AUGKD: Adaptive Universal Graph Knowledge Distillation

Junqi Liu[1], Yu Xie[1], Tianjun Ma[1], Wei Zheng[1], Jiangjiang Zhang[1]
[1]Key Laboratory of Computational Intelligence and Chinese Information Processing
of Ministry of Education, Shanxi University, Taiyuan 030006, China.

Graph data distillation frameworks have shown great potential in compressing graph representations and improving learning efficiency. Nevertheless, their robustness and generalization remain limited when applied to different graph neural network architectures. To address these limitations, we propose Adaptive Universal Graph Knowledge Distillation Framework (AUGKD). AUGKD adaptively enhances node representations by adapting to dataset heterogeneity and designing an importance-based node feature enhancement strategy. Then we introduce a label propagation–guided feature extraction module within the multilayer perceptron to mitigate over-smoothing, and employs a weight-adaptive mechanism that enables robust generalization to heterophilic graphs through adaptive and negative propagation coefficients. Experiments on multiple benchmark datasets demonstrate that AUGKD achieves robust and effective performance on both homophilic and heterophilic graphs benchmark datasets, significantly outperforming the baselines.

Index Terms—**Graph Data Distillation, Heterophilic Graphs, Adaptive Mechanisms, Graph Neural Networks.**

I. Introduction

GRAPH-STRUCTURED data is ubiquitous in real-world systems, spanning academic graphs [1], knowledge graphs [2], and social networks [3]. These graph structures play a central role in numerous applications, including complex system modeling, relationship analysis, and pattern recognition. Recently, deep learning methods have demonstrated remarkable effectiveness in analyzing graph data, achieving remarkable performance in tasks such as node classification [4], link prediction [5], [6], and recommendation systems [7], [8]. However, existing graph data distillation frameworks remain constrained by limited robustness and insufficient exploitation of the representational capacity of graph neural network frameworks.

Semi-supervised node classification is a core task in graph learning, where most existing methods rely on the homophily assumption that neighboring nodes tend to have similar attributes. Under this assumption, many methods have been developed [9]. In recent years, many deep learning-based classification methods have been proposed. Inspired by Chebyshev graph convolution, Kipf et al. [10] introduced a simpler variant of graph convolution called GCN, which opened up the application of GCN in the field of graphs. Petar et al. [11] proposed Graph Attention Networks (GAT), which addressed some drawbacks of graph convolution and its approximate methods by using masked self-attention layers. However, previous methods exhibit a low-pass filtering effect, where features become over-smoothed after propagating through a certain number of layers. Therefore, Zeng et al. [12] proposed MGIGNN, which retrieves globally similar nodes from a small candidate set and leverages memorized global information to improve graph neural network performance in both transductive and inductive settings. A-DropEdge [13] preserves key node connections and propagates information through multiple stochastic branches to enhance aggregation and capture complex relational dependencies. In contrast to the aforementioned approaches, Qian et al. [14] analyzed over-smoothing from an entropy-based perspective, showing that alleviating over-smoothing in high-entropy regions can significantly improve the discriminative capability of graphs. Most of these methods are designed under the assumption of homophily, which limits their applicability across diverse datasets and architectures.

However, many real-world graphs do not always exhibit homophily. In heterophilic scenarios, nodes with dissimilar labels are more likely to be connected, thereby challenging the assumptions that conventional graph neural network architectures are built upon [15] . The diversity of node and edge types, neighbor sampling biases, multimodal information, and issues such as data sparsity and label imbalance pose major challenges for conventional graph neural network frameworks on heterophilic graphs. To investigate the heterogeneous connectivity patterns among nodes in biological networks, Duan et al. [16] applied advanced graph neural network techniques to single-cell RNA sequencing (scRNA-seq) data, which demonstrates that methods explicitly designed to handle heterogeneity can effectively capture and interpret the complex patterns inherent in single-cell transcriptomic data. To further investigate heterogeneous relationships, EG-GCN [17] was proposed, which employs group-wise aggregation to distinguish between homophilic and heterophilic neighbors, while mitigating the limitations imposed by latent homophily. From a graph-centric perspective, TFE-GNN [18] was proposed, which adaptively extracts homophilic and heterophilic features from the graph according to varying degrees of homophily, while simultaneously leveraging the initial node features. These studies collectively aim to identify and differentiate homophilic and heterophilic nodes, thereby enhancing the understanding of structural heterogeneity in real-world graphs.

Knowledge distillation [19], [20], which can be effectively applied within deep learning frameworks, has made significant progress in recent years, further improving its effectiveness in graph representation learning. For example, Liu et al. [21] proposed DDK, a large language model distillation framework

Manuscript received January 30, 2026; revised April 10, 2026. Corresponding author: Yu Xie (email: sxlljcxy@gmail.com).

that dynamically adjusts the distillation dataset based on teacher-student performance discrepancies, improving stability and efficiency. AhmadKhan et al. [22] proposed HYDRA-FL, a hybrid knowledge distillation method that simultaneously enhances the robustness and accuracy of federated learning under adversarial conditions. Xu et al. [23] proposed Speculative Knowledge Distillation, which leverages an alternating sampling strategy to align teacher and student models. Meanwhile, a difficulty-aware knowledge distillation method, DA-KD, [24] is proposed, which dynamically adjusts the sample difficulty in the distillation dataset to improve the efficiency and stability of the distillation process. Collectively, these studies establish knowledge distillation as a principled framework for improving model efficiency and robustness, and for promoting consistent knowledge transfer across heterogeneous learning domains.

Existing graph data distillation frameworks exhibit suboptimal performance on heterophilic graphs. For graph condensation, these architectures are generally shallow, thereby hindering the modeling of global structural features. To address these limitations—particularly over-smoothing and poor adaptability to heterophilic graphs, we propose a universal adaptation-based graph knowledge distillation framework that transfers knowledge from a teacher to a student model while enriching it with additional structural and feature information. The framework incorporates three complementary components: (i) a feature extraction module integrating adaptive label propagation with multilayer perceptrons to mitigate over-smoothing, (ii) an importance-guided data augmentation mechanism within the distillation process to capture salient structural and feature information, and (iii) a weight adaptation mechanism to effectively model heterophilic relationships. We highlight the main contributions of this paper as follows:

- We propose a unified adaptation-based knowledge distillation framework that designs an importance-guided data augmentation strategy to effectively capture salient structural and feature information.
- A feature extraction framework is proposed in which adaptive label propagation guides multilayer perceptrons to alleviate over-smoothing in graph representations.
- We introduce a weight adaptation mechanism that captures heterophilic relationships by adaptively modulating inter-node dependencies in a context-aware manner.
- Extensive experiments demonstrate the effectiveness of the proposed framework in transferring teacher knowledge while augmenting structural and feature information.

