STRUCTURE-SENSITIVE GRAPH DICTIONARY EMBEDDING FOR GRAPH CLASSIFICATION

Paper ID: 431

ABSTRACT

Graph structure expression plays an important role in distinguishing various graphs. In this work, we propose a Structure-Sensitive Graph Dictionary Embedding (SS-GDE) framework to transform input graph into the space of graph dictionary for the graph classification task. Instead of a naive use of base graph dictionary, we propose variational graph dictionary adaptation (GDA) to generate a personalized dictionary (named adapted graph dictionary) for catering each input graph. In particular for the adaptation, the Bernoulli sampling is introduced to adjust substructures of base graph keys, which increases the expression capacity of base dictionary tremendously. To make cross-graph measurement sensitive as well as stable, multi-sensitivity Wasserstein encoding is proposed to produce the embeddings by designing multi-scale attention on optimal transport. To optimize the framework, we introduce mutual information as the objective, which just deduces to variational inference of adapted graph dictionary. We perform our SS-GDE on multiple datasets of graph classification, and the experimental results demonstrate the effectiveness and the superiority over the state-of-the-art methods.

1 INTRODUCTION

Graph is usually composed of one node set and one edge set, where nodes represent individual objects and edges denote relationships among them. Due to rather flexible structures, graphs have been widely-used to model ubiquitous irregular data in various real-world and scientific fields, such as biological graphs Duvenaud et al. (2015); Hamilton et al. (2017) and social networks. To obtain powerful representation ability on graphs, graph deep learning is flourishing in recent years, and has made many milestone progresses on several graph-related tasks including community detection, document classification, et al. Among them, an essential issue is graph classification, which endeavors to learn the most discriminant representation of graphs in certain measurement metrics.

The existing graph classification methods generally fall into two main streams: graph kernel-based algorithms and graph neural networks (GNNs). As the traditional and classical representative, graph kernels Shervashidze et al. (2009); Borgwardt & Kriegel (2005) measure similarities of graphs in a lifting high-dimensional space of structure statistics quantified by graphlets, where structural information is well preserved. As a contrast, GNNs Kipf & Welling (2016); Veličković et al. (2017); Xu et al. (2018) attempt to extract high-level and discriminative features by stacking multiple neural network layers, wherein the adjacent information is aggregated in an iterative manner. Hence, GNNs are capable of exploiting local structures of graph to some extent, and have achieved better results in graph classification due to the essence of deep architecture.

Even though much considerable progress has been made by GNNs in graph classification, the over-smoothness Xu et al. (2018) limits the expression ability of features because of the structure confusion during information aggregation. When retrospecting graph kernels again, the good preservation ability of graph structures is really fascinating to GNNs. Hereby, we naturally pose such an issue why not assemble the powerful abilities of GNN for deep representation and graph kernel for structure preservation. Probably inspired from this idea, recently, an interesting work, deep graph dictionary learning Zhang et al. (2021), raised a novel solution by encoding input graph with graph dictionary in a GNN architecture, and meantime achieved promising performance. For graph dictionary learning, however, two major problems remain to be addressed: i) a fixed graph dictionary usually has limited capacity to express a giant amount of graphs, as combinatorial-explosive graph structures with exponential-order magnitude would be overwhelming for such a dictionary; ii) the similarity measurement across input graph and graph dictionary key should be stable, also sensitive to local structural variations, which guarantees high representation ability on graph dictionary.

To tackle the two issues above, in this work, we propose a Structure-Sensitive Graph Dictionary Embedding (SS-GDE) framework to facilitate graph representation modeling for graph classification. To fully liberate graph dictionary, we propose variational graph dictionary adaptation (VGDA) to conduct individual structure selections from graph dictionary keys for each input graph. Such a selective adjustion tremendously expands the capacity of original fixed graph dictionary (called base graph dictionary, BGD), and generates a personalized specific dictionary adapted for each input graph, which we call adapted graph dictionary (AGD) versus base graph dictionary. To effectively choose the corresponding substructures from base graph keys, we introduce a Bernoulli sampling to be learnt during the variational inference in VGDA. For the measurement between input graph and adaptive dictionary keys, we employ cross-graph Wasserstein distance, which is rather stable as verified in Zhang et al. (2021). But to increase the sensitivity for cross-graph correlation, we propose multi-sensitivity Wasserstein encoding (MS-WE) by introducing multi-scale attention on optimal transport, which could adaptively capture those important local correlation patterns for the final accurate representation of input graph. To optimize the proposed framework, we introduce mutual information as the objective, which just deduces to variational inference of adapted graph dictionary. To evaluate the SS-GDE framework, extensive experiments are conducted on multiple graph classification datasets, and the experimental results validate the effectiveness and the superiority over the state-of-the-art methods.

In summary, the contributions of our work are four-fold: i) propose a structure-sensitive graph dictionary embedding framework to promote deep graph learning for graph classification; ii) propose variational graph dictionary adaptation to release the potential capacity of base graph dictionary; iii) design multi-sensitivity Wasserstein encoding to guarantee the sensitivity as well as stability of cross-graph measurement; iv) report new state-of-the-art results on some datasets.

2 Related work

In this section, we first review the previous methods of graph classification, then introduce works related to inherently interpretable models and Wasserstein distance learning.

