KGNN: Harnessing Kernel-based Networks for Semi-supervised Graph Classification

Wei Ju
School of Computer Science,
Peking University
juwei@pku.edu.cn

Junwei Yang
School of Computer Science,
Peking University
yjwtheonly@pku.edu.cn

Meng Qu
Mila - Québec AI Institute,
Université de Montréal
meng.qu@umontreal.ca

Weiping Song
School of Computer Science,
Peking University
weiping.song@pku.edu.cn

Jianhao Shen
School of Computer Science,
Peking University
jhshen@pku.edu.cn

Ming Zhang*
School of Computer Science,
Peking University
mzhang_cs@pku.edu.cn

ABSTRACT

This paper studies semi-supervised graph classification, which is an important problem with various applications in social network analysis and bioinformatics. This problem is typically solved by using graph neural networks (GNNs), which yet rely on a large number of labeled graphs for training and are unable to leverage unlabeled graphs. We address the limitations by proposing the Kernel-based Graph Neural Network (KGNN). A KGNN consists of a GNN-based network as well as a kernel-based network parameterized by a memory network. The GNN-based network performs classification through learning graph representations to *implicitly* capture the similarity between query graphs and labeled graphs, while the kernel-based network uses graph kernels to *explicitly* compare each query graph with all the labeled graphs stored in a memory for prediction. The two networks are motivated from complementary perspectives, and thus combing them allows KGNN to use labeled graphs more effectively. We jointly train the two networks by maximizing their agreement on unlabeled graphs via posterior regularization, so that the unlabeled graphs serve as a bridge to let both networks mutually enhance each other. Experiments on a range of well-known benchmark datasets demonstrate that KGNN achieves impressive performance over competitive baselines.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Neural networks**; **Semi-supervised learning settings**.

KEYWORDS

Graph Classification, Graph Neural Networks, Graph Kernels, Semi-supervised Learning

*Corresponding author

ACM Reference Format:
Wei Ju, Junwei Yang, Meng Qu, Weiping Song, Jianhao Shen, and Ming Zhang. 2022. KGNN: Harnessing Kernel-based Networks for Semi-supervised Graph Classification. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498429

1 INTRODUCTION

Graph-structured data are ubiquitous in a wide range of domains. Examples include social networks [31], biological reaction networks [28], molecules [34]. For graph-structured data, one fundamental problem is graph classification, which aims at analyzing and predicting the property of the entire graph. Such a problem has various downstream applications, including predicting the properties of molecules [13] and analyzing the functionality of compounds [19].

Graph classification is typically formalized as a supervised learning task, and many recent works propose to use graph neural networks (GNNs) [23, 24, 40] to solve the problem. The basic idea is to learn effective graph representations with nonlinear message passing schemas. At each step, a node receives messages from all its neighbors, which are further aggregated to update the node representation. Finally, a readout function is applied to integrate all the node representations into a representation of the whole graph. With this shared message passing framework, the learned graph representations can implicitly capture the similarity between query graphs and labeled graphs in the latent space. Despite the good performance, GNNs usually require a large amount of labeled data for training and fail to leverage unlabeled data. However, data annotation often requires domain experts, which is highly expensive, especially in specific domains such as biomedicine [13].

This motivates us to study semi-supervised graph classification, i.e., using both labeled and unlabeled data for graph classification. The unlabeled data serve as a regularizer, which helps a model better explore the inherent graph semantic information even with a limited amount of labeled data. Indeed, there are a handful of works along this line [13, 24, 34], and they typically employ semi-supervised learning techniques to train GNN models. These approaches usually integrate self-training [22] or knowledge distillation [14] into GNNs. However, these methods suffer from two key limitations: **(1) Unable to well explore graph similarity**. Graph classification relies on comparing the query graphs with labeled graphs. Existing methods [13, 24, 34] typically compute the similarity of graphs in an implicit way by projecting graphs into a latent space with a GNN encoder. However, such implicit methods often cannot well explore the similarity of graphs. **(2) Suffering from labeled data scarcity**. Besides, experimental results show that the performance of existing methods is still unsatisfactory especially when labeled data are very scarce. The reason is that these methods are not able to obtain high-quality annotated data to improve model training. Therefore, we are looking for an approach that is able to better consider the relationship among graphs and meanwhile overcome the challenge of scarce labeled data.

In this paper, we propose such a method called the Kernel-based Graph Neural Network (KGNN). The key idea of KGNN is to enhance GNNs with graph kernels, which are able to explicitly measure the graph similarity of graphs. To leverage graph kernels effectively, we introduce two modules in a KGNN, i.e., a GNN-based network and a kernel-based network. The GNN-based network is parameterized by existing GNNs, which uses message passing mechanisms to learn useful graph representations for graph classification. In contrast, the kernel-based network employs a memory network, where the memory stores the given labeled graphs. Given a query graph, we compare it with all the graphs in the memory using graph kernels, and further integrate the labels of the most similar graphs to predict the label of the query graph.

The GNN-based network and kernel-based network explore graph similarity from different angles, i.e., message passing and graph kernels respectively. Although they are naturally complementary, how to jointly train both networks which enables us to distill the knowledge between each other is nontrivial. We solve the problem with a novel posterior regularization framework [7], which encourages both networks to collaborate with each other and maximize their agreement on unlabeled data. Each training iteration of posterior regularization consists of two steps. In the first step, the kernel-based network is updated by projecting the GNN-based network into a regularized subspace, yielding a stronger kernel-based network. In the second, we distill the knowledge learned by the kernel-based network into the GNN-based network, so that allowing the GNN to better explore graph similarity and overcome the challenge caused by the scarcity of labeled data.

To summarize, the main contributions of this work are as follows:

- We propose a novel approach for semi-supervised graph classification, which consists of a GNN-based network and a kernel-based network to fully capture the graph similarity and overcome the scarcity of labeled data.
- We develop a novel posterior regularization framework to combine the advantages of graph neural networks and graph kernels in a principled way, such that they can mutually enhance each other via optimizing the two modules with an EM-style algorithm.
- We conduct extensive experiments on a range of well-known benchmarks to demonstrate the effectiveness of our KGNN.

2 RELATED WORK

2.1 Graph Neural Networks

Our work is related to graph neural networks (GNNs), which are able to learn useful graph representations via supervised learning.

