Fine-Grained Learning Behavior-Oriented Knowledge Distillation for Graph Neural Networks

Kang Liu, Zhenhua Huang, *Senior Member, IEEE*, Chang-Dong Wang, *Senior Member, IEEE*, Beibei Gao, and Yunwen Chen, *Member, IEEE*

***Abstract*— Knowledge distillation (KD), as an effective compression technology, is used to reduce the resource consumption of graph neural networks (GNNs) and facilitate their deployment on resource-constrained devices. Numerous studies exist on GNN distillation, and however, the impacts of knowledge complexity and differences in learning behavior between teachers and students on distillation efficiency remain underexplored. We propose a KD method for fine-grained learning behavior (FLB), comprising two main components: feature knowledge decoupling (FKD) and teacher learning behavior guidance (TLBG). Specifically, FKD decouples the intermediate-layer features of the student network into two types: teacher-related features (TRFs) and downstream features (DFs), enhancing knowledge comprehension and learning efficiency by guiding the student to simultaneously focus on these features. TLBG maps the teacher model's learning behaviors to provide reliable guidance for correcting deviations in student learning. Extensive experiments across eight datasets and 12 baseline frameworks demonstrate that FLB significantly enhances the performance and robustness of student GNNs within the original framework.**

***Index Terms*— Feature knowledge decoupling (FKD), gradient correction, graph neural networks (GNNs), knowledge distillation (KD), learning behavior.**

I. Introduction

GRAPHS serve as a way for representing real-world data structures [1], [2], with nodes representing entities and edges representing their relationships. Mining graph data can yield valuable insights to help us understand complex dynamic relationships in various domains. For example, in social networks [3], graph data mining reveals community structures, influence patterns, and information or trend dissemination. In bioinformatics [4], [5], it aids in understanding protein interactions, genetic pathways, and disease spread. However, the inherent irregularity and disorder of graph data pose significant challenges to its mining and analysis. In response to these challenges, graph neural networks (GNNs) have emerged as a crucial innovation, effectively addressing the nonlinear and heterogeneous nature of graph structures [6], [7], [8], [9], [10], [11]. This has greatly promoted the development of the field of graph learning, marking a significant contribution to the development and application of graph-based analysis techniques [12], [13], [14], [15].

Despite their potential in academic research, GNNs rely on high-quality labeled data. To obtain these data, a more complex model and substantial computing resources are required, especially huge graph structures. Thus, deploying them on resource-limited devices (i.e., iPhones and cameras) [16] poses considerable obstacles.

In recent years, knowledge distillation (KD) [17] has emerged as an efficient method for model compression to solve the problem of deploying networks on resource-limited devices. Its core operation is to transfer "dark knowledge" from a larger, more complex network (named the teacher) to a smaller, compact one (named the student). The crucial objective is for the student to approach or even exceed the teacher's performance, reduce computational demands, and speed up inference time. As a result, many researchers [18], [19], [20] have integrated KD into GNNs to address challenges such as acquiring labeled data and deploying network architectures with limited computational resources.

In the existing studies, Yang et al. [21] are the pioneers of using KD for GNNs by designing the local structure preservation (LSP) method. This approach closely aligns a student's node structural feature with a teacher's node structural feature. Subsequent studies [22], [23], [24] broaden this focus to encompass local and global structures, capturing more complex graph structural information. For instance, He et al. [23] develop an adversarial KD framework (GraphAKD). This framework identifies differences between teacher and student outputs by preserving internode and interclass correlations. Wang et al. [24] introduce adversarial ideas into online KD and propose an innovative online adversarial distillation (OAD) framework. In addition, some studies also explore model augmentation approaches [25], [26], [27]. Feng et al. [25] propose a general and principled framework. This framework employs reinforcement learning to exchange node- and structure-level knowledge between two shallow GNNs. They effectively avoid requiring a deep and well-optimized teacher to distill knowledge. However, current learning frameworks primarily train students to absorb coarse-grained static knowledge [28]. This provides highly abstract information to students, hinders

Manuscript received 26 January 2024; revised 3 June 2024; accepted 26 June 2024. This work was supported in part by the Natural Science Foundation of China under Grant 62172166, Grant 61772366, and Grant 62276277; and in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515011380 and Grant 2022B1515120059. *(Corresponding authors: Beibei Gao; Zhenhua Huang.)*

Kang Liu and Zhenhua Huang are with the School of Computer Science, South China Normal University, Guangzhou 510631, China (e-mail: kang_liu_2019@126.com; jukiehuang@163.com).

Chang-Dong Wang is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510000, China (e-mail: changdongwang@hotmail.com).

Beibei Gao is with the School of Philosophy and Social Development, South China Normal University, Guangzhou 510631, China (e-mail: 20200659@m.scnu.edu.cn).

Yunwen Chen is with the Research and Development Department, DataGrand Inc., Shanghai 201203, China (e-mail: chenyunwen@datagrand.com).

Digital Object Identifier 10.1109/TNNLS.2024.3420895

their deep understanding of specific tasks, and impacts their learning efficiency. Furthermore, the prevalent transfer of static knowledge between the teacher and student presents two issues: 1) it overlooks the dynamic aspects of a teacher's knowledge comprehension and 2) it potentially diminishes a student's flexibility and adaptability in handling tasks.

To address these issues, we propose a fine-grained learning behavior (FLB)-oriented method, named FLB. FLB aims to deepen a student's knowledge comprehension and pay more attention to a teacher's learning behavior. It comprises two crucial components: feature knowledge decoupling (FKD) and teacher learning behavior guidance (TLBG). In KD, knowledge encompasses logit-based knowledge from the output layer and information-rich feature-based knowledge from the intermediate layer. Our FKD component focuses more on processing feature-based knowledge. Specifically, it decouples the student's feature knowledge into teacher-related features (TRFs) and downstream features (DFs). TRF pays more attention to learning from a teacher, while DF targets learning from downstream task labels. Meanwhile, we maintain the mutual independence of TRF and DF to avoid interference. FKD can deepen a student's knowledge comprehension and enhance learning efficiency by providing refined knowledge. TLBG designs a gradient network that maps teachers' learning behaviors, helping the students handle related tasks more effectively. The gradient network mainly captures a teacher's dynamic learning behavior by tracking gradient changes. Subsequently, it transfers this as potential "dark knowledge" to a student. This approach encourages the student to mimic a teacher's learning behavior during learning. The advantages of the gradient network include: 1) it focuses more on the teacher's gradient fluctuations and weight adjustment processes and guides a student to learn the teacher's decision-making logic and strategies and 2) it reveals changes and optimizations in the teacher's internal operations, thereby aligning the student closer to the teacher in terms of output accuracy. Consequently, the student can demonstrate improved flexibility and adaptability when encountering similar tasks.

In summary, FLB significantly improves a student's ability to grasp knowledge and effectively corrects a student's data handling and decision-making processes. This method dramatically enhances a student's flexibility and adaptability in similar tasks. In addition, the student achieves greater efficiency and accuracy in making predictions.

The main contributions of this article can be summarized as follows.

1) We propose an FLB method to boost the student's learning efficiency by providing more fine-grained knowledge and analyzing the teacher's learning behavior. This method also enhances the student's flexibility and adaptability in coping with tasks.
2) We propose an FKD component that separates the student's intermediate-layer feature knowledge into TRF and DF, ensuring their independence and noninterference. Compared to existing methods, this component offers the student more fine-grained knowledge to enhance the understanding of knowledge.
3) We propose a TLBG component to capture the teacher's learning behavior through gradient changes, transferring this as a form of "dark knowledge" for the student. This enables the student to adopt a learning behavior similar to that of the teacher. Unlike existing frameworks, TLBG focuses on the transfer of dynamic learning states between the teacher and the student. It can adjust the student's learning behavior by mapping the teacher's dynamic update process.
4) We evaluate the effectiveness of FLB, FKD, and TLBG within diverse KD frameworks based on GNNs. The experimental results show that our approach further improves the accuracy of existing GNN-based KD frameworks.

Section II introduces the related work to this article. Section III reviews the background of KD, GCN, and the definition of node classification. Section IV provides the details of the proposed method. Section V validates the performance of our proposed method on different datasets and frameworks. Section VI concludes this article.

II. Related Work

A. Graph Neural Networks

GNNs are initially proposed by Gori et al. [29] to process substantial amounts of graph-structured data encountered in practical applications [30]. Since then, GNNs have been widely used in various fields [31], [32]. Initially, GNNs mainly focus on effectively aggregating information from neighboring nodes to enhance node feature representations. They employ recursive network structures to update each node's state iteratively until stability is achieved. A significant milestone in the evolution of GNNs was achieved with the introduction of ChebNet. It utilizes Chebyshev polynomials to approximate spectral filters of the graph Laplacian operator, thereby performing convolution operations on graphs. Meanwhile, graph convolutional network (GCN) [33] designs a variant of a convolutional neural network (CNN) for semi-supervised learning on graph-structured data. GCN is particularly noted for its computational efficiency and adeptness in handling graph data, signifying a pivotal advancement in GNNs. Building upon this foundation, several enhanced models have emerged, including the graph attention network (GAT) [34], which introduces an attention mechanism allowing nodes to focus more on their neighbors' features. Similarly, GraphSAGE [35] extends the utility of GCN to large-scale graphs by employing local neighborhood sampling and feature aggregation techniques.

Furthermore, recent advancements in graph learning have also significantly deepened our comprehension and utilization of GNNs. For instance, Li et al. [12] propose the multimodal graph learning framework based on 3-D Haar semi-tight framelet transforms. This highlights the utility of capturing intermodal relationships and leveraging spectral-based graph structure learning to enhance prediction accuracy. PEGFAN [13] introduces a mathematical framework for extending traditional graph learning techniques to accommodate heterophilous graph structures. BLoG [14] offers a novel perspective on leveraging the local node and global graph information. This technique emphasizes balancing local feature extraction with global structural coherence. Huang et al. [15] investigate the feasibility of the GCNs with random weights in extracting meaningful graph representations. This method sheds light on the intrinsic capabilities of GCNs in capturing graph topology and feature distributions without extensive training.

B. Knowledge Distillation

As a model compression method, KD was first proposed by Hinton et al. [17]. Its main objective is to distill a complex high-performance teacher into a lightweight student while retaining comparable performance. This concept

has sparked extensive research [36], [37], [38], [39], [40]. In KD, knowledge is categorized into logit-based and feature-based. Logit-based knowledge [28], [41], [42] emphasizes label regularization to enhance the student model performance. For instance, DKD [28] divides the traditional KD process into two independent components: target class KD (TCKD) and nontarget class KD (NCKD). It addresses the efficiency and flexibility limitations of conventional methods. Li et al. [41] adopt a curriculum-based strategy to introduce a dynamic, learnable temperature parameter in the distillation process. Feature-based knowledge [43], [44], [45], [46] utilizes intermediate features from teachers, often outperforming logit-based methods. Among them, FitNets [43] first introduces intermediate-layer features for knowledge matching between the student and the teacher. AT [46] utilizes attention transfer, enabling students to learn from the teacher's attention maps. Park et al. [45] expand research beyond individual data samples by considering their structural relationships.

C. KD on GNNs

Numerous recent studies have integrated KD with GNNs [21], [47], [48], [49]. Among them, LSP [21] is the first distillation method tailored for GCN. Subsequently, combination of parameterized label propagation and feature transformation (CPF) [47] designs a student network using label propagation and feature transformation mechanisms to fully harness valuable prior knowledge. To eliminate data dependency, graph-less neural networks (GLNNs) [50] distills knowledge from GNNs into multi-layer perceptrons (MLPs), sparking a wave of GNN-to-MLP distillation models [51], [52], [53], [54], [55]. Wu et al. [51] propose a knowledge-inspired reliable distillation (KRD) framework, which evaluates the reliability of knowledge in GNNs by gauging the invariance of information entropy against noise perturbations. Full-frequency GNN-to-MLP (FF-G2M) [54] distinguishes low- and high-frequency knowledge from GNNs for injection into MLPs, enhancing the student model with multifaceted knowledge. Yang et al. [55] design a vector-quantized variational autoencoder (VQ-VAE) to tackle the scalability and latency challenges associated with GNNs.

