Mutual GNN-MLP Distillation for Robust Graph Adversarial Defense

Bowen Deng, Jialong Chen, Yanming Hu, Chuan Chen, and Tao Zhang

***Abstract*—Current adversarial defenses for graph neural networks (GNNs) face critical limitations that hinder their real-world application: 1) inadequate adaptability to graph heterophily, 2) lack of generalizability to early GNNs like GraphSAGE, and 3) low inference scalability, which is problematic for resource-constrained scenarios. To tackle these issues, we propose the mutual GNN-MLP distillation (MGMD) framework. MGMD leverages the complementary strengths of GNN and MLP, harnessing the unique advantages of both models, hence enhancing adaptability to graph heterophily and fortifying defenses against structure and/or node feature attacks. Since not intruding GNNs or MLPs, MGMD can seamlessly integrate with simple early GNNs widely used downstream. And the distilled MLP enables extremely high inference scalability. Our decision boundary analysis formally demonstrates MGMD's adversarial robustness and adaptability to graph heterophily. To mitigate potential convergence issues stemming from inductive bias conflicts between heterogeneous MLP and GNN, we introduce a novel learning rate scheduler inspired by our convergence analysis of the involved MLP. Experiments on seven homophilic and three heterophilic graphs demonstrate the effectiveness of the proposed scheduler and the advantages of MGMD over prior methods.**

***Index Terms*—Graph neural network, adversarial robustness, graph knowledge distillation, graph heterophily, semi-supervised learning**

I. Introduction

GNNS [1], [2], [3], [4], emerging as the most promising tools for graph data analysis, have been utilized in a wide array of domains, such as material discovery [5], financial risk management [6], image recognition [7], community detection [8], traffic forecasting [9], [10], suicidal ideation detection [11], and recommendation systems [12], [13]. Despite their enormous success, GNNs, like other deep learning models, are vulnerable to adversarial attacks — malicious perturbations of data [14], [15], [16]. For GNNs, perturbations to the graph structure are particularly destructive, as one edge perturbation can influence all feature dimensions. In contrast, the effects of one node feature perturbation can be mitigated or even nullified by GNNs' neighborhood aggregation [17]. Consequently, our study, like most prior works, focuses primarily on the adversarial robustness against structure-only attacks. Nevertheless, we also discuss the proposed method under node feature and joint attacks.

Bowen Deng and Tao Zhang are with the School of Systems Science and Engineering, Sun Yat-Sen University, Guangzhou, China (e-mail: dengbw3@mail2.sysu.edu.cn; zhangt358@sysu.edu.cn).

Jialong Chen, Yanming Hu, and Chuan Chen are with the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China (e-mail: chenjlong7@sysu2.edu.cn huym27@sysu2.edu.cn; chenchuan@mail.sysu.edu.cn).

Corresponding authors: Chuan Chen; Tao Zhang.

Heterophily and homophily refer to the tendency of neighboring nodes to have different and similar labels and features, respectively [18], [19], [20]. Graph learning methods designed for homophilic graphs often lose effectiveness on heterophilic graphs [21], [22], [23]. This issue is also evident in adversarial robustness research. Graph attacks modify properties especially homophily degree, leading purification defense methods to detect and counter adversarial attacks by identifying these changes and purifying the message passing process [24]. Reported alterations include decreased homophily [17], [25], [26], [27], [28], increased variance in node feature distributions [29], changed singular values of the adjacency matrix [30], [26], elevated adjacency matrix rank [26], and amplified distribution shift [31]. However, most purification methods require substantial computation to analyze edges or all node pairs to identify and mitigate suspicious graph components, leading to high costs. Critically, these attack patterns are derived from heuristics [29] or studies limited to homophilic graphs [17], [25], [26], [30], [27], [31], making their adaptability to heterophilic graphs uncertain. Our experiments, alongside [32], indicate that previous purification methods fail on heterophilic graphs, highlighting their inadequate adaptability to heterophily.

In fact, there have been attempts to understand and counter adversarial attacks on heterophilic graphs. Zhu et al. [28] highlight the distinctions in evasion attack patterns on homophilic versus heterophilic graphs, and claim that some heterophily-focused GNN designs, such as those in H_2GCN [18], enhance adversarial robustness. Lei et al. [33] observe that successful attacks increase the homophily gap between training and test graphs, and propose EvenNet to tackle this gap. Although the adversarial robustness of these models is only theoretically proven under evasion attack settings, both H_2GCN and EvenNet are empirically shown to be robust against poisoning attacks as well. However, these robust GNNs cannot be directly dropped in to replace the simple GNNs (e.g., SAGE [34]), that are already integrated into downstream application systems [35], [13], [36], [37].

Beyond adversarial robustness, inference scalability is also paramount for industrial applications. In non-graph models, inference acceleration can be achieved through techniques such as pruning and quantization, which reduce multiplication-and-accumulation (MAC) operations [38]. However, the computational savings from pruning and quantization are marginal when compared to the extensive aggregation operations ne-

cessitated by GNNs. Current purification defenses and heterophilic GNNs heavily rely on neighbor aggregation during inference, which inherently limits their scalability. To mitigate the structure dependency, GLNN [39] transfers the knowledge from a pretrained GNN to a MLP that utilizes only node features via offline GNN-to-MLP distillation. Subsequently, NOSMOG [40] incorporates relationship distillation and adversarial training on node features into GLNN. Nevertheless, relying soly on a GNN teacher renders these offline GNN-MLP methods susceptible to poisoning structure attacks.

Overall, existing methods face significant challenges.

- **Inadequate adaptability to heterophily.** Most of prior purification defenses lack considerations for heterophily, resulting in deficient adaptability to inherently heterophilic graphs.
- **Absent generalizability to simple GNNs.** The robustness mechanisms of heterophilic GNNs are inherently tied to the GNN itself and cannot be transferred to simple GNNs, such as SAGE, which are already integrated into downstream systems.
- **High inference latency.** Current (defense) methods, except for GNN-MLP distillation ones, do not eliminate the dependency on the graph structure during inference, leading to high inference latency.
- **Low adversarial robustness.** Offline GNN-MLP distillation methods with high inference scalability lack adversarial robustness.

To address the aforementioned problems, we propose a novel online mutual GNN-MLP knowledge distillation method MGMD that merges the knowledge of heterogeneous MLP and GNN. The core motivation behind this method is that heterogeneous models like MLP and GNN focus on different perspectives of graph data (node features versus graph structure), and thus contain complementary information. As shown in Table I, MLP and SAGE have distinct sets of correct predictions on both clean and perturbed graph data, and the union of these two correct prediction sets (i.e., the last row of Table I) is larger than one alone set. On the clean heterophilic UAI, we observe that MLP can outperform SAGE, with the superiority becoming even more pronounced under MetaAttack [16] attacks. These seed experimental results suggest that transferring knowledge from MLP to GNN can improve adversarial robustness and adaptability to heterophily. But traditional GNNs and distillation methods with only GNN-to-MLP direction, such as GLNN [39], cannot achieve this. By contrast, our mutual distillation framework MGMD manages to do it provably. While fusing the knowledge of GNNs and MLPs holds great promise, these heterogeneous models possess different architectures and inductive biases, potentially leading to substantially different or even conflicting predictions. Such knowledge conflicts can cause online mutual distillation to fail. And this failure is typically manifested by the model being pulled in different directions and struggling to converge well, as demonstrated by the experimental results in Section IV-F. To address this issue, we analyze the convergence of MGMD-MLP and find that learning rate decay and alternating optimization facilitate its convergence.

TABLE I
MLP AND SAGE HAVE COMPLEMENTARY KNOWLEDGE. THE TEST ACCURACY (%) ON (POISONED) HOMOPHILIC CORA AND HETEROPHILIC UAI IS REPORTED. THE DATASETS ARE INTRODUCED IN SECTION IV-B. MLP-SAGE CORRESPONDS TO THE UNION OF THE CORRECT PREDICTIONS FROM SAGE AND MLP.

	Cora	Cora-Meta-15	UAI	UAI-Meta-15
MLP	59.96	59.96	61.08	61.08
SAGE	83.50	71.38	55.09	41.24
MLP-SAGE	88.68	82.80	70.09	66.91

Consequently, we design a new learning rate scheduler, a novel joint scheduler tailored for heterogeneous GNN-MLP mutual distillation.

In our **MGMD** framework, the participating MLP and GNN are referred to as MGMD-MLP and MGMD-GNN, respectively. When incorporating SAGE [34], we refer to the variant as MGMD-SAGE. Our primary contributions can be outlined as follows.

- We introduce MGMD, an online mutual distillation framework for GNN-MLP, featuring an adversarial robustness mechanism that works on both homophilic and heterophilic graphs. MGMD integrates seamlessly with SAGE and offers inference scalability that significantly surpasses traditional GNNs (Section III-B).
- We investigate the adversarial robustness of GNN-MLP distillation methods, exposing the vulnerability of current offline methods to poisoning attacks. Our MGMD is resilient against such attacks.
- Theorem 1 demonstrates that, when any level of heterophilic graph is successfully attacked, the MLP-to-GNN distillation in MGMD can correct the compromised GNN knowledge, pushing the GNN outputs further from the optimal decision boundary compared to without MLP correction. Furthermore, Theorem 1 can accurately anticipate certain experimental outcomes (Section III-C).
- We pinpoint the optimization challenges faced by heterogeneous GNN-MLP mutual distillation and analyze the convergence of MGMD-MLP in Theorem 2. Based on this analysis, we developed a novel learning rate scheduler for MGMD (Section III-D).
- Experimental results on seven homophilic and three heterophilic graphs demonstrate the scheduler's effectiveness and highlight MGMD's superior adversarial robustness, clean accuracy, and inference scalability compared to fifteen baseline methods. Our evaluation is thorough, covering both pristine data and data compromised by several potent attacks (Section IV).

II. RELATED WORK

A. GNN Defense Methods

Of the four main types of adversarial defenses, the first is **adversarial training**. This method involves perturbing the clean adjacency matrix with techniques like random flips [14], gradient projection descent [41], or Nettack [42] during the training process, which provides some robustness against

evasion attacks. However, it can reduce training efficiency, fail to prevent poisoning attacks, and force a trade-off between robustness and clean accuracy [43]. The second type is **preprocess purification**, where vulnerable components such as high-rank adjacency SVD components [30] or dissimilar node pairs [17] are removed before reaching models. A recent method is GARNET [32], which estimates a clean graph from the top-k largest singular components minimally affected by adversarial attacks. However, its scalability is constrained by k-truncated singular value decomposition (k-TSVD). The third type is **purification by learning**. This line involves learning clean graphs during training via reducing the influence of edges connecting dissimilar nodes [17], assigning small weights to dubious nodes [29], and learning a dense adjacency matrix to reflect sparseness, low-rankness, and homophily [26]. A recent approach, STABLE [27] utlizes graph contrastive learning to learn robust node features and accordingly construct a homophilic k-nearest neighbors graph. The fourth type is **heterophilic design**. Many attack algorithms introduce heterophily into homophilic graphs to undermine homophilic GNNs. In response, GNNs designed for heterophilic graphs, such as H_2GCN [18] and EvenNet [33], can somewhat adapt to varying levels of homophily, providing inherent robustness against current adversarial attacks.

MGMD have advantages over these methods. **1)** Unlike type 1 models, it defends against both poisoning and evasion attacks without sacrificing clean accuracy for robustness. **2)** Unlike types 2 and 3, MGMD inherently adapts to heterophily without additional purification costs or significant training overhead. **3)** Compared to type 4, MGMD can be integrated into downstream applications [35], [36], [13] seamlessly, rather than requiring replacing the simple GNNs with heterophilic ones. **4)** Unlike all four types, MGMD achieves inference scalability comparable to MLPs. **5)** Most importantly, its adversarial robustness stems from counteracting the decrease in prediction score of the ground-truth class, making it applicable to any graph and any *effective* attacks.

B. GNN-MLP Distillation

MGMD introduces an innovative online and mutual style to GNN-MLP distillation, distinguishing itself from offline and unidirectional models such as GLNN [39] and NOSMOG [40]. GLNN distill the GNN knowledge to MLPs that do not rely on graph structures, by matching temperatured logits [44], [45]. However, this approach fails to fully align input node features with the label space, capture the soft structural representational similarities among nodes, and withstand node feature noise. To mitigate these issues, NOSMOG integrates structure embeddings, such as those from DeepWalk [46], into node features, distills relative node similarity, and employs projected gradient descent adversarial training (PGD-AT) [47] on node features to address noise.

The distinctions between these offline GNN-MLPs and MGMD are as follows. **1)** GLNN and NOSMOG are susceptible to poisoning structure attacks, whereas MGMD is robust against both poisoning and evasion structure attacks. **2)** Offline methods are limited by their pre-trained teachers, but MGMD overcomes this limitation through mutual distillation. **3)** MGMD trains robust GNNs and MLPs simultaneously in a single phase, avoiding the complex two-phase process required by offline distillation. **4)** MGMD addresses both structural and node feature adversarial robustness, with a focus on structural robustness, which is often more destructive and prevalent in graph machine learning but is neglected by NOSMOG. **5)** Unlike the offline methods that lack the theoretical analysis of distillation benefits, the advantages of MGMD are substantiated by Theorem 1. **6)** While offline methods are primarily used to enhance inference speed, MGMD offers additional benefits such as adversarial robustness and adaptability to heterophily, significantly expanding the range of applications for GNN-MLP distillation.

III. Methodology

A. Preliminaries

An undirected graph with $N = |\mathcal{V}|$ nodes and $M = |\mathcal{E}|$ edges are denoted as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The nonzero element $\mathbf{A}_{ij} = 1$ of adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$ represents an edge $(i,j) \in \mathcal{E}$ between node i and j. And the i-th element of diagonal degree matrix $\mathbf{D}$ is the degree of node i. Real-world graphs exhibit varying degrees of homophily, i.e., the tendency to link similar nodes. Various homophily measures exist [18], [19], [20], with the edge-based homophily [18] prevalent as the most widely adopted quantification. Given the node label vector $\mathbf{y} \in \mathbb{R}^N$, the edge-based homophily ratio (HR) is defined as the fraction of edges linking same-label nodes

$$h(\mathcal{G}, \mathbf{y}) = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \mathbb{1}(y_i, y_j) \tag{1}$$

where $\mathbb{1}(y_i, y_j) = 1$ when $y_i = y_j$ and 0 otherwise.

With the propagation matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, the GraphSAGE [34] widely used in downstream applications can be formulated as (without sampling)

$$\mathbf{H}^{(l+1)} = \phi\left(\mathbf{H}^{(l)}\mathbf{W}_1^{(l)} + \mathbf{P}\mathbf{H}^{(l)}\mathbf{W}_2^{(l)}\right), \tag{2}$$

where $\phi(\cdot)$ is the activation function, $\mathbf{H}^{(l)}$ denotes the input features of the l-th layer, and $\mathbf{W}^{(l)}$ is the weight matrix. For node i, the information is actually aggregated within the neighborhood $\mathcal{N}(i)$

$$\mathbf{h}_i^{(l+1)} = \left(\mathbf{W}_1^{(l)}\right)^\top \mathbf{h}_i^{(l)} + \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} \left(\mathbf{W}_2^{(l)}\right)^\top \mathbf{h}_j^{(l)}. \tag{3}$$

An L-layer GNN maps the input node features $\mathbf{H}^{(0)} = \mathbf{X} \in \mathbb{R}^{N\times d}$ to the predictions $\mathbf{Z} = f_\theta(\mathcal{G}) = f_\theta(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N\times C}$ over C categories.

B. Proposed Framework: MGMD

The knowledge fusion between GNN and MLP is achieved via the bidirectional knowledge distillation, which is comprised of GNN-to-MLP and MLP-to-GNN distillation, as illustrated in Fig. 1. The whole training objective $\mathcal{L}_{pro}$ of

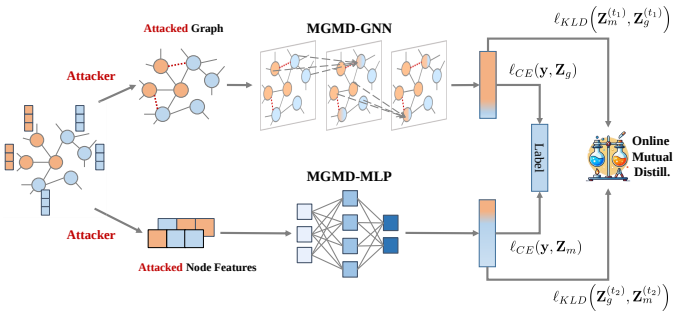


Fig. 1. The MGMD framework. During training, a GNN and an MLP collaboratively extract information from the graph structure and node features via empirical risk minimization and mutual distillation. After training, either MGMD-GNN or MGMD-MLP can be deployed.

MGMD is divide into GNN and MLP parts, i.e., Eq. (4b) and Eq. (4c),

$$\mathcal{L}_{pro} = \mathcal{L}_g + \mathcal{L}_m \tag{4a}$$

$$\mathcal{L}_g = \frac{1}{|\mathcal{V}_L|}\sum_{i\in\mathcal{V}_L} \ell_{CE}\left(y_i, [\mathbf{Z}_g]_i\right) + \frac{\alpha_1 t_1^2}{|\mathcal{V}_{obs}|}\sum_{i\in\mathcal{V}} \ell_{KLD}\left(\left[\mathbf{Z}_m^{(t_1)}\right]_i, \left[\mathbf{Z}_g^{(t_1)}\right]_i\right) \tag{4b}$$

$$\mathcal{L}_m = \frac{1}{|\mathcal{V}_L|}\sum_{i\in\mathcal{V}_L} \ell_{CE}\left(y_i, [\mathbf{Z}_m]_i\right) + \frac{\alpha_2 t_2^2}{|\mathcal{V}_{obs}|}\sum_{i\in\mathcal{V}} \ell_{KLD}\left(\left[\mathbf{Z}_g^{(t_2)}\right]_i, \left[\mathbf{Z}_m^{(t_2)}\right]_i\right), \tag{4c}$$

where ℓ_{KLD} is the Kullback-Leibler divergence (KLD), ℓ_{CE} is the cross-entropy loss, α_1 and α_2 are separately the distillation loss weights of MGMD-GNN and MGMD-MLP, $\mathbf{Z}_g$ and $\mathbf{Z}_m$ denote the prediction matrices of MGMD-GNN and MGMD-MLP, and the superscripts t_1 and t_2 of $\mathbf{Z}^{(t_1)}$ and $\mathbf{Z}^{(t_2)}$ are the temperatures of softmaxing the unnormalized predictions. $\mathcal{V}_L$ is the labeled training node set and $\mathcal{V}_{obs}$ is the set of nodes with observable features during training. These two sets are identical in inductive tasks, but different in transductive and semi-inductive tasks. More set split details of different task scenarios can be found in Section IV-A5.

1) Robustness against Structure-only Attacks: After successful simultaneous mutual distillation training guided by Eq. (4a), MGMD-GNN and MGMD-MLP usually achieve comparable acccuracy (see the experimental Section IV). Thus we prefer to deploy the latter for fast inference. Since MLPs do not need graph structures, evasion structure attacks cannot infect them. That is, *MGMD can be completely immune to evasion structure attacks.* And we hence focus on poisoning attacks here. Regarding poisoning, GNN is infected by poisoned structures during training, while MLP only absorbs clean node features. So MLP-to-GNN distillation can purify the incorrect knowledge of MGMD-GNN with the clean knowledge from MGMD-MLP (as proved in Theorem 1), imparting poisoning robustness to MGMD-GNN (and consequently to MGMD-MLP through GNN-to-MLP distillation).

2) Robustness against Joint Attacks: In addition to structure-only attacks, the complementary knowledge fusion mechanism of MGMD also makes it resistant to node feature or joint attacks. Since pure node feature attacks are inefficient to harm GNNs [17] and can be countered by directly adopting techniques (e.g., adversarial training) developed in non-graph literature [48], we move onto more ticklish joint attacks that modifies both the structure and node features.