Graph Classification. Many recent techniques have been proposed to solve the graph classification problem. Some early approaches dedicated to building kernel functions to measure similarities among graphs. These kernel-based methods decompose graphs into sub-structure such as random-walks (Gärtner et al., 2003), shortest path (Borgwardt & Kriegel, 2005), graphlets (Shervashidze et al., 2009) and subtrees (Shervashidze et al., 2011). GNN can directly operate on graph-structured data to extract expressive graph-level representation by stacking multiple neural network layers, which can aggregate neighbor node features and have achieved promising performance in the graph classification task. Besides, various convolution (Kipf & Welling, 2016; Niepert et al., 2016; Luo et al., 2017) and pooling operations were proposed to learn robust node features and graph representations in recent years. Graph Convolutional Network(GCN) (Kipf & Welling, 2016) proposed a layer-wise propagation rule based on a first-order approximation of spectral convolution on graphs via the Chebyshev polynomial iteration. Graph Attention Network (GAT) (Veličković et al., 2017) highlighted more information nodes by assigning different weights to different nodes in the neighborhood. In addition to the above-mentioned methods, many pooling strategies have emerged, which can be categorized as node selection (Lee et al., 2019; Li et al., 2020; Nouranizadeh et al., 2021), graph coarsening (Ying et al., 2018; Yuan & Ji, 2020) and other methods (Baek et al., 2021; Li et al., 2019). SAGPool (Lee et al., 2019) employed self-attention mechanism, which uses graph convolution to calculate the attention scores to distinguish the nodes that should be dropped and the nodes that should be retained. DiffPool (Ying et al., 2018) learned a differentiable soft assignment to cluster nodes at each layer.

Wasserstein Distance learning. Wasserstein distance measures the difference between two probability distributions defined on a given metric space by leveraging the OT principle and has been widely used in machine learning and pattern matching fields. Numerous algorithms (Frogner et al., 2015; Titouan et al., 2019; Chen et al., 2020) were proposed to learn representation from graphs or measure this distance. For instance, (Togninalli et al., 2019) proposed that calculate the Wasserstein distance between the node feature vector distributions of two graphs to find subtler differences.

Fused Gromov-Wasserstein (FGW) (Titouan et al., 2019) was introduced to consider both features and structure information in the optimal transport problem. Some other works (Schmitz et al., 2018; Rolet et al., 2016; Vincent-Cuaz et al., 2021) attempted to conduct dictionary learning in W-space, e.g., using W-distance for fitting data term.

Inherently Interpretable models. The interpretability of models refers to understanding their internal mechanism. Many works have been proposed to extract meaningful data patterns for prediction by post-hoc methods (Ying et al., 2019; Luo et al., 2020) as well as inherently interpretable models (Wu et al., 2022b; Miao et al., 2022). However, To make the model more interpretable, it may need to sacrifice prediction performance. DIR (Wu et al., 2022b) proposed to create multiple distributions by conducting interventions to the training distribution and to filter out the spurious and unstable patterns. GSAT (Miao et al., 2022) introduced stochastic attention to block the task-irrelevant graph components information while learning stochasticity-reduced attention to select task-relevant subgraphs to provide the interpretability of the model.

Compared to those existing methods above, our SS-GDE has apparently different aspects: (1) Instead of a naive use of base graph dictionary, we introduce the Bernoulli sampling to adjust substructures of base graph keys to generate a personalized dictionary adapted for each input graph. Such a selective adjustion significantly expands the capacity of original fixed graph dictionary. Furthermore, we introduce mutual information as objective, which deduces to the variational inference of adapted graph dictionary. (2) We design the MS-WE module to use cross-graph Wasserstein distance for the stability of embedding and further introduce multi-sensitivity regularization to improve the sensitivity of structure variations. Hence, those important local correlation patterns could be well captured for the accurate representation of input graphs.

3 The proposed Method

In the following parts, vectors/matrices are denoted with lowercase/uppercase letters in boldface, and $\square^\intercal$ represents the transpose. A calligraphic symbol may either indicate a tuple, e.g. the graph tuple $\mathcal{G}_i$ consisting of node/edge sets, or simply a set, e.g. a node set $\mathcal{V}_i$. $\square^D$ means that the matrix/vector corresponds to the graph dictionary, e.g. the graph dictionary denoted as $\mathcal{D} = \{\mathcal{G}_1^D, \cdots, \mathcal{G}_K^D\}$ (K is the number of the graph keys). In this section, we first overview the whole architecture of our proposed SS-GDE framework, then describe those main learning processes in detail.

3.1 Overview

The whole architecture of the proposed SS-GDE framework is shown in Fig. 1. In general, it contains two main learning modules: variational graph dictionary adaptation (VGDA) and multi-sensitivity Wasserstein encoding (MS-WE). Before VGDA, a base graph dictionary (BGD) $\widetilde{\mathcal{D}}$ is first constructed to support the subsequent embedding representation of input graph. For the rough graphs in dictionary or input set, graph convolution neural network (GCNN) may be used to learn primary expression of each node. Given an input graph $\mathcal{G}$, the VGDA learns an adapted graph dictionary $\mathcal{D}$ from base dictionary through learning Bernoulli sampling during cross-correlating input and dictionary keys. Such a process generates more expressive structural dictionaries for the next embedding representation. The detail of VGDA could be found in Section 3.2. The adapted graph dictionary $\mathcal{D}$ is fed into the MS-WE module to produce the embedding of input graph. To make better cross-graph embedding, in MS-WE, we use cross-graph Wasserstein distance for the stability of embedding and introduce multi-sensitivity regularization for the sensitivity of structure variations. The detail of VGDA could be found in Section 3.3. Finally, the resulting embeddings pass through fully-connected layers for low-dimensional representations. To optimize the SS-GDE framework, mutual information is introduced as the objective, which deduces the variational inference of the AGD, as derived in Section 3.4. In the training process, the whole architecture as well as the base graph dictionary can be optimized in an end-to-end tuning mode through back-propagation.

