學號：B03901039　系級：電機三　姓名：童寬

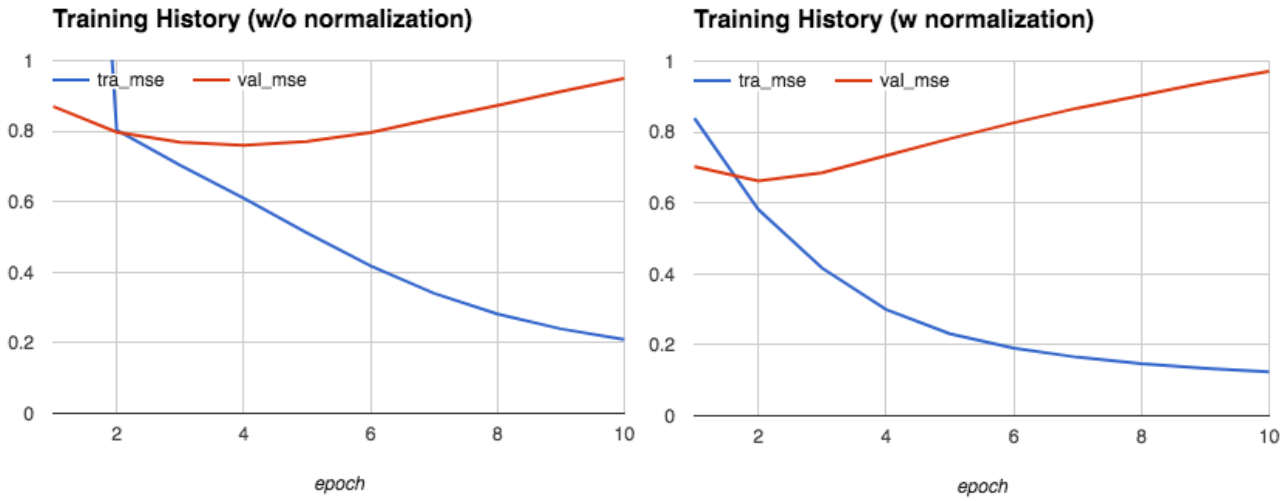
1. (1%)請比較有無 normalize (rating) 的差別。並說明如何 normalize.
以下兩個模型都是作業說明投影片第八頁的方式完成 (有加 bias) (latent dimension = 128)
我用以下的 code 去 normalize ratings：

```
ratings = (ratings - np.mean(ratings)) / np.std(ratings)
```

也就是將每一筆 rating 減掉全部的平均，再除以標準差
並且在 testing 時以下的 code 去還原 ratings：

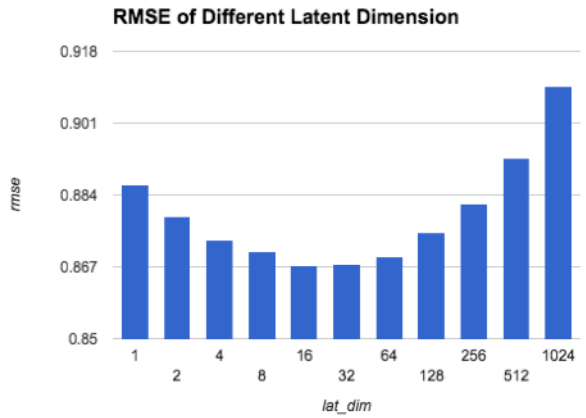
```
res = (res * np.std(ratings)) + np.mean(ratings)
```

實驗結果如下 (左圖是沒有 normalize，右圖是有 normalize)：



- 右圖的數據，我有將 mse 乘以 np.std(ratings)，讓它們的 scale 變回一樣
- 左圖第一個 epoch 的 tra_mse 太大 (3.552439286)，為了讓兩張圖的 scale 一樣，就不把它畫出來
- 右圖的 tra_mse 下降得更快，最低的 val_mse 也較左圖低
- 分別取在 validating set 上表現最好的模型，其在 testing set (Kaggle public and private) 的結果如下：
- 沒有 normalize 的 rmse：0.87516
- 有 normalize 的 rmse：0.86141，在 rating 上 normalize 能帶來不少的進步

2. (1%)比較不同的 latent dimension 的結果。
所有實驗都是以作業說明投影片第八頁的方式完成 (有加 bias) (epoch = 100)
我嘗試不同的 latent dimension，並取在 validating set 表現最好的模型去做 testing (Kaggle public and private)，得到實驗結果如下：



