DC-GNN: Decoupled Graph Neural Networks for Improving and Accelerating Large-Scale E-commerce Retrieval

Chenchen Feng*†
fcc19@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Yu He*
herve.hy@alibaba-inc.com
Alibaba Group
Beijing, China

Shiyang Wen
shiyang.wsy@alibaba-inc.com
Alibaba Group
Beijing, China

Guojun Liu
guojun.liugj@alibaba-inc.com
Alibaba Group
Beijing, China

Liang Wang
liangwang.wl@alibaba-inc.com
Alibaba Group
Beijing, China

Jian Xu
xiyu.xj@alibaba-inc.com
Alibaba Group
Beijing, China

Bo Zheng
bozheng@alibaba-inc.com
Alibaba Group
Beijing, China

ABSTRACT

In large-scale E-commerce retrieval, the Graph Neural Networks (GNNs) has become one of the stage-of-the-arts due to its powerful capability on topological feature extraction and relational reasoning. However, the conventional GNNs-based large-scale E-commerce retrieval suffers from low training efficiency, as such scenario normally has billions of entities and tens of billions of relations. Under the limitation on efficiency, only shallow graph algorithms can be employed, which severely hinders the GNNs representation capability and consequently weakens the retrieval quality. In order to deal with the trade-off between training efficiency and representation capability, we propose the Decoupled Graph Neural Networks (DC-GNN) to improve and accelerate the GNNs-based large-scale E-commerce retrieval. Specifically, DC-GNN decouples the conventional framework into three stages: pre-train, deep aggregation, and CTR prediction. By decoupling the graph operations and the CTR prediction, DC-GNN can effectively improve the training efficiency. More importantly, it can enable deeper graph operations to adequately mine higher-order proximity to boost model performance. Extensive experiments on large-scale industrial datasets demonstrate that DC-GNN gains significant improvements in both model performance and training efficiency.

CCS CONCEPTS

• **Information systems** → **Information retrieval**.

*Both authors contributed equally to this work.
†Work done as an intern at Alibaba Group.

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524203

KEYWORDS

Graph Neural Networks, information retrieval, graph acceleration

ACM Reference Format:
Chenchen Feng, Yu He, Shiyang Wen, Guojun Liu, Liang Wang, Jian Xu, and Bo Zheng. 2022. DC-GNN: Decoupled Graph Neural Networks for Improving and Accelerating Large-Scale E-commerce Retrieval . In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3487553.3524203

1 INTRODUCTION

In recent decades, online E-commerce platforms (such as eBay, Amazon, Taobao) have become increasingly popular in people's daily life [9, 34]. These platforms involve search, recommendation, and advertising systems, which are essential parts to help users better find what they need from billions of products, and have attracted more and more attention from both academia and industry [5]. The mainstream solution of these systems typically follows a multi-stage cascade architecture [25]. Taking Taobao sponsored search system as an example, the system can be roughly divided into two stages: retrieval stage and ranking stage. The retrieval stage is responsible for collecting a mini set of relevant products from an enormous corpus with minimum computational resources. The ranking stage is then to rank the mini-set products and determine their impression positions based on sponsored auctions [4]. In both stages, the prediction of click-through rate (CTR) which aims to estimate the probability that a user will click an advertisement (ad) under a search query, plays a vital role [23]. Differently, the CTR prediction in retrieval stage aims to efficiently search top hundreds of candidates from billions of products, while the CTR prediction in ranking stage is sophisticated to sort the retrieved hundreds of candidates. In this paper, we focus on the retrieval stage where the CTR prediction model is normally designed as a two-tower architecture [19] due to the tremendous amount of candidate products. The two-tower CTR model learns ctr-aware embedding vectors between users (and/or queries) and ads, and then approximate nearest

neighbors (ANN) algorithms can be feasibly applied to efficiently search top candidates in online retrieve module.

Most recently, the Graph Neural Networks (GNNs) has become one of the state-of-the-arts for large-scale E-commerce retrieval [6]. Various graph models, such as GCN [20], GAT [39], and GraphSAGE [11], have shown great potential on topological feature extraction and relational reasoning. The core of GNNs is to iteratively aggregate information from neighbors to the target node, which can effectively capture high-order proximity in graphs to alleviate data sparsity problems [24]. The conventional GNNs-based large-scale E-commerce retrieval faces with two major challenges. First, this retrieval scenario normally has billions of entities and tens of billions of relations. Such huge amount of graph data leads to a rapid expansion of training samples, which can severely curtail the graph model's training efficiency. Meanwhile, most GNNs approaches inherently rely on an expensive message-passing procedure to propagate information through the graph. As the number of graph layers growing continuously, the neighbors and computational complexity are increasing exponentially. Hence, the considerable graph data and complex graph operations simultaneously set great limitations on training efficiency of the conventional GNNs-based large-scale E-commerce retrieval. Second, under the limitation on efficiency, only shallow graph algorithms can be adopted. For instance, most current industrial graph models achieve their peak performance by stacking only few layers (e.g., 1 or 2 layers) [24]. Conceptually, shallow graph operations only capture the information of a very limited neighborhood for each target node, which severely hinders the GNNs representation capability, as a larger neighborhood would provide the model with more information [21].

Current researches on tackling the above challenges can be divided into two categories: graph pre-training and lightweight graph models. Graph pre-training mainly consists of skip-gram based embedding models [30, 32, 33] and GNNs pre-training methods [10, 17, 35]. The primary goal of graph pre-training is to learn transferable prior knowledge that can be generalized to downstream tasks and facilitate the model training. However, simultaneously learning node attributes as well as deep structure information remains challenging for graph pre-training methods. Besides, considering the efficiency, deep graph operations still cannot be adopted to mine higher-order proximity that is of significance to improve the model performance. On the other hand, various lightweight graph models have been investigated to make deep graph operations more scalable on large-scale industrial graphs [1, 14, 42]. These approaches propose to reduce the complexity of GNNs by simplifying the graph operators to improve the model training efficiency [21, 41]. Though have made some progress, they can only promote negligible acceleration when deployed in large-scale E-commerce retrieval. Unfortunately, simplifying the graph operators will significantly deteriorate the graph representation capability, which is intolerable in attribute graphs.

