Billion-Scale Bipartite Graph Embedding: A Global-Local Induced Approach

Xueyi Wu
East China Normal University
xueyi.wu.1@stu.ecnu.edu.cn

Yuanyuan Xu
University of New South Wales
yuanyuan.xu@unsw.edu.au

Wenjie Zhang
University of New South Wales
wenjie.zhang@unsw.edu.au

Ying Zhang
University of Technology Sydney
ying.zhang@uts.edu.au

ABSTRACT

Bipartite graph embedding (BGE), as the fundamental task in bipartite network analysis, is to map each node to compact low-dimensional vectors that preserve intrinsic properties. The existing solutions towards BGE fall into two groups: metric-based methods and graph neural network-based (GNN-based) methods. The latter typically generates higher-quality embeddings than the former due to the strong representation ability of deep learning. Nevertheless, none of the existing GNN-based methods can handle billion-scale bipartite graphs due to the expensive message passing or complex modelling choices. Hence, existing solutions face a challenge in achieving both embedding quality and model scalability. Motivated by this, we propose a novel graph neural network named AnchorGNN based on global-local learning framework, which can generate high-quality BGE and scale to billion-scale bipartite graphs. Concretely, AnchorGNN leverages a novel anchor-based message passing schema for global learning, which enables global knowledge to be incorporated to generate node embeddings. Meanwhile, AnchorGNN offers an efficient one-hop local structure modelling using maximum likelihood estimation for bipartite graphs with rational analysis, avoiding large adjacency matrix construction. Both global information and local structure are integrated to generate distinguishable node embeddings. Extensive experiments demonstrate that AnchorGNN outperforms the best competitor by up to 36% in accuracy and achieves up to 28 times speed-up against the only metric-based baseline on billion-scale bipartite graphs.

PVLDB Reference Format:
Xueyi Wu, Yuanyuan Xu, Wenjie Zhang, and Ying Zhang. Billion-Scale Bipartite Graph Embedding: A Global-Local Induced Approach. PVLDB, 17(2): 175-183, 2023.
doi:10.14778/3626292.3626300

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/iBoom2333/AnchorGNN.

1 INTRODUCTION

Bipartite graphs naturally arise as a data structure to model relationships between two types of entities (nodes) with a wide range of applications, covering recommender systems [26, 39], search engines [9], drug discovery [44] and image object recognition [13]. Bipartite graph embedding (BGE), which aims to learn the vectorized representation of each node of the bipartite graph, is a fundamental task that underpins the aforementioned applications [18, 26, 36, 40, 45]. For instance, in the recommender system, a top-K item list recommended to a user is obtained by ranking the dot product of user embedding and item embedding; in drug discovery, BGE can aid in drug-target interaction predictions by link prediction (see Figure 2d). Hence, learning high-quality BGE is essential in boosting downstream applications.

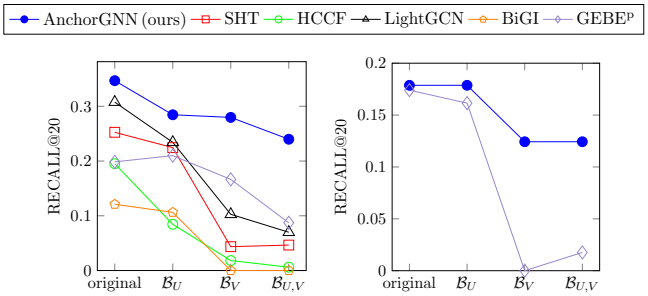
The existing solutions for bipartite graph embedding can be roughly classified into two categories: *metric-based* and *GNN-based* methods. The former [16, 45] learns embeddings based on predefined similarity/proximity metrics between nodes over their constructed approximate paths. Specifically, BiNE [16] performs a large number of biased random walks within nodes' multi-hop neighborhood to build the approximate path context, which is not scalable on large graphs like MovieLens. GEBE$^{\text{p}}$ [45] assigns approximate path importance via a probability mass function and optimizes the learning objective with eigen-decomposition, thus scaling to billion-scale bipartite graphs. However, the embedding quality of both methods is sensitive to predefined measures over the approximate structure.

To enhance embedding quality, many research efforts have been dedicated to designing various GNN models to generate BGE [12, 21, 27, 35, 38, 39, 41, 46]. Upon graph convolutional network (GCN) [24], several studies [21, 34, 46] focus on multi-hop local structure using neighborhood-based message passing (MP), among which LightGCN [21] achieves state-of-the-art performance in efficiency and effectiveness via simplifying the message passing. To further improve representations of two-type nodes, follow-up studies [12, 38, 39] attempt to explore global and local learning simultaneously, i.e., capturing high-order homogeneous interactions and heterogeneous interactions in a unified framework. Among them, global learning is derived based on local learning. For example, SHT [39] first conducts local learning using neighborhood aggregation and then employs global learning based on local learning to generate node embeddings. Generally, GNN-based methods are able to learn higher-quality embeddings than metric-based solutions, but they fail to handle billion-scale bipartite graphs (e.g., MAG) due to the expensive message passing, complex global learning and combination of multiple self-augmentation techniques, as shown in Figure 1. This leads to poor applicability in real-life scenarios, where bipartite graphs consist of millions of nodes and billions of edges.

In a nutshell, existing solutions make some progress in either model scalability or embedding quality, but none of them achieves satisfactory performance in both aspects. Motivated by this and

*Xueyi Wu and Yuanyuan Xu are the joint first authors.

Proceedings of the VLDB Endowment, Vol. 17, No. 2 ISSN 2150-8097.
doi:10.14778/3626292.3626300



(a) MovieLens (10 million edges) **(b) MAG (1 billion edges)**

Figure 1: Top-20 recommendation performance of comparative methods on MoiveLens and MAG. $\mathcal{B}_U$: only binarize the embeddings of U, $\mathcal{B}_V$: only binarize the embeddings of V, $\mathcal{B}_{U,V}$: binarize both. We only report results within 3 days.

