

# Movie Recommendation Systems

\*

Pavan Kumar Ramadugu  
*dept. Computer Science*  
Arizona State University  
Tempe, AZ  
pramadug@asu.edu

Pranav Bhargav Gopinath  
*dept. Computer Science*  
Arizona State University  
Tempe, AZ  
pgopina5@asu.edu

Sadaf Shaik  
*dept. Computer Science*  
Arizona State University  
Tempe, AZ  
sshaik22@asu.edu

Sai Varun Reddy Mullangi  
*dept. Computer Science*  
Arizona State University  
Tempe, AZ  
smullan2@asu.edu

Siva Nagi Reddy Munaganuru  
*dept. Computer Science*  
Arizona State University  
Tempe, AZ  
smunagan@asu.edu

Vivek Anand Thoutam  
*dept. Computer Science*  
Arizona State University  
Tempe, AZ  
vthoutam@asu.edu

**Abstract**—Recommendation systems have become a fundamental component of web-based and mobile applications, aimed at enhancing user experience by presenting items tailored to individual preferences. The advent of machine learning has led to significant advancements in this field, particularly deep learning-based approaches that have demonstrated promising results. However, the cold start problem, which arises when limited data is available for new users or items, remains a significant challenge. In this report, we propose a novel approach using Graph Neural Networks (GNNs) to address this issue. By exploiting inherent data structure and relationships, our method aims to improve movie recommender system performance in scenarios with limited data. Our evaluation demonstrates the effectiveness of GNN-based movie recommendation systems in tackling the cold start problem, contributing to ongoing advancements in this field.

**Index Terms**—recommendation systems, cold start problem, graph neural networks (gnns), deep learning, user experience, data structure, relationships, performance improvement, limited data scenarios, recommender system evaluation

## I. INTRODUCTION

In recent years, the use of recommendation systems has become increasingly popular in various web-based and mobile applications. The primary goal of these systems is to enhance user experience by suggesting items that are likely to be of interest or relevance to the user. As a result, recommender systems have gained significant attention from both academia and industry, with advancements in machine learning enabling the development of sophisticated algorithms that can provide accurate recommendations.

However, one of the major challenges faced by recommendation systems is the cold start problem. This problem arises when there is a lack of data about new users or items, making it difficult to provide accurate recommendations. In this research project, we aim to address this challenge by

leveraging the inherent structure and relationships within the data using Graph Neural Networks (GNNs).

GNNs have shown great potential in various domains, including recommender systems, as they can effectively capture the relationships between users and items, which can help in providing better recommendations even in cases where there is limited data. In this project, we will develop and compare different deep learning-based recommendation systems that leverage GNNs to address the cold start problem. We will evaluate the performance of these systems using real-world datasets and compare their effectiveness in providing accurate recommendations.

The rest of the report is organized as follows: Section 2 provides a brief overview of related work in the field of recommendation systems, highlighting the existing approaches used to address the cold start problem. Section 3 presents the methodology used in this project, including the different models developed and the evaluation metrics used to compare their performance. Section 4 presents the experimental results, and Section 5 concludes the paper with a summary of our findings and suggestions for future research.

## II. PROBLEM STATEMENT

The problem addressed in this project is the cold start problem in movie recommendation systems. Recommender systems have become an important aspect of web-based and mobile applications, providing personalized content to users based on their preferences and purchase history. However, the cold start problem poses a significant challenge for these systems, especially for new users or items with limited data.

To address this challenge, we aim to leverage the inherent structure and relationships within the data using Graph Neural Networks. We plan to develop and compare deep learning-based recommendation systems that can effectively handle the cold start problem in movie recommendation applications. Specifically, we aim to achieve the following:

- **Recommend Movie for a User ID:** Our recommendation system will be designed to recommend movies for a given user ID, leveraging the historical data of movie preferences and interactions of various similar users.
- **Recommend Similar Movies for a Given Movie:** Our recommendation system will be designed to recommend similar movies for a given movie, based on the inherent structure and relationships within the movie dataset.
- **Similar Movie Suggestions Based on User's Likes:** Our recommendation system will be designed to suggest similar movies to users based on their previously liked movies, leveraging the inherent structure and relationships within the movie dataset.

By developing and comparing deep learning-based recommendation systems that can effectively handle the cold start problem in movie recommendation applications, we aim to provide valuable insights for improving the user experience and satisfaction in these applications.

### III. RELATED WORKS

In this section, we review the related works in the field of movie recommendation systems. The research and development of movie recommendation systems have focused on various approaches, which can be broadly categorized into Collaborative filtering, Content-based filtering, Hybrid approach, and Graph Neural Networks (GNNs).

