Graph–Graph Similarity Network

Han Yue, Pengyu Hong, and Hongfu Liu, *Member, IEEE*

***Abstract*— Graph learning aims to predict the label for an entire graph. Recently, graph neural network (GNN)-based approaches become an essential strand to learning low-dimensional continuous embeddings of entire graphs for graph label prediction. While GNNs explicitly aggregate the neighborhood information and implicitly capture the topological structure for graph representation, they ignore the relationships among graphs. In this article, we propose a graph–graph (G2G) similarity network to tackle the graph learning problem by constructing a SuperGraph through learning the relationships among graphs. Each node in the SuperGraph represents an input graph, and the weights of edges denote the similarity between graphs. By this means, the graph learning task is then transformed into a classical node label propagation problem. Specifically, we use an adversarial autoencoder to align embeddings of all the graphs to a prior data distribution. After the alignment, we design the G2G similarity network to learn the similarity between graphs, which functions as the adjacency matrix of the SuperGraph. By running node label propagation algorithms on the SuperGraph, we can predict the labels of graphs. Experiments on five widely used classification benchmarks and four public regression benchmarks under a fair setting demonstrate the effectiveness of our method.**

***Index Terms*— Graphs, metric learning, neural networks, supervised learning.**

I. Introduction

MACHINE learning, as a part of artificial intelligence, is the study of computer algorithms that automatically perform tasks through training by data. In recent decades, the topics of machine learning have received a great deal of interest from both academic and industrial areas due to the exponential increase in computing power and available data. Machine learning methods have been successfully applied to many tasks such as classification, regression, and clustering.

One of the most popular types of the machine learning algorithms is artificial neural network (ANN), which is based on a collection of connected neurons. Compared with the conventional regression and statistical models, ANNs [1], [2], [3] have the ability to perform nonlinear modeling and implicitly detect interactions between input features, thus are relatively effective and competitive. In fact, ANNs have been used in various kinds of real-world applications such as object detection [4], speech recognition [5], music generation [6], sentiment analysis [7], and question answering [8], which impact on huge parts of our daily life. As mentioned above, ANNs have the ability to handle a diversity of data structures such as images, sounds, texts, and graphs. We focus on dealing with graph-structured data in this article.

Nowadays, there is an increasing number of applications on graph-structured data ranging from e-commerce to biological molecules. In the graph structure, each node represents an entity, and edges linking two nodes denote the relationship between the two entities. For instance, in citation networks, each node denotes an article, and edges indicate the citationships between articles. Another example is that in social networks, nodes and edges, respectively, represent users and their friendships. As graphs are always irregular with various numbers of nodes and edges, it is difficult to apply convolutional neural networks (CNNs) [9] to the graph domain directly. Encouraged by the success of CNNs, graph neural networks (GNNs) [10] have recently become the most popular tool for graph-structured data by effectively combining node features and graph topology. The research on GNNs is mainly developing into two lines. The first category is to infer labels of individual nodes by a given graph structure, and the second one is to predict the labels of unseen graphs by a given set of graphs with different structures and sizes. In this article, we focus on the second one, the graph learning problem.

The majority of the GNN-based methods [11], [12], [13], [14], [15], [16], [17], [18], [19] mainly involve transforming, propagating, and aggregating node features across the graph, and some other methods [20], [21] have been proposed to learn the graph embeddings in an unsupervised way by adopting an autoencoder framework. In graph learning, the representations generated by convolution are high-level node representations, where the unfixed structures and number of nodes bring in huge difficulties. It is essential to align all the graphs to the same distribution. On the other hand, current graph learning methods seek independent graph representation. Unfortunately, the relationships among different graphs are ignored.

In this article, we propose a graph–graph (G2G) similarity network to tackle the supervised graph learning problem by constructing a SuperGraph from all the input graphs. While previous GNN-based approaches focus on generating graph-level representations to denote the identities of graphs, we present to find the relationship between graphs and make the prediction with this global information. Specifically, we capture high-level node representations by graph convolutional network (GCN) [22]. Then we use the adversarial autoencoder [23] to align all the graphs to a prior space distribution to get robust representations. The max-pooling method is used to downsample and collapse node representations into

Manuscript received 20 May 2021; revised 6 July 2022; accepted 23 October 2022. This work was supported by the National Science Foundation (NSF) under Grant OAC-1920147 and Grant DMR-1933525. *(Corresponding author: Han Yue.)*

The authors are with the Michtom School of Computer Science, Brandeis University, Waltham, MA 02453 USA (e-mail: hanyue@brandeis.edu; hongpeng@brandeis.edu; hongfuliu@brandeis.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3218936.

Digital Object Identifier 10.1109/TNNLS.2022.3218936

graph representations. Next, we design a neural network for learning the similarity between graphs by the generated graph representations, which is the key part for the construction of the SuperGraph. On the SuperGraph, we run the K-nearest neighbor algorithm [24] for classification and GCN for regression, and then get the predictions for the graphs. In summary, our contributions are as follows.

1) We address the problem of measuring the similarity between graphs. Compared with the traditional metrics that are predefined and fixed, our method is more flexible and efficient with the guidance of graph labels and network inference.
2) We propose a novel G2G model consisting of GCN, heterogeneous space alignment, G2G similarity network, and SuperGraph to learn effective graph-level representations under a common representation space and explore the complex relationships between graphs, converting the supervised graph learning problem into a semisupervised node label propagation problem, and finally predict labels for graphs.
3) Extensive experimental results demonstrate that our G2G model outperforms other baseline models on all five public classification datasets and three of the four regression datasets. We also provide similarity visualizations and an ablation study on G2G, which demonstrates the effectiveness of our G2G similarity network.

This article is organized as follows. In Section II, we review related works on graph classification, graph regression, and metric learning, and discuss the difference between previous studies and our work. In Section III, we describe our proposed G2G model and the objective function in detail. Section IV and Section V show the experimental results on classification and regression, respectively. In Section VI, we summarize this article and draw a conclusion.

II. Related Works

In this section, we briefly review related work on graph classification, graph regression, and metric learning. Our work is also connected with GNN and generative adversarial network, whose related work is discussed comprehensively and thoroughly in their surveys [25], [26]. At the end of each part, we explain how our method is different from previous studies.