For a target node i, an efficient joint attack algorithm typically avoids extensively modifying the features of node i's neighbors after adding perturbation edges connected to node i opting instead to attack the next target node. Similarly, if the features of node i or its neighbors are perturbed, efficient attack algorithms usually do not significantly alter the structure of node i's neighborhood, again preferring to move on to the next target node.

In the former scenario, MLP is almost always more accurate than GNN in predicting the label of node i, allowing MLP-to-GNN distillation to cleanse the poisoned GNN. In the latter scenario, GNN can significantly mitigate the impact of feature perturbations, enabling GNN-to-MLP distillation to cleanse the poisoned MLP. Therefore, MGMD exhibits robustness against joint attacks. If an attacker sacrifices attack efficiency to perturb both the structure and features within node i's neighborhood, they must use part of the budget that could have been allocated to other target nodes. This trade-off between attack surface and attack effectiveness forms the basis of MGMD's robustness, which remains resilient even under strong white-box adaptive attacks [49]. Our joint attack experiments (Sections IV-E and IV-D) provide empirical evidence for this.

Additionally, Tian et al. [40] have demonstrated that node feature adversarial training (NFAT) can effectively defend node feature noise. Integrating this strategy can further enhance robustness against node feature attacks. Notably, NFAT is orthogonal to and compatible with MGMD, meaning it can be integrated into MGMD when additional defenses are needed against node feature onslaughts. However, such integration is more concerned with practice, while this paper focuses on the theoretical basis of GNN-MLP mutual distillation. So we leave such integration for future work.

C. Theoretical Analysis of MGMD

The most significant distinction between MGMD and offline GNN-MLP distillation methods lies in the MLP-to-GNN distillation, which has often been preconceived in the past as not beneficial since MLPs usually perform much worse than GNNs on most graph tasks. So we dive into the effect of MLP-to-GNN distillation on the graphs generated with the prevalent contextual stochastic block model (CSBM) [50], [51] and our extended aCSBM, and then discuss how the resulting theorem relates to the adversarial robustness, clean accuracy, and heterophily adaptability. The benefits of MLP-to-GNN distillation are justified by Theorem 1, given auxiliary Proposition 1 and Proposition 2, and the proofs can be found in the supplementary material. Before introducing Theorem 1, we first formally describe the used random graph models (Section III-C1) and then the optimal decision boundaries of GraphSAGE absorbing these random graphs (Section III-C2).

Later, the benefits of MLP-to-GNN distillation are demonstrated by analyzing the distances between the SAGE outputs and the optimal decision boundaries (Section III-C3).

1) The CSBM and aCSBM graph data : The contextual stochastic block model (CSBM) [50] widely adopted for GNN analysis [52], [53], [51], [54], [55], [56], [31], can flexibly generate random graphs with community structures and node features. And we adopt the two-class CSBM from [51]. Let $y_i \in \{0,1\}$ be the label for node i. The edge generation probability between intra-class nodes is denoted by $0 \leq p \leq 1$ and that of inter-class ones by $0 \leq q \leq 1$. The initial features of node i is sampled from a Gaussian distribution $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}(y_i), \sigma^2\mathbf{I})$ where $\boldsymbol{\mu}(y_i) = \boldsymbol{\mu}_0$ if $y_i = 0$ otherwise $\boldsymbol{\mu}_1$. Actually $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$ ($\boldsymbol{\mu}_0 \neq \boldsymbol{\mu}_1$) are two cluster centers of raw node features. The values of p and q determine the density and homophily ratio of the generated graph $\mathcal{G} \sim \mathrm{CSBM}\left(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, p, q\right), p^2 + q^2 \neq 0, p, q \in [0,1]$. Though prevalent, CSBM tends to produce dense graphs when p or q is not small enough [51], because the edge generation is performed between all pairs of nodes. To extend our analysis to more common sparse graphs, we also consider an adapted setting where p and q denote intra-/inter-class neighbor ratios instead of generation probabilities. We denote this adapted model as $\mathcal{G} \sim \mathrm{aCSBM}\left(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, p, q\right), p + q = 1, p, q \in [0,1]$, which decouples the neighborhood sizes from p and q to enable sparsity even when p or q is large.

2) The Optimal Decision Boundary of SAGE :

Proposition 1. *Given a two-class $\{0,1\}$ CSBM graph $\mathcal{G} \sim \mathrm{CSBM}\left(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, p, q\right)$ [51] or aCSBM graph $\mathcal{G} \sim \mathrm{aCSBM}\left(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, p, q\right)$, the node feature vector of node i is sampled from a Gaussian $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{y_i}, \sigma^2\mathbf{I})$. Then the node embedding $\mathbf{h}_i$ obtained via a SAGE layer (3) follows the Gaussian distribution $\mathbf{h}_i \sim \mathcal{N}\left(\mathbb{E}_{y_i}\left[\mathbf{h}_i\right], \mathbb{D}_{y_i}\left[\mathbf{h}_i\right]\right)$, where*

$$\mathbb{E}_{y_i}\left[\mathbf{h}_i\right] = \mathbf{W}_1^\top \boldsymbol{\mu}_{y_i} + \mathbf{W}_2^\top \frac{p\boldsymbol{\mu}_{y_i} + q\boldsymbol{\mu}_{1-y_i}}{p+q} \tag{5a}$$

$$\mathbb{D}_{y_i}\left[\mathbf{h}_i\right] = \sigma^2 \mathbf{W}_1^\top \mathbf{W}_1 + \frac{\sigma^2(p^2+q^2)}{(p+q)^2} \mathbf{W}_2^\top \mathbf{W}_2. \tag{5b}$$

Proposition 2. *Following the setting in Proposition 1, the decision boundary $\mathcal{P}_h$ of the optimal linear classifier on the embedding $\mathbf{h}_i$ is the hyperplane crossing the midpoint $\mathbf{m}_h$ and orthogonal to the line connecting two cluster centers, i.e., $\mathbb{E}_0\left[\mathbf{h}_i\right]$ and $\mathbb{E}_1\left[\mathbf{h}_i\right]$.*

$$\mathcal{P}_h = \left\{\mathbf{h} \mid \mathbf{s}_h^\top \mathbf{h} - \mathbf{s}_h^\top \mathbf{m}_h\right\} \tag{6a}$$

$$\mathbf{m}_h = \frac{\left(\mathbf{W}_1^\top + \mathbf{W}_2^\top\right)\left(\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1\right)}{2} \tag{6b}$$

$$\mathbf{s}_h = \frac{\left(\mathbf{W}_1^\top + \frac{p-q}{p+q}\mathbf{W}_2^\top\right)\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right)}{\left\|\left(\mathbf{W}_1^\top + \frac{p-q}{p+q}\mathbf{W}_2^\top\right)\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\right)\right\|_2} \tag{6c}$$

3) The Robustness Benefits of MLP-to-GNN Distillation : These two propositions delineate the optimal decision boundary for a one-layer SAGE on CSBM/aCSBM graph data. We can thus justify the benefits of MLP-to-GNN distillation by proving that the SAGE trained with extra MLP-to-GNN distillation gradients produces embeddings farther away from the optimal decision boundary.

Theorem 1. *Consider the binary classification task on a CSBM or aCSBM graph (in Proposition 1) using a one-layer SAGE paired with an MLP. After one forward-backward optimization step on a node i, the SAGE outputs with and without MLP-to-GNN distillation gradients are denoted as $\mathbf{h}_i^{kd}$ and $\mathbf{h}_i$, respectively. If the MLP has a higher prediction probability than the SAGE on the ground-truth class (**MGMD Condition**), then in expectation:*

- *the optimal linear classifier defined by the decision boundary $\mathcal{P}_h$ in Proposition 2 has a lower misclassification probability on $\mathbf{h}_i^{kd}$ than $\mathbf{h}_i$ no matter how heterophilic the graph is (or is changed to be by adversarial attacks);*
- *the misclassification probability gap between using $\mathbf{h}_i^{kd}$ and $\mathbf{h}_i$ gets minimized at the demarcation point of 0.5 homophily ratio, and is maximized as the homophily ratio approaches to 0 or 1.*

Proof Sketch: The proof details can be found in the supplementary material. Denoting by $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2)$ the SAGE weights updated solely with normal supervised training loss (e.g., ℓ_{CE}) and by $\mathbf{W}^{kd} = \left(\mathbf{W}_1^{kd}, \mathbf{W}_2^{kd}\right)$ the weights updated with both normal supervised and MLP-to-GNN distillation losses (4b), the main proof path is as follows.

Step1 We consider the nodes from class 0. Proposition 1 gives the distribution of outputs $\mathbf{h}$ and $\mathbf{h}^{kd}$ obtained by feeding the raw (a)CSBM data to a SAGE layer with $\mathbf{W}$ and $\mathbf{W}^{kd}$, respectively. Proposition 2 gives the ideal optimal decision boundaries $\mathcal{P}_h$ and $\mathcal{P}_h^{kd}$ for $\mathbf{h}$ and $\mathbf{h}^{kd}$, separately.

Step2 By finding out the KLD distillation gradients in Eq. (4b), we construct the relationship between $\mathbf{W}$ and $\mathbf{W}^{kd}$, and thus that between $\mathbf{h}$ and $\mathbf{h}^{kd}$. With $\mathcal{P}_h$ and $\mathbf{h}$, we can get the probability of node i being misclassified $\mathbb{P}(\mathbf{h}_i$ is misclassified $)$ as $\mathbb{P}\left(\mathbf{s}_h^\top \mathbf{h}_i - \mathbf{s}_h^\top \mathbf{m}_h \leq 0\right)$. Then we prove that, in expectation, $\mathbb{P}(\mathbb{E}\left[\mathbf{h}_i\right]$ is misclassified $)$ is greater than $\mathbb{P}(\mathbb{E}\left[\mathbf{h}_i^{kd}\right]$ is misclassified $)$ irrespective of the graph homophily ratio. This is the first conclusion.

Step3 Lower $\mathbb{P}(\mathbb{E}\left[\mathbf{h}_i\right]$ is misclassified $)$ corresponds to higher distance $R = \mathbf{s}_h^\top \mathbb{E}\left[\mathbf{h}_i\right] - \mathbf{s}_h^\top \mathbf{m}_h$ between $\mathbb{E}\left[\mathbf{h}_i\right]$ and the optimal decision boundary $\mathcal{P}_h$. Similarly we can get R^{kd}. By examining the excess of R^{kd} over R in relation to p and q, we reach the second conclusion.

As the case for the nodes from class 0 is symmetric to that from class 1, we skip the proof for those from class 1. ■

The proof sketch of Theorem 1 serves as an instance of the framework we designed for analyzing GNN-MLP distillation. In this framework, we first examine how distillation affects the weights of GNNs and, consequently, their embeddings. We then quantify the impact of distillation by analyzing the decision boundaries of these GNN embeddings. This framework is adaptable and can be used to analyze various scenarios by modifying the GNNs involved and employing different random graph models.

Remark 1 (adversarial robustness)**.** *Per the first conclusion of Theorem 1, MLP-to-GNN distillation can shift the GNN outputs away from the decision boundary when MLP has more correct predictions, thereby mitigating or even eliminating structure poisoning effects. Although our theorem, like other GNN theories [51], [54], [33], [31], [56], is proven only on stochastic graphs, we conjecture that MGMD defends against structure poisoning attacks on real-world graphs.*

Remark 2 (heterophily adaptability)**.** *Theorem 1 shows that MLP-to-GNN distillation can enhance the participant GNN (and consequently the involved MLP via reverse distillation) when MGMD Condition is met. Since this condition is unaffected by heterophily levels, the performance gains can be guaranteed on both homophilic and heterophilic graphs once the condition is satisfied. And this condition is easy to meet on heterophilic or/and perturbed graphs.*

Remark 3 (minimal effect)**.** *The second conclusion of Theorem 1 shows that the minimal effect of MLP-to-GNN distillation occurs at a homophily ratio of 0.5, with maximum effect at extreme homophily ratios approaching 0 or 1. As shown in Table II, the homophily rate of real-world graph data is often far from 0.5. Therefore, the effect of MLP-to-GNN distillation would be considerable in practice.*

Remark 4 (accompanying response)**.** *The MGMD Condition in Theorem 1 does not require a very high MLP probability on the true class, only higher than the (expected) small value of poisoned GNN. Since more drastic attacks can amplify this probability gap and hence increase the likelihood of satisfying the MGMD Condition, we conjecture that MGMD's robustness advantages over SAGE increase with attacks become more destructive.*

Remark 5 (complementary benefits)**.** *Given their respective emphases on node attributes and structures, MLPs and GNNs may exhibit divergence in the correctly classified node sets $\mathcal{V}_{mlp}$ and $\mathcal{V}_{gnn}$. While implied by offline methods [39], [57] , GNN-to-MLP distillation can enhance MLPs on $\mathcal{V}_{gnn}$, Theorem 1 indicates that MLP-to-GNN distillation can improve GNNs on $\mathcal{V}_{mlp}$. This suggests that the mutual distillation in MGMD can confer complementary benefits.*

D. Proposed Scheduler

The MGMD optimization objective Eq. (4a) comprises four potentially conflicting parts, making the learning more challenging than alone MLP and GNN. To mitigate potential convergence issues arising from knowledge conflicts between the heterogeneous MLP and GNN, we study the convergence conditions of MGMD-MLP in Theorem 2. Based on this, we employ the cosine annealing (CA) learning rate schedule [58]. Additionally, we adopt an alternating learning strategy that silences each participant in turn to further stabilize the training. Collectively, these two points lead to our new scheduler.

Theorem 2. *Given a MGMD-MLP trained with the loss function (4c) and assume that $\exists u > 0$,*

$$\mathrm{Tr}\left\{(w_1 - w_2)^\top \left[\nabla \mathcal{L}_m(w_1) - \nabla \mathcal{L}_m(w_2)\right]\right\} \geq u\|w_1 - w_2\|_F^2, \quad (7)$$

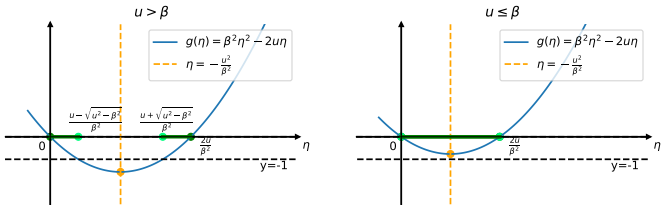


Fig. 2. Plot of $g(\eta) = \eta^2\beta^2 - 2\eta u$. The orange lines are the symmetry axes.The green segments on the horizontal axis are the feasible regions of the convergence inequality system in (8a).

one global or local optimal MLP weight w^* of the last layer can be found by gradient descent if

$$0 \leq \left(1 + \eta^2\beta^2 - 2\eta u\right) < 1 \quad (8a)$$

$$\beta = \frac{1}{|\mathcal{V}_L|}\sigma(\mathbf{H}^\top \mathbf{S}^\top \mathbf{S})\sigma(\mathbf{H}) + \frac{\alpha t_2}{N}\sigma^2(\mathbf{H}), \quad (8b)$$

where η is the learning rate, $\mathbf{H}$ is the input feature matrix of last MLP layer, $\sigma(\cdot)$ is the matrix spectral norm, and $\mathbf{S} \in \{0,1\}^{|\mathcal{V}_L| \times N}$ is the row selection matrix to extract the training node rows from the $\mathbf{X} \in \mathbb{R}^{N \times d}$ into $\mathbf{SX} \in \mathbb{R}^{|\mathcal{V}_L| \times d}$.

Proof: See the supplementary material. ■

Although Theorem 2 only considers the last MLP layer, similar quadratic inequalities on η can be derived for all layers by extending the main proof tool. Still, this simplified theorem sufficiently motivates us.

a) Cosine annealing (CA) part: Let $g(\eta) = \eta^2\beta^2 - 2\eta u$, then the convergence inequality (8a) becomes $-1 \leq g(\eta) < 0$. The zero points of $g(\eta)$ are 0 and $2u/\beta^2$, and the points $\left(\frac{u-\sqrt{u^2-\beta^2}}{\beta^2}, -1\right)$ and $\left(\frac{u+\sqrt{u^2-\beta^2}}{\beta^2}, -1\right)$ are on $g(\eta)$ if $u > \beta$. Fig. 2 illustrates two cases of $g(\eta)$, with a gap between the feasible regions in the left case. For a fixed learning rate η_0, the possibilities in the left case include:

- $\eta_0 > \frac{2u}{\beta^2}$: it does not meet the convergent condition
- $\frac{u-\sqrt{u^2-\beta^2}}{\beta^2} < \eta_0 < \frac{u+\sqrt{u^2-\beta^2}}{\beta^2}$: it does not satisfies the convergent condition
- $\frac{u+\sqrt{u^2-\beta^2}}{\beta^2} \leq \eta_0 \leq \frac{2u}{\beta^2}$ or $0 < \eta_0 \leq \frac{u-\sqrt{u^2-\beta^2}}{\beta^2}$: the learning rate η may fall outside these two regions as the training progresses, due to the variations in u and β, which alter the feasible regions.

For the right case in Fig. 2, similar issues persist, just with different feasible regions. To address these issues, a simple solution is to chose a very small η_0 close to origin such that η_0 always lies in the feasible region even if u and β change. However, this may yield intolerably slow training.

To resolve this issue, we propose to use annealing during each schedule period. With an appropriately large initial learning rate η_0 and a cold lower bound $\eta_{\min}$ near origin, it enables fast training in early epochs. Then the learning rate adapts to stay in the feasible regions for more epochs than a fixed η_0 as it can walk across or within feasible regions even if these regions change with u and β during training. Finally, η decays to $(0, \eta_{\min}], \eta_{\min} \ll \left(u - \sqrt{u^2 - \beta^2}\right)/\beta^2$, an interval insensitive to the change of u and β . For the effectiveness

of cosine annealing with warm restarts in achieving fast convergence and improved accuracy, [58], its used as the annealing part of our scheduler.

b) Quasi-alternating (QA) part: The heterogeneity between GNN and MLP models can lead to knowledge conflicts exacerbated by poisoned structures. High learning rates for both models can result in rapid, intense knowledge exchange, potentially causing confusion. In contrast, maintaining a high learning rate for one model while keeping the other's learning rate low allows the latter's knowledge to remain relatively stable and accessible. Additionally, according to Theorem 2, alternating learning can also stabilize the training dynamics of MGMD-MLP. That is, when MGMD-GNN keeps silenced and near stable knowledge, u in (7) is likely to be less oscillatory since the gradients from GNN-to-MLP distillation are less fluctuated with stable GNN knowledge. Furthermore, inspired by the alternating iterative turbo decoding [59], we conjecture that an alternating knowledge exchange mechanism can help MGMD mitigate errors induced by poisoning structure attacks.

c) The proposed learning rate scheduler: The above analysis leads to our new scheduler formulated below.

$$\eta_T = \begin{cases} \eta_{\min} + \frac{(\eta_{\max}-\eta_{\min})}{2}\left(1+\cos\left(\frac{2T_{\text{cur}}\,\pi}{T_0}\right)\right) & T_{\text{cur}} < \frac{T_0}{2} \\ \eta_{\min} & T_{\text{cur}} \geq \frac{T_0}{2} \end{cases}, \tag{9}$$

where $\eta_{\min}$ and $\eta_{\max}$ are respectively the minimal and maximal learning rates, $T_{\text{cur}} = (T+B) \mod T_0$ denotes how many epochs have gone since the last period restart, B is the offset of being preparing, and a minimal schedule period has T_0 epochs. Per one minimal period, our scheduler does annealing in the first half period and silencing in the rest epochs. In MGMD, the offsets for GNN and MLP are respectively set to $B = T_0/2$ and $B = 0$, sparking an alternating training. And in the silence phase, a small $\eta_{\min}$ is always kept, making the proposed scheduler "quasi"-alternating. An example of applying our scheduler to MGMD is shown in Fig. 3

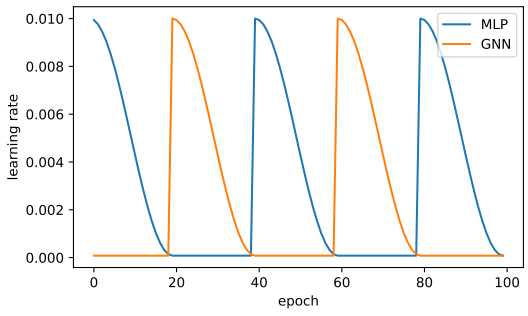


Fig. 3. Scheduler illustration with $T_0 = 20$, $\eta_{\min} = 0.00008$, and $\eta_{\max} = 0.01$. We set $B = 0$ for MGMD-MLP and $B = T_0/2$ for MGMD-GNN.

