Label Attentive Distillation for GNN-Based Graph Classification

Xiaobin Hong[1], **Wenzhong Li**[1*], **Chaoqun Wang**[2], **Mingkai Lin**[1], **Sanglu Lu**[1]
[1]State Key Laboratory for Novel Software Technology, Nanjing University
[2]The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), China
{xiaobinhong, mingkai}@smail.nju.edu.cn, chaoqunwang@link.cuhk.edu.cn, {lwz, sanglu}@nju.edu.cn

Abstract

Graph Neural Networks (GNNs) have emerged as a powerful tool for modeling graph-structured data, exhibiting remarkable potential in applications such as social networks, recommendation systems, and molecular structures. However, the conventional GNNs perform node-level feature aggregation from neighbors without considering graph-label information, which leads to the misaligned embedding problem that may cause a detrimental effect on graph-level tasks such as graph classification. In this paper, we propose a novel label-attentive distillation method called LAD-GNN for graph representation learning to solve this problem. It alternatively trains a teacher model and a student GNN with a distillation-based approach. In the teacher model, a label-attentive encoder is proposed to encode the label information fusing with the node features to generate ideal embedding. In the student model, the ideal embedding is used as intermediate supervision to urge the student GNN to learn class-friendly node embedding to facilitate graph-level tasks. Generally, LAD-GNN is an enhanced GNN training approach that can be incorporated with arbitrary GNN backbone to improve performance without significant increase of computational cost. Extensive experiments with 7 GNN backbones based on 10 benchmark datasets show that LAD-GNN improves the SOTA GNNs in graph classification accuracy. The source codes of LAD-GNN are publicly available on https://github.com/XiaobinHong/LAD-GNN.

Introduction

Graph Neural Networks (GNNs) (Kipf and Welling 2016, 2017) are adept at converting unstructured data into low-dimensional representations by effectively capturing both node features and topological dependencies. The GNN learning tasks broadly focused on *node classification*, *link prediction*, and *graph classification* (You et al. 2021; Wang et al. 2021), which have demonstrated their effectiveness in various fields such as protein-to-protein interaction (Nouranizadeh et al. 2021), molecular medicine (Li et al. 2022b) information retrieval (Chen et al. 2022), etc. This paper focuses on graph classification, a graph-level task that aims to learn a graph representation with GNNs to predict the graph labels.

[*]Corresponding author.

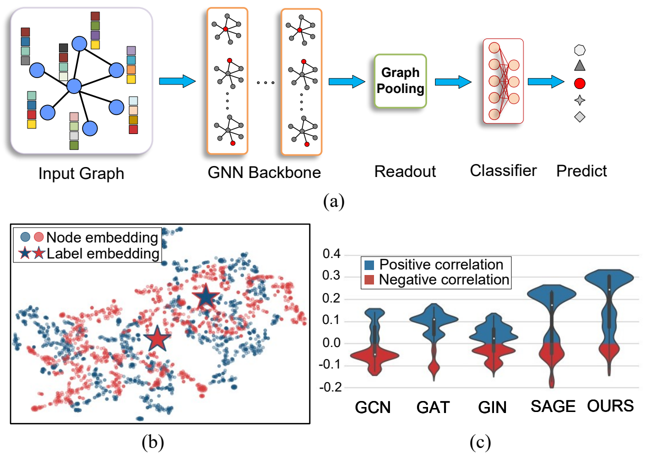


Figure 1: (a) The pipeline of conventional GNN-based graph classification. (b) Illustration of misaligned node embeddings and label embeddings for GraphSAGE on the MUTAG dataset, where different colors denote different graph classes. (c) The violin plots of correlation statistics of node/label embeddings on the MUTAG dataset. It shows that our method has fewer negative correlations and much higher positive correlation scores compared to the other GNNs.

The conventional pipeline of GNN-based graph classification is shown in Fig. 1 (a), which consists of the following processes: (1) The input graph is fed into a GNN backbone to generate node embedding by aggregating neighbors' information via message passing; (2) A graph-level representation is formed by applying a readout function (e.g., graph pooling (Duvenaud et al. 2015)) on the node embedding; (3) The graph representation is fed to a classifier, which is trained with supervised labels and then applied for graph classification. However, a major issue of the conventional GNN-based pipeline for graph classification lies in that it conducts node embeddings without considering graph-level information. As a result, the readout function pools the diverse local node embeddings to form a global graph representation, which exists the *embedding misalignment* problem that can jeopardize the accuracy of graph-level task. For example, Fig. 1 (b) shows the label embeddings and node embeddings with the popular

GNN named GraphSAGE (Hamilton 2017) on the MUTAG dataset, where different colors represent different classes. According to the figure, the node embeddings are not in line with their corresponding label embeddings, where a large number of blue and red node embeddings are mixed up in the feature space. We call it the "embedding misalignment" phenomenon, where the diverse unaligned node embeddings generated by GNNs lead to less discriminative representation among graph classes. For further explanation, we show in Fig. 1(c) the statistics of correlation coefficients between node embeddings and the corresponding label embeddings for four widely used GNN backbones, i.e., GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), GIN (Xu et al. 2019), and GraphSAGE on the MUTAG dataset. It shows that there is a large ratio of node embeddings negatively correlated to the ground truth labels in all GNN backbones. To address this issue, we propose a label-attentive design for GNNs to integrate global graph information into node embedding to improve graph-level tasks. As shown in Fig. 1(c), the OURS approach yields a much higher proportion of positive correlations to the ground truth, which are more favorable for an ideal graph representation.

Specifically, this paper proposes a novel **L**abel **A**ttentive **D**istillation method named **LAD-GNN** for graph representation learning to overcome the embedding misalignment problem. The pipeline of the proposed LAD-GNN is illustrated in Fig. 2, which consists of a two-phrase training processes and an inference step described as follows. (1) *Label-attentive teacher training:* we propose an auxiliary neural network called *label-attentive encoder* that encodes the ground-truth into label embedding, which is attentively combined with the node embedding generated by the GNN backbone to form an ideal embedding. (2) *Distillation-based student learning:* we train a student GNN to generate class-friendly node embedding by distilling knowledge from the teacher. The intuition is that the teacher model trained with augmented labels can generate an informative feature map to provide effective supervision for the student model. We adopt a multi-task training paradigm to train the student GNN to minimize a classification loss and an auxiliary distill loss, which can inherit the class-specific knowledge from the teacher model and encourage the student to generate class-friendly node embedding to facilitate graph-level tasks. When model deployment, only the student model works without graph labels, ensure no information leakage.