A. Graph Classification

Graph classification is the task of predicting the class labels of unseen graphs. Early popular approaches adopt graph kernels, which allow kernel-based methods such as support vector machine (SVM) [27] to work directly on graphs for graph classification. By decomposing graphs into small substructures (e.g., shortest paths [28], random walks [29], subtree structures [30], or graphlets [31]), the kernel function compares two graphs by comparing all the pairwise substructures. However, graph kernels are restricted to substructures with few nodes because of the combinatorial complexity of substructure enumeration. Recently, GNNs have been popular because they can extract features from graphs efficiently. Graph SAmple and aggreGatE (GraphSAGE) [32] leverages node feature information to efficiently generate node embeddings for previously unseen data. Graph attention networks (GATs) [33] use the attention mechanism to aggregate neighborhood representations with different weights. While the above methods are mainly designed for learning meaningful node representations, they face a computational challenge and are unable to generate graph-level representations. To solve this problem, other GNN-based approaches are often combined with pooling operations, which not only downsample the nodes to generate smaller representations but also obtain a compact representation on the graph level. For example, DiffPool [16] proposes to softly assign nodes to a set of clusters on the basis of a supervised criterion. Deep graph CNN (DGCNN) [15] enables learning from global graph topology by sorting vertex features instead of summing them up.

All the works mentioned above focus on learning independent graph representations, while our work intends to learn effective graph-level representations under a common representation space and explore the relationships between graphs. We classify graphs based on their relationships.

B. Graph Regression

Graph regression aims to predict labels for graphs, which is similar to graph classification except that the labels are continuous. Many graph regression tasks with public datasets are to predict the properties of molecules and materials, where the topological graphs are defined by atoms and chemical bonds. Recently, several works have been done for this kind of task by the machine learning methods. SchNet [34] adopts a constant cutoff distance and uses the atomic number and atom coordinates for prediction. Message passing neural networks (MPNNs) [35] includes the existing graph models and modifies the update functions for crystal structures. [36] extends SchNet and MPNNs with an edge update network, which allows the information exchanged between atoms to be dependent on the sending and receiving atoms. Crystal graph CNNs (CGCNNs) [37] proposes to encode both atomic information and bonding interactions between atoms as the representation of crystal structures, which uses not only the topological information but also the spatial information. It then achieves predictions through convolution and pooling layers. Crystal graph neural network (CGNN) [38] presents that the distance features are inessential and proposes a model without using spatial information.

Similar to the graph classification methods, previous graph regression methods seek to learn graph representations based on the intrinsic properties of graphs. Differently, we propose to involve both the informative graph-level features and the similarity between graphs when predicting labels for graphs.

C. Metric Learning

Many distance metrics are based on calculating the difference or similarity between two data points. The traditional distance metrics are commonly used in machine learning, such as Euclidean distance [39], cosine distance [40], and Wasserstein distance [41], [42]. Differently, metric learning aims to learn distance functions, which are then used to support tasks such as classification and regression. Some methods [43], [44] learn a linear transformation of the input data, while others [45], [46], [47], [48], [49] use deep neural networks

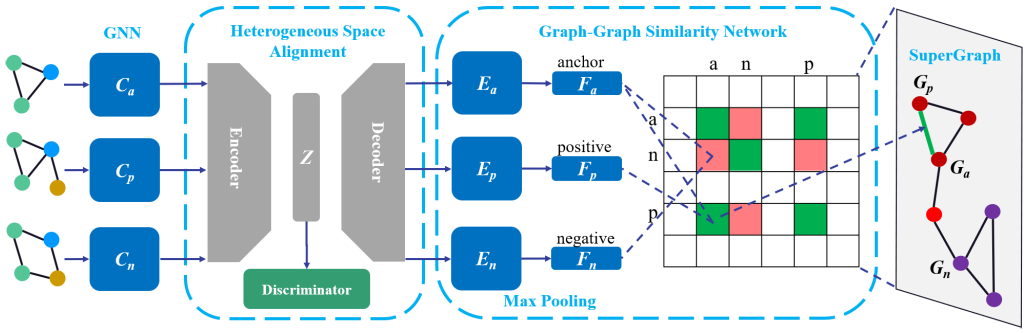


Fig. 1. G2G model overview. There are four components in the framework: GCN, heterogeneous space alignment, G2G similarity network, and SuperGraph. Graph feature representations and a similarity matrix are learned by the network for SuperGraph construction. After that, a node label propagation algorithm is applied to the SuperGraph for the prediction of graph labels.

for nonlinear metric learning, which is known as deep metric learning. Higher-order Siamese graph convolutional neural network (HS-GCN) [46] incorporates higher order proximity in GCNs and leverages it for the similarity learning task on brain networks. SimGNN [48] designs a neural network with graph-level embedding interaction and pairwise node comparison to calculate similarity scores between graphs based on a specific similarity metric. GraphSim [47] directly matches two sets of node embeddings for graph similarity computation. Graph matching network (GMN) [49] learns a similarity score between a pair of graphs through a cross-graph attention-based matching mechanism. Driven by loss functions, deep metric learning learns feature embedding from the input data, which shows the importance of the loss function definition. Contrastive loss [50] is one of the most popular pairwise losses, which minimizes the distance between pairs of the same class (positive) and maximizes the distance between pairs of different classes (negative). Different from contrastive loss, triplet loss [51] is defined based on an anchor sample, a positive sample, and a negative sample. Triplet loss defines the difference by relative similarity, and thus suits more scenarios compared with the contrastive loss. A-softmax [52] projects the Euclidean space of features to an angular space and introduces an angular margin to increase the separability between classes.

Previous distance metrics and deep metric learning methods are all designed to compute similarity scores for pairs of graphs. Instead of getting a similarity score for a pair of graphs each time, our method directly outputs a similarity matrix that shows the pairwise similarities between all the graphs.

III. Methodology

In this section, we first elaborate on the graph learning problem, then introduce our proposed G2G model, followed by the designated objective function in detail.

A. Problem Definition

A graph can be represented by $G = (V, X, A)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertexes, $X \in \mathbb{R}^{n\times d}$ denotes the features of each vertex, and $A \in \{0, 1\}^{n\times n}$ represents the adjacency matrix. Given a set of labeled graphs $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_N, y_N)\}$ where $y_i \in \mathcal{Y}$ is the corresponding label of graph $G_i \in \mathcal{G}$, the graph learning problem can be defined as to learn a mapping function $f : \mathcal{G} \to \mathcal{Y}$, which maps graphs to labels. To achieve the goal, we further define a procedure $g : \mathcal{G} \to (F, S)$ to convert graphs into finite-dimensional representations $F \in \mathbb{R}^{N\times D}$ and a similarity matrix $S \in \mathbb{R}^{N\times N}$ denoting the similarity between graphs. N is the number of graphs in $\mathcal{G}$, and D is the dimension of the produced representations. A mapping function $h : (F, S) \to \mathcal{Y}$ is used to predict labels for graphs.

