**Recap and Overview**

* Short Recap of what we did in blog post 2
* Our goal in this blog post:
  + Improve the performance of the transformer
  + implement more DL applications for movie recommendation task: autoencoders and word2VEc
  + Generate movie recommendations based on users or movie info

**Transformer**

* **Refer to Drew’s note:** https://docs.google.com/document/d/1ocNi6YfP7zrHb6yApEC3u8uHWTogShUBNxp-C7u6Umo/edit?ts=607c8cfe

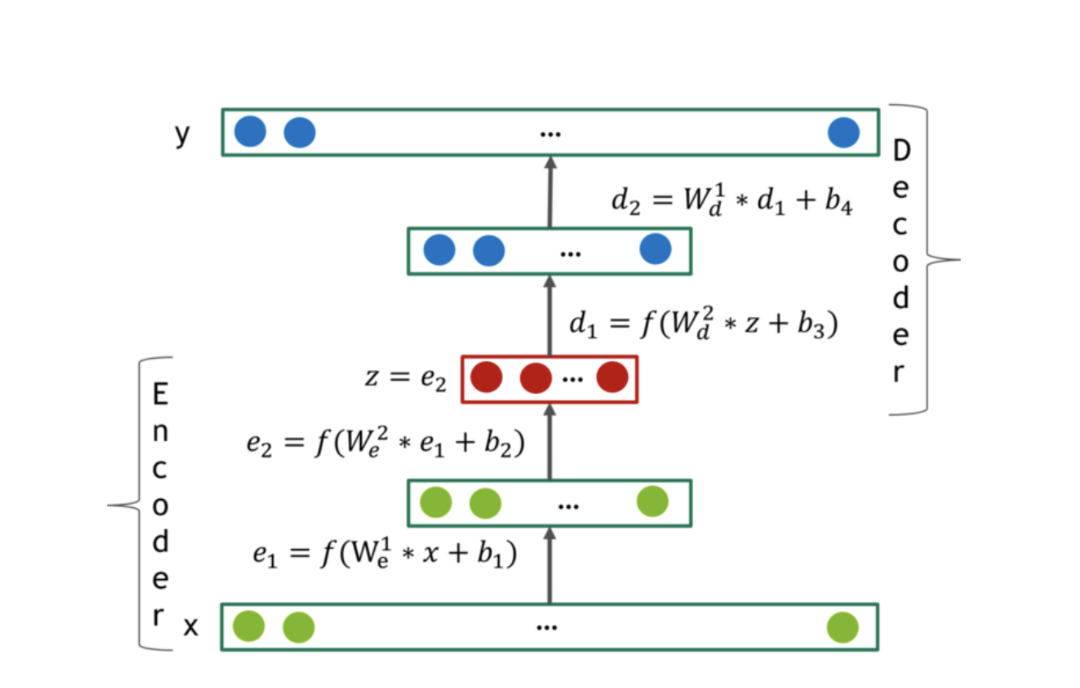
**AutoEncoders**

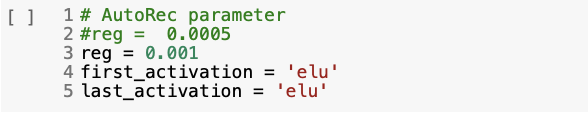
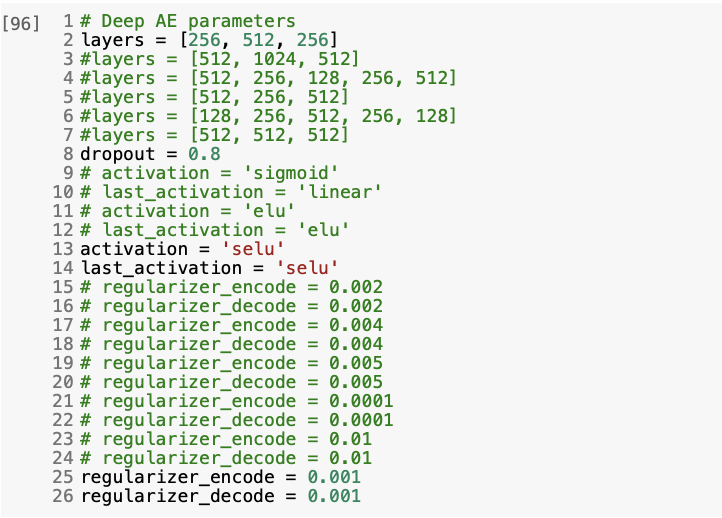
* Overview of autoencoders: what are they and how do they work for recommendation task specifically (just collaborative filtering)
* Data preprocessing: split the data into random approximately 80% train, 10% val, 10% test, the function dataPreprocessor transforms dataset 2D matrix with rows = users, columns = movies, and each entry = ratings of a specify user for a specific movie, this is needed for the autorec. In the function, the init\_value is the default rating for unobserved ratings, which range from 0 to 5. If average is set to True , the unobserved rating will be set as the average rating of the user.
* Utility functions: need to write some custom loss functions: masked\_mse, masked\_rmse, and masked\_rmse\_clip based on the “Autorec : Autoencoder meets Collaborative Filtering” paper by Sedhain because we only take into consideration the error where the rating is not zero in the test set. MMSE is not provided in keras so needs a custom function. Overall explanation is for MMSE is here:

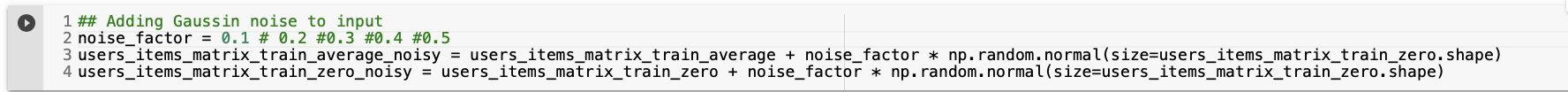


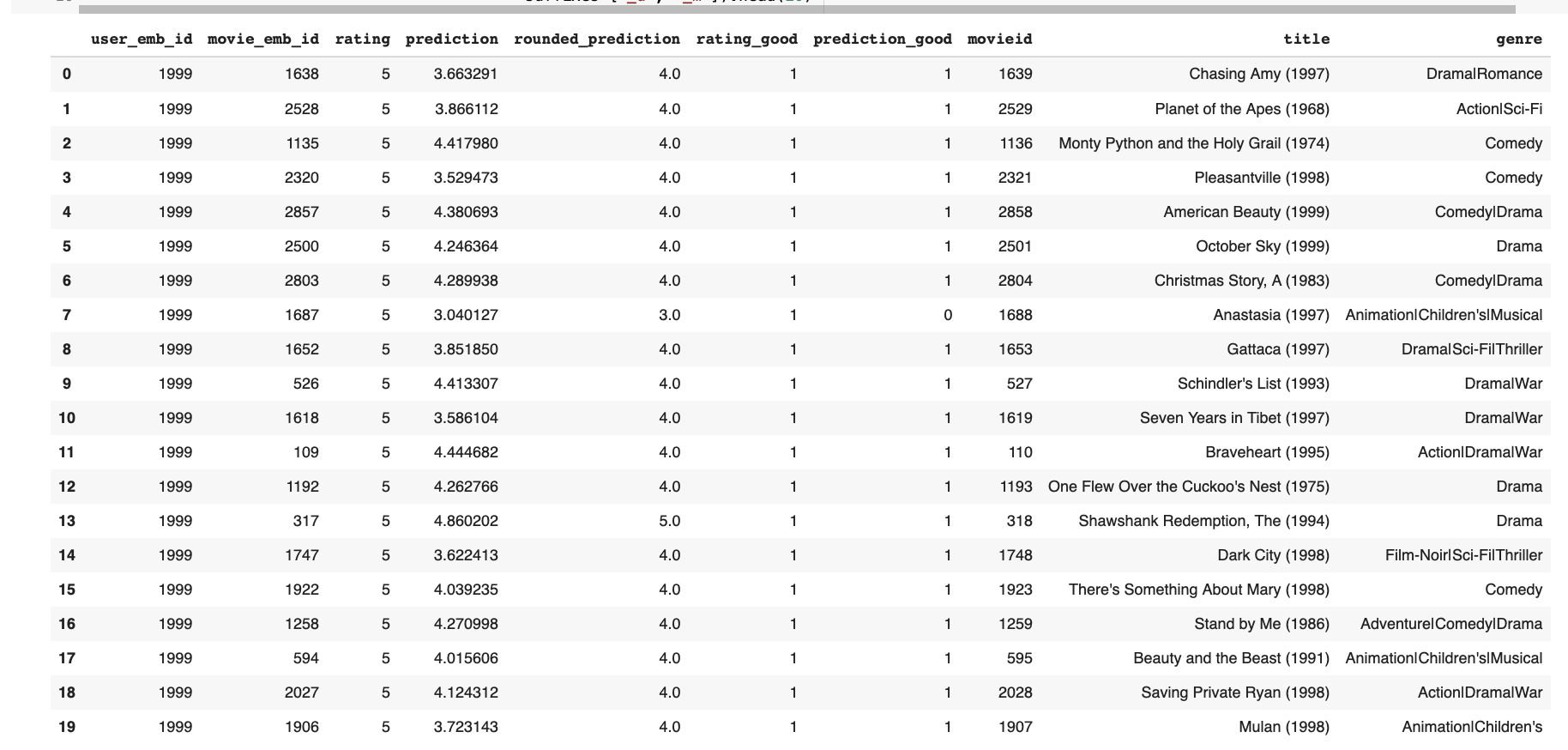
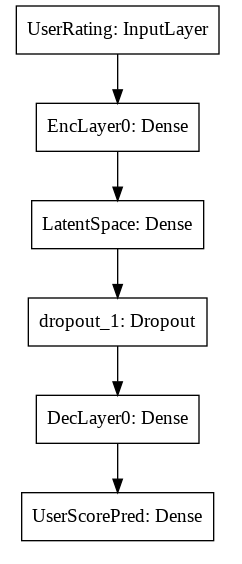
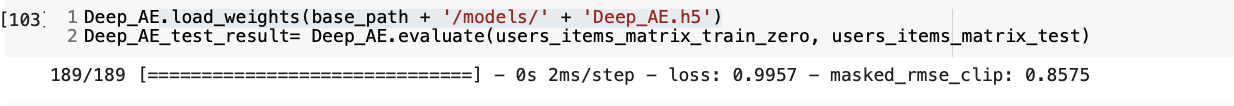
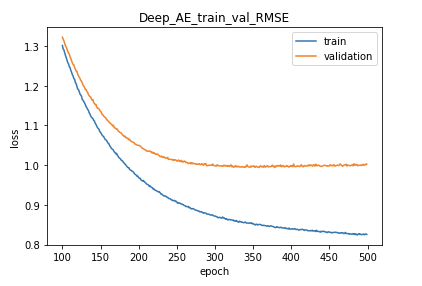
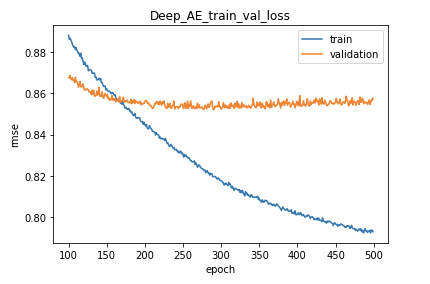
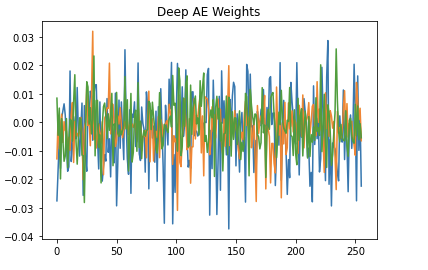
where ri is the actual rating and yi is the reconstructed rating. Mi is a mask function where mi =1 where ri is non-zero else mi=0.

