Boosting Graph Neural Networks via Adaptive Knowledge Distillation

Zhichun Guo[1], Chunhui Zhang[2], Yujie Fan[3], Yijun Tian[1], Chuxu Zhang[2], Nitesh V. Chawla[1]

[1]University of Notre Dame, Notre Dame, IN 46556
[2]Brandeis University, Waltham, MA 02453
[3]Case Western Reserve University, Cleveland, OH 44106
{zguo5,yijun.tian,nchawla}@nd.edu
{chunhuizhang,chuxuzhang}@brandeis.edu, yxf370@case.edu

Abstract

Graph neural networks (GNNs) have shown remarkable performance on diverse graph mining tasks. While sharing the same message passing framework, our study shows that different GNNs learn distinct knowledge from the same graph. This implies potential performance improvement by distilling the complementary knowledge from multiple models. However, knowledge distillation (KD) transfers knowledge from high-capacity teachers to a lightweight student, which deviates from our scenario: GNNs are often shallow. To transfer knowledge effectively, we need to tackle two challenges: how to transfer knowledge from compact teachers to a student with the same capacity; and, how to exploit student GNN's own learning ability. In this paper, we propose a novel adaptive KD framework, called *BGNN*, which sequentially transfers knowledge from multiple GNNs into a student GNN. We also introduce an adaptive temperature module and a weight boosting module. These modules guide the student to the appropriate knowledge for effective learning. Extensive experiments have demonstrated the effectiveness of BGNN. In particular, we achieve up to 3.05% improvement for node classification and 6.35% improvement for graph classification over vanilla GNNs.

Introduction

Recent years have witnessed the significant development of graph neural networks (GNNs). Various GNNs have been developed and applied to different graph mining tasks (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Velickovic et al. 2018; Klicpera, Bojchevski, and Günnemann 2019; Xu et al. 2019; Wu et al. 2019; Jin et al. 2021; Guo et al. 2022a). Although most GNNs can be unified into the Message Passing Neural Networks (Gilmer et al. 2017), their learning abilities diverge (Xu et al. 2019; Balcilar et al. 2020). In our preliminary study, we observe that the graph representations learned by different GNNs are not similar, especially in deeper layers. It suggests that different GNNs may encode complementary knowledge due to their different aggregation schemes. Based on this observation, it is natural to ask: *can we boost vanilla GNNs by effectively utilizing complementary knowledge learned by different GNNs from the same dataset?*

An intuitive solution is to compose multiple models into an ensemble (Hansen and Salamon 1990; Breiman 2001) that would achieve better performance than each of its constituent models. However, ensemble is not always effective especially when the base classifiers are strong learners (Zhang et al. 2020). Thus, we seek a different approach to take advantage of knowledge from different GNNs: knowledge distillation (KD) (Hinton et al. 2015; Romero et al. 2014; Touvron et al. 2021), which distills information from one (teacher) model to another (student) model. However, KD is always accompanied by model compression (Yim et al. 2017; Heo et al. 2019; Yuan et al. 2019), where the teacher network is a high-capacity neural network, and the student network is a compact and fast-to-execute model. Standing by this situation, there could be a significant performance gap between students and teachers. But this kind of performance gap may not exist in our scenario: GNNs are all very shallow due to the over-smoothing issue (Zhao and Akoglu 2019; Li, Han, and Wu 2018; Alon and Yahav 2020). Hence, it is more difficult to distill extra knowledge from teacher GNNs to boost the student GNN. To achieve this goal, two major challenges arise: *the first one* is how to transfer knowledge from a teacher GNN into a student GNN with the same capacity that can produce the same even better performance (*teaching effectiveness*); *the second one* is how to push the student model to play the best role in learning by itself, which is ignored in the traditional KD where the student's performance heavily relies on the teacher (*learning ability*).

In this work, we propose a novel framework, namely BGNN, which combines the knowledge from different GNNs in a "boosting" way to strengthen a vanilla GNN through knowledge distillation. To improve the teaching effectiveness, we propose two strategies to increase the useful knowledge transferred from the teachers to the student. One is the sequential training strategy, where the student is encouraged to focus on learning from one teacher at a time. This allows the student to learn diverse knowledge from individual GNNs. The other one is an adaptive temperature module. Unlike existing KD methods that use a uniform temperature for all samples, the temperature in BGNN is adjustable based on the teacher's confidence in a specific sample. To enhance the learning ability, we develop a weight boosting module. This module redistributes the weight of samples, making the student GNN pay more attention to the misclassified samples. Our proposed BGNN is a general model which can be applied to both graph classification and node classification tasks. We

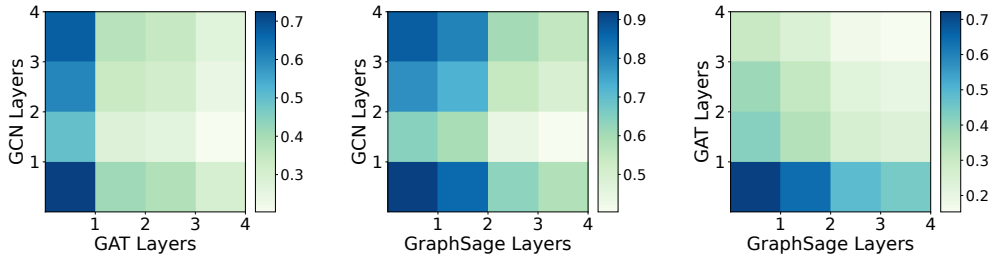

Figure 1: CKA similarity between graph representation at different layers of GNNs on Enzymes.

conduct extensive experimental studies on both tasks, and the results demonstrate the superior performance of BGNN compared with a set of baseline methods.

