Performance of Covariance Neural Networks on Rating Prediction

Timothy Axel

Supervisor(s): Elvin Isufi, Andrea Cavallo, Chengen Liu

EEMCS, Delft University of Technology, The Netherlands

Abstract

Recommender systems help users navigate vast catalogs of content through recommendations, of which rating prediction remains an important task. Traditional methods such as collaborative filtering often struggle to model higher-order relationships between users and items, as well as suffer from the cold start problem when the number of users and items is still low. Graph Neural Networks (GNNs) have shown promise in this area, although they are often limited by their focus on local graph structures. This study explores the application of Covariance Neural Networks (VNNs) for rating prediction, leveraging covariance matrices to leverage global statistical dependencies and model higher-order relationships. Using the MovieLens-100k dataset, we evaluate the performance of VNNs against baselines and other models, using RMSE as the metric of evaluation. Our results demonstrate that VNNs outperform simple matrix completion techniques, but are limited by their susceptibility to oversmoothing. This work highlights the potential of VNNs for recommender systems while underscoring the need for careful architectural design to balance performance and stability.

1 Introduction

Recommender systems play an important role in many online applications, helping users in navigating large catalogs of items, including movies, books, or other products. One of the main tasks for these systems is rating prediction, where new, undiscovered items to the user are assigned ratings based on predictions of how the user might have rated the item.

Traditional approaches to tackle this task have included neighborhood-based methods[1], matrix factorization[2], and other pattern-based methods centered around user-item interaction (otherwise known as collaborative filtering). However, these methods fail when dealing with complex, higher-order relationships, which are often found in real-world user preferences and item attributes. Moreover, they become subject to the cold-start and user preference transfer problems, which happen to new users and users with changing preferences, respectively.

Graph-based approaches have been introduced to address the cold-start and preference transfer limitations previously outlined, falling under the moniker of GNN (Graph Neural Networks) [3]. One promising subspace of GNN are GCNs, or Graph Convolutional Networks, which models the convolution operations found in convolutional neural networks using graph collaborative filters. However, these models are typically limited to capturing local graph structures, as they often involve aggregation of the attributes and interactions of neighboring nodes, and thus may not fully utilize the statistical dependencies present in user-item relationships.

This research aims to explore the application of Covariance Neural Networks (VNNs) [4] for rating prediction tasks using the MovieLens-100k dataset[5] as a benchmark, whereupon the covariance matrix is used as a graph collaborative filter.

While variants of VNNs have been proposed to accomplish regression tasks, for instance, Spatiotemporal VNNs for time-series predictions [6], bias-mitigating data processing using Fair VNNs [7], and a variant compatible with sparse matrices (Sparse VNN) [8], research on VNNs in rating prediction tasks remains open to exploration, motivating this study.

1.1 Contributions

This study aims to make the following contributions:

1. **Performance Evaluation of Covariance Neural Networks for Rating Prediction**. We propose techniques to create rating predictions using VNNs, as well as an evaluation of the performance of such a model. The model takes, as input, user-item matrices generated from each set, and makes predictions for each row of users. Performance is then evaluated using RMSE, as is often used when benchmarking performance of rating prediction systems.
2. **Stability of VNNs for Rating Prediction**. We evaluate the effect of perturbations to see if the stability advantage of VNNs could transfer to rating prediction tasks.

1.2 Background

Collaborative Filtering

Collaborative Filtering (CF) is a widely-popular technique in building recommender systems that leverages past user-item interactions to make predictions. Often, CF methods do not require auxiliary information such as user demographics or item metadata, making them suitable for large-scale domains where such data may be unavailable or sparse.

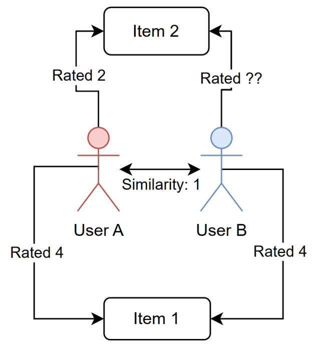

Figure 1: An example scenario in collaborative filtering, using user similarity to create ratings. User A and User B are very similar. Thus, if User A rates Item 2 with a low rating, User B is likely to do the same.

There are two main families of CF techniques: memory-based and model-based methods. Memory-based approaches, such as user-based or item-based nearest neighbor algorithms, compute similarities between users or items using a distance measure, such as cosine similarity or Pearson correlation [9]. For instance, in user-based CF, recommendations are made to a user based on ratings provided by similar users, pictured in Figure 1. These approaches have the advantage of being easy to interpret, but may face scalability issues as the number of users and items grow.

