BI-GCL: EFFICIENT SEARCH ON NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent research shows the promising potential of node continuous embedding methods in Top-K network node similarity search, which often involves finding nearest neighbors measured by similarity in a continuous embedding space. However, these methods poorly scale to searching on large networks, since their embeddings demand significant storage and entail tremendous computation costs. In this paper, we introduce a graph contrastive learning framework for compressing continuous node embeddings into binary codes that enable customized bits per dimension, striking a balance between retrieval accuracy, speed, and storage. Specifically, a recurrent binarization with GNNs is presented, which consists of two components, a GNN encoder for learning node continuous representations, and a residual multilayer perception module for encoding representations to binary codes. The whole architecture is trained end-to-end by jointly optimizing three losses, i.e., contrastive loss from giving closely aligned representations to positives, information bottleneck loss from superfluous information minimization, and representation distillation loss from aligning binary codes and their continuous counterparts. Extensive experiments demonstrates that our method achieves approximately 6x-19x faster retrieval and 16x-32x space reduction compared to traditional continuous-based embedding methods. Moreover, it reaches comparable or superior performance than state-of-the-art continuous- and hash-based network embedding methods on several real-world networks.

1 INTRODUCTION

Learning high-quality node embedding is the theme of efficient network node similarity search (retrieval), which aims to find a small set of ranked nodes that are most similar to specific queries. This has practical applications: recommending social media friends with shared interests, locating related papers or authors in citation networks, e-commerce item suggestions, etc. Many successful graph embedding methods are mainly designed for node classification or link prediction, with limited attention to node similarity search. Directly employing them for similarity search might yield sub-optimal results for a few problems. The first challenge is that many node embedding models learn continuous (float-valued) node embeddings, which are inefficient for similarity search on large networks with hundreds of thousands or millions of nodes. Filtering candidates in a continuous embedding space is highly costly due to inefficient embedding similarity calculations (e.g., cosine similarity) using massive floating-point operations (#FLOP). The worst-case linear search time complexity for continuous embeddings is even up to $O(Nd)$, where N is corpus size and d is embedding dimension. The other challenge is high memory usage, e.g., 256-dimensional continuous embeddings using standard double precision numbers for 10 million nodes require 19GB memory, which are impractical for general devices; Though traditional embedding hashing that transforms continuous embeddings into hash codes can give improved retrieval speed and memory consumption, it may suffer from poor retrieval accuracy in large-scale retrieval (e.g., million-scale one as shown in Table x) due to less discriminative representations of hash codes (only 1 bit per dimension).

In this paper, we propose to learn novel binary codes as alternatives to conventional continuous/hash embeddings for network nodes, so as to achieve a balanced goal of retrieval performance, speed, and memory requirement. Recently, (Shan et al., 2018) propose to compress an arbitrary continuous embedding into a recurrent binary embedding (RBE) for efficient sponsored search (Edelman et al., 2007). RBE progressively refines a base binary vector (compressed from a continuous vector) with binary residual vectors to narrow the retrieval performance gap between hash codes and

continuous counterparts, while offering much faster searching speed and reduced memory usage than continuous embeddings. This motivates us to borrow the idea of RBE for network node binary representation learning. However, there are some problems for RBE deployment: i) Network noise: As RBE is compressed from its continuous counterpart, it may introduce superfluous network noise from the continuous embedding, potentially degrading representation robustness. For instance, a node's continuous embedding, learned from its own features or those of its neighbors, may contain irrelevant data that negatively impacts predictions for the current node (Wu et al., 2020). ii) Degraded representation expressivity: Compressing continuous embeddings into equally expressive binary codes is NP-hard due to discrete constraints (Håstad, 2001). Prior binarization methods often overlook the transfer of representation knowledge from continuous to binary embeddings (Tan et al., 2020; He et al., 2020), resulting in reduced expressivity for accurately retrieving and ranking relevant items to queries.

To address aforementioned problems, we propose an algorithmic framework for learning binary representations (RBE) for network nodes with graph contrastive learning (Bi-GCL), which aims to effectively binarize network nodes into recurrent binary embeddings with *the Information Bottleneck principle* and *self-supervised representational distillation*. Specifically, by introducing a new GCL's objective and a probabilistic recurrent binary block, we resolve objective mismatching and enable end-to-end unsupervised training of binary embeddings with the Straight-Through Estimator. We employ the Information Bottleneck principle to learn the shared information between two augmentation views while minimizing their specific details. This helps to capture truly useful information that is predictable for identifying nodes while eliminating noises. We leverage a pre-trained continuous embedding model (as Teacher) to transfer its representational knowledge into a binary embedding model (as Student) in a self-supervised fashion, which improves our binary embedding expressivity.

Our contributions are summarized as follows:

- We propose unsupervised binary network embeddings with novel contrastive learning for efficient node similarity search.
- We enhance binary embedding learning with the Information Bottleneck principle to improve robustness against network noises.
- We introduce self-supervised representational distillation to bridge the retrieval performance gap between continuous embeddings and binary embeddings.
- Extensive experiments show our method outperforms cutting-edge approaches, achieving comparable or superior retrieval with significantly reduced speed and memory usage.