IV. Experiments

In this section, we empirically study the proposed across four aspects: adversarial robustness, clean accuracy, inference scalability, and the effectiveness of proposed scheduler.

TABLE II
THE DATA STATISTICS OF THE USED LARGEST CONNECTED COMPONENT GRAPHS. HR REFERS TO THE HOMOPHILY RATIO.

Dataset	#Nodes	#Edges	#Features	#Classes	HR
Texas	183	279	1703	5	0.061
Polblogs	1222	16714	1490	2	0.906
Citeseer	2110	3668	3703	6	0.736
Chameleon	2277	31371	2325	5	0.230
Cora	2485	5069	1433	7	0.804
CoraML	2810	7981	2879	7	0.784
ACM	3025	13128	1870	3	0.821
UAI	3067	28311	4973	19	0.364
Pubmed	19717	44338	500	3	0.802
Arxiv	169343	1157799	128	40	0.654

A. Experimental Settings

1) Datasets: We consider public graph datasets: Cora, Citeseer, Pubmed, UAI [60], ACM [61], Polblogs [62], Chameleon, Texas [19], Arxiv [63] and CoraML [64]. The statistics of the largest connected components (LLCs) of these graphs are summarized in Table II.

2) Data Splitting: Unless otherwise noted, we follow [16]. The LLC of each graph is taken and split with 10% nodes for training, 10% validation, and 80% testing. Furthermore, we repeat such 1:1:8 data splitting with 5 random seeds on each graph, and the results averaged over these 5 distinct splits are reported as the eventual performance on that graph.

3) Baselines: We compare our MGMD with the following baselines. Method only using node features: MLP. Simple GNNs: GCN [65], SAGE [34], and SGC [66]. Purification-based defenses: SVD [30], Jaccard [17], RGCN [29], Guard [25], ProGNN [26], STABLE [27], and GARNET [67] Heterophilic GNNs: GPRGNN [55] and EvenNet [33]. Offline GNN-MLP distillation: GLNN [39]. Random propagation: GRAND [68]. Robust aggregation: Soft-median-GDC (SM-GDC) [69].

4) Evaluation Metrics & Attack Methods: The primary metrics for evaluation are node classification accuracy and model inference speed. Robustness is measured by the classification accuracy on perturbed graphs, which are generated with several powerful attacks: MetaAttack [16], GR-BCD [69] for large graph, the SOTA graph injection attack G^{2}A2C [70], an adaptive attack (AA) we designed following [49]. We abbreviate settings like Cora-Meta-20, which stands for the Cora dataset attacked by MetaAttack with a budget[1] of 20%.

5) Transductive vs. Semi-inductive: In the *transductive setting*, the supervised signals come from the training set $\mathcal{V}_L$ while the distillation signals are from $\mathcal{V}_{obs} = \mathcal{V}_L \cup \mathcal{V}_{val} \cup \mathcal{V}_{test}$, where $\mathcal{V}_L$, $\mathcal{V}_{val}$, and $\mathcal{V}_{test}$ are the training, validation, and test node sets, respectively. In the *inductive setting*, distillation is performed on the observed set $\mathcal{V}_{obs} = \mathcal{V}_L \cup \mathcal{V}_{val}$ and testing is done on a disjoint $\mathcal{V}_{test}$. In the *semi-inductive setting*, $\mathcal{V}_{test}$ is further divided into two disjoint observed and inductive subsets $\mathcal{V}_{test} = \mathcal{V}_{test}^{trans} \cup \mathcal{V}_{test}^{ind}$. Distillation is carried out on $\mathcal{V}_{obs} = \mathcal{V}_L \cup \mathcal{V}_{val} \cup \mathcal{V}_{test}^{trans}$. For instance, the production setting proposed in GLNN [39] is semi-inductive. Across all

[1]A budget of 5% (15%) means the attacker can flip $0.05|\mathcal{E}|$ ($0.15|\mathcal{E}|$) entries in the adjacency matrix.

three settings, $\mathcal{V}_L$, $\mathcal{V}_{val}$, $\mathcal{V}_{test}^{ind}$, and $\mathcal{V}_{test}^{trans}$ are disjoint, and we report the accuracy on $\mathcal{V}_{test}$ when the model performs best on the validation set $\mathcal{V}_{val}$. Under *inductive* and *semi-inductive settings* with inductive test nodes, the edges between $\mathcal{V}_{obs}$ and these nodes are not observable and used during training.

B. The Robustness against Structure-only MetaAttack

MetaAttack (with a GCN surrogate) [16], an effective meta-learning-based [71] method, is adopted by almost all GNN defense works for robustness evaluation. For each graph split, we run the MetaAttack implemented in DeepRobust [72] at 5%, 10%, 15%, and 20% attack budgets. The accuracy under a budget, e.g., 20%, is averaged over the results from 5 random splits. Table III presents the results on two homophilic and two heterophilic graphs at 5% and 15% attack budgets. The full MetaAttack results on eight graphs with 5%, 10%, 15%, and 20% budgets are in the supplementary material. Since MLPs ignore graph structures, each MLP row has an identical performance across budgets. So we only report MLP at the budget of 5%.

As shown by Tables III, MGMD-MLP and MGMD-SAGE rank first or second on all attacked real-world graphs, supporting the robustness declared in Remark 1. Further, the improvement of MGMD over standalone SAGE grows with the attack budget, e.g. from around 3.5% at Meta-5 to about 14.9% at Meta-15 on Polblogs (Table III), confirming Remark 4. On the extremely heterophilic Texas, the defenses with homophilic GNNs fail with 53% accuracy. In contrast, the heterophily-adapted EvenNet outperforms them by >9% across all budgets, and our MGMD-MLP exceeds EvenNet by about 10%. In fact, significant advantages can be observed on almost all homophilic and heterophilic graphs, validating the heterophily adaptability declared in Remark 2.

C. The Robustness against Structure-only GR-BCD

Gradient-based attacks require dense adjacency matrices, which can lead to intractable time and space complexity on large graphs. To reduce the costs, PR-BCD and GR-BCD [69] sample only a block of the adjacency matrix at each gradient descent step. Though GR-BCD is a greedy relaxation of PR-BCD, their attack effects are quite similar. Thus we evaluate models with more practical GR-BCD. We use the pre-attacked datasets provided by [69], and the high accuracy and low training time of MGMD on large Arxiv dataset is reported in Table IV. The results demonstrates the training scalability and adversarial robustness of MGMD. Such scalablility is reasonable: while the inference scalability of MGMD intrinsically comes from the distilled MLP, the extra training overhead of MGMD is also negligible since only one extra MLP is trained.

D. The Robustness against Joint Injection Attack

Most attacks focus on modifying existing clean structures. However, in many scenarios (e.g., social media), attackers cannot add or remove existing edges among nodes, but can easily create new nodes and interactions. This more realistic setting motivates graph injection attacks (GIAs), which link virtual nodes (with virtual features) into the original graph to fool GNNs. Considering that such attacks are more likely to occur, we evaluate MGMD's robustness against the SOTA GIA G^{2}A2C [70]. In a GIA, a virtual node v is first added for a target node u and then connected to the original graph. Since the injected nodes have fake node features, GIAs are inherently joint attacks. We adopt the default attack setting from [70], which considers the full graph instead of the largest connected component (LCC), follows the public fixed semi-supervised split [65], and links the virtual node via one edge to the graph after generating it for one target node.

The misclassification rate is reported in Table V, demonstrating MGMD's robustness against G^{2}A2C. This result is expected because our defense mechanism envelops GIAs. During neighborhood aggregation, the embedding of target node u will be interfered by the virtual node v. Its score on the correct category will decrease significantly if the GIA is successful. According to Theorem 1, one key to MGMD's robustness is that the MLP can provide relatively clean information. The point becomes whether the MLP is affected by the virtual node v to the extent that the MLP has a lower ground-truth score than the GNN and thus becomes unqualified to provide clean information. Since the injected nodes are not included in the original set $\mathcal{V}$, they do not affect the training of MGMD-MLP and the node features of target node u, meaning that MGMD-MLP is still qualified to provide clean information about u.

E. The Robustness against Joint Adaptive Attack

Mujkanovic et al. [49] find that many graph defenses lose robustness under adaptive attack settings, where attackers have perfect knowledge of the model, parameters, data, and defense measures. Although this assumption of full knowledge is not very realistic, such settings can reflect robustness under worst-case scenarios and are thus important to robustness evaluation.

Adaptive attacks aim to bypass defenses, and the core defense mechanism of MGMD is mutual distillation. We design an adaptive attack following [49]. To undermine the mutual distillation mechanism, we use Eq. (4a) as the attack loss and make attack decisions with PGD [73], [41]. Additionally, considering the importance of node features to MGMD, we modify the existing implementation [49] to enable joint attacks on node features and graph structure.

We use the pre-split LCC datasets provided by [49]. The poisoning adaptive robustness over 5, 10, 15, 20% budgets are considered and reported in Fig. 4, from which we can see the accuracy of MGMD only slightly drops with the attack budgets increasing though the defend mechanism has already been exposed to attackers. And this is consistent with our discussion in Section III-B that MGMD based on complementary knowledge fusion, possesses robustness against feature-only and joint attacks.

F. The Effectiveness of the Proposed Scheduler

To validate the effectiveness of cosine annealing (CA) and quasi-alternating (QA) learning designed in Section

TABLE III
THE ROBUSTNESS RESULTS (%) ON FOUR DATASETS ATTACKED BY METAATTACK. THE TOP TWO PERFORMING MODELS ARE HIGHLIGHTED IN BOLD, WITH THE BEST FURTHER UNDERLINED.

	Polblogs (HR=0.906)		Citeseer (HR=0.736)		UAI (HR=0.364)		Texas (HR=0.061)	
	5%	15%	5%	15%	5%	15%	5%	15%
MLP	52.21±0.61		66.01±1.37		61.74±2.11		65.71±4.42	
MLPw4	51.72±0.87		66.43±1.73		62.71±1.91		68.98±3.32	
GCN	77.18±1.76	67.53±0.99	72.03±1.23	64.74±2.70	56.72±4.68	54.22±3.17	49.25±5.43	49.39±2.29
SGC	77.71±1.79	66.95±1.36	71.94±1.31	64.51±2.44	58.78±3.34	56.52±2.64	53.88±2.23	55.24±2.12
SAGE	90.39±0.66	77.34±3.74	72.68±1.25	70.40±1.05	60.02±3.21	60.18±2.65	62.99±3.39	64.35±2.99
RGCN	75.42±1.29	66.18±0.64	71.71±2.04	64.02±1.90	49.89±2.85	48.40±2.74	52.93±1.89	49.52±8.10
SVD	92.43±0.70	73.44±1.77	69.82±0.86	65.15±2.01	48.65±1.14	44.87±1.18	49.66±4.02	48.57±5.66
Jaccard	50.88±1.69	50.88±1.69	72.18±1.81	66.96±2.71	54.08±4.18	50.64±2.69	49.25±5.43	49.39±2.29
Guard	51.58±0.57	51.58±0.57	69.79±1.24	67.35±0.62	20.28±10.99	20.36±8.27	48.03±12.96	47.76±11.40
ProGNN	85.97±5.16	72.78±3.43	71.60±1.84	65.12±2.38	49.22±5.22	38.43±11.55	47.89±10.06	45.31±14.47
STABLE	92.80±2.38	88.55±0.38	74.33±1.08	73.32±1.14	51.78±2.08	47.63±2.26	52.27±2.82	50.52±3.24
EvenNet	87.04±1.45	68.06±1.50	74.08±1.02	70.95±1.71	67.8±2.029	66.91±2.18	62.45±2.70	63.27±2.85
GPR	69.45±1.08	56.13±2.01	73.40±1.04	69.82±1.89	35.38±9.24	34.75±11.11	54.15±2.75	51.02±7.37
GPR-GARNET	72.91±0.84	59.57±1.74	74.05±0.80	74.49±1.50	32.39±5.32	28.17±2.75	54.97±6.39	57.55±3.95
GLNN	91.62±1.35	77.46±3.73	74.25±1.20	71.92±1.38	62.46±2.91	62.02±2.00	66.40±2.57	66.53±5.45
GLNNw4	91.19±1.41	77.12±3.58	74.01±1.32	71.94±1.38	62.62±2.39	62.75±1.99	67.21±2.70	66.40±4.92
MGMD-SAGE	**93.95±1.34**	**92.27±2.26**	**75.01±0.75**	**74.81±0.41**	**69.86±0.58**	**69.52±0.46**	**68.84±5.65**	**71.02±2.30**
MGMD-MLP	**93.99±0.76**	**93.95±0.34**	**75.31±1.18**	**74.79±0.64**	**68.31±0.59**	**69.10±0.45**	**72.11±2.06**	**73.20±1.53**

TABLE IV
ACCURACY AND TRAINING TIME (INCLUDING PURIFICATION TIME) ON ARXIV ATTACKED BY GR-BCD

	Arxiv		Arxiv-GR-BCD-5		Arxiv-GR-BCD-10	
	Acc. (%) ↑	Time (s) ↓	Acc. (%) ↑	Time (s) ↓	Acc. (%) ↑	Time (s) ↓
GCN	72.05	30	48.39	46	42.87	46
GCN-GARNET	68.12	1096	60.07	1161	60.06	1186
SAGE	70.46	28	61.28	29	58.34	28
SAGE-GARNET	67.21	1163	64.41	1225	63.25	1237
GPR	69.56	38	54.74	58	47.89	59
GPR-GARNET	65.67	1120	56.63	1201	53.93	1236
MGMD-SAGE	70.56	110	64.71	161	63.33	138
MGMD-MLP	70.99	110	65.12	161	64.23	138

TABLE V
MISCLASSIFICATION RATE (%) ↓ ON THE GRAPHS ATTACKED BY G^2A2C. GARNET* ADDITIONALLY PERFORMS TSVD ON THE NODE FEATURES. THE MEAN AND STD OVER FIVE RUNS ARE REPORTED. THE TOP TWO PERFORMING MODELS ARE HIGHLIGHTED IN BOLD, WITH THE BEST FURTHER UNDERLINED.

	Cora-G^2A2C	Citeseer-G^2A2C	Pubmed-G^2A2C
SAGE	21.30±0.79	34.54±2.14	48.20±4.55
EvenNet	18.26±0.83	31.06±0.36	**22.34±0.34**
GCN	20.68±1.00	37.84±0.59	42.98±2.81
GCN-GARNET	18.06±0.48	33.90±0.68	25.08±0.39
GCN-GARNET*	18.92±0.95	30.54±0.42	46.22±7.66
GPR	18.62±0.62	31.22±0.99	38.44±1.10
GPR-GARNET	18.28±1.03	31.42±0.48	25.42±0.60
GPR-GARNET*	19.54±0.46	29.84±0.77	32.76±0.88
MGMD-SAGE	**17.96±0.50**	**26.46±1.26**	**22.60±0.46**
MGMD-MLP	**18.04±0.39**	**26.32±0.52**	22.82±1.54

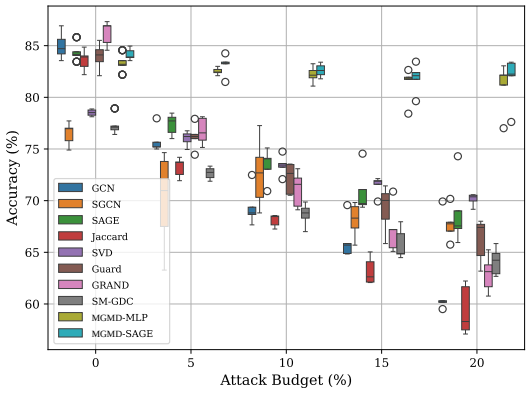
III-D, we conduct ablation experiments on multiple attacked datasets with different learning rate schedulers, including fixed scheduler (Fixed), cosine annealing scheduler (CA), and quasi-alternating scheduler (QA). The CA scheduler is adopted from [58] and is formulated as $\eta_T = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{T_{\text{cur}}}{T_0}\pi\right)\right)$, differing from our scheduler (9) only in the absence of silence time during one minimal period T_0. To solely check the QA impact, we construct the quasi-alternating scheduler that replaces the formula of first half period in (9) with $\eta_T = \eta_{\max}$.

For the experiment on each dataset, the initial learning rates of Fxied, CA, QA, and proposed schedulers are all set as $\eta_{\max}$, and the latter three schedulers share the same T_0 and $\eta_{\min}$. The results on Cora, Citeseer, Polblogs, ACM, Texas, and UAI-Meta are presented in Fig. 5. The results indicate that within the MGMD framework, using either QA or CA alone leads to better performance than not adjusting the learning rate on all graphs except Texas-Meta-20. This exception is likely due to the extreme heterophily of the Texas dataset, which drastically exacerbates the knowledge conflicts between MLP and homophilic SAGE such that using QA or CA alone is insufficient to mitigate these conflicts and instead causes interference with the better-learned MLPs. Despite this, the proposed scheduler, which combines QA and CA, significantly outperforms all other schedulers. These results demonstrate that cosine annealing and alternating learning in the proposed scheduler are generally beneficial for MGMD.

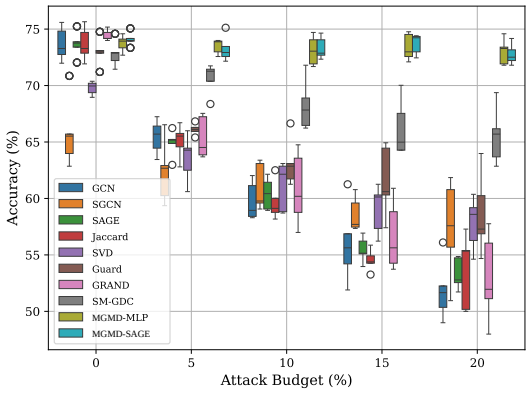
G. High Inference Scalability

Under the realistic semi-inductive setting introduced in Section IV-A5 [39], we measure the inference latency on 10 inductive test nodes and the overall accuracy on both inductive and transductive test nodes on an Ubuntu20.04 server equipped with RTX4090.

