applied sciences

MDPI

Article

A Lightweight Method for Graph Neural Networks Based on Knowledge Distillation and Graph Contrastive Learning

Yong Wang and Shuqun Yang *

School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; wangyong74@outlook.com
* Correspondence: shqyang@sues.edu.cn

Abstract: Graph neural networks (GNNs) are crucial tools for processing non-Euclidean data. However, due to scalability issues caused by the dependency and topology of graph data, deploying GNNs in practical applications is challenging. Some methods aim to address this issue by transferring GNN knowledge to MLPs through knowledge distillation. However, distilled MLPs cannot directly capture graph structure information and rely only on node features, resulting in poor performance and sensitivity to noise. To solve this problem, we propose a lightweight optimization method for GNNs that combines graph contrastive learning and variable-temperature knowledge distillation. First, we use graph contrastive learning to capture graph structural representations, enriching the input information for the MLP. Then, we transfer GNN knowledge to the MLP using variable temperature knowledge distillation. Additionally, we enhance both node content and structural features before inputting them into the MLP, thus improving its performance and stability. Extensive experiments on seven datasets show that the proposed KDGCL model outperforms baseline models in both transductive and inductive settings; in particular, the KDGCL model achieves an average improvement of 1.63% in transductive settings and 0.8% in inductive settings when compared to baseline models. Furthermore, KDGCL maintains parameter efficiency and inference speed, making it competitive in terms of performance.

Keywords: graph neural network; lightweight technology; knowledge distillation; graph contrastive learning

Citation: Wang, Y.; Yang, S. A Lightweight Method for Graph Neural Networks Based on Knowledge Distillation and Graph Contrastive Learning. *Appl. Sci.* **2024**, *14*, 4805. https://doi.org/10.3390/app14114805

Academic Editor: Keun Ho Ryu

Received: 19 April 2024
Revised: 27 May 2024
Accepted: 30 May 2024
Published: 2 June 2024

1. Introduction

Graph neural networks (GNNs) have shown great potential in processing non-Euclidean data and have achieved good performance in various graph machine learning tasks [1]. The success of modern graph neural networks relies on the message passing scheme. This scheme's core principle is to center on each node, aggregate information from its neighboring nodes, and combine it with the node's own representation to update its representation [1]. Due to the node dependency of graph data and the inherent limitations of the message passing scheme [2–4], deploying graph neural networks on end-devices with restricted computational and storage resources poses a significant challenge. This difficulty hinders the application of graph neural networks in real-world scenarios for rapid and high-performance inferencing. To meet the demands for quick inference in real-world scenarios, the multilayer perceptron (MLP) remains the optimal choice, although it performs poorly on non-Euclidean data [3].

Graph neural networks have good performance advantages on graph-structured data. Although the MLP performs poorly on graph data, its computational efficiency is much higher than that of GNNs due to its simple structure, ease of computing node outputs and updating parameters, and the highly parallelizable nature of fully connected layer computations. To fully leverage the performance advantages of GNNs and the efficiency advantages of MLPs, some studies have combined both into a single framework [3–5]. A

mainstream method for combining GNN and MLP models is knowledge distillation, which uses the soft label (i.e., probability distribution) outputs of the teacher model (in this case, the GNN) as supervision signals to train the student model (the MLP) and complete the transfer of knowledge from the teacher model to the student model [3]. The distilled student model MLP is then deployed for fast inference. The MLP model only inputs the content features of nodes and does not include graph structure features. Through knowledge distillation, the student model MLP can imitate the ability of the teacher model GNN, while its inference speed is much faster than that of the GNN. However, this approach has the following two drawbacks: first, the student model MLP is unable to directly capture graph structural information and can only utilize node content features as input; second, the accuracy of the student model MLP is far inferior to that of the teacher model GNN, and it is more sensitive to noise and interference in graph data.

There are two primary methods for lightweight graph neural networks through knowledge distillation. In one approach, both the teacher and student models are GNNs [6–9], named GNN-to-GNN. It involves transferring knowledge from a large, complex teacher model to a smaller, more efficient student model through knowledge distillation. GNN-to-GNN distillation reduces the parameter scale of the GNN model and decreases inference time by reducing the number of layers and neurons in the student model; however, it does not fundamentally address the issue of node dependency during the inference process. Consequently, it fails to effectively improve inference efficiency on large-scale graph datasets. The other approach is heterogeneous knowledge distillation, in which the teacher model is a GNN and the student model is an MLP or another model [3,4,10–12]. The output probability distribution of the GNN guides the training of the MLP, allowing the MLP to imitate the GNN, thus effectively transferring its knowledge.

Graph contrastive learning is an unsupervised learning method that aims to learn the structural features of graph data without requiring any label information [13]. The core idea of this approach is to train the model by generating different views of the graph, thereby learning the intrinsic structure of the graph and the relationships between nodes. This is achieved by creating two or more "views" of the graph by adding or deleting nodes or edges, as well as perturbing attributes [14]. A contrastive loss function is designed to train the model, which maximizes the similarity between different views of the same graph and minimizes the similarity between views of different graphs. Common contrastive loss functions include InfoNCE Loss [15] and Contrastive Loss. InfoNCE Loss aims to increase the similarity between different views of the same graph and decrease the similarity between views of different graphs, while Contrastive Loss aims to minimize the distance between positive sample pairs and maximize the distance between negative sample pairs. During training, model parameters are adjusted through backpropagation and gradient descent algorithms to minimize the contrastive loss. This enables the model to distinguish similar and dissimilar structural features in the graph, thereby learning the structural feature information of the graph data in an unsupervised manner.