In this graph learning problem, there are three challenges we have to deal with. The first one is how to get a fixed-length vector representation for each graph. A graph consists of vertexes, vertex features, and the relationship between vertexes. To apply standard machine learning methods, it is necessary to find a way to extract useful feature vectors that contain both feature and relationship information of nodes from these input graphs. The second one is how to align the graphs to the same distribution. The existing GNN-based methods aggregate the node-level embeddings to present a graph, which cannot guarantee that all the graph representations are in the same feature space due to the differences in sizes and structures of graphs. Thus, it is essential to learn a common graph-level representation space. The third one is how to calculate a meaningful similarity between graphs. With vector representations of graphs, Euclidean distance, cosine distance, Wasserstein distance, and graph alignment techniques can be used for graph similarity. However, these predefined metrics are inflexible to capture the similarity across multiple graphs.

B. Model

The goal of graph learning is to predict labels associated with the graphs. While most GNN-based approaches generate graph-level representations for graphs using the node features and graph structure information, they ignore the connections between graphs. To exploit this relationship instead of only focusing on the identities of graphs, we present a model to calculate the similarity between graphs and build a SuperGraph by the similarity and graph-level representations of graphs, which converts the graph learning problem to the node label propagation problem and thus involves the relationship between graphs.

Specifically, to solve the graph learning problem and tackle the above challenges, we proposed the G2G model illustrated in Fig. 1. We first use the existing GNN-based methods to

combine both the node features and the relationship between nodes of a graph. Then an adversarial autoencoder is used to learn an invariant space for all the graph representations, which is essential for similarity calculation. The generator produces fake graph representation, while the discriminator judges whether the input graph representation is real or fake. The generator learns a universal graph space, and all the graph representations from GNN are fed into the unified generator to eliminate the different feature spaces' issue. After the alignment, a pooling layer is added to get finite-dimensional vector representations for graphs with different numbers of nodes. With the features for graphs, a G2G similarity network trained by the guidance of a label matrix with part of predefined values is designed to get the similarity between graphs, solving the problem of similarity calculation. For classification, we adopt zero-one loss and triplet loss. The zero-one loss works as a guide for training, and the triplet loss makes the generated pairwise similarity meaningful by encouraging each graph to be closer to graphs that belong to the same class with it than graphs of different classes. For regression, we adopt mean squared error as the loss function, guiding the training of the G2G similarity network. The feature representations of graphs, together with the similarity matrix, can formulate another graph, which is called the SuperGraph in this article. Each node of the SuperGraph denotes an input graph, and the weights of edges in the SuperGraph are denoted by the similarity matrix. Then the graph learning problem is transformed into the node label propagation problem. Finally, a standard machine learning method for node label propagation can be used on the SuperGraph to predict the labels for the graphs, which captures not only the features of graphs but also the relationship between graphs. In general, our model takes both graph structure similarity and graph label similarity into consideration. The graph structure similarity is captured by the GNN and the heterogeneous space alignment part, and the Supergraph takes the graph labels into account. The model takes graphs as the input and outputs the feature representations of graphs and the similarity between graphs in the first stage. Then for the second stage, we run a node propagation method on the SuperGraph to get the predictions of graph labels.

C. Objective Function

Our model consists of four parts: GNN, heterogeneous space alignment, G2G similarity network, and SuperGraph. In this section, we use $\theta = \{\theta_G, \theta_Q, \theta_P, \theta_S\}$ to denote the trainable parameters' set of four parts in the G2G model. Specifically, θ_G denotes the trainable parameters in the GNN part, θ_Q is the trainable parameters in the encoder, θ_P is the trainable parameters in the decoder, and θ_S represents the trainable parameters in the G2G similarity network. Our goal is to minimize the objective function by adjusting θ by the model and dataset. Each part is detailed as follows.

1) Graph Neural Network: We use GNN to make use of both node features and the relationship of nodes in a graph. While the features of nodes are of the same dimension, a shared GNN can be used for all the graphs. It is flexible to adopt various GNNs for this part, and here we use GCN [22], a standard technique, for illustration. The GCN embedding of the static inputs is

$$C_\theta^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}C_\theta^{(l)}\theta_G^{(l)}\right). \tag{1}$$

Here, $\tilde{A} = A + I_n$ is the adjacency matrix A with added self-connections. $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix, $\tilde{D}$ is the degree matrix of $\tilde{A}$, and $\theta_G^{(l)}$ is a layer-specific trainable weight matrix. $C^{(l)}$ denotes the matrix in the lth layer, and $C^{(0)} = X$. We use $\sigma(\cdot) = \mathrm{ReLU}(\cdot)$ as the activation function.

2) Heterogeneous Space Alignment: This part is designed as an adversarial autoencoder [23], which includes an autoencoder and a discriminator, aiming to match the aggregated posterior of the hidden codes with an arbitrary prior distribution using an adversarial training procedure. The autoencoder is popularly used for data embedding, which provides informative low-dimensional representations of input data by mapping them to the latent space. While the latent code space is free of any structure, the autoencoder can easily result in poor representation when dealing with graph data due to the differences in sizes and structures of graphs. Therefore, a discriminator is introduced as a complementary regularizer to handle this problem [21], [23], [53], which provides more robust space representation learning by enforcing the latent codes to follow some prior data distribution.

Specifically, let C be the input and z be the hidden code of the autoencoder. Denote by $q(z|C)$ and $q(C|z)$ an encoding distribution and the decoding distribution, respectively. Then the aggregated posterior distribution of $q(z)$ on the hidden codes can be computed as $q(z) = \int_C q(z|C)p(C)dc$, where $p(C)$ is the marginal distribution of inputs. Let $p(z)$ be the prior distribution one wants to impose on the codes. The network minimizes the reconstruction error for the autoencoder, and meanwhile guides $q(z)$ to match $p(z)$ by attaching an adversarial network on top of the hidden codes.

Let E be the generated embeddings. Here, we use $Q(X) = \sigma(X \times \theta_Q)$ to denote the encoder and $P(X) = \sigma(X \times \theta_P)$ the decoder. Then $E_\theta = P(Q(C_\theta))$. $D(\cdot)$ denotes the discriminator telling apart the true hidden codes sampled based on z. Then the reconstruction loss of the autoencoder looks like

$$\mathcal{L}_{\mathcal{R};\theta} = \sum_i^N \|E_{i;\theta} - C_{i;\theta}\|_2^2. \tag{2}$$

We use the max-pooling method to get vector representations from the hidden layer of the autoencoder. The adversarial loss is calculated by the following equation:

$$\begin{aligned}\mathcal{L}_{\mathcal{A};\theta} = {} & \mathbb{E}_{z\sim p(z)}[\log D(z)] \\ & + \mathbb{E}_{C\sim p(C)}[\log(1 - D(\mathrm{maxpool}(Q(C_\theta))))].\end{aligned} \tag{3}$$

3) G2G Similarity Network: The features of graphs F are generated by a pooling layer. Here, we also use the max-pooling method, that is, $F_\theta = \mathrm{maxpool}(E_\theta)$. Then to get the similarity between graphs, we use a neural network to simulate the calculation process, which is shown as follows:

$$S_\theta = \sigma(F_\theta \cdot \theta_S) \cdot \sigma(F_\theta \cdot \theta_S)^\top. \tag{4}$$

