GEM-Distill: Graph Ensemble Multi-level Distillation for MLP-based Graph Classification

Minghao Qin, Donghai Guan, Weiwei Yuan, Shuai Xu, and Çetin Kaya Koç, *Life Fellow*, *IEEE*

***Abstract*—Graph Neural Networks (GNNs) achieve strong performance on graph learning tasks, but their message-passing computation hinders deployment in resource-constrained settings. GNN-to-MLP (G2M) knowledge distillation improves inference efficiency by transferring GNN knowledge to lightweight MLP students; however, existing methods mainly target node classification and generalize less effectively to graph classification, where supervision is sparse, teacher global semantics are difficult to transfer, and MLP students have limited structural expressiveness. This paper presents GEM-Distill, a graph ensemble multi-level distillation framework for MLP-based graph classification. GEM-Distill assigns graph-, subgraph-, and node-level objectives to granularity-specific MLP experts and coordinates them through a sequential coarse-to-fine training strategy, mitigating cross-granularity optimization conflicts while preserving complementary structural supervision. It further introduces Virtual-Neighbor Guided Graph Distillation (VNGD), a training-only graph-level distillation module that enriches global supervision by mixing semantically related training graphs. Experiments on standard and large-scale graph classification benchmarks show that GEM-Distill achieves competitive or stronger performance than existing G2M baselines while retaining efficient MLP-style inference. Additional analyses on robustness, parameter-aligned comparisons, efficiency, and teacher architectures further support the effectiveness and practicality of the framework.**

***Index Terms*—Graph neural networks, Knowledge distillation, Graph classification, GNN-to-MLP knowledge distillation.**

I. Introduction

GRAPH Neural Networks (GNNs) [1], [2], [3], [4] have demonstrated outstanding capabilities in processing graph-structured data, driving breakthrough advancements in graph machine learning across various fields, including recommendation systems [5], bioinformatics [6], and social network analysis [7]. Nevertheless, the message-passing paradigm of GNNs requires nodes to iteratively aggregate information from their neighbors at each layer, which introduces significant computational and communication overhead on large-scale graph data. This limitation severely constrains the deployment and application of GNN models in latency-sensitive and resource-constrained real world scenarios [8].

To improve the efficiency of GNNs, various acceleration techniques have been investigated, including quantization, pruning, sampling, architecture compression, and knowledge distillation. These techniques can lower the cost of GNN inference, but they usually retain the message-passing dependency and may require careful architecture- or dataset-specific design. In contrast, knowledge distillation provides a model-agnostic way to transfer the predictive behavior or intermediate representations of a strong GNN teacher to a lightweight student [9], [10], [11]. This makes distillation particularly attractive for preserving accuracy while improving deployment efficiency.

Multi-Layer Perceptrons (MLPs) offer a natural student architecture for efficient graph inference. Unlike GNNs, MLPs do not perform neighborhood aggregation at inference time; they predict directly from node or graph features, making them computationally efficient, easy to parallelize, and practical for resource-constrained environments. Nevertheless, this efficiency comes at the cost of limited structural awareness: an MLP cannot explicitly model graph topology or high-order neighborhood dependencies. GNN-to-MLP (G2M) distillation addresses this trade-off by injecting structural knowledge from a pre-trained GNN teacher into an inference-efficient MLP student. Early and recent G2M methods have demonstrated the promise of this direction through soft-label distillation, ensemble learning, multi-granularity supervision, and vector-quantized structural targets [12], [13], [14], [15], [16]. Recent studies further improve MLP-based graph inference by refining teacher supervision, confidence-guided knowledge transfer, class-view distillation, cross-layer alignment, and positional encoding enhanced multi-level distillation [17], [18], [19], [20].

Despite these advantages, MLP students have an inherent limitation: they do not explicitly encode graph topology. As a result, they often struggle to distill meaningful structural information from graph data, leading to suboptimal performance on downstream tasks. Existing G2M methods thus rely on knowledge distillation (KD) [9] to inject structural information from a pre-trained GNN teacher into the MLP student. This strategy has shown promising results on node classification, where dense node-level labels and abundant feature interactions provide relatively rich supervision. Despite promising performance on node classification, G2M approaches fall short when extended to graph classification tasks [21], [22]. This is primarily due to three major challenges:

1) ***Sparse Learning Signals:*** G2M methods succeed in node classification partly because node-level datasets offer high-dimensional feature spaces, where GNNs and MLPs yield comparable equivalence classes [23], [12].

This work is supported by the National Natural Science Foundation of China (No. 62472220).*(Corresponding author: Donghai Guan.)*

The authors are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: minghao_q@nuaa.edu.cn; dhguan@nuaa.edu.cn; yuanweiwei@nuaa.edu.cn; xushuai7@nuaa.edu.cn; cetinkoc@ucsb.edu).

TABLE I
ABLATION STUDY RESULTS IN MUGSI (LEFT: SINGLE COMPONENTS; RIGHT: MULTIPLE COMPONENTS AND FULL MODEL)

	w/ GraphKD	w/ ClusterKD	w/ NodeKD	w/o GraphKD	w/o ClusterKD	w/o NodeKD	MuGSI$_{MLP}$
PROTEINS	76.28±2.59*	76.55±2.60	76.37±2.96*	77.18±3.14	**76.01±1.59**↓	76.64±3.37	77.63±3.45
BZR	85.81±3.09	85.42±3.01*	85.20±3.43*	**85.18±4.33**↓	85.42±2.49	85.68±2.78	**85.68±2.56**↓
NCI1	66.24±1.62*	66.04±1.26*	66.15±1.83	66.30±1.70	66.21±1.84	**66.15±1.68**↓	67.23±1.12
REDDIT-B	85.04±1.23	85.61±1.49*	85.31±1.10*	**85.55±1.50**↓	85.13±1.15	85.63±1.51	86.70±1.72

In node classification, dense label distributions and feature interactions provide abundant supervisory signals for student learning. By contrast, in graph classification, labels derive only from the teacher's final graph-level predictions, forcing students to rely on globally pooled features. This yields extremely sparse alignment signals, severely limiting knowledge transfer—especially when training data or class numbers are small, where MLP students often underperform [12], [13].

2) ***Difficulty in Transferring the Teacher's Global Knowledge:*** Modern GNN teachers are increasingly expressive and often encode high-dimensional global semantics [24], [25]. For graph classification, this knowledge is compressed into graph-level embeddings or logits. A shallow MLP student can easily fail to absorb such complex global representations, especially when the training graphs are limited or highly diverse.
3) ***Limited Expressive Capacity of MLP Students:*** In graph classification, alongside the sparsity of graph signals and the complexity of the teacher's outputs, the inherently simple structure of an MLP student prevents the model from capturing complex topological structures or high-order semantics, which considerably undermines the effectiveness of information extraction.

Recent graph-classification-oriented G2M methods, especially MuGSI [15], attempt to alleviate sparse supervision by distilling graph-, subgraph-, and node-level knowledge into a single MLP student. Different granularities provide heterogeneous and sometimes competing objectives: graph-level signals emphasize global semantics, subgraph-level signals capture mesoscopic structural patterns, and node-level signals focus on fine-grained local information. When these objectives are forced into one shared student, their gradients may interfere with each other, causing cross-granularity optimization conflicts. Experimental results (Table I) show that single-objective variants can sometimes outperform multi-objective variants in MuGSI, and adding extra granularities may even degrade performance. Furthermore, graph-level distillation contributed least to performance gains, suggesting MLP students struggle to absorb high-order global knowledge distilled from GNN teachers. All evaluations followed the open-source MuGSI framework.

Present Work. To address these challenges, we propose GEM-Distill (Graph Ensemble Multi-level Distillation for MLP-based Graph Classification), a multi-level expert distillation framework for graph classification. Its design follows the three issues identified above. (1) To mitigate sparse supervision and cross-granularity interference, GEM-Distill assigns graph-, subgraph-, and node-level objectives to separate MLP experts, enriching structural supervision without forcing heterogeneous objectives into a shared student. (2) To facilitate global knowledge transfer, we introduce Virtual-Neighbor Guided Graph Distillation (VNGD), a training-time graph-level distillation module that constructs semantic virtual neighbors and performs sample mixing among related training graphs. (3) To enhance student expressiveness, GEM-Distill adaptively ensembles the granularity-specific experts and reweights hard samples during sequential distillation, enabling complementary supervision to be integrated across structural scales.

The main contributions of this work are summarized as follows:

- We propose a hierarchical and sequential GNN-to-MLP distillation framework that assigns graph-, subgraph-, and node-level objectives to granularity-specific MLP experts, mitigating interference among heterogeneous structural objectives.
- We introduce VNGD, a training-time graph-level distillation module that enriches global supervision through semantic virtual neighbors and sample mixing.
- We conduct extensive experiments on standard and large-scale graph classification benchmarks, showing that GEM-Distill achieves competitive or stronger performance than existing G2M baselines while preserving efficient MLP-style inference.

The code will be made public at: https://github.com/Universeping/GEM-Distill.

II. RELATED WORK

Graph Neural Networks (GNNs) can be broadly categorized into spectral and spatial methods. Spectral methods, such as GCN [3], Spectral CNN [26], and ChebyNet [27], define graph convolutions from graph signal processing perspectives, while spatial methods, such as GAT [4], GraphSAGE [28], and GIN [1], aggregate information from local neighborhoods. Although these models differ in formulation, they generally rely on message passing and therefore incur inference latency, memory overhead, and communication cost on large graphs. To reduce these costs, prior studies have explored quantization [29], [30], pruning [31], [32], architecture compression, and knowledge distillation.

GNN-to-GNN Distillation. Knowledge distillation transfers knowledge from a strong teacher to a compact student [9]. In graph learning, GNN-to-GNN distillation methods train smaller GNN students to mimic larger teacher GNNs. LSP [11] preserves local structure, TinyGNN [10] searches compact student architectures, and RDD [33] designs reliable distillation objectives. These methods reduce model size, but their

students still perform message passing during inference, which limits deployment acceleration.

GNN-to-MLP Distillation. To remove inference-time graph dependency, GNN-to-MLP distillation trains an MLP student to mimic a GNN teacher. GLNN [12] uses soft-label distillation, while later methods incorporate more structural signals. KRD [34] studies reliable distillation, NOSMOG [35] aligns structural representations, and FF-G2M [36] leverages spectral information. Recent methods further explore vector-quantized structural targets, self-supervised alignment, task-agnostic distillation, and multi-teacher transfer [16], [37], [38], [39]. Related advances transfer high-order hypergraph knowledge to MLPs and theoretically explain how MLP students acquire implicit graph inductive biases [40], [41]. Nevertheless, most existing methods focus on node-level prediction or specialized graph domains, leaving graph classification with multi-granularity supervision underexplored.