* Two broad autoencoders models: AutoRec, which take the partially observed ratings vector of a user, project it into a low dimensional latent space and then reconstruct back to the output space to predict the missing rating. Deep AE, which has more hidden layers, uses activation functions with non-zero negative parts and unbounded positive parts and uses dropout layers after the latent layer to avoid overfitting and learn robust representations.
* Deep AE model - cite Autorec paper



* Parameters tested:
  + AutoRec:
    - 
    - 
  + Deep AE:
    - 
  + Also tested with the different train matrices with different default rating values as training sets, found that that the matrix with average ratings as default converged faster but with noise and when the model goes deeper, the zero default rating converged faster and with less noise
  + Adding more layers does not help because deeper easier to get overfitted and increasing the regularization parameters will bring the test performance down.
  + So, in our project, using three hidden layers is the best option and [512, 256, 512] and [256, 512, 256] have similar performance but [256, 512, 256] has half the number of parameter so use [256, 512, 256] in further trials to train model faster and reduce overfitting.
* Other method tested:
  + Loading in larger datasets: MovieLens 10M (subsetting 2M then 5M), even wrote training loop to generate subsets of 1M data point at a time and train, but colab keeps timing out
  + Add Noise:
    - Adding additive Gaussian noise and multiplicative dropout noise, tested with noise\_factor = 0.1, 0.2, 0.3, 0.4, 0.5



* + - * adding Gaussian Noise did not improve both models AutoRec and AutoRec because default rating has an impact on the performance, adding noise is changing the default rating
    - Add More Features:
      * concatenated the side information: gender, age and occupation and after transforming to one hot encoding format to the rating user-item matrix
      * Adding the side information has a limited impact on both models AutoRec and Deep AE . This is because the estimates are biased towards users with a lot of ratings, which the dataset already has a lot of information. Moreover, compared with 3952 rating features, 30 side information features will have limited effect. But according when the users have fewer ratings, the side information will have more effect.
  + Groups Specific Recommendations:
    - generate the user-user similarity matrix and cluster them into different groups then train an autoencoder for each group, focus on the age and gender distribution and select a group with most people, age\_group\_2 + gender\_group\_1. This group has 1538 users and train an autoencoder for this group. The test RMSE was only 0.89. But this result may cause by the limited number of users in the training set, as we have 3952 features but only 1538 samples.
    - We extended this experiment by trying to generate different user-user similarity matrix using different demographic subsets with the largest representation, such as those who have occupation group 0 ("other or not specified"),
    - But the recommendation is better for user 2000, who is a male of age group 2 (got 73% of his ratings correctly)
* Best model for AutoEncoder:
  + Best Parameters: layers = [256, 512, 256], dropout = 0.8, activation and last\_activation = 'selu’, regularization alpha of encoder and decoder = 0.001
  + Model Structure:
  + 
  + Scores:
  + 
  + Plots:
  + 
  + 
  + 
* Recommendation Pipeline: predict ratings and recommend unseen movies for user with ID 2000 (got 70% of his ratings correctly)
  + 
  + 

**Word2Vec**

* Overview of Word2Vecs: what are they and how do they work for recommendation task specifically (both collaborative and content filtering)

**Evaluation and Reflections**

* **Challenges we faced:** 
  + understanding 3 different complex DL applications, figuring out how to build them from scratch then tune the, understanding more about recommendation tasks - emphasize that we haven’t learn this in class so we have done a lot of self-learning, research, trial and errors on our own
  + Loading in larger movielens dataset, 10M, 20M movielens datasets → ultimately only able to load in 10M datasets for the transformer
* **Plan:**
  + If
* **Reflections**

**References**

**AutoRec paper: (Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015, May). Autorec: Autoencoders meet collaborative filtering. In Proceedings of the 24th International Conference on World Wide Web (pp. 111-112). ACM.),** [**https://github.com/gtshs2/Autorec**](https://github.com/gtshs2/Autorec)**,** [**http://users.cecs.anu.edu.au/~u5098633/papers/www15.pdf**](http://users.cecs.anu.edu.au/~u5098633/papers/www15.pdf)