Neural Networks 151 (2022) 70–79

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

GHNN: Graph Harmonic Neural Networks for semi-supervised graph-level classification

Wei Ju [a,1], Xiao Luo [b,1], Zeyu Ma [c], Junwei Yang [a], Minghua Deng [b,*], Ming Zhang [a,*]

[a] School of Computer Science, Peking University, Beijing, 100871, Beijing, China
[b] School of Mathematical Sciences, Peking University, Beijing, 100871, Beijing, China
[c] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, 518055, Guangdong, China

ARTICLE INFO

Article history:
Received 25 October 2021
Received in revised form 19 January 2022
Accepted 10 March 2022
Available online 24 March 2022

Keywords:
Graph classification
Graph neural networks
Graph kernels
Semi-supervised learning

ABSTRACT

Graph classification aims to predict the property of the whole graph, which has attracted growing attention in the graph learning community. This problem has been extensively studied in the literature of both graph convolutional networks and graph kernels. Graph convolutional networks can learn effective node representations via message passing to mine graph topology in an implicit way, whereas graph kernels can explicitly utilize graph structural knowledge for classification. Due to the scarcity of labeled data in real-world applications, semi-supervised algorithms are anticipated for this problem. In this paper, we propose Graph Harmonic Neural Network (GHNN) which combines the advantages of both worlds to sufficiently leverage the unlabeled data, and thus overcomes label scarcity in semi-supervised scenarios. Specifically, our GHNN consists of a graph convolutional network (GCN) module and a graph kernel network (GKN) module that explore graph topology information from complementary perspectives. To fully leverage the unlabeled data, we develop a novel harmonic contrastive loss and a harmonic consistency loss to harmonize the training of two modules by giving priority to high-quality unlabeled data, thereby reconciling prediction consistency between both of them. In this manner, the two modules mutually enhance each other to sufficiently explore the graph topology of both labeled and unlabeled data. Extensive experiments on a variety of benchmarks demonstrate the effectiveness of our approach over competitive baselines.

1. Introduction

Graphs are widely used to extract various complicate relations in the real world. Recently, numerous works have attempted to extend the convolutional neural networks (CNNs) to graph-structured data (Kipf & Welling, 2017; Veličković et al., 2017; Wu et al., 2020; Xu, Hu, Leskovec, & Jegelka, 2019). These methods are typically called graph neural networks (GNNs), which have been applied to various important tasks, e.g., node classification and link prediction in different domains. Among them, more attention has been applied to the problem of graph classification, which refers to predicting the labels of the whole graph based on the attributive and structural information in the graph. This problem is critical with a wide range of applications (Kojima et al., 2020), such as determining the activity of new compounds in molecular biology and pharmacology (Hao et al., 2020; Lu et al., 2019).

In recent years, numerous algorithms have been developed for graph classification (Lee, Lee, & Kang, 2019; Ying et al., 2018; Zeng & Xie, 2021). As a representative GNN, graph convolutional networks (GCNs) have seen tremendous success. GCNs typically use neighbor-aware message passing mechanisms to embed node attributes and structural information into node representations (Hamilton, Ying, & Leskovec, 2017; Kipf & Welling, 2017; Veličković et al., 2017). To facilitate graph classification, numerous graph pooling methods (Bianchi, Grattarola, & Alippi, 2020; Lee et al., 2019; Yuan & Ji, 2020) have been also developed to aggregate all the node representations into a graph representation. Thus, the learned graph representation could effectively capture the structural-semantic information contained in the graph for effective graph classification.

Of note, these methods are often trained in a supervised manner (Lee et al., 2019; Ying et al., 2018; Zhang, Cui, Neumann, & Chen, 2018). Hence, in the training process, these methods require a large number of labeled graphs, which is typically expensive and time-consuming to obtain in real-world applications (Hao et al., 2020). For example, it takes about five hours to compute the characteristics of a molecule with just 20 atoms using density functional theory (Engel & Dreizler, 2013), which can

* Corresponding authors.
E-mail addresses: dengmh@pku.edu.cn (M. Deng), mzhang_cs@pku.edu.cn (M. Zhang).

[1] Equal contribution with alphabetical order.

https://doi.org/10.1016/j.neunet.2022.03.018
0893-6080/

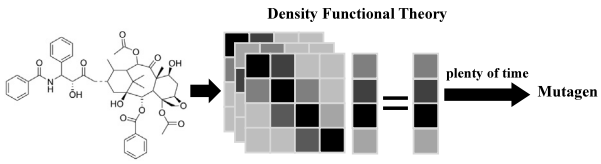


Fig. 1. An example of label annotation in chemistry. It takes approximately 5 h to calculate the properties of the above molecule, indicating the graph label scarcity in practice.

be illustrated in Fig. 1. Owing to the shortage of labeled graphs, the majority of existing approaches perform poorly. To address this problem, we note that there are a large number of unlabeled graphs available in a variety of disciplines. Though there is no access to their label property, the topologies of these unlabeled graphs can well be used as guidance to regularize the graph encoder. In this spirit, this paper investigates semi-supervised graph classification, which aims to leverage both labeled and unlabeled data for graph property prediction.

There have been a few works for semi-supervised graph classification (Hao et al., 2020; Li et al., 2019; Sun, Hoffmann, Verma, & Tang, 2020). These works usually combine GCNs with traditional semi-supervised learning approaches such as self-training (Lee et al., 2013) and knowledge distillation. For example, SEAL-AI (Li et al., 2019) leverages a classifier to annotate unlabeled samples. ASGN (Hao et al., 2020) and InfoGraph (Sun et al., 2020) both involve the teacher–student framework to explore graphs from different views. However, these methods typically focus on semi-supervised learning algorithms but neglect to sufficiently explore structural information in substructures such as paths. Specifically, in GCN architectures, graph topology is predominantly involved while propagating node representations along edges, implying structural information is not fully explored in these methods. Furthermore, since the number of labeled graphs in semi-supervised graph classification is scarce, traditional GCNs cannot well classify graph samples with less exploration of structural information.

Another area of research along this line is based on recent advancements in graph kernels (Chen, Jacob and Mairal, 2020; Du et al., 2019; Kriege, Johansson, & Morris, 2020; Long, Jin, Wu, & Song, 2021). Graph kernels are classical techniques for measuring similarities between pairwise substructures or graphs, allowing for graph clustering, comparison, and classification. Their core is to decompose graphs into various substructures and leverage kernel functions to explore the graph similarity. Graph kernels are impressed with the ability to explicitly explore graph topology information, which is vital for graph classification, especially in the chemical fields (Hao et al., 2020). Although graph kernels can provide a brand new view for modeling graph topology, these methods typically define substructures or feature vectors with some hand-crafted rules, which are quite heuristic and may result in poor generalization and sub-optimal performance.

To overcome the aforementioned issues, we propose a novel framework called Graph Harmonic Neural Network (GHNN), which combines the advantages of both graph convolutional networks and graph kernels. Firstly, we use a neighbor-aware message passing mechanism to learn structured node representations followed by a graph-level pooling operation. In this way, the summarized graph representation would implicitly capture graph structural information. Secondly, we employ a graph kernel network leveraging the random walk kernel to compare each graph with a number of hidden graphs, which generates a new graph representation that explicitly explores graph structural information. To couple topology information from two worlds, we develop a semi-supervised optimization framework consisting of a novel harmonic contrastive loss and a harmonic consistency loss, where two modules are encouraged to collaborate with each other via giving priority to high-quality unlabeled data and producing consistent predictions. We conduct experiments on a wide range of graph classification benchmark datasets. The results indicate that our proposed GHNN outperforms competitive baselines by a large margin. Overall, our contributions can be highlighted as follows:

- We propose a novel framework called GHNN for semi-supervised graph classification, consisting of a GCN module and a GKN module, which sufficiently explore the unlabeled graphs to capture the graph topology and overcome the scarcity of labeled graphs.
- To couple the structural information from different perspectives, we present a well-designed semi-supervised framework consisting of a novel harmonic contrastive loss and a harmonic consistency loss to enable two modules to collaborate with each other and encourage prediction consistency.
- Experiments on a range of well-known bioinformatics and social network benchmark datasets demonstrate that GHNN achieves state-of-the-art performance.

