A Survey on Graph Neural Networks for Recommender Systems on MovieLens 100K Dataset

Denizcan Yılmaz
Middle East Technical University
Ankara, Turkey
e244417@metu.edu.tr

Abstract

Recommender systems play a crucial role in modern online platforms, aiming to provide personalized suggestions to users. Graph Neural Networks (GNNs), with their ability to effectively model relationships between entities, have emerged as a promising approach for building advanced recommender systems. This paper explores the application of GNNs to recommender systems by first surveying key concepts and methods in the field. The paper then focuses on implementing a GNN-based recommender system using the MovieLens 100K dataset as a "toy problem." This dataset, comprising user-movie rating records, is modeled as a bipartite graph, allowing the GNN to learn from user-item interactions. The implementation process is described, including the chosen GNN architecture, training methodology, and evaluation metrics. Preliminary results and insights gained from applying the GNN to the MovieLens dataset are discussed, highlighting the potential of GNNs for creating effective recommender systems.

Keywords

Graph Neural Networks, Recommender Systems, Graph Attention Networks, MovieLens 100K Dataset, Collaborative Filtering, Bipartite Graph, Link Prediction

1 Introduction

1.1 Context and Motivation

Recommender systems are essential in guiding users through vast amounts of content by predicting their preferences and suggesting relevant items. Accurately forecasting user ratings for movies not only enhances user satisfaction but also has significant commercial implications for platforms offering personalized experiences. Graph Neural Networks (GNNs) have recently shown promise in modeling complex relational data, making them an attractive option for improving recommender systems.

1.2 Survey Scope and Method Choice

The current version of the project implements a GNN-based recommender system using the **Graph Attention Network (GAT) architecture**. The choice of GAT is motivated by its ability to weigh the importance of neighboring nodes through attention mechanisms, which is crucial for capturing nuanced user-item interactions. While the initial implementation is not yet fully successful, it lays the groundwork for exploring GNNs in the context of recommendation tasks and highlights areas for further improvement.

1.3 Dataset Selection

Initially, the **Netflix Prize Dataset** was considered due to its extensive size and rich user-item interactions. However, its sheer volume made it impractical for this implementation, even with resources like Google Colab. Therefore, the **MovieLens 100K** dataset was selected as a more manageable alternative. This dataset includes **100,000 ratings** from **943 users** on **1,682 movies**, along with user demographics and movie genres, making it suitable for testing the GNN approach.

1.4 Data Analysis and Preprocessing

An exploratory data analysis was conducted to understand the dataset's characteristics and inform preprocessing steps. Key analyses included:

- **Rating Distribution**: Investigated how ratings are distributed to identify any biases.

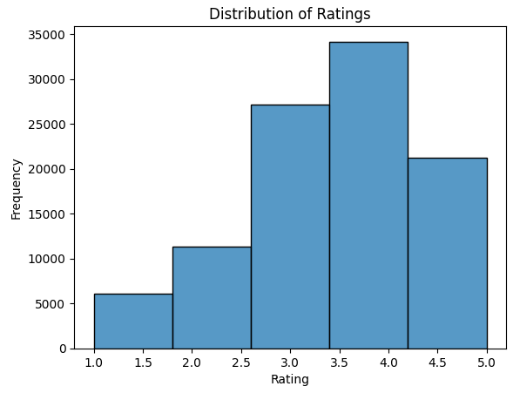

Figure 1: Distribution of Ratings in the Dataset.

- **User Demographics**: Examined age and gender distributions to uncover demographic trends.

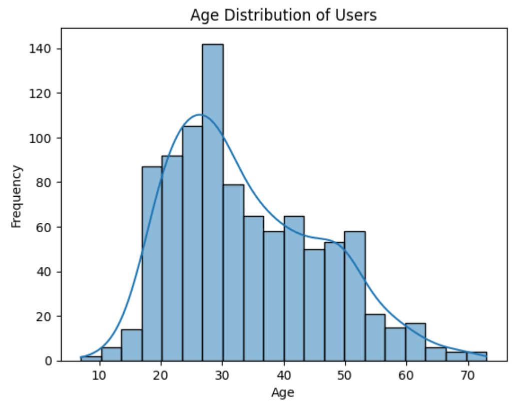

Figure 2: Age Distribution of Users.