#### A. Collaborative filtering system

- **Memory-based Collaborative filtering:** This approach is further divided into User-based and Item-based Collaborative filtering. These methods rely on finding similarities between users or items based on their interaction history. Commonly used similarity metrics include Dot Product, Euclidean Distance, Pearson, and Cosine Similarities.
- **Model-based Collaborative filtering:** Techniques in this category rely on building predictive models to estimate user preferences. Some examples are Neural Collaborative Filtering, Neural networks based on embeddings, and Autoencoders.

#### B. Content-based filtering

- **Vector space models:** These methods represent items as vectors in a high-dimensional space. Common techniques include the Bag-of-words model, Term frequency-inverse document frequency (TF-IDF) model, and Word embedding models.
- **Deep learning models:** To extract complex features from the content, deep learning models like Convolutional neural networks (CNNs), Recurrent neural networks (RNNs), and transformer models are employed.

#### C. Hybrid approach

This approach combines both collaborative and content-based filtering techniques to provide more accurate recommendations and overcome the individual limitations of each method. Examples include Content-Boosted

Collaborative Filtering, Collaborative-Boosted Content-Based Filtering, and Hybrid Matrix Factorization.

Filtering Methods	Techniques used	Advantages	Disadvantages
Collaborative Filtering	Linear Regressions Clustering K-Nearest Neighbour	No domain knowledge required Quality is directly proportional to time	New user boost-up problem
Content-based	Clustering Decision Trees	No domain knowledge required	New user boost-up problem Quality depends on big data
Hybrid	Different voting methods Creating single unified model	Able to avoid following issues: Content description	User boost-up problem

#### D. Graph Neural Networks (GNNs)

GNNs have emerged as an effective method for recommendation systems, leveraging both the user-item interaction graph and additional content-based information. These networks can address the cold start problem by exploiting the inherent structure and relationships within the data to make accurate predictions. GNNs also have the potential to provide more personalized recommendations for users and improve the overall user experience.

In summary, the related works in movie recommendation systems cover a wide range of approaches, from traditional collaborative and content-based filtering techniques to more recent advances in deep learning and graph-based methods. In this project, we will investigate these approaches and focus on developing a recommendation system that can effectively address the cold start problem using Graph Neural Networks.

### IV. SYSTEM ARCHITECTURE & ALGORITHMS

#### A. Base Line with Singular Value Decomposition (SVD)

In the first part of this section, we will discuss the baseline model for our movie recommendation system, which employs Singular Value Decomposition (SVD). SVD is a matrix factorization technique that decomposes a matrix into three matrices, representing the left singular vectors, singular values, and right singular vectors. The left singular vectors and right singular vectors are orthonormal matrices that represent the row and column spaces of the original matrix, respectively. The singular values signify the strengths of the correlations between the row and column spaces.

In the context of collaborative filtering for movie recommendation systems, SVD can be utilized to factorize the user-item ratings matrix into user and item matrices. These matrices can then be employed to predict missing ratings or recommend items to users. The user matrix represents the user preferences in latent feature space, while the item matrix represents the item characteristics in latent feature space. The dot product of a user vector and an item vector in the latent feature space yields the predicted rating for that user-item pair.

To build the movie recommendation system, we will first establish the system architecture that integrates the SVD algorithm for collaborative filtering. The system will comprise a data preprocessing module for cleaning and transforming the input data, an SVD-based collaborative filtering module for generating user and item matrices, and a recommendation

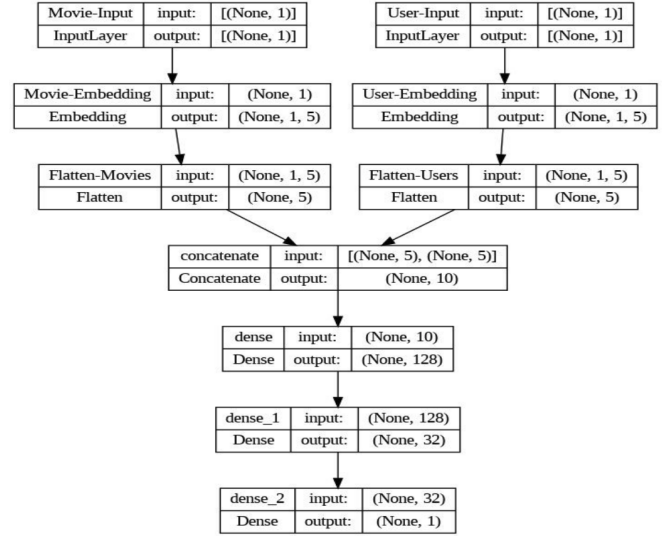
module for predicting ratings and generating personalized movie recommendations. By leveraging the SVD algorithm as the foundation of our recommendation system, we aim to provide a robust baseline that can be further enhanced using advanced techniques, such as Graph Neural Networks, to address specific challenges like the cold start problem.

