Robust Self-Supervised Structural Graph Neural Network for Social Network Prediction

Yanfu Zhang
yaz91@pitt.edu
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA, USA

Hongchang Gao
hongchang.gao@temple.edu
Computer and Information Sciences
Temple University
Philadelphia, PA, USA

Jian Pei
jpei@cs.sfu.ca
Computer Science
Simon Fraser University
Vancouver, BC, Canada

Heng Huang
henghuanghh@gmail.com
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA, USA

ABSTRACT

The self-supervised graph representation learning has achieved much success in recent web based research and applications, such as recommendation system, social networks, and anomaly detection. However, existing works suffer from two problems. Firstly, in social networks, the influential neighbors are important, but the overwhelming routine in graph representation-learning utilizes the node-wise similarity metric defined on embedding vectors that cannot exactly capture the subtle local structure and the network proximity. Secondly, existing works implicitly assume a universal distribution across datasets, which presumably leads to sub-optimal models considering the potential distribution shift. To address these problems, in this paper, we learn structural embeddings in which the proximity is characterized by 1-Wasserstein distance. We propose a distributionally robust self-supervised graph neural network framework to learn the representations. More specifically, in our method, the embeddings are computed based on subgraphs centering at the node of interest and represent both the node of interest and its neighbors, which better preserves the local structure of nodes. To make our model end-to-end trainable, we adopt a deep implicit layer to compute the Wasserstein distance, which can be formulated as a differentiable convex optimization problem. Meanwhile, our distributionally robust formulation explicitly constrains the maximal diversity for matched queries and keys. As such, our model is insensitive to the data distributions and has better generalization abilities. Extensive experiments demonstrate that the graph encoder learned by our approach can be utilized for various downstream analyses, including node classification, graph classification, and top-k similarity search. The results show our algorithm outperforms state-of-the-art baselines, and the ablation study validates the effectiveness of our design.

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

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512182

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; • **Networks** → **Network algorithms**.

KEYWORDS

Graph neural networks, self-supervised learning, non-Euclidean distance

ACM Reference Format:
Yanfu Zhang, Hongchang Gao, Jian Pei, and Heng Huang. 2022. Robust Self-Supervised Structural Graph Neural Network for Social Network Prediction. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512182

1 INTRODUCTION

Many real-world web based research and applications involve structural data which can be represented using graphs, such as social networks [28], natural languages [25], *etc.* Graph Neural Networks (GNNs) [10, 13, 16, 16, 17, 22, 50, 52, 53] have shown superior performance in learning graph embeddings from structured web data, and the effectiveness has been verified by many downstream tasks, *e.g.* text processing [35, 49], fraud detection [54], recommendation systems [56, 62], and social network link prediction [39, 47]. For example, textGNN [63] extends the twin tower model to employ the user interactions in natural language understanding [44]; in anomaly detection, attention-based GNNs allow users to deduce the root cause of a detected anomaly [11]; in recommendation systems [31], interest aware message-passing can avoid the influence of high-order neighbors with no common interest of a user [24].

By far, most works focus on the analysis for one single graph or a fixed set of graphs. Recently, self-supervised graph representation learning achieved some success both in research and many real web applications [64]. The data sparsity problem is common due to cost of collecting and labelling the data. On the other hand, many problems require domain knowledge, and traditional unsupervised methods lack clear guidance in model designing. Given the success of transfer learning in other machine learning areas, a natural question is the feasibility of generalizable, transferrable, and robust graph representation learning. Inductive representation learning [17] is one of the early works that notice the generalization

of GNNs to "unseen" nodes during the learning. Graph Contrastive Coding [33], inspired by the success of contrastive learning, takes one step further to consider the transferability of GNNs. Inspired by self-supervised GNN, there is also evidence showing that the robustness of representations can be enhanced by graph augmentations [55]. At a high level, the prevailing self-supervised GNN pipeline learns the node embeddings via characterizing their local patterns (*i.e.* subgraphs centered at the nodes of interest) and makes use of the instance discrimination [48] framework to learn the relationships between these embeddings. There are some success attempts to apply self-supervised learning to real problems. For example, in recommendation systems, the quality of users and items representations from historical interactions is central to the success of collaborative filtering. Via incorporating item contents into the learning scheme, the contrastive-based methods can enforces dimension-wise similarity between feature representation and collaborative embedding, which avoid some noise edges, such as the irrelevant interaction of users with a large bulk of items. With the dramatic social media boom, the social recommender system can find the high-order connectivity information from the social influential users using self-supervised methods [56]. and capture the behavior of users even for those having few interactions with items. In fraud detection in finance and spammer discovery in social media, the hop-count distribution is different for anonymous nodes and normal nodes, and predicting shortest path length between a pair of nodes using self-supervised learning methods can provide some evidence to anomaly detection [54].

Although the success of aforementioned self-supervised methods in social networks and related applications, several important problems remain to address. Firstly, the node representation learned in most existing self-supervised methods is focused on the node-wise proximity, and the proximity of local structures is less considered. Secondly, the local consistency of the users are seldom considered, which is related to distribution shift. Sub-graph sampling techniques are widely exploited in the training of GNNs as a consequence of the size of modern large-scale graph data. However, additional distribution shifts would be introduced by different subgraph sampling techniques, while prior works implicitly assume that the data distribution is universal for different sources. The distributional shift potentially deteriorates the generalization and transferability. These problems urge *network proximity defined on the embedding of local structures* and invoke an explicit consideration on the *distributional robustness* of the self-supervised GNNs.