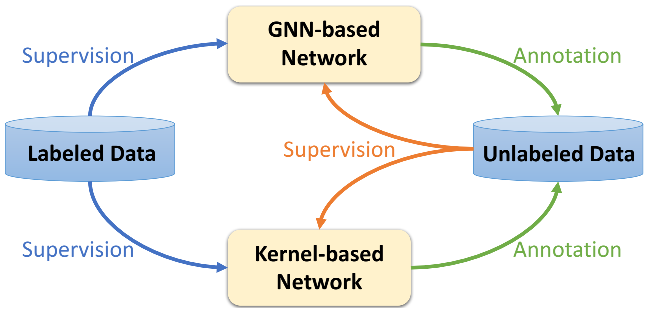


Figure 1: Illustration of the framework. KGNN consisting of a GNN-based network and a kernel-based network receives supervision from labeled data to annotate the unlabeled data, which improve both modules in turn. Two modules are jointly optimized and mutually enhance each other.

Most existing methods [16, 23, 38, 40] inherently use a message-passing neural network [10] to learn node representations, and then aggregate them as graph representations. Because of their effectiveness in learning graph representations, they achieve state-of-the-art results. However, graph representations derived from GNNs often fail to effectively leverage graph similarity for prediction. Besides, these methods rely on a large number of labeled data for training. With KGNN, besides learning effective graph representations derived from GNNs, we also benefit from graph kernels to explicitly incorporate graph similarity in a semi-supervised framework.

2.2 Graph Kernels

Kernels on graphs are originally introduced by [9, 15], and the basic idea is to decompose graphs into atomic substructures and use kernel functions to capture the graph similarity, which can be used in kernel methods for graph classification. Many early methods[9, 15] define feature vectors as counts of label paths using a random walk or shortest path. However, these methods cannot be scaled to large graphs and suffer from high computational complexity. To alleviate these weaknesses, many later graph kernels are designed based on limited-sized substructures [30] or neighborhood aggregation [29]. Some recent studies also try to combine graph kernels and graph neural networks [4, 6]. GCKNs [4] connect two methods via multilayer graph kernels. GNTKs [6] prove that their kernels are equivalent to infinitely wide GNNs trained by gradient descent. Similar to these methods, our proposed KGNN also combines the advantages of both worlds but from different views. To be specific, our approach not only imposes graph kernels as structured constraints to guide the training process of GNNs, but also uses unlabeled data to assist model training by maximizing the agreement of the two networks on unlabeled data, while their works can only leverage labeled data in a supervised way.

2.3 Semi-supervised Learning

Another category of related work is semi-supervised learning. The self-training method is one of the earliest ideas along this line. It requires a trained classifier to periodically predict class labels of the unlabeled data and add high-quality classified samples to the training set. For example, the entropy minimization method [11] requires the classifier to output low-entropy predictions on unlabeled data. Another line of research focuses on perturbation-based approaches. The core idea is to encourage the mapping consistency between the input and output when noise is applied to the input, which is so-called consistency regularization. Among them, the temporal ensembling model [21] maintains an exponential moving average of label predictions while the mean teacher model [35] averages network parameters to obtain a stable target output.

Besides, there are also some recent studies [13, 24, 34] proposed for semi-supervised graph classification. SEAL-AI [24] approaches the problem in the perspective of the hierarchical graph related to self-training. InfoGraph [34] and ASGN [13] learn graph representations via contrastive learning and active learning respectively, which both can be boiled down to the teacher-student framework. Compared with existing works, our work focuses on combining the advantages of graph neural networks and graph kernels via posterior regularization, incorporating graph similarity from implicit and explicit perspectives respectively, while their works fail to enhance GNNs from the view of topological similarity.

3 PROBLEM DEFINITION & PRELIMINARY

3.1 Problem Definition

Definition: Semi-supervised Graph Classification. Let $G = (V, E, \mathbf{X})$ represent a graph, where V is the set of nodes, E is the set of edges. We use $\mathbf{X} \in R^{|V| \times d}$ denotes the node feature matrix, where d is the dimension of features. For semi-supervised graph classification, let $\mathcal{G} = \{\mathcal{G}^L, \mathcal{G}^U\}$ denote a set of graphs, in which $\mathcal{G}^L = \left\{G_1, \cdots, G_{|\mathcal{G}^L|}\right\}$ are labeled graphs and $\mathcal{G}^U = \left\{G_{|\mathcal{G}^L|+1}, \cdots, G_{|\mathcal{G}^L|+|\mathcal{G}^U|}\right\}$ are unlabeled graphs. Additionally, we use $\mathbf{y}^L$ to represent the labels corresponding to $\mathcal{G}^L$. The goal of semi-supervised graph classification is to learn a label distribution $p(\mathbf{y}^U|\mathcal{G}, \mathbf{y}^L)$, which can assign labels to unlabeled graphs $\mathcal{G}^U$.

3.2 Graph Kernels

Given two graphs G_i and G_j, the graph kernel $K(G_i, G_j)$ measures the similarity between them and is defined as:

$$K\left(G_i, G_j\right) = \sum_{u \in V_i} \sum_{v \in V_j} k_{base}\left(f_{G_i}(u), f_{G_j}(v)\right), \tag{1}$$

where the base kernel k_{base}, i.e., the inner product on Hilbert space, compares substructures centered at nodes v_i and v_j, and $f_G(v)$ is the feature vector counting the number of appearances of each substructure (e.g., subtrees, paths, graphlets) in the graph G.

Afterward, the matrix of pairwise similarities among graphs defined by graph kernels can be passed to kernel-based methods such as the Support Vector Machine to perform graph classification.

Weisfeiler-Lehman Kernel. Inspired by the Weisfeiler-Lehman test of graph isomorphism [37], Weisfeiler-Lehman kernel [29] follows the idea of the iterative relabeling process to augment the label of each node through the sorted set of labels of its neighbors in each iteration. The augmented labels are further compressed into a new label. Then the final feature for each graph is the concatenation of label counts for each iteration.