This group of experiments are conducted on perturbed graphs. For more stringent robustness evaluation, in our semi-inductive experiments, the poisoning modifications are still generated in a transductive style. That is, the attacker is aware of the full graph while making attack decisions. This attack manipulates both the training graph (supported

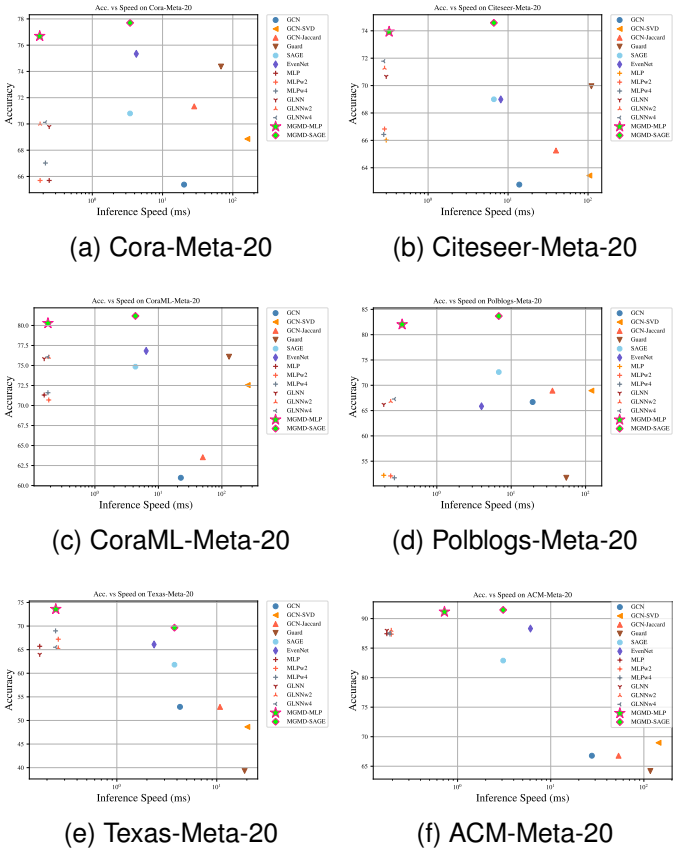


(a) Poisoning Adaptive Attack on Cora



(b) Poisoning Adaptive Attack on Citeseer

Fig. 4. The robustness against poisoning joint **adaptive attacks**

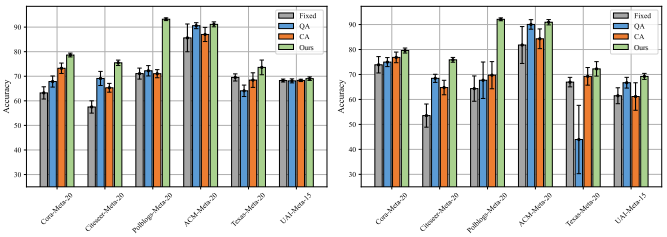


(a) Cora-Meta-20 (b) Citeseer-Meta-20

(c) CoraML-Meta-20 (d) Polblogs-Meta-20

(e) Texas-Meta-20 (f) ACM-Meta-20

Fig. 6. Acc. (%) vs. inference speed (ms) in the **semi-inductive** setting like [39]. That is 20% test nodes are excluded as inductive test nodes during training and validation while 80% test nodes are transductive ones observable across training/validation/test stages. The x-axis is logarithmically scaled.



(a) MGMD-MLP (b) MGMD-SAGE

Fig. 5. Ablation of the proposed learning rate scheduler. Fixed indicates no learning rate change, QA enables quasi-alternating learning, CA utilizes the cosine annealing with warm restart, and ours combines QA and CA.

by $\mathcal{V}_L \cup \mathcal{V}_{val} \cup \mathcal{V}_{test}^{trans}$) and the test graph (supported by $\mathcal{V}_L \cup \mathcal{V}_{val} \cup \mathcal{V}_{test}^{trans} \cup \mathcal{V}_{test}^{ind}$), so we regard it as a hybrid attack comprised of both poisoning and evasion elements.

Results on semi-inductive Cora-Meta-20, Citeseer-Meta-20, ACM-Meta-20, Texas-Meta-20, Polblogs-Meta-20, and CoraML-Meta-20 (in Fig. 6) show that MGMD-MLP achieves competitive accuracy compared to MGMD-SAGE, substantially higher than all other methods, while matching the inference speed of MLPs. Notably, MGMD-MLP infers within 1ms on all graphs, making it promising for high-throughput industrial tasks.

H. No Clean Accuracy Sacrifice

As discussed in Remark 5, Theorem 1 supports the clean accuracy improvement of MGMD. To verify this, we conduct experiments on eight clean graphs, and the results are shown in Table VI. The results indicate that MGMD-SAGE and MGMD-MLP achieve higher accuracy than standalone SAGE and MLP, respectively. This justifies our knowledge fusion motivation. And such improvements are observed across all eight graphs with varying homophily ratios, confirming Remark 2. On the extremely heterophilic Texas dataset (see Table VI), MGMD exceeds SAGE by about 9 percentage points, EvenNet by 6 percentage points, and GLNN by 5 percentage points. However, the improvement on other datasets, such as CoraML with less extreme homophily ratios, is less significant. This matches with the second conclusion of Theorem 1 and Remark 3.

I. Additional Observations

We have some interesting observations from the robustness (Table III) and clean accuracy experiments (Table VI).

- GLNNs with SAGE teachers exhibit poisoning robustness approaching SAGE, as expected with GNN-to-MLP distillation. Interestingly, GLNNs even improve over their teachers regarding adversarial robustness in many cases,

TABLE VI
ACCURACY (%) COMPARISON ON CLEAN GRAPHS. THE MEAN AND STD OVER FIVE SPLITS ARE REPORTED. THE TOP THREE PERFORMING MODELS ARE HIGHLIGHTED IN BOLD, WITH THE BEST FURTHER UNDERLINED.

	homophilic (HR>0.5)					heterophilic (HR<0.5)		
	Cora	Citeseer	Polblogs	ACM	CoraML	Texas	Chameleon	UAI
MLP	65.69±1.43	66.02±1.37	52.21±0.61	87.44±0.28	71.31±0.65	65.71±4.42	42.65±0.86	61.74±2.11
MLPw4	67.02±1.12	66.43±1.73	51.72±0.87	87.39±0.69	71.60±0.94	**68.98±3.32**	41.08±2.05	62.71±1.91
GCN	84.20±0.92	73.40±2.01	94.79±1.20	89.65±0.75	85.94±0.68	51.29±6.46	**56.69±2.68**	63.51±1.29
SGC	82.58±0.75	73.57±1.49	94.68±0.90	90.03±0.79	84.66±0.64	53.06±1.86	50.91±2.30	62.28±2.32
SAGE	83.56±0.86	74.53±1.01	94.40±0.77	90.31±0.90	84.51±1.09	64.76±1.76	51.66±2.21	60.56±3.83
RGCN	83.85±0.63	72.94±1.68	94.87±0.81	89.40±2.08	86.19±0.57	51.57±2.17	**55.74±1.46**	54.80±1.68
SVD	77.72±0.37	69.65±1.53	93.58±0.85	86.41±1.49	81.09±0.61	51.02±2.85	47.87±2.25	50.58±1.82
Jaccard	82.95±0.68	73.50±1.72	50.88±1.69	89.65±0.75	84.81±0.47	51.29±6.46	45.32±1.27	61.09±1.05
Guard	78.33±1.15	70.14±2.30	51.58±0.57	89.23±1.14	77.06±1.07	48.44±8.61	40.89±2.27	32.84±2.31
ProGNN	83.84±0.77	73.72±0.99	94.83±0.51	90.17±0.76	85.64±0.73	51.84±3.31	53.63±1.39	57.65±1.01
STABLE	83.09±0.58	74.44±0.56	94.68±0.45	85.40±0.83	83.62±0.46	50.27±4.70	46.66±1.57	56.47±0.48
EvenNet	**84.89±0.35**	74.46±0.80	**95.24±0.55**	90.54±0.57	**86.48±0.31**	67.21±1.22	51.73±1.22	**70.07±1.16**
GPR	84.43±0.83	74.88±1.23	94.68±0.43	92.13±0.70	**86.38±0.66**	56.33±8.76	51.30±0.87	34.99±1.93
GPR-GARNET	83.61±0.59	74.87±0.58	93.03±0.77	**92.35±0.48**	85.98±0.90	56.87±2.91	50.31±0.91	33.53±7.85
GLNN	83.17±0.68	75.14±0.84	94.15±0.63	91.90±0.45	84.87±0.86	67.48±3.38	48.36±2.19	62.54±3.34
GLNNw4	83.23±0.79	**75.60±0.52**	94.58±0.82	91.89±0.51	84.99±1.00	68.44±4.60	49.36±2.07	62.94±2.79
MGMD-SAGE	**84.94±0.51**	**75.20±0.70**	**95.22±0.24**	**93.15±0.86**	85.93±0.91	**72.79±3.22**	**55.88±1.12**	**69.90±0.92**
MGMD-MLP	**84.50±0.58**	**75.81±0.68**	**95.32±0.41**	**93.22±0.71**	**86.54±0.75**	**73.06±1.64**	53.43±1.45	**68.97±0.66**

e.g., by ~2% on Citeseer-Meta-15 (see Table III). This may result from the expressive power gap between GNNs and MLPs [53], [39], which makes the poisoned structure knowledge fail to fully transfer, allowing MLPs to receive less poisoned structure knowledge.

- The clean accuracy gains of MGMD over standalone models, while present, are less pronounced than the robustness improvements. As shown by Table VII, MGMD-MLP and MGMD-SAGE improve less significantly over MLP and SAGE on clean graphs than on perturbed graphs. This aligns with Theorem 1 as the MGMD Condition of higher MLP ground-truth probability is less frequently met on clean graphs, limiting distillation benefits. However, on perturbed graphs, the degraded accuracy of SAGE makes satisfying the MGMD Condition easier, enabling more corrections from MLP-to-GNN distillation.
- The improvement brought by MGMD on heterophilic graphs is more significant than on homophilic graphs, as revealed by Table VII. On homophilic graphs, SAGE is able to fully extract the information from the data, meaning most of the MLP knowledge is already acquired by SAGE. As a result, fusing the MLP knowledge does not lead to significant changes. However, on heterophilic graphs, MLP has exclusive knowledge that SAGE cannot obtain. Thus, integrating MLP knowledge in these cases brings significant improvement.

TABLE VII
THE AVERAGE ACCURACY IMPROVMENT OVER SAGE (COMPUTED FROM TABLES III AND VI)

	Clean homophilic graphs	Perturbed homophilic graphs
MGMD-SAGE	1.42	6.31
MGMD-MLP	1.62	6.81
	Clean heterophilic graphs	Perturbed heterophilic graphs
MGMD-SAGE	7.20	7.93
MGMD-MLP	6.16	8.80

V. CONCLUSIONS AND DISCUSSION

A. Conclusions

To address key limitations of existing GNN adversarial defenses, namely: **1)** inadequate adaptability to heterophily; **2)** absent generalizability to early GNNs such as SAGE; **3)** low inference scalability, this study provides an efficient and theoretically grounded solution MGMD. MGMD pioneers online and mutual GNN-MLP distillation that merges the complementary knowledge between GNNs and MLPs. It integrates seamlessly with early GNNs like SAGE and achieves inference scalability comparable to MLPs. We analyze the benefits of MLP-to-GNN distillation in Theorem 1, which indicates that MGMD-MLP can correct the wrong knowledge of MGMD-GNN, enhancing both adversarial robustness and clean accuracy irrespective of homophily ratios. Furthermore, we analyzes the potential knowledge conflicts from the perspective of convergence in Theorem 2, which inspires our new learning rate scheduler. Experiments are conducted on ten diverse graph datasets, encompassing both homophilic and heterophilic types, with various splits and attack methods. And the results demonstrate the proposed scheduler's effectiveness, highlights MGMD's exceptional inference scalability, superior adversarial robustness, and improved clean accuracy in comparison to prior methods.

B. Discussion

Beyond the proposed framework and scheduler, our study may have broader impacts on related fields. **1) New insight**.

Since standalone MLPs generally perform worse than standalone GNNs, previous works only focus on distilling the knowledge of GNNs to MLPs for fast inference, overlooking the beneficial knowledge that MLPs can contribute to GNNs. This study confirms that the knowledge from MLPs can be beneficial to and transferred to GNNs, prompting the community to reconsider the potential of MLPs beyond just accelerating inference. This broadens the application scope of GNN-MLP distillation. **2) New analysis framework**. To prove Theorem 1, we developed a novel framework for analyzing the distillation effects with consideration of graph homophily ratios. Unlike past distillation works, which lacked theoretical analysis of distillation effects, we formally proved, using the developed analysis framework, that MLP-to-GNN distillation can help SAGE produce more discriminative embeddings on both homophilic and heterophilic graphs. Many predictions (i.e., the remarks in Section III-C3) made by Theorem 1 have been validated through experiments, demonstrating the effectiveness of both Theorem 1 and the analysis framework. This framework thus provides a reference for future theoretical analysis of GNN-MLP distillation. **3) Key to heterogeneous model knowledge distillation**. MLPs and GNNs are fundamentally different models. Simply combining them in mutual distillation poses risks of failure, as shown by the fixed learning rate ablation baseline in Section IV-F. This work identifies this issue and provides an effective, validated solution through the proposed scheduler. This contribution may serve as a basic optimization style for future sophisticated mutual distillation methods.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24,, Jan. 2021.

[2] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. Yu, "Graph Self-Supervised Learning: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2022.

[3] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.

[4] X. Zheng, Y. Wang, Y. Liu, M. Li, M. Zhang, D. Jin, P. S. Yu, and S. Pan, "Graph Neural Networks for Graphs with Heterophily: A Survey," Feb. 2024.

[5] P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. van Hoesel, H. Schopmans, T. Sommer, and P. Friederich, "Graph neural networks for materials science and chemistry," *Communications Materials*, vol. 3, no. 1, pp. 1–18, Nov. 2022.

[6] S. Motie and B. Raahemi, "Financial fraud detection using graph neural networks: A systematic review," *Expert Systems with Applications*, vol. 240, p. 122156, Apr. 2024.

[7] S. Wang, X. Liu, L. Liu, W. Tu, X. Zhu, J. Liu, S. Zhou, and E. Zhu, "Highly-efficient incomplete large-scale multi-view clustering with consensus bipartite graph," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9776–9785.

[8] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, S. Y. Philip, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1149–1170, 2021.

[9] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. Macao, China: AAAI Press, Aug. 2019, pp. 1907–1913.

[10] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Virtual Event CA USA: ACM, Aug. 2020, pp. 753–763.

[11] S. Ji, S. Pan, X. Li, E. Cambria, G. Long, and Z. Huang, "Suicidal ideation detection: A review of machine learning methods and applications," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 1, pp. 214–226, 2020.

[12] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 417–426.

[13] A. Pal, C. Eksombatchai, Y. Zhou, B. Zhao, C. Rosenberg, and J. Leskovec, "Pinnersage: Multi-modal user embedding framework for recommendations at pinterest," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2020, pp. 2311–2320.

[14] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial Attack on Graph Structured Data," in *International Conference on Machine Learning*, vol. 80. PMLR, Jul. 2018, pp. 1115–1124.

[15] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial Attacks on Neural Networks for Graph Data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. London United Kingdom: ACM, Jul. 2018, pp. 2847–2856.

[16] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.

[17] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial Examples for Graph Data: Deep Insights into Attack and Defense," in *Proceedings of the International Joint Conference on Artificial Intelligence*, ijcai.org, 2019, pp. 4816–4823.

[18] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 7793–7804.

[19] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," in *International Conference on Learning Representations*, 2020.

[20] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim, "Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates Inc., 2021, pp. 20 887–20 902.

[21] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup, "Revisiting Heterophily For Graph Neural Networks," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 1362–1375. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/092359ce5cf60a80e882378944bf1be4-Paper-Conference.pdf

[22] X. Li, R. Zhu, Y. Cheng, C. Shan, S. Luo, D. Li, and W. Qian, "Finding Global Homophily in Graph Neural Networks When Meeting Heterophily," in *Proceedings of the 39th International Conference on Machine Learning*. PMLR, Jun. 2022, pp. 13 242–13 256.

[23] N. Liao, S. Luo, X. Li, and J. Shi, "LD2: Scalable heterophilous graph neural network with decoupled embeddings," in *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

[24] J. Gilmer, S. Schoenholz, P. Riley, O. Vinyals, and G. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70. PMLR, Jul. 2017, pp. 1263–1272.

[25] X. Zhang and M. Zitnik, "GNNGUARD: Defending graph neural networks against adversarial attacks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., Dec. 2020, pp. 9263–9275.

[26] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 66–74.

[27] K. Li, Y. Liu, X. Ao, J. Chi, J. Feng, H. Yang, and Q. He, "Reliable representations make A stronger defender: Unsupervised structure refinement for robust GNN," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22.

New York, NY, USA: Association for Computing Machinery, 2022, pp. 925–935.

[28] J. Zhu, J. Jin, D. Loveland, M. T. Schaub, and D. Koutra, "How Does Heterophily Impact the Robustness of Graph Neural Networks? Theoretical Connections and Practical Implications," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 2637–2647.

[29] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1399–1407.

[30] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All you need is low (rank): Defending against adversarial attacks on graphs," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, ser. WSDM '20. New York, NY, USA: ACM, 2020, pp. 169–177.

[31] K. Li, Y. Liu, X. Ao, and Q. He, "Revisiting graph adversarial attack and defense from a data distribution perspective," in *International Conference on Learning Representations*, 2023.

[32] C. Deng, X. Li, Z. Feng, and Z. Zhang, "GARNET: Reduced-Rank Topology Learning for Robust and Scalable Graph Neural Networks," in *Proceedings of the First Learning on Graphs Conference*. PMLR, Dec. 2022, pp. 3:1–3:23, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v198/deng22a.html

[33] R. Lei, Z. Wang, Y. Li, B. Ding, and Z. Wei, "EvenNet: Ignoring odd-hop neighbors improves robustness of graph neural networks," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 4694–4706.

[34] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 1025–1035.

[35] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *Proceedings of the 24th International Conference on Knowledge Discovery & Data Mining*. London United Kingdom: ACM, Jul. 2018, pp. 974–983.

[36] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Virtual Event China: ACM, Jul. 2020, pp. 639–648.

[37] D. Pujol-Perich, J. Suarez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of Graph Neural Networks for robust Intrusion Detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 4, pp. 111–117, Jun. 2022.

[38] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 126–136, Jan. 2018.

[39] S. Zhang, Y. Liu, Y. Sun, and N. Shah, "Graph-less Neural Networks: Teaching Old MLPs New Tricks via Distillation," in *International Conference on Learning Representations*, 2022.

[40] Y. Tian, C. Zhang, Z. Guo, X. Zhang, and N. Chawla, "Learning MLPs on graphs: A unified view of effectiveness, robustness, and efficiency," in *International Conference on Learning Representations*, 2023.

[41] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. Macao, China: AAAI Press, Aug. 2019, pp. 3961–3967.

[42] J. Chen, Y. Wu, X. Lin, and Q. Xuan, "Can adversarial network attack be defended?" *CoRR*, vol. abs/1903.05994, 2019.

[43] T. Pang, M. Lin, X. Yang, J. Zhu, and S. Yan, "Robustness and Accuracy Could Be Reconcilable by (Proper) Definition," in *Proceedings of the 39th International Conference on Machine Learning*. PMLR, Jun. 2022, pp. 17 258–17 277.

[44] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," Mar. 2015.

[45] M. Phuong and C. Lampert, "Towards Understanding Knowledge Distillation," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 5142–5151.

[46] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 701–710.

[47] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *International Conference on Learning Representations*, Mar. 2022.

[48] N. D. Singh, F. Croce, and M. Hein, "Revisiting adversarial training for imagenet: Architectures, training and generalization across threat models," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 13 931–13 955.

[49] F. Mujkanovic, S. Geisler, S. Günnemann, and A. Bojchevski, "Are Defenses for Graph Neural Networks Robust?" in *Advances in Neural Information Processing Systems*, May 2022.

[50] Y. Deshpande, S. Sen, A. Montanari, and E. Mossel, "Contextual Stochastic Block Models," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[51] Y. Ma, X. Liu, N. Shah, and J. Tang, "Is Homophily a Necessity for Graph Neural Networks?" in *International Conference on Learning Representations*, Mar. 2022.

[52] A. Baranwal, K. Fountoulakis, and A. Jagannath, "Effects of Graph Convolutions in Multi-layer Networks," in *The Eleventh International Conference on Learning Representations*, Sep. 2022.

[53] L. Chen, Z. Chen, and J. Bruna, "On Graph Neural Networks versus Graph-Augmented MLPs," in *The 9th International Conference on Learning Representations*, 2021.

[54] E. Chien, W.-C. Chang, C.-J. Hsieh, H.-F. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon, "Node Feature Extraction by Self-Supervised Multi-scale Neighborhood Prediction," in *International Conference on Learning Representations*, Jan. 2022.

[55] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive Universal Generalized PageRank Graph Neural Network," in *International Conference on Learning Representations*, Feb. 2022.

[56] L. Gosch, D. Sturm, S. Geisler, and S. Günnemann, "Revisiting Robustness in Graph Machine Learning," in *The Eleventh International Conference on Learning Representations*, Feb. 2023.