TABLE I
STATISTICS OF FIVE DATASETS, MUTAG, ENZYMES, PROTEINS, D&D, AND NCI1

Dataset	# Graph	# Class	Avg. # Node	Avg. # Edge	# Node Label	Node Attr.
MUTAG [54]	188	2	17.93	19.79	7	No
ENZYMES [55]	600	6	32.63	62.14	3	Yes
PROTEINS [56]	1113	2	39.06	72.82	3	Yes
D&D [57]	1178	2	284.32	715.66	89	No
NCI1 [58]	4110	2	29.87	32.30	37	No

To guide the calculation of similarity matrix, we first build a label matrix Y. For classification, $Y_{i,j} = 1$ if $y_i \neq y_j$, and otherwise $Y_{i,j} = 0$. Then we use both zero-one loss and triplet loss [51] as the constraints, which are designed as

$$\mathcal{L}_{\mathcal{Z};\theta} = \sum_{i}^{N}\sum_{j}^{N}((1-S_{i,j;\theta})^2 \odot Y_{i,j} + S_{i,j;\theta}^2 \odot (1-Y_{i,j})) \quad (5)$$

$$\mathcal{L}_{\mathcal{T};\theta} = \sum_{(G_a,G_p,G_n)} \max\left(S_{a,p;\theta}^2 - S_{a,n;\theta}^2 + \alpha, 0\right) \quad (6)$$

where α is the margin of triplet loss, and (G_a, G_p, G_n) denotes a triplet of graphs. G_a is the anchor graph, G_p is the positive graph, and G_n is the negative graph. Here, positive/negative means the samples that have the same/different labels with a chosen sample (also known as anchor). Combining (5) and (6), the loss for G2G similarity network is

$$\mathcal{L}_{\mathcal{S};\theta} = \lambda_Z \mathcal{L}_{\mathcal{Z};\theta} + \lambda_T \mathcal{L}_{\mathcal{T};\theta} \quad (7)$$

where λ_Z and λ_T are the weights of zero-one loss and triplet loss, respectively.

For regression, $Y_{i,j} = |y_i - y_j|$. We then adopt mean squared error as the loss function to train the G2G similarity network, which is formalized as follows:

$$\mathcal{L}_{\mathcal{S};\theta} = \frac{1}{N^2}\sum_{i}^{N}\sum_{j}^{N}(Y_{i,j} - S_{i,j;\theta})^2. \quad (8)$$

Combining (2), (3), (7), and (8), our overall objective function is shown as follows:

$$\min_{\theta} \mathcal{L}_{\mathcal{R};\theta} + \lambda_A \mathcal{L}_{\mathcal{A};\theta} + \lambda_S \mathcal{L}_{\mathcal{S};\theta} \quad (9)$$

where λ_A and λ_S are the hyperparameters controlling the weights of $\mathcal{L}_{\mathcal{A}}$ and $\mathcal{L}_{\mathcal{S}}$, respectively. Then we use stochastic gradient descent (SGD) [61] to optimize the objective function in the discriminator and Adam [62] to minimize other parts.

4) SuperGraph: With the generated features F and the similarity matrix S, we get a topological graph representation, which is called the SuperGraph in this article. In the SuperGraph, each node corresponds to an input graph, and the weight of each edge denotes the relationship between the corresponding graphs of the two vertexes. With this SuperGraph, we turn the graph learning problem into the node label propagation problem, and many machine learning methods such as GCN [22] can be applied to it without limitation. In our experiments, we use the K-nearest neighbor algorithm [24] for the classification task and GCN for the regression task on the SuperGraph and get the predictions.

IV. EXPERIMENTS ON CLASSIFICATION

In this section, we first describe the datasets and experimental setup for the graph classification task, and then evaluate our model. Finally, we provide exploitative experiments for ablation study and comparison with different distance metrics.

A. Dataset

We evaluate our model on five public datasets, graphs in which are derived from small chemical compounds or protein molecules: MUTAG [54], ENZYMES [55], PROTEINS [56], D&D [57], and NCI1 [58]. Table I shows the statistics and properties of each dataset. In these datasets, each graph belongs to only one class, and all the nodes are labeled. For ENZYMES and PROTEINS, node attributes are provided, which are used as node features in our experiment. For the other three datasets, we use one-hot node labels as the node features. For all the datasets, we perform ten-fold cross validation [63] to evaluate model performance and report the accuracy averaged over ten folds. Specifically, we use the splits of the datasets provided by [64], which provides a grounding for rigorous evaluations of graph classification models. While they did not include a split for MUTAG, we generate a ten-fold split for MUTAG and test all the methods on it.

B. Baselines and Experimental Settings

We select five popular methods with strong architectural differences as baselines: GraphSAGE [32], graph isomorphism network (GIN) [59], edge-conditioned convolution (ECC) [60], DGCNN [15], and DiffPool [16]. GraphSAGE first performs sum, mean, or max-pooling neighborhood aggregation. GIN extends it with arbitrary aggregation functions on multisets. ECC weights neighbor aggregation according to specific parameters learned for each edge label. DGCNN proposes a SortPooling layer, which performs pooling by sorting vertex features into a meaningful order. DiffPool learns a differentiable soft cluster assignment for nodes at each layer of a deep GNN to hierarchically pool graph nodes. All the five methods use the information within a graph to predict its label. The settings of each baseline method follow the work of [64] and the displayed experimental results of the baseline methods in Table II on all the datasets except for MUTAG. For MUTAG, we run the tool provided by [64] with their settings.[1]

[1]Since we follow the setting of [64] for fair comparisons, the performance of the baseline methods might be different from the ones reported in the original articles.

TABLE II
EXPERIMENTAL RESULTS (% ACCURACY WITH STANDARD DEVIATION). THE BEST SCORES ARE IN **BOLD**

Methods	MUTAG	ENZYMES	PROTEINS	D&D	NCI1
GraphSAGE [32]	83.3 ± 9.0	58.2 ± 6.0	73.0 ± 4.5	72.9 ± 2.0	76.0 ± 1.8
GIN [59]	83.9 ± 7.8	59.6 ± 4.5	73.3 ± 4.0	75.3 ± 2.9	80.0 ± 1.4
ECC [60]	75.9 ± 6.4	29.5 ± 8.2	72.3 ± 3.4	72.6 ± 4.1	76.2 ± 1.4
DGCNN [15]	80.3 ± 10.9	38.9 ± 5.7	72.9 ± 3.5	76.6 ± 4.3	76.4 ± 1.7
DiffPool [16]	81.2 ± 9.3	59.5 ± 5.6	73.7 ± 3.5	75.0 ± 3.5	76.9 ± 1.9
G2G (Ours)	**90.3 ± 6.4**	**68.0 ± 3.6**	**78.3 ± 2.2**	**77.0 ± 4.8**	**80.3 ± 2.4**

