DMSD: VALIDATION-GUIDED DUAL-BRANCH MLP SELF-DISTILLATION FOR HETEROPHILIC GRAPH LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Transferring graph structural knowledge to multilayer perceptrons (MLPs) offers a practical approach to efficient node classification. However, the effectiveness of topological supervision varies across graphs, particularly under heterophily, where local relations and broader propagation contexts provide different learning signals. Our analysis shows that neither route consistently dominates across the evaluated datasets, motivating an alternative to fixed topological supervision. We propose DMSD, a validation-guided, teacher-free dual-branch MLP self-distillation framework for heterophilic graph learning. DMSD independently trains local-relation and propagation-based candidates, each integrating topology-guided and attribute-guided self-distillation to learn complementary structural and feature representations without an external GNN teacher. Rather than uniformly averaging their predictions, DMSD computes a shared pair of fusion weights for each data split by applying a fixed-temperature sigmoid to the candidates' validation accuracy difference. Their predictions are then combined in log-probability space, without using test labels or training an additional gating network. Experiments on ten node classification datasets show that DMSD achieves the highest mean accuracy among the compared methods on eight datasets, including all five heterophilic graphs, with an average gain of 3.97 percentage points over the strongest baseline on each heterophilic dataset. Ablation results further show that validation-guided soft fusion improves macro-averaged accuracy over uniform fusion by 2.98 percentage points.

1 INTRODUCTION

Graph neural networks (GNNs) have become an important framework for learning from non-Euclidean structured data and have achieved strong performance in node classification, link prediction, and graph classification (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Xu et al., 2019). Modern GNNs follow the message-passing paradigm and learn node representations by aggregating node features and structural information from neighboring nodes (Gilmer et al., 2017). GNNs have been widely applied to recommender systems (Fan et al., 2019; He et al., 2020; Wu et al., 2023b), fraud detection (Liu et al., 2021), information retrieval (Mao et al., 2020), and text-attributed graph learning (Xu & Ding, 2026). Subsequent studies have explored scalable neighborhood sampling (Hamilton et al., 2017), attention-based aggregation (Veličković et al., 2018), higher-order structural modeling (Xu et al., 2019), adaptive propagation (Chien et al., 2021), learning on heterophilic graphs (Pei et al., 2020; Zhu et al., 2020; Lim et al., 2021), and robust graph learning (Lei et al., 2026). Together, these developments have broadened the applicability of GNNs to graphs with different scales and complex topologies.

Despite this progress, deploying GNNs in large-scale and latency-sensitive applications remains challenging because of their inference cost. Message passing requires repeated access to graph structure and neighborhood features. With multiple propagation layers, more nodes are involved in each prediction, further increasing computation and memory costs (Zhang et al., 2022; Yang et al., 2024; Feng et al., 2024). In contrast, multilayer perceptrons (MLPs) make predictions from node features alone and require no graph access at inference. Motivated by this efficiency, GNN-to-MLP methods use knowledge distillation to transfer structural knowledge from a pretrained GNN to an

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

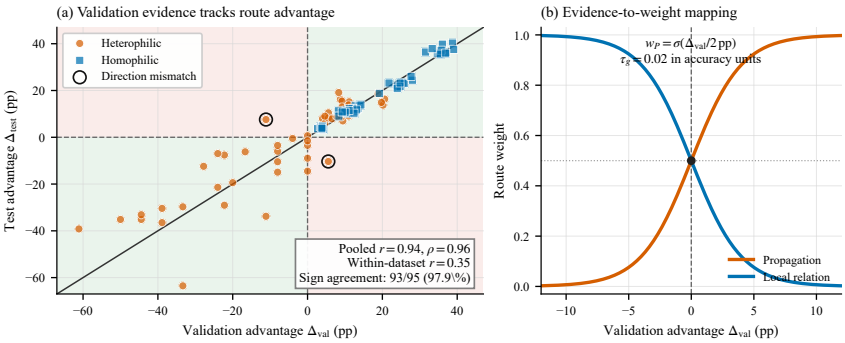

Figure 1: Validation evidence for topology-route fusion. (a) The validation advantage of propagation over local-relation learning closely tracks its test advantage across datasets and data splits. Outlined markers denote the few direction mismatches. (b) A fixed-temperature sigmoid maps the validation advantage to continuous route weights.

MLP student (Hinton et al., 2015; Zhang et al., 2022; Wu et al., 2023a; Yang et al., 2024; Feng et al., 2024; Lu et al., 2024). The student learns from structural supervision during training but requires only node features at inference. However, training a separate GNN teacher adds computational cost and makes the supervision dependent on teacher quality. Teacher-free graph self-distillation instead generates supervisory signals within the model. It removes the external teacher and its pretraining stage, enables MLP-based prediction, and promotes stable representations through internal consistency learning (Wu et al., 2024; Gao et al., 2026).

Although teacher-free graph self-distillation removes the need for an external GNN teacher, its structural supervision is still often constructed from predefined topology objectives or fixed combination rules (Wu et al., 2024; Gao et al., 2026). Local-relation learning and propagation-based learning introduce different structural inductive biases: the former preserves direct node-pair relations, whereas the latter captures broader structural context. This distinction is particularly important on heterophilic graphs, where adjacent nodes may belong to different classes and propagation may introduce class-inconsistent information (Pei et al., 2020; Zhu et al., 2020; Chien et al., 2021; Lim et al., 2021). As shown in Figure 1, our paired analysis further indicates that neither route consistently dominates across graphs and data splits. Consequently, a fixed route may introduce a topological bias that is unsuitable for the current task, whereas uniform fusion may dilute the more reliable structural signal. Both choices can reduce the quality of MLP representations and limit distillation performance.

To address this limitation, we propose DMSD, a teacher-free dual-branch MLP self-distillation framework for learning node representations on heterophilic graphs. For each topology route, DMSD builds two complementary branches: topology-guided self-distillation (TSD) and attribute-guided self-distillation (ASD). The TSD branch constructs structural supervision from either direct node relations or propagated structural context, yielding a local-relation candidate and a propagation candidate. The ASD branch learns complementary attribute information directly from the original node features without relying on graph edges. Each branch follows an online–target architecture, in which an exponential moving average target encoder provides stable self-distillation signals. A cross-branch consistency objective further aligns the topology-guided and attribute-guided representations. DMSD then treats the validation accuracy difference between the two candidates as observable evidence of their relative reliability and maps it to a continuous fusion weight using a fixed-temperature sigmoid function. As shown in Figure 1, clear validation evidence assigns more weight to the stronger route, whereas similar validation performance preserves information from both routes. The controller uses neither test labels nor an additional trainable gating network. At inference, the MLP encoders perform no learned message passing, but the propagation candidate still requires graph-dependent preprocessing before fusion.

The main contributions of this work are summarized as follows: (i) We systematically analyze the relative performance of local-relation learning and propagation-based learning across different graphs and data splits. Our analysis reveals the complementary behavior of the two topology routes. Based on this finding, we propose a soft controller driven by validation evidence, without using test labels or an additional trainable gating network. (ii) We propose DMSD, a teacher-free dual-branch

MLP self-distillation framework. It learns complementary representations through topology-guided self-distillation (TSD) and attribute-guided self-distillation (ASD), using MLP encoders without learned message-passing layers. (iii) Extensive experiments across ten benchmark datasets show that DMSD achieves the highest mean accuracy among all compared methods on eight datasets. DMSD improves macro-averaged accuracy over the standalone MLP and GCN baselines by **11.74** and **8.26** percentage points, respectively. On the five heterophilic graphs, it further outperforms the strongest competing method on each dataset by an average of **3.97 percentage points**.

