Neighbor Interaction Aware Graph Convolution Networks for Recommendation

Jianing Sun[1], Yingxue Zhang[1], Wei Guo[2], Huifeng Guo[2], Ruiming Tang[2], Xiuqiang He[2], Chen Ma[3], Mark Coates[4]
1. Huawei Noah's Ark Lab, Montreal Research Center, Montreal, QC, Canada
{jianing.sun, yingxue.zhang}@huawei.com
2. Huawei Noah's Ark Lab, Shenzhen, China
{guowei67, huifeng.guo, tangruiming, hexiuqiang1}@huawei.com
3. School of Computer Science, McGill University, Montreal, QC, Canada
{chen.ma2}@mail.mcgill.ca
4. Department of Electrical and Computer Eningeering, McGill University, Montreal, QC, Canada
{mark.coates}@mcgill.ca

ABSTRACT

Personalized recommendation plays an important role in many online services. Substantial research has been dedicated to learning embeddings of users and items to predict a user's preference for an item based on the similarity of the representations. In many settings, there is abundant relationship information, including user-item interaction history, user-user and item-item similarities. In an attempt to exploit these relationships to learn better embeddings, researchers have turned to the emerging field of Graph Convolutional Neural Networks (GCNs), and applied GCNs for recommendation. Although these prior works have demonstrated promising performance, directly apply GCNs to process the user-item bipartite graph is suboptimal because the GCNs do not consider the intrinsic differences between user nodes and item nodes. Additionally, existing large-scale graph neural networks use aggregation functions such as sum/mean/max pooling operations to generate a node embedding that considers the nodes' neighborhood (i.e., the adjacent nodes in the graph), and these simple aggregation strategies fail to preserve the relational information in the neighborhood. To resolve the above limitations, in this paper, we propose a novel framework **NIA-GCN**, which can explicitly model the relational information between neighbor nodes and exploit the heterogeneous nature of the user-item bipartite graph. We conduct empirical studies on four public benchmarks, demonstrating a significant improvement over state-of-the-art approaches. Furthermore, we generalize our framework to a commercial App store recommendation scenario. We observe significant improvement on a large-scale commercial dataset, demonstrating the practical potential for our proposed solution as a key component of a large scale commercial recommender system. Furthermore, online experiments are conducted to demonstrate that NIA-GCN outperforms the baseline by 10.19% and 9.95% in average in terms of CTR and CVR during ten-day AB test in a mainstream App store.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401123

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Graph Convolution Networks, Recommendation, Collaborative Filtering

ACM Reference Format:
Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, Mark Coates. 2020. Neighbor Interaction Aware Graph Convolution Networks for Recommendation In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20), July 25-30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401123

1 INTRODUCTION

With the rapid development of the Internet and mobile devices, our daily activities connect closely to online services, such as online shopping, online news and videos. The amount of online information has been growing explosively, leading to information overload. Recommender systems are a powerful information filter for guiding users to find the items of interest in a gigantic and rapidly expanding pool of candidates. Rapid and accurate prediction of users' preferences is the ultimate goal of today's recommender systems [31]. The core idea behind recommender systems is collaborative filtering (CF) [20]. The basic assumptions underpinning CF are that similar users tend to like the same item and items with similar audiences tend to receive similar ratings from an individual.

Recommender systems heavily rely on historical interactions between users and items to make accurate recommendations. However, two major challenges faced by existing recommender systems are the problems of *data sparsity* and *cold start*. Data sparsity means that most users have interacted with only a few items, and symmetrically most items have been interacted with by only a few users. The cold start scenario refers to the task of making recommendations to a user that has not interacted with any items (i.e., a new

user), or finding users for an item that has not been interacted with by any users (i.e., a new item).

Recently, graphs have been used to represent the relational information present in recommendation datasets. Incorporating and exploiting the graph representation has shown to be effective for alleviating the data sparsity and cold start problems [22, 37, 40]. Such works exploit user-item interaction graphs [37, 40] and item-item co-occurrence graphs [22]. The intuition motivating the involvement of graphs in recommender systems is that we include more relational information and increase the connectivity among users and items, leading to improved recommendation quality.

Graph Convolutional (Neural) Networks (GCNs) [1, 12, 17, 25, 35, 38, 40, 41] have proven to be among the best performing architectures for a variety of graph learning tasks. GCNs strive to learn how to iteratively aggregate feature information from local graph neighborhoods using neural networks. The convolution operation in a GCN is a two-stage process consisting of a *neighborhood aggregation stage* and a *center-neighbor combination stage*. The *neighborhood aggregation stage* learns the representation of a neighborhood by transforming and aggregating the feature information from the neighborhood. The *center-neighbor combination stage* learns the representation of the central node by combining the representation of the neighborhood with the features of the central node.

By building an explicit graph representation on the user-item interactions, the effectiveness of applying GCNs has been demonstrated in [3, 5, 21, 37, 40]. PinSAGE [40] constructs a bipartite graph on the pin-board structure in the Pinterest framework; Neural Graph Collaborative Filtering (NGCF) [37] builds a user-item bipartite graph structure and further emphasizes the interactions between the neighbors and the center node during the neighborhood aggregation process. [21] clusters the users into communities and make predictions in each community collaboratively. In order to better incorporate information from heterogeneous interaction types (search, guide, click, etc.) or interaction motives, several graph-based solutions have been proposed in [3, 5, 24].

Despite the effectiveness of these methods, we perceive three important limitations. *First*, during the neighborhood aggregation of a GCN layer, permutation invariant aggregation functions such as element-wise mean/sum or max-pooling are usually performed as the aggregator to generate the neighborhood embedding [11, 12, 17]. This aggregation destroys pairwise relational information between individual neighbor-neighbor pairs. The GCN therefore is forced to ignore the fine structure of neighborhood relations. In addition, user-user and item-item feature interactions are also very important signals and have been demonstrated to be useful for CTR prediction [10, 23], sequential recommendation [34] and collaborative filtering [37]. Most CF methods take the general form $u_i \cdot v_j$ to learn the low-rank relationships between embeddings. [2] shows that first-order neural network layers are inefficient in modeling low-rank relations. This suggests that aside from aggregating the first-order neighborhood information, the pairwise relationships between individual neighbors should also be modeled.

Second, the user-item bipartite graph in recommender systems is, by its very nature, a heterogeneous graph, containing two distinct entities: user nodes and item nodes. If we take a user node as an example, the one-hop neighborhood contains the item nodes that the user has interacted with before. The two-hop neighborhood contains other user nodes that share a similar click/check/download history with the central node. Traditional GNNs [12, 17, 40] do not consider the nature of node types when the information is being aggregated across neighborhoods.

