Deep Graph Convolutional Networks with Hybrid Normalization for Accurate and Diverse Recommendation

Wei Guo[1†], Yang Yang[2†], Yaochen Hu[3], Chuyuan Wang[1], Huifeng Guo[1], Yingxue Zhang[3], Ruiming Tang[1], Weinan Zhang[2], Xiuqiang He[1]
† Co-first authors with equal contributions
[1]Noah's Ark Lab, Huawei, [2]Shanghai Jiao Tong University, [3]Huawei Noah's Ark Lab, Montreal Research Center
{guowei67,yaochen.hu,wangchuyuan,huifeng.guo,yingxue.zhang,tangruiming,hexiuqiang1}@huawei.com,
{yyang0324,wnzhang}@sjtu.edu.cn

ABSTRACT

Collaborative filtering (CF) methods based on Graph Convolutional Networks (GCNs) have attracted great research interests and achieved the state-of-the-art performance in recent years. However, all the existing GCN-based CF models achieve their best performance with shallow layers, which fail to model higher-order collaborative signal. Besides, most GCN-based CF models utilize the same normalization rule for neighbor information aggregation, leading a popularity-related importance (symmetric normalization) or equal importance (left normalization) for neighbors. Due to the intrinsic differences over nodes, different normalization rules are suitable for them to aggregate neighbor information. In this work, we propose a novel *Deep Graph Convolutional Network with Hybrid Normalization* (DGCN-HN) to alleviate the above limitations. First, a deep graph convolutional network for recommendation is designed which consists of **residual connection** and **holistic connection** to alleviate the over-smoothing problem. It allows the effective training of GCN with deeper layers. Then, a **hybrid normalization layer** and a **simplified attention network** are proposed to flexibly model the importance of neighbors by adaptively incorporating information from different normalization rules. Comprehensive experiments on three real-world datasets show that DGCN-HN achieves the best accuracy against the state-of-the-art methods (12.12% and 12.77% relative improvement on average in terms of *Recall*@20 and *NDCG*@20). Further, we conduct a diversity study compared with the best baseline. It demonstrates that our proposed solution can achieve more diverse recommendation results. Besides, in our extensive case study, we show that our proposed method can benefit more for users having few interaction history, which can be used for alleviate the data sparsity problem.

1 INTRODUCTION

Nowadays the demand of diverse and personalized recommendation becomes more and more stronger because of the information overload in the web services, such as online video, news feeding and shopping. It is a practical problem to recommend the most suitable items for individual users from a rapidly expanding pool of candidates in real-world recommender systems. *Collaborative Filtering* (CF) [19], which learns user and item representations from their historical interaction data, is an effective way to capture user preference by calculating the similarity between the user and items. Various methods based on CF have been proposed for recommendation [11, 15, 18, 25].

Recently, with the success of graph convolutional networks (GCNs) [7, 13, 20, 23] on graph structure data, many practitioners use GCNs and its variants to improve the performance of CF-based model as the user-item interactions can be naturally viewed as a bipartite graph. Graph Convolutional Matrix Completion (GC-MC) [2] treats the recommendation task as link prediction and employs a graph auto-encoder framework on the user-item bipartite graph. Neural Graph Collaborative Filtering (NGCF) [21] propagates embeddings of users and items via utilizing multiple GCN layers to capture high-order connectivity. Note that NGCF keeps the same network design as GCNs, like feature transformation and non-linear activation, which are found redundant in CF [3, 10]. Therefore, LR-GCCF [3] removes non-linear activation, and LightGCN [10] takes a further step that removes all transformation parameters and activation function existing in the convolutional layers, which greatly improves the performance over NGCF.

Although GCN-based CF models have made great progress and achieved the state-of-the-art performance, they still suffer from two limitations. *Firstly*, most of the current GCN-based CF models achieve their best performance with shallow models, i.e., Pinsage [24] is 2-layer model, LR-GCCF and LightGCN [3, 10] are 3-layer or 4-layer models. Though deeper versions of LightGCN have access to more neighbor information, it performs worse in recommendation. Such shallow architectures limit their ability to extract useful information from higher-order neighbors, especially for sparse nodes that have very few interactions, which makes it hard to learn high-quality representation.

Secondly, most GCN-based CF models use a fixed normalization rule for all nodes in graph during neighbor information aggregation. For example, GraphSAGE [7] uses left normalization (also called as random walk based normalization) that assigns same normalized weights for different neighbors, while NGCF [21] and LightGCN [10] use symmetric normalization that assigns smaller normalized weights for popular neighbors and bigger weights for unpopular neighbors. Each normalization has its own advantages. Without loss of generalization, we take the viewpoint of users for illustration. When some popular items are not in the user's interests, left normalization performs worse than symmetric normalization as it

DLP-KDD 2021, August 15, 2021, Singapore

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

assigns the same importance to interested items and uninterested items, regardless their popularity. On the other hand, if some popular items are in the user's interests, symmetric normalization will cause the aggregation process fail to learn useful information from these popular and interested items, because the normalized weights of these items assigned by symmetric normalization are small (as they are very popular). Thus using the same normalization rule for all nodes is inflexible and sub-optimal. Besides, we explore the effect of left normalization and symmetric normalization from the perspective of accuracy and diversity.

We use a toy example to validate our argumentation as shown in Figure 1. *Firstly*, considering higher-order connection, user u_2 purchased iPhone, Computer before as he is interested in electronic product. Thus he may be interested in Drone, which needs higher-order neighbors to capture this underlying interested item. It shows that failing to model higher-order neighbors may limit the recommendation performance. *Secondly*, we can see that the iPhone was purchased by many people, like u_1, u_2 and u_3. Different users may buy iPhone with different motivations. u_1 bought it for need not for her interests, as her other purchased items are stationery (i.e., Pen and Book). On the other hand, u_3 bought iPhone driven by his great interests on technology products as he bought some other technology products (Computer and iWatch) which are not so popular in the graph. If we use left normalization for all nodes' information aggregation, then the electronic product items will be recommended to u_1 even though she has no interest in it. However, if we use symmetric normalization for all nodes' information aggregation, even though iPhone contains more related interest information for u_3, it has little importance because of its popularity during aggregation process. Thus either using left normalization to aggregation items' information via the same importance, or using symmetric normalization to distribute small weight for popular items for all nodes' aggregation process is unreasonable in some cases.