GNN-to-MLP in Graph Classification. Most current G2M distillation frameworks focus on solving node classification within a single graph, but lack designs for graph classification task. Existing graph-level or graph-free distillation methods [42], [43], [44] provide useful attempts, and MuGSI [15] introduces multi-granularity structural distillation for graph classification. However, challenges remain, including the limited expressivity of MLP students, optimization conflicts among multiple objectives, and the difficulty of effectively modeling global graph semantics for graph classification tasks.

III. Preliminaries

A. Notations

Let $G = (\mathcal{V}, \mathcal{E})$ denote a simple undirected graph, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}| = N$, and $\mathcal{E}$ is the set of edges. Each node $\nu_i \in \mathcal{V}$ is associated with a d-dimensional feature vector $x_i \in \mathbb{R}^d$; all node features are collected in $X \in \mathbb{R}^{N\times d}$. The adjacency matrix $A \in \mathbb{R}^{N\times N}$ encodes the connectivity, where $A_{i,j} = 1$ if $(\nu_i, \nu_j) \in \mathcal{E}$ and $A_{i,j} = 0$ otherwise.

B. Graph Classification Problem

Given a set of graphs $\mathcal{G} = \{G_i\}_{i=1}^N$, where each G_i is associated with its node feature matrix X^i and adjacency matrix A^i, the objective of graph classification is to learn a mapping $f : G_i \to y_i$, where $y_i \in \{0, 1\}^C$ is a one-hot vector representing the label of G_i among C classes. The dataset $\mathcal{G}$ is typically partitioned into labeled graphs $\mathcal{G}_L$ (for training and validation) and unlabeled graphs $\mathcal{G}_U$ (for testing). The goal is to train f on $\mathcal{G}_L$ to accurately predict the class labels of graphs in $\mathcal{G}_U$.

C. Graph Neural Networks

We consider standard message-passing Graph Neural Networks (GNNs). For an L-layer GNN, at the l-th layer, each node ν_i updates its representation as follows:

$$m_i^{(l)} = \text{AGGREGATE}^{(l)}\left(\{h_j^{(l-1)} : \nu_j \in \mathcal{N}_i\}\right), \quad (1)$$

$$h_i^{(l)} = \text{UPDATE}^{(l)}\left(h_i^{(l-1)}, m_i^{(l)}\right), \quad (2)$$

where $h_i^{(l)}$ is the embedding of node ν_i at layer l, $m_i^{(l)}$ is the aggregated information from its neighbors $\mathcal{N}_i$, and $h_i^{(0)} = x_i$ is the initial node feature.

For graph-level tasks, a readout function integrates node representations at the final layer to generate a graph embedding:

$$h_G = \text{READOUT}\left(\{h_i^{(L)} : \nu_i \in \mathcal{V}\}\right), \quad (3)$$

which is then input to a classifier to obtain predictions at graph level.

D. Graph-to-MLP Knowledge Distillation

Knowledge distillation (KD) [9] was originally proposed to transfer knowledge from a large "teacher" model to a compact "student" model. In graph learning, several works such as GLNN [12] and CPF [13] extend KD to distill knowledge from a GNN teacher to an MLP student, commonly via the Kullback-Leibler (KL) divergence between their softened predictions:

$$\mathcal{L}_{SL} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} D_{KL}\left(\sigma(z_i^g/\tau), \sigma(z_i^m/\tau)\right), \quad (4)$$

where z_i^g and z_i^m are the logits of the GNN and MLP for node i, $\sigma(\cdot)$ is the softmax function, and $\tau > 0$ is the temperature.

The student is also supervised using the cross-entropy (CE) loss with ground-truth labels:

$$\mathcal{L}_{CE} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \text{CE}\left(\sigma(z_i^m), y_i\right), \quad (5)$$

The total loss is defined as

$$\mathcal{L} = \lambda\mathcal{L}_{CE} + (1-\lambda)\mathcal{L}_{SL}, \quad (6)$$

where $\lambda \in [0, 1]$ balances the contribution of each loss.

IV. Methodology

GEM-Distill is a conflict-aware GNN-to-MLP distillation framework for graph classification. Given a pre-trained GNN teacher, it assigns graph-, subgraph-, and node-level objectives to three granularity-specific MLP experts rather than imposing these heterogeneous objectives on a single student. Each expert distills the teacher representation at its assigned structural scale and also minimizes prediction-level divergence from the teacher outputs, thereby preserving complementary global, mesoscopic, and local supervision while reducing direct optimization interference. The overall architecture is shown in Fig. 1.

During initial data processing, node features in graph classification scenarios, such as social networks, bioinformatics, and molecular graphs, can be sparse or incomplete. While GNNs can exploit graph connectivity to mitigate feature deficiency, MLPs lack this advantage. We therefore augment the original node features with Laplacian eigenvector positional encodings [45]. Following prior graph learning and graph distillation studies [46], [15], these encodings introduce global structural information before being fed to the MLP experts.

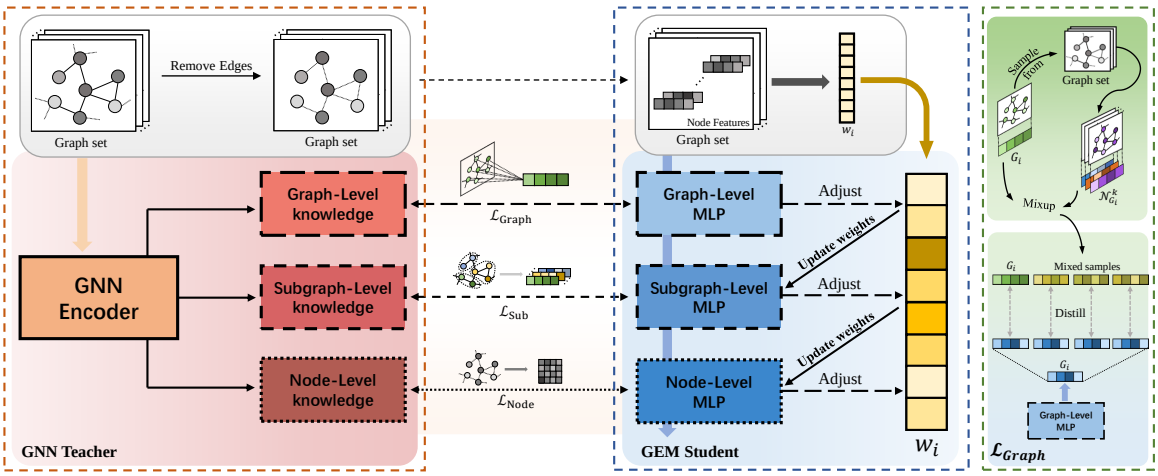

Fig. 1. The figure illustrates the knowledge distillation process in GEM-Distill. First, a pre-trained GNN teacher outputs representations at the graph, subgraph, and node levels ($\mathcal{L}_{Graph}$, $\mathcal{L}_{Sub}$, $\mathcal{L}_{Node}$), which are used to guide three corresponding MLP-based student experts. Both sample and ensemble weights of the students are updated sequentially. For clarity, soft label distillation loss ($\mathcal{L}_{SL}$) and cross-entropy loss ($\mathcal{L}_{CE}$) are omitted from the diagram.

A. Multi-level Distillation

Graph data inherently exhibits a hierarchical structure, where knowledge can be naturally decomposed into multiple granularities such as node-, subgraph-, and graph-level representations [15], [47], [48]. Prior studies have demonstrated that adopting an appropriate granularity is crucial, as overly coarse or excessively fine levels may hinder knowledge transfer and optimization. In graph classification, these three granularities correspond to complementary structural abstractions. Graph-level knowledge captures holistic semantics that are directly aligned with graph labels and is essential for transferring the teacher's global representation. Subgraph-level knowledge characterizes mesoscopic structural units, such as communities or functional substructures, which bridge global graph semantics and local node patterns. Node-level knowledge preserves fine-grained topological and feature interactions, which are often weakened when replacing message-passing GNNs with structure-independent MLP students.

Motivated by this coarse-to-fine hierarchy, we adopt a three-level design in our distillation framework, where each student expert specializes in one granularity and receives independent supervision. This design is also motivated by the optimization conflict observed in multi-granularity G2M distillation: when Graph-, Subgraph-, and Node-level objectives are imposed on a single student, their gradients may compete because they emphasize different structural scales. GEM-Distill instead assigns these objectives to independent MLP experts and coordinates them through sequential training and adaptive ensemble weighting. Importantly, while the implementations of some level-specific distillation modules follow prior work, our novelty lies in reorganizing these granularities into a conflict-aware, ensemble-driven distillation scheme. This shifts the methodological contribution from designing each granularity in isolation to resolving cross-level optimization conflicts and integrating complementary knowledge across the graph hierarchy.

1) Graph-level: Holistic Knowledge Transfer. Most GNN-based methods for graph-level prediction rely on iterative message passing and aggregation, ultimately pooling node features to produce a graph-level representation. Prior GNN-to-MLP distillation methods [15], [36], [37] often focus only on the final embedding or output logits, neglecting inter-graph dependencies and shared information. Recent studies have suggested that leveraging sample difficulty or local neighborhood information can enhance the effectiveness of knowledge distillation. Some recent efforts explore neighborhood mixup or Hard-aware strategies at the node-level [49], [34], [50], but their assumptions do not naturally carry over to graph classification: unlike nodes, individual graphs do not have explicit neighbors. To overcome this fundamental gap, we propose a *Virtual-Neighbor Guided Graph-level Distillation (VNGD)* mechanism. Specifically, since graphs themselves lack explicit inter-graph neighborhoods, we construct a virtual neighbor graph by selecting semantically similar samples in the representation space and establishing weighted virtual edges among them. This allows each graph to receive enriched supervisory signals from its virtual neighbors. By integrating uncertainty-based weighting and similarity-driven mixup [51], the student model benefits from more informative and stable guidance, thereby enhancing graph-level distillation performance. The virtual-neighbor graph is constructed only over labeled training graphs and is not used during validation or inference.