Instead of recursively updating neighbor embeddings at different depths without considering their node types, learning different neighborhood embeddings from the user and item nodes independently can be more effective.

Third, because the neighborhood embeddings aggregated from each layer are from different entities, they represent different information. It is important for the model to combine (aggregate) the different neighborhood embeddings corresponding to different layer depths. We call this *cross-depth ensemble*, and to the best of our knowledge, existing works have not explored such a strategy.

In this paper, we propose a novel graph convolutional neural network-based recommender system framework, Neighbor Interaction Aware Graph Convolutional Neural Networks, **NIA-GCN**, which has three key innovations:

- *Capturing the relational information between neighbors during neighborhood aggregation*: In NIA-GCN, we propose a **Pairwise Neighborhood Aggregation (PNA) layer** to capture relationships between pairs of neighbors at each GCN layer. PNA applies element-wise multiplication between every two neighbor embeddings to learn user-user and item-item relationships, which are important signals in recommendation.
- *Exploit the heterogeneous nature of the user-item bipartite graph*: In our proposed model, instead of recursively updating neighborhood embeddings, we introduce parallel graph convolution networks **(Parallel-GCNs)**, which can independently aggregate information from node entities at different depths with respect to the central node. Such an approach is a much better match to the heterogeneous nature of the user-item bipartite graph.
- *Cross-Depth Ensemble*: We propose a novel **Cross-Depth Ensemble (CDE)** layer to capture the user-user, item-item and user-item relationships in neighborhoods in the graph. This layer allows predictions to take into account more complicated relationships across different depths in the graph.

We conduct empirical studies on four public benchmark datasets. We also generalize our method to a large-scale commercial dataset, with more than 25 million user-item interaction records. Extensive results and ablation studies demonstrate the superiority of NIA-GCN over the strongest state-of-the-art graph-based models. We conduct an online experiment to demonstrate the superiority of NIA-GCN over a competitive method. It improves CTR and CVR by 10.19% and 9.95% on average across a ten-day AB test.

2 RELATED WORK

2.1 Model-based Collaborative Filtering Models

Model-based CF methods learn the similarities between items and users by fitting a model to the user-item interaction data. Latent factor models are common, such as probabilistic Latent Semantic Analysis (pLAS [15]) and the most widely used approach, Matrix Factorization (MF [20]). Koren proposed SVD++ [19], a model which combines information about a user's "neighbors", i.e., items she

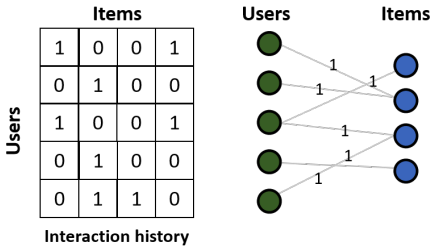


Figure 1: The user-item bipartite graph in recommendation.

has interacted with, and matrix factorization for prediction. Factorization machines [29] provide a mechanism to incorporate side information such as user demographics and item attributes.

MF-based methods are limited because they are confined to the inner-product mechanism to measure the similarity between embeddings of users and items. Recently, neural networks have been incorporated into collaborative filtering architectures [7, 14]. These use a combination of fully-connected layers, convolution, inner-products and sub-nets to capture complex similarity relationships.

2.2 Graph-based Recommendation

In the collaborative filtering and recommender system literature, there has been a considerable research effort focusing on developing techniques to employ graph representations and thus learn from the pairwise relationship and contextual information. Early works [9, 39] used label propagation and random walks on the user-item interaction graph to derive similarity scores for user-item pairs. More recent works have started to apply GNNs [26, 33, 35, 37, 40]. In addition to these approaches, there has been a body of work that has employed spectral-based GNNs for recommendation [28, 42]. Although these methods can be effective, they perform operations in the spectral domain, requiring the identification of eigenvectors of the Laplacian of the graph, which is prohibitively computationally expensive for large graphs and prevents these methods from being applied in large-scale recommender systems. Graph Convolutional Matrix Completion (GCMC) [35] treats the recommendation problem as a matrix completion task and employs a graph convolution autoencoder. PinSage [40] applies a graph neural network on the item-item graph formed by modeling the similarity between items. PinSage has been reported to provide significant performance gains for the Pinterest recommendation system. GNNs have also been employed for the task of social recommendation in [6, 32] and the incorporation of knowledge graph information in news recommendation [36], such as social interactions or item attributes. Another line of work deals with the complex and heterogeneous interaction types between user and item in large-scale e-commerce networks [3, 5, 24] which is different from the problem that we address in this paper (a single type of interaction between user and item). However, our proposed approach has the potential to generalize to the case of multiple interaction types.

Beyond the recommendation setting, there is related work in which GNNs have been employed to learn node embeddings for other tasks. In [38] Xu et al. identified conditions under which the aggregation schemes on graphs can represent universal functions over the multiset of a node and its neighbors. They proposed the Graph Isomorphism Network (GIN), a GNN involving multi-layer perceptrons (MLPs), as a concrete realization for the graph classification task. For the anomaly detection task, GeniePath [25] integrated LSTMs to aggregate information across different GCN layers. Although these works were designed for different graph learning tasks, they can act as a viable alternative for recommendation. We have included them as baselines to have a more comprehensive comparison with existing works. This allows us to compare some of the techniques used in these methods (e.g., attention, LSTMs, MLPs) with the corresponding modules proposed in our paper.

3 PROBLEM STATEMENT

Graph-based recommender systems build interaction graphs from historical responses. The set of user-item interactions can be readily modeled as a bipartite graph (as shown in Figure 1) $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, where each $u \in \mathcal{U}$ represents a user and each $v \in \mathcal{V}$ represents an item. Each edge in $\mathcal{E}$ represents an interaction between a user u and an item v. We embed every user u and item v into an unified d-dimensional embedding space and the node vectors $\mathbf{e}_u^* \in \mathbb{R}^d$ and $\mathbf{e}_v^* \in \mathbb{R}^d$ indicate the latent representation of u and v, as learned via our Graph Neural Network (GNN) based framework.

4 METHODOLOGY

In this section, we introduce three general mechanisms we propose for a graph-based recommendation model. *First*, a **Pairwise Neighborhood Aggregation graph convolution layer (PNA layer)** captures the relational information and commonalities between node pairs in the neighborhood, while maintaining simplicity and permutation invariance. *Second*, a parallel graph convolution mechanism, **Parallel-GCNs** aggregates information independently from different receptive fields (different neighborhood depths). *Third*, a **Cross-Depth Ensemble (CDE)** layer aggregates latent embeddings derived from different depths using Parallel-GCNs. The overall framework of our proposed architecture is depicted in Figure 2.