We empirically evaluate our method in graph classification tasks on 10 benchmark datasets with 7 commonly used GNN backbones and execute performance comparisons with other 9 GNN training methods (such as manual/automated graph augmentation methods and graph distillation methods). The experimental results demonstrate that LAD-GNN significantly outperforms the original GNN backbones: it achieves up to 16.8% accuracy improvement on the IMDB-BINARY dataset with the GraphSAGE backbone. We also perform extensive experiments for parameter sensitivity and visualization for detailed analyses and asses.

Our major contributions are summarized as follows.

- We propose a novel label-attentive distillation method called LAD-GNN for graph representation learning. It introduces a teacher model trained with an auxiliary label encoder to generate ideal embedding, and proposes a distillation-like approach to train a student GNN with intermediate supervision to learn class-friendly node embedding. Generally, LAD-GNN is an enhanced GNN training approach that can be incorporated with arbitrary GNNs on graph-level tasks without significant increase of computational cost.
- We introduce a novel label-attentive encoding architecture design to encode the graph labels into the latent space fusing with nodes embedding, and propose an auxiliary distillation supervision method to solve the embedding misalignment problem, which can enhance the GNN backbone to capture informative class-specific information to facilitate graph-level task.
- We perform extensive experiments using 7 GNN backbones on 10 benchmark datasets to validate graph classification performances, and execute comparisons with the state-of-the-art GNN training methods. Experimental results justify the superiority and effectiveness of the proposed method.

Related Works

Graph Neural Networks

Graph Neural Networks (GNNs) have received tremendous attention due to their superiority in a wide variety of graph learning tasks (Park et al. 2020; Hong et al. 2021; Wang et al. 2022). The pioneering work of GNN was the Graph Convolutional Networks (GCN) (Kipf and Welling 2017), which for the first time introduced the spectral convolution to graph data and employed the Chebyshev polynomials to accelerate its training. Based on GCN, graph attention networks (Veličković et al. 2018) dynamically learn the weights (attention scores) on the edges when performing message passing and introduce the attention mechanism into neighborhood aggregation. Graph Isomorphism Network (GIN) (Xu et al. 2019) used the injective multiset function for neighbor aggregation, which had been shown to be as powerful as the 1-WL test in distinguishing graph structures. As the prevalence of Transformer framework in CV and NLP tasks, some studies (Rong et al. 2020a; Wu et al. 2022; Müller et al. 2023) applied Transformer for GNNs, which incorporated the correlations between nodes for neighborhood aggregation. The customizing GNNs for graph-level tasks mainly focus on refining graph readout strategies, which can be categorized into pooling methods (Bianchi, Grattarola, and Alippi 2020; Liu et al. 2022), attention mechanism methods (Nouranizadeh, Matinkia, and Rahmati 2021; Chen et al. 2023), and information theoretic-based methods (Gao et al. 2021; Han et al. 2022).

Label Enhancement Methods

Labels are commonly used as supervision in the computing loss function at the end of the output. There were plenty of works that used label-enhanced techniques to boost model training (Bengio, Weston, and Grangier 2010; Sun et al. 2017; Yang et al. 2021; Peng et al. 2022). The usage of label-enhanced GNNs can be categorized under label-enhanced node embedding and graph structure optimization. In label-enhanced node embedding (Wang 2021; Shi et al. 2020; Li et al. 2022a), the labels were encoded to concatenate with or sum up the node attributes to enhance feature representation. Within the context of label-enhanced graph structure optimization (Chen et al. 2019; Yang et al. 2021), labels played a crucial role in the process of refining the adjacency matrix to facilitate the unimpeded propagation of topological information among nodes that share a common label. While the existing label-enhancement GNNs mostly focused on improving node-level tasks, our work for the first time introduced a label-attentive approach to enhance GNN learning for graph-level tasks.

Knowledge Distillation

Our method shares some common principles with knowledge distillation (KD) (Hinton et al. 2015; Zhang et al. 2020b). Originally KD aims to reduce the model size when deployed on devices with limited computational resources. In KD, a large teacher model is trained first, and its predictions are used as soft labels to supervise the training of a small student model. There were a few works that adopted KD for GNNs (Guo et al. 2023). Yang et al. proposed a graph knowledge distillation framework, which can inject the knowledge of an arbitrarily learned GNN model into a well-designed student model (Yang, Liu, and Shi 2021). Jing et al. trained a multi-talented student GNN that amalgamates knowledge from a couple of teacher GNNs with heterogeneous architectures to handle distinct tasks (Jing et al. 2021). Different from the existing GNN knowledge distillation works, our proposed distillation-like method introduces a novel label-attentive encoder to generate ideal embedding, which is used as intermediate supervision to train the student GNN to generate task-friendly graph presentations.

Methodology

Problem Formulation

Given a graph dataset with known labels $\mathcal{D} = (\mathcal{G}, \mathcal{Y}) = \{(G^i, y^i)\}_{i=1}^N$, where $G^i \in \mathcal{G}$ is the i-th graph in the dataset. Denoted by $G^i = (\mathbf{A}^i, \mathbf{X}^i)$, where $\mathbf{A}^i \in \mathbb{R}^{n_i \times n_i}$ is the adjacency matrix describing the link relationship of the nodes set $\mathcal{V}^i$; $\mathbf{X}^i \in \mathbb{R}^{n_i \times d}$ is the node's feature vector; and n_i, d are the nodes number and feature dimension of the i-th graph G^i respectively. The goal of graph representation learning is to learn a low-dimensional embedding for each graph to predict the label. Let $\mathcal{Y} = \{y^i\}_{i=1}^N$ be the label set, where y^i denotes the y^i-th class label and N is the dataset scale. Without loss of generality, we omit the index i thereafter to simplify the description.

