ANLY 535: Movie Recommendation System Based on Deep Learning

Aditya Malik, Nihar Garlapati, Gaurav Gade, Divyani Kanawat

Harrisburg University of Science and Technology

April 23rd, 2021

Abstract

A recommender system, or a recommendation system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They play an important role in the vast majority of business which offer services directly to consumers such as ecommerce, social media, online streaming websites etc. Making the right recommendation for the next product, music or movie increases user retention and satisfaction, leading to sales and profit growth. Companies competing for customer loyalty invest on systems that capture and analyses the user's preferences and offer products or services with higher likelihood of purchase.

For this project, our primary goal is to create a movie recommendation system which will provide the user a list of recommended movies based on his or her past usage of the service. Movie recommendation systems have been built using standard machine learning algorithms to provide satisfactory results to consumers. This project aims to improve those results further by using deep learning neural networks to provide more accurate recommendations to consumers and thereby improve their customer experience.

Our model has an accuracy of 75% and performs better than models built using standard machine learning algorithms, with higher precision and recall scores. 'Genre', 'Rating', and 'Release Year' were found to be the most important factors driving a successful movie recommendation. The proposed model will help guide organizations in providing better recommendations to customers and improve their services correspondingly.

Keywords: Machine Learning, Deep Learning, Neural Networks, Movie Recommendations

Introduction

In general, a recommendation system provides suggestions based on the users' preferences, past choices, browsing history etc. Recommendation systems are currently used for a large variety of tasks at large internet companies, including ad click-through rate (CTR) prediction and rankings. Although these methods have had long histories, these approaches have only recently embraced neural networks.

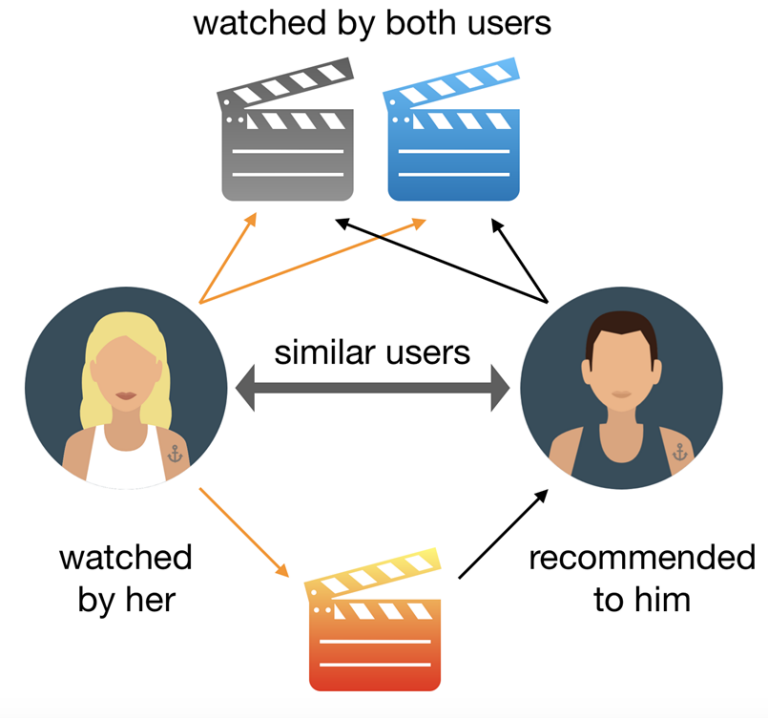
Recommendation systems plays an important role in e-commerce and online streaming services. Netflix, Amazon, Goodreads are some of the examples where recommendation systems are used to better serve the customers. Making the right recommendation for the next product undoubtably increases user retention and satisfaction, leading to increase in sales and profit. Companies competing for customer loyalty invest on systems that capture and analyses the user's preferences and offer products or services with higher likelihood of purchase.

As compared to the models that we have used previously to develop recommendation systems, Deep learning can model the non-linear interaction. Therefore, we have captured complex patterns in user interactions. For example, linear model would look at ratings as the prediction variable. However, with deep learning we explored the ratings as genres, categories of movies, tags etc. We can use embedded layers and drop-outs to increase the accuracy. For this project, we have used the Keras and Pytorch frameworks, regularization, 'Relu' activation, dropouts and Adam optimizer.

As future scope, we can use Deep neural network to include unstructured content information such as text, images, audio.