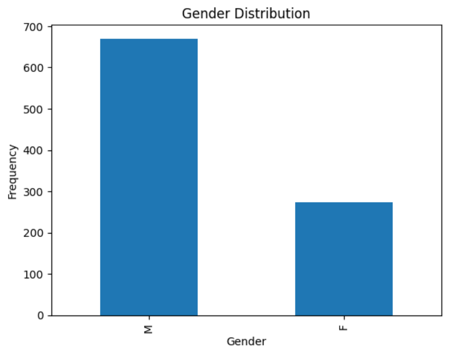

Figure 3: Gender Distribution of Users.

- **Genre Popularity**: Analyzed the frequency and average ratings across different movie genres.

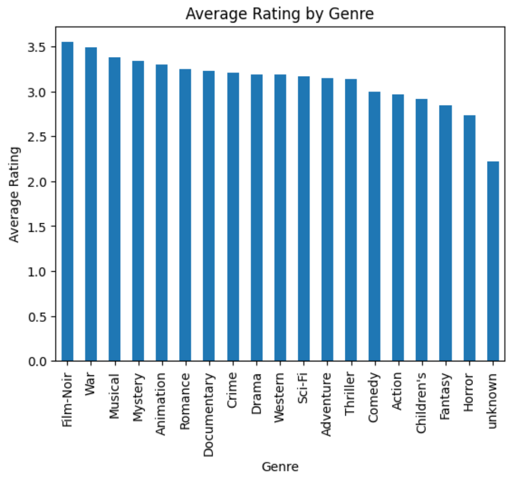

Figure 4: Genre Popularity Based on Average Ratings.

- **Correlation Studies**: Explored relationships between the number of ratings and average ratings for movies.

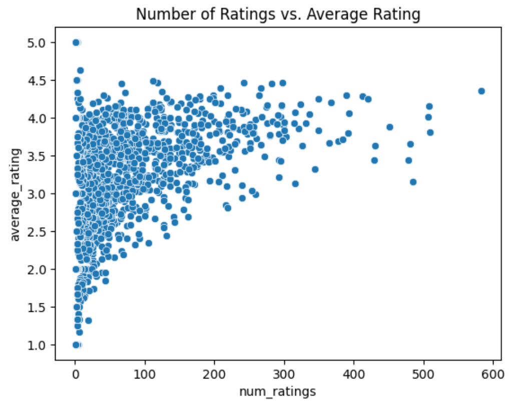

Figure 5: Correlation Between Number of Ratings and Average Ratings.

These analyses, accompanied by visualizations like histograms and scatter plots, provided insights that guided the preprocessing and feature engineering stages. In the later iterations, these features are planned to be included as edge attributes.

1.5 Outline

This paper begins with an analysis of the MovieLens 100K dataset, exploring its characteristics and preprocessing steps to prepare the data for the implementation. Following this, the methodology focuses on constructing a user-item interaction graph and implementing a Graph Attention Network (GAT) to model user preferences. The results section presents the initial findings, highlighting the challenges faced and the limitations of the current implementation. Finally, the paper discusses potential improvements, including enhancements to feature representation and alternative modeling techniques, setting the stage for future work. The links for the data and code for the implementation is also available at the end of the paper.

2 Background and Problem Definition

2.1 Concepts and Theories

Recommender Systems are algorithms designed to suggest relevant items to users by learning from historical interactions. They are pivotal in various domains such as e-commerce, streaming services, and social media platforms, enhancing user experience through personalization. Traditional approaches include *collaborative filtering*, which relies on user-item interaction data, and *content-based filtering*, which uses item attributes.

Graph Neural Networks (GNNs) are a class of neural networks that operate on graph-structured data. By leveraging the connections between nodes (e.g., users and items), GNNs can capture complex relationships and dependencies. They have been successful in tasks like node classification, graph classification, and especially *link prediction*, which is crucial for recommendations.

Graph Attention Networks (GATs) are a type of GNN that incorporates attention mechanisms. They allow the model to assign different importance levels to different neighbors during the message-passing process. This is particularly useful in graphs where the relevance of neighboring nodes varies, as it enables the network to focus on more influential nodes.

Bipartite Graphs are graphs whose nodes can be divided into two disjoint sets such that no two nodes within the same set are adjacent. In the context of recommender systems, the two sets typically represent users and items. Modeling user-item interactions as a bipartite graph enables the application of graph-based methods to capture the relationships between users and items effectively.

