DISTILLING INTERACTION SPACES FOR GRAPH TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Transformers (GTs), with sophisticated attention mechanisms, naturally support beneficial distant node interactions and have shown strong potential across graph applications. Yet, in semi-supervised node classification, their flexibility makes interaction-structure acquisition particularly important: a single interaction graph can over-constrain a GT, whereas unrestricted structure learning can be weakly constrained by scarce labels and co-adapt with attention. To bridge this gap, by a constrained bi-level optimization reformulation, we disentangle graph searching and attention learning to reduce detrimental structure–attention co-adaptation and improve generalizability. Furthermore, inspired by weak-to-strong distillation, we let a structurally simpler GNN teacher define a *task-informed interaction space* and allow a more expressive GT student to adaptively search for a suitable candidate within it. This way not only flexibly encodes *task-specific implicit constraints* on graphs tailored for downstream tasks, but also preserves the model adaptiveness of the GT beyond hand-crafted explicit heuristics. Technically, we propose a two-stage framework: 1) the **Disentangling Stage** extracts several diverse and task-specific prototypical graphs as a *graph basis* that spans a reasonable interaction space, with their task relevance enhanced by our Multi-Aspect Cooperative Distillation; 2) the **Fusing Stage** conducts adaptive search in this space via a simple node-wise linear fuser trained with our Meta Self-Update mechanism. Our framework fully mines the value of teacher knowledge and better releases students' potential. It empirically outperforms state-of-the-art counterparts while retaining generality and practical efficiency.

1 INTRODUCTION

Graph Transformers (GTs) extend graph representation learning by allowing attention to combine information beyond observed neighborhoods (Min et al., 2022; Wu et al., 2021; 2022). Recent scalable and sparse GTs have substantially reduced the cost of broader interactions (Wu et al., 2022; 2024), while studies of over-globalization and hierarchical interaction masks increasingly show that *which nodes are allowed to interact* can be as consequential as the attention function itself (Xing et al., 2024; 2025). This makes the structural side of attention particularly important for semi-supervised node classification, where observed edges need not contain all task-relevant relations but only a small fraction of nodes provide labels. We call the weighted structural prior on candidate interactions the *potential graph* $G^{(p)}$; attention then learns how strongly to use those candidates. Because $G^{(p)}$ determines the relations available before attention reweights them, its choice jointly controls the model's structural inductive bias, the effective interaction search range, and the number of candidate interactions that must be handled. *We therefore study a basic interaction-structure acquisition problem: how can an expressive GT obtain task-relevant interactions when supervision is scarce?*

Two common strategies expose a central tension in this problem. A fixed or carefully constructed graph provides a strong prior, but also makes an early structural commitment: attention cannot recover excluded relations even when they would better match later model representations, and a single global choice need not suit every node or layer. Our single-prototype comparison provides direct motivation for this concern (Tab. 9). At the other extreme, learning interactions with much greater freedom (Wu et al., 2022; Zhang et al., 2022) places the burden of structure discovery on the same few labels that also train attention, representations, and the classifier. Structure and attention can then co-adapt to labeled nodes without identifying relations that transfer well to unlabeled ones; our joint-learning

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.

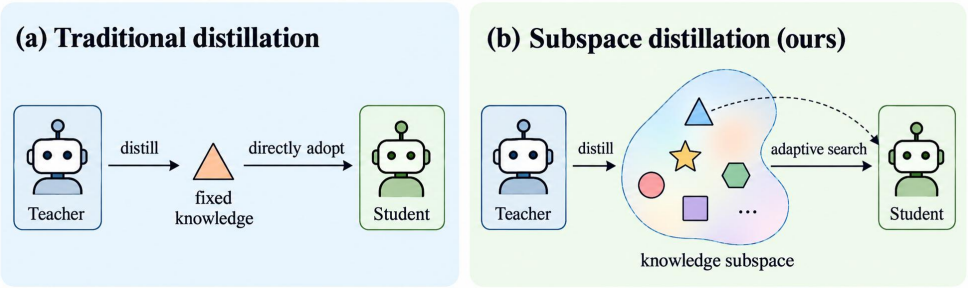


Figure 1: **From Fixed Knowledge Transfer to Subspace Distillation.** Traditional KD transfers fixed knowledge for a student to imitate or directly adopt. In contrast, subspace distillation transfers a constrained knowledge space, allowing a student to adaptively select a suitable realization. Thus, a teacher specifies a range of plausible solutions rather than one fixed answer. Our interaction-space distillation instantiates this principle for graph structures.

study illustrates this risk for the tested model (Fig. 15). *The challenge is therefore not merely to learn a "better graph", but to retain useful structural freedom without asking scarce labels to supervise an essentially unrestricted structural search.*

Our key insight is to elevate the modeling object from one interaction graph to a constrained interaction space. We represent this space by a graph basis of task-relevant prototypes whose admissible fusions span a parameterized family of weighted graphs. Compared with a singleton graph, this family leaves substantially more room for model adaptation; compared with unrestricted graph learning, it explicitly bounds the structural hypotheses that need to be considered. Moreover, node- and layer-dependent fusion allows different nodes and layers to occupy different positions in the same space, rather than forcing the whole model to select one discrete graph. This middle ground is especially relevant as recent GTs increasingly rely on multiple masks or routing patterns, while those candidate patterns are often specified by architectural heuristics rather than acquired from task knowledge (Xing et al., 2025). The remaining question is how to obtain such a task-informed space without returning to either hand-crafted rules or unconstrained structure learning.

Knowledge distillation (KD) provides a natural route. Traditional KD usually transfers a fixed prediction, representation, relation, or structural target for a student to imitate. We instead use a teacher to specify a constrained range of plausible structural knowledge and leave the final realization to a student (Fig. 1). Motivated by weak-to-strong generalization (Burns et al., 2023), a teacher need not be the stronger final predictor: its role is to provide a task-informed, structurally biased search region that a more expressive GT can further adapt within. We call this **Interaction-Space Distillation**, a structural instance of *subspace distillation*: a teacher constrains where learning should search rather than prescribing a unique solution. This view leads directly to two implementation questions: **(Q1)** *How can a teacher construct a task-informed interaction space?* and **(Q2)** *How can a student identify a suitable, student-adaptive interaction structure within that space?* Our **Disentangling Stage** and **Fusing Stage** address Q1 and Q2, respectively. Fig. 10 contrasts this teacher-derived interaction space with fixed or heuristic candidate sources. Fig. 11 visualizes why it occupies a middle ground between rigid fixed-graph learning and unconstrained structure search.

For Q1, we need structural anchors that carry richer task information than the scarce labels alone. We obtain them from a task-trained message-passing GNN teacher. Its local aggregation imposes a stronger structural bias and much smaller interaction freedom than the GT student, making its predictions a comparatively constrained source of task guidance. Stage 1 disentangles this guidance into multiple structural views. Because accurate holistic predictions do not guarantee that every aspect is task-relevant, our **Multi-Aspect Cooperative Distillation (MACD)** transfers holistic task knowledge back to the aspect heads. Prediction similarity and confidence then convert these task-aligned views into sparse prototypical graphs, which form a graph basis that spans the task-informed interaction space rather than one copied teacher graph. Additional teacher-choice results and practical selection guidance are discussed in Sec. L.9 of Appx.

For Q2, the student must exploit the remaining freedom inside that space. Stage 2 therefore learns a node-wise fuser conditioned on student representations, so different nodes and layers can combine the same prototypes differently before attention further refines their weights. However, restricting the candidate family alone does not determine how the student should search it: training the fuser and attention only with the same scarce-label objective can reintroduce structure–attention co-adaptation. We therefore separate student fitting from structural selection through a constrained bi-level formulation implemented by alternating optimization. Our **Meta Self-Update** mechanism

