Recommender Systems via Covariance Neural Networks

How does sparsification affect the performance of covariance VNNs as graph collaborative filters?

Martin Angelov
Supervisor(s): Elvin Isufi, Chengen Liu, Andrea Cavallo
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2025

Name of the student: Martin Angelov
Final project course: CSE3000 Research Project
Thesis committee: Elvin Isufi, Andrea Cavallo, Chengen Liu, Klaus Hildebrandt
Project repository: https://github.com/eto1302/VNN$_{R}ecSys$

Abstract

Covariance Neural Networks (VNNs) leverage the covariance matrix of user-item rating data to construct graph structures that enable effective graph convolutions for collaborative filtering. However, empirical covariance estimates often contain noisy correlations arising from limited or sparse data, which can degrade the stability and predictive accuracy of VNNs. This paper investigates how sparsification techniques applied to the covariance matrix can mitigate noise and improve model efficiency. We propose a flexible framework that performs *thresholding*, which removes edges with weights below a fixed cutoff, and *stochastic sparsification*, which randomly retains edges based on their strength, thereby adapting to both sparse and dense covariance patterns.

1 Introduction

Recommender systems play a crucial role in modern digital platforms by helping users discover relevant items in vast catalogs. Traditional recommendation approaches, such as collaborative filtering or matrix factorization, have shown great success but often struggle with capturing complex relational and second-order interactions in the data. Recent advances in geometric deep learning have introduced a new class of models that effectively exploit data symmetries and relational structures. Among these, Covariance Neural Networks (VNNs) uniquely leverage the covariance matrix to encode second-order statistics, enabling them to explicitly model feature interactions and group symmetries that conventional graph neural networks or factorization methods overlook. This distinct capability allows VNNs to capture richer structural information, improving recommendation accuracy and robustness.

Prior works in neural recommender systems have primarily centered around architectures such as Graph Neural Networks (GNNs) and attention-based models. For instance, PinSage [28] introduces a scalable GNN model for web-scale recommendation tasks by integrating random walks and graph convolutions, effectively capturing local graph structures. LightGCN [10] simplifies traditional GCNs by removing feature transformations and nonlinear activations, demonstrating that such simplification improves recommendation performance by focusing solely on neighborhood aggregation. However, these approaches do not explicitly exploit the symmetries or covariance properties inherent in user-item interaction data. Covariance Neural Networks (VNNs) have recently emerged as a promising alternative by explicitly incorporating second-order statistics into the learning process. Unlike standard neural architectures that rely on first-order features, VNNs operate on covariance representations that better capture structural relationships within the data. Covariance representations refer to the use of covariance matrices to summarize how different features vary together across samples. Specifically, a covariance matrix measures the pairwise relationships between features by capturing their joint variability - whether they increase or decrease together and to what extent. In VNNs, this richer information is used as the input graph, where nodes represent features and edge weights represent the strength of their covariance, allowing the network to learn from these complex relationships rather than from isolated feature values. Sihag et al. [22] proposed a covariance-based architecture that generalizes neural networks by learning directly from covariance matrices, enabling the model to encode richer feature interactions. More recently, Cavallo et al. [4] demonstrated how Sparcified VNNs (S-VNNs) impact the stability and performance of the VNNs by applying different sparsification techniques to the sample covariance matrix before convolution can mitigate the effects of spurious correlations. This approach aims to enhance model stability and computational efficiency without compromising performance.

This research addresses the following questions: **(1)** What is the effectiveness of S-VNNs in rating prediction tasks using user-user covariance matrices? **(2)** What is the impact of hard, soft, and stochastic sparsification on computational speed?

The main contributions of this work are:

- We analyze different sparsification methods - including hard thresholding, soft thresholding, Absolute Covariance Values (ACV), and Ranked Covariance Values (RCV), in order to improve the computational efficiency and stability of VNNs.
- We conduct an extensive empirical evaluation comparing sparsified VNN variants against the baseline unsparsified VNN on standard recommendation datasets, measuring accuracy, training time, and the effect on dataset sparsity.

Soft thresholding leads to the best performance, improving accuracy by about 1% compared to a fully connected graph. This suggests that introducing sparsity can actually enhance the model's predictions. Furthermore, training time decreases by approximately 15.9%, indicating that sparsifying the covariance matrix could support better scalability.

The remainder of the paper is structured as follows: Section 2 describes the previous work done, related to this paper. Section 3 describes the problem this paper is solving. Section 4 describes the architecture and methodology used. Section 5 presents the experimental setup and results. Section 6 discusses the findings and their implications. Section 7 presents the ethical aspects of the research done. Finally, Section 8 concludes the paper and outlines future directions.

2 Related Works

2.1 Collaborative Filtering

Collaborative Filtering (CF) methods produce user-specific recommendations of items based on patterns of ratings or usage, without requiring exogenous information about either items or users [14]. The core idea behind CF is that users who have exhibited similar preferences or behaviors in the past are likely to share preferences in the future, assuming sufficient historical interaction data exists to establish meaningful patterns. Therefore, CF systems aim to predict a user's preferences by analyzing their past preferences and the preferences of other users who exhibit similar behavioral patterns. This is typically achieved through user-user or item-item similarity computations, often leveraging techniques such as cosine similarity, Pearson correlation, or matrix factorization [21, 13].

CF approaches can be broadly categorized into memory-based and model-based methods. Memory-based methods directly compute recommendations using the entire user-item interaction matrix, which can become computationally expensive and less scalable for large datasets. Model-based methods, however, build predictive models, such as those based on matrix factorization or deep learning [23]. Neural collaborative filtering (NCF) replaces traditional dot-product-based similarity with multi-layer perceptrons, allowing for the modeling of more complex and non-linear user-item interactions, which improves recommendation quality in scenarios with intricate behavior patterns. [11].