2. Related work

2.1. Graph convolutional networks

Graph Convolutional Networks (GCNs) (Hamilton et al., 2017; Kipf & Welling, 2017; Lin, Gao, & Li, 2020; Xu et al., 2019) have grown in popularity due to their simplicity to deal with graph-structured data. The majority of GCNs follow the message passing mechanisms, which combine node attributes and graph structural information for node representations. GCNs have been successfully applied into different applications such as node representation learning (Liu, Wen, Kang, Luo, & Tian, 2021), node classification (Chong, Ding, Yan, & Pan, 2020; Fu et al., 2021; Fu, Liu, Zhang, Zhou and Tao, 2021) and node clustering (Pan & Kang, 2021). As for graph classification (Lee et al., 2019; Zeng & Xie, 2021), there are two groups of pooling strategies to obtain graph-level representation. The first group is global pooling (Gilmer, Schoenholz, Riley, Vinyals, & Dahl, 2017), which adds a simple pooling operation, such as global summation of all node embeddings. The second group is hierarchical pooling (Gao & Ji, 2019; Ying et al., 2018), which aggregates messages on coarser graphs for the graph representation. Nevertheless, GCNs fail to model substructures in graphs such as paths (Long et al., 2021), which is critical when the labels are scarce. Compared with previous works, our GHNN explores graph topology from both implicit and explicit manners to overcome the scarcity of label annotations in real-world applications.

2.2. Graph kernels

In machine learning and data mining, kernel methods have been extensively investigated and utilized in various tasks (Camastra & Verri, 2005; Elisseeff & Weston, 2001). Kernel methods leverage kernel functions to assess pairwise similarities between input samples. In this way, data samples are implicitly projected onto high-dimensional space to provide richer features. Kernel methods have been extended into graph data, bringing in different graph kernels (Gärtner, Flach, & Wrobel, 2003; Kashima, Tsuda, & Inokuchi, 2003). Graph kernels usually begin with decomposing graphs into atomic substructures and using kernel functions to capture graph similarity. Early approaches leverage counts of label pathways through the random walk or the shortest path to generate feature vectors, but they may have scalability issues and are very computationally intensive. Recent researches

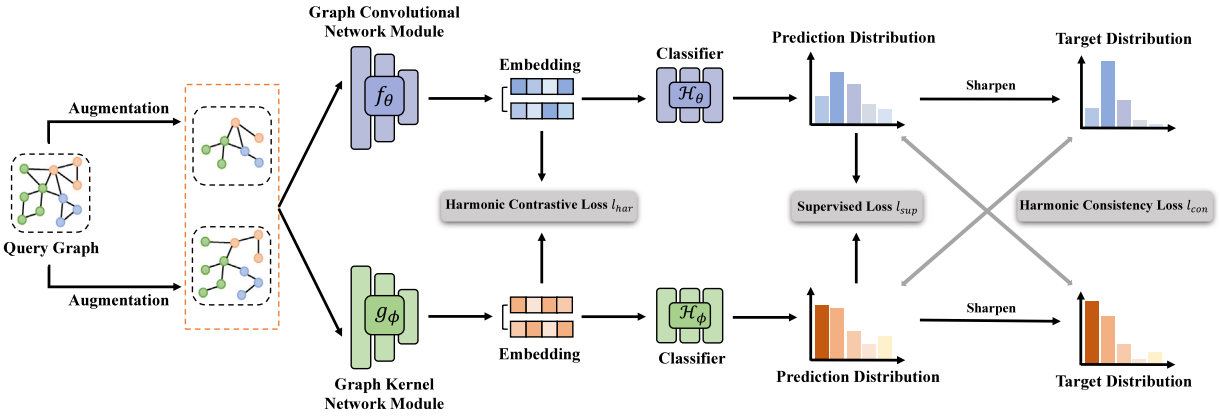


Fig. 2. The schematic of the proposed framework GHNN. Our GHNN shapes a semi-supervised graph classification framework with two modules (i.e., a GCN module and a GKN module) to collaborate with each other and encourage prediction consistency. The framework is optimized by supervised loss, harmonic contrastive loss, and harmonic consistency loss.

have sought to merge graph neural networks with graph kernels due to the limitations of GCNs (Chen, Jacob et al., 2020; Du et al., 2019). With an explicit multilayer kernel, GCKN (Chen, Jacob et al., 2020) improves graph neural networks to explore more topology information. GNTK (Du et al., 2019) demonstrates that a graph kernel is equivalent to infinitely wide graph neural networks trained by standard gradient descent. Our proposed solution integrates the best of both worlds but from distinct perspectives. To be more specific, our method not only investigates graph structure both implicitly and explicitly, but also uses unlabeled data to facilitate model training in semi-supervised circumstances.

2.3. Semi-supervised graph classification

Semi-supervised learning has lately received a lot of attention. Self-training (Grandvalet & Bengio, 2005; Lee et al., 2013) and consistency regularization (Laine & Aila, 2017; Tarvainen & Valpola, 2017) are two mainstream techniques for semi-supervised learning. The first technique leverages a trained classifier to predict the class labels of unlabeled data iteratively and add high-quality classified samples into the training set. The second technique expects the model to output consistent predictions when fed perturbed versions of the input data. Recently, various semi-supervised graph classification algorithms (Hao et al., 2020; Li et al., 2019; Sun et al., 2020) have been proposed to solve the scarcity of label annotations in real-world applications. SEAL-AI (Li et al., 2019) solves the issue from the view of a hierarchical graph and self-training. Furthermore, InfoGraph (Sun et al., 2020) and ASGN (Hao et al., 2020) use contrastive learning and active learning to learn graph representations, respectively. Unlike prior approaches which only leverage GCN to model graph topology implicitly, we use graph kernels to capture topology explicitly.

3. Methodology

This paper introduces a novel framework GHNN in Fig. 2 for semi-supervised graph classification. We begin with a formal definition of the problem in Section 3.1. Then, we introduce our graph convolutional module and graph kernel network module in Sections 3.2 and 3.3, respectively. Finally, we propose a joint semi-supervised harmonic optimization framework in Section 3.4, where two modules are encouraged to benefit from each other for making the best of both labeled and unlabeled data.

3.1. Problem definition

In this section, we introduce the definition and notations of the graph, and formally define the semi-supervised graph classification task.

Definition 1 (*Graph*)**.** A graph is formally defined as $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges between nodes in V. The topology information of the whole graph is described by the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, where n is the number of nodes. A graph is typically associated with a node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d'}$, where each row $\boldsymbol{x}_i \in \mathbb{R}^{d'}$ represents the feature vector of node i and d' is the dimension of node features.

