Lighter Graph Convolutional Networks for Recommendation

Yanping Zheng
Renmin University of China
Beijing, China
zhengyanping@ruc.edu.cn

Zhewei Wei*
Renmin University of China
Beijing, China
zhewei@ruc.edu.cn

Jiajun Liu
Data 61, CSIRO
Pullenvale, Queensland, Australia
jiajun.liu@csiro.au

Frank de Hoog
Data 61, CSIRO
Black mountain, Queensland
Australia

Xu Chen
Renmin University of China
Beijing, China

Yuhang Ye
Jiadeng Huang
Huawei Poison Lab
Shenzhen, China

ABSTRACT

Graph-based recommender models leverage graph structures and Graph Convolutional Networks (GCNs) to model user-item relationships and learn embeddings, improving recommendations' accuracy. However, existing methods frequently require a large number of parameters, complicating training and constraining their applicability in large-scale data contexts. To solve this issue, this paper proposes the Lighter-X framework, an innovative solution that significantly reduces the number of parameters in the model. First, we introduce a low-rank random matrix to replace the conventional input feature and transform the graph-based recommender model into the decoupled GCN framework. Moreover, by applying this strategy to representative models across different domains, we demonstrate its plug-and-play property. It proves to be generalizable to different models, effectively reducing parameter counts and enhancing computational efficiency. We conducted extensive experiments, and the results demonstrate the effectiveness of Lighter-X, which can significantly reduce the number of parameters while achieving comparable performance with base models. In particular, in a real network with millions of interactions, the number of parameters for Lighter-X is just 1% of LightGCN's, yet it performs better.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Information systems → Information retrieval**.

KEYWORDS

Graph Neural Networks, Recommender Systems, Collaborative Filtering

ACM Reference Format:
Yanping Zheng, Zhewei Wei, Jiajun Liu, Frank de Hoog, Xu Chen, Yuhang Ye, and Jiadeng Huang. 2024. Lighter Graph Convolutional Networks for Recommendation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Zhewei Wei is the corresponding author. The work was partially done at Gaoling School of Artificial Intelligence, Peng Cheng Laboratory, Beijing Key Laboratory of Big Data Management and Analysis Methods and MOE Key Lab of Data Engineering and Knowledge Engineering.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
2024. ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recommender systems are information filtering systems designed to connect users with relevant information. Their primary role is to assist users in discovering useful content and, simultaneously, to ensure that information reaches interested users. Collaborative filtering is a common technique in recommender systems, and the idea behind it is that users with similar past preferences are likely to have comparable future preferences. Conventional approaches to collaborative filtering, such as matrix decomposition, concentrate on generating a hidden vector representation of users and items through the decomposition of the user-item interaction matrix. However, these methods do not utilize the structural information underlying different users and items [24]. To solve this problem, people propose to formulate the user-item interactions as a bipartite graph and enhance the utilization of collaborative filtering information via neighbor convolution. By stacking more convolutional layers, the users and items with longer distances can be associated and share similar propagated gradients in the optimization process [8]. Actually, traditional matrix decomposition methods can be seen as graph-based recommender models with only one convolutional layer.

Despite effectiveness, graph-based recommender models usually contain a large number of parameters and need complex convolutional operations, which hinders their application in real-world scenarios. This problem necessitates the studies of more efficient graph-based recommender models. For example, LightGCN [10] is a simplified graph convolutional network (GCN) designed for recommender systems and performs well in collaborative filtering tasks. It simplifies the model by omitting feature transformations and nonlinear activation functions in traditional GNN-based models such as NGCF [20]. These elements theoretically increase the representational power of the model but, in practice, may complicate the training process and reduce model effectiveness. Especially in recommender systems, each node in the interaction graph usually has only one user or item ID as an input, which makes the complex structure unnecessary. LightGCN removes these complex operations and focuses on the neighbor aggregation process, which reduces the complexity of the model and improves training efficiency and recommendation accuracy.

Although LightGCN has fewer parameters than previous models, being limited to embeddings for each user and item, it still possesses a large number of training parameters, expressed as $n \times d$, where n represents the total number of users and items and d is the dimension of the embedding. This raises some concerns about the practicality of LightGCN in real-world scenarios. Specifically, as the dataset size grows, leading to an increase in n, the number of training parameters in the LightGCN model increases significantly. This escalation in model complexity and size may pose challenges in training and deploying the model effectively in large-scale applications. Recent works introduce polynomial-based filters [9] and Graph Contrastive Learning (GCL) [2, 25] to improve recommendation accuracy. However, these models generally use LightGCN as their backbone network, which would also run into issues of huge parameter size when dealing with large-scale datasets, limiting their practical applications.

In recent years, some work has been made to optimize and simplify models like LightGCN, with the majority of these efforts being centered on approaches derived from matrix decomposition perspectives [13–15]. For example, SVD-GCN [15] uses the technique of Singular Value Decomposition (SVD), which reduces LightGCN [10] parameters by calculating truncated SVD and obtaining a low-rank representation of users and items directly. Although methods such as SVD-GCN [15] demonstrate promising results in reducing parameters and improving performance, computing SVD on large-scale graphs will still be a computationally intensive task requiring large amounts of processing power and memory, which is challenging even with optimized algorithms [5]. **Is it possible to devise a lighter and more efficient method to minimize the number of parameters in recommender systems?** To achieve this, a deeper understanding and analysis of GNNs's operational mechanisms in recommendation tasks is essential.

In this paper, we attempt to provide a positive answer to this question. Our contributions can be summarized as follows:

- Based on a further understanding of GNN models and recommender systems, we propose a lighter framework (**Lighter-X**), which reduces the parameters in a cost-effective way by employing compressed sensing theory, and optimizes the computational efficiency under the decoupled framework to improve the scalability of the model.
- Employing the Lighter-X framework, we improve existing recommender models and construct *LighterGCN*, *LighterJGCF* and *LighterGCL*. Detailed complexity analysis of these models demonstrates that the proposed Lighter-X can effectively reduce model parameter size and computational complexity.
- We conducted extensive experimental validation on several datasets and demonstrated that the proposed method is able to achieve comparable or even better results with a lower number of parameters than the original model.