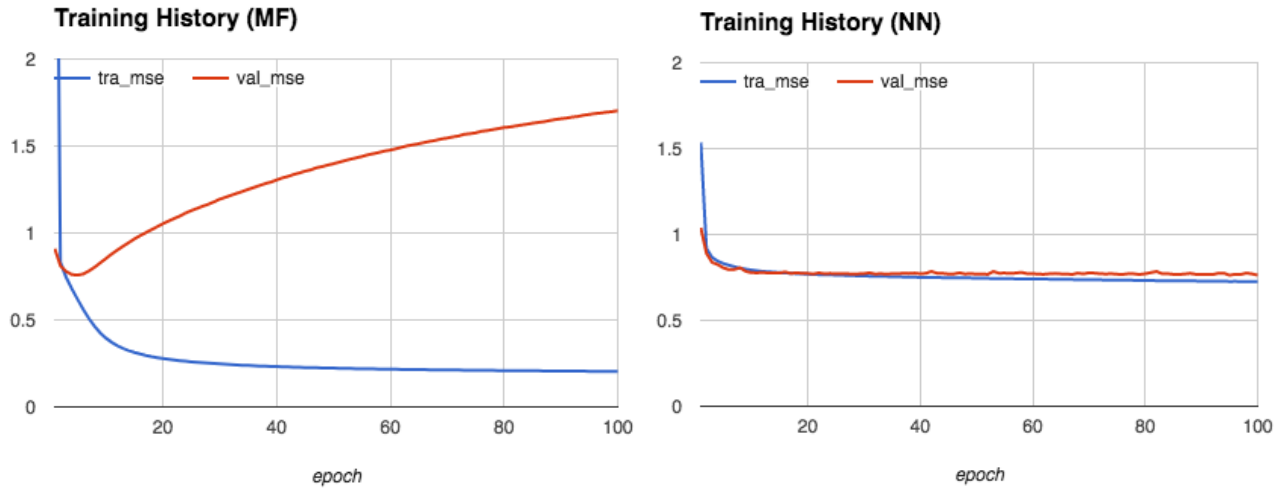
- lat_dim = 16 有最好的結果 (rmse = 0.86741)
- 在 training data overfit 的速度隨著維度增加而上升。維度為 1 使用的是第 96 個 epoch 的模型，維度為 1024 使用的是第 2 個 epoch 的模型

3. (1%)比較有無 bias 的結果。
所有實驗的 latent dimension 皆設為 16 (epoch = 30)
我將比較以下四個模型：沒有 bias，只有 user bias，只有 movie bias，有 user 跟 movie bias
並取在 validating set 表現最好的模型去做 testing (Kaggle public and private)，實驗結果如下：