[57] Y. Tian, S. Pei, X. Zhang, C. Zhang, and N. V. Chawla, "Knowledge Distillation on Graphs: A Survey," Jan. 2023.

[58] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *International Conference on Learning Representations*, Nov. 2016.

[59] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[60] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective Classification in Network Data," *AI Magazine*, vol. 29, no. 3, pp. 93–93, Sep. 2008.

[61] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous Graph Attention Network," in *The World Wide Web Conference*. San Francisco CA USA: ACM, May 2019, pp. 2022–2032.

[62] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. election: Divided they blog," in *Proceedings of the 3rd International Workshop on Link Discovery*, ser. LinkKDD '05. New York, NY, USA: Association for Computing Machinery, Aug. 2005, pp. 36–43.

[63] 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," in *Advances in Neural Information Processing Systems*, 2020.

[64] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations*, 2018.

[65] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[66] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, Jun. 2019, pp. 6861–6871.

[67] C. Deng, X. Li, Z. Feng, and Z. Zhang, "GARNET: Reduced-Rank Topology Learning for Robust and Scalable Graph Neural Networks," in *Proceedings of the First Learning on Graphs Conference*. PMLR, Dec. 2022, pp. 3:1–3:23.

[68] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.
[69] S. Geisler, T. Schmidt, H. Şirin, D. Zügner, A. Bojchevski, and S. Günnemann, "Robustness of Graph Neural Networks at Scale," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., Nov. 2021, pp. 7637–7649.
[70] M. Ju, Y. Fan, C. Zhang, and Y. Ye, "Let Graph Be the Go Board: Gradient-Free Node Injection Attack for Graph Neural Networks via Reinforcement Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, Jun. 2023, pp. 4383–4390.
[71] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Jul. 2017, pp. 1126–1135.
[72] Y. Li, W. Jin, H. Xu, and J. Tang, "DeepRobust: A Platform for Adversarial Attacks and Defenses," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, May 2021, pp. 16 078–16 080.
[73] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *International Conference on Learning Representations*, Feb. 2018.
[74] T. P. Minka, "Old and new matrix algebra useful for statistics," *See www. stat. cmu. edu/minka/papers/matrix. html*, vol. 4, 2000.
[75] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
[76] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and Deep Graph Convolutional Networks," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 1725–1735.
[77] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.
[78] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.
[79] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. K. Prasanna, "GraphSAINT: Graph sampling based inductive learning method," in *International Conference on Learning Representations*, 2020.
[80] M. Fey, J. E. Lenssen, F. Weichert, and J. Leskovec, "GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings," in *Proceedings of the International Conference on Machine Learning*, vol. 139. PMLR, Jul. 2021, pp. 3294–3304.
[81] Z. Shi, X. Liang, and J. Wang, "LMC: Fast training of GNNs via subgraph sampling with provable convergence," in *The 11th International Conference on Learning Representations*, 2023.
[82] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs," in *arXiv Preprint arXiv:2103.09430*, 2021.
[83] H. Federer, *Geometric Measure Theory*. Springer, 2014.
[84] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 3839–3848.
[85] B. Gao and L. Pavel, "On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning," Aug. 2018.
[86] W. Falcon and The PyTorch Lightning team, "PyTorch lightning," Mar. 2019.
[87] M. Fey and J. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *International Conference on Learning Representations*, 2019.
[88] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," Oct. 2017.

APPENDIX A
LIST OF SYMBOLS

Part of used symbols are summarized here.

TABLE VIII
LIST OF PART OF USED SYMBOLS

Symbol	Meaning
$\mathcal{G}=(\mathcal{V},\mathcal{E})$	The clean graph
$\mathcal{E}$	The set of graph edges
$\mathcal{V}$	The set of graph nodes
$\mathcal{V}_L/\mathcal{V}_{val}/\mathcal{V}_{test}$	The set of training/validation/test nodes
$\mathcal{V}_{obs}$	The set of graph nodes observed during training
$\lvert\mathcal{V}\rvert$	The capacity of set $\mathcal{V}$
C	The number of node classes
$\mathbf{A}$	The graph adjacency matrix
$\mathbf{I}$	The identity matrix with an appropriate shape
$\mathbf{D}=\operatorname{diag}(\mathbf{A}\cdot\mathbf{1})$	The graph degree matrix
$\hat{\mathcal{G}}=\left(\hat{\mathbf{A}},\hat{\mathbf{X}}\right)$	The attacked (perturbed) graph
$\operatorname{Tr}\{\mathbf{M}\}$	The trace of matrix $\mathbf{M}$
$\sigma(\mathbf{M})$	The spectral norm of matrix $\mathbf{M}$
$\phi'(\mathbf{X})\in\mathbb{R}^{m\times n}$	The element-wise derivative of $\phi(\mathbf{X})$ w.r.t. $\mathbf{X}\in\mathbb{R}^{m\times n}$
$\mathbf{S}\in\{0,1\}^{\lvert\mathcal{V}_L\rvert\times N}$	The row selection matrix to pick out the training node rows
$\delta\mathbf{W}=\partial\mathcal{L}/\partial\mathbf{W}$	The gradients of the scalar loss $\mathcal{L}$ w.r.t. a matrix $\mathbf{W}$
$\lVert\mathbf{M}\rVert_2$	The L_2 operator norm of matrix $\mathbf{M}$
$\mathbf{L}=\mathbf{I}-\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$	The symmetric Laplacian matrix
$\mathbf{P}\in\mathbb{R}^{N\times N}$	The propagation matrix of GNN
$\mathbf{y}\in\{1,\cdots,C\}^N$	The ground-truth label vector of all nodes
$y_i\in\{1,\cdots,C\}$	The ground-truth label of node i
$\mathbf{H}^{(0)}=\mathbf{X}\in\mathbb{R}^{N\times d}$	The initial feature matrix
$\mathbf{H}^{(l)}$	The input feature matrix of the l-th GNN layer
$\mathbf{Z}\in\mathbb{R}^{N\times C}$	The normalized logits of N graph nodes over C classes
$\mathbf{h}_i$	The input column feature vector of node i
$\mathbf{Z}_m^{(t_1)}$	The MLP logits normalized by t_1-softmax function
$[\mathbf{M}]_i$	Pick out the i-th row of the matrix $\mathbf{M}$ as a row vector
$\boldsymbol{\mu}_0/\boldsymbol{\mu}_1$	The initial class centers of one two-class CSBM graph
$\mathbf{s}_h=\frac{\left(\mathbf{W}_1^\top+\frac{p-q}{p+q}\mathbf{W}_2^\top\right)(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)}{\lVert\left(\mathbf{W}_1^\top+\frac{p-q}{p+q}\mathbf{W}_2^\top\right)(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\rVert_2}$	The line connecting the two class centers of the SAGE outputs
$\mathbf{m}_h=\frac{\left(\mathbf{W}_1^\top+\mathbf{W}_2^\top\right)(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1)}{2}$	The midpoint between two class centers
$\mathcal{P}_h=\left\{\mathbf{h}\mid\mathbf{s}_h^\top\mathbf{h}-\mathbf{s}_h^\top\mathbf{m}_h\right\},$	The decision boundary for the GNN outputs on CSBM graphs
$\mathbf{A}\circ\mathbf{B}$	The Hadamard/element-wise product between two matrices
$f\circ g$	The composition of functions f and g
$\lVert f\rVert_{Lip}$	The Lipschitz constant of scalar-valued function f
$\nabla f(x)$	The gradient of the scalar function f w.r.t. x
$\lVert\mathbf{M}\rVert_F$	The Frobenius norm of matrix $\mathbf{M}$

APPENDIX B
PROOFS OF SOME PROPOSITIONS

A. Proof of Proposition 1

Proof: For any node i, we denote the neighbor set by $\mathcal{N}_i$ and the neighborhood size by N_i. (3) shows that $\mathbf{h}_i$ is a linear combination of affine mapped $\mathbf{x}_i$ and $\mathbf{x}_j, j \in \mathcal{N}_i$. Since all node features are sampled from Gaussians, $\mathbf{h}_i$ still follows a Gaussian. As there are $\frac{pN_i}{p+q}$ intra-class neighbors and $\frac{qN_i}{p+q}$ inter-class neighbors, we readily get that

$$\mathbb{E}\left\{\frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{x}_j\right\} = \frac{1}{N_i}\left\{\mathbb{E}\left[\sum_{j\in\mathcal{N}(i)\cap\{j,y_j=y_i\}}\mathbf{x}_j\right]+\mathbb{E}\left[\sum_{j\in\mathcal{N}(i)\cap\{j,y_j\neq y_i\}}\mathbf{x}_j\right]\right\} \tag{10a}$$

$$= \frac{1}{N_i}\left(\frac{pN_i}{p+q}\boldsymbol{\mu}_{y_i}+\frac{qN_i}{p+q}\boldsymbol{\mu}_{1-y_i}\right) = \frac{p\boldsymbol{\mu}_{y_i}+q\boldsymbol{\mu}_{1-y_i}}{p+q} \tag{10b}$$

$$\mathbb{D}\left\{\frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{x}_j\right\} = \frac{1}{N_i^2}\left\{\mathbb{D}\left[\sum_{j\in\mathcal{N}(i)\cap\{j,y_j=y_i\}}\mathbf{x}_j\right]+\mathbb{D}\left[\sum_{j\in\mathcal{N}(i)\cap\{j,y_j\neq y_i\}}\mathbf{x}_j\right]\right\} \tag{10c}$$

$$= \frac{1}{N_i^2}\left[\frac{(pN_i)^2}{(p+q)^2}\sigma^2\mathbf{I}+\frac{(qN_i)^2}{(p+q)^2}\sigma^2\mathbf{I}\right] = \frac{p^2+q^2}{(p+q)^2}\sigma^2\mathbf{I} \tag{10d}$$

Thus we have

$$\frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{x}_j \sim \mathcal{N}\left(\frac{p\boldsymbol{\mu}_{y_i}+q\boldsymbol{\mu}_{1-y_i}}{p+q}, \frac{p^2+q^2}{(p+q)^2}\sigma^2\mathbf{I}\right) \tag{11a}$$

$$\mathbf{W}_2^\top\frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{x}_j \sim \mathcal{N}\left(\mathbf{W}_2^\top\frac{p\boldsymbol{\mu}_{y_i}+q\boldsymbol{\mu}_{1-y_i}}{p+q}, \mathbf{W}_2^\top\frac{p^2+q^2}{(p+q)^2}\sigma^2\mathbf{W}_2\right) \tag{11b}$$

Owing to that $\mathbf{W}_1^\top\mathbf{x}_i \sim \mathcal{N}\left(\mathbf{W}_1^\top\boldsymbol{\mu}_{y_i}, \sigma^2\mathbf{W}_1^\top\mathbf{W}_1\right)$, $\mathbf{h}_i = \mathbf{W}_1^\top\mathbf{x}_i + \frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{W}_2^\top\mathbf{x}_j$ is the sum of two Gaussian distributions, amounting to the results in Proposition 1. ■

B. Proof of Proposition 2

Proof: Proposition 1 suggests that the processed feature distributions of class 0 and 1 are two Gaussians separately centered at $\mathbb{E}_0[\mathbf{h}_i]$ and $\mathbb{E}_1[\mathbf{h}_i]$ with the same distribution 'radius', i.e., variance. Hence the optimal decision boundary of GNN outputs is naturally the hyperplane defined in (6a), analogous to [51]. To get the optimal decision boundary, we directly compute the midpoint between $(\mathbb{E}_0[\mathbf{h}_i], \mathbb{E}_1[\mathbf{h}_i])$, and the line $\mathbb{E}_0[\mathbf{h}_i] - \mathbb{E}_1[\mathbf{h}_i]$ connecting these two centers. According to Proposition 1, we readily get the following.

$$\mathbf{m}_h = \frac{\mathbb{E}_0[\mathbf{h}_i]+\mathbb{E}_1[\mathbf{h}_i]}{2} = \frac{1}{2}\left[\mathbf{W}_1^\top(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1)+\mathbf{W}_2^\top\frac{p\boldsymbol{\mu}_0+q\boldsymbol{\mu}_1+p\boldsymbol{\mu}_1+q\boldsymbol{\mu}_0}{p+q}\right] \tag{12a}$$

$$= \frac{1}{2}\left[\mathbf{W}_1^\top(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1)+\mathbf{W}_2^\top(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1)\right] \tag{12b}$$

$$= \frac{\left(\mathbf{W}_1^\top+\mathbf{W}_2^\top\right)(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1)}{2} \tag{12c}$$

$$\mathbb{E}_0[\mathbf{h}_i] - \mathbb{E}_1[\mathbf{h}_i] = \mathbf{W}_1^\top\boldsymbol{\mu}_0+\mathbf{W}_2^\top\frac{p\boldsymbol{\mu}_0+q\boldsymbol{\mu}_1}{p+q}-\left(\mathbf{W}_1^\top\boldsymbol{\mu}_1+\mathbf{W}_2^\top\frac{p\boldsymbol{\mu}_1+q\boldsymbol{\mu}_0}{p+q}\right) \tag{13a}$$

$$= \mathbf{W}_1^\top(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)+\mathbf{W}_2^\top\frac{p(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)+q(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)}{p+q} \tag{13b}$$

$$= \mathbf{W}_1^\top(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)+\mathbf{W}_2^\top\frac{(p-q)(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)}{p+q} \tag{13c}$$

$$= \left(\mathbf{W}_1^\top+\frac{p-q}{p+q}\mathbf{W}_2^\top\right)(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1) \tag{13d}$$

$$\mathbf{s}_h = \frac{(\mathbb{E}_0[\mathbf{h}_i]-\mathbb{E}_1[\mathbf{h}_i])}{\|\mathbb{E}_0[\mathbf{h}_i]-\mathbb{E}_1[\mathbf{h}_i]\|_2} \tag{14}$$

■

C. Proof of Proposition 3

Lemma 1. *For a $C-$category classification task, when training a one-layer MLP $\mathbf{h}=\mathbf{W}\mathbf{x}$ with t-softmax and cross-entropy functions, the t-softmax logits and gradients w.r.t. $\mathbf{h}$ are*

$$z_i=\frac{\exp\left(h_i/t\right)}{\sum_{k=1}^{C}\exp\left(h_k/t\right)},\quad \mathcal{L}=\sum_{i=1}^{C}y_i\log\frac{1}{z_i},\quad \delta\mathbf{h}=\frac{\partial\mathcal{L}}{\partial\mathbf{h}}=\frac{\mathbf{z}-\mathbf{y}}{t},$$

where c is the true label of $\mathbf{x}$ and $\mathbf{y}$ is the corresponding ground-truth one-hot vector.

Proof: The derivative of $\mathcal{L}$ w.r.t. unnormalized logits $\mathbf{h}$ are composed by

$$\delta h_i=\frac{\partial\mathcal{L}}{\partial h_i}=-\sum_{k=1}^{C}y_k\frac{\partial\log z_k}{\partial h_i}=-\sum_{i=1}^{C}\frac{y_k}{z_k}\frac{\partial z_k}{\partial h_i} \tag{15a}$$

$$=-\frac{y_i}{z_i}\frac{\partial z_i}{\partial h_i}-\sum_{k\neq i}^{C}\frac{y_k}{z_k}\frac{\partial z_k}{\partial h_i}=-\frac{y_i}{z_i}\frac{1}{t}z_i(1-z_i)-\sum_{k\neq i}^{C}\frac{y_k}{z_k}\frac{1}{t}(-z_kz_i) \tag{15b}$$

$$=\frac{-y_i+y_iz_i}{t}+\frac{1}{t}z_i\sum_{k\neq i}^{C}y_k \tag{15c}$$

$$=\frac{-y_i+z_i\sum_{k=i}^{C}y_k}{t}=\frac{z_i-y_i}{t}. \tag{15d}$$

■

Lemma 2. *Denote by $\mathbf{P}$ the propagation matrix, by $\mathbf{W}$ the learnable weights, by $\mathbf{B}$ the learnable bias, by $\mathbf{H}$ the input features, and by $\delta\mathbf{H}'=\partial\mathcal{L}/\partial\mathbf{H}'$ the derivative matrix of scalar loss $\mathcal{L}$ w.r.t. the output features $\mathbf{H}'$ activated by an element-wise activation function $\phi(\cdot)$, then it follows that all related gradients of the layer*

$$\mathbf{X}=\mathbf{P}\mathbf{H}\mathbf{W}+\mathbf{B},\ \mathbf{H}'=\phi(\mathbf{X}) \tag{16}$$

are

$$\delta\mathbf{W}=\frac{\partial\mathcal{L}}{\partial\mathbf{W}}=(\mathbf{P}\mathbf{H})^{\top}\left(\delta\mathbf{H}'\circ\phi'(\mathbf{X})\right) \tag{17a}$$

$$\delta\mathbf{H}=\frac{\partial\mathcal{L}}{\partial\mathbf{H}}=\mathbf{P}^{\top}\left(\delta\mathbf{H}'\circ\phi'(\mathbf{X})\right)\mathbf{W}^{\top} \tag{17b}$$

$$\delta\mathbf{B}=\frac{\partial\mathcal{L}}{\partial\mathbf{B}}=\delta\mathbf{H}'\circ\phi'(\mathbf{X}), \tag{17c}$$

where $\phi'(\mathbf{X})$ denotes the element-wise differentiation and $\circ$ is the Hadamard product.

Proof: The differential of loss $\mathcal{L}$ is

$$d\mathcal{L}=\mathrm{Tr}\left\{\left(\frac{\partial\mathcal{L}}{\partial\mathbf{H}'}\right)^{\top}d\mathbf{H}'\right\}$$

$$=\mathrm{Tr}\left\{\left(\frac{\partial\mathcal{L}}{\partial\mathbf{H}'}\right)^{\top}\left[\frac{\partial\phi(\mathbf{X})}{\partial\mathbf{X}}\circ(\mathbf{P}\mathbf{H}d\mathbf{W}+\mathbf{P}d\mathbf{H}\mathbf{W}+d\mathbf{B})\right]\right\}. \tag{18a}$$

The Hadamard and Frobenius products commute in a trace operation, so we have the following when focusing on $\mathbf{W}$.

$$d\mathcal{L}=\mathrm{Tr}\left\{(\delta\mathbf{H}')^{\top}\left[\phi'(\mathbf{X})\circ(\mathbf{P}\mathbf{H}d\mathbf{W})\right]\right\}=\mathrm{Tr}\left\{\left(\delta\mathbf{H}'\circ\phi'(\mathbf{X})\right)^{\top}\mathbf{P}\mathbf{H}d\mathbf{W}\right\}$$

$$=\mathrm{Tr}\left\{\left[\mathbf{H}^{\top}\mathbf{P}^{\top}\left(\delta\mathbf{H}'\circ\phi'(\mathbf{X})\right)\right]^{\top}d\mathbf{W}\right\}$$

According to the relationships between matrix derivatives and total differential, i.e., $d\mathcal{L}=\mathrm{Tr}\left\{\frac{\partial\mathcal{L}}{\partial\mathbf{R}}^{\top}d\mathbf{R}\right\}$ [74], we readily get the gradient w.r.t. $\mathbf{W}$. Similarly, we can get the rest results by manipulating $\mathbf{H}$ and $\mathbf{B}$. ■

Lemma 3. *Denote by $\mathbf{W}$ the learnable weights, by $\mathbf{B}$ learnable bias, by $\mathbf{H}$ the input features, and by $\delta\mathbf{H}'=\partial\mathcal{L}/\partial\mathbf{H}'$ the derivative matrix of scalar loss $\mathcal{L}$ w.r.t. the output features $\mathbf{H}'$ activated by an element-wise activation function $\phi(\cdot)$, then we can get that all involved gradients of the layer*

$$\mathbf{X}=\mathbf{H}\mathbf{W}+\mathbf{B},\ \mathbf{H}'=\phi(\mathbf{X}) \tag{20}$$

are

$$\delta\mathbf{W} = \frac{\partial\mathcal{L}}{\partial\mathbf{W}} = \mathbf{H}^\top\left(\delta\mathbf{H}' \circ \phi'(\mathbf{X})\right) \tag{21a}$$

$$\delta\mathbf{H} = \frac{\partial\mathcal{L}}{\partial\mathbf{H}} = \left(\delta\mathbf{H}' \circ \phi'(\mathbf{X})\right)\mathbf{W}^\top \tag{21b}$$

$$\delta\mathbf{B} = \frac{\partial\mathcal{L}}{\partial\mathbf{B}} = \delta\mathbf{H}' \circ \phi'(\mathbf{X}), \tag{21c}$$

where $\phi'(\mathbf{X})$ denotes element-wise differentiation and $\circ$ is the Hadamard product.