Related Work

There have been several recommendation systems that have been built for similar projects. In our project, the main challenge was to be identify user interaction with movie ratings, genres and be able to accurately classify movies based on these reviews. In the training phase, we analyzed user activity (positive review, genres watched etc) to build the recommendation model. Then, in the test phase, we select the movies that have not been watched and pass them to the model for that specific user. Then, predict movies that user will like.



Multi-Layer Perceptron Based Recommendation: MLP is a feed-forward neural network with multiple hidden layers between the input layer and the output layer. (Le, 2019). This model has been used in the past to solve both regression and classification problems that were initially introduced for app recommendation in Google Play.

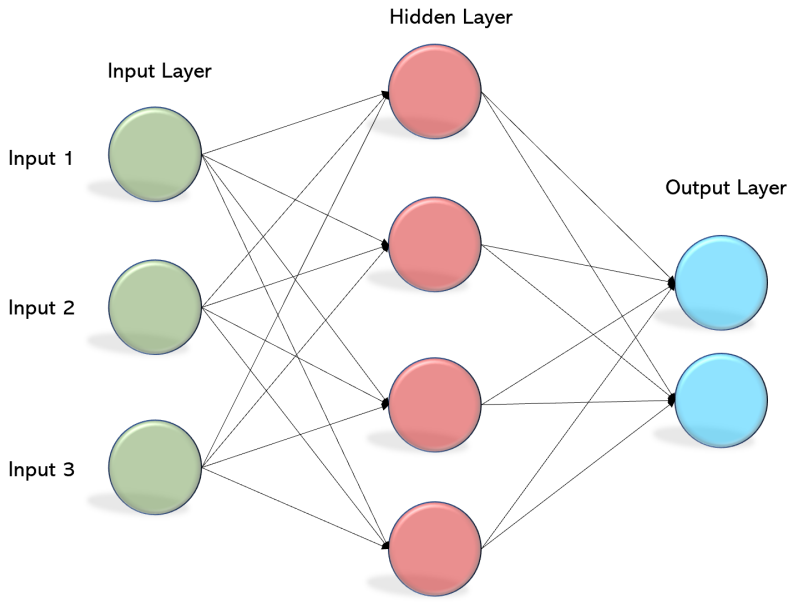


Figure Reference: https://towardsdatascience.com

Autoencoder Based Recommendation is another method which is an unsupervised model attempting to reconstruct its input data in the output layer. It has been used to recommend artwork and other collector items based on user activity.

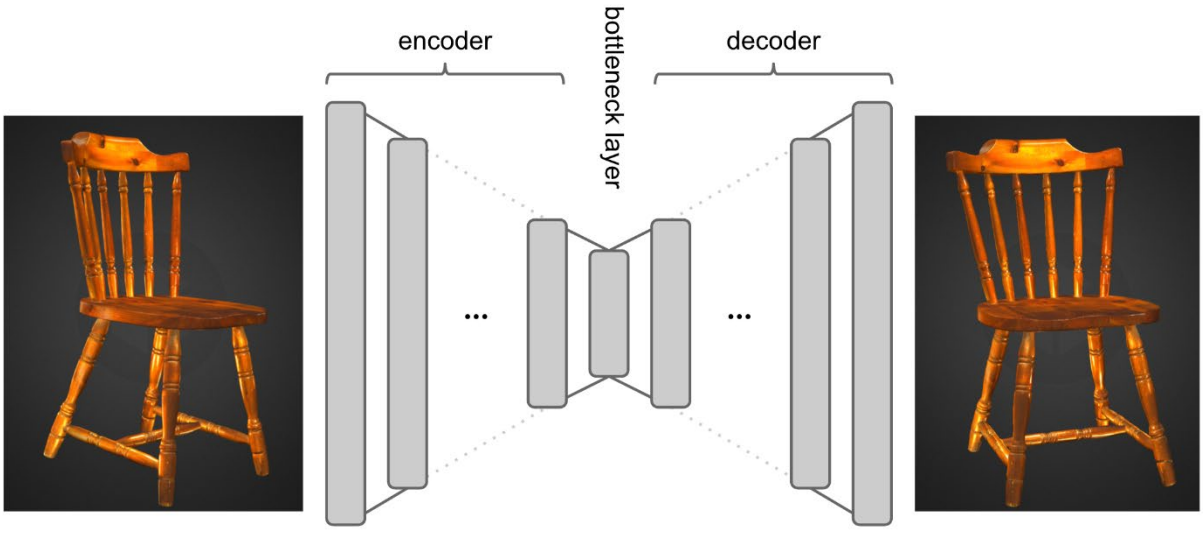


Figure Reference: https://towardsdatascience.com

CNN based recommendation that can capture the global and local features, thus significantly enhancing the model's efficiency and accuracy. It is very powerful in processing unstructured multi-media data. This is effective for visual images as well as for extracting features from data like music signals. It has been used in the past for building music recommendation systems.