2 Related Work

Knowledge distillation for GNNs. Existing graph knowledge distillation methods mainly transfer knowledge from a teacher GNN to a smaller student GNN (GNN-to-GNN distillation) or to MLPs (GNN-to-MLP distillation). In GNN-to-GNN distillation, prior studies preserve structural knowledge by matching predictions, local relations, or hidden representations (Yang et al., 2021; Yan et al., 2020; Deng & Zhang, 2021). However, the student GNN still requires access to graph structure and message passing during inference. GNN-to-MLP distillation removes this dependence by transferring structural knowledge to an MLP that takes only node features as input (Zhang et al., 2022; Zheng et al., 2022; Tian et al., 2023). For example, GLNN uses class predictions from a teacher GNN to supervise the student MLP (Zhang et al., 2022); VQGraph uses a discrete codebook to represent fine-grained local structures (Yang et al., 2024); LightHGNN transfers high-order structural knowledge from hypergraph neural networks to MLPs (Feng et al., 2024); and T2-GNN models teacher knowledge at both the feature and structure levels (Huo et al., 2023). Although these methods provide richer structural supervision, they usually rely on separately pretrained graph teachers. Teacher-free graph self-distillation instead produces supervisory signals within the model, often using an online encoder and a momentum-updated target encoder (Grill et al., 2020; Thakoor et al., 2022). TGS uses neighborhood structure to guide self-distillation (Wu et al., 2024), while SSD learns topology and feature invariance in separate blocks and exchanges knowledge between them through cross-distillation (Gao et al., 2026). Unlike SSD's graph-encoder-based topology block, DMSD uses MLP online and EMA target encoders in both its topology- and attribute-guided branches. It instantiates this dual-branch architecture as independently trained local-relation and propagation-based candidates, then combines their predictions using validation evidence.

Graph learning under heterophily. Most message-passing GNNs aggregate neighborhood information under the homophily assumption that adjacent nodes have similar semantics. However, in heterophilic graphs, adjacent nodes often belong to different classes. Direct aggregation may therefore introduce class-inconsistent information into node representations and degrade model performance (Pei et al., 2020; Zhu et al., 2020). Existing methods address this problem by adapting propagation or decoupling feature and structural information. GPR-GNN learns generalized PageRank weights over different propagation orders to control the contribution of multi-hop neighborhood information (Chien et al., 2021). FAGCN uses signed message aggregation to preserve high-frequency information that is useful under heterophily (Bo et al., 2021). LINKX encodes node features and adjacency information separately, avoiding their direct coupling through neighborhood aggregation (Lim et al., 2021). Recent analyses further show that conclusions on heterophilic graph learning can be sensitive to datasets, data splits, and model-selection protocols (Platonov et al., 2023). Although these methods improve representation learning under heterophily, they mainly design or learn topology-processing mechanisms within graph-dependent encoders and generally still require graph structure during inference. They do not directly address how observable evidence from the current task can estimate the relative reliability of different topology routes in message-passing-free MLP self-distillation. To address this gap, DMSD constructs local-relation and propagation-based candidates and maps their validation accuracy difference to a continuous fusion weight. This design adaptively combines the two topology biases without using test labels or training an additional gating network.

3 Preliminaries

Notation and Graph Neural Networks. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ denote an attributed graph, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and $\mathcal{E}$ are the node set and edge set, respectively. Here, $N = |\mathcal{V}|$ is the

number of nodes. The graph structure is represented by an adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$. We have $A_{ij} = 1$ if there is an edge between nodes v_i and v_j, and $A_{ij} = 0$ otherwise. The node feature matrix is denoted by $\mathbf{X} \in \mathbb{R}^{N\times F}$, where its i-th row $\mathbf{x}_i \in \mathbb{R}^F$ represents the feature vector of node v_i, and F is the feature dimension. For a node classification task with C classes, the label matrix is denoted by $\mathbf{Y} \in \{0,1\}^{N\times C}$, where $\mathbf{y}_i$ is the one-hot label vector of node v_i. Only a subset of nodes is usually labeled. We denote the labeled and unlabeled node sets by $\mathcal{V}^L$ and $\mathcal{V}^U$, respectively, where $\mathcal{V}^L \cap \mathcal{V}^U = \emptyset$ and $\mathcal{V}^L \cup \mathcal{V}^U = \mathcal{V}$. Graph neural networks learn node representations by aggregating neighborhood information through message passing (Gilmer et al., 2017). Let $\mathcal{N}(v)$ denote the neighbor set of node v, and let $\mathbf{h}_v^{(l)}$ denote its representation at the l-th layer, with $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. A general message-passing layer can be written as

$$\mathbf{h}_v^{(l)} = \mathrm{UPDATE}^{(l)}\left(\mathbf{h}_v^{(l-1)}, \mathrm{AGG}^{(l)}\left(\left\{\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}(v)\right\}\right)\right), \tag{1}$$

where $\mathrm{AGG}^{(l)}(\cdot)$ aggregates the representations of neighboring nodes from the previous layer, and $\mathrm{UPDATE}^{(l)}(\cdot)$ combines the aggregated information with the node representation to produce the updated representation.

EMA-Based Teacher-Free Self-Distillation. Traditional knowledge distillation usually relies on a separately pretrained teacher model. In contrast, teacher-free self-distillation generates supervision from the model itself and removes the additional teacher-training stage. This framework typically consists of an online encoder and a target encoder with the same architecture (Grill et al., 2020; Thakoor et al., 2022). The online encoder is optimized by backpropagation, while the target encoder receives no gradients and is updated by the exponential moving average of the online parameters. Since the target parameters change slowly, the target encoder can provide more stable learning targets. Given an input representation $\mathbf{U} \in \mathbb{R}^{N\times F}$, a schematic formulation is

$$\begin{aligned} \mathbf{Z} &= f_\theta(\mathbf{U}), \qquad \overline{\mathbf{Z}} = \mathrm{sg}(f_\xi(\mathbf{U})), \\ \mathcal{L}_{\mathrm{sd}} &= \mathcal{D}\big(\mathbf{Z}, \overline{\mathbf{Z}}\big), \qquad \xi \leftarrow m\xi + (1-m)\theta, \end{aligned} \tag{2}$$

where θ and ξ denote the parameters of the online and target encoders, respectively, and $\mathbf{Z}, \overline{\mathbf{Z}} \in \mathbb{R}^{N\times d}$, with d the encoder output dimension. $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation, $\mathcal{D}(\cdot,\cdot)$ is a consistency measure, and $m \in [0,1)$ is the momentum coefficient. The specific objectives, including pairwise relations and regularization, are defined in Sec. 4.1.

4 DMSD

We propose DMSD, a teacher-free dual-branch MLP self-distillation framework for heterophilic graphs. It adaptively combines candidate models with different topological inductive biases using validation evidence. Specifically, DMSD trains a local-relation candidate and a propagation-based candidate with the same architecture but independent parameters. Both candidates consist of topology-guided self-distillation (TSD) and attribute-guided self-distillation (ASD). The two branches learn stable and complementary representations through EMA target encoders and cross-branch consistency (Sec. 4.1). DMSD then generates a continuous weight from the validation accuracy difference between the two complete candidates and fuses their predictions in log-probability space (Sec. 4.2). The overall framework is shown in Fig. 2.

4.1 Learning Complementary Dual-Branch Candidates

Constructing Local and Propagation Candidates. DMSD separately trains a local-relation candidate $r = L$ and a propagation-based candidate $r = P$. The two candidates share the same dual-branch MLP architecture and training protocol but have independent parameters. Each candidate comprises a topology-guided self-distillation (TSD) branch and an attribute-guided self-distillation (ASD) branch. Their online encoders and EMA target encoders follow the mechanism introduced in Sec. 3, with independent parameter pairs $(\theta_b^{(r)}, \xi_b^{(r)})$ for each branch $b \in \{T, A\}$ and candidate $r \in \{L, P\}$. Each target is updated as $\xi_b^{(r)} \leftarrow m\xi_b^{(r)} + (1-m)\theta_b^{(r)}$. All branch encoders have

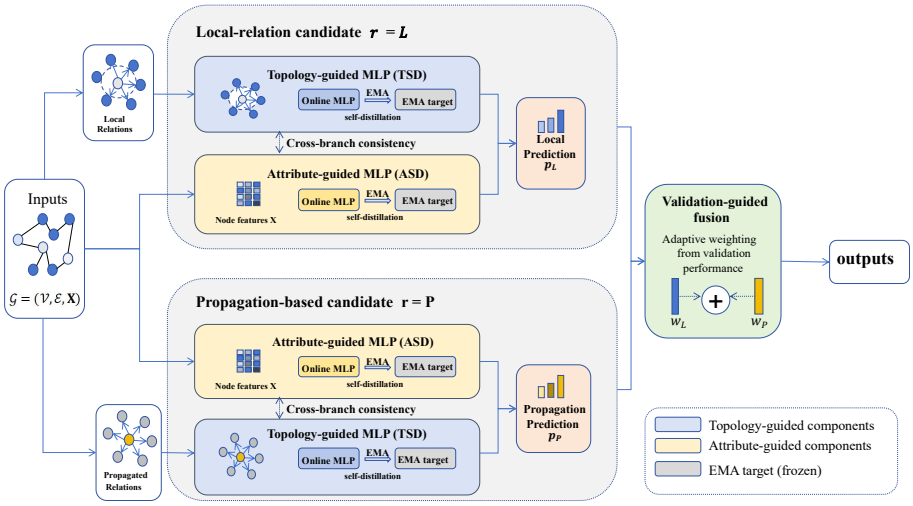

Figure 2: Overview of DMSD. Local-relation and propagation candidates are trained independently. Each candidate combines topology-guided and attribute-guided self-distillation, with optional cross-branch consistency. Validation evidence then determines a soft weight for log-probability fusion. Neither node count nor test labels are used by the controller.