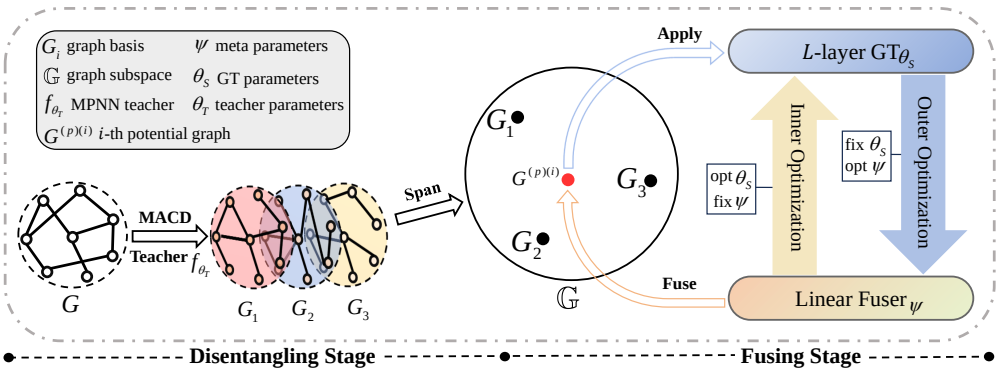


Figure 2: The comprehensive illustration of the proposed high-level 2-stage Interaction-Space Distillation framework. It contains two stages, *i.e.*, Stage 1 (Disentangling Stage) and Stage 2 (Fusing Stage). In Stage 1 (Sec. 2.2), we distill several prototypical graphs $\{G_i\}$ from a task-trained teacher (*e.g.*, a message-passing GNN), whose task relevance is enhanced by our MACD module (Sec. 2.2.1). These prototypical graphs form a graph basis that spans the task-informed interaction space $\mathbb{G}$ (labeled as a "graph subspace" in the legend only to echo the more general subspace-distillation view in Fig. 1; in the GT setting, this object is precisely the interaction space). In Stage 2 (Sec. 2.3), to search for a student-adaptive potential graph $G^{(p)(i)}$ (*e.g.*, shown via a red point in this figure) for the attention mechanism of the i-th layer of the student GT, we present a simple linear fuser to node-wisely, locally, and smoothly fuse those prototypical graphs $\{G_i\}$ into potential graphs $\{G^{(p)(i)}\}$ (Eq. 14 in Sec. 2.3.1), which can be trained via the proposed Meta Self-Update mechanism (Sec. 2.3.2). In the inner optimization, we fix the parameters of the linear fuser (denoted by ψ) and optimize the parameters of the student GT (denoted by θ_S); in the outer optimization, we fix θ_S and optimize ψ. We provide more technical details in Sec. 2.3 and Sec. B.6, as well as further discussion in Sec. L.

supplies the outer update with a label-efficient surrogate constructed from teacher predictions and selected historical student predictions, avoiding the need for a separate labeled meta-training set.

Together, these two stages form **Interaction-Space Distillation**, with several concrete methodological advantages. *First, task-informed constraints replace purely hand-crafted structural choices*: the search region is distilled from a teacher that has already absorbed supervision from the target task. *Second, the framework preserves student freedom instead of enforcing teacher imitation*: a weaker but structurally biased teacher can restrict implausible search directions without fixing the final graph or becoming the student's performance ceiling. *Third, the interaction space provides controlled flexibility*: a compact graph basis bounds the candidate relations, while node- and layer-wise fusion still spans many student-specific structures. *Finally, the framework separates space construction, structure selection, and attention weighting, so each can use suitable supervision and be improved largely independently*. Sparse prototypes keep aggregation support compact and remain compatible with scalable attention. In short, *the teacher defines the feasible structural region, the student chooses a structure within it, and attention decides how to use that structure*. This extends KD beyond fixed targets or model compression by transferring a useful interaction space—that is, a constrained structural hypothesis space—to a stronger learner.

Our main contributions are summarized as follows: **(1)** we formulate the interaction-space acquisition problem for label-scarce Graph Transformers and introduce **Interaction-Space Distillation**, a high-level KD paradigm that transfers a task-informed interaction space (i.e., a candidate structural family) rather than one fixed graph; **(2)** we develop a **disentangling–fusing framework**, where MACD builds a task-aligned graph basis that spans the interaction space, and node-wise fusion with Meta Self-Update enables student-adaptive search under scarce labels; and **(3)** extensive experiments on nine real-world node-classification benchmarks show that our framework outperforms many state-of-the-art GNN and GT counterparts, while comprehensive distillation comparisons and ablations further verify the benefits of the distilled interaction space and adaptive search.

2 METHOD

We first formalize the distinction between the interaction space, the potential graph, and the attention weights. We then develop the two stages in Fig. 2: constructing task-informed prototypes and learning student-adaptive structures through fusion and alternating optimization. Appx. Fig. 12 gives a compact visual summary of how these objects differ and interact. The appendix provides full preliminaries, related work, derivations, and algorithms.

Notation. Let $G = (V, E)$ be the observed graph with $n = |V|$ nodes, adjacency matrix $A \in \{0,1\}^{n\times n}$, and node features $X \in \mathbb{R}^{n\times f}$. Semi-supervised node classification observes labels $y_u \in [1, C]$ only for a small labeled set $V_L \subset V$ and predicts labels for the remaining nodes. We denote the layer-l node representations by $H^{(l)} \in \mathbb{R}^{n\times d}$ with $H^{(0)} = X$, and use $L(\cdot,\cdot)$ for cross-entropy. For a student layer, $G^{(p)[l]}$ denotes the weighted *potential graph* that specifies candidate interactions and their structural priors, $\{\alpha_{i,j}\}^{[l]}$ denotes attention coefficients, and $G^{(agg)[l]}$ denotes the resulting aggregation graph. $\mathrm{RowNorm}(\cdot)$ normalizes each row to sum to one. Superscripts (T) and (S) denote teacher and student quantities; full notation and GNN/GT preliminaries appear in Sec. I and Tabs. 10–11.

2.1 Motivation and Comprehensive Framework

Consider the following message-passing form for an L-layer graph model:

$$H^{(l+1)} = \sigma\left(G^{(agg)[l]} \cdot \mathrm{MLP}^{(l)}_{\theta_l}\left(H^{(l)}\right)\right), \quad \forall\, l \in [0, L), \tag{1}$$

Here, $G^{(agg)[l]}$ is the weighted aggregation graph at layer l, $\sigma(\cdot)$ is a nonlinear activation, and $\mathrm{MLP}^{(l)}_{\theta_l}$ transforms the node features. For example, GCN uses $G^{(agg)[l]} = \hat{A}$. For the attention models considered here, we write $G^{(agg)[l]} = \mathrm{RowNorm}(G^{(p)[l]} \odot \{\alpha_{i,j}\}^{[l]})$, where RowNorm normalizes each row. The potential graph $G^{(p)[l]}$ specifies the candidate relations and their prior weights; attention refines these weights. For a fixed or learnable potential graph, the training objective is

$$\mathrm{fix}/\min_{G^{(p)}} \min_{\{\alpha_{i,j}\},\,\theta} L_{TE}\left(\{\alpha_{i,j}\}, \theta;\, G^{(p)}\right) = \frac{1}{|V_L|}\sum_{i\in V_L} L\left(p_i, y_i\right), \tag{2}$$

where $p_i = \mathrm{Softmax}(\mathrm{MLP}^{(L)}_{\theta_L}(H^{(L)}_i))$ predicts y_i, $L(\cdot,\cdot)$ is cross-entropy, and θ contains the remaining student parameters. We omit layer indices for brevity. A global GT can use a complete potential graph, while a local GT uses the observed graph or an augmented version. Learned interactions and structural encodings provide further choices (Sec. L.3).

Our motivation concerns both the candidate family and its training signal. First, the comparisons in Tab. 9 show that adaptive combinations outperform the original graph, a random prototype, and the best individual prototype in the reported setting. Second, the tested global GT produces attention-induced graphs with relatively low homophily on citation networks, even with sparsity regularization (Fig. 3). Adding a learnable potential graph under the same training objective does not resolve the performance gap (Fig. 15). We refer to this observed risk of detrimental structure–attention co-adaptation as *Cooperative Collapse*; Appx. Fig. 16 further contrasts this risk with our constrained bi-level alternative. These experiments motivate our design.

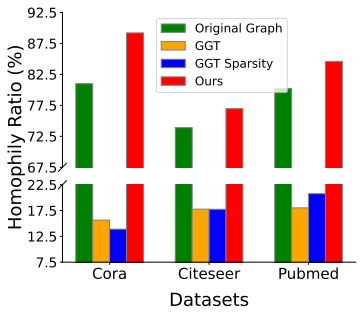


Figure 3: Homophily ratios on three citation networks: the original graph, attention-induced graphs from the tested global GT with and without sparsity regularization, and our learned graph. Higher homophily here indicates more label-consistent candidate relations.

We therefore introduce **(1)** a task-informed interaction space $\mathbb{G}$ that constrains candidate graphs and **(2)** a separate structure-selection objective. Ideally, a structure should be evaluated after fitting the student under it. This gives the following constrained bi-level formulation:

$$\begin{aligned} &\min_{G^{(p)}\in\mathbb{G}} \ \widehat{L_{GE}}\left(\{\alpha_{i,j}\}^*, \theta^*;\, G^{(p)}\right), \\ &\text{s.t.} \ \ \{\alpha_{i,j}\}^*, \theta^* = \arg\min_{\{\alpha_{i,j}\},\theta} L_{TE}\left(\{\alpha_{i,j}\}, \theta;\, G^{(p)}\right). \end{aligned} \tag{3}$$

Here $\widehat{L_{GE}}$ is a *generalization-oriented surrogate* used in alternating optimization (Sec. 2.3.2). This connects Q1 and Q2: Stage 1 constructs the task-informed family, while Stage 2 learns a student-specific graph within it. The approximation to bi-level optimization is discussed in Sec. G.