2 BACKGROUND AND PRELIMINARY

2.1 Notations

A recommender system usually contains a user set U, an item set I, and a user-item interaction matrix $\mathbf{R} \in \{0,1\}^{|U|\times|I|}$, where $\mathbf{R}_{ui} = 1$ indicates that there exists an interaction between user u and item i. The GNN-based recommendation system utilizes graph structures to represent the relationships between users and items. Specifically, the data is constructed into a bipartite graph $G = (V, E)$, where the node set $V = U \cup I$ contains all users and items. The edges in the edge set E represent interactions, defined as $E = R'$, where $R' = \{(u,i)|\mathbf{R}_{ui} = 1, u \in U, i \in I\}$. The adjacency matrix of the bipartite graph is denoted as $\mathbf{A}$. $\mathbf{D}$ is a diagonal degree matrix. Formally, the recommendation task is to estimate user u's preference scores for any item $i \in I$ based on the learned user representation $\boldsymbol{e}_u$ and item representation $\boldsymbol{e}_i$:

$$\hat{\boldsymbol{y}}_{u,i} = f(\boldsymbol{e}_u, \boldsymbol{e}_i), \tag{1}$$

where the function $f(\cdot)$ can be the dot product, cosine similarity, multilayer perceptrons (MLPs), etc.

2.2 Graph Neural Networks

Graphs have attracted a lot of interest in the field of machine learning because of their outstanding expressiveness. GNNs are neural network architectures designed for graph data, aiming to derive embeddings enriched with neighborhood information. Typically, each node on the graph has an h-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^h$, which represents the attribute information of that node, e.g., the user's age, gender, occupation, etc. The features of the n nodes are stacked into a feature matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\top$. The GNN learns a new representation from the input features $\mathbf{X}$ through multiple layers, i.e., $\mathbf{H}^{(0)} = \mathbf{X}$. The most representative work, GCN [12], adopts normalized aggregation along with self-loop updating methods. Consequently, the formula for the $(\ell+1)$-th convolutional layer is expressed as:

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\widetilde{\mathbf{P}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell+1)}\right), \tag{2}$$

where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops, $\mathbf{I}$ refers to the identity matrix, and $\widetilde{\mathbf{D}}$ is the degree matrix of $\widetilde{\mathbf{A}}$. $\mathbf{H}^{(\ell)}$ and $\mathbf{W}^{(\ell)}$ represent the embedding matrix and the learnable weight matrix at layer ℓ, respectively. $\widetilde{\mathbf{P}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ is the normalized adjacent matrix. σ represents the activation function.

The application of conventional GNN models to large-scale graphs is challenging primarily due to the limitations of full-batch training. In order to improve the scalability of GNN models, a series of works decouple the feature propagation in GCNs from training process, such as SGC [22], PPRGo [1] and AGP [19]. In general, feature propagation can be computed by

$$\mathbf{Z} = \sum_{\ell=0}^{L} w_\ell \mathbf{Z}^{(\ell)} = \sum_{\ell=0}^{L} w_\ell \mathbf{P}^\ell \mathbf{X}. \tag{3}$$

where L is the number of layers, and the weight w_ℓ corresponds to the importance of layer ℓ's representation. Each layer's propagation $\mathbf{Z}^{(\ell)}$ is derived from the preceding layer through a transformation, i.e. $\mathbf{Z}^{(\ell)} = \mathbf{P}\mathbf{Z}^{(\ell-1)}$. The initial representation $\mathbf{Z}^{(0)}$ is set to the feature matrix $\mathbf{X}$. Typically, the feature propagation matrix $\mathbf{Z}$ is input into a model, such as a Multilayer Perceptron (MLP), for training on downstream tasks. For instance, in node classification, a two-layer MLP might be used: $\widehat{\mathbf{Y}} = \mathrm{softmax}(\sigma(\mathbf{Z}\mathbf{W}_1)\mathbf{W}_2)$. The predicted probability matrix $\widehat{\mathbf{Y}} \in \mathbb{R}^{n\times|C|}$, and each element $\widehat{\mathbf{Y}}_{vc}$ represents the probability that node $v \in V$ belongs to class $c \in C = \{1, \cdots, |C|\}$. Contrary to directly learning the probabilities, the recommender model concentrates on learning node embeddings. Using a single-layer simple MLP, the embedding matrix $\mathbf{E}$ can be obtained as $\mathbf{E} = \mathbf{Z}\mathbf{W}$.

Table 1: Performance comparison for original and decoupled GNN models. EqualLightGCN denotes the decoupled GNN version corresponding to the original model LightGCN that employs identity matrix as the input feature, EqualJGCF and EqualLightGCL represent the equivalent decoupled versions corresponding to the original JGCF and LightGCL.

Method	MovieLens-1M				LastFM			
	hit@10	mrr@10	recall@10	ndcg@10	hit@10	mrr@10	recall@10	ndcg@10
LightGCN	0.7533	0.4563	0.1688	0.265	0.6088	0.3389	0.1952	0.1878
EqualLightGCN	0.7533	0.4562	0.1689	0.265	0.6083	0.3388	0.1951	0.1877
JGCF	0.7811	0.4822	0.1863	0.2823	0.6279	0.3513	0.2054	0.1971
EqualJGCF	0.7811	0.4822	0.1863	0.2823	0.6279	0.3513	0.2054	0.1971
LightGCL	0.7303	0.447	0.1592	0.2539	0.6295	0.3676	0.205	0.2018
EqualLightGCL	0.7301	0.4471	0.1593	0.254	0.6295	0.3648	0.2064	0.202

2.3 GNN-based Recommender Systems

GNN-based recommender systems can learn more powerful node embeddings by mining collaborative signals from high-order neighbors. NGCF [20] is designed based on the standard GCN architecture. LightGCN [10] simplifies NGCF by removing the weight matrices and the activation function in each layer. This simplification leads to a more efficient and less complex model while still capturing the essential elements of user-item interactions in recommender systems. Formally, the embedding calculation in LightGCN can be represented by:

$$\mathbf{E} = \frac{1}{L+1}\sum_{\ell=0}^{L}\mathbf{E}^{(\ell)} = \frac{1}{L+1}\sum_{\ell=0}^{L}\mathbf{P}^{\ell}\mathbf{E}^{(0)} \tag{4}$$

where L is the number of layers, and $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the propagation matrix. The final embedding matrix $\mathbf{E}$ is computed by aggregating the embeddings across all layers. $\mathbf{E}^{(0)}$ is the initial embedding matrix, which stands out as the sole parameter matrix of LightGCN and serves as the foundational embeddings to derive subsequent layer embeddings $\mathbf{E}^{(\ell)}$. Specifically, the embedding matrix of each layer $\ell \in \{1, \cdots, L\}$ is computed by $\mathbf{E}^{(\ell)} = \mathbf{P}\mathbf{E}^{(\ell-1)} = \mathbf{P}^{\ell}\mathbf{E}^{(0)}$. The iterative application of $\mathbf{P}$ effectively captures information at different neighborhood scopes within the graph. To achieve enhanced recommendation performance, several studies introduce polynomial graph filters [9] and graph contrastive learning [2, 23, 25].