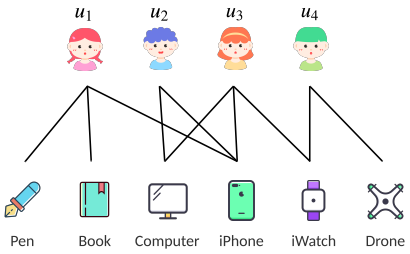

Figure 1: An example of user-item interaction graph.

To tackle the above issues, we propose a novel graph convolutional network-based recommendation framework, *Deep Graph Convolutional Network with Hybrid Normalization* (DGCN-HN).

We *first* design a deep graph convolutional network for recommendation which includes **residual connection** and **holistic connection** to avoid over-smoothing and train very deep graph convolutional network. The **residual connection** is to sum the previous layer's node representation into the next layer's representation, which can facilitate shallow layer's feature reuse and gradient back-propagation. The **holistic connection** lets node representations from all layers connect to the final representation to get the combined representations. As different nodes may need different aggregation layers, this connection can keep the previous information to avoid over-smoothing. Figure **??** shows that our proposed deep graph convolutional network for recommendation enables successfully training of GCNs up to 8 layers deep with the performance increasing steadily.

Secondly, we extend existing methods from fixed normalization to hybrid normalization by combining left normalization and symmetric normalization adaptively for flexible modeling of neighbor importance. Some nodes prefer left normalization for aggregating neighbor information with equal importance and more diverse recommendation, while some nodes prefer symmetric normalization to assign neighbor nodes with popularity-based importance and get more personalized recommendation. The detailed analysis of accuracy and diversity trade-off can be seen in Sec 2.3. By employing the hybrid normalization, our model can learn a more flexible weight for each neighbor, leading to more accurate as well as more diverse recommendation. Moreover, to learn node representations by adaptively incorporating information from different normalization rules, a simplified attention network is proposed, to automatically learn different weights for information from different normalization rules.

To summarize, the main contributions of this paper are as follows:

1) We introduce a deep graph convolutional network for recommendation to address the over-smoothing problem in the user-item bipartite graph representation learning. Based on it, we learn a 10-layer model with great performance.

2) We propose to employ an adaptive hybrid normalization for flexible modeling the importance of different neighbor nodes during aggregation process so that it can adaptively incorporate neighbor information, which improves both the accuracy and the diversity of recommendation.

3) We perform extensive experiments on three real-world large-scale datasets. The results demonstrate the superiority of our model over state-of-the-art models, in terms of both accuracy and diversity.

2 PRELIMINARIES

2.1 Problem Formulation

We begin by describing the graph-based collaborative filtering problem and introducing the necessary definitions and notations. In a typical recommendation scenario, we have a set of M users $\mathcal{U} = \{u_1, u_2, ..., u_M\}$ and a set of N items $\mathcal{V} = \{v_1, v_2, ..., v_N\}$. As user-item interactions can be regarded as a user-item bipartite graph, we can denote a user-item interaction matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$ to represent the interaction relations, where $y_{uv} = 1$ means user u has interaction with item v before, otherwise $y_{uv} = 0$. The user-item bipartite graph can be also represented as $\mathcal{G} = \langle \mathcal{U} \cup \mathcal{V}, \mathbf{A} \rangle$, where $\mathbf{A}$ is the adjacent matrix

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{Y} \\ \mathbf{Y}^\intercal & 0 \end{pmatrix}. \tag{1}$$

Our problem is to learn user and item embeddings, where the inner products between a user embedding and an item embedding

represent how likely the user will choose the item. Eventually, we recommend the top-K items to users.

2.2 Graph Convolutional Networks

Graph Convolutional Networks [13] was proposed to learn the node embedding in a graph. The basic idea is to recursively aggregate the embedding from neighbors to extract the structure proximity information. Specifically, let $\mathbf{H}^l \in \mathbb{R}^{(M+N)\times d^l}$ be the representation of nodes at layer l, where d^l is the dimension of features at layer l. The forward propagation for embedding is defined as:

$$\mathbf{H}^{l+1} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^l\mathbf{W}^l), \tag{2}$$

where $\mathbf{W}^l \in \mathbb{R}^{d^l \times d^{l+1}}$ is the transformation matrix in layer l with trainable parameters, $\tilde{\mathbf{A}}$ is a *normalization* of adjacent matrix and $\sigma(\cdot)$ is some non-linear activation function. $\mathbf{H}^0$ is defined as the initial node embeddings or features. We denote $\mathbf{D}$ as the degree matrix of $\mathbf{A}$. There are three typical normalization methods, i.e., *left normalization* or *random walk normalization* $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$, *symmetric normalization* $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and *right normalization* $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$. We could also add self-loops to adjacency matrix as $\mathbf{A}' = \mathbf{A} + \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, and we denote the corresponding degree matrix as $\mathbf{D}'$. [13] adopts a symmetric normalization over the adjacent matrix with self-loops. We can also represent Eq.(2) in a message passing framework with embedding vectors. To differentiate between users and items, let $\mathbf{h}_u^{(l)}$ and $\mathbf{h}_v^{(l)}$ represent the embedding of user u and item v in layer l, we have

$$\mathbf{h}_u^{(l+1)} = \sigma\Big(\sum_{v\in N_u} \tilde{\mathbf{A}}_{uv}\mathbf{h}_v^{(l)}\mathbf{W}^l\Big), \tag{3}$$

$$\mathbf{h}_v^{(l+1)} = \sigma\Big(\sum_{u\in N_v} \tilde{\mathbf{A}}_{vu}\mathbf{h}_u^{(l)}\mathbf{W}^l\Big), \tag{4}$$

where N_u, N_v denotes the set of neighbor nodes of node u and v. Specifically, taking central node of u and neighbor node of v as as an example, $\tilde{\mathbf{A}}_{uv}$ is the corresponding element in $\tilde{\mathbf{A}}$, $\tilde{\mathbf{A}}_{uv} = \frac{1}{|N_u|}$ for *left normalization*, $\tilde{\mathbf{A}}_{uv} = \frac{1}{\sqrt{|N_u|}\sqrt{|N_v|}}$ for *symmetric normalization* and $\tilde{\mathbf{A}}_{uv} = \frac{1}{|N_v|}$ for *right normalization*.