2 Related Work

Network Node Similarity Search & Graph Embedding Early works (Jeh & Widom, 2002; Zhao et al., 2009; Zhang et al., 2015) measure node similarity by structural similarity, but they are slow and miss similarities defined by node features. In recent years, graph embedding methods that are based on matrix factorization (Qiu et al., 2018), random walks (Grover & Leskovec, 2016) or deep learning Veličković et al. (2018) have achieved state-of-the-art performance and efficiency in information retrieval. Among them, graph neural networks (GNNs) have shown great effectiveness and scalability in generating embeddings Ying et al. (2018); Wu et al. (2019b). Very recently, self-supervised graph contrastive learning (GCL) (Zhu et al., 2020; Thakoor et al., 2021; Wu et al., 2021; Yu et al., 2022) has further boost GNNs' performance on various tasks. However, all these methods' learned continuous embeddings significantly degenerate similarity computation efficiency and increase storage cost. Hereby, Discrete network embedding comes as a solution to these problems. Early discrete methods borrow the idea of deep hashing in computer vision (Lai et al., 2015; Lin et al., 2015) by first learning continuous representations, and then binarizing the representations with a separated post-step, which might give sub-optimal binarization. Recently, (Yang et al., 2018; Shen et al., 2018b; Tan et al., 2020) introduce end-to-end graph hashing learning. However, their learned binary codes with only 1 bit per dimension are less discriminative than continuous embeddings, which will degrade performance in large-scale retrieval.

Learning by Information Bottleneck The Information Bottleneck (IB) principle formulates the goal of representation learning as an information trade-off between predictive power and representation compression, which is formally presented as the following objective:

$$\mathcal{R}_{IB} = I(Z;Y) - \beta I(Z;X), \tag{1}$$

where X denotes the original data, Y, Z denote the label and the corresponding representation of X, and β is the parameter that controls the trade-off. A high value of $\mathcal{R}_{IB}$ suggests that representation Z retains sufficient information to predict Y, without preserving excessive information from X. Recently, some works integrate the IB principle into the graph learning process for supervised representation enhancement(Wu et al., 2020), vital subgraph extraction (Yu et al., 2020), and meta-path aggregation (Yang et al., 2021). However, one drawback is that they use a fixed value of β, which might be too small or too large for entries with different robustness.

Knowledge Distillation Knowledge distillation (KD) (Hinton et al., 2015) aims to transfer the knowledge from a high-capacity teacher model to a smaller one without losing too much generalization power, which is also well investigated in embedding (representation) distillation (Tian et al., 2019; Aguilar et al., 2020; Sun et al., 2020). The idea is to directly align embeddings from the student with those from the teacher, e.g., minimizing the MSE between both embeddings. However, existing work mainly tackles homogeneous embedding distillation, overlooking transfers between types like continuous to discrete.

3 THE PROPOSED MODEL

In this section, we first formally introduce the problem of binary code learning for network nodes and notations. Then we outline the framework of our method Bi-GCL, and detail each component of our method.

3.1 PROBLEM STATEMENT AND NOTATIONS

We are given a graph $G = \{\mathcal{V}, \mathcal{E}\}$ as input, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent the sets of nodes and edges, respectively. We define the feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and the adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N \times N}$, where $x_i \in \mathbb{R}^F$ corresponds to the feature of node v_i, and $\boldsymbol{A}_{ij} = 1$ if and only if there is an edge between nodes v_i and v_j. Our goal is to learn a nonlinear recurrent binary function that maps an arbitrary node v to a recurrent binary code Shan et al. (2018) with customized bits per dimension. Specifically, given $\boldsymbol{X}$ and $\boldsymbol{A}$ as the inputs, the proposed Bi-GCL applies the augmentation function $aug(\boldsymbol{X}, \boldsymbol{A})$ to generate two augmented views $\tilde{G}_1 = (\tilde{\boldsymbol{X}}_1, \tilde{\boldsymbol{A}}_1)$ and $\tilde{G}_2 = (\tilde{\boldsymbol{X}}_2, \tilde{\boldsymbol{A}}_2)$. We consider $v_i^{(1)}$ and $v_i^{(2)}$ as two nodes in augmented views $\tilde{G}_1$ and $\tilde{G}_2$, generated from v_i, with $\boldsymbol{h}_i^{(1)}$ and $\boldsymbol{h}_i^{(2)}$ as two intermediate float vectors, $\boldsymbol{b}_i^{(1)}$ and $\boldsymbol{b}_i^{(2)}$ as two binary codes learned on $v_i^{(1)}$ and $v_i^{(2)}$ using a graph convolutional encoder ϕ and a recurrent binarization block ψ. The codes are learned by optimizing the multi-view graph contrastive loss, the information bottleneck loss, and the representation distillation loss.

3.2 RECURRENT BINARIZATION WITH GRAPH CONVOLUTIONAL ENCODER

Graph Augmentation. Given a graph G, we corrupt it at both structure and attribute levels by random masking of edges and node features following the approach in (Zhu et al., 2020). By graph corruption, we generate two augmented graph views $\tilde{G}_1$ and $\tilde{G}_2$, which provide diverse node contexts for Bi-GCL to contrast with, as is discussed in section x.

Graph Convolutional Encoder. We use a 2-layer K-head GAT (Veličković et al., 2018) to encode each node on the augmented views $\tilde{G}_1$ and $\tilde{G}_2$. The embedding of a node v_i at layer l is computed by: $h_i^l = \sigma(\text{Aggr}(\{\alpha_{ij}^{l,k}\boldsymbol{\Theta}^{l,k}h_j^{l-1} | j \in \mathcal{N}_i \cup \{i\}, k = 0, ..., K\}))$, $h_i^0 = x_i \in \tilde{\boldsymbol{X}}_*$, where $\alpha_{ij}^{l,k}$ is the edge coefficient of k-th head at layer l, $\boldsymbol{\Theta}^{l,k} \in \mathbb{R}^{F' \times F}$ is the learnable weight matrix, Aggr refers to the feature aggregator, i.e., average aggregator or concat aggregator, and $\sigma(\cdot)$ is a nonlinear transformation such as PRelu. The last layer's output is taken as the node representation h, the intermediate input for the subsequent recurrent binarization block that generates the binary code.