along the GNN-based methods, our goal is to develop an effective and scalable BGE solution by addressing the following challenges:
Challenge I: *How to efficiently capture local structure with limited resources?* For a bipartite graph, all edges only exist between two disjoint node sets U and V, known as "one-hop structure", and the learning objective of existing GNN-based methods is mainly to preserve the one-hop structure. Meanwhile, they rely on neighborhood-based message passing to strengthen the representation of neighboring nodes, thus achieving success in local structure learning [21, 34, 46]. However, such message passing heavily relies on the adjacency matrix, leading to expensive storage and computation costs linear to the number of edges in the graph [21]. This makes these methods not scalable to large graphs, as demonstrated in Figure 1b and Section 5.3. These two observations inspire us to efficiently maintain the important one-hop structure for BGE learning.
Challenge II: *How to better model global-local learning?* We observe that although existing global-local learning methods process U and V separately, they still follow the same manner. For example, SHT [39] constructs U-hyperedge and V-hyperedge propagations using the same routine. To explore roles of U and V, we create three binarization variants ($\mathcal{B}_U$, $\mathcal{B}_V$ and $\mathcal{B}_{U,V}$) based on original float-precision embeddings, using the binarization function $\texttt{sign}(\cdot) : \mathbb{R}^d \rightarrow \{-1, 1\}^d$. By comparing performance under $\mathcal{B}_U$ and $\mathcal{B}_V$ in Figure 1, we observe that U and V contribute differently to BGE performance for almost all BGE methods. This indicates that the modeling choice of existing global-local learning methods may be unsuitable. Second, these methods learn global information combined with the local neighborhood aggregation, leading to inferior scalability compared to those local learning methods (e.g., LightGCN), as confirmed in Section 5.3. Hence, effective global-local learning is essential for large-scale graphs.
Challenge III: *How to alleviate the over-smoothing problem?* Existing solutions typically follow the path of the bipartite graph to generate BGE. Nevertheless, this makes them suffer from the over-smoothing problem [37]: node representations tend to be indistinguishable within neighborhood, thus compromising the embedding quality. By comparing the performance under original embeddings and $\mathcal{B}_{U,V}$ in Figure 1, we observe that existing methods show a dramatic performance decrease under $\mathcal{B}_{U,V}$ (especially GNN-based ones, with an average drop of 89% accuracy). This is because both GNN-based and metric-based methods excessively rely on paths of bipartite graphs, which leads to indistinguishable node embeddings.

To address the above three challenges, in this paper, we propose a novel Graph Neural Network with a new anchor-based message passing schema, called AnchorGNN, for bipartite graph embedding. To capture global information, we construct effective node-anchor propagations in our anchor-based message passing, in which anchor nodes with global knowledge are directly learned and optimized during the training process (**Challenge II**). Regarding that edges on the bipartite graph only exist between two types of node sets, we introduce a novel one-hop local structure modeling: learning one-hop proximity between two node sets using maximum likelihood estimation, which can avoid the usage of $O(|E|)$ adjacency matrix (**Challenge I**). Then, global and local learning are integrated to learn high-quality node embeddings and effectively alleviate the over-smoothing problem, as we see the better and more stable performance on AnchorGNN and its variants in Figure 1 (**Challenge III**). Furthermore, we optimize the loss function with a negative sampling strategy, enabling AnchorGNN to process billion-scale graphs with limited resources (e.g., single GPU), while existing GNN-based methods run out of memory.

The main contributions are summarized below:

- We propose a novel global-local induced AnchorGNN for BGE, which significantly advances the state of the art in both embedding quality and model scalability.
- We propose a new anchor-based message passing schema to explore global learning, which helps generate distinguishable node embeddings in the global context as well as speeds up model convergence.
- We offer efficient local structure modelling using maximum likelihood estimation underlying the cross-entropy loss. To our best knowledge, we rationally analyze its advantages for bipartite graph learning for the first time.
- Extensive experiments demonstrate that our AnchorGNN outperforms baselines by a large margin in accuracy and is significantly faster than the only metric-based baseline on billion-scale bipartite graphs.

2 PROBLEM STATEMENT

2.1 Preliminaries

Let $G = (U, V, E)$ be a bipartite graph, where U and V denote disjoint node sets for two types of nodes. The nodes in U and V are called source nodes and destination nodes, respectively. $E \subseteq U \times V$ denotes the edge set. $|U|$ and $|V|$ are the number of nodes in U and V, respectively, and $|E|$ is the number of edges in E. The set of neighbors of a node $x \in U \cup V$ is denoted by $\mathcal{N}(x)$. We have $\mathcal{N}(u) \subseteq V$ and $\mathcal{N}(v) \subseteq U$ for $u \in U$ and $v \in V$. We denote matrices in bold uppercase (e.g. $\mathbf{U}$) and denote vectors in bold lowercase (e.g. $\mathbf{x}$). The κ-th element of vector $\mathbf{x}$ is denoted by $\mathbf{x}[\kappa]$. Following convention [16, 45], we consider undirected bipartite graphs here.

2.2 Problem Definition

Bipartite graph embedding aims to map each node $u_i \in U$ and each node $v_j \in V$ into the d-dimensional vectors $\mathbf{u}_i$ and $\mathbf{v}_j$, respectively.

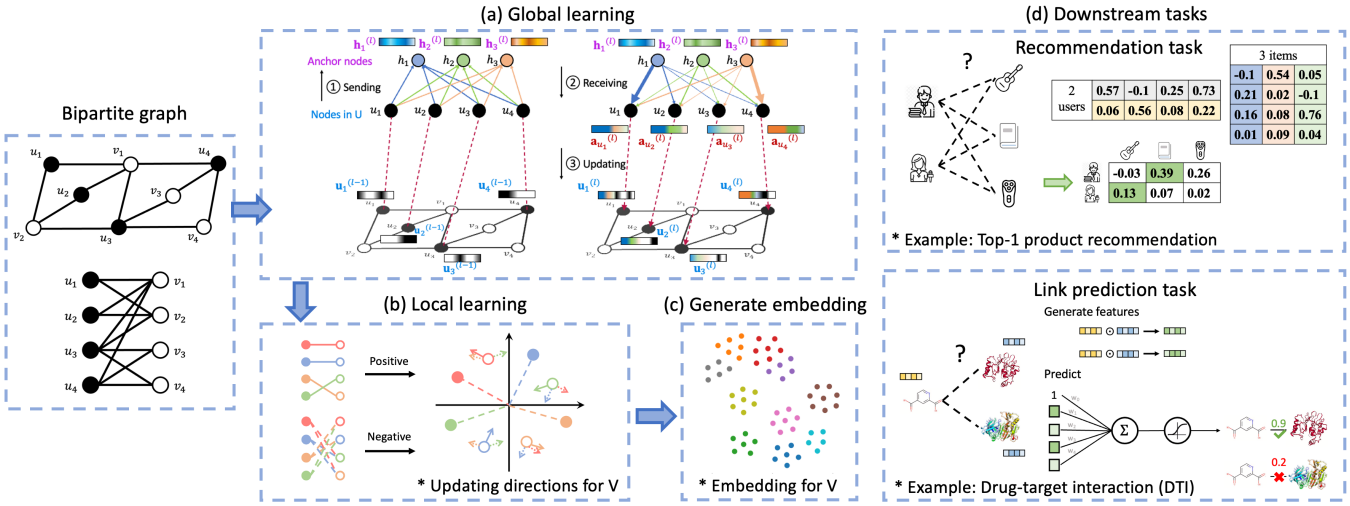


Figure 2: Overview of AnchorGNN and its downstream tasks. (a) Anchor-based message passing. (b) Local learning in partial-structure mode. (c) Visualization of hit set in top-20 recommendation on Yelp using t-SNE [32]. Different colors denote different users. (d) Applications of product recommendation and drug discovery for two downstream tasks, respectively.