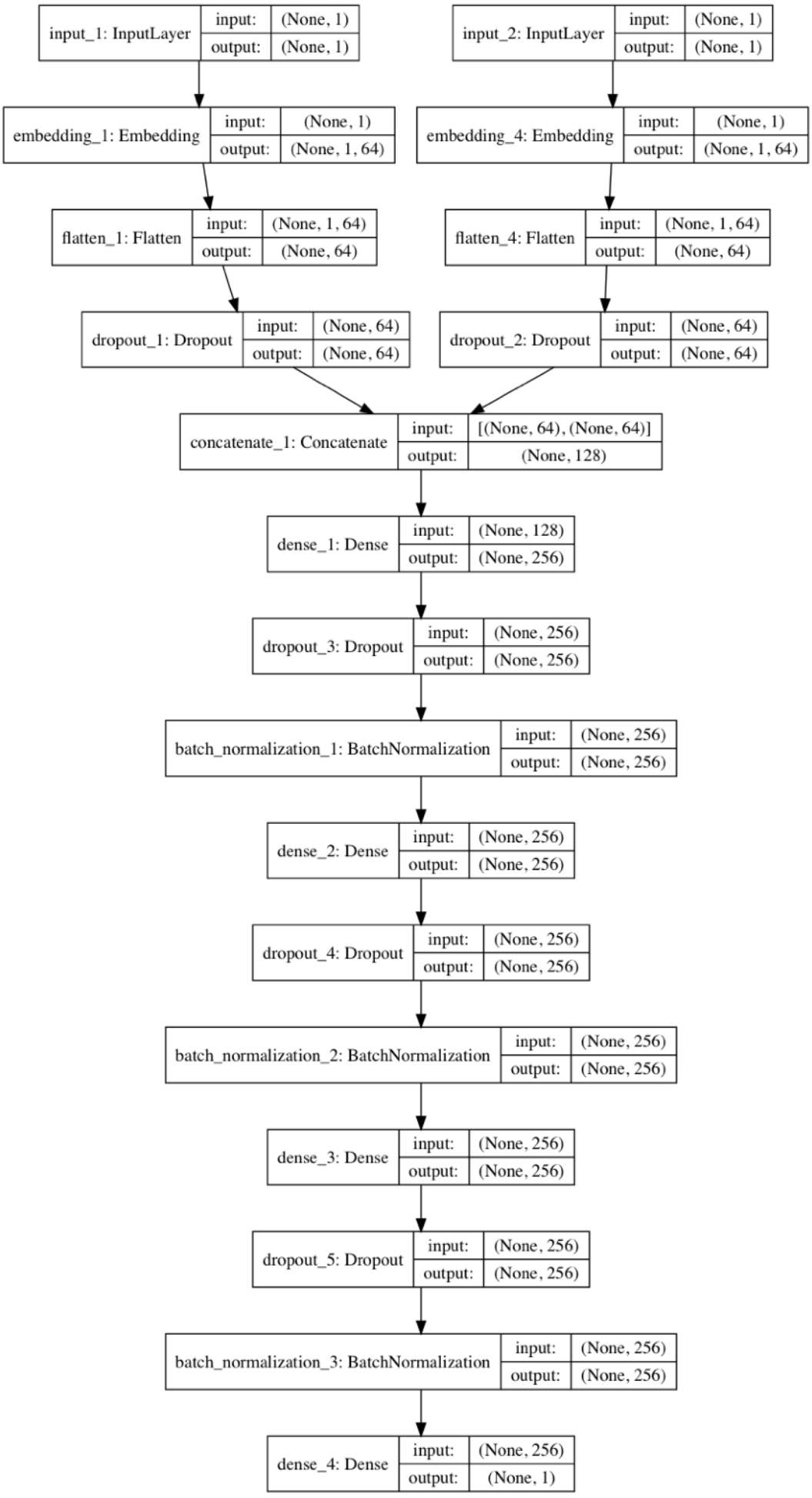
模型	rmse
沒有 Bias	0.86886
只有 User Bias	0.86831
只有 Movie Bias	0.86801
User 跟 Movie Bias	0.86741

- 從上到下結果越來越好。只有 Movie Bias 比只有 User Bias 好這點滿有趣的，可能代表電影本身對 ratings 的影響較大

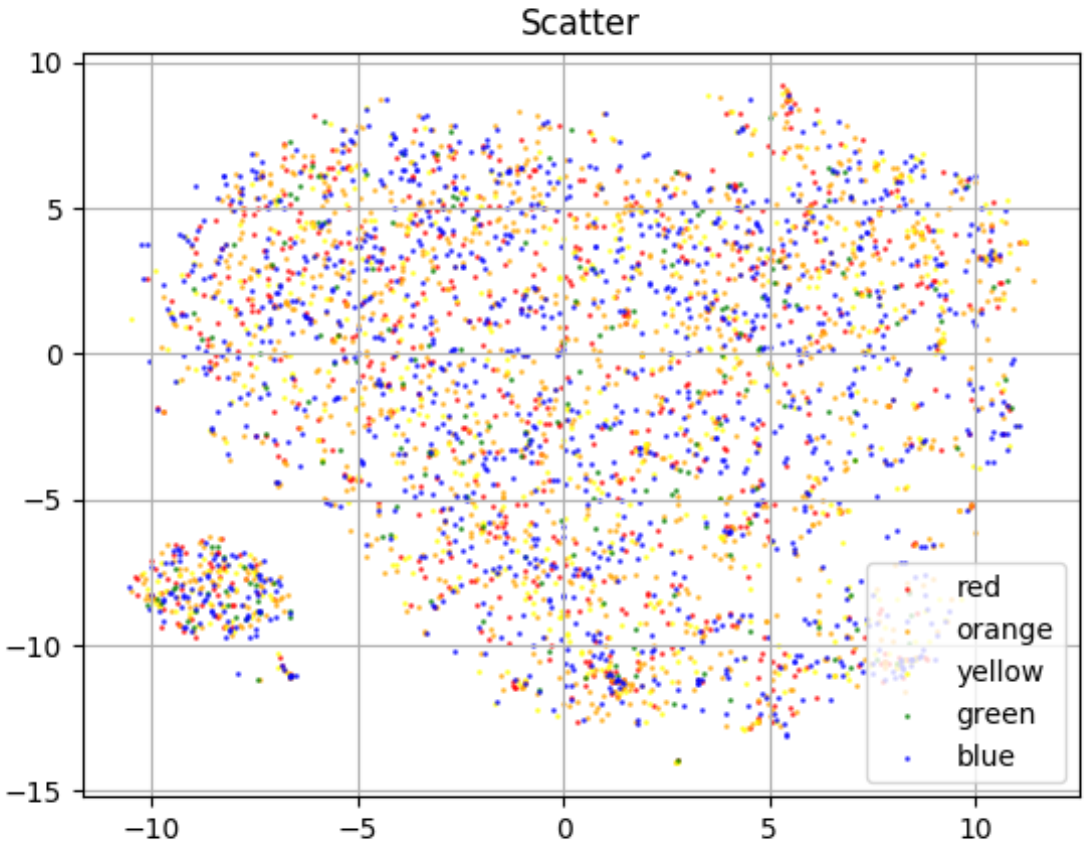
4. (1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。
我將 user 跟 movie 的 embedding concat 在一起，再把它送到 DNN，DNN 最後一層只有一個神經元，直接輸出 rating，把問題視為 regression。完整的模型架構因為長度太長，故擺放在下一頁，以下是實驗結果：
(MF 跟 NN 的 epoch 皆設為 100，latent dimension 皆設為 64)