Recurrent Binarization Block. Inspired by (Shan et al., 2018), given an arbitrary float vector h learned by the GAT encoder, this block compresses the vector into a recurrent binary one. To this end, we first compress the float vector h into a binary code b_0 composed of either -1 or +1:

$$b_0 = \rho\left(\boldsymbol{W}_0(h)\right), \tag{2}$$

where $\boldsymbol{W}_0$ represents a fully-connected layer, $\rho(\cdot)$ is a probabilistic binary layer (Shen et al., 2018a) formulated as:

$$\rho\left(\cdot\right) \equiv sign(\sigma(\cdot) - t), \tag{3}$$

where $\sigma(\cdot)$ denotes a sigmoid function, and t denotes a sample from the uniform distribution $[0, 1]$.

The binary embedding b_0 is subsequently converted back into the continuous embedding $\tilde{h_0} = ||R_0(b_0)||$ using multi-layer perception (MLP) R_0 following the approach in (Gan et al., 2023). The difference between the original h and the reconstructed $\tilde{h_0}$ indicates the representation loss due to the binarization process. This loss can be further minimized by repeatedly executing the previous steps to binarize the residual components. The residual binary vector can be formulated as $r_0 = \rho(W_1(h - \tilde{h_0}))$, which is further added to the base binary $b_1 = b_0 + \frac{1}{2}r_0$. We choose the weight $\frac{1}{2}$ to ease the similarity calculation with only xor and $popcount$ (Shan et al., 2018).

Up until this point, we have presented the process of recurrent binarization when the loop count is set to 1. The loop can be tailored to strike a balance between accuracy and efficiency, the entire process of recurrent binarization, can be formally defined as:

$$\begin{cases} b_0 = \rho(\boldsymbol{W}_0(h)), \\ \hat{h}_{u-1} = ||R_{u-1}(b_{u-1})||, \\ r_{u-1} = \rho(\boldsymbol{W}_u(h - \hat{h}_{u-1})), \\ b_u = b_{u-1} + 2^{-u} r_{u-1}. \end{cases} \tag{4}$$

The binary code with $u + 1$ bit(s) per dimension (recurrent binary embedding) is the output of the binarization process in Eq. 4.

3.3 Training Schemas

Contrastive Learning Inspired by the recent success of GCL (Zhu et al., 2020; Yu et al., 2022), we propose to use a contrastive objective to get semantically similar nodes close in the representation space, while pushing dissimilar ones apart. Treating $\boldsymbol{b}_i^{(1)}$ as the anchor, the positives are: i) the inter-view embedding of $v_i^{(2)}$, i.e., $\boldsymbol{b}_i^{(2)}$; ii) embeddings of intra-view neighbors most frequently visited by random walks starting from $v_i^{(1)}$, i.e., $\{\boldsymbol{b}_j^{(1)} | v_j^{(1)} \in \text{RandomWalkSampler}(v_i^{(1)})\}$, where RandomWalkSampler selects nodes with the highest Top-T visit counts with respect to $v_i^{(1)}$ within $\tilde{G}_1$. The embeddings of nodes in the two views $\notin \{\boldsymbol{b}_i^{(1)} \cup \boldsymbol{b}_i^{(2)} \cup \{\boldsymbol{b}_j^{(1)} | v_j^{(1)} \in \text{RandomWalkSampler}(v_i^{(1)})\}\}$ are regarded as negatives. We define the contrastive loss between the two views associated with the anchor $\boldsymbol{b}_i^{(1)}$ as:

$$\ell\left(\boldsymbol{b}_i^{(1)}\right) = -\log\left\{\frac{1}{|P(\boldsymbol{b}_i^{(1)})|}\frac{\sum_{p\in P(\boldsymbol{b}_i^{(1)})} e^{\theta\left(\boldsymbol{b}_i^{(1)}, \boldsymbol{p}\right)/\tau}}{\sum_{p\in P(\boldsymbol{b}_i^{(1)})} e^{\theta\left(\boldsymbol{b}_i^{(1)}, \boldsymbol{p}\right)/\tau} + \sum_{n\in N(\boldsymbol{b}_i^{(1)})} e^{\theta\left(\boldsymbol{b}_i^{(1)}, \boldsymbol{n}\right)/\tau}}\right\}, \tag{5}$$

where $P(\boldsymbol{b}_i^{(1)})$ and $N(\boldsymbol{b}_i^{(1)})$ refer to positives and negatives of $\boldsymbol{b}_i^{(1)}$ respectively, τ is a temperature parameter, and $\theta(\cdot, \cdot)$ is defined as the cosine similarity. Since two views are symmetric, given $\boldsymbol{b}_i^{(2)}$ as the anchor, the contrastive loss $\ell\left(\boldsymbol{b}_i^{(1)}\right)$ can be similarly defined according to Eq. 5. The final contrastive loss between the two augmented views, averaged over all nodes is defined as:

$$\mathcal{L}_{CL} = \frac{1}{2N}\sum_{i=1}^{N}[\ell(\boldsymbol{b}_i^{(1)}) + \ell(\boldsymbol{b}_i^{(2)})], \tag{6}$$