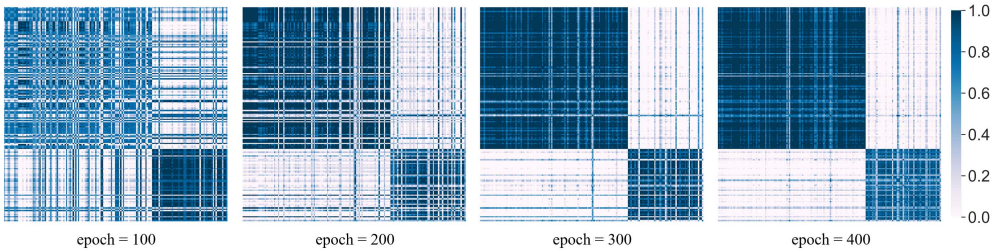


Fig. 2. Similarity matrices of the training set of MUTAG along different iterations. The final similarity matrix of MUTAG with epoch=500 is in Fig. 3.

In experiments, we use pseudonodes to pad the input graphs to the same size. The features of pseudonodes are zero-vectors, and the pseudonodes do not link to any other nodes. All the network parameters of our proposed G2G model are initialized by Xavier uniform [65]. The hyperparameter settings are as follows by default. The weights of the adversarial loss λ_A, the zero-one loss λ_Z, and the triplet loss λ_T are set to 0.1, 1.0, and 1.0, respectively. The margin of triplet loss α is 0.5. We adopt a two-layer GCN with hidden dimensions of 128 and 64 and apply batch normalization after each layer of the model. The dimension of the hidden code of the autoencoder is set to 32. We run the training stage 500 epochs for each dataset. The Adam optimizer with a learning rate of 0.0003 is used to minimize the loss for the G2G model, and the SGD optimizer with a learning rate of 0.0001 is used to optimize the trainable parameters in the discriminator. For the k-nearest neighbors (KNNs) classifier, we set $k = 5$ by default.

C. Graph Classification Results

The classification results are listed in Table II. As can be seen, our proposed G2G model achieves the best scores on all the five datasets. The experiments demonstrate that our method consistently outperforms the baseline methods on all the five datasets. Moreover, our results excel the second-best method by nearly 5%, 10%, and 5% on MUTAG, ENZYMES, and PROTEINS and pass all the significance tests with p-value <0.05. While on the other two datasets, our results also pass the significance tests against other baseline methods, except for DGCNN on D&D and GIN on NCI1, which indicates that our model performs significantly better than the baseline models in most cases. The reason is that previous GNN-based graph classification methods build a classifier on top of the graph-level representations, while we transform it to label propagation on the SuperGraph. Instead of using independent graph information like other methods, our model incorporates the relationship between graphs by the generated pairwise similarity matrix. Together with the zero-one loss which drives the training process of the G2G similarity network, the triplet loss further promotes graphs closer to their corresponding positive graphs than to the negative ones under our learned distance metric.

To further analyze the performance of the G2G model, we visualize the similarity matrix of the SuperGraph generated by our model. Fig. 2 displays the similarity matrices on one training set of MUTAG along with iterations, where the darker color denotes the higher similarity. Here, we organize the graphs by their ground-truth labels for visualization, where graphs belonging to the same class are put together. The X-axis and Y-axis indicate the organized indices of graphs (from top left to bottom right). It is expected to see that the similarity matrix becomes more and more block-structured with more iterations, which indicates the effectiveness of the G2G model. Moreover, Fig. 3 shows the similarity matrices on one training set of MUTAG, ENZYMES, D&D, and NCI1, and their similarity matrices on the corresponding testing sets. The classes can be easily identified on the training set, and the dark squares lie on the diagonal of the matrix, demonstrating the effectiveness of our designed objective function for training. Although the performance drops a little on the testing set, the dark squares are still recognizable on the diagonal on MUTAG, D&D, and NCI1. These demonstrate the generalization of our G2G model. The structure of ENZYMES is not as significant as on other datasets. This might result from the multiclasses of graphs. Instead of two classes in the other four datasets, ENZYMES includes six kinds of class labels, which brings more challenges to the classification problem.

D. Ablation Study

To study the impact of the G2G similarity network and the function of SuperGraph, we run an ablation study on the five datasets. The results are reported in Table III. We can clearly see that with heterogeneous space alignment part,

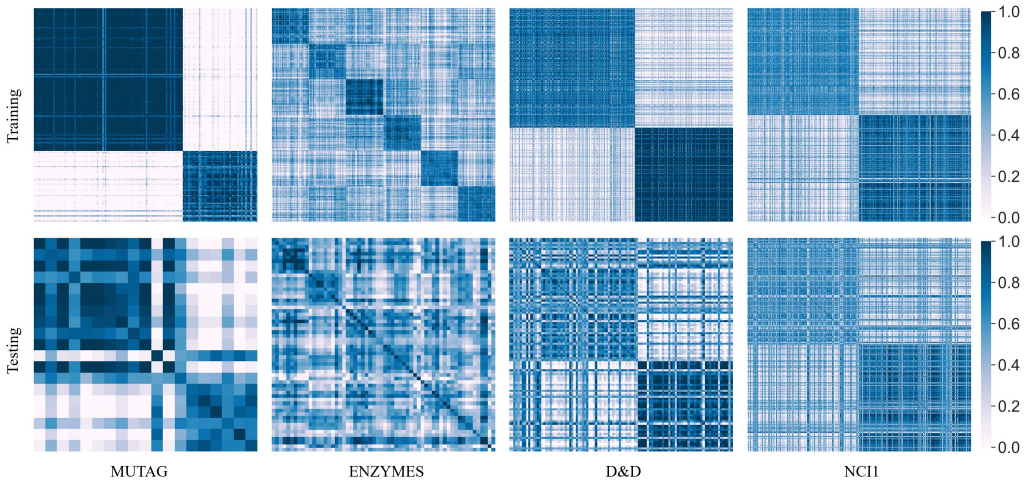


Fig. 3. Visualization of final similarity matrices of the training and test data on MUTAG, ENZYMES, D&D, and NCI1.