3.2 Variational Graph Dictionary Adaptation

We use $\mathcal{G}_i = (\mathcal{V}_i, \mathbf{X}_i, \mathbf{A}_i)$ $\{\mathbf{X}_i \in \mathbb{R}^{n\times d}, \mathbf{A}_i \in \mathbb{R}^{n\times n}\}$ to denote the i-th input graph, and $\widetilde{\mathcal{G}}_j^D = (\widetilde{\mathcal{V}}_j^D, \widetilde{\mathbf{X}}_j^D, \widetilde{\mathbf{A}}_j^D)$ $\{\widetilde{\mathbf{X}}_i^D \in \mathbb{R}^{n_d\times d}, \widetilde{\mathbf{A}}_i^D \in \mathbb{R}^{n_d\times n_d}\}$ to denote the j-th dictionary key in the BGD $\widetilde{\mathcal{D}}$.

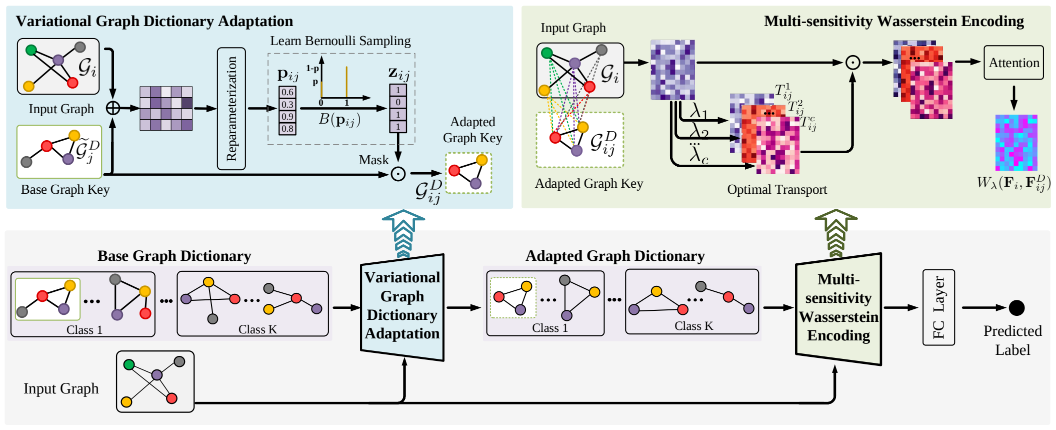


Figure 1: The architecture of the proposed SS-GDE framework. It contains two main modules named the variational graph dictionary adaptation (VGDA) and multi-sensitivity Wasserstein encoding (MS-WE). Given one input graph, as the first step, a base graph dictionary (BGD) is constructed to support the subsequent embedding. Then, the input graph and BGD are fed into the VGDA module to learn the adapted graph dictionary (AGD) corresponding to each input. In this process, the Bernoulli sampling is learned to extract substructures from the BGD by cross-correlating input and its dictionary keys. Next, the learned AGD is fed into the MS-WE module to produce the embedding of the input graph. To make better cross-graph embedding, the MS-WE employs the multi-sensitivity regularization before the optimal transport to improve the sensitivity of structure variations, and further introduces the attention mechanism to capture salient structural patterns. Finally, the obtained embeddings pass through fully-connected layers for low-dimensional representations to facilitate the classification. More details can be found in the main body.

After GCNN for primary encoding, the learned features corresponding to $\mathbf{X}_i$ and $\widetilde{\mathbf{X}}_j^D$ are denoted as $\mathbf{F}_i$ and $\widetilde{\mathbf{F}}_j^D$. For each input graph, the VGDA aims to learn a more suitable AGD denoted as $\mathcal{D}$ from the BGD $\widetilde{\mathcal{D}}$. This is done by learning Bernoulli sampling factors that adaptively select substructures from the BGD. Formally, given $\mathcal{G}_i$ and $\widetilde{\mathcal{G}}_j^D$, together with the sampling factor $\mathbf{z}_{ij}$, the selection function for generating the corresponding graph $\mathcal{G}_{ij}^D \in \mathbb{R}^{n\times d}$ in $\mathcal{D}$ can be denoted as:

$$\mathcal{G}_{ij}^D = s(\widetilde{\mathcal{G}}_j^D, \mathbf{z}_{ij}), \text{ s.t. } \mathbf{z}_{ij} = g_\phi(\mathbf{F}_i, \widetilde{\mathbf{F}}_j^D), \; p(\mathbf{z}_{ij}) \sim B(\mathbf{p}_{ij}). \tag{1}$$

Specifically, $\mathbf{z}_{ij}$ conforms to the Bernoulli distribution based on the probability vector $\mathbf{p}_{ij}$, and the elements in $\mathbf{z}_{ij}$ are either 0 or 1. Based on the selection function $s(\cdot)$ with $\mathbf{z}_{ij}$, a certain number (i.e. n) of nodes, together with the corresponding edges, can be selected from $\widetilde{\mathcal{G}}_j^D$. $g_\phi(\cdot)$ is the function to learn the sampling factor $\mathbf{z}_{ij}$ with the parameters denoted as ϕ.