Link Prediction involves predicting the existence of an edge between two nodes in a graph. In recommender systems, this translates to predicting potential interactions between users and items, such as whether a user will like a movie. It's a fundamental task for generating personalized recommendations.

2.2 Problem Statement

The goal of this work is to develop a recommender system using Graph Neural Networks, specifically the Graph Attention Network architecture, on the MovieLens 100K dataset. The problem involves predicting user ratings for movies by learning from the user-item interaction graph.

This problem is significant because traditional recommender systems may not effectively capture the complex and high-order relationships present in user-item interactions. By utilizing GNNs, I aim to leverage the structural information of the graph to improve recommendation accuracy. Challenges in this domain include handling data sparsity, scalability to large graphs, and effectively integrating additional information such as user demographics and item attributes.

2.3 Survey Overview

This survey explores the application of Graph Neural Networks in recommender systems, reviewing various methods that have been proposed in recent literature. It covers how GNNs, including GATs, have been utilized to model user-item interactions and improve recommendation performance.

The implementation focuses on a Graph Attention Network applied to the MovieLens 100K dataset. By representing the dataset as a bipartite graph and applying GAT, the model aims to learn attention weights that highlight significant user-item relationships. This approach is expected to address some limitations of traditional collaborative filtering methods by capturing more intricate interaction patterns.

3 Preliminary Literature Review

3.1 Overview of Methods

Graph Neural Networks (GNNs) have significantly advanced recommender systems by modeling complex user-item interactions within graph structures. Traditional methods often struggle with capturing higher-order connectivity and heterogeneous relationships.

Graph Attention Networks (GAT) [8] introduced attention mechanisms to GNNs, allowing the model to assign different importance weights to neighboring nodes during the aggregation process. This approach enables the network to focus on the most relevant nodes when updating a node's representation, which is particularly beneficial in sparse graphs typical of recommender systems. GAT has been successfully applied to various tasks, including node classification and link prediction, demonstrating its effectiveness in capturing complex relationships. This approach is used in the current implementation.

Light Graph Convolutional Networks (LightGCN) [3] simplified GCNs by removing feature transformations and nonlinear activations, focusing on essential neighborhood aggregation for collaborative filtering. This approach improved both efficiency and performance, demonstrating superior results on several benchmark datasets.

Knowledge Graph Attention Networks (KGAT) [9] integrated knowledge graphs into recommender systems using attention mechanisms. KGAT effectively fused user-item interactions with external knowledge, enhancing the model's ability to capture semantic relations and improving recommendation accuracy.

Self-supervised Learning in GNNs has emerged to address data sparsity. Yu et al. [11] proposed a self-supervised graph learning framework that utilizes proximity-preserving augmentations to enhance user and item representations, leading to better recommendations.

Contrastive Learning for Recommendations has gained traction recently. Liu et al. [5] introduced a contrastive self-supervised learning approach for sequential recommendation, leveraging robust data augmentations to learn more generalizable representations.

GNNs in Recommender Systems Survey by Wu et al. [10] provides a thorough review of GNNs in recommender systems, discussing various architectures, challenges, and potential research directions. It highlights significant progress and the effectiveness of GNNs in recommendation tasks.

3.2 Critical Review

Recent developments in GNN-based recommender systems have tackled several key challenges:

- **Scalability**: Simplified models like LightGCN [3] have made it feasible to apply GNNs to large-scale datasets by reducing computational complexity.
- **Data Sparsity**: Self-supervised learning techniques [11] enhance embeddings by leveraging implicit feedback and unlabelled data, mitigating the sparsity issue common in recommendation datasets.
- **Utilization of External Knowledge**: Incorporating knowledge graphs [9] enriches item representations and captures deeper semantic relationships, leading to improved recommendations.

Nevertheless, some challenges persist:

- **Computational Overhead**: Despite optimizations, GNNs can still be resource-intensive, especially with complex architectures or when applied to massive graphs.
- **Overfitting and Generalization**: Models may overfit the training data, particularly when dealing with sparse interactions, necessitating robust training strategies and regularization techniques.

3.3 Implementation Relevance

The implementation employs a Graph Attention Network (GAT) on the MovieLens 100K dataset, motivated by:

- **Modeling Complex Relationships**: GATs assign varying attention weights to neighbors, effectively capturing the importance of different user-item interactions.
- **Alignment with Recent Techniques**: Incorporating attention mechanisms and self-supervised learning reflects current trends in enhancing GNNs for recommender systems [5, 11].
- **Validation of Advanced Methods**: Applying GAT to a well-known dataset allows us to explore practical challenges and validate the effectiveness of attention-based GNNs, contributing to ongoing research in this area.