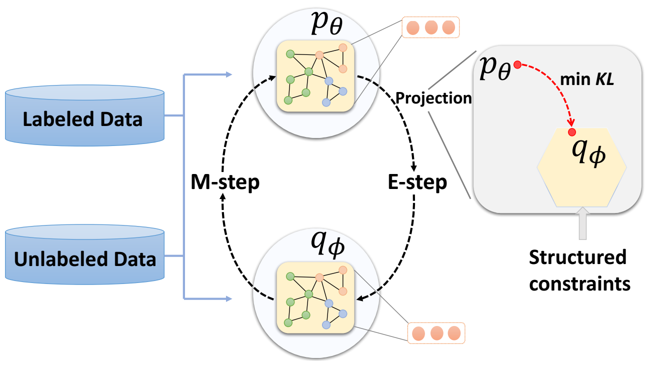


Figure 2: Framework overview. Yellow squares are the entire graph for classification. Orange triple circles are graph representations. KGNN consists of a GNN-based network p_θ and a kernel-based network q_ϕ, which is trained by alternating between an E-step and an M-step. In the E-step, the kernel-based network q_ϕ is updated by projecting the GNN-based network p_θ to a regularized subspace (red dashed arrow). In the M-step, the GNN-based network p_θ is updated from the knowledge distilled by the kernel-based network q_ϕ.

3.3 Posterior Regularization

Posterior regularization (PR) [7] is a principled framework to impose constraints from the desired distribution $q(\mathbf{y})$ on probabilistic models $p_\theta(\mathbf{y}|\mathbf{x})$ as follows:

$$\mathcal{L}(q; \boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) - \min_{q \in Q} \mathrm{KL}\left(q(\mathbf{y}) \| p_\theta(\mathbf{y}|\mathbf{x})\right), \tag{2}$$

where $\mathcal{L}(\boldsymbol{\theta})$ is the log-likelihood of model $p_\theta(\mathbf{y}|\mathbf{x})$, and Q is a constraint set, with respect to the expectations of constraint features $\psi(\mathbf{x}, \mathbf{y})$ that are bounded by $\mathbf{b}$:

$$Q = \left\{q(\mathbf{y}) : \mathbb{E}_q[\psi(\mathbf{x}, \mathbf{y})] \le \mathbf{b}\right\}. \tag{3}$$

To optimize the posterior regularized likelihood $\mathcal{L}(q; \boldsymbol{\theta})$, PR presents an iterative procedure akin to an EM-style algorithm:

$$\mathrm{E}: q^{t+1} = \arg\min_{q \in Q} \mathrm{KL}(q(\mathbf{y}) \| p_{\theta^t}(\mathbf{y}|\mathbf{x})), \tag{4}$$

$$\mathrm{M}: \boldsymbol{\theta}^{t+1} = \arg\max_{\theta} \mathcal{L}(\boldsymbol{\theta}) + \mathbb{E}_{q^{t+1}}[\log p_\theta(\mathbf{y}|\mathbf{x})]. \tag{5}$$

For better interpretation, it can be viewed as performing coordinate ascent on $\mathcal{L}(q; \boldsymbol{\theta})$. Starting from an initial parameter estimate θ^0, the algorithm iterates two block-coordinate ascent steps until a convergence criterion is reached.

4 KGNN: KERNEL-BASED GRAPH NEURAL NETWORK

4.1 Overview

In this paper, we introduce our approach for semi-supervised graph classification by incorporating both labeled and unlabeled data. Existing methods typically combine graph neural networks with semi-supervised learning techniques. However, these methods are not able to fully leverage graph similarity information for classification, which leads to insufficient expressiveness. Also, their performance is not yet satisfactory when the labeled data are limited.

We address the above limitations by proposing the Kernel-based Graph Neural Network (KGNN), which combines methods based on graph kernels and graph neural networks. The idea is to use graph kernels as structured constraints to guide the training of graph neural networks via posterior regularization, allowing the graph neural network to explicitly exploit graph similarity and receive additional supervision to boost performance. In particular, there are a kernel-based network and a GNN-based network in a KGNN, and we use an EM-style algorithm to jointly train both networks. In the E-step, the kernel-based network represented as a memory network is updated by projecting the GNN-based network to a subspace constrained by the graph similarity. In the M-step, the GNN-based network is improved by distilling the knowledge from the kernel-based network. In this way, both networks can effectively mutually enhance each other. An illustration of the framework is presented in Figure 2. Next, we first introduce the GNN-based network and the kernel-based network. Finally, the training algorithm is explained.

4.2 GNN-based Network

Due to the appealing representation learning ability, graph neural networks have been widely used in graph classification tasks. For example, InfoGraph [34] first learns node representations with GIN [38], and then summarizes the node representations into the graph-level representation, which serves as graph features for the classification task. Note that in InfoGraph [34], the label dependency between different graphs is ignored. We follow the same idea and factorize the joint label distribution as:

$$p(\mathbf{y}^U|\mathcal{G}, \mathbf{y}^L) = \prod_{g \in \mathcal{G}^U} p(\mathbf{y}^g|\mathcal{G}, \mathbf{y}^L). \tag{6}$$

Furthermore, to learn graph-level representation for a graph $G = (V, E, \mathbf{X})$, GNN-based methods solely rely on the node attributes $\mathbf{X}$ and graph structure E. In other words, it is independent of other graphs, we therefore define the conditional probability of label $\mathbf{y}^g$ for graph G as follows:

$$p(\mathbf{y}^g|\mathcal{G}, \mathbf{y}^L) = p(\mathbf{y}^g|\mathbf{X}, E). \tag{7}$$

To predict the graph label $\mathbf{y}^g$, a graph neural network first learns node representations by aggregating and updating messages from its neighborhoods. Based on the node representations, a READOUT function summarizes them into the whole graph representation. This procedure can be formalized in the following way:

$$\begin{aligned} \mathbf{h}_v^{(l)} &= U_\theta^{(l)}\left(\mathbf{h}_v^{(l-1)}, A_\theta^{(l)}\left(\left\{\mathbf{h}_u^{(l-1)}\right\}_{u \in \mathcal{N}(v)}\right)\right), \\ H(G) &= \text{READOUT}\left(\left\{\mathbf{h}_v^L\right\}_{v \in V}\right), \\ p_\theta(\mathbf{y}^g|\mathbf{X}, E) &= \text{MLP}_\theta\left(H(G)\right), \end{aligned} \tag{8}$$

where $\mathbf{h}_v^{(l)}$ is the feature vector of node v at the l-th layer. $\mathbf{h}_v^{(0)}$ is often initialized as node features (i.e., $\mathbf{X}_v$) and $\mathcal{N}(v)$ are neighborhoods to node v. The message-passing phase runs for L steps and is defined in terms of aggregation functions $A_\theta^{(l)}(\cdot)$ and node updating functions $U_\theta^{(l)}(\cdot)$, which have a series of possible implementations [12, 18, 36]. After a permutation invariant function

