A CNN Autoencoder boosted Collaborative Filtering Model

Tanish Kapoor
Department of Computer Science Engineering
Delhi Technological University
Delhi, India
tanishkapoor_co22a7_58@dtu.ac.in

Jyoti
Department of Computer Science
Shyama Prasad Mukherji College for Women, University of Delhi
Delhi, India
varunivaruni094@gmail.com

Kritika Soni
Department of Computer Science
Shyama Prasad Mukherji College for Women, University of Delhi
Delhi, India
kritikasoni0605@gmail.com

Shweta Tyagi
Department of Computer Science
Shyama Prasad Mukherji College for Women, University of Delhi
Delhi, India
shwetacs@spm.du.ac.in

Sonia Kumari
Department of Computer Science
Shyama Prasad Mukherji College for Women, University of Delhi
Delhi, India
sonia.kumari@spm.du.ac.in

Abstract*— An important role has been played by Recommender Systems in various domains of our daily routine, especially in media and e-commerce applications. Recommender Systems help to personalize the user experience. However, some challenges are still faced such as data sparsity and cold-start scenarios. Due to these challenges the effectiveness of recommender system is limited. This paper proposes a hybrid recommendation model which will addresses these issues. Our proposed model integrates Collaborative Filtering (CF) with Convolutional Neural Networks (CNN) for both recommendation and matrix completion tasks. Implemented on MovieLens 100K dataset, the proposed model leverages user and item metadata to enrich the sparse user-item interaction matrix. To handle the problem of sparsity, the proposed CNN autoencoder (CNN-AE) performs matrix completion by considering rating patterns through a matrix-as-image transformation. Thereafter, a weighted hybridization mechanism referred to as CF+CNN-AE is applied. This mechanism uses per-user alpha tuning to optimally blend predictions from CF and CNN-AE for improved accuracy. The model's performance is evaluated using standard metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Experimental results reveal that the proposed CF+CNN-AE model achieves superior accuracy compared to both traditional CF and the standalone CNN-AE approach.

Keywords*— Recommender System, Hybrid Model, Collaborative Filtering, CNN Autoencoder, Cold-start, Sparsity, MovieLens.

I. INTRODUCTION

Recommender systems are widely used in platforms like online shopping and movie streaming to suggest personalized content [1]. Among various approaches, collaborative Filtering (CF) is one of the most common, as it predicts user preferences based on similarities with other users [2]. CF techniques are classified as memory-based (neighbourhood-based) and model-based [1], [3], [4]. In literature, we can find various approaches to tackle the problems associated with CF such as sparsity, cold-start, new-user, new-item and scalability [5], [6], [7], [8], [9]. These approaches are developed to generate more accurate and reliable recommendations. Still, CF struggles with data sparsity and cold-start problems, where limited data leads to poor recommendations.

To address the problems of sparsity and cold-start, this paper presents a hybrid recommendation model that merges memory-based CF with convolutional autoencoder. Section 2 presents a comprehensive review of the existing literature. In Section 3, the architecture of the proposed hybrid model is discussed. In Section 4, the experimental setup and results are presented. Section 5 concludes the paper and outlines potential avenues for future research.

II. RELATED WORKS

Recommender systems have evolved a lot, and there are some techniques which are used in to enhance the accuracy and efficiency of recommendations.

A. Matrix Factorization Approaches

CF based recommendation models have used many matrix factorization methods like Singular Value Decomposition (SVD). This strategy has been used to extract hidden patterns from user-item interaction data [10], [11]. These techniques shrink the large dimension matrices into smaller dimension matrices that help to identify the preferences. To capture more complex and non-linear user behaviours, deep learning model was introduced for matrix factorization [12]. Using this approach, more accurate predictions and recommendations are regenerated using this approach.

B. Integrating Autoencoders with Collaborative Filtering

A significant advancement for CF tasks is marked by AutoRec model by adapting autoencoders[13]. It treats incomplete data matrices as inputs to reconstruction process; it effectively learns to predict the missing data. AutoRec demonstrates good performance on datasets, it often results better on matrix factorization, especially when the data is highly sparse. Further, a hybrid collaborative recommendation model was developed using dual-autoencoder to alleviate cold-start problem [14]. Guo et al proposed another hybrid recommender system based on autoencoder and latent feature analysis for optimal recommendation performance [15].