Despite their success, traditional CF methods often face limitations when dealing with higher-order interactions, non-linear relationships, and dynamic user behavior (for example, shifting preferences). They assume that preferences can be modeled using linear combinations and often fail to capture the context or structure behind interactions. Additionally, they struggle with cold-start problems (when a user or item has few or no interactions). These shortcomings motivate the exploration of more structure-focused models, such as Graph Neural Networks, which provide a principled way to incorporate structure and underlying relationships into the recommendation process.

Graph Neural Networks in Recommender Systems

Graph Neural Networks (GNNs) have emerged as a powerful framework for learning from graph-structured data. In recommender systems, GNNs enable the modeling of complex user-item relationships by treating interactions as edges in a graph, with users and items as nodes. This graph structure allows models to capture relational information across nodes, paving the way for better collaborative-filtering techniques. One key technique utilized by GNNs is message passing, which aggregates information across nodes that are within a set distance from a source node in order to update its representation.

Covariance Neural Networks (VNNs)

Covariance Neural Networks (VNNs) [4] were initially introduced as a more stable alternative to Principal Component Analysis (PCA), capable of achieving similar performance whilst being stable to perturbations in the input. This is achieved through the architecture's unique way of modeling relationships. Instead of capturing local, first-order interactions, VNNs utilize second-order relationships, particularly the covariance, via their graph collaborative filters. This helps the model learn larger, underlying trends that span either the entire data (global) or across a wide area (for instance, a neighborhood of features). This difference in paradigm likewise contributes to the VNN's stability to perturbation, as it no longer needs to rely on local relationships that could be absent from its input. This advantage is what we hope to extract from the use of VNNs in rating prediction tasks. In a space where user preferences constantly shift and missing interactions are often the norm, stability to perturbation translates to better generalizability of the model's performance.

VNNs can be seen as performing a two-stage transformation on its input. First, they construct a covariance representation of node features, before applying learnable transformations to the aforementioned representation facilitated through multilayer perceptrons (MLPs). This design enables for learning covariance-aware embeddings, which can be used for rating prediction tasks. A diagram of this can be seen in Figure 2.

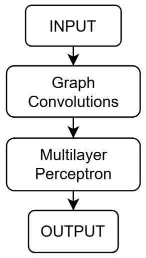

Figure 2: Diagram of the VNN architecture

The use of covariance matrices also make VNNs inherently more robust towards sparse or incomplete data, a common scenario in recommender systems. In traditional collaborative filtering approaches, missing entries pose a challenge for prediction tasks, especially for new users or less popular items. VNNs are able to mitigate this problem by abstracting learning away from individual entries, and instead leveraging global (or neighborhood) structures and their characteristics, possibly leading to improved generalization.

In the context of rating prediction, the ability to model complex statistical dependencies presents an advantage. User preferences often hold underlying structures not immediately obvious through ratings alone, such as preference to certain genres. This would often not be encoded into the graph of traiditonal models unless manually and intentionally done so, even though they are reflected in the relationships between users and items. By using covariance-based filters, VNNs can uncover and exploit these patterns to make more accurate predictions. While the topic of VNNs itself is still an emerging subject, their theoretical advantages have already inspired adaptaions, as mentioned in section 1. Much like these adaptations, we expect to harness some of the advantages born of VNNs in our rating prediction model.

Covariance for Collaborative Filtering

Although previous work on rating prediction tasks using VNNs remains scarce, there have been attempts to utilize the covariance matrix for collaborative filtering. Xiao, et. al [10] demonstrated that the covariance between items or users can be used as a similarity measure to accomplish rating prediction tasks using a memory-based approach. Thus, we hope to create a model that provides the best of both worlds by combining the advantage of stability native to VNNs with the ability to capture more complex relations through covariance distance measures.

2 Methodology

This section describes the approach taken to design a rating prediction model using VNNs, and the subsequent evaluation of its performance.

2.1 Pipeline

To leverage VNNs in rating prediction tasks, we propose the following steps: preprocessing, model training, and evaluation. In the initial preprocessing step, we split the dataset into train, test, and validation sets. The training set is then used for two purposes: the computation of the covariance matrix for our VNN's graph shift operator (GSO) in order to perform convolutions, and the creation of input and output sets for model training. Model training then begins with the creation of a VNN using our computed covariance matrix. Using the LocalGNN [11] architecture, we create a model that is fed interaction matrices created in the preprocessing step, and evaluate its predictions on their respective masked truths. The best model according to validation loss is then fed the test set to obtain the model's generalization performance. Finally, we introduce perturbations in the graph structure to evaluate the model's stability. The following subsections delve into more detail regarding key steps in this pipeline.