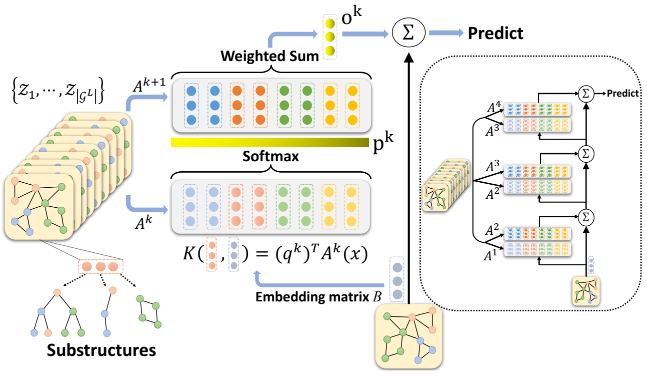


Figure 3: Illustration of the kernel-based network q_ϕ. It is parameterized by a memory network that incorporates the graph kernels to capture the topological similarity information of graphs. The multi-hop mechanism helps in learning correlations between memories. For simplicity, we omit the subscript ϕ of embedding matrix A_ϕ and B_ϕ.

READOUT$(\cdot)$ such as some pooling functions [16, 23, 40], the final graph-level representation $H(G)$ is fed into an MLP$_\theta$ (Multi-Layer Perceptron) classifier for classification. In this paper, we follow InfoGraph [34] and use GIN [38] to learn node representations for its high expressive power, and then use the sum operation to produce the graph-level representations (i.e., the READOUT function).

4.3 Kernel-based Network

Graph neural networks can effectively utilize node attributes and graph local structures under the message-passing framework, however, they are insufficient in capturing more global graph topology and ignore the relations between different graphs. This motivates us to additionally incorporate graph kernels, which have been proven useful for encoding graph topology and similarity information [20].

Traditional kernel-based methods perform graph classification tasks in a two-step approach that similarity features are separated from the training phase and are not optimized for downstream tasks, hence the performance is usually unsatisfactory. To address the above limitation, we propose a kernel-based network $q(\mathbf{y}^U|\mathcal{G}, \mathbf{y}^L)$ and implement it with a memory network, which shares a similar idea to classical kernel-based methods[27], but haves larger model capacity and can be optimized in an end-to-end manner. Moreover, memory networks can be naturally and effectively coupled with kernel tricks [8] and introduced to further compare graph-graph similarity via graph kernels for prediction.

In detail, with the assumption of label dependency in Eq. 6, we model the conditional probability $q(\mathbf{y}^g|\mathcal{G}, \mathbf{y}^L)$ as follows

$$q_\phi(\mathbf{y}^g|\mathcal{G}, \mathbf{y}^L) = \text{MemNN}_\phi(\mathcal{G}), \tag{9}$$

where the memories of MemNN$_\phi$ consists of the labeled data set $\mathcal{G}^L$ represented as the feature vector set $\{z_1, \cdots, z_{|\mathcal{G}^L|}\}$ derived from graph kernels (see. Sec. 3.2). Then each graph are converted into memory vectors by a set of trainable embedding matrix set $A_\phi = \{A_\phi^1, \dots, A_\phi^{K+1}\}$, where each A_ϕ^k maps memory content into continuous space at each layer k. The query graph (one of the labeled data) is also encoded via another trainable embedding matrix B_ϕ to obtain a hidden representation q^k as a reading head. Afterward, we measure the similarity between q^k and each memory vector $A^k_\phi(z_i)$ by inner product, corresponding to the graph kernel computed on Hilbert space $\mathcal{H}$ and then followed by a softmax function. The model loops over K hops and it computes the attention weights at hop k for each memory G_i using:

$$p^k_i = \text{Softmax}((q^k)^\top A^k_\phi(z_i)), \tag{10}$$

where $\text{Softmax}(x_i) = e^{x_i}/\Sigma_j e^{x_j}$. Here, p^k is a soft memory selector that decides the memory relevance with respect to q^k. Then, the model reads out the memory o^k by the weighted sum over A^{k+1}_ϕ [1],

$$o^k = \sum_i p^k_i A^{k+1}_\phi(z_i). \tag{11}$$

Then, the query vector is updated for the next hop by using $q^{k+1} = q^k + o^k$. The above step leads to the final graph representation o^K, which will be fed into an MLP_ϕ classifier for prediction as follows:

$$q_\phi(\mathrm{y}^g|\mathcal{G}, \mathrm{y}^L) = \text{MLP}_\phi\left(o^K\right). \tag{12}$$

In this way, the representation of graphs can be combined with similarity features of multiple graphs from memories, and the multi-hop attention mechanism helps in learning correlations between memories. Thus, the derived kernel-based network enables high flexibility and is able to effectively exploit graph topology similarity. An illustration of the kernel-based network is presented in Figure 3.

4.4 Optimization Framework with PR

Graph kernels offer effective prior knowledge to explicitly capture graph similarity. To inject this knowledge into graph neural networks efficiently, posterior regularization (PR) is utilized to provide a principled way to impose structured constraints on the posterior distribution of the probabilistic models. However, PR typically uses a fully-specified constraint (see Eq. (3)), which is often sub-optimal. To address this limitation, we represent the desired distribution $q(\mathbf{y})$ in (2) as the kernel-based network described in Section 4.3, where this manipulation leads to a more efficient and flexible training algorithm. For simplicity, the GNN-based network and the kernel-based network will be simply written $p_\theta(\mathbf{y}|\mathbf{x})$ and $q_\phi(\mathbf{y}|\mathbf{x})$ respectively with an abuse of notation in the rest of the paper.

Based on this, the kernel-based network enables learnable constraints and should be a good complement for the GNN-based network to incorporate the topology information. To this end, we formulate the training objective as:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}(\theta) - \min_\phi \text{KL}\left(q_\phi(\mathbf{y}|\mathbf{x}) \| p_\theta(\mathbf{y}|\mathbf{x})\right). \tag{13}$$

Note that another benefit of modeling the kernel-based network as the memory network is the differentiability of our training objective Eq. (13). It is easy to use stochastic gradient descent algorithms to optimize model parameters. Next, we will give the specific training algorithm for updating parameters in Section 4.5.