The success of graph collaborative filtering can be attributed to their effective implementation of low-pass filtering. JGCF utilizes Jacobi polynomial bases to approximate graph signal filters, facilitating efficient frequency decomposition and signal filtration. As a result, JGCF enables distinct modeling of low and high-frequency signals, combining the effects of low-frequency ($\mathbf{E}_{low}$) and mid-frequency ($\mathbf{E}_{mid}$) signals to produce the final embeddings $\mathbf{E} = concat(\mathbf{E}_{low}, \mathbf{E}_{mid})$. The low-frequency signal $\mathbf{E}_{low}$ is derived by summing up L embeddings:

$$\mathbf{E}_{low} = \frac{1}{L+1}\sum_{\ell=0}^{L}\mathbf{J}_{\ell}^{a,b}(\mathbf{P})\mathbf{E}^{(0)}, \tag{5}$$

where $\mathbf{J}_{\ell}^{a,b}(x)$ represents the Jacobi basis, defined by:

$$\mathbf{J}_{\ell}^{a,b} = \begin{cases} 1, & \ell = 0 \\ \frac{a-b}{2} + \frac{a+b+2}{2}x, & \ell = 1 \\ \left(\theta_\ell z + \theta'_\ell\right)\mathbf{J}_{\ell-1}^{a,b}(x) - \theta''_\ell \mathbf{J}_{\ell-2}^{a,b}(x), & \ell \geq 2, \end{cases} \tag{6}$$

and

$$\begin{aligned} \theta_\ell &= \frac{(2\ell+a+b)(2\ell+a+b-1)}{2\ell(\ell+a+b)}, \\ \theta'_\ell &= \frac{(2\ell+a+b-1)\left(a^2-b^2\right)}{2\ell(\ell+a+b)(2\ell+a+b-2)}, \\ \theta''_\ell &= \frac{(\ell+a-1)(\ell+b-1)(2\ell+a+b)}{\ell(\ell+a+b)(2\ell+a+b-2)}, \end{aligned} \tag{7}$$

where $a > -1$ and $b > 1$ are parameters to control the signal filter. The mid-frequency signals are obtained by calculating $\mathbf{E}_{mid} = tanh(\beta\mathbf{E}^{(0)} - \mathbf{E}_{low})$), where β is a weighting factor that modulates the impact of low and high-frequency components.

To mitigate the issue of sparse information in recommender systems, some work has introduced contrastive learning to improve the performance of recommendations. The core principle of contrastive learning involves modifying the original graph structure to produce a new set of representation vectors from the augmented graph. The goal is to align these newly generated vectors with those derived from the original graph, while simultaneously distancing the vectors of dissimilar nodes from one another. LightGCL [2] utilizes singular value decomposition and reconstruction to guide data augmentation. Firstly, singular value decomposition is performed on the interaction matrix $\mathbf{R}$ to obtain $\mathbf{R} = \mathbf{U}\mathbf{Q}\mathbf{V}^{\top}$, where $\mathbf{U}$ / $\mathbf{V}$ is a $|U| \times |U|/|I| \times |I|$ orthogonal matrix and $\mathbf{Q}$ is a diagonal matrix storing the singular values of $\mathbf{R}$. The principal components of a matrix are usually associated with top-k singular values, so LightGCL utilizes top-k singular values and singular vectors to construct the perturbed interaction matrix $\hat{\mathbf{R}}$. The perturbed neighbor matrix $\hat{\mathbf{P}} = [[\mathbf{0}, \hat{\mathbf{R}}], [\hat{\mathbf{R}}^{\top}, \mathbf{0}]]$ is then substituted into Equation 4 to obtain the perturbed embedding:

$$\begin{aligned} \hat{\mathbf{E}} &= \sum_{\ell=0}^{L}\hat{\mathbf{E}}^{(\ell)}, \\ \hat{\mathbf{E}}^{(\ell)} &= \hat{\mathbf{P}} \cdot \mathbf{E}^{(\ell-1)}, \end{aligned} \tag{8}$$

where $\mathbf{E}^{(\ell-1)}$ denotes unperturbed embedding of the previous layer, and $\hat{\mathbf{E}}^{(0)}$ is set as $\mathbf{E}^{(0)}$. The $\frac{1}{L+1}$ is omitted because LightGCL employs the sum function for aggregation instead of the mean.

These GNN-based recommender systems have achieved promising results, but also face challenges in scalability. The increase in the number of users and items in the recommender system causes the graph size to grow rapidly, which makes it difficult to train GNNs efficiently. On the other hand, these models explicitly learn an embedding $\boldsymbol{e} \in \mathbb{R}^d$ for each user and item, which causes the parameter size of the model to become very large $\mathbf{E} \in \mathbb{R}^{n*d}$, especially in systems with a large number of users and items. This not only leads to high computational resource demands, but also risks model convergence issues or overfitting. Overall, scalability remains an open research challenge for applying GNNs to large-scale recommender systems with many users and items.

3 EQUIVALENCE BETWEEN LIGHTGCN AND DECOUPLED GNNS

Though LightGCN [10] and decoupled GNN models such as SGC [22], AGP [19] and PPRGo [1] all aim to simplify GNNs, they take different approaches to achieving this. As mentioned earlier, LightGCN simplifies GCN models by removing the weight matrices and nonlinear activation functions in each layer. It aggregates neighbor embeddings and uses the weighted sum of embeddings at each layer to form the final embeddings. This simplification focuses on reducing the complexity of each layer of the graph convolution and how the layers are combined. In contrast, decoupled GNN models [19, 22] simplify GNN computation by decoupling the feature propagation from the model training. This decoupling allows for a more efficient feature propagation process, which can be achieved through the pre-computing or constraining of propagation steps. For example, SGC [22] employs a single propagation step while AGP [19] pre-computes Personalized PageRank vectors. Therefore, in terms of model architecture, the relationship between LightGCN and decoupled GNNs is:

- LightGCN can be seen as a simplified decoupled GNN. It removes weight matrices and nonlinear functions, simplifying the graph convolutions of each layer into sequential linear transformations. This is consistent with decoupled GNNs.
- However, LightGCN is not entirely decoupled, as it requires aggregation of node representations at each layer.