2.3 Normalization and Diffusion Algorithms

In this section, we aim to draw the link between the existing diffusion-based algorithm in recommendations and the commonly used normalization techniques in graph neural networks. And we discuss their relationship to the metric of relevance and diversity of the recommendation results.

[26] proposes a hybrid methods over *heat-spreading* algorithm and *probabilistic-spreading* algorithm. In *heat-spreading*, weights on users and items are recursively updated by averaging the weights on their neighbors. In *probabilistic-spreading*, weights are evenly distributed to neighbors, then every node collects the weights from neighbors and updates its own weight by the sum of those received weights. [26] shows that *heat-spreading* algorithm favors unpopular items and gets better diversity in recommended results, while *probabilistic-spreading* algorithm favors popular items and gets more accurate recommendation. Their hybrid approach over such two algorithms can achieve a balance between accuracy and diversity.

Interestingly, the normalization methods have similar propagation properties with the diffusion algorithms in [26]. Specifically, *left normalization* is similar to *heat-spreading* algorithm in the sense that embedding from neighbors are averaged in Eq.(3) and Eq.(4) with $\tilde{\mathbf{A}}_{uv} = \frac{1}{|N_u|}$. In *right normalization* with $\tilde{\mathbf{A}}_{uv} = \frac{1}{|N_v|}$, embedding of neighbors are divided by their own degree, and eventually propagated to the central nodes, which is similar to *probabilistic-spreading* algorithm. Besides, *symmetric normalization* with $\tilde{\mathbf{A}}_{uv} = \frac{1}{\sqrt{|N_u|}\sqrt{|N_v|}}$ acts as a balance between *left normalization* and *right normalization*. Although the intrinsic structure in graph neural network models and the original diffusion algorithms are not exactly equivalent, they at least share similar trend in terms of the favor on unpopular items and popular items. Inspired by the hybrid method for diffusion algorithms, we propose a hybrid normalization method, in order to achieve both accurate and diverse recommendation results.

3 METHOD

In this section, we introduce Deep Graph Convolutional Networks with Hybrid Normalization (DGCN-HN), which is shown in Figure 2. It consists of three main components. *Firstly*, a deep graph convolutional network for recommendation with residual connection and holistic connection is designed to deepen the model. *Secondly*, a hybrid normalization layer is introduced for flexible modeling of neighbor importance to improve recommendation accuracy and diversity by combining left normalization and symmetric normalization simultaneously. *Thirdly*, a simplified attention network is proposed to adaptively incorporate information from the representation of different normalization methods.

3.1 Embedding Layer

In web-scale recommender system, user IDs and item IDs are usually encoded as one-hot vectors. We first transform the IDs of users and items into the embedding vectors:

$$\mathbf{e}_i = \mathbf{E} \cdot \mathbf{x}_i \tag{5}$$

where $\mathbf{E} \in \mathbb{R}^{(M+N)\times d_0}$ is the initial embedding table for users and items, $\mathbf{x}_i$ is the ID (one-hot vector) of an user or an item and $\mathbf{e}_i \in \mathbb{R}^{d_0}$ is the gathered embedding, and d_0 is the dimension of initial embedding.

3.2 Deep Graph Convolutional Network for Recommendation

Most of existing GCN-based collaborative filtering methods [3, 21] update the node representations in (l+1)-th layer by normalizing and aggregating the representations of their neighbors in l-th layer. The representations of user u and item v in (l+1)-th layer can be formulated as follows:

$$\mathbf{h}_u^{(l+1)} = \mathrm{agg}(\tilde{\mathbf{A}}_{uv}\mathbf{h}_v^{(l)}, \forall v \in N_u) \tag{6}$$

$$\mathbf{h}_v^{(l+1)} = \mathrm{agg}(\tilde{\mathbf{A}}_{vu}\mathbf{h}_u^{(l)}, \forall u \in N_v) \tag{7}$$

where $\mathrm{agg}(\cdot)$ is an aggregation mapping function which maps the normalized neighbor representations into an updated center node representation.

Even though GCN makes great progress for node representation learning in CF task, we find that most GCN-based models

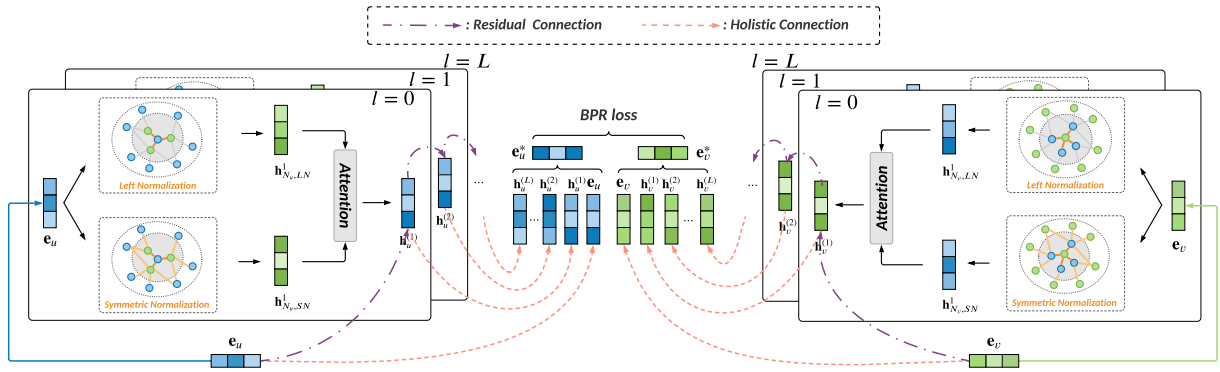

Figure 2: Overall structure of DGCN-HN framework. Arrowed lines present the flow of information. Blue nodes represent users and green nodes represent items. It takes the initial embedding of users and items as input, then predicts the preference of users to items. Best viewed in color.