[1]Here is A^{k+1}_ϕ since we use adjacent weighted tying [33].

Algorithm 1 Joint Learning Algorithm of KGNN

```
Input: Labeled data $\mathcal{G}^L$, unlabeled data $\mathcal{G}^U$.
Parameter: Model parameter $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$
Output: Jointly learned $p_\theta(\mathbf{y}|\mathbf{x})$ and $q_\phi(\mathbf{y}|\mathbf{x})$.
1: Initialize model parameter $\boldsymbol{\theta}$ on labeled data $\mathcal{G}^L$.
2: while not convergence do
3:     // E-step
4:     Annotate unlabeled data $\mathcal{G}^U$ with $p_\theta$.
5:     Optimize model parameter $\boldsymbol{\phi}$ with Eq. (15).
6:     // M-step
7:     Annotate unlabeled data $\mathcal{G}^U$ with $q_\phi$.
8:     Optimize model parameter $\boldsymbol{\theta}$ with Eq. (16).
9: end while
```

4.5 Training Algorithm

In the training phase, our goal is to maximize the posterior regularized likelihood (13) that can be optimized by an iterative procedure akin to an EM-style algorithm, alternated by an E-step and an M-step. In the E-step, the goal is to optimize the kernel-based network by minimizing the KL divergence between the kernel-based network and the GNN-based network. In the M-step, we optimize the GNN-based network by the original likelihood of model $p_\theta(\mathbf{y}|\mathbf{x})$ and additional guidance derived from the kernel-based network. Next, we introduce the details of the E-step and M-step.

E-step. In this step, we optimize the parameter ϕ. Recall that the optimization of the kernel-based network parameter ϕ is performed by minimizing the KL divergence in Eq. (4). However, directly optimizing the likelihood function requires dealing with the partition function in q_ϕ, which can be difficult. To address the restriction, we propose to instead minimize the reverse KL divergence:

$$\begin{aligned} &\min_\phi \text{KL}(p_\theta(\mathbf{y}|\mathbf{x}) \| q_\phi(\mathbf{y}|\mathbf{x})) \\ &= \min_\phi -\mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})}[\log q_\phi(\mathbf{y}|\mathbf{x})] + const. \end{aligned} \tag{14}$$

Besides, we notice that q_ϕ is a standard discriminative classifier and can be also trained with labeled data. Therefore, we optimize q_ϕ by maximizing the following objective function:

$$\begin{aligned} \boldsymbol{\phi} = \arg\max_\phi &\sum\nolimits_{(\mathbf{x},\mathbf{y})\in L} \log q_\phi(\mathbf{y}|\mathbf{x}) \\ &+ \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})}[\log q_\phi(\mathbf{y}|\mathbf{x})]. \end{aligned} \tag{15}$$

M-step. In this step, we optimize the parameter θ. More specifically, besides optimizing the original likelihood of model p_θ, we will fix q_ϕ and further update p_θ. Afterwards, the parameter θ can be updated by maximizing the following overall objective function:

$$\begin{aligned} \boldsymbol{\theta} = \arg\max_\theta &\sum\nolimits_{(\mathbf{x},\mathbf{y})\in L} \log p_\theta(\mathbf{y}|\mathbf{x}) \\ &+ \mathbb{E}_{q_\phi(\mathbf{y}|\mathbf{x})}[\log p_\theta(\mathbf{y}|\mathbf{x})]. \end{aligned} \tag{16}$$

We summarize the detailed optimization algorithm for KGNN in Algorithm. 1. The model alternates between optimizing the objective function with regard to ϕ and θ, which is referred to as E-step and M-step. During each step, both labeled and unlabeled data are leveraged for training. In addition to training two networks with labeled data, we essentially select samples with high confidence from the unlabeled data for regularization to let both networks mutually enhance each other. Specifically, given the unlabeled data $\mathbf{x}$ and their predicted categories $\mathbf{y}$, we rank these data according to the probability of distribution $\mathbf{y}$ (i.e., $q_\phi(\mathbf{y}|\mathbf{x})$ and $p_\theta(\mathbf{y}|\mathbf{x})$) and then select top-$k$ data from the corresponding ranked list. This is similar to the way we do in self-training methods [22]. Furthermore, to eliminate the influence of incorrect annotations in selected samples, we collect the intersection of the training samples predicted via both networks and regard them as the additional labeled set to enlarge the labeled data, experimental analysis in the next section shows that consistent prediction data are helpful for improving the final performance. The whole iterative procedure stops until convergence or the unlabeled data are exhausted.

5 EXPERIMENT

To verify the effectiveness of our proposed method, we first introduce the experimental settings and then present the detailed experiment results and performance analysis.

5.1 Experimental Settings

5.1.1 Evaluation Datasets. We conduct extensive experiments on seven widely-used semi-supervised graph classification datasets, including two bioinformatics datasets (i.e., PROTEINS [3] and DD [5]), four social network datasets (i.e., IMDB-B, IMDB-M, REDDIT-B, REDDIT-M-5k [39]) and one scientific collaboration dataset (i.e., COLLAB [39]). We summarize the statistics of datasets in Table 1. Following InfoGraph [34], we use all-ones encoding as input node features when node attributes are not available in the datasets.

Data Preparation. As these datasets contain labels for all graphs, we need to construct the datasets applicable for semi-supervised learning scenarios. For each dataset, we first split graphs into TRAIN, VAL and TEST sets according to a 7:1:2 ratio. Based on the TRAIN data, we further sample 2/7 of the graphs as labeled data, which is denoted as TRAIN-L and use the rest as unlabeled data TRAIN-U. We use the validation data VAL for hyper-parameter selection and report the results on TEST data set.