Definition 2 (*Semi-supervised Graph Classification*)**.** Given a set of labeled graphs $\mathcal{G}^L = \left\{G_1, \ldots, G_{|\mathcal{G}^L|}\right\}$ and a set of unlabeled graphs $\mathcal{G}^U = \left\{G_{|\mathcal{G}^L|+1}, \ldots, G_{|\mathcal{G}^L|+|\mathcal{G}^U|}\right\}$, semi-supervised graph classification task aims to learn a classifier that (1) fits the labeled graphs $\mathcal{G}^L$; and (2) captures the information in the unlabeled graphs $\mathcal{G}^U$, at the same time.

3.2. Graph convolutional network module

Graph convolutional networks (GCNs), which implicitly capture graph topological information, have demonstrated their effectiveness in learning the representation of graph-structured data (Gilmer et al., 2017; Kipf & Welling, 2017; Veličković et al., 2017; Xu et al., 2019), such as social networks and molecules. GNNs iteratively update the representation of each node by aggregating information from their neighbor nodes. In its most general form, a GNN consists of K hidden layers, where at the kth layer, each node $v \in V$ aggregates and updates messages from its 1-hop neighboring nodes $\mathcal{N}(v)$, leading to the feature vector

$$\begin{aligned}\mathbf{h}_{\mathcal{N}(v)}^{(k)} &= AGGREGATE_{\theta}^{(k)}\left(\left\{\mathbf{h}_u^{(k-1)} | u \in \mathcal{N}(v)\right\}\right)\\ \mathbf{h}_v^{(k)} &= UPDATE_{\theta}^{(k)}\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(k)}\right)\end{aligned} \tag{1}$$

where $\mathcal{N}(v)$ denotes the neighbors of v, $\mathbf{h}_v^{(k)}$ denotes the embedding of node v at the kth layer. Here $AGGREGATE_{\theta}^{(k)}$ and $UPDATE_{\theta}^{(k)}$ are two trainable functions at the kth layer, respectively. Common aggregate and updating functions could be averaging or summation over all elements. After K iterations of Eq. (1), the

graph-level representation is obtained by pooling the final set of node representations expressed as

$$f_\theta(G) = \mathrm{POOL}\left(\left\{\mathbf{h}_v^{(K)}\right\}_{v\in V}\right) \tag{2}$$

where θ is the parameter set of the graph convolutional network module and $f_\theta(\cdot)$ is the pooling function which represents averaging or a more sophisticated graph-level pooling function (Lee et al., 2019; Ying et al., 2018; Zhang et al., 2018). In this way, Eq. (2) summarizes the node representations into a higher-level graph representation and explores graph structural information implicitly.

3.3. Graph kernel network module

In the previous section, GCNs entangle the node features and graph topology, and the graph structural information is only utilized when propagating node representations along edges, which fails to model high-order substructures such as paths (Chen, Jacob et al., 2020; Long et al., 2021). As a consequence, we leverage a random walk graph kernel to capture the graph structural information explicitly.

For this reason, we introduce a graph kernel network module consisting of a number of hidden graphs parameterized by trainable adjacency matrices, as shown in Fig. 3. Specifically, there are N hidden graphs $G'_1, G'_2, \ldots, G'_N$. Each hidden graph G'_i of size n is assumed as the undirected graph without self-loops for fewer parameters ($n(n-1)/2$ trainable parameters in adjacency matrix $\mathbf{W}_i \in \mathbb{R}^{n\times n}$). These hidden graphs are anticipated to learn the structure that helps distinguish the available categorizations. We compare each input graph with hidden graphs using a differentiable function based on the random walk kernel, driven by the observation that random walk kernels can measure the similarity of two graphs using the number of common walks in the graph pair (Kashima et al., 2003; Nikolentzos & Vazirgiannis, 2020).

We begin with the introduction of graph direct product. Denote two graphs as $G = (V, E)$ and $G' = (V', E')$ and their direct product $G_\times = (V_\times, E_\times)$ is a graph where

$$\begin{aligned} V_\times &= \left\{\left(v, v'\right) : v \in V \wedge v' \in V'\right\} \\ E_\times &= \left\{\left\{\left(v, v'\right), \left(u, u'\right)\right\} : \{v, u\} \in E \wedge \left\{v', u'\right\} \in E'\right\} \end{aligned} \tag{3}$$

According to the definition, performing a random walk on direct product $G_\times$ of G and G' is equivalent to a concurrent random walk on two graphs (Vishwanathan, Schraudolph, Kondor, & Borgwardt, 2010). Given that the random walk kernels calculate all the pairs of matching walks on two graphs, we can infer the number of matching walks through the adjacency matrix $\mathbf{A}_\times$ of $G_\times$ if we assume a uniform distribution to characterize both the starting and stopping probabilities over nodes of two graphs. Given $P \in \mathbb{N}$, the P-step random walk kernel between two graphs G and G' is defined as:

$$k\left(G, G'\right) = \sum_{i=1}^{|V_\times|}\sum_{j=1}^{|V_\times|}\left[\sum_{p=0}^{P}\lambda_p \mathbf{A}_\times^p\right]_{ij} \tag{4}$$

where $\lambda_0, \ldots, \lambda_P$ are positive and real-valued weights. To simplify the calculation, we only count the number of common walks of length exactly p over two compared graphs:

$$k^{(p)}\left(G, G'\right) = \sum_{i=1}^{|V_\times|}\sum_{j=1}^{|V_\times|}\left[\mathbf{A}_\times^p\right]_{ij} \tag{5}$$

Furthermore, we generalize the above equation to graphs associated with multi-dimensional node attributes. Let $\mathbf{X}$ and $\mathbf{X}' \in \mathbb{R}^{n\times d'}$ denote the matrices containing the node attributes of the graph G and G' respectively. We first measure the similarity between the node attributes of two graphs, i.e., $\mathbf{S} = \mathbf{X}'\mathbf{X}^T \in \mathbb{R}^{|V'|\times|V|}$. Next, $\mathbf{S}$ is transformed into a vector $\mathbf{s} \in \mathbb{R}^{|V'||V|}$ by stacking the columns of the matrix one after another. Note that the (i, j)th element of matrix $A_\times^p$ is equal to the number of walks of length p between the ith node and jth node of $G_\times$. Each node of $G_\times$ corresponds to a pair of nodes, one from graph G and the other from graph G'. Each node of $G_\times$ can be assigned with a real value that quantifies the similarity between the attributes of the two nodes it represents. These values are contained in vector $\mathbf{s}$. As a result, we reformulate the random walk kernel as:

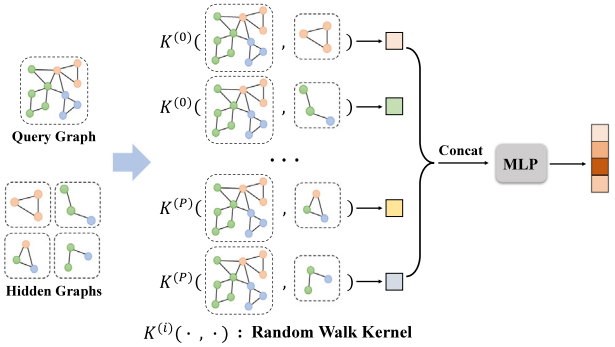


Fig. 3. An illustration of the graph kernel network module. It compares each query graph against a number of hidden graphs to produce the graph representation in an end-to-end manner.

$$k^{(p)}\left(G, G'\right) = \sum_{i=1}^{|V_\times|}\sum_{j=1}^{|V_\times|}\mathbf{s}_i\mathbf{s}_j\left[\mathbf{A}_\times^p\right]_{ij} \tag{6}$$

Finally, given $\mathcal{P} = \{0, \ldots, P\}$ and parameterized hidden graph set $\mathcal{G}_h = \{G'_1, \ldots, G'_N\}$, we can obtain a matrix $\mathbf{H} \in \mathbb{R}^{N\times(P+1)}$ where $\mathbf{H}_{ij} = k^{(j-1)}(G, G'_i)$ for each input graph G. Then, the matrix $\mathbf{H}$ is flattened as $\hat{\mathbf{H}} \in \mathbb{R}^{N(P+1)}$ and fed into a fully-connected layer (MLP) to produce the graph-level representation denoted as $g_\phi(G) = \mathrm{MLP}(\hat{\mathbf{H}})$, where ϕ is the parameter set of the graph kernel network module.