output dimension d. The online representations are given by

$$\begin{aligned} \mathbf{Z}_T^{(r)} &= f_{\theta_T^{(r)}}(\mathbf{S}_r), & \mathbf{Z}_A^{(r)} &= f_{\theta_A^{(r)}}(\mathbf{X}), \\ \mathbf{S}_L &= \mathbf{X}, & \mathbf{S}_P &= \mathbf{X}^{(K)} = \widehat{\mathbf{A}}^K \mathbf{X}, \end{aligned} \tag{3}$$

where $\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}$, $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\widetilde{\mathbf{D}}$ is the corresponding degree matrix. Here, K denotes the number of parameter-free propagation steps. Both candidates use the original node attributes as input to ASD, while their TSD branches differ in input construction and self-distillation objectives.

Local-Relation TSD. For the local candidate, observed edges identify the node pairs used for training without neighborhood feature aggregation. We transform their raw feature similarity into a soft target and encourage the learned representations to preserve the corresponding pairwise relation:

$$\begin{aligned} \mathcal{L}_{\mathrm{TSD}}^{(L)} = \frac{1}{|\widetilde{\mathcal{E}}|} \sum_{(i,j)\in\widetilde{\mathcal{E}}} \left[\mathrm{sim}\left(\mathbf{z}_{T,i}^{(L)}, \mathbf{z}_{T,j}^{(L)}\right) - \sigma(\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j)) \right]^2 \\ + \lambda_{\mathrm{neg}} \mathcal{R}_{\mathrm{pair}}^{(L)}. \end{aligned} \tag{4}$$

Here, $\widetilde{\mathcal{E}}$ includes self-loops, $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity, and $\sigma(\cdot)$ is the sigmoid function. The regularizer $\mathcal{R}_{\mathrm{pair}}^{(L)}$ penalizes randomly sampled online–target node pairs whose cosine similarity exceeds a margin, discouraging representation collapse. We set $\lambda_{\mathrm{neg}} = 0.1$. This objective uses feature-derived relation targets rather than assuming that adjacent nodes belong to the same class.

Propagation-Based TSD. To capture structural context beyond direct connections, the propagation-based candidate first computes K-step parameter-free propagated features, as defined in Eq. (3). It then adopts a neighborhood-aware contrastive self-distillation objective (Gao et al., 2026). Let $s_{ij} = \mathrm{sim}(g(\mathbf{z}_{T,i}^{(P)}), \bar{\mathbf{z}}_{T,j}^{(P)})/\tau$, where g denotes the online prediction head, τ is the contrastive temperature, and the bar indicates a stop-gradient target representation. The objective is

$$\mathcal{L}_{\mathrm{TSD}}^{(P)} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|\widetilde{\mathcal{N}}(i)|} \sum_{j\in\widetilde{\mathcal{N}}(i)} \left[-s_{ij} + \log(C_i + \exp(s_{ij})) \right]. \tag{5}$$

Here, $\widetilde{\mathcal{N}}(i)$ includes node i, and C_i denotes the global contrastive term defined in Appendix A, Eq. (A.5). While the local-relation objective in Eq. (4) models direct node-pair relations, the

propagation-based objective in Eq. (5) uses propagated features to capture broader structural context. These objectives provide distinct topological inductive biases for subsequent candidate fusion.

ASD and Cross-Branch Consistency. In both candidates, ASD learns from the original node attributes by enforcing consistency between the normalized online and EMA target representations. A margin-based regularizer on randomly sampled node pairs further discourages representation collapse (Gao et al., 2026). Within each candidate, cross-branch consistency aligns the online representations of TSD and ASD. The effective objective for updating the online parameters is

$$\begin{aligned} \mathcal{L}_{\mathrm{ASD}}^{(r)} &= \mathrm{MSE}\Big(\mathrm{norm}(\mathbf{Z}_A^{(r)}), \mathrm{norm}(\overline{\mathbf{Z}}_A^{(r)})\Big) + 0.1\,\mathcal{R}_{\mathrm{pair},A}^{(r)}, \\ \mathcal{L}^{(r)} &= \mathcal{L}_{\mathrm{TSD}}^{(r)} + \mathcal{L}_{\mathrm{ASD}}^{(r)} + \lambda_{\mathrm{cross}}\,\mathrm{MSE}\Big(\mathbf{Z}_T^{(r)}, \mathbf{Z}_A^{(r)}\Big). \end{aligned} \tag{6}$$

Here, $\mathrm{norm}(\cdot)$ denotes row-wise ℓ_2 normalization, and λ_{cross} controls the cross-branch consistency term. After pretraining, we freeze the online encoders and train a linear classifier for each candidate using the branch readout configuration selected on the validation set. This yields two sets of candidate predictions, which are subsequently combined using validation evidence as described in Sec. 4.2.

4.2 Validation-Evidence Soft Fusion

After the candidate learning described in Sec. 4.1, we obtain two dual-branch MLP candidates with their respective classifiers, one trained using local relations and the other using propagated structural information. We aim to adaptively combine their predictions according to their relative performance on the current task. To this end, we introduce a validation-evidence soft fusion mechanism that maps their validation accuracy difference to a continuous weight and combines their predictions in log-probability space.

Assessing Candidate Reliability with Validation Evidence. We use the accuracy difference between the two candidates on the same validation set as a proxy for their relative reliability. Let A_L^{val} and A_P^{val} denote the validation accuracies of the local-relation and propagation-based candidates, respectively. We define the validation evidence as

$$\Delta_{\mathrm{val}} = A_P^{\mathrm{val}} - A_L^{\mathrm{val}}. \tag{7}$$

A positive value favors the propagation-based candidate, whereas a negative value favors the local-relation candidate. This task-specific measure directly reflects their relative validation performance and requires no additional network to evaluate the candidates.

Mapping Validation Evidence to Continuous Weights. We map the validation accuracy difference to candidate weights using a fixed-temperature sigmoid function:

$$w_P = \sigma\left(\frac{\Delta_{\mathrm{val}}}{\tau_g}\right), \qquad w_L = 1 - w_P, \tag{8}$$

where $\tau_g > 0$ controls sensitivity to the accuracy difference. We fix $\tau_g = 0.02$, with accuracies expressed in $[0, 1]$. The mapping favors the candidate with higher validation accuracy and approaches equal weighting as the difference vanishes. For each data split, a single pair of graph-level weights is computed on the validation set and shared across all prediction nodes. This procedure uses neither test labels nor additional trainable gating parameters.

Fusing Candidate Predictions. With the validation-derived weights, we combine the local-relation and propagation-based candidates to obtain the final node predictions. Specifically, let $\mathbf{O}_L$ and $\mathbf{O}_P$ denote the logits produced by their respective classifiers, and let $\mathbf{P}_r = \mathrm{softmax}(\mathbf{O}_r)$ be the corresponding class probabilities. We perform weighted fusion in log-probability space:

$$\mathbf{S} = w_L \log \mathbf{P}_L + w_P \log \mathbf{P}_P, \qquad \widehat{y}_i = \arg\max_c S_{ic}. \tag{9}$$

Here, S_{ic} is the fused score for assigning node i to class c. Maximizing this score is equivalent to maximizing $P_{L,ic}^{w_L} P_{P,ic}^{w_P}$, thereby combining the two candidates according to their relative validation performance. Both candidates are retained during deployment, while the fusion weights remain fixed after validation.

5 EXPERIMENTS

Datasets and Evaluation. We evaluate DMSD on ten public node classification datasets, including five heterophilic graphs, Texas, Cornell, Wisconsin, Chameleon, and Squirrel (Pei et al., 2020), and five homophilic graphs, Cora, Citeseer, Pubmed (Yang et al., 2016), Amazon Computers, and Amazon Photo (Shchur et al., 2018). Dataset statistics are summarized in Appendix B.1 (Table B.1). For each dataset, we select approximately 10% of the nodes for training through class-balanced sampling, reserve another 10% for validation, and use the remaining nodes for testing. All methods use the same dataset versions and ten fixed data splits. We report the mean and standard deviation of test accuracy across these splits.