Besides these model compression methods, KD for graph augmentation (KDGA) [56] tackles negative augmentation issues arising from distribution shifts between original and augmented graphs through homogeneity analysis. Guo et al. [57] transfer knowledge from various GNNs in an "enhanced" manner to boost traditional GNN performance. Yun et al. [18] distill the head and tail category student model to classify the related nodes. This effectively addresses category imbalance. To mitigate the adverse effects of incomplete information on the student model, two teachers based GNN (T2-GNN) [58] introduces distinct feature- and structure-level teachers to offer targeted guidance to students. To improve the fairness of GNNs without sacrificing utility, Zhu et al. [27] propose a FairGKD framework.

Although current KD methods applied to GNNs demonstrate considerable performance, they often involve coarse-grained static knowledge, which can pose challenges to the student's learning process. First, due to the highly generalized nature of the transferred knowledge, a student may not comprehend it in a way similar to that of a teacher. Second, static knowledge only reflects the current learning status of the network, which might not suffice for adapting to task changes.

To address the above issues, we propose an FLB method to refine the student's learning knowledge and correct its learning behavior. In FLB, the FKD component separates the student's feature knowledge into TRF and DF. It effectively addresses the issue of coarse-grained knowledge, which is challenging to comprehend. The TLBG component captures the teacher's learning behavior through observed gradient changes and imparts this understanding as potential "dark knowledge" to the student. It effectively alleviates the student's lack of adaptability and flexibility in coping with similar tasks.

III. Preliminaries

A. Notions and Problem Statement

For a given connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ representing the set of nodes and $\mathcal{E}$ representing the set of edges, the goal is to perform classification prediction on the unlabeled set of nodes $\mathcal{V}_U \subset \mathcal{V}$ in graph $\mathcal{G}$, i.e., the task of node classification on the graph.

B. Graph Neural Networks

In GNNs, a graph is typically represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathbf{X} \in \mathbb{R}^{n \times d}$ represents the d-dimensional features of n nodes. For a given node v, GNNs aggregate messages from its neighboring nodes $\mathcal{N}_v$. Then, GNNs use the message-passing paradigm to update the node v's embedding features h_v, which can be represented as

$$h_v^{l+1} = \mathrm{COM}\left(h_v^l, \left[\mathrm{AGG}\left(\left\{h_u^l | \forall u \in \mathcal{N}_v\right\}\right)\right]\right) \tag{1}$$

where h_v^l and h_v^{l+1} represent the embedding of the node v at the lth and $(l+1)$th layers, respectively. $\mathrm{COM}(\cdot)$ represents the combination function, and $\mathrm{AGG}(\cdot)$ represents the neighbor aggregation function. h_v^0 is initialized using node features $\mathcal{X}_v$. Furthermore, the embedding of the entire graph can be represented as

$$h_{\mathcal{G}}^l = \mathrm{READOUT}\left\{h_v^l | v \in \mathcal{V}\right\} \tag{2}$$

where READOUT is a graph-level pooling function.

C. Knowledge Distillation

Different from traditional methods, KD utilizes the outputs of the teacher to guide the student in learning, thereby obtaining a distillation loss defined as

$$\mathcal{L}_{\mathrm{KD}} = \mathcal{H}\left(\sigma_S\left(Z^S; \tau\right), \sigma_T\left(Z^T; \tau\right)\right) \tag{3}$$

where Z^S and Z^T represent the output distributions of the student and teacher, respectively. $\mathcal{H}(\cdot)$ is the Kullback–Leibler divergence (KL-divergence), and σ represents the softmax function with temperature τ. A higher τ (with a limit of $\tau \to \infty$) results in the output distribution being closer to the zero-mean distribution [17]. However, this may cause the student to be unable to capture effective information from the teacher, reducing the accuracy of the student's predicted probabilities. Conversely, a lower τ leads to a sharper output distribution. This potentially renders the student insensitive to a teacher's "dark knowledge," thereby limiting the student's performance. Generally, τ is set within the range of 1–10.

In addition, partial KD also utilizes the intermediate layers of information from teachers and students for matching, and the distillation loss typically uses $\mathcal{L}_{\mathrm{mse}}$ loss

$$\begin{aligned} \mathcal{L}_{\mathrm{mse}} &= \mathrm{MSE}\left(F^S, F^T\right) \\ &= \left\| f\left(F^S\right) - g\left(F^T\right) \right\|_2^2 \end{aligned} \tag{4}$$

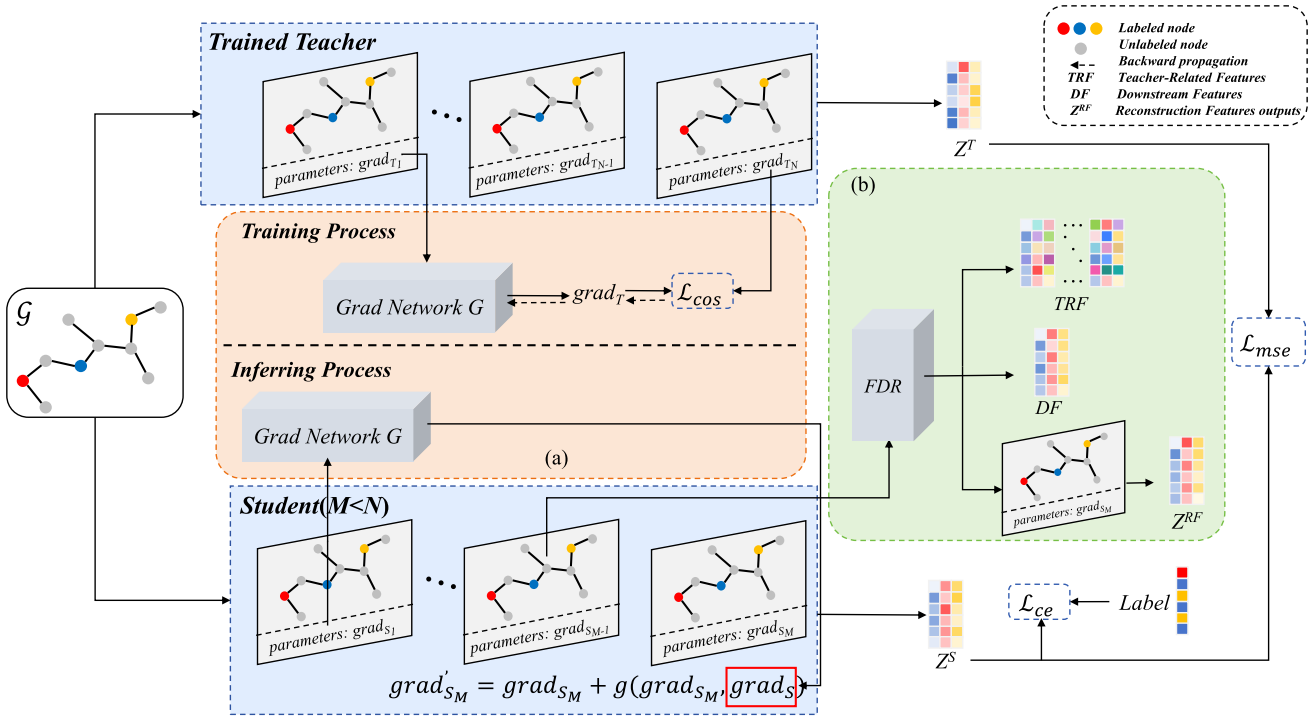

Fig. 1. Overview of the FLB method. We show the main components. (a) TLBG component designs a gradient network G to map the teacher learning process in the training process. In the inferring process, provide a pretrained G to the student to adjust its final layer gradients. (b) FKD component decouples and reconstructs the penultimate layer's features in the student, providing it with fine-grained features to deepen its knowledge comprehension.

where F^S and F^T represent the intermediate-layer features of the student and teacher, respectively. $f(\cdot)$ and $g(\cdot)$ are transformation functions used for dimension matching (e.g., linear transformation, convolution transformation, and interpolation transformation). $\|\cdot\|_2^2$ represents the square of the Euclidean norm.

IV. Methodology

In this section, we will elaborate on our proposed FLB-oriented KD approach for GNNs, namely, FLB. Specifically, we first provide an overview of the FLB methodology in Section IV-A. Then, we introduce the details of its core components in Sections IV-B and IV-C. Finally, the optimization objectives and training inference process of this method are detailed in Section IV-D.

A. Overview of FLB

In this section, we provide the overview of FLB, as illustrated in Fig. 1. The FLB methodology enhances the student performance through two aspects: 1) mimicking the dynamic learning process of the teacher to correct the student's learning trajectory and 2) refining coarse-grained knowledge to facilitate easier comprehension by the student. We model the two aspects as the TLBG component and the FKD component. For the TBLG component, as illustrated in Fig. 1(a), we devise an MLP-based gradient network, denoted as G. We pretrain G synchronously with the teacher to obtain the mapping of the teacher's learning process. This design ensures that the student can imitate the teacher's learning process while absorbing the teacher's knowledge. We will discuss the TLBG in Section IV-C. Fig. 1(b) illustrates that the FKD component incorporates a three-branch feature decoupling and reconstruction (FDR) network. This network's architecture is aligned with that of the student (e.g., GCN or MLPs) to ensure the rationality of our component. Each branch in the FDR is constrained by a corresponding loss function, with the design specifics elaborated in Fig. 2 and discussed in Section IV-B.

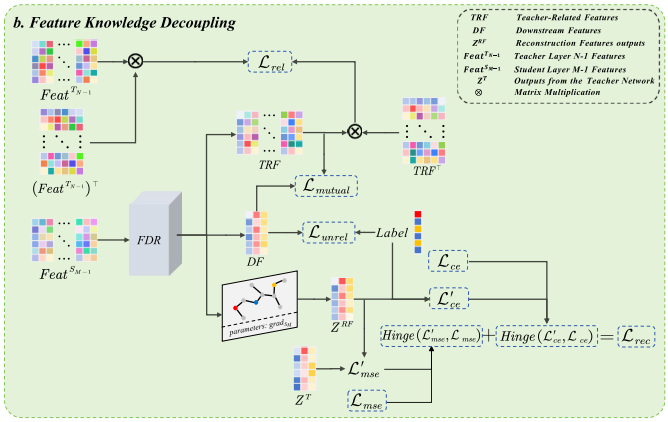

Fig. 2. Structure diagram of FKD.

B. Feature Knowledge Decoupling

In KD, "knowledge" is coarse-grained abstract information. Due to the relatively smaller scale of the student model, it may fail to absorb knowledge in the same manner as the teacher. To the best of our knowledge, in KD methods for CNN, DKD [28] has been designed to decouple the logit-based knowledge of the student. However, given that GNNs capture relational information between nodes, features from intermediate layers provide richer information than the final output of the network. With this motivation, we design the FKD component. This component decouples the student's intermediate-layer features (denoted as $\text{Feat}^{S_{M-1}}$, where M represents the number of layers in the student) into TRFs and DFs.

We compel the student to learn from both features individually and concurrently to enhance its performance. To clearly delineate this process, the FKD component is designed as a three-branch FDR network, aligning with the architecture of the student network. According to the functions of the three branches, we define them as the TRFs branch, the DFs branch, and the reconstruction features branch.

For the TRFs branch, we anticipate that it will extract information from the student's intermediate-layer features that are relevant to the teacher's intermediate-layer features, represented as TRF. To ensure that the TRF focuses solely on teacher features, we employ supervision using the teacher's intermediate-layer features, that is,

$$\mathcal{L}_{\text{rel}} = \text{MSE}\big(\text{TRF}, \text{Feat}^{T_{N-1}}\big) \tag{5}$$

where $\text{Feat}^{T_{N-1}}$ represents the $(N-1)$th layer features of the teacher and N represents the number of layers in the teacher.