Let the teacher's graph-level representation for G_i be $h^g_{G_i}$. For each labeled training graph G_i, we select its k-nearest neighbors [52] (in representation space) sharing the same label as its virtual neighbors:

$$\begin{aligned} \mathcal{N}_i &= \{j \mid G_i, G_j \in \mathcal{G}_L \wedge y_i = y_j \wedge i \neq j\}, \\ \mathcal{N}_i^{\mathrm{k}} &= \text{Top-K}_{j \in \mathcal{N}_i} \left[\mathrm{sim}(h^g_{G_i}, h^g_{G_j})\right]. \end{aligned} \tag{7}$$

where y_i denotes the ground-truth label of a training graph G_i. The label condition in (7) is used only for constructing supervised virtual neighbors within $\mathcal{G}_L$. The similarity function $\mathrm{sim}(\cdot,\cdot)$ is defined as:

$$\operatorname{sim}\left(h^g_{G_i}, h^g_{G_j}\right) = \frac{\left\langle h^g_{G_i}, h^g_{G_j} \right\rangle}{\left\|h^g_{G_i}\right\|_2 \cdot \left\|h^g_{G_j}\right\|_2}, \tag{8}$$

We then construct a virtual edge set:

$$E_{virtual} = \bigcup_{i=1}^{|\mathcal{G}_L|} \left\{(i,j) \mid j \in \mathcal{N}_i^{\mathrm{k}}\right\}, \tag{9}$$

For each central-neighbor pair, define a supervision weight:

$$\Delta_{j\to i} = \frac{\exp\left[\mathrm{H}\left(z^g_{G_i}\right) + \mathrm{H}\left(z^m_{G_i}\right)\right]}{1 + \exp\left[\mathrm{H}(z^g_{G_j})\right]}, \tag{10}$$

$$\pi_{j\to i} = 1 - \exp\left(-\eta \cdot \operatorname{sim}\left(h^g_{G_i}, h^g_{G_j}\right) \cdot \Delta_{j\to i}\right), \tag{11}$$

where $G_j \in \mathcal{N}_i^{\mathrm{k}}$, $z^g_{G_i}$ and $z^m_{G_i}$ are the predicted logits of teacher and student models, $\mathrm{H}(\cdot)$ denotes entropy (to quantify prediction uncertainty), and η is a sensitivity hyperparameter, and specify that $\pi_{i\to i} = 1$. For each sample, multiple mixed samples can be synthesized using:

$$\hat{h}^g_{i,j} = \lambda\cdot\pi_{j\to i}\cdot h^g_{G_j} + (1-\lambda\cdot\pi_{j\to i})\cdot h^g_{G_i}, \lambda \sim \mathrm{Beta}(\alpha,\alpha), \tag{12}$$

where $\mathrm{Beta}(\alpha,\alpha)$ is a beta distribution parameterized by α. Mixing by $\pi_{j\to i}$ enables the teacher's confident and similar neighbors to provide more effective signals for the student. The graph-level distillation objective is:

$$\mathcal{L}_{Graph} = \frac{1}{|\mathcal{G}_L|}\sum_{i\in\mathcal{G}_L}\frac{1}{|\mathcal{N}_i^{\mathrm{k}}|}\sum_{j\in\mathcal{N}_i^{\mathrm{k}}}\left\|\frac{\hat{h}^g_{i,j}}{\left\|\hat{h}^g_{i,j}\right\|_2} - \frac{h^m_{G_i}}{\left\|h^m_{G_i}\right\|_2}\right\|_2, \tag{13}$$

where $\|\cdot\|_2$ is the L2 norm and the student MLP aligns with the mixed graph representation from the teacher GNN. During validation and inference, VNGD is not executed: no virtual-neighbor graph is constructed for $\mathcal{G}_U$, and the trained MLP experts directly make predictions using the learned ensemble weights.

2) Subgraph-level: Structural Knowledge Transfer. Subgraph-level topology can enhance GNN expressivity beyond standard k-hop neighborhoods, as evidenced by recent subgraph-aware architectures [53], [54], [28], [1]. To capture such mesoscopic structures, we follow [15] and employ Louvain clustering [55] as a lightweight default partitioning algorithm. This component is not tied to Louvain; alternative community detection or graph partitioning methods, such as Leiden [56], can also be used to instantiate the subgraph-level expert. We adopt Louvain to keep the evaluation focused on the proposed conflict-aware distillation framework rather than on optimizing the clustering algorithm itself. A READOUT function subsequently aggregates the node features within each subgraph to yield subgraph embeddings:

$$h_{S_i} = READOUT\left(\left\{h_v^{(L)} : v \in S_i\right\}\right), \tag{14}$$

where S_i denotes the i-th subgraph. The resulting subgraph representations matrix for teacher GNN and student MLP are $H^G_{sub} \in R^{C\times d_G}$ and $H^M_{sub} \in R^{C\times d_M}$, respectively, where C is the number of subgraphs, and d the embedding dimensions.

To transfer subgraph knowledge, we employ a similarity-preserving objective, inspired by representational similarity distillation [35], which transfers knowledge via the similarity matrix between subgraphs. It is formally defined as follows:

$$S_G = H^G_{sub}\cdot(H^G_{sub})^T, S_M = H^M_{sub}\cdot(H^M_{sub})^T, \tag{15}$$

and define the Frobenius norm loss:

$$\mathcal{L}_{Sub} = \left\|\frac{S_G}{\|S_G\|_2} - \frac{S_M}{\|S_M\|_2}\right\|_F^2, \tag{16}$$

3) Node-level: Fine-grained Knowledge Transfer. Since subgraph- and graph-level representations are ultimately derived from node embeddings, the node-level expert transfers fine-grained local topology from the GNN teacher to the MLP student. The distillation objective follows a neighbor-similarity principle [15], aligning teacher and student conditional neighbor distributions induced by random-walk contexts [57]. For each node v, let $\mathcal{P}_v^K$ denote the set of K-step random-walk paths starting from v, and let a sampled path be $p_v = (v_1, \cdots, v_K), v_1 = v$. For a neighbor $u \in p_v$, the conditional likelihood is defined as:

$$p(u \mid v) = \frac{\exp(h_u^T \cdot h_v)}{\sum_{w\in p_v}\exp(h_w^T \cdot h_v)}, u \in p_v, \tag{17}$$

where the distribution is computed from the corresponding teacher or student node embeddings. The node-level distillation loss aligns these conditional distributions:

$$\mathcal{L}_{Node} = \frac{1}{|\mathcal{V}|}\left[\sum_{v\in\mathcal{V}}\frac{1}{|\mathcal{P}_v^K|}\sum_{p_v\in\mathcal{P}_v^K}\mathcal{D}_{KL}\left(p(\cdot \mid v), q(\cdot \mid v)\right)\right], \tag{18}$$

where $\mathcal{D}_{KL}$ denotes the Kullback-Leibler divergence, and $p(\cdot \mid v)$ and $q(\cdot \mid v)$ denote the teacher and student conditional distributions, respectively.

B. Ensemble Multi-level

When integrating the three student models, we aim for subsequent students to focus on samples that previous students found challenging, leveraging their own learned knowledge for further refinement. Ensemble learning often relies on re-weighting samples such that subsequent learners emphasize previously misclassified instances [58]. Such re-weighting principles like boosting have been widely adopted in Graph Network and knowledge distillation [14], [59]. Motivated by this, we extend sample-adaptive weighting to graph-level multi-class distillation, we apply SAMME [60] to update the weight:

Specifically, each graph sample G_i is initialized with weight $w_i = \frac{1}{N}, N = |\mathcal{G}_L|$. For each Task-specific student model M_t, the weighted error is determined as:

$$err^{(M_t)} = \frac{\sum_{i=1}^n w_i \cdot (1 - \exp\left[-I\left(G, M_t\right)\right])}{\sum_{i=1}^n w_i}, \tag{19}$$

A key design here is the discrepancy function $I(G, M_t)$, which jointly capture sample difficulty via both KL-divergence and task loss:

$$I(G, M_t) = \mathcal{D}_{KL}\left(z^g_{G_i}, z^{M_t}_{G_i}\right) + \lambda_t\mathcal{L}_t, \tag{20}$$

where, λ_t is the loss-weight hyperparameter, and t specifies the task category as $Graph$, Sub, or $Node$. By unifying knowledge consistency and task fidelity, this discrepancy definition enables coarse-grained students to guide the boosting process, encouraging fine-grained students to concentrate on samples that remain difficult in terms of knowledge transfer. This establishes collaboration across levels rather than leaving students to operate in isolation.

The ensemble weight $\alpha^{(M_t)}$ for each student model is then:

$$\alpha^{M_t} = \log \frac{1 - err^{(M_t)}}{err^{(M_t)}} + \log(K-1), \tag{21}$$

where K denotes the number of classes. The sample weights w_i are adaptively updated based on both the $\alpha^{(M_t)}$ and $err^{(M_t)}$ and normalized:

$$w_i \leftarrow w_i \cdot \exp\left(\alpha^{M_t}\left(1 - \exp\left[-I\left(G, M_t\right)\right]\right)\right), i = 1, \cdots, N. \tag{22}$$

The distillation and ensemble losses for each student are defined as:

$$\mathcal{L}_{KD}^{M_t} = \sum_{i \in \mathcal{G}_L} w_i \cdot \left(L_{SL} + 3\lambda_t \mathcal{L}_t\right), \tag{23}$$

$$\mathcal{L}_{Ensemble} = \frac{1}{3}\left(\mathcal{L}_{KD}^{M_G} + \mathcal{L}_{KD}^{M_S} + \mathcal{L}_{KD}^{M_N}\right), \tag{24}$$

C. Training Strategy

In addition to the above distillation losses, each student model is supervised with the conventional cross-entropy loss over ground-truth graph labels:

$$\mathcal{L}_{CE}^{M_t} = \frac{1}{|\mathcal{G}_L|} \sum_{i \in \mathcal{G}_L} CE\left(\sigma(z_i^m), y_i\right), \tag{25a}$$

$$\mathcal{L}_{CE} = \frac{1}{3}(\mathcal{L}_{CE}^{M_G} + \mathcal{L}_{CE}^{M_S} + \mathcal{L}_{CE}^{M_N}), \tag{25b}$$

where $CE(\cdot)$ denotes the cross-entropy loss. The total loss function for the GEM-Distill objective is defined as:

$$\mathcal{L}_{GEMD} = \mathcal{L}_{CE} + \mathcal{L}_{Ensemble}, \tag{26}$$

The final prediction y_i for graph G_i is obtained by AdaBoost-based aggregation of predictions from all student models:

$$\tilde{y}_i = \arg\max_c \sum \left(\bar{\alpha}^{(M_t)} \sigma\left(z_{G_i}^{M_t}\right)\right), \tag{27}$$

where $\arg\max_c$ selects the highest scoring class, and $\bar{\alpha}^{(M_t)}$ denotes the normalized ensemble weight. The entire GEM-Distill algorithm is summarized in Algorithm 1.

D. Discussion

1) Multi-Grained Framework Design in GEM-Distill: To mitigate cross-granularity optimization conflicts, GEM-Distill avoids entangling all distillation objectives within a single MLP student. Instead, it decouples the distillation of graph-, subgraph-, and node-level structural knowledge into independent expert models. Inspired by curriculum learning [61] and AdaBoost-style reweighting [62], [60], these experts are organized into a coarse-to-fine sequential framework within each training epoch. Formally, each step in the curriculum can be viewed as a reweighted training distribution:

$$Q_t \propto W_t(z)P(z), \quad \forall z \in D_{train}, \tag{28}$$

where D_{train} denotes the training set, and $W_t(z)$ controls the emphasis assigned to sample z. In GEM-Distill, the graph-level expert first transfers macroscopic global semantics; subsequently, the subgraph-level expert captures mesoscopic structural patterns under the updated sample weights; finally, the node-level expert provides fine-grained local supervision for samples that remain difficult. As outlined in Algorithm 1, this "graph→subgraph→node" progression operates as an intra-epoch coordination strategy rather than three disjoint training phases.

