學號：B02901093 系級： 電機四 姓名：吳岳

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

Normalize Rating 的有無在validation Loss的差距非常明顯，我如果用Normalize Rating 配上latent dimension = 20 再加上 bias，loss最低可以到0.6128。然而，我如果沒使用Normalize Rating(latent dimension, bias 條件一樣)，loss 最低是0.7771。

做法：首先，先計算rating 的mean跟std。再將所有的rating減掉mean並除以Std，這樣可以確保rating的平均為0，標準差為1。而在跑testing的時候，記得把training 的std & mean 用pickle等檔型預存下來，將預測出的testing data * std_train + mean_train。

2. **(1%)比較不同的latent dimension的結果。**

在Bias & Normalization都有設定的前提下，我測試了Latent Dimension = 15, 20, 25, 30, 35, 40 六種的狀況，結果如下：

Dimension	15	20	25	30	35	40
Val_loss	0.6133	0.6156	0.6153	0.6171	0.6139	0.6128

從我自己的測試中，這些不同的Dimension對Val_Loss並沒有太大的影響。另外，若在重複做同樣的測試，我發現Loss最低的Dimension從40變成25，40反而變成次高的Loss，推論在這個區間下Dimension跟Loss並沒有太大的相關性。

3. **(1%)比較有無bias的結果。**

我比較Latent Dimension = 20、做Normalization之下Bias有無對於Val_Loss的影響，結果如下：

Bias Or Not	Yes	No
Loss	0.6156	0.6220

根據測試結果，在沒有Bias的情況下Val_Loss是較高的，因為Bias有考慮使用者的評分傾向(ex:習慣評低分或高分)，能比較精確的呈現預

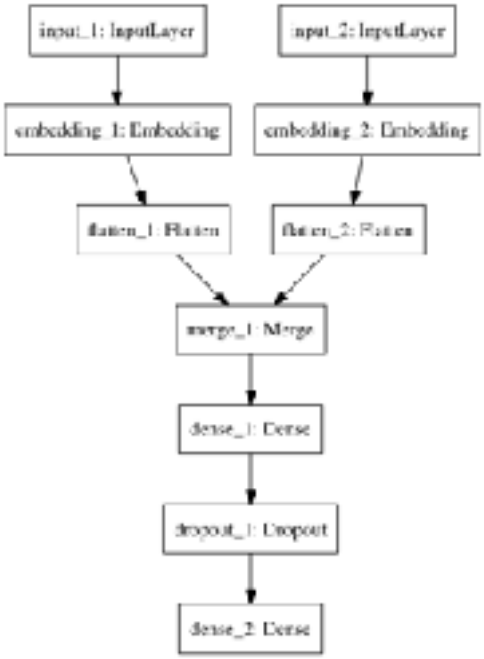
測結果。

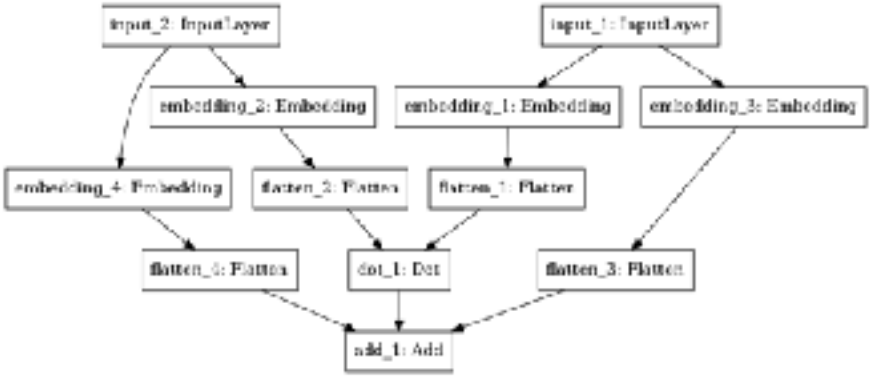
4. **(1%)請試著用DNN來解決這個問題，並且說明實做的方法(方法不限)。並比較MF和NN的結果，討論結果的差異。**

我Loss最低的DNN是用Linear Regression，因Classification的結果並沒有想像中來得好。另外，我只有放一層Dense(Unit = 256)，User & Movie從InputLayer輸入之後經過Embedding、Flatten、Merge(Concat merge)的處理之後就直接輸出，根據Mean Square Error 找最低的Loss，並用Batch = 128，patience = 10。在Kaggle上的Public的結果是0.859。(架構如左下)

此外，我MF的方法其實就是用助教的Code，在嘗試不同Latent Dimension之後選用Dimension= 40，Kaggle上的成績是0.875。(架構如右下)

差異除了Loss之外，架構上MF看起來比較胖是因為Bias也做了Embedding跟Flatten，另外，DNN參數調整的彈性較高，可以再去調整Dense的層數、Dropout跟是否要做Batch Normalization，這些都可以讓DNN的結果遠高於MF。