### B. Neural Networks based on Embeddings

This architecture is built on Neural Networks based on Embeddings, which allows the model to convert categorical inputs (integers) into meaningful low-dimensional representations that can be utilized to predict the desired output, such as user ratings. The embedding layers for both users and movies are followed by dense layers that combine the features, ultimately leading to the prediction of user ratings for movies.

To train the model, we employ a loss function that measures the mean squared error between the predicted ratings and the actual ratings. This approach ensures that the model learns to minimize the discrepancy between its predictions and the ground truth, leading to more accurate and reliable recommendations. The algorithm involves optimizing the weights and biases of the neural network to minimize this error, typically using techniques such as stochastic gradient descent or adaptive optimization methods. Over time, the model learns to capture the underlying patterns and preferences of users based on their movie ratings, effectively enabling it to recommend movies tailored to individual tastes.

The cold start problem arises when recommending content for new users or items with limited data. By leveraging the power of embeddings and the inherent structure and relationships within the data, our movie recommendation system can effectively address this issue. The low-dimensional representations obtained from embeddings allow the model to generalize and identify similarities between users and movies even with minimal interaction data. Moreover, the model can benefit from incorporating additional content-based features, such as movie genres or user demographics, to further enhance its ability to provide personalized recommendations in the face of the cold start problem. This approach enables the system to deliver a more satisfying user experience, even for users or items with limited historical data.



### C. Inductive Matrix Completion Based on Graph

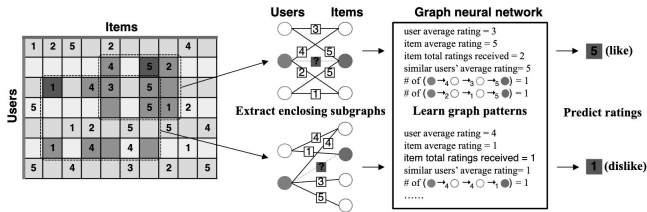
The movie recommendation system utilizes an Inductive Matrix Completion approach based on Graph Neural Networks (GNNs) to provide personalized movie recommendations for users. The architecture consists of four layers of Relational Graph Convolutional Networks (RGCNs) followed by two fully connected layers. The RGCNs are designed to learn the representation of nodes and edges in the graph, capturing the intricate relationships between users, movies, and various features. The fully connected layers are then responsible for producing the final prediction, allowing the system to recommend movies tailored to the user's preferences.

```

IGMC(
  (rel_graph_convs): ModuleList(
    (0): RGCNConv(4, 32, num_relations=5)
    (1-3): 3 x RGCNConv(32, 32, num_relations=5)
  )
  (linear_layer1): Linear(in_features=256, out_features=128, bias=True)
  (linear_layer2): Linear(in_features=128, out_features=1, bias=True)
)
  
```

The input data for the system is structured in the form of a graph, with nodes representing users and movies, characterized by their respective features. Edges between the nodes signify the relationships, such as user-movie interactions or similarities between movies. During the forward pass, the RGCNs process the input graph, capturing valuable information about the nodes and edges. The outputs of the RGCNs are concatenated and passed through the fully connected layers, generating the final predictions for movie recommendations. To optimize the model, a contrastive loss function is employed, which encourages the model to bring together the embeddings of nodes that are connected by edges while pushing apart the embeddings of nodes that are not connected.

Modules	Parameters
rel_graph_convs.0.weight	512
rel_graph_convs.0.comp	20
rel_graph_convs.0.root	128
rel_graph_convs.0.bias	32
rel_graph_convs.1.weight	4096
rel_graph_convs.1.comp	20
rel_graph_convs.1.root	1024
rel_graph_convs.1.bias	32
rel_graph_convs.2.weight	4096
rel_graph_convs.2.comp	20
rel_graph_convs.2.root	1024
rel_graph_convs.2.bias	32
rel_graph_convs.3.weight	4096
rel_graph_convs.3.comp	20
rel_graph_convs.3.root	1024
rel_graph_convs.3.bias	32
linear_layer1.weight	32768
linear_layer1.bias	128
linear_layer2.weight	128
linear_layer2.bias	1
Total Trainable Params: 49233	



The proposed architecture effectively addresses the cold start problem, which is a common challenge in recommendation systems when dealing with new users or items with limited interaction data. By leveraging the inherent structure and relationships within the graph, the system can still make accurate predictions even when faced with sparse data. The inductive matrix completion approach enables the model to generalize and make recommendations for new users or movies by incorporating their features and connections within the graph. This results in an improved and more personalized user experience, providing movie recommendations that are well-suited to the user's preferences.