To summarize, our contributions are listed as follows:

- Through empirical study, we show that the representations learned by different GNNs are not similar, indicating that they encode different knowledge from the same input.
- Motivated by our observation, we propose a novel framework BGNN that transfers knowledge from different GNNs in a "boosting" way to elevate a vanilla GNN.
- Rather than using a uniform temperature for all samples, we design an adaptive temperature for each sample, which benefits the knowledge transfer from teacher to student.
- Empirical results have demonstrated the effectiveness of BGNN. Particularly, we achieve up to 3.05% and 6.35% improvement over vanilla GNNs for node classification and graph classification, respectively.

Related Work and Background

Graph Neural Networks. Most GNNs follow a message-passing scheme, which consists of message, update, and readout functions to learn node embeddings by iteratively aggregating the information of its neighbors (Xu et al. 2019; Wu et al. 2019; Klicpera, Bojchevski, and Günnemann 2019). For example, GCN (Kipf and Welling 2017) simplifies graph convolutions, which takes averaged aggregation method to aggregate the neighbors' information; GraphSage (Hamilton, Ying, and Leskovec 2017) fixes the number of sampled neighbors to perform aggregation; GAT (Velickovic et al. 2018) proposes an attention mechanism (Vaswani et al. 2017) to treat neighbors differently in aggregation. The aim of this work is not to design a new GNN architecture, but to propose a new framework to boost existing GNNs by leveraging the diverse learning abilities of different GNNs.

GNN Knowledge Distillation. There have been many models that apply KD framework on GNNs for better efficiency in different settings (Yang, Liu, and Shi 2021; Zheng et al. 2022; Zhang et al. 2020; Deng and Zhang 2021; Feng et al. 2022). For example, Yan et al. (Yan et al. 2020) proposed TinyGNN to distillate a large GNN to a small GNN. GLNN (Zhang et al. 2022) was proposed to distillate GNNs to MLP. All of these work distillate knowledge by penalizing the softened logit differences between a teacher and a student following (Hinton et al. 2015). Besides this vanilla KD, Yang et al. (Yang et al. 2020) proposed LSP, a local structure preserving based KD method in computer vision area, to transfer the knowledge effectively between different GCN models. Wang et al. (Wang et al. 2021) propose a novel multi-teacher KD method, MulDE, for link prediction based on knowledge graph embeddings. LLP (Guo et al. 2022b) is another KD framework specifically for link prediction tasks. In this work, we also take logits-based KD method to distillate knowledge but design two modules to increase useful knowledge transferred from the teachers to the student and play the best of the student model. Different from combining teachers' knowledge in a parallel way in MulDE, we utilize sequential training strategy to combine different teacher models.

Background and Preliminary Study

Background

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where $\mathcal{V}$ stands for all nodes and $\mathcal{E}$ stands for all edges. Each node $v_i \in \mathcal{V}$ in the graph has a corresponding D-dimensional feature vector $\boldsymbol{x}_i \in \mathbb{R}^D$. There are N nodes in the graph. The entire node features matrix is $\mathbf{X} \in \mathbb{R}^{N \times D}$.

Graph Neural Networks. A GNN iteratively updates node embeddings by aggregating information of its neighboring nodes. We initialize the embedding of node v as $\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v$. Its embedding in the l-th layer is updated to $\boldsymbol{h}_v^{(l)}$ by aggregating its neighbors' embedding, which is formulated as: $\boldsymbol{h}_v^{(l)} = \text{UPDATE}_l\big(\boldsymbol{h}_v^{(l-1)}, \text{AGG}_l\big(\{\boldsymbol{h}_u^{(l-1)} : \forall u \in \mathcal{N}(v)\}\big)\big)$, where AGG and UPDATE are aggregation function and update function, respectively, $\mathcal{N}(v)$ denotes the neighbors of node v. Furthermore, the whole graph representation can be computed based on all nodes' representations as: $\boldsymbol{h}_G = \text{READOUT}(\{\boldsymbol{h}_v^{(l)} | v \in \mathcal{V}\})$, where READOUT is a graph-level pooling function.

Node Classification. Node classification is a typical supervised learning task for GNNs. The target is to predict the label of unlabeled node v in the graph. Let $\mathbf{Y} \in \mathbb{R}^{N \times C}$ be the set of node labels. The ground truth of node v will be y_v, a C-dimension one-hot vector.

Graph Classification. Graph classification is commonly used in chemistry tasks like molecular property prediction (Hu et al. 2019; Guo et al. 2021). Graph classification is to predict the graph properties. Here, the ground truth matrix $\mathbf{Y} \in \mathbb{R}^{M \times C}$ is the set of graph labels, where M and C are the number of graphs and graph categories, respectively.

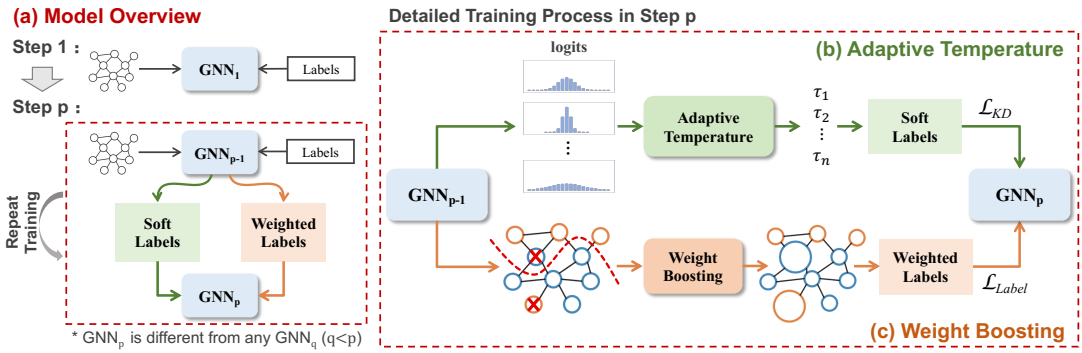

Figure 2: The overall framework of BGNN. (a) shows our sequential training process, where we train a teacher GNN_1 with labels. Then we repeatedly take the generated GNN model from previous step as the teacher to generate soft labels and update the training nodes' weight to train a new initialized student. (b) presents the adaptive temperature module, where the temperature is adjusted based on the teacher's logits distribution for each node. (c) shows the weight boosting module, where the weight of the nodes misclassified by the teacher GNN are boosted (nodes with large size in the above figure).