5.1.2 Compared Methods. We compare our KGNN with the following methods, which can be divided into three categories: traditional graph methods, traditional semi-supervised learning methods and graph-specific semi-supervised learning methods. Traditional graph methods include Graphlet Kernel (GK) [30], Shortest Path Kernel (SP) [2], Weisfeiler-Lehman Kernel (WL) [29], Deep Graph Kernel (DGK) [39], Sub2Vec [1] and Graph2Vec [26]. Traditional semi-supervised learning methods include EntMin [11], Π-Model [35], Mean-Teacher [35] and VAT [25]. Graph-specific semi-supervised methods include InfoGraph [34], ASGN [13], GraphCL [42] and JOAO [41]. Note that we don't compare with SEAL-AI [24] because it requires explicit relations among the graph instances, which are not available in our datasets.

5.1.3 Parameter Settings. For the proposed KGNN, we use GIN [38] to parameterize the GNN-based network p_θ, consisting of three graph convolutional layers and one sum-pooling layer, followed by the softmax function, which is the same as InfoGraph [34]. The kernel-based network q_ϕ is implemented as a memory network with 3 hops, in which the Weisfeiler-Lehman subtree kernel [29] serves as the base graph kernel. For a fair comparison, we set the batch size to 32 and the number of epochs to 20 for all datasets. The embedding dimensions of hidden layers are set as 32 for bioinformatics datasets while 64 for social network and scientific collaboration datasets. We apply the dropout [32] with a ratio of 0.5 in our KGNN model. In each iteration, p_θ and q_ϕ are trained using Adam optimizer [17] where initial learning rate is 0.01 and weight decay is 0.0005.

Table 1: Statistics of the evaluation datasets.

Datasets	Graph Num.	Avg. Nodes	Avg. Edges	Classes
PROTEINS	1113	39.06	72.82	2
DD	1178	284.32	715.66	2
IMDB-B	1000	19.77	96.53	2
IMDB-M	1500	13.00	65.94	3
REDDIT-B	2000	429.63	497.75	2
REDDIT-M-5k	4999	508.52	594.87	5
COLLAB	5000	74.49	2457.78	3

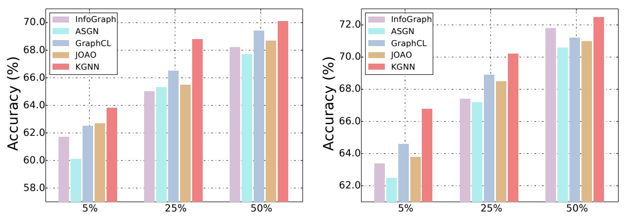
5.2 Results

We report the quantitative results of semi-supervised graph classification with half of the labeled data TRAIN-L in Table 2. From this table, we make the following observations. (1) We can observe that most of the traditional graph methods perform worse than other methods, which shows that graph neural networks possess the strong representation-learning ability to extract more useful information from graph-structured data. (2) The methods with traditional semi-supervised learning techniques (EntMin, Π-Model, Mean-Teacher and VAT) show worse performance compared with the graph-specific semi-supervised learning approaches, indicating that recent graph semi-supervised learning techniques are more suitable for the challenging graph classification task. (3) Among the previous state-of-the-art approaches, GraphCL achieves almost the best performance on most datasets. Specifically, other graph-specific semi-supervised learning approaches are not as effective as GraphCL, maybe the reason is that GraphCL takes advantage of the idea of instance discrimination for contrastive learning and specific graph augmentation strategies, and thus further improve the performance. (4) Our framework KGNN achieves the best performance on six of seven datasets, which demonstrates the effectiveness of our framework. From the results, we attribute the performance gain to two aspects: (i) Introduction of graph kernels. Graph kernels acting as structured constraints are able to guide the training process of graph neural networks. (ii) Introduction of posterior regularization framework. Instead of simply integrating the graph neural networks and graph kernels, we use a principled way to combine both worlds, which can mutually enhance each other.

Performance on Different Amounts of Labeled Data. We vary rates of the labeled data for training to show the performance of different models in Figure 4. We take the PROTEINS and IMDB-B as examples. Here we omit the traditional methods since they do not have competitive performance. From the results, we find that the performance of all the methods further improves as the number of available labeled data increases, demonstrating that adding more labeled data is an efficient approach to boost the

Table 2: Results on seven benchmark datasets. We present the mean together with the standard deviation of prediction accuracy over five runs (in %) using different random seeds. The results in bold indicate the best performance.

Methods	Datasets						
	PROTEINS	DD	IMDB-B	IMDB-M	REDDIT-B	REDDIT-M-5k	COLLAB
GK [30]	64.8 ± 2.3	53.2 ± 1.4	54.5 ± 1.7	32.3 ± 2.4	57.8 ± 2.7	34.3 ± 0.8	55.7 ± 1.1
SP [2]	65.2 ± 2.6	55.3 ± 2.1	52.0 ± 1.6	37.7 ± 1.9	68.3 ± 3.7	30.4 ± 1.3	64.1 ± 1.3
WL [29]	63.5 ± 1.6	57.3 ± 1.2	58.1 ± 2.3	33.3 ± 1.4	61.8 ± 1.3	37.0 ± 0.9	62.9 ± 0.7
DGK [39]	64.4 ± 1.7	60.5 ± 0.8	55.6 ± 2.2	34.6 ± 1.3	66.2 ± 2.4	36.5 ± 2.4	61.3 ± 1.2
Sub2Vec [1]	52.7 ± 4.5	46.4 ± 3.2	44.9 ± 3.5	31.8 ± 2.7	63.5 ± 2.3	35.1 ± 1.5	60.8 ± 1.4
Graph2Vec [26]	63.1 ± 1.8	53.7 ± 1.6	61.2 ± 2.6	38.1 ± 2.2	67.7 ± 2.3	38.1 ± 1.4	63.6 ± 0.9
EntMin [11]	62.7 ± 2.7	59.8 ± 1.3	67.1 ± 3.7	37.4 ± 1.2	66.9 ± 3.5	38.7 ± 2.8	63.8 ± 1.6
Π-Model [35]	63.2 ± 1.2	61.8 ± 1.8	67.0 ± 3.4	39.0 ± 3.5	67.1 ± 2.9	39.0 ± 1.1	63.7 ± 1.0
Mean-Teacher [35]	64.3 ± 2.1	60.6 ± 1.8	66.4 ± 2.7	38.8 ± 3.6	68.7 ± 1.3	39.2 ± 2.1	63.6 ± 1.4
VAT [25]	64.1 ± 1.2	59.9 ± 2.6	67.2 ± 2.9	39.6 ± 1.4	70.8 ± 4.1	38.9 ± 3.2	64.1 ± 1.1
InfoGraph [34]	68.2 ± 0.7	67.5 ± 1.4	71.8 ± 2.3	42.3 ± 1.8	75.2 ± 2.4	41.5 ± 1.7	65.7 ± 0.4
ASGN [13]	67.7 ± 1.2	68.5 ± 0.6	70.6 ± 1.4	41.2 ± 1.4	73.1 ± 2.3	42.2 ± 0.8	65.3 ± 0.8
GraphCL [42]	69.4 ± 0.8	68.7 ± 1.2	71.2 ± 2.5	**43.7 ± 1.3**	75.2 ± 1.7	42.3 ± 0.9	66.4 ± 0.6
JOAO [41]	68.7 ± 0.9	67.9 ± 1.3	71.0 ± 1.9	42.6 ± 1.5	74.8 ± 1.6	42.1 ± 1.2	65.8 ± 0.4
KGNN (Ours)	**70.9 ± 0.5**	**70.5 ± 0.6**	**72.5 ± 1.6**	43.3 ± 0.7	**76.0 ± 0.9**	**44.8 ± 0.6**	**67.4 ± 0.5**