Observation 1. *In terms of embedding learning and model parameters, LightGCN serves as a specialized form of decoupled GNN, where the input feature matrix is set as an identity matrix.*

While LightGCN is not fully decoupled, from the perspective of parameter and embedding learning, it is equivalent to a decoupled GNN. This can be easily proved. If we set w_ℓ in Equation 3 to $1/(L+1)$, and $\mathbf{X} = \mathbf{I}$, where $\mathbf{I}$ denotes an $n \times n$ identity matrix, then Equation 3 can be written as $\mathbf{Z} = 1/(L+1) \sum_{\ell=0}^{L} \mathbf{P}^\ell \mathbf{I}$. Substituting this into the embedding calculation gives $\mathbf{E} = \mathbf{ZW} = 1/(L+1) \sum_{\ell=0}^{L} \mathbf{P}^\ell \mathbf{IW}$, which is the same as Equation 4, where $\mathbf{E}^{(0)}$ corresponds to the parameter matrix $\mathbf{W}$ in decoupled GNNs. This observation aligns perfectly with the statement in recommender systems that **the IDs of users and items are used as input features**. In these systems, users and items have no actual features beyond their IDs, equivalent to a one-hot encoding input. This is analogous to decoupled GNNs with an identity matrix as the feature matrix.

Based on Observation 1, we found that in LightGCN, the computations for multi-layer graph convolutions can be pre-computed by utilizing the decoupled GNN model architecture. This strategy circumvents the need for computationally intensive aggregation operations at each layer. Similarly, the multi-layer graph convolution computations in JGCF [9] and LightGCL [2] can also be precomputed, utilizing the identity feature matrix. To further validate this observation, we conducted a comparison between the original model and its equivalent decoupled GNN version (EqualRecs) on two datasets: MovieLens-1M and LastFM. All Equal models employ the identity matrix precomputation. As shown in Table 1, the experimental results demonstrate that decoupled Equal-X models achieve comparable performance to the original models. It means that we can further improve the efficiency of LightGCN-based recommendation models by applying precomputation techniques.

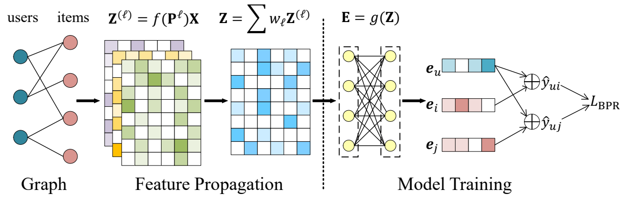
users
items
$\mathbf{Z}^{(\ell)} = f(\mathbf{P}^\ell)\mathbf{X}$
$\mathbf{Z} = \sum w_\ell \mathbf{Z}^{(\ell)}$
$\mathbf{E} = g(\mathbf{Z})$
e_u
e_i
e_j
$\hat{y}_{ui}$
$\hat{y}_{uj}$
L_{BPR}
Graph
Feature Propagation
Model Training

Figure 1: An overview of the decoupled framework for graph-based recommendation.

4 THE LIGHTER-X METHOD

The primary goal of Lighter-X is to reduce the parameter count of the recommendation model. According to Equation 3, the parameter size of GNN depends on the size of the input features. Taking a single-layer simple MLP as an example, in $\mathbf{E} = \mathbf{ZW}$, the size of the weight matrix $\mathbf{W}$ is $h * d$, where h is the dimension of the features on each node $v \in V$, and d represents the dimension of the output representation vector. In LightGCN, the feature matrix $\mathbf{X}$ is an identity matrix of dimension $n \times n$, and the corresponding $\mathbf{W}$ matrix needs to be of size $n \times d$, which is discussed in Section 3. However, the size of the learnable weight matrix being directly related to the number of samples n does not conform to the common principles of machine learning. This limits the model's ability to generalize to unseen data and severely restricts the model's training on large datasets. In order to reduce the scale of $\mathbf{W}$, we propose to use a low-rank matrix instead of the identity feature matrix, which can naturally reduce the number of model parameters to $h \times d$, where $h \ll n$. Therefore, the problem has transformed into finding an appropriate low-rank matrix. **What, then, is the proper low-rank matrix?**

A potential ideal solution. Singular Value Decomposition (SVD) is a common and effective method to obtain a low-rank matrix representation. In SVD-GCN [15], the identity feature matrix is replaced by the top-k singular values of interaction matrix $\mathbf{R}$ and the corresponding singular vectors. This is equivalent to extracting the main features through an all-pass filter after performing the SVD decomposition of $\mathbf{R}$. Consequently, truncated SVD decomposition stands as a potentially optimal top-k approximation method for selective feature filtering, yet it remains computationally intensive. Efficiently computing SVD decomposition on large-scale graphs is still a challenging and unresolved problem [5].

Random sampling approximation: a low-cost compromise solution. A widely adopted approach for swiftly computing SVD is Randomized Singular Value Decomposition (RSVD). In its initial phase, RSVD applies dimensionality reduction to the input matrix through multiplication with a random matrix. This expedited preliminary reduction of dimensions facilitates a more efficient computation of SVD. By employing a random sampling technique, this method approximates the feature space of the original matrix, thereby enhancing computational efficiency while maintaining a high level of accuracy. Therefore, we believe that random sampling is a promising approach for accelerating convolutional operations through effective dimensionality reduction at an early stage in the network. On the other hand, in recommendation systems, data is usually very sparse, and traditional data processing and analysis methods may encounter inefficiency issues [8]. Employing random matrices for rapid dimensionality reduction provides an effective solution. According to compressed sensing theory, even when data is significantly compressed, as long as the signal possesses sparse characteristics, it is possible to accurately reconstruct the original signal from a small number of observations through optimization algorithms [7]. In other words, the compressed matrix can retain sufficient core features of the data.