A. Basic Notation

Suppose $G = (V, E)$ is an undirected graph composed of a set of nodes $V = \{v_1, v_2, \cdots, v_N\}$ and a set of edges E, with N denoting the number of nodes. $X \in \mathbb{R}^{N \times F}$ is the node feature matrix associated with the nodes, $H \in \mathbb{R}^{N \times F'}$ represents the hidden layer representation, $Z \in \mathbb{R}^{N \times C}$ is the final embedding representation, $P \in \mathbb{R}^{N \times C}$ is the label prediction, $P = softmax(Z)$, C is the number of node categories, F and F' denote the feature dimensions. $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix, where $a_{ij} = 1$ indicates the existence of a connection between nodes v_i and v_j, and vice versa. In an undirected graph $a_{ij} = a_{ji}$. $\tilde{A}$ is the adjacency matrix added with self-loops, and $\tilde{D}$ is the diagonal matrix of $\tilde{A}$, $\tilde{D}_{(i,i)} = \sum_j \tilde{A}_{(i,j)}$. $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ represents the symmetrically normalized adjacency matrix with self-loops, while another type is the simple normalized matrix $\hat{A} = \tilde{D}^{-1}\tilde{A}$.

B. Graph Neural Networks

In general, most existing graph neural networks operate under the message-passing framework, in which node representations are iteratively updated by aggregating information from their local neighborhoods. Given their ability to model complex structural dependencies, graph neural networks have become a central component of many graph data distillation frameworks. Methods for learning node embeddings in these networks are typically formulated within this paradigm:

$$\begin{aligned} a_i^{(k)} &= \text{PROPAGATION}^{(k)}\big(\{h_i^{(k-1)}, \{h_j^{(k-1)} \mid j \in \mathcal{N}_i\}\}\big), \\ h_i^{(k)} &= \text{TRANSFORMATION}^{(k)}\Big(a_i^{(k)}\Big). \end{aligned} \tag{1}$$

Here, k denotes the number of propagation layers; h_i and h_j represent the hidden embeddings of the central node i and its neighbors, respectively; $\mathcal{N}_i$ is the neighborhood of node i, and the initial representation $h_i^{(0)}$ is initialized with the input feature X_i. The function PROPAGATION$(\cdot)$ aggregates information from nodes neighborhood, propagating source nodes features along the edges toward target nodes. TRANSFORMATION$(\cdot)$ maps the aggregated information into updated node representations. In conventional graph neural networks, the normalized adjacency matrix $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is typically employed as the propagation coefficient, which depends solely on the node degrees. However, such fixed coefficients often constrain the representational capacity of node features and impede their effective differentiation, ultimately leading to over-smoothing in deep graph neural networks.

Homophily Definition. In this work, we quantify the degree of homophily in a graph using the edge homophily ratio, which measures the proportion of edges connecting nodes that share the same class label. Formally, it is defined as: $\mathcal{H} = \frac{|(v_i,v_j):a_{ij}=1 \wedge y_i=y_j|}{|m|}$, where m denotes the set of all edges. A higher value of $\mathcal{H}$ (i.e., closer to 1) indicates that most edges connect nodes within the same class, reflecting strong homophily, whereas a lower value (closer to 0) implies that edges predominantly link nodes from different classes, corresponding to heterophilic structures.

Data Distillation. Building on the structural principles of homophily and heterophily, we focus on data distillation, a framework that facilitates efficient knowledge transfer between models. Specifically, knowledge distillation transfers information from a large, well-trained *teacher* model to a compact *student* model, enabling the student to achieve comparable performance while reducing computational and memory overhead [25]. This process serves two main objectives: compressing models for resource-constrained environments and enhancing

performance through guided learning. The distillation objective is formulated as $\mathcal{L}_{kd} = \mathrm{DIV}(k^T, k^S)$, where $\mathrm{DIV}(\cdot)$ denotes the distance between teacher and student representations, which can be measured using divergence loss or Euclidean distance. Here, k^T and k^S represent the knowledge of the teacher and student, which may correspond to hidden layer embeddings (Z^T, Z^S) or probability distributions over classes (P^T, P^S). Through this process, the student inherits the representational capacity of the teacher, achieving a lightweight model without sacrificing performance.

Label Propagation. Beyond knowledge transfer via data distillation, label propagation represents a classical method in semi-supervised graph learning. Grounded in the homophily principle, which posits that adjacent nodes are likely to share the same label, this algorithm iteratively propagates labels from labeled nodes to their unlabeled neighbors to generate predictions. Formally, let $f_{LP}(v)$ denote the final prediction of node v, and $f_{LP}^k(v)$ represent the prediction at the k-th propagation step. Here, C denotes the number of node classes, and V_L and V_U denote the sets of labeled and unlabeled nodes, respectively. The initial label assignments are given by:

$$f_{LP}^0(v) = \begin{cases} (0, \ldots, 1, 0, \ldots, 0) \in \mathbb{R}^C, & \forall v \in V_L. \\ \left(\frac{1}{C}, \ldots, \frac{1}{C}\right) \in \mathbb{R}^C, & \forall v \in V_U. \end{cases} \tag{2}$$

During propagation, nodes with labels are initialized as one-hot vectors, whereas unlabeled nodes are initialized with a uniform distribution over classes, reflecting equal initial probabilities. The label update rule at the k-th propagation step is formulated as:

$$f_{LP}^k(v) = (1-\alpha)\,\frac{1}{|\mathcal{N}_v|}\sum_{u\in\mathcal{N}_v} f_{LP}^{k-1}(u) + \alpha\, f_{LP}^{k-1}(v). \tag{3}$$

where $\alpha \in [0,1]$ balances the contribution of the central node and its neighbors, and $\mathcal{N}_v$ denotes the neighborhood of node v. Being parameter-free, label propagation is computationally simpler than graph neural networks, yet Eq. (3) shows that it performs neighborhood aggregation analogous to message passing in graph neural networks, albeit without trainable parameters. Although label propagation serves as a lightweight approach, its capacity to model rich structural patterns is limited. To overcome this limitation, we introduce an adaptive mechanism that enhances structural feature modeling in the student network.