The advantages of this contrastive objective are two-fold. First, it allows us to flexibly preserve more semantic similarity relations between nodes than using only traditional self-supervised graph

contrastive objectives, i.e., InfoNCE (Oord et al., 2018) or NT-Xent (Zhu et al., 2020), which will push away similar samples of an anchor, especially those that are most similar to the anchor as hard negatives across all the training process, e.g., InfoNCE and NT-Xent treat the first-order neighbors of an anchor as hard negatives and might violate the homophily assumption suggesting that connected nodes tend to share similar semantic classes in many networks and should be close to each other. On the other hand, in Bi-GCL, the number of positives per node can be properly limited to learn a even embedding distribution that helps preserve intrinsic characteristics of nodes (Yu et al., 2022). It is worth noting that the NT-Xent is a special case of the proposed contrastive loss, by making the inter-view same node as the sole positive. Second, it preserves the hard positive/negative mining property from NT-Xent (Zhu et al., 2020), and generalizes this property to all positives, thus avoiding the explicit hard mining that is delicate but critical part of many losses, e.g., triplet loss (Khosla et al., 2020). Notably, given an anchor, we consider all the inter-view samples except the inter-view same node as negatives, part of which naturally form hard negatives (e.g., inter-view counterparts with respect to intra-view positives). This improves model's ability to identify positives from hard negatives, e.g., after convolution propagation on two views with different augmentations, an intra-view neighbor u of an anchor is more similar to that anchor compared to the inter-view counterpart of u, and the model is forced to distinguish u as a positive from such hard negative. As is shown in our experiments, it is a useful trick to boost performance.

Improving under the knowledge distillation To fill the expressivity gap between continuous embeddings and binary ones, we maximize a lower-bound to the mutual information between the continuous embeddings and binary codes, aiming to transfer structured representational knowledge, i.e., dependencies between representation output dimensions (Tian et al., 2019). Formally, given a mini-batch of binary codes $[b_1; b_2; ...; b_N]$ learned on the original graph G, and their continuous counterparts (intermediate representations) $[h_1; h_2; ...; h_N]$, we employ InfoNCE to maximize the mutual information between b_i and h_i:

$$\mathcal{L}_{KD} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\theta(\boldsymbol{b}_i,\boldsymbol{h}_i)/\tau}}{\sum_{j=1}^{N}e^{\theta(\boldsymbol{b}_i,\boldsymbol{h}_j)/\tau}}, \tag{7}$$

where τ is a temperature parameter, and $\theta(\cdot,\cdot)$ is defined as the cosine similarity. Intuitively, Eq. 7 enforces $\boldsymbol{b}_i$ to be close to $\boldsymbol{h}_i$ and meanwhile pushes $\boldsymbol{b}_i$ away from continuous embeddings other than $\boldsymbol{h}_i$. It is worth noticing that the reason we choose to use InfoNCE instead of MSE, another commonly used function in KD, is that the architectures of generating continuous embeddings and binary codes as well as their representation forms are very different, in which case forcing binary codes (as students) to approach the exact values as continuous embeddings (as teachers) seems overly stringent and unnecessary for a good student, which is line with the conclusion in (He et al., 2021). Instead, Eq. 7 only attempts to ensure the information in the continuous embeddings is also captured in the binary code, which is more effective than MSE shown in our experiments.

We further reveal that maximizing the objective in Eq. 7 is also equivalent to minimize the quantization error, which is often overlooked in previous works, as they mainly focus on KD between homogeneous representations, i.e., continuous to continuous. Formally, the quantization error can be interpreted as: $\min ||\mathbf{h}-\mathbf{b}||^2$, where $\mathbf{h}$ is the continuous embedding and $\mathbf{b}$ is the binary code. We expand this equation to get: $\|\mathbf{h}-\mathbf{b}\|^2 = \|\mathbf{h}\|^2 + \|\mathbf{b}\|^2 - 2\|\mathbf{h}\|\|\mathbf{b}\|\cos\theta_{hb}$. In retrieval, we evaluate the similarities of binary codes by measuring their cosine similarities. Consequently, the magnitude of vector $\mathbf{h}$ can be ignored by normalizing it to match the norm of vector $\mathbf{b}$, i.e., $||\mathbf{h}|| = \sqrt{K}$ (i.e., $K = ||\mathbf{b}||^2$) and interpret the quantization error as to only the angle θ_{hb} between $\mathbf{h}$ and $\mathbf{b}$: $||\mathbf{h}-\mathbf{b}||^2 = 2K - 2K\cos\theta_{hb} = 2K(1-\cos\theta_{hb})$. Given that $2K$ is a constant, maximizing the cosine similarity between $\mathbf{h}$ and $\mathbf{b}$ will minimize the quantization error. This results in a more accurate binary code approximation.

Improving under the Information Bottleneck Principle We adopt the Information Bottleneck principle to retain the minimum sufficient information in learned binary codes. By dosing so, we can improve representation robustness against network noises that may negatively affect the binarization. Specifically, we encourage the divergence between the binary codes of the augmented views while maximizing the information relevant to the node retrieval task. Accordingly, in Eq. 1, we rewrite the input variables X as the intermediate float embeddings $H \triangleq \{h_i^{(1)}, h_i^{(2)}\}_{i=1}^{N}$, representations Z as

binary codes $B \triangleq \{b_i^{(1)}, b_i^{(2)}\}_{i=1}^N$, and reformat Eq. 1 as our final joint optimization objective:

$$\min_{\phi;\psi} \mathcal{L}_{CL} + \mathcal{L}_{KD} + \beta I(B;H), \tag{8}$$

For the third term in Eq. 9, by definition, we have $I(B,H) = KL\left(\mathbb{P}(b|h)||\mathbb{P}(h)\right)$, where $\mathbb{P}(h)$ denotes the distribution of h, $KL(\cdot||\cdot)$ is the KL-divergence, $\mathbb{P}(b|h) \sim Bernoulli(\sigma(\text{Concat}_{u=0}^{U}(\boldsymbol{W}_u(\gamma_u)))), \gamma_0 = h, \gamma_{u(u>0)} = h - \hat{h}_{u-1}$ (see section 3.2). From the non-negativeness of KL-divergence: $I(B,H) \leq KL\left(\mathbb{P}(b|h)||\mathbb{P}(h)\right)$, where $q(b)$ could be any distribution of b. we reduce Eq. 9 to:

$$\min_{\phi;\psi} \mathcal{L}_{CL} + \mathcal{L}_{KD} + \beta KL\left(\mathbb{P}(b|h)||q(b)\right), \tag{9}$$