Proof: The proof can be finished in a way analogous to Lemma 2 or just by substituting $\mathbf{P} = \mathbf{I}$ into the conclusion of Lemma 2. ■

Proposition 3. *For an L-layer GraphSAGE (stacked by (2)) with cross entropy loss $\mathcal{L}$, $\delta\mathbf{W}$ is the matrix derivative, i.e., $\delta\mathbf{W} = \partial\mathcal{L}/\partial\mathbf{W}$, and the gradients of weights and hidden features are*

$$\delta\mathbf{X}^{(L-1)} = \mathbf{S}^\top\delta\mathbf{X}^{(L-1)}_{\mathcal{V}_L} = \frac{\mathbf{S}^\top\mathbf{S}}{t|\mathcal{V}_L|}\left(\mathbf{Z}^{(t)} - \mathbf{Y}\right) \tag{22a}$$

$$\delta\mathbf{W}_2^{(L-1)} = \left(\mathbf{P}\mathbf{H}^{(L-1)}\right)^\top\delta\mathbf{X}^{(L-1)} \tag{22b}$$

$$\delta\mathbf{W}_1^{(L-1)} = \left(\mathbf{H}^{(L-1)}\right)^\top\delta\mathbf{X}^{(L-1)} \tag{22c}$$

$$\delta\mathbf{H}^{(L-1)} = \delta\mathbf{X}^{(L-1)}\left(\mathbf{W}_1^{(L-1)}\right)^\top + \mathbf{P}^\top\delta\mathbf{X}^{(L-1)}\left(\mathbf{W}_2^{(L-1)}\right)^\top \tag{22d}$$

$$\delta\mathbf{W}_2^{(l)} = \left(\mathbf{P}\mathbf{H}^{(l)}\right)^\top\left(\delta\mathbf{H}^{(l+1)} \circ \phi'(\mathbf{X}^{(l)})\right) \tag{22e}$$

$$\delta\mathbf{W}_1^{(l)} = \left(\mathbf{H}^{(l)}\right)^\top\left(\delta\mathbf{H}^{(l+1)} \circ \phi'(\mathbf{X}^{(l)})\right) \tag{22f}$$

$$\delta\mathbf{H}^{(l)} = \left(\delta\mathbf{H}^{(l+1)} \circ \phi'(\mathbf{X}^{(l)})\right)\left(\mathbf{W}_1^{(l)}\right)^\top + \mathbf{P}^\top\left(\delta\mathbf{H}^{(l+1)} \circ \phi'(\mathbf{X}^{(l)})\right)\left(\mathbf{W}_2^{(l)}\right)^\top \tag{22g}$$

where t is the softmax temperature, $\mathbf{Y} \in \mathbb{R}^{N\times C}$ is stacked by one-hot ground-truth row vectors, $\phi'(\mathbf{X}^{(l)})$ is the derivative of element-wise activation function $\phi(\cdot)$ w.r.t. $\mathbf{X}^{(l)} = \mathbf{H}^{(l)}\mathbf{W}_1^{(l)} + \mathbf{P}\mathbf{H}^{(l)}\mathbf{W}_2^{(l)}$, and $\mathbf{S} \in \{0,1\}^{|\mathcal{V}_L|\times N}$ is the row selection matrix to pick out the rows corresponding to the training nodes, e.g., $\mathbf{X}^{(L-1)}_{\mathcal{V}_L} = \mathbf{S}\mathbf{X}^{(L-1)} \in \mathbb{R}^{|\mathcal{V}_L|\times C}$.

Proof: In L-layer GraphSAGE, the data flow during training is

$$\begin{aligned}\mathbb{R}^{N\times d} \ni \mathbf{X} = \mathbf{H}^{(0)} &\xrightarrow{1st\ conv} \mathbf{X}^{(0)} \xrightarrow{1st\ \phi} \mathbf{H}^{(1)} \xrightarrow{2nd\ conv} \mathbf{X}^{(1)} \xrightarrow{2nd\ \phi} \mathbf{H}^{(2)} \to \cdots \\ &\to \mathbf{H}^{(L-1)} \xrightarrow{L-th\ conv} \mathbf{X}^{(L-1)} \xrightarrow{t-\text{softmax}} \mathbf{H}^{(L)} = \mathbf{Z} \in \mathbb{R}^{N\times C} \\ &\xrightarrow{pick\ out\ \mathcal{V}_L} \mathbf{Z}_{\mathcal{V}_L} \xrightarrow{with\ \mathbf{Y}_{\mathcal{V}_L}=\mathbf{S}\mathbf{Y}} \mathcal{L} \in \mathbb{R}\end{aligned} \tag{23}$$

We first derive the gradients of $\mathbf{X}^{(L-1)}$ and then backpropagate through the data flow. Softmax is row-wise, and the cross entropy loss is also computed per row/node. The averaged loss over nodes from the training set $\mathcal{V}_L$ is usually taken as the final objective. According to Lemma 1 we have

$$\delta\mathbf{X}^{(L-1)}_{\mathcal{V}_L} = \frac{1}{|\mathcal{V}_L|}\left(\mathbf{Z}^t_{\mathcal{V}_L} - \mathbf{Y}_{\mathcal{V}_L}\right) = \frac{\mathbf{S}}{|\mathcal{V}_L|t}\left(\mathbf{Z}^{(t)} - \mathbf{Y}\right). \tag{24}$$

Taking $\mathbf{X}^{(L-1)}_{\mathcal{V}_L} = \mathbf{S}\mathbf{X}^{(L-1)}$ as the layer in Lemma 3 leads to (22a). Each SAGE layer comprises two parts respectively reducible to Lemmas 2 and 3. Recursively applying these two lemmas per layer through the inverse data flow (23) gives the remaining derivatives. ■

D. Proof of Proposition 4

Proposition 4. *For a k-dimensional Gaussian random vector $\mathbf{x} \in \mathbb{R}^k$ that admits the mean $\mathbf{u} \in \mathbb{R}^k, \lambda = \sum_{i=1}^k \mu_i^2$ and covariance matrix $\sigma^2\mathbf{I} \in \mathbb{R}^{k\times k}$, the expectation and variance of the squared L_2 norm are respectively*

$$\mathbb{E}\left[\|\mathbf{x}\|_2^2\right] = k\sigma^2 + \lambda \tag{25}$$

$$\mathbb{D}\left[\|\mathbf{x}\|_2^2\right] = 4\sigma^2\lambda + 2k\sigma^4 = 2\sigma^2(2\lambda + k\sigma^2) \tag{26}$$

Proof: Let $x_i = \sigma y_i + u_i$ such that $y_i \sim \mathcal{N}(0,1)$, then we have

$$\|x\|_2^2 = \sum_{i=1}^k x_i^2 = \sum_{i=1}^k (\sigma y_i + u_i)^2 = \sum_{i=1}^k \sigma^2 y_i^2 + \sum_{i=1}^k u_i^2 + 2\sum_{i=1}^k \sigma u_i y_i. \tag{27}$$

Since $y_i^2 \sim \chi^2(k)$ and $\sum_{i=1}^k u_i^2 + 2\sum_{i=1}^k \sigma u_i y_i \sim \mathcal{N}(\lambda, 4\sigma^2\lambda)$, $\|\mathbf{x}\|_2^2$ is actually a generalized chi-square distribution whose mean and variance are

$$\mathbb{E}\left[\|\mathbf{x}\|_2^2\right] = \lambda + \sum_{j=1}^{k} \sigma^2(1+0) = k\sigma^2 + \lambda \tag{28a}$$

$$\mathbb{D}\left[\|\mathbf{x}\|_2^2\right] = 4\sigma^2\lambda + 2\sum_{j}^{k} \sigma^4(1+0) = 4\sigma^2\lambda + 2k\sigma^4 = 2\sigma^2(2\lambda + k\sigma^2) \tag{28b}$$

■

APPENDIX C
PROOF OF THEOREM 1

Proof: A one-layer SAGE can be formulated as $\mathbf{h}_i = (\mathbf{W}_1)^\top \mathbf{x}_i + \frac{1}{N_i}\sum_{j\in\mathcal{N}(i)} (\mathbf{W}_2)^\top \mathbf{x}_j$. With both empirical risk minimization and MLP-to-GNN distillation, the loss function concentrating on node i can be formulated as

$$\mathcal{L}_g = \frac{1}{|\mathcal{V}_L|}\ell_{CE}\left(y_i, [\mathbf{Z}_g]_i\right) + \frac{\alpha_1 t_1^2}{N}\ell_{KLD}\left(\left[\mathbf{Z}_m^{(t_1)}\right]_i, \left[\mathbf{Z}_g^{(t_1)}\right]_i\right). \tag{29}$$

Denoting the gradients from the two terms in (29) separately by $\nabla\mathcal{L}_{g,CE}$ and $\nabla\mathcal{L}_{g,KLD}$, one forward-backward optimization step on node i with both terms leads to the SAGE weights that output $\mathbf{h}_i^{kd}$ while that with only the first term $\mathbf{h}_i$.

Though the full-batch update is common for most GNNs [65], [75], [76], [33], [27], the sampling-based GNNs using ego-nets [34], [77], [13] or subgraphs [78], [79], [80], [81] are popular for large graphs [82]. Since the analysis of the finest ego-net granularity can extend to subgraph and whole-graph cases where the distillation effects of multiple nodes are aggregated, we concentrate on one node here for extendibility. For one sampled ego-net centering at node i, Proposition 3 can be accordingly adapted by appropriately adjusting $\mathbf{P}$, $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{Y}$.

Since minimizing the KLD is equivalent to minimizing the CE given the target distribution P

$$\ell_{KLD}(P,Q) = \sum_{x\in\mathcal{X}} p(x)\log\frac{1}{q(x)} - \sum_{x\in\mathcal{X}} p(x)\log\frac{1}{p(x)} = \ell_{CE}(P,Q) - H(P), \tag{30}$$

the gradient of $\ell_{KLD}(P,Q)$ turns out to be that of $\ell_{CE}(P,Q)$. Thus the gradients from MLP-to-GNN knowledge distillation, i.e., the second term in (29), are

$$\frac{\partial}{\partial\mathbf{W}_1}\left[\alpha_1 t_1^2\ell_{KLD}\left(\left[\mathbf{Z}_m^{(t_1)}\right]_i, \left[\mathbf{Z}_g^{(t_1)}\right]_i\right)\right] = \frac{\alpha_1 t_1^2}{N}\frac{1}{t_1}\frac{\partial\ell_{CE}\left(\left[\mathbf{Z}_m^{(t_1)}\right]_i, \left[\mathbf{Z}_g^{(t_1)}\right]_i\right)}{\partial\mathbf{W}_1} \tag{31a}$$

$$= \frac{\alpha_1 t_1}{N}\mathbf{x}_i\left(\left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i\right) \tag{31b}$$

$$\frac{\partial}{\partial\mathbf{W}_2}\left[\alpha_1 t_1^2\ell_{KLD}\left(\left[\mathbf{Z}_m^{(t_1)}\right]_i, \left[\mathbf{Z}_g^{(t_1)}\right]_i\right)\right] = \frac{\alpha_1 t_1}{N}\left[\mathbf{P}\mathbf{X}\right]_i^\top\left(\left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i\right) \tag{31c}$$

$$= \frac{\alpha_1 t_1}{N}\left(\frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{x}_j\right)\left(\left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i\right) \tag{31d}$$

$$= \frac{\alpha_1 t_1}{N}\bar{\mathbf{x}}_i\left(\left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i\right), \tag{31e}$$

according to the Proposition 3 variant adapted to the ego-net granularity. Let $\mathbf{W}_1$ and $\mathbf{W}_2$ be the weights updated by normal supervised training. Denoting by $\mathbf{W}_1^{kd}$ and $\mathbf{W}_2^{kd}$ the weights updated with both normal and distillation gradients, then we have

$$\mathbf{W}_1^{kd} = \mathbf{W}_1 - \eta\mathbf{x}_i\left(\left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i\right) = \mathbf{W}_1 - \eta\mathbf{x}_i\mathbf{r}_i \tag{32a}$$

$$\mathbf{W}_2^{kd} = \mathbf{W}_2 - \eta\bar{\mathbf{x}}_i\left(\left[\left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i\right]\right) = \mathbf{W}_2 - \eta\bar{\mathbf{x}}_i\mathbf{r}_i, \tag{32b}$$

where $\mathbf{r}_i$ denotes $\left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i$ and $\eta = \eta'\alpha_1 t_1/N$ integrates the learning rate η' and other hyperparameters.

As the case for the nodes from class 0 is symmetric to that from class 1, we only prove for the nodes from class 0. For the node i with label $y_i = 0$, $\mathbf{W}_1$ and $\mathbf{W}_2$ lead to a Gaussian distribution whose expectation is, according to Proposition 1,

$$\mathbb{E}\left[\mathbf{h}_i\right] = \mathbb{E}\left[\mathbf{W}_1^\top\mathbf{x}_i + \mathbf{W}_2^\top\frac{1}{N_i}\sum_{j\in\mathcal{N}(i)}\mathbf{x}_j\right] = \mathbb{E}\left[\mathbf{W}_1^\top\mathbf{x}_i + \mathbf{W}_2^\top\bar{\mathbf{x}}_i\right] \tag{33a}$$

$$= \mathbf{W}_1^\top\boldsymbol{\mu}_0 + \mathbf{W}_2^\top\frac{p\boldsymbol{\mu}_0 + q\boldsymbol{\mu}_1}{p+q}. \tag{33b}$$

By contrast, $\mathbf{W}_1^{kd}$ and $\mathbf{W}_2^{kd}$ result in

$$\mathbf{h}_i^{kd} = \left(\mathbf{W}_1^{kd}\right)^\top \mathbf{x}_i + \left(\mathbf{W}_2^{kd}\right)^\top \frac{1}{N_i} \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \tag{34a}$$

$$= \left(\mathbf{W}_1^\top - \eta \mathbf{r}_i^\top \mathbf{x}_i^\top\right) \mathbf{x}_i + \left(\mathbf{W}_2^\top - \eta \mathbf{r}_i^\top \bar{\mathbf{x}}_i^\top\right) \bar{\mathbf{x}}_i, \tag{34b}$$

whose expectation is

$$\mathbb{E}\left[\mathbf{h}_i^{kd}\right] = \mathbb{E}\left[\mathbf{W}_1^\top \mathbf{x}_i + \mathbf{W}_2^\top \bar{\mathbf{x}}_i\right] - \mathbb{E}\left[\eta \mathbf{r}_i^\top \mathbf{x}_i^\top \mathbf{x}_i + \eta \mathbf{r}_i^\top \bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_i\right] \tag{35a}$$

$$= \mathbb{E}\left[\mathbf{h}_i\right] - \eta \mathbf{r}_i^\top \left(\mathbb{E}\left[\mathbf{x}_i^\top \mathbf{x}_i\right] + \mathbb{E}\left[\bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_i\right]\right). \tag{35b}$$

According to Proposition 4 and (11a), we have

$$\mathbb{E}\left[\mathbf{x}_i^\top \mathbf{x}_i\right] = \mathbb{E}\left[\|\mathbf{x}_i\|_2^2\right] = \boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 + d\sigma^2 \tag{36}$$

$$\mathbb{E}\left[\bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_i\right] = \mathbb{E}\left[\|\bar{\mathbf{x}}_i\|_2^2\right] = \frac{(p\boldsymbol{\mu}_0^\top + q\boldsymbol{\mu}_1^\top)(p\boldsymbol{\mu}_0 + q\boldsymbol{\mu}_1) + d\left(p^2+q^2\right)\sigma^2}{(p+q)^2}. \tag{37}$$

Substituting them into (35b) gives the output expectation of GNN with MLP-to-GNN distillation

$$\mathbb{E}\left[\mathbf{h}_i^{kd}\right] = \mathbb{E}\left[\mathbf{h}_i\right] - \eta \mathbf{r}_i^\top \left[\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 + d\sigma^2 + \frac{(p\boldsymbol{\mu}_0^\top + q\boldsymbol{\mu}_1^\top)(p\boldsymbol{\mu}_0 + q\boldsymbol{\mu}_1) + d\left(p^2+q^2\right)\sigma^2}{(p+q)^2}\right]. \tag{38}$$

Substituting $\mathbf{W}_1^{kd}$ and $\mathbf{W}_2^{kd}$ into Proposition 2, we get the ideal decision boundary on $\mathbf{h}^{kd}$

$$\mathcal{P}_h^{kd} = \left\{\mathbf{h}^{kd} \mid \left(\mathbf{s}_h^{kd}\right)^\top \mathbf{h}^{kd} - \left(\mathbf{s}_h^{kd}\right)^\top \mathbf{m}_h^{kd}\right\}, \tag{39a}$$

$$\mathbf{s}_h^{kd} = \frac{\left(\mathbf{W}_1^\top - \eta \mathbf{r}_i^\top \mathbf{x}_i^\top + \frac{p-q}{p+q}\mathbf{W}_2^\top - \frac{p-q}{p+q}\eta \mathbf{r}_i^\top \bar{\mathbf{x}}_i^\top\right)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)}{\|numerator\|_2} \tag{39b}$$

$$= \frac{\mathbf{s}_h - \eta\left(\mathbf{r}_i^\top \mathbf{x}_i^\top - \frac{p-q}{p+q}\mathbf{r}_i^\top \bar{\mathbf{x}}_i^\top\right)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)}{\|numerator\|_2} \tag{39c}$$

$$\mathbf{m}_h^{kd} = \frac{\left[\mathbf{W}_1^\top + \mathbf{W}_2^\top - \eta\left(\mathbf{r}_i^\top \mathbf{x}_i^\top + \mathbf{r}_i^\top \bar{\mathbf{x}}_i^\top\right)\right](\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1)}{2}, \tag{39d}$$

where $numerator$ means the corresponding numerator formula.

Proof of the first conclusion in Theorem 1.

In analogy with [51], for any node i we have the follows

$$\mathbb{P}(\mathbf{h}_i \text{ is mis-classified }) = \mathbb{P}\left(\mathbf{s}_h^\top \mathbf{h}_i - \mathbf{s}_h^\top \mathbf{m}_h \le 0\right) \tag{40}$$

$$\mathbb{P}(\mathbf{h}_i^{kd} \text{ is mis-classified }) = \mathbb{P}\left(\left(\mathbf{s}_h^{kd}\right)^\top \mathbf{h}_i^{kd} - \left(\mathbf{s}_h^{kd}\right)^\top \mathbf{m}_h^{kd} \le 0\right), \tag{41}$$

which turns out to be, in expectation,

$$\mathbb{P}(\mathbb{E}\left[\mathbf{h}_i\right] \text{ is mis-classified }) = \mathbb{P}\left(R = \mathbf{s}_h^\top \mathbb{E}\left[\mathbf{h}_i\right] - \mathbf{s}_h^\top \mathbf{m}_h \le 0\right) \tag{42}$$

$$\mathbb{P}(\mathbb{E}\left[\mathbf{h}_i^{kd}\right] \text{ is mis-classified }) = \mathbb{P}\left(R^{kd} = \left(\mathbf{s}_h^{kd}\right)^\top \mathbb{E}\left[\mathbf{h}_i^{kd}\right] - \left(\mathbf{s}_h^{kd}\right)^\top \mathbf{m}_h^{kd} \le 0\right). \tag{43}$$

If $R^{kd} > R$, R^{kd} is less likely to be ≤ 0 than R and thus less likely to be misclassified. Now let's prove $R^{kd} > R$ holds no matter how heterophilic the graph is (or is changed by attacks to be).