Compared Methods. We compare DMSD with nine representative methods. These include an attribute-only MLP, the message-passing GCN (Kipf & Welling, 2017), and the adaptive propagation model GPR-GNN (Chien et al., 2021); GNN-to-MLP distillation methods GLNN (Zhang et al., 2022), VQGraph (Yang et al., 2024), AdaGMLP (Lu et al., 2024), and FF-G2M (Wu et al., 2023a); and teacher-free self-distillation methods TGS (Wu et al., 2024) and SSD (Gao et al., 2026). All methods use the same data splits and evaluation metric, and model selection is performed on the validation set.

5.1 MAIN RESULTS

Table 1 summarizes the node classification results of DMSD and nine baselines on ten datasets. DMSD achieves the highest mean accuracy on eight datasets, including all five heterophilic graphs and three homophilic graphs.

Performance on Heterophilic Graphs. DMSD achieves the highest mean accuracy on all five heterophilic datasets, with an average gain of **3.97 percentage points** over the strongest baseline on each dataset. The largest improvement occurs on Wisconsin (**9.51 percentage points**), followed by Squirrel (**3.76**) and Texas (**3.38**). The strongest baseline varies across datasets, whereas DMSD consistently outperforms the compared methods. The contributions of its candidates and fusion strategy are examined in the following analyses and ablations.

Performance on Homophilic Graphs. DMSD also remains competitive on homophilic graphs, achieving the highest mean accuracy on Cora, Pubmed, and Amazon Photo, with gains of **0.04**, **0.72**, and **0.55 percentage points**, respectively, over the strongest baseline. It ranks second on Amazon Computers, trailing GLNN by **1.27 percentage points**, and falls below TGS on Citeseer by **1.82 percentage points**. These results demonstrate its applicability beyond heterophilic graphs, although its advantage is dataset-dependent.

Table 1: Node classification accuracy (%) on ten datasets, reported as mean ± standard deviation over ten fixed data splits. Bold and underlined values denote the highest and second-highest mean accuracies, respectively. All configurations are selected using validation data only.

Dataset	Base Models			GNN-to-MLP				Self-Distillation		Ours
	MLP	GCN	GPR-GNN	VQGraph	GLNN	AdaGMLP	FF-G2M	SSD	TGS	**DMSD**
Heterophilic graphs										
Chameleon	42.53±1.48	<u>58.10±1.79</u>	52.23±6.17	52.20±1.72	54.27±0.95	53.85±0.74	36.44±3.71	56.87±1.49	38.17±2.62	**59.02±1.37**
Squirrel	30.53±1.36	39.19±0.73	38.20±4.81	33.91±1.18	37.80±1.09	37.86±1.86	24.62±2.26	<u>39.35±0.51</u>	28.80±2.93	**43.11±0.66**
Cornell	48.14±10.07	40.97±7.63	47.86±11.35	<u>53.10±7.14</u>	38.41±9.36	49.31±12.78	35.24±14.76	52.21±7.77	46.34±5.56	**55.38±7.47**
Texas	61.28±12.49	48.04±6.78	57.84±10.10	<u>67.77±6.15</u>	55.41±14.32	57.09±15.34	32.91±17.70	53.51±5.86	61.01±3.05	**71.15±4.09**
Wisconsin	59.10±12.07	46.07±10.40	<u>65.47±7.20</u>	60.15±17.39	53.78±9.59	64.08±8.25	43.63±14.73	54.23±6.11	47.36±8.42	**74.98±6.33**
Homophilic graphs										
Cora	57.80±2.43	80.24±1.17	81.57±0.99	75.41±2.76	79.77±1.92	80.83±0.94	80.22±1.24	<u>82.17±0.80</u>	81.68±0.96	**82.21±1.02**
Citeseer	58.10±2.04	68.09±1.58	<u>72.67±0.98</u>	65.72±3.37	71.36±4.02	69.26±12.89	71.73±3.03	70.79±1.47	**72.93±1.06**	71.11±0.97
Pubmed	82.58±0.50	85.12±0.44	<u>85.70±0.19</u>	85.52±0.27	85.48±0.34	85.41±0.21	84.95±0.23	84.74±0.13	84.64±0.66	**86.42±0.28**
Computers	79.64±0.91	84.14±0.75	86.04±1.52	81.04±0.84	**87.68±0.49**	86.25±0.59	80.22±2.25	82.94±0.63	83.18±0.85	<u>86.41±0.78</u>
Photo	85.66±0.80	90.22±0.53	91.52±0.54	89.49±1.18	91.28±0.39	89.40±1.13	91.45±0.45	90.85±0.32	<u>92.44±0.36</u>	**92.99±0.41**

5.2 Model Analysis

Complementarity of Topological Candidates. We compare candidates with the same architecture, training protocol, and data splits, differing only in TSD topology modeling. Appendix C.1 (Fig. C.1) shows dataset-dependent advantages: local relations perform better on some heterophilic graphs, while propagation is more effective on others. Neither route consistently dominates, motivating validation-guided combination. The gains over individual candidates and uniform fusion are examined in Sec. 5.3.

Reliability of Validation Evidence. We examine whether validation performance provides useful evidence of the two candidates' relative test performance. For each dataset and split, we compute the propagation-minus-local accuracy differences on the validation and test sets. As shown in Fig. 1(a), the pooled Pearson and Spearman correlations are **0.94** and **0.96**, respectively. Excluding validation ties, the two differences agree in sign in **97.9%** of cases ($n = 95$). These results suggest that validation advantage usually identifies the better-performing candidate on the test set across the evaluated datasets and splits. However, the within-dataset correlation is lower (**0.35**), indicating that the strong pooled association partly reflects between-dataset differences.

Accuracy–Efficiency Trade-off. Table 2 compares macro-averaged accuracy, training time, and end-to-end inference latency across ten datasets. DMSD's computational cost includes both complete candidates, the required propagation preprocessing, and prediction fusion. Retaining both candidates during deployment incurs higher inference latency than the evaluated single-MLP distillation methods. Its accuracy gains thus come at an additional computational cost, reflecting a trade-off between predictive performance and inference efficiency rather than a pursuit of minimum latency. Detailed measurement protocols and additional resource measurements are provided in the appendix. The corresponding accuracy–time plots are provided in Appendix C.4.

Table 2: Macro-averaged accuracy and time costs on ten datasets. Bold denotes the best value in each column.

Method	Acc. (%) ↑	Train (s) ↓	Infer. (ms) ↓
MLP	60.54	31.78	1.660
GCN	64.02	53.09	3.079
GPR-GNN	67.91	73.87	3.318
VQGraph	66.43	187.39	**0.284**
GLNN	65.52	74.66	1.497
AdaGMLP	67.33	42.47	1.318
FF-G2M	58.14	23.75	0.422
SSD	66.77	**10.63**	7.032
TGS	63.66	1063.38	1.917
DMSD	**72.28**	27.56	18.850

Accuracy comes from Table 1; timing comes from separate efficiency measurements, not the same runs. DMSD's timed run achieves 71.67% macro accuracy. Its timing includes both candidates and full input-to-logits prediction. Values are rounded.

5.3 Ablation Study

We analyze the contributions of candidate representation learning and candidate fusion strategies in Table 3.

Contribution of Self-Distillation Components. The full model achieves a macro-averaged accuracy of **72.28%**. Removing TSD, ASD, or cross-branch consistency reduces this score to **52.29%**, **69.78%**, and **66.67%**, respectively. The largest drop occurs without TSD, highlighting the importance of topology-guided supervision for candidate representation learning. Removing ASD and cross-branch consistency leads to decreases of **2.50** and **5.61 percentage points**, respectively, supporting the contributions of attribute-guided supervision and cross-branch consistency.

Effect of the Fusion Strategy. Both validation-based hard selection and soft fusion achieve higher macro-averaged accuracy than the two fixed routes, supporting the use of validation evidence to adapt to dataset-dependent candidate advantages. Soft fusion achieves **72.28%**, outperforming uniform fusion and validation-based hard selection by **2.98** and **0.91 percentage points**, respectively. It also achieves the highest mean accuracy among the compared fusion strategies on all ten datasets. These results support the effectiveness of validation-evidence fusion, although its gains vary across

datasets. The additional improvement over hard selection is relatively modest, suggesting that continuous weighting provides an incremental benefit beyond validation-based candidate selection.