However, given the disparity in network architecture design between the teacher and the student, we utilize the feature correlation matrix of TRF and $\text{Feat}^{T_{N-1}}$ to evaluate the gap between the two. Equation (5) is modified as

$$\mathcal{L}_{\text{rel}} = \text{MSE}\big(\text{matrix}_{\text{TRF}}, \text{matrix}_{\text{Feat}^{T_{N-1}}}\big) \tag{6}$$

where

$$\begin{aligned}\text{matrix}_{(A)} &= AA^T \\ &= \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{1m} & \cdots & a_{nm} \end{bmatrix}\end{aligned} \tag{7}$$

where A represents the feature information of TRF or $\text{Feat}^{T_{N-1}}$, n is the number of nodes, and m denotes the dimension of the network's hidden layers.

Similarly, the DF branch should extract DFs from the student's features, represented as DF. Here, we utilize downstream task labels to constrain DF, that is,

$$\mathcal{L}_{\text{unrel}} = \text{CE}(\text{DF}, y) \tag{8}$$

where $\text{CE}(\cdot)$ denotes the cross-entropy loss function and y signifies the labels for the downstream tasks.

Note that, to minimize the mutual impact of TRF and DF during the student training process, it is imperative to ensure their independence and lack of correlation. Consequently, we impose additional constraints on these features utilizing the distance correlation measurement method introduced in [59], i.e.,

$$\begin{aligned}\mathcal{L}_{\text{mutual}} &= \text{dcor}(\text{TRF}, \text{DF}) \\ &= \frac{v_n^2(\text{TRF}, \text{DF})}{\sqrt{v_n^2(\text{TRF}, \text{TRF}) \times v_n^2(\text{DF}, \text{DF})}}\end{aligned} \tag{9}$$

where $v_n^2(\cdot, \cdot)$ represents the product of two numbers before and after.

The purpose of the reconstruction feature branch is to ensure the effective decoupling of the first two branches. Specifically, we concatenate the decoupled TRF and DF along the column direction, denoted as Concatenate(TRF, DF). Then, we utilize the reconstruction feature branch to adjust its dimensions so that it aligns them with the original intermediate-layer feature dimensions of the student. We denote the output of the reconstruction feature branch as RF. Indeed, RF can be regarded as the new intermediate-layer features of the student, obtained after decoupling and reconstructing the student's intermediate-layer features.

Note that, we do not directly compare the differences between the student's original intermediate-layer features and the post-reconstructed intermediate-layer features. Instead, we supervise this branch by evaluating the changes in loss throughout the student's entire training process. In other words, we constrain the related loss of the post-reconstructed output results using the loss associated with the student's original output results. This is specifically represented as

$$\begin{aligned}\mathcal{L}_{\text{rec}} = &\text{Hinge}\big(\mathcal{L}'_{\text{ce}}\big(Z^{\text{RF}}, y\big), \mathcal{L}_{\text{ce}}\big(Z^S, y\big)\big) \\ &+ \text{Hinge}\big(\mathcal{L}'_{\text{mse}}\big(Z^{\text{RF}}, Z^T\big), \mathcal{L}_{\text{mse}}\big(Z^S, Z^T\big)\big)\end{aligned} \tag{10}$$

where Z^{RF} represents the output values obtained by inputting RF into the last layer of the student, i.e., $Z^{\text{RF}} = \text{Graphcovn}_{\theta \backsim S_M}(\text{RF})$, where $\text{Graphcovn}_{\theta \backsim S_M}(\cdot)$ represents the last layer of the student with M layers. $\mathcal{L}'$ represents the relevant loss from Z^{RF}, while $\mathcal{L}$ represents the relevant loss from Z^S. $\text{Hinge}(\mathcal{L}', \mathcal{L}) = \max(0, \zeta - (\mathcal{L}' - \mathcal{L}))$. ζ represents the evaluation boundary threshold. $\text{Hinge}(\mathcal{L}', \mathcal{L})$ is used to assess the disparity between Z^{RF} and Z^S, thereby further constraining the TRF and DF features. It ensures that the FKD component can effectively perform its decoupling function and provides the student with more fine-grained knowledge.

In summary, we define the following overall loss function for the FKD component:

$$L_{\text{feat}} = \mathcal{L}_{\text{rel}} + \mathcal{L}_{\text{unrel}} + \mathcal{L}_{\text{mutual}} + \mathcal{L}_{\text{rec}}. \tag{11}$$

C. Teacher Learning Behavior Guidance

The TLBG component is designed to capture the teacher's complex dynamic learning behavior and correct the student's learning process. We model the TLBG component as a network G based on MLPs, which has the same number of layers and the dimensions of hidden layers as the student. In the training phase, G is trained synchronously with the teacher to obtain a mapping of the teacher's learning process. Subsequently, in the inference phase, G is utilized to correct the gradients of the last layer of the student. Therefore, the student can learn the teacher's knowledge and imitate the teacher's learning process. Next, we provide the detailed descriptions of the training and inference phases of network G. The pseudocode of the TLBG is described in Algorithm 1.

1) Training Process: In this phase, we take the gradient from the first layer of the teacher network as input and use the gradient value from the last layer for supervision. This design aims to map the teacher's learning and updating process during each iteration, that is, the network's update process from the first layer to the last layer into the gradient network G. Specifically, we match the output values of the network G with the gradients of the last layer of the teacher network using a similarity function, which is

$$\mathcal{L}_S = 1 - \text{Similarity}\Big(\text{grad}, \text{grad}_{\theta_{T_N}}\Big) \tag{12}$$

where $\text{Similarity}(\cdot)$ denotes the similarity function, for which we employ cosine similarity. grad and $\text{grad}_{\theta_{T_N}}$ represent the gradients of G and the last layer of the teacher, respectively. By leveraging this approach, we save the teacher's learning behavior within G. Notably, our gradient network G can only reflect the changes from the first to the last layer in

Algorithm 1 TLBG Component

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Node Features $\mathbf{X}$, teacher network T with parameters $\theta_{\mathcal{T}}$, gradient network G with parameters $\theta_{\mathcal{G}}$.
Output: teacher network T, gradient network G

Initialize T and G
Generate subgraphs g_i and node features $\mathrm{X}_{\boldsymbol{g_i}}$ of subgraphs, where $g_1 \cup g_2 \cup \cdots \cup g_b = \mathcal{G}$ and $i = 1, 2, \ldots, b$.
while stop training condition not satisfied **do**
  if $grad_{\theta_{T_1}}$ from the first layer of T **then**
    Save the current gradient value as $state = \{grad_{\theta_{T_1}}\}$
  else
    $state$ is None.
  end if
  Input g_i and $\mathrm{X}_{\boldsymbol{g_i}}$ to T to obtain Z_T.
  Compute $\mathcal{L}_{ce}$
  Update $\theta_{\mathcal{T}}$ with $\frac{\partial \mathcal{L}_{ce}}{\partial \theta_T}$ and obtain $grad_{\theta_{T_1}}$, $grad_{\theta_{T_N}}$.
  if $state$ **then**
    Input $grad_{\theta_{T_1}}$ to G to obtain $grad$
    Compute $\mathcal{L}_s$ according to Eq. (12).
    Update $\theta_{\mathcal{G}}$ with $\frac{\partial \mathcal{L}_s}{\partial \theta_G}$
  end if
end while
return T and G

Algorithm 2 FLB Method

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Node Features $\mathbf{X}$, teacher network T, gradient network G, student network S with parameters $\theta_{\mathcal{S}}$, Feature Decoupling Reconstruction FDR.
Output: student network S with parameters $\theta_{\mathcal{S}}$

Initialize S and FDR
Generate subgraphs g_i and node features $\mathrm{X}_{\boldsymbol{g_i}}$ of subgraphs, where $g_1 \cup g_2 \cup \cdots \cup g_b = \mathcal{G}$ and $i = 1, 2, \ldots, b$.
while stop training condition not satisfied **do**
  if $grad_{\theta_{S_1}}$ from the first layer of S **then**
    Save the current gradient value as $state = \{grad_{\theta_{S_1}}\}$
  else
    $state$ is None.
  end if
  Input g_i and $\mathrm{X}_{\boldsymbol{g_i}}$ to S to obtain Z_S and intermediate layer features S_{feat}.
  Input g_i and $\mathrm{X}_{\boldsymbol{g_i}}$ to T to obtain Z_T and intermediate layer features T_{feat}.
  Input g_i and S_{feat} to FDR to obtain TRF, DF and RF.
  Compute L_{total} according to Eq. (14).
  Compute $grad_{\theta_{S_1}}$, $grad_{\theta_{S_M}}$ according to $\frac{\partial \mathcal{L}_{total}}{\partial \theta_S}$.
  if $state$ **then**
    Input $grad_{\theta_{S_1}}$ to G to obtain $grad'$
    Update $grad_{\theta_{S_M}}$ according to Eq. (13).
  end if
  Update $\theta_{\mathcal{S}}$
end while

the teacher network. We cannot guarantee that the output of network G will exhibit the same change results as those in the intermediate layers of the teacher network. In other words, we are solely considering the overall changes from input to output in the teacher network and their impact on the overall changes in the student.

2) Inferring Process: In this phase, following the same behavioral pattern as during the G training process, we use the gradient of the first layer of the student network as input into the pretrain G. Subsequently, we use the output of G as a correction value for the gradient of the last layer of the student network, denoted as grad′. Finally, we employ grad′ for a secondary update of the last layer of the student network, which is expressed as

$$\widehat{\mathrm{grad}}_{\theta_S} = \mathrm{grad}_{\theta_S} + \lambda F\left(\mathrm{grad}', \mathrm{grad}_{\theta_S}\right) \tag{13}$$

where $\widehat{\mathrm{grad}}_{\theta_S}$ represents the modified gradient, grad_{θ_S} is the original gradient of the student, λ is a scaling factor that measures how much corrective action needs to be applied, and $F(\cdot)$ is an adjustment factor for the student gradient. Rather than directly applying the modified gradient grad′, we quantify the discrepancy between grad′ and grad_{θ_S} in terms of a proportional relationship. This discrepancy is then accentuated via an exponential function, ensuring that its values are constrained between 0 and 1. The function $F(\cdot)$ is meticulously designed to facilitate the student's optimization of network parameters by imitating the teacher's learning behavior. Consequently, $F(\cdot)$ is formulated as $F(\mathrm{grad}', \mathrm{grad}_{\theta_S}) = \exp(\mathrm{grad}_{\theta_S}/\mathrm{grad}') - 1$.

D. Optimization Objective

The FLB method we propose not only enhances the learning efficiency of the student but also strengthens its comprehension ability, as well as its flexibility and adaptability to tasks. The optimization objective of this method is

$$L_{\mathrm{total}} = \alpha \mathcal{L}_{\mathrm{ce}} + (1-\alpha)(\beta \mathcal{L}_{\mathrm{mse}} + (1-\beta)\mathcal{L}_{\mathrm{feat}}) \tag{14}$$

where $\mathcal{L}_{\mathrm{ce}}$ represents the CE loss between the student and the downstream task labels and $\mathcal{L}_{\mathrm{mse}}$ denotes the mean-squared error loss between the student and the teacher. $\mathcal{L}_{\mathrm{feat}}$ reflects the sum of feature losses as shown in (11). Meanwhile, we show the pseudocode of the proposed FLB method in Algorithm 2.

The overall process of our proposed method can be described as follows. Initially, the teacher network T and gradient network G involved in the TLBG component are initialized and synchronously pretrained until convergence as described in Algorithm 1. Then, the pretrained T and G are utilized, in conjunction with FKD, to train student network S as outlined in Algorithm 2. Finally, only the trained S is used to predict task data in the inference phase.

V. Experiments

In this section, we first describe the experimental setup and integrate FLB as a plug-and-play method into different frameworks to compare its performance. This incorporation aims to demonstrate the generality of the FLB method. Next, we conduct extensive ablation studies to evaluate the effectiveness of the FKD and TLBG components. Then, we perform the sensitivity analysis on the hyperparameters involved in this article. In addition, we analyze the impact of teachers with different numbers of layers on student performance. Finally, we provide related analyses of the computation and time efficiency of the method to illustrate the reliability of the FLB method more effectively.