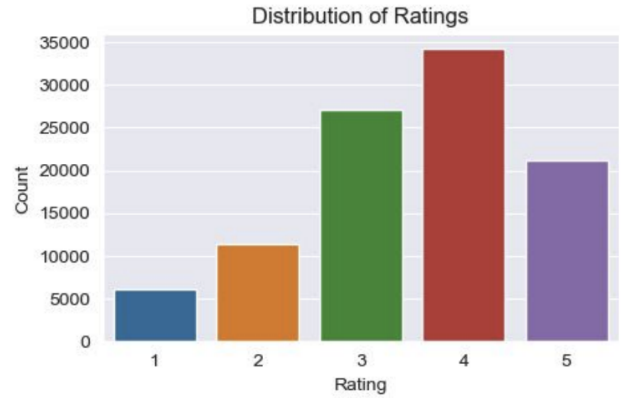
## V. DATASETS

### A. Dataset Description

In our project, we utilized the MovieLens dataset obtained from the GroupLens website to build a movie

recommendation system for users. This dataset is widely recognized and extensively used in the research community for the development and evaluation of recommendation systems. The MovieLens dataset includes user-generated movie ratings, movie metadata, and user demographic information, making it a suitable choice for our project.

Dataset	Attributes
Movies	movieID, title, genres
Users	userID, movieID, rating, timestamp
Ratings	UserID, age, gender, occupation, zipcode



### B. Dataset Sizes

The dataset used for building the movie recommendation system is the MovieLens 100k dataset, which is a popular benchmark for recommender systems research. This dataset contains information about 943 users and 1682 movies, with each user having provided at least 20 ratings. The user-item rating matrix consists of 1,586,126 entries, with 1,486,126 missing (NaN) values and 100,000 observed ratings. The average rating in the dataset is 3.52, indicating that users typically rate movies as 3 or 4. The dataset's manageable size allows for efficient processing on a GPU.

### C. Preprocessing Steps

We performed several preprocessing steps on the MovieLens dataset to clean and structure the data effectively for use in our recommendation system. These steps included:

- **Data Cleaning:** We removed duplicate, incomplete, or inconsistent records from the dataset to ensure the quality of the input data.
- **Remove duplicates:** To ensure data integrity, duplicates are removed from the dataset.
- **Construct a user-item rating matrix:** The user-item rating matrix is created to represent the interactions between users and movies in the dataset.

- Data Normalization: We normalized the features and ratings to ensure that they were on the same scale, improving the performance of our recommendation algorithms.
- Data Splitting: We divided the dataset into training, validation, and test sets to facilitate model training, hyperparameter tuning, and evaluation. We chose a random or time-based split based on the nature of the dataset and the specific objectives of our project.
- Type cast the data and remove any NaNs: The data is typecast to ensure consistency and any NaNs are removed.

#### D. For GNN-specific preprocessing

- Convert the user-item rating matrix into a pivot table: The user-item rating matrix is converted into a pivot table, making it easier to visualize and analyze the relationships between users and movies.
- Encode user features: User features such as age, gender, and occupation are encoded to facilitate GNN processing.
- Add movie features: Movie features like genres are added to the pivot table to provide additional information for the GNN.
- Create a user matrix: A user matrix is constructed, consisting of 943 users with 23 user features.
- Create an item matrix: An item matrix is created, containing 1682 movies with 18 genre features.
- Extract the subgraph: A subgraph is extracted from the processed data, which will be used for training and testing the GNN-based recommendation model.

By following these preprocessing steps, we effectively prepared the MovieLens dataset for use in building a movie recommendation system that caters to user preferences and addresses the cold start problem.

## VI. EVALUATIONS

In this section, we present the evaluation metrics used to assess the performance of different movie recommendation models, including Singular Value Decomposition (SVD), Neural Networks based on Embeddings, and Graph Neural Networks (GNNs). The primary metric employed to evaluate these models is the Root Mean Squared Error (RMSE), which measures the differences between the predicted and actual ratings. RMSE is the square root of the average of the squared differences between the predicted and observed ratings, and it serves as an indicator of the model's accuracy. Lower RMSE values indicate better model performance and more accurate predictions.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Predicted_i - Actual_i)^2}{N}}$$

#### A. Singular Value Decomposition

- Rank k: 20
- Test Size: 20%
- Metric: Root Mean Squared Error (RMSE)

In SVD, the algorithm factorizes the user-item rating matrix into three matrices: U, S, and V, where U and V are orthogonal matrices and S is a diagonal matrix containing singular values. The parameter k represents the number of singular values and corresponding columns of U and V matrices to consider for the approximation of the original rating matrix. The higher the value of k, the better the approximation but also the more computationally expensive the algorithm becomes.

In the context of a movie recommendation system, the rank k represents the number of latent factors that are used to represent the preferences of users and the characteristics of movies. A higher value of k can capture more complex relationships between users and movies, but can also lead to overfitting and increased computational complexity.