Preliminary Study on GNNs' Representation

Next, we perform a preliminary study to answer the following question: *do different GNNs encode different knowledge from the same input graphs?* We train 4-layer GCN, GAT, and GraphSage on Enzymes in a supervised way. After the training, we utilize Centered Kernel Alignment (CKA) (Kornblith et al. 2019) as a similarity index to evaluate the relationship among different representations. The higher CKA means the compared representations are more similar. We take the average of all the embeddings at each layer as the representations of that layer. Figure 1 illustrates the CKA between representations of each layer learned from GCN, GAT, and GraphSage on Enzymes. We observe that the similarities between the learned representations at different layers in GCN, GAT, and GraphSage are diverse. For example, the CKA value between the representation from layer 1/2/3/4 of GCN and that from GAT is around 0.7/0.35/0.4/0.3. It indicates that different GNNs may encode different knowledge.

We posit that different aggregation schemes in these GNNs cause difference in the learned representations. In particular, GCN aggregates neighborhoods with predefined weights; GAT aggregates neighborhoods using learnable weights and GraphSage randomly samples neighbors during aggregation. Given such differences, it is promising to boost one GNN by incorporating the knowledge from other GNNs and it motivates us to design a framework that can take advantage of the diverse knowledge from different GNNs.

The Proposed Framework

In this section, we introduce the framework BGNN to boost GNNs by utilizing complementary knowledge from other GNNs. An illustration of the model framework is shown in Figure 2. Our framework adopts a sequential training strategy to encourage the student to focus on learning from one single teacher at a time. To adjust the information distilled from the teacher, we propose an adaptive temperature module to adjust the soft labels from teachers. Further, we propose a weight boosting mechanism to enhance the student model training.

Model Overview

To boost a GNN, we take it as the student and we aim to transfer diverse knowledge from other teacher GNNs into it. In this work, we utilize the KD method proposed by (Hinton et al. 2015), where a teacher's knowledge is transferred to a student by encouraging the student model to imitate the teacher's behavior. In our framework, we pre-train a teacher GNN (GNN_T) with ground-truth labels and keep its parameters fixed during KD. Then, we transfer the knowledge from GNN_T by letting the student GNN (GNN_S) optimize the soft cross-entropy loss between the student network's logits $\boldsymbol{z}_v$ and the teachers' logits $\boldsymbol{t}_v$. Let τ_v be the temperature for node v to soften the logits distribution of teacher GNN.

Then we incorporate the target of knowledge distillation into the training process of the student model by minimizing the following loss:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{label}} + \lambda \mathcal{L}_{KD} = \mathcal{L}_{\text{label}} - \lambda \sum\nolimits_{v \in \mathcal{V}} \hat{\boldsymbol{y}}_v^{\text{T}} \log(\hat{\boldsymbol{y}}_v^{\text{S}}), \\ \text{with} \quad & \hat{\boldsymbol{y}}_v^{\text{T}} = \text{softmax}(\boldsymbol{t}_v/\tau_v),\ \hat{\boldsymbol{y}}_v^{\text{S}} = \text{softmax}(\boldsymbol{z}_v/\tau_v), \end{aligned} \tag{1}$$

where $\mathcal{L}_{\text{label}}$ is the supervised training loss w.r.t. ground-truth labels and λ is a trade-off factor to balance their importance.

Following the above objective, we adopt a sequential process of distillation, where we can freely boost the student GNN with one teacher GNN or multiple teachers. Such a sequential manner encourages the student model to focus on the knowledge from one single teacher. In contrast, when using multiple teachers simultaneously, the student may receive mixed noisy signals which can harm the distillation process.

As illustrated in Figure 2(a), we first train a teacher GNN_1 using true labels and train a student GNN_2 with dual targets of predicting the true labels and matching the logits distribution of GNN_1. The logits distribution has been softened by our proposed adaptive temperature (Figure 2(b)) for each node and the weight of the nodes misclassified by the teacher GNN are boosted when predicting true labels (Figure 2(c)). The parameters of teacher GNN_1 are not updated when we train GNN_2. Such process is repeated for p steps and the

knowledge from $\text{GNN}_{p-1}, \ldots, \text{GNN}_1$ can be transferred into the last student GNN_p.

Distillation via Adaptive Temeperature

Instead of leveraging KD for compressing GNN models, we aim to use KD for boosting the student by transferring knowledge between GNNs sharing the same capacity. To achieve this goal, we need a more powerful KD method to fully take advantage of the useful knowledge in the teacher GNN so as to produce better performance.