Sensitivity to the Fusion Temperature. We assess the suitability of a shared temperature through sensitivity analysis and leave-one-graph-out (LOGO) validation. Across ten LOGO folds, $\tau_g = \mathbf{0.02}$ is selected nine times, while **0.01** is selected once, providing empirical support for using a shared temperature of **0.02**. Detailed settings, the selection procedure, and complete results are provided in Appendix C.2.

Table 3: Component ablations and fusion strategies on ten datasets. Entries are test accuracy (%, mean ± standard deviation across ten splits); the last row averages dataset means. All columns are taken from the same ablation experiment with matched data splits. Full Soft uses a fixed fusion temperature of 0.02.

Dataset	Full Soft	w/o TSD	w/o ASD	w/o Cross	Always Local	Always Propagation	Validation Hard	Uniform Fusion
Chameleon	59.02±1.37	41.78±1.17	56.10±2.41	56.11±2.17	42.90±1.48	58.23±1.34	58.23±1.34	58.43±1.52
Squirrel	43.11±0.66	31.70±1.54	39.90±0.67	41.15±1.74	33.12±0.86	41.58±1.14	41.58±1.14	41.12±1.63
Cornell	55.38±7.47	52.83±2.73	52.10±5.38	51.28±3.76	54.55±9.68	50.55±4.44	55.03±7.50	55.14±6.88
Texas	71.15±4.09	61.22±9.80	67.09±8.09	62.84±7.79	70.54±3.85	33.99±7.41	70.54±3.85	48.78±8.41
Wisconsin	74.98±6.33	64.38±4.68	71.89±6.96	72.94±4.52	71.14±5.62	63.48±6.66	71.94±5.20	74.58±4.74
Cora	82.21±1.02	53.12±1.90	80.88±1.19	67.82±1.93	59.66±1.60	81.71±0.90	81.71±0.90	80.59±1.06
Citeseer	71.11±0.97	54.77±1.43	68.67±0.89	58.01±1.43	33.54±1.51	70.78±1.09	70.78±1.09	69.59±1.67
Pubmed	86.42±0.28	78.19±0.60	84.72±0.49	85.37±0.24	82.52±0.46	85.45±0.27	85.45±0.27	86.39±0.29
Computers	86.41±0.78	42.93±2.57	85.73±1.04	82.32±0.65	73.41±0.81	85.74±0.51	85.74±0.51	85.89±0.50
Photo	92.99±0.41	41.97±2.53	90.73±0.39	88.85±0.57	82.03±0.51	92.65±0.44	92.65±0.44	92.47±0.31
Macro Avg.	72.28	52.29	69.78	66.67	60.34	66.42	71.37	69.30

5.4 Robustness to Edge Perturbations

We randomly delete or add edges at rates of 10%, 20%, and 30% on ten datasets and compare DMSD, GCN, and SSD under matched data splits and perturbation settings. For each perturbed graph, DMSD retrains both candidates and determines their fusion weights using validation evidence. Across all six perturbation settings, DMSD achieves the highest macro-averaged accuracy among the three methods. Under 30% edge deletion and addition, it reaches **69.73%** and **67.71%**, respectively, outperforming SSD by **3.76** and **4.04 percentage points**. Its advantage primarily lies in higher absolute accuracy after perturbation, rather than consistently smaller performance degradation. These findings do not establish the test-time robustness of a fixed model or adversarial robustness. Complete curves, per-dataset results, and statistical tests are provided in Appendix C.3.

6 Conclusion

We propose DMSD, a teacher-free dual-branch MLP self-distillation framework for heterophilic graphs. By integrating topology-guided and attribute-guided learning, DMSD constructs candidates with distinct topological inductive biases and combines their predictions using continuous weights derived from validation evidence, without an additional trainable gating network. Experiments on ten datasets show that DMSD achieves the highest mean accuracy among the compared methods on eight datasets, including all five heterophilic graphs. Future work will explore lighter fusion strategies to further improve the trade-off between accuracy and deployment cost.

AI use statement

In this work, we used generative AI tools, including OpenAI Codex, to aid and polish writing through Chinese–English translation and language editing. For retrieval and discovery, the tools helped identify potentially relevant literature, whose citations were checked against the original sources. For research ideation and execution, they provided feedback on methodology and experimental design, assisted with implementing and debugging experimental scripts, and helped organize

and interpret outputs. We also used them to draft and revise sections of the paper and to assist with manuscript structure, LaTeX formatting, tables, and plotting code. We did not use generative AI to create datasets, labels, or experimental measurements. All reported results were produced by executing the described code on the stated datasets. The authors reviewed the AI-assisted text and code, checked numerical claims against experiment logs, and take full responsibility for the final content of this work.

Reproducibility statement

To support reproducibility, Section 4 specifies the model architecture, training objectives, validation-evidence controller, and validation-only deployment procedure. Section 5 describes the datasets, fixed data splits, evaluation protocol, main implementation settings, software, and hardware. The appendix provides the complete preprocessing procedure, hyperparameters and search spaces, random seeds, baseline sources and configurations, edge-perturbation protocol, efficiency measurement procedure, and additional results. We also provide anonymized source code, configuration files, split masks, and scripts for reproducing the reported tables and figures as supplementary material.

References

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 3950–3957. AAAI Press, 2021. doi: 10.1609/AAAI.V35I5.16514.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

Xiang Deng and Zhongfei Zhang. Graph-free knowledge distillation for graph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2321–2327. International Joint Conferences on Artificial Intelligence Organization, 2021. doi: 10.24963/ijcai.2021/320.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, pp. 417–426. Association for Computing Machinery, 2019. doi: 10.1145/3308558.3313488.

Yifan Feng, Yihe Luo, Shihui Ying, and Yue Gao. Lighthgnn: Distilling hypergraph neural networks into mlps for 100x faster inference. In *International Conference on Learning Representations*, volume 2024, pp. 29822–29842, 2024.

Yuan Gao, Yuchen Li, Bingsheng He, Hezhe Qiao, Guoguo Ai, and Hui Yan. SSD: self-supervised distillation for heterophilic graph representation learning. *IEEE Trans. Knowl. Data Eng.*, 38(1): 631–644, 2026. doi: 10.1109/TKDE.2025.3621758.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1263–1272. PMLR, 2017.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284. Curran Associates, Inc., 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng 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, pp. 639–648. Association for Computing Machinery, 2020. doi: 10.1145/3397271.3401063.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, abs/1503.02531, 2015.

Cuiying Huo, Di Jin, Yawen Li, Dongxiao He, Yu-Bin Yang, and Lingfei Wu. T2-GNN: graph neural networks for graphs with incomplete features and structure via teacher-student distillation. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 4339–4346. AAAI Press, 2023. doi: 10.1609/AAAI.V37I4.25553.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Runlin Lei, Lu Yi, Mingguo He, Pengyu Qiu, Zhewei Wei, Yongchao Liu, and Chuntao Hong. Robustness in text-attributed graph learning: Insights, trade-offs, and new defenses. In *International Conference on Learning Representations*, volume 2026, pp. 49084–49131, 2026.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20887–20902. Curran Associates, Inc., 2021.

Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Pick and choose: A gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, pp. 3168–3177. Association for Computing Machinery, 2021. doi: 10.1145/3442381.3449989.

Weigang Lu, Ziyu Guan, Wei Zhao, and Yaming Yang. AdaGMLP: Adaboosting gnn-to-mlp knowledge distillation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2060–2071, 2024. doi: 10.1145/3637528.3671699.

Kelong Mao, Xi Xiao, Jieming Zhu, Biao Lu, Ruiming Tang, and Xiuqiang He. Item tagging for information retrieval: A tripartite graph neural network based approach. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2327–2336. Association for Computing Machinery, 2020. doi: 10.1145/3397271.3401438.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS*, 2018.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2022.

Yijun Tian, Chuxu Zhang, Zhichun Guo, Xiangliang Zhang, and Nitesh Chawla. Learning MLPs on graphs: A unified view of effectiveness, robustness, and efficiency. In *The Eleventh International Conference on Learning Representations*, 2023.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Lirong Wu, Haitao Lin, Yufei Huang, Tianyu Fan, and Stan Z. Li. Extracting low-/high- frequency knowledge from graph neural networks and injecting it into mlps: An effective gnn-to-mlp distillation framework. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 10351–10360. AAAI Press, 2023a. doi: 10.1609/AAAI.V37I9.26232.