4.1 Pairwise Neighborhood Aggregation (PNA) Graph Convolution Layer

Graph-based recommendation models build up the representation for each node by iteratively combining the embedding of the node itself with the embeddings of the nodes in its local neighborhood. Most existing methods split this process into two steps: *neighborhood aggregation*, in which an aggregation function operates over a multiset of vectors to aggregate the embeddings of neighbors, and *center-neighbor combination* that combines the aggregated neighborhood vector with the central node embedding. We now elaborate upon these two steps.

Neighborhood aggregation: By considering each neighbor as a *feature* of the central node, the PNA layer captures neighborhood feature interactions by applying element-wise multiplication on every neighbor-neighbor pair. Denote by $\mathbf{q}_i$ and $\mathbf{q}_j$ the i^{th} and j^{th} rows of the neighborhood embedding matrix $\mathbf{Q}^k \in \mathbb{R}^{N_k \times d}$ at layer k, where N_k is the number of depth-k neighbors. The PNA

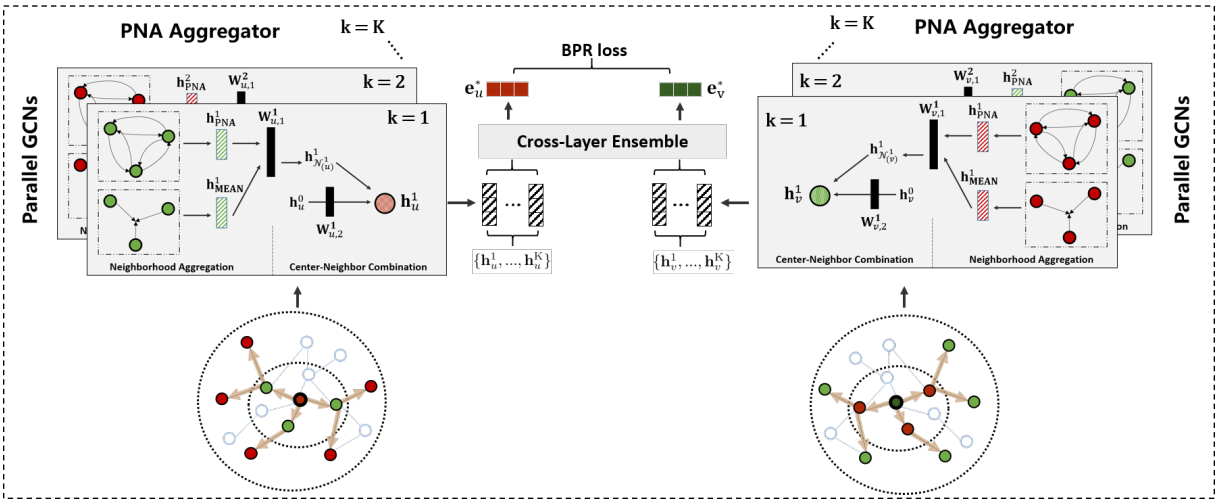


Figure 2: Overall structure of NIA-GCN framework. Arrowed lines present the flow of information. Solid black rectangular boxes denote densely-connected MLPs. Cross-hatched blue rectangles denote embeddings learned from the neighborhood. Rectangles with black slash denote central node embeddings learned from different modules.

computation can be formulated and further simplified as:

$$\mathbf{h}^k_{\text{PNA}} = \frac{1}{N_k}\sum_{i=1}^{N_k}\sum_{j=i+1}^{N_k}\mathbf{q}^k_i \odot \mathbf{q}^k_j = \frac{1}{2N_k}\left(\sum_{i=1}^{N_k}\sum_{j=1}^{N_k}\mathbf{q}^k_i \odot \mathbf{q}^k_j - \sum_{i=1}^{N_k}\mathbf{q}^k_i \odot \mathbf{q}^k_i\right)$$
$$= \frac{1}{2N_k}\left(\sum_{i=1}^{N_k}\mathbf{q}^k_i\sum_{j=1}^{N_k}\mathbf{q}^k_j - \sum_{i=1}^{N_k}\mathbf{q}^{k\,2}_i\right) = \frac{1}{2N_k}\left(\left(\sum_{i=1}^{N_k}\mathbf{q}^k_i\right)^2 - \sum_{i=1}^{N_k}\left(\mathbf{q}^k_i\right)^2\right). \tag{1}$$

PNA can thus be computed in linear complexity $O(N)$.

So far, $\mathbf{h}^k_{\text{PNA}}$ only contains pairwise relational information of the neighborhood. The direct first-order neighborhood information, a coarser summary of the entire neighborhood, is preserved by a sum aggregator. We concatenate these two forms of neighborhood information and pass the resulting vector through a standard MLP to generate the local neighborhood embedding $\mathbf{h}^k_{\mathcal{N}^k_{(u)}}$:

$$\mathbf{h}^k_{\text{MEAN}} = \frac{1}{N_k}\sum_{i}^{N_k}\mathbf{q}^k_i\,, \tag{2}$$

$$\mathbf{h}^k_{\mathcal{N}^k_{(u)}} = \mathbf{W}^k_{u,1}\left[\mathbf{h}^k_{\text{PNA}}\ ;\ \mathbf{h}^k_{\text{MEAN}}\right]. \tag{3}$$

Here [;] represents concatenation, $\sigma(\cdot)$ is the tanh activation function, and $\mathbf{W}^k_{u,1}$ is the layer-k (user) aggregator weight. It is shared across all central user nodes at layer k. The neighborhood aggregation function we specify is permutation invariant/equivariant in the same manner as the more commonly used aggregation operations such as summation and mean [18, 27].

Center-neighbor combination: After the aggregation process, every central node is assigned a new embedding by combining its aggregated neighborhood vector with the central node embedding vector itself. The layer-k embedding of the target user u can be represented as:

$$\mathbf{h}^k_u = \sigma\left[\mathbf{W}^k_{u,2}\,\mathbf{h}^0_u\ ;\ \mathbf{h}^k_{\mathcal{N}^k_{(u)}}\right],\quad \mathbf{h}^0_u = \mathbf{e}_u\,, \tag{4}$$

where $\mathbf{e}_u$ is the initial embedding for target node u, $\mathbf{W}^k_{u,2}$ is the weight matrix for central node transformation at layer k. The same operations (with different weight matrices) are applied to generate the layer-k item embedding $\mathbf{h}^k_v$ of item v.