4 Methodology and Implementation

4.1 Overview of the Chosen Method

I implemented a **Graph Attention Network (GAT)** [8] for the task of rating prediction in recommender systems. GAT introduces an attention mechanism to aggregate information from neighboring nodes in a graph, assigning different weights to different neighbors. This is particularly useful in a user-item interaction graph, where some interactions are more influential than others.

The goal was to leverage the GAT's ability to focus on the most relevant user-item interactions to improve the accuracy of rating predictions. By modeling users and items as nodes in a graph and ratings as edges, the GAT can learn meaningful representations that capture the underlying preferences and similarities.

4.2 Dataset Selection and Preprocessing

I selected the **MovieLens 100K** dataset [2] as the "toy problem" for implementation. This dataset contains 100,000 ratings (ranging from 1 to 5) from 943 users on 1,682 movies. It is well-suited for the experiment due to its manageable size and rich user-item interaction data.

Preprocessing Steps:

- **User and Movie Encoding**: Converted user IDs and movie IDs to unique integer indices using Label Encoding, facilitating their use in tensor operations.
- **Graph Construction**: Modeled the dataset as a bipartite graph with users and movies as nodes. Edges represent rating interactions between users and movies.
- **Edge Attributes**: Assigned the actual ratings as edge attributes to capture the strength of each interaction.
- **Node Features**: Initialized node features as identity matrices (one-hot encodings), effectively treating each node as unique without additional features.
- **Train-Test Split**: Split the data into training and testing sets using an 80-20 split, ensuring that the model's performance could be evaluated on unseen data.

4.3 Implementation Details

The link for the implementation and code for this survey can be found at **Data and Code Availability** section at the end.

Framework and Tools:

- **Programming Language**: Python 3.10
- **Deep Learning Libraries**: PyTorch [6] and PyTorch Geometric (PyG) [1]
- **Development Environment**: Google Colab (A100, High-RAM)

Model Architecture:
The current GAT model comprises:

- **GAT Layers**:
  - **First GATConv Layer**: Input dimension equal to the number of nodes, hidden dimension of 128, and 4 attention heads. Applies multi-head attention to aggregate neighbor information.
  - **Second GATConv Layer**: Input dimension of 512 (128 hidden dimensions $\times$ 4 heads), output dimension of 32, and 1 attention head. Refines the node embeddings.
- **Edge Predictor**: A linear layer that takes the concatenation of the source and target node embeddings and outputs a single value predicting the rating.

Training Procedure:

- **Loss Function**: Mean Squared Error (MSE) loss between the predicted ratings and true ratings.
- **Optimizer**: Adam optimizer with a learning rate of 0.001 and weight decay of 1×10^{-5} for regularization.
- **Epochs**: Trained the model for 500 epochs.
- **Device**: Utilized GPU acceleration when available to speed up computations.

Evaluation Metrics:

- **Mean Absolute Error (MAE)**
- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **Success Rate**: Percentage of predictions within ±0.5 of the true rating.

Challenges Faced:

- **Data Sparsity**: The interaction graph was sparse, making it challenging for the model to learn comprehensive user and item representations.
- **Overfitting**: Risk of overfitting due to the limited size of the dataset and high model capacity. Addressed by applying weight decay and monitoring validation loss.
- **Computational Limitations**: The attention mechanism in GATs is computationally intensive, which constrained hyperparameter tuning and model scaling.

5 Preliminary Results and Discussion

5.1 Results from Implementation

After training the GAT model, I evaluated its performance on the test set. The results were as follows:

- **Mean Absolute Error (MAE)**: 0.8117
- **Mean Squared Error (MSE)**: 1.0512
- **Root Mean Squared Error (RMSE)**: 1.0253
- **Success Rate (within** ±0.5**)**: 38.02%