In our experiments, we compute: $q(b_i^{(1)}) = p(b|v_i^{(2)})$; and $q(b_i^{(2)})$ for the view $v_i^{(1)}$ can be defined similarly. Intuitively, by encouraging the encoding distributions from different views of the same node close to each other, the model can learn the shared information between the two views and eliminate superfluous information from each view. Any representation containing all shared information from both views would have necessary label information, and the view-specific information is redundant and label-irrelevant.

To select an appropriate value of β to control the trade-off between sufficiency and robustness of the binary code, we propose to assign personalized β to nodes with different robustness. Specifically, we perform "virtual" adversarial attacks that generate adversarial perturbation in the representation space to measure the robustness of nodes, as nodes prone to attacks are regarded nonrobust and should receive higher values of β. We first employ one-step PGD (Madry et al., 2017) that injects zero-initialized perturbation δ into the embedding representation, and then calculate the gradient of the loss with respect to δ. The ℓ_2 norm of the gradient $g(\delta)_i$ of the perturbation δ on the embedding of an arbitrary node v_i is negatively correlated to the node robustness. Consequently, we can do the followings to give a personalized β_i to a node v_i:

$$\beta_i = \min\left(\max\left(\frac{||g(\delta)_i||_F}{||g(\delta)_{MEAN}||_F}\beta_{base}, \beta_{min}\right), \beta_{max}\right), \tag{10}$$

where $||g(\delta)_{MEAN}||_F$ is the mean of $||g(\delta)||_F$ with respect to all nodes, β_{base} is a base value to fine-tune personalized β_i, and $\beta_{min/max}$ guarantees the value scale locate in appropriate scope.

3.4 Retrieval with Binary Codes

(Shan et al., 2018) decomposed the cosine similarity of RBE into the dot product of hash codes (Eq. 11), with q and r denoting query and reference. This enables efficient hash code calculation through bit-wise operations like population count, XOR, and logical right shift (Eq. 12), where x, y are binary vectors in $\{1,-1\}^m$ (m is the dimensionality of RBE).

$$\begin{aligned}\mathcal{D}(b_u^q, b_u^r) \propto & \frac{1}{||b^r||}(b_0^q \cdot b_0^r + \sum_{j=0}^{u-1}\sum_{i=0}^{u-1}(\frac{1}{2})^{j+i+2} r_j^q \cdot r_i^r \\ & + \sum_{j=0}^{u-1}(\frac{1}{2})^{j+1} b_0^q \cdot r_j^r + \sum_{i=0}^{u-1}(\frac{1}{2})^{i+1} b_0^r \cdot r_i^q)\end{aligned} \tag{11}$$

$$x \cdot y = (popc(x \wedge y) >> 1) + m \tag{12}$$

Computations are efficient on GPU (Shan et al., 2018) or CPU (Gan et al., 2023), and can be deployed with various ANN algorithms (e.g., HNSW (Malkov & Yashunin, 2018)). In this work, we only focus on RBE learning for graph nodes, and simply use brute-force search in experiments.

4 Experiments

Datasets We use five real-world networks including citation networks (Bojchevski & Günnemann, 2018; Hu et al., 2020), social networks (Zeng et al., 2019) and a product co-purchasing network (Hu et al., 2020). Brief statistics of the datasets are shown in Table 1.

Table 1: The Statistics of Datasets.

Dataset	Nodes	Edges	Features	Classes	Validation/Testing
Cora	19,793	126,842	8,710	70	1400:1400
Flickr	89,250	899,756	500	7	700:700
Reddit2	232,965	23,213,838	602	41	840:840
OGBN-Arxiv	169,343	1,166,243	128	40	800:800
OGBN-Products	2,449,029	61,859,140	100	47	940:940

Evaluation Metric In Top-K node similarity search evaluation, we select two widely-used evaluation metrics Precision@K and MAP@K.

Baselines In this work, we include the following embedding models: (1) unsupervised continuous embedding methods including BGRL (Thakoor et al., 2021) GRACE (Zhu et al., 2020), DGI (Veličković et al., 2018), GAE (Kipf & Welling, 2016b), VGAE (Kipf & Welling, 2016b), (2) hashing methods including unsupervised ones: BANE (Yang et al., 2018), DNE (Shen et al., 2018b), HashGNN (Tan et al., 2020), and (3) supervised continuous embedding methods including GCN (Kipf & Welling, 2016a), SGC (Wu et al., 2019a).

We exclude potential baselines, e.g., PTE (Tang et al., 2015), BiNE (Gao et al., 2018), MF (Luo et al., 2014), Node2Vec (Grover & Leskovec, 2016), GraphSage (Hamilton et al., 2017)MVGRL (Hassani & Khasahmadi, 2020), since the above competing methods (Tan et al., 2020; Zhu et al., 2020; Thakoor et al., 2021) have validated the superiority.