Data Augmentation. Beyond label propagation and knowledge distillation, data augmentation serves as a complementary approach to enhance model generalization. The core idea is to generate additional samples by transforming the original training dataset, thereby expanding the data distribution and improving predictive performance. In graph neural networks, data augmentation can be broadly categorized into feature augmentation [26] , formalized as:

$$\begin{aligned} p(Y \mid X, A) &= \sum_{\hat{A}\in[0,1]^{N\times N}} p(Y \mid X, \hat{A})\, p(\hat{A} \mid X, A). \\ p(Y \mid X, A) &= \sum_{\hat{X}\in[0,1]^{N\times F}} p(Y \mid \hat{X}, A)\, p(\hat{X} \mid X, A). \end{aligned} \tag{4}$$

Here, X and A denote the input node features and adjacency matrix, respectively. Y is the predicted label, $\hat{X}$ represents augmented features, and $\hat{A}$ represents the augmented adjacency matrix. Structural augmentation typically involves edge deletion or addition, which may introduce distributional shifts between the original and augmented graphs, potentially causing negative enhancement. To mitigate this, importance-based augmentation [27] has received increasing attention.

II. Methodology

This section is organized as follows. We begin by introducing the overall framework of graph knowledge distillation and outlining its complete workflow. Next, we detail the importance-based feature enhancement strategy and the student architecture that integrates adaptive label propagation with a two-layer multilayer perceptron. Finally, we present the pseudocode and analyze the computational complexity of the proposed framework. The overall structure of AUGKD is illustrated in Figure 1.

A. Graph Knowledge Distillation Framework

Graph neural networks can be viewed as black-box models that take a graph structure G and a labeled node set V_L as input and learn a mapping function f that outputs class probability distributions. For a given node $v \in V$ and class $y \in Y$, the model estimates a probability $f(v, y)$ such that $\sum_{y\in Y} f(v, y) = 1$. For labeled nodes, the probability corresponding to their ground-truth class is set to $f(v, y) = 1$, while $f(v, y') = 0$ for all other classes. For convenience, we denote the complete probability vector across all classes as $f(v) \in \mathbb{R}^{|Y|}$. In our framework, the teacher model can be any graph neural networks architecture, producing a probability distribution $f_{\mathrm{TEA}}(v)$ for each node v. The student model, parameterized by Θ, outputs its own distribution $f_{\mathrm{STU};\Theta}(v)$. The objective of knowledge distillation is to minimize the discrepancy between the student's soft predictions and those of the teacher, thereby transferring the knowledge embedded in the teacher model to the student. Formally, the optimization target can be expressed as:

$$\min_{\Theta} \sum_{v\in V} D_{KL}(f_{\mathrm{TEA}}(v),\, f_{\mathrm{STU};\Theta}(v))\,. \tag{5}$$

The $D_{KL}(\cdot)$ denotes the distance between two probability distributions, such as Kullback–Leibler divergence or cross-entropy. In this work, Euclidean distance is adopted as the distillation metric.

B. Importance-based Node Feature Augmentation

Feature-level augmentation perturbs node representations by randomly masking feature dimensions. However, uncontrolled randomness may suppress critical features, causing unstable training and weakened generalization. To mitigate this, importance-guided data augmentation retains structurally and semantically salient components while perturbing less informative ones, thereby improving the fidelity of the augmented representations [27] . We develop an importance-aware

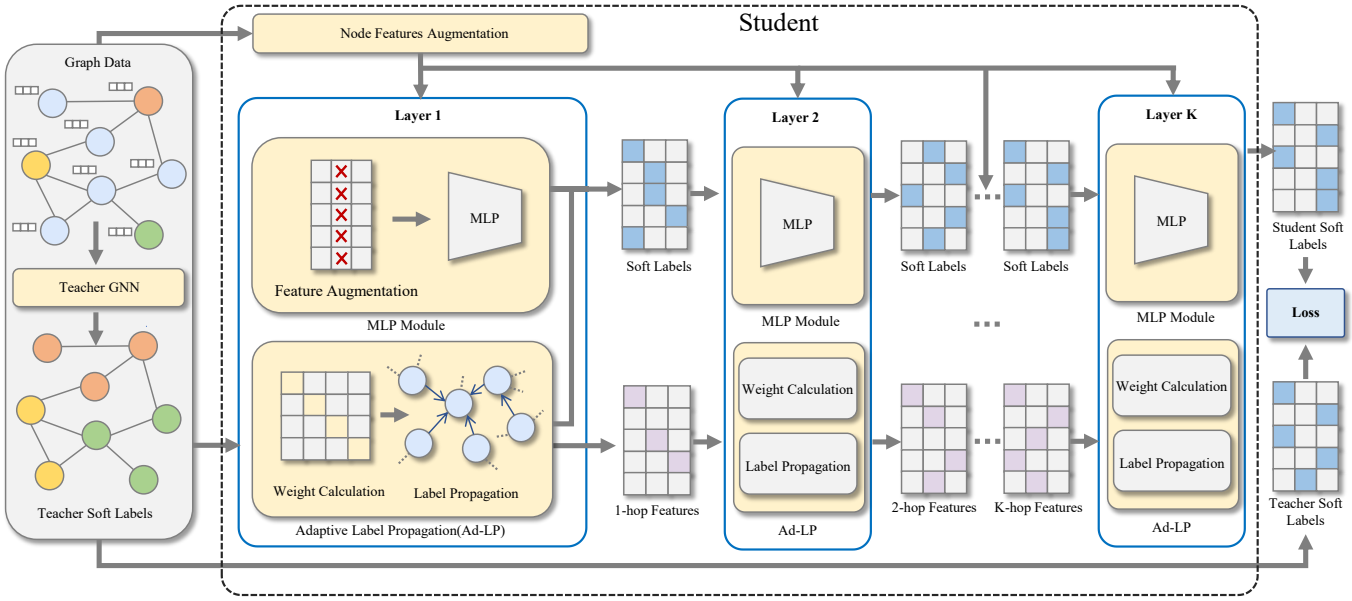


Fig. 1: Overall architectural framework of the proposed AUGKD model, illustrating the complete process of Teacher module, adaptive label propagation, features augmentation, and student module that collectively define its structural design.

strategy that accounts for the intrinsic structural characteristics of graphs. Feature importance manifests differently in homophilic and heterophilic settings, distinct formulations are employed for each type. For homophilic graphs, where local neighborhood information dominates, the importance of the i-th feature dimension is defined as $w_i^f = \log(\sum_{v \in V} x_{vi}^f d_v)$ which implicitly incorporates the influence of node degrees d_v within the local neighborhood. Conversely, for heterophilic graphs, where global consistency is more significant and neighborhood influence should be minimized, the importance is defined as $w_i^f = \log(\sum_{v \in V} x_{vi}^f)$. Here, x_{vi}^f denotes the i-th feature component of node v, and d_v represents the degree of node v. This adaptive design enables the augmentation process to selectively perturb less informative dimensions while preserving key discriminative information.