We denote matrices $\mathbf{U} \in \mathbb{R}^{|U|\times d}$ and $\mathbf{V} \in \mathbb{R}^{|V|\times d}$ as the embedding vectors of all nodes in U and V, respectively.

Problem definition. Given an unweighted bipartite graph $G = (U, V, E)$, BGE is to learn a map function $f : U \cup V \rightarrow \mathbb{R}^d$, where each node in G is mapped to a d-dimensional embedding vector, e.g., $\mathbf{u}_i \in \mathbb{R}^d = f(u_i)$.

3 THE PROPOSED ANCHORGNN

Here, we present an overview of our AnchorGNN model (i.e., $f(\cdot)$ function) in Figure 2. Concretely, we first present how to explore global learning via a new anchor-based message passing strategy, and then introduce local learning modelling with our loss function, followed by the rational analysis of the proposed approach.

3.1 Global Learning by Anchor-Based MP

To address the poor scalability of existing global learning methods, this study proposes a novel anchor-based message passing schema to model global learning, where we use global shared anchor nodes for node-anchor propagations. First, we introduce anchor nodes.

3.1.1 Anchor nodes. In AnchorGNN, the anchor node serves as the intermediate hub among nodes of bipartite graphs, and we give the definition of the anchor node below.

Definition 1 (Anchor node). *The anchor node is the learned virtual node with distilled global knowledge from the bipartite graph, which is designed for global information sharing among source nodes. During the message passing process, each anchor node is assumed to be connected with all source nodes.*

Definition 2 (Anchor node set). *The anchor node set is defined by $H = \{h_1, h_2, \cdots, h_{|H|}\}$, where we learn the low-dimensional embedding $\mathbf{h}_j \in \mathbb{R}^m$ for each anchor node $h_j \in H$ during the training process. The number of anchor nodes $|H|$ and the dimensionality of anchor node embedding m are both hyperparameters.*

Generally, global information sharing is conducted for two types of nodes using the same message passing schema [12, 38, 39]. However, global information sharing in our anchor-based MP for either U or V would make all node representations affected by anchor nodes owing to our global-local learning framework. Here, our anchor-based MP is conducted only on source nodes in U and we leave more detailed explanations in Section 3.3. Next, we formulate our proposed anchor-based message passing, where its each layer involves three steps: *sending*, *receiving* and *updating* (Figure 2a).

3.1.2 Sending. Given a source node $u \in U$, a message is constructed at the l-th layer based on its current representation $\mathbf{u}^{(l-1)}$. The node sends messages to all anchor nodes using a parameterized function, which is defined as

$$\mathbf{s}_u^{(l)} = \text{SEND}(\mathbf{u}^{(l-1)}). \tag{1}$$

Here SEND$(\cdot)$ is the multilayer perceptron (MLP), which is employed for building virtual edges between the node u and all anchor nodes in H, as shown in Figure 2a-①. These edges can be regarded as a latent space projection from graph nodes to anchor nodes. $\mathbf{s}_u^{(l)} \in \mathbb{R}^m$ is then utilized in the receiving step.

3.1.3 Receiving. With constructed edges, we can aggregate the global information from anchor nodes to the node in U in this step. Suppose we employ aggregators of the neighborhood-based MP, such as *sum*-, *mean*- and *max*-pooling methods [21, 42], the output of aggregators (e.g., sum$(\{\mathbf{h}_j^{(l)} | h_j \in H\})$) is a constant vector for each node, which fails to provide the distinguishable knowledge for node representations. Inspired by the interaction estimation in [22, 43], we employ the relevance score to estimate the interaction between the node u and each anchor node h_j, and formulate it by

$$\mathbf{r}_u^{(l)} = \Big\Vert_{j=1}^{|H|} \text{ATTENTION}(\mathbf{s}_u^{(l)}, \mathbf{h}_j^{(l)}), \tag{2}$$

where we employ the dot-product attention in [33]. We concatenate (i.e., "||") these scores to build interaction messages $\mathbf{r}_u^{(l)} \in \mathbb{R}^{|H|}$, so that learned global knowledge is leveraged into global interactions. Then, each node u will receive messages from all anchor nodes, and we normalize and aggregate these messages by

$$\mathbf{a}_u^{(l)} = \text{RECEIVE}(\text{LN}(\mathbf{r}_u^{(l)})), \tag{3}$$

where receiving messages for each node u are represented as $\mathbf{a}_u^{(l)} \in \mathbb{R}^d$. RECEIVE$(\cdot)$ function is the neural networks, e.g., MLP. LN$(\cdot)$ is the layer normalization [10] to guarantee a fair message receiving. By Eqs. (2)-(3), each node u receives information equally from interactions between itself and all anchor nodes. The receiving step is shown in Figure 2a-②.

3.1.4 Updating. Last, we combine the node's current representation with aggregated messages to update it by

$$\mathbf{u}^{(l)} = \text{UPDATE}(\mathbf{u}^{(l-1)}, \mathbf{a}_u^{(l)}) = \mathbf{u}^{(l-1)} + \sin(\mathbf{a}_u^{(l)}), \tag{4}$$

where we employ $\sin(\cdot)$ as the activation function and update a node's representation by element-wise summation. At the L-th layer, we have an embedding vector $\mathbf{u} = \mathbf{u}^{(L)}$ for node u as a result. Upon three steps, our MP route follows $u \to h \to u$ route instead of the path of $u \to v \to \cdots \to u$, avoiding multiple neighborhood dependencies. Each anchor node is learned and optimized based on the input graph during the training process, thus preserving global context. Correspondingly, global information is leveraged to help generate distinguishable node representation for each u based on the node-anchor propagations, as shown in Figure 2a.

3.2 Local Learning by Loss Function

Upon our anchor-based MP, each source $u \in U$ obtains global information and produces the embedding $\mathbf{u}$, but the one-hop $U-V$ structure constraint is not guaranteed. However, the commonly-used Bayesian personalized ranking loss [29] in existing GNN-based methods [21, 27, 34, 35, 38, 39, 47] shows inferior capabilities for structure learning as shown in experiments of [34]. Hence, we point out a new modelling direction: based on only two types of nodes of the bipartite graph, we propose to enforce constraints on node types to capture one-hop local structure information.