Experiment Settings. We built our model using Python 3.9 and PyTorch 1.13.1. The experiments are conducted on a Linux-based system, equipped with an NVIDIA RTX4090 GPU, Intel Xeon(R)-8383C CPU @ 2.7GHz, and 256 GB of RAM. For all the baselines, we follow the official reported hyperparameter settings. For our method, we tuned the learning rate, the weight decay, the temperature parameter within $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 10^{-2}\}$, $\{10^{-6}, 10^{-5}, 10^{-4}\}$, and $\{0.5, 0.7, 0.9\}$, respectively. We execute $1 \sim 30$ random walks per node with length $\in \{1, 5, 10, 15, 20, 25, 30\}$ to pick up semantically similar neighbors for objective in Eq. 5. All models are initialized using the default normal initializer and optimized with the Adam optimizer (Kingma & Ba, 2014). We use entire network nodes as training nodes to learn representations for unsupervised methods, and employ the default dataset training splits for supervised methods. For datasets including OGBN-Products, OGBN-Arxiv, Reddit2 and Cora, we randomly select 40 nodes per class as queries, and 200 nodes per class specially for Flickr, with half of nodes per class are for validation and the rest ones for testing. The total query count with respect to all datasets are summarized in Table 1. The embedding dimension of all methods is set to 128. We use cosine similarity to measure embedding similarity, and apply brute-force search for Top-K similarity search. Particularly, in our experiments, due to large scale of big graphs like Reddit2 and OGBN-Arxiv that cannot fit into GPU memory entirely, we apply the random walk based subsampling method proposed in (Zeng et al., 2019) to sample a mini-batch of subgraphs from the original graph in the training process.

4.1 Performance Demonstration

We evaluate Top-K similarity search by varying K in $\{20, 40, 60, 80, 100\}$. A detailed comparison of the Top@40 and Top@60 results is provided in Table 2, and the rest results are in Supplementary Materials. From Table 2, we draw the following observations:

- **Our model offers a competitive similarity searching capability to state-of-the-art continuous embedding models.**
  (1). We leverage our model to generate two types of recurrent binary embeddings, $\text{RBE}^{(1)} \in \{-1.5, -0.5, 0.5, 1.5\}^{128}$ (recurrent binarization loop=1), and $\text{RBE}^{(2)} \in \{-1, 1\}^{128}$ (recurrent binarization loop=0). $\text{RBE}^{(1)}$ consistently give comparable or superior performance than all unsupervised and supervised continuous embedding baselines on all datasets. Specifically, $\text{RBE}^{(1)}$ significantly outperform most baselines on the *precision* metric, e.g. 3%$\sim$5% improvement than the most competitive methods GRACE and DGI; see for example on the datasets Reddit2 and OGBN-Arxiv. In terms of the ranking ability measured by the *MAP* metric, $\text{RBE}^{(1)}$ show com-

parable performance (slightly better or worse) compared to the most cutting-edge methods such as DGI and SGC, but RBE$^{(1)}$ is able to retrieve more semantically similar nodes. We notice that GRACE, a special case of our method Bi-GCL, is the method that gives consistently superior performance than other baselines. We attribute this to its negative mining strategy that pushes all samples except the node in the same view away from the anchor, which helps to mitigate the popular bias and preserve the intrinsic characteristics of nodes. However, as mentioned before, GRACE might deliver sub-optimal performance since it separates semantically similar samples. Bi-GCL, on the other hand, can use importance-aware random walks to effectively and efficiently preserve similarity relations between nodes. On small-scale graphs, RBE$^{(2)}$ gives comparable or even superior performance to RBE$^{(1)}$; see for example on the datasets OGBN-Arxiv and Flickr. However, RBE$^{(2)}$ become much less dicriminative for tackling large-network retrieval. We argue that one can flexibly choose the type of binary codes learned by Bi-GCL according to the network scale, in order to make a balance between retrieval performance, speed and memory usage.

- **Compared to all hashing models, our model presents remarkable performance improvement.**
(1) RBE$^{(1)}$ outperforms all hashing models, particularly by large margins on large graph such as Reddit2, OGBN-Arxiv and OGBN-Products, since the indiscriminative representations of traditional hashing models severely degrade the retrieval performance. For example, as DNE learns binary code from only network structure and do not leverage the essential information on node attributes, it cannot learn discriminative binary codes to perform well for similarity search. (2) With the help of two-stage retrieval, the ranking ability of (unsupervised) HashGNN is superior than another two hashing methods DNE and BANE, but is still worse than RBE$^{(1)}$. Among hashing methods, HashGNN gives the closest performance to our RBE$^{(1)}$, but it uses Triplet loss to capture semantically similar samples, which is sensitive and delicate to the hard negative mining. Therefore, its retrieval performance is still far behind RBE$^{(1)}$, e.g., HashGNN falls short by 5% in Precision@40 on billion-scale dataset OGBN-Products.

Table 2: Performance Comparison on Retrieval Experiment.