C. Convolutional Neural Networks for Recommendation Tasks

CNNs are best known for their success in image processing and have become a cornerstone of computer vision tasks. Beyond these traditional applications, CNNs have also been explored in the domain of recommender systems. Ying et al. [16] introduced PinSage model, a scalable graph convolutional neural network designed for web-scale recommendation tasks. In another approach, Guo [17] proposed a method that integrates CNN-based visual feature extraction with CF to improve movie recommendations. Additionally, Liu et al. [18] proposed SECNCF, a model that stacks user and item embeddings and applies convolutional layers to learn rich interaction patterns, demonstrating improvements in the traditional CF methods.

III. Proposed Work

To enhance the performance and accuracy of recommendation system, especially in the presence of sparse data, we propose a hybrid architecture that integrates CF based techniques with deep learning methods. The core idea is to utilize the strengths of both traditional memory-based approaches and neural networks to achieve better generalization and robustness. The model uses both user and item features, along with the rating matrix, in a different way to improve the prediction accuracy. By converting the rating matrix into an image like format, the CNN autoencoder learns hidden patterns and helps fill in missing ratings. A weighted blending method is then used to combine the outputs from CF and CNN autoencoder, adapting to each user.

We start by utilizing the MovieLens 100k dataset[1], which contains explicit user ratings for movies, along with user demographic information and movie metadata. The set of users is represented by U and set of items is denoted by I .

Let the user-item interaction matrix be represented by $R = [r_{i,j}]$. The size of the matrix R is $m \times n$ where m represents number of users $i.e.\ |U| = m$ and n represents number of items $i.e.\ |I| = n$. Each row of the matrix represents user latent factor and each column is representing item latent factor.

The proposed model works in three phases as outlined below:

A. Memory- Based Collaborative Filtering

As a foundational layer, a memory-based CF algorithm is employed [4], [19]. Specifically, to learn latent representations of users and items, a user-based CF approach is used. The Pearson correlation coefficient is utilized as the similarity measure, to identify neighbours for a target (active) user, [4], [19]. The similarity between users u and v , denoted as $PSim(u,v)$, is computed using the following formula given in equation (1).

$$PSim(u,v) = \frac{\Sigma_{i\epsilon I}(R_{u,i}-R'_u)\times(R_{v,i}-R'_v)}{\sqrt{\left(\Sigma_{i\epsilon I}(R_{u,i}-R'_u)^2\times\Sigma_{i\epsilon I}(R_{v,i}-R'_v)^2\right)}} \tag{1}$$

where $R_{u,i}$ is the rating of item i for user u and R'_u is the average rating of user u.

Further, to predict missing rating by capturing the underlying structure of user-item interactions, the latent factors of the neighbourhood set are used. The rating prediction formula is given by equation (2) [4], [19].

$$PR_{u,i} = \frac{\Sigma_{v\in Nbr(u)} R_{v,i}\times PSim(u,v)}{\Sigma_{v\in Nbr(u)} PSim(u,v)} \tag{2}$$

While this approach provides solid baseline performance, it tends to suffer in sparse data scenarios and fails to generalize well to new users or movies.

B. CNN-Based Autoencoder for Matrix Completion

To complement the CF model, a CNN-based Autoencoder (CNN-AE) technique is introduced. This model takes the enriched user-item matrix R as input and attempts to reconstruct the full matrix by learning its hidden structure. Convolutional layers are used to extract localized patterns from the data, while encoder-decoder structure captures and reconstructs non-linear dependencies. This component helps in filling missing entries more accurately by leveraging both spatial and contextual relationship.

In user-item matrix R , missing ratings (zeros) represent unrated objects. To ensure that values fall between 0 and 1 for known entries, all ratings were normalized by dividing by the maximum (5 stars) and centring on the user mean. After padding each user's rating vector to the next square dimension (using zeros for missing entries), it was reshaped into a 2D array (height × width) with two input channels: a binary mask that indicated observed items and the normalized ratings. Additionally, to enhance the model's understanding of user preferences, additional features such as age group, occupation and movie genres are integrated as additional channels.