Concretely, we formulate one-hop structure learning as a maximum likelihood optimization problem, i.e., max $\prod_{u \in U} Pr(\mathcal{N}(u)|u)$, where $Pr(\mathcal{N}(u)|u) = \prod_{v \in \mathcal{N}(u)} Pr(v|u)$, and $Pr(v|u)$ denotes the probability of nodes u and v forming an edge when given u. Hence, we can learn one-hop structure directly on the edge set without adjacency matrix construction. We estimate the probability $Pr(v_j|u_i)$ for edge (u_i, v_j) via the softmax function:

$$Pr(v_j|u_i) = \frac{\exp(\mathbf{u}_i^T \mathbf{v}_j)}{\sum_{v_k \in N_S(u_i) \cup \{v_j\}} \exp(\mathbf{u}_i^T \mathbf{v}_k)}, \tag{5}$$

where $N_S(u_i) \cup \{v_j\}$ is the set of candidate neighboring nodes for u_i, $N_S(u_i) \subseteq V$ is the set of negative samples for u_i under training mode S, which will be introduced in Section 4.1. Then we construct the cross-entropy loss $\mathcal{L}_{CE}$ [11] to maximize log probabilities for one-hop structure learning: $\mathcal{L}_{CE} = -\sum_{(u_i,v_j) \in E} \log(Pr(v_j|u_i))$. By substituting $Pr(v_j|u_i)$ with Eq. (5), we reformulate $\mathcal{L}_{CE}$ as

$$\mathcal{L}_{CE} = \sum\nolimits_{(u_i,v_j) \in E} -\mathbf{u}_i^T \mathbf{v}_j + \log(\sum\nolimits_{v_k \in N_S(u_i) \cup \{v_j\}} \exp(\mathbf{u}_i^T \mathbf{v}_k)). \tag{6}$$

To prevent overfitting, we use the regularization technique and obtain the final optimization objective:

$$\min \quad \mathcal{L}_{CE} + \lambda \|\Theta\|^2, \tag{7}$$

where λ is the hyperparameter. To demonstrate the rationality behind our design, we construct an in-depth analysis below.

3.3 Analysis on Loss Function

Although the cross-entropy loss is widely used in the classification task [15, 25, 28], we are the first to analyze its advantages over bipartite graph structure learning in detail. The advantages lie in three aspects:

(1) One-hop local structure learning. For the bipartite graph, two node sets U and V have no internal edge and the edge set is $E \subseteq U \times V$. Given this property, a practical approach to model one-hop structure for node $u \in U$ is to maximize $\prod_{v \in \mathcal{N}(u)} Pr(v|u)$ for all v in neighboring set $\mathcal{N}(u) \subseteq V$. Thus, $\mathcal{L}_{CE}$ can constrain the U-V proximity via $Pr(v|u)$, leading to the embeddings of v and v' to be closer or clustered together when they are connected to the same u, as demonstrated in Figure 2c. Additionally, the softmax of $\mathcal{L}_{CE}$ in Eq. (5) ensures that the sum of output probability is 1 given any u_i, thus each node is treated equally regardless of their degrees. Furthermore, since we directly estimate the probability distribution $P(V|u)$ for one-hop local structure learning, where $P(V|u) = [Pr(v_1|u); Pr(v_2|u); ...]^T \in \mathbb{R}^{|V|}$, we cut the $O(|E|)$ storage cost of the adjacency matrix. The above three properties ensure the effective learning of the one-hop graph structure on the bipartite graph with limited resources.

(2) Adaptation for downstream tasks. An undirected bipartite graph can be seen as the combination of two directed subgraphs, thus we can capture the one-hop structure by modelling $P(V|u)$ and $P(U|v)$. The reason why we model $P(V|u)$ rather than $P(U|v)$ is that U and V play different roles for some downstream tasks like recommendation [19] (see Figure 1), where we evaluate the performance by ranking $\mathbf{u}_i^T \mathbf{v}_j$ for all $v_j \in V$ when given u_i. In such case, modelling $P(V|u)$ is more suitable than modelling $P(U|v)$. Meanwhile, this modelling choice also shows the effectiveness for the link prediction in Section 5.2.

(3) Dependency of V on U. We take batch size = 1 for demonstration. During the gradient computation, given a single u_i, the partial derivative for the κ-th element of $\mathbf{v}_j$ for candidate v_j is $\frac{\partial \mathcal{L}_{CE}}{\partial \mathbf{v}_j[\kappa]} = \frac{\partial \mathcal{L}_{CE}}{\partial a_j} \frac{\partial a_j}{\partial z_j} \frac{\partial z_j}{\partial \mathbf{v}_j[\kappa]} = (a_j - y_t) \cdot \mathbf{u}_i[\kappa]$, where a_j is the probability calculated by softmax in Eq. (5), $z_j = \mathbf{u}_i^T \mathbf{v}_j$, and y_t is the 0/1 label for node pair (u_i, v_j), i.e., $y_t = 1$ if $(u_i, v_j) \in E$, otherwise $y_t = 0$. Since $0 < a_j < 1$ for the probability a_j, $\frac{\partial \mathcal{L}_{CE}}{\partial \mathbf{v}_j[\kappa]}$ has an opposite sign to $\mathbf{u}_i[\kappa]$ for $y_t = 1$, and has the same sign to $\mathbf{u}_i[\kappa]$ for $y_t = 0$. In other words, the gradient/updating direction of $\mathbf{v}_j$ depends on $\mathbf{u}_i$ with the guarantee of $\mathcal{L}_{CE}$ (see Figure 2b). The dependency of $\mathbf{V}$ on $\mathbf{U}$ allows us to focus on anchor-based MP only among U while ensuring global learning for V. The effectiveness of our modelling approach is validated by experiments in Section 5.5.

3.4 Anchor-based vs. Neighborhood-based MP

We compare anchor-based MP against neighborhood-based MP in LightGCN [21], a simple but SOTA BGE solution, from three views. Note that we set $m < d$, $|H| < d$, and $L = 1$. **(1) Time complexity of MP.** According to Eqs. (1)-(4), the time complexity of anchor-based MP in a mini-batch manner is $O(B \cdot (md + m|H| + |H|d))$, bounded by $O(B \cdot d^2)$, where B denotes batch size. Its time complexity depends only on hyperparameters rather than $|E|$, and provides more feasibility and efficiency over LightGCN's neighborhood-based MP ($O(|E|Ld)$ per batch) when $|E|$ is large. **(2) Storage cost of MP.** Anchor-based MP route does not follow the path of the bipartite graph, thus its overall storage is $O(md + m|H| + |H|d)$, bounded by $O(d^2)$, while neighborhood-based MP requires $O(|E|)$ additional storage for adjacency matrix. **(3) Distinguishability.** Neighborhood-based MP solutions' reliance on paths makes them vulnerable to the over-smoothing problem. In contrast, the combination of node-anchor propagations and our proposed local learning helps learn distinguishable embedding regardless of the path.

