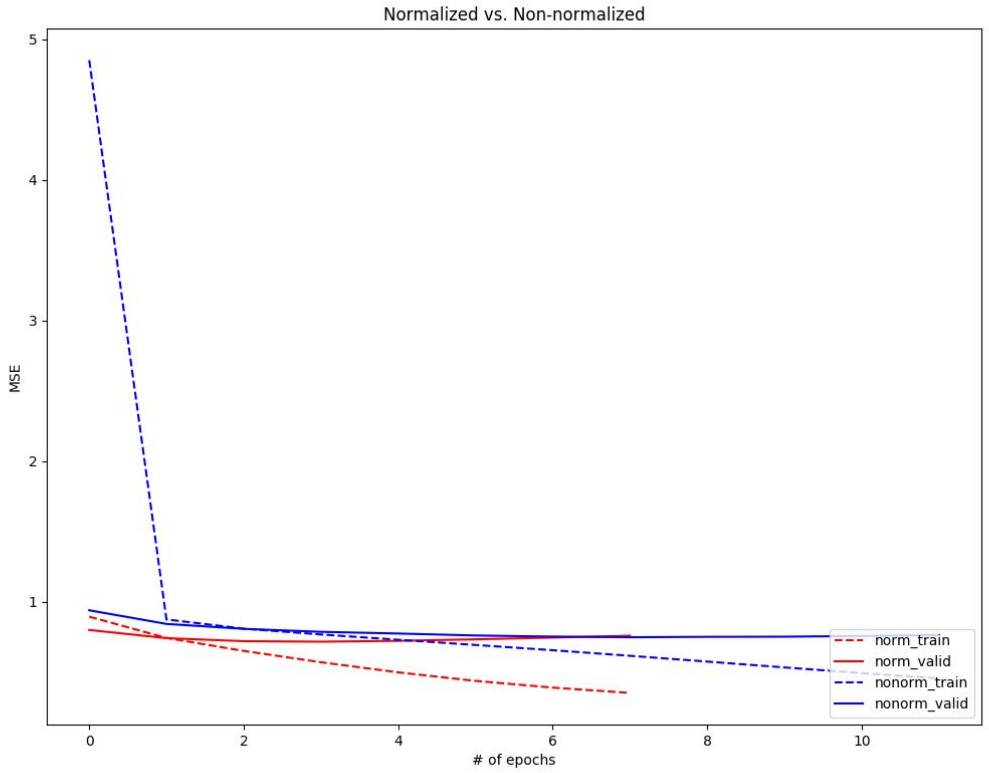
學號：R05943139 系級：電子碩一 姓名：張育瑄

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

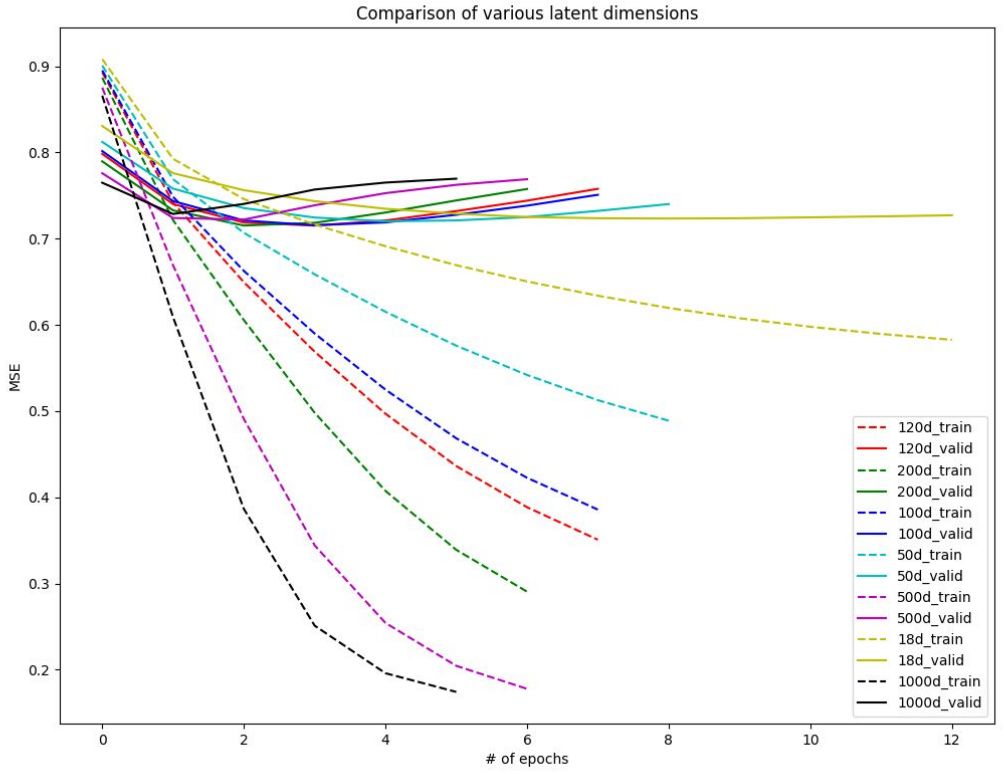


上圖為有 normalize 與沒有 normalize 的 training 過程比較，可以看出有 normalize 的表現明顯較好（在 Kaggle 上也是）。

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

與助教不同，我參考 Andrew Ng 在 Coursera 上的 ML 課程內容做 normalize，上圖中的公式為 MF 要 minimize 的 objective，θ 為 user 的 latent factor，x 為 movie 的 latent factor，y 為真正的 rating，假設有一種情況是，某個 testing