Model	OGBN-Products				Reddit2				OGBN-Arxiv			
	MAP@40	Precision@40	MAP@60	Precision@60	MAP@40	Precision@40	MAP@60	Precision@60	MAP@40	Precision@40	MAP@60	Precision@60
DGI	0.8222	0.4318	0.8104	0.4263	0.6951	0.4331	0.6456	0.4170	0.6673	0.2674	0.6034	0.2488
GRACE	0.8502	0.4604	0.8204	0.4436	0.8775	0.7366	0.8548	0.7209	0.6790	0.3471	0.6254	0.3285
BGRL	0.8518	0.4927	0.8292	0.4740	0.7090	0.2472	0.6451	0.2203	0.7054	0.3052	0.6550	0.2807
GAE	0.8133	0.4288	0.8086	0.4105	0.6356	0.3073	0.5756	0.2858	0.6662	0.2893	0.6126	0.2669
VGAE	0.8085	0.4224	0.8011	0.3996	0.6308	0.2839	0.5655	0.2656	0.6590	0.2885	0.5978	0.2694
DNE	0.7561	0.3747	0.7432	0.3556	0.5942	0.2391	0.5258	0.2134	0.6420	0.2417	0.5868	0.2229
BANE	0.7641	0.4099	0.7529	0.3809	0.6115	0.2744	0.5463	0.2751	0.6555	0.2545	0.5991	0.2382
HashGNN	0.7852	0.4525	0.7781	0.4067	0.7133	0.7072	0.6844	0.6850	0.6789	0.3217	0.6325	0.3021
SGC-supervised	0.8593	0.4964	0.8317	0.4775	0.8845	0.7765	0.8680	0.7604	0.6964	0.3567	0.6468	0.3320
GCN-supervised	0.8499	0.4887	0.8154	0.4646	0.8764	0.7670	0.8583	0.75659	0.6865	0.3655	0.6328	0.3476
GAT-supervised	0.8520	0.4905	0.8254	0.4716	0.8699	0.7679	0.8487	0.7565	0.6827	0.3613	0.6291	0.3433
Raw Feature	0.7756	0.4047	0.7642	0.3970	0.6408	0.1822	0.5658	0.1626	0.6391	0.2283	0.5782	0.2092
RBE^1	0.8541	0.5052	0.8275	0.4864	0.8807	0.7804	0.8645	0.7710	0.6871	0.3938	0.6441	0.3777
RBE^2	0.7742	0.4807	0.7549	0.4645	0.7412	0.6923	0.7158	0.6778	0.6863	0.3839	0.6373	0.3688

Model	Cora				Flickr			
	MAP@40	Precision@40	MAP@60	Precision@60	MAP@40	Precision@40	MAP@60	Precision@60
DGI	0.7106	0.3617	0.6552	0.3208	0.5310	0.2252	0.4621	0.2156
GRACE	0.6987	0.3231	0.6419	0.2859	0.5367	0.2227	0.4633	0.2132
BGRL	0.6831	0.2690	0.6243	0.2348	0.5278	0.2036	0.4508	0.1938
GAE	0.6352	0.2703	0.5696	0.2367	0.5199	0.1900	0.4313	0.1828
VGAE	0.6549	0.2719	0.5905	0.2377	0.5159	0.1823	0.4234	0.1743
DNE	0.5878	0.2470	0.5359	0.2111	0.4916	0.1677	0.4084	0.1596
BANE	0.6095	0.2581	0.5404	0.2228	0.5017	0.1704	0.4110	0.1625
HashGNN	0.6404	0.3522	0.5892	0.3130	0.5261	0.2212	0.4512	0.2096
SGC-supervised	0.7154	0.4248	0.6653	0.3886	0.5264	0.2220	0.4442	0.2155
GCN-supervised	0.6912	0.3610	0.6336	0.3377	0.5332	0.2234	0.4540	0.2173
GAT-supervised	0.6837	0.3525	0.6322	0.3290	0.5225	0.2190	0.4405	0.2043
Raw Feature	0.6163	0.2120	0.5443	0.1831	0.4128	0.1955	0.3632	0.1878
RBE^1	0.7009	0.3933	0.6510	0.3591	0.5245	0.2235	0.4503	0.2167
RBE^2	0.6658	0.3784	0.6177	0.3462	0.5310	0.2255	0.4568	0.2145

4.2 Ablation Study

We test the effect of the Information Bottleneck principle and the representational distillation for retrieval performance improvement, and report the results in Table 4. We have conclusions that

the RBE model learned with the Information Bottleneck principle and the representation distillation achieves better performance across all datasets.

Table 3: Ablation Study on the Representational Distillation.

Model	OGBN-Product		Reddit2		OGBN-Arxiv		Cora		Flickr	
	MAP@40	Precision@40	MAP@40	Precision@40	MAP@40	Precision@40	MAP@40	Precision@40	MAP@40	Precision@40
RBE	0.8357	0.4865	0.86242	0.76581	0.6650	0.3728	0.5918	0.3766	0.5025	0.2022
RBE+OUR Distillation	0.8461	0.4968	0.8711	0.7720	0.6765	0.3822	0.6912	0.3836	0.5135	0.2115
RBE+MSE Distillation	0.8541	0.5052	0.8807	0.7804	0.6871	0.3938	0.7009	0.3933	0.5245	0.2235

Table 4: Ablation Study on Information Bottleneck.

Model	OGBN-Product		Reddit2		OGBN-Arxiv		Cora		Flickr	
	MAP@40	Precision@40	MAP@40	Precision@40	MAP@40	Precision@40	MAP@40	Precision@40	MAP@40	Precision@40
RBE	0.8462	0.4959	0.8712	0.7711	0.6765	0.3830	0.6021	0.3822	0.5138	0.2149
RBE+IB	0.8541	0.5052	0.8807	0.7804	0.6871	0.3938	0.7009	0.3933	0.5245	0.2235

4.3 Retrieval Time and Memory usage

We mainly compare the runtime of computing dot product with respect to recurrent binary embeddings and continuous embeddings. We empirically found that computing the dot product of two recurren binary embeddings is about 6x faster ($\text{RBE}^{(1)}$) or 19x faster ($\text{RBE}^{(2)}$)than the continuous counterparts'. $\text{RBE}^{(1)}$ with 2 bits per dimension and $\text{RBE}^{(2)}$ with 1 bit respectively reduce memory consumption by 16 times and 32 times theoretically.