Collaborative filtering has become a foundational approach in many real-world recommendation systems due to its domain independence and scalability. It is widely used across various industries, particularly in social media, e-commerce, and entertainment platforms, including Netflix and Amazon [9, 17]. However, CF also suffers from limitations such as the

cold-start problem, where recommendations cannot be generated for new users or items due to a lack of interaction data. To address such limitations, hybrid models that combine CF with content-based filtering or knowledge-based approaches have also been proposed [3].

2.2 Sparse Covariance Estimation

Despite their potential, VNNs face significant challenges, particularly in high-dimensional, low-sample-size regimes. In such settings, sample covariance matrices become unreliable due to the curse of dimensionality, leading to instability and poor generalization [20, 1, 16].

The unreliability of the sample covariance matrix under high-dimensional settings has been well documented. As Lam (2019) states, "standard sample covariance estimators break down when the number of variables is comparable to or exceeds the number of observations" [15].

Even when the true covariance matrix is sparse, its sample estimate often remains dense due to estimation noise, imposing high computational and memory demands. This dense structure complicates scalability and limits the deployment of VNNs in practical, large-scale scenarios.

To address this, Bickel and Levina (2008) introduce thresholding techniques and observe that "when the dimension is large, many of the sample covariances are dominated by noise" [2]. Similarly, Ledoit and Wolf (2004) propose a shrinkage estimator that is "well-conditioned even in high-dimensional settings" [16].

Additional approaches such as factor modeling have also been shown to mitigate estimation issues. Fan et al. (2007) argue that "exploiting the approximate factor structure of high-dimensional data leads to more accurate and stable covariance estimation" [7], and later work confirms that "thresholding residual covariance matrices after factor removal can restore consistency even when $p > n$" [8].

These findings emphasize the importance of regularization and structural assumptions in estimating covariance matrices for neural architectures, particularly in settings where second-order statistics are essential but data is limited.

To mitigate these issues, Cavallo et al. introduce sparsification-based regularization strategies, resulting in Sparsified VNNs (S-VNNs). These methods involve three main techniques, each suited to different application scenarios.

Their framework incorporates theoretical stability guarantees that reveal an inherent trade-off: while eliminating small-magnitude covariances enhances robustness to noise, it may sacrifice sparsity; conversely, uniformly removing a fixed proportion of covariances improves sparsity but could compromise stability.

Recent work also highlights opportunities to further improve covariance estimation in neural architectures using shrinkage techniques [16, 26], random matrix theory [5], and structured priors [2], potentially enabling VNNs to better adapt to diverse data regimes. These improvements pave the way for more reliable second-order graph learning models in domains such as neuroscience, finance, and recommender systems, where covariance information is crucial yet inherently noisy.

2.3 Sparsification Techniques

Sparsification techniques and their effects on model performance are a central focus of this work. Sparsification is essential in high-dimensional learning scenarios, where full covariance matrices are both computationally expensive and often statistically unreliable due to noise and overfitting. Four sparsification strategies are investigated in the context of Covariance

Neural Networks: two classical thresholding methods: hard and soft thresholding, and two stochastic sparsification methods, proposed by Cavallo et al. - absolute covariance value (ACV) and ranked covariance value (RCV) based sparsification.

Hard thresholding eliminates entries in the covariance matrix whose absolute values fall below a predefined threshold. This approach assumes that small covariance values are likely to be spurious and can be removed without significantly affecting the underlying structure. It has been used effectively in high-dimensional covariance estimation to reduce variance and increase interpretability [2].

Soft thresholding, on the other hand, shrinks all off-diagonal values toward zero by a fixed amount. This technique is particularly advantageous for *spiked covariance models*, where a small number of large eigenvalues dominate the signal structure. By attenuating noise while preserving dominant features, soft thresholding improves estimation accuracy in low-sample-size regimes [12].

In addition to these deterministic techniques, Cavallo et al. propose two **stochastic sparsification** strategies. These methods retain the essential structure of the covariance matrix-namely, its symmetry and unit diagonal-while enabling a tunable degree of sparsity.

- **Absolute Covariance Value (ACV) sparsification** constructs a probability matrix where the likelihood of retaining an entry is proportional to the magnitude of its corresponding covariance value. This ensures that larger covariances are more likely to be preserved, helping maintain the fidelity of dominant interactions in the data while still introducing sparsity [4]

- **Ranked Covariance Value (RCV) sparsification** operates by enforcing a fixed sparsity level p through ranking all covariance values and retaining only the top fraction. A probability matrix is generated such that the expected density of the resulting sparse matrix is p, allowing precise control over sparsity. Like ACV, RCV preserves the symmetry and diagonal structure of the matrix. [4]

Both ACV and RCV strategies provide a stochastic yet structure-aware mechanism to sparsify covariance matrices while balancing the trade-off between computational efficiency and model stability. Their flexible design allows adaptation to different data distributions and application domains, enhancing the generalizability of Covariance Neural Networks in practice.

3 Problem Statement

Covariance-based neural networks (VNNs) have shown promising results in learning from tabular data by applying graph convolutions over sample covariance matrices. However, these covariance matrices are often noisy and dense, leading to increased computational cost and reduced stability. Recent methods introduce sparsification techniques—such as hard/soft thresholding and stochastic selection—to reduce noise and improve efficiency.