set 中的 user 沒有評價過任何 movie，那麼這個 user 的 latent factor 就會是 0，導致這個 user 對任何 movie 的 rating 我們都會 predict 成 0，很明顯這樣的情況不是我們樂見的，因此，應該先將所有 movie 的平均 rating 算出來，然後把 training set 中所有的 rating 減掉它所屬的 movie 的平均，最後 predict 時再加回來，這樣一來，沒有評價過任何 movie 的 user 的 rating 就會被 predict 為它所屬的 movie 的平均 rating，而不是 0。

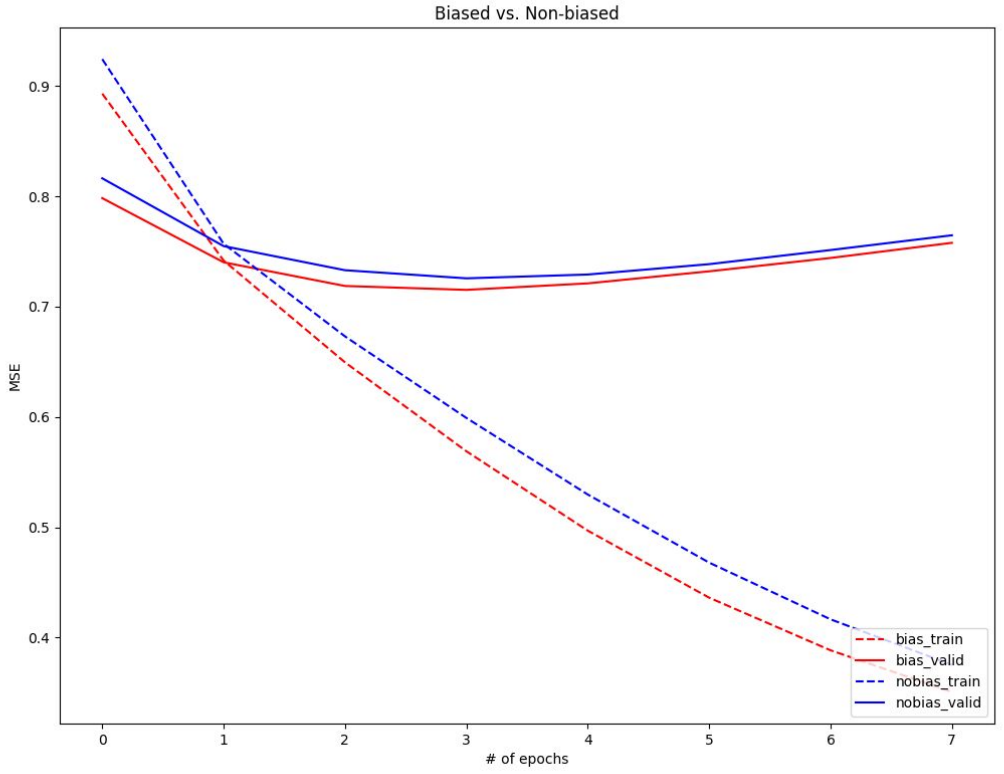
2. (1%)比較不同的latent dimension的結果。



上圖為我嘗試七種 latent dimension（18、50、100、120、200、500、1000）的 training 過程比較，右下角的 legend 以表現好壞由上往下排序，可以觀察到以下幾個現象：