4 OPTIMIZATION & COMPLEXITY ANALYSIS

4.1 Optimization for Large Bipartite Graphs

We design two training modes S including *full-structure mode* (by default) and *partial-structure mode* (for billion-scale graphs). In full-structure mode, we adopt $N_S(\cdot) = V$ in Eq. (6), which can learn one-hop structure thoroughly. However, it would take $O(B|V|d)$ to compute $\mathcal{L}_{CE}$, which is costly on large-scale graphs. Hence, we introduce a partial-structure mode with the negative sampling strategy to optimize the training process. Concretely, we randomly sample $|N|$ nodes from V for each u in training edges from uniform distribution [22, 29] to form the negative sample set $N_S(u)$, where $|N|$ is a hyperparameter. In this mode, our local structure learning in Eq. (6) is optimized by partial structure, i.e., one positive sample and $|N|$ negative samples in Figure 2b. This mode enables AnchorGNN to process billion-scale bipartite graphs with decent performance and competitive training speed, which will be discussed in Section 5.4.

4.2 Complexity Analysis of AnchorGNN

4.2.1 Time complexity. AnchorGNN's time complexity is mainly related to that of $\mathcal{L}_{CE}$ and anchor-based MP. On general graphs, $\mathcal{L}_{CE}$ takes $O(B|V|d)$. The overall time complexity is $O(B|V|d + B \cdot (md + m|H| + |H|d))$, which can be reduced to $O(B|V|d)$ when $d < |V|$, as the second part (MP) is bounded by $O(B \cdot d^2)$ (Section 3.4). On billion-scale graphs, we use negative sampling, thus $\mathcal{L}_{CE}$ takes $O(B|N|d)$. The overall time complexity is $O(B \cdot (|N|d + md + m|H| + |H|d))$, which is bounded by $O(B \cdot d^2)$ when $|N| < d$.

4.2.2 Space complexity. The overall space complexity consists of the storage of parameters for U and V, and additional parameters in MP. It takes $O((|U| + |V|) \cdot d + (md + m|H| + |H|d))$ storage, which can be reduced to $O((|U| + |V|) \cdot d)$ when $d < |U| + |V|$.

5 EXPERIMENTAL STUDY

We evaluate our AnchorGNN against 8 competitors on 10 real-world bipartite graphs on two fundamental tasks (i.e., top-K recommendation and link prediction). All experiments are conducted on a Linux server with an Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz CPU, 251G RAM and a GeForce RTX 3090 GPU (24GB).

Table 1: Statistics of Datasets.

Alias	Name	$\|U\|$	$\|V\|$	$\|E\|$
WK	Wikipedia	15,000	3,214	64,095
PT	Pinterest	55,187	9,916	1,480,995
YP	Yelp	31,668	38,048	1,561,406
AB	Amazon-Book	52,643	91,599	2,984,108
ML	MovieLens	69,878	9,708	9,995,471
LF	Last.fm	358,680	63,958	17,262,164
MD	MIND	876,956	97,509	18,149,915
NF	Netflix	463,770	17,768	100,396,376
OK	Orkut	2,783,196	8,730,857	327,037,487
MG	MAG	10,539,041	1,302,979	1,087,329,592

5.1 Experimental Settings

5.1.1 Datasets. Table 1 summarizes the statistics of ten public datasets used in our experiments. Here, we collect Yelp and Amazon-Book from [16], and the remaining datasets from [45]. Eight of them are general graphs, and Orkut and MAG are two large-scale graphs.

5.1.2 Implementation and Setting. We implement our model using the PyTorch framework. For a fair comparison, we set the embedding dimensionality $d = 64$ for all competitors. The model parameters are initialized with the Xavier method [17] and the normalization trick [10] is used during the learning process. We train AnchorGNN with Adam optimizer [23], with a learning rate of 0.0002. We set the network layer $L = 1$, the batch size $B = 1000$, the number of anchor nodes $|H| = 16$, and the embedding dimensionality of anchor nodes $m = 8$ for all datasets. Note that AnchorGNN is not sensitive under these configurations. We employ negative sampling for the two largest datasets (Orkut and MAG), with the number of samples $|N| = 10$ for each training edge, which is an affordable cost to train such large graphs. For all datasets, we perform a grid search for the L2 regularization coefficient $\lambda \in [0, 0.005]$.

5.1.3 Baseline Approaches. We compare our AnchorGNN against 8 baseline methods covering 3 groups: ① GNN-based methods: LightGCN [21], SHT [39], HCCF [38] and BiGI [12]; ② metric-based methods: GEBE^p [45] and BiNE [16]; ③ homogeneous network embedding (HONE) methods: LINE [30] and node2vec [18]. We use the open-source implementation of baselines under their default settings [1–8]. Note that we set the batch size as 10,000 on Netflix for GNN-based methods, ensuring feasible methods could generate results within three days. If one method fails to terminate within three days or runs OOM on one dataset, we do not report it.

5.2 Embedding Quality

We evaluate the embedding quality by two fundamental downstream tasks: top-K recommendation and link prediction.

5.2.1 Top-K Recommendation. Top-K recommendation is a personalized ranking task, which is widely used for existing BGE studies [16, 21, 45]. We use five bipartite graphs, split them with $8:2$ for training and testing under the 10-core setting [20]. The remaining

Table 2: Top-20 recommendation performance. The best result is highlighted in bold, and the second best is underlined.

Method	Yelp		MovieLens		Last.fm		Netflix		MAG	
	RECALL	NDCG	RECALL	NDCG	RECALL	NDCG	RECALL	NDCG	RECALL	NDCG
AnchorGNN	**0.070**	**0.058**	**0.347**	**0.437**	**0.279**	**0.268**	**0.217**	**0.361**	**0.179**	**0.286**
SHT	0.045	0.037	0.253	0.315	0.152	0.138	-	-	-	-
HCCF	0.035	0.029	0.195	0.250	0.066	0.073	-	-	-	-
LightGCN	0.060	0.049	0.308	0.391	0.232	0.220	0.159	0.291	-	-
BiGI	0.001	0.001	0.121	0.129	-	-	-	-	-	-
GEBE^p	0.041	0.035	0.199	0.263	0.119	0.115	0.121	0.214	0.174	0.278
BiNE	0.012	0.009	-	-	-	-	-	-	-	-
node2vec	0.020	0.016	-	-	-	-	-	-	-	-
LINE	0.009	0.007	0.097	0.138	0.037	0.036	0.046	0.083	-	-

Table 3: Link prediction performance. The best result is highlighted in bold, and the second best is underlined.