Analysis of KD. Before we introduce the detailed technique, we first analyze the reason behind KD's success by comparing the gradient of $\mathcal{L}_{KD}$ and supervised loss. Specifically, for any single sample, we compute the gradient of $\mathcal{L}_{KD}$ for its c-th output (Detailed derivation is in Section A of Appendix):

$$\begin{aligned}\frac{\partial \mathcal{L}_{KD}}{\partial z_{v,c}} &= -\frac{\partial}{\partial z_{v,c}} \sum\nolimits_{j=1}^{C} \hat{y}^{\mathrm{T}}_{v,j} \log(\hat{y}^{\mathrm{S}}_{v,j}) = \frac{1}{\tau_v}(\hat{y}^{\mathrm{S}}_{v,c} - \hat{y}^{\mathrm{T}}_{v,c}) \\ &= \frac{1}{\tau_v}\Big(\frac{e^{z_{v,c}/\tau_v}}{\sum_{j=1}^{C} e^{z_{v,j}/\tau_v}} - \frac{e^{t_{v,c}/\tau_v}}{\sum_{j=1}^{C} e^{t_{v,j}/\tau_v}}\Big). \end{aligned} \tag{2}$$

Let $*$ denote the true label for the single sample, i.e., $y_{v,*} = 1$. Then the gradient of KD loss for this sample is:

$$\begin{aligned}\frac{\partial \mathcal{L}_{KD}}{\partial z_v} &= \frac{1}{\tau_v} \sum\nolimits_{c=1}^{C} (\hat{y}^{\mathrm{S}}_{v,c} - \hat{y}^{\mathrm{T}}_{v,c}) \\ &= \frac{1}{\tau_v}\Big((\hat{y}^{\mathrm{S}}_{v,*} - \hat{y}^{\mathrm{T}}_{v,*}) + \sum\nolimits_{c=1,c\neq *}^{C} (\hat{y}^{\mathrm{S}}_{v,c} - \hat{y}^{\mathrm{T}}_{v,c})\Big). \end{aligned} \tag{3}$$

In the above equation, the first term $(\hat{y}^{\mathrm{S}}_{v,*} - \hat{y}^{\mathrm{T}}_{v,*})$ corresponds to transferring the knowledge hidden in the distribution of the logits of the correct category; the second term is responsible for transferring the dark knowledge from the wrong categories. This dark knowledge contains important information about the similarity between categories, which is the key point for the success of KD (Hinton et al. 2015). We rewrite the first term as: $\hat{y}^{\mathrm{S}}_{v,*} - \hat{y}^{\mathrm{T}}_{v,*} y_{v,*}$. Note that the gradient for the sample of the cross entropy between student logits z_v and ground truth label is $\hat{y}^{\mathrm{S}}_{v,*} - y_{v,*}$ when τ_v is 1. We can view $\hat{y}^{\mathrm{T}}_{v,*}$ as the importance weight of the true label information. If the teacher is confident for this sample (i.e. $\hat{y}^{\mathrm{T}}_{v,*} \approx 1$), the ground truth related information will play a more important role in gradient computation. Thus, both true label information and similarity information hidden in the wrong categories contribute a lot to the success of KD. The confidence of the teacher is the key to balancing these two parts' importance in KD training process.

Detailed Technique. From the above analysis, temperature directly controls the trade-off between true label knowledge and dark knowledge. Additionally, existing work (Zhang and Sabuncu 2020) has proved that temperature scaling helps to yield more calibrated models. To transfer more useful knowledge from teachers to students, we make the predefined hyper-parameter temperature adaptive for all nodes: each node is associated with an adaptive temperature based on the confidence of teachers for each node. Specifically, we take the entropy of the teacher's logits $\boldsymbol{t}_v$ to measure the teachers' confidence for each node. The lower entropy means more confident the teacher is for the specific node. Then we compute the temperature for each node with learnable parameters by the following formulation:

$$\text{Confidence}(\boldsymbol{t}_v) = -\sum\nolimits_{c=1}^{C} t_{v,c} log(t_{v,c}), \tag{4}$$

$$\tau_v = \sigma\Big(\text{MLP}\big(\text{Confidence}(\boldsymbol{t}_v)\big)\Big), \tag{5}$$

where σ represents sigmoid operation. We use $\tau_v = \tau_v \times (\tau_{max} - \tau_{min}) + \tau_{min}$ to limit the temperature into a fixed range $[\tau_{min}, \tau_{max}]$. In the experiments, we discover that involving the distribution of the logits of the teacher in the temperature is more effective than only considering the teachers' confidence after two training steps:

$$\tau_v = \sigma\Big(\text{MLP}\big(\textsc{Concat}(\boldsymbol{t}_v, \text{Confidence}(\boldsymbol{t}_v))\big)\Big), \tag{6}$$

where CONCAT represents the concatenation function.

Weight Boosting

As we mentioned earlier, we aim to transfer knowledge from teachers to students with the same capacity. In this case, we should not only enhance the knowledge transferred from teachers to the student but also enhance the supervised training process of the student GNN itself. For the supervised training procedure, inspired by Adaboost algorithm (Freund, Schapire, and Abe 1999; Sun, Zhu, and Lin 2021), we propose to boost the weights of misclassified nodes by the teacher GNN, which encourages the student GNN to pay more attention to these misclassified samples and learn them better. For better efficiency, we drop the ensemble step of boosting and take the student GNN generated by the last training step for the latter prediction in test data. In specific, firstly, we initialize the node weights $w_i = 1/N_{train}, i = 1, 2, 3, ..., N_{train}$ on training set. Next, we pre-train a teacher GNN with samples of the training set in a supervised way. Then we boost the weights of training samples (i.e. graphs for graph classification and nodes for node classification) that are misclassified by GNN_{T}. Here, we apply SAMME.R (Hastie et al. 2009) to update the weight:

$$w_i = w_i \cdot \exp\Big(-\frac{C-1}{C} \boldsymbol{y}_i^{\top} \log \boldsymbol{p}_i\Big), i = 1, 2, ..., N_{train}, \tag{7}$$

where $\boldsymbol{p}_i = \text{softmax}(\boldsymbol{z}_i)$.