Overall Framework

The overall framework of LAD-GNN is shown in Fig. 2. It contains a two-phase process that trains a teacher model and a student model iteratively. Firstly, it proposes a *label-attentive teacher training* method to train a teacher GNN to generate label-augmented node embedding. Specifically, it introduces an auxiliary neural network called *label-attentive encoder* that encodes the ground-truth into label embedding and then combines the label embedding with the nodes embedding generated by the GNN backbone using an attention mechanism to form an ideal embedding, which is fed into the readout function and classification head to predict the graph label. The label-attentive encoder is jointly trained with the GNN backbone to minimize the classification loss. Secondly, it applies a *distillation-based student learning* method to train a student GNN model. In this phase, the ideal embedding from the teacher model performs as intermediate supervision to distill the student. As shown in the figure, the student model shares a classification head with the teacher model, and it trains the student GNN to minimize both the classification loss and the distillation loss, which can inherit the knowledge from the teacher model to generate class-friendly nodes embedding to facilitate graph-level task.

Before getting into the details of label-attentive distillation, we first introduce the following general components for GNN-based graph classification.

GNN Backbone. It is used to extract the node-level features $\mathbf{H} = \{\mathbf{H}_v | v \in \mathcal{V}\}$, where $\mathbf{H}_v \in \mathbb{R}^{d'}$ denotes node v's embedding which synthesizes the topology and initial node attributes. A GNN layer can be simplify formalized as:

$$\mathbf{H}_v^{(l+1)} = \text{UPT}(\mathbf{H}_v^{(l)}, \text{AGG}(\{\mathbf{H}_u^{(l)} | u \in \mathcal{N}_v\})), \forall v \in \mathcal{V} \quad (1)$$

where $\mathbf{H}_v^{(l+1)}$ denotes the v-th latent node representation of the $(l+1)$-th layer; $\mathbf{H}^{(0)} = \mathbf{X}$ is initialized by the node feature matrix; $\mathcal{N}_v$ denotes the neighbors of node v; AGG and UPT are the aggregation and update function respectively. The nodes embedding $\mathbf{H}$ is aggregated by the GNN backbone as:

$$\mathbf{H} = f(\mathbf{A}, \mathbf{X}; \theta_f). \quad (2)$$

Readout Function. It is used to form a graph-level representation from the node embedding, which can be regarded as a graph pooling operator:

$$\mathbf{Z}_G = \text{POOL}(\{\mathbf{H}_v | v \in \mathcal{V}\}), \quad (3)$$

where $\mathbf{Z}_G \in \mathbb{R}^{d'}$ denotes the representation of graph G, and d' is the feature dimension. The average pooling (Duvenaud et al. 2015) and max pooling (Xu et al. 2019) are the common pooling operations, which treat all nodes equally. Considering the importance of different nodes, there are also some customized pooling operations, such as subgraph selector (Wu et al. 2020; Li et al. 2020) and attention mechanism (Lee, Lee, and Kang 2019).

Classification Head. It is used to predict graph labels based on the pooled graph representations: $\widehat{\mathcal{Y}} = g(\mathbf{Z}; \phi_g)$. where $\widehat{\mathcal{Y}} \in \mathbb{R}^{N \times c}$ is the output one-hot prediction with c classes, ϕ_g is the classification head parameters and the Multi-Layer Perceptron (MLP) is commonly employed.

Label-Attentive Teacher Training

As illustrated in Fig. 2, the teacher model consists of a GNN backbone and a label-attentive encoder. Given a sample (G, y), we feed the graph G to the GNN backbone to generate node representations. To align the node embedding with the global label, we introduce a label-attentive encoder architecture as follows.

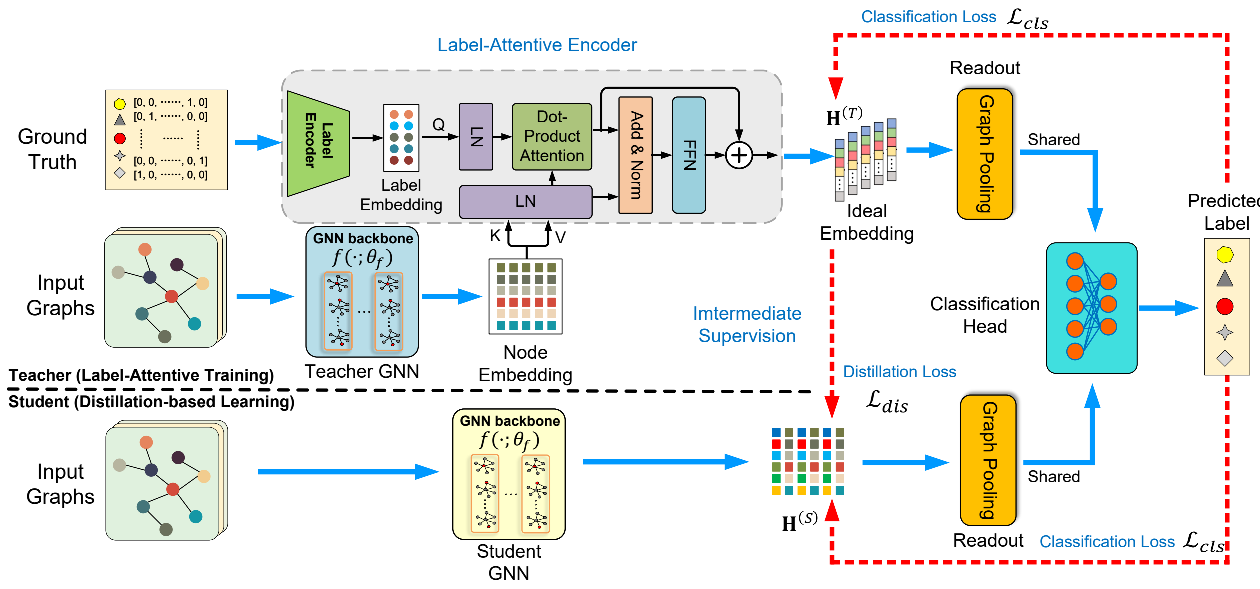


Figure 2: The overall framework of LAD-GNN. The pipeline consists of three steps: (1) Label-attentive training using known labels to train a teacher GNN; (2) Distillation-based learning to train a student GNN; (3) Inference using the student GNN for graph classification (without label input).