2.2 Covariance Matrix Computation

The covariance matrix is calculated in the preprocessing step, using only data from the training set to prevent data leakage. The covariance matrix is at the core of any VNN, acting as the graph by which connections between different users or items are modeled. The covariance itself measures how two random variables change together. A positive covariance implies the variables tend to increase or decrease together, while negative covariance signal that the variables tend to move in opposite directions. Subsequently, a covariance close to zero indicates that two variables are independent, uninfluenced by changes in the other variable. A covariance matrix may be constructed by computing covariances between each user-user pair in the standardized interaction matrix, using the formula:

$$Cov(x, y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{N} \tag{1}$$

Here x and y represent the rating sets of two users, with $\bar{x}$ and $\bar{y}$ representing their respective means. N is the number of ratings per user. As the interaction matrix between users and items are often sparse, NaN ("Not a Number") values populate much of the matrix. In cases where a NaN value is present, imputation was done using zeroes. An example covariance matrix's shape is given in Table 1.

$Cov(u_1, u_1)$	$Cov(u_1, u_2)$	$Cov(u_1, u_3)$	$Cov(u_1, u_4)$
$Cov(u_2, u_1)$	$Cov(u_2, u_2)$	$Cov(u_2, u_3)$	$Cov(u_2, u_4)$
$Cov(u_3, u_1)$	$Cov(u_3, u_2)$	$Cov(u_3, u_3)$	$Cov(u_3, u_4)$
$Cov(u_4, u_1)$	$Cov(u_4, u_2)$	$Cov(u_4, u_3)$	$Cov(u_4, u_4)$

Table 1: The covariance matrix for a dataset with 4 users. Note that the diagonals are simply the variance of the users and that the matrix is symmetric.

2.3 VNN Model Architecture

As previously mentioned in subsection 2.1, our approach uses the LocalGNN architecture to create a VNN. The architecture consists of two types of layers: The Graph Filtering Layers (GFLs) and the readout layer. The GFL layers perform graph convolutions on the input matrix, applying message passing from each node to nodes within a set number of hops. The model performs graph convolutions using the covariance matrix as its graph shift operator, represented as [4]:

$$\mathbf{H}(\hat{\mathbf{C}_n})\mathrm{x} \tag{2}$$

Here, x represents the input, $\hat{\mathbf{C}_n}$ the covariance matrix, and $\mathbf{H}(\hat{\mathbf{C}_n}) = \sum_{k=0}^{m} h_k \hat{\mathbf{C}}_n^k$, the covariance filter of the graph. This results in node embeddings for each user, which is fed into a multilayer perceptron in the readout layer to generate predictions. A diagram of the architecture can be found in Figure 3

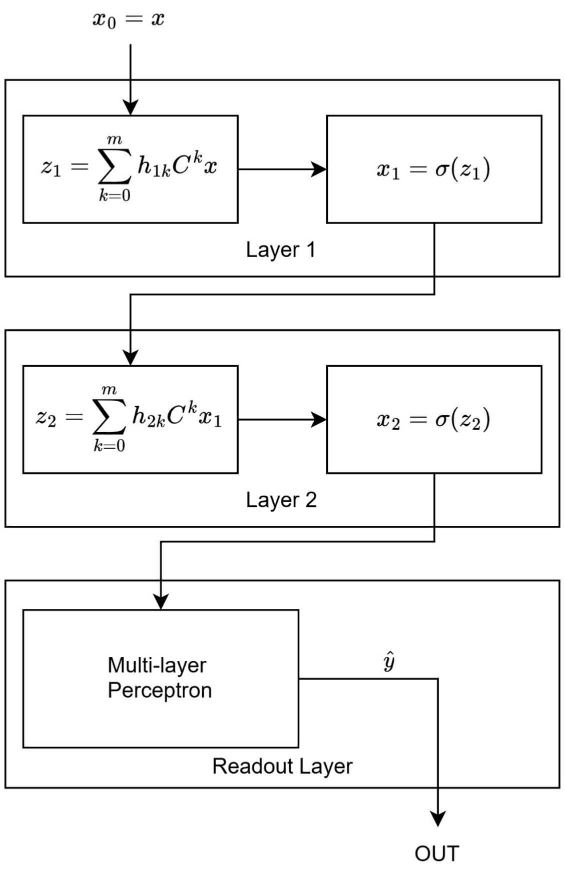

Figure 3: The architecture of a 2-layer VNN. x refers to the input interaction matrix. At each layer, a non-linearity activation function σ is used on the convolution output. The final layer, the readout layer, has a multi-layer perceptron that has been abstracted for brevity.