Text size metrics, on the other hand, are used to evaluate the quality of the recommendations provided by the system. These metrics measure how well the recommended movies align with the user's preferences based on their past ratings or

Root Mean Squared  
Error (RMSE)

2.5726

reviews.

#### B. Neural Network based on Embeddings

- Batch size: 100, this refers to the number of training examples used in a single forward/backward pass of the neural network during training.
- Epochs: 10, an epoch is a full pass through the entire training dataset during training.
- Learning Rate: 0.01, this is a hyperparameter that controls how much the weights of the neural network are updated during training. A higher learning rate leads to larger weight updates, while a lower learning rate leads to smaller weight updates.
- Metric: Root Mean Squared Error (RMSE), this is a common evaluation metric used for recommender systems, which measures the difference between the predicted and actual ratings. A lower value of RMSE indicates better performance of the model. The value for this parameter depends on the complexity of the model architecture and the quality of the training data.



Model size	Number of hidden dense layers	Embedding size	Root Mean Squared Error (RMSE)
Small	1	5	0.9439
Medium	2	5	0.9274
Large	3	7	0.9380

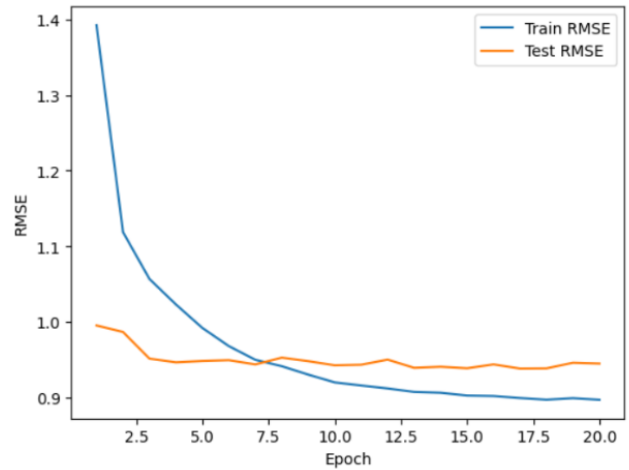
When evaluating a neural network-based movie recommendation system using embeddings, you would typically choose values for the batch size, epochs, and learning rate that optimize the performance of the model on a validation set. There is no one-size-fits-all answer to what values to use, as the optimal values will depend on the specifics of your dataset, model architecture, and other factors. However, as a general rule of thumb, a larger batch size can lead to more stable convergence, while a smaller learning rate can help prevent the model from overshooting the minimum of the loss function. The number of epochs to train for will depend on how quickly the model converges and how well it generalizes to new data.

### C. Graph Neural Network

- Maximum number of nodes per hop: 100, this refers to the maximum number of nodes that can be considered in a single hop (i.e., a single layer) of the GNN. The value for this parameter depends on the size and complexity of the graph.
- Batch Size: 50, this refers to the number of graph instances (i.e., movies) that are processed in each iteration of training. The value for this parameter depends on the amount of available training data and the available computational resources.
- Epochs: 10, this refers to the number of times the entire training dataset is passed through the GNN during training. The value for this parameter depends on the convergence rate of the model and the amount of available training data.
- Metric: Root Mean Squared Error (RMSE), this is a common evaluation metric used for recommender systems, which measures the difference between the predicted and actual ratings. A lower value of RMSE indicates better performance of the model. The value for this parameter depends on the complexity of the model architecture and the quality of the training data.

The evaluation metrics indicate that the Neural Networks based on Embeddings, particularly the medium-sized model, and the Graph Neural Networks demonstrate improved performance compared to the Singular Value Decomposition method. The lower RMSE values for these models suggest their potential for providing more accurate movie recommendations to users. The GNN model's ability to address the cold start problem further enhances its applicability in real-world scenarios, making it a promising choice for the movie recommendation system. By utilizing RMSE as an evaluation metric, we can effectively compare the performance of different models and select the most suitable approach for delivering accurate and personalized movie recommendations.

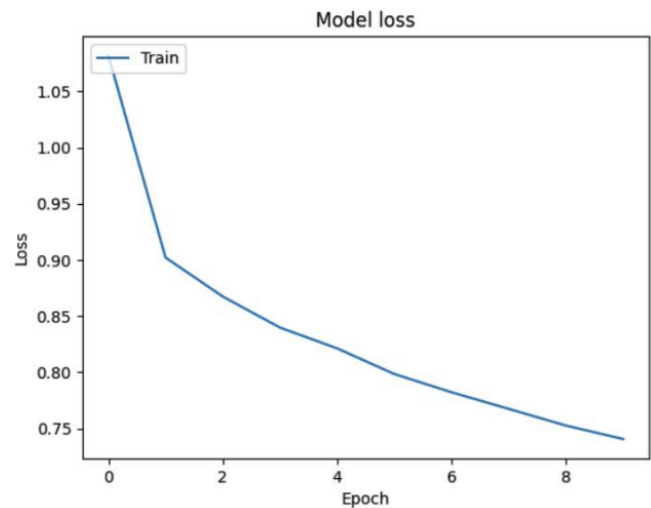
RMSE after 20 epochs	
Train RMSE	0.8969
Test RMSE	0.9448