Label-Attentive Encoder. The label-attentive encoder consists of a label encoder and several layers of attention mechanisms and operations to form an ideal embedding.

Taking the ground-truth labels y_G as input, the label encoder $h(\cdot)$ encodes the input to a latent embedding. In practice, we propose to use a Multi-Layer Perceptron (MLP):

$$\mathbf{H}_l = h(y_G | G \in \mathcal{G}), \tag{4}$$

where $\mathbf{H}_l$ is the label embedding of graph G.

The attention mechanisms work similarly to the popular Transformer (Vaswani et al. 2017) architecture. The label embedding and the node embedding from the teacher GNN go through a *layernorm* (LN) to alleviate covariate shift, and then are fed into a *scaled dot-product attention* layer for feature fusion. The *add & normalization* operation is used to alleviate internal covariate shift and enhance the independence between features. The *feed-forward network* (FFN) processes the output of the attention layer to form a higher-level latent representation. This architecture enables the model to capture intricate relationships between label and node embeddings, in the meanwhile enhancing model expressiveness by adding nonlinearity. The overall operations can be formulated as:

$$\begin{aligned} \mathbf{H}_v^{'} =& \text{Attention}(\mathbf{H}_v, \mathbf{H}_l) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}} \cdot \tau)V, \\ \mathbf{H}_v^{(T)} &= \text{FFN}(\text{LN}(\mathbf{H}_v^{'} + \mathbf{H}_v)) + \mathbf{H}_v^{'}, \end{aligned} \tag{5}$$

where $\mathbf{H}_l$ is the label embedding generated by the label encoder; $\mathbf{H}_v$ is the node embedding generated by the GNN backbone; $Q = \mathbf{H}_l\mathbf{W}^Q$ is the label embedding projection; $K = \mathbf{H}_v\mathbf{W}^K$ and $V = \mathbf{H}_v\mathbf{W}^V$ are the node embeddings projections; and τ is the attention temperature coefficient (Zhang et al. 2021).

Teacher Model Training. The teacher model employs the GNN backbone and the label-attentive encoder to generate ideal embedding, which flows into the readout function and the shared classification head to output the prediction label $\widehat{\mathcal{Y}}$. The objective function of teacher model training is the cross entropy for graph classification:

$$\mathcal{L}_{cls} = \frac{1}{N}\sum_{i=1}^{N} -(y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})). \tag{6}$$

Distillation-based Student Learning

As shown in Fig. 2, the student model learns from the teacher model with knowledge distillation, and it shares the classification head with the teacher model. During inference, a graph G is input into the student GNN to generate node representations, which are fed into the readout function and the classification head to predict its corresponding label $\hat{y}$.

Specifically, after the teacher model is trained to converge, we use the teacher model's output, i.e., the ideal embedding, as intermediate supervision to guide the student's GNN backbone to learn enhanced node embedding with a distillation-like method. We use $\mathbf{H}^{(T)}, \mathbf{H}^{(S)}$ to denote the node embeddings extracted from the teacher GNN and the student GNN respectively. To distill the class-dependent knowledge from the teacher to the student model, we aim to minimize the following distillation loss representing as a Mean Square Error (MSE):

$$\mathcal{L}_{dis} = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{H}_i^{(T)}, \mathbf{H}_i^{(S)}||_2^2. \tag{7}$$

The training of student model is a multi-task paradigm that urges the student GNN to generate ideal embedding and

Algorithm 1: LAD-GNN Algorithm

Input: graph datasets $\mathcal{D}$, labels $\mathcal{Y}$, hyper parameter λ, τ.
Output: optimized model parameters, prediction of the graph labels $\widehat{\mathcal{Y}}$.

```
1: Randomly initializes the model parameters and splits
   the dataset to train/val/test.
2: for Teacher training epochs do
3:     for batchs do
4:         Label Encoding: $\mathbf{H}_l = h(y_G | G \in \mathcal{G})$;
5:         Neighbor Aggregation: $\mathbf{H}^{(T)} = f_T(\mathbf{A}, \mathbf{X}; \theta_{f_T})$;
6:         $\mathbf{H}^{(T)} = \text{Attention}(\mathbf{H}^{(T)}, \mathbf{H}_l)$ in Eq. 5;
7:         $\widehat{\mathcal{Y}} = g(\text{POOL}(\mathbf{H}_v^{(T)} | v \in \mathcal{V}); \phi_g)$;
8:         Cross-Entropy Loss Computation in Eq. 6;
9:         Save the optimized model parameters.
10:    end for
11: end for
12: for Student training epochs do
13:    for batchs do
14:        Neighbor Aggregation: $\mathbf{H}^{(S)} = f_S(\mathbf{A}, \mathbf{X}; \theta_{f_S})$;
15:        Load teacher model for ideal embedding: $\mathbf{H}^{(T)}$;
16:        $\mathcal{L}_{dis} = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{H}_i^{(T)}, \mathbf{H}_i^{(S)}||_2^2$;
17:        $\widehat{\mathcal{Y}} = g(\text{POOL}(\mathbf{H}_v^{(S)} | v \in \mathcal{V}), \phi_g)$;
18:        $\mathcal{L} = \mathcal{L}_{cls} + \lambda \cdot \mathcal{L}_{dis}$.
19:    end for
20: end for
```

pushes the classifier to optimize graph-level tasks. Therefore the student model is trained with the same set of training samples as the teacher's to minimize the following comprehensive objective function:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \cdot \mathcal{L}_{dis}, \tag{8}$$

where $\mathcal{L}_{dis}$ is the classification loss defined in Eq. 6; $\mathcal{L}_{dis}$ is the distillation loss defined in Eq. 7, and λ is the hyperparameter for balancing the classification and distillation loss.