Here, a crucial step is to derive $\mathbf{z}_{ij}$ based on $\mathbf{F}_i$ and $\mathbf{F}_j^D$. Rather than learning $\mathbf{z}_{ij}$ in a deterministic way, e.g. constructing networks with the attention mechanism, we infer $\mathbf{z}_{ij}$ through a probabilistic manner. Compared to the deterministic way, this stochastic learning endows the framework better generalization ability and interpretability. However, it's rather non-trivial to infer $\mathbf{z}_{ij}$ through the Bayes rule $p(\mathbf{z}_{ij}|\mathbf{F}_i, \mathbf{F}_j^D) = p(\mathbf{z}_{ij})p(\mathbf{F}_i, \mathbf{F}_j^D|\mathbf{z}_{ij})/p(\mathbf{F}_i, \mathbf{F}_j^D)$ due to the intractability. Hence, we resort to the variational inference to approximate the intractable true posterior $p(\mathbf{z}_{ij}|\mathbf{F}_i, \mathbf{F}_j^D)$ with $q_\phi(\mathbf{z}_{ij}|\mathbf{F}_i, \mathbf{F}_j^D)$ and meanwhile constrain the KL-divergence $D_{KL}(q_\phi(\mathbf{z}_{ij}|\mathbf{F}_i, \mathbf{F}_j^D)||p(\mathbf{z}_{ij}|\mathbf{F}_i, \mathbf{F}_j^D))$ between them. Specifically, in Section 3.4, we describe how this KL-divergence can be deduced from the mutual information in detail. As a result, for the input $\mathbf{F}_i$ and BGD $\widetilde{\mathcal{D}}$, the corresponding AGD $\mathcal{D}_i = \{\mathcal{G}_{i1}^D, \mathcal{G}_{i2}^D, \cdots, \mathcal{G}_{iK}^D\}$ can be obtained.

3.3 Multi-sensitivity Wasserstein Embedding

In the MS-WE module, we use the Wasserstein metric for cross-graph correlation across input graph and adapted graph keys. Moreover, the multi-sensitivity regularization is introduced to improve the sensitivity to structure variations. Formally, given the input graph $\mathcal{G}_i$, the cross-graph correlation with the adapted graph key $\mathcal{G}_j^D$ can be formularized as:

$$h_{ij}^{\lambda} = W_{\lambda}(\mathbf{F}_i, \mathbf{F}_{ij}^D) = \langle \mathbf{T}_{ij}^{\lambda}, \mathbf{M}_{ij} \rangle. \tag{2}$$

Each element in $\mathbf{M}_{ij}$ calculates pair-wise squared Euclidean distances between cross-graph nodes from $\mathcal{G}_i$ and $\mathcal{G}_{ij}^D$, respectively. $\langle \mathbf{A}, \mathbf{B} \rangle = tr(\mathbf{A}^\intercal \mathbf{B})$. $\mathbf{T}_{ij}^{\lambda}$ represents the OT matrix between $\mathbf{F}_i$ and $\mathbf{F}_{ij}^D$ based on $\mathbf{M}_{ij}$, and the OT problem can be solved by the Sinkhorn's fixed point iterations:

$$\mathbf{T}_{ij}^{\lambda} = \mathbf{u}_{ij} \mathbf{1}_{N_1} \odot \mathbf{K}_{ij} \odot \mathbf{1}_{N_2} \mathbf{v}_{ij}^\intercal, \text{ s.t. } \mathbf{K}_{ij} = e^{-\lambda \mathbf{M}_{ij}}. \tag{3}$$

$\mathbf{u}_{ij}$ and $\mathbf{v}_{ij}$ are initialized as all-1 vectors and kept updating during the Sinkhorn iteration. More details can be found in (Cuturi, 2013). Hence, given the AGD $\mathcal{D}_i = \{\mathcal{G}_{i1}^D, \mathcal{G}_{i2}^D, \cdots, \mathcal{G}_{iK}^D\}$, the corresponding embedding $\mathbf{h}_i^{\lambda} = [h_{i1}^{\lambda}, \cdots, h_{iK}^{\lambda}]$ can be obtained for $\mathcal{G}_i$.

Specifically, the regulation λ controls the sensitivity of local information between nodes across two graphs. The larger value of λ, the less sensitive to the local correlation across graphs. As large structural variation exists among graphs, one unified λ may not well handle it for all graph samples. To address this issue, we conduct the MS-WE through two steps: (1) to calculate multi-sensitivity (e.g. C-sensitivity) embeddings denoted as $[\mathbf{h}_i^{\lambda_1}, \cdots, \mathbf{h}_i^{\lambda_C}]$ based on Eqn. (2); (2) aggregating multi-sensitivity embeddings with the attention mechanism:

$$\widehat{\mathbf{h}}_i = \sum_{j=1}^{C} \alpha_j \mathbf{h}_i^{\lambda_j}, \text{ s.t. } \alpha_j = \frac{e^{(\mathbf{h}_i^{\lambda_j})^\intercal \mathbf{w}_m}}{\sum_{l=1}^{C} e^{(\mathbf{h}_i^{\lambda_l})^\intercal \mathbf{w}_m}}. \tag{4}$$

Specifically, $\mathbf{w}_m$ denotes the projection parameter for the attention mechanism.

3.4 The Framework Optimization

To train our model, we employ the mutual information to measure the relationship between the obtained embeddings and their corresponding labels based on the whole embedding process $f_{\Psi,\phi}(\cdot) : (\mathcal{G}, \widetilde{\mathcal{D}}) \to \mathcal{Y}$. Here, ϕ denotes the parameter set for learning the Bernoulli sampling set $\mathcal{Z} = \{\mathbf{z}_{i1}, \cdots, \mathbf{z}_{iK}\}$, and Ψ denotes the set of the other parameters involved in the learning process. Then, the optimization is as follows:

$$\begin{aligned}
\tilde{\Psi}, \tilde{\phi} &= argmax\ I(\mathbf{y}, f_{\Psi,\phi}(\mathcal{G}, \widetilde{\mathcal{D}})) \\
&= argmax\ -H(\mathbf{y}|f_{\Psi,\phi}(\mathcal{G}, \widetilde{\mathcal{D}})) + H(\mathbf{y}) \\
&= argmax\ -H(\mathbf{y}|f_{\Psi,\phi}(\mathcal{G}, \widetilde{\mathcal{D}})) \\
&= argmax\ E_{\mathbf{y}|\mathcal{G},\widetilde{\mathcal{D}}}(\log p_{\Psi,\phi}(\mathbf{y}|\mathcal{G}, \widetilde{\mathcal{D}})) \\
&= argmax\ E_{\mathbf{y}|\mathcal{G},\widetilde{\mathcal{D}}}(\log \int p_{\Psi}(\mathbf{y}|\mathcal{G}, \widetilde{\mathcal{D}}, \mathcal{Z}) p_{\phi}(\mathcal{Z}|\mathcal{G}, \widetilde{\mathcal{D}}) d\mathcal{Z}) \\
&\geq argmax\ E_{\mathbf{y}|\mathcal{G},\widetilde{\mathcal{D}}}(E_{q_{\phi}(\mathcal{Z})}(\log p_{\Psi}(\mathbf{y}|\mathcal{G}, \mathcal{D})) \\
&\qquad\qquad - D_{KL}(q_{\phi}(\mathcal{Z})||p_{\phi}(\mathcal{Z}|\mathcal{G}, \widetilde{\mathcal{D}})).
\end{aligned} \tag{5}$$

Here, $I(\cdot)$ means the mutual information. $H(\mathbf{y})$ is the constant entropy for the random variable of labels. Specifically, for each input $\mathcal{G}_i$, we assume that those sampling factors $\mathbf{z}_{i1}, \cdots, \mathbf{z}_{iK}$ are independent and identically distributed, and each of them conforms the Bernoulli distribution. Hence, the KL divergnecy can be rewritten as:

$$\begin{aligned}
D_{KL}(q_{\phi}(\mathcal{Z})||p_{\phi}(\mathcal{Z}|\mathcal{G}, \widetilde{\mathcal{D}})) &= \sum_i \sum_{j=1}^{K} D_{KL}(q_{\phi}(\mathbf{z}_{ij})||p_{\phi}(\mathbf{z}_{ij}|\mathcal{G}_i, \widetilde{\mathcal{G}}_j^D)) \\
&= \sum_i \sum_{j=1}^{K} \mathbf{p}_{ij} \log \frac{\mathbf{p}_{ij}}{\widehat{\mathbf{p}}_{ij}} + (1 - \mathbf{p}_{ij}) \log \frac{1 - \mathbf{p}_{ij}}{1 - \widehat{\mathbf{p}}_{ij}}.
\end{aligned} \tag{6}$$

$q_{\phi}(\mathbf{z}_{ij})$ is the expected Bernoulli distribution, i.e. $q_{\phi}(\mathbf{z}_{ij}) \sim B(\widehat{\mathbf{p}}_{ij})$ where $\widehat{\mathbf{p}}_{ij}$ denotes the pre-defined probability vector, and $\mathbf{p}_{ij}$ denotes the learned probability vector. For the intractable

$p_\phi(\mathbf{z}_{ij}|\mathcal{G}_i, \widetilde{\mathcal{G}}_j^D)$, to make the variational inference optimizable, we introduce the reparameterization trick to derive $p_\phi(\mathbf{z}_{ij}|\mathcal{G}_i, \widetilde{\mathcal{G}}_j^D) \sim B(\mathbf{p}_{ij})$ with the following formulation:

$$\mathbf{p}_{ij} = \sigma_r(\mathbf{M}_{ij}^\intercal \mathbf{w}_r), \tag{7}$$

where $\mathbf{w}_r \in \mathbb{R}^n$ is the projection vector, and $\mathbf{M}_{ij}$ calculates the pairwise squared Euclidean distance between two cross-graph nodes from $\mathcal{G}_i$ and $\widetilde{\mathcal{G}}_j^D$.

By applying the reparameterization, the whole framework can be optimized through back-propagation according to Eqn. (5). Specifically, the first term (denoted as L_y) in Eqn. (5) constraints the accuracy for the classification task, while the second term (denoted as L_{KL}) in Eqn. (5) minimizes the distribution difference between $q_\phi(\mathcal{Z})$ and $p_\phi(\mathcal{Z}|\mathcal{G}, \mathcal{D})$. To better balance the influence of the two terms, we introduce a trade-off coefficient β in Eqn. (5) to transform the whole optimization objective as:

$$L = L_y - \beta L_{KL}. \tag{8}$$

4 Experiments

In this section, we first introduce the public datasets used to evaluate our model, as well as describe the implementation details. We then compare the proposed SS-GDE model with multiple state-of-the-art methods. Finally, we conduct ablation analysis to dissect the SS-GDE.

4.1 Datasets

Table 1: Summary of Graph Datasets

Datasets	Graphs Num	Average Nodes	Average Edges	Node labels	Classes
MUTAG	188	17.9	2.2	7	2
PTC	344	25.6	2.0	19	2
PROTEINS	1113	39.1	3.7	3	2
IMDB-BINARY	1000	19.8	9.8	-	2
IMDB-MULTI	1500	13.0	10.1	-	3
COLLAB	5000	74.5	65.9	-	3

To comprehensively evaluate the effectiveness of SS-GDE, we conduct experiments on six widely used public datasets in the graph classification task. These datasets can be divided into two categories: bioinformatics datasets (MUTAG (Debnath et al., 1991), PTC (Helma et al., 2001), PROTEINS (Borgwardt et al., 2005)) and social network datasets (COLLAB, IMDB-BINARY and IMDB-MULTI (Yanardag & Vishwanathan, 2015)). The corresponding summary can be found in Table 1.