TABLE I
SUMMARY OF DATASETS

Dataset	Nodes	Edges	Features	Classes
Cora [60]	2708	5278	1433	7
Citeseer [61]	3327	4552	3703	6
Pubmed[1]	19717	44324	500	3
Co-cs [62]	18333	81894	6805	15
A-photo [63]	7487	119043	745	8
Cornell [64]	183	277	1703	5
Texas [64]	183	279	1703	5
Chameleon [65]	2277	3142	2325	5

TABLE II
NETWORK ARCHITECTURES IN KD. "-T" AND "-S" REPRESENT THE TEACHER AND STUDENT NETWORK ARCHITECTURE UNDER THE MODEL COMPRESSION FRAMEWORKS, RESPECTIVELY. "-TS" REPRESENTS THE SAME TEACHER–STUDENT NETWORK ARCHITECTURE UNDER THE MODEL AUGMENTATION FRAMEWORK. THE TOTAL SUM OF PARAMETERS IN THE STUDENT MODEL AND THE FDR NETWORK WITHIN THE FKD COMPONENT IS RECORDED IN PARENTHESES. "/" REPRESENTS THAT NO ATTENTION HEADS ARE UTILIZED

Dataset	Network	Layers	Hidden	Attention heads	Parameters
Cora	GCN-T	3	256,256,7	/	0.43M
	GAT-T	3	256,256,7	8,8,1	0.44M
	GraphSAGE-T	4	256,256,256,7	/	1.00M
	GCN-S	2	64,7	/	0.09M (0.11M)
	MLPs-S	2	64,7	/	0.09M (0.11M)
	GCN-TS	3	256,256,7	/	0.43M (0.69M)
Citeseer	GCN-T	3	512,512,6	/	2.16M
	GCN-S	2	64,6	/	0.24M (0.25M)
	MLPs-S	2	64,6	/	0.24M (0.25M)
	GCN-TS	3	512,512,6	/	2.16M (3.21M)
Pubmed	GCN-T	3	512,512,3	/	0.52M
	GCN-S	2	64,3	/	0.03M (0.04M)
	MLPs-S	2	64,3	/	0.03M (0.04M)
Co-cs	GCN-T	3	512,512,15	/	3.75M
	GCN-S	2	64,15	/	0.44M (0.45M)
	MLPs-S	2	64,15	/	0.44M (0.45M)
A-photo	GCN-T	4	512,512,512,8	/	1.82M
	GCN-S	2	64,8	/	0.05M (0.06M)
	MLPs-S	2	64,8	/	0.05M (0.06M)
Cornell	GCN-TS	3	256,256,5	/	0.50M (0.76M)
Texas	GCN-TS	3	256,256,5	/	0.50M (0.76M)
Chameleon	GCN-TS	3	256,256,5	/	0.66M (0.92M)

A. Experimental Setup

1) Datasets: In the field of GNNs, we conduct experiments on eight public datasets. The statistical details of datasets are presented in Table I.

2) Baseline Methods: By integrating the FLB method into the current state-of-the-art frameworks and comparing it with their original forms, we demonstrate its effectiveness. This involves a comparison with various model compression frameworks, such as KD [17], LSP [21], GraphAKD [23], OAD [24], CPF [47], NOSMOG [52], FF-G2M [54], KRD [51], and VQGraph [55]. In addition, we compare it with model augmentation frameworks such as KD [17], T2-GNN [58], KDGA [56], LTE4G [18], and FairGKD [27].

[1]https://pubmed.ncbi.nlm.nih.gov/

TABLE III
OPTIMAL HYPERPARAMETER VALUES FOR EACH DATASET

Architecture	Dataset	α	β	λ
GCN-to-GCN Model Compression	Cora	0.3	0.9	0.5
	Citeseer	0.9	0.2	0.3
	Pubmed	0.7	0.6	0.4
	Co-cs	0.9	0.2	1.0
	A-photo	0.9	0.5	1.0
GCN-to-GCN Model Augmentation	Cora	0.2	0.4	0.1
	Citeseer	0.1	0.6	0.3
	Cornell	0.9	0.5	0.5
	Texas	0.5	0.8	0.6
	Chameleon	0.8	0.1	0.6

3) Teacher–Student Network Architectures: Except for KD, our proposed FLB method follows the original teacher–student network architecture settings in the frameworks. For KD, we only select three classical GNN models as our teacher architectures on the Cora dataset: GCN, GAT, and GraphSAGE, while the teacher architectures for other datasets and all student architectures are consistently set as GCN. To compare the differences between the teacher and student network architectures, we have also enumerated the detailed information for each architecture, as shown in Table II.

4) Implementation Details: In this article, we implement relevant experiments in the PyTorch framework [66] and DGL library [67]. In Section IV-B, we retain the original experimental settings of all methods except for KD. For KD, we optimize the models using the Adam optimizer with a learning rate of 0.01 and a weight decay rate of $5e^{-4}$. For ζ involved in the Hinge$(\cdot)$ function in (10), we uniformly set it to 0.2. The output results are the average of five separate runs. Moreover, we use three evaluation metrics, namely, accuracy (Acc), area under the ROC curve (AUC), and balanced F score ($F1$), to measure the performance of the models under the baseline frameworks. For the hyperparameters in (13) and (14), we utilize grid search to determine the optimal values for each dataset, using Acc as the primary criterion. The optimal hyperparameter values for each dataset are displayed in Table III. The detailed experimental analysis is shown in Section IV-D. All experimental results presented in Tables IV–VII are obtained under identical environmental conditions.

B. Performance Comparison With Baseline

In this section, we integrate the FLB method into state-of-the-art baseline frameworks and conduct performance comparison experiments when the teacher architecture is GCN, GAT, or GraphSAGE, and the student architecture is GCN. For the teacher architecture of GCN, we conduct experiments on all baseline frameworks and datasets, as shown in Tables IV–VI. For the teacher architecture of GAT and GraphSAGE, we only conduct experiments on six baseline frameworks (KD, LSP, KRD, FF-G2M, KDGA, and T2-GNN) and the Cora dataset, as shown in Table VII.

1) Results With the Teacher Model as GCN: Tables IV–VI present the experimental results obtained by integrating the FLB method into the frameworks of model compression and augmentation. We have made the following observations.

a) Results of GCN-to-GCN model compression: In Table IV, we show the results of the teacher–student architecture with different hidden layer dimensions and different numbers of layers. Incorporating FLB has increased Acc, AUC, and $F1$ compared to the original results of the KD, LSP, GraphAKD, and OAD baseline frameworks. Notably,

TABLE IV

NODE CLASSIFICATION EVALUATION RESULTS (%) UNDER THE GCN-TO-GCN MODEL COMPRESSION FRAMEWORKS. "+FLB" INDICATES THAT IT IS ADDITIONALLY EQUIPPED WITH OUR METHOD. Δ REPRESENTS THE PERFORMANCE IMPROVEMENT AFTER ADDING THE FLB METHOD UNDER THE CURRENT FRAMEWORK

Dataset	Metric	KD [17]		Δ	LSP [21]		Δ	GraphAKD [23]		Δ	OAD [24]		Δ
			+FLB			+FLB			+FLB			+FLB	
Cora	Acc	80.95	81.65	**+0.70**	76.51	79.73	**+3.22**	83.76	84.58	**+0.82**	80.50	81.70	**+1.20**
	AUC	95.19	95.27	**+0.08**	84.30	86.61	**+2.31**	97.13	97.14	**+0.01**	95.50	96.06	**+0.56**
	F1	75.88	76.60	**+0.72**	64.68	65.76	**+1.08**	82.33	83.06	**+0.73**	79.80	80.50	**+0.70**
Citeseer	Acc	68.84	70.52	**+1.68**	59.11	61.83	**+2.72**	72.54	73.54	**+1.00**	65.60	69.90	**+4.30**
	AUC	88.64	88.92	**+0.28**	82.09	85.21	**+3.12**	89.66	89.77	**+0.11**	87.90	87.93	**+0.03**
	F1	63.44	65.70	**+2.26**	54.21	55.84	**+1.63**	68.26	69.12	**+0.86**	65.60	66.48	**+0.88**
Pubmed	Acc	77.22	78.20	**+0.98**	70.80	72.67	**+1.87**	80.88	81.60	**+0.72**	78.90	80.30	**+1.40**
	AUC	90.71	90.89	**+0.18**	82.47	84.02	**+1.55**	92.42	92.62	**+0.20**	90.70	91.29	**+0.59**
	F1	76.77	77.61	**+0.84**	69.70	72.13	**+2.43**	79.72	80.51	**+0.79**	78.50	79.69	**+1.19**
Co-cs	Acc	88.90	89.87	**+0.97**	81.49	83.53	**+2.04**	83.87	84.58	**+0.71**	89.43	91.03	**+1.60**
	AUC	98.28	98.49	**+0.21**	87.84	88.32	**+0.48**	97.76	97.84	**+0.08**	98.70	98.99	**+0.29**
	F1	84.90	85.86	**+0.96**	67.48	75.57	**+8.09**	81.37	82.47	**+1.10**	86.60	88.25	**+1.65**
A-photo	Acc	90.67	91.49	**+0.82**	85.96	87.30	**+1.34**	89.89	90.54	**+0.65**	90.36	90.90	**+0.54**
	AUC	98.84	98.94	**+0.10**	96.83	96.89	**+0.06**	98.67	98.68	**+0.01**	98.90	98.90	**+0.00**
	F1	88.51	89.40	**+0.89**	90.19	90.56	**+0.37**	87.93	88.57	**+0.64**	88.70	89.23	**+0.53**

TABLE V

NODE CLASSIFICATION EVALUATION RESULTS (%) UNDER THE GCN-TO-MLP MODEL COMPRESSION FRAMEWORKS. "+FLB" INDICATES THAT IT IS ADDITIONALLY EQUIPPED WITH OUR METHOD. Δ REPRESENTS THE PERFORMANCE IMPROVEMENT AFTER ADDING THE FLB METHOD UNDER THE CURRENT FRAMEWORK