2.2 Stage 1 (Disentangling Stage): Interaction-Space Distillation

Stage 1 addresses Q1 by converting the teacher's task knowledge into K prototypical graphs $\{G_k\}_{k=1}^K$, each derived from an aspect $\mathcal{A}_k$. Borrowing the concept of a basis from linear algebra, we regard these

prototypes as a graph basis whose combinations *span* the interaction space used in Stage 2. Here, spanning denotes the generation of admissible graph candidates from the prototypes; the concrete coefficient constraints and fusion operators are specified in Sec. 2.3.1. The construction extracts teacher views, builds their prototype graphs, and then spans the interaction space as follows:

$$\mathbb{G} = \mathrm{Span}\left(G_1, G_2, \cdots, G_K\right), \tag{4}$$

$$G_k = \mathrm{Build}\left(R_k, S_k\right), \ \ \forall\, k \in [1, K], \tag{5}$$

$$\{(R_1, S_1), (R_2, S_2), \cdots, (R_K, S_K)\} = \mathrm{DRL}\left(f_{\theta_T}, A, X\right). \tag{6}$$

Here, DRL denotes disentangled pre-training, while Build maps an aspect representation $R_k \in \mathbb{R}^{n\times d/K}$ and auxiliary information S_k into a weighted graph. Span denotes the interaction space generated by the graph basis under the admissible fusion rule in Sec. 2.3.1; in our implementation, the coefficients are nonnegative and normalized, and candidate supports are inherited from the prototypes. Thus, Span keeps the basis/spanning analogy while its concrete realization is governed by our graph-fusion operator. The auxiliary information includes prediction confidence.

Disentangled pre-training. We implement DRL by extending the routing mechanism of DisenGCN (Ma et al., 2019) to a message-passing teacher f_{θ_T}. For each node u, probabilistic routing assigns neighboring messages to K aspects, yielding embeddings $\{c'_k\}_{k=1}^K$, with $R_{k,u} = c'_k$. Their concatenation forms the holistic embedding h_u used for prediction (Eq. 8). Full routing and message-passing details are given in Sec. B.1 and Algorithm 3; Sec. M discusses alternative teacher views.

2.2.1 MULTI-ASPECT COOPERATIVE DISTILLATION (MACD)

The prototype graphs should be informative about the task while retaining useful differences across aspects. Holistic supervision alone does not guarantee the first property: a classifier can perform well on the concatenated embedding h_u even when some individual aspects have poor predictive quality. We observe this gap in the vanilla teacher's Average Aspect Accuracy (Sec. N.1). The aspect heads $P_u^{(T)[k]} = \mathrm{Softmax}(\mathrm{MLP}(c'_k))$ therefore need additional task guidance. Direct losses $L(P_u^{(T)[k]}, y_u)$ are available only on labeled nodes; the holistic prediction instead supplies soft targets for all nodes. Figure 4 in Appx. gives the complete MACD flow from aspect disentanglement to prototypical-graph construction.

To exploit this information, **MACD** performs distillation *within the teacher*: the holistic prediction $P_u^{(T)}$ guides the K aspect predictions $\{P_u^{(T)[k]}\}_k$. These aspect heads are the students of this internal distillation step, not the final GT student. We minimize the following confidence-weighted Kullback–Leibler (KL) divergence:

$$L^{(Dis)} = \frac{1}{K \cdot S_{total}^{(T)}} \sum_{k=1}^{K} \sum_{u\in V} S_u^{(T)} \cdot \mathrm{KL}\left(\mathrm{Detach}\left(P_u^{(T)}\right) \| P_u^{(T)[k]}\right), \quad S_{total}^{(T)} = \sum_{u\in V} S_u^{(T)}, \tag{7}$$

where the holistic prediction $P_u^{(T)}$ (*i.e.*, the teacher) and the aspect predictions $P_u^{(T)[k]}$ (*i.e.*, the students) of node $u \in V$ are produced from its embedding h_u and its disentangled embeddings c'_k at aspects $\mathcal{A}_k$ ($k \in [1, K]$), respectively:

$$P_u^{(T)} = \mathrm{Softmax}\left(\mathrm{MLP}\left(h_u\right)\right), \quad P_u^{(T)[k]} = \mathrm{Softmax}\left(\mathrm{MLP}\left(c'_k\right)\right), \quad h_u = \Big\|_{k=1}^{K} c'_k \in \mathbb{R}^d. \tag{8}$$

The $\mathrm{Detach}(\cdot)$ operation stops gradients through the holistic target in this KL term, while the teacher remains trained by its supervised objective. Because teacher predictions can be unreliable, especially on ambiguous nodes, we discount their contribution using the Inverted Normalized Entropy (INE):

$$S_u^{(T)} = \mathrm{Confidence}\left(P_u^{(T)}\right) = \mathrm{INE}\left(P_u^{(T)}\right) = 1 - H\left(P_u^{(T)}\right) / \log C \in [0, 1], \tag{9}$$

where $H(P) = -\sum_{c=1}^{C} P_c \log P_c$ and $C > 1$ is the number of classes. The score is zero for a uniform prediction and approaches one for a confident prediction, so uncertain teacher targets contribute less.

Finally, with the supervised loss of the teacher $L_{Sup}^{(T)} = \frac{1}{|V_L|} \sum_{u\in V_L} L\left(P_u^{(T)}, y_u\right)$, we pre-train the teacher by minimizing

$$L_{Pre}^{(T)} = L_{Sup}^{(T)} + \alpha_{Dis} \cdot L^{(Dis)}, \tag{10}$$

with trade-off $\alpha_{Dis} \geq 0$. Optional diversity regularization and further MACD discussion are provided in Sec. B.2.

2.2.2 WEIGHTING AND GRAPH CONSTRUCTION

The next step converts each task-aligned view into a graph on which the student can aggregate information. We use prediction similarity as a proxy for task-related affinity, motivated by the role of label-consistent interactions in GT generalization (Li et al., 2024b). Homophily measures the proportion of same-label neighbor pairs, but the true labels of most nodes are unavailable here. We therefore compare $\{P_u^{(T)[k]}\}_u$ rather than the raw representations R_k: the classification head in Eq. 8 filters each aspect through the prediction task.

Teacher knowledge is also uneven across nodes. We refer to this as *knowledge imbalance*: uncertain predictions may yield less reliable neighborhoods than confident ones. To reduce their influence, the edge score combines predictive similarity with the confidences of both endpoints. This can favor a less similar but more confident neighbor. For each aspect, node u retains the K' highest-scoring other nodes and its self-connection:

$$W^{(k)}(u,v) = \left(S_u^{(T)[k]}\right)^a \cdot \left(S_v^{(T)[k]}\right)^a \cdot \exp\left\{-\tau' \cdot \mathrm{KL}\left(P_u^{(T)[k]} \middle\| P_v^{(T)[k]}\right)\right\} \in [0,1], \tag{11}$$

$$G_{k,u,v} = W^{(k)}(u,v) \cdot \mathbb{I}\left(v \in \{u\} \cup \underset{j}{\text{ArgTop-}K'}\left(\left\{W^{(k)}(u,j)\right\}_{j \in V \setminus u}\right)\right) \in [0,1], \tag{12}$$

where $\mathbb{I}(\cdot)$ is the indicator, $K', a, \tau' > 0$, and $G_{k,u,v} \geq 0$ is the weight of edge $v \to u$ in G_k.

Each prototype has at most $n(K'+1)$ directed entries. However, evaluating all pairwise scores still costs $\mathcal{O}(n^2 C)$ per aspect because the predictions have C components. To avoid explicitly forming the pairwise score matrix, Proposition B.1 rewrites top-K' selection as Maximum Inner Product Search (MIPS) between $[P_u^{(T)[k]} \| a]$ and $[\tau' \log P_v^{(T)[k]} \| \log S_v^{(T)[k]}]$. This permits accelerated retrieval with Faiss (Johnson et al., 2019). Sections B.3 and E give the conditions and complete derivation.

Algorithm 1 summarizes Stage 1; Appendix Algorithm 4 gives its complete training and graph-construction procedure.