Bioinformatics datasets. MUTAG is a nitro compounds dataset containing 188 samples with seven discrete node labels. These samples are divided into two classes. PROTEINS comprises 1113 protein structures of secondary structure elements (SSEs) with three discrete node labels. PTC consists of 344 chemical compound networks divided into two categories, showing carcinogenicity for male and female rats. Moreover, each node is annotated with 19 labels.

Social Network Datasets. IMDB-BINARY and IMDB-MULTI are both movie collaboration datasets derived from IMDB, where each graph represents a movie with nodes corresponding to actors/actresses. If two actors appear in the same film, there will be an edge between their corresponding nodes. BINARY and MULTI stand for the number of classes. COLLAB is a scientific dataset where each graph represents a collaborative network between an affiliated researcher and other researchers from 3 physical domains, labeled as the researcher's physical field.

4.2 Experiment Setup

Implementation Details. For input graph initialization, each node of the input graph is described with a one-hot vector according to its node label, and the edges are set the same as those pre-defined in the datasets. For the base graph dictionary construction, multiple groups of graph keys, where each group corresponds to a fixed number of samples from each class, are randomly selected from the training set. In this experiment, the fixed number of keys is set to 14 for all datasets. To learn primary expression with GCNN, two encoders with the same structure, i.e. the three-layer GCNs, are employed for input samples and the base graph dictionary, respectively. Specifically, the output dimensions of the three layers are 256,128,32. In the MS-WE module, 8-sensitivity regularization is employed. Then, the output embeddings of the MS-WE module further pass two fully connected layers for classification. In the training stage, the framework is trained for 500 epochs with a learning rate of 0.001 and the weight decay of 10^{-4}. In this process, almost all the parameters in the framework, together with the base graph dictionary, are optimized through backpropagation by using the Adam optimizer. One exception is the three-layer GCN that corresponds to the base dictionary. Specifically, its parameters are optimized according to the three-layer GCN of the inputs through the momentum mechanism. The momentum coefficient is set to 0.999. The trade-off parameter β in Eqn. (8) is set to 0.001, while the pre-defined probability vector $\widehat{\mathbf{p}}_{ij}$ in Eqn. (6) is set to all-0.5 vectors.

Protocol. According to the previous literature, the same 10-fold cross-validation protocol is strictly followed to evaluate the classification performance of our proposed method for a fair comparison. Specifically, We randomly split the dataset into ten sections, where nine sections are the training set, and the remaining one section is the testing set. We use the average accuracy and stand deviation of the ten folds as the final reported performance.

4.3 Experiment Results

We compare the SS-GDE framework with a range of state-of-the-art methods as follows: (1) graph kernel-based methods: GK (Shervashidze et al., 2009), DGK (Yanardag & Vishwanathan, 2015), WL (Shervashidze et al., 2011), (2) Neural network based methods: PSCN (Niepert et al., 2016), GCN (Kipf & Welling, 2016), NgramCNN (Luo et al., 2017), HRN (Wu et al., 2022a), GIN-0 (Xu et al., 2018), U2GNN (Nguyen et al., 2022), GNTK (Du et al., 2019), PPGN (Maron et al., 2019), SLIM (Zhu et al., 2022), CAL (Sui et al., 2022), GLA (Yue et al., 2022), and WGDL (Zhang et al., 2021).

Table 2 shows the experimental results of SS-GDE and the comparison with other approaches on the six public datasets. We have the following observations:

1. Generally, the performance of graph kernel-based methods (GK, DGK, WL) is usually lower than those GNN-based methods. This may be attributed to the low expression power caused by the usually employed hand-crafted features, and the limited feature learning ability of the two-stage learning process instead of the global optimization. Among the graph kernel-based methods, WL achieves relatively high performance on most datasets with the average performance gain of 3% over GK and DGK, as it defines a family of efficient kernels based on Weisfeiler-Lehman sequences of graphs. However, its performance is still lower than those of GNNs that stack neural network layers into a deep architecture.

2. GNNs, especially GCN variants involved methods, achieve good experimental for the classification task. Specifically, several recent works improve GCN by either developing the pooling algorithm or introducing the attention mechanism for structure modeling, which further promotes the performance. According to Table 2, our SS-GDE model achieves the state-of-the-art performances on five public datasets, and also a comparable performance on the left COLLAB dataset. This demonstrates the robustness of our framework against graph variation. In contrast, previous methods cannot perform well on all the six datasets, while the second highest performances on different datasets are obtained by six different methods. Compared with the GNTK on the other five datasets, the average performance gain of more than 4% is obtained by our SS-GDE. Moreover, compared to the WGDL using a naive fixed graph dictionary and single-regularization in cross-graph modeling, the improvement of 2% on average is obtained by the SS-GDE. All above verify the effectiveness of our framework.

Table 2: Comparsion with the state-of-the-art-methods. The best accuracies are in bold. $*$ indicates the second highest performance.