To sum up, preprocessing consisted of

- Input: Take an $m \times n$ matrix, R of raw ratings of n items given by m users.
- Normalization: Ratings are scaled to [0,1] and missing entries are filled with the user's mean rating.
- Padding and reshaping: The rating matrix is padded or reshaped to form a square of size (height × width). This square is then stacked into a 3D shape of (height, width, 2), where the two channels are
    - filled ratings
    - mask (1=filled, 0= original)
- Adding metadata as extra channels: Other features such as age group, occupation and movie genres are included as additional channels.
    - age group is mapped to a normalized value per user adding 5 channels
    - occupation is mapped to one-hot encoding or padding creating 10 channels
    - movie genres (multi-hot encoding or paddings) creating 18 channels
- Output: The rating matrix along with user metadata and item metadata represented by a 3D "image" with total 35 channels out of which 2 are for ratings and mask, 5 for age group, 10 for occupation and 18 for genres.

[1] https://www.kaggle.com/datasets/prajitdatta/movielens-100k-dataset

Further, proposed CNN-AE model takes output of the preprocessing step to understand latent user preferences and predict missing ratings. The working of the proposed model consists of the following steps:

1) Encoder acting as a pattern detector is designed to extract localized and hierarchical features from the enriched user-item matrix. To aid the progressive abstraction of data, it consists of two convolutional blocks both having L2 kernel regularization, batch normalisation and ReLU activation:

- First block starts with a two-dimensional convolution layer with 32 filters each of size (3x3) and to preserve spatial dimensions, 'same' padding is used. To provide translation invariance and capture local feature patterns, a max pooling layer (2x2) is used to divide the spatial dimensions in half.

- Second block doubles the feature map's depth, using a two-dimensional convolutional layer of 64 filters, also uses 'same' padding and (3x3) kernels. A second max-pooling layer (2×2, same padding) cuts the size of the spatial dimensions in half again, and a SpatialDropout2D (rate = 0.5) is used to stop overfitting by randomly dropping whole feature maps.

2) Decoder acting as encoder's mirror uses compressed latent representation enabling the model to learn a compact, low-dimensional representation while accurately predicting missing entries It reconstructs a two dimensional 'image' of the user-item matrix. Through a series of meticulously designed layers all having L2 Regularization, the decoder systematically restores the structure of the spatial dimensions and the input:

- In the first block, a two-dimension convolutional layer consisting 64 filters with (3x3) kernels and batch normalization is used to preserve spatial information. To increase the spatial dimensions, an upsampling layer of size (2x2) is incorporated resulting in gradually restoration of matrix's shape.

- Reducing filters to 32, a 2D convolutional layer with (3x3) kernels and batch normalization is introduced in second block of decoder. To bring the output closer to initial input size, an additional (2x2) upsampling layer is used to expand spatial dimension.

- The output layer which reconstructs the ratings channel is a two-dimensional convolutional layer having single filter, sigmoid activation and a (3x3) kernel. A final cropping layer is introduced to compensate for any excess introduced during upsampling operations by ensuring that the output and input dimensions precisely match.

To detect missing items ratings, this mirror architecture reconstructs the 2D rating picture. A last cropping stage trims any excess so that the output dimensions exactly match the original $m \times n$ rating matrix. This ensures a one-to-one correspondence between the reconstructed image and the input matrix R, allowing accurate imputation of all missing ratings. Consequently, each user's ratings are now learned by the proposed model.

C. Adaptive Hybrid Fusion

This traditional memory-based CF offers a baseline but typically performs badly on sparse data. For handling the problem of sparsity, the proposed hybrid model CF+CNN-AE adopts an adaptive fusion mechanism that combines the predictions from the two models, CF and CNN-AE. The final predicted rating is obtained by combing the outputs of the CF model and the CNN-AE with weights α and $1 - \alpha$. Thus, if $PR^1_{u,i}$ and $PR^2_{u,i}$ are the predicted rating of an item i for user u by CF and CNN-AE models respectively, then the prediction by the hybrid approach CF+CNN-AE, $PR^3_{u,i}$, is given by the formula (3).

$$PR^3_{u,i} = \alpha\, PR^1_{u,i} + (1-\alpha) PR^2_{u,i} \quad (3)$$

These weights are not fixed and are dynamically optimized for each user through a grid search over the validation set. This allows the model to adjust its reliance on each component depending on the input data characteristics, ultimately improving the accuracy of predictions.

