Deep Wasserstein Graph Discriminant Learning for Graph Classification

Tong Zhang[1*], **Yun Wang**[1*], **Zhen Cui**[1†], **Chuanwei Zhou**[1],
Baoliang Cui[2], **Haikuan Huang**[2], **Jian Yang**[1]
[1]Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education,
Jiangsu Key Lab of Image and Video Understanding for Social Security,
School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China.
[2] Alibaba Group, Hangzhou, China
{tong.zhang, yun.wang, zhen.cui, cwzhou}@njust.edu.cn, moqing.cbl@taobo.com, haikuan.hhk@alibaba-inc.com,
csjyang@njust.edu.cn

Abstract

Graph topological structures are crucial to distinguish different-class graphs. In this work, we propose a deep Wasserstein graph discriminant learning (WGDL) framework to learn discriminative embeddings of graphs in Wasserstein-metric (W-metric) matching space. In order to bypass the calculation of W-metric class centers in discriminant analysis, as well as better support batch process learning, we introduce a reference set of graphs (aka graph dictionary) to express those representative graph samples (aka dictionary keys). On the bridge of graph dictionary, every input graph can be projected into the latent dictionary space through our proposed Wasserstein graph transformation (WGT). In WGT, we formulate inter-graph distance in W-metric space by virtue of the optimal transport (OT) principle, which effectively expresses the correlations of cross-graph structures. To make WGDL better representation ability, we dynamically update graph dictionary during training by maximizing the Wasserstein Discriminant loss, i.e. the ratio of inter-class versus intra-class Wasserstein distance. To evaluate our WGDL method, comprehensive experiments are conducted on six graph classification datasets. Experimental results demonstrate the effectiveness of our WGDL, and state-of-the-art performance.

Introduction

Graph modeling has become an active topic in the machine learning field due to its huge application potential in recent years. Generally, a graph consists of a set of nodes and edges, where the nodes can represent any types of entities and the edges describe their connection relationship. Due to the flexible structure and powerful representation ability, graphs have been widely used to characterize ubiquitous irregular data in the real world such as citation graphs and social networks. Accordingly, there exist various graph-related pattern recognition tasks, among which graph classification is one of the most fundamental tasks.

Numerous algorithms have been proposed for graph classification in the past decade years. Graph kernel-based algorithms and graph neural networks (GNNs) are two mainstreams of methods. Graph kernels (Shervashidze et al. 2011; Borgwardt and Kriegel 2005; Shervashidze et al. 2009) often have explicit expressions and are easy to train because they usually have convex optimization solutions. However, as these methods often take hand-crafted (shallow) features as input, their representation ability in capturing complex interactions between nodes might be limited. In contrast, GNNs learn representation from graphs by stacking multiple neural network layers, in each of which multi-hop node aggregation is performed. In this way, GNNs are powerful in capturing graph structural information, and thus have achieved promising results in graph classification.

Although considerable success has been achieved by previous works above, however, few of them directly considered topology correspondence between graphs. Topological characteristics play a crucial role in identifying a graph, and well capturing topology correspondence may be of great significance to learn discriminative graph representation. However, topology correspondence may not be easily modeled as it requires understanding semantic structure among nodes. Based on this consideration, Wasserstein distance (W-distance) (Cuturi 2013) may be leveraged to measure the topology correlation between two graphs. With an optimal transport (OT) matrix considering all probabilistic couplings between two probability distributions, W-distance provides a powerful tool to compute distances between two complex distributions. From this perspective, W-metric would be a good selection to measure the correspondence of topology distributions spanned by those graph nodes.

To do discriminative analysis of graphs in W-metric space, however, we have to confront three aspects of issues: i) The integration of W-metric into GNNs is necessary, but difficult due to the non-Euclidean property. ii) There exists a high complexity in computing inter-class and intra-class scatter, as the explicit calculation on mean and covariance of graphs are rather difficult in W-metric space. iii) The minibatch-based processing of deep learning architectures makes it difficult to access the global covariance of all samples within one or across different classes.

To address the aforementioned problems, we propose a deep Wasserstein Graph Discriminant Learning (WGDL) to deal with the graph classification task. In WGDL, graph dis-

[*]Tong Zhang and Yun Wang have equal contribution.
[†]The corresponding author.

criminant learning is fully performed in W-metric space. Specifically, we construct a graph dictionary as the reference set to avoid those explicit statistic computation on mean and covariance of graph samples. Meantime, the use of graph dictionary can be in favour of minibatch-based processing during the training of neural network. Taking graph dictionary as the bridge, every given input graph can be transformed into dictionary space and produce succinct features through our proposed Wasserstein graph transformer (WGT). In this transformation, graph correlation is measured in W-metric space through the regularized W-distance with optimal transport (OT) matrices. To learn more discriminative features, graph dictionary is designed to be dynamically updated during training. In the constrain of a maximum Wasserstein discriminant loss (WD-Loss), i.e. ratio of inter-class versus intra-class W-distance, the encoded dictionary keys are optimized to be more compact in each class meanwhile have better inter-class dispersion. Finally, we construct a fully end-to-end training network including graph encoding, W-distance computation of graphs and dictionary learning. We test the proposed method on six popular graph classification datasets, and experimental results demonstrate the effectiveness of our methods. The contributions of this work can be summarized as follows:

- We propose a novel deep Wasserstein graph discriminant learning framework for graph discriminant analysis, where graph convolution learning and graph distance learning are performed in W-metric space.
- We propose a dynamic graph dictionary defined in W-metric space, which should be the first time to our knowledge. Further, we introduce a maximum ratio principle of inter-class versus intra-class in W-metric space to learn better representation ability for graph dictionary.
- We define a Wasserstein graph transformer by using the optimal transport mechanism, which can convert those irregular graphs into the normalized dictionary space under W-metric.
- We verify the effectiveness of our method, and report the state-of-the-art results on several graph classification datasets.