This study analyzes the limitations of lightweight graph neural networks through knowledge distillation and examines the structural representation capability of graph contrastive learning technology. Based on this analysis, we propose a lightweight method for graph neural networks that combines knowledge distillation with graph contrastive learning. Specifically, the graph contrastive learning technique is first utilized to extract structural features from graph data, which are then concatenated with node content features. The concatenated node features are enhanced and then fed into an MLP. Then, knowledge distillation with variable temperature is employed to transfer knowledge from the teacher model, where the soft label outputs serve as supervisory signals for training the student model, thus facilitating knowledge transfer. To enhance the robustness of the student model MLP, an adversarial feature enhancement strategy is employed to minimize the impact of noise on the accuracy and stability of the MLP. This enables the generation of a noise-resistant, high-efficiency, and high-performance student MLP model, which we call KDGCL. In order to comprehensively evaluate our proposed model, extensive experiments were

conducted on seven datasets in both transductive (tran) and inductive (ind) settings. We concluded that our model learns through integrating graph structure feature information, where feature enhancement and knowledge distillation can improve the performance and robustness of the student model MLP. The contributions of the proposed method are summarized as follows:

(1) This study proposes a lightweight graph neural network method that combines knowledge distillation and graph contrastive learning. By integrating graph contrastive learning and feature enhancement, we can efficiently distill the knowledge from the GNN and transfer it to a student model.
(2) The effectiveness and robustness of the graph contrast learning module and feature enhancement module are proven using seven public datasets, three teacher model architectures, and two experimental conditions. The experimental results demonstrate that the KDGCL model outperforms NOSMOG on six out of seven datasets.
(3) The variable temperature module we designed enables the temperature to change with the progress of distillation. Compared with using a globally unified temperature, the student MLP obtained with distillation has better performance and stronger stability.

2. Related Works

Many graph neural networks [16–20] have been specifically proposed for processing graph-structured data. Most of these graph neural networks follow the message passing paradigm scheme, which aggregates neighbor node information to update node representations [21]. Graph convolutional neural networks (GCNs) calculate graph node representations through computational propagation rules similar to convolutional neural networks [19]. Graph attention networks (GATs) introduce an attention mechanism to aggregate the feature representation of neighbor nodes with different weights [16]. The GraphSAGE approach samples neighbor nodes and applies an aggregation function to aggregate the sampled node feature information and learn node features, enabling its good scalability [18]. GAEs embed graph nodes into low-dimensional space through the encoder and reconstruct the graph structure through the decoder to learn node representations [22,23]. DeepGCNs [24] and GCNII [20] introduce residual connections to address the issues of over-smoothing and over-fitting in graph neural networks. However, research has shown that graph neural networks in the message passing scheme can only exploit local graph structures and have been proven not to perform better than WL graph isomorphism tests [17,25,26]. Inspired by positional encoding in natural language processing technology, some studies have enhanced the graph learning process through integrating node position information [3,4,11], such as Laplacian feature mapping and random walk methods [27–30], thereby enhancing the node feature encoding ability of graph neural networks.

Contrastive learning obtains positive and negative sample pairs through data enhancement and negative sampling and uses an objective function to increase the similarity between positive sample pairs and reduce the similarity between negative sample pairs to obtain discriminative features [15,31,32]. While graph data come in various forms and are not easy to label, through contrastive learning, graph node feature representations can be learned. For example, InfoGraph learns node representations by maximizing graph-level representation and mutual information between substructures at different scales [33]. SimGRACE utilizes two GNN models as encoders to acquire two similar views for comparison, demonstrating strong performance in terms of generalization and robustness [14]. GCC uses contrastive learning to learn the structural representation of graphs through inter-network and cross-network subgraph instance discrimination [34]. GraphCL [31] and GraphMAE [23] also use contrastive learning to obtain better graph node representations. Utilizing contrastive learning with graph data can effectively address the issue of graph data labels, enabling the learning of node representations and structural features that are beneficial for diverse downstream tasks.

The acceleration and compression of GNNs have attracted the attention of many researchers [3–8,11,12]. Common methods include pruning [35], quantification [36–38],

and knowledge distillation (KD) [3–8,11,12]. Among them, KD has been widely used as it can effectively decrease a model's size while increasing its inference speed. Previous works have applied KD to teach student GNNs with fewer parameters but comparable performance to the teacher GNNs. However, time-consuming message passing with multi-hop neighborhood fetching is still required during the inferencing process of the student GNN. For example, while LSP [6] and TinyGNN [7] preserve local structure information in the KD process, neighbor aggregation is still required during message passing, which consumes a lot of time. To address the high-lattency issue, recent works have combined the low latency of MLP models with the high performance of GNNs. These approaches involve training an MLP-based student model using knowledge distillation from a complex, high-performing GNN model, such as Graph-MLP [2], GLNN [3], and so on. The MLP only takes node features as input during the inference process and does not contain graph structure information (i.e., an adjacency matrix). Due to the lack of necessary graph structure information, the performance of the MLPs obtained by distillation for inference is far inferior to that of the teacher GNNs. Therefore, in the NOSMOG model [4], the authors used the graph embedding algorithm DeepWalk [29] to learn the graph node positions and input them into the MLP obtained through distillation together with the node features for stitching, which substantially improved the performance [4]. However, as the random walk algorithm is used, it cannot obtain the graph structure features well and is susceptible to noise and interference. The performance of the trained student model is unstable, and the global unified temperature is used in the distillation process, which does not enable knowledge transfer to be performed well. In our work, we obtain a better graph structure representation through graph contrastive learning and introduce a variable distillation temperature to promote knowledge transfer. At the same time, the node features are enhanced, which improves the robustness and anti-noise characteristics of the student model.