To address the aforementioned challenging problems, in this paper, we proposed a novel distributionally robust graph contrastive learning framework, dubbed learning *structural embeddings*. In contrast to existing works that focus on the node-wise proximity, our method learns the local-structural level proximity via gathering an embedding set describing both a node and its neighbors, *i.e.* a subgraph surrounding the node of interest. The difference between two nodes is then characterized via the Wasserstein distance defined on the distribution of nodes of corresponding subgraphs. To alleviate the distributional shift, we further design a distributionally robust contrastive learning framework by constraining the maximal inconsistency for similar subgraphs. The Wasserstein distance to measure the proximity for self-supervised learning is considerably challenging in computation because the contrastive loss function based on the Wasserstein distance becomes a difficult bi-level optimization problem. In this paper, we employ a differentiable implicit layer to deal with the Wasserstein distance, making our framework end-to-end trainable. Our contributions are summarized as follows:

- We propose new graph embeddings at a local-structural level via learning the embedding sets for the subgraphs, which are composed of the nodes of interest and their neighbors. The Wasserstein distance is adopted as the network proximity.
- We introduce a distributionally robust contrastive learning framework. To circumvent the difficult minimax problem, the original problem is relaxed to an asymptotic formulation. We resort to the differentiable optimization methods to compute the Wasserstein distance, which makes the full network end-to-end trainable.
- Extensive experiments are conducted on various tasks and representative datasets. The results demonstrate that our algorithm outperforms other state-of-the-art methods in several important downstream analysis. The ablation study validates the effectiveness of our approach.

1.1 Notations

Throughout the paper, the bold capital and bold lowercase symbols are used to represent matrices and vectors, respectively. If all elements of a matrix $\boldsymbol{A}$ are greater than or equal to 0, we denote it by $\boldsymbol{A} \geq 0$. We use $G = \{V, E\}$ to represent a graph. Here V is the node set, and E is the edge set. Finally, a $n \times n$-identity matrix is denoted by $\mathbf{I}_n$, $\mathbf{1}_n$ is a n-dimension one vector, and $\mathbf{0}$ denotes a zero matrix.

2 RELATED WORKS

2.1 Graph Representation Learning

Graph representation learning is featured by mining the topological structures of graphs and encoding nodes with low-dimensional embeddings. Representative works, including Word2Vec [26], DeepWalk [32], and LINE [41], collect local patterns and learn mappings from graphs to vectors. To capture the highly non-linear property of attributed graphs, Deep Attribute Embedding Network (DANE) [15] upgrades the shallow models in the aforementioned model to deep networks.

There are increasing interest in GNN [22], which is developed from graph convolution and can simultaneously exploit the structural knowledge and the enriched side information in attribute graphs. GCN [22] shows the superiority of the first-order graph convolution for semi-supervised node classification. GraphSage [17, 52, 53] and Message Passing Neural Network (MPNN) [16] broaden the application of GNNs to the analysis for large-scale graphs via a more efficient aggregation mechanism. Graph Attention Network (GAN) [45] introduces the attention mechanism which is shown to be effective in general network analysis. recently, the relation between GNNs and graph isomorphism problem is studied, based on which Graph Isomorphism Network (GIN) [50] is designed.

In this paper we use GIN as the backbone models in our self-supervised graph neural networks.

2.2 Graph Contrastive Learning

Recently, contrastive learning has attracted a surge of attention in a machine learning community. A typical scheme is to select an anchor, a positive instance, and a negative instance, and maximize

the margin between the similar pair (anchor *v.s.* positive) and the negative pair (anchor *v.s.* negative). Instance discrimination [48] is a popular self-supervised framework that achieves state-of-the-art performance in many computer vision tasks, for example, SimCLR [8] and SimCLRv2 [9].

Graph contrastive learning requires different objectives due to the unique data structure, such as Jensen-Shannon estimator [38, 46], the noise-contrastive estimation (NCE) [55] and parametric estimation method using projection head [18]. To generate graph views, different augmentation methods are proposed. One of the most commom method is node attribute masking [55] which applies the feature transformations. For a given graph, edge perturbation [33, 55] can randomly adds or drops edges. Another direction is sampling-based transformation. For example, ego-nets sampling, such as in DGI [46], InfoGraph [38] and MVGRL [18], can be viewed as the unification of the contrast between graph-level and node-level representations.

In this work, we adopt the InfoNCE loss [30] and instance discrimination [48] methods, which are demonstrated to have good performance in graph pre-training [33].

2.3 Network Proximity

There are several perspectives concerning the definition for the proximity between different nodes in networks. A variety of works are based on the neighborhood similarity computed locally from the neighborhood of the vertices, for example, Jaccard similarity and cosine similarity. On the other hand, the structural similarity consider the similarity *w.r.t.* local patterns. Models of this genre include structural diversity [43], motif [5, 27], and spectral methods [12]. A more fine-grained definition [41] considers the first-order proximity, second-order proximity, and high-order proximity. GraRep [6] considers the connectivity between different nodes via explicitly constructing the probability transition matrix. Alternatively, DeepWalk [32] explores the connectivity and the local pattern via random walk with restart. For attribute graphs, GNN [22] and its variants [45, 52] are powerful tools to leverage the side information in computing network proximity. Another related topic to the network proximity is the graph matching problem, where a principally similar idea to the graph neural network is developed. The representative works in this direction include the Weisfeiler-Lehman Isomorphism Test [36] and the related graphlet methods [37].

2.4 Deep Implicit Layers

Deep neural networks are heavily dependent on the gradient-based optimization, *e.g.* Momentum Stochastic Gradient Descent [40] and Adam [21]. However, constrained optimization is seldom integrated into deep neural networks due to the difficulty of the automatic differentiation concerning the boundary.

Recently, neural ordinary differential equations [7] provides a new interpretation for the residual neural layers, which states that each residual layer can be viewed as one differential operation. Inspired by this perspective, it is shown that deep neural networks can be utilized to solve convex constrained problems, referred to as differentiable optimization [1]. OptNet [2] is an initial study, in which a special deep layer is designed to solve quadratic programming problems. Of note, in regular tasks, the parameters in the convex constrained problems of interest usually are computed using the outputs from preceding layers. An important result of the deep implicit layer is that the gradient of the parameters can be computed using the implicit function theorem.

In this paper, we use Wasserstein distance to characterize the node proximity, which can be formulated as a linear programming problem. Given the success of deep earth move distance in few-shot learning [57], we adopt the deep implicit layer to compute Wasserstein distance, which admits a fully end-to-end graph neural network.

3 PROPOSED METHOD

At a high level, we sample subgraphs spanned from a node and learn the embeddings via a graph encoder to capture the local pattern. Each node is an individual class and the similarity of subgraphs is characterized by the network proximity defined on the embeddings. Figure 1 illustrates the pipeline of our approach. We consider a subgraph triplet $\left(G_{q_i}, (G_{k_i}, G_{k_j})\right)$, in which G_{q_i} and G_{k_i} are from the same node and G_{k_j} is from a different one. We encourage a large margin between the proximity for similar pair (G_{q_i}, G_{k_i}) and that for dissimilar pair (G_{q_i}, G_{k_j}).