Based on these observations, we construct a viable input feature matrix $\mathbf{X} = \mathbf{P} \cdot \mathrm{S}$ through random sampling in a cost-effective manner, where $\mathrm{S} \in \mathbb{R}^{n \times h}$ is a random matrix. Since the user-item interactions in recommendation systems can be represented as a bipartite graph, the corresponding $\mathbf{P}$ matrix is a block matrix. Therefore, the input feature matrix $\mathbf{X}$ is expressed as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{0}^{|U|} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{0}^{|I|} \end{bmatrix} \begin{bmatrix} \mathbf{0}^{|U|} & \mathrm{S}_2 \\ \mathrm{S}_1 & \mathbf{0}^{|I|} \end{bmatrix}, \tag{9}$$

where $\mathbf{0}^n$ is a zero matrix of dimension $n \times n$, $\mathbf{B} = \mathbf{D}_u^{-\frac{1}{2}} \mathbf{R} \mathbf{D}_i^{-\frac{1}{2}}$, $\mathbf{D}_u$ and $\mathbf{D}_i$ represent the diagonal degree matrix of users and items, respectively. S_1 / S_2 is a $|U| \times h_1$ / $|I| \times h_2$ random matrix used to compress the matrices $\mathbf{R}$ and $\mathbf{R}^\top$, respectively.

To ensure sampling quality, we follow a rigorous theoretical basis for reconstructing sparse signals, known as the Restricted Isometry Property (RIP), to design suitable random matrices. This usually involves the selection of specific types of random matrices (e.g., Gaussian random matrices, Bernoulli random matrices, etc.) that have been shown theoretically and empirically to be able to satisfy the RIP condition:

$$(1-\delta)\|\boldsymbol{p}\|^2 \leq \|\mathrm{S} \cdot \boldsymbol{p}\|^2 \leq (1+\delta)\|\boldsymbol{p}\|^2, \tag{10}$$

where $\boldsymbol{p}$ is a row in the matrix $\mathbf{B}/\mathbf{B}^\top$ representing the sparse signal vector of user $u \in U/i \in I$. In addition, the RIP constrains the dimension of the random matrix to be related to the sparsity of the data. Assuming r represents sparsity, the dimension of the random matrix h should satisfy:

$$h = c \cdot r \log(n/r), \tag{11}$$

where c is a constant. Compressed Sensing established that a noise-free signal and a sampling matrix S that satisfies RIP can achieve zero error in recovery (complete recovery). In practice the recovery process can be solved by the Basis Pursuit algorithm [3, 4]. This guarantees that the sampled signals ($\mathbf{X} = \mathbf{PS}$) preserve the information from the original signal matrix, and indicates that in our formulation, the sampled signals $\mathbf{B}\mathrm{S}_1$ and $\mathbf{B}^\top\mathrm{S}_2$ fully capture the noise-free $\mathbf{B}$ and $\mathbf{B}^\top$ matrices when S_1 and S_2 both satisfy RIP. Therefore, the resulting low-rank sampled signals could retain sufficient key information from the user-item interaction matrix to support subsequent recommendation tasks.

Decoupled framework for graph-based recommendation. The coupled model structure is another important factor that limits the scalability of traditional GCN [12] and LightGCN [10]. Specifically, these models usually require convolutional operations to be carried out on the entire graph, which is computationally costly on large-scale graphs. Specifically, these models typically require that convolutional operations be performed on the entire graph, which is computationally expensive and not easily achievable on large-scale graph data. A series of studies [1, 19, 22] have proposed to improve the scalability of GCNs by decoupling feature propagation from training process in the original model, allowing the computationally intensive convolution operations to be pre-computed. This implies that the model executes the costly and time-consuming operation just once and obviates performing expensive convolution computations repeatedly during the training process, markedly enhancing the scalability of traditional GCNs. Therefore, we propose to construct graph-based recommendations based on the decoupling framework, as shown in Fig 1. In the feature propagation stage, we complete the convolution related operation and obtain the feature propagation matrix $\mathbf{Z}$. The subsequent neural network takes $\mathbf{Z}$ as input and is trained to generate the final user and item embeddings. This training process is commonly guided by the Bayesian Personalized Ranking (BPR) loss.

4.1 LighterGCN

LighterGCN employs the aforementioned low-rank approximation and decoupling framework to obtain the final embedding matrix. Formally, the way LighterGCN learns embeddings by:

$$\begin{aligned} \mathbf{E} &= MLP(\mathbf{Z}) = MLP(\sum_{\ell=0}^{L} w_\ell \mathbf{Z}^{(\ell)}), \\ \mathbf{Z}^{(\ell)} &= \mathbf{P}^\ell \mathbf{X}, \end{aligned} \tag{12}$$

where $\mathbf{X}$ is the random sampling result obtained Equation 9, and its rank h is much smaller than the number of nodes n. Based on this low-rank input feature matrix $\mathbf{X}$, LighterGCN performs graph convolutional operations to obtain the feature propagation matrix $\mathbf{Z}$. Finally, we train an MLP to get the final embedding $\mathbf{E}$. Therefore, LighterGCN successfully reduces the number of parameters from the original $O(nd)$ to $O(hd)$, where $h \ll n$, and improves computational efficiency significantly.

4.2 Polynomial-based collaborative filtering

Spectral-based methods define graph convolution by introducing filters from the graph signal processing perspective, where graph convolution operations are interpreted as removing noise from graph signals. Polynomial-based graph collaborative filtering is equivalent in form to applying different polynomial bases to compute the aggregation weights for each convolutional layer, such as using Jacobi polynomial bases in JGCF [9]. Under the Lighter-X framework, we can naturally incorporate polynomial-based GCF by aggregating the propagation matrix Z based on different polynomial bases. This allows leveraging the representational power

Table 2: The comparison of time complexity between baseline and proposed models. n, m, $|U|$ and $|I|$ represent the number of nodes, edges, users and items, respectively. B represents the batch size, n_B denotes the number of nodes in a batch, L is the number of layers in the model, d refers to the embedding size, h is the dimension of the feature matrix, and q is the required rank. T denotes the number of iterations in training and is equal to m_{train}/B, where m_{train} is the number of training samples.