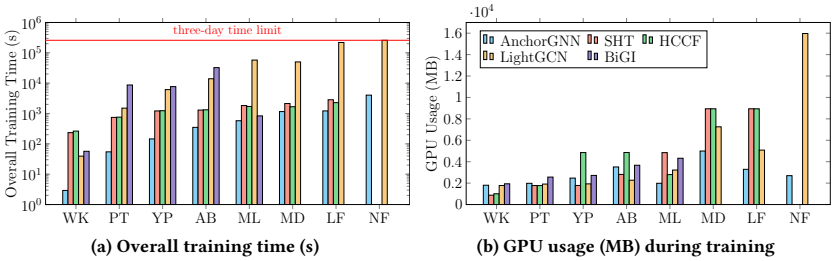
Method	Wikipedia		Pinterest		Amazon-Book		MIND		Orkut	
	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
AnchorGNN	**0.928**	**0.938**	**0.965**	**0.959**	0.954	0.955	**0.977**	**0.974**	**0.877**	**0.912**
SHT	0.862	0.899	0.952	0.937	0.946	0.948	0.961	0.960	-	-
HCCF	0.889	0.916	0.917	0.895	0.901	0.900	0.941	0.939	-	-
LightGCN	0.862	0.892	0.963	0.953	**0.956**	**0.959**	0.955	0.965	-	-
BiGI	0.920	0.930	0.781	0.741	0.834	0.800	-	-	-	-
GEBE^p	0.824	0.871	0.943	0.939	0.908	0.919	0.916	0.925	0.863	0.893
BiNE	0.807	0.862	0.688	0.660	0.755	0.776	-	-	-	-
node2vec	0.657	0.607	0.940	0.927	0.925	0.919	-	-	-	-
LINE	0.743	0.790	0.798	0.777	0.662	0.720	0.876	0.882	0.799	0.870

task setting follows the baselines [21, 39, 45], and we adopt two popular evaluation metrics: RECALL@K and Normalized Discounted Culmulative Gain (NDCG@K) (the higher the better). We set $K = 20$ and observe from Table 2 that our AnchorGNN significantly outperforms eight baselines across all evaluation metrics on all datasets. In almost all cases, GNN-based methods yield superior performance than metric-based methods and HONE methods, demonstrating that advanced deep learning techniques can help generate higher-quality bipartite graph embedding. Our AnchorGNN beats the best competitor LightGCN (local learning method) on general graphs by a large margin, up to 36% in RECALL and 24% in NDCG. Furthermore, AnchorGNN achieves better performance than existing global-local methods (i.e., SHT, HCCF and BiGI). These validate the effectiveness of our proposed global-local learning framework. For the billion-scale dataset MAG, all existing GNN-based methods run GPU OOM. In contrast, AnchorGNN achieves 0.179 in RECALL and 0.286 in NDCG, which is averaged 2.8% higher than the only competitor (metric-based) GEBE^p that can finish in three days.

5.2.2 Link Prediction. Link prediction aims to predict whether an edge exists between two given nodes $u_i \in U$ and $v_j \in V$. We use five bipartite graphs and set the training and testing ratio as $6:4$. We use the Hadamard product to build the feature vector for each node pair and conduct experiments following [16, 18, 45]. We evaluate link prediction in terms of two metrics: the area under the ROC curve (AUC-ROC) and the Precision-Recall curve (AUC-PR) (the higher the better). It is observed from Table 3 that AnchorGNN consistently outperforms baselines in almost all cases, except Amazon-Book dataset. Concretely, on MIND, our AnchorGNN achieves 1.6% AUC-ROC and 0.9% AUC-PR improvements compared to the best competitor. For the large-scale bipartite graph Orkut, our AnchorGNN yields 0.877 in AUC-ROC and 0.912 in AUC-PR, outperforming the best competitor by a significant margin of up to 1.6% in AUC-ROC and 2.1% in AUC-PR. To sum up, AnchorGNN can generate higher-quality embeddings, which facilitates the performance of downstream tasks. Both tasks endorse the effectiveness of our proposed techniques in global learning and local learning, especially for large-scale bipartite graph learning.

5.3 Resource Consumption of GNN-based BGE

We assess the resource consumption of GNN-based BGE methods on 8 general bipartite graphs in terms of overall training time and GPU memory usage. For a fair comparison, we exclusively consider baselines that utilize GPU training. Observe from Figure 3 that (1) In terms of training time in Figure 3a, our AnchorGNN is faster than all GNN-based methods, achieving $1.4-60$ times speed-up. (2) AnchorGNN may require more GPU memory on smaller datasets (e.g., Wikipedia (WK)) compared to GNN-based methods when the graph size is not significantly larger than the number of nodes. This is because our GPU memory is related to the number of nodes, while the GPU memory of GNN-based methods is related to the graph size $|E|$. However, the memory usage of AnchorGNN is significantly lower than other methods as the graph size increases (see Figure 3b). Specifically, AnchorGNN only requires 2697MB of GPU memory on Netflex (NF), which is 83% cheaper than LightGCN's 15969MB.



(a) Overall training time (s) (b) GPU usage (MB) during training

Figure 3: Overall training time & GPU usage on 8 general graphs.

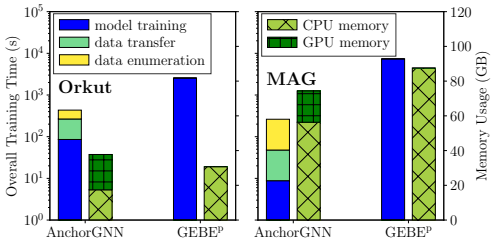
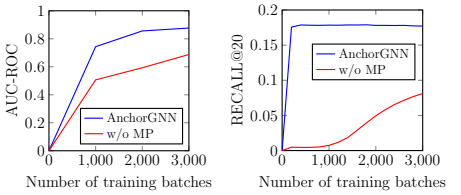
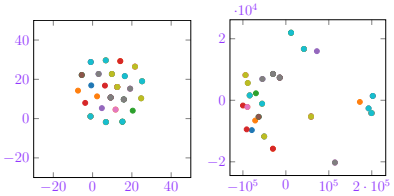


Figure 4: Resource usage on large graphs.



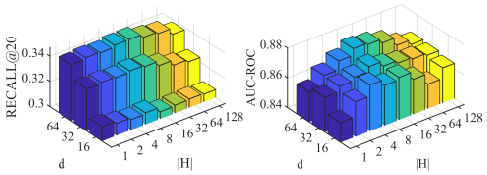
(a) Orkut (b) MAG

Figure 5: Training curves.



(a) AnchorGNN (b) GEBE^p

Figure 6: Distribution of V on MAG.



(a) MovieLens (b) Orkut

Figure 7: Hyper-parameter studies.

(3) Although SHT and HCCF combine global learning with edge sampling strategy to improve efficiency on small datasets, they fail to handle larger graphs (e.g., Netflex (NF)) due to complex global-local learning and combinations of multiple self-augmentation techniques. In summary, AnchorGNN exhibits superior scalability and efficiency compared to other GNN-based methods.