Without loss of generality, we can always establish an appropriate coordinate system or apply suitable appropriate affine transform beforehand such that the two class centers of raw node features admit $\boldsymbol{\mu}_0 = -\boldsymbol{\mu}_1$ and the midpoints $\mathbf{m}_h$ and $\mathbf{m}_h^{kd}$ become zeros. Then we have

$$R = \mathbf{s}_h^\top \mathbb{E}\left[\mathbf{h}_i\right] \tag{44}$$

$$R^{kd} = \left(\mathbf{s}_h^{kd}\right)^\top \mathbb{E}\left[\mathbf{h}_i^{kd}\right] \approx \mathbf{s}_h^\top \mathbb{E}\left[\mathbf{h}_i^{kd}\right], \tag{45}$$

where $\mathbf{s}_h^{kd}$ from (39b) is approximately equal to $\mathbf{s}_h$ due to the small $\eta = \eta' \alpha_1 t_1 / N$ that results from a small learning rate η' (usually at an order of magnitude less than 10^{-2}) and a large N (usually at an order of magnitude greater than 10^3). Then we rewrite R^{kd} with (38)

$$R^{kd} = \mathbf{s}_h^\top \mathbb{E}\left[\mathbf{h}_i^{kd}\right] \tag{46a}$$

$$= \mathbf{s}_h^\top \left\{\mathbb{E}\left[\mathbf{h}_i\right] - \eta \mathbf{r}_i^\top \left[\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 + d\sigma^2 + \frac{(p\boldsymbol{\mu}_0^\top + q\boldsymbol{\mu}_1^\top)(p\boldsymbol{\mu}_0 + q\boldsymbol{\mu}_1) + d(p^2+q^2)\sigma^2}{(p+q)^2}\right]\right\} \tag{46b}$$

$$= \mathbf{s}_h^\top \mathbb{E}\left[\mathbf{h}_i\right] - \eta \mathbf{s}_h^\top \mathbf{r}_i^\top B = R - \eta \mathbf{s}_h^\top \mathbf{r}_i^\top B, \tag{46c}$$

where $\mathbf{s}_h^\top \mathbf{r}_i^\top$ determines the relative magnitude between R^{kd} and R, and B controls the magnitude gap between them. Recap that $\mathbf{r}_i = \left[\mathbf{Z}_g^{(t_1)}\right]_i - \left[\mathbf{Z}_m^{(t_1)}\right]_i$ where these two terms are the predicted distributions for node i from GNN and MLP, respectively.

Considering that $\boldsymbol{\mu}_0 = -\boldsymbol{\mu}_1$ on the CSBM graph, the formula below is certainly positive.

$$B(p,q) = \boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 + d\sigma^2 + \frac{(p-q)^2 \boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 + d\left(p^2+q^2\right)\sigma^2}{(p+q)^2} \tag{47a}$$

$$= \frac{2\left(p^2+q^2\right)\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 + 2\left(p^2+q^2+pq\right)d\sigma^2}{(p+q)^2} > 0 \tag{47b}$$

On the aCSBM graph, (47b) reduces to a quadratic form of p since $p = 1-q$

$$B(p) = 2\left(d\sigma^2 + 2\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0\right)p^2 - 2\left(d\sigma^2 + 2\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0\right)p + 2\left(d\sigma^2 + \boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0\right), \tag{48}$$

where the derivative is $dB/dp = 2(2p-1)(\sigma^2 + 2\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0)$ and the (global) minimum is hence $p = q = 0.5$. Hence, the minimum value of $B(p)$ is $(3d\sigma^2 + 2\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0)/2 > 0$. Therefore, for both CSBM and aCSBM graphs, $R^{kd} > R$ requires $\mathbf{s}_h^\top \mathbf{r}_i^\top < 0$ in (46c).

Let $\left[\mathbf{Z}_g^{(t_1)}\right]_i = [\phi_1, 1-\phi_1]$ and $\left[\mathbf{Z}_{m,}^{(t_1)}\right]_i = [\phi_2, 1-\phi_2]$. Then the delta between them turns out to be $\mathbf{r}_i = [\phi_1 - \phi_2, \phi_2 - \phi_1], \phi_1, \phi_2 \in [0,1]$. We further denote that

$$\mathbf{s}_h = \begin{bmatrix} x_0' \\ y_0' \end{bmatrix} = \frac{\left(\mathbf{W}_1^\top + \frac{p-q}{p+q}\mathbf{W}_2^\top\right)\boldsymbol{\mu}_0}{\|numerator\|_2} = \frac{\mathbf{W}\boldsymbol{\mu}_0}{\|\mathbf{W}\boldsymbol{\mu}_0\|_2}$$

Then we have

$$\mathbf{s}_h^\top \mathbf{r}_i^\top = (x_0', y_0') \begin{bmatrix} \phi_1 - \phi_2 \\ \phi_2 - \phi_1 \end{bmatrix} = (\phi_1 - \phi_2)(x_0' - y_0'). \tag{49}$$

As stated in the theorem condition, i.e., MGMD Condition, the MLP has a higher prediction probability than the GNN on the ground-truth class, meaning that $\phi_1 < \phi_2$. Therefore, $\mathbf{s}_h^\top \mathbf{r}_i^\top < 0$ (i.e., $R^{kd} > R$) requires $x_0' < y_0'$, which can be satisfied by appropriately establishing the coordinate system (and hence appropriately assigning coordinates to $\boldsymbol{\mu}_0$) for arbitrary $2 \times d$ weight matrix $\mathbf{W}$.

Proof of the second conclusion in Theorem 1

According to (46a), the distillation effect has an amplitude factor B, which is determined by the class centers and heterophily. So we can build the relationship between heterophily and the MLP-to-GNN distillation effect.

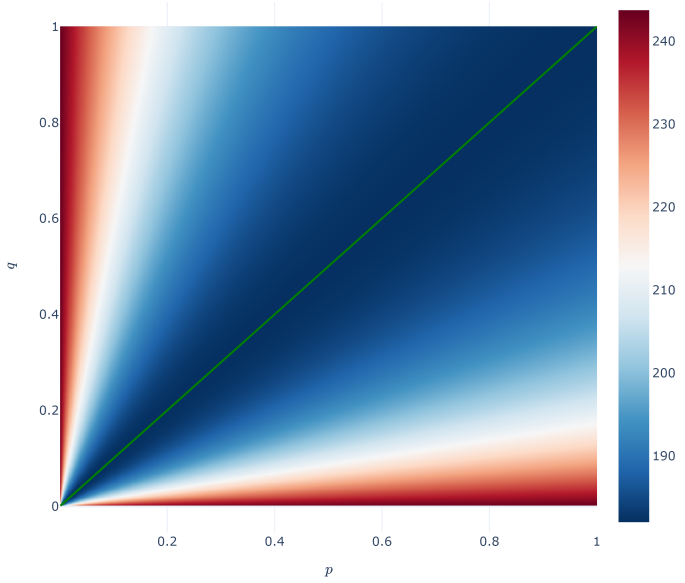


Fig. 7. The value of B changes with p and q, and gets minimized when $p = q$, which is highlighted as the green line in the graph. In this picture, we set $d = 120$, $\sigma^2 = 1$, $\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0 = 2$.

To determine the influence of graph heterophily on B, we examine the partial derivatives of (47b) w.r.t. p and q, which are

$$\frac{\partial B}{\partial p} = \frac{2q(p-q)(d\sigma^2 + 2\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0)}{(p+q)^3} \tag{50}$$

$$\frac{\partial B}{\partial q} = \frac{2p(q-p)(d\sigma^2 + 2\boldsymbol{\mu}_0^\top \boldsymbol{\mu}_0)}{(p+q)^3}. \tag{51}$$

Hence the stationary points of $B(p,q)$ are on the line $p = q$. Despite the negative definite Hessian indicating that they are saddle points, for any fixed $q \in (0,1]$, $B(p)$ has a local minimum $p = q$. Therefore, these saddle points are actually minima, meaning that the weakest MLP-to-GNN distillation effect emerges at the heterophilic demarcation of HR $= p/(p+q) = 0.5$. For an intuitive understanding, we depict the value of B with respect to various p and q in Figure 7, which also shows that the strongest effect is approached at the extreme homophily ratios, i.e., the left-top and right-bottom corners in the figure. Regarding the aCSBM graph, a similar analysis of (48) can lead to the same conclusion. ■

APPENDIX D
PROOF OF THEOREM 2

A. Auxiliaries for Theorem 2 Proof

Definition 1. *(Lipschitz constant) A function $f : \mathcal{X} \to \mathcal{Y}$ is K-Lipschitz (continuous) w.r.t. a norm $\|\cdot\|$ if there is a constant K such that*

$$\forall x_1, x_2 \in \mathcal{X}, \|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|. \tag{52}$$

The smallest K admits the inequality is one of the Lipschitz constants of f and denoted as $\|f\|_{Lip}$.

Theorem 3 (Rademacher [83], Theorem 3.1.6; [84], Theorem 1)**.** *If $f : \mathbb{R}^n \to \mathbb{R}^m$ is a locally Lipschitz continuous function[2], then f is differentiable almost everywhere. Moreover, if f is Lipschitz continuous, then*

$$\|f\|_{Lip} = \sup_{x \in \mathbb{R}^n} \|\nabla f(x)\|_2, \tag{53}$$

where $\|\mathbf{M}\|_2 = \sup_{\|x\|\leq 1} \|\mathbf{M}x\|_2$ is the operator norm of matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$.

Theorem 4 (Banach fixed-point theorem)**.** *Let $(\mathcal{X}, \mathcal{D})$ be a non-empty complete metric space with a contraction mapping $f : \mathcal{X} \to \mathcal{X}$ such that*

$$\mathcal{D}\left(f(x_1), f(x_2)\right) \leq q\mathcal{D}\left(x_1, x_2\right), \exists q \in [0,1), \tag{54}$$

then there is a unique fixed-point $f(x^*) = x^*$ that can be found by generating a sequence $\{x_n \mid x_{n+1} = f(x_n)\}_{n\in\mathbb{N}}$ with an initial point $x^{(0)} \in \mathcal{X}$. $\mathcal{D}$ is usually a vector or matrix norm.

Proposition 5. *Given two functions $f : \mathcal{X} \to \mathcal{U}$ and $g : \mathcal{U} \to \mathcal{Y}$ whose Lipschitz constants are respectively $\|g\|_{Lip}$ and $\|f\|_{Lip}$, the Lipschitz constant of the composite function $g \circ f : \mathcal{X} \to \mathcal{Y}$ satisfies the inequality*

$$\|g \circ f\|_{Lip} \leq \|g\|_{Lip}\|f\|_{Lip}. \tag{55}$$

Proof: Let $u = f(x)$ and $y = g(u)$. The gradient of $g \circ f$ is

$$\nabla(g \circ f)(x) = \nabla g(f(x))\nabla f(x). \tag{56}$$

Then we have

$$\sup_{\|x\|\leq 1} \|[\nabla g(f(x))][\nabla f(x)]\|_2 \leq \sup_{\|u\|\leq 1} \|[\nabla g(u)]\|_2 \sup_{\|x\|\leq 1} \|[\nabla f(x)]\|_2, \tag{57}$$

which amounts to the desired according to Theorem 3. ■

Lemma 4 ([85], Proposition 4)**.** *The t-softmax function is Lipschitz with respect to L_2 vector norm and the Lipschitz constant is $1/t$. That is, for all $x_1, x_2 \in \mathbb{R}^n$*

$$\|\operatorname{softmax}_t(x_1) - \operatorname{softmax}_t(x_2)\|_2 \leq \frac{1}{t}\|x_1 - x_2\|_2. \tag{58}$$

Proposition 6. *The Lipschitz constant w.r.t. the Frobenius norm of the linear mapping operator $\mathbf{Y}_{m\times n} = \mathbf{A}_{m\times k}\mathbf{X}_{k\times n}$ is the spectral norm of $\mathbf{A}$, and that of the row-wise t-softmax function $\mathbf{Y}_{m\times n} = \operatorname{softmax}_t(\mathbf{X}_{m\times n})$ is $1/t$.*

Proof: 1) According to Theorem 3, the Lipschitz constant of matrix-vector multiplication $\mathbf{y}_{m\times 1} = \mathbf{A}_{m\times k}\mathbf{x}_{k\times 1}$ w.r.t. L_2 vector norm can be directly obtained, which is the spectral norm $\sigma(\mathbf{A})$, as shown in [84].

$$\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \leq \sigma(\mathbf{A})\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \tag{59}$$

[2]The functions whose restriction to some neighborhood around any point is Lipschitz are *locally* Lipschitz.

Given two matrices $\mathbf{X}_1 = [\mathbf{x}_{11}, \mathbf{x}_{12}, ..., \mathbf{x}_{1k}]$ and $\mathbf{X}_2 = [\mathbf{x}_{21}, \mathbf{x}_{22}, ..., \mathbf{x}_{2k}]$ composed by column vectors, we have

$$\mathbf{Y}_1 = [\mathbf{y}_{11}, \mathbf{y}_{12}, ..., \mathbf{y}_{1k}] = \mathbf{A}\mathbf{X}_1 = [\mathbf{A}\mathbf{x}_{11}, \mathbf{A}\mathbf{x}_{12}, ..., \mathbf{A}\mathbf{x}_{1k}] \tag{60}$$

$$\mathbf{Y}_2 = [\mathbf{y}_{21}, \mathbf{y}_{22}, ..., \mathbf{y}_{2k}] = \mathbf{A}\mathbf{X}_2 = [\mathbf{A}\mathbf{x}_{21}, \mathbf{A}\mathbf{x}_{22}, ..., \mathbf{A}\mathbf{x}_{2k}]\,. \tag{61}$$

The squared L_2 vector norms of all column vectors amount to the squared Frobenius norm

$$\|\mathbf{X}_1 - \mathbf{X}_2\|_F^2 = \sum_{j=1}^{k} \|\mathbf{x}_{1j} - \mathbf{x}_{2j}\|_2^2 \tag{62}$$

$$\|\mathbf{Y}_1 - \mathbf{Y}_2\|_F^2 = \sum_{j=1}^{k} \|\mathbf{y}_{1j} - \mathbf{y}_{2j}\|_2^2 \tag{63}$$

Thus according to (59), we have

$$\|\mathbf{Y}_1 - \mathbf{Y}_2\|_F^2 \leq \sigma^2(\mathbf{A})\|\mathbf{X}_1 - \mathbf{X}_2\|_F^2, \tag{64}$$

meaning that the Lipschitz constant of linear mapping w.r.t. the Frobenius norm is $\sigma(\mathbf{A})$.

2) By distributing the Frobenius norm onto rows and using Lemma 4 on each row, we can get $1/t$ in a way like the proof of the first conclusion. ■

B. Main Proof of Theorem 2

Proof: For the symbol simplicity, we replace $\mathcal{L}_m$with L_m and $\mathbf{W}$ with w here. The local or global optimal MLP weight w^* should satisfy $\nabla_w L_m(w^*) = 0$, meaning that $w^* = w^* - \eta\nabla_w L_m(w^*)$. We can thus construct a function $G(w) = w - \eta\nabla_W L_m(w)$ where η is the step size of gradient decent and w^* is the fixed-point of $G(w)$. We now derive the existence condition of w^* based on Banach fixed-point theorem, i.e., Theorem 3.

Let the space metric $\mathcal{D}$ be Frobenius matrix norm. If follows that for any two weights $w_1, w_2 \in \mathcal{W}$

$$\begin{aligned}\|G(w_1) - G(w_2)\|_F^2 =& \|w_1 - w_2 - \eta\left(\nabla L_m(w_1) - \nabla L_m(w_2)\right)\|_F^2 &\text{(65a)}\\ =& \|w_1 - w_2\|_F^2 + \eta^2\|\nabla L_m(w_1) - \nabla L_m(w_2)\|_F^2 \\ & - 2\,\mathrm{Tr}\left\{\eta(w_1 - w_2)^\top\left[\nabla L_m(w_1) - \nabla L_m(w_2)\right]\right\}. &\text{(65b)}\end{aligned}$$

To apply Banach fixed-point theorem, we need construct the inequality relationship between $\|G(w_1) - G(w_2)\|_F^2$ and $\|w_1 - w_2\|_F^2$, so the second and third terms in (65b) should be tackled. Since the assumption in Theorem 2 copes with the third term, we move on to the second term now.

The loss function L_m of MGMD-MLP is (4c), the gradient of which w.r.t. MLP weights can be obtained by recursively applying Lemmas 1 and 3. For simplicity, we denote by $f_t(\cdot)$ the t-softmax function and only consider the weight of the last MLP layer. In spite of this, our proof can be extended to an arbitrary MLP layer since the main tool, i.e., Proposition 5, can be extended to an arbitrary layer[3]. However, such a generalization would lead to highly cumbersome formulas without providing additional insights. To proceed, we expand the second term to

$$\begin{aligned}&\nabla L_m(w_1) - \nabla L_m(w_2)\\ =&\frac{\mathbf{H}^\top}{N_{tr}}\mathbf{S}^\top\mathbf{S}\left(f(\mathbf{H}w_1) - \mathbf{Y}\right) + \frac{\alpha t_2}{N}\mathbf{H}^\top\left(f_{t_2}(\mathbf{H}w_1) - \mathbf{Z}_g^{t_2}\right)\\ &\qquad - \frac{\mathbf{H}^\top}{N_{tr}}\mathbf{S}^\top\mathbf{S}\left(f(\mathbf{H}w_2) - \mathbf{Y}\right) - \frac{\alpha t_2}{N}\mathbf{H}^\top\left(f_{t_2}(\mathbf{H}w_2) - \mathbf{Z}_g^{t_2}\right) &\text{(66a)}\\ =&\frac{\mathbf{H}^\top}{N_{tr}}\mathbf{S}^\top\mathbf{S}\left[f(\mathbf{H}w_1) - f(\mathbf{H}w_2)\right] + \frac{\alpha t_2}{N}\mathbf{H}^\top\left[f_{t_2}(\mathbf{H}w_1) - f_{t_2}(\mathbf{H}w_2)\right], &\text{(66b)}\end{aligned}$$

where $\mathbf{H}$ is the input feature matrix of last MLP layer and $N_{tr} = |\mathcal{V}_L|$ is the size of training set.

We construct two functions

$$g_1(w) = \frac{\mathbf{H}^\top}{N_{tr}}\mathbf{S}^\top\mathbf{S}f(\mathbf{H}w) \tag{67}$$

$$g_2(w) = \frac{\alpha t_2}{N}\mathbf{H}^\top f_{t_2}(\mathbf{H}w), \tag{68}$$

and then (66b) turns out to be

$$\nabla L_m(w_1) - \nabla L_m(w_2) = g_1(w_1) - g_1(w_2) + g_2(w_1) - g_2(w_2). \tag{69}$$

[3]The extended proof also employs that the common activation functions (e.g., ReLU, LeakyReLU, Sigmoid, Tanh, and Sigmoid) are both 1-Lipschitz.