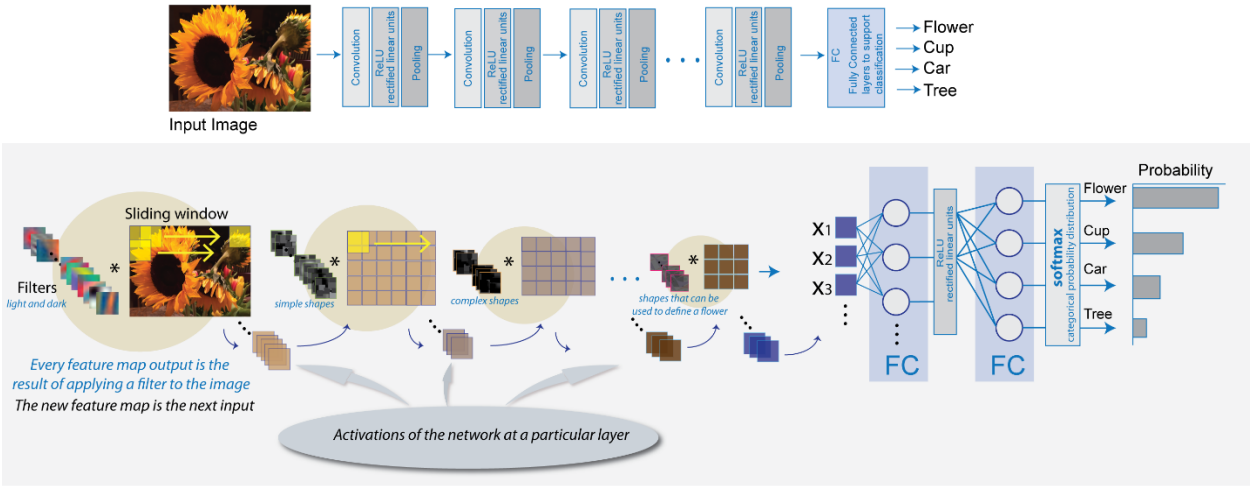


Figure Reference: https://towardsdatascience.com

Data Exploration & Preparation

We will be using the 'MovieLens 20M Dataset' for this project. It contains ratings and free-text tagging activities from MovieLens, a movie recommendation service. It contains 20,000,263 ratings and 465,564 tag applications across 27,278 movies. These data were created by 138,493 users between January 09, 1995 and March 31, 2015. Users were selected at random for inclusion. All selected users had rated at least 20 movies to ensure good data.

The dataset consists of 6 different .csv files which we're using to build our data model. Over these 6 files, there are 19 distinct variables in total. We're considering the 'rating' variable in the 'rating.csv' file as the response variable and the other 18 variables will be used to successfully predict the users' response.

When building this model, we selected a subset of patients to improve processing speed and time. We limited the final dataset to all records for a randomized set of 5,000 users. This amounted to 13,740 movie records and 750,512 movie ratings by these users. We also found that the majority of patients tend to rate movies between 3 and 5 stars. This helped us when building the logic to determine if a user would 'like' the recommended movie.

