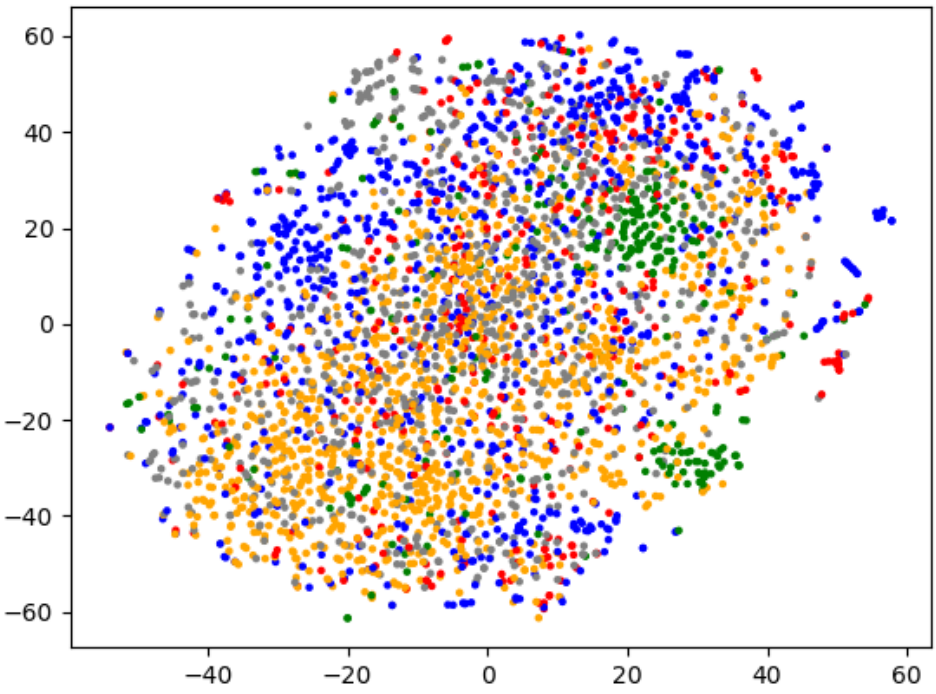






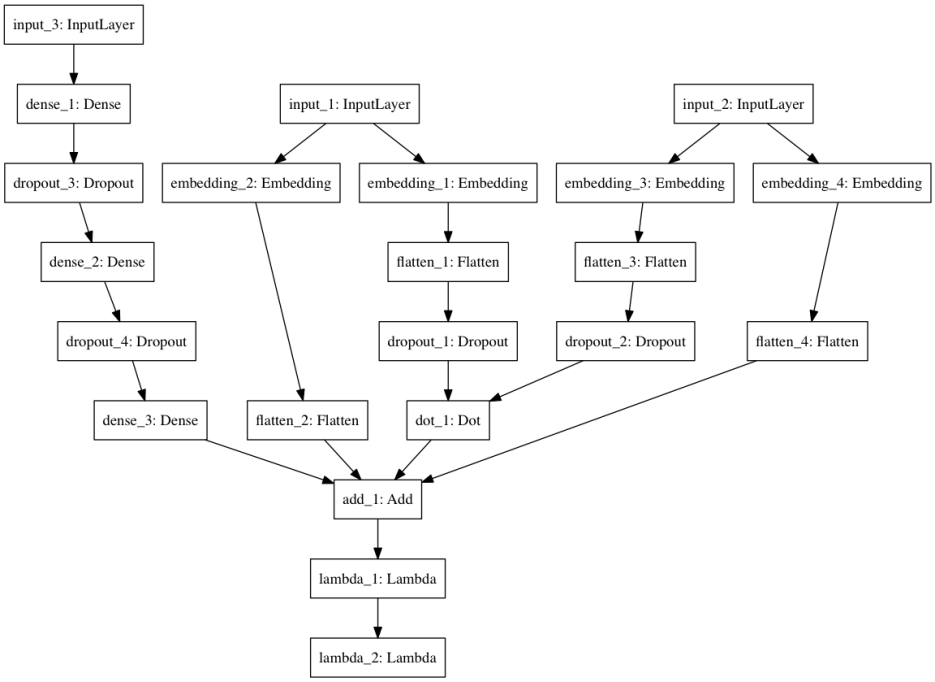


大概可以看出 Animation, Children's, Fantasy 這三類比較偏向右中與右下；Thriller, Horror, Sci-fi 這三類比較偏向上面那一大環；Adventure, Action, Crime 多位於上偏右的地方；Drama, Musical 則盤據整個下半部（稍微偏左），根據以上的分類重新在平面上標記，如下頁圖：

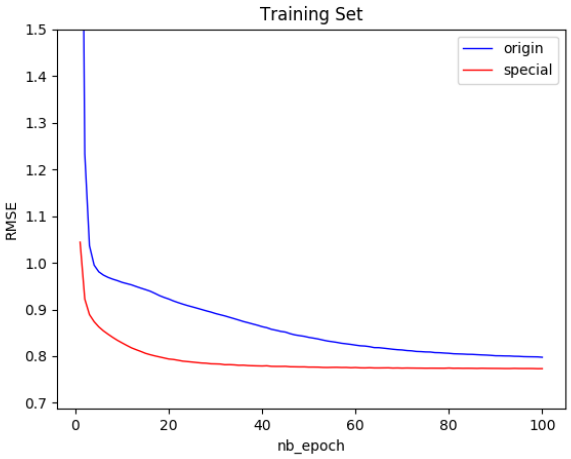


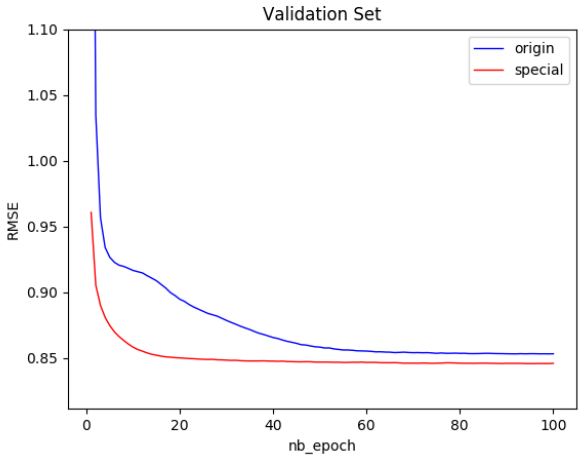
6. **(BONUS)(1%) 試著使用除了 rating 以外的 feature，並說明你的做法和結果，結果好壞不會影響評分。**

為了使用額外提供的 User 和 Movie 資訊，我將 User 的 2 種性別、8 種年齡層、20 種職業類別，及 Movie 的 18 種類別，弄成一串 48 維的 one-hot vector，然後依序經由 unit 為 64, 32, 1 的 Dense Layer，目的是想得到一個綜合以上資訊的新的 bias，並加到第一小題中的 Model 中，與其他 bias 一起加到最後 output 上，在此 Model 中有對 Rating 做 Normalize。模型架構如下圖，input_1, input_2 分別為 UserID, MovieID input，而 input_3 則為 48 維的額外資訊 vector input，最後兩個 Lambda Layer 則為標準化。



將訓練過程的 RMSE 記錄下來如下圖，可以看出加入了其他資訊抽取 bias 的 Model 不論是在 training 或 validation 都有較好的表現（validation 的 RMSE 約低 0.008）。而從各項 RMSE 也可以看出有加入其他資訊訓練的結果較好。





	Training	Validation	Public	Private
origin	0.79770	0.85315	0.85411	0.85616
special	0.77377	0..84534	0.84958	0.84835