The coupling mechanism among these experts can be interpreted through the lens of boosting. Conventional boosting constructs an ensemble model by assigning weights to sequential weak learners:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x). \tag{29}$$

GEM-Distill inherits this paradigm but tailors it for the knowledge distillation scenario. Rather than measuring sample difficulty solely by classification error, it employs a distillation-aware divergence metric defined in (20), which integrates teacher-student predictive consistency with granularity-specific distillation losses. Consequently, the weighted error, expert weights, and subsequent sample-weight updates are sequentially computed via (19)–(22). This adaptive mechanism directs subsequent experts toward hard samples that remain difficult for previous experts. Ultimately, the final prediction is aggregated from the outputs of the three experts using the learned ensemble weights in (27).

2) Complexity Analysis: The computational cost of GEM-Distill comes from teacher pretraining, preprocessing, and student training. Let $|\mathcal{G}|$ be the number of graphs, $|\bar{\mathrm{V}}|$ the average number of nodes per graph, x the input feature dimension, and h the hidden dimension. Training the GNN teacher takes $\mathcal{O}(|\mathcal{G}||\bar{\mathrm{V}}|xh)$ time. The additional preprocessing includes Louvain clustering with $\mathcal{O}(|\bar{\mathrm{V}}|\log|\bar{\mathrm{V}}|)$ time per graph and exact cosine-similarity KNN for VNGD with $\mathcal{O}(|\mathcal{G}|^2 h)$ time.

For student training, each MLP expert costs $\mathcal{O}(|\mathcal{G}||\bar{\mathrm{V}}|h)$, and the sequential ensemble introduces $\mathcal{O}(|\mathcal{G}||\bar{\mathrm{V}}|K_e)$ for sample-weight updates and expert aggregation, where $K_e = 3$ in our experiments. Node-level random-walk sampling costs $O(RT\log\bar{d})$, where R is the number of sampled paths, T is the path length, and $\bar{d}$ is the average node degree; since $\log\bar{d}$ is typically small, this term is approximated as $\mathcal{O}(RT)$. Thus, GEM-Distill introduces explicit preprocessing and training overhead compared with a single-MLP baseline, but these costs are bounded and are empirically reported in the efficiency analysis in Section V-H. During inference, the trained MLP experts make predictions using learned ensemble weights without virtual-neighbor construction, subgraph partitioning, or random-walk sampling.

Algorithm 1 Algorithm for GEM-Distill

Input: Pre-trained GNN teacher, dataset $\mathcal{G} = \mathcal{G}_L \cup \mathcal{G}_U$, labels $\mathcal{Y}$, and hyperparameters.

Output: Predictions $\hat{\mathcal{Y}}_U$, trained experts $\mathcal{M}_G, \mathcal{M}_S, \mathcal{M}_N$, and ensemble weights α^{M_l}.

Initialize $w_i = 1/N$ for each $G_i \in \mathcal{G}_L$ and initialize all MLP experts.
Compute LaPE features X_{LaPE}, set $X = \text{CONCAT}(X, X_{LaPE})$, and obtain teacher logits $\hat{\mathcal{Y}}_{SL}$ and multi-level representations.
for epoch $= 1, \ldots, T$ **do**
  Refresh training-time virtual edges via (7).
  for $l \in \{\text{Graph}, \text{Subgraph}, \text{Node}\}$ **do**
    Compute $\mathcal{L}_{CE}^{M_l}$ via (25a).
    Compute level loss $\mathcal{L}_l$: Graph uses $\pi_{j \to i}$ and $\tilde{h}_{i,j}^g$ via (11)–(13); Subgraph uses S_G, S_M via (15)–(16); Node uses sampled walks and $p(u \mid v)$ via (17)–(18).
    Compute $I(G, M_l)$, err^{M_l}, and α^{M_l} via (20)–(21), and update w_i via (22).
    Compute $\mathcal{L}_{SL}$ and $\mathcal{L}_{KD}^{M_l}$ via (4) and (23).
  end for
  Compute $\mathcal{L}_{Ensemble}$ and $\mathcal{L}_{GEMD}$ via (24) and (26), and update all experts.
end for
Predict $\hat{\mathcal{Y}}_U$ using learned ensemble weights via (27).
return $\hat{\mathcal{Y}}_U$, trained experts, and ensemble weights.

V. Experiments

In this section, we evaluate GEM-Distill by answering the following research questions: **Q1:** How does GEM-Distill compare with representative G2M distillation methods? **Q2:** Is GEM-Distill effective across different GNN teacher architectures? **Q3:** Are the gains preserved under parameter-aligned comparisons with enlarged baselines? **Q4:** How does the order of graph-, subgraph-, and node-level learning affect performance? **Q5:** What is the contribution of each component in GEM-Distill? **Q6:** How robust is GEM-Distill under dynamic graph perturbations? **Q7:** What are the training, memory, and inference costs of multi-granularity distillation? **Q8:** How sensitive is GEM-Distill to key hyperparameters?

A. Experiment Setup

1) Datasets: We evaluate GEM-Distill on nine graph-classification benchmarks: eight datasets from TUDataset [63] and the large-scale ogbg-molhiv dataset from the Open Graph Benchmark [64]. The TUDataset benchmarks include four bioinformatics datasets (PROTEINS, NCI1, BZR, and DD) and four social-network datasets (REDDIT-BINARY, IMDB-BINARY, IMDB-MULTI, and COLLAB). For social-network datasets without node attributes, we use one-hot node-degree features. IMDB-MULTI and COLLAB are multi-class tasks, and the remaining datasets are binary classification tasks. Dataset statistics are summarized in Table II.

TABLE II
Datasets Statistics

Dataset	#Tasks	#Graphs	Ave. #Nodes	Ave. #Edges
PROTEINS	2	1113	39.06	72.82
NCI1	2	4110	29.87	32.30
BZR	2	405	35.75	38.36
DD	2	1178	284.32	715.66
REDDIT-BINARY	2	2000	429.63	497.75
IMDB-BINARY	2	1000	19.77	96.53
ogbg-Molhiv	2	41127	25.5	27.5
COLLAB	3	5000	74.94	2457.78
IMDB-MULTI	3	1500	13.00	65.94

2) Baselines: We compare GEM-Distill with seven representative baselines: MLP as a feature-only reference; GLNN [12] as a soft-label G2M distillation baseline; NOSMOG [35] and AdaGMLP [14] as node-classification-oriented G2M methods adapted to graph classification; VQGraph [16] and SimMLP [37] G2M methods based on vector-quantized structural targets and self-supervised embedding alignment, respectively; and MuGSI [15] as a strong graph-classification-oriented multi-granularity G2M baseline. For fair comparison, all baselines are re-implemented under their original experimental settings, and their inputs are augmented with Laplacian eigenvector positional encoding.

3) Data Splitting and Evaluation Protocol: We employ two distinct evaluation protocols tailored to the dataset properties. For the standard TUDatasets, we follow [1] to perform 10-fold cross-validation via stratified sampling (90% train, 10% test). The teacher GNN is selected based on the highest test accuracy across folds. To ensure statistical robustness, all student models are trained independently five times, reporting the mean and standard deviation of their test accuracies. For the large-scale ogbg-molhiv dataset, we adopt the standard OGB scaffold split (80%:10%:10% for train/validation/test) [65], [66]. The model is evaluated using the ROC-AUC metric, reporting the test performance associated with the best validation ROC-AUC.

4) Hyperparameter: We select the teacher model with the best validation accuracy by grid search. For GIN, GCN, and GraphGPS, we search the number of layers $L \in \{2, 3, 5\}$, hidden dimension $H \in \{16, 32, 64, 128\}$, and dropout rate $p \in \{0, 0.5\}$. For KPGIN, we search $L \in \{2, 3, 4\}$, $p \in \{0, 0.5\}$, hops $K \in \{3, 4\}$, and combine function $F \in \{\text{attention}, \text{geometric}\}$. For MLP students, we search $L \in \{3, 4\}$ and $h \in \{32, 64, 128\}$ without dropout. All models are trained for 350 epochs with Adam [67], batch size 32, an initial learning rate of 8×10^{-3}, a decay factor of 0.6, and 30-epoch patience.

For a fair comparison, all baselines are re-implemented under the same data splits, input features, teacher GIN model, optimizer, and training schedule. The method-specific hyperparameters are searched as follows. For GLNN, we search the soft-label loss weight $\mathcal{L}_{SL} \in \{1.0, 1e\text{-}1, 1e\text{-}2, 1e\text{-}3\}$. For NOSMOG, we set $\mathcal{L}_{SL} = 1.0$, search the similarity loss weight in $\{1.0, 1e\text{-}1, 1e\text{-}2, 1e\text{-}3\}$, and do not use adversarial feature augmentation. For VQGraph, we set $\mathcal{L}_{SL} = 1.0$ and search the codebook size $K \in \{128, 256, 512, 1024\}$ and