Datasets	MUTAG	PTC	PROTEINS	IMDB-BINARY	IMDB-MULTI	COLLAB
GK	81.66±2.11	57.26±1.41	71.67±0.55	65.87±0.98	43.89±0.38	72.84±0.28
DGK	82.66±1.45	57.32±1.13	71.68±0.50	66.96±0.56	44.55±0.52	73.09±0.25
WL	82.72±3.00	56.97±2.01	73.70±0.50	72.86±0.76	50.55±0.55	79.02±1.77
PSCN	92.63±4.21	62.29±5.68	75.89±2.76	71.00±2.29	45.23±2.84	72.60±2.15
NgramCNN	$94.99 \pm 5.63^*$	68.57±1.72	75.96±2.98	71.66±2.71	50.66±4.10	-
GCN	87.20±5.11	-	75.65±3.24	73.30±5.29	51.20±5.13	81.72±1.64
GIN-0	89.40±5.60	64.60±7.00	76.20±2.80	75.10±5.10	52.30±2.80	80.20±1.90
GNTK	90.00±8.50	67.90±6.90	75.60±4.20	76.90±3.60	52.80±4.60	**83.60 ± 1.22**
PPGN	90.55±8.70	66.17±6.54	77.20±4.73	73.00±5.77	50.46±3.59	81.38±1.42
U2GNN	89.97±3.65	69.63±3.60	$78.53 \pm 4.07^*$	77.04±3.45	$53.60 \pm 3.53^*$	77.84±1.48
SLIM	83.28±3.36	$72.41 \pm 6.92^*$	77.47±4.34	77.23±2.12	53.38±4.02	78.22±2.02
CAL	89.24±8.72	-	76.28±3.65	74.40±4.55	52.13±2.96	82.08±2.40
HRN	90.4±8.9	65.7±6.4	-	77.5±4.3	52.8±2.7	81.80±1.2
GLA	91.05±0.86	-	77.45±0.38	-	-	81.54±0.14
WGDL	94.68±2.63	70.89±5.15	77.29±2.91	$79.70 \pm 3.59^*$	53.45±4.96	80.50±1.17
SS-GDE	**96.78 ± 2.77**	**72.96 ± 6.39**	**80.42 ± 4.29**	**80.70 ± 4.72**	**55.60 ± 3.54**	$82.36 \pm 1.44^*$

4.4 Ablation Study

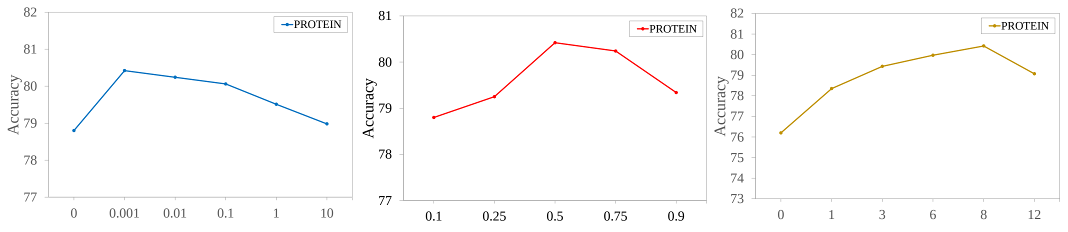
As our SS-GDE framework has achieved superior performance compared to existing state-of-the-art methods, it is also meaningful to conduct additional experiments to make clear how each module promotes the classification task as well as the sensitivity of the hyper-parameters in our framework. For this purpose, we set the variant of our SS-GDE framework, which uses just the fixed base graph dictionary without dictionary adaptation, and meantime only single regularization for cross-graph modeling, as the baseline. Then, we conduct the following ablation study.

1. **The effectiveness of the VGDA module.** To make clear the benefit of our designed VGDA module, we just remove the VGDA module from the SS-GDE (i.e. "BaseLine+MS-WE" in Table 3), and compared the performance.
2. **The effectiveness of the MS-WE module.** To verify the effectiveness of the MS-WE, we remove the MS-WE module from the SS-GDE framework, resulting in the variaint named "BaseLine+VGDA" in Table 3). The single regularization λ is set to 0.1.
3. **The influence of the parameter β.** β plays an essential role in balancing cross-entropy loss and Kullback-Leibler divergence for framework optimization. To quantify its influence, we vary the ratio in $\{0, 0.001, 0.01, 0.1, 1, 10\}$.
4. **The influence of pre-defined probability vector $\widehat{\mathbf{p}}_{ij}$ in Eqn. (6).** $\widehat{\mathbf{p}}_{ij}$ represents the expected probability for the Bernoulli sampling. For easy implementation, we set the elements in $\widehat{\mathbf{p}}_{ij}$ to be the same value. Hence, the value actually means the expected sampling ratio of nodes from the base dictionary keys. Here, we set the value in the range of $\{0.1, 0.25, 0.5, 0.75, 0.9\}$
5. **The influence of the sensitivity number C in the MS-WE module.** We vary the value of C in the range in [0, 12]. For each C, the regularization parameters $\{\lambda_1, \cdots, \lambda_C\}$ in Eqn. (4) are selected uniformly from the range $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 3, 5, 10, 20, 100\}$. Specifically, when C is 0, only the pair-wise squared Euclidean distances across graph are calculated, without the optimal transport.

Table 3: The results of the ablation study

	BaseLine	BaseLine+VGDA	Baseline+MS-WE	Baseline+VGDA+MS-WE (SS-GDE)
PROTEINS	77.29±2.91	78.35±4.30	79.25±3.69	**80.42 ± 4.29**
IMDB-MULTI	53.45±4.96	55.20±3.97	54.47±4.37	**55.60 ± 3.54**

The results are shown in Fig. 2 and Table 3. We have the following observations:



(a) β (b) $\widehat{\mathbf{p}}_{ij}$ (c) C

Figure 2: (a) The influence of parameter β, (b) the influence of pre-defined probablity vector $\widehat{\mathbf{p}}_{ij}$, (c)the influence of C (i.e. number of λ)

1. Our designed VGDA module plays an important role in promoting the graph learning for classification. In Table 3, on the PROTEINS and IMDB-MULTI datasets, additionally using the VGDA effectively improves the accuracy of more than 1% ("BaseLine" vs "BaseLine+VGDA", "BaseLine+MS-WE" vs "SS-GDE"). This verifies the importance of conducting individual substructure selection from the base graph dictionary.