To address the aforementioned issues, we propose the Decoupled Graph Neural Networks (DC-GNN), which decouples the conventional end-to-end GNNs-based two-tower CTR prediction framework for large-scale E-commerce retrieval into three stages: pre-train, deep aggregation, and CTR prediction. More specifically, the pre-train stage is encouraged to learn rich node attributes with carefully designed supervised and self-supervised multi-tasks. Next, DC-GNN applies deep aggregation with linear diffusion operators to efficiently capture and preserve higher-order proximity in graphs to further enhance the node embedding. During this aggregation, each target node combines the messages passed from its different k-order neighbors, thus can effectively distill additional important information from deep graph structure. After the first two stages, each node in the graph will preserve graph attributive and structural features to provide feature embedding for CTR prediction stage. It is worthwhile mentioning that each stage of DC-GNN can work compatibly with the existing state-of-the-art (SOTA) modules.

To sum up, the contributions of this work can be summarized as follows:

- We propose a new framework, DC-GNN, which decouples the traditional GNNs-based two-tower CTR prediction paradigm for retrieval into three stages to simultaneously improve training efficiency and representative capability. Each stage of DC-GNN can work compatibly with the existing SOTA methods to further promote the performance.
- In the pre-train stage, we design a supervised link prediction task and a self-supervised multi-view graph contrastive learning task to jointly learn the rich node attributes and enhance the robustness of GNNs.
- In the deep aggregation stage, a set of linear diffusion operators are employed to further mine and preserve higher-order proximity in graphs to enhance the node embedding, while reducing the exponential computational complexity of GNNs to be linear with the graph layers.
- We perform extensive experiments on large-scale industrial datasets, demonstrating significant improvements of DC-GNN in terms of both model training efficiency and representative capability. The contributions of each proposed stage of DC-GNN are also verified.

The rest of the paper is organized as follows. Section 2 reviews the related works, followed by the introduction of the DC-GNN framework in Section 3. Then, Section 4 presents the experimental results and analysis, before wrapping up with a conclusion in Section 5.

2 RELATED WORK

In this section, we review related works of graph pre-training and lightweight GNNs.

2.1 Graph Pre-Training

2.1.1 Skip-gram based model. Early attempts to pre-train graph representation are skip-gram based embedding models inspired by word2vec [22, 27], LINE [38], node2vec [8] and metapath2vec [2]. These works learn the latent representations of nodes by treating the truncated random walks in graphs as the equivalent of sentences. Most of them explore neighborhood and structural similarity of nodes and follow the underlying assumption that nodes closely connected should be projected into the embedding space at a closer distance. The learned embedding via above methods is bundled with the training graph, leading to relatively poor generalization.

2.1.2 Pre-training GNNs. The primary goal of pre-training GNNs is to learn transferable prior knowledge that can be generalized to

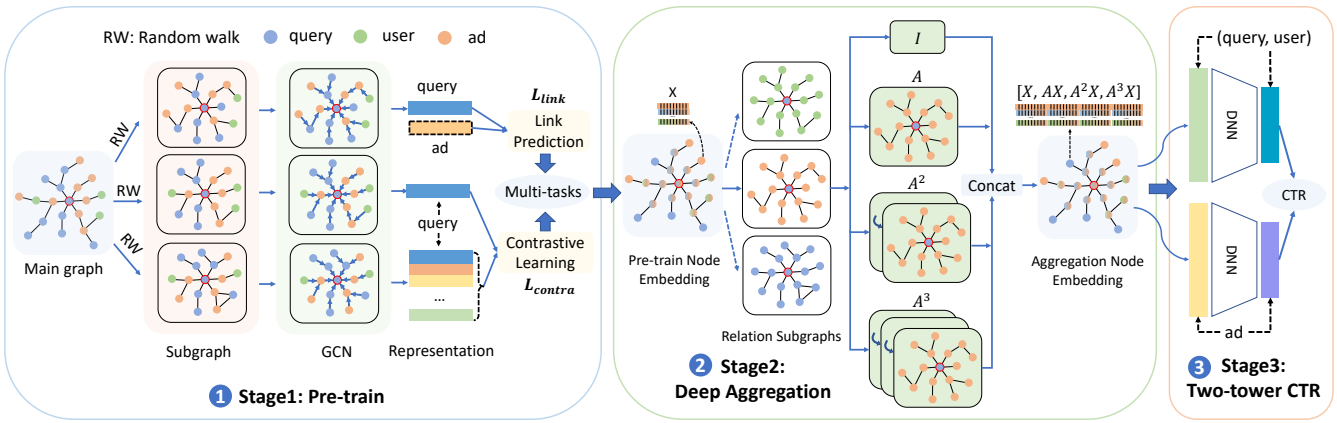

Figure 1: DC-GNN Framework. In pre-train stage, multi-tasks are designed to learn rich node attributes and improve generalization. In deep aggregation stage, heterogeneous linear diffusion operators are proposed to capture higher-order graph structures. Two-tower CTR stage takes as input previous generated node embedding to perform prediction.

downstream tasks [26], while reducing the labeling burden and making the most of abundant unlabeled data. GNNs pre-training methods typically optimize GNNs with carefully designed related tasks. For example, [17] explores three self-supervised pre-training tasks to capture structural information of a graph, while [36] optimizes the graph-level representation via maximizing the mutual information between graph-level representation and the representation of substructures of different scales. [16] designs a self-supervised attributed graph representation task to pre-train GNNs. In contrast to the above either focus on graph-level or node-level methods, [15] pre-trains graph with different strategies at both local and global levels to enhance the generalization. On another line, contrastive learning has emerged as powerful strategy for graph representation learning [29, 40, 46]. For instance, [31] designs a subgraph instance discrimination task and leverage contrastive learning to learn transferable graph structural representations. [37, 43, 46] explore different graph augmentation schemes in contrastive learning — node drop, edge drop and random walk — to generate multiple graph views. [12] uses a graph diffusion to generate an additional view of a graph to achieve better representation performance.