3. Preliminary

The multilayer perceptron (MLP) is a fundamental feedforward artificial neural network consisting of one or multiple hidden layers, with each hidden layer comprising multiple neuron nodes. Typically, an MLP consists of input, hidden, and output layers, where each neuron node is connected to all nodes in the subsequent layer, forming a fully connected network. The formal formula for a two-layer MLP is as follows:

$$y = f(w_{out} \cdot f(w_{h2} \cdot f(w_{h1} \cdot x + b_{h1}) + b_{h2}) + b_{out}) \tag{1}$$

where x is the input vector, y is the output vector, w_{h1} and w_{h2} are the weight matrices from the input layer to the hidden layer, w_{out} is the weight matrix from the hidden layer to the output layer, b_{h1} and b_{h2} are the bias terms of the hidden layer, b_{out} is the bias term of the output layer, and $f(\cdot)$ is the activation function.

Let $G = (V, E, C)$ be defined, where V is the set of nodes, E is the set of edges, and $C \in R^{N \times d_c}$ represents the d_c dimensional content features of the nodes, with N being the total number of nodes. In the node classification task, the model predicts the class probabilities for any node $v \in V$, where the true node label is $Y \in R^K$ and K is the total number of node classes.

For a given node v, the GNN aggregates information from the neighbors $N(v)$ of node v to learn a node embedding $h_v \in R^{d_n}$ of dimension d_n. In particular, for the node embedding of the l^{th} layer, the neighbor node embeddings are first aggregated and then combined with the embeddings of the previous layer $(l-1)$ to obtain the node embedding of this layer [18]. The entire calculation process is as follows:

$$h_v^{(l)} = UPDATE\left(h_v^{(l-1)}, AGGR\left(\left\{h_u^{(l-1)} : u \in N(v)\right\}\right)\right) \tag{2}$$

For KDGCL, there are three key modules: the graph contrastive learning module, the knowledge distillation module, and the node feature enhancement module. See the overall model architecture section for detailed information.

4. Methodology

We introduce the KDGCL framework in this section. As shown in Figure 1, the whole architecture includes three modules. Module (a) is a graph contrastive learning module that perturbs the original graph to obtain two graph views and applies contrastive learning to acquire graph structural feature representations. Module (b) is a knowledge distillation module that uses variable temperature knowledge distillation to transfer the teacher model's knowledge to the student model. Module (c) is a feature enhancement module that enhances the combined node content and graph structural features before inputting them into the student MLP. This improves the student model's classification accuracy, stability, and robustness against interference.

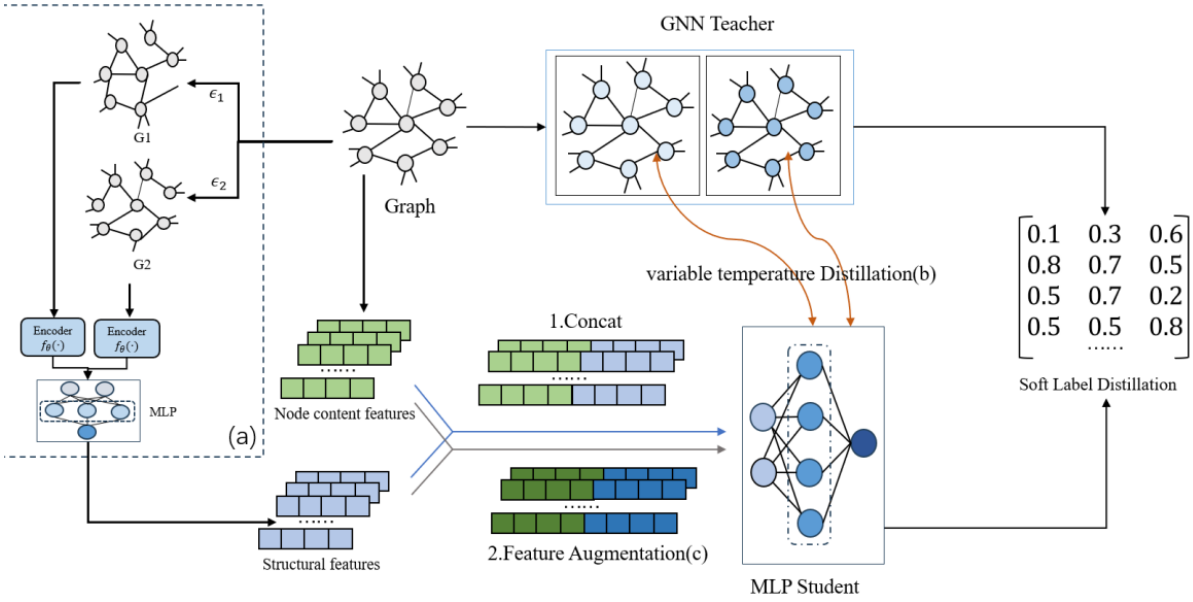

Figure 1. The overall KDGCL framework.

4.1. Graph Contrastive Learning Module

Figure 2 shows the graph contrastive learning framework. First, edges and node features are deleted, and the original input graph is masked to generate two graph views. Then, two shallow GNN encoders with shared parameters encode the two views separately. A contrastive loss function trains the model by bringing the different views of the same graph closer together while pushing away the views of different graphs. This process ultimately allows the model to learn effective graph structural representations.

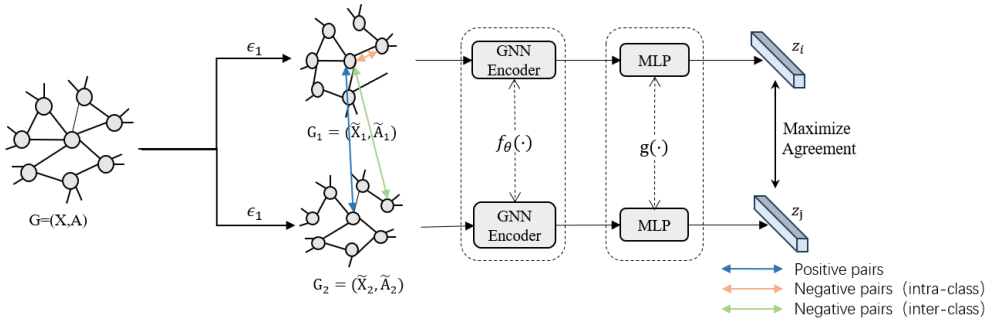

Figure 2. Graph contrastive learning module.