3.4. Semi-supervised optimization framework

In this section, we elaborate on how to fully leverage the unlabeled data and integrate the graph representations derived from the two modules to explore graph topology from different perspectives for semi-supervised classification.

3.4.1. Supervised loss

In the semi-supervised scenario, although labeled graphs are scarce, they are also indispensable since they can guide the model to produce task-oriented graph representations for classification. To achieve this, we build two distinct multi-layer perception (MLP) classifiers $\mathcal{H}_\theta(\cdot)$ and $\mathcal{H}_\phi(\cdot)$ to map the embedding vectors from different modules to label predictions, respectively (i.e., $\Phi_1(y \mid G) = \mathcal{H}_\theta(f_\theta(G))$ and $\Phi_2(y \mid G) = \mathcal{H}_\phi(g_\phi(G))$). The supervised classification loss for two modules is defined in the form of cross-entropy:

$$\mathcal{L}_{sup} = \frac{1}{|\mathcal{G}^L|}\sum_{G_j\in\mathcal{G}^L}\left[-\log\Phi_1(y_j \mid G_j) - \log\Phi_2(y_j \mid G_j)\right] \tag{7}$$

in which y_j is the corresponding ground truth class.

3.4.2. Harmonic contrastive loss

As the labeled data is very scarce in the semi-supervised scenario, we instead attempt to sufficiently leverage the unlabeled

data, which could be potentially beneficial to enhance the performance. However, the structural information in the unlabeled data is hard to be comprehensively captured in any single module without extra guidance. In addition, since two modules explore structural-semantic information from different views, there exists a distinction between semantic representations derived from their respective embedding spaces. To combine the advantages of the two modules and further encourage mutual communication between both of them, we expect to develop a novel objective to harmonize the optimization process of the two modules, so as to better model graph topology and explore the unlabeled data.

Graph Contrastive Learning. Motivated by recent success in graph contrastive learning (You, Chen, Shen, & Wang, 2021; You et al., 2020), our framework employs graph augmentations (You et al., 2020) to construct generalized and robust representation pairs to fully explore the unlabeled data. Generally, four basic data augmentation strategies are adopted: (1) Edge deletion: We randomly remove some edges from the graph following a default uniform distribution. It is premised on the assumption that semantic information is unaffected by changes in edge connection patterns. (2) Node deletion: We randomly remove certain nodes from the graph, along with all connected edges. Also, the dropping probability of each node follows a default i.i.d. uniform distribution. (3) Attribute masking: We sample some nodes at random and then mask certain of their attributes randomly. It assumes that the graph representation will be robust in the presence of incomplete node attributes. (4) Subgraph: We sample a subgraph from the graph using a random walk. The underlying assumption is that the graph's semantics can be largely preserved in its local structure. As such, augmented graphs are generated by randomly selecting one of the augmentation operations.

Formally, the given graph G first performs stochastic graph augmentations to obtain two correlated views $\hat{G}^{(1)}$ and $\hat{G}^{(2)}$, as a positive pair. Then, the GCN module $f_\theta(\cdot)$ is used to extract graph-level representations $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ for augmented graphs $\hat{G}^{(1)}$ and $\hat{G}^{(2)}$. The noise-contrastive estimation loss (Oord, Li, & Vinyals, 2018) is employed to enforce maximizing the consistency between positive pairs $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}\}$ compared with negative pairs. In practice, we randomly sample a minibatch of M graphs, producing $2M$ random augmented graphs $\{\hat{G}_m^{(1)}, \hat{G}_m^{(2)}\}_{m=1}^{M}$. Given a positive pair $\hat{G}_m^{(1)}$ and $\hat{G}_m^{(2)}$, we treat the other $(M-1)$ augmented graphs within a minibatch as negative examples. Here we re-annotate $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ as $\mathbf{z}_m^{(1)}$ and $\mathbf{z}_m^{(2)}$ for the mth graph in the minibatch. Denote $\mathbf{z}_m^{(1)} \star \mathbf{z}_m^{(2)}$ as the cosine similarity of $\mathbf{z}_m^{(1)}$ and $\mathbf{z}_m^{(2)}$. We compare two graph representations for the mth graph as follows:

$$\ell_m^1 = -\log \frac{e^{\mathbf{z}_m^{(1)} \star \mathbf{z}_m^{(2)}/\tau}}{\sum_{m'=1}^{M} e^{\mathbf{z}_m^{(1)} \star \mathbf{z}_{m'}^{(2)}/\tau}} \tag{8}$$

where τ is a temperature parameter set to 0.5 following You et al. (2020). Similarly, the output of the GKN module is denoted as $\mathbf{z}'^{(1)}_m$ and $\mathbf{z}'^{(2)}_m$ for the augmented pair $\hat{G}_m^{(1)}$ and $\hat{G}_m^{(2)}$ respectively, hence the contrastive learning loss for the mth graph is defined as:

$$\ell_m^2 = -\log \frac{e^{\mathbf{z}'^{(1)}_m \star \mathbf{z}'^{(2)}_m/\tau}}{\sum_{m'=1}^{M} e^{\mathbf{z}'^{(1)}_m \star \mathbf{z}'^{(2)}_{m'}/\tau}} \tag{9}$$

We can obviously observe that for each unlabeled sample, the contrastive learning loss for the two modules is trained independently without interacting with each other explicitly, which could cause prediction inconsistency between two classified categories. This inconsistency may come from the quality of the unlabeled data. The over-training of low-quality data may deteriorate the overall performance for semi-supervised graph classification.

To alleviate the above issues, we develop an unsupervised harmonic contrastive loss to give priority to high-quality unlabeled data and strengthen the correlation of two modules. Given the mth unlabeled data, the harmonic contrastive loss can be defined as follows:

$$\ell_m^{har} = (1+\beta_m^2)\ell_m^1 + (1+\beta_m^1)\ell_m^2 \tag{10}$$

where β_m^1, β_m^2 are two key but dynamic harmonic factors in our methods. Following the assumption of traditional semi-supervised learning that the quality of unlabeled data depends on its pseudo-labels (Grandvalet & Bengio, 2005; Lee et al., 2013), we define the harmonic factors as the negative entropy of pseudo-labels:

$$\beta_m^r = \exp\{\sum_c \Phi_r(y_m = c|G_m) \log \Phi_r(y_m = c|G_m)\} \tag{11}$$