- MF 很快就在 training set 上 overfit (在 validation 上變差)
- NN 在 100 epoch 裡還沒 overfit 收斂，如果繼續訓練下去，應該能得到更好的結果
- 分別取在 validating set 上表現最好的模型，其在 testing set (Kaggle public and private) 的結果如下：
- MF 的 rmse：0.87119
- NN 的 rmse：0.87301


input_1: InputLayer | input: (None, 1) | output: (None, 1)
input_2: InputLayer | input: (None, 1) | output: (None, 1)
embedding_1: Embedding | input: (None, 1) | output: (None, 1, 64)
embedding_4: Embedding | input: (None, 1) | output: (None, 1, 64)
flatten_1: Flatten | input: (None, 1, 64) | output: (None, 64)
flatten_4: Flatten | input: (None, 1, 64) | output: (None, 64)
dropout_1: Dropout | input: (None, 64) | output: (None, 64)
dropout_2: Dropout | input: (None, 64) | output: (None, 64)
concatenate_1: Concatenate | input: [(None, 64), (None, 64)] | output: (None, 128)
dense_1: Dense | input: (None, 128) | output: (None, 256)
dropout_3: Dropout | input: (None, 256) | output: (None, 256)
batch_normalization_1: BatchNormalization | input: (None, 256) | output: (None, 256)
dense_2: Dense | input: (None, 256) | output: (None, 256)
dropout_4: Dropout | input: (None, 256) | output: (None, 256)
batch_normalization_2: BatchNormalization | input: (None, 256) | output: (None, 256)
dense_3: Dense | input: (None, 256) | output: (None, 256)
dropout_5: Dropout | input: (None, 256) | output: (None, 256)
batch_normalization_3: BatchNormalization | input: (None, 256) | output: (None, 256)
dense_4: Dense | input: (None, 256) | output: (None, 1)