Lirong Wu, Haitao Lin, Zhangyang Gao, Guojiang Zhao, and Stan Z. Li. A teacher-free graph knowledge distillation framework with dual self-distillation. *IEEE Trans. Knowl. Data Eng.*, 36 (9):4375–4385, 2024. doi: 10.1109/TKDE.2024.3374773.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5):97:1–97:37, 2023b. doi: 10.1145/3535101.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Ruiyao Xu and Kaize Ding. Gnn-as-judge: Unleashing the power of llms for graph learning with gnn feedback. In *International Conference on Learning Representations*, volume 2026, pp. 29844–29871, 2026.

Bencheng Yan, Chaokun Wang, Gaoyang Guo, and Yunkai Lou. Tinygnn: Learning efficient graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1848–1856. Association for Computing Machinery, 2020. doi: 10.1145/3394486.3403236.

Cheng Yang, Jiawei Liu, and Chuan Shi. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proceedings of the Web Conference 2021*, pp. 1227–1237. Association for Computing Machinery, 2021. doi: 10.1145/3442381.3450068.

Ling Yang, Ye Tian, Minkai Xu, Zhongyi Liu, Shenda Hong, Wei Qu, Wentao Zhang, Bin CUI, Muhan Zhang, and Jure Leskovec. Vqgraph: Rethinking graph representation space for bridging gnns and mlps. In *International Conference on Learning Representations*, volume 2024, pp. 16490–16509, 2024.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 40–48, 2016.

Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old MLPs new tricks via distillation. In *International Conference on Learning Representations*, 2022.

Wenqing Zheng, Edward W Huang, Nikhil Rao, Sumeet Katariya, Zhangyang Wang, and Karthik Subbian. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. In *International Conference on Learning Representations*, 2022.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7793–7804. Curran Associates, Inc., 2020.

A Additional Method Details

This appendix expands the intermediate definitions omitted from the main text. It does not introduce additional training objectives.

A.1 Detailed Self-Distillation Objectives

Local-relation candidate. For $r = L$, the online and target representations are

$$\mathbf{Z}_T = f_{\theta_T}(\mathbf{X}), \qquad \overline{\mathbf{Z}}_T = \mathrm{sg}(f_{\xi_T}(\mathbf{X})). \tag{A.1}$$

For $(i, j) \in \widetilde{\mathcal{E}}$, we use

$$a_{ij} = \sigma(\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j)), \qquad p_{ij} = \mathrm{sim}(\mathbf{z}_{T,i}, \mathbf{z}_{T,j}). \tag{A.2}$$

The random-pair regularizer in Equation 4 is

$$\mathcal{R}^{(L)}_{\mathrm{pair}} = \frac{1}{|\mathcal{P}^-|} \sum_{(i,j)\in\mathcal{P}^-} [\mathrm{sim}(\mathbf{z}_{T,i}, \overline{\mathbf{z}}_{T,j}) - \mu]_+, \tag{A.3}$$

where $\mathcal{P}^-$ contains randomly sampled node pairs, $[u]_+ = \max(0, u)$, and the margin is $\mu = 0.2$.

Propagation candidate. For $r = P$, define $\mathbf{Z}_T = f_{\theta_T}(\mathbf{X}^{(K)})$ and $\overline{\mathbf{Z}}_T = \mathrm{sg}(f_{\xi_T}(\mathbf{X}^{(K)}))$. The score and global normalizer used in Equation 5 are

$$s_{ij} = \frac{\mathrm{sim}(g(\mathbf{z}_{T,i}), \overline{\mathbf{z}}_{T,j})}{\tau}, \tag{A.4}$$

$$C_i = \sum_{k=1}^{N} \left[\exp\left(\frac{\mathrm{sim}(\mathbf{z}_{T,i}, \overline{\mathbf{z}}_{T,k})}{\tau} \right) + \exp\left(\frac{\mathrm{sim}(\mathbf{z}_{T,i}, \mathbf{z}_{T,k})}{\tau} \right) \right]. \tag{A.5}$$

Attribute branch. The ASD representations are $\mathbf{Z}_A = f_{\theta_A}(\mathbf{X})$ and $\overline{\mathbf{Z}}_A = \mathrm{sg}(f_{\xi_A}(\mathbf{X}))$. The ASD loss in Equation 6 combines the following terms, with coefficient 0.1 on the random-pair regularizer:

$$\mathcal{L}_{\mathrm{attr}} = \frac{1}{Nd} \| \mathrm{norm}(\mathbf{Z}_A) - \mathrm{sg}(\mathrm{norm}(\overline{\mathbf{Z}}_A)) \|_F^2, \tag{A.6}$$

$$\mathcal{R}_{\mathrm{pair},A} = \frac{1}{|\mathcal{P}_A|} \sum_{(i,j)\in\mathcal{P}_A} [\mathrm{sim}(\mathbf{z}_{A,i}, \overline{\mathbf{z}}_{A,j}) - \mu]_+. \tag{A.7}$$

The candidate superscript is omitted above because the same definitions apply independently to both candidates. The normalization is row-wise, and the margin is $\mu = 0.2$. The cross-branch coefficient $\lambda_{\mathrm{cross}} \in \{0, 1\}$ is selected using validation data. The implementation also records the mean squared error between the two stop-gradient target representations. This term does not contribute to the online-parameter gradients and is therefore omitted from the effective objective in Equation 6.

A.2 Candidate Readout and Prediction

The input and branch embeddings retained after pretraining are

$$\begin{aligned} \mathbf{X}^{(r)}_T &= \begin{cases} \mathbf{X}, & r = L, \\ \mathbf{X}^{(K)}, & r = P, \end{cases} \\ \mathbf{H}^{(r)}_T &= f_{\theta^{(r)}_T}(\mathbf{X}^{(r)}_T), \qquad \mathbf{H}^{(r)}_A = f_{\theta^{(r)}_A}(\mathbf{X}). \end{aligned} \tag{A.8}$$

After pretraining, the target encoders and predictor are discarded. The online encoders remain frozen during linear-classifier training. The candidate representation is

$$\mathbf{H}_r(\alpha) = \alpha \mathbf{H}^{(r)}_T + (1 - \alpha)\mathbf{H}^{(r)}_A, \qquad \alpha \in \{0, 0.1, \ldots, 1\}. \tag{A.9}$$

Thus, $\alpha = 0$ uses ASD only, $\alpha = 1$ uses TSD only, and intermediate values fuse both branches. For each dataset, one readout configuration is selected by mean validation accuracy over the ten splits and applied to both candidates, following the evaluation protocol. For each validation-selected candidate representation $\mathbf{H}_r$, the linear classifier returns $\mathbf{O}_r = \mathbf{H}_r \mathbf{W}_r + \mathbf{b}_r$. Their final prediction follows Equations 8 and 9.

B Experimental Details

B.1 Datasets and Data Splits

Datasets and versions. We evaluate on five heterophilic graphs (Texas, Cornell, Wisconsin, Chameleon, and Squirrel) and five homophilic graphs (Cora, Citeseer, Pubmed, Amazon Computers, and Amazon Photo), as introduced in Section 5. Texas, Cornell, and Wisconsin are WebKB graphs; Chameleon and Squirrel use the Geom-GCN-preprocessed WikipediaNetwork data. The citation and Amazon graphs are loaded through DGL dataset interfaces. For Amazon Computers and Amazon Photo, we retain all nodes returned by DGL (13,752 and 7,650, respectively) rather than restricting the graphs to the largest connected component as in Shchur et al. (2018). The Cornell experiments use the corrected Geom-GCN raw files rather than the older cached WebKB version. The recorded SHA-256 identifier of the corrected node-feature-and-label file begins with `38561ad44e29199a`, distinguishing it from the older file beginning with `cf5a3ca346cdd121`. Dataset statistics are summarized in Table B.1.

Graph and feature preprocessing. For WebKB and WikipediaNetwork graphs, the edge lists are converted to undirected graphs. WebKB feature vectors are row-normalized by dividing each row by its feature sum, with the denominator bounded below by one. For the remaining datasets, we retain the features returned by their loaders without additional normalization in the training script. Before DMSD training, existing self-loops are removed and one self-loop is added per node. The edge counts in Table B.1 refer to unique undirected node pairs excluding self-loops; edge homophily is the fraction of these pairs whose endpoint labels agree. Experiments use a transductive setting: graph structure and attributes are available for all nodes, whereas classification supervision uses training labels only.