2.4 Evaluation Metrics

The model's performance is evaluated using Root Mean Squared Error (RMSE). This is a standard metric used in most learning applications, and the better a model performs, the lower its RMSE will be. RMSE is given as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \bar{x}_i)^2}{N}} \tag{3}$$

Here, x stands for the set of true values, and $\bar{x}$ the set of predictions. N is the number of predictions. This metric was chosen over another popular alternative, Mean Absolute Error (MAE), as RMSE punishes large outliers more strictly, thanks to the squaring operation it uses.

During training, the model uses Mean Squared Error (MSE) loss, due to its differentiable properties useful in backpropagation. MSE is given as:

$$MSE = RMSE^2 = \frac{\sum_{i=1}^{N} (x_i - \bar{x}_i)^2}{N} \tag{4}$$

x represents the set of true values, and $\bar{x}$ the set of predictions. N is, once again, the number of predictions. Much like RMSE, the lower the MSE is, the better the model performs.

2.5 Algorithm to Introduce Edge Removal Perturbations

To investigate the stability property of VNNs, we utilize the following algorithm to introduce perturbations:

1. Train the model on the dataset.
2. Create a new covariance matrix, using a reduced sample from the training data.
3. Change the graph structure of the trained model by changing its GSO and replacing it with the newly-computed covariance matrix
4. Use the modified model to predict the test set again and log its performance.

This algorithm simulates missing data–as is often present in real-world data, allowing us to observe how performance degrades when less data is available for the VNN to learn.

2.6 Algorithm to Introduce Perturbations for Robustness Analysis

To investigate the effect of the covariance matrix collaborative filter, we propose the following algorithm to introduce perturbations:

1. Create a covariance matrix using a sample taken from the training data.
2. Train the model, using the perturbed covariance matrix as the GSO.
3. Evaluate the model as usual on the test set.

3 Experimental Setup

3.1 Dataset Selection

We conduct our experiments on the MovieLens-100k dataset [5], which consists of 100,000 ratings made by 943 users towards 1682 movies. This choice was motivated by the dataset's widespread use in rating prediction tasks, allowing for easier comparability with other related works. Ratings are extracted from the dataset into an interaction matrix, each cell representing a rating for a given user-movie pair.

3.2 Preprocessing Methods

In this subsection, we outline our approach for splitting data and shaping input and outputs for model training.

Splitting

The dataset is first split into training, validation, and test sets randomly, with an 8:1:1 split among each set, respectively. This training set will serve two purposes: to be used to create covariance matrices, acting as the graph shift operator of our VNN architecture, and to create input (X) and output (y) matrices for each epoch of model training. Meanwhile, the validation set helps select the best model to evaluate on the test set, yielding a final, comparable score. For our sets, we utilize the 'u1' split of the MovieLens-100k dataset to provide better comparability with other methods.

Masking

To create input and ground truth for model training, an interaction matrix is first created using ratings from the training set. Some values are then masked from this interaction matrix to be used as the true values for evaluation and backpropagation. The remainder of the interaction matrix is used as the input matrix for forward passes. The shape of each model input is the same, at n movies * m users. An example of this masking operation can be found in Figure 4

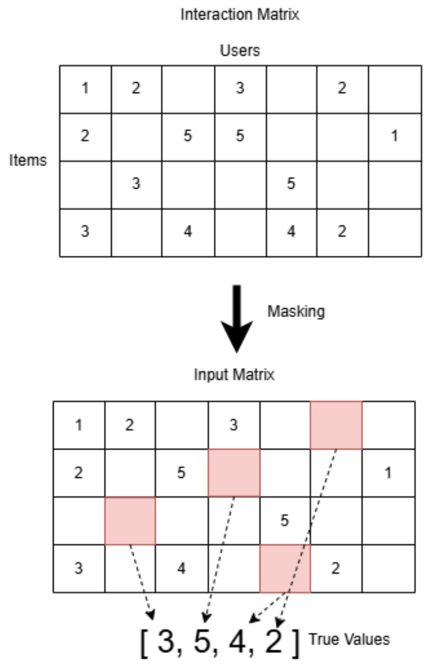

Figure 4: An example masking operation for an interaction matrix of 4 items and 7 users. The empty cells in the matrix represent missing data. The red cells represent masked values, each connecting to a corresponding true value, which is used for evaluating predictions

3.3 Hyperparameter Search

Model tuning and hyperparameter search is done using the Optuna library [12]. Given the large search space, we utilize random search over other, more exhaustive methods, such as grid search, in hyperparameter optimization. To increase robustness of the results, we perform 3 runs using each combination of hyperparameters during the search, across 50 total combinations randomly sampled from the space. For the sake of reproducibility, the hyperparameters, along with their final values are:

1. $[1, 64, 64]$ for the size of the node embeddings of each layer of the GNN. This means the final VNN has 2 layers with signals of size 64 for each node.