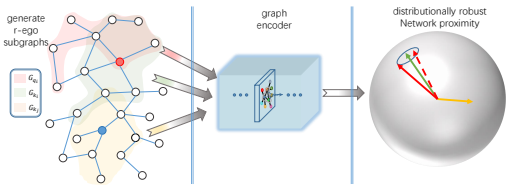

Figure 1: Subgraphs generated from the same node (colored in red) are similar, and from different nodes are dissimilar (red *v.s.* blue). To learn a robust encoder, we fix the keys, *i.e.* G_{k_i} and G_{k_j}, and focus on the most difficult query G_{q_i} (solid red line) instead of random queries (dashed red lines).

Compared to prior works, our approach is robust in two senses. Firstly, we expand the embedding space and exploit Wasserstein distance as the network proximity. Sampling subgraphs is a standard step for modern large-scale graph analysis. In this paper, we propose to use *structural embeddings* defined on subgraphs crawling around the nodes of interest, *i.e.* the comprehensive information gathered from the center nodes and their neighbors. We use the structural embeddings, which are sets of node embeddings, to explicitly capture the subtle difference between subgraphs. More specifically, we characterize the non-linear and non-local relationships between these embeddings by Wasserstein distance. Of note, we propose to utilize an automatic differentiable solver to compute the Wasserstein distance, which leads to an end-to-end trainable network. Secondly, we focus on the most ambiguously similar pairs, which leads to our formulation insensitive to data distributions. For example, sampling techniques and their parameters may introduce fluctuation concerning the distribution of subgraphs. We propose an asymptotically distributionally robust contrastive learning framework, which is reluctant to distribution shift and easy for computation.

In the following, we first formalize our self-supervised GNN and the associated network proximity in §3.1. Then, in §3.2 we propose our distributionally robust contrastive learning framework and formulate an asymptotic scenario to make the problem tractable. At last, in §3.3 we introduce an end-to-end trainable neural network featured by a differentiable implicit layer to compute the Wasserstein distance. We summarize the full algorithm and discuss some details in §3.4.

Notations: we use the bold capital and bold lowercase symbols to represent matrices and vectors. $\boldsymbol{A} \geq 0$ denotes all elements of a matrix $\boldsymbol{A}$ are greater than or equal to 0. $G = \{V, E\}$ is a graph, V is the node set, and E is the edge set. A $n \times n$-identity matrix is denoted by $\mathbf{I}_n$, $\mathbf{1}_n$ is a n-dimension one vector, and $\mathbf{0}$ denotes a zero matrix.

3.1 Self-Supervised Graph Neural Network via Wasserstein Proximity

We adopt the r-ego network [33] to represent the local structure of node i, defined as follow,

Definition. ***r-ego network*** *[33] For graph $G = (V, E)$, the r-neighbors of a node $v \in V$ are defined as $\mathcal{N}_v = \{u | d(u, v) \leq r\}$, where $d(u, v)$ is the shortest path length between node u and v. The r-ego network of v, denoted by G_v, is the subgraph induced by $\mathcal{N}_v$.*

r-ego networks can be augmented via graph sampling techniques, *e.g.* random walks with restart [42]. We consider two subgraphs via augmenting the same node i, denoted as G_{q_i} and G_{k_i}, and other subgraphs from different nodes $\{j\} \subset V$, denoted as $\{G_{k_j}\}$. In instance discrimination learning, each r-ego network is viewed as a distinct instance. Therefore, G_{q_i} and all G_{k_j} are considered to be similar, and G_{q_i} and G_{k_i} are considered to be dissimilar. Specifically, G_{q_i} is referred to as the *query*, and the elements in the set $\{G_{k_j}\} \cup \{G_{k_i}\}$ are referred to as the *keys*.

Our self-supervised GNN are then encouraged to maximize the margin between similar instances and dissimilar instances. We define $f_q, f_k : V \times \mathcal{S} \to \mathcal{Y}$, where $\mathcal{S}$ is the augmentation function space and $\mathcal{Y}$ is the embedding space. $f_q(\cdot)$ and $f_k(\cdot)$ map an augmentation of a node to the query embeddings and the key embeddings respectively, and for simplicity we omit the augmentation function. Let $d_g : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a network proximity function, then $d_g\left(f_q(i), f_k(j)\right)$ represents the proximity between query node i and key node j. Finally, let $\ell_d(\cdot)$ be the objective function for our self-supervised graph neural network, which is defined on the proximity between a group of nodes and detailed in the following.

Prior works usually defines $\mathcal{Y} \subseteq \mathbb{R}^n$, *i.e.* a n-dimension vector space. In this paper, we propose to enlarge the capacity of the embedding space by defines $\mathcal{Y} \subseteq \mathbb{R}^{n \times m}$. The computation of our method is illustrated in Fig. 2.

In the sampling framework, the node representations are computed on the induced subgraph instead of the full graphs, in which potential bias is introduced due to the perturbation in the sampled r-ego subgraph G_i. To address this problem, an alternative method is to expand the node representations to *subgraph representations* which allows some uncertainty. More specifically, we use an *embedding set* for r-ego subgraph $G_i = (V_i, E_i)$, instead of the fixed vector representation. Let $\boldsymbol{y}_{ic}$ be the embedding for a node $v_{ic} \in G_i$, the embedding set for G_i can be denoted as $\mathcal{Y}_i = \{\boldsymbol{y}_{ic} | c = 1, 2, \ldots, C\}$ with C be the cardinality of V_i. To learn an embedding set, we can use a two-stage subgraphs sampling on top of a general graph neural network backbone: given the node i, in the first stage, we sample G_i to obtain V_i, and in the second stage we sample $\bar{G}_i = \{G_{ic} | c = 1, 2, \ldots, C\}$. All G_{ic} are fed into the backbone model and the node embeddings are gathered to form a subgraph embedding.