Stage		Computation	LightGCN	JGCF	LightGCL	LighterGCN	LighterJGCF	LighterGCL
Pre-processing		Normalization	$O(2m)$	$O(2m)$	$O(2m)$	$O(2m)$	$O(2m)$	$O(2m)$
		SVD	-	-	$O(qm)$	-	-	$O(qm)$
		Graph Convolution	-	-	-	$O(2mLh)$	$O(2mLh)$	$O(2mLh+2qnLh)$
Training	One Batch	t_{conv}: Graph Convolution	$O(2mLd)$	$O(2mLd)$	$O(2mLd+2qnLd)$	$O(3Bhd)$	$O(3Bhd)$	$O(3Bhd+n_Bhd)$
		t_{bpr}: BPR Loss	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$
		t_{ssl}: InfoNCE Loss	-	-	$O(Bd + Bn_Bd)$	-	-	$O(Bd + Bn_Bd)$
	Total		$(t_{\text{conv}} + t_{\text{bpr}} + t_{\text{ssl}})T$					
Inference		Graph Convolution	$O(2mLd)$	$O(2mLd)$	$O(2mLd)$	$O(nhd)$	$O(nhd)$	$O(nhd)$
		Calculate Scores	$O(\|U\|\|I\|d)$	$O(\|U\|\|I\|d)$	$O(\|U\|\|I\|d)$	$O(\|U\|\|I\|d)$	$O(\|U\|\|I\|d)$	$O(\|U\|\|I\|d)$

of varied bases while precomputing the aggregations to reduce computational complexity.

LighterJGCF. We use a low-rank random matrix as input features and precompute polynomial features of each level. The precomputed results are then fed into the subsequent MLP to learn the final embeddings of users and items. Specifically, we utilize the low-rank feature matrix $\mathbf{X}$ and the decoupled framework introduced in Section 4.1 to reformulate Equation 5 into the following form:

$$\begin{aligned} \mathbf{E}_{low} &= MLP(\mathbf{Z}) = MLP(\sum_{\ell=0}^{L} w_\ell \mathbf{Z}^{(\ell)}), \\ \mathbf{Z}^{(\ell)} &= \mathbf{J}_\ell^{a,b}(\mathbf{P})\mathbf{X}. \end{aligned} \tag{13}$$

Similarly, we obtain $\mathbf{E}_{mid} = tanh(\beta MLP(\mathbf{X}) - \mathbf{E}_{low})$. Taking a single-layer simple MLP as an example, the dimension of the model parameter matrix is $h \times d$, which is much smaller than the scale of the JGCF model ($n \times d$), since $\mathbf{X}$ is a low-dimensional feature matrix. In addition, the polynomial basis functions can be pre-computed to speed up the process of graph convolution.

4.3 GCL for recommendation

The core of GCL for recommendation lies in obtaining a perturbed adjacency matrix $\hat{\mathbf{A}}$ through different data augmentation techniques and substituting it into the embedding formula to derive the corresponding perturbed embedding. For example, LightGCL [2] employs truncated SVD to obtain $\hat{\mathbf{A}}$. Within the Lighter-X framework, we can adopt the same precomputation approach to acquire the perturbed propagation matrix $\hat{\mathbf{Z}}$, and similarly gain the perturbed embedding matrix. This improves the computational efficiency of GCL-based recommendation models.

LighterGCL. Since LightGCN forms the foundation of LightGCL's embedding learning, its parameter scale is $n \times d$, equivalent to that of LightGCN. Therefore, we make LighterGCL have LighterGCN as its backbone to get the embedding $\mathbf{E}$, which also has a parameter scale of $h \times d$ and is much smaller than LightGCL. To further increase the efficiency and scalability of the model, we can pre-compute the perturbation component $\hat{\mathbf{Z}}$ in LighterGCL by using the low-rank input matrix $\mathbf{X}$. Specifically, we precompute the perturbed representations $\hat{\mathbf{Z}}^{(\ell)}$ of each layer utilizing the perturbed adjacency matrix $\hat{\mathbf{P}}$ and input features $\mathbf{X}$, which is a low-rank random matrix. The perturbed embeddings are then obtained by aggregating the pre-computed $\hat{\mathbf{Z}}^{(\ell)}$ and feeding it into MLP for training:

$$\begin{aligned} \hat{\mathbf{E}} &= MLP(\hat{\mathbf{Z}}) = MLP(\sum_{\ell=0}^{L} w_\ell \hat{\mathbf{Z}}^{(\ell)}), \\ \hat{\mathbf{Z}}^{(\ell)} &= \hat{\mathbf{P}} \cdot \mathbf{P}^{\ell-1}\mathbf{X}. \end{aligned} \tag{14}$$

where $\hat{\mathbf{Z}}^{(0)} = \mathbf{X}$. As a result, the repetitive perturbation generation computation in conventional approaches is circumvented by employing the low-rank feature matrix $\mathbf{X}$ and decoupling framework in LighterGCL. This substantially diminishes the time and space complexity of LighterGCL, rendering it more appropriate for scenarios involving large-scale graph learning.

Table 3: The statistics of datasets.

Dataset	#User	#Item	#Interaction	Sparsity
LastFM	1,892	17,632	92,834	99.72%
MovieLens-1M	6,040	3,952	1,000,209	95.81%
MovieLens-20M	138,493	27,278	20,000,263	99.47%
Yelp-2018	31,668	38,048	1,561,406	99.87%
Alimama	884,607	9,824	5,818,903	99.93%
HuaweiAds	1,692,592	25,158	3,504,103	99.99%

4.4 Analysis

GNN-based recommendation models typically suffer a significant computational cost as a result of the requirement to execute convolution operations on the complete graph repeatedly during the training stage. In contrast, we decouple the costly feature propagation from the training process, enabling models to pre-compute the convolutional operations. This avoids repetitive computations

Table 4: Performance comparison at public datasets, metric@10 is presented.

Dataset		LastFM			MovieLens-1M			MovieLens-20M			Yelp2018		
		recall	ndcg	*#params*	recall	ndcg	*#params*	recall	ndcg	*#params*	recall	ndcg	*#params*
Base Models	LightGCN	0.1952	0.1878	*2.50M*	0.1688	0.265	*1.25M*	0.2129	0.273	*21.15M*	0.0560	0.0450	*4.46M*
	JGCF	0.2054	0.1971	*2.50M*	0.1863	0.2823	*1.25M*	0.2185	0.2804	*21.15M*	0.0687	**0.0556**	*4.46M*
	LightGCL	0.205	0.2018	*2.50M*	0.1592	0.2539	*1.25M*	0.1172	0.1578	*21.15M*	0.0617	0.0496	*4.46M*
Lighter-X	LighterGCN	0.1946	0.1882	*0.40M*	0.1818	0.2731	*0.19M*	0.2108	0.278	*1.70M*	0.0566	0.0451	*0.07M*
	LighterJGCF	**0.2095**	0.1952	*0.40M*	**0.1883**	**0.2839**	*0.19M*	**0.2268**	**0.2882**	*1.70M*	**0.0694**	0.0538	*0.07M*
	LighterGCL	0.2059	**0.2021**	*0.40M*	0.1753	0.2642	*0.19M*	0.1688	0.2217	*1.70M*	0.0627	0.0497	*0.07M*