Algorithm 1 Stage 1: Interaction-Space Distillation (simplified, Alg. 4 for full version)

Require: A, X, V_L; teacher f_{θ_T}; K aspects; top-K' neighbors.
Ensure: Prototypes $\{G_k\}_{k=1}^K$ and $\mathbb{G} = \mathrm{Span}(G_1, \ldots, G_K)$.
1: **for** each teacher pre-training epoch **do**
2: Run L-layer MP-DRL to obtain holistic embeddings $H^{(L)}$, aspect embeddings $\{c'_k\}$, and aspect messages $\{z_{k,u}\}$.
3: Compute holistic predictions/confidence $P_u^{(T)}, S_u^{(T)}$ and aspect predictions $P_u^{(T)[k]}$.
4: Compute $L_{Sup}^{(T)}$, MACD loss $L^{(Dis)}$ (Eq. 7), and optional diversity loss $L^{(Div)}$.
5: Update the teacher and prediction heads by minimizing $L_{Pre}^{(T)}$ (Eq. 10).
6: **end for**
7: **for** $k = 1, \ldots, K$ **do**
8: Compute aspect confidence $S_u^{(T)[k]}$ and pair scores $W^{(k)}(u,v)$ (Eq. 11).
9: Retrieve top-K' neighbors by the equivalent MIPS formulation and build G_k via Eq. 12.
10: **end for**
11: **return** $\{G_k\}_{k=1}^K$ and the spanned interaction space $\mathbb{G}$.

2.3 STAGE 2 (FUSING STAGE): STUDENT-ADAPTIVE GRAPH SEARCH

Stage 2 addresses Q2 by learning how the student should combine the structural alternatives from Stage 1. A node-wise fuser first parameterizes student-adaptive potential graphs. We then define its selection objective and update procedure, so that the learned combinations reflect both the student's representations and the available task supervision.

2.3.1 A NODE-WISE FUSER

A fixed prototype or graph-wise mixture applies the same choice of aspect importance throughout the graph. We instead allow each node to combine prototype neighborhoods according to its current

student representation, while sharing the fuser parameters across nodes. Abstractly, this gives

$$G^{(p)} = \mathrm{Fuse}_{\psi}\left(\{G_k\}_k\right), \ ^{1} \tag{13}$$

For a precise definition, let $\Delta^{K-1} = \{\beta \in \mathbb{R}_+^K : \sum_k \beta_k = 1\}$. The admissible neighborhood family is $\mathbb{G}_u = \{\mathcal{M}_{k=1}^K(\beta_k G_{k,u}) : \beta \in \Delta^{K-1}\}$, and the graph family is $\mathbb{G} = \{G : G_{u,:} \in \mathbb{G}_u \ \forall u\}$. The shared fuser realizes a parameterized subset of this family:

$$G_u^{(p)(l)} = \mathcal{M}_{k=1}^K\left(\beta_{u,k}^{(l)} \cdot G_{k,u}\right) \in \mathbb{R}^{1\times n}, \tag{14}$$

$$\boldsymbol{\beta}_u^{(l)} = \left\{\beta_{u,k}^{(l)}\right\}_k = \mathrm{Softmax}\left(\mathrm{MLP}_{\psi^{(l)}}^{(l)}\left(H_u^{(l)}\right)\right) \in \mathbb{R}^K, \tag{15}$$

Here, $\mathcal{M}$ is element-wise Sum or Max, and $\psi^{(l)}$ denotes the parameters of the layer-specific MLP. Sum gives convex row combinations, whereas Max is generally nonlinear. Both define continuous weighted candidates on the prototype union. Their geometry and boundary cases are discussed in Sec. L.7, while Fig. 5 in Appx. gives the full node-wise fusion illustration. Parameter sharing encourages similar student representations to use similar mixtures without requiring all nodes to share one mixture. We compare this design with graph-wise fusion and single prototypes in Sec. 3.

2.3.2 Meta Self-Update Mechanism

The fuser determines a potential graph, but learning its parameters requires a criterion for structural selection. Using only the same labeled-node loss as the student can favor combinations that cooperate with the current predictor on the training set. Motivated by Eq. 3, we instead update the fuser using a *generalization-oriented surrogate* $\widehat{L_{GE}}$. The practical difficulty is that labels are too limited to assume a large additional meta-training set. Teacher predictions offer supervision across the graph, yet their errors should not become permanent targets as the student improves.

Meta Self-Update therefore maintains a meta-training set $\mathcal{D}_M = (\mathcal{V}_M, \mathcal{Y}_M)$ with $\mathcal{V}_M = V$ and teacher-initialized targets $\mathcal{Y}_M = \{P_u^{(T)}\}_u$. A fixed monitor set $\mathcal{D}_{mt}$ contains high-confidence teacher-pseudo-labeled nodes. After an inner update, a new best agreement with these monitor labels triggers confidence-weighted ensembling of the student's predictions into $\mathcal{Y}_M$. This incorporates selected student states while retaining a fixed reference. Monitor agreement is a selection proxy, not ground-truth validation accuracy. Section B.5 gives the thresholds and ensembling weights, and Fig. 6 in the appendix visualizes the complete inner-loop, historical-success integration, and outer-loop update.

At an outer step, the current fitted student $\mathcal{S}^*$ is held fixed while the fuser minimizes

$$\widehat{L_{GE}}\left(\mathcal{S}^* = (\{\alpha_{i,j}\}^*, \theta^*);\ G^{(p)}\right) = \frac{1}{C'}\sum_{u\in\mathcal{V}_M}\left(S_u^{(\mathcal{Y}_M)}\right)^{\eta} \cdot L\left(P_u^{(\mathcal{S}^*)},\ \mathrm{SP}\left(\mathcal{Y}_{M,u}\right)\right). \tag{16}$$

Here $C' = \sum_{u\in\mathcal{V}_M}(S_u^{(\mathcal{Y}_M)})^{\eta}$ normalizes the confidence weights; the hyper-parameters $\eta, \xi \geq 0$ control the confidence of the supervision and ensembling, respectively. The score $S_u^{(\mathcal{Y}_M)}$ is the confidence of $\mathcal{Y}_{M,u}$ (Eq. 9), and $\mathrm{SP}(\cdot)$ sharpens a distribution; for simplicity, we use $\mathrm{Onehot}(\mathrm{Argmax}(\cdot))$. We alternate $T^{(in)}$ inner steps with $T^{(out)}$ outer steps for $T^{(r)}$ rounds. The outer update holds the fitted student fixed rather than differentiating through its inner training trajectory (Appendix Algorithm 5).

Inner optimization. With fuser parameters fixed, the student minimizes its labeled-node loss $L_{Sup}^{(\mathcal{S})} = |V_L|^{-1}\sum_{u\in V_L} L(P_u^{(\mathcal{S})}, y_u)$. Two optional refinements help it use the teacher-derived structure before modifying the attention weights. A *near-to-far curriculum* initially favors nearly uniform attention on candidate neighbors and then permits more selective weighting. A *confidence-aware implicit sparsity regularizer* encourages the unmasked attention scores to concentrate on teacher-supported neighbors, rather than relying only on a sparse aggregation mask. Appx. Fig. 9 provides a compact visual summary. The complete losses, curriculum schedule, confidence weights, and kernel computation appear in Secs. B.6 and K, together with their roles in inner fitting.

Algorithm 2 summarizes the alternating student/fuser updates; Appendix Algorithm 5 gives the full procedure with monitor construction, ensembling weights, student resets, and optional refinements.

[1]The prototype family is shared across student layers, while potential graphs and fuser parameters are layer-specific. We omit l where the distinction is not needed.

Table 1: The results of performance comparison (with the best bolded and the runner-ups underlined)