In our model, two views G_1 and G_2 of graph G are generated through the operations ϵ_1 and ϵ_2. The G_1 and G_2 are embedded by the encoder and are denoted by $U = f\left(\widetilde{X}_1, \widetilde{A}_1\right)$ and $V = f\left(\widetilde{X}_2, \widetilde{A}_2\right)$. $\widetilde{X}_1$ and $\widetilde{X}_2$ are node feature matrices of G_1 and G_2, and $\widetilde{A}_1$ and $\widetilde{A}_2$ are adjacent matrices of G_1 and G_2.

After encoding with the GNN, the view representations are fed into an MLP to further enhance the feature representation. The contrastive loss function is then applied to train the model, aiming to maximize the similarity between different views of the same graph while minimizing the similarity between views of different graphs. For any node v_i, the embedding in the G_1 view is u_i, and the embedding in the G_2 view is v_i. Therefore, v_i and u_i form a positive sample pair. In addition, all sample pairs formed in the two views are negative sample pairs. We define $\theta(u, v) = s(g(u), g(v))$, where s is the cosine similarity and g is the non-linear mapping function, which is implemented here with two layers of the MLP. We define the loss of each positive sample pair as:

$$l(u_i, v_i) = log \frac{e^{\frac{\theta(u_i, v_i)}{\tau}}}{e^{\frac{\theta(u_i, v_i)}{\tau}} + \sum_{k=1}^{N} 1_{[k \neq i]} e^{\frac{\theta(u_i, v_k)}{\tau}} + \sum_{k=1}^{N} 1_{[k \neq i]} e^{\frac{\theta(u_i, u_k)}{\tau}}} \tag{3}$$

Here,$1_{[k \neq i]} \in \{0, 1\}$ is the indicator function (if k $\neq$ i, it is equal to 1; otherwise, it is 0), N is the total number of nodes, and τ is the temperature parameter. Negative nodes are not explicitly sampled here. Given a positive sample pair, all other node pairs are defined as negative sample pairs. Therefore, there are two sources of negative sample pairs, namely, intra-class and inter-class sample pairs, corresponding to the second and third terms of the denominator in the above formula, respectively. As the two views are symmetric, all have the same definition for $l(v_i, u_i)$. The overall objective function of the final graph contrastive learning module is defined as follows, which is the average of the loss values of all positive pairs:

$$L = \frac{1}{2N} \sum_{i=1}^{N} [l(u_i, v_i) + l(v_i, u_i)] \tag{4}$$

Here, N is the number of graph data nodes.

To summarize, the graph contrastive learning module first generates two views G_1 and G_2 of the graph G, then uses a graph neural network as an encoder to encode G_1 and G_2 to obtain node embeddings U and V, and finally optimizes the parameters by maximizing L. An effective graph structure embedding representation is then obtained.

4.2. Knowledge Distillation Module

Figure 3 shows the knowledge distillation module framework. The teacher model is a complex and highly accurate GNN, from which knowledge is distilled using distillation techniques and transferred to the student model MLP; in particular, the soft label outputs from the teacher GNN serve as the supervisory signals to train the student MLP.

Given a pre-trained teacher GNN, for any label node $v \in V^L$, the real label value is y_v, and the output of node v from the teacher GNN is a soft label $z_v = Softmax\left(\frac{q^t}{\tau}\right)$. The purpose of knowledge distillation is to use the output of the teacher GNN z_v and node true label y_v to train a lightweight student model MLP. The objective function is formally defined as follows:

$$L = \sum_{v \in V^L} L_{SL}(\hat{y}_v, y_v) + \lambda \sum_{v \in V} L_{KD}(\hat{y}_v, z_v) \tag{5}$$

Here, L_{SL} is the cross-entropy loss between the student model prediction $\hat{y}_v = Softmax\left(\frac{q^s}{\tau}\right)$ and the real label y_v (i.e., the supervision loss), L_{KD} is the KL divergence loss between the student model prediction output $\hat{y}_v$ and the output z_v of the teacher model GNN (i.e., the teacher model to the distillation loss of the student model with respect to the

teacher model), and λ is a factor that balances the supervision loss and the distillation loss. The purpose of the supervision loss here is to prevent the teacher model from passing incorrect knowledge to the student model.