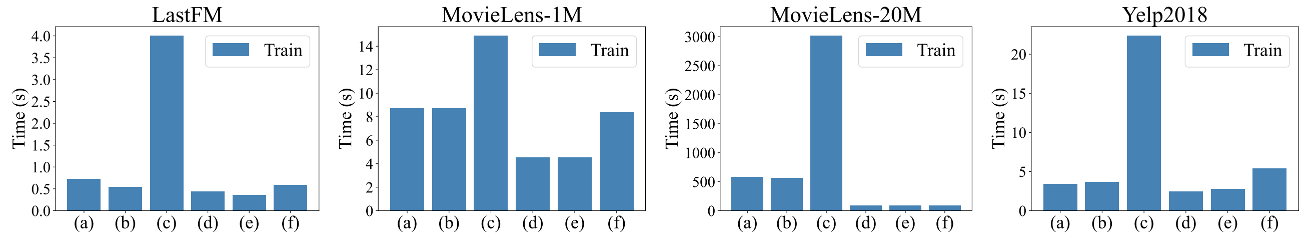

Figure 2: Time cost per epoch. The methods are denoted as follows: (a) LightGCN, (b) JGCF, (c) LightGCL, (d) LighterGCN, (e) LighterJGCF, and (f) LighterGCL.

throughout the training stage and significantly improves the efficiency. As shown in Table 2, we compare the model's preprocessing, per-batch training complexity, total training complexity, and inference complexity against baseline models.

Compared with the original model, although Lighter-X needs to perform Graph Convolution in the preprocessing stage, it only needs to be performed once. In contrast, baseline methods need to repeat the convolution of the whole graph in each training batch. Therefore, Lighter-X enhances training efficiency by pre-computing graph convolutions, thereby eliminating repetitive computations and significantly reducing the computational cost.

Additionally, the remarkable theoretical foundation and performance that are intrinsic to the base model are maintained by Lighter-X. For example, JGCF [9] improves recommendation performance without increasing the time complexity by integrating polynomial functions. LighterJGCF attentively maintains this characteristic, guaranteeing that its computational requirements do not surpass those of LighterGCN while attaining exceptional recommendation results. Similarly, LighterGCL incorporates graph augmentation to produce more robust embeddings. This is achieved without the necessity for synthetic graph generation to facilitate explicit data augmentation, inheriting from the efficiency characteristics of LightGCL [2].

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. We conduct experiments on six different datasets. (1) **LastFM** contains the listening history of users on the Last.fm online music system. (2) **MovieLens-1M** and (3) **MovieLens-20M** contain movie rating data from the MovieLens website, with each record reflecting a user's rating for a particular movie. (4) **Yelp2018** is collected from users' reviews of merchants on Yelp[1]. (5) **Alimama** contains user behaviors on taobao.com platform[2]. We construct interaction graphs using users' purchase relationships with product categories. (6) **HuaweiAds** is a dataset containing around 3.5 million users' behaviors toward the advertisements shown on Huawei devices (mobile phone, HuaweiPad, etc). It collects a 2-hour click log from one day in 2023. Table 3 summarizes the statistics of above-mentioned dataset.

Baselines. We regard three representative models LightGCN [10], JGCF [9] and LightGCL [2] as important baseline methods and conduct a comprehensive comparison of their performance and training efficiency against Lighter-X. Furthermore, we evaluate our models against other recommendation systems, including BPR [16], NeuMF[11], NGCF [20], DGCF [21], GDE [14], GTN [6], RGCF [17], and DirectAU [18], on the Yelp2018 dataset.

Implementation Details. For all baselines and our proposed methods, we implement using RecBole [26, 27], an open-source recommendation algorithm framework, and set hyperparameters based on their suggestions. All methods are optimized with Adam and initialize model parameters using the Xavier distribution. To ensure a fair comparison, we standardize the embedding size across all methods: 64 for Yelp2018 to align with other baselines, 32 and 64 for HuaweiAds to support business processing needs, and 128 for all other datasets. For Lighter-X, we direct the configuration of the input random matrix based on RIP theory, and c is turned in [1, 10].

5.2 Experiments on Public Datasets

Evaluation Protocols. In this experiment, for each user, we randomly select 80% and 10% of her interactions as the training and validation sets, while the others are left for testing. We use Recall and NDCG as the evaluation metrics and let the recommender models generate 10 items to compare with the ground truth.

Effectiveness. The overall performances of our framework against different base models are presented in Table 4. We can see, our framework can achieve comparable or even better performances than the base model across all the evaluation metrics and datasets. These results are encouraging, since our framework has much fewer parameters, which may demonstrate that, in traditional graph-based recommender models, a large number of parameters could be redundant and useless in terms of improving performance. Usually, recommender systems need to handle a large amount of real-time

[1]https://www.yelp.com/
[2]https://tianchi.aliyun.com/dataset/56

Table 5: Performance comparison on Yelp2018 dataset.

Method	recall@k		ndcg@k	
	k=10	k=20	k=10	k=20
BPR	0.0452	0.0764	0.0355	0.0460
NeuMF	0.0313	0.0548	0.0235	0.0316
NGCF	0.0459	0.0778	0.0364	0.0472
DGCF	0.0527	0.0856	0.0419	0.0528
LightGCN	0.0560	0.0913	0.0450	0.0569
GDE	0.0483	0.0808	0.0383	0.0493
GTN	0.0603	0.0984	0.0483	0.0611
RGCF	0.0633	0.1026	0.0503	0.0637
DirectAU	0.0557	0.0907	0.0435	0.0553
JGCF	0.0687	0.1105	**0.0556**	0.0694
LighterJGCF	**0.0694**	**0.1109**	0.0538	**0.0699**

Table 6: Performance comparison on Alimama dataset.

Method	recall@k		ndcg@k		#params
	k=10	k=20	k=10	k=20	
LightGCN	0.172	0.196	0.1538	0.1607	*114.49M*
JGCF	OOM	OOM	OOM	OOM	*114.49M*
LightGCL	0.1889	0.2526	0.1231	0.1413	*114.49M*
LighterGCN	0.2162	0.2855	0.1488	0.1684	*0.09M*
LighterJGCF	**0.2241**	**0.298**	**0.1538**	**0.1749**	*0.09M*
LighterGCL	0.1967	0.2557	0.1415	0.1583	*0.09M*

data, which requires high training and inference speed. The above experiments demonstrate that our lightweight framework holds great promise to achieve this goal. At last, our framework serves as an efficient plug-and-play strategy, which makes it more flexible and practical in real-world scenarios.