Data splits and evaluation. We generate ten fixed splits instead of using the datasets' supplied split masks. For a graph with N nodes and C classes, we sample $\mathrm{round}(0.1N/C)$ training nodes per class without replacement. If a class contains fewer nodes than this quota, all its nodes are used for training. We then sample $\mathrm{round}(0.1N)$ validation nodes uniformly without replacement from the remaining nodes; validation sampling is not class-balanced. All remaining nodes form the test set. Rounding follows the implementation's nearest-integer rule, with half-integer ties rounded to the even integer. Consequently, split fractions are approximately 10%/10%/80%, rather than exactly fixed. The split seeds are 77, 194, 419, 47, 121, 401, 210, 164, 629, and 242. All methods use the same dataset versions and node partitions. We report the mean and standard deviation of test accuracy across these splits. Model-selection and repeated-training details are given in Appendix B.2.

Table B.1: Statistics of the ten offline graphs used in our experiments. Edges denote unique undirected node pairs after removing self-loops, and homophily is computed on the same edge set. Heterophilic and homophilic graphs are shown in separate blocks.

Type	Dataset	Nodes	Features	Edges	Classes	Homophily
Hete.	Chameleon	2,277	2,325	31,371	5	0.230
	Squirrel	5,201	2,089	198,353	5	0.222
	Cornell	183	1,703	277	5	0.123
	Texas	183	1,703	279	5	0.061
	Wisconsin	251	1,703	450	5	0.178
Homo.	Cora	2,708	1,433	5,278	7	0.810
	Citeseer	3,327	3,703	4,552	6	0.736
	Pubmed	19,717	500	44,324	3	0.802
	Computers	13,752	767	245,861	10	0.777
	Photo	7,650	745	119,081	8	0.827

B.2 Training and Model Selection

Training configuration. Both candidates use two-layer MLP encoders with hidden and output dimensions of 2048, batch normalization and ReLU between linear layers, and no dropout. Pre-

training runs for 50 epochs with model seed 1024. Detached embeddings are reused across the ten data splits. Linear classifiers are trained with cross-entropy loss for 500 epochs; the checkpoint with highest validation accuracy is retained, with ties resolved in favor of the later checkpoint.

Dataset-specific settings. Table B.3 lists settings shared by both candidates. As in Appendix A.2, α weights TSD and $1-\alpha$ weights ASD. The table gives effective weights for single-branch readouts. Fusion weights are computed separately for each split using validation accuracy, with a fixed fusion temperature of 0.02.

Table B.2: Shared DMSD training settings.

Parameter	Value
Encoder layers	2
Hidden / output dimension	2048 / 2048
Optimizer (both stages)	Adam
Pretraining epochs	50
Pretraining learning rate	5×10^{-4}
Pretraining weight decay	10^{-6}
Linear-classifier epochs	500
Linear-classifier learning rate	10^{-2}
Linear-classifier weight decay	10^{-5}
EMA momentum	0.99
Propagation steps K	3
Propagation contrastive temperature	0.75
Negative-pair margin	0.2
Negative-pair regularization weight	0.1
Fusion temperature τ_g	0.02
Model seed	1024
Number of data splits	10

Table B.3: Dataset-specific DMSD settings.

Dataset	Readout	α	λ_{cross}
Chameleon	TSD only	1.0	1
Squirrel	Weighted sum	0.9	1
Cornell	Weighted sum	0.4	1
Texas	Weighted sum	0.8	1
Wisconsin	Weighted sum	0.4	1
Cora	Weighted sum	0.3	1
Citeseer	ASD only	0.0	1
Pubmed	Weighted sum	0.5	1
Computers	Weighted sum	0.5	1
Photo	Weighted sum	0.3	1

Repeated evaluation. The split seeds are 77, 194, 419, 47, 121, 401, 210, 164, 629, and 242. The ten repetitions correspond to different data splits, not ten independent pretraining runs. Test labels do not determine classifier checkpoint selection or fusion weights.

B.3 Efficiency Measurement Protocol

Experimental Environment. All methods in Table 2 are timed on the same server equipped with a single NVIDIA RTX 3090 GPU with 24 GB of memory, 14 allocated vCPUs from an Intel Xeon Gold 6330 processor at 2.00 GHz, and 90 GB of RAM. The software environment consists of Ubuntu 22.04, Python 3.10, PyTorch 2.1.0, and CUDA 12.1.

Training Time. We measure DMSD's training cost as the sum of the sequential training times of its local-relation and propagation-based candidates. For each candidate, the measurement includes topology preprocessing, model initialization, self-supervised pretraining, embedding extraction, and linear-classifier training and validation across all ten data splits. It excludes dataset loading and hyperparameter search.

Inference Latency. For DMSD, we report full-graph input-to-logits latency rather than classifier-only latency on cached embeddings. For each candidate, timing includes the encoder and linear classifier, together with propagation preprocessing when required by the selected readout. DMSD's total latency is the sum of the two candidate latencies and the separately measured log-probability fusion cost. Graph data and features are already resident on the execution device; disk loading and host-to-device transfer are excluded. Each timed operation uses five warm-up iterations followed by twenty measured repetitions, with GPU synchronization immediately before and after the timed repetitions.

Aggregation and Reporting. Candidate inference latencies are averaged over the classifiers trained on the ten splits. Training time and inference latency are then averaged equally across the ten datasets. The accuracy values in Table 2 come from the main classification experiments, whereas timing comes from separate efficiency runs, in which DMSD achieves 71.67% macro-averaged accuracy. Thus, the accuracy and timing values do not represent measurements from the same run.

C Additional Experimental Results

C.1 Complementarity of Topological Candidates

Figure C.1 reports paired test accuracy differences between the two candidates on the ten fixed splits of each dataset.

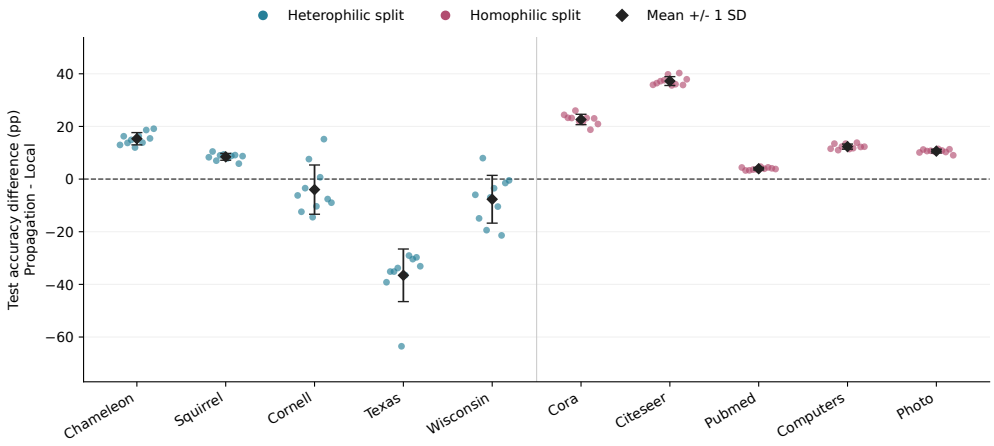

Figure C.1: Propagation-minus-local test accuracy across ten splits. Positive values favor propagation; diamonds and error bars denote mean $\pm$ 1 SD.

C.2 Fusion-Temperature Sensitivity and LOGO Validation

The fusion temperature τ_g controls the sensitivity of the weights to validation accuracy differences. Smaller temperatures approach hard selection, whereas larger temperatures approach uniform fusion. We evaluate $\tau_g \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0\}$ using leave-one-graph-out (LOGO) validation. In each fold, we hold out one dataset and select the temperature using the macro-averaged validation accuracy on the remaining nine datasets, then apply it to the held-out dataset. As shown in Fig. C.2, nine folds select 0.02; only the fold holding out Wisconsin selects 0.01. These results support using 0.02 as a shared temperature, without implying that it is optimal for every dataset. This

verification concerns temperature selection only; it does not imply that all model configurations are selected without validation labels from the target graph.