Figure 3 illustrates the full process of the PNA layer. Our proposed PNA layer remains computationally simple but has much greater representation capacity.

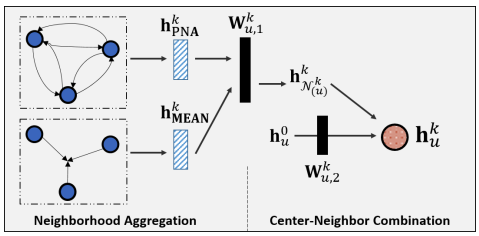


Figure 3: PNA layer. A user node as the central node is assumed. The case of item node as central node is symmetric.

4.2 Parallel-GCNs

To address the heterogeneity of the user-item interaction graph, we design a parallel graph convolution mechanism (Parallel-GCN). After forward sampling the required neighborhood sets up to depth K (for computational efficiency, due to the very large sizes of the graphs, we do not process all nodes in the neighborhood), we learn central node embedding vectors by aggregating independently from each of the K neighborhoods (one per depth). That is, instead of recursively updating the node embedding at layer-$(k-1)$ with the neighbors in layer-k, we learn multiple central node embeddings from neighbors at depths $1, ..., K$ directly, as shown in Figure 4. The output embeddings of Parallel-GCNs for users and items at

different layer are $\{\mathbf{h}_u^1, ..., \mathbf{h}_u^K\}$ and $\{\mathbf{h}_v^1, ..., \mathbf{h}_v^K\}$ for the users and items, respectively.

Most existing GCN algorithms [12, 17, 40] employ the same aggregation and transformation function at each layer for every node. Instead of sharing aggregators for both user nodes and item nodes, we process them separately by learning two sets of aggregators for two different entities on the bipartite graph.

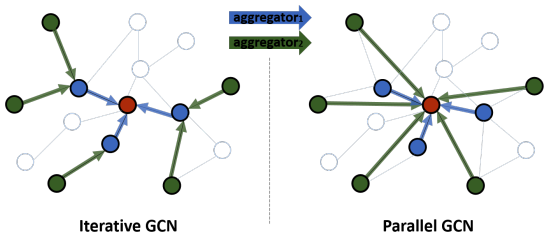


Figure 4: Difference between proposed Parallel-GCN and Iterative GCN. Arrowed lines present the flow of aggregation.

4.3 Cross-Depth Ensemble (CDE) for Final Representation

After obtaining the central node embedding vectors from K parallel GCN layers with different depths of neighborhood, it is important to determine how to merge these different embeddings effectively. In this work we propose a new method to generate the final user and item representations based on the learned embeddings from the neighbors in different depths. In particular, the final representation is derived from K PNA layer outputs and the outputs of the sum aggregator applied to each of the K GCN layers. We combine these $2K$ information vectors using pairwise multiplication, as shown in lines 12-21 in Algorithm 1. Algorithm 1 provides the pseudo-code of the full algorithm.

Figure 5 illustrates the process of CDE when the number of layers $k = 2$ and the central node is a user node. We empirically find that our CDE significantly outperforms traditional aggregation methods, such as element-wise sum or concatenation.

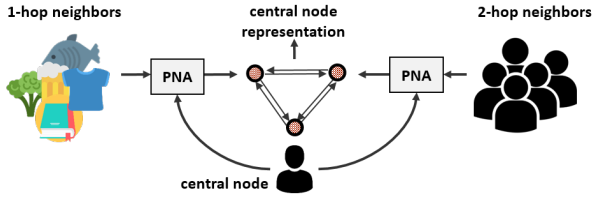


Figure 5: Cross-Depth Ensemble.

4.4 Model Training

We adapt our model to allow forward and backward propagation for mini-batches of triplet pairs $\{u, i, j\}$, where i is a positive item (one the user has interacted with) and j is a random sampled negative item. We optimize our model with respect to the widely-used

Algorithm 1: NIA-GCN embedding generation algorithm (forward propagation, $K = 2$)

Input : Bipartite Graph $\mathcal{G}(\mathcal{U}, \mathcal{V}, \mathcal{E})$; input embedding vectors $\{\mathbf{e}_u, \forall u \in \mathcal{U}\}$ and $\{\mathbf{e}_v, \forall v \in \mathcal{V}\}$; depth $K = 2$; neighborhood function $\mathcal{N}_u : u \rightarrow 2^{\mathcal{U}}$ and $\mathcal{N}_v : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{e}_u^*$ for all $u \in \mathcal{U}$ and $\mathbf{e}_v^*$ for all $v \in \mathcal{V}$

1 $\mathbf{h}_u^0 \leftarrow \mathbf{e}_u, \forall u \in \mathcal{U}, \mathbf{h}_v^0 \leftarrow \mathbf{e}_v, \forall v \in \mathcal{V}$.
2 **for** $k = 1, 2$ **do**
3 **for** $u \in \mathcal{U}$ **do**
4 $\mathbf{h}^k_{\mathcal{N}^k_{(u)}} \leftarrow \mathbf{W}^k_{u,1} \left[\mathbf{h}^k_{\text{PNA}} \,;\, \mathbf{h}^k_{\text{MEAN}}\right]$;
5 $\mathbf{h}_u^k \leftarrow \sigma\left[\mathbf{W}^k_{u,2}\, \mathbf{h}_u^0 \,;\, \mathbf{h}^k_{\mathcal{N}^k_{(u)}}\right]$;
6 **end**
7 **for** $v \in \mathcal{V}$ **do**
8 $\mathbf{h}^k_{\mathcal{N}^k_{(v)}} \leftarrow \mathbf{W}^k_{v,1} \left[\mathbf{h}^k_{\text{PNA}} \,;\, \mathbf{h}^k_{\text{MEAN}}\right]$;
9 $\mathbf{h}_v^k \leftarrow \sigma\left[\mathbf{W}^k_{v,2}\, \mathbf{h}_v^0 \,;\, \mathbf{h}^k_{\mathcal{N}^k_{(v)}}\right]$;
10 **end**
11 **end**
12 **for** $u \in \mathcal{U}$ **do**
13 $\mathbf{h}_u \leftarrow \left[\mathbf{h}_u^1 \,;\, \mathbf{h}_u^2 \,;\, \mathbf{h}_u^1 \odot \mathbf{h}_u^2\right]$
14 **end**
15 **for** $v \in \mathcal{V}$ **do**
16 $\mathbf{h}_v \leftarrow \left[\mathbf{h}_v^1 \,;\, \mathbf{h}_v^2 \,;\, \mathbf{h}_v^1 \odot \mathbf{h}_v^2\right]$
17 **end**
18 $\mathbf{e}_u^* \leftarrow \mathbf{h}_u, \forall u \in \mathcal{U}, \mathbf{e}_v^* \leftarrow \mathbf{h}_v, \forall v \in \mathcal{V}$.