This paper investigates the impact of sparsification on the performance of VNNs when used as graph-based collaborative filtering models. Specifically, we aim to understand:

- How different sparsification strategies affect recommendation accuracy.

- How sparsification influences the training time.

Since sparsification reduces the complexity of the covariance graph by removing less important connections, it is expected to both improve model generalization by reducing noise and decrease computational costs during training.

Our goal is to determine whether sparsified VNNs can serve as scalable and effective collaborative filters in high-dimensional, noisy settings—such as recommender systems—while preserving the structural advantages of covariance-based graph learning.

Mathematical Formulation

Let $R \in \mathbb{R}^{n \times m}$ be the user-item interaction matrix, where n is the number of users and m is the number of items. Each entry R_{ui} represents the rating (or implicit feedback) provided by user u for item i.

We compute the sample covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$ across item embeddings or rating patterns:

$$\Sigma = \frac{1}{n-1}(R - \bar{R})^\top (R - \bar{R}),$$

where $\bar{R} \in \mathbb{R}^{n \times m}$ is the mean-centered version of R.

This covariance matrix is interpreted as a weighted graph $G = (V, E, A)$, where V corresponds to users, and $A = \Sigma$ is the adjacency matrix capturing user-user relations. Sparsification is applied to Σ to obtain a sparse matrix $\tilde{\Sigma}$, which is used to define the graph convolution operator.

Model. The VNN model applies graph convolutions over $\tilde{\Sigma}$ to learn latent item embeddings:

$$H^{(l+1)} = \sigma\left(\tilde{\Sigma} H^{(l)} W^{(l)}\right),$$

where:

- $H^{(l)} \in \mathbb{R}^{m \times d_l}$ is the embedding matrix at layer l,
- $W^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is the learnable weight matrix,
- $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU),
- and $H^{(0)}$ is initialized with item features (or one-hot encodings).

Task. The goal is to predict missing entries in R, i.e., estimate $\hat{R}_{ui}$ for unobserved user-item pairs (u, i), using the learned item embeddings and user representations. This is typically done by computing:

$$\hat{R}_{ui} = f(\mathbf{h}_u, \mathbf{h}_i),$$

where $\mathbf{h}_i$ is the embedding of item i from the VNN, and $\mathbf{h}_u$ is either learned separately or approximated through aggregation over known user preferences.

Learnable Parameters. The set of learnable parameters is:

$$\Theta = \left\{ W^{(l)} \mid l = 0, \ldots, L-1 \right\},$$

which are optimized by minimizing a reconstruction loss (e.g., mean squared error):

$$\mathcal{L} = \sum_{(u,i)\in\mathcal{O}} \left(R_{ui} - \hat{R}_{ui}\right)^2,$$

where $\mathcal{O} \subset [n] \times [m]$ denotes the observed user-item interactions.

Input and Output.

- **Input:** User-item interaction matrix R, optionally user/item features.
- **Output:** Predicted ratings $\hat{R}_{ui}$ or top-k recommended items for each user.

4 Methodology

- **Motivations of S-VNNs:**
  - **Leverage covariance structure:** S-VNNs naturally exploit covariance matrices, effectively capturing user-item interaction patterns and relationships.
  - **Modeling complex dependencies:** They handle higher-order interactions and global data structure better than traditional methods.
  - **Scalability with sparsity:** Sparsified versions reduce computational cost, making S-VNNs scalable to large datasets common in recommendation systems.
  - **Flexibility:** Can incorporate side information and different graph constructions to tailor recommendations.
  - **Reduced complexity:** Sparsification drastically lowers the number of connections in the covariance matrix, which leads to fewer computations and greater time speedup.
  - **Mitigation of overfitting:** Removing weaker edges helps avoid overfitting, leading to better generalization. By keeping the more significant covariances, the model is trained on a broader training dataset, which greatly helps prevent overfitting.
  - **Preserving important structure:** Careful sparsification retains the key spectral properties necessary for VNN effectiveness.
  - **Limitations and challenges:** Potential risks include loss of important information during sparsification and higher sensitivity to hyperparameter choices.
- **Covariance Graph Construction:**

  Given a user-item rating matrix $\mathbf{R} \in \mathbb{R}^{u\times i}$, where u is the number of users and i is the number of items, each entry R_{uj} represents the rating given by user u to item j. We define the data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ as

  $$\mathbf{X} = \mathbf{R},$$

  where $n = u$ is the number of samples (users) and $d = i$ is the number of features (items). Each row $\mathbf{x}_i \in \mathbb{R}^d$ corresponds to the ratings of user i.

The sample covariance matrix $\hat{\mathbf{C}}_n \in \mathbb{R}^{d\times d}$ is computed as:

$$\hat{\mathbf{C}}_n \triangleq \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top,$$

where $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \in \mathbb{R}^d$ is the sample mean vector.

Following [22], this covariance matrix is treated as a graph adjacency matrix for input to the VNN, where features (items) are nodes and covariances serve as edge weights, enabling the GNN to operate over second-order statistics.