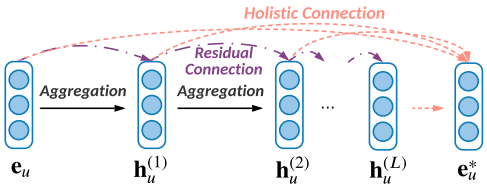

Figure 3: A sketch map of deep graph convolutional network for recommendation which includes residual connection and holistic connection.

of recommendation achieve the best performance with a shallow network structure. For example, Pinsage [24] is a 2-layer GCN, LR-GCCF and LightGCN [3, 10] are the 3-layer or 4-layer GCNs. The deeper layer will lead to the performance degradation because of the over-smoothing issue, which states that representations from different nodes become indistinguishable due to repeated propagation. Therefore, these models only achieve sub-optimal performance since they fail to capture the higher-order information, which is useful especially for sparse nodes.

To further explore higher-order information and alleviate the over-smoothing issue, we follow a similar architecture as ResGCN [14]. ResGCN [14] extends GCN into deeper layer with residual connection and output combination [8]. It shows great success in the point cloud classification and segmentation task. However, the aggregation layer is over parameterized in ResGCN if we perform it directly in the collaborative filtering task. Prior works have demonstrated empirically that the feature transformations and non-linear activation functions are not necessary and might even degrade the recommendation performance [3, 10]. Therefore, we remove the transformation matrix and activation function in each aggregation layer. Our proposed deep graph convolutional network for recommendation can be depicted as in Figure 3. The linear aggregation process of each layer can be re-formulated as a **residual connection**:

$$\mathbf{h}_u^{(l+1)} = \sum_{v \in N_u} \tilde{\mathbf{A}}_{uv} \mathbf{h}_v^{(l)} + \mathbf{h}_u^{(l)}, \quad \mathbf{h}_u^{(0)} = \mathbf{e}_u \tag{8}$$

$$\mathbf{h}_v^{(l+1)} = \sum_{u \in N_v} \tilde{\mathbf{A}}_{vu} \mathbf{h}_u^{(l)} + \mathbf{h}_v^{(l)}, \quad \mathbf{h}_v^{(0)} = \mathbf{e}_v. \tag{9}$$

After L layers' aggregation, instead of using the final embedding of each node for prediction, **holistic connection** combines the embedding of all the layers (including the initial embedding) as each node's final feature representation, which is formulated as:

$$\mathbf{e}_u^* = \frac{1}{L+1} \sum_{i=0}^{L} \mathbf{h}_u^{(i)} \tag{10}$$

$$\mathbf{e}_v^* = \frac{1}{L+1} \sum_{j=0}^{L} \mathbf{h}_v^{(j)} \tag{11}$$

Here we use an element-wise average aggregator as the information fusion strategy of the final layer. Other alternatives such as concatenation, attention aggregators can be further explored. Representations of different layers contain information of different receptive fields. Combining representations of all the layers is more informative and is effective for alleviating the over-smoothing problem.

It is noted that NGCF [21], LR-GCCF [3] and LightGCN [10] use a similar representation combination strategy to incorporate all the layers as the final representation. However, they still fail to scale up to the depth of a 5-layer architecture. This suggests that utilizing the holistic connection alone in the context of collaborative filtering task is not sufficient to extend GCN to a deep model.

3.3 Hybrid Normalization for Flexible Modeling of Neighbor Importance

As depicted in Section 1, most of the existing GCN-based CF methods use one normalization rule for weighting different neighbors. While for different nodes, the importance of neighbors is distinguished by the preference of central nodes, which results in a sub-optimal result. As a result, we propose to use a hybrid normalization strategy to assign weights to neighbors flexibly. It contains two types of neighbor information.

3.3.1 Symmetric Normalization Representation Propagation. One typical benefit of symmetric normalization is that it can distinguish the importance of each neighbor by its popularity. Taking the user side as an example. Following with the linear aggregation process in Eq.(8), the representation propagation from items to users when

using symmetric normalization is defined as:

$$\mathbf{h}_u^{(l+1)} = \sum_{v \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_v|}} \mathbf{h}_v^{(l)} + \mathbf{h}_u^{(l)} \tag{12}$$

3.3.2 Left Normalization Representation Propagation. In the recommendation scenario, sometimes the importance of the neighbor nodes should not be linked to their neighbor nodes' degree. For example, a user may have interest in the popular item, however using symmetric normalization will assign this item a small normalized weight. As left normalization is only based on the popularity (degree) of the central node in the normalization process, the neighbor nodes are assigned the same importance weights. Utilizing left normalization avoids representations of neighbor nodes being normalized to a very small value such that useful information is lost. Taking the user side as an example, the embedding propagation from item to user with left normalization is defined as:

$$\mathbf{h}_u^{(l+1)} = \sum_{v \in N_u} \frac{1}{|N_u|} \mathbf{h}_v^{(l)} + \mathbf{h}_u^{(l)}. \tag{13}$$

3.3.3 Simple Average for Representation Propagation. To combine the advantages of these two normalizations, we apply the average operation over these two normalization. The embedding propagation from item to user with simple average is defined as:

$$\mathbf{h}_u^{(l+1)} = \mathbf{h}_u^{(l)} + \frac{1}{2} \sum_{v \in N_u} \left(\frac{1}{|N_u|} + \frac{1}{\sqrt{|N_u|}\sqrt{|N_v|}}\right) \mathbf{h}_v^{(l)}. \tag{14}$$

Thanks to combining both symmetric and left normalization, this formulation distinguishes the importance of each neighbor node, in the mean time, it limits the range of importance to avoid useful information loss. If the popular item is the user interested in, the user representation can still capture the information of this item.