IV. Experimentation and Results

In our experiment, we made use of the MovieLens 100K dataset, which comprises 1,682 movies with 100,000 users' ratings. We constructed the standard user–item rating matrix (943 × 1682) with rows for persons and columns for movies. To make sure recommendations are sent to people who are not visible, we divided the dataset by user. Thirty percent of customers were retained as test users, and seventy percent were chosen for training. This scheme is repeated to choose five random samples of training and test users. These samples are named as Sample1, Sample2, Sample3, Sample4, and Sample5. We also withheld 20% of each test user's known ratings as validation subset used to tune hyperparameters and remaining 80% as training input for each sample.

A. Evaluation Metrics

The accuracy of our rating-prediction methods is assessed on each test user's held-out ratings (20% of that user's known ratings) using two standard measures: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). For the CNN-AE component during training, we employ a masked Huber loss that only penalizes reconstruction errors on the observed entries, effectively ignoring missing values. The formulas for computation of MAE, RMSE and masked Huber are given in Equations (4), (5) and (6) respectively.

$$\text{MAE} = \frac{\sum_{i=1}^{N_u} |R_{u,i} - PR_{u,i}|}{N_u} \quad (4)$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N_u} |R_{u,i} - PR_{u,i}|^2}{N_u}} \quad (5)$$

$$\text{Huber} = \frac{1}{\sum m_{u,i}} \sum_{u,i} m_{u,i}\, l_\delta (R'_{u,i} - \hat{R}_{u,i}) \quad (6)$$

Where, $R_{u,i}$ represents the actual rating of item i for user u, $PR_{u,i}$ represents the predicted rating of item i for user u and N_u denotes the number of items rated by user u, where $m_{u,i}$ = 1 if rating $R'_{u,i}$ is observed and 0 otherwise, and l_δ is the Huber function that is quadratic for small errors and linear beyond the threshold δ.

B. *Model Architecture*

Three models namely, memory-based CF, CNN-AE and a hybrid model CF+CNN-AE are implemented. In the CF model, we used user-to-user similarity, where the Pearson Correlation of each test user's (partial) rating vector is compared to all training users on co-rated items. To forecast test user ratings, we calculated a weighted sum of the ratings of the top 30 most similar users.

The CNN-AE sees the enriched user-item matrix as a multi-channel image. We train it with a masked Huber loss that only penalizes reconstruction errors on entries that have been seen.

The hybrid CF+CNN-AE linearly blends the two predictors. We hold back 20% of each user's training ratings for α tuning. To do this, we do a grid search over each user determining weight α in [0,1] to find the value that gives the lowest RMSE. Then, the best α is used to combine the CF and CNN-AE predictions for all unseen items.

C. *Training Configuration*

The CNN-AE model was trained using the Adam optimizer with an initial learning rate of 0.001 and a batch size of 32. Learning rate reduction on plateau was set up with a factor of 0.5, patience of 4 epochs, and a minimum learning rate of 1×10^{-6}. Early stopping was implemented by monitoring validation loss with a patience of 8 epochs for restoring the optimum weights. To reduce overfitting, a dropout rate of 0.3 and a spatial dropout of 0.5 were implemented in the encoder blocks. Early stopping usually happened between 20 and 75 epochs across various random seeds, even though training was allowed for up to 100 epochs.

D. *Results*

The results have been examined using several ways. Firstly, we analyzed the average training and validation loss of CNN-AE model by plotting a graph between the change in loss as we increase the size of the epochs. The graph of average training and validation loss across seeds is shown in Fig. 1.