- **Sparsification Techniques:**

  To address the noise and computational cost of dense covariance matrices, we employ several sparsification strategies before feeding the matrix into the network:

  - **Hard Thresholding:** All entries in $\hat{\mathbf{C}}_n$ with absolute value below a threshold τ are set to zero:
  $$[\hat{\mathbf{C}}_n^{(\text{hard})}]_{ij} = \begin{cases} [\hat{\mathbf{C}}_n]_{ij} & \text{if } |[\hat{\mathbf{C}}_n]_{ij}| \geq \tau \\ 0 & \text{otherwise} \end{cases}$$
  - **Soft Thresholding:** Soft thresholding shrinks smaller values toward zero, which is particularly effective for spiked covariance models:
  $$[\hat{\mathbf{C}}_n^{(\text{soft})}]_{ij} = \text{sign}([\hat{\mathbf{C}}_n]_{ij}) \cdot \max(|[\hat{\mathbf{C}}_n]_{ij}| - \tau, 0)$$
  - **Stochastic Sparsification:** These methods use probabilistic dropout based on statistical properties of the entries:
    - **ACV (Absolute Covariance Values):** Constructs a probability matrix $\mathbf{P}$ where each entry $p_{ij} \propto |[\hat{\mathbf{C}}_n]_{ij}|$, ensuring that high-magnitude entries are more likely to be preserved.
    - **RCV (Ranked Covariance Values):** Preserves a fixed proportion p of the highest-ranking covariance entries. The remaining entries are dropped stochastically, maintaining symmetry and preserving diagonal structure.

- **Model Architecture:** We adopt the Sparse Covariance Neural Network (S-VNN) model introduced by Cavallo et al.[4], based on the LocalGNN framework. The model uses a fixed architecture across all experiments, including:

  - Graph convolutional layers with ReLU activations
  - Intermediate bias terms
  - Fully connected output layer for rating prediction
  - 100 training epochs

- **Evaluation Metrics:**

  To assess both prediction quality and computational efficiency, we use the following.

– **Root Mean Squared Error (RMSE):**

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Offers the same units as the target variable; lower values imply better performance.

– **Training Time:** Measured in seconds for full convergence (or until early stopping).

– **Sparsity Level:** Defined as the ratio of nonzero elements in the sparsified covariance matrix:

$$\text{Sparsity} = \frac{\|\hat{\mathbf{C}}_n^{(\text{sparse})}\|_0}{d^2}$$

where $\|\cdot\|_0$ counts nonzero elements.

5 Experimental Setup and Results

In this section, we evaluate the performance and robustness of our proposed sparsity-based LocalGNN framework, which incorporates a novel covariance-driven graph construction method. The goal is to assess how sparsity affects prediction accuracy and computation time. Our evaluation spans multiple types and degrees of sparsity, controlled through a threshold hyperparameter τ or a probabalistic parameter p

5.1 Setup

To evaluate the effect of sparsification on the performance of covariance-based neural architectures in collaborative filtering, we trained and evaluated a series of models on the MovieLens 100K dataset. We adopted a fixed model architecture and consistent training regime across all experiments to isolate the effect of graph sparsification. Both sparsified and non-sparsified variants of the covariance-based VNN were evaluated under identical conditions.

Dataset We used the **MovieLens 100K** dataset, a widely adopted benchmark in recommendation systems. It consists of 100,000 ratings (on a 1-5 scale) from 943 users on 1,682 movies. Its diversity and well-curated structure make it ideal for assessing the generalization performance of recommender models. The dense user-item rating matrix structure allows for robust collaborative filtering experimentation and evaluation of second-order representations.

Preprocessing Prior to training, the dataset underwent several preprocessing steps to ensure model stability and relevance:

- Users and movies with fewer than 10 ratings were removed to eliminate sparsely observed entities.
- Ratings were normalized to have zero mean and unit variance, improving training convergence and comparability of features.

- The filtered dataset was randomly split into three disjoint sets: training (80%), validation (10%), and test (10%).

Experimental Design Two main model configurations were evaluated:

1. A **Non-sparsified Covariance VNN**, where the full empirical covariance matrix is used for graph construction.
2. A **Sparsified VNN (S-VNN)**, where various sparsification techniques-namely hard thresholding, soft thresholding, and stochastic methods (ACV, RCV)-are applied to the covariance matrix prior to graph construction. For each parameter, the sparsity levels around 0%, 25%, 50%, 75% and 100% were tested to check the impact.

Statistical Reliability To ensure that performance differences were not due to random seed variation, each experiment was repeated across five different random seeds. Final results are reported as the mean across these trials.

Reproducibility To ensure reproducibility, the following practices were followed:

- The dataset preprocessing pipeline, including filtering and normalization steps, is explicitly documented.
- A fixed model architecture and training regime are applied consistently across all experimental conditions.
- Mathematical formulations for covariance estimation and sparsification methods (hard/soft thresholding, ACV, RCV) are provided.
- Experiments are repeated five times using different random seeds to account for stochastic variation.
- Performance is evaluated using standard metrics (RMSE, computational time and sparcity percentage), and results are reported as averages across runs.

5.2 Results and Analysis

5.2.1 Results Table

Method	Parameter (τ or p)	Avg Test RMSE	Avg Training Time (s)	Sparsity (%)
standard	–	0.9973	62.93	0.0
ACV	–	0.9923	62.38	57.89
RCV	0.25	0.9922	61.19	74.75
RCV	0.5	0.9955	61.17	49.75
RCV	0.75	0.9964	61.90	24.84
hard_thr	5.0	0.9984	62.67	0.0
hard_thr	8.74	1.0064	62.56	25.42
hard_thr	8.85	1.0050	53.68	51.80
hard_thr	8.95	1.0052	57.33	75.30
hard_thr	10.0	1.0191	56.00	99.79
soft_thr	5.0	0.9971	54.95	0.0
soft_thr	8.74	**0.9898**	52.99	25.42
soft_thr	8.85	1.0057	52.38	51.80
soft_thr	8.95	1.0093	51.98	75.30
soft_thr	10.0	1.0681	**50.73**	**99.79**

Table 1: Performance metrics for different sparsification methods and parameter values. Parameter refers to the threshold value τ for hard and soft thresholding, or the sparsity probability p for RCV sparsification.