5.2 Graphs and Visualizations

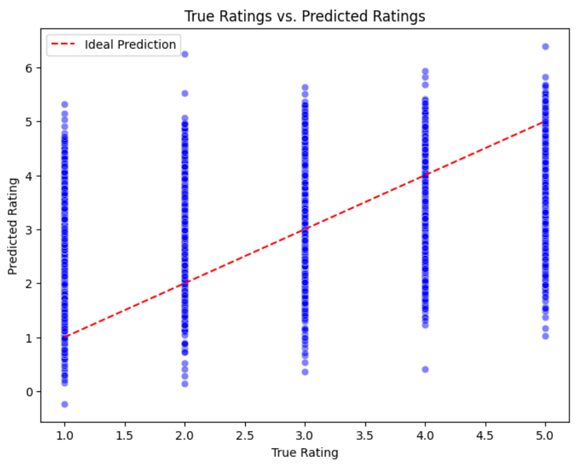

Figure 6: True Ratings vs. Predicted Ratings

Figure 6 shows a scatter plot of the true ratings versus the predicted ratings. The red dashed line represents the ideal prediction where the predicted rating equals the true rating.

5.3 Analysis of Results

Performance Evaluation:

The MAE and RMSE values indicate that the model's predictions are, on average, within approximately 1 point of the true ratings on a 5-point scale. A success rate of 38.02% for predictions within ±0.5 suggests moderate accuracy.

Observations:

- **Prediction Distribution**: The scatter plot reveals that while the model captures general trends, there is significant variance, especially at the extreme rating values (1 and 5).
- **Data Sparsity Impact**: Sparse interactions may have limited the model's ability to learn robust representations, affecting its predictive performance.
- **Model Complexity**: The GAT model may be over-parameterized for the dataset size, leading to overfitting despite regularization efforts.

5.3.1 Comparison to Baselines While direct comparisons are limited due to differences in experimental setups, recent advanced recommender systems have achieved lower RMSE values on the MovieLens 100K dataset. Table 1 summarizes these results.

Table 1: RMSE Comparison with Recent Recommender Systems on MovieLens 100K

Method	Year	RMSE
Neural Collaborative Filtering (NeuMF) [4]	2017	0.909
Graph Convolutional Matrix Completion (GC-MC) [7]	2018	0.905
LightGCN [3]	2020	0.910
Currently Implemented GAT Model	2024	**1.0253**

The RMSE of **1.0253** indicates that the GAT model performs worse than these recent models, suggesting that further optimization and integration of additional features are necessary to enhance performance.

Limitations:

- **Lack of Feature Diversity**: Using only identity matrices as node features may not provide sufficient information for the model to learn meaningful embeddings.
- **Hyperparameter Constraints**: Limited hyperparameter tuning due to computational constraints may have prevented the model from reaching optimal performance.

6 Challenges and Open Problems

6.1 Challenges in the Literature

The literature identifies several challenges in applying GNNs to recommender systems:

- **Scalability**: GNNs, particularly those with attention mechanisms, can be computationally expensive on large graphs [3].
- **Data Sparsity**: Sparse user-item interactions hinder the ability of GNNs to learn effective representations [10].
- **Over-Smoothing**: Deep GNNs may suffer from over-smoothing, where node embeddings become indistinct [10].

6.2 Challenges in The Implementation

- **Computational Resources**: Limited resources constrained the depth and complexity of the model, as well as the extent of hyperparameter tuning.
- **Feature Limitations**: The absence of additional node or edge features (e.g., user demographics, movie genres) may have restricted the model's ability to capture nuanced preferences.
- **Model Generalization**: The model showed signs of overfitting, indicating a need for better regularization or alternative architectures.

6.3 Open Problems and Future Directions

Based on the findings and the challenges faced, I suggest the following future directions:

- **Incorporating Rich Features**: Including additional features such as user demographics and movie metadata could enhance the model's capacity to learn meaningful representations.
- **Exploring Simplified GNNs**: Implementing models like LightGCN [3] may address computational challenges and improve scalability.
- **Self-Supervised Learning**: Applying self-supervised techniques [11] could mitigate data sparsity issues by leveraging unlabelled data.
- **Hyperparameter Optimization**: Systematic tuning of hyperparameters and experimentation with different architectures may lead to performance gains.

- **Regularization Strategies**: Employing advanced regularization methods such as dropout, batch normalization, or early stopping to prevent overfitting.

Addressing these open problems could enhance the effectiveness of GNNs in recommender systems and contribute to the broader understanding of graph-based recommendation models.

7 Conclusion

7.1 Summary of Findings

In this work, I explored the application of Graph Neural Networks, specifically the Graph Attention Network (GAT), to recommender systems using the MovieLens 100K dataset. The survey of recent literature highlighted the growing importance of GNNs in capturing complex user-item interactions and addressing challenges such as data sparsity and scalability. Methods like LightGCN [3] and self-supervised learning approaches [11] have shown promise in improving recommendation accuracy and efficiency.