TABLE III
COMPARISON OF ABSOLUTE AND RELATIVE IMPROVEMENTS ACHIEVED BY GEM-DISTILL AND OTHER BASELINE MODELS

	PROTEINS	BZR	DD	NCI1	IMDB-B	REDDIT-B	IMDB-M	COLLAB	ogbg-molhiv
GIN	79.25±3.22	93.09±1.89	77.67±2.86	82.43±1.12	79.60±3.02	91.35±1.58	55.40±2.82	83.34±1.40	75.16
MLP	76.46±2.71	83.05±2.63	79.25±2.50	65.69±1.53	76.60±3.40	83.39±1.87	53.73±3.28	73.86±1.72	68.35±2.13
GLNN	75.74±2.08	85.42±2.99	79.45±3.49	67.18±1.65	79.10±4.12	86.30±1.14	52.07±3.48	77.82±1.48	68.40±1.95
NOSMOG	77.09±3.16	84.18±2.73	81.15±2.50	**68.28±1.95**	78.20±4.40	85.70±2.59	53.87±3.32	78.82±1.79	71.16±1.27
VQGraph	78.36±2.88	86.27±1.89	80.22±1.97	67.83±2.08	78.80±3.88	86.65±2.31	51.27±1.62	78.86±0.82	71.05±2.10
AdaGMLP	77.90±2.80	85.65±3.13	80.89±3.97	67.88±2.17	79.40±3.66	85.85±1.80	52.87±3.61	79.22±1.38	71.43±2.49
SimMLP	76.65±2.64	83.43±3.17	80.22±3.25	65.72±1.84	77.2±3.29	83.45±2.03	53.20±3.06	77.41±1.25	72.87±0.62
MuGSI	77.63±3.45	85.68±2.56	80.53±3.37	67.23±1.12	79.40±4.12	86.70±1.72	52.73±3.57	78.38±1.51	71.93±2.45
GEM-Distill (ours)	**79.16±2.64**	**86.91±3.40**	**82.51±3.14**	68.27±1.88	**79.80±3.68**	**87.95±1.40**	**54.20±2.95**	**79.58±1.04**	**73.67±1.95**
Δ_{MLP}	2.70 (3.53%)	3.86 (4.84%)	3.26 (4.11%)	2.58 (3.93%)	3.20 (4.18%)	4.56 (5.47%)	0.47 (0.87%)	5.72 (7.74%)	5.32 (7.78%)
Δ_{GLNN}	3.42 (4.52%)	1.49 (1.74%)	3.06 (3.85%)	1.09 (1.62%)	0.70 (0.88%)	1.65 (1.91%)	2.13 (4.09%)	1.76 (2.26%)	5.27 (7.70%)
Δ_{NOSMOG}	2.07 (2.69%)	2.73 (3.24%)	1.36 (1.68%)	-0.01 (-0.01%)	1.60 (2.05%)	2.25 (2.63%)	0.33 (0.61%)	0.76 (0.96%)	2.51 (3.53%)
$\Delta_{VQGraph}$	0.80 (1.02%)	0.64 (0.74%)	2.29 (2.85%)	0.44 (0.65%)	1.00 (1.27%)	1.30 (1.50%)	2.93 (5.71%)	0.72 (0.91%)	2.62 (3.69%)
$\Delta_{AdaGMLP}$	1.26 (1.62%)	1.26 (1.47%)	1.62 (2.00%)	0.39 (0.57%)	0.40 (0.50%)	2.10 (2.45%)	1.33 (2.52%)	0.36 (0.45%)	2.24 (3.14%)
Δ_{SimMLP}	2.51 (3.27%)	3.47 (4.16%)	2.29 (2.85%)	2.55 (3.88%)	2.60 (3.37%)	4.50 (5.39%)	1.00 (1.88%)	2.17 (2.80%)	0.80 (1.10%)
Δ_{MuGSI}	1.53 (1.97%)	1.23 (1.44%)	1.98 (2.46%)	1.04 (1.55%)	0.40 (0.50%)	1.25 (1.44%)	1.47 (2.79%)	1.20 (1.53%)	1.74 (2.42%)
Δ_{GIN}	-0.09 (-0.11%)	-6.18 (-6.64%)	4.84 (6.23%)	-14.16 (-17.18%)	0.20 (0.25%)	-3.40 (-3.72%)	-1.20 (-2.17%)	-3.76 (-4.51%)	-1.49 (-1.98%)

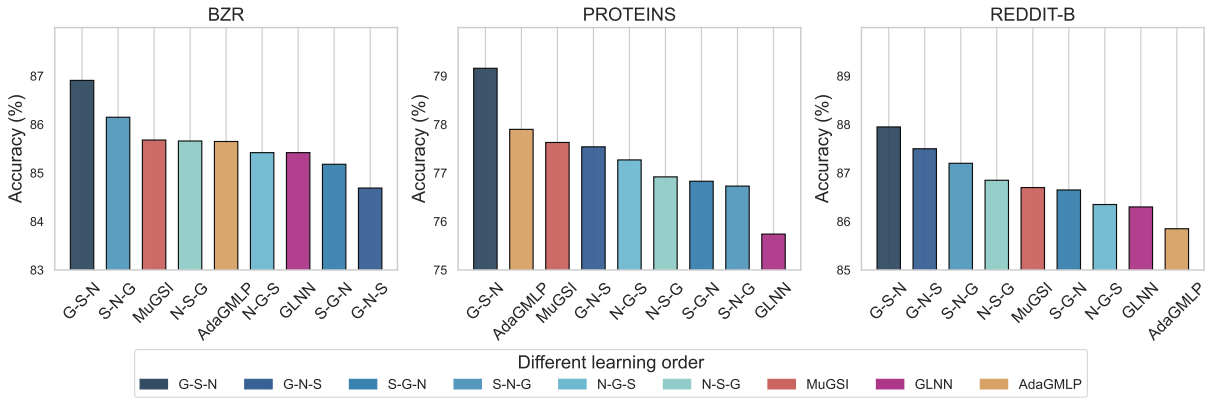

Fig. 2. The figure shows the experimental results of GEM-Distill across all possible permutations of graph-level (G), subgraph-level (S), and node-level (N) learning tasks. In the notation, G, S, and N represent graph-level, subgraph-level, and node-level tasks, respectively, and "-" indicates the order of training

the soft code assignment loss weight in $\{1.0, 1e\text{-}1, 1e\text{-}2\}$. For AdaGMLP, we search the ensemble size $K \in \{2, 3, 4\}$, balancing parameter $\lambda \in \{0.1, 0.2, \ldots, 0.9\}$, and divergence sensitivity $\beta \in \{0.5, 1, 2, 3, 4\}$, with node alignment omitted. For MuGSI, we set $\mathcal{L}_{SL} = 1.0$ and search the granularity weights (λ, μ, η) over $\{1.0, 1e\text{-}1, 1e\text{-}2\}$, $\{1.0, 1e\text{-}1, 1e\text{-}2\}$, and $\{1e\text{-}4, 1e\text{-}5\}$, respectively. For SimMLP, we set $\mathcal{L}_{SL} = 1.0$ and search the self-supervised alignment loss weight in $\{1.0, 1e\text{-}1, 1e\text{-}2, 1e\text{-}3\}$. For GEM-Distill, we use temperature $\tau = 0.4$, $k = 20$ for $\mathcal{L}_{Graph}$, $\eta \in \{1, 3, 5, 10\}$ with exponential decay and decay step 250, $\alpha = 0.25$ in (12), and random-walk path length 8 for $\mathcal{L}_{Node}$; the loss weights are searched over $\lambda_{Graph} \in \{30, 10\}$, $\lambda_{Sub} \in \{1, 1e\text{-}1, 1e\text{-}2\}$, and $\lambda_{Node} \in \{2e\text{-}1, 2e\text{-}2, 2e\text{-}3\}$. All experiments are implemented using PyTorch Geometric [68] 2.2.0, PyTorch 1.13.0, and PyTorch Lightning 1.8.1. Random-walk sampling uses NetworkX, and Louvain clustering uses the python-louvain package. Experiments are run on servers with Intel(R) Xeon(R) Platinum 8336C CPUs and four NVIDIA GeForce RTX 3090 GPUs.

B. Classification Performance Comparison (Q1)

To answer Q1, we compare GEM-Distill with representative G2M baselines, including classical methods and recent models, and further include the large-scale ogbg-molhiv benchmark. The results are reported in Table III. The best and second-best student results are highlighted in bold and underlined, respectively. For TUDatasets, we report classification accuracy; for ogbg-molhiv, we report ROC-AUC.

GEM-Distill achieves the best mean student performance on eight of the nine benchmarks and remains competitive on NCI1, where the difference from NOSMOG is only 0.01 percentage points. Its consistent gains over MLP and GLNN indicate that multi-level teacher supervision is more effective than feature-only training or soft-label-only distillation for graph classification. Compared with recent baselines, GEM-Distill improves over VQGraph by 0.44%–2.93% on TUDatasets and by 2.62 ROC-AUC points on ogbg-molhiv, and also outperforms SimMLP on all benchmarks, including a 0.80-point gain on ogbg-molhiv. Relative to MuGSI, the most directly related multi-granularity G2M baseline, GEM-Distill obtains positive mean gains on all datasets, including 1.98%, 1.20%, and 1.74 points on DD, COLLAB, and ogbg-molhiv, respectively. These results support the effectiveness of decoupling graph-, subgraph-, and node-level objectives into granularity-specific experts. At the same time, the small margins on high-variance datasets such as IMDB-BINARY and NCI1 should be interpreted cautiously; our main observation is that GEM-Distill provides the strongest overall mean performance among MLP-style students while retaining message-passing-free inference.

C. Different GNN Teacher Comparison (Q2)

TABLE IV
CLASSIFICATION ACCURACY ON DIFFERENT TEACHER ARCHITECTURES

Teacher	Student	PROTEINS	IMDB-BINARY	DD	BZR
-	MLP	76.46±2.71	76.60±3.40	79.25±2.50	83.05±2.63
GCN	-	75.75±2.65	79.30±4.17	76.48±1.37	87.16±3.42
	GLNN	76.68±3.24	77.70±4.16	79.67±3.05	83.45±3.87
	MuGSI	75.39±3.15	77.10±3.60	80.04±2.93	83.45±2.12
	AdaGMLP	74.84±3.57	78.30±2.91	80.39±2.43	85.92±4.19
	GEM-Distill	**77.36±3.28**	**78.90±2.92**	**80.64±3.52**	**86.41±4.85**
GAT	-	76.29±2.47	77.10±3.67	77.92±3.85	87.17±5.02
	GLNN	75.12±3.62	77.20±3.36	81.31±3.05	84.20±3.52
	MuGSI	75.84±2.31	76.80±3.49	80.56±2.35	84.94±3.36
	AdaGMLP	76.01±2.92	78.50±4.50	80.47±3.81	86.90±5.01
	GEM-Distill	**78.08±2.83**	**78.80±3.91**	**82.00±3.37**	**87.16±3.62**
KPGIN	-	77.27±3.02	80.3±4.37	81.07±2.83	88.40±3.26
	GLNN	76.55±3.66	73.70±3.34	76.32±3.54	83.46±3.98
	MuGSI	75.84±2.50	73.50±2.27	80.89±2.14	83.95±3.71
	AdaGMLP	78.44±3.64	72.20±2.25	78.18±3.86	85.43±2.71
	GEM-Distill	**78.99±3.90**	**74.70±2.83**	**81.49±2.33**	**86.43±3.51**
GraphGPS	-	78.45±3.81	77.6±2.63	80.3±2.39	84.45±3.07
	GLNN	73.43±3.84	73.50±3.14	81.32±2.41	78.77±1.06
	MuGSI	74.76±3.14	74.90±4.14	80.81±1.55	82.21±2.58
	AdaGMLP	75.03±3.77	72.20±2.20	81.40±2.66	79.02±1.56
	GEM-Distill	**75.74±4.00**	**75.70±2.75**	**82.26±2.15**	**83.45±2.82**

To answer Q2, we examine whether GEM-Distill is tied to a specific teacher architecture by evaluating four teachers: GCN [3], GAT [4], KPGIN [69], and GraphGPS [70]. GCN and GAT represent conventional message-passing teachers, KPGIN provides a more expressive GNN teacher, and GraphGPS serves as a Graph Transformer teacher. For each teacher, we compare GEM-Distill with GLNN, MuGSI, and AdaGMLP under the same teacher-student setting, with MLP reported as a teacher-free reference.