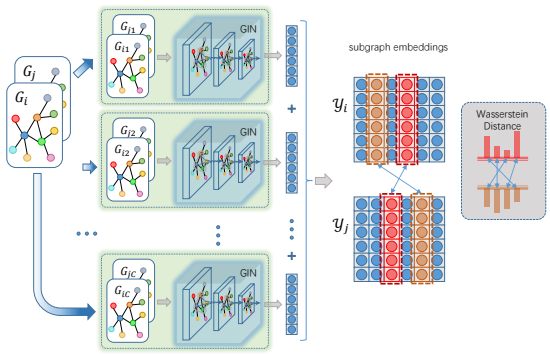

Figure 2: The r-ego subgraphs surrounding the nodes of interest are fed into the structural GNN to obtain the embedding sets. Wasserstein distance is then computed as the network proximity.

The embedding sets are composed of multiple node representations therefore having better representation abilities and larger capacity for capturing the subtle difference in sampled subgraphs. The network proximity defined on the embedding sets can be interpreted as a matching problem. We adopt the 1-Wasserstein distance to evaluate the difference between $\mathcal{Y}$ and $\tilde{\mathcal{Y}}$,

$$\mathcal{W}\left(\mathcal{Y}, \tilde{\mathcal{Y}}\right) = \min_{x_{ij}} \sum_{i,j=1}^{C} x_{ij} c(\boldsymbol{y}_i, \tilde{\boldsymbol{y}}_j), \quad c(\boldsymbol{y}_i, \tilde{\boldsymbol{y}}_j) = 1 - \frac{\boldsymbol{y}_i^\intercal \tilde{\boldsymbol{y}}_j}{|\boldsymbol{y}_i||\tilde{\boldsymbol{y}}_j|}, \quad (1)$$

here x_{ij} is a match and we reserve the constraints concerning x_{ij} for the next section. The cost function $c(\boldsymbol{y}_i, \tilde{\boldsymbol{y}}_j)$ measures the dissimilarity between $\boldsymbol{y}_i$ and $\tilde{\boldsymbol{y}}_j$.

3.2 (Asymptotically) Distributionally Robust Contrastive Learning

3.2.1 Distributionally Robust Contrastive Learning. : The support of the embeddings $\mathcal{Y}_i$ for node i is exactly the feasible set for r-ego subgraphs G_i. Prior works consider an empirical expectation form for the objective functions. However, as aforementioned, the distributions of r-ego subgraphs are presumably affected by the specific sampling method. Local structures potentially lead to significantly different distributions for r-ego subgraphs, and an example is given in Fig. 3. As such, it may introduce bias into the learned global information. To avoid this bias, robustness is desired for our self-supervised graph neural networks.

In this paper, we focus on the difficult queries instead of equally treating all queries. Specifically, for given keys and candidate r-ego subgraphs, we only consider the most difficult query, defined as the

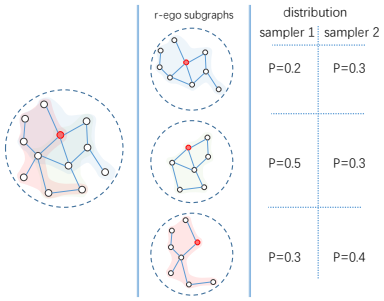

Figure 3: An example shows the distribution of r-ego subgraphs are determined by the sampling methods. The node of interest is colored in red, and three possible subgraphs are shaded in different colors (middle column). Under different sampling techniques (*e.g.* altering the backward jump probability in random walks), the distribution of subgraphs are presumably different (right column).

one that has the worst similarity with the matched key. Formally, we define,

$$\mathcal{L} = \min_{f_q, f_k} \max_{G_{q_i}} -\frac{1}{N} \sum_{i=1}^{N} \log \frac{1}{\mathcal{Z}} \exp\left(\mathcal{W}_{\pi_i}\left(G_{q_i}, G_{k_i}\right)\right), \tag{2}$$

here $\mathcal{W}_{\pi_i}$ is Wasserstein distance dependent on a data-related constrain π_i whose details will be discussed in the next section. $\mathcal{Z}$ is a normalization term,

$$\mathcal{Z} = \exp\left(\mathcal{W}_{\pi_i}\left(G_{q_i}, G_{k_i}\right)\right) + \sum_{j=1}^{m} \exp\left(\mathcal{W}_{\pi_j}\left(G_{q_i}, G_{k_j}\right)\right). \tag{3}$$

We refer (2) as the *distributionally robust contrastive learning*, because a contrastive objective is adopted and the it can be represented as a distributionally robust optimization (DRO) problem.

The distributionally robust formulation is weakly related to the support of the embedding set $\mathcal{Y}_i$. Rather, compared to the expectation form, it directly constrains the most diverse match between the query and key. This property assures the learned model is *robust* to distribution shift caused by either the sampling methods for r-ego subgraphs or the data difference.

3.2.2 Asymptotically Distributionally Robust Contrastive Learning. : The DRO problem defined by (2) can be solved via the duality form. In detail, a feasible method is to solve the inner maximal problem and the outer minimal problem alternatively, for example, the projected gradient descent in adversarial training. However, in our case the inner problem is difficult and the reasons are two-folded. Firstly, the closed-form solution is difficult to derive, due to the complexity of the subgraph distributions and the computation of the Wasserstein distance. Secondly, the graph data are structural and discrete, which means that gradient-based methods cannot be immediately adopted. To overcome the challenge, we propose an asymptotic relaxation.