3.4 Simplified Attention Network for Adaptive Combination

As demonstrated above, neighbor representations by different normalization methods are of different importance in learning the new representation of the central node. For better combination of these representations, we introduce a novel simplified attention network for adaptively combining these neighbor representations with central node's representation. Let $\mathbf{h}_{N_u,LN}^{(l)}$ and $\mathbf{h}_{N_u,SN}^{(l)}$ denote the left normalization and symmetric normalization based neighbor representation of user u at l-th layer. Taking the user side as an example, the adaptive representation propagation process is formulated as:

$$\mathbf{h}_{N_u,LN}^{(l+1)} = \sum_{v \in N_u} \frac{1}{|N_u|} \mathbf{h}_v^{(l)} \tag{15}$$

$$\mathbf{h}_{N_u,SN}^{(l+1)} = \sum_{v \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_v|}} \mathbf{h}_v^{(l)} \tag{16}$$

$$\mathbf{h}_u^{(l+1)} = \mathbf{h}_u^{(l)} + \alpha_{u,LN}^{(l+1)} \mathbf{h}_{N_u,LN}^{(l+1)} + \alpha_{u,SN}^{(l+1)} \mathbf{h}_{N_u,SN}^{(l+1)} \tag{17}$$

where $\alpha_{u,*}^{l}$ indicates the normalized attention scores of different neighbor representations at l-th layer.

When calculating attention scores, we not only consider the neighbor self-information, but also consider the similarity between the central representation and the neighbor representation. Taking into consideration such similarity information makes the importance of each neighbor node be related to the central node, i.e., a neighbor node, which is more similar to the central node, is more important and takes more weights when aggregation. Formally, the attention representation network is defined as:

$$z_{u,*}^{(l+1)} = \mathbf{W}_1^{(l)} \sigma\Big(\mathbf{W}_2^{(l)} (\mathbf{h}_{N_u,*}^{(l+1)} + \mathbf{h}_{N_u,*}^{(l+1)} \odot \mathbf{h}_u^{(l)})\Big), \tag{18}$$

where the output $z_{u,*}^{(l+1)}$ is the attention score before normalization, $\mathbf{W}_1^{(l)} \in \mathbb{R}^{1 \times d^t}$,$\mathbf{W}_2^{(l)} \in \mathbb{R}^{d^t \times d^l}$ are feature transformation matrices, σ is the activation function, d^t is the hidden layer size of the attention network, $\odot$ denotes the Hadamard product.

Empirically, we find that Eq.(18) is difficult for optimization, so we propose a simplified attention network that removes the feature transformation matrix and activation function, and uses a simple average aggregation operation to compress the vector to a scalar:

$$z_{u,*}^{(l+1)} = \text{ave}(\mathbf{h}_{N_u,*}^{(l+1)} + \mathbf{h}_{N_u,*}^{(l+1)} \odot \mathbf{h}_u^{(l)}), \tag{19}$$

where the function $\text{ave}(\cdot)$ takes the average of every element in the input vector. Then the attention scores are normalized by a Softmax function:

$$\alpha_{u,*}^{(l)} = \frac{\exp(z_{u,*}^{(l)})}{\sum_{k \in \{LN,SN\}} \exp(z_{u,k}^{(l)})}. \tag{20}$$

3.5 Model Training

We train our model by minimizing the pairwise *Bayesian Personalized Ranking* (BPR) [17] loss:

$$\mathcal{L} = \sum_{(u,i,j) \in O} -\log\sigma(\mathbf{e}_u^* \cdot \mathbf{e}_i^* - \mathbf{e}_u^* \cdot \mathbf{e}_j^*) + \lambda ||\Theta||_2^2 \tag{21}$$

where $O = \{(u,i,j)|(u,i) \in O^+, (u,j) \in O^-\}$ denotes the training dataset. O^+ indicates observed positive user-item interactions and O^- indicates unobserved user-item interactions. Note that our model introduces no extra parameters except for the node embeddings before aggregation process, i.e., $\Theta = \{\mathbf{e}_u, \mathbf{e}_v | u \in \mathcal{U}, v \in \mathcal{V}\}$. σ is the Sigmoid function and λ is the L_2 regularization coefficient.

3.6 Model Discussion

To distinguish ourselves from prior works, we compare our model with five recent works from five aspects, which is shown in table 1. *First*, since the transformation matrix and non-linear activation function have been demonstrated empirically to be redundant in recommendation task [3, 10], we remove those over-parameterized and non-linear components as well. *Second*, LightGCN, NGCF, and LR-GCCF incorporate the holistic connection to alleviate the over-smoothing problem, however, with the increase of the numbers of layers, the information of previous layers will be further weakened and the impact of the low-order features becomes less dominant. The residual connection allows the architecture to re-emphasize feature from low-order receptive field. This design allows both the low-order and high-order collaborative filtering signals to be well utilized. *Third*, our model is the first work in the GCN based CF models that considers a hybrid normalization strategy (symmetric normalization and left normalization) simultaneously, which

Table 1: Comparison of Different Graph Variants.

	GC-MC	Pinsage	NGCF	LR-GCCF	LightGCN	DGCN-HN
Remove Transformation Matrix	×	×	×	×	✓	✓
Remove Non-linear Activation Function	×	×	×	✓	✓	✓
Residual Connection	×	×	×	×	×	✓
Holistic Connection	×	×	✓	✓	✓	✓
Normalization	Symmetric/Left	Left	Symmetric	Symmetric	Symmetric	**Symmetric**&**Left**

Table 2: Dataset statistics.

Dataset	User	Item	Instance	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062

can well balance the recommendation accuracy and diversity as explained in section 3.3.

4 EXPERIMENTS

4.1 Experiment Setup

4.1.1 Datasets. Following [10, 21, 22], we evaluate the effectiveness of our proposed model on three datasets: *Gowalla*, *Yelp2018*, and *Amazon-Book*. These benchmark datasets are open sourced real-world data with various domains, sizes, and sparsity levels.