Our DC-GNN framework differs from graph pre-training methods in two aspects. First, simultaneously learning graph attributive and deep structural properties remains challenging for graph pre-training methods. In contrast, DC-GNN focuses on learn node attributes with multi-tasks in pre-train stage, while effectively capturing higher-order proximity in deep aggregation stage. Second, considering the training efficiency and complexity in large-scale industrial graph, complex graph operations are difficult to apply in graph pre-training methods. By decoupling the attributive and deep structural properties learning, DC-GNN can significantly improve the GNNs representation capability and training efficiency.

2.2 Lightweight GNNs

Scaling GNNs to industrial settings is known as a major challenge, as graph normally mount billions of entities and tens of billions of relations. Meanwhile, in typical graphs the exponential growth of neighborhood size with increase of the filter receptive field corresponds to the significant computational complexity. Many sampling-based methods have been investigated to reduce the computational complexity [11]. Most Recently, Various lightweight GNNs have been proposed to break the scalability bottleneck. For example, [14, 42] propose to remove nonlinearities and collapse weight matrices between graph consecutive layers to reduce the complexity of GNNs. [1, 21] explore the relationship between GNNs and PageRank [28] and propose an efficient approximation of information diffusion in graphs to construct a fast-training graph model. [7] proposes a scalable inception architecture that combines graph convolutional filters of different sizes to accelerate training.

Our DC-GNN framework outperforms lightweight GNNs in large-scale E-commerce retrieval in two aspects. First, lightweight GNNs concentrate on optimizing graph operators, which can only promote negligible acceleration when deployed in industrial settings. DC-GNN decouples such conventional framework so that can gain great efficiency improvements. Second, simple simplification of graph operators can deteriorate the graph representation capability, especially in attribute graphs. In contrast, DC-GNN can effectively learn graph attributes and structures, enhancing model representation capability and training efficiency.

3 METHODOLOGY

In this section, we first provide an overview of the proposed DC-GNN framework. Then, we elaborate on each stage, starting with the multi-tasks based pre-train stage, followed by the deep aggregation stage. Finally, we briefly introduce the two-tower CTR prediction stage.

3.1 Notations

Before introducing the method, we first give the definitions of notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ and $\mathcal{E} = \{e_1, e_2, ..., e_{N\times N}\}$ are node set and edge set respectively. $\mathcal{G}$ has node attributes $\{\mathcal{X}_v \in \mathbb{R}^{N\times F} \mid v \in \mathcal{V}\}$ of dimension F. Assume that the adjacency matrix of $\mathcal{G}$ is $\mathcal{A} \in \{0, 1\}^{N\times N}$ which is

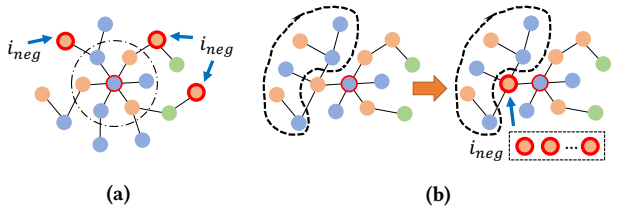

Figure 2: Hard Negative Mining. (a) Hardness adjustable k-hop negatives. (b) Structure negatives.

normally sparse and $\mathcal{A}_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$. We denote the set of neighbors of a node v as $\mathcal{N}_v$. The objective of pre-train stage is to learn a GNNs encoder, i.e. $f^1 : (\mathcal{X}_v, \mathcal{A}) \rightarrow \mathbb{R}^{N \times F'}$, taking as input the node attributes and graph structures, that can generate node embedding h_v^1 in low dimensionality F'. Note that $h_v^1 = f^1(\mathcal{X}_v, \mathcal{A})$ can be further used in downstream tasks. Similarly, we define the node representation as h_v^2 and the embedding dimensionality as F'' in deep aggregation stage. Correspondingly, the objective of deep aggregation stage is $f^2 : (h_v^1, \mathcal{A}) \rightarrow \mathbb{R}^{N \times F''}$. Then, the CTR prediction stage takes h_v^2 as input to acquire the prediction score for retrieval.

3.2 DC-GNN Framework

DC-GNN essentially decouples the traditional end-to-end GNNs-based two-tower CTR prediction framework for retrieval into three stages, aiming to simultaneously improving the GNNs representation capability and training efficiency. As shown in Fig. 1, in the pre-train stage, we design a supervised link prediction task and a self-supervised multi-view graph contrastive learning task to jointly learn node attributes and improve the robustness of GNNs. Next, in the deep aggregation stage, we propose heterogeneous linear diffusion operators to efficiently mine higher-order graph structural properties for further improving the representation capability. Finally, the enhanced node embedding provides dense feature input for the two-tower CTR prediction stage.

3.3 Pre-train

To effectively learn rich node attributive information, we first design a supervised link prediction task in the pre-train stage. Furthermore, we supplement a self-supervised multi-view graph contrastive learning task to enhance the GNNs robustness and generalization. Stage 1 in Fig. 1 represents the procedure of pre-train. The large-scale heterogeneous industrial graph basically includes *query*, *user* and *ad* nodes and each node carries rich attributes. The edges in graph represents click behaviours. Considering the efficiency, we perform random walks (RW) on the heterogeneous graph to generate three subgraphs for each target node. The first subgraph is used for link prediction task and the remaining subgraphs for multi-view graph contrastive learning task.