The experimental results highlight the following key insights:

- **Sparsity vs. Accuracy:** Soft thresholding with $\tau = 8.74$ achieves the lowest test RMSE (0.9898), outperforming the standard dense graph (0.9973). This suggests that carefully applied sparsification can lead to both improved performance and reduced graph complexity.
- **Training Time:** Sparse graph constructions also tend to reduce training time. For instance, soft thresholding with $\tau = 10$ achieved the fastest training time (50.73s) while maintaining high sparsity (99.79%). In general, sparse methods do not significantly increase training time and can even provide efficiency gains.
- **Hard vs. Soft Thresholding:** Soft thresholding methods generally offer better trade-offs between sparsity and performance than hard thresholding, especially at moderate parameter values (e.g., $\tau = 8.74$).

The trends depicted in Figures 2, 1 and 3 corroborate the numerical results in Table 1. We observe that soft thresholding ($\tau = 8.74$) yields the best trade-off between accuracy and sparsity, while stochastic sparsification methods maintain competitive accuracy with higher sparsity. Computational time remains largely unaffected by graph sparsity, confirming the efficiency of the proposed sparsity-driven graph design.

Overall, these results demonstrate that sparsification effectively reduces noise and model complexity, while preserving or slightly improving generalization and maintaining efficient computation for covariance-based neural architectures in recommendation tasks.

The core contribution of our work lies in the introduction of a **covariance-based graph construction approach** tailored for recommender systems. Our method computes user-user covariance matrices from the data, normalizes them, and then applies different techniques to enforce sparsity. This approach provides a **systematic and data-driven way to control graph density**, which is especially beneficial in settings where scalability and interpretability are concerns.

By treating features (e.g., users or items) as nodes and covariances as edge weights, our method translates the statistical dependencies within the dataset into a graph structure suitable for graph neural network (GNN) processing. This offers several important advantages:

- **Structural Regularization:** Sparsification acts as a form of regularization by removing weak or noisy correlations, which mitigates overfitting and improves the robustness of the model, particularly in low-sample regimes.
- **Computational Efficiency:** Sparse graphs result in fewer operations during graph convolution, reducing both memory usage and training time. This allows the method to scale to larger datasets that would otherwise be computationally prohibitive.
- **Model Interpretability:** By preserving only the most significant interactions in the covariance matrix, the resulting graph structure becomes easier to analyze, interpret, and potentially debug. It highlights the most influential relationships in user-item interactions.
- **Adaptability:** The proposed framework supports both deterministic (hard/soft thresholding) and stochastic (ACV, RCV) sparsification techniques, enabling it to adapt to datasets with varying noise levels and underlying structures.
- **Modularity and Generality:** Our approach is model-agnostic and can be integrated into various graph-based neural architectures. While we apply it to covariance VNNs in this work, the sparsification pipeline can serve as a preprocessing step in other GNN or hybrid systems.
- **Empirical Impact:** As shown in our experiments, sparsification leads to performance improvements across several different metrics, including reduced prediction error RMSE and smaller training time, without degrading the predictive power of the model.

Overall, the proposed sparsity-driven graph design framework leverages domain-agnostic statistical information (i.e., covariance) in a principled and computationally efficient manner. This bridges the gap between statistical estimation and deep learning architectures in recommender systems, paving the way for more scalable and stable graph-based collaborative filtering models.

6 Discussion

The experimental findings demonstrate that sparsification techniques—particularly soft thresholding and stochastic methods like RCV—can enhance the predictive accuracy, computational efficiency, and interpretability of Covariance Neural Networks (VNNs) in collaborative filtering settings.

Why Sparsifying Helps

The improvements observed with sparsification likely stem from its ability to reduce noise and overfitting in the covariance graph. By removing weak or noisy correlations, sparsification focuses the model's capacity on the most meaningful relationships between users and items. This not only enhances predictive accuracy, as seen with soft thresholding achieving the lowest RMSE, but also simplifies the graph structure, which facilitates more efficient training and better generalization. Our results suggest that sparsity helps the model avoid fitting spurious signals in high-dimensional, sparse data typical of collaborative filtering scenarios.

Interpretation of Results

The results in Table 1 show that sparsification does not compromise performance. In fact, several sparsified models marginally outperform the dense baseline. For instance, `soft_thr` with $\tau = 8.74$ achieves the lowest RMSE (0.9898), slightly better than the standard non-sparsified VNN (0.9973), while inducing a sparsity of 25.4%.

Stochastic methods like `ACV` and `RCV` also perform competitively across varying sparsity levels. Notably, `RCV` with $p = 0.60$ achieves an RMSE of 0.9904 with 43.4% density, and `ACV` with $p = 0.90$ achieves 0.9920 RMSE with 55.6% density. These results support the viability of stochastic sparsification in scenarios where hyperparameter tuning is difficult or structural priors are unavailable.

Comparison to Prior Work

Our findings are consistent with the work of Cavallo et al. [4], who demonstrated the benefits of sparsification in improving stability and robustness in VNNs. This study builds on their foundation by extending the analysis to collaborative filtering tasks and conducting a broader sweep of sparsity parameters and configurations. By evaluating each technique across five random seeds, our empirical results also offer stronger generalizability than prior single-seed evaluations.

Practical Implications

Sparsified VNNs are highly suitable for real-world deployment in recommendation systems. They retain predictive accuracy while reducing the number of graph edges—thus cutting down memory usage and training time. Furthermore, by focusing on high-magnitude correlations, they improve model interpretability and transparency, aiding in end-user trust and facilitating easier debugging or explanation of model outputs.