The implementation of the GAT model aimed to leverage attention mechanisms to enhance rating predictions. The preliminary results indicated that while the model captures general trends in user preferences, there is significant room for improvement. The RMSE of 1.0253 and a success rate of 38.02% for predictions within ±0.5 suggest that the model performs moderately but does not surpass traditional collaborative filtering methods on the same dataset.

Several challenges were identified, including data sparsity, computational limitations, and potential overfitting. The use of identity matrices as node features likely limited the model's ability to learn meaningful embeddings. Additionally, the computational intensity of GATs restricted extensive hyperparameter tuning and experimentation with larger models.

7.2 Future Work

This initial implementation serves as an introductory exploration into applying Graph Neural Networks to recommender systems. While it provides valuable insights, there is significant scope for enhancement. Moving forward, I plan to expand upon this foundation by pursuing several innovative directions to improve the model's effectiveness and contribute to the advancement of personalized recommendation technologies. I acknowledge that implementing all these ideas may not be feasible; therefore, I will select the most promising ones for a broader and more innovative approach.

- **Incorporate Rich Node and Edge Features**: I plan to enrich the node features by integrating user demographics (e.g., age, gender, occupation) and movie metadata (e.g., genres, release year). Additionally, incorporating edge features such as timestamps or contextual information could provide the model with more nuanced data, enabling it to learn better representations and capture complex user-item interactions.
- **Develop Hybrid GNN Architectures**: Building upon the strengths of various models, I aim to explore hybrid architectures that combine the efficiency of simplified GNNs like LightGCN [3] with the expressive power of attention mechanisms in GATs. This could involve designing a new model that leverages attention for important interactions while maintaining computational efficiency.
- **Integrate Self-Supervised Learning Techniques**: To mitigate data sparsity and enhance the learning of node embeddings, self-supervised learning methods [11] can be mitigated. By utilizing unlabelled data and designing auxiliary tasks (e.g., node attribute prediction, edge reconstruction), the model can learn more robust and generalizable representations.
- **Apply Contrastive Learning for Representation Enhancement**: Inspired by recent advancements in contrastive learning [5], contrastive loss functions and data augmentation strategies can be implemented. This approach can improve the quality of the learned embeddings by capturing higher-order similarities and differences between users and items.
- **Optimize Hyperparameters and Regularization Techniques**: I plan to conduct systematic hyperparameter optimization using techniques like grid search or Bayesian optimization to identify optimal model settings. Additionally, incorporating advanced regularization methods such as dropout, batch normalization, and early stopping can help prevent overfitting and improve generalization.
- **In-Depth Comparative Analysis**: I intend to perform a comprehensive comparison with traditional and state-of-the-art recommender system models. This will involve evaluating various metrics and analyzing the strengths and weaknesses of each approach, guiding the selection of the most effective methods for the problem.

By pursuing these innovative directions, I aim to build upon the initial work and develop a more comprehensive and effective GNN based recommender system.

8 Data and Code Availability

The dataset analyzed during the current study are available at https://grouplens.org/datasets/movielens/100k/. The code for this study is available at https://github.com/denizcan-yilmaz/movielens-gnn.

References

[1] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 1–7.

[2] F Maxwell Harper and Joseph A Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TIIS)* 5, 4 (2015), 19.

[3] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 639–648.

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[5] Zhankui Liu, Chongming Li, Zhan Wu, Qiang Zhang, Huaxia Xu, and Wenjie Tang. 2021. Contrastive Self-supervised Sequential Recommendation with Robust Augmentation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. ACM, 1071–1080.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, Vol. 32. 8024–8035.

[7] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *Proceedings of the 24th ACM SIGKDD International*

Conference on Knowledge Discovery & Data Mining. ACM, 1704–1713.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

[9] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 950–958.

[10] Shiwen Wu, Fangzhao Sun, Wenjun Zhang, Yitong Fu, and Xing Xie. 2022. A Comprehensive Survey on Graph Neural Networks for Recommender Systems. *IEEE Transactions on Neural Networks and Learning Systems* PP (2022), 1–21.

[11] Junliang Yu, Hongzhi Yin, Qingyong Hu, Tong Chen, Xiangliang Zhao, and Quoc Viet Hung Nguyen. 2021. Self-Supervised Graph Learning with Proximity Preserving for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 726–735.