Dataset	Metric	KD [17]		Δ	CPF [47]		Δ	KRD [51]		Δ	FF-G2M [54]		Δ	VQGraph [55]		Δ
			+FLB			+FLB			+FLB			+FLB			+FLB	
Cora	Acc	61.08	63.06	**+1.98**	84.37	84.82	**+0.45**	82.66	84.20	**+1.54**	83.02	85.02	**+2.00**	82.62	82.86	**+0.24**
	AUC	84.96	86.15	**+1.19**	97.45	97.77	**+0.32**	97.14	97.26	**+0.12**	97.09	97.28	**+0.19**	96.37	96.52	**+0.15**
	F1	55.41	57.13	**+1.72**	82.33	82.56	**+0.23**	82.78	83.45	**+0.67**	82.89	83.82	**+0.93**	79.80	80.27	**+0.47**
Citeseer	Acc	59.46	60.63	**+1.17**	72.43	76.30	**+3.87**	74.22	74.66	**+0.44**	73.36	74.20	**+0.84**	75.25	75.75	**+0.50**
	AUC	84.51	84.40	**-0.11**	89.75	90.16	**+0.41**	91.7	92.51	**+0.81**	91.21	91.29	**+0.08**	90.36	90.48	**+0.12**
	F1	55.51	56.16	**+0.65**	65.64	66.36	**+0.72**	71.54	72.08	**+0.54**	70.97	71.10	**+0.13**	68.33	69.35	**+1.02**
Pubmed	Acc	73.79	74.94	**+1.15**	79.63	80.15	**+0.52**	81.14	81.32	**+0.18**	79.54	80.36	**+0.82**	78.41	79.18	**+0.77**
	AUC	87.31	87.34	**+0.03**	93.86	93.93	**+0.07**	94.15	94.22	**+0.07**	93.38	93.48	**+0.10**	92.96	93.04	**+0.08**
	F1	73.59	75.06	**+1.47**	79.94	80.48	**+0.54**	81.83	82.6	**+0.77**	81.14	81.33	**+0.19**	78.40	79.32	**+0.92**
Co-cs	Acc	88.29	89.02	**+0.73**	91.67	92.19	**+0.52**	93.56	94.12	**+0.56**	92.86	93.98	**+1.12**	93.83	94.31	**+0.48**
	AUC	98.08	98.22	**+0.14**	99.24	99.26	**+0.02**	99.73	99.73	**+0.00**	99.67	99.70	**+0.03**	99.46	99.58	**+0.12**
	F1	82.84	83.72	**+0.88**	89.32	89.72	**+0.40**	91.07	91.83	**+0.76**	91.96	92.16	**+0.20**	91.65	93.58	**+1.93**
A-photo	Acc	78.76	79.47	**+0.71**	93.72	94.44	**+0.72**	92.30	93.81	**+1.51**	92.28	92.60	**+0.32**	91.51	92.60	**+1.09**
	AUC	96.92	97.31	**+0.39**	99.50	99.61	**+0.11**	99.19	99.48	**+0.29**	99.04	99.37	**+0.33**	98.88	99.13	**+0.25**
	F1	76.60	77.47	**+0.87**	92.30	92.83	**+0.53**	91.12	92.23	**+1.11**	92.36	92.57	**+0.21**	88.48	89.12	**+0.64**

adding FLB to the LSP framework demonstrates exceptional performance across all five datasets. Specifically, FLB achieves accuracy improvements of 2.04% and 8.09% for Acc and $F1$ on the Co-cs dataset, respectively. Meanwhile, we also observe that the AUC results show significant enhancement under the LSP framework. This illustrates that the FLB method is more effective in avoiding potential information loss caused by only considering local structures. This further reduces the error rate of the framework and improves its robustness.

Acc, AUC, and $F1$ values have also been improved for the KD, GraphAKD, and OAD frameworks. However, the increase varies across datasets (e.g., in the Citeseer dataset, OAD's Acc increased by 4.30%, while in the A-photo dataset, it only improved by 0.54%). This indicates that the performance of FLB may be influenced by the specific framework structure and dataset characteristics.

b) Results of GCN-to-MLPs model compression: We further apply the FLB method to five baseline frameworks of KD, CPF, KRD, FF-G2M, and VQGraph, utilizing the GCN-to-MLPs as the teacher–student architecture, as illustrated in Table V. Compared with the original framework, Acc, AUC, and $F1$ have improved after integrating the FLB method, especially for the KD framework. In contrast to the modest gains in AUC and $F1$, the notable increase in Acc suggests that the FLB method substantially bolsters these frameworks' positive and negative classification capabilities. Moreover, the slight enhancements in AUC and $F1$ indicate that the FLB method enhances the robustness of the five baseline frameworks, particularly in datasets with balanced distributions.

Further analysis of the data in Table V reveals that the FLB method performs exceptionally well on certain datasets. For example, in the Citeseer dataset, the Acc of the CPF

TABLE VI

NODE CLASSIFICATION EVALUATION RESULTS (%) UNDER THE GCN-TO-GCN MODEL AUGMENTATION FRAMEWORKS. "+FLB" INDICATES THAT IT IS ADDITIONALLY EQUIPPED WITH OUR METHOD. Δ REPRESENTS THE PERFORMANCE IMPROVEMENT AFTER ADDING THE FLB METHOD UNDER THE CURRENT FRAMEWORK

Dataset	Metric	KD [17]	+FLB	Δ	T2-GNN [58]	+FLB	Δ	KDGA [56]	+FLB	Δ	LTE4G [18]	+FLB	Δ	FairGKD [27]	+FLB	Δ
Cora	Acc	80.89	81.76	**+0.87**	83.36	85.23	**+1.87**	83.78	84.14	**+0.36**	82.70	83.03	**+0.33**	87.00	87.10	**+0.10**
	AUC	95.59	95.65	**+0.06**	96.82	97.12	**+0.30**	94.57	97.01	**+2.44**	92.20	92.88	**+0.68**	92.82	92.99	**+0.17**
	F1	74.98	75.74	**+0.76**	82.14	83.87	**+1.73**	66.69	83.57	**+16.88**	81.20	82.16	**+0.96**	46.52	47.31	**+0.79**
Citeseer	Acc	69.37	71.18	**+1.81**	70.50	73.26	**+2.76**	73.40	73.82	**+0.42**	69.90	70.78	**+0.88**	92.30	92.87	**+0.57**
	AUC	88.50	89.17	**+0.67**	89.90	91.13	**+1.23**	87.35	89.06	**+1.71**	81.60	82.74	**+1.14**	64.63	66.51	**+1.88**
	F1	64.91	65.92	**+1.01**	66.52	69.64	**+3.12**	67.64	68.43	**+0.79**	65.30	66.88	**+1.58**	48.00	48.93	**+0.93**
Cornell	Acc	59.46	61.08	**+1.62**	70.27	71.89	**+1.62**	61.08	61.74	**+0.66**	21.50	40.28	**+18.78**	86.49	89.19	**+2.70**
	AUC	69.95	70.86	**+0.91**	76.77	79.67	**+2.90**	66.79	72.73	**+5.94**	51.30	54.78	**+3.48**	65.05	74.73	**+9.68**
	F1	33.63	38.25	**+4.62**	49.94	52.90	**+2.96**	42.14	56.35	**+14.21**	15.10	26.84	**+11.74**	60.55	71.97	**+11.42**
Texas	Acc	56.76	58.91	**+2.15**	68.11	69.73	**+1.62**	66.47	67.09	**+0.62**	38.70	56.02	**+17.32**	67.57	75.68	**+8.11**
	AUC	42.94	62.29	**+19.35**	77.68	79.30	**+1.62**	67.55	76.13	**+8.58**	52.60	56.42	**+3.82**	38.71	48.39	**+9.68**
	F1	24.94	32.23	**+7.29**	43.16	46.60	**+3.44**	24.79	53.34	**+28.55**	20.40	32.48	**+12.08**	47.14	51.95	**+4.81**
Chameleon	Acc	57.02	57.93	**+0.91**	58.66	62.94	**+4.28**	61.23	61.77	**+0.54**	53.80	54.55	**+0.75**	80.04	80.26	**+0.22**
	AUC	79.93	80.55	**+0.62**	84.75	86.28	**+1.53**	84.19	84.99	**+0.80**	72.20	72.93	**+0.73**	78.14	78.16	**+0.02**
	F1	56.96	58.00	**+1.04**	58.41	62.81	**+4.40**	59.72	64.28	**+4.56**	52.90	53.21	**+0.31**	46.54	46.63	**+0.09**

framework increased by 3.87% after integrating the FLB method, while in the A-photo dataset, the Acc of the VQGraph framework increased by 1.09%. These results indicate that the FLB method not only improves the overall classification ability of the models but also shows significant advantages on specific datasets and frameworks.

c) Results of GCN-to-GCN model augmentation: In addition, incorporating the FLB method into five model augmentation frameworks (KD, T2-GNN, KDGA, LTE4G, and FairGKD) results in significant performance improvements under the Acc, AUC, and $F1$ evaluation criteria, as shown in Table VI. Notably, this improvement is particularly evident in smaller datasets, such as Cornell and Texas. This can be attributed to the issues with unbalanced data distribution in smaller datasets, which affect the models' data processing abilities. The FLB method can address this by providing the model with more fine-grained feature knowledge. This enriched feature information not only enhances the model's comprehension of feature knowledge but also alleviates the challenges brought by unbalanced data distribution. The consistent improvements across different datasets and metrics also underscore the method's effectiveness and flexibility.

2) Results on Cora With Different GNN Architectures: Table VII presents the performance of six baseline frameworks on the Cora dataset under different teacher–student architectures, including GAT-to-GCN, GraphSAGE-to-GCN, GAT-to-GAT, GraphSAGE-to-GraphSAGE, GAT-to-MLPs, and GraphSAGE-to-MLPs. We have reached the following conclusions.

a) Results of model compression experiments where the teacher architecture is GAT: By analyzing the experimental results in columns 2–4 of Table VII, we derive the following insights. When the teacher architecture is GAT, incorporating our FLB method to distill the heterogeneous student architectures significantly improves Acc, AUC, and $F1$ values. These improvements are especially notable in the LSP method. However, for distillation between homogeneous architectures, we observed a 0.10% decrease in Acc in the KDGA method after integrating FLB. We believe that this may be because GAT typically uses a multihead attention mechanism to stabilize the training process and enhance model performance.

TABLE VII

NODE CLASSIFICATION EVALUATION RESULTS (%) ON CORA, WHERE THE TEACHER UTILIZES GAT AND SAGE (ABBREVIATION FOR GRAPHSAGE) AND THE STUDENT UTILIZES GCN, GAT, SAGE, AND MLPS. "+FLB" INDICATES THAT IT IS ADDITIONALLY EQUIPPED WITH OUR METHOD. Δ REPRESENTS THE PERFORMANCE IMPROVEMENT AFTER ADDING THE FLB METHOD UNDER THE CURRENT FRAMEWORK

Framework	GAT-to-GCN			SAGE-to-GCN		
Metric	Acc	AUC	F1	Acc	AUC	F1
KD [17]	80.15	95.59	74.28	80.84	95.62	76.27
KD [17]+FLB	80.91	95.90	74.75	81.32	95.73	75.98
Δ	**+0.76**	**+0.31**	**+0.47**	**+0.48**	**+0.11**	**-0.29**
LSP [21]	74.78	82.73	60.08	73.84	82.60	58.26
LSP [21]+FLB	78.16	85.93	65.07	77.96	85.52	63.86
Δ	**+3.38**	**+3.20**	**+4.99**	**+4.12**	**+2.92**	**+5.60**
Framework	GAT-to-GAT			SAGE-to-SAGE		
Metric	Acc	AUC	F1	Acc	AUC	F1
KDGA [56]	81.60	95.29	77.97	84.52	94.24	67.86
KDGA [56]+FLB	81.50	95.83	80.63	85.20	97.24	83.83
Δ	**-0.10**	**+0.54**	**+2.66**	**+0.68**	**+3.00**	**+15.97**
T2-GNN [58]	84.31	96.92	82.09	84.71	95.75	83.18
T2-GNN [58]+FLB	85.71	97.31	83.91	86.12	97.02	85.92
Δ	**+1.40**	**+0.39**	**+1.82**	**+1.41**	**+1.27**	**+2.74**
Framework	GAT-to-MLPs			SAGE-to-MLPs		
Metric	Acc	AUC	F1	Acc	AUC	F1
KRD [51]	81.68	96.91	81.85	81.52	96.92	82.51
KRD [51]+FLB	83.10	96.95	82.22	83.00	96.64	82.89
Δ	**+1.42**	**+0.04**	**+0.37**	**+1.48**	**-0.28**	**+0.38**
FF-G2M [54]	83.18	97.38	83.30	83.82	97.26	83.57
FF-G2M [54]+FLB	84.80	97.38	83.40	85.20	97.34	83.63
Δ	**+1.62**	**+0.00**	**+0.10**	**+1.38**	**+0.08**	**+0.06**

* G represents a billion floating-point operations.