Method	Category	Cora	Citeseer	Pubmed	Actor	Cornell	Texas	Wisconsin	Arxiv	Flickr	Avg. Rank
MLP	Classic	$58.30_{\pm 0.11}$	$58.68_{\pm 0.04}$	$72.94_{\pm 0.05}$	$35.62_{\pm 0.18}$	$72.70_{\pm 1.99}$	$77.84_{\pm 1.14}$	$79.61_{\pm 1.01}$	$32.84_{\pm 0.29}$	$42.01_{\pm 0.47}$	9.56
GCN	GNN	$82.06_{\pm 0.69}$	$71.60_{\pm 0.32}$	$79.58_{\pm 0.26}$	$27.88_{\pm 0.07}$	$53.51_{\pm 1.21}$	$69.19_{\pm 1.48}$	$57.25_{\pm 1.25}$	$53.77_{\pm 0.34}$	$38.40_{\pm 2.23}$	11.44
GAT	GNN	$82.84_{\pm 0.67}$	$72.28_{\pm 0.72}$	$78.52_{\pm 0.32}$	$28.71_{\pm 0.25}$	$55.14_{\pm 0.48}$	$68.65_{\pm 1.48}$	$58.82_{\pm 2.40}$	$\underline{55.73}_{\pm 0.46}$	$40.32_{\pm 1.75}$	9.56
GraphSAGE	GNN	$81.40_{\pm 0.52}$	$71.68_{\pm 0.13}$	$78.50_{\pm 0.44}$	$36.24_{\pm 0.39}$	$63.78_{\pm 2.42}$	$75.14_{\pm 1.21}$	$76.08_{\pm 0.88}$	$51.42_{\pm 0.21}$	$41.42_{\pm 0.89}$	8.00
SuperGAT$_{SD}$	GNN	$82.70_{\pm 0.60}$	$72.50_{\pm 0.80}$	$\underline{81.30}_{\pm 0.50}$	$30.18_{\pm 0.25}$	$54.59_{\pm 1.21}$	$69.73_{\pm 1.21}$	$58.04_{\pm 1.07}$	$51.52_{\pm 0.37}$	$36.24_{\pm 4.15}$	10.11
APPNP	GNN	$\underline{84.10}_{\pm 0.43}$	$72.14_{\pm 0.22}$	$80.02_{\pm 0.19}$	$33.47_{\pm 0.38}$	$61.08_{\pm 1.48}$	$71.35_{\pm 1.48}$	$65.10_{\pm 0.88}$	$55.60_{\pm 0.12}$	$43.07_{\pm 0.85}$	7.67
GT	GT	$77.58_{\pm 0.22}$	$66.96_{\pm 0.36}$	$76.48_{\pm 0.13}$	$37.15_{\pm 0.40}$	$61.62_{\pm 2.26}$	$74.60_{\pm 1.48}$	$71.76_{\pm 1.75}$	OOM	OOM	11.86
SAN	GT	$77.60_{\pm 0.23}$	$68.64_{\pm 0.92}$	$76.62_{\pm 0.22}$	$37.79_{\pm 0.20}$	$63.24_{\pm 1.48}$	$75.14_{\pm 2.26}$	$77.25_{\pm 1.75}$	OOM	OOM	9.29
Graphormer	GT	$63.08_{\pm 0.27}$	$61.08_{\pm 0.04}$	OOM	OOM	$62.70_{\pm 2.26}$	$76.76_{\pm 1.48}$	$72.16_{\pm 0.88}$	OOM	OOM	11.20
ANS-GT	GT	$77.68_{\pm 0.81}$	$64.16_{\pm 1.16}$	$77.98_{\pm 1.38}$	$\underline{38.29}_{\pm 0.61}$	$\underline{74.92}_{\pm 1.86}$	$76.22_{\pm 2.26}$	$76.47_{\pm 1.96}$	$41.83_{\pm 0.62}$	$21.86_{\pm 1.93}$	9.00
Nodeformer	GT	$79.02_{\pm 0.57}$	$69.66_{\pm 0.13}$	$76.06_{\pm 0.98}$	$34.80_{\pm 0.48}$	$68.11_{\pm 1.21}$	$77.84_{\pm 1.71}$	$76.47_{\pm 4.80}$	$39.47_{\pm 1.08}$	$40.31_{\pm 4.42}$	9.11
NAGphormer	GT	$79.51_{\pm 0.90}$	$67.34_{\pm 0.80}$	$78.32_{\pm 0.40}$	$37.33_{\pm 0.20}$	$63.78_{\pm 1.48}$	$71.89_{\pm 1.48}$	$66.27_{\pm 0.88}$	$52.00_{\pm 0.29}$	$38.59_{\pm 1.16}$	9.67
GOAT	GT	$83.18_{\pm 1.27}$	$71.99_{\pm 1.26}$	$79.13_{\pm 0.38}$	$37.66_{\pm 0.51}$	$64.32_{\pm 2.02}$	$76.76_{\pm 2.76}$	$73.33_{\pm 1.57}$	$52.46_{\pm 1.12}$	$35.53_{\pm 3.13}$	6.56
Exphormer	GT	$82.77_{\pm 1.38}$	$71.63_{\pm 1.19}$	$79.46_{\pm 0.35}$	$35.53_{\pm 0.62}$	$62.16_{\pm 2.42}$	$75.68_{\pm 1.71}$	$70.98_{\pm 2.29}$	$41.12_{\pm 0.69}$	$22.79_{\pm 3.09}$	9.56
SGFormer	GT	$82.38_{\pm 0.70}$	$71.82_{\pm 0.18}$	$80.64_{\pm 0.52}$	$37.80_{\pm 0.47}$	$68.65_{\pm 2.42}$	$\underline{78.92}_{\pm 1.21}$	$\underline{80.00}_{\pm 0.88}$	$45.73_{\pm 1.61}$	$40.13_{\pm 2.49}$	$\underline{5.22}$
DIFFormer	GT	$83.32_{\pm 0.52}$	$\mathbf{74.46}_{\pm 0.42}$	$78.16_{\pm 0.32}$	$34.51_{\pm 0.76}$	$60.00_{\pm 1.21}$	$68.11_{\pm 2.26}$	$63.92_{\pm 1.07}$	$53.60_{\pm 0.64}$	$\mathbf{44.25}_{\pm 0.86}$	8.67
Ours	GT	$\mathbf{84.88}_{\pm 0.38}$	$\underline{74.05}_{\pm 0.35}$	$\mathbf{82.58}_{\pm 0.18}$	$\mathbf{38.74}_{\pm 0.20}$	$\mathbf{81.08}_{\pm 2.42}$	$\mathbf{83.24}_{\pm 2.26}$	$\mathbf{84.71}_{\pm 1.47}$	$\mathbf{57.28}_{\pm 0.23}$	$\underline{43.81}_{\pm 1.41}$	**1.22**

Table 2: The results of distillation comparison

No.	Model	Cora	Citeseer	Pubmed	Actor	Cornell	Texas	Wisconsin	ArXiv	Flickr
(A)	*Vanilla Student* $\mathcal{S}$	75.40	67.90	76.10	36.57	67.56	75.67	78.43	46.74	28.24
(B)	*Vanilla Teacher* $\mathcal{T}$	81.50	62.10	77.80	37.23	72.97	75.67	76.47	53.38	33.19
(C)	*Vanilla Teacher* $\mathcal{T}$ *(AAA)*	13.37	16.23	32.83	19.97	17.56	9.40	20.09	6.84	25.03
(D)	*Improved Teacher* $\mathcal{T}^+$	82.20	<u>72.80</u>	81.40	37.17	70.27	75.67	<u>80.39</u>	54.68	42.30
(E)	*Improved Teacher* $\mathcal{T}^+$ *(AAA)*	<u>82.30</u>	72.12	<u>81.48</u>	36.11	74.32	<u>81.08</u>	79.41	55.02	42.30
(F)	*Vanilla Teacher* $\mathcal{T} \rightarrow$ *Student* $\mathcal{S}$ *(Single)*	81.38	67.76	78.30	<u>37.61</u>	<u>77.30</u>	75.14	77.65	56.12	38.56
(G)	*Vanilla Teacher* $\mathcal{T} \rightarrow$ *Student* $\mathcal{S}$ *(Space)*	53.70	51.50	69.40	34.54	70.27	72.97	78.43	55.98	<u>43.28</u>
(H)	*Improved Teacher* $\mathcal{T}^+ \rightarrow$ *Student* $\mathcal{S}$ *(Single)*	82.06	71.96	80.26	36.13	74.05	80.00	79.61	<u>56.72</u>	40.32
(I)	*Improved Teacher* $\mathcal{T}^+ \rightarrow$ *Student* $\mathcal{S}$ *(Space)*	**84.88**	**74.05**	**82.58**	**38.74**	**81.08**	**83.24**	**84.71**	**57.28**	**43.81**

Table 3: The results of ablation studies

No.	Method	Cora	Citeseer	Pubmed	Actor	Cornell	Texas	Wisconsin	ArXiv	Flickr
(1)	*w.o.* $L^{(Dis)}$	53.70	51.50	69.40	34.54	70.27	72.97	78.43	53.84	39.36
(2)	node-wise fuser $\Rightarrow$ graph-wise fuser	<u>84.04</u>	<u>73.38</u>	81.50	37.04	<u>80.00</u>	78.92	77.25	56.56	42.21
(3)	$\widehat{L_{GE}} \Rightarrow$ training loss	83.96	73.26	81.80	<u>37.98</u>	78.92	78.38	<u>81.18</u>	<u>56.86</u>	<u>42.27</u>
(4)	$\widehat{L_{GE}} \Rightarrow$ teacher's predictions	83.82	73.36	81.76	37.33	78.92	78.38	<u>81.18</u>	56.75	42.13
(5)	graph search $\Rightarrow$ random graph selection	82.00	72.20	<u>81.90</u>	35.59	70.27	<u>81.08</u>	78.43	54.48	40.84
(6)	graph search $\Rightarrow$ original graph	77.60	68.32	76.60	36.25	70.27	76.76	80.39	54.38	29.46
$(\cdot)$	Ours	**84.88**	**74.05**	**82.58**	**38.74**	**81.08**	**83.24**	**84.71**	**57.28**	**43.81**

Algorithm 2 Stage 2: Student-Adaptive Graph Search (simplified, Alg. 5 for full version)

Require: A, X, V_L; $\{G_k\}$; teacher predictions/confidence; student $\mathcal{S}$.
Ensure: Trained $\mathcal{S}$ and potential graphs $\{G^{(p)(l)}\}$.