- 維度越高，training set 上的 MSE 就降得越快越低，overfitting 也越嚴重。
- 100 ～ 200 維表現較好，而其中又以 120 維表現最好。

3. (1%)比較有無bias的結果。



上圖為有加 bias 與沒加 bias 的 training 過程比較，可以很明顯地看到，有加 bias 的 MSE 不管是在 training set 還是 validation set 上的表現都比較好。

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。

Layer (type)	Output Shape	Param #	Connected to
UserID (InputLayer)	(None, 1)	0	
MovieID (InputLayer)	(None, 1)	0	
UserLatent (Embedding)	(None, 1, 120)	840000	UserID[0][0]
MovieLatent (Embedding)	(None, 1, 120)	600000	MovieID[0][0]
FlattenedUserLatent (Flatten)	(None, 120)	0	UserLatent[0][0]
FlattenedMovieLatent (Flatten)	(None, 120)	0	MovieLatent[0][0]
ConcatenatedUserMovieLatent (Con	(None, 240)	0	FlattenedUserLatent[0][0] FlattenedMovieLatent[0][0]
Dense-1 (Dense)	(None, 64)	15424	ConcatenatedUserMovieLatent[0][0]
Dense-2 (Dense)	(None, 128)	8320	Dense-1[0][0]
Dense-3 (Dense)	(None, 256)	33024	Dense-2[0][0]
Rating (Dense)	(None, 1)	257	Dense-3[0][0]

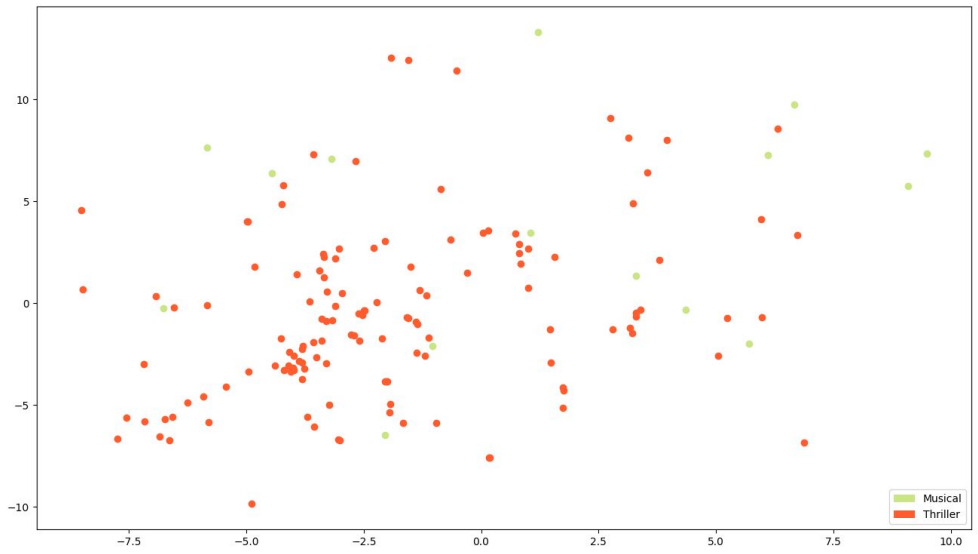
Total params: 1,497,025
Trainable params: 1,497,025
Non-trainable params: 0

跟 MF 一樣，使用一個 latent factor 表示 user，一個 latent factor 表示 movie，但是不將其內積，而是把它們 concatenate 起來看成是一大組 feature，再通過 DNN 對 output rating 做 regression。

以上圖的 DNN 結構 train 出的結果在 Kaggle 上的成績為 0.87004，通過 strong baseline，但還遠遠不及我用 MF 做出來的成績（Kaggle 0.84991）。

MF 與 DNN 會有如此大的差距，我認為最主要的原因是，DNN 將 user 以及 movie 的 latent factor 視為兩組沒有互動的參數，因此 DNN 只會知道某些 user 跟 movie 的排列組合 rating 比較高，某些比較低，而不會知道造成這些 rating 背後的 user 的 latent factor 與 movie 的 latent factor 之間的互動關係，所以表現自然差了一截。