The deletion probability of the i-th feature component, denoted as p_i^f, is defined as: $p_i^f = \min\left(\frac{w_{\max}^f - w_i^f}{w_{\max}^f - w_{\text{mean}}^f} \cdot p_f, p_\tau\right)$, where $w_{\max}^f$ and w_{mean}^f denote the maximum and mean importance values across all feature dimensions, respectively. The hyperparameter p_f controls the overall average deletion rate, while p_τ specifies its upper bound. As evident from the definition, p_i^f is inversely proportional to the importance score w_i^f: the higher the feature importance, the lower its deletion probability. A binary mask $\varepsilon_i^f \sim Bernoulli(1 - p_i^f)$ is then sampled for each feature component to determine whether it should be retained or masked out. Accordingly, the augmented feature representation is obtained as $\hat{X}_v = X_v \odot \varepsilon^f$, where $\odot$ denotes the Hadamard product. This mechanism ensures that salient features are preserved while less informative components are selectively suppressed, enabling more robust and semantically consistent feature augmentation.

C. Student Model Architecture

The proposed student network consists of multiple layers, each coupling an adaptive label propagation module with a multilayer perceptron through trainable fusion parameters. Unlike conventional graph neural networks that rely on deep and highly nonlinear architectures, our design prioritizes computational efficiency without compromising expressive power. To further alleviate information loss during deep feature extraction, the student model incorporates the teacher's soft predictions as auxiliary inputs. By jointly leveraging these soft targets with the original graph topology and node attributes, the model effectively recovers both node-level and structural cues that are often weakened in the teacher's latent representations.

Adaptive Label Propagation Module. Conventional label propagation, being parameter-free, lacks the flexibility to capture the intricate structural dependencies inherent in graph data. Furthermore, it assigns equal importance to all neighboring nodes, neglecting their varying degrees of influence on the central node. To alleviate these limitations, we design an adaptive label propagation mechanism that dynamically modulates propagation weights based on feature similarity between nodes. This adaptive weighting allows the framework to accentuate informative neighbors in homophilic graphs while attenuating misleading connections in heterophilic scenarios. The propagation process is formally defined as follows:

$$f_{\text{ADLP}}^k(v) = \sum_{u \in \mathcal{N}_v} w_{uv}^k \, f_{\text{ADLP}}^{k-1}(u) \tag{6}$$

w_{uv}^k represents the weight between nodes u and v in the k-th layer, and the label of the central node is the weighted sum of its neighbor nodes. Eq.(2) is used to initialize $f_{ADLP}^0(u)$,

and the calculation formula for w_{uv}^k is as follows:

$$w_{uv}^k = \begin{cases} 0, & a_{uv} = 0, \\ \dfrac{\tanh\left(g^{(k)T}\left[h_u^{k-1} \,\|\, h_v^{k-1}\right]\right)}{\sqrt{d_u d_v}}, & a_{uv} = 1. \end{cases} \tag{7}$$

Here, a_{uv} represents the relationship between nodes u and v. If there is no connection between them, the weight is set to 0; if there is a connection between them, the weight is set to the normalized self-attention value of the two. $||$ denotes concatenation operation, $g^{(k)} \in \mathbb{R}^{2F'}$ is the attention vector, and h_u^k is the k-th layer representation of node u. The hyperbolic tangent ($\tanh$) activation function is employed, allowing the propagation weights to take negative values—an essential property for effectively modeling heterophilic graphs [21]. Here, the update of hidden layer representation also uses adaptive weights $h_v^k = \sum_{u \in N_v} w_{uv}^k h_u^{k-1}$, and the enhanced feature matrix $\hat{X}$ is used for initialization at layer zero.

Multi-layer Perceptron Module. The adaptive label propagation component enriches the student model with additional structural information. To further enhance node-level feature learning, we incorporate a multi-layer perceptron module, denoted as f_{MLP}. The multi-layer perceptron module takes the enhanced feature $\hat{X}$ as input and performs node-level prediction. Specifically, two layers of nonlinear transformations followed by a softmax($\cdot$) activation are applied to map the learned features into class probability distributions. The formulation is given as follows:

$$f_{\mathrm{MLP}}(v) = \mathrm{softmax}\Big(f_2\Big(f_1(\hat{X}_v, \Theta_1), \Theta_2\Big)\Big) \tag{8}$$

Label Fusion. The final student model is obtained by integrating the adaptive label propagation module f_{ADLP} and the multi-layer perceptron module f_{MLP}. To enable adaptive fusion between structural and feature-based predictions, a trainable parameter α_v is assigned to each node v, controlling the relative contribution of the two components. The overall formulation is expressed as follows:

$$\begin{aligned} f_{\mathrm{AUGKD}}^k(v) &= \alpha_v \sum_{u \in N_v} w_{uv}^k f_{\mathrm{AUGKD}}^{k-1}(u) + (1-\alpha_v) f_{\mathrm{MLP}}(v) \\ f_{\mathrm{AUGKD}}^1(v) &= \alpha_v \sum_{u \in N_v} w_{uv}^1 f_{\mathrm{ADLP}}^0(u) + (1-\alpha_v) f_{\mathrm{MLP}}(v) \end{aligned} \tag{9}$$

In each layer, the prediction from the adaptive label propagation branch is fused with that from the multi-layer perceptron branch under the control of α_v. Through the distillation process, knowledge is effectively transferred from the teacher to the student model, while simultaneously enriching the student representation with complementary structural and feature information.

D. Optimization

Assuming the student model has K layers, the distillation objective in Eq.(5) can be further elaborated as follows: $\min_\Theta \sum_{v \in V_U} \| f_{\mathrm{TEA}}(v) - f_{\mathrm{AUGKD};\Theta}^K(v) \|_2$. Here, $\| \cdot \|_2$ denotes the L2 norm. The parameter set Θ includes the balancing parameter α between f_{ADLP} and f_{MLP} , the self-attention vector in the f_{ADLP} module, and the parameters in the f_{MLP} module. By minimizing the discrepancy between the teacher and student models, the knowledge encoded in the teacher is effectively transferred to the student. In summary, the overall procedure of the proposed AUGKD is presented in Algorithm 1.