Related Work

In this section, we first review the previous methods of graph classification, then introduce those works related to W-distance learning.

Graph Classification

Graph classification has been widely investigated in previous literatures, and existing methods can be roughly divided into two main types, i.e. graph kernel based methods and graph neural networks (GNNs). Graph kernel based methods enumerate substructures of a whole graph through graph decomposition, and further build graph kernels based on the similarities among these components. Graph kernels proposed in the early stage include graphlets (Shervashidze et al. 2009), shortest paths (Borgwardt and Kriegel 2005), Weisfeiler-Lehman kernel (Shervashidze et al. 2011), and other graph kernels (Orsini, Frasconi, and De Raedt 2015; Kriege and Mutzel 2012). Although good performances have been achieved, however, they may suffer from the drawback of "diagonal dominance issue", which may happen when substructures are assigned with large sizes. To solve this problem, in recent years, Zhang et. al (Zhang et al. 2018) proposed the return probability-based graph kernel which can effectively exploit various node attributes while being scalable to large datasets.

GNNs are powerful neural networks designed to work directly on the graph-structured data. Specifically, inspired by the success of the standard Convolutional neural network (CNN) (Krizhevsky, Sutskever, and Hinton 2012), various graph convolution operations have been studied, yielding multiple graph CNN variants (Niepert, Ahmed, and Kutzkov 2016; Defferrard, Bresson, and Vandergheynst 2016), including graph convolutional network (GCN) (Kipf and Welling 2016), PATCHY-SAN (referred to as PSCN) (Niepert, Ahmed, and Kutzkov 2016), Diffusion-convolutional neural networks (DCNN) (Atwood and Towsley 2016), and NgramCNN (Luo et al. 2017). Considering their powerful representation ability, these methods are adapted to the graph classification task where promising performances are achieved. For instance, DCNN (Atwood and Towsley 2016) models the graph by scanning a diffusion process across each vertex, NgramCNN (Luo et al. 2017) introduces the n-gram block to serialize each graph, based on which graph representation learning and classification are fulfilled.

Wasserstein Distance Learning

W-distance is a powerful tool to compute the distances between two complex distributions by leveraging the OT principle. To take the advantage of the OT matrix, various algorithms (Titouan et al. 2019; Simou, Thanou, and Frossard 2020; Flamary et al. 2018; Bécigneul et al. 2020) are proposed to either learn representation from graphs or measure the distances between two data sets. For instance, Simou et al. (Simou, Thanou, and Frossard 2020) proposed a graph representation framework with W-distances to simultaneously learn a low-dimensional space and coordinates for nodes. Some other works (Schmitz et al. 2018; Rolet, Cuturi, and Peyré 2016) attempted to conduct dictionary learning in W-space, e.g. to use the W-distance as the fitting error between each original point and its reconstruction.

In contrast to these existing methods above, our WGDL has several different aspects:

- We introduce W-metric to measure the distance of graphs, and further derive graph convolution network in W-metric space. It is different from the traditional graph convolution methods defined in Euclidean space.
- We introduce dictionary learning on graphs into a deep architecture, different from those graph kernels and dictionary learning methods (Schmitz et al. 2018; Rolet, Cuturi, and Peyré 2016) that only operate on the general feature vector space (not involve topological structures).
- We introduce W-metric discriminant analysis on graphs by maximizing the WD-Loss in W-metric space.