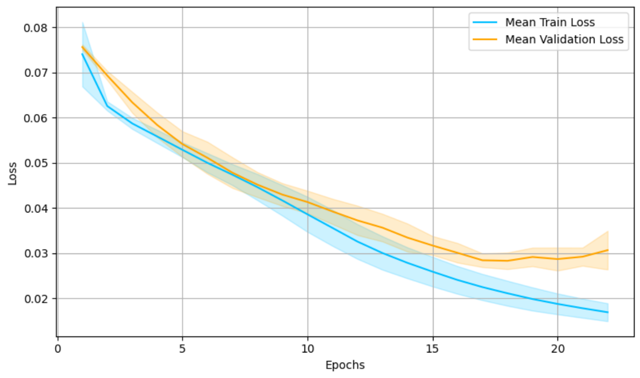

Fig.1: Average Training and Validation loss across Seeds

Both the mean training and mean validation loss curves decrease sharply during the first 5–7 epochs, indicating that the CNN-AE quickly captures major latent structures in the user–item matrix. After approximately epoch 10, the rate of loss reduction slows, and by epoch 15 both curves begin to flatten. This leveling suggests that most learnable patterns have been encoded and the model is converging. Throughout all epochs, the mean validation loss remains only slightly above the mean training loss. The shaded regions show ± one standard deviation across seeds and overlap substantially. The narrow variance and close tracking show no overfitting and consistent performance across runs. At subsequent epochs, there are no noticeable spikes or widening gaps between the training and validation curves. It is also observed that the final training and validation losses converged to roughly 0.008 and 0.007, respectively.

Second, for each sample, we calculated the RMSE and MAE over all held-out ratings that is 20% of each test user's known ratings to quantitatively evaluate accuracy. We calculated the errors between predicted and actual ratings on 20% of known ratings of test users. Finally, the RMSE and MAE were averaged over all test users held-out ratings. Thus, the overall assessment gauges each method's accuracy in forecasting missing ratings. The computed average accuracy of each model is summarized in Table 1. The result in the given table concludes that the hybrid model CF+CNN-AE, a combination of CF and CNN-AE approach performs better than the standard CF and CNN-AE approaches.

TABLE 1: Performance comparison based on RMSE and MAE

		CF	**CNN-AE**	**CF+CNN-AE**
Sample 1	MAE	0.7595	0.9764	0.7396
	RMSE	0.9235	1.1357	0.8916
Sample 2	MAE	0.7744	0.9702	0.7444
	RMSE	0.9335	1.1541	0.8924
Sample 3	MAE	0.7304	0.9491	0.7124
	RMSE	0.8937	1.1312	0.8625
Sample 4	MAE	0.7621	0.9636	0.7357
	RMSE	0.9314	1.1414	0.8871
Sample 5	MAE	0.7589	0.9286	0.7297
	RMSE	0.9238	1.1110	0.8825

Table 1 reports the average values of the evaluation metrics within the five random samples which show different behaviour of each method. CF itself also meets the sparsity problem in constructing the user–item matrix, which results in a moderate error. The CNN-AE encodes the latent structure more flexibly, but gives higher prediction errors than CF showing that reconstruction alone is insufficient in sparse training. By contrast, the hybrid CF+CNN-AE model that linearly blends predictions made by CF and CNN-AE, produces more accurate predictions than either component alone.

Importantly, the per-user blending weight α that was estimated via grid search in favor of CF's contribution was on average higher than for CNN-AE. It is noticed from experiments that optimal α values were rather clustered around 0.5–1.0 across users. This pattern highlights that while deep autoencoding enhances accuracy, CF-based approach is still important for accurate rating prediction at sparsity levels. In summary, the combination approach is utilizing the strengths of the two paradigms to complement each other. This shows that deep neural models can enrich the traditional CF algorithms.

V. Conclusion and Future Work

We proposed a hybrid model, CF+CNN-AE, that combines a convolutional autoencoder (CNN-AE) with a neighbourhood-based collaborative filtering (CF) module to address the problem of data sparsity in recommender systems. In this model, the CNN-AE treats the incomplete user-item rating matrix as an image, learning latent features to impute missing values. Consequently, the CF component leverages observed ratings to exploit user-item similarity. We evaluated the model on the MovieLens 100k dataset using MAE and RMSE. The experimental results demonstrate that the integration of CNN-AE with CF significantly performs better than both the conventional user-based CF and the standalone CNN-AE in the terms of prediction and accuracy, often heavily weighing the CF's prediction per user but appreciating CNN-AE output.

In order to further improve and make more general the hybrid recommendation framework, it becomes interesting to think about user preferences and item popularity that evolves over time. Introducing a notion of time into the model e.g., via the recurrent layers or time-decay components would make the recommendations more flexible and possibly more effective. Furthermore, adaptation to implicit feedback such as clicks, views and play counts instead of relying entirely on explicit ratings, would render the system more suitable for real-world scenarios where such data is more abundant.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Trans Knowl Data Eng*, vol. 17, no. 6, pp. 734–749, Jun. 2005, doi: 10.1109/TKDE.2005.99.

[2] D. Billsus and M. J. Pazzani, "Learning Collaborative Information Filters," in *Proceedings of the Fifteenth International Conference on Machine Learning*, in ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 46–54.