For simplicity, let $\ell_w(G_{q_i}) = \log \frac{1}{\mathcal{Z}} \exp\left(\mathcal{W}_{\pi_i}\left(G_{q_i}, G_{k_i}\right)\right)$. We re-write (2) as follow,

$$\max_{G_{q_i}} \ell_w(G_{q_i}) = \sum \left(\mathbb{I}_i(G_{q_i}) \ell_w(G_{q_i})\right) \tag{4}$$

here $\mathbb{I}_i(G_{q_i})$ is an indicator function,

$$\mathbb{I}_i(G_{q_i}) = \begin{cases} 1, & G_{q_i} = \arg\max_{G_{q_i}} \ell_w(G_{q_i}) \\ 0, & otherwise \end{cases}.$$

Problem (2) is transformed into an integer optimization problem. To make the problem tractable, we relax $\mathbb{I}_i(G_{q_i})$ with $h_i(G_{q_i})$,

$$h_i(G_{q_i}) = \left(1 + \exp\left(\frac{1}{\tau}\left(\hat{\mathcal{W}}_i - \mathcal{W}_{\pi_i}\left(G_{q_i}, G_{k_i}\right)\right)\right)\right)^{-1}, \tag{5}$$

here τ is a temperature parameter, $\hat{\mathcal{W}}_i$ is threshold indicating the maximal Wasserstein distance. The relaxation is referred to as *asymptotically Distributional robustness* because when $\tau \to 0$ and $\hat{\mathcal{W}}_i \to \max_{G_{q_i}} \mathcal{W}_{\pi_i}\left(G_{q_i}, G_{k_i}\right)$, $h_i(G_{q_i})$ converges to the distributionally robust objective. $\hat{\mathcal{W}}_i$ can be empirically estimated and updated during the optimization, and the details are given in §3.4.

3.3 Computing Wasserstein Distance via Deep Implicit Layer

(2) involves the computation of the 1-Wasserstein distance between the queries and the keys, which can be formulated as a linear programming problem,

$$\mathcal{W}_{\pi_i}(G_1, G_2) = \min_{x_{ij}} \sum_i \sum_j c_{ij} x_{ij}, \tag{6}$$
$$s.t. \quad x_{ij} \geq 0, \quad \sum_i x_{ij} = s_j, \quad \sum_j x_{ij} = d_i, \qquad \forall i, j,$$

here c_{ij}, s_j and d_i are parameters computed from the embeddings, which can be viewed as functions defined on some input $\boldsymbol{\theta}$. In our case, $\boldsymbol{\theta}$ can be interpreted as the embeddings and other model parameters. Directly integrating (6) in our deep model will lead to intractable gradients. To solve this challenged problem, we resort to a deep implicit layer encoding the Wasserstein distance computation. A deep implicit layer exploits the (Karush–Kuhn–Tucker) KKT conditions to solve a convex problem with the parameters fixed, and computes the gradients *w.r.t.* $\boldsymbol{\theta}$ using the implicit function theorem. As such, we can build a deep neural network by integrating the graph encoder and the Wasserstein layer, which can solve (6) and can be trained in an end-to-end manner. For self-containedness, the detailed derivation of the deep implicit layer encoding the Wasserstein distance is given in the following.

Let $N = m \times n$, $f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{c}^\intercal \boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^N$, and the k-th entry is x_{ij} with $i = \lfloor k/n \rfloor$ and $j = k \mod n$. $\boldsymbol{c} : \boldsymbol{\Theta} \to \mathbb{R}^N$ is the vectorized form of the cost function. We first re-write (6) as,

$$\mathcal{W}_{\pi_i}(G_1, G_2) = \min f(\boldsymbol{x}, \boldsymbol{\theta}), \tag{7}$$
$$s.t. \qquad G(\boldsymbol{\theta})\boldsymbol{x} \leq \boldsymbol{0}, \quad h(\boldsymbol{x}, \boldsymbol{\theta}) = 0,$$

here $G(\boldsymbol{\theta}) = diag(\boldsymbol{-1})$, and $diag(\cdot)$ maps the given vector to a diagonal matrix. Let $\boldsymbol{s} \in \mathbb{R}^m$ and $\boldsymbol{d} \in \mathbb{R}^n$ be the vectorized $s_j(\boldsymbol{\theta})$ and $d_i(\boldsymbol{\theta})$. $h(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{A}\boldsymbol{x} - [\boldsymbol{s}, \boldsymbol{d}]$, where $[\boldsymbol{s}, \boldsymbol{d}]$ is the concatenation of $\boldsymbol{s}$ and $\boldsymbol{d}$. $\boldsymbol{A} \in \mathbb{R}^{N \times (n+m)}$ arranges the linear constraints, , defined

as,

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & & \\ & \ddots & \\ & & \mathbf{1} \\ diag\,(\mathbf{1}) & \cdots & diag\,(\mathbf{1}) \end{bmatrix}. \tag{8}$$

(7) is a constrained convex problem. Let $\boldsymbol{v}$ and $\boldsymbol{\lambda}$ be the dual variables for the equalities and inequalities in the constraints, its Lagrangian is,

$$L\,(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = \boldsymbol{c}^\intercal \boldsymbol{x} + \boldsymbol{\lambda} G(\boldsymbol{\theta})\boldsymbol{x} + \boldsymbol{v} h(\boldsymbol{x}, \boldsymbol{\theta}). \tag{9}$$

It is easy to verify that (7) satisfies the Slater's condition and the twice differentiability. As such, the KKT conditions of (9) are the necessary and sufficient optimality conditions for (7). More specifically, define $g\,(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\theta}) = [\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\theta}), diag(\boldsymbol{\lambda}) G(\boldsymbol{\theta})\boldsymbol{x}, h(\boldsymbol{x}, \boldsymbol{\theta})]^\intercal$. If $g(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) = 0$ for some $\tilde{\boldsymbol{z}} = (\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}})$ where $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{v}}$ are both feasible, then the KKT conditions are satisfied and $\tilde{\boldsymbol{x}}$ is optimal. We can compute the partial Jacobian regarding $\boldsymbol{z}$ (omitting the last two columns) and $\boldsymbol{\theta}$ as,

$$D_{\boldsymbol{z}} g(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) = \begin{bmatrix} D_{\boldsymbol{x}} \nabla_{\boldsymbol{x}} L(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) \\ diag(\tilde{\boldsymbol{\lambda}}) D_{\boldsymbol{x}} G(\boldsymbol{\theta})\tilde{\boldsymbol{x}} \\ D_{\boldsymbol{x}} h(\boldsymbol{x}, \boldsymbol{\theta}) \end{bmatrix}, D_{\boldsymbol{\theta}} g(z, \theta) = \begin{bmatrix} D_{\theta} \nabla_{\boldsymbol{x}} L(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) \\ diag(\tilde{\boldsymbol{\lambda}}) D_{\theta} G(\boldsymbol{\theta})\tilde{\boldsymbol{x}} \\ D_{\theta} h(\boldsymbol{x}, \boldsymbol{\theta}) \end{bmatrix}. \tag{10}$$

The following theorem characterizes the differentiability of convex optimization.