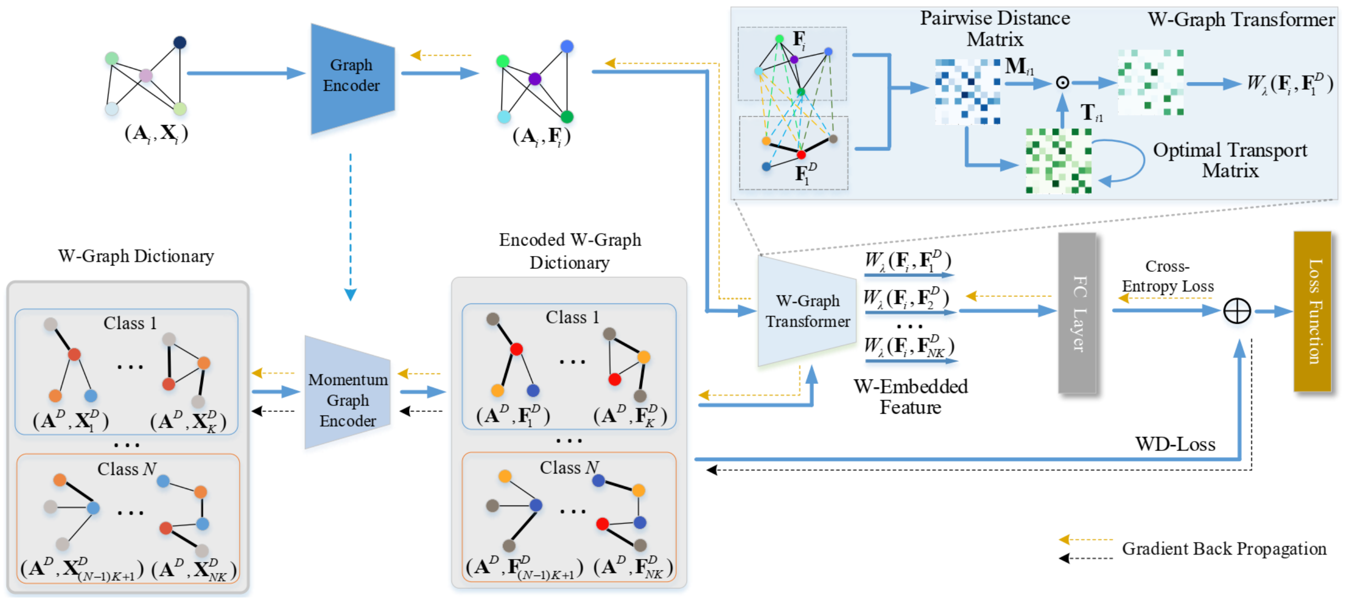

$(\mathbf{A}_i, \mathbf{X}_i)$
Graph Encoder
$(\mathbf{A}_i, \mathbf{F}_i)$
Pairwise Distance Matrix
W-Graph Transformer
$\mathbf{F}_i$
$\mathbf{M}_{i1}$
$\mathbf{T}_{i1}$
$W_\lambda(\mathbf{F}_i, \mathbf{F}_1^D)$
Optimal Transport Matrix
$\mathbf{F}_1^D$
W-Graph Dictionary
Encoded W-Graph Dictionary
Class 1
Class N
$(\mathbf{A}^D, \mathbf{X}_1^D)$
$(\mathbf{A}^D, \mathbf{X}_K^D)$
$(\mathbf{A}^D, \mathbf{X}_{(N-1)K+1}^D)$
$(\mathbf{A}^D, \mathbf{X}_{NK}^D)$
Momentum Graph Encoder
$(\mathbf{A}^D, \mathbf{F}_1^D)$
$(\mathbf{A}^D, \mathbf{F}_K^D)$
$(\mathbf{A}^D, \mathbf{F}_{(N-1)K+1}^D)$
$(\mathbf{A}^D, \mathbf{F}_{NK}^D)$
W-Graph Transformer
$W_\lambda(\mathbf{F}_i, \mathbf{F}_1^D)$
$W_\lambda(\mathbf{F}_i, \mathbf{F}_2^D)$
$W_\lambda(\mathbf{F}_i, \mathbf{F}_{NK}^D)$
W-Embedded Feature
FC Layer
Cross-Entropy Loss
Loss Function
WD-Loss
Gradient Back Propagation

Figure 1: The architecture of our WGDL. The WGDL consists of three main processes: graph encoding, graph dictionary learning, Wasserstein graph transforming. Given an input graph, we use graph convolution network (aka graph encoder) to extract more robust feature representation. More details could be found in the section of the proposed method. And, we introduce a dictionary of graphs as the reference set. The input graph can be transformed into the space spanned by graph dictionary. In graph transforming, we use W-metric to define the correlation between the input query and the keys of dictionary. We name the transformation process as Wasserstein graph transforming. The W-metric discriminant loss is further imposed on graph dictionary to learn better representation ability. Finally, graph dictionary is dynamically optimized together with graph encoder and Wasserstein graph transformer. The entire learning of all modules is an end-to-end process. More details could be found in the mainbody.

The Proposed Method

In this section, we will first give an overview of the proposed WGDL model, then describe the learning processes in the proposed framework in detail.

Overview

The whole architecture of our deep WGDL framework is shown in Fig. 1. In general, it consists of three main leaning processes, i.e. graph encoding, Wasserstein graph transforming, and graph discriminant learning. Given a certain graph as the input, the deep WGDL model aims to classify it into a certain category by learning robust and discriminative features. To this end, a dynamic graph dictionary is constructed to encode those input graphs into vectors in a much lower dimension through the Wasserstein graph transforming (aka query process). For input graphs and dictionary keys, two corresponding graph encoders with the same architecture of stacked GCN layers are first employed for graph encoding. Then, the feature of each input graph is further encoded through Wasserstein graph transformer in a dictionary lookup manner by calculating its W-distance with respect to all the encoded dictionary keys. After this step, the input graphs can be transformed to succinct query-resulted vectors by characterizing semantic correspondence between graph-structured data. Specifically, each key of dictionary denotes an individual graph attached with a certain class label, and it will be dynamically updated during the training process. Based on the framework above, we expect to learn discriminative features as well-separated dictionary keys. To better learn representation ability of network, we use the supervised label information with the cross-entropy loss to guide the network optimization. At the same time, the discriminant analysis on dictionary is done by introducing the principle of inter-class versus intra-class in W-metric space, which would make dictionary keys more compact with each class and dispersed between classes. Finally, the whole architecture could be optimized in an end-to-end training mode.

Graph Encoding

A graph encoder and a momentum graph encoder with the same architecture are employed in the WGDL framework for graph feature learning, and they are used to model input graphs and dictionary keys, respectively. To obtain robust representation, the graph encoder stacks several layers of graph convolutional networks (GCNs), which are powerful in learning graph topology. For the i-th input graph sample $\mathcal{G}_i = (\mathcal{V}_i, \mathbf{A}_i, \mathbf{X}_i)$ and j-th dictionary key $\mathcal{G}_j^D = (\mathcal{V}_j^D, \mathbf{A}_j^D, \mathbf{X}_j^D)$, the corresponding encoding functions $f(\cdot)$ and $f_D(\cdot)$ perform aggregation to their nodes based on the adjacency relationship, yielding the corresponding encoded features $\mathbf{F}_i = f(\mathbf{X}_i, \mathbf{A}_i, \Phi)$ and $\mathbf{F}_j^D = f_D(\mathbf{X}_j^D, \mathbf{A}_j^D, \Phi_D)$.