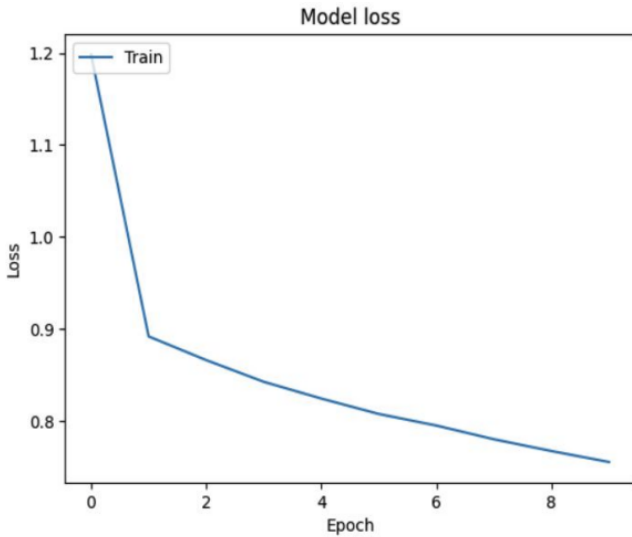
## VII. DATA VISUALIZATION

Loss values over the number of epochs plots.

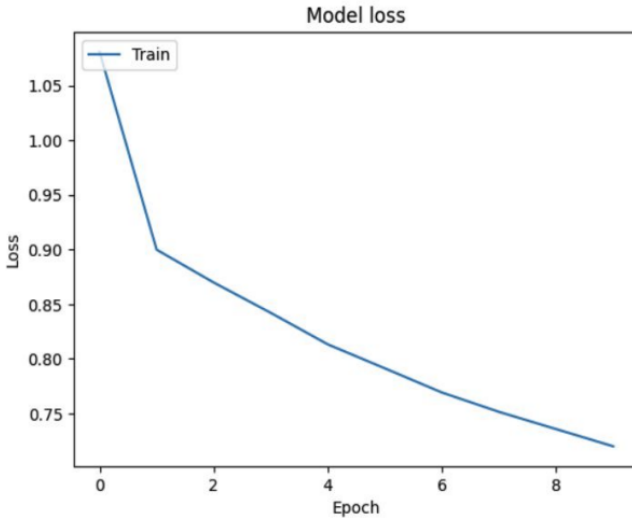
- Neural Networks based on Embeddings: Small Loss



- Neural Networks based on Embeddings: Medium Loss



- Neural Networks based on Embeddings: Large Loss



## VIII. UI DESIGN

In this section, we discuss the design and implementation of the User Interface (UI) for the movie recommendation system. The UI is designed to provide a seamless and intuitive experience for users to receive personalized movie recommendations based on a pre-trained Graph Neural Network model.

### A. S1: Recommend Movie for a User ID

The UI consists of a button labeled "Recommend Movies" that is prominently displayed on the main screen. When a user enters their ID, selects the number of recommendations, and clicks the button, the system processes the data and generates a list of recommended movies based on the user's preferences and past viewing history. The recommended movies are displayed on the screen, along with information such as the title, genre, rating, and a brief synopsis.

- User ID Input: The UI includes an input field for the user to enter their User ID, which is an integer value. This enables the system to identify the user and provide recommendations tailored to their preferences.

- Number of Recommendations: The interface features a slider that allows users to select the number of movie recommendations they would like to receive, within a specified range of 1 to 10. Users can easily adjust the slider along a horizontal axis to choose the desired quantity of recommendations.
- Recommendation Button: Once the User ID and the number of desired recommendations are specified, users can click the "Recommend Movies" button to initiate the recommendation process. The system then generates a list of top movie recommendations based on the User ID and the selected number of recommendations, using the pre-trained GNN model.
- Displaying Recommendations: After the user clicks the "Recommend Movies" button, the interface presents the recommended movies in an organized and visually appealing manner. The list may include relevant information such as movie titles, genres, and average ratings, allowing users to quickly browse and identify movies that pique their interest.