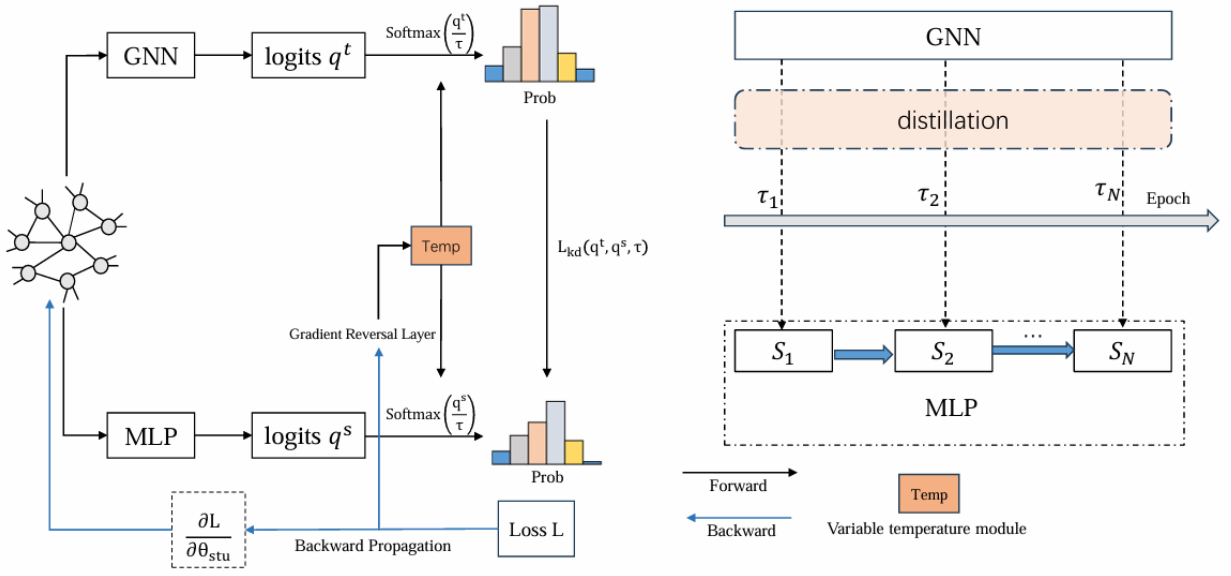

Figure 3. Knowledge distillation module.

During the knowledge distillation process, the knowledge mastered by the student model grows progressively, so maintaining a constant distillation temperature for knowledge distillation will result in suboptimal distillation results. Similar to how teachers teach students knowledge, they tend to transition from simple knowledge to complex knowledge, known as the process of progressive learning. Inspired by this learning process, we simulated the difficulty of the knowledge distillation process by varying the distillation temperature. Notably, changing the distillation temperature does not introduce additional computational costs into the distillation process.

In order to ensure that the distillation temperature is non-negative and remains within a certain range, the distillation temperature τ is determined by the output of the teacher model and the student, as shown in Figure 4. It is scaled using the following formula:

$$\tau = \tau_{init} + \tau_r \left(\sigma \left(T_{pred} \right) \right) \tag{6}$$

Here, τ_{init} is the initial temperature, $\tau(r)$ represents the range of τ, $\sigma(\cdot)$ is the activation function, and T_{pred} is the predicted value (here, τ_r is set from 1 to 15).

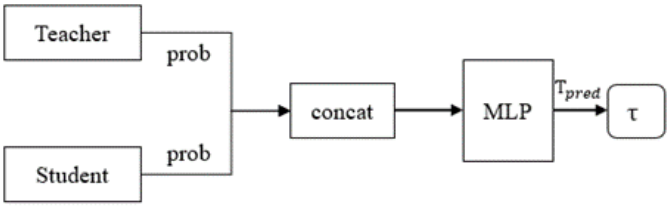

Figure 4. Variable temperatures during knowledge distillation [39].

4.3. Node Feature Enhancement Module

Figure 5 shows the node feature enhancement module framework. In this module, the perturbation feature matrices X_1 and X_2 are first generated by randomly discarding elements from the feature matrix. Then, the perturbed feature matrices are utilized to generate an enhanced feature matrix. To enrich the node feature matrix and enhance the

anti-interference ability of the model, we utilize a two-layer MLP for non-linear transformation of the perturbation feature matrix, resulting in a node information matrix that incorporates structural features. The model can be formalized as follows:

$$X_v = MLP(\,[X_1, X_2]\,) \tag{7}$$

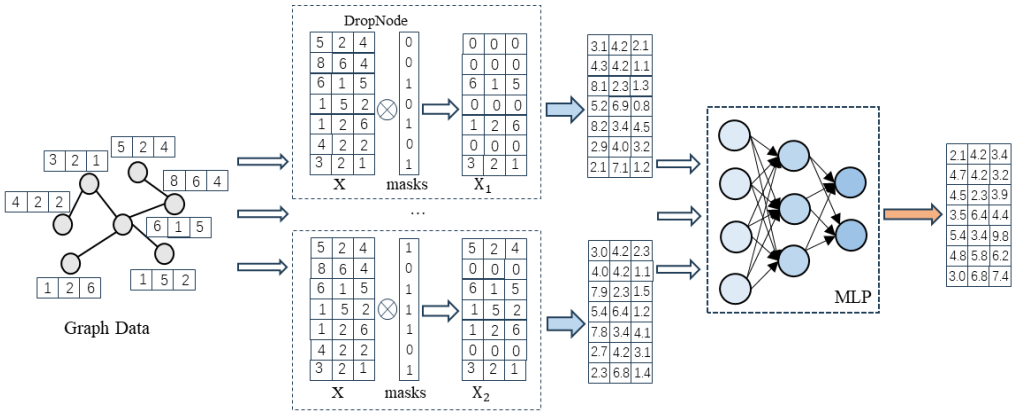

Figure 5. Node feature enhancement module schematic.

Additionally, noise was incorporated into the node feature input of the MLP during the implementation phase in order to facilitate noise robustness learning. The results also demonstrated that the model trained in this way is more stable. The formalization of this process is as follows:

$$\hat{y}_v = MLP(X_v) \tag{8}$$

Here, X_v is the node feature after feature enhancement through connecting node content features and structural features. Then, $\hat{y}_v$ is used to calculate the knowledge distillation loss.

5. Experiments

5.1. Datasets

Our evaluation is based on seven public datasets (Cora [40], Citeseer [40], Pubmed [41], A-computer [42], A-photo [42], Ogbn-Arxiv [43], and Ogbn-Products [43]). Table 1 shows the statistics of the public datasets.

Table 1. Statistics of the datasets.

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2485	5069	1433	7
Citeseer	2110	3668	3703	6
Pubmed	19,717	44,324	500	3
A-computer	13,381	245,778	767	10
A-photo	7487	119,043	745	8
Ogbn-Arxiv	169,343	1,166,243	128	40
Ogbn-Products	2,449,029	61,859,140	100	47

Table 1 lists the datasets utilized in this study, where #Nodes and #Edges denote the number of nodes and edges in each dataset, respectively. Additionally, #Features represents the feature dimensions within the dataset, and #Classes indicates the number of categories included in the dataset.