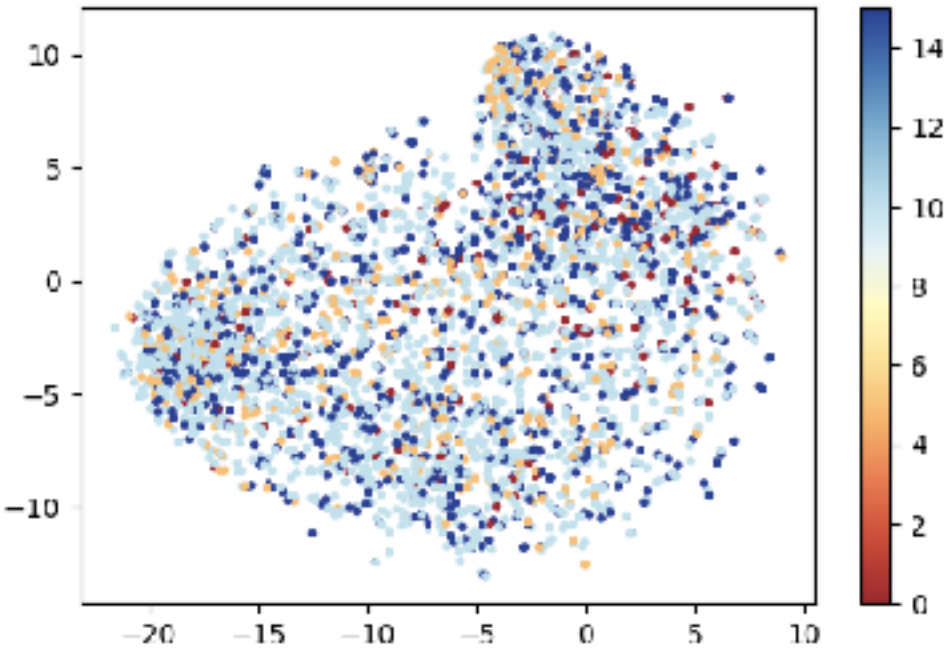


5. (1%)請試著將movie的embedding用t降維後，將movie category當作label來作圖。

我主要將movie category分成4大類，如下表：

Category	Type
0	Animation, Childen's
1	Crime, Horror, Thriller
2	Drama, Comedy, Musical
3	Adventure, Action

另外，我得到的降為圖片如右，可以發現Category幾乎都混在在一起，沒有很明顯的呈現區別。

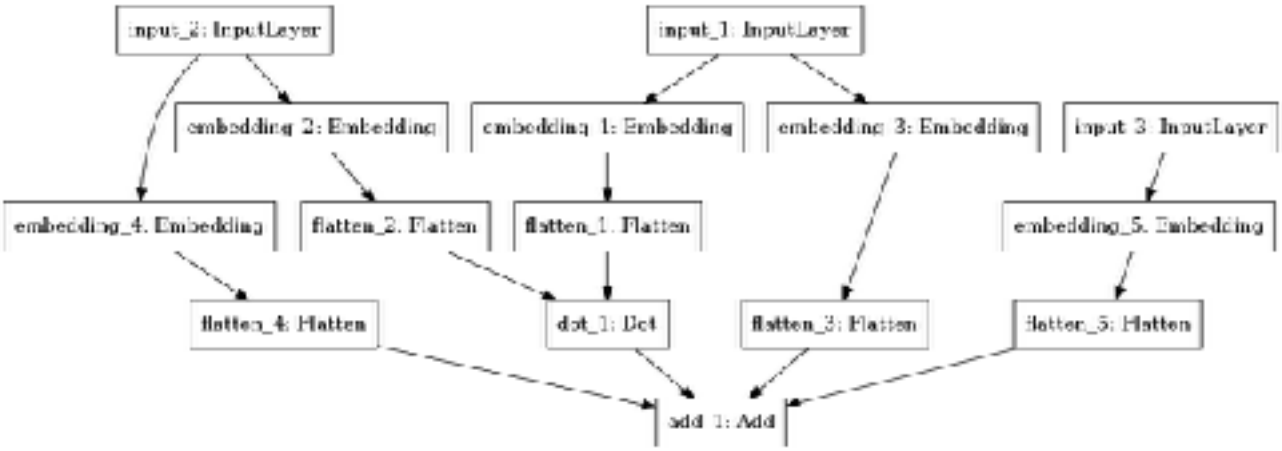


從這樣訓練過後的embedding降維對應到不同的Movie category，感覺跟最後輸出的Rating相關度不高，畢竟使用者也不見得會根據Movie的類別決定分數給低還是給高。

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。

參照第五題的做法，我把Movie Category當成Bias也加進去了。分類參考上題4大類 +「其他類別」+ Unknown Movie共六大類，Latency Dimension = 20、有做Normalization。

架構如下圖：



最後，我的Val_Loss是0.6095，比第二題同樣參數下的結果還低，Kaggle上的Error是0.87059 (過Strong Baseline)。