Theorem 1. *(Differentiability of a Convex Optimization Problem [4]) Given a convex problem, assume (1) Slater condition holds, (2) all derivative exists, (3) $\{i|\lambda_i = 0$ and $f_i(x,\theta)\} = \emptyset$, and (4) $D_{\boldsymbol{x}} g(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\theta})$ is non-singular. If $g(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\theta}) = 0$, the solution mapping has a single-valued localization s around $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}}$ that is continuously differentiable in a neighborhood $\boldsymbol{Q}$ of $\boldsymbol{\theta}$ with Jacobian satisfying, $D_{\boldsymbol{\theta}} s(\boldsymbol{\theta}) = -D_{\boldsymbol{z}} g(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\theta})^{-1} D_{\boldsymbol{\theta}} g(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\theta})$ for every $\boldsymbol{\theta} \in \boldsymbol{Q}$.*

Remark. *Theorem 1 is an immediate result from the Implicit Function Theorem [23]. Recall that in our problem, $\boldsymbol{\theta}$ are the embeddings of subgraphs. Theorem 1 states that the gradient w.r.t. $\boldsymbol{\theta}$ can be computed according to* (10). *In other words, backward propagation is feasible.*

3.4 Algorithm and Implementation

The first stage of sampling is based on neighbor sampling, and the second stage is based on random walk with restart. To accelerate the training, we use the one-hop neighbors in the first stage. We use Graph Isomorphism Network (GIN) as the backbone model in our structural GNN. For the Wasserstein distance, let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two embedding sets, s_j in (6) is defined by,

$$s_j = \frac{C\bar{s}_j}{\sum_{i=1}^{C} \bar{s}_j}, \text{ where } \bar{s}_j = \max\left\{\sum_{k=1}^{C} \boldsymbol{u}_j^\intercal \boldsymbol{v}_k, 0\right\}, \tag{11}$$

and d_i is defined similarly. We initialize $\hat{\mathcal{W}}_i$ and τ in (5) with 0 and 3 respectively. $\hat{\mathcal{W}}_i$ is estimated dynamically and τ is gradually decreasing during training. The full algorithm is summarized in Algo. 1.

Our implementation is based on the DGL package [1] and OptNet [2], which is designed for solving quadratic programming (QP) problems. Specifically, the backward propagation is accomplished via automatic differentiation through a customized QP solver [2]. To tailor our problem to fit into OptNet, we only need to substitute the objective function $\boldsymbol{c}^\intercal \boldsymbol{x}$ with a quadratic form $\frac{1}{2}(\boldsymbol{c}^\intercal \boldsymbol{x})^2$, and the above derivation remains untouched. It is easy to verify that the surrogate quadratic objective has the same optimums as the original linear programming problem (6).

Algorithm 1: Distributionally Robust Self-Supervised Graph Neural Network

Input: Graph G, key size C
Output: Model Parameter W

Initialize τ, $\hat{\mathcal{W}}_i$;
while *epoch not end* **do**
  for $i \in V$ **do**
    Sample r-ego subgraphs, G_{q_i} and G_{k_i} ;
    Sample r-ego subgraphs, G_{k_j}, $j = 1, 2, \ldots, C$;
    Update W *w.r.t.* (4) and (5), ;
    $\hat{\mathcal{W}}_i \leftarrow 0.999 \times \max\left\{\hat{\mathcal{W}}_i, \mathcal{W}_{\pi_i}\left(G_{q_i}, G_{k_i}\right)\right\}$;
    $\tau \leftarrow 0.999 \times \tau$;
  end
end

4 EXPERIMENTAL RESULTS

In this section, we evaluate our approach for downstream graph analysis, which includes two steps. We first pre-train the model using several large-scale graph datasets. Then we finetune the pre-trained model according to the specific tasks. In §4.1 we describe the pre-training setting of our robust self-supervised graph neural network. In §4.2 we detail the finetuning and evaluate our approach on three standard downstream benchmarks via comparing the performance with related baselines. In §4.3 we conduct the ablation study to demonstrate the effectiveness of our design.

4.1 Training Self-Supervised Graph Neural Network

We follow the experimental setting in GCC [33] and use six graph datasets from NetRep [34] and SNAP [3, 52] covering academic graphs and social networks. In detail, we use Academia and two DBLP datasets for academic graphs, and Facebook, IMDB, and LiveJournal datasets for social networks. The statistics of the pre-training datasets are summarized in Table 1.

For our robust self-supervised method, we adopt a 5-layer 64-hidden-units GIN as the backbone model. We use one-hop neighborhood sampling with 5 neighbors in the first stage, and a random walk with restart probability 0.8 in the second stage. The query encoder is trained using Adam optimizer with learning rate of 0.001, $\beta_1 = 0.5$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We use the momentum contrast (MoCo) method [19] to update the key encoders. The mini-batch size is 32, and the dictionary size is 16,384. The momentum is set to 0.9. The pre-train takes 50,000 steps and the learning rate is decayed by $\frac{1}{10}$ in step 10,000 and step 30,000. We include two versions

[1] https://docs.dgl.ai/index.html

[2] https://github.com/locuslab/qpth

of our approach, *proposed*$^{\Delta}$ which omits the distributional robust consideration, and *proposed* which is the full approach.

4.2 Downstream Analysis

We consider three standard benchmark tasks for graph learning algorithms, *i.e.* node classification, graph classification, and top-k similarity search. Through the experiments, we use GCC [33] and its variants as the self-supervised baselines. Supervised baselines are detailed in the associated experiments to be discussed in the following.

4.2.1 Node Classification. : In this task, we predict the unknown node labels in a partially labeled network. We adopt US-Airport and H-index as the benchmark datasets. US-Airport contains the airline activity levels among 1190 airports. H-index is an academic dataset indicating the distribution of the h-index of authors extracted from OAG [58]. We use the pre-trained query encoder to extract features and a logistic regression classifier to make the final prediction. Specifically, we finetune the query encoder and the classifier for 90 epochs using a batch size of 128.