As shown in Table IV, GEM-Distill achieves the best student performance in all 16 teacher-dataset settings. Compared with the strongest competing student under each teacher, the average gains are 0.51, 0.83, 0.79, and 0.90 percentage points for GCN, GAT, KPGIN, and GraphGPS, respectively. The GraphGPS results are particularly informative: with a Graph Transformer teacher, GEM-Distill still obtains the highest student accuracy on PROTEINS, IMDB-BINARY, DD, and BZR, improving over the best competing student by 0.71%, 0.80%, 0.86%, and 1.24%, respectively. These results indicate that the proposed granularity-specific expert decomposition is not limited to a particular GNN teacher. They also show that a stronger teacher does not automatically yield a stronger MLP student; for example, KPGIN attains high teacher accuracy on IMDB-BINARY, whereas all MLP-based students remain clearly below the teacher. This supports the need for student-side mechanisms that improve the transfer of complex graph representations into topology-free MLPs.

D. Parameter-aligned Fairness Comparison (Q3)

To answer Q3, we construct parameter-aligned baselines to examine whether the gain of GEM-Distill mainly comes from a larger parameter budget or a direct ensemble effect. We enlarge MLP, GLNN, and MuGSI to obtain Fat-MLP, Fat-GLNN, and Fat-MuGSI, respectively, while keeping the same data splits, teacher model, optimizer, training schedule, and distillation losses as their original counterparts. We also include a 3×GLNN ensemble with a larger parameter budget than GEM-Distill. Thus, Table V compares GEM-Distill against both enlarged single-student baselines and a direct multi-MLP ensemble.

GEM-Distill achieves the highest mean performance on all five datasets. Compared with the strongest non-GEM-Distill result on each dataset, it improves the mean accuracy by 1.53%, 1.23%, 1.02%, 0.40%, and 0.63% on PROTEINS, BZR, DD, IMDB-BINARY, and NCI1, respectively. The enlarged baselines show only limited and inconsistent gains: Fat-MLP and Fat-GLNN improve on some datasets but degrade on others, and Fat-MuGSI performs worse than the original MuGSI on four of the five datasets. This suggests that simply increasing the capacity of a shared multi-granularity student does not remove the optimization conflict among graph-, subgraph-, and node-level objectives. In addition, GEM-Distill outperforms the 3×GLNN ensemble despite using fewer trainable student parameters, indicating that its advantage is not explained by parameter stacking or direct ensembling alone, but by granularity-specific expert decomposition and sequential coordination.

E. Different Learning Order Comparison (Q4)

In GEM-Distill, graph-, subgraph-, and node-level objectives are optimized by independent MLP experts. Within each epoch, these experts are processed in a predefined order, and the sample weights updated by an earlier expert are passed to the subsequent expert, enabling boosting-style coordination across complementary granular supervision. This design naturally raises Q4: how does the order of learning granularities affect GEM-Distill? To investigate this, we evaluate all six permutations of graph-, subgraph-, and node-level learning orders on BZR, PROTEINS, and REDDIT-BINARY, representing small-, medium-, and large-scale datasets, respectively. In Fig. 2, G, S, and N denote graph-, subgraph-, and node-level objectives.

The results show that the graph-subgraph-node order yields the best overall performance. This order follows a coarse-to-fine principle related to curriculum learning [61] and coarse-to-fine optimization [71]: the graph-level expert first transfers global semantics and updates the sample weights, the subgraph-level expert then refines mesoscopic structural patterns under the updated weights, and the node-level expert finally provides fine-grained local supervision for samples that remain difficult. In contrast, reversed or randomized orders, particularly those starting from node-level objectives, may emphasize noisy local signals before stable global semantics are established. Under the boosting-style reweighting mechanism [62], [60], placing finer-grained supervision later allows high-weight hard samples to receive more targeted guidance, thereby improving the coordination of complementary granular objectives.

F. Ablation Study (Q5)

1) Independent contributions of each level: To isolate the effect of granularity specialization from the number of ensemble members, all variants in Table VI use three experts and the

TABLE V
PARAMETER-ALIGNED COMPARISON BETWEEN GEM-DISTILL AND ENLARGED BASELINE MODELS

Datasets	MLP/50K	Fat-MLP/120K	GLNN/40K	Fat-GLNN/120K	MuGSI/40K	Fat-MuGSI/120K	3×GLNN/150K	GEM-Distill/120K
PROTEINS	76.46±2.71	76.20±3.02	75.74±2.08	75.12±3.01	77.63±3.45	76.38±3.18	76.06±3.07	**79.16±2.64**
BZR	83.05±2.63	83.69±3.76	85.42±2.99	82.71±3.50	85.68±2.56	84.18±3.18	84.91±4.01	**86.91±3.40**
DD	79.25±2.50	79.93±2.52	79.45±3.49	80.58±2.67	80.53±3.37	81.49±2.28	80.82±2.88	**82.51±3.14**
IMDB-BINARY	76.60±3.40	77.07±3.65	79.10±4.12	77.50±3.78	79.40±4.12	78.10±4.12	78.67±3.58	**79.80±3.68**
NCI1	65.69±1.53	65.99±1.97	67.18±1.65	66.30±2.62	67.23±1.12	66.62±2.04	67.64±2.31	**68.27±1.88**

TABLE VI
FIXED THREE-EXPERT ABLATION FOR GRANULARITY SPECIALIZATION

Dataset	GEM-Distill	3×Graph-level	3×Subgraph-level	3×Node-level	K*-S-N	G-K*-N	G-S-K*	SoftKD*
BZR	86.91±3.40	86.18±3.45	83.95±3.36	**87.16±3.24**	85.67±3.97	85.91±2.93	85.91±2.89	84.91±4.01
PROTEINS	**79.16±2.64**	77.64±2.92	77.28±3.82	78.27±3.31	77.10±4.52	77.55±3.58	76.64±3.44	76.06±3.07
DD	**82.51±3.14**	82.17±3.22	82.51±2.95	82.25±3.10	81.48±3.28	81.31±3.26	82.14±2.71	80.82±2.88
REDDIT-BINARY	87.95±1.40	86.95±1.77	**88.45±0.90**	87.95±1.23	86.65±2.08	86.65±1.47	87.10±1.52	84.40±1.70

same AdaBoosting ensemble mechanism. GEM-Distill denotes the complete G-S-N design. The 3×Graph-level, 3×Subgraph-level, and 3×Node-level variants repeat a single granularity three times, while K*-S-N, G-K*-N, and G-S-K* replace one specialized expert with a standard SoftKD expert. SoftKD* denotes an ensemble of three SoftKD experts.

The results show that granularity-specific supervision is more informative than a pure SoftKD ensemble: all variants containing graph-, subgraph-, or node-level experts outperform SoftKD* on the evaluated datasets, and replacing a specialized expert with K* generally reduces performance relative to GEM-Distill. Same-granularity ensembles can be competitive on specific datasets, such as 3×Node-level on BZR and 3×Subgraph-level on REDDIT-BINARY, indicating that the most useful structural scale may depend on dataset characteristics. However, these variants are less stable across datasets. In contrast, GEM-Distill achieves the best result on PROTEINS, matches the best mean accuracy on DD, remains competitive on BZR and REDDIT-BINARY, and consistently outperforms the SoftKD-replacement variants. These observations suggest that the gain does not come merely from using three experts or boosting-style weighting, but from coordinating complementary graph-, subgraph-, and node-level supervision while avoiding the conflict of imposing all granularities on a single student.

2) Virtual-Neighbor Guided Graph Distillation: To evaluate the contribution of VNGD in graph-level distillation, we compare it with a conventional embedding-alignment baseline. The baseline directly aligns the normalized graph embeddings of the teacher and student:

$$\mathcal{L}^{*}_{Graph} = \mathbb{E}_{G_i \sim \mathcal{D}_L} \left\| \frac{h^g_{G_i}}{\left\| h^g_{G_i} \right\|_2} - \frac{h^m_{G_i}}{\left\| h^m_{G_i} \right\|_2} \right\|_2, \qquad (30)$$

where $h^g_{G_i}$ and $h^m_{G_i}$ denote the teacher and student graph embeddings, respectively. Unlike this point-wise alignment objective, VNGD enriches graph-level supervision during training by constructing semantic virtual neighbors and mixing related training graphs.

As shown in Table VII, VNGD consistently outperforms the embedding-alignment baseline on all evaluated datasets. This indicates that semantic-neighbor-based mixup provides more informative graph-level supervision than directly matching teacher and student graph embeddings. Notably, the embedding-alignment variant remains competitive with prior baselines on most datasets, suggesting that the overall multi-granularity sequential ensemble is effective, while VNGD further strengthens the transfer of global graph semantics without adding inference-time operations.

G. Dynamic Environment Robustness (Q6)

To answer Q6, we evaluate the robustness of GEM-Distill and the teacher GNNs under dynamic structural perturbations using the first fold of REDDIT-BINARY. For each graph, we randomly remove 20 nodes and then reinsert them one by one, assuming that such limited node removal does not change the graph label. This process is repeated 20 times per graph, producing graph variants with node-missing rates from 0% to 4%. We report the average prediction error rate and the average prediction entropy after each reinsertion step; incorrect predictions are assigned maximum entropy to reflect prediction uncertainty.

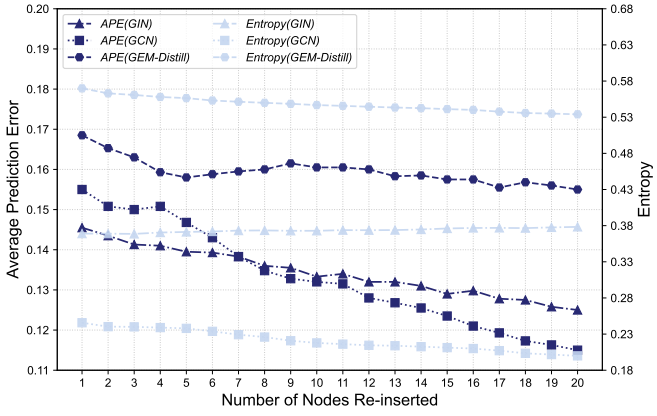

Fig. 3. Average prediction error rate and entropy of GIN, GCN, and GEM-Distill during sequential node reinsertion.