Complexity Analysis

The proposed LAD-GNN algorithm is summarized in Algorithm 1. LAD-GNN consists of a two-phase training process for the teacher and student models. The teacher jointly trains the GNN backbone with the label-attentive encoder by a classification loss $\mathcal{L}_{cls}$. The student shares a similar model architecture with the teacher except for the label-attentive encoder. LAD-GNN conducts a distillation-like training for the student with an objective function combining the distillation loss $\mathcal{L}_{dis}$ and the classification loss $\mathcal{L}_{cls}$. Note that compared with the conventional GNN learning methods, the time complexity of LAD-GNN mainly lies in the distillation loss computation in the student training phrase, and the time complexity for model inference is the same as that of the GNN backbones. As an example, the time complexity of a SOTA GNN backbone named MEWISPool (Nouranizadeh et al. 2021) is $\mathcal{O}(|\mathcal{V}|(kd + |\mathcal{E}|)) + \mathcal{O}(|\mathcal{V}|^3))$, where k is the maximum degree of the graph and d is the dimension of nodes features. By applying the proposed LAD-GNN training approach on the MEWISPool backbone, the time complexity of training the teacher model consumes $\mathcal{O}(|\mathcal{V}|(kd + |\mathcal{E}|)) + \mathcal{O}(|\mathcal{V}|^3)) + \mathcal{O}(Ld)$, where L is the number of layers of the label-attentive encoder; the time complexity of student training consumes $\mathcal{O}(|\mathcal{V}|(kd + |\mathcal{E}|)) + \mathcal{O}(|\mathcal{V}|^3)) + \mathcal{O}(Ld) + \mathcal{O}(|\mathcal{V}|^2)$. In summary, the increase of computational complexity lies in the term $\mathcal{O}(Ld)$, which is neglectable since $L \ll |\mathcal{V}|$ and $d \ll |\mathcal{V}|$ holds in practice.

Experiments

In this section, we assess the performance of LAD-GNN in comparison with the 7 GNN backbones based on 10 open graph datasets, and then we evaluate LAD-GNN on graph classification tasks compared with 9 other GNN training strategies. We further discuss the sensitivity of hyperparameters and visualize the graph representation of different approaches. We implement LAD-GNN in PyTorch v1.12, and the experiments are conducted on a GPU-equipped PC with an NVIDIA GeForce RTX 3090Ti.

We assess the graph classification performances on 10 open datasets, which include the Chemical Molecules datasets MUTAG, PTC (Debnath et al. 1991), and NCI1 (Wale, Watson, and Karypis 2008), the Bioinformatics graph datasets PROTEINS and ENZYMES (Borgwardt et al. 2005), and the Social Network datasets COLLAB, IMDA-BINARY, IMDB-MULTI, REDDIT-BINARY, and REDDIT-MULTI-5K (Yanardag and Vishwanathan 2015). These graph datasets contain non-attribute graphs, and we use the node's degree as the initial node feature following the literature (Duong et al. 2019). For the detailed statistics of these datasets please refer to the supplementary materials.

Performance Comparison with GNN Backbones

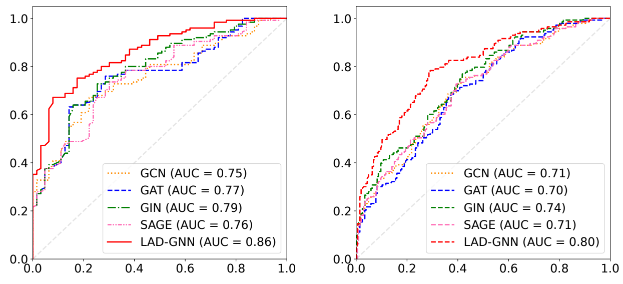
We conduct experiments based on 7 GNN backbones, which include 4 commonly used message-passing protocol GNNs (i.e., **GCN** (Kipf and Welling 2017), **GAT** (Veličković et al. 2018), **GraphSAGE** (Hamilton 2017) and **GIN** (Xu et al. 2019)) and 3 state-of-the-arts graph classification models (i.e., **DGCNN** (Zhang et al. 2018), **SAGPool** (Lee, Lee, and Kang 2019), and **MEWISPool** (Nouranizadeh et al. 2021)). LAD-GNN is integrated with each GNN backbone to boost graph classification performance, and the results are reported in Table 1, where each row reports the classification accuracy of the original GNN backbone and the results after applying LAD-GNN, and each column reports the performance of one dataset in the table. For GCN, GAT, and GraphSAGE, since the original models are proposed for the node classification task, we reproduce these models based on the PyG (PyTorch Geometric) library (Fey and Lenssen 2019) and applied them for graph classification, where the node aggregation layer numbers of them are all set to 2. For GIN, DGCNN, SAGPool, and MEWISPool, whose models are proposed for the graph classification task in the original papers, we run their open-source codes on the 10 datasets. To remove the unwanted bias towards the training data, for all experiments on the datasets, we evaluate the model performance with a ten-fold cross-validation setting. For a fair comparison, all datasets are randomly split to train/validation/test sets following the 0.8/0.1/0.1 protocol in each