4.4 Case Study

In this subsection, we conduct a case study on relevant paper search on the arxiv dataset to intuitively show the effectiveness of our model.We randomly selected a highly-cited article titled *Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation* focused on the subject of medical image segmentation to serve as the query paper.We apply node similarity search to retrieve the top-5 similar papers by our model and baselines.Among the top-5 related papers, we combined the title and abstract of the paper to check whether they are relevant papers. We can get the results that the five papers retrieved by Bi-GCL mainly focus on the topic of *segmentation in the biomedical field*. Among other methods, BRGL performs the best, with only one paper not matching the target theme. On the other hand, GCN's results are less satisfactory, with just one paper fully aligning with the intended topic. It is worth highlighting that the papers retrieved by Bi-GCL are typically highly cited, indicating that our model not only returns papers with high relevance but also those that generally possess substantial academic influence. The detailed results of the relevant papers returned by all methods can be found in the appendix.

5 Conclusion

This research introduces Recurrent Binary Embeddings (RBE) as a solution to the challenges of computational inefficiency and memory intensity with continuous embeddings, as well as poor retrieval performance with hash codes in network node similarity search. Inspired by Graph Contrastive Learning (GCL), Bi-GCL provides retrieval performance comparable to or better than continuous/hash embeddings, allows efficient similarity calculations through binary dot products, and offers low memory consumption. Finally, it enables a trade-off between accuracy, computation, and memory overheads.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 7350–7357, 2020.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In International Conference on Learning Representations, 2018.

Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. American economic review, 97(1):242–259, 2007.

Yukang Gan, Yixiao Ge, Chang Zhou, Shupeng Su, Zhouchuan Xu, Xuyuan Xu, Quanchao Hui, Xiang Chen, Yexin Wang, and Ying Shan. Binary embedding-based retrieval at tencent. arXiv preprint arXiv:2302.08714, 2023.

Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. Bine: Bipartite network embedding. In The 41st international ACM SIGIR conference on research & development in information retrieval, pp. 715–724, 2018.

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

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. Advances in neural information processing systems, 30, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In International conference on machine learning, pp. 4116–4126. PMLR, 2020.

Johan Håstad. Some optimal inapproximability results. Journal of the ACM (JACM), 48(4):798–859, 2001.

Haoyu He, Xingjian Shi, Jonas Mueller, Sheng Zha, Mu Li, and George Karypis. Distiller: A systematic study of model distillation methods in natural language processing. In Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, pp. 119–133, 2021.

Tao He, Lianli Gao, Jingkuan Song, Xin Wang, Kejie Huang, and Yuanfang Li. Sneq: Semi-supervised attributed network embedding with attention-based quantisation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 4091–4098, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. Advances in neural information processing systems, 33:22118–22133, 2020.

Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 538–543, 2002.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. Advances in neural information processing systems, 33:18661–18673, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308, 2016b.

Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3270–3278, 2015.

Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep learning of binary hash codes for fast image retrieval. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 27–35, 2015.

Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Transactions on Industrial Informatics, 10(2):1273–1284, 2014.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017.

Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence, 42(4):824–836, 2018.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 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, pp. 459–467, 2018.

Ying Shan, Jian Jiao, Jie Zhu, and JC Mao. Recurrent binary embedding for gpu-enabled exhaustive retrieval from billion-scale semantic vectors. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2170–2179, 2018.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Ricardo Henao, and Lawrence Carin. NASH: Toward end-to-end neural architecture for generative semantic hashing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2041–2050, Melbourne, Australia, July 2018a. Association for Computational Linguistics. doi: 10.18653/v1/P18-1190. URL https://aclanthology.org/P18-1190.

Xiaobo Shen, Shirui Pan, Weiwei Liu, Yew-Soon Ong, and Quan-Sen Sun. Discrete network embedding. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 3549–3555, 2018b.

Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. Contrastive distillation on intermediate representations for language model compression. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 498–508, 2020.

Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. Learning to hash with graph neural networks for recommender systems. In Proceedings of The Web Conference 2020, pp. 1988–1998, 2020.

Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1165–1174, 2015.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped representation learning on graphs. In ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In International Conference on Learning Representations, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

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

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

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pp. 346–353, 2019b.

Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. Advances in Neural Information Processing Systems, 33:20437–20448, 2020.

Hong Yang, Shirui Pan, Peng Zhang, Ling Chen, Defu Lian, and Chengqi Zhang. Binarized attributed network embedding. In 2018 IEEE International Conference on Data Mining (ICDM), pp. 1476–1481. IEEE, 2018.

Liang Yang, Fan Wu, Zichen Zheng, Bingxin Niu, Junhua Gu, Chuan Wang, Xiaochun Cao, and Yuanfang Guo. Heterogeneous graph information bottleneck. In IJCAI, pp. 1638–1645, 2021.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 974–983, 2018.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In International Conference on Learning Representations, 2020.

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

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In International Conference on Learning Representations, 2019.

Jing Zhang, Jie Tang, Cong Ma, Hanghang Tong, Yu Jing, and Juanzi Li. Panther: Fast top-k similarity search on large networks. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1445–1454, 2015.

Peixiang Zhao, Jiawei Han, and Yizhou Sun. P-rank: a comprehensive structural similarity measure over information networks. In Proceedings of the 18th ACM conference on Information and knowledge management, pp. 553–562, 2009.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131, 2020.