Here, $\mathbf{X}_i \in \mathbb{R}^{N_1 \times d}, \mathbf{A}_i \in \mathbb{R}^{N_1 \times N_1}, \mathbf{X}_j^D \in \mathbb{R}^{N_2 \times d}, \mathbf{A}_j^D \in \mathbb{R}^{N_2 \times N_2}$, in which N_1, N_2 denote the numbers of nodes and d is the feature dimension, Φ, Φ_D represent the parameter sets of the two encoders. Formally, taking $f(\cdot)$ of two-layer GCNs as an example, it can be written as:

$$\begin{aligned}\mathbf{F}_i &= f(\mathbf{X}_i, \mathbf{A}_i, \Phi) \\ &= \sigma(\widehat{\mathbf{A}}_i \sigma(\widehat{\mathbf{A}}_i \mathbf{X}_i \mathbf{W}^{(0)}) \mathbf{W}^{(1)}). \end{aligned} \tag{1}$$

In this equation, $\mathbf{W}^{(0)}, \mathbf{W}^{(1)} \in \Phi$ are two weighting matrices for feature projection, and $\sigma(\cdot)$ is a non-linear activation function. Specifically, a little different from the standard GCN, here $\widehat{\mathbf{A}}_i = \mathbf{A}_i + \mathbf{I}_{N_1}$ while the Laplacian normalization is not performed. According to (Sandryhaila and Moura 2013), from the view of spectral filtering, similar learning effects can be achieved no matter whether Laplacian normalization is performed because the adjacent matrix and its Laplacian norm have the same eigenvectors.

As same as the architecture of $f(\cdot)$, $f_D(\cdot)$ performs similar transformation on those dictionary keys, but uses different parameters denoted as Φ_D. In the optimization stage, different updating strategies are taken for the parameter sets Φ and Φ_D. Specifically, Φ is updated through the gradient back-propagation, while Φ_D is optimized according to the momentum update mechanism (He et al. 2020) to make it evolve more smoothly to obtain better encoded results. Formally. the update of Φ_D can be written as

$$\Phi_D = m\Phi_D + (1-m)\Phi, \tag{2}$$

where m means a momentum coefficient.

Wasserstein Graph Transforming

The dynamic Wasserstein graph dictionary targets at projecting its query graphs to discriminative vectors in a low-dimension Euclidean space. Let $\mathcal{S} = \{\mathbf{F}_1^D, \mathbf{F}_2^D, \cdots, \mathbf{F}_{NK}^D\} (\mathbf{F}_j^D \in \mathbb{R}^{N_2 \times d_1}, j \in [1, NK])$ represents the encoded features of K dictionary keys in each class with N classes, and $\mathbf{F}_i$ denotes the i-th query feature (encoded input feature) attached with a class label $\mathbf{y}_i$. Formally, the W-distance calculation can be written as:

$$z_{ij} = W_\lambda(\mathbf{F}_i, \mathbf{F}_j^D) = \langle \mathbf{T}_{ij}^\lambda, \mathbf{M}_{ij} \rangle, \tag{3}$$

$$\text{s.t.}\ , \quad \mathbf{T}_{ij} \mathbf{1}_{N_2} = \mathbf{1}_{N_1}/N_1, \tag{4}$$

$$\mathbf{T}_{ij}^T \mathbf{1}_{N_1} = \mathbf{1}_{N_2}/N_2, \tag{5}$$

where $\mathbf{T}_{ij} \in \mathbb{R}_+^{N_1 \times N_2}$. In Eqn. (3), z_{ij} is the j-th element of the i-th query-resulted feature denoted as $\mathbf{z}_i = [z_{i1}, \cdots, z_{iNK}]$, and $\langle \mathbf{A}, \mathbf{B} \rangle = tr(\mathbf{A}^T\mathbf{B})$. $\mathbf{M}_{ij}$ is the pairwise distance matrix in Euclidean space, and the element $M_{ij}(r,l)$ in the r-th row and l-th column calculates the squared Euclidean distance between the r-th node of $\mathbf{F}_i$ and l-th node of $\mathbf{F}_j^D$. $\mathbf{T}_{ij}^\lambda$ is the solution of an entropy-smoothed optimal transport problem:

$$\mathbf{T}_{ij}^\lambda = \underset{\mathbf{T}_{ij}}{\arg\min} \quad \lambda \langle \mathbf{T}_{ij}, \mathbf{M}_{\mathbf{F}_i, \mathbf{F}_j^D} \rangle - \Omega(\mathbf{T}_{ij}). \tag{6}$$

Here, $\Omega(\mathbf{T}_{ij}) = -\sum_{rl} T_{ij}(r,l) \log(T_{ij}(r,l))$ where $T_{ij}(r,l)$ is the element in the r-th row and l-th column of $\mathbf{T}_{ij}$, and $\mathbf{T}_{ij} \in \mathbb{R}_+^{N_1 \times N_2}$. $\Omega(\mathbf{T}_{ij})$ can be seen as a discrete joint probability distribution calculating the entropy of $\mathbf{T}_{ij}$. Specifically, the optimization problem in Eqn. (6) can be efficiently solved through Sinkhorn's fixed point iterations (Cuturi 2013), and the solution can be written as:

$$\begin{aligned}\mathbf{T}_{ij} &= diag(\mathbf{u}_{ij}) \mathbf{K}_{ij} diag(\mathbf{v}_{ij}) \\ &= \mathbf{u}_{ij} \mathbf{1}_{N_2}^T \odot \mathbf{K}_{ij} \odot \mathbf{1}_{N_1} \mathbf{v}_{ij}^T, \end{aligned} \tag{7}$$

where $\odot$ represents elementwise production, and $\mathbf{K}_{ij}$ is calculated based on the distance matrix $\mathbf{M}_{ij}$ with $\mathbf{K}_{ij} = e^{-\lambda \mathbf{M}_{ij}}$. In Sinkhorn iterations, $\mathbf{u}_{ij}$ and $\mathbf{v}_{ij}$ are kept updating. Taking the k-th iteration as an example, the update takes the following form:

$$\mathbf{v}_{ij}^k = \frac{\mathbf{1}_{N_2}/N_2}{\mathbf{K}_{ij}^T \mathbf{u}_{ij}^{k-1}}, \tag{8}$$

$$\mathbf{u}_{ij}^k = \frac{\mathbf{1}_{N_1}/N_1}{\mathbf{K}_{ij} \mathbf{v}_{ij}^k}. \tag{9}$$