Bayesian Personalized Ranking (BPR) [30] loss:

$$
\begin{aligned}
loss = & \sum_{(u,i,j) \in O} -\log \sigma(\mathbf{e}_u^* \cdot \mathbf{e}_i^* - \mathbf{e}_u^* \cdot \mathbf{e}_j^*) \\
& + \gamma \Big(\sum_{(u,u_p)} ||\mathbf{e}_u^* - \mathbf{e}_{u_p}^*||_2^2 + \sum_{(i,i_p)} ||\mathbf{e}_i^* - \mathbf{e}_{i_p}^*||_2^2 \Big) \\
& + \lambda ||\Theta||_2^2 + \beta (||\mathbf{e}_u^*||_2^2 + ||\mathbf{e}_i^*||_2^2 + ||\mathbf{e}_j^*||_2^2),
\end{aligned}
\tag{5}
$$

where $O = \{(u, i, j) | (u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-)\}$ denotes the training batch. $\mathcal{R}^+$ indicates observed positive interactions. $\mathcal{R}^-$ indicates sampled unobserved negative interactions. u_p is a sample of the 10-nearest neighbors of u, which computer based on the cosine similarity between user's history records. i_p is a sample of the 10-nearest neighbors of i. Here u_p and i_p are used to regularize the distance between users & items with their positive neighbors. Θ is the model parameter set and $\mathbf{e}_u^*$, $\mathbf{e}_i^*$, and $\mathbf{e}_j^*$ are the learned representations. λ and β are regularization terms to prevent overfitting.

5 EXPERIMENTAL RESULTS

We perform experiments on four public datasets and one industrial dataset to fully evaluate our model. Further, we conduct extensive ablation studies on each proposed component (PNA, Parallel-GCNs and CDE). We also provide a visualization comparison between BPRMF and our method for the learned representations on CDs and Gowalla.

Table 1: The overall performance comparison. Underline indicates the second best model performance. We conduct Wilcoxon signed rank test. The result indicates a statistically significant difference between the scores of the best and second-best algorithms for all evaluation metrics.

	Gowalla		Amazon-Electronics		Amazon-CDs		Amazon-Movies	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
BPRMF	0.1608	0.1259	0.0476	0.0278	0.0895	0.0571	0.0667	0.0436
NeuMF	0.1644	0.1298	0.0527	0.0293	0.1008	0.0604	0.0820	0.0511
GC-MC	0.1648	0.1302	0.0411	0.0230	0.1091	0.0632	0.0638	0.0401
PinSAGE	0.1620	0.1276	0.0454	0.0258	0.1265	0.0799	0.0872	0.0559
PinSAGE-lstm	0.1679	0.1318	0.0480	0.0280	0.1269	0.0805	0.0886	0.0565
NGCF	0.1688	0.1320	0.0546	0.0301	0.1258	0.0792	0.0866	0.0555
GIN	0.1637	0.1288	0.0483	0.0286	0.1272	0.0795	0.0905	0.0576
NIA-GCN	**0.1726**	**0.1358**	**0.0646**	**0.0375**	**0.1487**	**0.0932**	**0.1058**	**0.0683**

5.1 Datasets

To evaluate the effectiveness of our method, we conduct extensive experiments on four popular benchmarks: *Gowalla*[1],*Amazon-Electronics*, *Amazon-CDs*, *Amazon-Movies*, and one industrial dataset. These benchmark datasets are publicly accessible, real-world data with various domains, sizes, and sparsity levels. We filter out users and items with fewer than 10 interactions for Amazon-CDs, Amazon-Electronics, and Amazon-Movies. For each user, we randomly select 20% of her rated items as ground truth for testing, The remaining 70% and 10% data constitutes the training and validation set. Table 2 summarizes the statistics of all the datasets.

Table 2: Statistics of evaluation datasets.

Dataset	#User	#Items	#Interactions	Density
Gowalla	29,858	40,981	1,027,370	0.084%
Amazon-Electronics	58,858	20,972	708,348	0.057%
Amazon-Movies	44,439	25,047	1,070,860	0.096%
Amazon-CD	43,169	35,648	777,426	0.051%
Industrial data	403,341	25,092	25,014,232	0.247%

Gowalla: This dataset was collected from Gowalla[2], a popular location-based social network that has had more than 600,000 users since November 2010. We treat locations as items to capture user preferences based on the check-in history.

Amazon-Electronics, **Amazon-Movies** and **Amazon-CDs**: Amazon review[3] is a popular dataset for product recommendations [13]. We select three subsets, Amazon-Electronics and Amazon-Movies.

Industrial data: We sampled and collected download/click histories from 403,341 users and 25,092 apps, with more than 25M interaction records, from a mainstream App store. The details of this dataset and the corresponding evaluation results are presented in Section 6.

[1]resplit from https://github.com/xiangwang1223/neural_graph_collaborative_filtering/tree/master/Data/gowalla
[2]https://snap.stanford.edu/data/loc-gowalla.html
[3]http://jmcauley.ucsd.edu/data/amazon/

5.2 Evaluation Metrics

For all experiments, we evaluate our model and baselines in terms of *Recall@k* and *NDCG@k* (we report with $k = 20$). *Recall@k* indicates the coverage of true (preferred) items as a result of top-k recommendation. *NDCG@k* (normalized discounted cumulative gain) is a measure of ranking quality.

- *Recall@k* indicates the coverage of true (preferred) items as a result of top-k recommendation. The value is computed by $\text{Recall@k} = \frac{|\mathcal{I}_u^+ \cap I_k(u)|}{|\mathcal{I}_u^+|}$, where the target user $u \in \mathcal{U}$, $\mathcal{I}$ is the set of all items, $I_k(u) \in \mathcal{I}$ is an ordered set of the top-k items, $\mathcal{I}_u^+$ is the set of true (preferred) items, and $|\mathcal{I}_u^+ \cap I_k(u)|$ is the number of true positives.
- *NDCG@k*: The Normalized Discounted Cumulative Gain (NDCG) computes a score for $I(u)$ which emphasizes higher-ranked true positives. It is computed as

$$\text{NDCG@}k = \frac{\text{DCG}_\text{k}}{\text{IDCG}_\text{k}} = \frac{\sum_{n=1}^{|\mathcal{I}|} \text{D}_k(n)[i_n \in \mathcal{I}_u^+]}{\sum_{n=1}^{|I|} D_k(n)}$$

where $D_k(n) = (2^{rel_n} - 1)/\log_2(n+1)$ for a relevancy score rel_n. We only consider binary responses, so we use a binary relevance score: $\text{rel}_n = 1$ if $i_n \in \mathcal{I}_u^+$ and 0 otherwise.