Models	MUTAG	COLLAB	ENZYMES	IMDB-B	IMDB-M	NCI1	PROTEINS	PTC	REDDIT-B	REDDIT-M
GCN	0.747±0.08	0.810±0.01	0.765±0.11	0.573±0.04	0.402±0.05	0.490±0.03	0.643±0.05	0.662±0.06	0.813±0.06	0.425±0.02
+LAD-GNN	**0.837**±0.07	**0.841**±0.06	**0.772**±0.12	**0.633**±0.04	**0.406**±0.03	**0.584**±0.06	**0.716**±0.03	**0.740**±0.08	**0.825**±0.03	**0.491**±0.07
GAT	0.737±0.12	0.693±0.03	0.778±0.08	0.546±0.07	0.377±0.05	0.727±0.04	0.692±0.05	0.691±0.08	0.727±0.02	0.487±0.01
+LAD-GNN	**0.805**±0.05	**0.712**±0.02	**0.813**±0.08	**0.576**±0.04	**0.414**±0.04	**0.731**±0.04	**0.703**±0.04	**0.706**±0.09	**0.745**±0.04	**0.488**±0.02
GraphSAGE	0.816±0.09	0.791±0.05	0.742±0.09	0.589±0.06	0.420±0.05	0.699±0.04	0.706±0.04	0.709±0.07	0.831±0.05	0.466±0.02
+LAD-GNN	**0.863**±0.10	**0.793**±0.05	**0.792**±0.10	**0.755**±0.08	**0.420**±0.06	**0.867**±0.07	**0.733**±0.05	**0.814**±0.10	**0.925**±0.02	**0.517**±0.04
GIN	0.842±0.06	0.659±0.03	0.733±0.18	0.701±0.03	0.434±0.03	0.729±0.04	0.721±0.02	0.620±0.06	0.717±0.02	0.410±0.02
+LAD-GNN	**0.854**±0.05	**0.678**±0.04	**0.765**±0.16	**0.714**±0.03	**0.441**±0.02	**0.747**±0.03	**0.730**±0.03	**0.657**±0.05	**0.725**±0.02	**0.450**±0.03
DGCNN	0.913±0.01	0.749±0.01	0.501±0.02	0.728±0.02	0.464±0.01	0.701±0.01	0.713±0.02	0.617±0.03	0.743±0.01	0.457±0.03
+LAD-GNN	**0.944**±0.02	**0.751**±0.02	**0.516**±0.01	**0.748**±0.02	**0.482**±0.01	**0.705**±0.02	**0.721**±0.03	**0.649**±0.02	**0.774**±0.02	**0.529**±0.04
SAGPool	0.763±0.08	0.725±0.02	0.426±0.06	0.589±0.05	0.419±0.06	0.694±0.03	0.588±0.05	0.614±0.06	0.837±0.02	0.484±0.02
+LAD-GNN	**0.784**±0.08	**0.733**±0.02	**0.428**±0.07	**0.593**±0.04	**0.434**±0.04	**0.722**±0.02	**0.601**±0.03	**0.649**±0.07	**0.858**±0.03	**0.508**±0.03
MEWISPool	0.926±0.03	0.745±0.01	0.383±0.01	0.772±0.01	0.474±0.04	0.711±0.05	0.699±0.03	0.714±0.02	-	-
+LAD-GNN	**0.947**±0.01	**0.771**±0.02	**0.433**±0.02	**0.779**±0.03	**0.511**±0.03	**0.723**±0.02	**0.746**±0.01	**0.746**±0.03	-	-

Table 1: The results on 10 graph classification datasets compared with 7 SOTA GNN backbones. In each row, the higher reports the original GNN model's performance, and the lower (shadows in gray) is the results with the proposed LAD-GNN.

Methods	PROTEINS	IMDB-BINARY	COLLAB	MUTAG	NCI109	NCI1	PTC
DropEdge	0.707±0.002	0.733±0.012	0.812±0.003	0.779±0.005	0.762±0.007	0.780±0.002	-
M-Mixup	0.706±0.003	0.736±0.004	0.811±0.005	0.798±0.015	0.788±0.005	0.803±0.003	-
$\mathcal{G}$-Mixup	0.715±0.006	0.748±0.004	0.811±0.009	0.805±0.002	0.654±0.043	0.686±0.037	-
JOAOv2	0.700±0.003	0.707±0.008	0.688±0.003	0.775±0.016	0.675±0.003	0.670±0.006	-
AD-GCL	0.699±0.008	0.712±0.008	0.670±0.008	0.837±0.010	0.634±0.003	0.641±0.004	-
AutoGCL	0.684±0.008	0.707±0.007	0.745±0.002	0.783±0.022	0.705±0.003	0.737±0.002	-
KD	0.763±0.035	0.808±0.026	0.812±0.017	0.878±0.121	-	-	0.756±0.053
GFKD	0.633±0.077	0.623±0.052	0.633±0.023	0.677±0.129	-	-	0.625±0.059
DFAD-GNN	0.690±0.061	0.675±0.049	0.689±0.011	0.765±0.073	-	-	0.669±0.037
IGSD	-	0.744±0.060	0.704±0.110	**0.902**±0.070	-	0.754±0.030	0.614± 0.170
LAD-GNN	**0.765**±0.056	**0.811**±0.042	**0.827**±0.054	0.899±0.113	**0.882**±0.065	**0.896**±0.059	**0.791**±0.062

Table 2: Comparison of LAD-GNN with 9 other GNN training methods on the graph classification task. The first 3 rows belong to manual graph augmentation methods, the second 3 rows are graph auto-augmentation methods, and the third 4 rows are graph distillation methods.



(a) ROC curves on MUTAG (b) ROC curves on PTC

Figure 3: The ROC curves on MUTAG and PTC datasets of GCN, GAT, GIN, SAGE, and the proposed LAD-GNN.

model. We report the average and standard deviation of test accuracy across the ten folds within the cross-validation.

As shown in Table 1, the proposed LAD-GNN which shadows in gray achieves a clear performance improvement compared with the original models in the accuracy of the graph classification task: it outperforms the GNN backbones and gains 3.3% average improvement in the absolute accuracy. In particular, LAD-GNN achieves 16.6% and 16.8% improvements on the IMDB-BINARY and NCI1 datasets respectively with GraphSAGE as the backbone. In summary, the experimental results show that LAD-GNN effectively improves the graph-level performance compared with the state-of-the-art GNN models across a range of datasets.

In addition, we plot the ROC curves of LAD-GNN and other backbones (GCN, GAT, GIN, and GraphSAGE) on the MUTAG and PTC datasets, and calculate the AUC values (area under the ROC curve) of each model for performance comparison. As shown in Fig. 3, the ROC curves of our model (the red line) are generally above all backbones. Furthermore, the AUC values of LAD-GNN are also greater than that of the comparison models, which proves the superiority of our proposed method.

