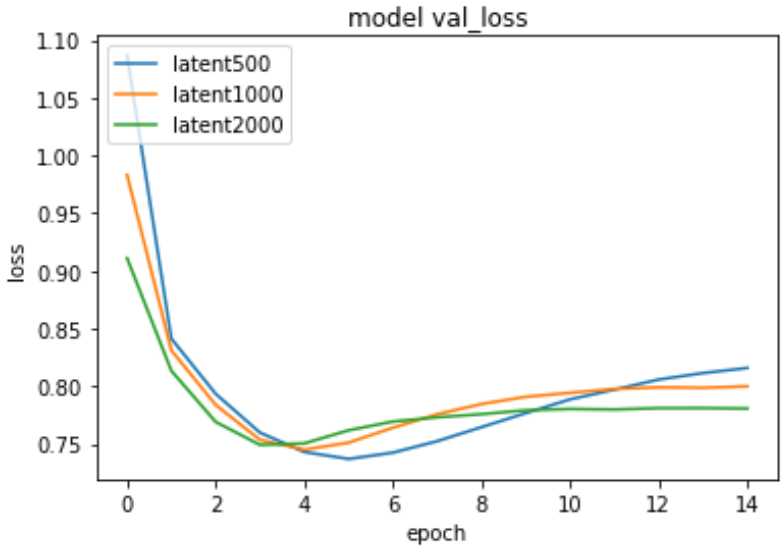
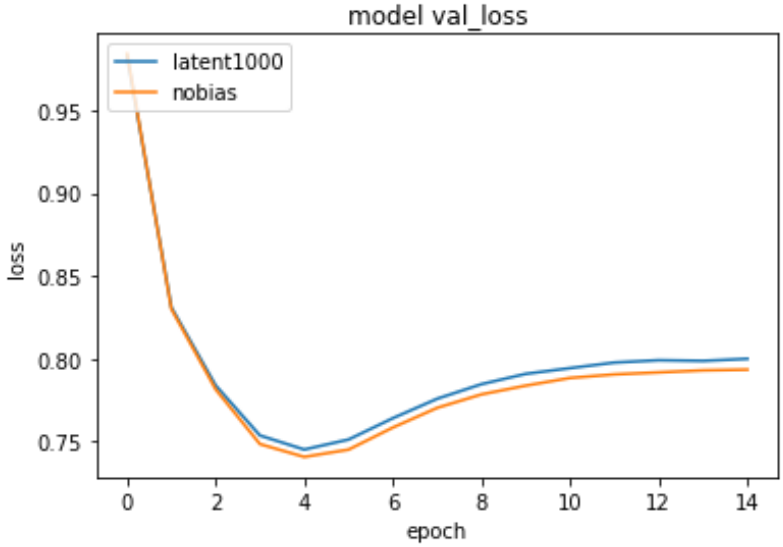
學號：r05229016 系級： 大氣碩二 姓名：羅章碩

1. **(1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.**
   (collaborator: r05229014 鄒適文)
   首先，我 normalize 的方法就是先對 rating 取平均和標準差，之後每個 label 都減掉平均之後再除上標準差，而最後 model predict 出來的結果再把標準差乘回去並加上平均，得到最後的預測結果。在 normalize 的部分我總共測試了兩組，第一種是有使用 EarlyStopping 並設定 patience 是 1，也就是當 val_loss 上升時就會停止訓練，另外一種則是固定設定 Epoch 是 15，不管有沒有出現 overfitting 當繼續訓練下去。
   第一組的結果為：沒有做 normalize 時在 private 和 public 的分數分別為 0.864,0.862，而有做 normalize 時結果為 0.876,0.875。
   第二組的結果為：沒有做 normalize 時在 private 和 public 的分數分別為 0.892,0.890，而有做 normalize 時結果為 0.893,0.892。
   從上述結果看到有做 normalize 不管怎麼樣都得到比較差的分數，因此我覺得在這個預測電影排行時，做 normalize 反而會減小 rating 之間的差距，而差距變小，對 model 而言會變成是這個電影跟另一個電影的 rating 相似，代表他們差距不夠大，這對 model 在預測哪個電影 rating 會比較高時會變得比較困難。
2. **(1%)比較不同的 latent dimension 的結果。**
   (collaborator: r05229014 鄒適文)
   這題跟上一題一樣分成兩組(有使用 early stop 和固定 epoch 為 15)，其中每一組包含了 latent 500、1000、2000 去做 training，在第一組有使用 early stop 的分數中，latent 為 500 時在 kaggle 上的結果最好，可達到大約 0.85 接近 0.86 的成績，而 latent 為 2000 時的結果則最差為大約 0.87，但如果是看第二組時可以發現答案是相反過來的，最好的是 latent 2000、最差是 latent 500。得到上述結果後，我就畫出 latent 分別為 500、1000、2000 在 val_loss 上的結果(下圖)。從這結果可以看到 latent 500 隨著 epoch 的增加，它的變化幅度是最大的，在 4~6 之間它的 val_loss 是最小的，但隨著 epoch 變多，它 over-fitting 的程度也增加最快(15 epoch training data 上的 loss 其實還是持續下降，但 val_loss 已經開始上升)，而這也可以解釋為什麼它在第一組中的結果可以最好，在第二組中是最差的。除此之外，可以看到 latent 2000 時它的變動幅度是最小的。從以上結果可以推斷當 latent 比較小時 loss 的變動幅度比較大，而 latent 越大時 loss 的變動幅度則比較小。