5.3 Baseline Algorithms

To demonstrate the effectiveness, we compare our proposed NIA-GCN model with the following methods:
Classical collaborative filtering methods:

- **BPRMF** [30]: A general learning framework for personalized ranking recommendation using implicit feedback.
- **NeuMF** [14]: NeuMF replaces the inner product with a MLP to learn the user-item interaction function.

Graph neural network-based CF methods:

- **GC-MC** [35]: This is a graph-based auto-encoder approach that treats recommendation as a matrix completion problem.

- **PinSAGE** [40]: PinSAGE is a recent industry application of graph representation learning for recommendation. It deploys GraphSage [12] on an item-item graph with both image and text information as the input node features with mean aggregator. A hit rate improvement of more than 20% is reported in [40].
- **PinSAGE-LSTM**: Same overall architecture as PinSAGE but we replaced the mean aggregator with LSTM aggregation proposed in [12]. Even if LSTM is an undesirable aggregator because it is not permutation invariant with respect to the ordering of neighborhood, it has better expressive capability and can model more complicated neighborhood relationships.
- **NGCF** [37]: NGCF is the state-of-the-art graph-based CF method. It explicitly integrates a bipartite graph structure into the embedding learning process to model the high-order connectivity in the user-item graph.

Others:

- **GIN**: GIN has a similar GCN architecture as in [12, 40], but it applies an MLP as the neighborhood aggregator. This has been theoretically proved to be capable of representing any permutation-invariant function of the multi-set of neighbourhood feature vectors.

Our proposed method:

NIA-GCN learns user and item representations on two parallel GCN layers that incorporate pairwise neighborhood interaction during layer-wise aggregation to preserve second-order similarity information. The model combines vector representations across GCN layers to capture their relational information.

5.4 Parameter Settings

We optimize all models using the Adam optimizer with the Xavier initialization procedure [8]. The embedding size is fixed to 64 and the batch size to 5000 for all baseline models and our algorithm. Grid search is applied to choose the learning rate and the coefficient of L_2 normalization over the ranges {0.0001, 0.001, 0.01} and $\{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$, respectively. Number of negative numbers is 10 for all baselines and our algorithm. As in [37], for GC-MC and NGCF, we also tune the dropout rate and network structure. Pre-training [14] is used in NeuMF and GC-MC to improve performance. All the baselines and our algorithm are fully trained with up to 500 epochs.

We implement our NIA-GCN model in Tensorflow. We apply two Parallel-GCN layers with 40 samples in the first layer and 2 samples in the second layer. The output dimension of each layer is fixed as 64 for all experiments. The regularization parameters in the objective function are set to $\lambda = 0.02$, $\beta = 0.02$, and $\gamma \in \{0.002, 0.005, 0.02\}$.

5.5 Comparison with Baselines

Table 1 reports the overall performance compared with baselines. Each result is the average performance from 5 runs with random weight initializations. We conducted Wilcoxon signed rank tests to evaluate the significance of the difference between the best-performing algorithm and the second-best algorithm. We make the following observations:

- NIA-GCN consistently yields the best performance for all datasets. More precisely, NIA-GCN improves over the strongest baselines w.r.t Recall@20 by **2.3%**, **21.7%**, **16.9%** and **17.2%** for Gowalla, Amazon-Electronics, Amazon-CDs and Amazon-Movies respectively. For the NDCG@20 metric, NIA-GCN outperforms the next best method **from 2.9% to 21.8%**. This suggests that exploiting the latent information by capturing the interactions between neighbors and across depths, and learning on a bipartite graph in parallel can lead to much more informative representations.
- By replacing the inner product with neural networks, NeuMF surpasses BPRMF in all cases. However, it fails in modeling the connectivity of users and items, and therefore performs worse than GC-MC and PinSAGE.
- Although both GC-MC and PinSAGE conduct neighborhood aggregation over a bipartite graph, PinSAGE achieves slightly better performance in most cases. A possible reason is that PinSAGE conducts neighborhood graph convolution on a subset of the neighbors via a sampling process instead of using all neighbors, and this can improve the generalization capabilities of the model.
- NGCF, in general, has the best performance or is comparable to the best baseline result in all datasets. However, the high-order embedding propagation process is likely to aggravate overfitting especially on sparse datasets, such as Amazon-Electronics and Amazon-CDs..
- GIN outperforms the other baselines on Amazon-CDs, although the outperformance is not statistically significant. It is not the best performing baseline for the other datasets. The stacked dense layers of GIN offer greater representative capacity [38], but the simpler models are easier to train and can exhibit less variance in predictive performance.
- PinSAGE-lstm outperforms PinSAGE on all datasets. Using LSTM as the neighborhood aggregator provides PinSAGE with larger expressive capability when capturing neighborhood information, compared with simply using the mean aggregator.

Table 3: Ablation studies on each proposed component.

	Architecture	Amazon-CDs	
		Recall@20	NDCG@20
1	Iterative GCNs - mean	0.1278	0.0810
2	Iterative GCNs - PNA	0.1308	0.0822
3	Parallel GCNs - PNA	0.1396	0.0878
4	**NIA-GCN**	**0.1487**	**0.0932**

5.6 Ablation studies

We conduct ablation studies on Amazon-CDs for each proposed component to justify its effectiveness, starting from iterative GCNs [12] with mean aggregator, then adding PNA, Parallel-GCNs and CDE one by one (Table 3 line 1-4).

We compare performance with the best baseline after grid search for the best hyperparameters. We make the following observations:

- All three main components of our proposed model, PNA, CDE, and Parallel-GCNs, are demonstrated to be functional and effective.
- As shown in line 1 and line 2, our devised aggregation module (PNA) greatly improves the performance under iterative GCNs mechanism, compared with mean aggregation. Moreover, it can

be effectively integrated under Parallel-GCNs as well. This proves that it can function as a general neighborhood aggregation approach for GCN-based algorithms and is capable of learning more accurate neighborhood information.
- As shown in line 2 and line 3, Parallel-GCNs lead to significant performance improvement on both datasets, meaning learning node embeddings in parallel can better fit the heterogeneous graph in recommender systems, compared with iteratively updating embeddings for each layer.
- As shown in lines 2-4, combining the PNA layer within Parallel-GCNs with cross-depth ensemble leads to further improvement, indicating that individual embeddings from different layers can effectively capture different information regarding users, items, and the complex user-item relationships.