5.4 Scalability

We evaluate the scalability of AnchorGNN against the only competitor (metric-based) GEBE^p on two billion-scale bipartite graphs. Observe from Figure 4 and Tables 2-3 that AnchorGNN outperforms GEBE^p in terms of time cost, memory usage, and effectiveness in almost all cases. Next, we present analyses from three perspectives.

(1) Time cost. According to the overall training time in Figure 4, AnchorGNN takes 434s and 261s on Orkut and MAG, respectively, achieving 5× and 28× speed-up compared to GEBE^p, respectively. Our advantage in efficiency is largely attributed to the fast convergence. We create a variant: AnchorGNN without anchor-based MP ("w/o MP") and show their training curves on Orkut and MAG in Figure 5. Observe that AnchorGNN converges faster and achieves better performance from scratch compared to its variant, indicating that the global information provided by anchor-based MP leads to a strong generalization capability. For example, AnchorGNN can yield SOTA performance within a small number of training batches, e.g., obtaining 0.176 in RECALL at batch 200 on MAG. This suggests that it is less dependent on full-edge training on large graphs. In contrast, GEBE^p has to factorize the edge weight matrix for the entire graph, which is time-consuming on large graphs.

(2) Memory usage. The memory usages of both methods mainly depend on data loading and model training. For data loading, their memory usages are approximately the same, where both use CPU memory. For model training, AnchorGNN takes GPU resources, but GEBE^p only costs CPU resources since the input matrix of GEBE^p takes $O(|E|)$ storage and cannot be loaded into GPU. In Figure 4, AnchorGNN's GPU usage on the two largest graphs is similar since its GPU usage is related to the number of nodes. In contrast, GEBE^p applies matrix operations to the edge matrix, resulting in higher memory usage on larger graphs like MAG. In a word, AnchorGNN exhibits more advantages than GEBE^p on larger graphs.

(3) Effectiveness. In Tables 2-3, AnchorGNN yields slightly better performance than GEBE^p on two largest graphs. The underlying reason lies in that hard negatives might not be sampled during sampling process, which can introduce less informative training samples. However, our AnchorGNN still achieves competitive performance in the partial-structure mode. To further demonstrate the effectiveness of AnchorGNN, we choose 10 users on MAG and visualize the representations of items that hit ground truths in the test set in Figure 6 (same setting as Figure 2c). For AnchorGNN, items with close representations gather for a user group (Figure 6a). In contrast, GEBE^p represents items in a long distance (as highlighted in its axes with a large span: $10^5 \times 10^4$ in Figure 6b), making them less likely to be recommended to the same user.

Overall, global-local induced AnchorGNN provides a feasible, scalable and effective solution for BGE on large graphs.

5.5 Investigation of AnchorGNN

Ablation Study. To evaluate the effectiveness of our proposed global learning and local learning components, we construct a variant without MP ("w/o MP", i.e., the only local learning component) and compare the accuracy and overall training time on 10 datasets on two tasks. Observe from Table 4 that (1) Our variant with merely local learning outperforms LightGCN (SOTA among local learning methods) in most cases, which confirms the effectiveness of one-hop structure learning of bipartite graphs for downstream tasks and is consistent with our analysis in Section 3.3. (2) Our anchor-based MP can significantly boost the performance of BGE, with up to 9.3%

Table 4: Ablation study of AnchorGNN in terms of accuracy and overall training time. The best result is highlighted in bold.

Top-20 Recommendation. (R: RECALL, N: NDCG, time: overall training time (s).)															
Method	**Yelp**			**MovieLens**			**Last.fm**			**Netflix**			**MAG**		
	R	N	time	R	N	time	R	N	time	R	N	time	R	N	time
AnchorGNN	**0.070**	**0.058**	**146**	**0.347**	**0.437**	579	**0.279**	**0.268**	**1219**	**0.217**	**0.361**	4038	**0.179**	0.286	**261**
w/o MP	0.065	0.053	248	0.332	0.414	**376**	0.258	0.245	1807	0.210	0.351	**2859**	**0.179**	**0.288**	37452

Link Prediction. (ROC: AUC-ROC, PR: AUC-PR, time: overall training time (s).)															
Method	**Wikipedia**			**Pinterest**			**Amazon-Book**			**MIND**			**Orkut**		
	ROC	PR	time	ROC	PR	time	ROC	PR	time	ROC	PR	time	ROC	PR	time
AnchorGNN	**0.928**	**0.938**	**2.9**	**0.965**	**0.959**	**54.6**	**0.954**	**0.955**	**350**	**0.977**	**0.974**	**1158**	**0.877**	**0.912**	**434**
w/o MP	0.904	0.922	5.1	0.956	0.947	69.8	0.945	0.941	779	0.973	0.970	2209	0.800	0.876	2886

improvement in recommendation and up to 9.6% improvement in link prediction, demonstrating that global information contributes to higher-quality node embeddings. (3) As introduced in Section 5.4, our anchor-based MP can speed up model convergence under the guide of global information. Consequently, AnchorGNN reduces the overall training time compared to its variant in most cases, achieving up to 143 times speed-up on MAG. On the other hand, AnchorGNN requires slightly more time to achieve a better performance than its variant on relatively dense graphs (MovieLens and Netflix). This suggests the advantage of our AnchorGNN is more pronounced on sparse bipartite graphs.

Hyper-parameter sensitivity. We evaluate the sensitivity of the number of anchor nodes $|H|$ and the embedding dimension d, where $|H| \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ and $d \in \{16, 32, 64\}$. Figure 7 shows the results on MovieLens and Orkut.

Sensitivity of $|H|$. AnchorGNN can achieve stable and decent performance for $|H| \in \{8, 16\}$ in most cases, but large $|H|$ can make the model overfit. Besides, adding a single anchor node also yields good performance, indicating the significance of global learning. In our experiments, we empirically set $|H| = 16$.

Sensitivity of d. A larger d leads to better performance in most cases since the embedding dimension determines the expressiveness of AnchorGNN. Throughout our experiments, we set $d = 64$.

6 RELATED WORK

Metric-based Bipartite Graph Embedding. The metric-based methods generate node embeddings by defining similarity and proximity metrics over approximate paths. For instance, BiNE [16] conducts biased random walks to generate vertex sequences as the approximate path context and learns embeddings from both edges and paths. This approach incurs significant overhead due to numerous random walks. GEBE^p [45] learns to preserve multi-hop similarity and proximity based on the assigned path importance via a probability mass function and then optimizes the objective function by the eigen-decomposition, thus it can process billion-scale bipartite graphs. Nevertheless, the reliance on approximate paths makes their embedding quality sensitive to predefined measures.