TABLE III

ABLATION STUDY RESULTS (% ACCURACY WITH STANDARD DEVIATION). THE BEST SCORES ARE IN **BOLD**. HERE C, H, G, AND S, RESPECTIVELY, DENOTE GCN, HETEROGENEOUS SPACE ALIGNMENT, G2G SIMILARITY NETWORK, AND SUPERGRAPH

Models	MUTAG	ENZYMES	PROTEINS	D&D	NCI1
C	83.0 ± 11.5	55.2 ± 6.6	70.4 ± 4.1	72.4 ± 2.5	72.6 ± 2.0
C + H	86.2 ± 7.1	57.7 ± 7.9	70.7 ± 2.8	72.7 ± 4.4	75.1 ± 2.4
C + H + G	87.1 ± 7.1	65.7 ± 6.6	73.0 ± 3.8	74.4 ± 3.6	77.9 ± 2.1
C + H + G + S	**90.3 ± 6.4**	**68.0 ± 3.6**	**78.3 ± 2.2**	**77.0 ± 4.8**	**80.3 ± 2.4**

TABLE IV

TIME CONSUMPTION (SECONDS) OF DIFFERENT METHODS. THE BEST SCORES ARE IN **BOLD**

Methods	MUTAG	ENZYMES	PROTEINS	D&D	NCI1
GraphSAGE [32]	42.5	133.8	263.1	349.7	783.7
GIN [59]	43.2	**119.9**	**209.2**	**349.6**	**534.8**
ECC [60]	92.4	338.9	641.8	986.6	1397.4
DGCNN [15]	41.6	132.4	227.8	378.5	758.4
DiffPool [16]	538.2	1603.9	2931.1	12082.2	10415.3
GMN [49] + Supergraph	425.5	2614.2	49193.7	48827.5	14876.0
SimGNN [48] + Supergraph	44.5	475.2	19787.7	14516.0	7841.9
G2G (Ours)	**18.9**	359.3	8264.5	8151.6	2268.7

the performance improves, which demonstrates the positive effect of adversarial autoencoder, because it learns a common graph-level representation space. While by adding a G2G similarity network to the "C + H" model, the performance is improved even without using the learned similarity. It is because the triplet loss optimizes the representations such that graphs belonging to the same class are closer to each other than those of different classes. The "C + H + G" model already outperforms the baseline methods on MUTAG and ENZYMES and has comparable results with them on PROTEINS, D&D, and NCI1. Furthermore, building the SuperGraph with the generated similarity matrix increased the performance again. Overall, the controlled ablation study in Table III shows that each component has a necessary and positive effect on the G2G model, which verifies our motivations.

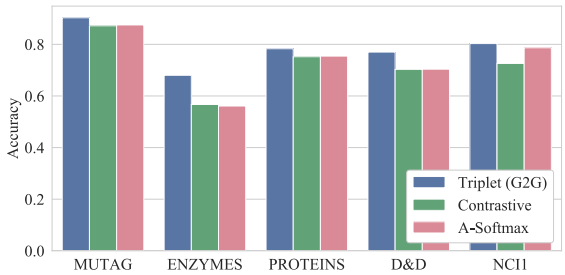


Fig. 4. Graph classification results based on different metric learning losses for the calculation of graph similarity.

E. Comparison With Different Metric Learning Losses

We future explore the impact of triplet loss by replacing it with contrastive loss [50] and A-Softmax loss [52]. Fig. 4

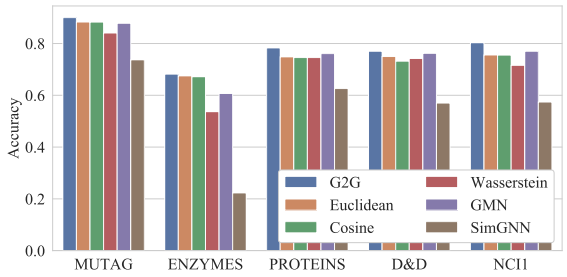


Fig. 5. Graph classification results based on different distance metrics for the calculation of graph similarity.

shows the classification results of G2G with different metric learning losses. Contrastive loss and A-Softmax loss achieve similar results, and triplet loss outperforms both. Among the three losses, triplet loss is the only one that considers relative similarity and thus more suits for learning a similarity matrix for all the graphs in our case.

F. Comparison With Different Distance Metrics

We run some experiments on the representations extracted by the G2G model with the Euclidean distance [39], cosine distance [40], and Wasserstein distance [41], [42]. We also include graph distances learned by GMN [49] and SimGNN [48] as the comparisons. We first calculate the pairwise distance of the graph-level features and then apply the K-nearest neighbor algorithm like G2G to get the predictions for each kind of distance metric. Fig. 5 exhibits the average accuracy of each distance metric on each dataset, indicating that our G2G similarity network is more effective than other distance metrics in the graph classification problem. Compared with the Euclidean distance, cosine distance, and Wasserstein distance, our method is a learned nonlinear transformation, which is more expressive and flexible. Compared with GMN and SimGNN, our method aligns graphs to the same distribution, which helps in graph representation learning. Both GMN and G2G apply triplet loss, while SimGNN does not use it and cannot capture the relative similarity, which hurts its performance on graph classification tasks when using Supergraph. In addition, our method learns a similarity matrix for all the graphs directly, which is much more efficient in constructing a Supergraph than GMN and SimGNN.

G. Hyperparameter Analysis

We analyze the hyperparameters of G2G by changing the value of one hyperparameter and fixing values of other hyperparameters to their default values described in Section IV-B. Fig. 6 shows the graph classification results on MUTAG. The performance of G2G achieves the best when $\lambda_A = 0.1$, and it keeps dropping as λ_A goes higher or lower. On the other hand, there are no significant trends in the setting of λ_Z and λ_T, so we set them to 1.0 by default. Finally, the performance of G2G drops significantly when $\alpha > 0.6$, indicating that a high margin may hurt the effectiveness of triplet loss because of the limited available samples. Similarly, a low margin ($\alpha < 0.5$) is not a good choice because there are too many easy samples. Therefore, we set the margin of triplet loss $\alpha = 0.5$.

H. Time Consumption

We report the time consumption (seconds) of all the methods in Table IV. All the experiments were conducted on a physical machine with Ubuntu 18.04, a total memory of 64 GB, an advanced micro devices (AMD) Ryzen Threadripper 2920X 12-Core Processor, and an NVIDIA GP102 graphics processing unit (GPU). G2G costs the least time on MUTAG but runs a longer time than GraphSAGE, GIN, ECC, and DGCNN on other datasets. The reason is that G2G calculates a similarity matrix of all the graphs. As the number of graphs increases, the time cost of methods with Supergraph grows N times longer than other methods. Compared with GMN and SimGNN, G2G consumes much less time because G2G calculates a similarity matrix directly, while GMN and SimGNN are designed to calculate similarities between a pair of graphs.

V. Experiments on Regression

In this section, we focus on graph regression. We describe the datasets and experimental setup, and then evaluate our model.

A. Dataset

We evaluate our model on four datasets consisting of chemical compounds: Formation Energy (FE), Band Gap (BG), Estimating the aqueous SOLubility (ESOL) [67], and Lipophilicity [68]. The statistics of the datasets are shown in Table V. The first two datasets, FE and BG, are both collected from The Materials [66], but focus on different properties of chemical compounds. For these four datasets, we also perform ten-fold cross validation [63] to evaluate model performance and report the rooted mean squared error averaged over ten folds.