It follows that

$$\begin{aligned}
&\|\nabla L_m(w_1) - \nabla L_m(w_2)\|_F^2 && \text{(70a)}\\
=&\|g_1(w_1) - g_1(w_2) + g_2(w_1) - g_2(w_2)\|_F^2 && \text{(70b)}\\
=&\|g_1(w_1) - g_1(w_2)\|_F^2 + \|g_2(w_1) - g_2(w_2)\|_F^2 \\
&\qquad + 2\|g_1(w_1) - g_1(w_2)\|_F \|g_2(w_1) - g_2(w_2)\|_F && \text{(70c)}\\
\leq&\|g_1(w)\|_{Lip}^2 \|w_1 - w_2\|_F^2 + \|g_2(w)\|_{Lip}^2 \|w_1 - w_2\|_F^2 \\
&\qquad + 2\|g_1(w)\|_{Lip} \|g_2(w)\|_{Lip} \|w_1 - w_2\|_F^2 && \text{(70d)}
\end{aligned}$$

To find the inequality relationship between $\|\nabla L_m(w_1) - \nabla L_m(w_2)\|_F^2$ and $\|w_1 - w_2\|_F^2$, we need to check out the Lipschitz constants of $g_1(w)$ and $g_2(w)$. According to Propositions 6 and 5, we have

$$\|g_1(w)\|_{Lip} = \frac{1}{N_{tr}} \sigma(\mathbf{H}^\top \mathbf{S}^\top \mathbf{S}) \sigma(\mathbf{H}) \tag{71}$$

$$\|g_2(w)\|_{Lip} = \frac{\alpha t_2}{N} \sigma^2(\mathbf{H}). \tag{72}$$

Then the upper bound (i.e., (73)) becomes

$$\|\nabla L_m(w_1) - \nabla L_m(w_2)\|_F^2 \leq U_1 = \left[\frac{1}{N_{tr}} \sigma(\mathbf{H}^\top \mathbf{S}^\top \mathbf{S}) \sigma(\mathbf{H}) + \frac{\alpha t_2}{N} \sigma^2(\mathbf{H}) \right]^2 \|w_1 - w_2\|_F^2. \tag{73}$$

Substituting (73) and the theorem assumption (7) into (65b) leads to

$$\begin{aligned}
&\|G(w_1) - G(w_2)\|_F^2 && \text{(74a)}\\
\leq&\|w_1 - w_2\|_F^2 + \eta^2 \left[\frac{1}{N_{tr}} \sigma(\mathbf{H}^\top \mathbf{S}^\top \mathbf{S}) \sigma(\mathbf{H}) + \frac{\alpha t_2}{N} \sigma^2(\mathbf{H}) \right]^2 \|w_1 - w_2\|_F^2 && \text{(74b)}\\
=&\left\{ 1 + \eta^2 \left[\frac{1}{N_{tr}} \sigma(\mathbf{H}^\top \mathbf{S}^\top \mathbf{S}) \sigma(\mathbf{H}) + \frac{\alpha t_2}{N} \sigma^2(\mathbf{H}) \right]^2 - 2\eta u \right\} \|w_1 - w_2\|_F^2 && \text{(74c)}\\
=&\left(1 + \eta^2 \beta^2 - 2\eta u\right) \|w_1 - w_2\|_F^2. && \text{(74d)}
\end{aligned}$$

■

APPENDIX E
MORE EXPERIMENTAL DETAILS

A. Experiment Environment

Part of the experiments are conducted on an Ubuntu20.04 server equipped with one Intel(R) Xeon(R) Gold 6240C CPU and four NVIDIA GeForce RTX 4090 GPUs. And the rest experiments are run on a Windows11 workstation equipped with one Intel(R) Core(TM) i7-12700 and one NVIDIA GeForce RTX 4090 GPU.

For RGCN, Jaccard, SVD, and ProGNN, we adopt the implementations from DeepRobust [72]. The implementation of Guard follows their official implementation[4]. The implementation of STABLE adheres to the official implementation[5]. The implementation of EvenNet is based on the official implementation[6]. The datasets and models used in adaptive attacks are adapted from the repo[7] of [49]. And we implement MLP, SGC, SAGE, GLNN and MGMD based on Lightning 2.0.3 [86] and PyG 2.3.1 [87], which is built on Pytorch 2.0.1 [88] and CUDA Toolkit 11.8.

Besides, to facilitate future research on GNN-MLPs, we have designed some LightningModules to eliminate boilerplate code so that researchers can focus on designing their distillation modules. These LightningModules support: **1)** offline, online, and hybrid distillation; **2)** multiple teachers and/or multiple students; **3)** plug-in (customized) feature and/or logit distillation strategy modules; **4)** mini-batch sampling or full-batch training and distillation; **5)** unified and friendly data interfaces for both clean and perturbed graph data; **6)** easy switch between transductive, inductive and semi-inductive evaluation settings; **7)** easy switch between poisoning and evasion attack evaluation settings; **8)** flexible experiment monitoring and management.

B. Hyperparameters

In all experiments, for each baseline model and every dataset, we incorporate the provided hyperparameters if supplied by the respective authors. For models lacking hyperparameter information, we perform grid search within the range specified in Table IX to select suitable hyperparameters. For MGMD, we choose the temperatures of GNN and MLP from [0.05, 10], and the distillation term weights from [0.1, 10], in terms of the validation accuracy.

[4]https://github.com/mims-harvard/GNNGuard
[5]https://github.com/likuanppd/STABLE
[6]https://github.com/leirunlin/evennet
[7]https://github.com/LoadingByte/are-gnn-defenses-robust

TABLE IX
THE SEARCH SPACES OF PART OF BASELINE HYPERPARAMETERS

Hyperparameter	Search space	Hyperparameter	Search space
learning rate	{0.001, 0.005, 0.01, 0.1}	STABLE-α	{-0.5, -0.1, 0.1, 0.6, 2}
weight decay	{0, 0.0001, 0.0005, 0.001}	STABLE-β	{1, 2}
dropout rate	{0., 0.3, 0.5, 0.7}	STABLE-k	{1, 5, 7}
max epochs	{300, 500}	ProGNN α	{1e-4, 5e-4}
TSVD order	{20, 50}	ProGNN β	{1.5, 5}
Guard threshold	{1e-6, 1e-2}	ProGNN λ	{0, 0.001}
Jaccard threshold	{0.01, 0.03}	GLNN distillation temperature	[0.1, 5]
GARNET TSVD order	{30, 500}	GLNN distillation weight	{0.5, 1.}

C. Full MetaAttack Results

The full MetaAttack evaluation results on five homophilic and three heterophilic graphs are shown in Tables X, XI, XII, and XIII. Since the MetaAttack provided by DeepRobust [72] encounters an unknown error when the attack budget is 20% and the random seed is 18 on UAI, we skip UAI-Meta-20.

TABLE X
THE FULL ROBUSTNESS RESULTS AGAINST METAATTACK ON CORA AND CITESEER. THE ACCURACY MEAN AND STD OVER FIVE SPLITS ARE REPORTED. THE TOP THREE PERFORMING MODELS ARE HIGHLIGHTED IN BOLD, WITH THE BEST FURTHER UNDERLINED.

	Cora (HR=0.804)				Citeseer (HR=0.736)			
	5%	10%	15%	20%	5%	10%	15%	20%
MLP	65.69±1.43				66.01±1.37			
MLPw2	66.58±0.94				66.82±1.83			
MLPw4	67.02±1.12				66.43±1.73			
GCN	79.80±1.36	75.20±2.17	70.67±3.23	64.93±3.37	72.03±1.23	68.98±2.46	64.74±2.70	61.39±3.27
SGC	79.43±1.67	74.27±2.06	68.70±2.58	63.08±2.59	71.94±1.31	68.84±2.19	64.51±2.44	61.23±2.67
SAGE	80.33±1.31	77.20±1.90	73.79±2.70	71.39±2.74	72.68±1.25	72.29±0.52	70.40±1.05	67.97±2.31
RGCN	79.12±1.30	74.92±2.20	70.30±2.47	63.92±2.69	71.71±2.04	68.46±2.25	64.02±1.90	61.07±1.97
SVD	77.12±0.49	74.73±1.83	71.97±2.27	69.40±2.44	69.82±0.86	68.56±1.76	65.15±2.01	62.46±1.12
Jaccard	80.36±0.74	77.14±1.26	74.54±1.50	71.63±1.26	72.18±1.81	70.10±1.92	66.96±2.71	64.44±2.76
Guard	76.76±0.78	74.93±1.79	74.49±1.96	74.00±1.85	69.79±1.24	68.89±1.33	67.35±0.62	67.15±1.28
ProGNN	78.87±1.51	74.28±1.83	70.25±3.21	64.33±3.55	71.60±1.84	69.31±2.00	65.12±2.38	61.17±2.74
STABLE	81.68±0.73	80.09±0.38	79.20±1.00	77.58±1.53	74.33±1.08	73.95±0.74	73.32±1.14	72.50±0.88
EvenNet	**83.40±0.96**	**80.80±0.63**	78.36±1.10	75.14±1.79	74.08±1.02	72.97±1.19	70.95±1.71	69.40±1.14
GPR	82.13±1.06	79.74±0.32	77.41±0.70	75.21±0.97	73.40±1.04	72.30±1.28	69.82±1.89	67.83±1.80
GPR-GARNET	81.14±1.17	79.69±1.2	**81.07±0.64**	**80.70±0.79**	74.05±0.80	**74.56±1.14**	**74.49±1.50**	**73.71±1.24**
GLNN	80.12±1.37	77.35±1.97	73.95±1.92	71.94±3.22	74.25±1.20	73.46±0.73	71.92±1.38	70.02±2.10
GLNNw2	80.22±1.56	77.31±1.87	74.14±3.10	71.90±3.18	**74.44±1.51**	73.67±0.36	72.13±1.36	70.11±2.75
GLNNw4	80.21±1.51	77.34±1.87	74.60±2.24	72.76±2.82	74.01±1.32	73.37±0.71	71.94±1.38	70.37±2.72
MGMD-SAGE	**83.05±1.05**	**81.13±1.45**	**79.53±2.17**	**79.59±1.00**	**75.01±0.75**	**75.23±1.21**	**74.81±0.41**	**75.87±0.96**
MGMD-MLP	**82.87±1.05**	**80.72±1.63**	**79.27±2.33**	**78.64±0.77**	**75.31±1.18**	**75.15±0.98**	**74.79±0.64**	**75.47±1.10**

TABLE XI
THE FULL ROBUSTNESS RESULTS AGAINST METAATTACK ON ACM AND CORAML. THE ACCURACY MEAN AND STD OVER FIVE SPLITS ARE REPORTED. THE TOP THREE PERFORMING MODELS ARE HIGHLIGHTED IN BOLD, WITH THE BEST FURTHER UNDERLINED.

	ACM (HR=0.821)				CoraML (HR=0.784)			
	5%	10%	15%	20%	5%	10%	15%	20%
MLP	87.44±0.28				71.31±0.65			
MLPw2	87.36±0.25				70.69±1.78			
MLPw4	87.39±0.69				71.60±0.94			
GCN	80.15±3.46	75.65±4.94	71.26±5.70	68.47±6.36	81.78±0.82	77.5±0.72	73.27±1.99	66.54±3.17
SGC	82.46±2.36	80.12±2.12	77.75±2.37	74.70±1.72	78.54±1.37	72.94±1.44	67.34±1.85	58.88±3.70
SAGE	86.18±2.35	85.38±3.07	84.70±3.48	83.57±3.637	83.09±0.70	81.08±0.85	79.00±1.57	76.67±2.07
RGCN	79.90±1.80	74.59±3.62	71.07±3.81	67.51±3.30	81.94±0.81	77.14±0.77	73.35±1.57	67.02±2.78
SVD	85.98±1.36	84.98±0.81	83.28±1.82	82.10±1.77	80.61±0.61	79.76±0.99	78.84±0.65	77.42±0.97
Jaccard	80.15±3.46	75.65±4.94	71.26±5.70	68.47±6.36	81.50±0.77	77.84±0.56	74.63±1.20	69.81±2.26
Guard	76.22±2.36	71.63±6.86	70.20±8.07	65.78±7.01	76.66±0.85	76.75±0.93	76.47±0.82	76.35±0.63
ProGNN	79.89±2.01	77.80±1.98	74.77±4.35	69.13±7.62	81.87±1.20	78.27±1.25	75.86±2.23	70.49±3.76
STABLE	82.61±1.73	78.02±3.23	75.78±4.79	72.03±4.78	82.54±0.38	81.13±0.91	79.58±1.27	76.55±3.21
EvenNet	88.09±1.31	87.99±1.57	88.34±1.84	87.82±1.51	**85.55±0.38**	**83.88±0.50**	82.69±0.92	80.35±1.61
GPR	89.93±1.19	89.51±1.54	89.10±1.81	89.26±1.31	**85.33±0.79**	83.11±0.82	80.90±1.78	79.07±1.74
GPR-GARNET	88.02±1.45	89.26±0.80	**90.12±0.71**	**90.11±0.49**	82.27±1.11	81.37±1.69	**84.51±0.86**	**83.48±1.04**
GLNN	89.37±1.09	89.24±1.38	88.65±2.29	88.48±1.50	83.82±0.66	82.14±0.92	80.29±1.91	77.87±1.74
GLNNw2	**90.29±1.05**	90.11±1.56	89.31±1.41	88.84±1.83	84.19±0.69	82.46±1.01	80.43±1.90	78.31±1.89
GLNNw4	89.72±1.42	**90.31±1.12**	89.35±1.58	88.16±2.56	84.32±0.50	82.76±0.85	80.45±2.43	79.00±2.14
MGMD-SAGE	**92.03±1.27**	**91.68±0.84**	**91.24±0.65**	**90.93±1.06**	**85.18±0.61**	**84.29±0.70**	**83.19±0.54**	**82.34±0.80**
MGMD-MLP	**92.12±0.60**	**91.74±0.93**	**91.31±0.59**	**91.12±1.01**	85.17±0.63	**84.33±0.74**	**83.82±0.66**	**82.28±0.56**

TABLE XIII
THE FULL ROBUSTNESS RESULTS AGAINST METAATTACK ON CHAMELEON AND UAI. THE ACCURACY MEAN AND STD OVER FIVE SPLITS ARE REPORTED. THE TOP THREE PERFORMING MODELS ARE HIGHLIGHTED IN BOLD, WITH THE BEST FURTHER UNDERLINED.

	Chameleon (HR=0.230)				UAI (HR=0.364)		
	5%	10%	15%	20%	5%	10%	15%
MLP	42.65±0.86				61.74±2.11		
MLPw2	42.91±0.48				64.13±1.40		
MLPw4	41.08±2.05				62.71±1.91		
GCN	36.30±3.98	31.70±3.28	28.91±1.56	27.39±1.95	56.72±4.68	55.47±3.56	54.22±3.17
SGC	34.92±3.12	29.78±2.42	27.39±3.31	27.26±3.30	58.78±3.34	58.42±3.69	56.52±2.64
SAGE	40.35±4.28	38.67±3.71	36.85±4.53	35.20±4.35	60.02±3.21	59.53±4.61	60.18±2.65
RGCN	37.53±3.77	31.43±1.62	29.57±2.52	28.34±2.29	49.89±2.85	50.00±3.249	48.40±2.74
SVD	39.36±2.03	32.22±4.23	29.66±2.41	28.42±2.19	48.65±1.14	47.24±1.53	44.87±1.18
Jaccard	41.16±1.51	40.76±0.63	36.71±2.63	37.97±0.96	54.08±4.18	51.99±8.29	50.64±2.69
Guard	39.55±2.19	38.61±2.26	36.88±1.57	36.13±1.51	20.28±10.99	18.26±3.07	20.36±8.27
ProGNN	37.20±1.93	31.92±3.20	29.78±3.06	28.77±2.74	49.22±5.22	46.25±3.99	38.43±11.55
STABLE	**42.68±3.76**	**41.61±3.10**	35.45±5.98	34.38±5.34	51.78±2.08	49.07±2.72	47.63±2.26
EvenNet	37.53±3.08	32.25±2.77	30.68±1.81	29.41±2.06	**67.8±2.029**	**67.48±1.64**	**66.91±2.18**
GPR	37.97±2.08	35.49±3.95	36.04±1.25	34.28±4.45	35.38±9.24	32.09±7.86	34.75±11.11
GPR-GARNET	42.41±2.59	40.54±2.32	**39.24±2.20**	**38.68±2.53**	32.39±5.32	29.82±1.81	28.17±2.75
GLNN	39.00±3.12	36.99±3.40	37.14±3.91	35.57±4.10	62.46±2.91	62.64±3.18	62.02±2.00
GLNNw2	39.62±3.26	37.31±3.23	35.41±5.28	33.80±5.84	62.89±3.10	63.11±2.77	63.15±1.22
GLNNw4	39.20±3.31	37.59±3.18	35.57±5.12	34.75±5.02	62.62±2.39	62.62±3.21	62.75±1.99
MGMD-SAGE	**47.89±0.69**	**46.30±1.80**	**46.27±1.16**	**45.69±1.04**	**69.86±0.58**	**69.06±0.50**	**69.52±0.46**
MGMD-MLP	**44.56±1.71**	**45.10±1.25**	**44.81±0.74**	**44.57±1.92**	**68.31±0.59**	**68.35±0.44**	**69.10±0.45**

TABLE XII
THE FULL ROBUSTNESS RESULTS AGAINST METAATTACK ON POLBLOGS AND TEXAS. THE ACCURACY MEAN AND STD OVER FIVE SPLITS ARE REPORTED. THE TOP THREE PERFORMING MODELS ARE HIGHLIGHTED IN BOLD, WITH THE BEST FURTHER UNDERLINED.

	Polblogs (HR=0.906)				Texas (HR=0.061)			
	5%	10%	15%	20%	5%	10%	15%	20%
MLP	52.21±0.61				65.71±4.42			
MLPw2	52.09±0.24				67.21±3.02			
MLPw4	51.72±0.87				68.98±3.32			
GCN	77.18±1.76	71.64±1.64	67.53±0.99	66.07±1.05	49.25±5.43	46.67±4.48	49.39±2.29	46.53±7.16
SGC	77.71±1.79	70.74±2.52	66.95±1.36	64.58±1.34	53.88±2.23	53.20±2.12	55.24±2.12	53.88±2.82
SAGE	90.39±0.66	82.60±1.31	77.34±3.74	73.31±2.74	62.99±3.39	63.67±3.35	64.35±2.99	65.31±3.33
RGCN	75.42±1.29	69.18±2.36	66.18±0.64	65.03±0.87	52.93±1.89	51.43±3.10	49.52±8.10	51.84±3.89
SVD	92.43±0.70	86.46±3.46	73.44±1.77	69.69±2.28	49.66±4.02	46.12±7.13	48.57±5.66	52.38±5.02
Jaccard	50.88±1.69	50.88±1.69	50.88±1.69	50.88±1.69	49.25±5.43	46.67±4.48	49.39±2.29	46.53±7.16
Guard	51.58±0.57	51.58±0.57	51.58±0.57	51.58±0.57	48.03±12.96	49.66±7.47	47.76±11.40	42.45±12.54
ProGNN	85.97±5.16	78.71±5.51	72.78±3.43	69.14±2.09	47.89±10.06	46.67±6.19	45.31±14.47	49.12±4.57
STABLE	**92.80±2.38**	86.28±4.17	**88.55±0.38**	**88.55±0.38**	52.27±2.82	51.47±3.00	50.52±3.24	47.91±7.55
EvenNet	87.04±1.45	**94.68±0.45**	68.06±1.50	64.05±1.25	62.45±2.70	66.26±2.95	63.27±2.85	63.13±3.95
GPR	69.45±1.08	61.82±1.45	56.13±2.01	55.42±2.32	54.15±2.75	54.01±6.91	51.02±7.37	48.71±6.08
GPR-GARNET	72.91±0.84	63.05±1.68	59.57±1.74	60.22±1.53	54.97±6.39	61.5±6.78	57.55±3.95	56.19±6.74
GLNN	91.62±1.35	83.52±2.32	77.46±3.73	73.35±2.65	66.40±2.57	67.35±6.14	**66.53±5.45**	68.44±5.10
GLNNw2	91.55±1.11	83.37±2.59	77.14±3.98	73.21±2.85	66.67±2.93	**67.62±6.17**	66.12±5.49	**69.66±4.82**
GLNNw4	91.19±1.41	84.27±2.49	77.12±3.58	73.25±2.73	**67.21±2.70**	67.48±6.58	66.40±4.92	68.03±4.74
MGMD-SAGE	**93.95±1.34**	**92.92±1.07**	**92.27±2.26**	**92.09±0.57**	**68.84±5.65**	**69.66±1.64**	**71.02±2.30**	**72.25±2.90**
MGMD-MLP	**93.99±0.76**	**94.01±0.59**	**93.95±0.34**	**93.21±0.56**	**72.11±2.06**	**74.01±2.90**	**73.20±1.53**	**73.61±2.99**