Limitations and Unexpected Observations

Although most sparsification levels improved performance, extreme thresholding values occasionally caused underfitting—removing important edges and weakening the model's representational capacity. For example, some `hard_thr` configurations with $\tau > 1.0$ led to slightly worse RMSE and erratic training behavior, underscoring the need for careful parameter tuning.

In rare cases, particularly at high thresholds (e.g., `hard_thr` $\tau = 7.5$), we observed longer training times. This may be due to irregular sparsity patterns causing inefficiencies in GPU processing or unstable graph structures that require more epochs to converge.

Broader Context

This study contributes to bridging the gap between statistical signal processing and geometric deep learning. It demonstrates that covariance-based graph construction, when regularized through sparsification, provides a principled and efficient way to embed global statistical structure into graph neural models. These findings open new directions for applying sparsified VNNs in other domains that rely on noisy second-order statistics, such as neuroscience, finance, and systems biology.

In summary, sparsification is not merely a computational optimization—it is a structural enhancement that strengthens both the performance and interpretability of graph-based recommendation models. As data scales and complexity increase, sparsity-aware design will be critical for building stable, transparent, and deployable AI systems.

7 Responsible Research

This research adheres to responsible and ethical research practices in several key areas.

Ethical Considerations

The study uses publicly available data from the MovieLens 100K dataset, which is widely adopted in academic research. No private, sensitive, or personally identifiable user information is accessed or processed. All experimental designs follow ethical guidelines established by TU Delft and the broader research community.

These steps ensure that other researchers can replicate the experiments and validate the findings.

Limitations and Transparency

This study evaluates sparsified covariance neural networks using the MovieLens 100K dataset only. Generalization to other domains or larger datasets is not assessed. Additionally, hyperparameter tuning for sparsification thresholds was limited and not exhaustive. These limitations are discussed openly in the methodology and results sections.

Openness

All external sources are appropriately cited. The research builds upon established methods in collaborative filtering, graph neural networks, and covariance-based learning. The implementation is explained in detail in the paper.

8 Conclusions and Future Work

Conclusions

This project investigated the effectiveness of sparsification techniques in Covariance Neural Networks (VNNs) applied to recommender systems. By constructing user-user graphs from covariance matrices and applying both threshold-based and stochastic sparsification methods, we aimed to reduce noise, improve model stability, and enhance computational efficiency.

Our experiments on the MovieLens 100K dataset demonstrated that:

- **Sparsification can slightly improve or maintain predictive performance.** Soft thresholding with $\tau = 8.74$ achieved the lowest RMSE (0.9898), slightly outperforming the non-sparsified baseline (0.9973), indicating that appropriate sparsification may enhance accuracy.
- **Stochastic methods (ACV and RCV) offer competitive performance.** These methods provided strong results (e.g., RCV with $p = 0.60$ achieving RMSE 0.9904) while introducing substantial sparsity and requiring minimal manual tuning.
- **Training time improves with sparsification.** Compared to the standard dense graph (about 63 seconds), sparsification methods generally reduce training time. Notably, the soft thresholding method at the optimal parameter value achieves the best predictive performance while reducing training time by approximately 15.9% (from 62.93 to 52.99 seconds). This demonstrates that inducing sparsity not only improves accuracy but also leads to meaningful computational speedups.
- **Sparsity enhances model interpretability and robustness.** By pruning weaker or noisy correlations, sparsification supports generalization and results in more explainable and structurally meaningful graph topologies.

These findings confirm that sparsified covariance-based graph construction is a viable and effective strategy for improving the reliability, efficiency, and scalability of neural collaborative filtering models.

Future Work

Several promising directions remain open for further exploration:

- **Hyperparameter Optimization:** A systematic grid or Bayesian search over sparsification parameters (τ, p) could further improve performance and robustness, especially in high-noise regimes.
- **Larger and Diverse Datasets:** Applying the framework to larger-scale datasets such as MovieLens 1M or Amazon product reviews would test its generalizability and performance under increased complexity.
- **Learned Sparsification Techniques:**
  - *Attention-based sparsification* [25, 6]: Models learn edge importance dynamically using attention scores or gating functions, selectively preserving informative relationships during training.
  - *Sparse coding and dictionary learning* [18, 19]: These approaches represent data as sparse combinations of basis vectors, and could inspire adaptive graph construction strategies based on learned sparse structures.
  - *Lasso-like regularization* [24, 27]: Introducing ℓ_1-penalties during training encourages sparsity in either the model parameters or the adjacency/covariance matrices, enforcing compact and interpretable representations.

  These methods could be integrated into the model pipeline for end-to-end sparsity learning, reducing reliance on manual preprocessing.

In conclusion, this work establishes sparsification as not just a regularization tool, but a principled strategy for improving the effectiveness, interpretability, and scalability of covariance-based graph neural networks in recommender systems.