3.3.1 Link prediction. Link prediction [45] is to predict whether two nodes in graph are likely to have an edge. As illustrated in Stage 1 in Fig. 1, the pre-train graph encoder aggregates information from neighbors to generate the target node embedding. We pre-train the graph encoder with the task that predicts whether there is an edge between *query* and *ad* nodes in the first subgraph. To optimize the model parameters, we frame the link prediction task as supervised learning and adopt the NCE loss [27]. The optimization objective can be defined as:

$$\mathcal{L}_{link} = \sum_{(q, i_p) \in \mathcal{E}} (-\log \sigma(f_s(q, i_p)) - \sum_k \log(1 - \sigma(f_s(q, i_n^k)))), \quad (1)$$

where f_s measures the similarity between two vectors, which is set as cosine similarity function. q and i_p denote a pair of positive samples, indicating that there is an edge between q and i_p in graph. i_n^k is the k-th negative sample. Note that we use the click relation as the edge in graph and thus this objective is aligned with the goal of CTR prediction stage to some extent, which means that we use the CTR to guide the node embedding update in pre-train.

Hard negatives mining (i_n^k) can promote the embedding learning by making the model better at differentiating between similar results [18]. In the link prediction task, to impose GNNs on node attributes learning, we explore two kind of hard negatives.

Hardness adjustable k-hop negatives. As shown in Fig. 2a, taking the target node as *query* as an example, we select the k-hop (k≥2) *ad* neighbors as negative samples. It is worthwhile noticing that we can adjust the hardness of *ad* negatives by changing the parameter k. The *ad* neighbors that are closer to the target node in graph are more difficult, while those farther are simpler. This kind of negatives ignore the graph structure and impose the GNNs on node attributive information learning.

Structure negatives. Another type of negatives is represented in Fig. 2b. We retain the neighborhood structure of the positive *ad* and replace the positive *ad* with a negative one coming from the global negative sampling strategy. Conceptually, such structure negatives generate a fake subgraph with the same topology as the original true one, and retain the randomness of the global negatives. Thus, in the case of similar graph structures of positives and negatives, the GNNs will pay more attention on the node attributes learning. In addition, since GNNs inherently relies heavily on the graph structure, such negatives can also help to alleviate the over-smoothing problems.

3.3.2 Multi-view graph contrastive learning. In the industrial settings, the node embedding update is usually carried out in the form of subgraphs due to the limitation on efficiency. We select random walk (RW) to generate subgraphs following [44], as it can preserve much semantics in local structure which is consistent with the objective of our pre-train stage. The generation of subgraphs inherently introduces randomness and interference, which implies that for the same target node there is a high probability each generated subgraph is different. Hence, to enhance the robustness of GNNs and capture nodes generalization characteristics, we supplement a self-supervised multi-view graph contrastive learning task. As shown in Fig. 1 Stage 1, the second and third subgraphs are two augmented views of the same target node, which are treated as positive pairs (i.e., q_1 and q_2). The augmented views of any different target nodes are treated as negative pairs (i.e., q_1 and v_2). The contrastive loss, InfoNCE [43], is adopted to maximize the agreement of the positive pairs and minimize that of the negative pairs. The

optimization objective for target node *query* can be defined as:

$$\mathcal{L}_{query} = \sum_{\substack{q_1 \in v_q, \\ v_q \in \mathcal{V}}} -\log \frac{exp(wf_s(q_1, q_2))}{\sum_v exp(wf_s(q_1, v_2))}, \tag{2}$$

where v_q represents the set of *query* nodes in graph. w is the *temperature* parameter in softmax, and f_{sim} the similarity measurement function. Similarly, we can drive the optimization objective for target node *user* and *ad*:

$$\mathcal{L}_{user} = \sum_{\substack{u_1 \in v_u, \\ v_u \in \mathcal{V}}} -\log \frac{exp(wf_s(u_1, u_2))}{\sum_v exp(wf_s(u_1, v_2))}, \tag{3}$$

$$\mathcal{L}_{ad} = \sum_{\substack{i_1 \in v_i, \\ v_i \in \mathcal{V}}} -\log \frac{exp(wf_s(i_1, i_2))}{\sum_v exp(wf_s(i_1, v_2))}, \tag{4}$$

where v_u and v_i respectively denote the set of *user* and *ad* nodes in graph. Combining the above three losses, we drive the objective function of the multi-view graph contrastive learning task as:

$$\mathcal{L}_{contra} = \mathcal{L}_{query} + \mathcal{L}_{user} + \mathcal{L}_{ad}. \tag{5}$$

Finally, we leverage a multi-task training strategy to jointly optimize the link prediction and multi-view graph contrastive learning tasks. The loss function of the pre-train stage can be defined as:

$$\mathcal{L}_{total} = \mathcal{L}_{link} + \lambda_1 \mathcal{L}_{contra} + \lambda_2 \|\theta\|_2^2, \tag{6}$$

where θ is the model parameters, and λ_1, λ_2 the hyper-parameters to balance the $\mathcal{L}_{contra}$ and L_2 regularization. After the first stage, each node embedding in graph has learned rich attributive information.

3.4 Deep Aggregation

Following the pre-train stage, deep aggregation aims to mine higher-order graph structures to further enhance the node embedding. It is well-recognized that the recursive message-passing procedure of GNNs incurs exponential computation complexity. Meanwhile, stacking multiple GNNs layers are prone to over-smoothing, as too many layers lead to indistinguishable node representations [24].