B. Baselines and Experimental Settings

We select two baseline methods that are popular applied on Materials [66] for comparison: GCN [22] and CGCNN [37]. While GCN is designed for node embedding, we add a max-pooling layer to GCN for the graph regression task. CGCNN is developed to learn material properties from crystal graphs, which encode both atomic information and bonding interactions between atoms. Both the baseline methods are based on the intrinsic properties of graphs. The settings of the baseline models follow the default values provided by [37].

In experiments, we adopt CGCNN as the GNN part in G2G. The dimensions of hidden layers are the same as described in Section IV-B. By default, the weights of the adversarial loss λ_A and the similarity network loss λ_S are set to 0.1 and 1.0, respectively. The learning rate of the Adam optimizer is set to 0.001 to minimize the objective function of G2G, and the learning rate of the SGD optimizer is set to 0.0001 for the discriminator. The results are evaluated based on root mean square error (RMSE).

C. Graph Regression Results

Table VI shows the graph regression results of all the baseline methods on the four regression datasets. Our G2G

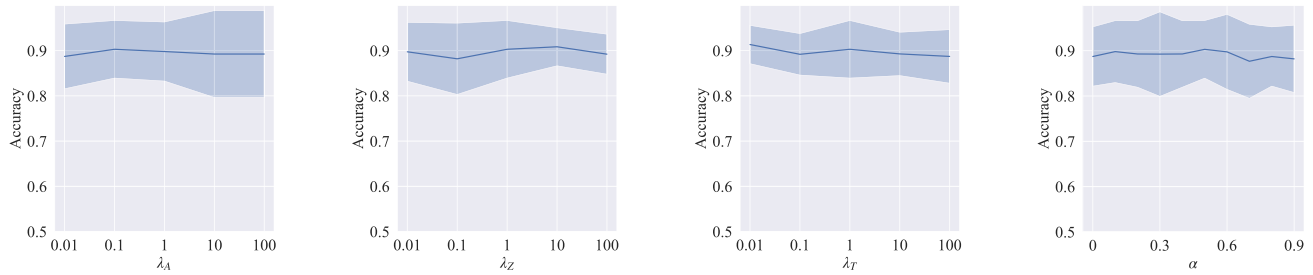


Fig. 6. Graph classification results on MUTAG by G2G with different hyperparameter values.

TABLE V
STATISTICS OF FOUR DATASETS, FE, BG, ESOL, AND LIPOPHILICITY

Dataset	Property	# Graph	Avg. # Node	Avg. # Edge	Node Attr.	Edge Attr.
FE [66]	formation energy	500	8.3	586.6	92	41
BG [66]	band gap	500	8.3	586.6	92	41
ESOL [67]	water solubility	1128	13.3	27.4	93	7
Lipophilicity [68]	octanol/water distribution coefficient	4200	27.0	59.0	93	7

TABLE VI
EXPERIMENTAL RESULTS (RMSE). THE BEST SCORES ARE IN **BOLD**

Methods	FE	BG	ESOL	Lipophilicity
GCN [22]	0.293	0.625	0.461	0.467
CGCNN [37]	**0.221**	0.582	0.432	0.473
G2G (Ours)	0.244	**0.496**	**0.418**	**0.459**

model outperforms GCN on all the four datasets and is also the best on three of them. Unlike GCN and CGCNN which make predictions based only on the intrinsic properties of graphs, our proposed G2G model uses the relationship information of graphs. G2G first learns the pairwise relationship between graphs, then builds a Supergraph and performs node label propagation, which helps with the regression task.

VI. CONCLUSION

In this article, we addressed the supervised graph learning problem and presented a G2G model, which explored the relationship between graphs. Specifically, we first used GNN and adversarial autoencoder to learn effective graph-level representations under a common representation space. Next, a G2G similarity network was designed to learn the similarity between graphs, and a Supergraph was built upon it, which transformed graph learning to node label propagation on the Supergraph. Finally, we adopted KNN/GCN to get predictions for the classification/regression tasks. The experiments on public datasets validated that our method performed better than other baseline models, and the G2G similarity network, as well as the SuperGraph, performed positive impacts on tackling the graph learning tasks.

Limitations: Compared with previous graph classification/regression methods, G2G needs to learn a similarity matrix for all the graphs before predicting, which would increase the time cost when the number of graphs is high. One possible solution is to calculate part of the similarity matrix and build Supergraph based on the partial information.

REFERENCES

[1] B. Du, L. Ru, C. Wu, and L. Zhang, "Unsupervised deep slow feature analysis for change detection in multi-temporal remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 9976–9992, Dec. 2019.

[2] Y. Liu, B. Du, W. Tu, M. Gong, Y. Guo, and D. Tao, "LogDet metric-based domain adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4673–4687, Nov. 2020.

[3] Z. Lu, C. Xu, B. Du, T. Ishida, L. Zhang, and M. Sugiyama, "LocalDrop: A hybrid regularization for deep neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3590–3601, Feb. 2021.

[4] L. Liu et al., "Deep learning for generic object detection: A survey," *Int. J. Comp. Vis.*, vol. 128, no. 2, pp. 261–318, 2020.

[5] J. Padmanabhan and M. J. J. Premkumar, "Machine learning in automatic speech recognition: A survey," *IETE Tech. Rev.*, vol. 32, no. 2, pp. 240–251, 2015.

[6] S. Ji, J. Luo, and X. Yang, "A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions," 2020, *arXiv:2011.06801*.

[7] A. Yadav and D. K. Vishwakarma, "Sentiment analysis using deep learning architectures: A review," *Artif. Intell. Rev.*, vol. 53, no. 6, pp. 4335–4385, Aug. 2020.

[8] E. Dimitrakis, K. Sgontzos, and Y. Tzitzikas, "A survey on question answering systems over linked data and documents," *J. Intell. Inf. Syst.*, vol. 55, no. 2, pp. 233–259, Oct. 2020.

[9] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *Handbook Brain Theory Neural Netw.* vol. 3361, no. 10, p. 1995, Apr. 2015.

[10] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.

[11] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," 2017, *arXiv:1709.05584*.

[12] J. Zhou et al., "Graph neural networks: A review of methods and applications," 2018, *arXiv:1812.08434*.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2020.

[14] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2018.

[15] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[16] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4800–4810.

[17] D. Wang, T. Wang, F. Zhao, and X. Zhang, "Improved graph-based semi-supervised learning for fingerprint-based indoor localization," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[18] D.-C. Juan et al., "Graph-RISE: Graph-regularized image semantic embedding," 2019, *arXiv:1902.10814*.