For the initialization of the update process above, $\mathbf{u}_{ij}^0$ is assigned as an all-1 vector $\mathbf{1}_{N_1}$.

Graph Discriminant Learning

To effectively promote the graph classification performance, our WGDL model targets at possessing two expected properties, i.e. the discriminability of the learned features and the separability of the encoded dictionary keys. For the discriminability of features, a supervised learning objective is adopted to guide the network optimization. And for the separability of the encoded dictionary, the distribution of encoded dictionary keys is made to be compact within each class while dispersed between classes.

To fulfill the purposes above, the WD-Loss denoted as E_I is specifically proposed together with the cross-entropy E_c for the optimization of the whole architecture, which can be formally expressed as:

$$E = E_c - \beta E_I, \tag{10}$$

where β is a trade-off parameter between E_c and E_I. Specifically, the cross-entropy E_c measures the divergence between the predicted probabilities and the actual labels, and can be calculated after passing the query result through full-connection layers followed with a softmax layer.

For the WD-Loss E_I, given the encoded dictionary key set $\mathcal{S} = \{\mathbf{F}_1^D, \mathbf{F}_2^D, \cdots, \mathbf{F}_{NK}^D\}$ ($\mathbf{F}_j^D \in \mathbb{R}^{N_2 \times d_1}$) attached with corresponding labels, it takes the following form:

$$E_I = \frac{\sum_{c,\ c'>c} \sum_{i \in \mathcal{I}_c, j \in \mathcal{I}_{c'}} T_{c,c'}^\lambda(i,j) W_\lambda(\mathbf{F}_i^D, \mathbf{F}_j^D)}{\sum_c \sum_{i,j \in \mathcal{I}_c} T_{c,c}^\lambda(i,j) W_\lambda(\mathbf{F}_i^D, \mathbf{F}_j^D)}, \tag{11}$$

$$\text{s.t. } \mathbf{T}_{c,c'}^\lambda = \arg\min \lambda \langle \mathbf{T}_{c,c'}, \mathbf{M}_{c,c'} \rangle - \Omega(\mathbf{T}_{c,c'}). \tag{12}$$

In the above equations, $\mathbf{T}_{c,c'}^\lambda$ is the transport matrix imposing weights on distances between graph samples belonging to classes c and c', $\mathbf{M}_{c,c'}$ is a W-distance matrix whose each element measures the W-distance between two graph samples in the dictionary from classes c and c', respectively. $\mathcal{I}_c$ is a sample index set that the samples indexed by its elements all belong to class c.

Datasets	Graphs Num	Average Nodes	Average Edges	Node Labels
MUTAG	188	17.93	19.79	7
PTC	344	14.29	14.69	19
NCI1	4110	29.87	32.30	37
PROTEIN	1137	39.06	72.82	3
IMDB-BINARY	1000	19.77	96.53	-
IMDB-MULTI	1500	13.00	65.94	-

Table 1: Summary of Graph Datasets.

Experiments

In the following parts, we first introduce the used datasets, then describe the experiment setup including the implementation details, parameter setting, and the employed protocol. Next, we compare the proposed WGDL model with multiple state-of-the-art methods. Finally, we analyze our model by conducting additional ablation experiments.

Dataset

The employed datasets in the experiments can be divided into two categories, i.e. the bioinformatics datasets and social networks datasets. The bioinformatics datasets include MUTAG (Debnath et al. 1991), PTC (Toivonen et al. 2003), PROTEINS (Borgwardt et al. 2005), and NCI1 (Wale, Watson, and Karypis 2008), while the social networks datasets contain IMDB-BINARY and IMDB-MULTI. Below, we briefly introduce them, and please also refer to Table 1.

Bioinformatics Datasets. MUTAG is a nitro compounds dataset containing 188 samples divided into 2 classes, and for each node there are 7 discrete labels; PTC is about compounds labeled according to carcinogenicity on rodents. The graphs in this dataset are divided into two categories with each node annotated with 19 labels. NCI1, collected by National Cancer Institute (NCI), is a balanced dataset of chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell, and they contain 4110 and 4127 chemical compounds, respectively. PROTEINS contains 1113 protein structures of secondary structure elements (SSEs) with 3 discrete node labels.

Social Networks Datasets. IMDB-BINARY and IMDB-MULTI are both movie collaboration datasets derived from IMDB with two and three classes respectively. Each graph represents a movie and every node represents an actor/actress. If they appear in the same movie, there will be an edge between them. IMDB-BINARY is derived from the Action and Romance genres.