To tackle the above two issues, we propose a set of heterogeneous linear diffusion graph operators which extend the scalable inception architecture [7] to heterogeneous graphs. As illustrated in Fig. 1 Stage 2, three relation subgraphs are first derived for each target node in the heterogeneous graph. Taking the target node as *query* as an example, its *query* subgraph, *user* subgraph, and *ad* subgraph can be respectively sampled and each subgraph only contains the nodes of one relation type. $\{\mathcal{A}, \mathcal{A}^2, \mathcal{A}^3, ...\}$ respectively represent pre-computed first-order, second-order, third-order and higher-order adjacency matrices of the relation subgraph, aiming at capturing and maintaining different k-order proximity in graphs. Note that distinguish the information of different k-order is necessary because it will force a more adequate learning of the graph structure, which is the explicit objective of our deep aggregation stage. Assume that the node representation learned from the first stage is $\mathcal{X}$, i.e., $h_v^1 = \mathcal{X}$, the node embedding enhanced by the second stage can be depicted as follows:

$$h_v^2 = f^2(\mathcal{X}, \mathcal{A}) = [\mathcal{X}, \mathcal{A}\mathcal{X}, \mathcal{A}^2\mathcal{X}, \mathcal{A}^3\mathcal{X}], \tag{7}$$

where h_v^2 denotes the node representation in deep aggregation. The representations of different relation graphs will be concat together.

The heterogeneous linear diffusion operation essentially is a parallel forward propagation strategy, which can effectively reduces the exponentially increasing computational complexity in graphs to be linear with the graph layers. Hence, it can accelerate the calculation at the graph operation level. In addition, we preserve the node locality (i.e., $\mathcal{X}$) in the final node representation to stay close to the target node to alleviate the over-smoothing problem, while leveraging the information from a larger neighborhood to enhance the node embedding. After this stage, each node in the graph, on the basis of the learned rich attributes, is enhanced by the distilled additional important information from deep graph structure. Next, the enhanced node embedding provides dense feature inputs for the two-tower CTR prediction stage.

3.5 CTR Prediction

Due to the tremendous amount of candidate products, the CTR model in retrieval stage is typically designed as a two-tower architecture. Therefore, in this stage, we establish a two-tower CTR prediction model where one tower is (*query,user*) and the other is *ad*. As illustrated in Fig. 1 Stage 3, the node embedding generated by the first two stages provides dense feature input for the CTR model. The objective function in this stage can be defined as follows:

$$\mathcal{L}_{CTR} = \sum(-\log \sigma(f_s((q,u), i_{clk})) - \sum_k \log(1 - \sigma(f_s((q,u), i_{pv}^k)))), \tag{8}$$

where f_s denote similarity measurement function. We use (*query,user*)-*ad* clicked results as the positives (i.e., (q,u) and i_{clk}), and the results impressed but not clicked as negatives (i.e., (q,u) and i_{pv}).

4 EXPERIMENTS

In this section, we conduct experiments to verify the model performance and training efficiency of the proposed DC-GNN framework. First, we compare the DC-GNN with SOTA competitors in Section 4.2, including lightweight graph models and graph pre-training methods. Next, in Section 4.3 we respectively evaluate the multi-tasks and the designed negatives in pre-train, and analyze the aggregation layers and neighbors in deep aggregation stage. In Section 4.4, we conduct the parameter studies of DC-GNN.

To sum up, the experiments are carried out to answer the following questions:

- **RQ1**: How dose the proposed DC-GNN perform compared with the SOTA approaches?
- **RQ2**: What are the contributions of each stage of DC-GNN?
- **RQ3**: What effect can be achieved by tuning the parameters?

4.1 Experimental Setup

4.1.1 Dataset. To evaluate the effectiveness and efficiency of DC-GNN, we conduct experiments on the industry-scale heterogeneous graph dataset collected from Taobao. The Taobao graph is constructed by collecting 7-days user click behavior logs on Taobao's sponsored search platform, which consists of three types of nodes, i.e. *query* (q), *user* (u), and *ad* (i), and three types of edges i.e. a *user* searches a *query* (u-q) and then clicks an *ad* (u-i, q-i). The graph carries billions of nodes and tens of billions of edges, and the statistics are shown in Tab. 1. Following [13, 24], we randomly

Table 1: Statistics of the Taobao dataset.

Categories	Types	Statistics	Total
Nodes	q*	305,728,622	916,111,617
	u*	348,409,723	
	i*	567,396,166	
Edges	q-i	2,343,149,048	10,246,182,743
	u-i	5,917,118,417	
	u-q	1,985,915,278	

* q, u, and i respectively denote *query*, *user*, and *ad*.

split the dataset into training and testing sets with the ratio of 9:1 for evaluation. As we can see, such large-scale industrial heterogeneous graph is suitable for analyzing the performance of our method and the competitors.

4.1.2 Evaluation Metrics. We adopt time consumption to evaluate training efficiency and two widely used evaluation metrics, *AUC* and *Hit-rate@K*, to evaluate the model performance. The less time consumption (↓) means a higher training efficiency and A higher *AUC* (↑) implies better prediction performance. As for the *Hit-rate@K*, we retrieve the top-k preferred ads for each pair of <query, user> based on CTR scores from the whole corpus with billions of products, and evaluate the top-k retrieved list against positive ads in the test set. Then the *Hit-rate@K* can be define as the proportion of positive ads hit in the top-k retrieved list over the total number of positive ads in the test set, i.e., $Hit\text{-}rate@K = \frac{NumberOfHits@K}{|Positives|}$. This metric directly measures the probability that the positive ads can be retrieved according to the applied CTR model. A higher *Hit-rate@K* (↑) indicates a better retrieval quality. In our experiments, K is separately set to be 100, 200, 500, and 1000.

4.1.3 Parameter Settings. The graph generator is implemented in the Alibaba Open Data Processing Service (ODPS) distributed data analysis framework [1] and the model is implemented with Tensorflow [2]. The basic embedding dimension of all models is set as 64 and the batch size is fixed to 1024. The models are optimized with AdaGrad [3] optimizer and the learning rate is 0.1 with the decay rate of 1. The L_2 regularization coefficient λ_2 is searched in the range of {1e-6, 1e-7, 1e-8}. In addition, we tune the GNNs layer size in the range of {1, 2, 3, 4, 5} to verify the model performance.