(a) **PROTEINS** (b) **IMDB-B**

Figure 4: Results on two datasets *w.r.t.* the amounts of the labeled data (i.e., 5%, 25% and 50%) and all the unlabeled data.

performance. Among all the methods, the proposed KGNN achieves the best performance, which indicates that explicitly incorporating graph kernels as structured constraints into graph neural networks can further facilitate the performance for graph classification.

5.3 Ablation Study

In the KGNN model, either the GNN-based network or the kernel-based network is enhanced by incorporating extra "labels" selected from both networks. A more straightforward way could be empowering every single network with "labels" generated by itself, which is known as the self-training method. Next, we investigate a few variants that are trained under the self-training framework:

- **GNN-Sup.** Our base model, which trains a GNN (i.e., p_θ network) solely on labeled data TRAIN-L in a fully supervised manner.
- **MemNN-Sup.** It is another model, which optimizes a MemNN (i.e., q_ϕ network) solely on labeled data in a supervised way.
- **GNN-Self.** It is a model variant where we optimize a GNN (i.e., p_θ) using a self-training strategy.
- **Ensemble-Self.** Besides using a single network, we first ensemble the GNN-based network p_θ and the kernel-based network q_ϕ by concatenating their graph representations before the final prediction layer, and train the ensembled model with self-training.
- **KGNN-Sep.** KGNN-Sep differs from the original KGNN in that we directly feed the label annotations by one network into the other network without checking the prediction consistency.

Results and Analysis. The results of the above model variants are summarized in Table 3. We can see that GNN-Sup outperforms MemNN-Sup on most datasets, maybe the reason is that GNN-based methods can take advantage of node attributes while graph kernel-based methods fail. Also, GNN-Self outperforms GNN-Sup on 5 out of 7 datasets. On DD and REDDIT-B datasets, GNN-self performs worse than GNN-Sup, which may attribute to the noisy "labels" generated by the model itself. Importantly, we find that Ensemble-Self performs well compared with the above three model variants in most cases, which implies that incorporating graph similarity explicitly indeed benefits the performance. Besides, by using a more principled way to combine two networks, KGNN-Sep outperforms the above four models on all datasets, showing the effectiveness of our proposed posterior regularization. Finally, with the intersection strategy to eliminate the influence of incorrect annotations, our proposed KGNN achieves the best performance on all datasets, which is in accordance with our expectations.

5.4 Parameter Analysis

Finally, we examine the KGNN's performance varies w.r.t. different parameter settings. Specifically, we investigate the effect of embedding dimensions of hidden layers and the number of hops in the memory network on two datasets PROTEINS and COLLAB.

Effect of the Embedding Dimensions. We first analyze the effect of the embedding dimensions of hidden layers h. We expect the

Table 3: Comparison with several variants for ablation study (in %). We present the mean together with the standard deviation of prediction accuracy over five runs (in %) using different random seeds. The results in bold indicate the best performance.

Methods	Datasets						
	PROTEINS	DD	IMDB-B	IMDB-M	REDDIT-B	REDDIT-M-5k	COLLAB
GNN-Sup	63.3 ± 1.4	62.5 ± 1.5	63.4 ± 2.1	39.2 ± 1.6	69.8 ± 1.1	38.6 ± 2.5	61.7 ± 1.5
MemNN-Sup	59.8 ± 2.4	60.4 ± 1.0	59.3 ± 2.7	38.3 ± 2.8	65.2 ± 1.2	39.0 ± 2.6	60.1 ± 2.2
GNN-Self	65.2 ± 1.6	64.0 ± 1.4	67.3 ± 1.8	41.1 ± 2.2	68.3 ± 1.8	42.1 ± 2.1	63.5 ± 1.6
Ensemble-Self	65.5 ± 2.4	66.1 ± 3.4	68.0 ± 2.8	41.0 ± 3.1	72.4 ± 0.3	41.5 ± 2.8	64.3 ± 1.0
KGNN-Sep	68.8 ± 2.9	67.4 ± 1.5	68.8 ± 2.2	42.4 ± 3.2	73.6 ± 2.7	42.8 ± 1.3	65.1 ± 1.2
KGNN (Ours)	**70.9 ± 0.5**	**70.5 ± 0.6**	**72.5 ± 1.6**	**43.3 ± 0.7**	**76.0 ± 0.9**	**44.8 ± 0.6**	**67.4 ± 0.5**

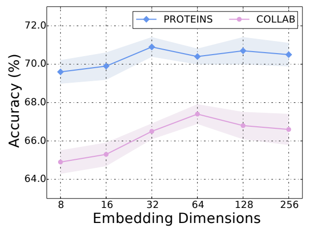


Figure 5: Performance *w.r.t.* the embedding dimensions.

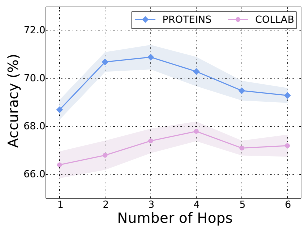


Figure 6: Performance *w.r.t.* the number of hops.

model to perform well as dimensions increase due to the benefit of augmented model capacity. We fix all other parameters to the ones that yield the best results and varied $h = \{8, 16, 32, 64, 128\}$, the obtained results are shown in Figure 5. We observe that larger embedding dimensions generally lead to better performance before saturation. The model needs larger embedding dimensions on COLLAB than PROTEINS to achieve the best performance as its dataset is large-scale and requires a larger capacity model to fit.