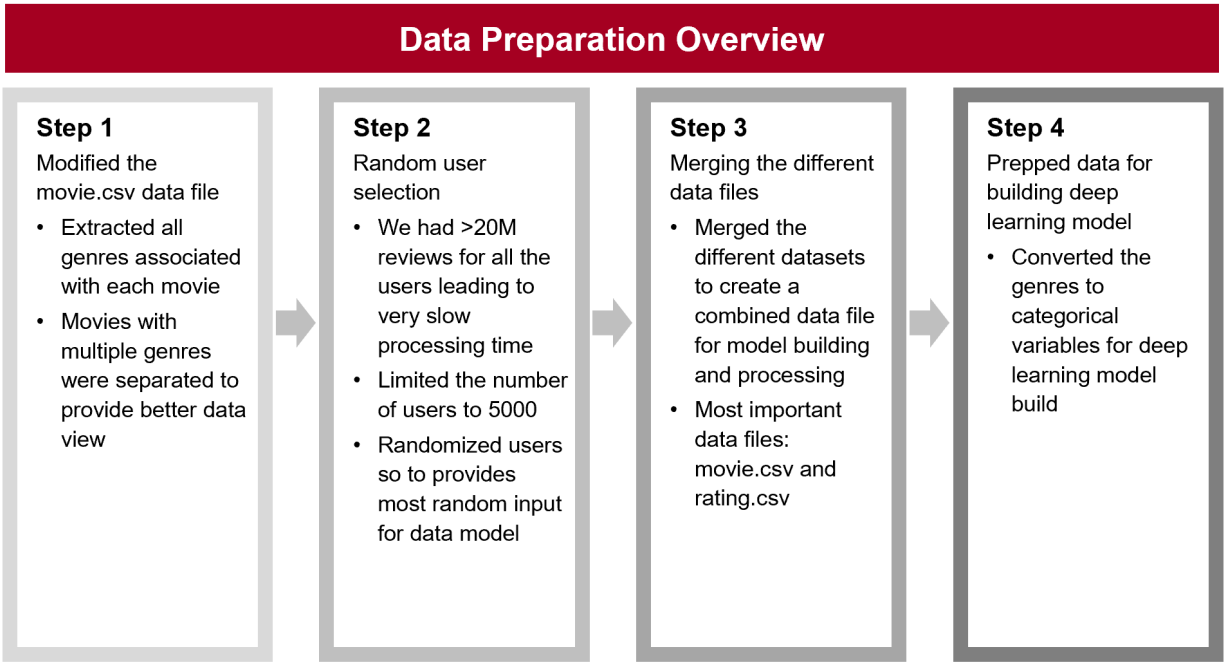
We also looked at how the 'genre' and 'release_year' variables were distributed. Looking at 'genre' (Figure, there was no clear dominant genre with 'Drama', 'Comedy', 'Thriller', 'Action', and 'Romance' having the highest movie records respectively. However, when we look at the 'release_year', we saw that 73% of movies were released during the 21st Century which hinted that the recommendation system might work better for younger users compared to older users.

To create the final dataset, we combined the above 6 data files into a single data file. Each movie was associated with multiple genres. Since genre was an important factor, we separated each genre out into separate columns to input them separately into the model. Finally, we converted the genres to categorical variables so the deep learning model could accurately read each genre.

Once the final dataset was created, we split it into training and test dataset with the training set being 90% and test set being the remaining 10%.