Performance Comparison with other GNN Training Strategies

In order to assess the effectiveness of the proposed training strategy, we compare LAD-GNN with 9 other GNN train-

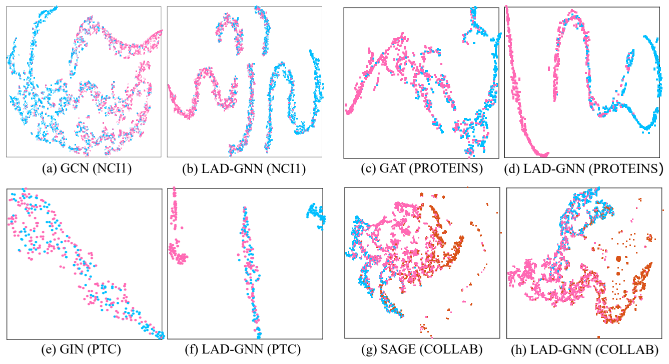


Figure 4: The t-SNE visualization of graph representations for LAD-GNN and the original GNN backbones.

ing methods for the graph classification task. The compared methods include the *manual graph augmentation methods* (i.e., **DropEdge** (Rong et al. 2020b), **M-Mixup** (Wang et al. 2021), **$\mathcal{G}$-Mixup** (Han et al. 2022)), the *graph auto-augmentation methods* (i.e., **JOAOv2** (You et al. 2021), **AD-GCL** (Suresh et al. 2021), **AutoGCL** (Yin et al. 2022)), and the *graph distillation methods* (i.e., **KD**, **GFKD** (Deng and Zhang 2021), **DFAD-GNN** (Zhuang et al. 2022), and **IGSD** (Zhang et al. 2020a)). The results are reported in Table 2, where the results of the graph augmentation methods (the upper 6 rows) follow the literature of AutoGCL (Yin et al. 2022), the three distillation methods (i.e. KD, GFKD, and DFAD-GNN) follow the DFAD-GNN (Zhuang et al. 2022), and IGSD follows the literature (Zhang et al. 2020a). It is shown that our LAD-GNN outperforms the manual/auto graph augmentation and distillation methods, achieving the best performance on all datasets except MUTAG, where it is outperformed by the self-distillation method IGSD (though the difference is small). In summary, LAD-GNN achieves a 5.6% accuracy improvement on NCI1 dataset compared with the second-place training method, and outperforms other methods except for a bit worse on MUTAG.

Hyperparameter Analysis

We further discuss the hyper-parameter sensitiveness of λ in Eq. 8 and τ in Eq 5. We tune the values of λ from 0.001 to 1000 and τ from 0.1 to 1.0, and test the graph classification performance on 4 datasets (i.e., PTC, NCI109, PROTEINS, and IMDB-BINARY). The results are presented in Fig 5. It indicates that different values of λ may result in the optimal convergence values for the parameters, and the best λ value for a given dataset can be determined using the validation set. τ is less sensitive with a low impact on the graph classification accuracy, and to ensure uniformity of hyperparameters across different datasets, we carefully considered and experimented with various values and ultimately selected $\tau = 0.1$ as the standard value for our experiments.

Visualization

In order to intuitively understand the quality of global graph representation learned by different methods, we visualize the graph embedding of different GNN backbones and the

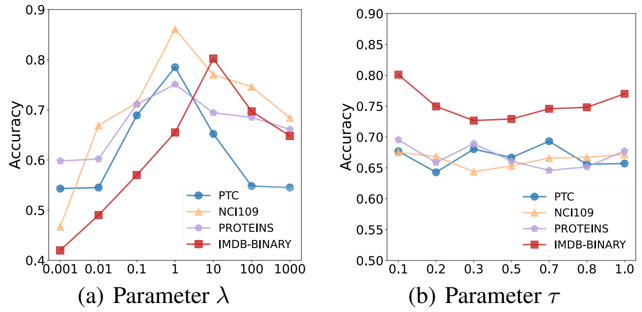


Figure 5: Hyperparameters sensitiveness of (a) λ and (b) τ on the 4 datasets.

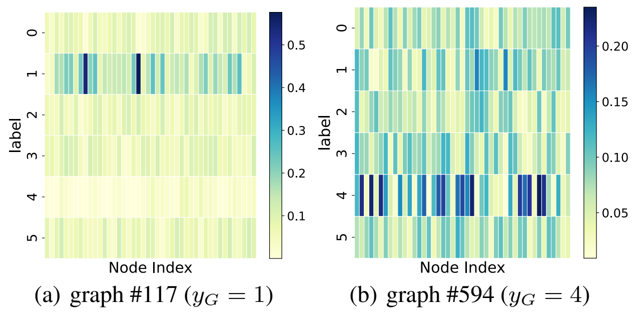


Figure 6: Visualization of attention scores on ENZYMES.

results are shown in Fig 4. The graph embedding is extracted by the readout function which is projected into a two-dimensional space with t-SNE for visualization. Compared with the original backbones, LAD-GNN gathers graphs in the same class more closely and provides more obvious boundaries between graphs in different classes.

We also visualize the label attention score heatmaps for two random graphs on ENZYMES dataset in Fig 6. In Fig. 6 (a), the attention scores of nodes more focus on label 1 (the ground truth). A similar result can be found in Fig. 6 (b), which shows that the node embeddings align with the ground truth of the proposed method.

Conclusion

In this paper, we focused on the performance issue of Graph Neural Networks (GNNs) in graph-level classification tasks. We found that conventional GNNs' node-level information aggregation approach forms misaligned embeddings that can jeopardize graph-level tasks. To address this issue, we propose a novel label attentive distillation method called LAD-GNN for graph classification. LAD-GNN introduces a teacher model with a label-attentive encoder architecture to encode the ground-truth labels into the latent space fusing with nodes embedding to form ideal embedding, and encourage the student GNN to learn class-friendly node embeddings that facilitate graph-level tasks with a self-distilled intermediate supervision method. Extensive experimental results justified the superiority and effectiveness of the proposed method.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 61972196, 61832008, 61832005), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Sino-German Institutes of Social Computing.

References

Bengio, S.; Weston, J.; and Grangier, D. 2010. Label embedding trees for large multi-class tasks. *Advances in Neural Information Processing Systems*, 23.

Bianchi, F. M.; Grattarola, D.; and Alippi, C. 2020. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, 874–883. PMLR.

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.

Chen, D.; Liu, X.; Lin, Y.; Li, P.; Zhou, J.; Su, Q.; and Sun, X. 2019. Highwaygraph: Modelling long-distance node relations for improving general graph neural network. *arXiv preprint arXiv:1911.03904*.

Chen, F.; Wang, J.; Wei, Y.; Zheng, H.-T.; and Shao, J. 2022. Breaking Isolation: Multimodal Graph Fusion for Multimedia Recommendation by Edge-wise Modulation. In *Proceedings of the 30th ACM International Conference on Multimedia*, 385–394.