We consider ProNE [60], GraphWave [12], and Struct2Vec [14] as supervised baselines. The cardinality of subgraphs sets is 5. The results are presented in Table 2. For node classification, we find the pre-trained graph neural networks are superior to the supervised models. For example, our model outperforms the best supervised baseline, Struc2Vec, by up to 2.7. Compared to self-supervised baselines, our algorithm shows improvement with better stability for different datasets. The distributional robust consideration also improves the model performance significantly.

4.2.2 Graph Classification. : In this task, we predict the labels for graphs. We consider five datasets [51], including IMDB-Binary (IMDB-B), IMDB-Multi (IMDB-M), COLLAB, Reddit Binary (RDT-B), and Reddit-Multi5k (RDT-M). We finetune the pre-trained encoder and a classifier parallel to the node classification. The main difference is that we use the full graphs instead of r-ego networks.

We consider DGK [51], Graph2Vec [29], InfoGraph [38], DGCNN [61], and GIN [50] as supervised baselines. The results are presented in Table 3. Of note, our approach consistently outperforms the self-supervised baselines on most datasets. For COLLAB, our approach is competitive to the best-performing self-supervised baselines. Meanwhile, our method is also competitive to the supervised baselines. For example, our method has the same performance with the best-performing baseline, GIN, on IMDB-B, and is also very close on IMDB-M. In COLLAB, our model surpasses the best performing supervised baseline by a large margin.

4.2.3 Top-k Similarity Search. : Given two graphs, the top-k similarity search attempts to find the most similar vertices in one graph for the vertices in the other graphs. We adopt the co-author graphs [59] of K-I (KDD *v.s.* ICDM) S-C (SIGIR *v.s.* CIKM), and S-I (SIGMOD *v.s.* ICDE) and define the ground truth similarity as the common authors in both conferences. We use top-10 accuracy, *i.e.* HITS@10 as the performance measurements. Of note, this is an unsupervised task, and we use the pre-trained model without finetuning. The baseline methods include random guess, RolX [20], Panther++ [59], and GraphWave [12]. The results are presented in Table 4. Compared to the in-place methods, such as Panther++, the self-supervised methods show competitive performances. Particularly, the proposed method show improvements compared to the state-of-the-art self-supervised methods.

4.3 Ablation Studies

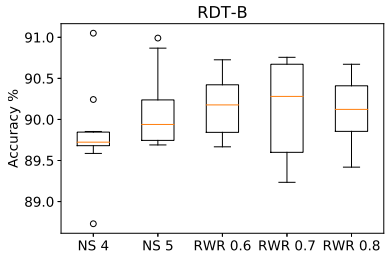

Figure 4: The model performance under different subgraph distributions. We test five different sampling settings for the second stage: for neighborhood sampling with different neighbor size, we consider NS 4 and NS 5; for random walk with restart with different restart probability, we consider RWR 0.6, RWR 0.7 and RWR 0.8. The results are based on 10-fold validation accuracy on RDT-B. Our model performs similar under different settings, which indicates our method is robust to distribution shift.

4.3.1 Model Robustness. : To evaluate the distributional robustness with the presence of distribution shift,we consider different sampling settings for building the structural embeddings, and the results are summarized in Fig. 4. We test five different sampling settings for the second stage: for neighborhood sampling with different neighbor size, we consider NS 4 and NS 5; for random walk with restart with different restart probability, we consider RWR 0.6, RWR 0.7 and RWR 0.8. The results are based on 10-fold validation accuracy on RDT-B. Different sampling settings will lead to different subgraph distributions. The results show that our method is insensitive to the choice of sampling settings.

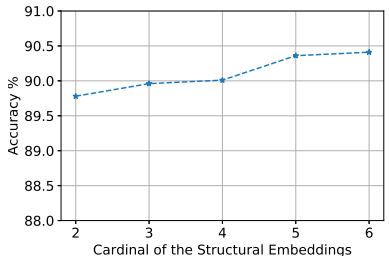

Figure 5: Model performance *v.s.* sampling size.

4.3.2 Cardinality of Embedding Set. : The first sampling stage in computing our structural embeddings is neighborhood sampling, and in the above experiments, we consider 5 neighbors within one-hop for each node of interest. In this section, we alter the value of C,

Table 1: The statistic for the pre-training datasets. Among these datasets, Academia, DBLP-SNAP, and DBLP-NetRep are academic datasets, and the rest are social networks.

Dataset	Academia	DBLP-SNAP	DBLP-NetRep	IMDB	Facebook	LiveJournal
# nodes	137,969	317,080	540,486	896,305	3,097,165	4,843,953
# edges	739,384	2,099,732	30,491,458	7,565,894	47,334,788	85,691,368

Table 2: Node Classification Results.

Dataset	US-Airport	H-index
# nodes	1190	5000
# edges	13,599	44,020
ProNE	62.3	69.1
GraphWave	60.2	70.3
Struc2vec	66.2	–
GCC-E2E	65.3	77.7
GCC-MoCo	65.8	76.1
GCC-rand*	63.6	77.2
GCC-E2E*	68.4	78.8
GCC-MoCo*	66.5	80.9
Proposed$^{\Delta}$	68.1	80.7
Proposed	**68.9**	**81.2**

Table 3: Graph Classification Results.

Dataset	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1000	1500	5000	2000	5000
# classes	2	3	3	2	5
avg.# nodes	19.8	13.0	74.5	429.6	508.5
DGK	67.1	44.7	73.4	78.0	40.8
Graph2Vec	71.1	50.2	–	76.9	48.2
InfoGraph	73.3	50.1	–	83.1	53.5
GCC-E2E	71.5	49.3	74.8	86.9	53.1
GCC-MoCo	72.4	49.3	79.0	90.1	53.4
DGCNN	70.2	48.0	73.7	–	–
GIN	**75.6**	51.5	80.2	89.4	54.5
GCC-rand*	75.5	51.0	78.6.	87.9	52.0
GCC-E2E*	71.2	47.6	79.1	86.1	48.6
GCC-MoCo*	73.3	50.5	81.1	88.0	53.2
Proposed$^{\Delta}$	74.9	53.0	80.7	89.9	55.4
Proposed	**75.6**	**53.1**	**81.2**	**90.4**	**55.9**

the cardinality of the embedding set, to show its relationship with the model performance. The results are summarized in Fig. 5. The results show that with the growth of the neighbor size, the model performance first increases then stay stable. The cardinality used in this paper is chosen to balance the computational efficiency and the performance.