4.2 Comparison with SOTA Methods (RQ1)

In this section, we compare DC-GNN with the SOTA methods in terms of both model performance and training efficiency. Tab. 2 shows the experimental results. The best results are highlighted in bold and the second best underlined.

4.2.1 Baseline methods. To verify the effectiveness of DC-GNN, we select representative baseline methods belonging to the following two categories: lightweight GNNs, including sampling-based GNNs (LasGNN, GraphSAGE [11]) and efficient propagation GNNs (SGC [42], LightGCN [14]), and graph pre-training models, including skip-gram methods (node2vec [8]) and pre-training GNNs methods (GCC [31], GCA [46]). The brief introduction of these baseline models are as follows:

- **LasGNN**: LasGNN is a sampling-based graph model adapted to the Taobao advertisement retrieval scenario. This method proposes a layer-wise sampling method to reduce the computational complexity, demonstrating good performance.
- **GraphSAGE** [11]: GraphSAGE proposes a sampling strategy to avoid the unpredictable memory and expected runtime, which contributes to a fixed per-batch space and time complexity.
- **SGC** [42]: SGC simplifies GNNs by successively removing nonlinearities and collapsing weight matrices between consecutive graph layers, effectively reducing the excess complexity.
- **LightGCN** [14]: LightGCN simplifies the design of GNNs by only including the most essential component — neighborhood aggregation — to make it more scalable on large-scale graphs.
- **node2vec** [8]: node2vec defines a flexible notion of neighborhood and proposes an improved random walk algorithm to capture the diversity of connectivity patterns in graphs.
- **GCC** [31]: GCC presents a self-supervised contrastive learning based graph pre-training framework to capture topological properties across graphs, achieving good generalization.
- **GCA** [46]: GCA proposes a general contrastive framework for graph representation learning with data augmentation on both topology and attribute levels to encourage model to learn important features.

Table 2: The overall performance of DC-GNN and the SOTA competitors.

Models	AUC	Hit-rate @100	Hit-rate @200	Hit-rate @500	Hit-rate @1000
LasGNN	0.6119	0.4898	0.5608	0.6465	0.7030
GraphSAGE	0.6021	**0.4978**	**0.5699**	0.6573	0.7149
SGC	0.5923	0.4674	0.5432	0.6354	0.6960
LightGCN	0.5920	0.4671	0.5454	0.6415	0.7051
node2vec-P$_f$	0.5532	0.2368	0.2843	0.3587	0.4250
node2vec-P$_t$	0.6135	0.4531	0.5262	0.6170	0.6783
GCC-P$_f$	0.5146	0.0492	0.0866	0.1678	0.2571
GCC-P$_t$	0.5979	0.3643	0.4443	0.5499	0.6248
GCA-P$_f$	0.5217	0.0877	0.1305	0.2085	0.2836
GCA-P$_t$	0.5800	0.3910	0.4529	0.5654	0.6462
DC-GNN-P$_f$	0.5736	0.3092	0.3797	0.4781	0.5530
DC-GNN-P$_t$	0.6297	0.4767	0.5523	0.6460	0.7080
DC-GNN	**0.6543**	0.4910	0.5677	**0.6650**	**0.7322**

4.2.2 Overall performance. Tab. 2 shows the performance comparison between DC-GNN and the competitors in terms of *AUC* and *Hit-rate@K*. For fairness, all methods are optimized by the same learning strategy. The best results are highlighted in bold and the second best underlined. Through the experimental results, we can obtain the following observations.

First, DC-GNN consistently yields the best performance in *AUC* among all SOTA methods. As illustrated in Tab. 2, compared with

[1] https://www.alibabacloud.com/product/maxcompute.
[2] https://www.tensorflow.org

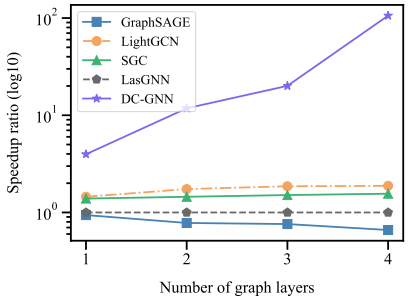

Figure 3: The comparison of training efficiency between DC-GNN and the competitors.

Table 3: Effect of multi-tasks in pre-train stage.

Metrics	AUC		
Tasks	LP*	CL**	LP + CL
DC-GNN-P$_f$	0.5563	0.5475	**0.5736**
DC-GNN-P$_t$	0.5897	0.6004	**0.6297**
DC-GNN	0.6346	0.6329	**0.6543**

* LP denotes the supervised link prediction task.

** CL denotes the self-supervised multi-view graph contrastive learning task.

the sampling-based lightweight GNNs, DC-GNN outperforms the best baseline LasGNN by 4.24%. This is mainly because that DC-GNN can capture important higher-order proximity in graphs which is inaccessible in sampling-based lightweight GNNs. As for the efficient propagation GNNs, DC-GNN achieves 6.20% improvements on the best baseline SGC. Such significant improvements validate that DC-GNN can learn effective features in large-scale attributes graph while efficient propagation GNNs degrade the performance due to the weakened representation ability.