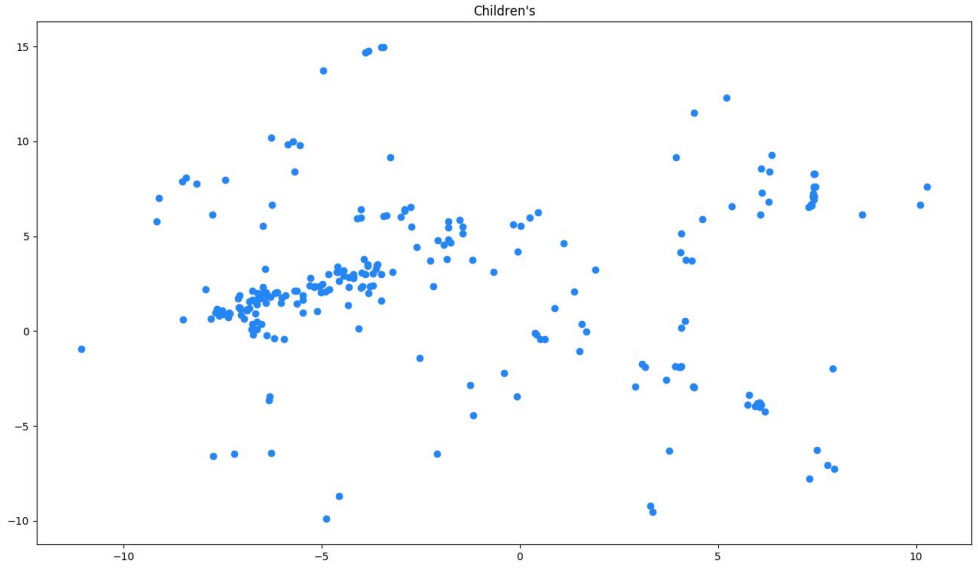
5. (1%)請試著將movie的embedding用tsne降維後，將movie category當作label來作圖。

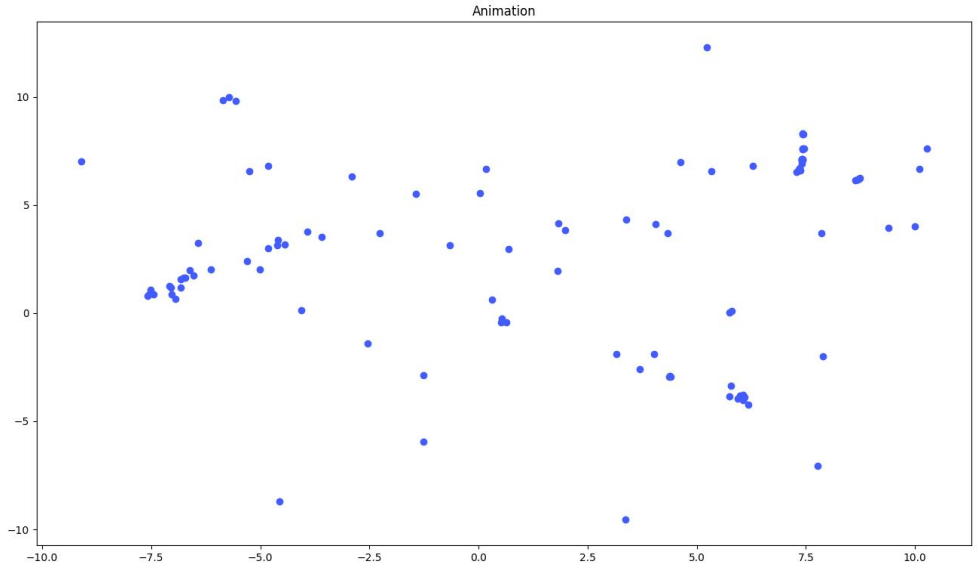


上圖為我嘗試將 musical 以及 drama 當作一類，而 thriller 和 action 另外一類，但從圖上可以看到，他們並沒有明顯的界線，完全違反我的直覺。