- **Gowalla**[1]: This location recommendation dataset is a popular location dataset that has check-ins information including times and locations made by users. The locations are treated as items in our case to predict user preferences based on the check-in history.
- **Yelp2018**[2]: This business recommendation dataset is obtained from the Yelp Data Challenge in 2018. It has more than 1 million records of the business like restaurants and bars.
- **Amazon-Book**[3]: Amazon review is a popular dataset which can be used for product recommendation [9]. It contains several subsets, such as Amazon-Electronics, Amazon-Movies and Amazon-CDs. We select Amazon-Book subset for book recommendation.

For a fair comparison, we use the same pre-processed datasets which are publicly available from [10]. Table 2 shows the statistics of the three datasets.

4.1.2 Baseline Models. We compare our approach with the following state-of-the-art methods, including classical CF method (BPRMF), GCN-based CF methods (GCMC, Pinsage and NGCF), Deep GCN-based method (ResGCN), Disentangled GCN-based methods (MacridVAE, DGCF) and Light-wise GCN-based methods (LightGCN).

4.1.3 Evaluation Metrics. Following [10], we adopt two widely-used metrics, namely *Recall@k* and *NDCG@k*, to evaluate the accuracy of proposed and baseline models for top-K recommendation.

- *Recall@k* indicates the coverage of positive items as a result of top-*k* recommendation. It does not consider the actual position of the items as long as it is inside the top-*k* recommendation list.

[1]https://snap.stanford.edu/data/loc-gowalla.html
[2]https://www.yelp.com/dataset/challenge
[3]http://jmcauley.ucsd.edu/data/amazon/

Table 3: The overall accuracy comparison. Underline indicates the second best model performance. Boldface denotes the best model performance. We conduct Wilcoxon signed rank test. Asterisks indicates a statistically significant difference between the scores of the best and second-best algorithms.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Model	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
BPRMF	0.1291	0.1109	0.0433	0.0354	0.0250	0.0196
GC-MC	0.1395	0.1204	0.0462	0.0379	0.0288	0.0224
PinSage	0.1380	0.1196	0.0471	0.0393	0.0282	0.0219
NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0266
ResGCN	0.1637	0.1343	0.0592	0.0482	0.0370	0.0282
MacridVAE	0.1618	0.1202	0.0612	0.0495	0.0383	0.0295
DGCF	0.1794	0.1521	0.0640	0.0522	0.0399	0.0308
LightGCN	<u>0.1830</u>	<u>0.1554</u>	<u>0.0649</u>	<u>0.0530</u>	<u>0.0411</u>	<u>0.0315</u>
DGCN-HN	**0.1905***	**0.1606***	**0.0710***	**0.0584***	**0.0505***	**0.0393***
%Improv.	4.10%	3.35%	9.40%	10.19%	22.87%	24.76%

- *NDCG@k*: The Normalized Discounted Cumulative Gain (NDCG) is a position-aware evaluation metric that it assigns larger weights on higher-ranked true positives.

Besides accuracy metrics, we also utilize two commonly-used metrics: *Coverage@k* and *Entropy@k* for diversity evaluation [4, 16].

- *Coverage@k* indicates the proportion of recommended items to the universal set. The upper bound of *Coverage@k* is 1, and the larger the better as more items are recommended to users [4].
- *Entropy@k* focuses on the distribution of different items to be recommended. It calculates the entropy value based on recommended times of different rounds. A larger entropy value indicates a better diversity as it indicates the heterogeneity of recommended items is high [16].

4.1.4 Parameter Settings. We implement our models using Tensorflow [1]. For a fair comparison, we copy the best performance of BPRMF, GC-MC, Pinsage, NGCF, MacridVAE and DGCF-1 from [22], LightGCN from [10] directly. To be consistent with [22] and [10], we set the embedding size as 64, initialize the embedding parameters with the Xavier initialization method [6] and optimize our model with Adam optimizer [12]. Moreover, we perform a grid search to search the best hyper-parameter settings. The learning rate is searched in the range of $\{10^{-3}, 5\times10^{-4}, 10^{-4}, 5\times10^{-5}, 10^{-5}\}$, and the coefficient of $L2$ regularization is tuned in the range of $\{10^{-1}, 5\times10^{-2}, 10^{-2}, 5\times10^{-3}, 10^{-3}, 5\times10^{-3}, 10^{-4}\}$. No extra hyper-parameters are needed to be tuned besides learning rate and $L2$ regularization coefficient, which makes our results easy to reproduce.

4.2 Accuracy Comparison

Table 3 reports the overall performance comparison of 10-layer DGCN-HN with baselines. We show the best accuracy we can obtain for each method. For *Recall@k* and *NDCG@k*, we set $k = 20$ by default. We conducted Wilcoxon signed rank tests [5] to evaluate the significance of the difference between the best-performing algorithm and the second-best algorithm.

We make the following observations:

- DGCN-HN consistently yields the best performance for all datasets. More precisely, DGCN-HN outperforms the strongest baselines by **4.10%**, **9.40%** and **22.87%** in terms of *Recall@20* (*3.35%*, *10.19%* and *24.76%* in terms of *NDCG@20*) on Gowalla, Yelp2018 and Amazon-Book, respectively. The significant improvement is attributed to the following two aspects: 1) by modeling higher-order graph structure information with a deep graph convolutional network, DGCN-HN is able to exploiting more user-item collaborative signals, which leads to better feature representations of users and items; 2) by using a hybrid normalization module, the flexible modeling of neighbor importance leads to better aggregation of neighbor information.
- BPRMF obtains the worst performance on all three datasets because it projects the single ID of a node to its embedding which might ignore underlying relation information. By leveraging the user-item bipartite graph to explicitly model relation information, GC-MC, PinSage and NGCF achieve significantly better performance compared with BPRMF. In particular, since NGCF captures higher-order user-item information, it performs better than GCMC and PinSage, which reflects the importance of modeling higher-order information of neighbors.
- ResGCN outperforms NGCF because the residual connection offers better representation capacity. However, it performs worse than LightGCN. Possible reason is that the complicated feature transformation and non-linear activation restricts its representation capacity for the CF task.
- MacridVAE and DGCF further achieves better performance compared with GC-MC, PinSage and NGCF on all datasets. The reason is that these methods exploit diverse relationships among users and items in the interaction history, and disentangle user intents into several latent factors, which leads a more fine-grained feature representation.
- LightGCN outperforms the other baselines on all datasets and serves as the best baseline. It removes the feature transformation and non-linear activation function, learns users and items embeddings with a linear embedding propagation function.