Initialize meta targets $\mathcal{D}_M = (V, \{P_u^{(T)}\})$, high-confidence teacher monitor set $\mathcal{D}_{mt}$, and threshold γ.
Initialize node-wise fusers $\{\text{Fuse}_\psi^{(l)}\}_{l=1}^{L}$.
for $r = 1, \ldots, T^{(r)}$ **do**
  Reinitialize/reset $\mathcal{S}$ and construct $G_u^{(p)(l)}$ by Eqs. 14–15.
  for $t = 1, \ldots, T^{(in)}$ **do** ▷ inner: fit the student
    Predict $\{P_u^{(S)}\}$ and set λ_{Cur} by the cosine schedule.
    Compute $L_{TE}^{(S)} = L_{Sup}^{(S)} + \lambda_{Cur} L^{(Cur)} + \lambda_{Reg} L^{(Reg)}$ and update $\mathcal{S}$.
    if $\text{Acc}(\mathcal{S}, \mathcal{D}_{mt}) > \gamma$ **then**
      Update γ and $\mathcal{Y}_M \leftarrow (\mathbf{1} - \boldsymbol{\lambda}) \odot \mathcal{Y}_M + \boldsymbol{\lambda} \odot \{P_u^{(S)}\}$, with confidence weights in Eq. 24.
    end if
  end for
  for $t = 1, \ldots, T^{(out)}$ **do** ▷ outer: select the structure
    Hold the fitted student fixed; compute $\widehat{L_{GE}}$ (Eq. 16) and update the fusers ψ.
  end for
end for
return $\mathcal{S}$ and $\{G^{(p)(l)}\}$.

3 Experiments

We evaluate predictive performance together with Q1/Q2. Beyond GNN/GT comparisons, distillation studies test Stage 1 task-informed prototypes, while ablations test Stage 2 student adaptation and its training objective. The appendix provides full comparisons, implementation and hyperparameter details, complexity analysis, and visualizations.

Benchmarks. We use nine benchmarks: citation graphs Cora, Citeseer, Pubmed (Sen et al., 2008) and large-scale OGBN-ArXiv (Hu et al., 2020a); image-relation graph Flickr (Zeng et al., 2020); webpage graphs Texas, Wisconsin, and Cornell (Pei et al., 2020); and the Actor co-occurrence

Table 4: Running time comparison on two large-scale real-world graph benchmarks (sec/epoch). More comparisons can be found in the full table, *i.e.*, Tab. 17.

Dataset	$\text{SuperGAT}_{\text{SD}}$	APPNP	GT	SAN	Graphormer	ANS-GT	Nodeformer	NAGphormer	GOAT	Exphormer	SGFormer	DIFFormer	Ours
ArXiv	2.31	0.04	OOM	OOM	OOM	24.54	1.36	1.56	58.77	1.95	0.11	0.54	0.46
Flickr	0.75	0.02	OOM	OOM	OOM	7.80	0.84	0.83	28.28	1.11	0.05	0.30	0.43

network (Tang et al., 2009). Table 14 lists public splits; ArXiv/Flickr use custom low-label splits with 20 training nodes per class. All methods use the same splits. Since labeled fractions differ, these ArXiv/Flickr results are not directly comparable to standard-split leaderboard results.

Baselines. The sixteen baselines are: **(1)** ***classic*** MLP; **(2)** ***five GNNs***: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017), $\text{SuperGAT}_{\text{SD}}$ (Kim & Oh, 2021), and APPNP (Gasteiger et al., 2019); and **(3)** ***ten Graph Transformers***: GT (Dwivedi & Bresson, 2020), SAN (Kreuzer et al., 2021), Graphormer (Ying et al., 2021), ANS-GT (Zhang et al., 2022), NodeFormer (Wu et al., 2022), GOAT (Kong et al., 2023), NAGphormer (Chen et al., 2022b), Exphormer (Shirzad et al., 2023), SGFormer (Wu et al., 2024), and DIFFormer (Wu et al., 2023).

Comparative experiments. Across ten initializations, Tab. 1 reports means with standard errors. Ours leads seven of nine benchmarks with average rank 1.22; DIFFormer is slightly better on Citeseer/Flickr. Relative gains over GT, NodeFormer, and GCN are 13.4%, 13.8%, and 21.1%, with Cornell gains of 31.6%, 19.0%, and 51.5%. The comparison spans homophilous and heterophilous graphs; additional GT results, including cases favoring CoBFormer, are in Sec. C.4.

Distillation comparison. Table 2 separates teacher-knowledge quality from transfer and tests a task-informed family against one holistic graph while allowing the student to surpass its teacher. It compares: **(A)** naive LGT *Vanilla Student* $\mathcal{S}$; **(B)-(C)** pre-trained vanilla DRL *Teacher* $\mathcal{T}$ (*e.g.*, DisenGCN), via holistic accuracy and *AAA* (Average Aspect Accuracy) of $\{P^{(\mathcal{T})[k]}\}_k$; **(D)-(E)** the same metrics for MACD-improved $\mathcal{T}^+$ (Sec. 2.2); **(F)-(G)** $\mathcal{T} \to \mathcal{S}$ with a single holistic graph versus interaction-space distillation (*Single/Space*); and **(H)-(I)** the corresponding $\mathcal{T}^+$ transfers (I is *Ours*).

Table 2 shows five effects. **(1) A vs. I**: interaction-space distillation improves the vanilla student on every benchmark. **(2) D vs. I**: the final student exceeds the improved teacher on all nine datasets, going beyond prediction-level imitation. **(3) BC vs. DE**: MACD substantially improves aspect predictions and often the holistic teacher. **(4) F vs. G vs. I**: multiple vanilla-teacher graphs alone are insufficient; G underperforms single-graph transfer F on several datasets, while the task-aligned family in I improves performance. **(5) H vs. I**: with the improved teacher fixed, adaptive fusion consistently outperforms its single holistic graph. Sec. C further shows that adaptive fusion exceeds the best individual prototype (Tab. 9).

Ablation studies. Table 3 drops part X ("*w.o.* X") or replaces it by Y ("$X \Rightarrow Y$"). We **(1)** drop MACD $L^{(Dis)}$ in Eq. 10; **(2)** replace node-wise $\text{Fuse}_{\psi}(\cdot)$ with a graph-wise fuser; **(3)-(4)** replace $\widehat{L_{GE}}$ in Eq. 3 by the training loss or filtered dark knowledge (*i.e.*, teacher predictions); and **(5)-(6)** replace graph search by random prototype selection or the input graph, testing whether search surpasses special single graphs. Remaining components (*e.g.*, confidence weighting) are ablated in Tab. 5.

Table 3 supports both stages: removing MACD weakens prototypes, graph-wise fusion reduces student adaptation, and replacing $\widehat{L_{GE}}$ by the training loss or fixed teacher targets hurts performance.

Efficiency and quality of learned graphs. Sparse candidates avoid dense all-pair attention during student aggregation. Per-epoch costs are 0.46s/0.43s on ArXiv/Flickr (Tab. 4), where GT, SAN, and Graphormer run out of memory. These timings exclude teacher training, graph construction, and all alternating rounds; Sec. H separates these costs, while Sec. C.6 reports multi-prototype memory use.

Figure 3 compares homophily for our learned, original, and tested global-GT-induced graphs. Motivated by Li et al. (2024a), this label-related diagnostic is higher for ours on all three citation networks, consistent with our predictive-similarity construction.

4 Conclusion and Limitation

We study task-informed interaction structures for GTs without fixing one teacher graph. Interaction-Space Distillation transfers a constrained family of sparse prototypes and lets the student learn node-/layer-specific combinations. MACD improves prototype relevance, while Meta Self-Update supplies a separate structure-selection surrogate. Our framework has two main limitations. First, its search range depends on the coverage of the teacher-derived prototypes, and Stage 2 cannot add edges beyond their union; Sec. J discusses a possible extension. Second, storing multiple prototype graphs introduces additional memory cost, although graph sampling may further reduce this overhead.

AI USE STATEMENT

AI assistance was used to revise the exposition and organization, edit English and LaTeX, and identify consistency issues. It was not used to run the reported experiments. The authors are responsible for verifying the final manuscript and all experimental claims.

REPRODUCIBILITY STATEMENT

The source code is available at `https://anonymous.4open.science/r/dgt-B862/`.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Arnaud Breloy, Sandeep Kumar, Ying Sun, and Daniel P Palomar. Majorization-minimization on the stiefel manifold with application to robust sparse pca. *IEEE Transactions on Signal Processing*, 69:1507–1520, 2021.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023. URL `https://arxiv.org/abs/2312.09390`.