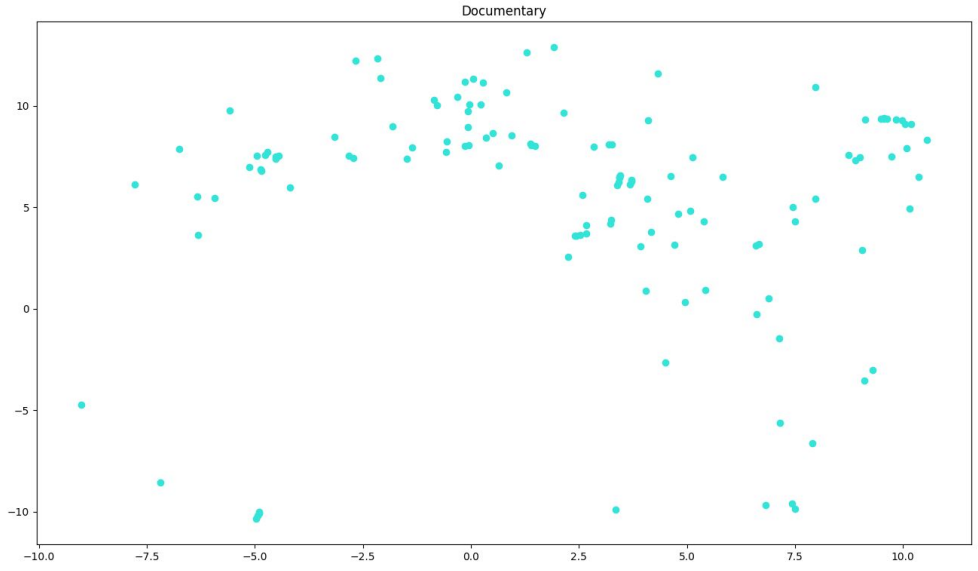
在嘗試將 movie 分類作圖後，我發現大部分都無法看出明顯的分界，我認為原因是每種 genre 之間常常是無法完全切割的，舉例來說，將 children's 與 animation 當作一類跟 thriller 與 horror 當作一類，在直覺上應該是天差地遠的類別，但由於一個 movie 的 genre 常常是 comedy 跟 horror 一起，也常常是 comedy 跟 children's 一起，才會造成它們之間會有許多重疊的區域，無法明顯分割。

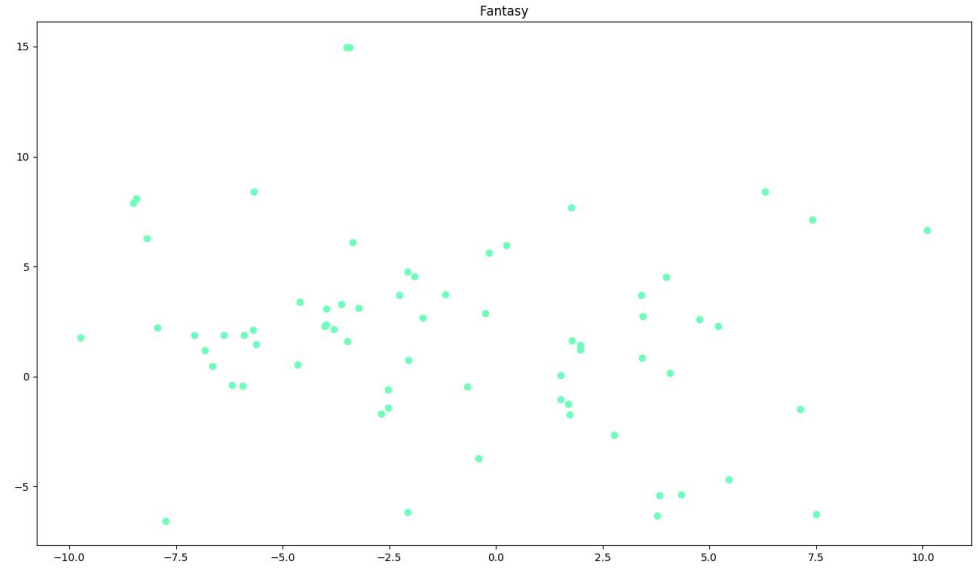
所以接下來我不分類，直接將 18 種 genre 分別作圖，再觀察其中的關聯。





首先，直覺上 children's 跟 animation 應該會最為相近，而從上面兩張圖我們可以看到，很明顯他們的 pattern 是接近的，並且距離也是接近的，符合我的直覺。





再來，documentary 是寫實的紀錄片，而 fantasy 是天馬行空的奇幻片，直覺上兩個就非常互斥，而從上面兩張圖可以看到，documentary 幾乎是偏右上角，fantasy 則是左下角，驗證了我的直覺。

總結來說，除非 genre 本身涵蓋的範圍非常明確並狹窄，否則很難與其它 genre 分別。

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。

使用 user 的 demographics 以及 movie 的 genres 作為 feature 丟進 DNN 對 rating 做 regression，但想當然爾，結果非常的差，連 Kaggle 的 simple baseline 都過不了。

直覺上來看，只用 user 的性別、年齡、職業及郵遞區號來代表一個 user 是遠遠不能代表他對於 movie 的 preference 的，其中職業跟郵遞區號幾乎沒用，而性別跟年齡雖然直覺上較有用，但在之前 MF train 好的 user latent factor 上做 tsne 降維後卻看不出明顯的分別。