Cora is a benchmark citation dataset where each node represents a paper and edges denote citation relationships between papers. Similarly, Citeseer also represents papers as nodes, but with higher feature dimensions compared to Cora. Pubmed is a benchmark dataset formed by diabetes-related papers in Pubmed data. Its node feature is the weighted

frequency of TF-IDF, and the category is the type of diabetes. A-computer and A-photo are two benchmark datasets extracted from Amazon co-purchase graphs. In these datasets, nodes represent products, while edges denote frequent co-purchases between products. The features encode product reviews using a bag-of-words model (BoW), and the labels represent product categories. Ogbn-Arxiv is a directed dataset that contains publicly available papers in the field of computer science on Arxiv. Each paper is a node, the citation relationship between papers is used as an edge, and its research field is the category of the paper. Ogbn-Products is an undirected dataset that represents the Amazon product co-purchase network. The nodes represent products, and the edges represent two products that are usually purchased by the same customer. Each node has category information, indicating the category of the product. Among the above-mentioned datasets, Arxiv and Products are relatively large-scale datasets and are often used for benchmark testing of graph neural networks. They are already benchmark datasets in the field of graph machine learning.

5.2. Experiment Setting

For this study, five widely used benchmark datasets were utilized: Cora [40], Citeseer [40], Pubmed [41], A-computer, and A-photo [42], as well as two large-scale OGB datasets: Ogbn-Arxiv [43] and Ogbn-Products [43], to evaluate the proposed model.

For the teacher model architecture, we selected GraphSAGE [18], GCN [19], and GAT [16] as teacher models in order to eliminate the impact of the teacher model architecture on the performance of the student model. For the experimental results, we recorded the average and standard deviation of ten runs using different random seeds, measured the model performance based on node classification accuracy, and conducted testing on the test dataset.

The experimental results were obtained under two setting conditions: transductive and inductive. The so-called transductive setting condition means that the GNN model also includes the data in the test set during the training process, thus lacking the ability to predict unknown labeled samples; meanwhile, the inductive setting condition means that the test dataset is not visible during the training phase. In this way, a judgment rule for data is learned, following which the learned rules are applied to predict or classify unseen data during the testing phase.

5.3. Model Performance

Table 2 details the node classification accuracy of the model on seven public datasets. Three teacher model architectures were used for the teacher model: GCN, GAT, and GraphSAGE. NOSMOG served as the baseline model for comparison. All experiments were conducted in both transductive and inductive settings, and the results in the production (prod) setting represent a combination of tran and ind results in a certain proportion. It can be observed that the performance impact of the three different teacher model architectures on the student model was not significant. This suggests that a well-performing teacher model does not necessarily transfer effective knowledge, and conversely, an average-performing teacher model may still distill into a better-performing student model. Therefore, the results of the student model were obtained using GraphSAGE as the teacher model architecture. The proposed KDGCL method demonstrated the ability to enhance the performance of the student model to varying degrees in both transductive and inductive settings, narrowing the gap in classification accuracy between the student model and the teacher GNN.

Additionally, KDGCL maintained the same parameter count as the baseline model, NOSMOG, thereby demonstrating superior competitiveness in practical environments.

Table 2. Node classification results on the seven datasets.

Dataset	Eval	GCN	GAT	SAGE	MLP	NOSMOG	Ours	ΔNOSMOG
Cora	tran	81.82 ± 1.26	81.89 ± 0.86	79.94 ± 2.43	59.18 ± 1.60	80.47 ± 1.39	82.83 ± 1.25	↑ 2.93%
	ind	82.01 ± 1.68	83.07 ± 2.06	80.96 ± 1.90	59.44 ± 3.36	80.19 ± 1.81	80.89 ± 1.79	↑ 0.87%
	prod	81.45	81.62	79.64	59.22	80.57	80.98	↑ 0.51%
Citeseer	tran	71.79 ± 1.62	70.97 ± 1.58	70.52 ± 1.35	58.51 ± 1.88	70.81 ± 2.42	73.46 ± 1.66	↑ 3.74%
	ind	71.79 ± 3.71	71.13 ± 3.21	68.81 ± 3.89	59.34 ± 4.61	71.06 ± 2.53	72.15 ± 2.28	↑ 1.53%
	prod	70.52	70.64	68.71	58.49	66.79	67.50	↑ 1.06%
Pubmed	tran	77.04 ± 2.45	75.48 ± 1.91	75.37 ± 2.27	68.39 ± 3.09	75.30 ± 2.77	76.88 ± 2.29	↑ 2.10%
	ind	76.40 ± 2.76	75.62 ± 2.65	75.31 ± 2.71	68.29 ± 3.26	75.41 ± 2.71	75.85 ± 3.35	↑ 0.58%
	prod	76.01	75.89	75.02	68.29	75.31	75.84	↑ 0.70%
A-computer	tran	83.52 ± 1.05	82.99 ± 1.32	81.94 ± 1.75	67.79 ± 2.16	83.05 ± 1.71	83.43 ± 1.59	↑ 0.46%
	ind	82.72 ± 1.91	83.48 ± 1.19	82.62 ± 1.53	67.88 ± 2.15	83.09 ± 1.99	83.88 ± 1.64	↑ 0.95%
	prod	82.77	82.69	82.49	67.70	82.88	83.46	↑ 0.70%
A-photo	tran	91.13 ± 1.31	91.70 ± 0.86	91.16 ± 0.48	77.29 ± 1.79	92.33 ± 0.53	92.95 ± 0.82	↑ 0.67%
	ind	91.74 ± 1.33	91.79 ± 1.22	91.18 ± 1.40	77.44 ± 1.50	92.52 ± 0.82	92.39 ± 0.73	↓ 0.14%
	prod	90.68	90.89	91.00	77.29	92.45	92.70	↑ 0.27%
Arxiv	tran	-	-	70.69 ± 0.24	55.57 ± 0.36	70.44 ± 0.31	70.98 ± 0.20	↑ 0.77%
	ind	-	-	70.66 ± 0.39	55.71 ± 0.36	67.67 ± 0.50	68.31 ± 0.53	↑ 0.95%
	prod	-	-	70.48	55.65	69.72 ± 0.87	70.42 ± 0.54	↑ 1.00%
Products	tran	-	-	77.77 ± 0.19	59.99 ± 0.11	77.35 ± 0.30	77.93 ± 0.22	↑ 0.75%
	ind	-	-	77.51 ± 035	59.98 ± 0.10	76.71 ± 0.42	77.38 ± 0.33	↑ 0.87%
	prod	-	-	77.42	59.25	77.19	77.82	↑ 0.82%