From Eq. (11), we can observe that when the entropy of pseudo-labels from one module is small, indicating the high quality of this sample, we put more emphasis on it in the training of the other module. In this way, we use a mutual supervision mechanism to harmonize the optimization of contrastive learning in two separate branches. These two branches explore graph structural information from complementary views and exchange its judgment of data quality, which may help to enhance the performance by sufficiently using the unlabeled data in semi-supervised scenarios. As such, the overall harmonic contrastive loss within a mini-batch is calculated as follows:

$$\mathcal{L}_{har} = \frac{1}{2M} \sum_{m=1}^{M} \ell_m^{har} \tag{12}$$

3.4.3. Harmonic consistency loss

Although harmonic contrastive learning can harmonize the optimization of the two modules, it could not be able to guarantee that the predictions of the two modules are consistent. To this end, we introduce a harmonic consistency loss into our framework to encourage the prediction consistency. We first simplify the symbols of prediction distribution, i.e., $\mathbf{q}_j^1 \overset{\text{def}}{=} \Phi_1(y \mid G_j)$ and $\mathbf{q}_j^2 \overset{\text{def}}{=} \Phi_2(y \mid G_j)$. Inspired by Xie, Girshick, and Farhadi (2016), we refine the prediction distribution to get a target prediction by a sharpening function ρ:

$$\left[\rho\left(\mathbf{q}_j^1\right)\right]_c := \frac{\left[\mathbf{q}_j^1\right]_c^{1/T}}{\sum_{c=1}^{\mathcal{C}} \left[\mathbf{q}_j^1\right]_c^{1/T}}, c = 1, \ldots, \mathcal{C} \tag{13}$$

where T is a temperature parameter set to 0.5 following Xie et al. (2016) and $\mathcal{C}$ denotes the number of categorizations. The sharpening operation can generate a stronger target distribution since it improves the purity of the prediction and focuses larger attention on data points with a solid prediction.

Afterward, we optimize the prediction consistency by using their high-confidence assignments as guidance (i.e., target distribution). To be more exact, our approach is trained by comparing label predictions to the target distributions. Since we seek to harmonize the learning process through enhancing the prediction consistency of two modules, we compare the prediction from one module to the sharpened prediction from the other module. Formally,

$$\mathcal{L}_{con} = \frac{1}{2|\mathcal{G}^U|} \sum_{G_j \in \mathcal{G}^U} [D_{KL}(\mathbf{q}_j^2 \| \rho(\mathbf{q}_j^1)) + D_{KL}(\mathbf{q}_j^1 \| \rho(\mathbf{q}_j^2))] \tag{14}$$

where D_{KL} denotes the KL-divergence to penalize the difference between two distributions. Our harmonic consistency loss makes the best of two complementary modules to generate reliable label predictions, which can be interpreted as a hybrid of pseudo-labeling (Lee et al., 2013) and consistency learning (Laine & Aila,

Algorithm 1 Learning Algorithm of GHNN

Input: Labeled data $\mathcal{G}^L$, unlabeled data $\mathcal{G}^U$
Parameter: GCN module parameter θ, GKN module parameter ϕ
Output: Jointly learned label distribution $\Phi\ (y|G)$

```
Initialize the model parameters.
while not convergence do
    Sample mini-batches B^L and B^U from the labeled data and the unlabeled data, respectively.
    Forward propagation B^L and B^U through graph augmentations and two modules.
    Calculate loss function in Eq. (15).
    Update model parameters through back propagation.
end while
```

2017; Sajjadi, Javanmardi, & Tasdizen, 2016). On the one hand, traditional pseudo-labeling methods (Lee et al., 2013) retain labels with the largest class probability over a predefined value. Moreover, we leverage a sharpening function (Xie et al., 2016) that preserves the maximum discriminative information for unlabeled samples. On the other hand, consistency learning (Laine & Aila, 2017; Sajjadi et al., 2016) aims to leverage the unlabeled data by assuming that the model should output similar predictions when fed into different views of the same sample. By comparing the prediction from one module to the sharpened prediction of the other module, our approach explores the prediction consistency that results in highly confident and module-invariant predictions.

3.4.4. Framework summary and discussion

To sum up, we combine the supervised classification loss $\mathcal{L}_{sup}$, harmonic contrastive loss $\mathcal{L}_{har}$, and harmonic consistency loss $\mathcal{L}_{con}$ in the final loss:

$$\mathcal{L} = \mathcal{L}_{sup} + \mathcal{L}_{har} + \mathcal{L}_{con} \tag{15}$$

In a nutshell, our model is optimized to predict the graph properties from the perspectives of both implicit and explicit topology exploration. The overall framework of the GHNN is illustrated in Algorithm 1.

Complexity Analysis. The computing complexity of our framework mainly depends on the propagation of two graph encoders. We begin with the review of the notations. For the graph G, $\|A\|_0$ denotes the number of nonzeros in its adjacency matrix, d' is the feature dimension, K is the layer number of GCN module and $|V|$ is the number of nodes. We find that the GCN module takes $\mathcal{O}(K\|A\|_0 d' + K|V|d'^2)$ computational time and the GKN module takes $\mathcal{O}(P(Nn(n+|V|) + \|A\|_0))$ for graph G. On the basis of the analysis, the computational complexity of our GHNN is linearly related to both $|V|$ and $\|A\|_0$.

4. Experiment

In this section, we carry out extensive experiments to evaluate the effectiveness of our proposed method GHNN. We attempt to answer the following research questions:

- **RQ1**: How does GHNN perform compared with state-of-the-art models for semi-supervised graph classification task?
- **RQ2**: How is the effectiveness of our proposed techniques (e.g., GCN module, GKN module, harmonic contrastive loss, and harmonic consistency loss)?
- **RQ3**: How do the model hyper-parameters in GHNN affect the final performance?
- **RQ4**: How do different augmentation strategies affect the final performance?

Table 1
Overview of the seven datasets.

Datasets	Sizes	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

In what follows, we first introduce the experimental settings and then answer the above three questions.

4.1. Experimental settings

Datasets. We validate the proposed approach on seven well-known benchmark datasets[2] following Sun et al. (2020), You et al. (2021, 2020), which are commonly used in graph classification. The dataset statistics are summarized in Table 1. Specifically, there are two bioinformatics datasets (i.e., PROTEINS Borgwardt et al., 2005 and DD Dobson & Doig, 2003), four social network datasets (i.e., IMDB-B, IMDB-M, REDDIT-B, REDDIT-M-5k Yanardag & Vishwanathan, 2015) and one scientific collaboration dataset (i.e., COLLAB Yanardag & Vishwanathan, 2015). Following InfoGraph (Sun et al., 2020), we use all-ones encoding as input node features when node attributes are not accessible in the datasets.

Data Split. To fit the datasets for semi-supervised scenarios, we split the original datasets with the ratio of 7:1:2 as the training set, validation set, and test set, respectively. More specifically, we further sample 2/7 of the training set as the labeled set, and use the rest training set as the unlabeled set, whose ground-truth label is unavailable to models. To validate the effectiveness of our framework in the semi-supervised scenario with limited labeled data, 50% of the labeled set is used as default in our experiment. The validation set is used for hyper-parameter selection and the test set is used for the model performance.

Baselines. We compare the proposed GHNN with representative and state-of-the-art approaches which can be divided into three families: traditional graph methods, traditional semi-supervised methods, and graph-specific semi-supervised methods, as described below.

Traditional graph methods:

- **Graphlet Kernel** (Shervashidze, Vishwanathan, Petri, Mehlhorn, & Borgwardt, 2009): It compares graph pairs by counting or sampling commonly limited size substructures called graphlet.
- **Shortest Path (SP) Kernel** (Borgwardt & Kriegel, 2005): It proposes graph kernels based on shortest paths, which measure the similarity of two graphs by comparing the labels and lengths of the shortest paths between all pairs of nodes in two graphs.
- **Weisfeiler–Lehman (WL) Kernel** (Shervashidze, Schweitzer, Van Leeuwen, Mehlhorn, & Borgwardt, 2011): It is based on Weisfeiler–Lehman test of isomorphism on graphs (Weisfeiler & Lehman, 1968) that follows the idea of the iterative relabeling process.