Overall Objective. By incorporating w_i into the supervised training process and combining KD training, the loss function of our method can be formulated as:

$$\begin{aligned}\mathcal{L}_{BGNN} &= \mathcal{L}_{Label} + \lambda \mathcal{L}_{KD} \\ &= -\sum\nolimits_{i=1}^{N_{train}} w_i \, \boldsymbol{y}_i \log(\boldsymbol{p}_i) - \lambda \sum_{v \in \mathcal{V}} \hat{\boldsymbol{y}}_v^{\mathrm{T}} \log(\hat{\boldsymbol{y}}_v^{\mathrm{S}}). \end{aligned} \tag{8}$$

Experiments

Experimental Setup

Datasets. We use seven datasets to conduct graph classification and node classification experiments. We follow the data split as in original papers (Sen et al. 2008; Namata et al. 2012) for Cora, Citeseer and Pubmed while the remaining

		Graph Classification			Node Classification			
Student	Method	COLLAB	IMDB	ENZYMES	CORA	CITESEER	PUBMED	A-COMPUTERS
GCN	NoKD	80.82±0.99	77.20±0.40	66.33±1.94	82.25±0.39	72.30±0.21	79.40±0.56	87.79±0.04
	KD (Hinton et al. 2015)	81.24±0.63	77.60±0.77	65.00±1.36	82.78±0.42	72.59±0.28	79.60±0.67	87.13±0.11
	LSP (Yang et al. 2020)	81.22±0.29	77.99±1.21	67.12±1.11	82.29±0.77	72.77±0.34	78.93±0.38	87.93±0.72
	BGNN	**82.73±0.34**	**79.33±0.47**	**69.44±0.79**	**83.97±0.17**	**73.87±0.24**	**80.73±0.25**	**89.58±0.03**
SAGE	NoKD	81.20±0.55	77.60±1.02	73.67±5.52	81.18±0.92	71.62±0.33	78.00±0.23	87.49±1.40
	KD (Hinton et al. 2015)	80.34±0.50	77.90±1.14	71.67±1.36	81.72±1.04	72.58±0.32	77.25±0.39	89.34±0.18
	LSP (Yang et al. 2020)	81.33±0.31	78.23±0.91	74.23±2.16	82.00±0.75	71.37±0.34	77.40±0.22	88.49±0.72
	BGNN	**82.67±0.57**	**79.67±0.47**	**78.33±1.00**	**83.30±0.35**	**73.90±0.16**	**79.03±0.09**	**89.85±0.12**
GAT	NoKD	79.28±0.47	76.20±1.33	74.33±2.00	83.44±0.17	72.46±0.75	79.36±0.19	87.98±0.29
	KD (Hinton et al. 2015)	78.98±1.01	77.80±1.25	72.67±2.36	83.50±0.27	72.52±0.20	78.91±0.18	86.90±0.25
	LSP (Yang et al. 2020)	78.99±1.39	78.00±0.93	74.43±3.27	83.36±0.57	71.90±0.36	79.53±0.26	86.90±0.48
	BGNN	**80.73±0.25**	**79.33±0.94**	**79.36±2.81**	**84.63±0.28**	**74.53±0.29**	**79.83±0.12**	**89.43±0.11**

Table 1: Classification performances under the single teacher setting. NoKD denotes the results of supervised training without KD. Bold text is used to highlight best results for each backbone. Shadow is to highlight best results for each dataset.

		Graph Classification			Node Classification			
Student	Method	COLLAB	IMDB	ENZYMES	CORA	CITESEER	PUBMED	A-COMPUTERS
GCN	BAN (Furlanello et al. 2018)	81.60±0.40	78.50±1.00	66.67±1.67	83.17±0.26	72.47±0.21	79.87±0.21	88.78±0.05
	MulDE (Wang et al. 2021)	80.86±1.18	77.50±1.20	67.33±0.90	82.53±0.05	72.33±0.09	78.73±0.09	88.41±0.12
	BGNN(s)	82.73±0.34	79.33±0.47	69.44±0.79	83.97±0.17	73.87±0.24	80.73±0.25	**89.58±0.03**
	BGNN(m)-ST	82.87±0.09	**79.67±0.27**	**71.12±2.45**	**84.83±0.25**	73.60±0.14	80.20±0.08	89.03±0.02
	BGNN(m)-TS	**83.40±0.15**	79.00±0.00	70.00±1.36	84.40±0.22	**74.87±0.25**	**80.90±0.00**	89.02±0.03
SAGE	BAN (Furlanello et al. 2018)	81.80±0.20	**80.73±0.20**	70.00±1.67	82.80±0.31	73.10±0.57	77.90±0.08	89.73±0.08
	MulDE (Wang et al. 2021)	81.00±0.91	77.60±1.50	77.00±0.14	81.67±0.09	68.83±0.34	78.13±0.34	88.05±1.35
	BGNN(s)	82.67±0.57	79.67±0.47	**78.33±1.00**	83.30±0.35	73.90±0.16	79.03±0.09	89.85±0.12
	BGNN(m)-TC	82.80±0.28	79.67±0.49	78.03±0.49	**83.90±0.22**	**74.67±0.33**	**79.23±0.05**	90.12±0.05
	BGNN(m)-CT	**83.30±0.23**	79.33±0.79	77.78±1.57	83.90±0.00	74.40±0.16	79.10±0.08	**90.40±0.16**
GAT	BAN (Furlanello et al. 2018)	80.10±0.50	79.50±0.50	76.83±1.50	83.93±0.21	72.90±0.08	79.57±0.12	87.66±0.15
	MulDE (Wang et al. 2021)	79.40±1.65	78.50±0.43	77.50±0.58	83.23±0.33	71.23±0.19	78.40±0.08	88.08±0.18
	BGNN(s)	80.73±0.25	79.33±0.94	79.36±2.81	84.63±0.28	74.53±0.29	79.83±0.12	**89.43±0.11**
	BGNN(m)-SC	81.53±0.41	**81.33±1.79**	78.89±0.79	**84.77±0.05**	74.20±0.08	**80.30±0.17**	89.27±0.04
	BGNN(m)-CS	**81.80±0.09**	80.33±0.94	**80.68±1.49**	84.30±0.22	**74.70±0.08**	79.83±0.05	89.07±0.11
Ensemble (Hansen and Salamon 1990)		82.27±0.09	78.67±0.94	80.00±3.60	82.63±0.05	72.43±0.26	78.17±0.31	90.40±0.30