TABLE VIII

ABLATION STUDY ON MODEL COMPRESSION GCN-TO-GCN AND GCN-TO-MLPS [METRIC: ACC (%)]. Δ REPRESENTS THE PERFORMANCE IMPROVEMENT UTILIZING DIFFERENT COMPONENTS

Dataset	Model	TLBG	FKD	GCN-to-GCN		GCN-to-MLPs	
				Acc	Δ	Acc	Δ
Cora	Teacher			81.73	-	81.73	-
	Student			80.95	-	61.08	-
		✓		81.15	**+0.20**	61.65	**+0.57**
			✓	81.29	**+0.34**	62.80	**+1.72**
		✓	✓	**81.65**	**+0.70**	**63.06**	**+1.98**
Citeseer	Teacher			69.53	-	69.53	-
	Student			68.84	-	59.46	-
		✓		69.30	**+0.46**	59.54	**+0.08**
			✓	70.07	**+1.23**	60.51	**+1.05**
		✓	✓	**70.52**	**+1.68**	**60.63**	**+1.17**
Pubmed	Teacher			77.40	-	77.40	-
	Student			77.22	-	73.79	-
		✓		77.65	**+0.43**	74.00	**+0.21**
			✓	77.61	**+0.39**	74.09	**+0.30**
		✓	✓	**78.20**	**+0.98**	**74.94**	**+1.15**
Co-cs	Teacher			89.32	-	89.32	-
	Student			88.90	-	88.29	-
		✓		89.33	**+0.43**	88.73	**+0.45**
			✓	89.39	**+0.49**	88.88	**+0.59**
		✓	✓	**89.87**	**+0.97**	**89.02**	**+0.73**
A-photo	Teacher			90.86	-	90.86	-
	Student			90.67	-	78.76	-
		✓		90.80	**+0.13**	79.08	**+0.32**
			✓	90.90	**+0.23**	79.05	**+0.29**
		✓	✓	**91.49**	**+0.82**	**79.47**	**+0.71**

TABLE IX

ABLATION STUDY ON MODEL AUGMENTATION GCN-TO-GCN [METRIC: ACC (%)]. Δ REPRESENTS THE PERFORMANCE IMPROVEMENT UTILIZING DIFFERENT COMPONENTS

Dataset	Model	TLBG	FKD	Acc	Δ
Cora	Teacher			81.73	-
	Student			80.89	-
		✓		81.67	**+0.78**
			✓	81.30	**+0.41**
		✓	✓	**81.76**	**+0.87**
Citeseer	Teacher			69.53	-
	Student			69.37	-
		✓		70.47	**+1.10**
			✓	70.38	**+1.01**
		✓	✓	**71.18**	**+1.81**
Cornell	Teacher			60.38	-
	Student			59.46	-
		✓		60.54	**+1.08**
			✓	60.00	**+0.54**
		✓	✓	**61.08**	**+1.62**
Texas	Teacher			57.14	-
	Student			56.76	-
		✓		57.84	**+1.08**
			✓	57.84	**+1.08**
		✓	✓	**58.91**	**+2.15**
Chameleon	Teacher			57.44	-
	Student			57.02	-
		✓		57.78	**+0.76**
			✓	57.62	**+0.60**
		✓	✓	**57.93**	**+0.91**

The introduction of the FLB method may cause interference among different attention heads when processing features and labels, thereby affecting the overall model performance. This phenomenon is particularly evident when the attention heads perform well on augmented graphs but cannot collaborate effectively on the original graph. This indicates that incorporating the FLB method in the GAT framework requires more caution and may necessitate further parameter adjustments to balance the overall performance of the framework.

b) Results of model compression experiments where the teacher architecture is GraphSAGE: In columns 5–7 of Table VII, compared to heterogeneous teacher–student pairs, when the teacher–student architectures are GraphSAGE, incorporating the FLB method significantly improves the performance of the original framework in terms of Acc, AUC, and $F1$ evaluation metrics, especially in $F1$ (e.g., a 15.97% increase in the KDGA framework and a 2.74% increase in the T2-GNN framework). This improvement is attributed to the sampling mechanisms of GraphSAGE, which, when integrated with the FLB method, can more effectively utilize this method to enhance the feature information of the original framework, significantly improving the capture of feature layer information. Conversely, in the case of heterogeneous teacher–student pairs, although integrating the FLB method can improve the classification accuracy of the original framework on relatively balanced data, it may confuse positive and negative samples in imbalanced data, especially in the KRD framework. We believe that this may be due to GraphSAGE's heavy reliance on aggregating information from each node's neighbors. The introduction of FLB may alter how this aggregated information is represented or utilized in the MLPs, leading to mismatches or less effective use of contextual information, thereby reducing AUC. Therefore, when using GraphSAGE as the teacher in heterogeneous teacher–student pairs, further optimization of the FLB method is needed to avoid negative impacts.

C. Ablation Study

In the classical KD framework, we utilize the datasets in Table I and three teacher–student architectures to evaluate the two components of the FLB method: FKD and TLBG. The experiment aims to highlight the effectiveness of each component. The experimental results corresponding to these architectures and their respective datasets are systematically recorded in Tables VIII and IX. In addition, to further substantiate the soundness of the FLB method, we visualized the two components in Figs. 3 and 4.

1) Effectiveness of FKD on Student Performance: According to the results in Tables VIII and IX, it is evident that the FKD component has the potential to enhance the performance of students with different architectures. Note that the performance improvements of 0.49% and 1.08% are observed for larger datasets such as Co-cs and smaller datasets such as Texas, respectively. This is attributed to the decoupling ability of FKD, which decomposes the student's feature knowledge into TRF and DF, thereby enabling the student to assimilate more fine-grained knowledge. This approach effectively addresses the challenges posed by highly integrated knowledge. Furthermore, combined with the data in the last column of Table II, we find that, compared to the model augmentation architecture, the FKD component in the model compression architectures can bring a greater performance improvement to the student models at a lower space resources occupancy rate.

We visualize the heatmaps of the correlations between the original student features ($\text{Feat}^{S_{M-1}}$), TRF, DF, RF from the FKD component, and the teacher features ($\text{Feat}^{T_{N-1}}$) in Fig. 3. We observe that Fig. 3(b) correlates more with the teacher

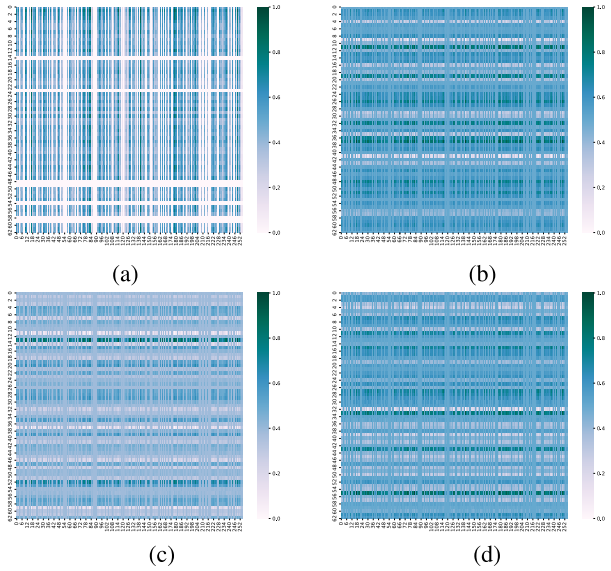

Fig. 3. Heatmap of the correlation between student and teacher features before and after decoupling on the Cora dataset. The vertical axis is the dimension of student-related features ($\text{Feat}^{S_{M-1}}$, TRF, DF, and RF). The horizontal axis is the dimension of teacher characteristics ($\text{Feat}^{T_{N-1}}$). (a) $\text{Feat}^{S_{M-1}}$ versus $\text{Feat}^{T_{N-1}}$. (b) TRF versus $\text{Feat}^{T_{N-1}}$. (c) DF versus $\text{Feat}^{T_{N-1}}$. (d) RF versus $\text{Feat}^{T_{N-1}}$.

features than the other three heatmaps. This indicates that the TRF obtained through the teacher-related branch effectively retains the teacher's feature information. Conversely, although Fig. 3(c) demonstrates a better correlation with the teacher features than Fig. 3(a), it still shows some discrepancy with the teacher features compared to Fig. 3(b). Meanwhile, Fig. 3(d) and (a) suggests that the RF, derived from coupling the TRF and DF through the reconstruction branch, can effectively learn different feature information from both the teacher and downstream tasks. This further demonstrates that the FKD component enhances the student's knowledge comprehension. We have also visualized the gradient changes in the student models with and without the FKD components in the first and third columns of Fig. 4. We find that FKD assists in correcting the gradient distribution during the optimization process of the student models.

2) Effectiveness of TLBG on Student Performance: Except for FKD, TLBG can also increase the prediction accuracy across various datasets. For instance, under the GCN-to-MLP teacher–student architecture, the student's Acc values improved by 0.57%, 0.08%, 0.21%, 0.45%, and 0.32% on the Cora, Citeseer, Pubmed, Co-cs, and A-photo datasets, respectively. In addition, an analysis of the gradient distributions in the first and second columns of Fig. 4 clearly shows that TLBG has effectively adjusted the gradient distribution of the student. This indicates that TLBG can correct the student's gradient distribution by mapping the dynamic changes in the teacher's gradients to enhance the flexible and adaptive learning process.

D. Hyperparameters Sensitivity Analysis

We also perform the experimental analysis to study the sensitivity of the three hyperparameters α, β, and λ involved in the FLB method. Specifically, we utilize GCN with varying numbers of layers and hidden layer dimensions as the teacher–student architecture for model compression on the Cora, Citeseer, Pubmed, Co-cs, and A-photo datasets. Using the Cora, Citeseer, Cornell, Texas, and Chameleon datasets, we adopt the teacher–student architectures with the same GCN structure for model augmentation. In these experiments, two hyperparameters are fixed to evaluate the sensitivity of the third hyperparameter. Fig. 5(a)–(c) illustrates the impact of the values of the three hyperparameters α, β, and λ on the classification accuracy of the student, respectively.

1) Impact of Hyperparameters on the Loss Function: In FLB, appropriate values for α and β are crucial to adjust the weight balance among the three types of loss. We vary α and β from 0.1 to 0.9 with a step size of 0.1 to investigate the effects of different weight combinations of the loss function on node classification. Fig. 5(a) and (b) shows the Acc results. For example, FLB achieves the optimal performance on the Texas dataset with $\alpha = 0.5$ and $\beta = 0.8$. In Fig. 5(a), the student's performance initially increases and then decreases with the α value. This indicates that a suitable weight balance adjustment between the first and the last two losses in (14) can enhance the classification accuracy. Fig. 5(b) illustrates that the student's performance follows a similar trend for a particular α value as β increases. It suggests that the network's performance can be optimized by balancing $\mathcal{L}_{\text{mse}}$ and $\mathcal{L}_{\text{feat}}$. Notably, when α and β reach optimal values, the loss function emphasizes feature loss more, aiding in capturing complex and deep data features.

Furthermore, in experiments, we observe that different datasets respond differently to the optimal values of α and β. For instance, the optimal α and β values differ between Cora and Texas datasets. This variation underscores that the loss function's optimal weight balance depends on the specific network structures and data characteristics. Consequently, achieving optimal performance requires careful parameter tuning for each specific application and dataset.

2) Impact of Hyperparameters on the Gradient Correction Effect: In FLB, λ in (13) is crucial to determine the gradient correction degree. We set the λ value within the range of 0.1–1.0, incrementing by 0.1, and the results are depicted in Fig. 5(c). The impact of λ varies between model compression and model augmentation datasets. In model compression, λ demonstrates relative insensitivity to network architectures with different hidden layer dimensions and layer numbers. This means that the student's performance remains relatively stable, even when appropriate gradient correction is applied. It is possibly because the network's design is inherently concise and efficient, limiting the impact of additional gradient correction on performance enhancement. Conversely, we observe more significant fluctuations in λ in model augmentation datasets, exhibiting distinct peaks and valleys. This variation suggests that gradient correction is more critical in this case. In model augmentation, the network is typically designed to be more complex and handle higher dimensional data. Therefore, appropriate gradient correction can significantly promote the learning process and avoid issues such as gradient vanishing or explosion, enhancing overall model performance.