[2] https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets.

- **Deep Graph Kernel (DGK)** (Yanardag & Vishwanathan, 2015): It presents a unified framework and leverages the dependency information between sub-structures by learning their latent representations.
- **Sub2Vec** (Adhikari, Zhang, Ramakrishnan, & Prakash, 2018): It presents an unsupervised algorithm and learns interpretable representations of arbitrary subgraphs to preserve graph properties.
- **Graph2Vec** (Narayanan et al., 2017): It is a neural representation learning approach to learn data-driven distributed representations for graphs.

Traditional semi-supervised methods:

- **EntMin** (Grandvalet & Bengio, 2005): This model adds a loss term to encourage the classifier to output "confident" (low-entropy) predictions on all unlabeled data, regardless of their categories.
- **Π-Model** (Tarvainen & Valpola, 2017): It creates two random augmentations for each sample and minimizes the square difference of these two predictions.
- **Mean-Teacher** (Tarvainen & Valpola, 2017): It uses an exponential moving average of parameters from previous training steps to obtain a stable target output for the unlabeled data.
- **VAT** (Miyato, Maeda, Koyama, & Ishii, 2018): It directly approximates a tiny perturbation added to the input that can most significantly affect the output of the prediction function.

Graph-specific semi-supervised methods:

- **InfoGraph** (Sun et al., 2020): It effectively learns the representations over graphs via maximizing the mutual information between the whole graph representations and the sub-structure representations of different scales.
- **ASGN** (Hao et al., 2020): It adopts a teacher–student framework that a teacher model produces graph representation from a local and global scale, and transfers the knowledge to a student model which targets the property prediction task.
- **GraphCL** (You et al., 2020): It designs some fundamental graph augmentation strategies to incorporate various priors and then conducts graph contrastive learning following the scheme of SimCLR (Chen, Kornblith, Norouzi and Hinton, 2020).
- **JOAO** (You et al., 2021): It improves GraphCL (You et al., 2020) with a bi-level optimization framework to automatically, adaptively, and dynamically select data augmentation for different datasets.

In this paper, we do not compare with SEAL-AI (Li et al., 2019) because the explicit relations among the graph instances are required, which are impractical in our datasets.

Configurations. We implement our proposed GHNN in Pytorch Geometric (Fey & Lenssen, 2019). For simplicity, we adopt a 3-layer GIN (Xu et al., 2019) followed by 1-layer sum-pooling as our GNN backbone. The hidden dimension of GNN is set as 64, the batch size as 64, and the number of epochs is set to 300 for all datasets for a fair comparison. Adam (Kingma & Ba, 2015) is used for optimization because of its effectiveness. The initial learning rate is set to 0.01 and decayed at the rate of 0.0005. The number of hidden graphs is set to 16 and their size equals 5 nodes in the graph kernel network module. The maximum length of random walk P is set to 3. We report the average results with standard deviations of 5 runs for all experiments.

4.2. Performance comparison (RQ1)

The results on seven datasets with a range of baselines are presented in Table 2 and have the following observations.

- The performance of traditional graph methods is not as good as other methods, maybe the reason is that hand-crafted features lead to low generalization and scalability. Instead, it also shows that graph neural networks have better feature extraction ability from graph-structured data.
- The graph-specific semi-supervised learning methods outperform traditional semi-supervised learning techniques (EntMin, Π-Model, Mean-Teacher, and VAT), indicating that the approaches tailoring for domain-specific data are more suitable for complex graph-related tasks than the graph-neural network combined with traditional semi-supervised learning techniques.
- Additionally, our method outperforms the previous state-of-the-art methods on six of seven datasets, which significantly demonstrates the superiority of our proposed model. Comparing our approach with baselines, the performance gain mainly comes from two aspects. On the one hand, we introduce the graph convolutional network module and the graph kernel network module to capture graph topology from complementary views. On the other hand, we optimize harmonic contrastive loss and harmonic consistency loss, allowing our approach to better give prior to high-quality unlabeled data and enhance the communication of the two modules, and further encourage the consistency between the two modules.

Effect of Rates of Labeled Data. To evaluate the effectiveness of our method with various rates of the labeled data in the semi-supervised scenarios, We take the PROTEINS, DD, IMDB-B, and REDDIT-M-5k as examples and exclude the traditional methods owing to their poor performance. From Fig. 4, when we fix all the unlabeled set, we can observe that increasing the size of labeled data effectively improve the performance for all methods. When labeled data is very scarce (e.g., accounting for 1%), the superiority of our GHNN shows that the exploration of graph topology from multi-view as well as the introduction of our novel harmonic semi-supervised learning framework are an efficient way to boost the performance.

Effect of Rates of Unlabeled Data. Labeling data is challenging as well as costly in specialized domains such as chemistry or biology, and their performance is not yet satisfactory when labeled data are scarce. To alleviate this problem, we aim to leverage a large number of unlabeled data to enhance the performance given a limited number of labeled data. Specifically, we fix the labeled set and compare our model performance with other baseline approaches under different rates of unlabeled data. In our experiment, we summarize the results on datasets IMDB-B and COLLAB in Fig. 5. One can observe that graph-specific semi-supervised learning methods outperform other methods in most cases, demonstrating the key role of contrastive learning in making the best use of the unlabeled data. Moreover, our GHNN achieves the best performance consistently, validating the superiority of our framework.

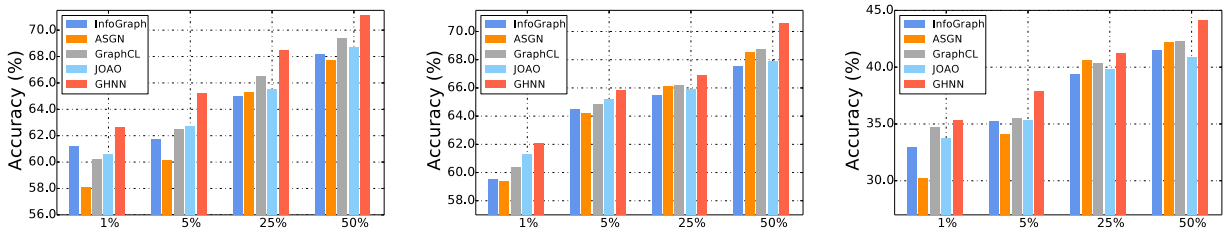
4.3. Ablation study (RQ2)

In this section, we conduct an ablation study to validate the contributions of every component of our GHNN:

- **GCN-Sup.** It trains a GCN module (i.e., f_θ) solely on the labeled data in a fully supervised way.

Table 2
Results on seven benchmark datasets. The best results are marked in bold.