[19] T. D. Bui, S. Ravi, and V. Ramavajjala, "Neural graph learning: Training neural networks using graphs," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 64–71.

[20] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.

[21] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2018, *arXiv:1802.04407*.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[23] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*.

[24] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, vol. 3. New York, NY, USA: Wiley, 1973.

[25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019, *arXiv:1901.00596*.

[26] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (GANs): A survey," *IEEE Access*, vol. 7, pp. 36322–36333, 2019.

[27] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.

[28] K. M. Borgwardt and H. Kriegel, "Shortest-path kernels on graphs," in *Proc. 5th IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2005, p. 8.

[29] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Mar. 2010.

[30] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler–Lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, Sep. 2011.

[31] N. Kriege and P. Mutzel, "Subgraph matching kernels for attributed graphs," 2012, *arXiv:1206.6483*.

[32] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.

[34] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "SchNet—A deep learning architecture for molecules and materials," *J. Chem. Phys.*, vol. 148, no. 24, Jun. 2018, Art. no. 241722.

[35] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[36] P. B. Jørgensen, K. W. Jacobsen, and M. N. Schmidt, "Neural message passing with edge updates for predicting properties of molecules and materials," 2018, *arXiv:1806.03146*.

[37] T. Xie and J. C. Grossman, "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties," *Phys. Rev. Lett.*, vol. 120, no. 14, Apr. 2018, Art. no. 145301.

[38] T. Yamamoto, "Crystal graph neural networks for data mining in materials science," Res. Inst. Math. Comput. Sci., Yokohama, Japan, 2019. [Online]. Available: https://github.com/Tony-Y/cgnn

[39] L. H. Lee, C. H. Wan, R. Rajkumar, and D. Isa, "An enhanced support vector machine classification framework by using Euclidean distance function for text document categorization," *Int. J. Speech Technol.*, vol. 37, no. 1, pp. 80–99, Jul. 2012.

[40] Y. Kim, "Convolutional neural networks for sentence classification," 2014, *arXiv:1408.5882*.

[41] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[42] C. Frogner, C. Zhang, H. Mobahi, M. Araya, and T. A. Poggio, "Learning with a Wasserstein loss," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2053–2061.

[43] D. Lim and G. Lanckriet, "Efficient learning of Mahalanobis metrics for ranking," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1980–1988.

[44] B. McFee and G. R. Lanckriet, "Metric learning to rank," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 775–782.

[45] G. Ma, N. K. Ahmed, T. L. Willke, and P. S. Yu, "Deep graph similarity learning: A survey," *Data Mining Knowl. Discovery*, vol. 35, no. 3, pp. 688–725, May 2021.

[46] G. Ma et al., "Deep graph similarity learning for brain data analysis," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2743–2751.

[47] Y. Bai, H. Ding, K. Gu, Y. Sun, and W. Wang, "Learning-based efficient graph similarity computation via multi-scale convolutional set matching," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3219–3226.

[48] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "SimGNN: A neural network approach to fast graph similarity computation," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 384–392.

[49] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," 2019, *arXiv:1904.12787*.

[50] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 2006, pp. 1735–1742.

[51] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[52] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "SphereFace: Deep hypersphere embedding for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 212–220.

[53] W. Yu et al., "Learning deep network representations with adversarially regularized autoencoders," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2663–2671.

[54] A. Debnath, R. L. D. Compadre, G. Debnath, A. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity," *J. Med. Chem.*, vol. 34, no. 2, pp. 786–797, Feb. 1991.

[55] I. Schomburg, "BRENDA, the enzyme database: Updates and major new developments," *Nucleic Acids Res.*, vol. 32, no. 90001, pp. 431–433, Jan. 2004.

[56] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, pp. 47–56, Jun. 2005.

[57] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *J. Mol. Biol.*, vol. 330, no. 4, pp. 771–783, Jul. 2003.

[58] N. Wale, I. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowl. Inf. Syst.*, vol. 14, no. 3, pp. 347–375, 2008.

[59] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*.

[60] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3693–3702.

[61] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.

[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[63] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *J. Mach. Learn. Res.*, vol. 11, pp. 2079–2107, Jul. 2010.

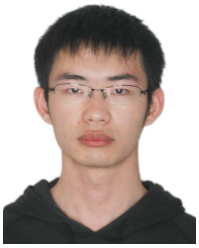
[64] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," 2019, *arXiv:1912.09893*.

[65] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[66] A. Jain et al., "Commentary: The materials project: A materials genome approach to accelerating materials innovation," *APL Mater.*, vol. 1, no. 1, 2013, Art. no. 011002.

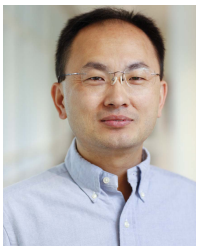
[67] J. S. Delaney, "ESOL: Estimating aqueous solubility directly from molecular structure," *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 3, pp. 1000–1005, May 2004.

[68] Z. Wu et al., "MoleculeNet: A benchmark for molecular machine learning," *Chem. Sci.*, vol. 9, no. 2, pp. 513–530, 2018.



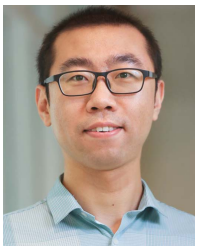
Han Yue received the bachelor's degree in computer science and technology from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, in 2012, and the master's degree in computer science and technology from the Institute of Software, University of Chinese Academy of Sciences, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the Michtom School of Computer Science, Brandeis University, Waltham, MA, USA.

His research interests include data mining and machine learning.



Pengyu Hong received the bachelor's and master's degrees in computer science from Tsinghua University, Beijing, China, in 1995 and 1997, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2001.

He is currently a Professor of computer science with Brandeis University, Waltham, MA, USA. His research interests focus on machine learning and its applications in bioinformatics, glycomics, materials research, neuroscience, medical informatics, FinTech, and so on.



Hongfu Liu (Member, IEEE) received the bachelor's and master's degrees in management information systems from the School of Economics and Management, Beihang University, Beijing, China, in 2011 and 2014, respectively, and the Ph.D. degree in computer engineering from Northeastern University, Boston, MA, USA, in 2018.

He is currently a tenure-track Assistant Professor affiliated with the Michtom School of Computer Science, Brandeis University, Waltham, MA, USA. His research interests generally focus on data mining and machine learning, with special interests in ensemble learning. He has served as the reviewers for many IEEE Transactions journals including the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), the IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), and the IEEE TRANSACTIONS ON BIG DATA (TBD). He has also served on the program committee for the conferences including Association for the Advancement of Artificial Intelligence (AAAI), International Joint Conferences on Artificial Intelligence (IJCAI), and Neural Information Processing Systems (NIPS).

Dr. Liu is the Associate Editor of the *IEEE Computational Intelligence Magazine*.