As shown in Fig. 3, GEM-Distill has slightly lower average prediction accuracy than the teacher GNNs but is

TABLE VII
EXPERIMENT RESULTS FOR DIFFERENT GRAPHKD METHODS

	PROTEINS	BZR	DD	NCI1	IMDB-B	REDDIT-B	IMDB-M	COLLAB
w/ Embedding-align	78.08±3.80	85.91±3.32	81.74±2.78	67.83±2.02	79.60±3.95	86.85±1.80	53.93±3.60	78.54±1.39
w/ VNG-Distill	**79.16±2.64**	**86.91±3.40**	**82.51±3.14**	**68.27±1.88**	**79.80±3.68**	**87.95±1.40**	**54.20±2.95**	**79.58±1.04**

TABLE VIII
COMPARISON OF TIME AND MEMORY CONSUMPTION FOR GEM-DISTILL AND OTHER BASELINES

Dataset	Metrics	Teacher GNNs			G2M Distillation Baselines					
		GIN	GraphGPS	KPGIN	GLNN	VQGraph	NOSMOG	MuGSI	AdaGMLP	**GEM-Distill**
PROTEINS	Pre-process Time (s)	0.01	14.76	86.72	1.05	1.03	1.19	1.72	1.19	1.68
	Training Time (s/Epoch)	0.85	3.47	1.78	0.34	0.62	0.39	3.07	0.59	2.96
	Evaluation Time (s/Epoch)	0.17	0.41	0.36	0.20	0.17	0.20	0.17	0.20	0.20
	Max Allocated Memory (MB)	42.36	147.42	193.30	8.19	21.65	60.54	13.06	26.64	26.21
DD	Pre-process Time (s)	0.07	140.78	1978.27	1.19	1.42	1.66	1.74	1.41	1.77
	Training Time (s/Epoch)	0.77	4.10	4.08	0.38	1.49	0.51	5.10	0.71	4.96
	Evaluation Time (s/Epoch)	0.33	1.03	0.95	0.36	0.35	0.35	0.35	0.34	0.43
	Max Allocated Memory (MB)	352.72	6409.04	2526.38	64.78	206.75	3103.90	81.15	194.37	164.37
REDDIT-BINARY	Pre-process Time (s)	0.13	321.43	71791.12	1.48	1.68	1.96	2.15	1.78	2.19
	Training Time (s/Epoch)	1.45	6.91	>24H	0.65	2.17	1.48	9.23	1.21	8.93
	Evaluation Time (s/Epoch)	0.47	1.91	–	0.47	0.44	0.43	0.47	0.55	0.59
	Max Allocated Memory (MB)	577.93	5096.46	OOM	96.16	210.92	8806.80	115.14	294.80	242.04

more stable under severe perturbations. After removing 20 nodes, the average prediction accuracy of GIN and GCN decreases by 14.09% and 25.81%, respectively, whereas GEM-Distill decreases by only 6.94%. The entropy results show a similar trend: the teacher GNNs generally produce higher uncertainty under perturbations, while GEM-Distill maintains smoother predictions. These results suggest that sequential multi-granularity distillation improves robustness to dynamic topological changes by transferring complementary structural supervision across graph, subgraph, and node levels.

H. Efficiency Analysis (Q7)

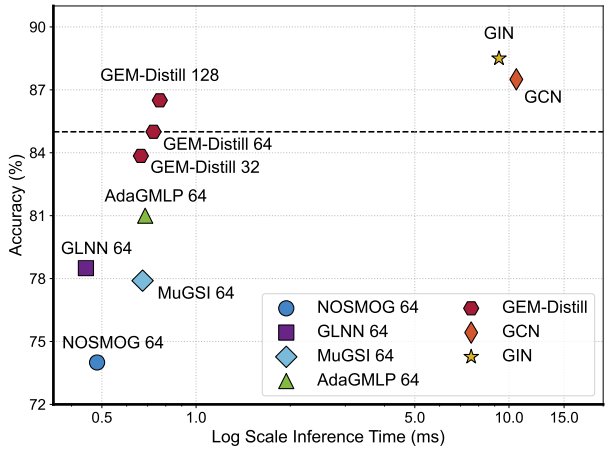

Fig. 4. Accuracy vs. Inference Time.

Fig. 4 compares classification accuracy and average CPU inference time on REDDIT-BINARY. All G2M students are configured as three-layer MLPs with hidden dimension 64 and sum pooling; AdaGMLP uses $K = 3$. Since both GEM-Distill and AdaGMLP allow independent expert inference, we report their parallel inference times for a fair comparison.

GEM-Distill requires slightly more inference time than single-MLP G2M baselines, but remains within the sub-millisecond range while achieving higher accuracy. With hidden dimension 64, GEM-Distill obtains 85.00% accuracy and an average CPU inference time of 0.73 ms, corresponding to a 12.73× speedup over GIN and a 14.44× speedup over GCN. When the hidden dimension is reduced to 32, GEM-Distill still achieves 83.85% accuracy, indicating that a smaller student can retain much of the performance. These results show that GEM-Distill provides a favorable accuracy-efficiency trade-off for MLP-style graph inference.

Table VIII reports one-time preprocessing, per-epoch training and evaluation time, and peak allocated GPU memory. Evaluation time denotes a complete held-out epoch, including data handling, host-to-device transfer, forward propagation, and metric aggregation, rather than pure inference latency. GEM-Distill has preprocessing costs comparable to MuGSI and other G2M methods but substantially lower than GraphGPS and KPGIN, showing that its KNN-based virtual-neighbor construction adds little overhead. Its training time is slightly lower than MuGSI but higher than GLNN and AdaGMLP due to its three expert-specific paths and multi-level structural objectives. Its evaluation time remains comparable to other G2M methods, with moderate overhead over single-student models from sequential three-expert execution. GEM-Distill also uses less GPU memory than AdaGMLP and substantially less than GraphGPS, KPGIN, and NOSMOG, maintaining practical efficiency while supporting multi-level knowledge transfer.

I. Hyper-parameters Analysis (Q8)

In this section, we conduct a sensitivity evaluation of the hyperparameters in GEM-Distill on both a small-scale dataset (BZR) and a large-scale dataset (REDDIT-BINARY). Specifically, we perform a grid search over the weighting coefficients for hierarchical tasks, λ_{graph}, λ_{Sub}, and λ_{Node}, with $\lambda_{graph} \in \{30, 10\}$, $\lambda_{Sub} \in \{1, 1e-1, 1e-2\}$, and

TABLE IX
COMPARISON BETWEEN THE PROPOSED SELECTION OF k AND GRID-SEARCHED KNN SIZES FOR VNGD.

Dataset	k_{sqrt}	k_{density}	Effective Range	Best Grid k	sqrt Result	Best Grid Result	Δ
PROTEINS	22	117	15–30	20	**79.62±3.49**	79.16±2.64	0.58%
BZR	13	69	15–30	20	85.17±3.88	**86.91±4.03**	2.00%
REDDIT-BINARY	25	234	10–25	20	87.10±1.64	**87.95±1.40**	0.97%

$\lambda_{Node} \in \{2e-1, 2e-2, 2e-3\}$. This results in 18 distinct hyperparameter combinations, indexed from 0 to 17. Fig. 5 presents the average of the best performance across three independent runs for each combination on both datasets.

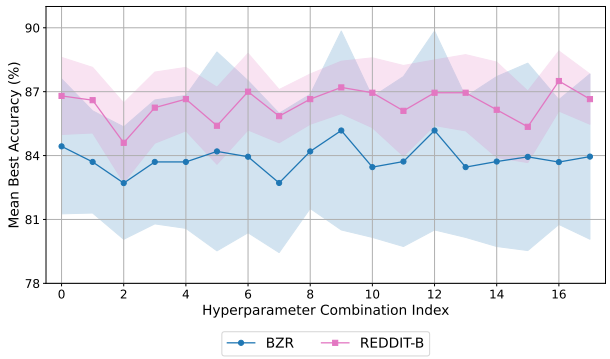

Fig. 5. Mean best accuracy for different hyper-parameter combinations for GEM-Distill.

As illustrated in Fig. 5, the hyperparameter sensitivity of the model is highly correlated between the two datasets, indicating strong robustness to parameter selection. The model is more sensitive to hyperparameter variations on the small-scale BZR dataset, as evidenced by a larger standard deviation and greater fluctuations in accuracy, likely due to the limited sample size. In contrast, the REDDIT-BINARY dataset, with a larger sample size, yields smaller standard deviations and more stable accuracy, demonstrating improved model stability on larger datasets. Overall, GEM-Distill exhibits low sensitivity to hyperparameter selection, particularly on large-scale datasets, allowing users to achieve near-optimal performance over a broad range of parameter values. Consequently, the proposed method requires minimal effort for hyperparameter tuning in practice and demonstrates strong generalization capability.

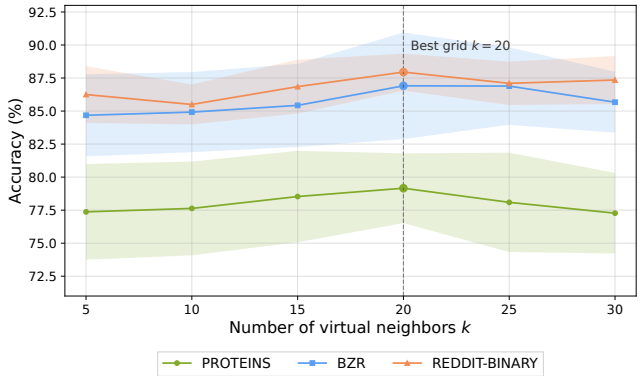

Fig. 6. Sensitivity of VNGD to the number of virtual neighbors k.

In the graph-level distillation component, the number of virtual neighbors k in the KNN construction is an important hyperparameter, since it controls the trade-off between semantic diversity and neighborhood noise. To avoid dataset-specific manual tuning, we further introduce a training-split-driven heuristic for selecting k. Let N_c denote the number of training graphs belonging to class c. We define a class-aware square-root rule as

$$k_{\text{sqrt}} = \text{clip}\Big(\text{round}\big(\sqrt{\text{median}_c\, N_c}\big), k_{\min}, k_{\max}\Big), \tag{31}$$

where $k_{\min} = 3$ and $k_{\max} = 25$ in our implementation. This rule adapts k to the typical number of available same-class training graphs while preventing the neighborhood size from growing linearly with the dataset scale. For comparison, we also examine a density-aware alternative based on teacher representations. Specifically, we compute a global same-class similarity threshold τ from the teacher graph embeddings. For each training graph G_i, we first define the reliable same-class neighbor set as

$$\mathcal{N}_i^{\tau} = \{j \mid y_j = y_i,\ j \neq i,\ \text{sim}(\mathbf{h}_i^g, \mathbf{h}_j^g) \geq \tau\}, \tag{32}$$

The density-aware estimate is then computed by

$$k_{\text{density}} = \text{clip}\left(\text{median}_i\, |\mathcal{N}_i^{\tau}|, k_{\min}, k_{\max}\right). \tag{33}$$

We evaluate different fixed choices $k \in \{5, 10, 15, 20, 25, 30\}$ on PROTEINS, BZR, and REDDIT-BINARY, and report the sensitivity curves in Fig. 6. The results show that moderate neighborhood sizes generally achieve better performance, whereas overly small neighborhoods provide insufficient intra-class diversity and overly large neighborhoods may introduce semantically distant virtual neighbors. Table IX further compares the proposed heuristic with the grid-searched effective ranges. The square-root rule selects $k = 22$, 13, and 25 on PROTEINS, BZR, and REDDIT-BINARY, respectively, which are within or close to the effective ranges observed in the grid search. In contrast, the density-aware rule selects much larger neighborhoods, indicating that same-class density alone may overestimate the number of reliable mixup partners. Therefore, we adopt the simpler and more stable square-root heuristic as the default strategy for VNGD.