[3] F. Ricci, B. Shapira, and L. Rokach, "Recommender Systems: Introduction and Challenges," *Recommender Systems Handbook, Second Edition*, pp. 1–34, Jan. 2015, doi: 10.1007/978-1-4899-7637-6_1.

[4] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", doi: 10.5555/2074094.

[5] S. Natarajan, S. Vairavasundaram, S. Natarajan, and A. H. Gandomi, "Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data," *Expert Syst Appl*, vol. 149, 2020, doi: 10.1016/j.eswa.2020.113248.

[6] S. Zhang, C. Li, L. Ma, and Q. Li, "Alleviating the sparsity problem of collaborative filtering using rough set," *COMPEL - The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 32, no. 2, 2013, doi: 10.1108/03321641311296918.

[7] R. Lavanya and B. Bharathi, "Movie Recommendation System to Solve Data Sparsity Using Collaborative Filtering Approach," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 5, 2021, doi: 10.1145/3459091.

[8] S. Tyagi and K. K. Bharadwaj, "Enhancing collaborative filtering recommendations by utilizing multi-objective particle swarm optimization embedded association rule mining," *Swarm Evol Comput*, vol. 13, pp. 1–12, Dec. 2013, doi: 10.1016/J.SWEVO.2013.07.001.

[9] S. Tyagi, P. Yadav, M. Arora, and P. Vashisth, "Predicting Users' Interest Through ELM-Based Collaborative Filtering," in *Data, Engineering and Applications*, Singapore: Springer Singapore, 2019, pp. 23–33. doi: 10.1007/978-981-13-6347-4_3.

[10] M. Ranjbar, P. Moradi, M. Azami, and M. Jalili, "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems," *Eng Appl Artif Intell*, vol. 46, pp. 58–66, Nov. 2015, doi: 10.1016/J.ENGAPPAI.2015.08.010.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer (Long Beach Calif)*, vol. 42, no. 8, pp. 30–37, 2009, doi: 10.1109/MC.2009.263.

[12] R. Lara-Cabrera, Á. González-Prieto, and F. Ortega, "Deep Matrix Factorization Approach for Collaborative Filtering Recommender Systems," *Applied Sciences 2020, Vol. 10, Page 4926*, vol. 10, no. 14, p. 4926, Jul. 2020, doi: 10.3390/APP10144926.

[13] S. Sedhain, A. K. Menony, S. Sannery, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*, Association for Computing Machinery, Inc, May 2015, pp. 111–112. doi: 10.1145/2740908.2742726.

[14] B. Dong, Y. Zhu, L. Li, and X. Wu, "Hybrid Collaborative Recommendation via Dual-Autoencoder," *IEEE Access*, vol. 8, pp. 46030–46040, 2020, doi: 10.1109/ACCESS.2020.2979255.

[15] S. Guo, X. Liao, G. Li, K. Xian, Y. Li, and C. Liang, "A Hybrid Recommender System Based on Autoencoder and Latent Feature Analysis," 2023, doi: 10.3390/e25071062.

[16] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Jul. 2018, pp. 974–983. doi: 10.1145/3219819.3219890.

[17] R. Guo, "Enhancing Movie Recommendation Systems Through CNN-Based Feature Extraction and Optimized Collaborative Filtering," *Applied and Computational Engineering*, vol. 81, no. 1, pp. 134–140, Nov. 2024, doi: 10.54254/2755-2721/81/20241080.

[18] L. Han, H. Wu, N. Hu, and B. Qu, "Convolutional Neural Collaborative Filtering with Stacked Embeddings," Oct. 15, 2019, *PMLR*. Accessed: Jun. 14, 2025. [Online]. Available: https://proceedings.mlr.press/v101/han19a.html

[19] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, Jan. 2009, doi: 10.1155/2009/421425.

[20] E. Zangerle and C. Bauer, "Evaluating Recommender Systems: Survey and Framework," *ACM Comput Surv*, vol. 55, no. 8, 2023, doi: 10.1145/3556536.

[21] O. Kuchaiev and B. Ginsburg, "Training Deep AutoEncoders for Collaborative Filtering," Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.01715

[22] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2014, Accessed: Jun. 13, 2025. [Online]. Available: https://arxiv.org/pdf/1412.6980