GNN-based Bipartite Graph Embedding. For bipartite graph modelling, another promising approach is to use deep learning techniques such as graph convolutional neural networks [21, 34, 48], transformers [14, 31, 39], and self-augmented learning [35, 38, 39, 41], to improve BGE quality, where most of the research in this line focuses on the recommendation task. Specifically, NGCF [34] integrates the user-item interaction into the propagation process, thus capturing the collaborative signal and high-order connectivity. LightGCN [21] removes the feature transformation and nonlinear activation during message passing and achieves SOTA performance. Based on LightGCN, SGL [35], SimGCL [47] and SHT [39] explore different self-supervised learning techniques to improve the model robustness or generalization capability. For example, SHT [39] proposes a self-supervised hypergraph transformer model to explore global collaborative effects among users and items, where global information is constructed depending on local learning by two-layer LightGCN. In addition, HCCF [38] develops a hypergraph-enhanced cross-view contrastive learning framework that iteratively performs local neighborhood aggregation and global message propagation, which can capture the intrinsic relations among users and items. BiGI [12] produces BGE by maximizing the mutual information between nodes' local and global representations, where initial node embeddings are first learned by the GCN. However, existing GNN-based methods fail to process billion-scale graphs due to expensive neighborhood-based MP and more complex modelling choices, as demonstrated in recent studies [45] and our experiments.

7 CONCLUSION

In this paper, we propose a novel AnchorGNN induced by global-local learning for large-scale bipartite graph embedding. Concretely, we propose a new anchor-based message passing schema to capture global information not depending on local aggregation, which can provide distinguishable messages for each node and speed up model convergence. Underlying the bipartite graph property, we leverage maximum likelihood estimation to model one-hop local structure learning, where our local learning not only avoids the memory cost of $O(|E|)$ adjacency matrix but also shows comparable performance against LightGCN [21]. Both global and local learning are integrated to effectively alleviate the over-smoothing problem and generate high-quality node embeddings. Extensive experiments on general and billion-scale graphs validate the superior performance in both embedding quality and model scalability.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China under grant 2018AAA0102502. Ying Zhang is supported by ARC LP210301046, DP210101393 and DP230101445.

REFERENCES

[1] 2015. LINE. https://github.com/tangjianpku/LINE.
[2] 2016. node2vec. https://github.com/eliorc/node2vec.
[3] 2018. BiNE. https://github.com/clhchtcjj/BiNE.
[4] 2020. LightGCN. https://github.com/gusye1234/LightGCN-PyTorch.
[5] 2021. BiGI. https://github.com/caojiangxia/BiGI.
[6] 2022. GEBEp. https://github.com/AnryYang/GEBEp.
[7] 2022. HCCF. https://github.com/akaxlh/HCCF.
[8] 2022. SHT. https://github.com/akaxlh/SHT.
[9] Ioannis Antonellis, Hector Garcia-Molina, and Chi-Chao Chang. 2008. Simrank++: query rewriting through link analysis of the click graph. *Proceedings of the VLDB Endowment* 1, 1 (2008), 408–421.
[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[11] Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*. Vol. 4. Springer.
[12] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite Graph Embedding via Mutual Information Maximization. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. ACM, 635–643.
[13] Chaoqi Chen, Jiongcheng Li, Zebiao Zheng, Yue Huang, Xinghao Ding, and Yizhou Yu. 2021. Dual Bipartite Graph Learning: A General Approach for Domain Adaptive Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. IEEE, 2683–2692.
[14] Lanting Fang, Kaiyu Feng, Jie Gui, Shanshan Feng, and Aiqun Hu. 2023. Anonymous Edge Representation for Inductive Anomaly Detection in Dynamic Bipartite Graph. *Proceedings of the VLDB Endowment* 16, 5 (2023), 1154–1167.
[15] Jun Gao, Jiazun Chen, Zhao Li, and Ji Zhang. 2021. ICS-GNN: lightweight interactive community search via graph neural network. *Proceedings of the VLDB Endowment* 14, 6 (2021), 1006–1018.
[16] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. BiNE: Bipartite Network Embedding. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 715–724.
[17] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.
[18] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.
[19] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2022. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2022), 3549–3568.
[20] Ruining He and Julian J. McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 144–150.
[21] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 639–648.
[22] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the International Conference on World Wide Web*. ACM, 173–182.
[23] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*.
[24] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.
[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.
[26] Zhao Li, Xin Shen, Yuhang Jiao, Xuming Pan, Pengcheng Zou, Xianling Meng, Chengwei Yao, and Jiajun Bu. 2020. Hierarchical Bipartite Graph Neural Networks: Towards Large-Scale E-commerce Applications. In *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 1677–1688.
[27] Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. 2022. Less is More: Reweighting Important Spectral Graph Features for Recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1273–1282.
[28] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. 2021. xFraud: explainable fraud transaction detection. *Proceedings of the VLDB Endowment* 15, 3 (2021), 427–436.
[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.
[30] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the International Conference on World Wide Web*. ACM, 1067–1077.
[31] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proceedings of the VLDB Endowment* 15, 6 (2022), 1201–1214.
[32] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. *Advances in Neural Information Processing Systems* 30 (2017).
[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 165–174.
[35] 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 International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 726–735.
[36] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph Neural Networks in Recommender Systems: A Survey. *Comput. Surveys* 55, 5 (2022), 1–37.
[37] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24.
[38] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy X. Huang. 2022. Hypergraph Contrastive Collaborative Filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 70–79.
[39] Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-Supervised Hypergraph Transformer for Recommender Systems. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2100–2109.
[40] Wei Xia, Quanxue Gao, Qianqian Wang, Xinbo Gao, Chris Ding, and Dacheng Tao. 2022. Tensorized Bipartite Graph Learning for Multi-View Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2022), 5187–5202.
[41] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive Learning for Sequential Recommendation. In *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 1259–1273.
[42] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the International Conference on Learning Representations*.
[43] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 3203–3209.
[44] Yoshihiro Yamanishi, Masaaki Kotera, Minoru Kanehisa, and Susumu Goto. 2010. Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics* 26, 12 (2010), 246–254.
[45] Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. 2022. Scalable and Effective Bipartite Network Embedding. In *Proceedings of the International Conference on Management of Data*. ACM, 1977–1991.
[46] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 974–983.
[47] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are Graph Augmentations Necessary?: Simple Graph Contrastive Learning for Recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1294–1303.
[48] Wentao Zhang, Xupeng Miao, Yingxia Shao, Jiawei Jiang, Lei Chen, Olivier Ruas, and Bin Cui. 2020. Reliable Data Distillation on Graph Convolutional Network. In *Proceedings of the International Conference on Management of Data*. ACM, 1399–1414.