Experiment Setup

Implementation Details. For each input graph, each node is initialized based on the node label by using a one-hot vector. Both the graph encoder and the momentum graph encoder stack 3 layers of GCNs, where the output dimensions are 512, 256, and 32 respectively. For the setting of the Wasserstein graph dictionary, generally, larger dataset with more classes may require more keys for higher representative ability. To avoid excessive parameters tuning, the key number for each class is set the same (10 for each class here) for all datasets. Accordingly, the graph dictionary is initialized by randomly selecting a fixed number of samples from each class in the training set. We set 2 fully connected layers to further learn the query-resulted vectors, and their output dimensions equal 64 and the class number, respectively. For the momentum graph encoder, similar with (He et al. 2020), we update it according to the graph encoder by following Eqn. 2 with the momentum coefficient m of 0.999, rather than applying the gradient descent. In contrast, during training, the feature and adjacent matrix of each dictionary key are both optimized. Specifically, the parameter λ for calculating the OT matrix in Eqn. 3 and Eqn. 11 , together with the trade-off parameter β in Eqn. 10, are set to 1. We train our model with Adam optimizer for 300 epochs, where the weight decay is 10^{-4} and the learning rate is 0.001.

Protocol. By strictly following the employed protocol in previous literatures, we evaluate the performance by 10-fold cross-validation on all employed datasets. Each dataset is divided into 10 sessions, of which nine sessions are used for training and the remaining one session for testing. The experiments are repeated 10 times in which process the session used for testing is enumerated. After this process, we use the average accuracy of the ten runs as the final performance.

Experimental Results

We compare the WGDL framework with several state-of-the-arts, including graph kernel based methods (GK (Shervashidze et al. 2009), DGK (Yanardag and Vishwanathan 2015), WL (Shervashidze et al. 2011), DWL (Morris, Kersting, and Mutzel 2017), P-WL-UC (Rieck, Bock, and Borgwardt 2019)), graph CNN methods (PSCN (Niepert, Ahmed, and Kutzkov 2016), NgramCNN (Luo et al. 2017), PPGN-1 (Maron et al. 2019), EigenGCN-3 (Ma et al. 2019), GNTK (Du et al. 2019)), feature based algorithms (DyF (Gómez, Chiêm, and Delvenne 2017), FB (Barnett et al. 2016)), and the neural network method (SAEN (Orsini, Baracchi, and Frasconi 2018)). The results are shown in Table 2. From the results, we have the following observations.

(1) In general, the graph kernel-based methods achieve lower performance than the other algorithms, which may attribute to the rough (hand-crafted) input features and shallow learning architecture of these kinds of methods. Among the graph kernel-based methods, WL achieves the best performance on most datasets, and even approaches or outperforms GNNs on the NCI1 dataset.

(2) GNNs and feature-based algorithms outperform those graph kernel-based methods, which may benefit from deep architectures. Specifically, graph CNNs are superior to other compared methods due to the node aggregation operation, which helps characterize the local topology.

(3) Our WGDL model achieves the best performance in most cases comparing to all the other methods. Compared to graph kernel methods, our proposed method improves the performance with about 10% higher accuracy on the MUTAG, PTC, and IMDB-BINARY datasets. Even for those

Datasets	MUTAG	PTC	NCI1	PROTEINS	IMDB-BINARY	IMDB-MULTI
GK	81.66±2.11	57.26±1.41	62.28±0.29	71.67±0.55	65.87±0.98	43.89±0.38
DGK	82.66±1.45	57.32±1.13	62.48±0.25	71.68±0.50	66.96±0.56	44.55±0.52
EigenGCN-3	79.50	-	77.00	76.60	-	-
WL	80.72±3.00	56.97±2.01	83.10±0.20	73.70±0.50	72.86±0.76	50.55±0.55
DWL	82.94±2.68	59.17±1.56	-	-	-	-
P-WL-UC	85.17±0.29	-	**85.62 ±0.27**	75.86±0.78	-	-
FB	84.66±2.01	55.58±2.30	62.90±0.96	69.97±1.34	72.02±4.71	47.34±3.56
PSCN	92.63±4.21	60.00±4.82	78.59±1.89	75.89±2.76	71.00±2.29	45.23±2.84
SAEN	84.99±1.82	57.04±1.30	77.80±0.42	75.31±0.70	71.26±0.74	49.11±0.64
DyF	88.00±2.37	57.15±1.47	68.27±0.34	75.04±0.65	72.87±4.05	48.12±3.56
GNTK	90.00±8.50	67.90±6.90	84.20±1.50	75.60±4.20	76.90±3.60	52.80±4.60
PPGN-1	90.55±8.70	66.17±6.54	83.19±1.11	77.20±4.73	72.60±4.90	50.00±3.15
NGRAMCNN	**94.99±5.63**	68.57±1.72	-	75.95±2.98	71.66±2.71	50.66±4.10
WGDL	94.68±2.63	**70.89**±**5.15**	80.30±2.45	**77.29**±**2.91**	**79.70**±**3.59**	**53.45**±**4.96**

Table 2: Comparsion with state-of-the-art-methods.

powerful graph CNN models, our proposed method still outperforms them with about 2% higher accuracy. The performance gain verifies the effectiveness of the WGDL model to learn between-graph correspondence.

Ablation Study

As our WGDL framework has achieved promising performance compared to existing state-of-the-art methods, it is meaningful and interesting to make clear how the modules or parameters setting, e.g. the graph dictionary and the number of keys, influence the performance of graph classification. For this purpose, we conduct several additional experiments to dissect our framework based on PROTEINS and IMDB-BINARY datasets as follows:

(1) The effectiveness of the dynamic Wasserstein dictionary. To evaluate the benefit of our designed dynamic Wasserstein dictionary, based on which graph correspondence is modeled, we simply remove the dictionary module from the WGDL framework, which results in a 3-layer GCN. Then, we compare the performance between these.