5.7 Embedding Visualization

Figure 6 provides a visualization of the representations derived from BPRMF and NIA-GCN. Nodes with the same color represent all the items from one user's clicked/visited history, including test items that remain unobserved during training. We random select six users from those who have more than 50 records. The visualization is constructed by projecting the learned embeddings onto a two-dimensional space using t-SNE.

We find that both BPRMF and NIA-GCN have the tendency to encode the items that are preferred by the same user close to one another. However, the clustering effect is less pronounced for BPRMF-generated embeddings. NIA-GCN generates much tighter clusters, showing a strong grouping effect for items that have been preferred by the same user. Also, the learned user representations have a tendency to be close to their item cluster in the latent space.

6 APPLICATION: CANDIDATE SELECTION FOR APP RECOMMENDATION

To further verify the effectiveness of NIA-GNN, we apply it in the recommender system of a mainstream App store, more specifically focusing on the task of candidate selection. In our recommendation scenario, there are tens of thousands of apps. If all the apps are included when forming a recommendation, then a significant latency arises, because it takes time to make prediction on a large number of apps using a complicated model (e.g., a deep neural network). Therefore, candidate selection is needed to select several hundreds of apps from the universal set for each user. A complicated ranking model can then be applied to the reduced candidate set with acceptable latency.

In this section, we first describe the offline dataset, baselines (DSSM and DSSM-FI) and how to apply graph-based models (PinSAGE and NIA-GCN) in candidate selection. We also design online experiments in a live recommender system to verify the performance of NIA-GCN. Specifically, we conduct a ten-day AB test to validate the superiority of NIA-GCN over DSSM in candidate selection. The AB test is conducted in a game recommendation scenario in the game center of a mainstream App store.

6.1 Offline Evaluation

6.1.1 Dataset. We sample and collect 12 consecutive days of user-app click records from the game recommendation center of a mainstream App store for training, and the next 3 days for testing. The user-app click record contains very sparse categorical features for each app (e.g., identification and category) and each user (e.g., download history). We prepare a industrial dataset by removing users and items with fewer than 10 interactions, which leads to approximately 25M records. The detailed statistics are shown in Table 2.

6.1.2 Baselines. We apply an embedding layer to transform sparse and categorical raw features into dense vectors for all baselines and our algorithms.

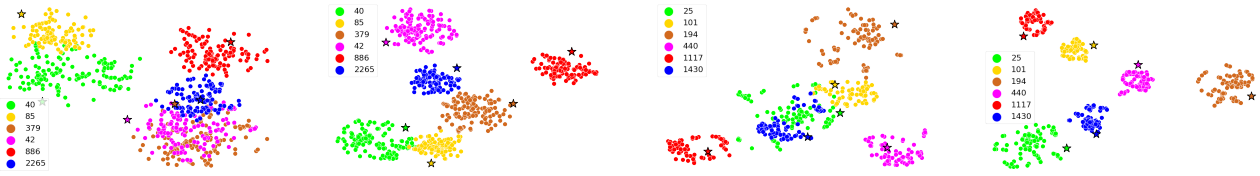
- **DSSM.** [16] proposed a deep structured semantic model (DSSM) with a dual-tower architecture to project queries and documents (in our application, users and apps) into a common low-dimensional space where the similarity between a query and a document is a pre-defined operation on the corresponding latent vectors. This is the current strategy for candidate selection in our commercial App recommendation.
- **DSSM-FI.** As neighborhood interactions are explicitly modeled in NIA-GNN, we propose to reinforce DSSM by modeling feature interactions (FI) within each tower architecture.
- **PinSAGE.** We apply a similar PinSAGE model as in Table 1.
- **NIA-GCN.** We generalize our algorithm for the candidate selection task in the recommender system. We build a bipartite graph based on the user-app click & download interactions.

Table 4: Industrial Application: Performance Comparison of Several Methods in Candidate Selection.

	Logloss	AUC(%)
DSSM	0.23439	97.826
DSSM-FI	0.23382	97.827
PinSAGE	0.24245	97.740
NIA-GCN	**0.22516**	**97.928**

6.1.3 Performance Comparison. For each user-app interaction pair, we random sample 5 apps to generate 5 negative pairs. The goal of this task is to maximize the probabilities of positive pairs to 1 and minimize the probabilities of negative pairs to 0. The probability is obtained by computing the cosine similarity between the user representation and item representation learned from the algorithms with a softmax function. We use the cross entropy loss as the loss function and adopt Adam to optimize baselines and our model.

In the industrial dataset, we use 11 categorical fields for users and 4 categorical fields for apps. By compressing each feature into a 40-dimension latent vector, we obtain user embeddings with $d = 440$ and app embeddings with $d = 160$. For DSSM and DSSM-FI, we apply a 3-layer neural network (with the structure of 512-256-64) for each of towers in the dual-tower architecture. For PinSAGE-mean and NIA-GNN, we apply two-layer GCNs with output dimension=[256, 128], and sample ten 1-hop neighbors and five 2-hop neighbors. We implement all algorithms in TensorFlow. Grid search is applied to choose the learning rate and the coefficient



(a) BPRMF (CDs) (b) NIA-GCN (CDs) (c) BPRMF (Gowalla) (d) NIA-GCN (Gowalla)

Figure 6: Visualization of the t-SNE transformed representations derived from BPRMF and NIA-GCN. Numbers in the legend are user IDs. Stars are users. Points with the same color represent the relevant items from the corresponding user.

of L_2 regularization. Early stopping is applied if the Logloss on the validation set does not increase for 3 successive epochs.

Table 4 shows the experimental results on the industrial dataset. We use two evaluation metrics in our experiments: AUC (Area Under ROC) and Logloss (cross entropy). As we can see, NIA-GNN has better performance than the best baseline (DSSM-FI) in terms of Logloss by 3.7% and in terms of AUC by 0.19%. As the absolute AUC is already near to 1, the AUC improvement achieved by NIA-GNN is not as significant as Logloss improvement. In recommendation tasks, a small improvement in offline evaluation can lead to a significant increase in online test performance. For example, as mentioned in [4], an algorithm with an 0.275% offline improvement led to a 3.9% improvement in an online test. Such an improvement in candidate selection in a recommender system is very valuable, as the App Store is a billion-dollar business.