Table 2: Classification performances under the multi teacher setting. For simplicity, we abbreviate GCN to C, GraphSage to S, and GAT to T. The order in which the abbreviation letters appear reflects their training order. For example, BGNN-ST indicates that GraphSage (S) and GAT (T) are the teachers in turn. We further denote the GNNs trained with single teacher as BGNN(s) and the GNNs trained with multiple teachers as BGNN(m). Best performances for each backbone and each dataset are marked with bold and shadow, respectively.

datasets are randomly split using an empirical ratio. The more details are in Section B of Appendix.

BGNN Training. We select GraphSage (Hamilton, Ying, and Leskovec 2017), GCN (Kipf and Welling 2017) and GAT (Velickovic et al. 2018) as GNN backbones and examine the performance of BGNN with different combinations of teachers and students. For all the GNN backbones, we use two-layer models. For the single teacher setting, we use one GNN as the teacher and a different GNN as the student. For the multiple teacher setting, we permute the three GNNs in six orders, where the first two serve as teachers and the third one is the student. During the training, we clamp the adaptive temperature in the range from 1 to 4, which is a commonly used temperature range in KD. Implementation and experiment details are shown in Section C of Appendix.

Baselines. For the single teacher setting, we compare BGNN with the student GNN trained without a teacher and the student GNN trained with different teacher GNNs using KD (Hinton et al. 2015) and LSP (Yang et al. 2020). KD is the most commonly used framework to distillate the knowledge

from teacher GNNs to student GNNs. LSP is the state-of-the-art KD framwork proposed for GNNs, which teaches the student based on local structure information instead of logits information. We set the temperature τ in KD to a commonly used value 4 (Stanton et al. 2021). For the multi teacher setting, we compare BGNN with two multi-teacher KD frameworks (i.e., BAN (Furlanello et al. 2018) and MulDE (Wang et al. 2021)). BAN (Furlanello et al. 2018) uses a similar sequential training strategy as ours. It follows the born-again idea (Breiman and Shang 1996) to train one teacher and multiple students sharing the same architecture, and the resulting model is an ensemble (Hansen and Salamon 1990) of all trained students. MulDE (Wang et al. 2021) trains the student by weighted-combining multiple teachers' outputs and utilizing KD. Here, the teachers' outputs are their generated logits for each node or graph. Additionally, we also include the ensemble that averages the prediction of supervised GCN, GraphSage and GAT as a baseline.

Evaluation. We evaluate the performance by classification accuracy. For graph classification tasks, we obtain the entire graph representation by sum pooling as recommended by (Xu et al. 2019). For each experiment, we report the average and standard deviation of the accuracy from ten training rounds.

How Does BGNN Boost the Vanilla GNNs?

This experiment examines whether our BGNN can boost the performance of the vanilla GNNs by transferring knowledge from multiple GNNs. Both the single teacher setting and the multiple teacher setting are considered.

For the single teacher setting, there are six possible teacher-student pairs: GCN-GAT, GCN-GraphSage, GraphSage-GCN, GraphSage-GAT, GAT-GCN and GAT-GraphSage. Each teacher-student pair is trained using our BGNN, KD and LSP. Note that for each student GNN, two teacher options are available. We report the better result of the two teachers for each of BGNN, KD and LSP, which is different from the setting of our ablation study. In Table 1, we can observe that BGNN outperforms the respective supervisedly trained GNN (NoKD) by significant margins for all datasets, regardless of the selection of student GNN. This confirms that our BGNN can boost the vanilla GNNs by transferring additional knowledge from other GNNs. However, we should also note that, the prediction power of the student GNN is still constrained by its own architecture. For example, for the Enzymes dataset, the original GAT without distillation (74.33%) outperforms the original GCN (66.33%) by a large margin. Although our BGNN can boost the performance of GCN from 66.33% to 69.44%, the boosted accuracy of GCN is still lower than that of the original GAT. This indicates that selecting an appropriate student GNN is still important.

For the multi-teacher setting, we examine the performance of BGNN with all the six permutations of the three GNNs. In Table 2, we find that BGNN(m) outperforms the respective BGNN(s) in most cases, and the supervised trained GNNs (NoKD) in all cases. This indicates that the extra teachers in the BGNN training successfully transfer additional knowledge into the student GNNs.

We further compare BGNN(m) with the average ensemble of the vanilla GNNs, which combines the knowledge of GNNs in a straightforward manner. In Table 2, BGNN(m) obtains better prediction results than those of the ensemble. The ensemble fails to leverage the mutual knowledge in the training stage, as it combines the prediction results directly. In contrast, our BGNN trains the student GNNs under the guidance of teachers. This combines the knowledge from teachers and students in a natural way, providing better prediction power to the student trained with BGNN.