2. $[64, 32, 1]$ for the number of nodes per each layer of the MLP at the readout layer of the model.

3. $[4, 4]$ as k for each layer's $(k-1)$-hop neighbors to consider. This means nodes within 3 hops are considered for message passing.

4. LeakyReLu as the non-linearity function used during training.

5. Adam optimizer with a weight decay of 0.0001 and a learning rate of 0.01.

6. 100 as the number of epochs.

7. 0.9 as the evaluation ratio, which means 0.9 of the training data is used as the input interaction matrix in model training, and 0.1 for ground truth used in evaluation.

3.4 Baseline Selection

To verify that the model is properly learning the dataset, we create the following baselines, keeping in mind the unique properties of the dataset.

1. Mean Prediction. All predictions are the global mean. This is useful given the distribution of the dataset, as seen in Figure 5, where most of the ratings are centered around 3 and 4.

2. Random Prediction. Predictions are made randomly, using the ratings' values [1,2,3,4,5] as discrete options. This serves as our upper bound, as a model that is unable to beat this prediction is very likely not learning.

3. Identity GSO. This simulates a situation where no neighborhood information is propagated – effectively, each node sees only itself. In this case, the GNN essentially acts like an MLP on isolated features, allowing us to investigate the effects of the covariance structure when compared to the final results.

4. All-ones GSO. The graph in this baseline is fully-connected and uniform, so every node's features are averaged together during filtering. This global smoothing adds useful aggregate information, which might improve prediction (much like the mean prediction baseline),

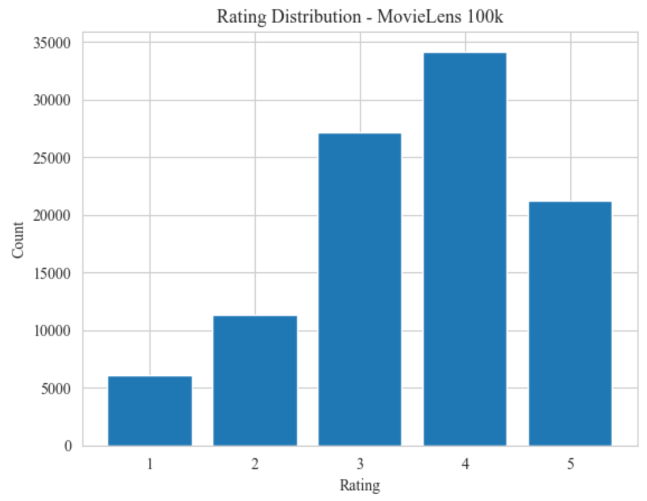

Figure 5: Distribution of the Ratings in the MovieLens-100k dataset

4 Results and Discussion

4.1 Model Performance

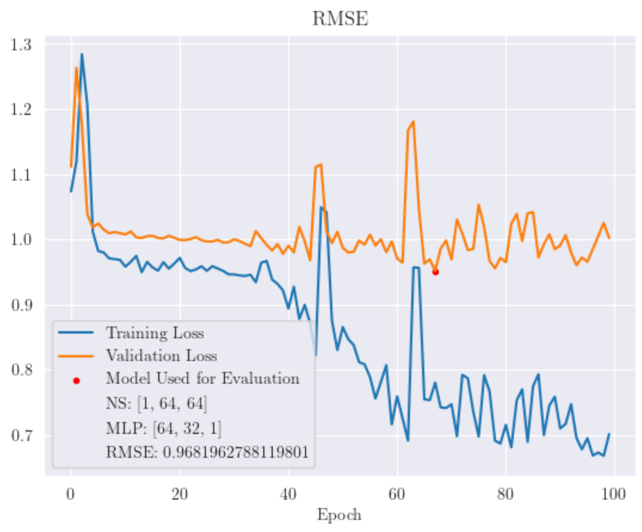

Figure 6: One of the runs with the best-found hyperparameters. 'NS' shows the dimensions of the node signals for this run, and 'MLP' the dimensions of the multilayer perceptron. Additionally, the test RMSE found for this run is printed in the legend.