Algorithm 1 AUGKD

Input: Original graph structure A, feature matrix X, propagation times K, MLP model: $f_{MLP}(\cdot)$, ADLP model: $f_{ADLP}(\cdot)$, trainable parameter $\alpha \in \mathbb{R}^{|N|}$, pre-trained teacher model $f_{TEA}(\cdot)$
Output: Node predictions Y_{STU}

```
Calculate node importance based on the feature matrix (homophilic graph):
Calculate the probability of feature component deletion based on w_i^f:
Initialize propagation labels
while not reaching the stopping condition do
    Generate masks ε_i^f for each component
    Mask features according to ε_i^f
    for k = 1 : K do
        Calculate edge weights w_uv^k
        Propagate features using weights:
        Propagate labels using weights:
        Calculate predicted labels f_MLP(v)
        Merge predicted labels f_AUGKD^k(v)
    end for
    Optimize parameters
end while
return The obtained optimal AUGKD framework
```

E. Complexity Analysis

The complexity of data augmentation on the graph is $\mathcal{O}(|N|F)$, where $|N|$ is the number of nodes and F is the feature dimension. The time complexity of two layers of nonlinear transformation is $\mathcal{O}\left(|N|F'\left(F+C\right)\right)$, where F' represents the dimension of the hidden layer and C represents the number of classes. The complexity of K layers adaptive label propagation is $\mathcal{O}(K|E|F')$, where $|E|$ is the number of edges. Therefore, the total time complexity of the AUGKD framework is: $\mathcal{O}\left(K\left(F'|E| + |N|F'\left(F+C\right)\right) + |N|F\right)$. Overall, the complexity is linear with the number of nodes and edges.

III. Experiments

To evaluate the performance of the proposed AUGKD framework, extensive experiments were conducted on six widely used benchmark datasets, with comparisons drawn against representative graph neural network baselines. Section 4.1 introduces the datasets and baseline algorithms and section 4.2. introduces the experimental setup. Section 4.3 presents experiments on homophilic graphs, whereas Section 4.4 reports results on heterophilic graphs. Section 4.5 provides ablation studies to assess the contribution of each model component. Section 4.6 evaluates the generalization capability of AUGKD, while Section 4.7 investigates its robustness. Finally, Section 4.8 examines the framework's behavior under over-smoothing conditions.

A. Dataset and Baselines

This section evaluates the performance of various frameworks on node classification tasks across both homophilic and heterophilic graphs. Specifically, experiments are conducted on

six benchmark datasets. The first three—Cora, Citeseer, and Pubmed—are citation networks [28]. These datasets are characterized by homophilic structures, where connected nodes tend to share the same class labels. In addition, three heterophilic datasets are adopted, namely Actor, Texas, and Cornell [29], where neighboring nodes often belong to different classes. Following prior work [30], only the largest connected component is used for each dataset. The key statistics of all datasets are summarized in Table 1.

TABLE I: Overview of the datasets statistics

Dataset	Nodes	Features	Edges	Classes	$\mathcal{H}$
Cora	2485	1433	5069	7	0.656
Citeseer	2110	3700	3679	6	0.578
Pubmed	19717	500	44338	3	0.644
Actor	7600	932	26659	5	0.008
Texas	183	1703	279	5	0.016
Cornell	183	1703	277	5	0.137

Teacher Models: Six different neural networks are chosen as benchmark algorithms, including GCN [10] , APPNP [31] , FAGCN [32] , GPRGNN [33] , BrenNet [34] , and MAPNET [35]. Student Models: CPF [30] and AUGKD are selected.

B. Experimental Setup

All the experiments are conducted using PyTorch on an NVIDIA GeForce RTX 3090 GPU card with 24 GB memory. Hyperparameters are set as follows: the search range for the learning rate is $\{0.001, \ldots, 0.01\}$, weight decay is $\{1e-3, 5e-4\}$, hidden units is $\{32, 64\}$, and all methods utilize the Adam optimizer. For both teacher and student models, the hyperparameters provided by the original paper are used and set according to the authors' recommendations. For AUGKD, the dropout range for MLP is $\{0.2, \ldots, 0.7\}$, with different values for different datasets; the propagation layer number $K = 10$; the range of feature augmentation magnitude p_f is $\{0.2, \ldots, 0.5\}$, and the upper limit $p_\tau = 0.7$ for feature deletion. AUGKD runs for 1000 epochs and uses early stopping, selecting the framework with the minimum validation loss for testing. For the homophilic graph node classification task, the experiment sparsely partitions the dataset, with each class having 20 labeled nodes, 500 nodes for validation, and the remaining nodes for testing. For the heterophilic graph node classification task, the dataset is densely partitioned, randomly splitting the node set into training/validation/testing sets at a ratio of $60\%/20\%/20\%$. For all tasks, the teacher and student models use the same seed for dataset partitioning to avoid additional supervision information during training. Each combination is experimented for 30 times, and the accuracy and standard deviation of each framework are reported.

C. Homophilic Graph Node Classification

In the experiments on homophilic graphs, three benchmaek datasets were used. Table 2 summarizes the accuracy of node classification, with comparison experimental methods to the baseline algorithms introduced in Section 4.1. VANILLA represents the accuracy reported in the original paper. Sparse data partitioning was employed, and the results for each combination are the average and standard deviation of 30 runs with randomly initialized weights. It can be observed that the proposed data distillation framework exhibits improvements on the graph neural network frameworks. In comparison with CPF, AUGKD demonstrates favorable performance in most cases. Specifically, on the Cora dataset, GCN shows the highest improvement, with an increase of 3.36%. A notable enhancement is also observed with MAPNET, achieving 86.51%. Despite utilizing an adaptive mechanism similar to MAPNET, why does data distillation still show improvement? This is attributed to the supplementation of additional node features during the distillation process. On the Citeseer dataset, GCN demonstrates an improvement of 4.43%, with the highest accuracy reaching 75.12% when the teacher model is FAGCN. On Pubmed, the highest accuracy is achieved at 82.6% by FAGCN. On BrenNet, if the heterophilic graph form $w_i^f = \log(\sum_{v \in V} x_{vi}^f)$ is chosen during the importance calculation stage, the improvement reaches 6.98%; however, if the homophilic graph form $w_i^f = \log(\sum_{v \in V} x_{vi}^f d_v)$ is selected, the improvement is only 4.67%. This could be due to the Pubmed dataset relying more on global information rather than neighborhood information.