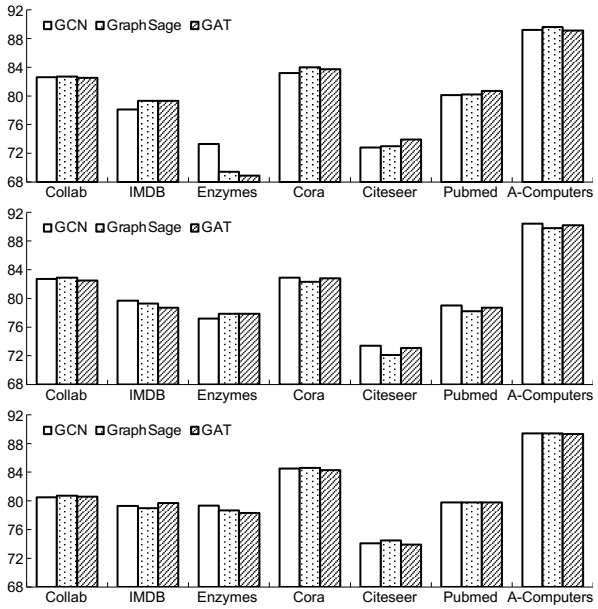

Figure 3: Results of BGNN with different teacher and student GNNs pairs. *Upper*: GCN as student. *Middle*: GraphSage as student. *Lower*: GAT as student.

Does BGNN Distill Knowledge More Effectively?

For the single teacher setting, we compare our BGNN with KD (Hinton et al. 2015) and LSP (Yang et al. 2020). In Table 1, we observe that our method can outperform both KD and LSP on both graph classification and node classification tasks. This may credit to both the adaptive temperature and weight boosting modules.

For the multi-teacher setting, we compare BGNN with MulDE and BAN. MulDE combines the teacher models in parallel, while our BGNN and BAN combines the teacher models in a sequential manner. As shown in Table 2, our BGNN achieves the best performance for all the seven datasets. BGNN also performs better than the respective MulDE and BAN in almost all settings (20 out of 21 cases). In addition, we find that the sequential methods (BGNN and BAN) outperform the parallel method (MulDE) in general.

A possible explanation is that all GNNs play an important role in the parallel MulDE training at the same time, which may not exploit the full potential of the more powerful GNNs. On the contrary, the sequential methods trains a student with a single teacher at each step. This allows the student to focus on learning from one specific teacher GNN. In this way, the prediction power of a single powerful teacher GNN may be

Variant	COLLAB	IMDB	ENZYMES	CORA	CITESEER	PUBMED	A-COMPUTERS
w/o Adaptive Temp	81.27±0.25	79.00±0.00	66.67±1.36	83.30±0.22	73.18±0.23	80.10±0.14	87.30±0.06
w/o Weight Boosting	82.53±0.09	77.67±0.47	68.33±1.36	83.17±0.42	72.80±0.17	79.80±0.14	88.56±0.00
BGNN	**82.53±0.47**	**79.33±0.47**	**68.89±0.79**	**83.70±0.37**	**73.87±0.24**	**80.73±0.25**	**89.12±0.01**

Table 3: Results of fusion selections (BGNN, BGNN without adaptive temperature, and BGNN without weight boosting). Here, we use GAT as the teacher and GCN as the student, the same as for Figure 4 and Figure 5.

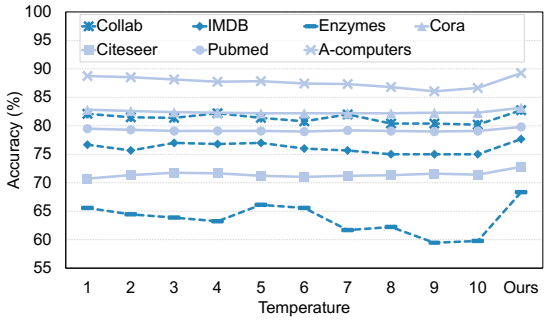

Figure 4: Results of BGNN with adaptive temperature and fixed temperatures from 1 to 10 (without weight boosting).

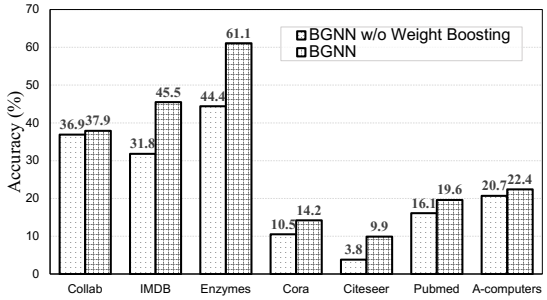

Figure 5: *Right:* Results of BGNN without weight boosting and BGNN on the mis-classifed nodes by the teacher GNN.

transferred to the student more effectively.

For the sequential methods, BAN uses the same architecture for both the teachers and the students. Therefore, it is not able to leverage information from different GNNs. In contrast, our BGNN achieves better results by combining knowledge of different models. Furthermore, it is worth mentioning that our model does not use the average ensemble step like BAN. This may lead to higher accuracy as well, assuming that the final student can inherit the power of all preceding GNNs.

How Does the Selection of Teacher GNNs Influence the Student GNNs?

We study the performance of the student GNNs when they are taught by different teachers. The results are shown in Figure 3. In 18 out of the 21 settings, the student learning from different architectures outperforms the respective ones learning from the same teacher GNNs. One exception is that, on Enzymes, the student GCN learning from GCN is more powerful than the GCNs learning from other GNNs. But for all the other datasets, we still find that GCN learns more from other GNNs. If we only consider the node classification task (the four datasets on the right part of each histogram), it becomes more obvious that a student may learn more from different architectures. In 11 out of the 12 settings, the students learning from the same architecture have the worst performance. However, for the multi-teacher setting, we do not find a clear clue of how to decide the order of teachers. As shown in Table 2, BGNN(m) usually outperforms BGNN(s), but it seems arbitrary when a specific order will be better.

Ablation Study

Does the adaptive temperature matter for knowledge distillation? In Table 3, we can see that the original BGNN performs better than its variation without the adaptive temperature (replacing the adaptive temperature with a fixed temperature). It means the adaptive temperature indeed helps BGNN learn more knowledge from the teacher GNN than the fixed temperature. Then, we further compare the adaptive temperature with the fixed temperatures from 1 to 10. Note that we do not train the models with the weight boosting to exclude its impact. In the left of Figure 4, we find that the adaptive temperature achieves the highest accuracy for all the seven datasets. This means that the adaptive temperature can make the best trade-off between the dark knowledge and true label information to maximize the transferred knowledge.