Model	RMSE
VNN (Ours)	0.9682
Random Predictions	1.8762
Global Mean	1.1537
Identity GSO	1.0695
All-ones GSO	0.9823

Table 2: Performance in RMSE compared to baselines

Following 5 runs with the best hyperparameters (an example run is shown in Figure 6, we obtain an RMSE of **0.9682 with a standard deviation of 0.00553**. This is an improvement over the two rudimentary baselines (shown in Table 2), implying that the VNN is indeed capable of learning relationships from the dataset. This hypothesis is also supported by the trends in model loss during training. As the model goes through more epochs, it is able to reduce the training and validation loss, up to a point where only the training loss keeps decreasing, at which point overfitting seems to have occurred. The VNN has also managed to beat the identity and all-ones baselines, providing evidence that these results cannot be entirely attributed to the readout layer of the VNN.

Model	RMSE
VNN (Ours)	0.968
MC [13]	0.973
IMC [14]	1.653
GMC [15]	0.996
GRALS [16]	0.945
sRGCNN [17]	0.929
GC-MC [18]	0.905

Table 3: Comparison of our VNN's performance with other models, with respect to RSME performance.

Comparing our VNN's performance to other models (shown in Table 3) yields interesting conclusions. The VNN is able to outperform some models (such as Matrix Completion [13] and Graph Matrix Completion [15]), but fails to beat other models (notably Graph Convolutional Matrix Completion (GC-MC) [18]). This seems to be in line with the complexity of the model architectures. The VNN is essentially performing matrix completion when it generates its output matrix, and although it is able to beat simple variants of matrix completion techniques, it struggles to outperform models utilizing larger, more complex architectures that capture more relationships. For instance, sRGCNN [17] combines a Multigraph Convolutional Neural Network (MGCNN) with a Recurrent Neural Network, while using unweighted 10-nearest neighbor graphs for training. 10-nearest neighbor graphs were also used in the Graph Regularized Alternating Least Squares (GRALS) model, potentially hinting that the VNN is missing a lot of relationships from this.

Furthermore, although hypothesized as having a contrary effect, given the high sparsity of the dataset as shown in Figure 7, modeling first-order relationships (user-item interactions, as is the case with the bipartite graphs in GC-MC) may be better than second-order relationships for this dataset, as global trends could be too noisy due to a lack of overlapping data, reducing the model's generalizability.

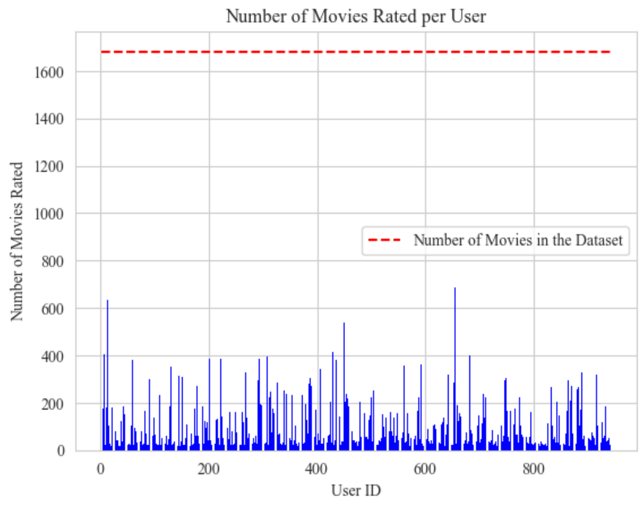

Figure 7: Number of ratings made by each user.

4.2 Robustness Analysis

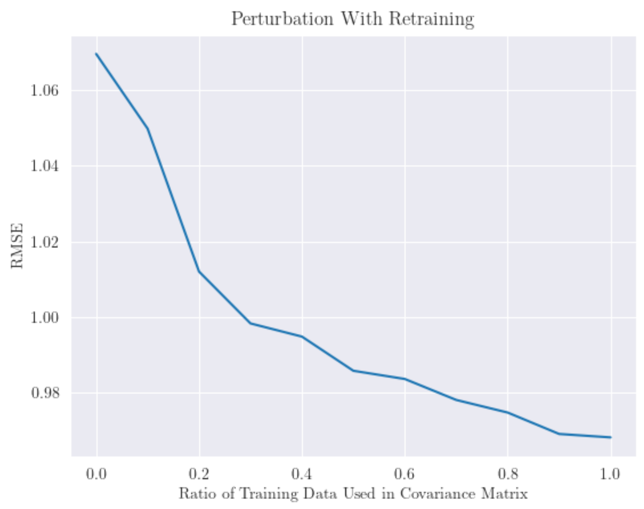

Figure 9: The performance of the VNN model as more data from the training set is supplied to create the covariance matrix for its GSO.