(2) The effectiveness of the OT principle in W-distance. To make clear the effectiveness of the OT principle, we just remove the OT projection (WGDL_No_OT in Table 3) in Eqn. 3, so that the W-distance is calculated only based on the point-wise Euclidean distance between two graphs.

(3) The effectiveness of the WD-Loss. To evaluate the WD-Loss, we remove it from the WGDL framework (WGDL_No_WD in Table 3) and compare them.

(4) The influence of the ratio between the dictionary key node number and input graph node number. To quantify its influence, we vary the ratio in $\{0.2, 0.6, 1.0, 1.4, 1.8\}$.

(5) The influence of the dictionary key number K of each class. The key number of each class in set in the range of $\{2, 6, 10, 14, 18\}$ to see how the accuracy changes.

	PROTEINS	IMDB-BINARY
3-layer GCN	75.85±2.97	77.30±3.20
WGDL_No_WD	76.39±2.68	78.70±3.65
WGDL_No_OT	73.51±3.84	77.40±2.99
WGDL	**77.29**±2.91	**79.70**±3.59

Table 3: The results of the ablation study.

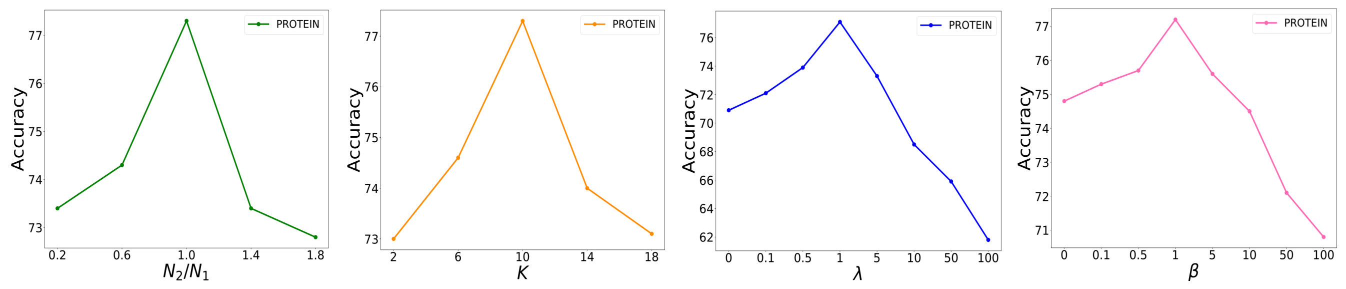
(6) The influence of the parameters λ and β. Both the ranges of the parameters λ and β are set in $\{0, 0.1, 0.5, 1, 5, 10, 50, 100\}$. When evaluating one parameter (e.g. λ) by varying its value, the other's value (e.g. β) is fixed to 1.

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

(1) Our designed dynamic dictionary effectively improves the graph classification performance. On the datasets, the WGDL framework outperforms a 3-layer GCN with about 2 percent higher accuracy. The performance gain comes from the specifically designed dynamic dictionary which additionally measures the topology correspondence between graphs compared to GCN.

(2) The OT principle also plays a crucial role in capturing graph topology correlation. According to Table 3, comparing WGDL_No_OT with WDGL, the performance obviously degrades without OT matrix in calculating the W-distance. The performance gap shows the importance of the OT matrix for well measuring the cross-graph topology correlation based on the Sinkhorn algorithm.

(3) The WD-Loss further promotes the performance of our WGDL framework. About 1% accuracy gain is achieved on both datasets by imposing the WD-Loss. This verifies



(a) (b) (c) (d)

Figure 2: The influence of the ratio between the dictionary key node number (N_2) and input graph node number (N_1) (a), the influence of the dictionary key number of each class (b), the influence of the λ (c) and β (d).

that the intra-class compactness and inter-class dispersion can effectively facilitate the graph discriminant learning.

(4) Both the numbers of the key nodes and graph keys in the dictionary should be set appropriately in a certain range. According to Fig. 2, for the number of key nodes, the best performance is achieved when the input graph and the key have almost equal numbers, while the performance degrades if further increasing the key node number. This may attribute to the reason that too many nodes in the key may introduce redundancy in the graph correspondence measurement. Similar to the node number, excessive keys may increase the correlation between subspaces while they tend to reduce the discriminability of representation. As shown in the experiment, performance degradation is observed when setting excessive keys.

(5) Overall, the proposed model is robust to the variations of the two parameters λ and β: even though two parameters are varied in a pretty large range of values, the performance of the model stays above 60%. Specifically, λ controls local information between the nodes across two graphs. For too large values of λ, the OT matrix would become little sensitive to the local correlation between two graphs and therefore would degrade the graph representation ability. And for the parameter β, it balances the influence of the cross-entropy loss and WD-Loss. In the case of large values of β, the large bias would be introduced into the training that reduces the discriminability of the graph representation (please see Eqn. (10)) and therefore degrades the performance.

Moreover, we additionally visualize the cross-key W-distances of the dictionary before and after training with 40 epochs in Fig. 3. According to Fig. 3, intuitively, our proposed WD-Loss is effective to endow intra-class compactness and inter-class dispersion, as the values of the diagonal patches of the matrices in the right column are near zero.

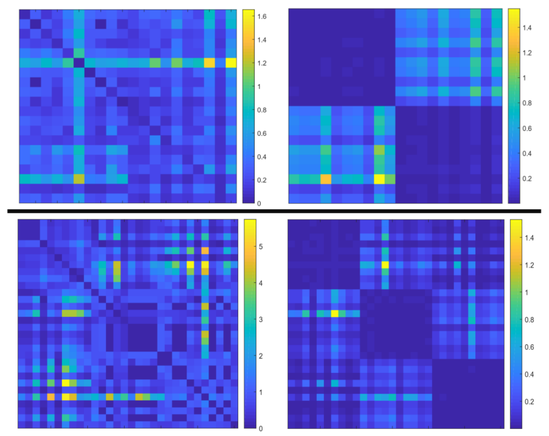


Figure 3: The visualization of cross-key W-distances of the dictionary. The first row is drawn based on a two-class dataset while the second is based on a three-class one. The left column shows the cross-key W-distances of the dictionaries before optimization, and the dictionaries related to the right column are optimized for 40 epochs.