Besides, DC-GNN demonstrates strong performance across graph pre-training methods. As illustrated in Tab. 2, P_f and P_t respectively represent that the embedding are fixed and fine-tunable. DC-GNN achieves 4.08% improvements over the best graph pre-training baseline node2vec-P_t, owing to the fact that deep aggregation stage in DC-GNN can effectively mine important higher-order graph structures to enhance the model performance. We also design two variants of DC-GNN, namely DC-GNN-P_f and DC-GNN-P_t, which directly feed the embedding generated by pre-train stage to the two-tower CTR prediction stage, to verify the performance of our pre-train stage. We observe that DC-GNN-P_f and DC-GNN-P_t respectively achieve 2.04% and 1.62% improvements over the corresponding best fixed and fine-tunable baselines. This indicates that our pre-train stage can better optimize the node embedding under the guidance of the CTR goal. Meanwhile, DC-GNN-$P_{(\cdot)}$ achieves smaller gaps between the fixed and fine-tunable embedding (5.61%) than other graph pre-training methods (node2vec (6.03%), GCC (8.33%) and GCA (5.83%)), implying that our pre-train stage can achieve better generalization and robustness. On the other hand, we evaluate the model performance in *Hit-rate@K* where K is separately set to be 100, 200, 500, and 1000. As shown in Tab. 2, DC-GNN clearly achieve competitive results compared with all baselines, and the advantages gradually enlarge as the K increases, especially in *Hit-rate@1000* (about 3% gains over the second best). The results indicate that our method can not only consistently maintain the recall accuracy over top candidates, but significantly improve the prediction results over mid- and long-tail candidates.

Regarding the training efficiency, we compare DC-GNN with the lightweight GNNs based methods. The training efficiency of LasGNN is set to be the baseline and the speedup ratios of other methods are shown in Fig. 3. It can be seen that, as the number of graph convolutional layers growing continuously, the training complexity of sampling-based lightweight graph models expands dramatically. The efficient propagation based lightweight GNNs, i.e. SGC and LightGCN, achieve higher training efficiency compared to the sampling-based models. This indicates that discarding linear transformation and non-linear activation module promote the faster training, but at the cost of impairing the GNNs representative ability. In contrast, DC-GNN decouples the graph operation and the CTR prediction, allowing the training time independent of the graph structure. Hence, the training efficiency of DC-GNN will not increase significantly as the number of graph layers grows. Note that when the number of graph layers reaches 4, the maximum speedup of DC-GNN exceeds 100. Conceptually, the speedup ratio of DC-GNN will be further enlarged with the increasing of graph layers. As a result, DC-GNN essentially provides a new CTR-based retrieval framework whose training efficiency does not rely on the graph structure and can capture graph's higher-order proximity to enhance the model performance.

4.3 Study on DC-GNN (RQ2)

In this section, we evaluate the effect of each proposed stage of DC-GNN, mainly focusing on the pre-train and deep aggregation stages. We first examine the effectiveness of the multi-tasks and negatives in pre-train. Then, we explore the effect of aggregation layers and neighbors in the deep aggregation stage.

4.3.1 Pre-train. To validate the effect of multi-tasks in pre-train stage, we compare the performance between DC-GNN with and without the specific link prediction task and multi-view graph contrastive learning task. As illustrated in Tab. 3, LP indicates that the model is individually optimized by the link prediction task, and CL the contrastive learning task. The best results are in bold. It can be seen that for DC-GNN and its variants the joint optimization of LP and CL achieves consistently the best performance. In addition, DC-GNN obtains the best results both in individual task and multi-tasks compared to DC-GNN-$P_{(\cdot)}$. This indicates that deep aggregation in DC-GNN plays a vital role in enhancing the model representation capability. The experimental results also demonstrate that the proposed link prediction and contrastive learning task are both indispensable for the pre-train stage.

We also examine the proposed k-hop and structure negatives in pre-train stage. These two negatives are designed to learn rich node attributes. In order to explore the effectiveness of the two kind of negatives, we compare the performance of model training with k-hop negatives, structure negatives and global negatives separately. The experimental results are shown in Fig. 4. It can be seen that

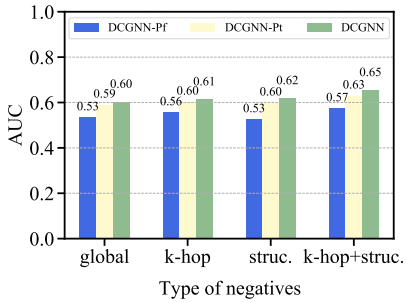

Figure 4: Effect of different negatives in pre-train stage.

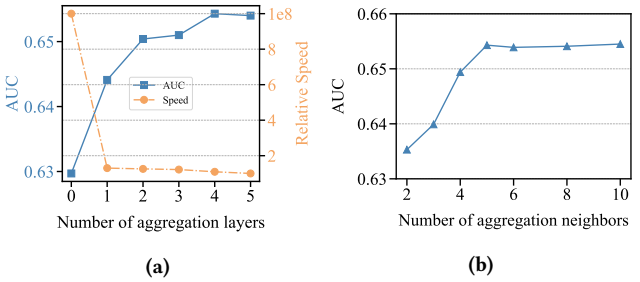

Figure 5: Effect of (a) aggregation layers and (b) aggregation neighbors in deep aggregation stage.

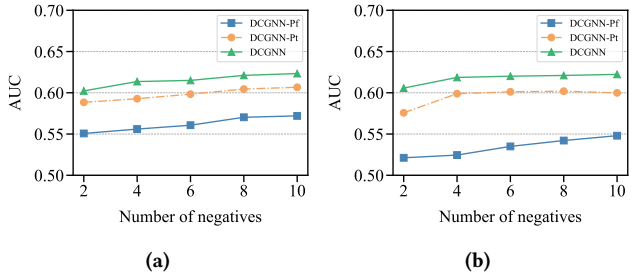

Figure 6: The parameter study of the number of (a) k-hop negatives and (b) structure negatives in pre-train stage.

the model performance optimized individually by k-hop negatives or structure negatives outperforms that by globally sampled negative samples. Jointly learning with k-hop and structure negatives consistently achieves the best results. The results demonstrate that hard negatives can promote the embedding learning by making the model better at differentiating between similar results.