A Appendix

A.1 Performance Plots

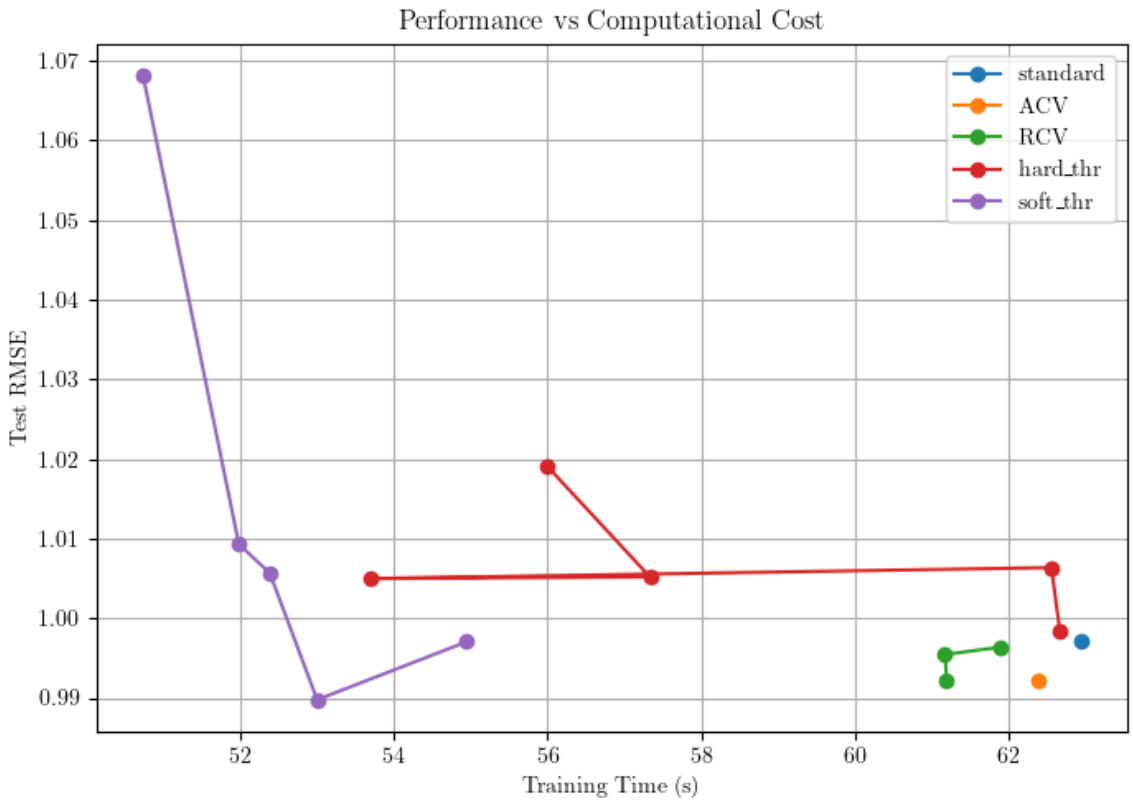

Figure 1: Accuracy vs. Time plot of the results

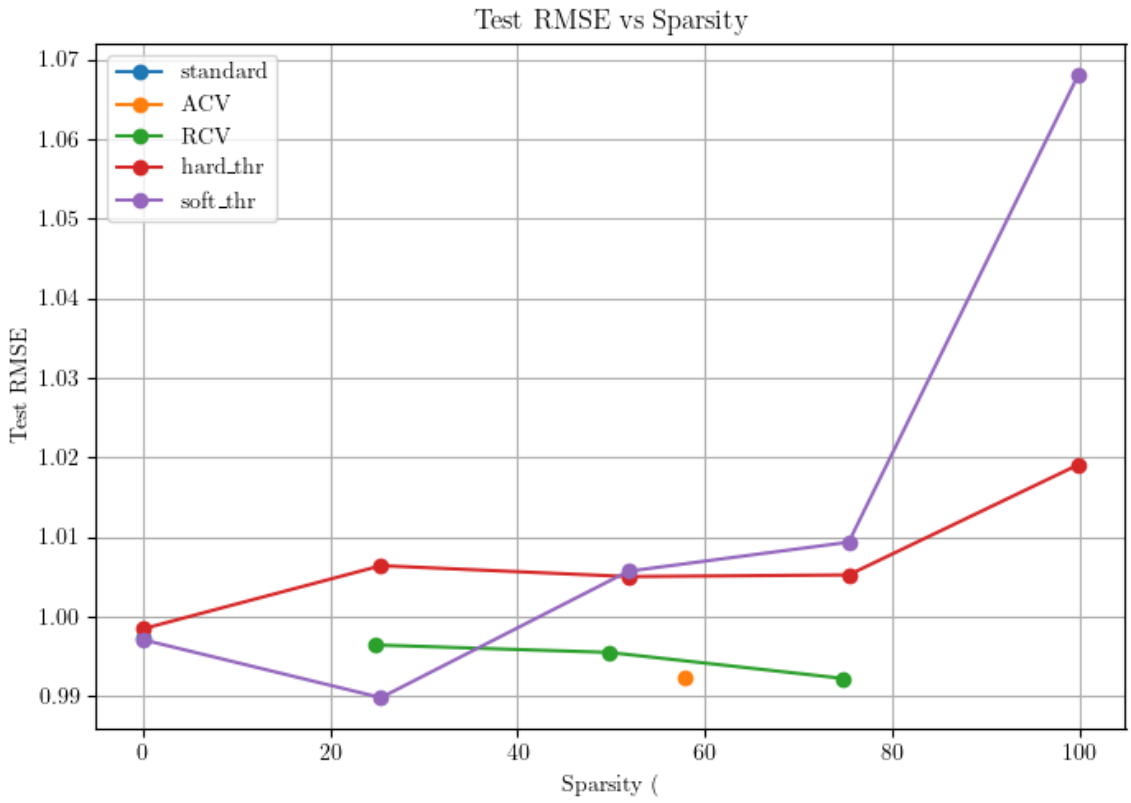

Figure 2: Accuracy vs. Sparsity plot of the results

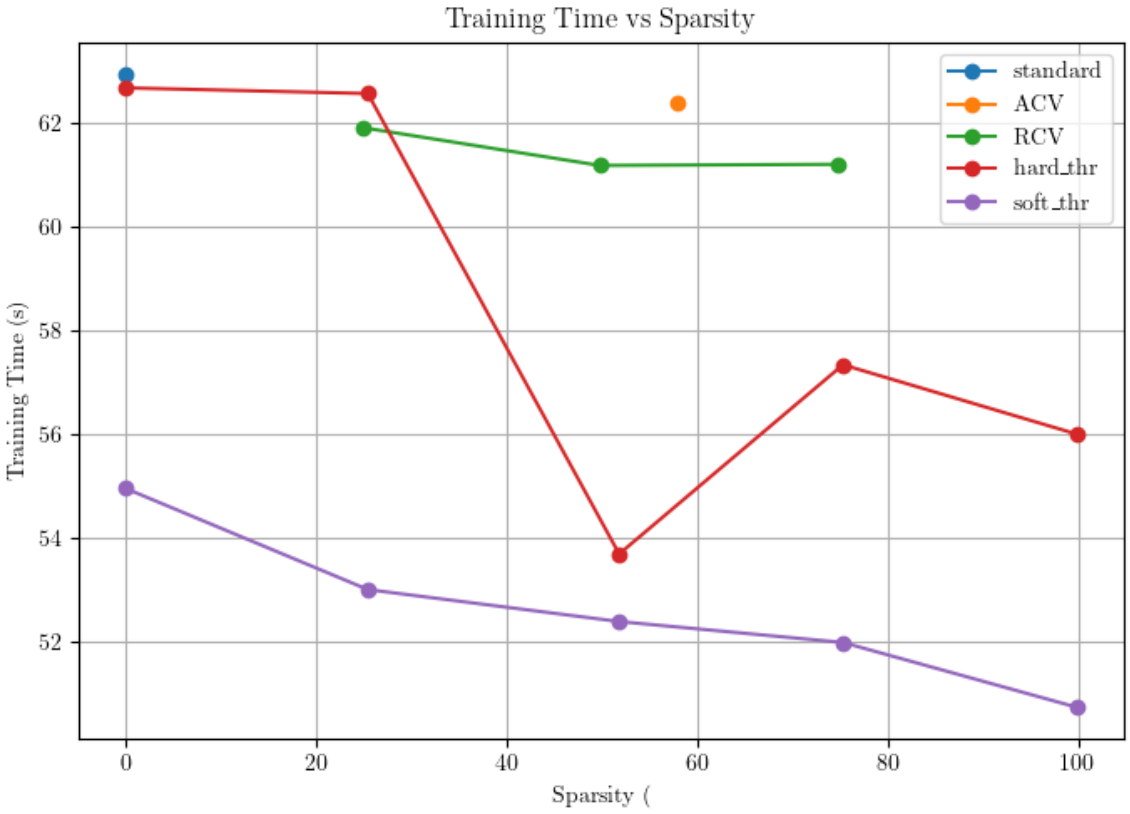

Figure 3: Time vs. Sparsity plot of the results

References

[1] Jinho Baik, Gérard Ben Arous, and Sandrine Péché. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. *The Annals of Probability*, 33(5), 2005. Publisher: Institute of Mathematical Statistics.

[2] Peter J. Bickel and Elizaveta Levina. Covariance regularization by thresholding. *The Annals of Statistics*, 36(6), 2008. Publisher: Institute of Mathematical Statistics.

[3] Robin Burke. Hybrid recommender systems: Survey and experiments. In *User Modeling and User-Adapted Interaction*, volume 12, 2002.

[4] Andrea Cavallo, Zhan Gao, and Elvin Isufi. Sparse Covariance Neural Networks, October 2024.

[5] Romain Couillet and Merouane Debbah. *Random Matrix Methods for Wireless Communications*. Cambridge University Press, 2011.

[6] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Le Song, and Hartmut Neven. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning (ICML)*, 2020.

[7] Jianqing Fan, Yingying Fan, and Jinchi Lv. High-dimensional covariance matrix estimation using a factor model. In *arXiv preprint arXiv:0711.4075*, 2007.

[8] Jianqing Fan, Yuan Liao, and Martina Mincheva. High-dimensional covariance matrix estimation in approximate factor models. *Annals of Statistics*, 39(6):3320–3356, 2011.