Conclusion

In this paper, a deep WGDL framework was proposed for the graph classification task. Considering the difficulty of calculating W-metric class centers in discriminant analysis, a graph dictionary was constructed as the reference set to avoid calculating statistics on the mean and covariance of graphs. Based on the constructed dictionary, the input graph can be transformed through the proposed WGT, in which process graph correlation is measured through the regularized W-distance in an OT principle. To learn more discriminative features, the graph dictionary was made to be dynamically updated in the training process. Accordingly, the WD-Loss was introduced, based on which the dictionary keys are optimized to have better intra-class compactness and inter-class dispersion. We evaluated the proposed model on multiple Bioinformatics and social network datasets, and dissected the framework with ablation analysis. The experimental results verify the effectiveness of our model.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61906094, 62072244, 61972204), the Natural Science Foundation of Jiangsu Province (Grant Nos. BK20190019 and BK20190452) and the Alibaba Innovative Research (AIR) programme.

References

Atwood, J.; and Towsley, D. 2016. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, 1993–2001.

Barnett, I.; Malik, N.; Kuijjer, M. L.; Mucha, P. J.; and Onnela, J.-P. 2016. Feature-based classification of networks. *arXiv preprint arXiv:1610.05868* .

Bécigneul, G.; Ganea, O.-E.; Chen, B.; Barzilay, R.; and Jaakkola, T. 2020. Optimal Transport Graph Neural Networks. *arXiv preprint arXiv:2006.04804* .

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

Borgwardt, K. M.; Ong, C. S.; Schönauer, S.; Vishwanathan, S.; Smola, A. J.; and Kriegel, H.-P. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21(suppl_1): i47–i56.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, 2292–2300.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. 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.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, 3844–3852.

Du, S.; Hou, K.; Póczos, B.; Salakhutdinov, R.; Wang, R.; and Xu, K. 2019. Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. In *NeurIPS*.

Flamary, R.; Cuturi, M.; Courty, N.; and Rakotomamonjy, A. 2018. Wasserstein discriminant analysis. *Machine Learning* 107(12): 1923–1945.

Gómez, L.; Chiêm, B.; and Delvenne, J. 2017. Dynamics Based Features For Graph Classification. *arXiv: Machine Learning* .

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9729–9738.

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

Kriege, N.; and Mutzel, P. 2012. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483* .

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Luo, Z.; Liu, L.; Yin, J.; Li, Y.; and Wu, Z. 2017. Deep learning of graphs with ngram convolutional neural networks. *IEEE Transactions on Knowledge and Data Engineering* 29(10): 2125–2139.

Ma, Y.; Wang, S.; Aggarwal, C.; and Tang, J. 2019. Graph Convolutional Networks with EigenPooling. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* .

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019. Provably Powerful Graph Networks. In *NeurIPS*.

Morris, C.; Kersting, K.; and Mutzel, P. 2017. Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, 327–336. IEEE.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*, 2014–2023.

Orsini, F.; Baracchi, D.; and Frasconi, P. 2018. Shift aggregate extract networks. *Frontiers in Robotics and AI* 5: 42.

Orsini, F.; Frasconi, P.; and De Raedt, L. 2015. Graph invariant kernels. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence*, volume 2015, 3756–3762. IJCAI-INT JOINT CONF ARTIF INTELL.

Rieck, B. A.; Bock, C.; and Borgwardt, K. 2019. A Persistent Weisfeiler-Lehman Procedure for Graph Classification. In *ICML*.

Rolet, A.; Cuturi, M.; and Peyré, G. 2016. Fast dictionary learning with a smoothed Wasserstein loss. In *Artificial Intelligence and Statistics*, 630–638.

Sandryhaila, A.; and Moura, J. M. 2013. Discrete signal processing on graphs. *IEEE transactions on signal processing* 61(7): 1644–1656.

Schmitz, M. A.; Heitz, M.; Bonneel, N.; Ngole, F.; Coeurjolly, D.; Cuturi, M.; Peyré, G.; and Starck, J.-L. 2018. Wasserstein dictionary learning: Optimal transport-based unsupervised nonlinear dictionary learning. *SIAM Journal on Imaging Sciences* 11(1): 643–678.

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

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, 488–495.

Simou, E.; Thanou, D.; and Frossard, P. 2020. node2coords: Graph Representation Learning with Wasserstein Barycenters. *arXiv preprint arXiv:2007.16056* .

Titouan, V.; Courty, N.; Tavenard, R.; and Flamary, R. 2019. Optimal Transport for structured data with application on graphs. In *International Conference on Machine Learning*, 6275–6284.

Toivonen, H.; Srinivasan, A.; King, R. D.; Kramer, S.; and Helma, C. 2003. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics* 19(10): 1183–1193.

Wale, N.; Watson, I. A.; and Karypis, G. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems* 14(3): 347–375.

Yanardag, P.; and Vishwanathan, S. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374.

Zhang, Z.; Wang, M.; Xiang, Y.; Huang, Y.; and Nehorai, A. 2018. Retgk: Graph kernels based on return probabilities of random walks. In *Advances in Neural Information Processing Systems*, 3964–3974.