Efficiency. In the above experiments, we demonstrate the effectiveness of our framework. A more significant advantage of our framework is its efficiency. In this experiment, we analyze the time cost of our framework in the training and inference phases. To evaluate the cost, we compare our framework with different base models for training one epoch. As shown in Figure 2, our framework can greatly reduce the time cost as compared with the base model. For example, on the MovieLens-20M dataset, the training time of LighterGCN is about 1/6 of LightGCN's. This result verifies the potential of our framework for efficient model training, which is crucial for practical recommender systems.

Performance comparison with other models. We benchmark the proposed LighterJGCF against other leading recommendation algorithms on the Yelp2018 dataset, as shown in Table 5. Optimized from JGCF, the prevailing state-of-the-art approach, LighterJGCF attains superior performance across all metric evaluations.

5.3 Online Experiments

Evaluation Protocols. Beyond the above experiments on public datasets, we also demonstrate the superiority of our framework in two real-world product environments including Alimama and HuaweiAds datasets. In specific, the Alimama dataset is divided into training, validation, and testing sets in an 8:1:1 ratio based on timestamps. The HuaweiAds dataset extracts the last interaction item of each user to form the testing set, while the others are used for training. The other settings follow the above experiments.

Table 7: Performance comparison on HuaweiAds dataset. d **is the embedding size.**

Setting	Method	recall@k				#param
		k=1	k=3	k=5	k=10	
d=32	LightGCN	0.1218	0.1724	0.1974	0.2352	*54.97M*
	LighterGCN	0.1418	0.1963	0.2163	0.2425	*0.98M*
d=64	LightGCN	0.1248	0.1792	0.2066	0.2483	*109.94M*
	LighterGCN	0.1541	0.2134	0.2316	0.2524	*0.99M*

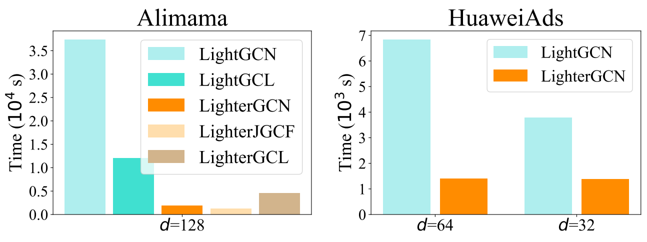

Figure 3: Total Running Time Comparison.

Effectiveness. Tables 6 and 7 show the experimental results on the Alimama and HuaweiAds datasets, respectively. We found that the model needs extra space to save the intermediate results since each ℓ-th ($\ell \geq 2$) layer of the JGCF needs to be computed based on the embedding of the first two layers, and thus suffers from the out-of-memory (OOM) problem on the Alimama dataset. However, LighterJGCF pre-computes this computation before training, eliminating the need for repeatedly allocating additional storage space during the training phase. We notice that the parameter scale reaches 114.49 million for LightGCN on the Alimama dataset and 109.04 million on the HuaweiAds dataset (d=64). However, with just 0.09 million parameters on the Alimama dataset, which is only 0.8% of the base model's parameters, Lighter-X achieves even better performance. Similarly, on the HuaweiAds dataset, LighterGCN attains superior performance while using just 1-1.7% of the parameter quantity of LightGCN, which agrees with the above experiments.

Efficiency. In Figure 3, we compare the running time of different models based on the Alimama and HuaweiAds datasets, respectively. We can see that the time cost of the base model is significantly lowered by applying our framework to it. For example, on the HuaweiAds dataset, Lighter-X reduces the total runtime by about 70% compared to the baseline of LightGCN. This further validates that Lighter-X can significantly accelerate training on industrial-scale datasets.

6 CONCLUSION

In this paper, we propose an efficient and plug-and-play strategy Lighter-X. With random sampling and decoupling frameworks, Lighter-X can significantly reduce the number of parameters and computational complexity of traditional models, which improves the training efficiency and makes it easier to be applied to large-scale real recommender systems. Based on LightGCN, we constructed the scalable LighterGCN model. We also demonstrate the generalization ability of the framework on filtering collaborative filtering and graph contrastive learning. Extensive empirical evaluation demonstrates that Lighter-X can effectively reduce the parameter size and improve the efficiency of existing recommender models, while still achieving comparable performance.

REFERENCES

[1] Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2464–2473.

[2] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=FKXVK9dyMM

[3] Emmanuel J Candès, Justin Romberg, and Terence Tao. 2006. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory* 52, 2 (2006), 489–509.

[4] Emmanuel J Candes and Terence Tao. 2005. Decoding by linear programming. *IEEE transactions on information theory* 51, 12 (2005), 4203–4215.

[5] Xinyu Du, Xingyi Zhang, Sibo Wang, and Zengfeng Huang. 2023. Efficient Tree-SVD for Subset Node Embedding over Large Dynamic Graphs. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[6] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2022. Graph trend filtering networks for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 112–121.

[7] Simon Foucart, Holger Rauhut, Simon Foucart, and Holger Rauhut. 2013. *An invitation to compressive sensing*. Springer.

[8] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.

[9] Jiayan Guo, Lun Du, Xu Chen, Xiaojun Ma, Qiang Fu, Shi Han, Dongmei Zhang, and Yan Zhang. 2023. On Manipulating Signals of User-Item Graph: A Jacobi Polynomial-based Graph Collaborative Filtering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 602–613.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[12] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[13] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1253–1262.

[14] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. Less is more: reweighting important spectral graph features for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1273–1282.

[15] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. SVD-GCN: A simplified graph convolution paradigm for recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1625–1634.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[17] Changxin Tian, Yuexiang Xie, Yaliang Li, Nan Yang, and Wayne Xin Zhao. 2022. Learning to denoise unreliable interactions for graph collaborative filtering. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 122–132.

[18] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1816–1825.

[19] Hanzhi Wang, Mingguo He, Zhewei Wei, Sibo Wang, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2021. Approximate graph propagation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1686–1696.

[20] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[21] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1001–1010.

[22] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[23] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 726–735.

[24] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[25] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1294–1303.

[26] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. 2022. RecBole 2.0: Towards a More Up-to-Date Recommendation Library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4722–4726.

[27] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4653–4664.