We observe a noticeable difference as the covariance matrix becomes less noisy, showing a decrease in RMSE of almost 0.1. This further bolsters the statement that the VNN is leveraging higher-order relationships to make predictions.

4.3 Stability Analysis

We observe vastly different outcomes based on the k-hop constraints. While the 1-hop VNNs present relatively stable outcomes irregardless of the number of graph filtering layers, the 2- and 3-hop-neighbor VNNs exhibit interesting patterns. While RMSE generally trends downwards for 1-hop neighbors as the training data becomes less perturbed, 2- and 3-hop-neighbor VNNs have interesting peaks that rise and fall as less perturbation is introduced. This is most likely caused

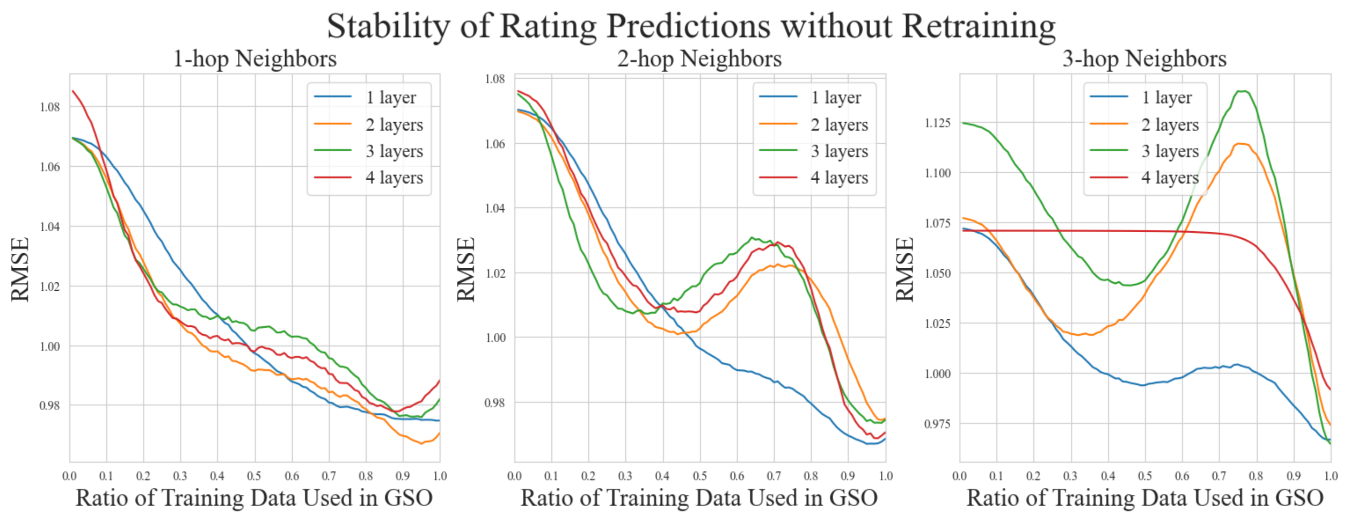

Figure 8: Effect of Perturbation on VNNs of different numbers of layers and k-hop constraints. For instance, each VNN in the 1-hop Neighbors graph only has layers that consider each node's surrounding 1-hop neighbors for message passing.

by oversmoothing [19], where node features converge as network depth is increased. This is supported by the fact that this behaviour is hardly seen in any of the 1-layer instead becoming generally more pronounced when more hops are introduced. Additionally, incorporating a larger k-hop seems to have a positive effect towards the final RMSE, signifying a tradeoff between stabililty and performance. Overall, the findings suggest a sensitivity in the VNN's structure, where extra care needs to be observed such that the model does not oversmooth. As was hypothesized earlier in subsection 4.1, this seems to support the idea that very sparse data leads to poorer performance for the VNN, due to its reliance on well-formed covariance matrices, which is difficult when data does not overlap or is missing a lot of values.

5 Conclusions and Future Work

5.1 Conclusions

This study investigated the use of Covariance Neural Networks in rating prediction tasks, particularly on movie ratings from the MovieLens-100k dataset, demonstrating their ability to capture higher-order relationships present in user-item interactions. Our experiments showed that VNNs achieve competitive performance in this area, outperforming basic matrix completion methods, but falling behind complex, sophisticated models such as GC-MC. Additionally, stability analysis on VNNs suggest a susceptibility to oversmoothing, particularly in deeper architectures.