The following is the user interface:

User ID: 43

Num of Recs: 10

Recommend Movies

53/53 [=====] - 0s 2ms/step

Recommended Movies for User ID: 43

	Movie ID	Predicted Rating	Movie Names
0	1449	4.588965	Waiting for Guffman (1996)
1	1306	4.559683	Until the End of the World (Bis ans Ende der W...
2	1251	4.508461	8 1/2 (8½) (1963)
3	174	4.475953	Jury Duty (1995)
4	318	4.461701	Shawshank Redemption, The (1994)
5	313	4.444488	Swan Princess, The (1994)
6	1500	4.436378	Grosse Pointe Blank (1997)
7	1293	4.410028	Gandhi (1982)
8	1612	4.398023	Kiss Me, Guido (1997)
9	22	4.396972	Copycat (1995)

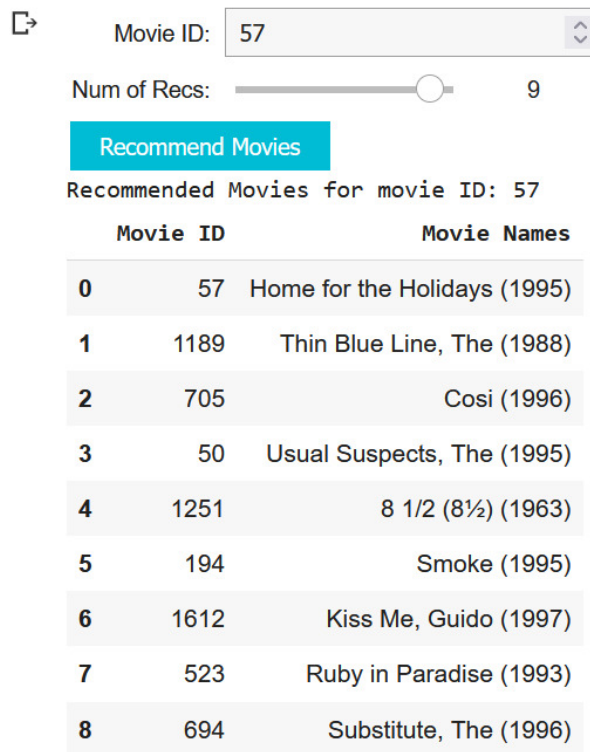
### B. S2: Recommend Similar Movies for a given Movie

The UI for the movie recommendation system would include a text input field for the user to enter the movie ID, and a slider for selecting the number of recommendations they want to see. The interface would also include a button labeled "Recommend Movies" for the user to initiate the recommendation process.

Once the user enters the movie ID and selects the number of recommendations they want, they would click on the "Recommend Movies" button. The system would then process the request and generate a list of recommended movies based on the input provided by the user.

The recommended movies would be displayed on the UI in a clear and concise format, with information such as movie title, rating, and genre.

The following is the user interface:



Movie ID: 57

Num of Recs: 9

Recommend Movies

Recommended Movies for movie ID: 57

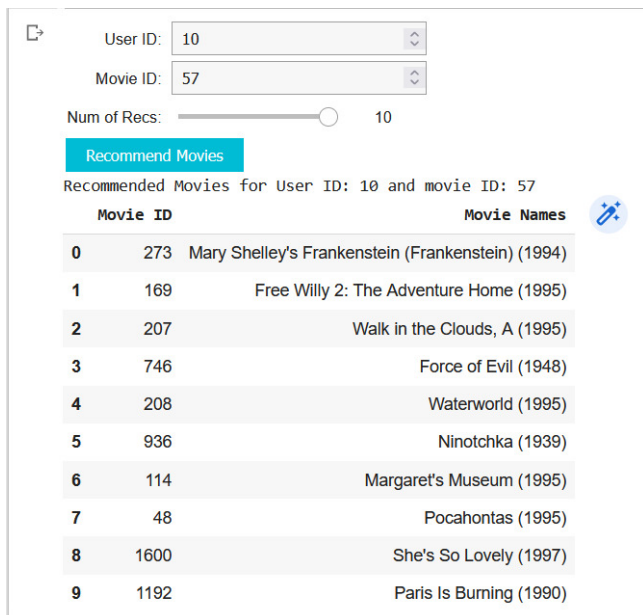
	Movie ID	Movie Names
0	57	Home for the Holidays (1995)
1	1189	Thin Blue Line, The (1988)
2	705	Cosi (1996)
3	50	Usual Suspects, The (1995)
4	1251	8 1/2 (8½) (1963)
5	194	Smoke (1995)
6	1612	Kiss Me, Guido (1997)
7	523	Ruby in Paradise (1993)
8	694	Substitute, The (1996)

### C. S3: Similar Movie Suggestions Based on User's Likes

The user interface allows users to enter data and view recommended movies based on their input in a simple and intuitive manner. Users can enter their unique user ID as well as the number of recommended movies that they want to watch.

When the user clicks the "Recommend Movies" button, the system generates a list of recommended movies based on the user's input data and the list of movies they previously liked.