In Table 2, ↑ indicates increase and ↓ indicates decrease. Under the same experimental settings, the proposed model was compared with a GNN, an MLP, and the best-performing NOSMOG. In the transductive setting, our method outperformed the baseline method on all datasets, achieving an increase of 1.63% in node classification accuracy. Particularly notable performance improvements were observed on the Citeseer and Cora datasets, reaching 3.74% and 2.93%, respectively. In the inductive setting, KDGCL outperformed the baseline model on 6 out of 7 datasets, with significant improvements observed on the Citeseer, A-computer, and Arxiv datasets (by 1.53%, 0.95%, and 0.95%, respectively). On average, the node classification accuracy was improved by 0.8% compared to the baseline model. Furthermore, comparing the standard deviations of the results between KDGCL and the baseline model, the standard deviation of the results obtained with our method was smaller than that for the baseline model, indicating that the proposed method is more stable.

6. Discussion

KDGCL contains different components (i.e., structural features, GCL; knowledge distillation, KD; and feature enhancement, FA). For this reason, ablation studies were conducted to analyze the contribution of different components to model performance by independently deleting each component. In Table 3, it can be seen that, when a certain component was removed, the model performance decreased, indicating that each component effectively contributes to the final model. It can be seen that the structural characteristics of graph-structured data have the greatest impact on model performance, especially on large-scale graph datasets. By capturing graph structural features through graph contrastive learning, KDGCL obtains better performance.

As can be seen in Table 3, the feature enhancement module (FA) contributed the least to the overall performance on different datasets. This is because the node features had been enhanced before passing through the MLP, and further enhancement may damage the enhanced features. The contribution of knowledge distillation in each dataset was moderate, as the node features after feature enhancement contain more effective information and use

fewer soft label outputs from the teacher model. The graph contrastive learning module had a significant impact on model performance, indicating that rich graph structure features can help improve the performance of the student model. Finally, KDGCL achieved improved performance on all datasets, demonstrating its overall effectiveness.

Table 3. Accuracy of different model variants.

Datasets	w/o GCL	w/o KD	w/o FA	KDGCL	Δ_{GCL}	Δ_{KD}	Δ_{FA}
Cora	80.22 ± 1.65	80.76 ± 1.89	81.20 ± 1.72	82.01 ± 1.36	↑ 2.23%	↑ 1.55%	↑ 1.00%
Citeseer	71.01 ± 2.31	71.51 ± 2.07	71.86 ± 3.01	72.31 ± 2.52	↑ 1.83%	↑ 1.12%	↑ 0.63%
Pubmed	74.89 ± 2.08	75.75 ± 2.19	74.99 ± 2.68	76.62 ± 2.54	↑ 2.31%	↑ 1.15%	↑ 2.17%
A-computer	83.75 ± 1.98	83.50 ± 1.79	83.78 ± 2.01	84.01 ± 1.76	↑ 0.31%	↑ 0.31%	↑ 0.27%
A-photo	92.40 ± 0.79	92.28 ± 0.98	93.24 ± 0.89	93.01 ± 0.86	↑ 0.66%	↑ 0.79%	↑ 0.83%
Arxiv	63.89 ± 0.57	67.40 ± 0.49	68.01 ± 0.88	68.74 ± 0.71	↑ 7.59%	↑ 1.99%	↑ 1.07%
Products	67.48 ± 0.45	77.30 ± 0.65	77.59 ± 0.78	78.03 ± 0.49	↑15.63%	↑ 0.94%	↑ 0.57%

7. Conclusions

In this study, we addressed the issues of low task accuracy, instability, and susceptibility to interference in a student MLP model after distillation from a teacher GNN model. Specifically, we proposed KDGCL—an optimization method for graph neural networks that combines knowledge distillation and graph contrastive learning. KDGCL consists of three modules: the contrastive learning module effectively captures graph structural features; the variable temperature knowledge distillation module facilitates more efficient knowledge transfer; and the feature enhancement module improves the stability and robustness of the student model. Experiments conducted under various conditions on seven datasets revealed that KDGCL outperforms GNNs and baseline models in terms of classification accuracy and stability. In transductive settings, KDGCL achieved an average improvement of 1.63% over baseline models; meanwhile, in inductive settings, KDGCL demonstrated an average improvement of 0.8% compared to baseline models. Additionally, ablation studies were carried out, which demonstrated the effectiveness of the components of the proposed method.