Effect of the Number of Hops. We further assess the effect of the number of hops in the memory network. We vary the number of hops from 1 to 6 while fixing all other parameters. As shown in Figure 6, we observe that increasing the number of hops results in better performance when the number is small, showing that the multi-hop mechanism has more broad attention and helps in learning correlations between memories. However, too many hops may hurt the performance, maybe the reason is that the query graph pays much attention to how similar it is with the in-memory graphs, which leads to overfitting and loses the generalization of the comparison similarity with the unknown graphs.

5.5 Case Study

In this section, we study the power of the kernel-based network to validate the strength of our framework, showing the superiority from the view of topological similarity. Figure 7 illustrates some of the results we obtained on REDDIT-M-5k, where two unlabeled samples are annotated with both GNN-based network p_θ and kernel-based network q_ϕ. We observe that both samples are classified into wrong categories by p_θ while q_ϕ corrects the wrong prediction successfully, which shows that our novel kernel-based network is a necessary supplement for the effective graph classification task.

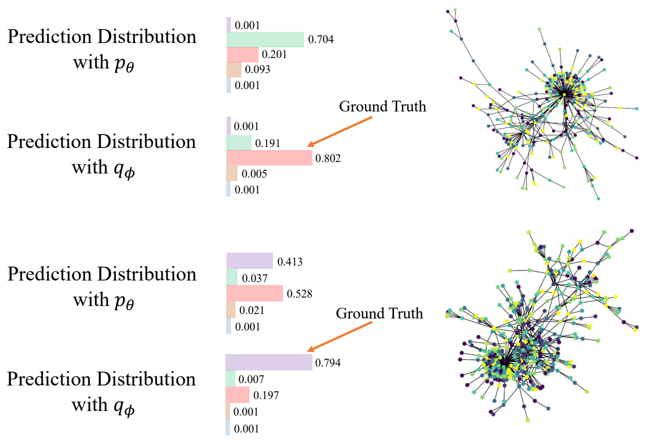


Figure 7: Illustration of two samples on REDDIT-M-5k.

The reason may be that the kernel-based network is able to capture more relationships between different substructures and thus acts as structured constraints to guide the GNN-based network.

6 CONCLUSION

This paper studies semi-supervised graph classification, which is a fundamental problem in graph-structured data modeling, and a novel approach called the KGNN is proposed. KGNN adopts the posterior regularization to incorporate graph kernels as structured constraints to guide the training process of graph neural networks under the EM-style framework. Specifically, in the M-step, we employ a GNN-based network, which can learn effective graph representations for prediction. In the E-step, we develop a kernel-based network using a memory network to capture the graph similarity efficiently. Experiments on a range of well-known benchmark datasets prove the effectiveness of the KGNN for graph classification. In the future, we plan to further extend our KGNN to a broader range of applications, such as drug discovery and material science.

ACKNOWLEDGMENTS

This paper is partially supported by National Key Research and Development Program of China with Grant No. 2018AAA0101902 as well as the National Natural Science Foundation of China (NSFC Grant No. 62106008 and No. 62006004).

REFERENCES

[1] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. 2018. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 170–182.

[2] Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *In Proceedings of the 5th IEEE International Conference on Data Mining*. 8–pp.

[3] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21 (2005), 47–56.

[4] Dexiong Chen, Laurent Jacob, and Julien Mairal. 2020. Convolutional Kernel Networks for Graph-Structured Data. In *Proceedings of International Conference on Machine Learning*. 1576–1586.

[5] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783.

[6] Simon S. Du, Kangcheng Hou, Ruslan Salakhutdinov, Barnabás Póczos, Ruosong Wang, and Keyulu Xu. 2019. Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 5724–5734.

[7] Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research* 11 (2010), 2001–2049.

[8] Cristina García and José Alí Moreno. 2004. The hopfield associative memory network: Improving performance with the kernel "trick". In *Ibero-American Conference on Artificial Intelligence*. Springer, 871–880.

[9] Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of Computational Learning theory and kernel machines*. 129–143.

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

[11] Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*. 529–536.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[13] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. 2020. ASGN: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 731–752.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning NIPS Workshop*.

[15] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of international conference on machine learning*. 321–328.

[16] Amir Hosein Khasahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. 2020. Memory-based graph networks. In *Proceedings of International Conference on Learning Representations*.

[17] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*.

[18] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations*.

[19] Ryosuke Kojima, Shoichi Ishida, Masateru Ohta, Hiroaki Iwata, Teruki Honma, and Yasushi Okuno. 2020. kGCN: a graph-based deep learning framework for chemical structures. *Journal of Cheminformatics* 12 (2020), 1–10.

[20] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. 2020. A survey on graph kernels. *Applied Network Science* 5, 1 (2020), 1–42.

[21] Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *Proceedings of International Conference on Learning Representations*.

[22] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*.

[23] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *Proceedings of International Conference on Machine Learning*. 3734–3743.

[24] Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wenbing Huang, and Junzhou Huang. 2019. Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*. 972–982.

[25] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 8 (2018), 1979–1993.

[26] 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).

[27] Dimitri Nowicki and Hava Siegelmann. 2010. Flexible kernel memory. *PloS One* 5, 6 (2010), e10955.

[28] Georgios A Pavlopoulos, Maria Secrier, Charalampos N Moschopoulos, Theodoros G Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pantelis G Bagos. 2011. Using graph theory to analyze biological networks. *BioData mining* 4, 1 (2011), 1–27.

[29] 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), 2539–2561.

[30] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Proceedings of International Conference on Artificial Intelligence and Statistics*. 488–495.

[31] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 555–563.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[33] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems*. 2440–2448.

[34] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proceedings of International Conference on Learning Representations*.

[35] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*. 1195–1204.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *Proceedings of International Conference on Learning Representations*.

[37] Boris Weisfeiler and Andrei A Lehman. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia* 2, 9 (1968), 12–16.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *Proceedings of International Conference on Learning Representations*.

[39] 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.

[40] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*. 4805–4815.

[41] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. *arXiv preprint arXiv:2106.07594* (2021).

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