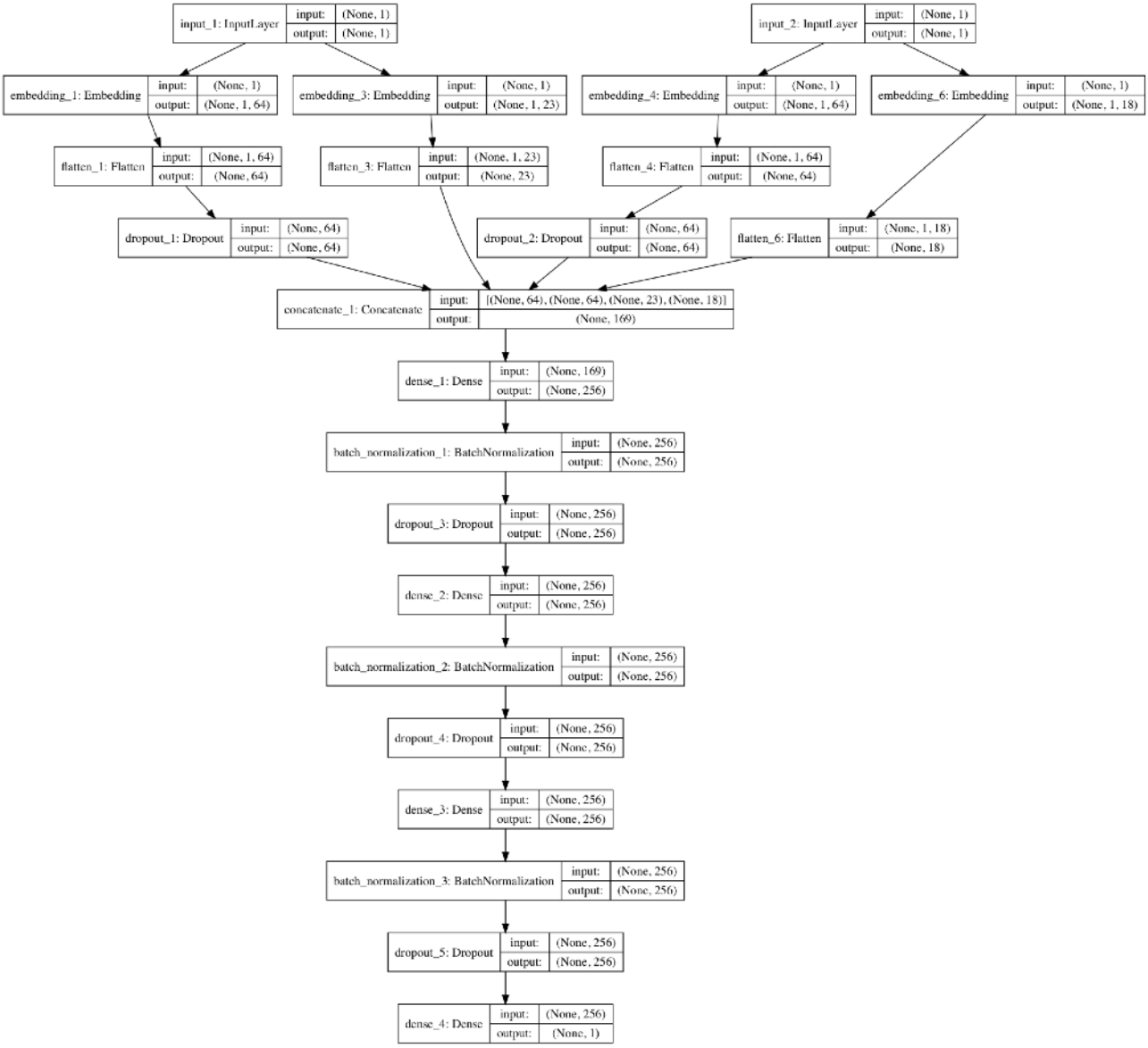
5. (1%)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。
我把所有類別分成五個大類，由以下顏色代表：
紅色：Action, Adventure, War, Western
橘色：Animation, Children's, Comedy
黃色：Crime, Film-Noir, Horror, Mystery, Thriller
綠色：Documentary, Drama, Musical, Romance
藍色：Fantasy, Sci-Fi
(每筆資料皆是由第一個 genre 去區分它屬於哪一大類)



- 圖的左下角有一群跟其他分離的資料，但每一種種類的數目差別並不會太明顯，每個種類在這裡都有一定數量的資料
- 藍色跟橘色的資料在 y 軸上普遍偏大，x 軸上普遍偏小
- 黃色跟紅色的資料分布得很平均
- 綠色的資料感覺比較多集中在左下角那一塊
- 我覺得我以第一個 genre 去區分這個方法，並不是太好。如果有像是屬於每個種類的比率，再以最高的那個去分類，畫圖出來區分的結果會更好

(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。

- User 的部分我把 Gender, Age 跟 Occupation 讀進來：
  Gender 用 0 表示 F (Female)，1 表示 M (Male)
  Age 把它們 normalize 當成連續資料
  Occupation 做 vectorization，也就是類似 one-hot 那樣的 encoding
- Movies 的部分我用了 Genres。因為總共有 18 個種類，所以就用一個 18 維的 vector 去表示 (以 0/1 表示有沒有該類別)
- 再將上述資料 (分別是 23 維跟 18 維) 跟原本 users 跟 movies 的 embedding concat 在一起，送到 dnn 裡面，最後直接以 rating 當作輸出。以下是模型架構跟實驗結果：



- 取在 validating set 上表現最好的模型，其在 testing set (Kaggle public and private) 的結果如下：
- rmse：0.84600，結果有比只用 rating 來得好