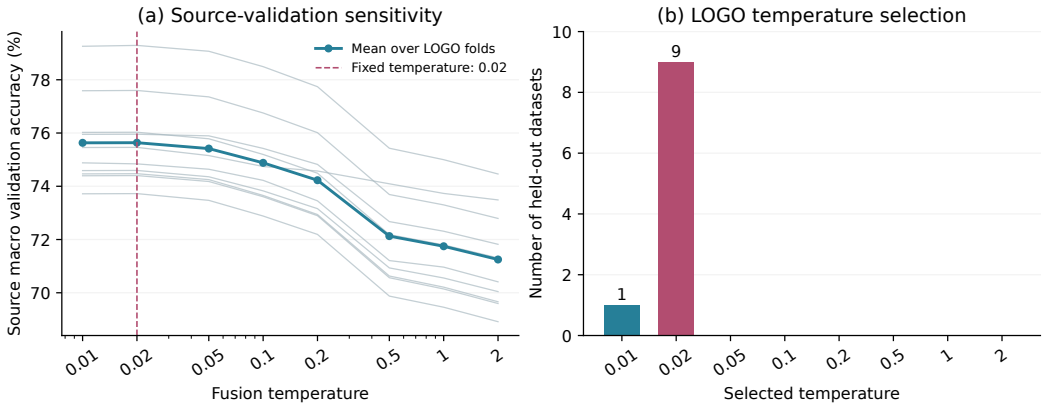

Figure C.2: Fusion-temperature analysis using leave-one-graph-out (LOGO) validation. (a) Thin curves show macro validation accuracy on the nine source datasets in each fold; the thick curve averages the ten overlapping folds. These curves do not report held-out test accuracy or confidence intervals. (b) Counts of temperatures selected using source validation scores: nine folds select 0.02 and one selects 0.01. The held-out dataset is excluded from temperature selection.

C.3 Robustness to Edge Perturbations

Perturbation construction. We evaluate random edge removal and addition at rates $r \in \{0.1, 0.2, 0.3\}$, together with a clean-graph reference. Let M be the number of unique undirected edges in the clean graph, excluding self-loops. Each condition changes $\lfloor rM \rfloor$ edges. For removal, we sample existing edges uniformly without replacement. For addition, we sample node pairs uniformly and reject self-pairs, existing edges, and previously accepted pairs until the required number of new edges is obtained. Perturbations are constructed from the clean graph at each rate, not applied successively across rates. The node set, features, and labels remain unchanged. No connectivity constraint is imposed, so edge removal may produce isolated nodes. Each retained or added undirected edge is represented in both directions, and self-loops are subsequently handled by the model's standard preprocessing. Node labels are used only to report homophily, not to select perturbed edges.

Paired evaluation and retraining. We compare DMSD, GCN, and SSD using matched data splits and perturbation settings. The ten seeds listed in Appendix B.2 are used for both split generation and edge perturbation, with the same seed assigned to the two operations in each run. DMSD's two candidates use the same perturbed graph and split, and are retrained for each condition with model seed 1024. We retain the dataset-specific settings in Table B.3, use 50 pretraining epochs and 500 linear-classifier epochs, and recompute fusion weights from validation accuracy with the fixed temperature $\tau_g = 0.02$. Test labels are not used for configuration or fusion-weight selection. This protocol evaluates learning on perturbed graphs, rather than test-time robustness of a fixed model or adversarial robustness. Figure C.3 shows the resulting accuracy curves.

Table C.1 reports per-dataset results under 30% edge removal and addition.

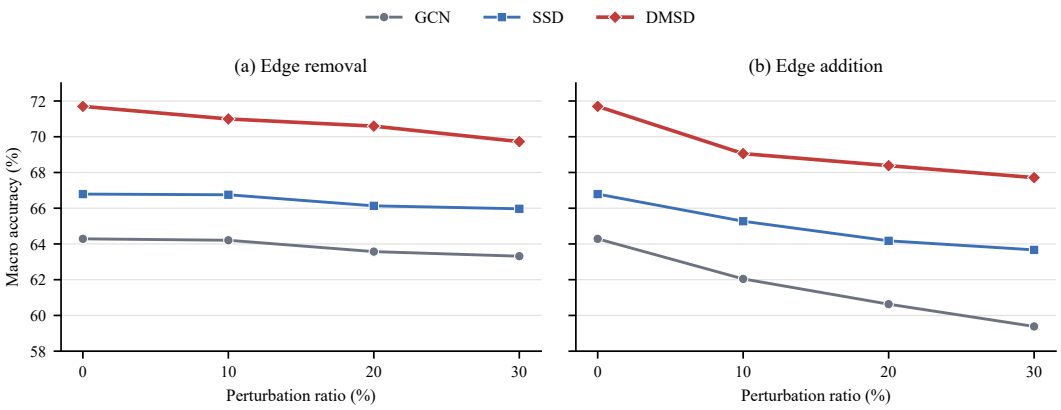

Figure C.3: Macro-averaged test accuracy across ten datasets under random edge removal and addition. Zero perturbation denotes the clean-graph condition in the robustness experiments.

Table C.1: Node classification accuracy (%) under 30% random edge perturbations. The best and second-best results are shown in bold and underlined, respectively.

Edge Removal (30%) Dataset	GCN	SSD	DMSD
Chameleon	**54.41±1.89**	53.62±1.43	54.33±2.00
Squirrel	36.61±0.92	37.87±0.81	**38.53±1.67**
Cornell	42.69±6.03	50.97±8.77	**55.66±7.57**
Texas	42.84±9.95	55.14±6.47	**65.47±3.22**
Wisconsin	48.66±5.23	55.32±5.10	**71.69±6.11**
Cora	77.49±0.66	**80.18±1.04**	79.19±1.12
Citeseer	66.69±2.06	**69.54±1.83**	68.80±1.72
Pubmed	83.54±0.68	84.21±0.25	**85.77±0.24**
Computers	**88.10±0.43**	82.31±0.96	85.52±0.68
Photo	92.13±0.43	90.54±0.31	**92.33±0.21**

Edge Addition (30%) Dataset	GCN	SSD	DMSD
Chameleon	47.48±1.31	**48.26±1.53**	47.19±1.93
Squirrel	31.46±1.02	34.67±1.01	**37.21±1.06**
Cornell	36.97±7.10	55.93±5.14	**58.97±4.29**
Texas	42.57±8.52	55.34±5.79	**62.84±6.61**
Wisconsin	42.29±11.89	55.47±5.51	**69.40±3.50**
Cora	75.07±1.17	77.81±1.13	**78.47±1.07**
Citeseer	63.43±2.05	**67.48±0.86**	64.43±1.84
Pubmed	80.28±1.16	80.42±0.23	**85.49±0.26**
Computers	**84.92±0.64**	76.22±0.60	82.51±0.84
Photo	89.36±0.67	85.08±0.86	**90.62±0.50**

Table C.2 reports macro-averaged accuracy for the clean graphs and all six perturbation conditions. The clean-graph reference is taken from the robustness runs, not from the main accuracy table. At 30% removal and addition, DMSD retains 96.97% and 94.21% accuracy on average, respectively. Retention is computed within each dataset relative to its clean-graph accuracy and then averaged across datasets; it is not the ratio of the two macro-averaged accuracies.

Table C.2: Macro-averaged test accuracy (%) for all edge-perturbation conditions.

Condition	GCN	SSD	DMSD
Clean	64.29	66.79	71.70
Removal 10%	64.21	66.75	71.00
Removal 20%	63.57	66.13	70.59
Removal 30%	63.31	65.97	69.73
Addition 10%	62.04	65.27	69.06
Addition 20%	60.63	64.18	68.38
Addition 30%	59.38	63.67	67.71

Paired statistical tests. Methods are paired by dataset, perturbation type, ratio, and seed. We first average the ten runs within each dataset and then use the ten datasets as paired statistical units. Friedman tests detect overall method differences at 30% removal and addition, with $p = 0.045$ and $p = 0.027$, respectively. For the four DMSD–baseline comparisons, we use two-sided exact Wilcoxon signed-rank tests with a joint Holm correction. The adjusted p-values for removal are 0.1113 against both GCN and SSD; for addition they are 0.0781 against GCN and 0.1113 against SSD. None of these pairwise comparisons meets the 0.05 threshold. Consequently, higher mean perturbed accuracy should not be interpreted as statistically significant pairwise superiority in these tests.

C.4 Accuracy–Efficiency Analysis

Figure C.4 visualizes the accuracy–time trade-offs reported in Table 2. The measurement scope is described in Appendix B.3.



Figure C.4: Accuracy–time trade-offs. Left: training time; right: inference latency. Time axes use logarithmic scales. Accuracy and timing follow the measurement sources described in Table 2.