4.3.3 Computational Time. : Our approach outperforms related baselines in several representative graph-related tasks. A potential disadvantage is that our approach requires longer computational time compared to related pretraining graph methods. Table. 5 describes the pretraining time for the baseline GCC and our approach using different subgraph size (denoted by superscripts). It can be observed that our approach takes moderately longer training time than GCC [33], and the per-step time is positive correlated with the size of subgraphs.

Table 4: Top-k Similarity Search ($k = 40$). Best-performing self-supervised methods are bold faced. Best performing non-self-supervised methods are denoted by *.

Dataset	K-I	S-C	S-I
|V|	2607	3548	2559
|E|	4774	7076	6668
# ground truth	697	874	898
Random	0.0566	0.0447	0.0521
RolX	0.1288	0.0984	0.1309
Panther++	0.1558	0.1185*	0.1320
GraphWave	0.1693*	0.0995	0.1470*
GCC-E2E	0.1564	**0.1247**	0.1336
GCC-MoCo	0.1521	0.1178	0.1425
Proposed	**0.1588**	0.1191	**0.1439**

Table 5: The comparison of per-step time for the baseline and our method using different subgraph size (denoted by superscripts).

Method	Time (ms)
GCC MoCo	4.31
Proposed^3	5.87
Proposed^4	5.93
Proposed^5	6.16

5 CONCLUSION

In this paper, we propose a robust self-supervised graph neural network. We design structural embeddings to explicitly represent both the nodes of interest and their neighbors and utilize 1-Wasserstein distance to characterize network proximity. We formulate an asymptotically distributionally robust contrastive learning framework, which is reluctant to distributional shift. We exploit a differentiable optimization framework to compute the network proximity, which makes our model end-to-end trainable. The experimental results demonstrate that our approach outperforms state-of-the-art baselines in various downstream graph analyses and our design is effective in improving model robustness.

ACKNOWLEDGEMENTS

This work was partially supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, 2040588.

REFERENCES

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. 2019. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430* (2019).

[2] Brandon Amos and J Zico Kolter. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*. PMLR, 136–145.

[3] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 44–54.

[4] Shane Barratt. 2018. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098* (2018).

[5] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 891–900.

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366* (2018).

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[9] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. 2020. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029* (2020).

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.

[11] Ailin Deng and Bryan Hooi. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4027–4035.

[12] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1320–1329.

[13] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc., 2224–2232. https://proceedings.neurips.cc/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf

[14] Daniel R Figueiredo, Leonardo Filipe Rodrigues Ribeiro, and Pedro HP Saverese. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada*. 13–17.

[15] Hongchang Gao and Heng Huang. 2018. Deep attributed network embedding. In *Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*).

[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*. PMLR, 1263–1272.

[17] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216* (2017).

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

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[20] Keith Henderson et al. 2012. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1231–1239.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[23] Steven G Krantz and Harold R Parks. 2012. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media.

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

[25] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602* (2018).

[26] Tomas Mikolov et al. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[27] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[28] Seth A Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. 2014. Information network or social network? The structure of the Twitter follow graph. In *Proceedings of the 23rd International Conference on World Wide Web*. 493–498.

[29] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).

[30] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[31] Marco Pennacchiotti and Siva Gurumurthy. 2011. Investigating topic models for social media user recommendation. In *Proceedings of the 20th international conference companion on World wide web*. 101–102.

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

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

[34] Scott C Ritchie et al. 2016. A scalable permutation approach reveals replication and preservation patterns of network modules in large datasets. *Cell systems* 3, 1 (2016), 71–82.

[35] Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. 2021. GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification. In *Proceedings of the Web Conference 2021*. 3733–3744.

[36] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, 9 (2011).

[37] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*. PMLR, 488–495.

[38] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2019).

[39] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Qing Meng, Wang Han, and Jiuxin Cao. 2021. Multi-level hyperedge distillation for social linking prediction on sparsely observed networks. In *Proceedings of the Web Conference 2021*. 2934–2945.

[40] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*. PMLR, 1139–1147.

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

[42] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *Proceedings of the Sixth International Conference on Data Mining (ICDM '06)*. IEEE Computer Society, USA, 613–622. https://doi.org/10.1109/ICDM.2006.70

[43] Johan Ugander, Lars Backstrom, Cameron Marlow, and Jon Kleinberg. 2012. Structural diversity in social contagion. *Proceedings of the National Academy of Sciences* 109, 16 (2012), 5962–5966.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

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

[46] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[47] Wei Wu, Bin Li, Chuan Luo, and Wolfgang Nejdl. 2021. Hashing-Accelerated Graph Neural Networks for Link Prediction. In *Proceedings of the Web Conference 2021*. 2910–2920.

[48] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3733–3742.

[49] Qianqian Xie, Jimin Huang, Pan Du, Min Peng, and Jian-Yun Nie. 2021. Graph topic neural network for document representation. In *Proceedings of the Web Conference 2021*. 3055–3065.

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

[51] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1365–1374.

[52] Rex Ying et al. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.
[53] Zhitao Ying et al. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*. 4805–4815.
[54] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. *arXiv preprint arXiv:2106.07594* (2021).
[55] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020).
[56] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. In *Proceedings of the Web Conference 2021*. 413–424.
[57] Chi Zhang et al. 2020. DeepEMD: Differentiable Earth Mover's Distance for Few-Shot Learning. *arXiv e-prints* (2020), arXiv–2003.
[58] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, et al. 2019. Oag: Toward linking large-scale heterogeneous entity graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2585–2595.
[59] Jing Zhang et al. 2015. Panther: Fast top-k similarity search on large networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1445–1454.
[60] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning.. In *IJCAI*, Vol. 19. 4278–4284.
[61] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[62] Yu Zheng, Chen Gao, Liang Chen, Depeng Jin, and Yong Li. 2021. DGCN: Diversified Recommendation with Graph Convolutional Networks. In *Proceedings of the Web Conference 2021*. 401–412.
[63] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*. 2848–2857.
[64] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*. 2069–2080.