5.2 Limitations

One limitation is the reliance on transductive learning methods [20] in the pipeline. Transductive methods require retraining to perform predictions on unseen items or users, resulting in poor scalability. Additionally, the sparse nature of real-world interaction matrices poses a challenge for covariance-based approaches, as imputation techniques can introduce bias.

5.3 Future Work

Future work could explore inductive methods in lieu of the transductive method explained in this study, in order to increase robustness and scalability. Inductive methods also tend to outperform transductive methods [21], which opens an interesting avenue for further research. Additionally, combining the VNN architecture with other GNN variants could enhance performance, as seen from the models that beat the VNN's performance. Testing on other datasets of varying sparsity could prove interesting as well, potentially providing deeper insights into the model's generalizability. On the matter of imputation, better techniques to reduce bias could be considered, including Expectation-Maximization [22]. Finally, regularization techniques to overcome the oversmoothing problem could be investigated to improve stability.

6 Responsible Research

Reproducibility and repeatability are at the forefront of considerations on the topic of ”Responsible Research”. In this study, several steps have been taken to promote this as much as possible. One of the ways this has been done is through a nigh-exhaustive description of every step taken in creating rating prediction models using VNNs, including detailing splits and masking procedures in great detail. A repository of the main pipeline can also be found at https://github.com/TimothyAxel/Rating_Prediction_with_VNNs. Results are kept as robust as possible through repeated, random initializations, allowing other researchers to obtain similar results provided they run the model enough times to minimize bias and variance. While the dataset used in this study consists of user ids and their preference of movies, the dataset has been scrubbed and anonymized before publication, mitigating the risk of data risks and leaks. Moreover, the dataset has proven reliable in the past, being used in many different studies prior to this one.

References

[1] A. N. Nikolakopoulos, X. Ning, C. Desrosiers, and G. Karypis, "Trust your neighbors: A comprehensive survey of neighborhood-based methods for recommender systems," 2021.

[2] D. Bokde, S. Girase, and D. Mukhopadhyay, "Matrix factorization model in collaborative filtering algorithms: A survey," *Procedia Computer Science*, vol. 49, pp. 136–146, 2015. Proceedings of 4th International Conference on Advances in Computing, Communication and Control (ICAC3'15).

[3] B. Khemani, S. Patil, K. Kotecha, and S. Tanwar, "A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions," *J. Big Data*, vol. 11, Jan. 2024.

[4] S. Sihag, G. Mateos, C. McMillan, and A. Ribeiro, "covariance neural networks," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 17003–17016, Curran Associates, Inc., 2022.

[5] F. M. Harper and J. A. Konstan, "The MovieLens datasets," *ACM Trans. Interact. Intell. Syst.*, vol. 5, pp. 1–19, Jan. 2016.

[6] A. Cavallo, M. Sabbaqi, and E. Isufi, "Spatiotemporal covariance neural networks," *arXiv [cs.LG]*, 2024.

[7] A. Cavallo, M. Navarro, S. Segarra, and E. Isufi, "Fair covariance neural networks," 2025.

[8] A. Cavallo, Z. Gao, and E. Isufi, "Sparse covariance neural networks," 2024.

[9] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," *Proceedings of ACM World Wide Web Conference*, vol. 1, 08 2001.

[10] Y. Xiao, J. Shi, W. Zheng, H. Wang, and C.-H. Hsu, "Enhancing collaborative filtering by user-user covariance matrix," *Math. Probl. Eng.*, vol. 2018, pp. 1–9, Nov. 2018.

[11] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 1034–1049, 2019.

[12] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[13] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comut. Math.*, vol. 9, pp. 717–772, Dec. 2009.

[14] P. Jain and I. S. Dhillon, "Provable inductive matrix completion," 2013.

[15] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix completion on graphs," 2014.

[16] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[17] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," 2017.

[18] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017.

[19] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," 2023.

[20] G. Lachaud, P. Conde-Cespedes, and M. Trocan, "Comparison between inductive and transductive learning in a real citation network using graph neural networks," in *2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 534–540, 2022.

[21] S. S. Ziaee, H. Rahmani, and M. Nazari, "MoRGH: movie recommender system using GNNs on heterogeneous graphs," *Knowl. Inf. Syst.*, vol. 66, pp. 7419–7435, Dec. 2024.

[22] Q. Ma and S. K. Ghosh, "Emflow: Data imputation in latent space via EM and deep flow models," *CoRR*, vol. abs/2106.04804, 2021.