4.3 Accuracy-Diversity Trade-off

Most of the existing graph-based CF models consider accuracy as the only evaluation metric without paying any attention to diversity of the recommendation results, which is not reasonable as the users may loss interest to the recommender system when popular items are recommended to her frequently. A good trade-off between accuracy and diversity, which makes the system recommends more diverse items while maintaining comparable accuracy, is very important. To evaluate the quality of trade-off between accuracy and diversity, we compare the accuracy and diversity of 10-layer DGCN-HN with the best baseline LightGCN on Gowalla

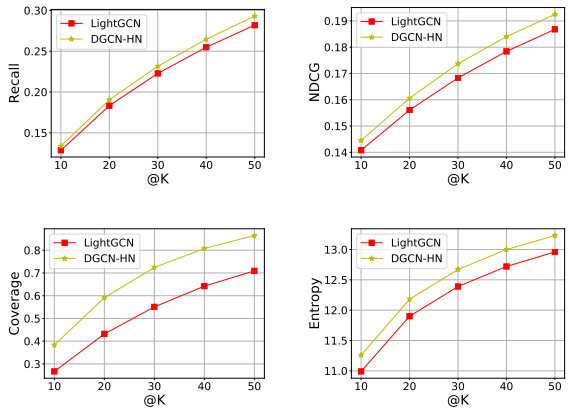

Figure 4: Accuracy-diversity comparison of our proposed DGCN-HN to LightGCN on Gowalla dataset.

with different values of k for *Recall@k*, *NDCG@k*, *Coverage@k* and *Entropy@k* in this section. Results are presented in Figure 4. We have the following observations:

- DGCN-HN achieves higher accuracy and diversity than LightGCN consistently with the growth of k for all metrics. In particular, the relative improvement with respect to the *Coverage@20* metric is **36.88%**, which shows the great ability of our proposed method in recommending diverse items to users.
- According to the accuracy-diversity dilemma [26], increasing one often results in sacrificing the other. However, DGCN-HN not only achieves better accuracy than LightGCN, but also gets higher diversity than LightGCN. This again demonstrates the superior performance of our proposed DGCN-HN.
- By combining left normalization and symmetric normalization which focus on diversity and accuracy separately, our proposed DGCN-HN increases accuracy and diversity simultaneously, but not a compromise of the two metrics. It provides new ideas to resolve the accuracy-diversity dilemma.

4.4 Ablation and Effectiveness Analysis

Table 4: Performance comparison of DGCN-HN with different choices of normalization schemes in graph convolution. Underline indicates the second best model performance. Boldface denotes the best model performance.

Dataset	Gowalla		Yelp2018	
Normalization	Recall@20	Coverage@20	Recall@20	Coverage@20
L	0.1670	**0.7230**	0.0662	**0.6311**
S	<u>0.1855</u>	0.2971	<u>0.0680</u>	0.2254
R	0.1578	0.1037	0.0599	0.1182
L&S	**0.1892**	<u>0.4968</u>	**0.0695**	<u>0.3745</u>
L&R	0.1647	0.4416	0.0526	0.2694
S&R	0.1702	0.2169	0.0523	0.1743
L&S&R	0.1744	0.3500	0.0541	0.2407

[1] *L*, *S* and *R* means left,symmetric and right normalization respectively.
[2] & denotes combination operation.

4.4.1 Impact of Normalization. In this paper, we propose to combine left normalization with symmetric normalization for better recommendation. To study its rationality, we study different choices here. Table 4 shows the *Recall@20* and *Coverage@20* of 8-layer DGCN-HN with different types of normalization. Due to the space limitation, the results in terms of *NDCG@20* and *Entropy@20* are omitted. We have the following observations:

- When a single normalization is used, left normalization achieves best diversity, while symmetric normalization obtains the best accuracy. Right normalization performs worst in both accuracy and diversity.
- Compared with other hybrid normalization, the combination of left normalization and symmetric normalization obtains not only the best accuracy, but also the best diversity.
- Based on comprehensive comparison of all the choices of normalization schemes, we can observe that left normalization achieves best diversity, but with unsatisfied accuracy. Our proposed combination of left normalization with symmetric normalization achieves the best accuracy and the second best diversity, which obtains the good trade-off between accuracy and diversity.

4.4.2 Impact of proposed component. To verify the effectiveness of individual components in the proposed model, we conduct an ablation analysis with 8-layer DGCN-HN on Gowalla and Yelp2018 datasets, and present the results in Table 5. Note that we only report the result of *Recall@20* and *Coverage@20* due to space limitation. We summarize the observations as follows:

- Four main components, including simplified attention network, hybrid normalization, residual connection and holistic connection, of our proposed model are demonstrated to be effective.
- As shown in line 1 and line 2, the introduced simplified attention network not only improves the accuracy, but also increases the diversity of recommendation.
- As shown in line 2 and line 3, the proposed hybrid normalization leads to significant performance improvement on both datasets compared with fixed symmetric normalization, especially on the diversity. This proves that it can be functioned as a module for GCN-based algorithms to learn more accurate and diverse neighborhood information.
- As shown in line 3, line 4 and line 5, the model performance degrades rapidly when removing residual connection and holistic connection with 8-layer graph convolutional network, indicating that residual connection and holistic connection are both important for training very deep graph convolutional network.