Chen, J.; Xiong, H.; Zheng, H.; Zhang, D.; Zhang, J.; Jia, M.; and Liu, Y. 2023. EGC2: Enhanced graph classification with easy graph compression. *Information Sciences*, 629: 376–397.

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.

Deng, X.; and Zhang, Z. 2021. Graph-free knowledge distillation for graph neural networks. *arXiv preprint arXiv:2105.07519*.

Duong, C. T.; Hoang, T. D.; Dang, H. T. H.; Nguyen, Q. V. H.; and Aberer, K. 2019. On node features for graph neural networks. *arXiv preprint arXiv:1911.08795*.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Gao, X.; Dai, W.; Li, C.; Xiong, H.; and Frossard, P. 2021. iPool–Information-Based Pooling in Hierarchical Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Guo, Z.; Zhang, C.; Fan, Y.; Tian, Y.; Zhang, C.; and Chawla, N. V. 2023. Boosting graph neural networks via adaptive knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 7793–7801.

Hamilton, Y. Z. . L. J., W. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Han, X.; Jiang, Z.; Liu, N.; and Hu, X. 2022. G-Mixup: Graph Data Augmentation for Graph Classification. *arXiv preprint arXiv:2202.07179*.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Hong, X.; Zhang, T.; Cui, Z.; Huang, Y.; Shen, P.; Li, S.; and Yang, J. 2021. Graph game embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7711–7720.

Jing, Y.; Yang, Y.; Wang, X.; Song, M.; and Tao, D. 2021. Amalgamating knowledge from heterogeneous graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15709–15718.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. In *International conference on machine learning*, 3734–3743. PMLR.

Li, M.; Chen, S.; Zhang, Y.; and Tsang, I. 2020. Graph cross networks with vertex infomax pooling. *Advances in Neural Information Processing Systems*, 33: 14093–14105.

Li, W.; Chen, J.; Gao, P.; and Huang, Z. 2022a. Label enhancement with label-specific feature learning. *International Journal of Machine Learning and Cybernetics*, 13(10): 2857–2867.

Li, Z.; Wu, Q.; Nie, F.; and Yan, J. 2022b. GraphDE: A Generative Framework for Debiased Learning and Out-of-Distribution Detection on Graphs. In *Advances in Neural Information Processing Systems*.

Liu, C.; Zhan, Y.; Wu, J.; Li, C.; Du, B.; Hu, W.; Liu, T.; and Tao, D. 2022. Graph pooling for graph neural networks: Progress, challenges, and opportunities. *arXiv preprint arXiv:2204.07321*.

Müller, L.; Galkin, M.; Morris, C.; and Rampášek, L. 2023. Attending to graph transformers. *arXiv preprint arXiv:2302.04181*.

Nouranizadeh, A.; Matinkia, M.; and Rahmati, M. 2021. Topology-Aware Graph Signal Sampling for Pooling in Graph Neural Networks. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, 1–7. IEEE.

Nouranizadeh, A.; Matinkia, M.; Rahmati, M.; and Safabakhsh, R. 2021. Maximum Entropy Weighted Independent Set Pooling for Graph Neural Networks. *arXiv preprint arXiv:2107.01410*.

Park, T.; Efros, A. A.; Zhang, R.; and Zhu, J.-Y. 2020. Contrastive learning for unpaired image-to-image translation. In *European conference on computer vision*, 319–345. Springer.

Peng, J.; Wang, H.; Yue, S.; and Zhang, Z. 2022. Context-aware co-supervision for accurate object detection. *Pattern Recognition*, 121: 108199.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020a. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33: 12559–12571.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020b. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; and Sun, Y. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Sun, X.; Wei, B.; Ren, X.; and Ma, S. 2017. Label embedding network: Learning label representation for soft training of deep networks. *arXiv preprint arXiv:1710.10393*.

Suresh, S.; Li, P.; Hao, C.; and Neville, J. 2021. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34: 15920–15933.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

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.

Wang, Y. 2021. Bag of tricks of semi-supervised classification with graph neural networks. *arXiv preprint arXiv:2103.13355*.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021. Mixup for node and graph classification. In *Proceedings of the Web Conference*, 3663–3674.

Wang, Z.; Liu, M.; Luo, Y.; Xu, Z.; Xie, Y.; Wang, L.; Cai, L.; Qi, Q.; Yuan, Z.; Yang, T.; et al. 2022. Advanced graph and sequence neural networks for molecular property prediction and drug discovery. *Bioinformatics*, 38(9): 2579–2586.

Wu, Q.; Zhao, W.; Li, Z.; Wipf, D.; and Yan, J. 2022. NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification. In *Advances in Neural Information Processing Systems*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

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.

Yang, C.; Liu, J.; and Shi, C. 2021. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the Web Conference*, 1227–1237.

Yang, H.; Yan, X.; Dai, X.; Chen, Y.; and Cheng, J. 2021. Self-enhanced gnn: Improving graph neural networks using model outputs. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Yin, Y.; Wang, Q.; Huang, S.; Xiong, H.; and Zhang, X. 2022. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8892–8900.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph contrastive learning automated. In *International Conference on Machine Learning*, 12121–12132. PMLR.

Zhang, H.; Lin, S.; Liu, W.; Zhou, P.; Tang, J.; Liang, X.; and Xing, E. P. 2020a. Iterative graph self-distillation. *arXiv preprint arXiv:2010.12609*.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*.

Zhang, S.; Zhang, X.; Bao, H.; and Wei, F. 2021. Attention temperature matters in abstractive summarization distillation. *arXiv preprint arXiv:2106.03441*.

Zhang, W.; Miao, X.; Shao, Y.; Jiang, J.; Chen, L.; Ruas, O.; and Cui, B. 2020b. Reliable data distillation on graph convolutional network. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 1399–1414.

Zhuang, Y.; Lyu, L.; Shi, C.; Yang, C.; and Sun, L. 2022. Data-Free Adversarial Knowledge Distillation for Graph Neural Networks. *arXiv preprint arXiv:2205.03811*.