As another interesting point, PinSAGE performs worse than DSSM, which suggests that applying graph-based models directly may not lead to superior performance. Combining the advantages of deep learning and graph based models may be promising to improve the recommendation performance.

6.2 Online A/B Test

6.2.1 Settings. For the control group, 5% of users are randomly selected and served by DSSM as the candidate selection strategy. For the experimental group, another 5% of users are served by NIA-GCN for candidate selection. After selecting candidates by NIA-GCN or DSSM, the same model is performed to rank the candidate apps selected, for both two groups of users. Due to resource limitations, we do not deploy all the methods in Table 1 and Table 4. Instead, we focus on the performance comparison between NIA-GCN and DSSM (note that DSSM is the current strategy for candidate selection in our system).

6.2.2 Metrics. We adopt two metrics to compare the online performance of NIA-GCN and DSSM, namely realistic Click Through Rate: $rCTR = \frac{\#downloads}{\#impressions}$ and realistic Conversion Rate: $rCVR = \frac{\#downloads}{\#users}$, where $\#downloads, \#impressions$ and $\#users$ are the number of downloads, impressions and visited users in one of the days during AB test.

6.2.3 Results. Figure 7 presents the results of online experiments. The blue and red histograms show rCTR and rCVR improvement of NIA-GCN over DSSM. More specifically, NIA-GCN improves rCTR and rCVR consistently across the whole AB test period and achieves 10.19% and 9.95% improvement on average in terms of rCTR and rCVR. As the ranking model is the same for the control group and the experimental group, such significant performance improvement is achieved completely by employing NIA-GCN instead of DSSM in the candidate selection.

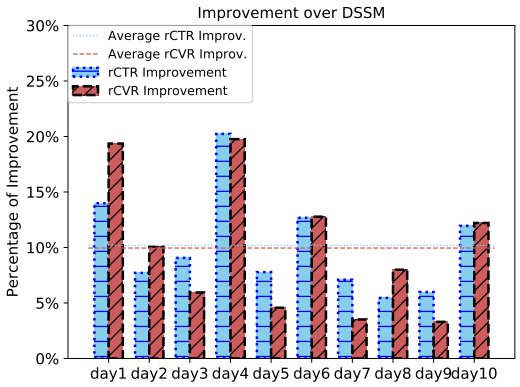


Figure 7: Results of Online A/B Test in Ten Days.

7 CONCLUSION

In this paper we have presented a novel graph-based recommendation model NIA-GCN with three key components: a pairwise neighborhood aggregator, a parallel graph convolution mechanism, and a cross-depth ensemble layer that builds the bridge between them. Extensive experiments on four real-world datasets demonstrate the effectiveness of our approach, and ablation studies quantitatively verify that each component makes an important contribution. Experiments on a large-scale commercial dataset demonstrate its potential industrial impact. Compared to the baseline, NIA-GCN yields better results in a ten-day online AB test, in a mainstream App store.

REFERENCES

[1] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[2] Alex Beutel, Paul Covington, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proc. ACM Int. Conf. Web Search and Data Mining*.

[3] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proc. Workshop Deep Learning for Recommender Systems*.

[5] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided Heterogeneous Graph Neural Network for Intent Recommendation. In *Proc. Int. Joint Conf. Artificial Intelligence*. 2873–2879.

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *Proc. Int. Conf. World Wide Web (WWW)*.

[7] Wenqi Fan, Yao Ma, Dawei Yin, Jianping Wang, Jiliang Tang, and Qing Li. 2019. Deep social collaborative filtering. In *Proc. ACM Conf. Recommender Systems*.

[8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. Int. Conf. Artificial Intelligence and Statistics*.

[9] Marco Gori and Augusto Pucci. 2007. ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In *Proc. Int. Joint Conf. Artificial Intelligence*.

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proc. Int. Joint Conf. Artificial Intelligence*.

[11] William Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. Adv. Neural Inf. Proc. Systems*.

[12] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. Adv. Neural Inf. Proc. Systems*.

[13] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proc. Int. Conf. World Wide Web*.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proc. Int. Conf. World Wide Web*.

[15] Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.* 22, 1 (2004), 89–115.

[16] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proc. ACM Int. Conf. Information & Knowledge Management*.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. Int. Conf. Learning Representations*.

[18] Risi Kondor and Shubhendu Trivedi. 2018. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proc. Int. Conf. Machine Learning (ICML)*.

[19] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*.

[20] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[21] Chong Li, Kunyang Jia, Dan Shen, CJ Shi, and Hongxia Yang. 2019. Hierarchical representation learning for bipartite graphs. In *Proc. Int. Joint Conf. Artificial Intelligence*.

[22] Feng Li, Zhenrui Chen, Pengjie Wang, Yi Ren, Di Zhang, and Xiaoyu Zhu. 2019. Graph Intention Network for Click-through Rate Prediction in Sponsored Search. In *Proc. ACM Int. Conf. Research and Development in Information Retrieval*.

[23] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*.

[24] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. Is a Single Vector Enough? Exploring Node Polysemy for Network Embedding. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*.

[25] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *Proc. AAAI Conf. Artificial Intelligence*.

[26] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory Augmented Graph Neural Networks for Sequential Recommendation. In *Proc. AAAI Int. Conf. Artificial Intelligence*.

[27] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. 2019. Invariant and equivariant graph networks. In *Proc. Int. Conf. Learning Representations (ICLR)*.

[28] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Proc. Adv. Neural Inf. Proc. Systems*.

[29] Steffen Rendle. 2010. Factorization Machines. In *Proc. IEEE Int. Conf. Data Mining*.

[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. Conf. Uncertainty in Artificial Intelligence*.

[31] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2010. *Recommender Systems Handbook* (1st ed.). Springer-Verlag, Berlin, Heidelberg.

[32] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based Social Recommendation via Dynamic Graph Attention Networks. In *Proc. ACM Int. Conf. Web Search and Data Mining*.

[33] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2019. Multi-graph Convolution Collaborative Filtering. In *Proc. IEEE Int. Conf. Data Mining*.

[34] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proc. ACM Int. Conf. Web Search and Data Mining*.

[35] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (Deep Learning Day)*.

[36] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proc. Int. Conf. World Wide Web*.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proc. ACM Int. Conf. Research and Development in Information Retrieval*.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proc. Int. Conf. Learning Representations (ICLR)*, Vol. abs/1810.00826.

[39] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *Proc ACM Conf. Recommender Systems*.

[40] 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 *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining*.

[41] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. 2019. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proc. AAAI Int. Conf. Artificial Intelligence*.

[42] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral collaborative filtering. In *Proc. ACM Conf. Recommender Systems*.