TABLE II: Comparative classification accuracy (%) achieved on homophilic graph datasets

Datasets	MODE	GCN	APPNP	FAGCN	GPRGNN	BERNET	MAPNET
Cora	VANILLA	81.5±0.1	83.7±0.0	84.1±0.1	83.5±0.0	83.0±0.1	84.6±0.0
	CPF	84.6±0.2	84.3±0.2	85.2±0.2	84.9±0.4	84.4±0.4	86.2±0.2
	AUGKD	**84.9±0.3**	**84.7±0.3**	**85.2±0.4**	**85.1±0.4**	**84.9±0.3**	**86.5±0.2**
Citeseer	VANILLA	70.3±0.0	71.5±0.0	72.7±0.0	71.7±0.1	72.8±0.0	73.9±0.0
	CPF	74.7±0.5	**74.0±0.5**	74.5±0.7	73.1±0.4	74.1±0.4	**75.5±0.3**
	AUGKD	**74.7±0.3**	73.8±0.4	**75.1±0.4**	**73.7±0.3**	**74.1±0.4**	75.2±0.3
Pubmed	VANILLA	79.0±0.0	79.2±0.0	79.4±0.0	75.9±0.1	70.4±0.0	79.4±0.0
	CPF	79.8±0.1	81.1±0.4	82.0±0.5	76.4±0.4	76.5±0.5	81.1±0.2
	AUGKD	**81.3±0.3**	**82.0±0.5**	**82.6±0.5**	**78.4±0.5**	**77.3±0.6**	**82.1±0.4**

D. Heterophilic Graph Node Classification

In the heterogeneous graph experiments, the Actor co-occurrence graph, Texas, and Cornell were utilized. Table 3 summarizes the accuracy of node classification, contrasting with the baseline algorithms introduced in Section 4.1. The heterophilic graph structure was employed in the importance calculation stage, with the dataset partitioned in a dense manner. All results represent the average and standard deviation of 30 runs with random weight initialization. Overall, AUGKD showed improvements across all graph neural networks and consistently outperformed CPF. The highest enhancement was observed with GCN, reaching 3.25%, 12.04%, and 18% on the three datasets. On Actor, the accuracy peaked at 42.42%, while on Texas and Cornell, it reached 99.61% and 98.83%, respectively. It's notable that CPF either didn't improve or experienced varying degrees of decline, especially evident on the Actor dataset. This discrepancy can be attributed to the structural characteristics of heterophilic graphs. Label propagation is based on the homophily assumption, which contradicts with heterophilic graphs. Although AUGKD also relies on label propagation, it incorporates adaptive mechanisms and introduces negative weights. This makes it less favorable for

predicting edge weights, which might decrease or even become negative.

TABLE III: Comparative classification accuracy (%) achieved on heterophilic graph datasets

Datasets	MODE	GCN	APPNP	FAGCN	GPRGNN	BERNET	MAPNET
Actor	VANILLA	32.1±0.1	38.9±0.2	39.2±0.1	39.5±0.0	39.8±0.0	40.7±0.0
	CPF	26.5±2.7	23.6±1.9	32.3±3.3	29.7±2.6	31.3±2.7	39.6±0.7
	AUGKD	**35.4±0.4**	**41.5±1.3**	**41.2±1.3**	**40.6±0.3**	**40.8±0.5**	**42.4±0.4**
Texas	VANILLA	74.5±0.1	91.5±0.1	85.7±0.0	93.3±0.0	92.3±0.0	92.2±0.0
	CPF	77.9±1.2	85.6±2.4	85.0±5.0	96.0±0.9	98.7±0.5	92.9±0.7
	AUGKD	**86.5±1.1**	**93.5±1.5**	**98.2±1.6**	**98.6±0.4**	**99.6±0.6**	**96.8±0.9**
Cornell	VANILLA	67.5±0.1	86.7±0.1	81.1±0.2	91.1±0.1	92.7±0.2	92.9±0.1
	CPF	81.7±1.6	82.9±4.8	75.8±3.6	93.9±1.0	98.3±0.3	95.6±0.6
	AUGKD	**85.5±1.5**	**91.0±1.8**	**87.1±2.4**	**96.5±0.8**	**98.8±0.4**	**96.2±0.4**

E. Ablation Study and Enhancement Strategy Selection

In the ablation experiments and the selection of enhancement strategies, the teacher model was consistently MAPNET, as it performed well in both homophilic and heterophilic graphs. Table 4 presents the results of the ablation experiments, where w/o mlp indicates removing the multilayer perceptron (MLP) in the label fusion stage and only using adaptive label propagation, w/o ad_lp indicates only using the MLP, w/o aug is without data augmentation, and AUGKD represents the complete framework. It can be observed that all variants experience varying degrees of decline, with the most significant decrease seen in w/o mlp. This is because label propagation solely relies on soft labels, which lose a considerable amount of valid information across different transformations of the teacher model, insufficient to support the student model's performance. w/o ad_lp utilizes only node features without leveraging structural features, hence failing to achieve the performance of the complete framework.

In the experiment of selecting enhancement strategies, w/ random represents random augmentation, w/ homophilic denotes homophilic graph enhancement strategy, w/ heterophilic represents heterophilic graph enhancement strategy, and w/o aug indicates no data augmentation. Table 5 displays the accuracy under different augmentation strategies. It can be observed that the accuracy of the Cora dataset is highest under w/ homophilic, while for the Actor dataset, it is highest under w/ heterophilic. This is because the homophilic graph enhancement strategy focuses more on the importance of the neighborhood, while the heterophilic graph enhancement strategy emphasizes global importance, consistent with the characteristics of the dataset. It is worth noting that the improvement is minimal under random augmentation, as random augmentation disrupts the integrity of the data, leading to a decline in framework performance.

TABLE IV: Quantitative results of the ablation experiments on framework components

Method	Cora	Actor
w/o mlp	71.13±0.30	21.56±0.81
w/o ad_lp	85.64±0.24	41.59±0.68
w/o aug	86.23±0.13	41.86±0.47
AUGKD	**86.51±0.19**	**42.42±0.44**

TABLE V: Comparative classification accuracy (%) across different importance selection strategies

Method	Cora	Actor
w/ random	86.07±1.57	41.23±0.33
w/ homophilic	**86.51±0.19**	42.12±0.34
w/ heterophilic	86.39±0.38	**42.42±0.44**
w/o aug	86.23±0.13	41.86±0.47
vanilla	84.58±0.00	40.7±0.07