2. According to Table 3, the designed MS-WE module further promotes the graph classification performance. The average performance gain of additionally using the MS-WE ("BaseLine" vs "BaseLine+MS-WE", "BaseLine+VGDA" vs "SS-GDE") on the two datasets is also more than 1%. This verifies the necessity of using the MS-WE to improve the sensitivity for cross-graph correlation.

3. Setting the trade-off parameter β in an appropriate range also improves the performance of our SS-GDE framework. As Fig. 2 (a) shows, the best performance is obtained when β is 0.001. In this situation, the trade-off parameter well balances the first accuracy-related term and the KL-divergence in Eqn. 8. When β is set to 0, it means no distribution constraint for the VGDA. In contrast, when β is set to 10, the KL-divergence may dominate the network optimization, while weakening the accuracy-related term.

4. According to Fig. 2 (b), the pre-defined probability vector $\widehat{\mathbf{p}}_{ij}$ also influences the performance, and should be set appropriately. The best performance is obtained when we set the value to 0.5. The large value of $\widehat{\mathbf{p}}_{ij}$, e.g. 1, means approximately using all the nodes in the base dictionary, degrades the influence of the variational inference. In contrast, the small value of 0.1 means only about 10% nodes are selected to form the AGD, which is insufficient in structure modeling.

5. As it is shown in Fig. 2 (c), the appropriate sensitivity number can improve the sensitivity of the cross-graph correlation. The highest performance is obtained when C is set to 8. Specifically, when C is 0, low performance is achieved no optimal transport is conducted to optimize the structure in the Wasserstein space. However, not higher value of C surely achieves better performance, as too many redundant cross-graph correlations are involved.

Conclusion

This paper proposed an SS-GDE framework to transform input graph into the space of graph dictionary for the graph classification task. Considering the limited expression capacity of a fixed graph dictionary for a giant amount of graphs, we propose variational graph dictionary adaptation (VGDA) to conduct individual structure selections from graph dictionary keys and generate a personalized dictionary adapted for each input graph. Besides, Bernoulli sampling is introduced to effectively choose the corresponding substructures. To increase the sensitivity and stability of cross-graph measurement,multi-sensitivity Wasserstein encoding is proposed to produce the embeddings by designing multi-scale attention on optimal transport. To optimize the proposed framework, we introduce mutual information as objective, which deduces to variational inference of adapted graph dictionary. We evaluated the SS-GDE on multiple datasets and dissected the framework with ablation analysis. The experimental results demonstrate the effectiveness and superiority of our model.

REFERENCES

Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *International Conference on Learning Representations (ICLR)*, 2021.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Liqun Chen, Zhe Gan, Yu Cheng, Linjie Li, Lawrence Carin, and Jingjing Liu. Graph optimal transport for cross-domain alignment. In *International Conference on Machine Learning*, pp. 1542–1553. PMLR, 2020.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. *Advances in neural information processing systems*, 28, 2015.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

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

Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

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

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. PMLR, 2019.

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

Maosen Li, Siheng Chen, Ya Zhang, and Ivor Tsang. Graph cross networks with vertex infomax pooling. *Advances in Neural Information Processing Systems*, 33:14093–14105, 2020.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

Zhiling Luo, Ling Liu, Jianwei Yin, Ying Li, and Zhaohui Wu. Deep learning of graphs with ngram convolutional neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 29(10): 2125–2139, 2017.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pp. 15524–15543. PMLR, 2022.

Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022*, pp. 193–196, 2022.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023. PMLR, 2016.

Amirhossein Nouranizadeh, Mohammadjavad Matinkia, Mohammad Rahmati, and Reza Safabakhsh. Maximum entropy weighted independent set pooling for graph neural networks. *arXiv preprint arXiv:2107.01410*, 2021.

Antoine Rolet, Marco Cuturi, and Gabriel Peyré. Fast dictionary learning with a smoothed wasserstein loss. In *Artificial Intelligence and Statistics*, pp. 630–638. PMLR, 2016.

Morgan A Schmitz, Matthieu Heitz, Nicolas Bonneel, Fred Ngole, David Coeurjolly, Marco Cuturi, Gabriel Peyré, and Jean-Luc Starck. Wasserstein dictionary learning: Optimal transport-based unsupervised nonlinear dictionary learning. *SIAM Journal on Imaging Sciences*, 11(1):643–678, 2018.

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

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

Yongduo Sui, Xiang Wang, Jiancan Wu, Min Lin, Xiangnan He, and Tat-Seng Chua. Causal attention for interpretable and generalizable graph classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1696–1705, 2022.

Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pp. 6275–6284. PMLR, 2019.

Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Cédric Vincent-Cuaz, Titouan Vayer, Rémi Flamary, Marco Corneli, and Nicolas Courty. Online graph dictionary learning. In *International Conference on Machine Learning*, pp. 10564–10574. PMLR, 2021.

Junran Wu, Shangzhe Li, Jianhao Li, Yicheng Pan, and Ke Xu. A simple yet effective method for graph classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3580–3586, 2022a.

Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks. *arXiv preprint arXiv:2201.12872*, 2022b.

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

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

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Han Yue, Chunhui Zhang, Chuxu Zhang, and Hongfu Liu. Label-invariant augmentation for semi-supervised graph classification. *arXiv preprint arXiv:2205.09802*, 2022.

Tong Zhang, Yun Wang, Zhen Cui, Chuanwei Zhou, Baoliang Cui, Haikuan Huang, and Jian Yang. Deep wasserstein graph discriminant learning for graph classification. In *AAAI*, 2021.

Yaokang Zhu, Kai Zhang, Jun Wang, Haibin Ling, Jie Zhang, and Hongyuan Zha. Structural landmarking and interaction modelling: A ”slim” network for graph classification. In *Proceedings of the Thirty-Sixth Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 9251–9259, 2022.