4.4.3 Impact of Network Depth. To investigate the influence of network depth to the performance of different models, we vary the number of layers L in the range of $\{1, ..., 8\}$. We show the results of *Recall*@20 on Gowalla dataset in Figure 5. LightGCN-final indicates that only the representation of last layer are used for prediction. We have the following findings:

- Increasing the depth of DGCN-HN improves the performance consistently. However, when increasing the depth, the performance of LightGCN, LightGCN-final and ResGCN degrade rapidly. Therefore, except DGCN-HN, the other three models achieve the best performance with a very shallow architecture. This observation verifies the effectiveness of our proposed method in modeling higher-order collaborative signals.
- LightGCN-final achieves its best performance with a 2-layer model, while LightGCN achieves its best performance with a 3-layer model. This indicates the effectiveness of holistic connection for prediction.
- ResGCN achieves its best performance with a 2-layer model. Possible reason is that the feature transformation and non-linear activation is hard to train in CF task, which restricts its capacity in modeling GCN into deep layer.

Table 5: Ablation studies of 8-layer DGCN-HN on each proposed component. Underline indicates the second best model performance. Boldface denotes the best model performance.

Dataset	Gowalla		Yelp2018	
Architecture	Recall	Coverage	Recall	Coverage
DHCN-HN	**0.1892**	**0.4968**	**0.0695**	**0.3745**
DHCN-HN w/o *SA*	<u>0.1882</u>	<u>0.4814</u>	<u>0.0687</u>	<u>0.3673</u>
DHCN-HN w/o *SA*,*HN*	0.1851	0.2863	0.0676	0.2204
DHCN-HN w/o *SA*,*HN*,*RC*	0.1693	0.1720	0.0614	0.1964
DHCN-HN w/o *SA*,*HN*,*RC*,*HC*	0.1175	0.0406	0.0387	0.0184

[1] *SA* and *HN* mean simplified attention network and hybrid normalization.
[2] *RC* and *HC* mean residual connection and holistic connection respectively.
[2] w/o denotes remove operation.

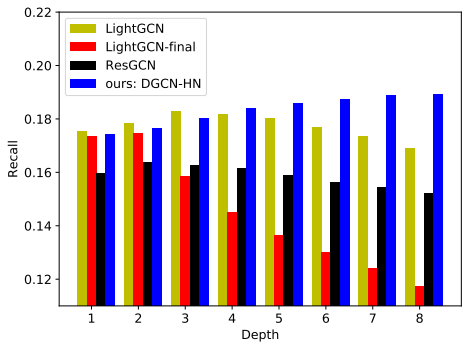

Figure 5: Performance comparison of different models with the impact of the network depth.

4.5 User Analysis

We make a fine-level analysis to investigate which group of users benefit from the higher-order collaborative signal more largely. We compare the performance of 10-layer DGCN-HN with LightGCN and LightGCN-final. We show the best accuracy we can obtain for each method. It's noticed that LightGCN-final achieves its best accuracy with a 2-layer model, while LightGCN achieves its best accuracy with a 3-layer model. Figure 6 shows the relative improvement (Y-axis in Figure 6) on Gowalla dataset with respect to different user groups. The users are partitioned into three groups according their interaction sparsity, i.e., the degree. We have the following observations:

- DGCN-HN outperforms LightGCN-final and LightGCN on all the user groups with respect to *Recall*@20 and *Coverage*@20. It

demonstrates that exploiting higher-order collaborative signal with deep graph convolutional network can lead to much more informative representation.

- The improvement for sparse users that have very few interactions are more significant than other users. It confirms that higher-order collaborative signals can help to alleviate the sparsity issue, as more items that the user are probably interested in are involved in training.
- The improvement over LightGCN-final with 2 layer are larger than LightGCN with 3 layer. It indicates that introducing more neighbor information can bring in more improvements.

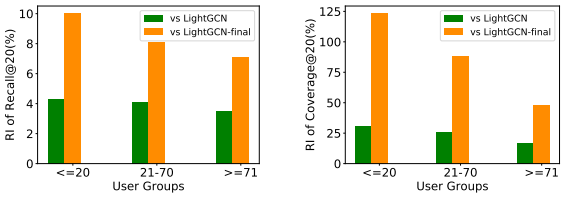

Figure 6: Relative improvements of our proposed DGCN-HN over different baselines on Gowalla dataset with respect to different user groups (partitioned by user degree).

5 CONCLUSIONS

In this paper, we propose a novel graph convolutional network based recommendation framework DGCN-HN with three main components: deep graph convolutional network for recommendation, hybrid normalization layer and simplified attention network. Extensive experiments on three real-world datasets demonstrate consistent improvement on recommendation accuracy over state-of-the-art methods. Further analysis on diversity highlight that our proposed method can bring more diverse recommendation. Besides, we show that our method can benefit more for users having fewer interaction records in our extensive case study, which confirms that higher-order collaborative signals can help to alleviate the sparsity issue.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 265–283.

[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[3] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 27–34.

[4] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. 2017. Learning to recommend accurate and diverse items. In *Proceedings of the 26th international conference on World Wide Web*. 183–192.

[5] Ben Derrick and Paul White. 2017. Comparing two samples from an individual Likert question. *International Journal of Mathematics and Statistics* 18, 3 (2017).

[6] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010 (JMLR Proceedings)*, Yee Whye Teh and D. Mike Titterington (Eds.), Vol. 9. JMLR.org, 249–256.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[9] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2002.02126* (2020).

[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] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[14] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *Proceedings of the IEEE International Conference on Computer Vision*. 9267–9276.

[15] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 689–698.

[16] Morteza Rashidi, Ab Razak Che Hussin, and Mehrbakhsh Nilashi. 2015. Entropy-based Ranking Approach for Enhancing Diversity in Tag-based Community Recommendation. *Journal of Soft Computing and Decision Support Systems* 3, 1 (2015), 1–7.

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

[18] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko (Eds.). ACM, 285–295.

[19] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009).

[20] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[21] 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.

[22] 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.

[23] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[24] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[25] Zhi-Dan Zhao and Ming-Sheng Shang. 2010. User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 Third International Conference on Knowledge Discovery and Data Mining*. IEEE, 478–481.

[26] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences* 107, 10 (2010), 4511–4515.