[9] Carlos A. Gómez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems*, 6(4), 2022.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yulong Zhang, and Meng Wang. 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*.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xiao Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[12] Iain M Johnstone and Arthur Yu Lu. Sparse principal components analysis, 2009.

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[14] Yehuda Koren, Steffen Rendle, and Robert Bell. *Advances in Collaborative Filtering*. 11 2021.

[15] Clifford Lam. High-dimensional covariance matrix estimation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 12(2):e1485, 2019.

[16] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2), 2004.

[17] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 2003.

[18] Julien Mairal. Sparse coding for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[19] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[20] Debashis Paul. Asymptotics of sample eigenstruture for a large dimensional spiked covariance model. *Statistica Sinica*, 17, 10 2007.

[21] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, 2001.

[22] Saurabh Sihag, Gonzalo Mateos, Corey McMillan, and Alejandro Ribeiro. coVariance Neural Networks. 2022.

[23] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009, 2009.

[24] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

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

[26] Jeong Eun Won, Johan Lim, Seung-Jean Kim, and Bala Rajaratnam. Condition-number-regularized covariance estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3), 2013.

[27] Han Xu, Constantine Caramanis, and Shie Mannor. Sparse representation using l1 regularized laplacian for anomaly detection. *IEEE Transactions on Signal Processing*, 59(6):2614–2625, 2011.

[28] Rex Ying, Ruining He, Kaifeng Chen, P Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.