F. Generalization Performance Experiment

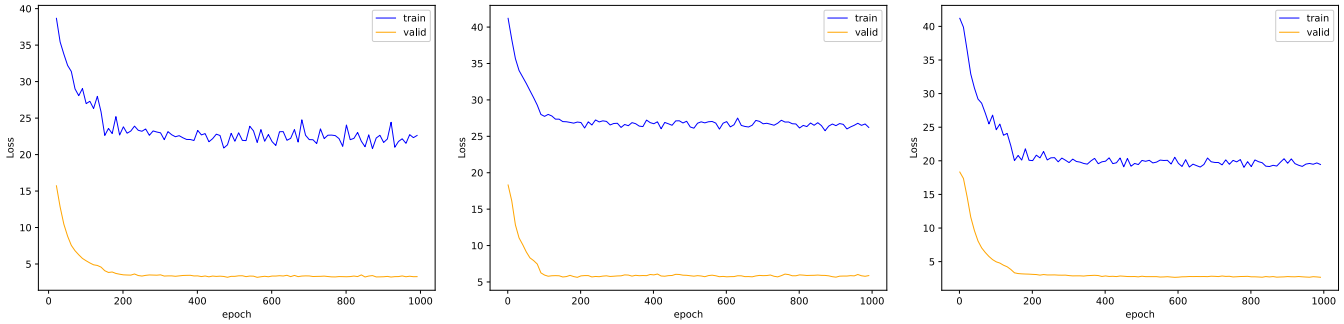
This section investigates the impact of removing data augmentation and random data augmentation on the framework's generalization ability. The experiment analyzes the loss of AUGKD on the training set and validation set on the Cora dataset. The GCN is selected as the teacher model, and a smaller difference between the two losses indicates better generalization performance of the model. The generalization performance of the AUGKD framework and its two variants without data augmentation (w/o aug) and with random data augmentation (w/ random) is illustrated in Figure 2. In the middle are the loss comparisons without data augmentation, where the lines are relatively smooth but with significant differences. On the left are the loss comparisons with random data augmentation, where the lines fluctuate significantly and there are large differences. On the right is the complete framework, with stable lines and smaller differences. The experiment demonstrates that importance-based data augmentation is more stable than random augmentation and leads to greater improvement in framework generalization performance.

G. Robustness Experiment

In this section, random feature deletion is used to study the robustness of AUGKD, and observe the framework's accuracy by perturbing the node features of the trained framework. The teacher models selected are GCN, APPNP, and MAPNET. Figure 3 presents the classification accuracy of various frameworks on the Cora dataset under different perturbation rates. It can be observed that the reduction in accuracy of AUGKD on all frameworks is consistently smaller than that of the original frameworks. When randomly deleting 50% of the features in Cora, the accuracy of AUGKD decreases by only 0.92% on GCN, whereas the original framework decreases by 4.05%; on APPNP, AUGKD decreases by 3.25%, while the original framework decreases by 5.18%; on MAPNET, AUGKD decreases by 0.71%, while the original framework decreases by 3.53%. This study demonstrates that combining graph neural networks with AUGKD can enhance the robustness of the original framework.

H. Over-smoothing Experiment

When the propagation step size increases, many graph neural networks face the problem of oversmoothing. To verify AUGKD's ability to alleviate oversmoothing, experiments with different propagation steps were conducted on three homophilic datasets, with GCN serving as the teacher model.



(a) w/ random (b) w/o aug (c) AUGKD

Fig. 2: Generalization performance of the proposed AUGKD framework on the Cora dataset, compared with its ablated variants without feature enhancement (w/o aug) and without the normalization layer (w/ random).

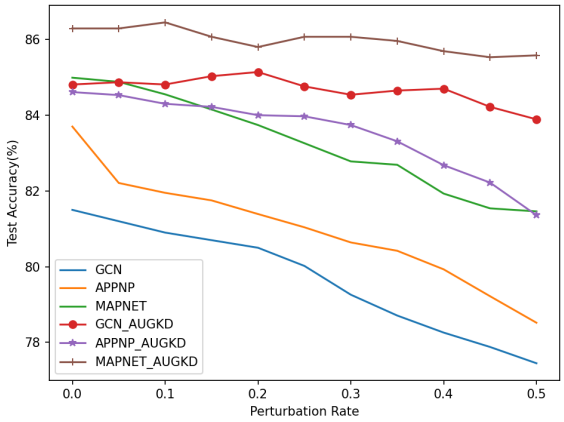


Fig. 3: Robustness evaluation on the Cora dataset — accuracy under varying perturbation rates.

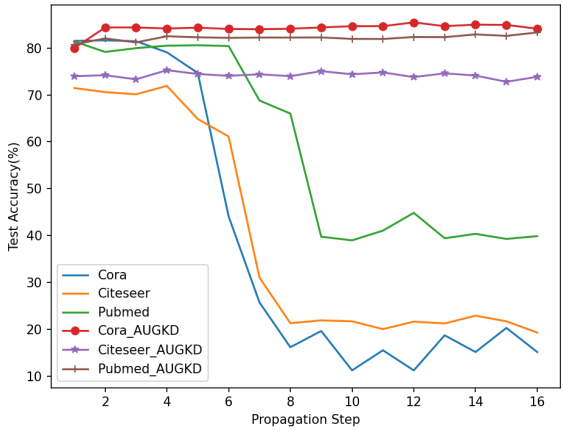


Fig. 4: Over-smoothing evaluation on the Cora dataset — accuracy under varying propagation steps.

Figure 4 displays the experimental results, where the propagation step size of AUGKD is controlled by the hyperparameter K, while GCN is adjusted by stacking different hidden layers. The experimental results indicate that as the propagation step size increases, the performance of GCN significantly decreases. On Cora, the accuracy drops from 81.5% to 15.18%, from 71.47% to 19.33% on Citeseer, and from 81.39% to 39.89% on Pubmed. In contrast, AUGKD shows no significant changes across the three datasets, indicating its insensitivity to propagation step size. This highlights the advantage of the adaptive mechanism, during training, if certain features are not conducive to the final prediction, the weights of these features are reduced.

IV. Conclusion

This paper presents a universal graph node classification framework, Adaptive Universal Graph Knowledge Distillation (AUGKD), to address the limited generalization capability of existing graph data distillation frameworks across diverse graph neural network architectures. At its core, a universal adaptation-based distillation paradigm is developed to enhance the performance of graph neural networks, particularly in heterophilic settings. In addition, an importance-guided node feature enhancement strategy adaptively selects augmentation methods according to dataset heterogeneity, thereby improving generalization and predictive robustness. Within this unified framework, a hybrid feature extraction module is devised by applying label propagation principles in the design of multilayer perceptrons to enrich feature diversity and effectively mitigate over-smoothing. Finally, a weight adaptation mechanism introduces adaptive coefficients and negative weights to better capture heterophilic relationships, further enhancing representational flexibility. Extensive experiments demonstrate that AUGKD achieves consistent performance gains across multiple benchmark datasets and can be readily applied to a wide range of graph neural network architectures. Despite its effectiveness, AUGKD is currently limited in handling large-scale, heterogeneous, and dynamic graphs. Future work will focus on addressing these limitations and integrating the method into unified hypergraph learning frameworks.

References

[1] M. Lin, X. Hong, W. Li, and S. Lu, “Unified graph neural networks pre-training for multi-domain graphs,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, pp. 12165–12173, 2025.