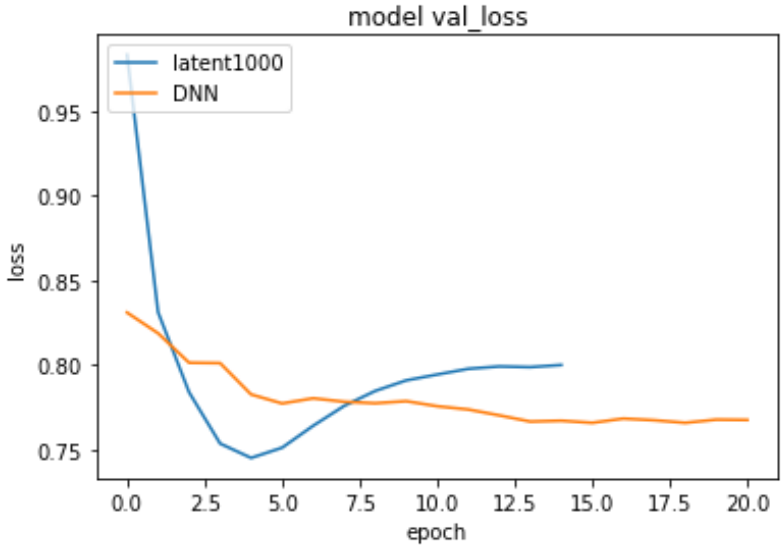
3. **(1%)比較有無 bias 的結果。**
   (collaborator: r05229014 鄒適文)
   這邊也同樣分為兩組。第一組中有加 bias 的分數為 0.864,0.862，而沒有 bias 則為 0.861,0.862，第二組中有加 bias 的分數是 0.892,0.890，而沒有 bias 的則為 0.893,0.891。從這兩組的結果來看有沒有加 bias 的影響其實不大，差距大概是 0.001~0.003 之間，沒有加 bias 在第一組中的成績好像有好一點點，而在第二組中則比較差一點點，但這差距實在是不大，所以我覺得對於相較於 latent dim 大小而言，有沒有 bias 好像影響不大。下圖是我把有 bias 和沒有加 bias 的 val loss 隨著 epoch 增加的變化畫出來，從這張圖也可以看到兩條線趨勢一樣而且值相差也不大，因此我覺得有沒有 bias 對結果影響並不大。



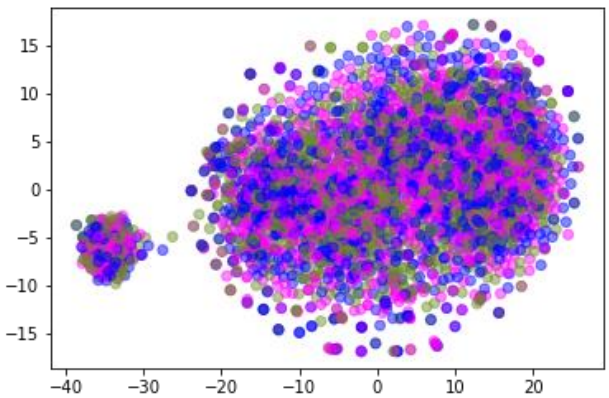
4. **(1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。**
   (collaborator: r05229014 鄒適文)
   我的 DNN 使用 keras 來實作，在把數據餵到 hidden layer 之前，先用 embedding

層將使用者 id 和 movie 的 id 做轉換，之後使用 flatten 在將他們兩個 merge 在一起，hidden layer 中我使用了三層的 fully-connected network(activation function 都是 selu)，而神經元數量分別為 256,128,64，並在每一層中加 dropout 來降低 overfitting 的機率，最後 output 一個數字當作 rating 結果，使用 mse 當 loss function，並用 adam 方式來做 gradient descend。在結果上來看，training 中都跑了 15 個 Epoch、batchsize 為 2048，DNN 有較好的結果(DNN 的 loss 大約 0.88，MF 大約 0.89)，因為 MF 已經出現 overfitting 的情形了，但是考慮 early stop 進去後，MF 可以得到較好的結果，DNN 的 loss 則一直降不下去了。

下圖為 MF latent1000 和 DNN val loss 的結果，可以看到 MF 的 loss 馬上就掉下去了，但是之後就又逐漸變高，容易有 overfitting，所以如果我取大約 epoch 3 的時候的 MF model 去做預測就可以得到不錯的結果，而 DNN 則是慢慢下降，但是到大約 epoch 12 的時候 loss 的值就下不去了，最後因為有設 early stop(patience 5)停在 Epoch 20。



5. **(1%)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。**
   (collaborator: r05229014 鄒適文)



總共分成三類。

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。