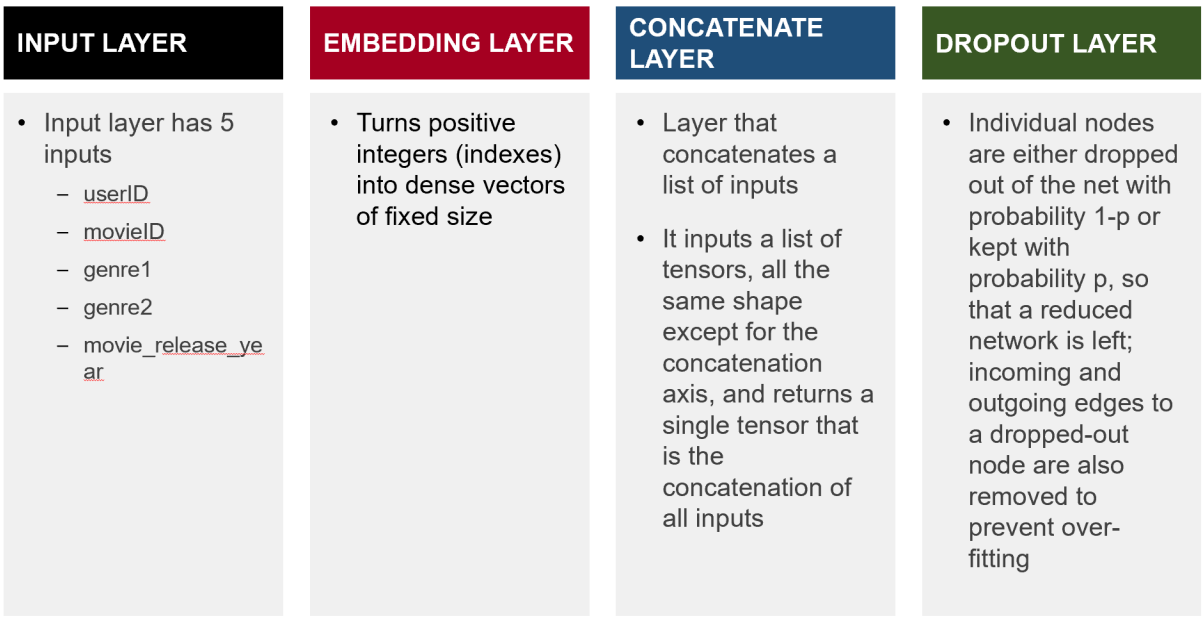
Technical Approach

Our deep learning model used the 'rating' field in the 'rating.csv' file as the base when predicting whether the user would 'like' the move or not. Our thought was 'If we can predict if an individual will like a certain movie or not, we can recommend movies to that individual based on our prediction'. To determine if the user liked a movie or not, we're using the rating field in the data file. Any review less than 3.5 is categorized as a dislike and any review above 3.5 is categorized as a like.



The deep learning neural network model has 4 layers built to provide optimal results. The input layer ingests 5 separate inputs (userID, movieID, genre1, genre2, releaseYear). The embedding layer turns positive integers (indexes) into dense vectors of fixed size. The

concatenate layer concatenates the above list of 5 inputs and inputs a list of tensors, all the same shape except for the concatenation axis, and returns a single tensor that is the concatenation of all inputs. Finally, in the dropout layer, individual nodes are either dropped out of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed to prevent over-fitting.



Model Evaluation

As mentioned above, the final dataset was split into training and test datasets with a 90-10 split. The model was trained on the training set during the modeling phase and was tested against the test dataset during the prediction phase. The model predicts which movies a user would like and also predicts what rating the user would give to a particular movie. Success is

defined on how accurately the model predicts user behavior based on the model recommendations.

Using the above approach, our deep learning neural network movie recommendation system has an accuracy of 75%. As ascertained from Figure 4, based on how the train and test accuracy changed over epochs, we can determine that the model is slightly underfitted. Based on these results, we can use our algorithm on all movies the individual didn't watch and send the user recommendations according to our prediction list.

Conclusion

Our team proposed to build a movie recommendation system using deep learning techniques learnt in the ANLY 535 course. We used the MovieLens 20M dataset for this project, with movie.csv and rating.csv containing the most important factors for predicting the user preferences. Our initial data analysis provided some interesting insights for model building with user 'like' distinction was determined by ratings distribution. The data for our recommendation model was randomly selected and then prepped to best suit the deep learning neural networks. The model we built had 4 layers in total: Input Layer, Embedding Layer, Concatenate Layer, and Dropout Layer. The model had an accuracy of 75%. One issue identified was 'underfitting' which can be resolved by adding more inputs such as movie tags and the crew in the movie. As next steps, the team wants to use the deep learning neural networks movie recommendation model on different data to compare results and try using more inputs to reduce underfitting.

References

GroupLens. (2018, August 15). MovieLens 20M Dataset. Kaggle. https://www.kaggle.com/grouplens/movielens-20m-dataset.

Roy, A. (2020, August 2). Introduction To Recommender Systems- 2: Deep Neural Network Based Recommendation Systems. Medium. https://towardsdatascience.com/introduction-to-recommender-systems-2-deep-neural-network-based-recommendation-systems-4e4484e64746.

Wittenauer, J. (2019, April 29). Deep Learning With Keras: Recommender Systems. Medium. https://medium.com/@jdwittenauer/deep-learning-with-keras-recommender-systems-e7b99cb29929.

Loy, J. (2020, October 19). Deep Learning based Recommender Systems. Medium. https://towardsdatascience.com/deep-learning-based-recommender-systems-3d120201db7e.

Ibrahim, M., Bajwa, I. S., Ul-Amin, R., & Kasi, B. (2019, August 4). A Neural Network-Inspired Approach for Improved and True Movie Recommendations. Computational Intelligence and Neuroscience. https://www.hindawi.com/journals/cin/2019/4589060/.