Methods	Datasets						
	PROTEINS	DD	IMDB-B	IMDB-M	REDDIT-B	REDDIT-M-5k	COLLAB
GK	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	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	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	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	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	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	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	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	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	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	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	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	69.4 ± 0.8	68.7 ± 1.2	71.2 ± 2.5	**43.7 ± 1.3**	75.8 ± 1.7	42.3 ± 0.9	66.4 ± 0.6
JOAO	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
GHNN (Ours)	**71.1 ± 0.3**	**70.6 ± 0.4**	**72.3 ± 0.6**	42.8 ± 0.4	**76.3 ± 0.7**	**44.1 ± 0.5**	**67.1 ± 0.3**

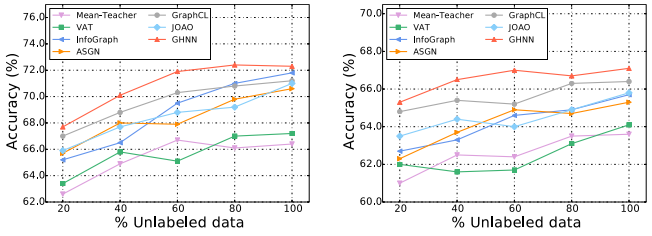


(a) PROTEINS (b) DD (c) REDDIT-M-5k

Fig. 4. Results on datasets *w.r.t.* the amounts of the labeled data (i.e., 1%, 5%, 25% and 50%) and all the unlabeled data.

Table 3
Comparison with several variants for ablation study (in %). The best results are marked in bold.

Methods	Datasets						
	PROTEINS	DD	IMDB-B	IMDB-M	REDDIT-B	REDDIT-M-5k	COLLAB
GCN-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
GKN-Sup	62.6 ± 0.8	61.7 ± 1.2	55.4 ± 1.7	32.7 ± 0.9	65.3 ± 0.6	33.4 ± 2.8	61.2 ± 1.0
GCN-Ensemble	69.3 ± 0.6	68.1 ± 0.5	66.7 ± 2.2	40.3 ± 0.7	75.2 ± 0.6	42.0 ± 0.8	65.7 ± 0.4
GKN-Ensemble	70.1 ± 1.1	66.7 ± 1.6	65.4 ± 2.6	38.7 ± 1.2	74.4 ± 0.9	40.2 ± 1.5	66.3 ± 0.6
GHNN w/o Aug	68.9 ± 0.4	69.2 ± 0.4	69.7 ± 0.3	41.4 ± 0.6	74.8 ± 0.7	41.5 ± 1.6	64.1 ± 1.3
GHNN w/o Fac	70.2 ± 0.5	69.5 ± 0.7	71.7 ± 0.6	42.1 ± 0.4	75.7 ± 0.7	42.9 ± 0.3	66.8 ± 0.2
GHNN w/o Con	70.8 ± 0.3	70.1 ± 0.2	70.8 ± 1.1	41.2 ± 0.5	75.6 ± 0.7	43.2 ± 0.8	66.1 ± 0.3
GHNN (Ours)	**71.1 ± 0.3**	**70.6 ± 0.4**	**72.3 ± 0.6**	**42.8 ± 0.4**	**76.3 ± 0.7**	**44.1 ± 0.5**	**67.1 ± 0.3**



(a) IMDB-B (b) COLLAB

Fig. 5. Results on datasets *w.r.t.* the amounts of the unlabeled data (i.e., 20%, 40%, 60%, 80% and 100%).

- **GKN-Sup.** It trains a GKN module (i.e., g_ϕ) solely on the labeled data in a fully supervised way.
- **GCN-Ensemble.** We replace the GKN module with another GCN module with re-initialization.
- **GKN-Ensemble.** We replace the GCN module with another GKN module with re-initialization.
- **GHNN w/o Aug.** We remove the whole harmonic contrastive loss in the overall loss.
- **GHNN w/o Fac.** We remove the dynamic harmonic factor in the harmonic contrastive loss. Here our harmonic contrastive loss is reduced to traditional contrastive loss (You et al., 2020).
- **GHNN w/o Con.** We remove the harmonic consistency loss in the overall loss.

The performance of all model variants is shown in Table 3. There are several findings from this Table. First, the performance of GCN-Sup is better than GKN-Sup on most datasets. The reason may be that hidden graphs in GKN modules cannot well explore both node attributes and graph topology information without enough guidance. Second, it can be seen that ensemble learning outperforms learning solely on the labeled data, indicating our harmonic semi-supervised framework has better capability to enhance the model performance. Third, both two ensemble models perform worse than our full model, showing the effectiveness of capturing graph topology information in both implicit and explicit views. Finally, our proposed GHNN achieves worse performance without either harmonic contrastive loss and

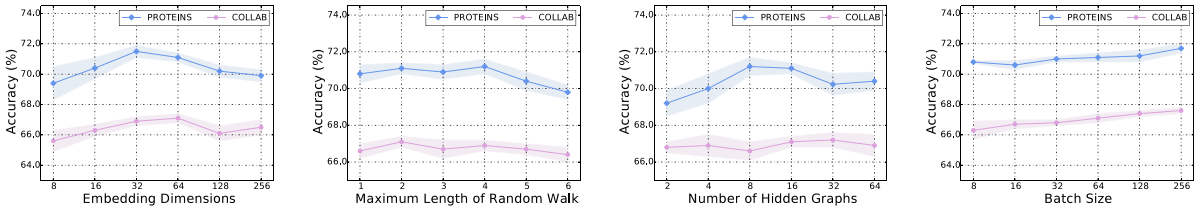


Fig. 6. Performance *w.r.t.* the embedding dimensions, the maximum length of random walk, and the number of hidden graphs, the batch size on two datasets PROTEINS and COLLAB.

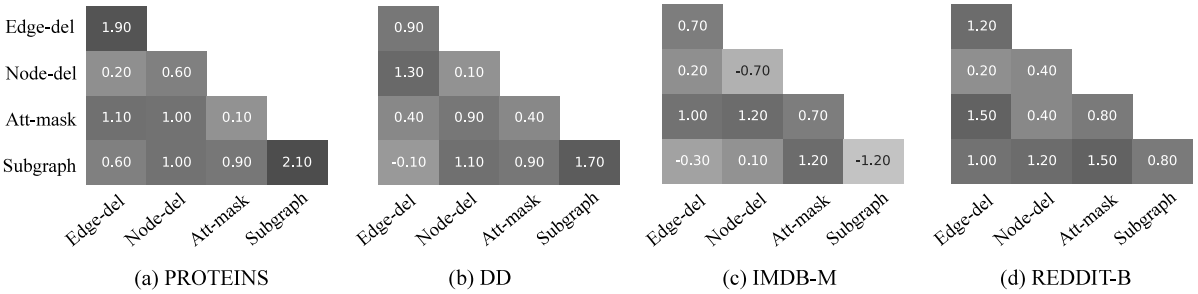


Fig. 7. Accuracy gain (in %) *w.r.t.* different augmented pairs, compared to training without any augmentation. Deeper colors imply better performance gains. The accuracies of baselines without augmentation are 69.6%, 69.3%, 42.3%, 75.5% for the datasets PROTEINS, DD, IMDB-M, and COLLAB respectively.

harmonic consistency loss, which validates that both our two novel harmonic losses can indeed benefit our semi-supervised learning framework for making the best of both labeled and unlabeled data. In view of these findings, we can verify the role and contribution of each component to the final model performance.

4.4. Parameter analysis (RQ3)

Here we conduct some hyper-parameter analysis on the embedding dimensions of hidden layers d, the maximum length of random walk P, and the number of hidden graphs N on both PROTEINS and COLLAB datasets in Fig. 6.

Effect of Embedding Dimensions of Hidden Layers. We first explore the effect of the embedding dimensions of hidden layers d. In particular, we search the dimensions in the range of $\{8, 16, 32, 64, 128\}$. The experimental results are summarized on the first one of Fig. 6. It can be observed that the expansion of dimensions does enhance the performance, but too large dimensions can lead to a degradation of model performance. The reason lies in the over-fitting phenomenon of the model, which unexpectedly learns much of the noise characteristics of the dataset.