Author Contributions: Conceptualization, Y.W. and S.Y.; methodology, Y.W.; validation, Y.W.; investigation, Y.W.; resources, S.Y.; data curation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, S.Y.; supervision, S.Y.; project administration, S.Y.; funding acquisition, S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China, grant number: 2020AAA0109300.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available in snap-stanford at https://ogb.stanford.edu/# (accessed on 10 April 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
2. Hu, Y.; You, H.; Wang, Z.; Wang, Z.; Zhou, E.; Gao, Y. Graph-MLP: Node classification without message passing in graph. *arXiv* **2021**, arXiv:2106.04051.
3. Zhang, S.; Liu, Y.; Sun, Y.; Shah, N. Graph-less neural networks: Teaching old MLPs new tricks via distillation. *arXiv* **2021**, arXiv:2110.08727.
4. Tian, Y.; Zhang, C.; Guo, Z.; Zhang, X.; Chawla, N.V. NOSMOG: Learning noise-robust and structure-aware MLPs on graphs. *arXiv* **2022**, arXiv:2208.10010.
5. Deng, X.; Zhang, Z. Graph-free knowledge distillation for graph neural networks. *arXiv* **2021**, arXiv:2105.07519.

6. Yang, Y.; Qiu, J.; Song, M.; Tao, D.; Wang, X. Distilling knowledge from graph convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 7074–7083.
7. Yan, B.; Wang, C.; Guo, G.; Lou, Y. Tinygnn: Learning efficient graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 1848–1856.
8. Feng, K.; Li, C.; Yuan, Y.; Wang, G. Freekd: Free-direction knowledge distillation for graph neural networks. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 357–366.
9. Chen, Y.; Bian, Y.; Xiao, X.; Rong, Y.; Xu, T.; Huang, J. On self-distilling graph neural network. *arXiv* **2020**, arXiv:2011.02255.
10. Chen, J.; Chen, S.; Bai, M.; Gao, J.; Zhang, J.; Pu, J. SA-MLP: Distilling graph knowledge from GNNs into structure-aware MLP. *arXiv* **2022**, arXiv:2210.09609.
11. Wu, L.; Lin, H.; Huang, Y.; Li, S.Z. Quantifying the knowledge in GNNs for reliable distillation into MLPs. In Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 37571–37581.
12. Wu, L.; Lin, H.; Huang, Y.; Fan, T.; Li, S.Z. Extracting low-/high-frequency knowledge from graph neural networks and injecting it into MLPs: An effective GNN-to-MLP distillation framework. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; pp. 10351–10360.
13. Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; Wang, L. Graph contrastive learning with adaptive augmentation. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 2069–2080.
14. Xia, J.; Wu, L.; Chen, J.; Hu, B.; Li, S.Z. Simgrace: A simple framework for graph contrastive learning without data augmentation. In Proceedings of the ACM Web Conference 2022, Lyon, France, 25–29 April 2022; pp. 1070–1079.
15. Oord, A.v.d.; Li, Y.; Vinyals, O. Representation learning with contrastive predictive coding. *arXiv* **2018**, arXiv:1807.03748.
16. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
17. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? *arXiv* **2018**, arXiv:1810.00826.
18. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
19. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
20. Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; Li, Y. Simple and deep graph convolutional networks. In Proceedings of the International Conference on Machine Learning, Online, 13–18 July 2020; pp. 1725–1735.
21. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1263–1272.
22. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *arXiv* **2016**, arXiv:1611.07308.
23. Hou, Z.; Liu, X.; Cen, Y.; Dong, Y.; Yang, H.; Wang, C.; Tang, J. Graphmae: Self-supervised masked graph autoencoders. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 594–604.
24. Li, G.; Muller, M.; Thabet, A.; Ghanem, B. DeepGCNs: Can GCNs go as deep as CNNs? In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9267–9276.
25. Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W.L.; Lenssen, J.E.; Rattan, G.; Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 4602–4609.
26. Chen, R.; Zhang, S.; Li, Y. Redundancy-free message passing for graph neural networks. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 4316–4327.
27. You, J.; Ying, R.; Leskovec, J. Position-aware graph neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 7134–7143.
28. Li, P.; Wang, Y.; Wang, H.; Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 4465–4478.
29. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
30. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
31. You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; Shen, Y. Graph contrastive learning with augmentations. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 5812–5823.
32. Xu, D.; Cheng, W.; Luo, D.; Chen, H.; Zhang, X. Infogcl: Information-aware graph contrastive learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 30414–30425.
33. Sun, F.-Y.; Hoffmann, J.; Verma, V.; Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv* **2019**, arXiv:1908.01000.
34. Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 1150–1160.
35. Zhou, H.; Srivastava, A.; Zeng, H.; Kannan, R.; Prasanna, V. Accelerating large scale real-time GNN inference using channel pruning. *arXiv* **2021**, arXiv:2105.04528. [CrossRef]

36. Bahri, M.; Bahl, G.; Zafeiriou, S. Binary graph neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 9492–9501.
37. Zhao, Y.; Wang, D.; Bates, D.; Mullins, R.; Jamnik, M.; Lio, P. Learned low precision graph neural networks. *arXiv* **2020**, arXiv:2009.09232.
38. Tailor, S.A.; Fernandez-Marques, J.; Lane, N.D. Degree-quant: Quantization-aware training for graph neural networks. *arXiv* **2020**, arXiv:2008.05000.
39. Li, Z.; Li, X.; Yang, L.; Zhao, B.; Song, R.; Luo, L.; Li, J.; Yang, J. Curriculum temperature for knowledge distillation. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; pp. 1504–1512.
40. Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; Eliassi-Rad, T. Collective classification in network data. *AI Mag.* **2008**, *29*, 93. [CrossRef]
41. Namata, G.; London, B.; Getoor, L.; Huang, B.; Edu, U. Query-driven active surveying for collective classification. In Proceedings of the 10th International Workshop on Mining and Learning with Graphs, Edinburgh, UK, 1 July 2012; p. 1.
42. Shchur, O.; Mumme, M.; Bojchevski, A.; Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv* **2018**, arXiv:1811.05868.
43. Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22118–22133.