Movie Recommendations Using the Deep Learning Approach. IEEE Xplore. (n.d.). https://ieeexplore.ieee.org/document/8424686./keywords#keywords.

Tables

Table 1

Model: "functional_3"

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	[(None, 1)]	0	
input_8 (InputLayer)	[(None, 1)]	0	
input_9 (InputLayer)	[(None, 1)]	0	
input_10 (InputLayer)	[(None, 1)]	0	
input_11 (InputLayer)	[(None, 1)]	0	
embedding_5 (Embedding)	(None, 1, 50)	250000	input_7[0][0]
embedding_6 (Embedding)	(None, 1, 50)	673500	input_8[0][0]
embedding_7 (Embedding)	(None, 1, 50)	5000	input_9[0][0]
embedding_8 (Embedding)	(None, 1, 50)	5000	input_10[0][0]
embedding_9 (Embedding)	(None, 1, 50)	125000	input_11[0][0]
reshape_5 (Reshape)	(None, 50)	0	embedding_5[0][0]
reshape_6 (Reshape)	(None, 50)	0	embedding_6[0][0]
reshape_7 (Reshape)	(None, 50)	0	embedding_7[0][0]
reshape_8 (Reshape)	(None, 50)	0	embedding_8[0][0]
reshape_9 (Reshape)	(None, 50)	0	embedding_9[0][0]
concatenate_1 (Concatenate)	(None, 250)	0	reshape_5[0][0] reshape_6[0][0] reshape_7[0][0] reshape_8[0][0] reshape_9[0][0]
dropout_2 (Dropout)	(None, 250)	0	concatenate_1[0][0]
dense_2 (Dense)	(None, 10)	2510	dropout_2[0][0]
activation_2 (Activation)	(None, 10)	0	dense_2[0][0]
dropout_3 (Dropout)	(None, 10)	0	activation_2[0][0]
dense_3 (Dense)	(None, 1)	11	dropout_3[0][0]
activation_3 (Activation)	(None, 1)	0	dense_3[0][0]
lambda_1 (Lambda)	(None, 1)	0	activation_3[0][0]

Total params: 1,061,021
Trainable params: 1,061,021
Non-trainable params: 0

Table 2

Model Accuracy:

```
Epoch 1/10
10555/10555 [==============================] - 70s 7ms/step - loss: 0.1878 - accuracy: 0.7263 - val_loss: 0.1812 - val_accuracy: 0.7355
Epoch 2/10
10555/10555 [==============================] - 70s 7ms/step - loss: 0.1854 - accuracy: 0.7328 - val_loss: 0.1806 - val_accuracy: 0.7370
Epoch 3/10
10555/10555 [==============================] - 70s 7ms/step - loss: 0.1849 - accuracy: 0.7342 - val_loss: 0.1803 - val_accuracy: 0.7376
Epoch 4/10
10555/10555 [==============================] - 73s 7ms/step - loss: 0.1840 - accuracy: 0.7362 - val_loss: 0.1802 - val_accuracy: 0.7392
Epoch 5/10
10555/10555 [==============================] - 74s 7ms/step - loss: 0.1835 - accuracy: 0.7370 - val_loss: 0.1799 - val_accuracy: 0.7390
Epoch 6/10
10555/10555 [==============================] - 84s 8ms/step - loss: 0.1830 - accuracy: 0.7377 - val_loss: 0.1794 - val_accuracy: 0.7382
Epoch 7/10
10555/10555 [==============================] - 80s 8ms/step - loss: 0.1826 - accuracy: 0.7387 - val_loss: 0.1793 - val_accuracy: 0.7405
Epoch 8/10
10555/10555 [==============================] - 76s 7ms/step - loss: 0.1823 - accuracy: 0.7388 - val_loss: 0.1792 - val_accuracy: 0.7395
Epoch 9/10
10555/10555 [==============================] - 72s 7ms/step - loss: 0.1820 - accuracy: 0.7396 - val_loss: 0.1795 - val_accuracy: 0.7396
Epoch 10/10
10555/10555 [==============================] - 74s 7ms/step - loss: 0.1822 - accuracy: 0.7395 - val_loss: 0.1795 - val_accuracy: 0.7409
2346/2346 [==============================] - 4s 2ms/step - loss: 0.1795 - accuracy: 0.7409
Accuracy: 74.09%
```