VI. CONCLUSION

In this work, we presented GEM-Distill, a novel framework for graph classification that utilizes a multi-level ensemble knowledge distillation strategy from GNNs to MLPs. Our approach introduces expert student networks operating at multiple levels of granularity, which enables the comprehensive extraction and integration of diverse knowledge from the

teacher model. Extensive experiments were conducted to systematically evaluate the performance of GEM-Distill. The results demonstrate that our method achieves highly competitive performance on graph classification tasks and shows strong potential for real-world applications. Nonetheless, several limitations remain, including the need for further optimization of granularity levels, exploration of more diverse distillation objectives, and improved student ensemble strategies. We leave these aspects for future work.

REFERENCES

[1] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[2] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[3] T. Kipf, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[5] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[6] J. Gasteiger, F. Becker, and S. Günnemann, "Gemnet: Universal directional graph neural networks for molecules," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 34, pp. 6790–6802, 2021.

[7] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426.

[8] Z. Jia, S. Lin, R. Ying, J. You, J. Leskovec, and A. Aiken, "Redundancy-free computation for graph neural networks," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 997–1005.

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

[10] B. Yan, C. Wang, G. Guo, and Y. Lou, "Tinygnn: Learning efficient graph neural networks," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 1848–1856.

[11] 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.*, 2020, pp. 7074–7083.

[12] S. Zhang, Y. Liu, Y. Sun, and N. Shah, "Graph-less neural networks: Teaching old mlps new tricks via distillation," *arXiv preprint arXiv:2110.08727*, 2021.

[13] 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. web Conf.*, 2021, pp. 1227–1237.

[14] W. Lu, Z. Guan, W. Zhao, and Y. Yang, "Adagmlp: Adaboosting gnn-to-mlp knowledge distillation," in *Proc. 30th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 2060–2071.

[15] T. Yao, J. Sun, D. Cao, K. Zhang, and G. Chen, "Mugsi: Distilling gnns with multi-granularity structural information for graph classification," in *Proc. ACM Web Conf.*, 2024, pp. 709–720.

[16] L. Yang, Y. Tian, M. Xu, Z. Liu, S. Hong, W. Qu, W. Zhang, B. CUI, M. Zhang, and J. Leskovec, "VQGraph: rethinking graph representation space for bridging GNNs and MLPs," in *Proc. Int. Conf. Learn. Representations*, 2024.

[17] Q. Li, X. Li, D. Li, and J. Wang, "From gnn to mlp: Enhancing learning with knowledge-augmented distillation and confidence guidance," *Inf. Fusion*, vol. 126, p. 103518, 2026.

[18] Y. Tian, S. Xu, and M. Li, "Class-view graph knowledge distillation: A new idea for learning mlps on graphs," *Neurocomputing*, vol. 637, p. 130035, 2025.

[19] H. Du *et al.*, "Learning robust mlps on graphs via cross-layer distillation from a causal perspective," *Pattern Recognit.*, vol. 162, p. 111367, 2025.

[20] L. Xu, Z. Wang, L. Bai, S. Ji, B. Ai, X. Wang, and P. S. Yu, "Multi-level knowledge distillation with positional encoding enhancement," *Pattern Recognit.*, vol. 163, p. 111458, 2025.

[21] Y. Luo, M. McThrow, W. Y. Au, T. Komikado, K. Uchino, K. Maruhashi, and S. Ji, "Automated data augmentations for graph classification," *arXiv preprint arXiv:2202.13248*, 2022.

[22] Z. Wang and J. Fan, "Graph classification via reference distribution learning: theory and practice," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 37, pp. 137 698–137 740, 2024.

[23] L. Chen, Z. Chen, and J. Bruna, "On graph neural networks versus graph-augmented mlps," *arXiv preprint arXiv:2010.15116*, 2020.

[24] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 04, 2020, pp. 5191–5198.

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

[26] L. Yi, H. Su, X. Guo, and L. J. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2282–2290.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 29, 2016.

[28] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1024–1034.

[29] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2015, pp. 1737–1746.

[30] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.

[31] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[32] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 28, 2015.

[33] W. Zhang, X. Miao, Y. Shao, J. Jiang, L. Chen, O. Ruas, and B. Cui, "Reliable data distillation on graph convolutional network," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2020, pp. 1399–1414.

[34] L. Wu, H. Lin, Y. Huang, and S. Z. Li, "Quantifying the knowledge in gnns for reliable distillation into mlps," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2023, pp. 37 571–37 581.

[35] 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. Int. Conf. Learn. Representations*, 2022.

[36] 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.*, vol. 37, no. 9, 2023, pp. 10 351–10 360.

[37] Z. Wang, Z. Zhang, C. Zhang, and Y. Ye, "Training mlps on graphs without supervision," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2025, pp. 697–706.

[38] R. G. V, A. Singh, A. Rai, H. Pal, D. Bagotia, A. Sethi, A. Malhotra, and S. Ranu, "Tag2m- a task-agnostic knowledge distillation framework for distilling gnn to mlp," in *Proc. 31th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2025, p. 2859–2870.

[39] B.-W. Yang, M.-Y. Chang, C.-H. Lu, and C.-Y. Shen, "Two heads are better than one: Teaching mlps with multiple graph neural networks via knowledge distillation," in *Database Systems for Advanced Applications: 29th International Conference, DASFAA 2024, Gifu, Japan, July 2–5, 2024, Proceedings, Part IV*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 452–462.

[40] Y. Feng, Y. Luo, S. Ying, and Y. Gao, "LightHGNN: Distilling hypergraph neural networks into MLPs for 100x faster inference," in *Proc. Int. Conf. Learn. Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=lHasEfGsXL

[41] Z. Yu, M. Lin, W. Li, Z. Yin, S. Xiao, and S. Lu, "Demystifying gnn-to-mlp knowledge transfer: Theoretical grounding and dual-stream distillation method," in *Proc. AAAI Conf. Artif. Intell.*, vol. 40, no. 33, 2026, pp. 28 005–28 013.

[42] J. Wang, M. Zhou, S. Zhang, and Z. Gong, "Generalized few-shot node classification with graph knowledge distillation," *IEEE Trans. Comput. Soc. Syst.*, vol. 12, no. 4, pp. 1805–1815, 2025.

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

[44] Y. Zhuang, L. Lyu, C. Shi, C. Yang, and L. Sun, "Data-free adversarial knowledge distillation for graph neural networks," *arXiv preprint arXiv:2205.03811*, 2022.

[45] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[46] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *J. Mach. Learn. Res.*, vol. 24, no. 43, pp. 1–48, 2023.

[47] H. Gao and S. Ji, "Graph u-nets," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2019, pp. 2083–2092.

[48] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 31, 2018.

[49] L. Wu, Y. Liu, H. Lin, Y. Huang, and S. Z. Li, "Teach harder, learn poorer: Rethinking hard sample distillation for gnn-to-mlp knowledge distillation," in *Proc. ACM Conf. Inf. Knowl. Manag*, 2024, pp. 2554–2563.

[50] G. Xu, Z. Liu, and C. C. Loy, "Computation-efficient knowledge distillation via uncertainty-aware mixup," *Pattern Recognit.*, vol. 138, p. 109338, 2023.

[51] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2019, pp. 21–29.

[52] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[53] B. P. Chamberlain, S. Shirobokov, E. Rossi, F. Frasca, T. Markovich, N. Hammerla, M. M. Bronstein, and M. Hansmire, "Graph neural networks for link prediction with subgraph sketching," *arXiv preprint arXiv:2209.15486*, 2022.

[54] D. Zeng, W. Liu, W. Chen, L. Zhou, M. Zhang, and H. Qu, "Substructure aware graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 9, 2023, pp. 11 129–11 137.

[55] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mechanics: Theory Exp.*, vol. 2008, no. 10, p. P10008, 2008.

[56] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, p. 5233, 2019.

[57] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

[58] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.*, vol. 8, no. 4, p. e1249, 2018.

[59] K. Sun, Z. Zhu, and Z. Lin, "Adagcn: Adaboosting graph convolutional networks into deep models," *arXiv preprint arXiv:1908.05081*, 2019.

[60] J. Zhu, H. Zou, S. Rosset, T. Hastie *et al.*, "Multi-class adaboost," *Statist. Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[61] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[62] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[63] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[64] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 33, pp. 22 118–22 133, 2020.

[65] B. Ramsundar, P. Eastman, P. Walters, and V. Pande, *Deep Learning for the Life Sciences: Applying Deep Learning to Genomics, Microscopy, Drug Discovery, and More*. O'Reilly, 2019.

[66] W. Hu*, B. Liu*, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[68] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[69] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang, "How powerful are k-hop message passing graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 4776–4790.

[70] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a General, Powerful, Scalable Graph Transformer," *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 35, 2022.

[71] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12 716–12 725.

Minghao Qin Minghao Qin received the B.S. degree in applied chemistry from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2024. He is currently pursuing the M.S. degree in computer science and technology with NUAA. His research interests include deep learning, graph neural networks, and knowledge distillation.

Donghai Guan is an associate professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. He received the Ph.D. degree in Computer Engineering from Kyung Hee University (KHU), Suwon, Korea in 2009. He was a research professor in KHU during 2009-2011, assistant professor in KHU during 2012-2014. His research interests include machine learning, intelligent systems, and ubiquitous computing.

Weiwei Yuan is a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. She received a Ph.D. from the Department of Computer Engineering, Kyung Hee University, South Korea, in 2010. She was a postdoc researcher in Kyung Hee University during 2012-2014. Her current research interests include pattern recognition, image processing, and social computing.

Shuai Xu (xushuai7@nuaa.edu.cn) received the B.S. degree in computer science from Northeastern University, China, in 2014, and the Ph.D. degree in computer science from Southeast University, China, in 2020. He is currently an associate professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), China. From 2018 to 2020, he visited the University of Göttingen, Germany, where he was a Research Assistant. His research interests include multimedia data mining and computer vision, and his research works have been published in renowned journals (such as IEEE TCSS and IPM) and conferences (such as CIKM and DASFAA). Now he serves as an associate editor of IEEE Journal of Social Computing.

Çetin Kaya Koç (Life Fellow, IEEE) received a B.S. degree in Electrical Engineering from İstanbul Technical University (1980) and a Ph.D. in Electrical and Computer Engineering from the University of California Santa Barbara (1988). His research interests include cryptographic engineering, random number generators, finite field and ring arithmetic, homomorphic encryption and machine learning. Koç has positions in multiple organizations, and his research is funded by several institutions, performed within Koç Lab that includes postdoctoral researchers, PhD and MS candidates, and advanced undergraduate students. Koç is the founding editor-in-chief of the Journal of Cryptographic Engineering, published by Springer since 2011, with focus on cryptographic hardware and software development. Koç was elected as an IEEE Fellow in 2007 and IEEE Life Fellow in 2023 for his contributions to cryptographic engineering.