In addition, different datasets exhibit varying sensitivities to λ, which may be related to their inherent characteristics and complexity. Some datasets may respond more to gradient adjustments, while others remain more robust against such corrections. Therefore, selecting an optimal λ value tailored to specific application scenarios and dataset properties is also critical for achieving optimal performance. Thus, the careful selection and adjustment of values, based on the model

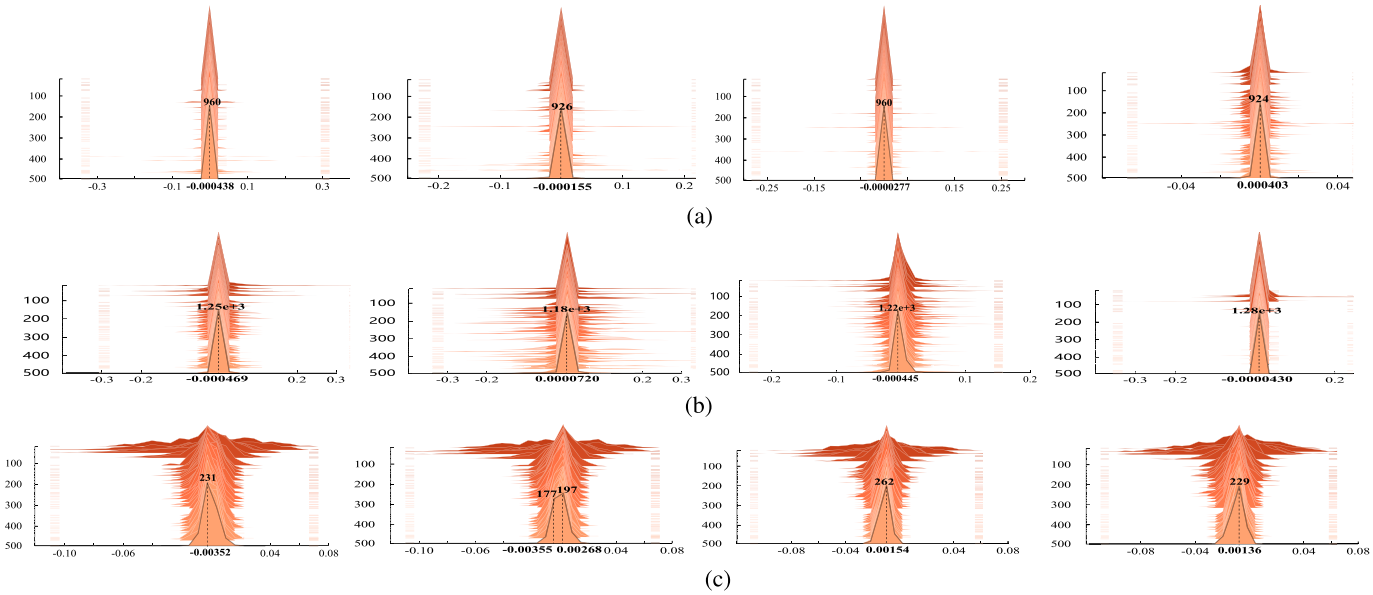

Fig. 4. Visualization of gradient changes. Sequences of (a)–(c), arranged from left to right, successively represent the student (only), student with TLBG, student with FKD, and student with FLB. (a) Model compression GCN-to-MLP and Co-cs dataset. (b) Model augmentation GCN-to-GCN and Texas dataset. (c) Model Compression GCN-to-GCN and Cora dataset.

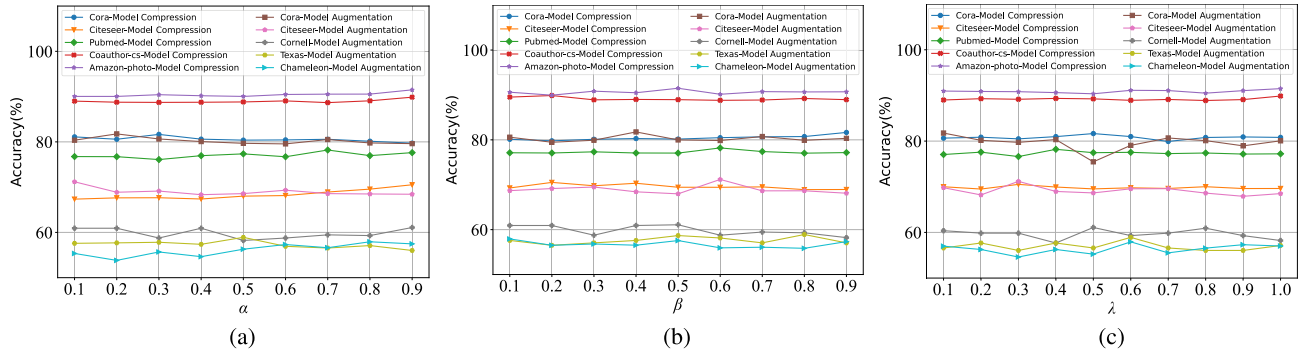

Fig. 5. Effect of hyperparameter values on student network performance on different datasets. (a) α values. (b) β values. (c) λ values.

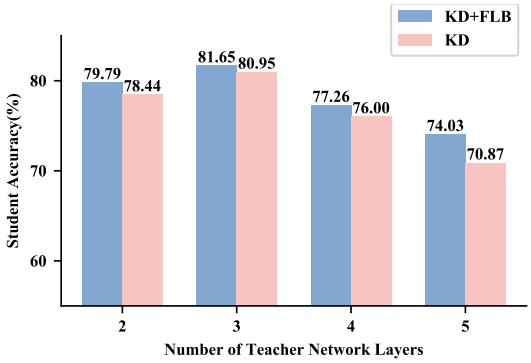

Fig. 6. Results of the teacher–student capacity gap on student performance based on GCN architecture, KD framework, and Cora dataset. The student model is consistently a two-layer GCN with a 64-D hidden layer. The hidden layer of the teacher model is set to 256 dimensions.

architecture and dataset characteristics, are essential to ensure training stability and optimal performance.

E. Teacher–Student Gap Analysis Based on GCN

In Fig. 6, we present the impact of increasing the complexity of the teacher network and the use of the FLB method on the student's prediction accuracy. Observations indicate that when GCN is used as the architecture for both teacher and student networks, a learning gap similar to that in CNNs exists between them. When the teacher network has three layers, the two-layer GCN student network achieves the optimal performance. This indicates that the complexity of the teacher network needs to match the learning capacity of the student network to effectively enhance its performance. In addition, we find that the FLB method can effectively mitigate the mismatch in learning capacity due to the difference in network sizes between the teacher and the student.

F. Computation and Time Efficiency Analysis

We also evaluate the computational and time efficiency of different frameworks equipped with our proposed FLB method, as shown in Table X. The floating-point operations (FLOPs) of most frameworks decrease by integrating the FLB method. This indicates that the FLB method is effective in reducing computational load. However, the training time varies across different teacher–student architectures. This variation might be due to the FLB method introducing additional processing of intermediate-layer features to the original framework, leading to extra time consumption.

TABLE X

RESULTS OF COMPUTATIONAL AND TIME EFFICIENCY DURING THE TRAINING PROCESS FOR NINE FRAMEWORKS UNDER THE CORA DATASET, BOTH WITH AND WITHOUT THE FLB METHOD. "FLOPS (G)" DENOTES THE NUMBER OF FLOATING-POINT OPERATIONS (IN BILLIONS) DURING THE TRAINING PROCESS. "IME(S)" INDICATES THE TOTAL TIME REQUIRED FOR TRAINING THE STUDENT MODEL. THE TEACHER–STUDENT MODELS FOR EACH FRAMEWORK ARE CONSISTENT WITH THOSE DESCRIBED IN TABLES IV–VI. DETAILED EXPERIMENTAL SETTINGS FOLLOW THE IMPLEMENTATION DESCRIBED IN SECTION V-A4. "+FLB" INDICATES THE INCLUSION OF OUR PROPOSED METHOD. ↑ AND ↓ DENOTE THE INCREASE AND DECREASE IN COMPUTATION AND TIME BEFORE AND AFTER EQUIPPING EACH FRAMEWORK WITH THE FLB METHOD, RESPECTIVELY

Metric	KD [17]		LSP [21]		GraphAKD [23]		CPF [47]		KRD [51]		FF-G2M [54]		T2-GNN [58]		KDGA [56]		LTE4G [18]	
		+FLB		+FLB		+FLB		+FLB		+FLB		+FLB		+FLB		+FLB		+FLB
FLOPs(G*)	4.3	3.3 (↓)	0.3	2.1 (↑)	82.7	79.7 (↓)	1.0	0.1 (↓)	93.0	77.0 (↓)	31.2	29.0 (↓)	15.3	5.5 (↓)	32.6	37.9 (↑)	0.05	0.01 (↓)
Time(s)	10.7	14.0 (↑)	507.7	74.0 (↓)	8.1	8.4 (↑)	34.2	74.1 (↑)	11.8	11.2 (↓)	14.1	9.1 (↓)	19.7	56.5 (↑)	33.5	27.4 (↓)	113.3	111.9 (↓)

* G represents a billion floating-point operations.

VI. CONCLUSION

In this article, we propose a novel KD method for GNNs, named FLB. This method can integrate into existing frameworks, enhancing their performance with fine-grained feature information and efficient learning behavior correction mechanisms. Specifically, FLB comprises two primary components: FKD and TLBG. The FKD component decouples the feature knowledge in the student into two fine-grained features: TRFs and DFs. This process allows the student to focus more on these two types of feature information separately, which deepens the student's comprehension of knowledge. Meanwhile, the TLBG component corrects the student's learning behavior by mapping the teacher's gradient changes. This promotes more efficient knowledge acquisition and enhances flexibility and adaptability to tasks. Extensive experiments confirm the effectiveness and robustness of the FLB method.

In future work, we aim to enhance the student's learning efficiency from two aspects. First, we will focus on the inheritance and extension relationships in the dynamic teacher–student learning processes. Second, we will investigate how to provide guidance from the teacher's knowledge for initializing the student. In addition, we plan to explore KD techniques that improve the student's generalization capabilities, enabling flexible handling of varied downstream tasks and different types of graph structures.

REFERENCES

[1] X. Li et al., "Sylvester equation induced collaborative representation learning for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 8811–8822, Oct. 2023.

[2] Z. Wang, J. Hu, G. Min, Z. Zhao, Z. Chang, and Z. Wang, "Spatial–temporal cellular traffic prediction for 5G and beyond: A graph neural networks-based approach," *IEEE Trans. Ind. Informat.*, vol. 19, no. 4, pp. 5722–5731, Apr. 2023.

[3] T. Konishi et al., "CG-GNN: A novel compiled graphs-based feature extraction method for enterprise social networks," in *Proc. 4th Workshop Intell. Cross-Data Anal. Retr.*, Jun. 2023, pp. 24–31.

[4] P. Coupeau, J.-B. Fasquel, J. Démas, L. Hertz-Pannier, and M. Dinomais, "Detecting cerebral palsy in neonatal stroke children: GNN-based detection considering the structural organization of basal ganglia," in *Proc. 20th Int. Symp. Biomed. Imag. (ISBI)*, 2023, pp. 1–4.

[5] Y. Xu et al., "Time-aware context-gated graph attention network for clinical risk prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 7557–7568, Jun. 2023.

[6] S. Xiong, B. Li, and S. Zhu, "DCGNN: A single-stage 3D object detection network based on density clustering and graph neural network," *Complex Intell. Syst.*, vol. 9, no. 3, pp. 3399–3408, Jun. 2023.

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

[8] J. Liu, Y. Chen, X. Huang, J. Li, and G. Min, "GNN-based long and short term preference modeling for next-location prediction," *Inf. Sci.*, vol. 629, pp. 1–14, Jun. 2023.

[9] W. Bi, L. Du, Q. Fu, Y. Wang, S. Han, and D. Zhang, "MM-GNN: Mix-moment graph neural network towards modeling neighborhood feature distribution," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Feb. 2023, pp. 132–140.

[10] L. Wei, H. Zhao, Z. He, and Q. Yao, "Neural architecture search for GNN-based graph classification," *ACM Trans. Inf. Syst.*, vol. 42, no. 1, pp. 1–29, Jan. 2024.