Exploratory Data Analysis

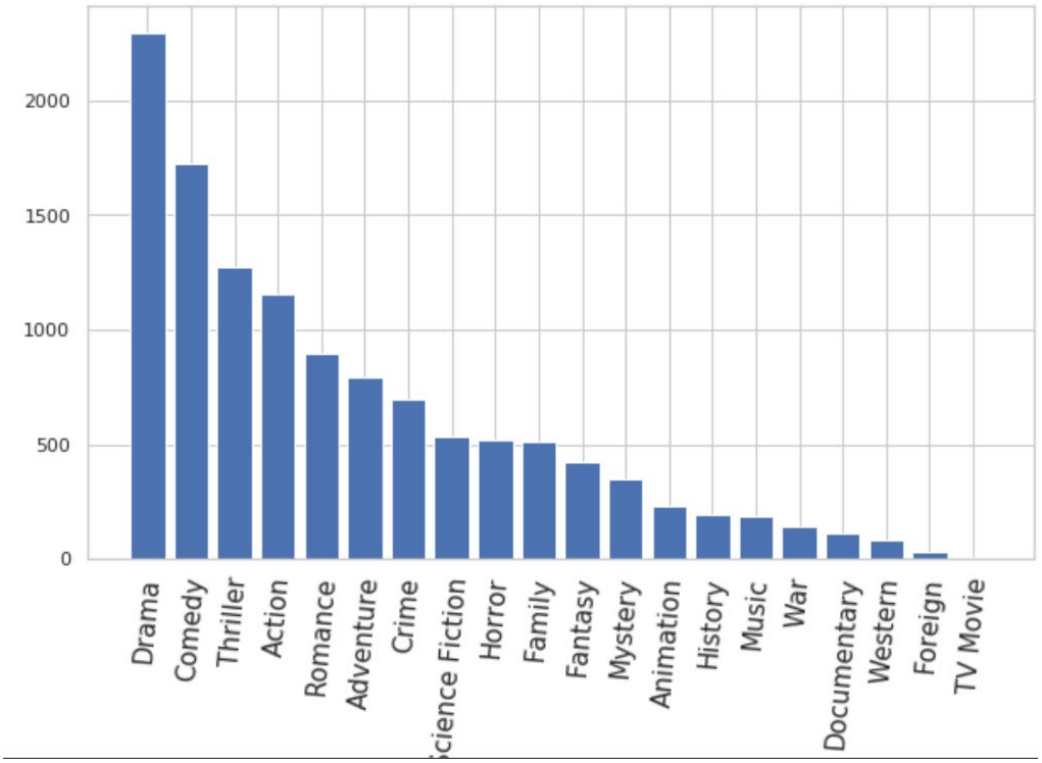


Figure 1: 'Genre' Distribution

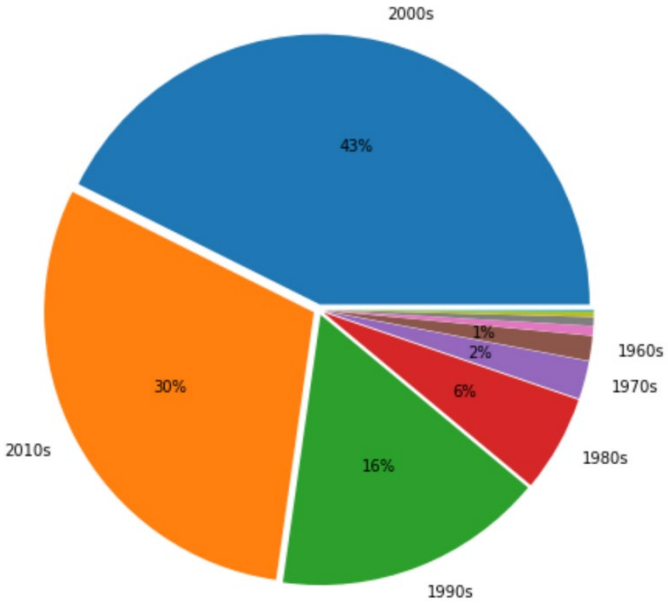


Figure 2: 'Release Year' Distribution