Does the weight boosting help? In Table 3, we observe that BGNN performs better than BGNN without the weight boosting, which confirms its effectiveness of weight boosting. We argue that the power originates from the improvement on the nodes mis-classified by the teacher GNN. This is verified by the right figure of Figure 5, which compares the performance with and without weight boosting on the mis-classified nodes. We see that BGNN obtains higher accuracy with weight boosting, because the weight boosting forces the student to pay more attention on these mis-classified nodes.

Conclusion

In this paper, we propose a novel KD framework to boost a single GNN by combining various knowledge from different GNNs. We develop a sequential KD training strategy to merge all knowledge. To transfer more useful knowledge from the teacher model to the student model, we propose an adaptive temperature for each node based on the teacher's confidence. Additionally, our boosting weight module helps the student to pick up the knowledge missed by the teacher GNN. The effectiveness of our methods has been proved on both node classification and graph classification tasks.

Acknowledgements

This work was supported by National Science Foundation under the NSF Center for Computer Assisted Synthesis (C-CAS), grant number CHE-2202693. We thank all anonymous reviewers for their valuable comments and suggestions.

References

Alon, U.; and Yahav, E. 2020. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*.

Balcilar, M.; Renton, G.; Héroux, P.; Gaüzère, B.; Adam, S.; and Honeine, P. 2020. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*.

Breiman, L. 2001. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*.

Breiman, L.; and Shang, N. 1996. Born again trees. *University of California, Berkeley, Berkeley, CA, Technical Report*.

Deng, X.; and Zhang, Z. 2021. Graph-Free Knowledge Distillation for Graph Neural Networks. In *IJCAI*.

Feng, K.; Li, C.; Yuan, Y.; and Wang, G. 2022. FreeKD: Free-direction Knowledge Distillation for Graph Neural Networks. In *KDD*.

Freund, Y.; Schapire, R.; and Abe, N. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*.

Furlanello, T.; Lipton, Z.; Tschannen, M.; Itti, L.; and Anandkumar, A. 2018. Born again neural networks. In *ICML*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*.

Guo, Z.; Nan, B.; Tian, Y.; Wiest, O.; Zhang, C.; and Chawla, N. V. 2022a. Graph-based Molecular Representation Learning. *arXiv preprint arXiv:2207.04869*.

Guo, Z.; Shiao, W.; Zhang, S.; Liu, Y.; Chawla, N.; Shah, N.; and Zhao, T. 2022b. Linkless Link Prediction via Relational Distillation. *arXiv preprint arXiv:2210.05801*.

Guo, Z.; Zhang, C.; Yu, W.; Herr, J.; Wiest, O.; Jiang, M.; and Chawla, N. V. 2021. Few-shot graph learning for molecular property prediction. In *WWW*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *NeurIPS*.

Hansen, L. K.; and Salamon, P. 1990. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*.

Hastie, T.; Rosset, S.; Zhu, J.; and Zou, H. 2009. Multi-class adaboost. *Statistics and its Interface*.

Heo, B.; Kim, J.; Yun, S.; Park, H.; Kwak, N.; and Choi, J. Y. 2019. A comprehensive overhaul of feature distillation. In *ICCV*.

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

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Jin, W.; Liu, X.; Zhao, X.; Ma, Y.; Shah, N.; and Tang, J. 2021. Automated self-supervised learning for graphs. *arXiv preprint arXiv:2106.05470*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of neural network representations revisited. In *ICML*.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

Namata, G.; London, B.; Getoor, L.; Huang, B.; and Edu, U. 2012. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*.

Stanton, S.; Izmailov, P.; Kirichenko, P.; Alemi, A. A.; and Wilson, A. G. 2021. Does knowledge distillation really work? *NeurIPS*.

Sun, K.; Zhu, Z.; and Lin, Z. 2021. Adagcn: Adaboosting graph convolutional networks into deep models. In *ICLR*.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *NeurIPS*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR*.

Wang, K.; Liu, Y.; Ma, Q.; and Sheng, Q. Z. 2021. Mulde: Multi-teacher knowledge distillation for low-dimensional knowledge graph embeddings. In *WWW*.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *ICML*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Yan, B.; Wang, C.; Guo, G.; and Lou, Y. 2020. Tinygnn: Learning efficient graph neural networks. In *KDD*.

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 *WWW*.

Yang, Y.; Qiu, J.; Song, M.; Tao, D.; and Wang, X. 2020. Distilling knowledge from graph convolutional networks. In *CVPR*.

Yim, J.; Joo, D.; Bae, J.; and Kim, J. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*.

Yuan, L.; Tay, F. E.; Li, G.; Wang, T.; and Feng, J. 2019. Revisit knowledge distillation: a teacher-free framework.

Zhang, S.; Liu, Y.; Sun, Y.; and Shah, N. 2022. Graph-less neural networks: Teaching old mlps new tricks via distillation. In *ICLR*.

Zhang, W.; Miao, X.; Shao, Y.; Jiang, J.; Chen, L.; Ruas, O.; and Cui, B. 2020. Reliable data distillation on graph convolutional network. In *SIGMOD*.

Zhang, Z.; and Sabuncu, M. 2020. Self-distillation as instance-specific label smoothing. *NeurIPS*.

Zhao, L.; and Akoglu, L. 2019. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*.

Zheng, W.; Huang, E. W.; Rao, N.; Katariya, S.; Wang, Z.; and Subbian, K. 2022. Cold Brew: Distilling Graph Node Representations with Incomplete or Missing Neighborhoods. In *ICLR*.