[11] S. Guo, T. Bai, and W. Deng, "Targeted shilling attacks on GNN-based recommender systems," in *Proc. ACM Int. Conf. Inf. Knowl. Manage (CIKM)*, 2023, pp. 649–658.

[12] M. Li, X. Zhuang, L. Bai, and W. Ding, "Multimodal graph learning based on 3D Haar semi-tight framelet for student engagement prediction," *Inf. Fusion*, vol. 105, May 2024, Art. no. 102224.

[13] J. Li, R. Zheng, H. Feng, M. Li, and X. Zhuang, "Permutation equivariant graph framelets for heterophilous graph learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2024, doi: 10.1109/TNNLS.2024.3370918.

[14] M. Li, L. Zhang, L. Cui, L. Bai, Z. Li, and X. Wu, "BLoG: Bootstrapped graph representation learning with local and global regularization for recommendation," *Pattern Recognit.*, vol. 144, Dec. 2023, Art. no. 109874.

[15] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu, "Are graph convolutional networks with random weights feasible?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2751–2768, Mar. 2023.

[16] Z. Huang et al., "Making accurate object detection at the edge: Review and new approach," *Artif. Intell. Rev.*, vol. 55, pp. 2245–2274, Sep. 2021.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[18] S. Yun, K. Kim, K. Yoon, and C. Park, "LTE4G: Long-tail experts for graph neural networks," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2022, pp. 2434–2443.

[19] C. K. Joshi, F. Liu, X. Xun, J. Lin, and C. S. Foo, "On representation knowledge distillation for graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4656–4667, Apr. 2024.

[20] C. Mai, Y. Chang, C. Chen, and Z. Zheng, "Enhanced scalable graph neural network via knowledge distillation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2024, doi: 10.1109/TNNLS.2023.3333846.

[21] Y. Yang, J. Qiu, M. Song, D. Tao, and X. Wang, "Distilling knowledge from graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Aug. 2020, pp. 7072–7081.

[22] Y. Tian, C. Zhang, Z. Guo, X. Zhang, and N. Chawla, "Learning MLPs on graphs: A unified view of effectiveness, robustness, and efficiency," in *Proc. ICLR*, 2023, pp. 1–12.

[23] H. He, J. Wang, Z. Zhang, and F. Wu, "Compressing deep graph neural networks via adversarial knowledge distillation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 534–544.

[24] C. Wang, Z. Wang, D. Chen, S. Zhou, Y. Feng, and C. Chen, "Online adversarial knowledge distillation for graph neural networks," *Exp. Syst. Appl.*, vol. 237, Mar. 2024, Art. no. 121671.

[25] K. Feng, C. Li, Y. Yuan, and G. Wang, "FreeKD: Free-direction knowledge distillation for graph neural networks," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 357–366.

[26] C. Wang et al., "Collaborative knowledge distillation for heterogeneous information network embedding," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 1631–1639.

[27] Y. Zhu, J. Li, L. Chen, and Z. Zheng, "The devil is in the data: Learning fair graph neural networks via partial knowledge distillation," in *Proc. ACM Int. Conf. Web Search Data Min.*, 2024, pp. 1012–1021.

[28] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Sep. 2022, pp. 11943–11952.

[29] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Mar. 2005, pp. 729–734.

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

[31] F. Guo, Y. Yuan, G. Wang, L. Chen, X. Lian, and Z. Wang, "Cohesive group nearest neighbor queries on road-social networks under multi-criteria," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 11, pp. 3520–3536, Nov. 2021.

[32] X. Guo et al., "A fast-response dynamic-static parallel attention GCN network for body–hand gesture recognition in HRI," *IEEE Trans. Ind. Electron.*, vol. 71, no. 6, pp. 5993–6004, Jun. 2024.

[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017, pp. 1–14.

[34] P. Velickovic et al., "Graph attention networks," in *Proc. ICLR*, 2018, pp. 1–12.

[35] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NeurIPS*, vol. 30, 2017, pp. 1024–1034.

[36] S. Kim and H. Kim, "Transferring knowledge to smaller network with class-distance loss," in *Proc. ICLR*, 2017, pp. 1–7.

[37] P. Passban, Y. Wu, M. Rezagholizadeh, and Q. Liu, "ALP-KD: Attention-based layer projection for knowledge distillation," in *Proc. 35th AAAI Conf. Artif. Intell. (AAAI)*, 2021, pp. 13657–13665.

[38] D. Chen et al., "Cross-layer distillation with semantic calibration," in *Proc. AAAI*, 2021, vol. 35, no. 8, pp. 7028–7036.

[39] S. Yang et al., "Learning from human educational wisdom: A student-centered knowledge distillation method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 6, pp. 4188–4205, Jun. 2024.

[40] Z. Huang, S. Yang, M. C. Zhou, Z. Li, Z. Gong, and Y. Chen, "Feature map distillation of thin nets for low-resolution object recognition," *IEEE Trans. Image Process.*, vol. 31, pp. 1364–1379, 2022.

[41] Z. Li et al., "Curriculum temperature for knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2023, vol. 37, no. 2, pp. 1504–1512.

[42] L. Xu, J. Ren, Z. Huang, W. Zheng, and Y. Chen, "Improving knowledge distillation via head and tail categories," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 5, pp. 3465–3480, May 2024.

[43] A. Romero et al., "FitNets: Hints for thin deep nets," in *Proc. ICLR*, 2015, pp. 1–13.

[44] P. Chen, S. Liu, H. Zhao, and J. Jia, "Distilling knowledge via knowledge review," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 5006–5015.

[45] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3967–3976.

[46] N. Komodakis and S. Zagoruyko, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. ICLR*, 2017, pp. 1–13.

[47] C. Yang, J. Liu, and C. Shi, "Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework," in *Proc. World Wide Web Conf. (WWW)*, 2021, pp. 1227–1237.

[48] X. Deng and Z. Zhang, "Graph-free knowledge distillation for graph neural networks," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 2321–2327.

[49] C. Yang, Q. Wu, and J. Yan, "Geometric knowledge distillation: Topology compression for graph neural networks," in *Proc. NeurIPS*, vol. 35, 2022, pp. 29761–29775.

[50] S. Zhang, Y. Liu, Y. Sun, and N. Shah, "Graph-less neural networks: Teaching old MLPs new tricks via distillation," in *Proc. ICLR*, 2022, pp. 1–21.

[51] L. Wu, H. Lin, Y. Huang, and S. Z. Li, "Quantifying the knowledge in GNNs for reliable distillation into MLPs," in *Proc. ICML*, 2023, vol. 202, no. 1564, pp. 37571–37581.

[52] Y. Tian, C. Zhang, Z. Guo, X. Zhang, and N. Chawla, "NOSMOG: Learning noise-robust and structure-aware MLPs on graphs," in *Proc. NeurIPS Workshop, New Frontiers Graph Learn.*, 2022, pp. 1–12.

[53] Y. Dong, B. Zhang, Y. Yuan, N. Zou, Q. Wang, and J. Li, "Reliant: Fair knowledge distillation for graph neural networks," in *Proc. Int. Conf. Data Mining (SDM)*, 2023, pp. 154–162.

[54] L. Wu, H. Lin, Y. Huang, T. Fan, and S. Z. Li, "Extracting low-/high-frequency knowledge from graph neural networks and injecting it into MLPs: An effective GNN-to-MLP distillation framework," in *Proc. AAAI Conf. Artif. Intell., AAAI*, 2023, no. 1163, pp. 10351–10360.

[55] L. Yang et al., "VQGraph: Rethinking graph representation space for bridging GNNs and MLPs," in *Proc. ICLR*, 2024, pp. 1–14.

[56] L. Wu, H. Lin, Y. Huang, and S. Z. Li, "Knowledge distillation improves graph structure augmentation for graph neural networks," in *Proc. NeurIPS*, vol. 35, 2022, pp. 11815–11827.

[57] Z. Guo, C. Zhang, Y. Fan, Y. Tian, C. Zhang, and N. V. Chawla, "Boosting graph neural networks via adaptive knowledge distillation," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2023, vol. 37, no. 6, pp. 7793–7801.

[58] C. Huo, D. Jin, Y. Li, D. He, Y.-B. Yang, and L. Wu, "T2-GNN: Graph neural networks for graphs with incomplete features and structure via teacher–student distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 4, pp. 4339–4346.

[59] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *Ann. Statist.*, vol. 35, no. 6, pp. 2769–2794, Dec. 2007.

[60] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Learning to classify text from labeled and unlabeled documents," in *Proc. Nat. Conf. Artif. Intell.*, 1998, pp. 792–799.

[61] C. L. Giles, K. D. Bollacker, and S. Lawrence, "CiteSeer: An automatic citation indexing system," in *Proc. 3rd ACM Conf. Digit. Libraries*, 1998, pp. 89–98.

[62] W. Hu et al., "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 22118–22133.

[63] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proc. KDD*, 2015, pp. 785–794.

[64] M. Craven et al., "Learning to extract symbolic knowledge from the world wide web," in *Proc. Nat. Conf. Artif. Intell.*, 1998, pp. 509–516.

[65] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[66] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, no. 721, 2019, pp. 8026–8037.

[67] M. Wang et al., "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2019, *arXiv:1909.01315*.

Kang Liu received the M.S. degree from the School of Information Engineering, Jingdezhen Ceramic University, Jingdezhen, Jiangxi, China, in 2022. She is currently pursuing the Ph.D. degree with the School of Computer Science, South China Normal University, Guangzhou, China.

Her research interests include knowledge distillation, graph representation learning, and efficient networks that can be used on resource-constrained edge computing devices.

Zhenhua Huang (Senior Member, IEEE) received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2008.

He is currently a Professor with the School of Computer Science, South China Normal University, Guangzhou, China. Since 2004, he has published three books and more than 80 papers in various journals and conference proceedings. His research interests mainly include deep learning, the Internet of Things, recommendation system, data mining, knowledge discovery, and big data.

Chang-Dong Wang (Senior Member, IEEE) received the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2013.

He was a Visiting Student at the University of Illinois at Chicago, Chicago, IL, USA, from January 2012 to November 2012. He joined Sun Yat-sen University in 2013, where he is currently an Associate Professor with the School of Computer Science and Engineering. He has published over 80 scientific papers in international journals and conferences, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, ACM Transactions on Knowledge Discovery from Data (ACM TKDD), ACM Transactions on Intelligent Systems and Technology (ACM TIST), IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART C (TSMC-C), ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Association for the Advancement of Artificial Intelligence (AAAI), International Joint Conference on Artificial Intelligence (IJCAI), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), International Conference on Data Mining (ICDM), ACM Conference on Information and Knowledge Management (CIKM), and SIAM International Conference on Data Mining (SDM). His current research interests include machine learning and data mining.

Dr. Wang's ICDM 2010 paper won the Honorable Mention for Best Research Paper Awards. He received the 2012 Microsoft Research Fellowship Nomination Award. He was awarded the 2015 Chinese Association for Artificial Intelligence (CAAI) Outstanding Dissertation. He was an Associate Editor of *Journal of Artificial Intelligence Research* (JAIR).

Beibei Gao received the M.S. and Ph.D. degrees in logic from South China Normal University, Guangzhou, China, in 2017 and 2020, respectively.

He is currently a Lecturer at the School of Philosophy and Social Development, South China Normal University. His research interests include logic and knowledge representation.

Yunwen Chen (Member, IEEE) received the Ph.D. degree from Fudan University, Shanghai, China, in 2008.

He had been the Chief Data Officer of Shanda Inc., Shanghai, the Senior Director of Tencent Inc., Shenzhen, China, and a Researcher with Baidu Inc., Beijing, China. He is the Founder and the CEO of DataGrand Inc., Shanghai, a leading artificial intelligence (AI) company in China. He has 32 patents and many publications. His research interests include data mining, natural language processing, deep learning, knowledge graph, and recommendation systems.

Dr. Chen is a member of Association for Computing Machinery (ACM) and a Senior Member of China Computer Federation (CCF).