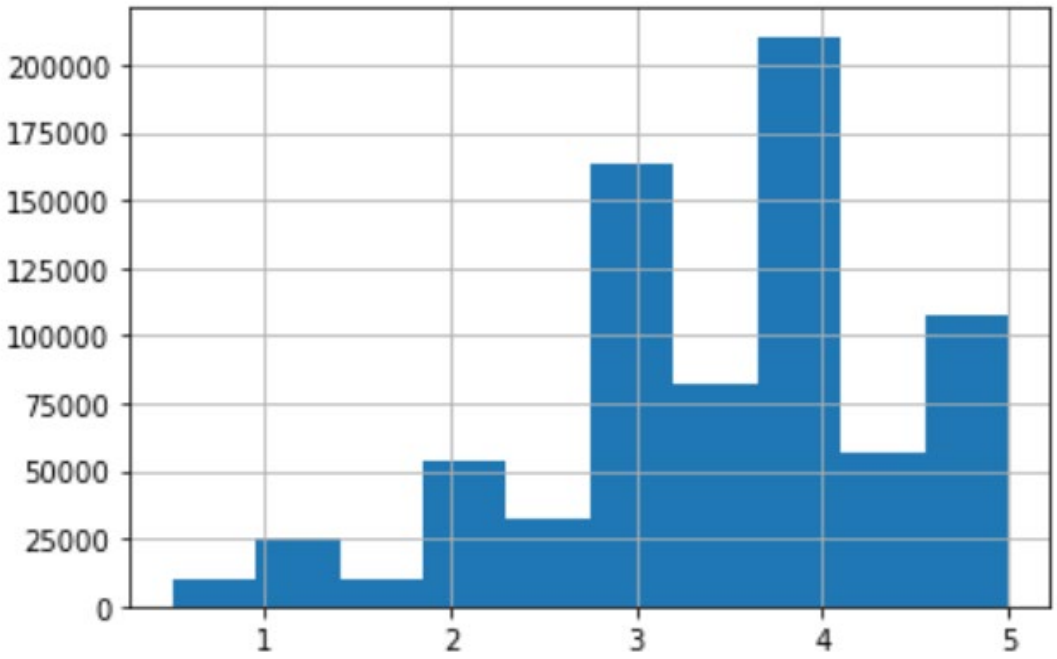


Figure 3: ‘Movie Rating’ Distribution

Model Performance Comparison

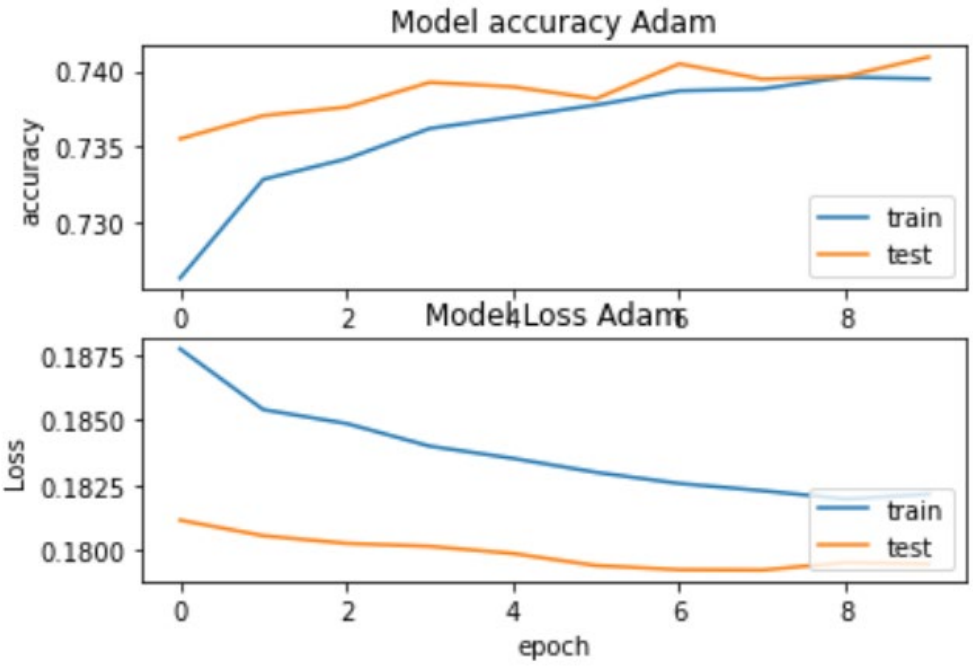


Figure 4: Change of training and test accuracy over multiple epochs