[2] M. Freund, R. Dorsch, S. Schmid, T. Wehr, and A. Harth, "Enriching RDF data with LLM-based named entity recognition and linking," in *Int. Knowl. Graph Semantic Web Conf.*, Springer, pp. 109–122, 2024.
[3] S. M. Oo and O. Hartig, "An algebraic foundation for knowledge graph construction," in *Eur. Semantic Web Conf. (ESWC)*, Springer, pp. 3–22, 2025.
[4] Z. Sheng, W. Guo, Y. Shao, W. Zhang, and B. Cui, "LLMs are noisy oracles: Noise-aware graph active learning," in *Proc. ACM SIGKDD*, pp. 2526–2537, 2025.
[5] Y. Zhao, H. Lin, S. Wen, J. Shen, and B. Hua, "SMA-GNN: A symbol-aware GNN for signed link prediction," in *Proc. ACM SIGKDD*, pp. 3957–3967, 2025.
[6] N. Firouzkouhi *et al.*, "Generalized fuzzy hypergraph for link prediction and identification of influencers in dynamic social media networks," *Expert Syst. Appl.*, vol. 238, p. 121736, 2024.
[7] J. Wang *et al.*, "Multimodal fusion framework based on knowledge graph for personalized recommendation," *Expert Syst. Appl.*, vol. 268, p. 126308, 2025.
[8] Y. Ding, Z. Zhang, and B. Wang, "Category-integrated dual-task graph neural networks for session-based recommendation," *Expert Syst. Appl.*, vol. 263, p. 125784, 2025.
[9] X. Song, B. Zhou, Y. Wang, and W. Liu, "Dynamic graph structure evolution for node classification with missing attributes," *Sci. Rep.*, vol. 15, no. 1, p. 25687, 2025.
[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
[11] P. Velickovic *et al.*, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.
[12] R. Zeng, J. Fang, S. Liu, Z. Meng, and S. Liang, "Enhancing graph neural networks via memorized global information," *ACM Trans. Web*, vol. 18, no. 4, pp. 1–34, 2024.
[13] E. Hssayni, A. Boufssasse, N.-E. Joudar, and M. Ettaouil, "Novel dropout approach for mitigating over-smoothing in graph neural networks," *Appl. Intell.*, vol. 55, no. 5, p. 354, 2025.
[14] F. Qian *et al.*, "Exploring the over-smoothing problem of graph neural networks for graph classification: An entropy-based viewpoint," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, pp. 3235–3244, 2025.
[15] J. Zhu *et al.*, "On the impact of feature heterophily on link prediction with graph neural networks," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 37, pp. 65823–65851, 2024.
[16] L. Duan *et al.*, "Ligand-receptor dynamics in heterophily-aware graph neural networks for enhanced cell type prediction from single-cell RNA-seq data," *Front. Mol. Biosci.*, vol. 12, Art. no. 1547231, 2025.
[17] S. Liu *et al.*, "Integrating co-training with edge discrimination to enhance graph neural networks under heterophily," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, pp. 18960–18968, 2025.
[18] R. Duan *et al.*, "Unifying homophily and heterophily for spectral graph neural networks via triple filter ensembles," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 37, pp. 93540–93567, 2024.
[19] D. Lopez-Paz *et al.*, "Unifying distillation and privileged information," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28, 2015.
[20] Y. Yang *et al.*, "Distilling knowledge from graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 9294–9302, 2019.
[21] J. Liu *et al.*, "Ddk: Distilling domain knowledge for efficient large language models," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 37, pp. 98297–98319, 2024.
[22] M. A. Khan *et al.*, "HYDRA-FL: Hybrid knowledge distillation for robust and accurate federated learning," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 37, pp. 50469–50493, 2024.
[23] W. Xu *et al.*, "Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2025.
[24] C. He *et al.*, "DA-KD: Difficulty-aware knowledge distillation for efficient large language models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2025.
[25] Y. Tian *et al.*, "Knowledge distillation on graphs: A survey," *ACM Comput. Surv.*, vol. 57, no. 8, pp. 1–16, 2025.
[26] Y. You *et al.*, "Graph contrastive learning with augmentations," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, pp. 5812–5823, 2020.
[27] Y. Zhu *et al.*, "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf. (WWW)*, pp. 2069–2080, 2021.
[28] J. Shen *et al.*, "GTAT: Empowering graph neural networks with cross attention," *Sci. Rep.*, vol. 15, no. 1, p. 4760, 2025.
[29] Y. Yang *et al.*, "Implicit vs unfolded graph neural networks," *J. Mach. Learn. Res.*, vol. 26, no. 82, pp. 1–46, 2025.
[30] C. Yang, J. Liu, and C. Shi, "Extract the knowledge of graph neural networks and go beyond it," in *Proc. Web Conf. (WWW)*, pp. 1227–1237, 2021.
[31] J. Klicpera *et al.*, "Predict then propagate: Graph neural networks meet personalized PageRank," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.
[32] D. Bo *et al.*, "Beyond low-frequency information in graph convolutional networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, pp. 3950–3957, 2021.
[33] E. Chien *et al.*, "Adaptive universal generalized PageRank graph neural network," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.
[34] M. He *et al.*, "BernNet: Learning arbitrary graph spectral filters via Bernstein approximation," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, pp. 14239–14251, 2021.
[35] Q. Ma *et al.*, "Adaptive propagation network based on multi-scale information fusion," in *Int. Conf. Artif. Neural Netw. (ICANN)*, Springer, pp. 51–62, 2023.

Junqi Liu is currently working toward the Ph.D. degree in computer science and technology from the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education of China, Shanxi University, Taiyuan, China. His research interests include deep learning, graph neural networks.

Yu Xie received the Ph.D. degree in pattern recognition and intelligent systems from Xidian University, Xi'an, China, in 2020. From 2019 to 2020, he was a research assistant with School of Computer Science and Engineering, Nangyang Technological University. He is currently an Associate Professor with School of Computer and Information Technology, Shanxi University. His research interests include graph machine learning and federated learning.

Tianjun Ma received the Ph.D. degree in Engineering from the Institute of Information Engineering, Chinese Academy of Sciences, China. He is currently a Lecturer with the School of Computer and Information Technology, Shanxi University, Taiyuan, China. His research interests include cryptography, blockchain, and network security protocols.

Wei Zheng received the Ph.D. degree with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China. He is currently a Lecturer with the School of Computer and Information Technology, Shanxi University, Taiyuan, China. His research interests include evolutionary algo rithms and their applications.

Jiangjiang Zhang received the Ph.D. degree with the Beijing University of Technology, China. He is currently a Lecturer with the School of Computer and Information Technology, Shanxi University, Taiyuan, China. His research interest includes data security, privacy protection, computational intelligence, and combinatorial optimization and modeling.