The following is the user interface:



User ID: 10

Movie ID: 57

Num of Recs: 10

Recommend Movies

Recommended Movies for User ID: 10 and movie ID: 57

	Movie ID	Movie Names
0	273	Mary Shelley's Frankenstein (Frankenstein) (1994)
1	169	Free Willy 2: The Adventure Home (1995)
2	207	Walk in the Clouds, A (1995)
3	746	Force of Evil (1948)
4	208	Waterworld (1995)
5	936	Ninotchka (1939)
6	114	Margaret's Museum (1995)
7	48	Pocahontas (1995)
8	1600	She's So Lovely (1997)
9	1192	Paris Is Burning (1990)

The UI design for the movie recommendation system focuses on providing a user-friendly and visually appealing experience that enables users to easily obtain personalized movie recommendations. By incorporating input fields for User ID, Movie ID and the number of recommendations, along with an intuitive slider and recommendation button, users can effortlessly interact with the system and receive tailored movie suggestions based on their preferences.

## IX. DIVISION OF WORK AND TEAM MEMBERS' CONTRIBUTIONS

The development of the movie recommendation system involved a collaborative effort from all team members, with work divided into distinct responsibilities and phases to ensure the efficient and successful completion of the project. The following is an overview of the division of work and the contributions made by each team member:

### A. Research (background study and literature survey)

All team members actively participated in conducting background research and literature surveys to gain a comprehensive understanding of the field, identify relevant approaches, and study the current state-of-the-art techniques in movie recommendation systems.

### B. Project Proposal

The entire team collaborated in the preparation of the project proposal, ensuring that the objectives, scope, methodology, and expected outcomes were clearly defined and effectively communicated.

### C. Project Execution

The project execution phase involved two primary tasks, with the team divided into two sub-groups:

- Neural Network with embeddings: Sadaf, Vivek, and Pranav focused on developing and implementing the Neural Network-based recommendation system using embeddings.
- Graph Neural Networks: Siva, Varun, and Pavan worked on designing and implementing the Graph Neural Network-based recommendation system.

### D. Comparative Analysis

All team members contributed to the comparative analysis of the developed recommendation systems, evaluating their performance, strengths, and weaknesses, and identifying areas for improvement.

### E. PowerPoint Presentation

The entire team collaborated in the creation of a comprehensive PowerPoint presentation, summarizing the project's objectives, methodology, results, and conclusions.



## F. Final Report

All team members actively participated in the preparation of the final project report, ensuring that the document accurately reflected the work carried out, the findings obtained, and the contributions made by each team member.

The successful completion of the movie recommendation system project was made possible through the collaborative efforts of all team members. Each member played a vital role in the various stages of the project, from research and proposal development to implementation, analysis, and presentation. This division of work allowed the team to efficiently develop and evaluate the recommendation systems, resulting in a valuable contribution to the field of movie recommendations.

## X. CONCLUSION

In conclusion, this project aimed to develop and evaluate three different deep learning-based movie recommendation systems, namely Singular Value Decomposition (SVD), Neural Networks based on Embeddings, and Graph Neural Networks (GNNs). We used the Movielens dataset for our experiments and evaluated the performance of each system for two scenarios: recommending similar movies given a particular movie and recommending movies to a user based on their likes.

Our primary goal was to address the cold start problem, which is a significant challenge for recommendation systems when recommending content to new users or items with limited data. We leveraged the inherent structure and relationships within the data using Graph Neural Networks to provide valuable insights for improving the user experience and satisfaction in movie recommendation applications.

Through our experiments, we found that all three recommendation systems showed promising results in terms of performance and were able to effectively handle the cold start problem. However, the GNN-based system showed the most promising results in terms of accuracy and was able to outperform the other two systems in both scenarios.

The SVD model achieved an RMSE of 2.5726. The Neural Networks based on Embeddings was evaluated using three different model sizes, and the best performance was achieved by the medium-sized model with an RMSE of 0.9274. Finally, the GNN model obtained a train RMSE of 0.8969 and a test RMSE of 0.9448 after 20 epochs.

In conclusion, this project provides insights into the development and evaluation of deep learning-based recommendation systems for movie recommendation applications, which can enhance the user experience and satisfaction. The techniques and approaches presented in this project can also be applied to other recommendation systems and domains beyond movies.

## REFERENCES

- [1] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 974-983).
- [2] Wang, H., Wang, N., Yeung, D. Y., & Shi, Q. (2019). Knowledge graph convolutional networks for recommender systems. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2106-2115).
- [3] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).
- [4] Wu, Y., DuBois, C., Wang, X., & Wilson, J. (2016). Collaborative denoising auto-encoders for top-N recommender systems. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 37-44).
- [5] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8), 30-37.
- [6] Singh, A. P., & Gordon, G. J. (2008). Relational learning via collective matrix factorization. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 650-658).