4.3.2 Deep aggregation. To investigate the effectiveness of aggregation layers and aggregation neighbors in deep aggregation, we conduct experiments with the layers increasing from 1 to 5 and neighbors increasing from 2 to 10. Through the experimental results in Fig. 5a, we can see that the model performance increase continuously with the growing of aggregation layers when stacking less than 4 layers. This indicates that the deep structure can mine higher-order neighborhood properties, providing the model with more information to enhance the node embedding. The peak performance of DC-GNN is obtained when stacking 4 layers. After 4 graph aggregation layers, the performance improvement of DC-GNN is smaller. Possible reasons are that increasing more layers is prone to over-smoothing problems that the node representations are indistinguishable. In addition, it can be seen that the proposed heterogeneous linear diffusion operators reduce the computational complexity to be linear with the increase of graph layers. This is because the different k-order proximity is captured and preserved in parallel for each node, making the computational complexity be independent of the graph structure. Taking into account the model performance and efficiency, we adopt 4 aggregation layers in deep aggregation stage of the DC-GNN framework in this work.

According to the effect of the number of aggregation neighbors, we can observe from Fig. 5b that the *AUC* gradually increase with the increasing of aggregation neighbors. When the number of neighbors reaches 5, the graph model achieves its peak performance. After aggregating more than 5 neighbors, the performance improvement of DC-GNN is relatively smaller. Possible reasons are that aggregating more neighbors to the target node significantly increases the model parameters, consequently complicating the network training. Considering the training efficiency, network complexity and model performance, we choose to aggregate 5 neighbors for each target node in the deep aggregation stage.

4.4 Ablation Study (RQ3)

In this section, we conduct parameter studies on the number of k-hop negative samples and structure negative samples. As illustrated in Fig. 6, the experimental results show that the model performance improves continuously with the increasing of the number of k-hop negatives or structure negatives. This motivate us believe that the our hard negatives mining strategies effectively promote the embedding learning. Besides, the k-hop negatives optimization yields better performance than the structure negatives, owing to the fact that the design of k-hop negatives is aligned with the CTR prediction optimization goal to some extent. Considering the training efficiency and model performance, we respectively choose 8 k-hop negatives and 8 structure negatives in the pre-train stage.

5 CONCLUSION

In this work, we propose DC-GNN which decouples the conventional GNNs-based CTR prediction paradigm for large-scale retrieval into three stages — pre-train, deep aggregation, and two-tower CTR prediction — to deal with the trade-off between model performance and training efficiency. The pre-train stage aims to learn rich node attributive information with carefully designed multi-tasks. Then, deep aggregation captures higher-order proximity in graphs to enhance the node embedding. Each stage in DC-GNN can work compatibly with the existing SOTA methods to further boost the model performance. Based on our analysis, the improvement in training efficiency is mainly attributed to the decomposition of graph operations and two-tower CTR prediction. Both multi-tasks based pre-train and deep aggregation stages contribute to the significant improvement of model performance. DC-GNN essentially provides a feasible alternative or complementary framework for large-scale E-commerce retrieval.

REFERENCES

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

[2] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

[3] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[4] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. 2005. *Internet Advertising and the Generalized Second Price Auction: Selling Billions of Dollars Worth of Keywords*. Working Paper 11765. National Bureau of Economic Research. https://doi.org/10.3386/w11765

[5] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. 2019. MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search. In *KDD*. ACM, 2509–2517.

[6] Hongliang Fei, Jingyuan Zhang, Xingxuan Zhou, Junhao Zhao, Xinyang Qi, and Ping Li. 2021. GemNN: gating-enhanced multi-task neural networks with feature interaction learning for CTR prediction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2166–2171.

[7] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198* (2020).

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[9] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, and Dawei Yin. 2020. Hierarchical User Profiling for E-commerce Recommender Systems. In *WSDM*. ACM, 223–231.

[10] Hakim Hafidi, Mounir Ghogho, Philippe Ciblat, and Ananthram Swami. 2020. Graphcl: Contrastive self-supervised learning of graph representations. *arXiv preprint arXiv:2007.08025* (2020).

[11] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[12] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*. PMLR, 4116–4126.

[13] Li He, Hongxu Chen, Dingxian Wang, Shoaib Jameel, Philip Yu, and Guandong Xu. 2021. Click-Through Rate Prediction with Multi-Modal Hypergraphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 690–699.

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

[15] W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. 2020. Strategies For Pre-training Graph Neural Networks. In *International Conference on Learning Representations (ICLR)*.

[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1857–1867.

[17] Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Pre-training graph neural networks for generic structural feature extraction. *arXiv preprint arXiv:1905.13728* (2019).

[18] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[19] 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 *CIKM*. ACM, 2333–2338.

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

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*.

[22] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems* 27 (2014), 2177–2185.

[23] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 539–548.

[24] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware Message-Passing GCN for Recommendation. In *Proceedings of the Web Conference 2021*. 1296–1305.

[25] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade Ranking for Operational E-commerce Search. In *KDD*. ACM, 1557–1565.

[26] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to pre-train graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4276–4284.

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[29] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*. 259–270.

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[31] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1150–1160.

[32] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 459–467.

[33] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 385–394.

[34] Daria Sorokina and Erick Cantú-Paz. 2016. Amazon Search: The Joy of Ranking Products. In *SIGIR*. ACM, 459–460.

[35] Arjun Subramonian. 2021. MOTIF-Driven Contrastive Learning of Graph Representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 15980–15981.

[36] Fan-Yun Sun, Jordon Hoffman, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *International Conference on Learning Representations*.

[37] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. *arXiv preprint arXiv:2106.05819* (2021).

[38] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[40] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In *International Conference on Learning Representations*.

[41] 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.

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

[43] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 726–735.

[44] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.

[45] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems* 31 (2018), 5165–5175.

[46] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*. 2069–2080.