Effect of Maximum Length of Random Walk. We then assess the influence of the maximum length of random walk P in the graph kernel module. We vary P in $\{1, 2, 3, 4, 5, 6\}$ while maintaining all other hyper-parameters fixed. The results are shown on the second one of Fig. 6. It can be seen that appropriately increasing the maximum length of random walk P can better explore different types of graph substructures, thus enriching model capacity. However, too-long random walks can lead to path deviation or repetition, which can seriously damage performance.

Effect of Number of Hidden Graphs. We further investigate the effect of numbers of hidden graphs N on the third one of Fig. 6. We vary N in $\{2, 4, 8, 16, 32, 64\}$. As we can see, increasing N almost brings in better performance when the number is small, implying that more hidden graphs can detect more diverse substructures and help in learning graph topology information. Nevertheless, too large N may have a negative impact on performance, because a large number of hidden graphs may introduce too many parameters and redundant information to fit the model.

Effect of Batch Size. Afterward, we evaluate the effect of the batch size M. We vary M in $\{8, 16, 32, 64, 128, 256\}$. It can be observed that a large batch size can consistently improve the performance of our GHNN on the last one of Fig. 6. This observation aligns with the case in the image domain. The reason is that a large enough batch can better represent the whole dataset, and thus contains more negative samples to drive the positive sample to learn more discriminative representations via contrastive learning. It is noted that too-large batch size may cause space complexity problems.

4.5. Augmentation analysis (RQ4)

Finally, to illustrate the impact of data augmentation, we explore four augmentation strategies (edge deletion, node deletion, attribute masking, and subgraph). From Fig. 7, we can see that contrastive learning based on augmentation strategy can indeed make full use of unlabeled data, allowing our method to learn more robust and more generalized representations. Moreover, different datasets may have their preference for some augmentation strategies, and we observed that some augmented pairs may hurt the model performance, which may be because the augmentation strategy used destroys the structure or attributes of the graph in the dataset, thus changing its semantics and leading to performance degradation.

5. Conclusion

In this paper, we propose a novel framework called the GHNN for semi-supervised graph classification. GHNN combines the graph convolutional network module and graph kernel network module to capture graph structural information in both implicit and explicit perspectives. Additionally, we further propose a novel semi-supervised framework to harmonize the training of two modules by prioritizing unlabeled data of high quality and encouraging prediction consistency between the two modules. By fusing the contrastiveness and consistency of the framework, GHNN obtains state-of-the-art performance on benchmark datasets. For future work, we are interested in exploring the effectiveness of our GHNN in other domains including natural language process and material discovery.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

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. 31871342, No. 62106008 and No. 62006004).

References

Adhikari, B., Zhang, Y., Ramakrishnan, N., & Prakash, B. A. (2018). Sub2vec: Feature learning for subgraphs. In *PAKDD*.

Bianchi, F. M., Grattarola, D., & Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In *ICML*.

Borgwardt, K. M., & Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *ICDM*.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., & Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, *21*, 47–56.

Camastra, F., & Verri, A. (2005). A novel kernel method for clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(5), 801–805.

Chen, D., Jacob, L., & Mairal, J. (2020). Convolutional kernel networks for graph-structured data. In *ICML*.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *ICML*.

Chong, Y., Ding, Y., Yan, Q., & Pan, S. (2020). Graph-based semi-supervised learning: A review. *Neurocomputing*, *408*, 216–230.

Dobson, P. D., & Doig, A. J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, *330*(4), 771–783.

Du, S. S., Hou, K., Salakhutdinov, R., Póczos, B., Wang, R., & Xu, K. (2019). Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *NeurIPS*.

Elisseeff, A., & Weston, J. (2001). A kernel method for multi-labelled classification. In *NeurIPS*.

Engel, E., & Dreizler, R. M. (2013). *Density functional theory*. Springer.

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch geometric. In *ICLR Workshop*.

Fu, S., Liu, W., Guan, W., Zhou, Y., Tao, D., & Xu, C. (2021). Dynamic graph learning convolutional networks for semi-supervised classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *17*(1s), 1–13.

Fu, S., Liu, W., Zhang, K., Zhou, Y., & Tao, D. (2021). Semi-supervised classification by graph p-Laplacian convolutional networks. *Information Sciences*, *560*, 92–106.

Gao, H., & Ji, S. (2019). Graph u-nets. In *ICML*.

Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *Proceedings of computational learning theory and kernel machines* (pp. 129–143).

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *ICML*.

Grandvalet, Y., & Bengio, Y. (2005). Semi-supervised learning by entropy minimization. In *NeurIPS*.

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *NeurIPS*.

Hao, Z., Lu, C., Huang, Z., Wang, H., Hu, Z., Liu, Q., et al. (2020). ASGN: An active semi-supervised graph neural network for molecular property prediction. In *KDD*.

Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *ICML*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kojima, R., Ishida, S., Ohta, M., Iwata, H., Honma, T., & Okuno, Y. (2020). kGCN: a graph-based deep learning framework for chemical structures. *Journal of Cheminformatics*, *12*, 1–10.

Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, *5*(1), 1–42.

Laine, S., & Aila, T. (2017). Temporal ensembling for semi-supervised learning. In *ICLR*.

Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In *ICML*.

Lee, D.-H., 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*.

Li, J., Rong, Y., Cheng, H., Meng, H., Huang, W., & Huang, J. (2019). Semi-supervised graph classification: A hierarchical graph perspective. In *WWW*.

Lin, W., Gao, Z., & Li, B. (2020). Shoestring: Graph-based semi-supervised classification with severely limited labeled data. In *CVPR*.

Liu, C., Wen, L., Kang, Z., Luo, G., & Tian, L. (2021). Self-supervised consensus representation learning for attributed graph. In *ACMMM*.

Long, Q., Jin, Y., Wu, Y., & Song, G. (2021). Theoretically improving graph neural networks via anonymous walk graph kernels. In *WWW*.

Lu, C., Liu, Q., Wang, C., Huang, Z., Lin, P., & He, L. (2019). Molecular property prediction: A multilevel quantum interactions modeling perspective. In *AAAI*.

Miyato, T., Maeda, S.-i., Koyama, M., & Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(8), 1979–1993.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., & Jaiswal, S. (2017). Graph2vec: Learning distributed representations of graphs. arXiv preprint arXiv:1707.05005.

Nikolentzos, G., & Vazirgiannis, M. (2020). Random walk graph neural networks. In *NeurIPS*.

Oord, A. v. d., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.

Pan, E., & Kang, Z. (2021). Multi-view contrastive graph clustering. In *NeurIPS*.

Sajjadi, M., Javanmardi, M., & Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NeurIPS*.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *12*(9), 2539–2561.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., & Borgwardt, K. (2009). Efficient graphlet kernels for large graph comparison. In *AISTATS*.

Sun, F.-Y., Hoffmann, J., Verma, V., & Tang, J. (2020). Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*.

Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. In *ICLR*.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., & Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, *11*, 1201–1242.

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

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *ICML*.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *ICLR*.

Yanardag, P., & Vishwanathan, S. (2015). Deep graph kernels. In *KDD*.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*.

You, Y., Chen, T., Shen, Y., & Wang, Z. (2021). Graph contrastive learning automated. In *ICML*.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., & Shen, Y. (2020). Graph contrastive learning with augmentations. In *NeurIPS*.

Yuan, H., & Ji, S. (2020). Structpool: Structured graph pooling via conditional random fields. In *ICLR*.

Zeng, J., & Xie, P. (2021). Contrastive self-supervised learning for graph classification. In *AAAI*.

Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. In *AAAI*.