Antoine Caubrière, Natalia Tomashenko, Antoine Laurent, Emmanuel Morin, Nathalie Camelin, and Yannick Estève. Curriculum-based transfer learning for an effective end-to-end spoken language understanding and domain portability. In *20th Annual Conference of the International Speech Communication Association (InterSpeech)*, pp. 1198–1202, 2019.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pp. 3469–3489. PMLR, 2022a.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, Vancouver, BC, Canada, 2018a. OpenReview.net. URL `https://openreview.net/forum?id=rytstxWAW`.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *The Eleventh International Conference on Learning Representations*, 2022b.

Jinsong Chen, Hanpeng Liu, John E. Hopcroft, and Kun He. Leveraging contrastive learning for enhanced node representations in tokenized graph transformers. In *Advances in Neural Information Processing Systems*, 2024.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735, 2020.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018b.

Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. DOI:10.1109/TPAMI.2022.3174515, 2022c.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui 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*, pp. 257–266, 2019.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Guanyi Chu, Xiao Wang, Chuan Shi, and Xunqiang Jiang. Cuco: Graph representation with curriculum contrastive learning. In *IJCAI*, pp. 2300–2306, 2021.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *International Conference on Learning Representations*, 2024.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pp. 1309–1318. PMLR, 2018.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

George H Dunteman. *Principal components analysis*, volume 69. Sage, 1989.

Emilien Dupont. Learning disentangled joint continuous and discrete representations. *Advances in neural information processing systems*, 31, 2018.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

Floris Geerts and Juan L Reutter. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2022.

Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. Eigengame: Pca as a nash equilibrium. In *International Conference on Learning Representations*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272, 2017.

Olga Golovneva, Tianlu Wang, Jason Weston, and Sainbayar Sukhbaatar. Contextual position encoding: Learning to count what's important. *arXiv preprint arXiv:2405.18719*, 2024.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

Lingbing Guo, Qiang Zhang, and Huajun Chen. Unleashing the power of transformer for graphs. *arXiv preprint arXiv:2202.10581*, 2022.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Dong-Jun Han, Jungwuk Park, Seokil Ham, Namjin Lee, and Jaekyun Moon. Improving low-latency predictions in multi-exit neural networks via block-dependent losses. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–9, 2023. doi: 10.1109/TNNLS.2023.3282249.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pp. 466–483. Springer, 2020.

Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

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

Haruo Hosoya. Group-based learning of disentangled representations with generalizability for novel contents. *arXiv preprint arXiv:1809.02383*, 2018.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pp. 2704–2710, 2020b.

Yisong Huang, Jin Li, Xinlong Chen, and Yang-Geng Fu. Training graph transformers via curriculum-enhanced attention distillation. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/480c36b153dc18bdf455b2ed43db5bae-Abstract-Conference.html.

Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 655–665, 2022.

Aref Jafari, Mehdi Rezagholizadeh, Pranav Sharma, and Ali Ghodsi. Annealing knowledge distillation. *arXiv preprint arXiv:2104.07163*, 2021.

Insu Jeon, Wonkwang Lee, Myeongjang Pyeon, and Gunhee Kim. Ib-gan: Disentangled representation learning with information bottleneck generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7926–7934, 2021.

Yinghui Jiang, Shuting Jin, Xurui Jin, Xianglu Xiao, Wenfan Wu, Xiangrong Liu, Qiang Zhang, Xiangxiang Zeng, Guang Yang, and Zhangming Niu. Pharmacophoric-constrained heterogeneous graph transformer model for molecular property prediction. *Communications Chemistry*, 6(1):60, 2023.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Chaitanya K Joshi, Fayao Liu, Xu Xun, Jie Lin, and Chuan Sheng Foo. On representation knowledge distillation for graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 1:2, 2019.

Ferath Kherif and Adeliya Latypova. Principal component analysis. In *Machine learning*, pp. 209–225. Elsevier, 2020.

Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International conference on machine learning*, pp. 2649–2658. PMLR, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C Bayan Bruss, and Tom Goldstein. Goat: A global transformer on large-scale graphs. In *International Conference on Machine Learning*, pp. 17375–17390. PMLR, 2023.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Hongkang Li, Meng Wang, Tengfei Ma, Sijia Liu, ZAIXI ZHANG, and Pin-Yu Chen. What improves the generalization of graph transformer? a theoretical dive into self-attention and positional encoding, 2024a. URL https://openreview.net/forum?id=aJl5aK9n7e.

Hongkang Li, Meng Wang, Tengfei Ma, Sijia Liu, ZAIXI ZHANG, and Pin-Yu Chen. What improves the generalization of graph transformer? a theoretical dive into self-attention and positional encoding, 2024b. URL https://openreview.net/forum?id=aJl5aK9n7e.

Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. Dynamic knowledge distillation for pre-trained language models. *arXiv preprint arXiv:2109.11295*, 2021.

Zheng Li, Xiang Li, Lingfeng Yang, Borui Zhao, Renjie Song, Lei Luo, Jun Li, and Jian Yang. Curriculum temperature for knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 1504–1512, 2023.

Valerii Likhosherstov, Krzysztof M Choromanski, Kumar Avinava Dubey, Frederick Liu, Tamas Sarlos, and Adrian Weller. Chefs' random tables: Non-trigonometric random features. *Advances in Neural Information Processing Systems*, 35:34559–34573, 2022.

Valerii Likhosherstov, Krzysztof M Choromanski, Kumar Avinava Dubey, Frederick Liu, Tamas Sarlos, and Adrian Weller. Dense-exponential random features: Sharp positive estimators of the gaussian kernel. *Advances in Neural Information Processing Systems*, 36, 2024.

Jing Liu, Tongya Zheng, and Qinfen Hao. Hire: Distilling high-order relational knowledge from heterogeneous graph neural networks. *Neurocomputing*, 507:67–83, 2022.

Xuebo Liu, Houtim Lai, Derek F Wong, and Lidia S Chao. Norm-based curriculum learning for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 427–436, 2020a.

Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. Independence promoted graph disentangled networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4916–4923, 2020b. doi: 10.1609/aaai.v34i04.5929.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *International conference on machine learning*, pp. 4212–4221. PMLR, 2019.

Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. A survey of product quantization. *ITE Transactions on Media Technology and Applications*, 6(1):2–10, 2018.

H. Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization, 2010.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020. URL http://jmlr.org/papers/v21/20-212.html.

Hoang D Nguyen, Xuan-Son Vu, and Duc-Trong Le. Modular graph transformer networks for multi-label image classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 9092–9100, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style,high-performance deep learning library. *Advances in neural information processing systems*, 32: 8024–8035, 2019.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Duy Nhat Phan, Nicolas Gillis, et al. An inertial block majorization minimization framework for nonsmooth nonconvex optimization. *Journal of Machine Learning Research*, 24(18):1–41, 2023.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

Michael B Richman. Rotation of principal components. *Journal of climatology*, 6(3):293–335, 1986.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571, 2020.

Mrinmaya Sachan and Eric Xing. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 453–463, 2016.

Ruslan Salakhutdinov and Sam T Roweis. Adaptive overrelaxed bound optimization methods. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 664–671, 2003.

Ruslan R Salakhutdinov, Sam T Roweis, and Zoubin Ghahramani. On the convergence of bound optimization algorithms. *arXiv preprint arXiv:1212.2490*, 2012.

Ryoma Sato. A survey on the expressive power of graph neural networks, 2020.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pp. 31613–31632. PMLR, 2023.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

Chengliang Tang, Nathan Lenssen, Ying Wei, and Tian Zheng. Wasserstein distributional learning via majorization-minimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 10703–10731. PMLR, 2023.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

Ajinkya Tejankar, Soroush Abbasi Koohpayegani, Vipin Pillai, Paolo Favaro, and Hamed Pirsiavash. Isd: Self-supervised learning by iterative similarity distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9609–9618, 2021.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Neng Wan, Dapeng Li, and Naira Hovakimyan. F-divergence variational inference. *Advances in neural information processing systems*, 33:17370–17379, 2020.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. Curriculum pre-training for end-to-end speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3728–3738, 2020a.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Shuhai Wang, Xin Liu, Xiao Pan, Hanjie Xu, and Mingrui Liu. Heterogeneous graph transformer for meta-structure learning with application in text classification. *ACM Trans. Web*, 17(3), may 2023. ISSN 1559-1131. doi: 10.1145/3580508. URL https://doi.org/10.1145/3580508.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33, 2020b.

Xin Wang, Hong Chen, Siao Tang, Zihao Wu, and Wenwu Zhu. Disentangled representation learning. *ArXiv*, abs/2211.11695, 2022a. URL https://api.semanticscholar.org/CorpusID:253735310.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(9):4555–4576, 2022b.

Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Curgraph: Curriculum learning for graph classification. In *Proceedings of the Web Conference 2021*, pp. 1238–1248, 2021.

Yukang Wei and Yu Bai. Dynamic temperature knowledge distillation. *arXiv preprint arXiv:2404.12711*, 2024.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020a.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871, 2019.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. Difformer: Scalable (graph) transformers induced by energy constrained diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems*, 36, 2024.

Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020b.

Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

Yu Xie, Shengze Lv, Yuhua Qian, Chao Wen, and Jiye Liang. Active and semi-supervised graph neural networks for graph classification. *IEEE Transactions on Big Data*, 8(4):920–932, 2022. doi: 10.1109/TBDATA.2021.3140205.

Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. Less is more: on the over-globalizing problem in graph transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54656–54672, 2024.

Yujie Xing, Xiao Wang, Bin Wu, Hai Huang, and Chuan Shi. Unifying and enhancing graph transformers via a hierarchical mask framework. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL `https://proceedings.neurips.cc/paper_files/paper/2025/hash/c8fa61687c7a85d392a0503b456070f1-Abstract-Conference.html`.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6095–6104, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7074–7083, 2020.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Adamo Young, Hannes Röst, and Bo Wang. Tandem mass spectrum prediction for small molecules using graph transformers. *Nature Machine Intelligence*, pp. 1–13, 2024.

Jin Yuan, Shikai Chen, Yao Zhang, Zhongchao Shi, Xin Geng, Jianping Fan, and Yong Rui. Graph attention transformer network for multi-label image classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 19(4), feb 2023. ISSN 1551-6857. doi: 10.1145/3578518. URL https://doi.org/10.1145/3578518.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems*, 35:21171–21183, 2022.

Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*, 2021.

Jiawei Zhou, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, and Lei Chen. Retrieval-based disentangled representation learning with natural language supervision. In *The Twelfth International Conference on Learning Representations*, 2023.

Tianyi Zhou, Shengjie Wang, and Jeffrey Bilmes. Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, 33:8602–8613, 2020.

Jiaming Zhuo, Yuwei Liu, Yintong Lu, Ziyi Ma, Kun Fu, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. DUALFormer: Dual graph transformer. In *International Conference on Learning Representations*, 2025.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A THE ORGANIZATION OF THE TECHNICAL APPENDIX

The main content of the technical appendix can be organized as follows:

- Sec. B and Sec. C provide the method details and experimental results omitted from the main text;
- We provide a table containing important notations and their explanation in Sec. D;
- In Sec. E, we give all the proofs omitted in the main text;
- Sec. F provides many details relating to the specific implementation of our framework;
- Sec. G provides all the important algorithms presented in the main text;
- Sec. H points out the overall time and space complexities of the proposed framework;
- In Sec. I, we provide the preliminaries moved from the main text together with related work on GNNs, Graph Transformers, distillation, disentangled representation learning, and meta-learning;
- Sec. J provides a possible implementation of Stage 3, where we first give a high-level abstract knowledge distillation pipeline and provide a specific implementation, which deserves further investigation;
- In Sec. K, we introduce how to implement a kernel-decomposition-based regularization;
- Sec. L provides discussion on the interpretation and limitations of the presented method, including the interaction-space trade-off and teacher selection;
- Sec. M presents some alternative ideas for the implementation of Stage 1 regarding, *e.g.*, how to obtain several prototypical graphs based on other methods, in addition to Disentangled Representation Learning;
- Motivation Experiments (Sec. N) provide several experiments to support our motivation in Sec. 2.1 of the main text;
- Sec. O provides all the tables omitted or incomplete in the main text;
- Sec. P gives us more details of the experiments, including experimental environments (Sec. P.1), dataset descriptions (Sec. P.2), and hyper-parameters (Sec. P.3);
- Sec. Q gives an efficiency analysis for real-world benchmarks;
- Sec. R provides some additional experiments to further evaluate the proposed framework;
- Sec. S presents many visualizations to facilitate better intuitive understandings.

B METHOD DETAILS OMITTED FROM THE MAIN TEXT

This section provides the complete technical details underlying the two stages in Sec. 2. We first derive the message-passing views and the full MACD objective used to construct task-informed prototypes. We then describe the MIPS reduction, the monitor and target updates in Meta Self-Update, and both auxiliary losses for student training. These details complement the structural overview in the main text and are retained separately to make the implementation and the role of each component explicit.

B.1 MESSAGE PASSING-BASED DRL

Our construction requires several teacher views, each of which can subsequently support a prototype graph. DisenGCN (Ma et al., 2019) provides a natural starting point because its routing mechanism groups neighboring messages into latent aspects. Below, we retain this routing procedure and describe how its aggregation step can incorporate the message and update operators of an MPNN teacher f_{θ_T}.

The routing model assumes that neighbors $v \in V$ connected to a center node $u \in V$ that are heavily due to the k-th aspect $\mathcal{A}_k$, $k \in [1, K]$ (*i.e.*, with large probability $p_{k,v}^{[u]}$, where $\sum_{k=1}^{K} p_{k,v}^{[u]} = 1$ and $p_{k,v}^{[u]} \in [0, 1]$) should be well-aligned in a sense that they can send consistent global messages [2]

[2]Here, *global messages* mean that $z_{k,v}$ is not explicitly related to $u \in V$.

$z_{k,v} = \mathrm{MSG}_k(h_v) \in \mathbb{R}^{d/K}$ from embeddings $h_v \in \mathbb{R}^d$ to the k-th aspect space $\mathbb{S}_k$. In other words, those $\left\{p_{k,j}^{[u]} \cdot z_{k,j}\right\}_{j \in \{u\} \cup N_u}$ should be well-clustered in $\mathbb{S}_k$, *e.g.*, well-fitting a prior such as a von Mises-Fisher (vMF) distribution.

To this end, given fixed u and $\{z_{k,j}\}_{k,j}$, it initializes the cluster centroids $\{c_k\}_k$ at $\{z_{k,u}\}_k$ and optimizes them alternatively with $\left\{p_{k,j}^{[u]}\right\}_{k,j}$ by maximizing the marginal likelihood via expectation-maximization (EM), *i.e.*, the *routing* mechanism, where the E steps update $\{p_{k,j}^{[u]}\}_{k,j}$ with a simple probabilistic estimation as:

$$p_{k,j}^{[u]} = \mathrm{Softmax}_k\left(z_{k,j}^T \, c_k/\tau\right), \tag{17}$$

via a temperature τ defined in vMF and the M steps update $\{c_k\}_k$ with the following summing formula:

$$c_k \propto z_{k,u} + \sum_{v \in N_u} p_{k,v}^{[u]} \cdot z_{k,v}. \tag{18}$$

After T_0 iterations, it treats the final $\{c_k\}_k$ as the disentangled representations of node u and their concatenation as node embedding h_u.

Instead, to adapt to more MPNNs, we can further aggregate the messages $\{z_{k,j}\}_j$ in $\mathbb{S}_k$, which naturally generalizes Eq. 18:

$$c'_k \propto \mathrm{UPD}\left(z_{k,u},\ \mathrm{AGG}\left(\{z_{k,v}\}_{v \in N_u};\ \{p_{k,v}^{[u]}\}_{v \in N_u}\right)\right), \tag{19}$$

via operators of updating $\mathrm{UPD}(\cdot,\cdot)$ and probabilistic weighted aggregations $\mathrm{AGG}(\cdot;\cdot)$. The embedding h_u will be updated by concatenating $\{R_{k,u}\}_k = \{c'_k\}_k$ (Eq. 8 in the main text). Algo. 3 in Appx. has more details. Notably, DRL methods are just examples of implementing this stage; please check Sec. M in Appx. for some possible alternative ideas.

B.2 More Details of Multi-Aspect Cooperative Distillation (MACD)

Sec. 2.2 presents the final form of MACD (Eq. 7). Its basic form, before detaching and confidence discounting, is:

$$L^{(Dis)} = \frac{1}{nK} \sum_{k=1}^{K} \sum_{u \in V} \mathrm{KL}\left(P_u^{(T)} \,\|\, P_u^{(T)[k]}\right), \tag{20}$$

where $\mathrm{KL}\left(\cdot \,\|\, \cdot\right)$ denotes the Kullback–Leibler divergence. Both the holistic and aspect predictions evolve during pre-training. If gradients pass through both arguments of this KL loss, the holistic prediction can move toward a poorly trained aspect instead of acting as its reference. This motivates stopping gradients through the target distribution; confidence discounting addresses the separate issue of unreliable teacher predictions. Figure 4 summarizes the complete flow, including optional diversity regularization and graph-basis construction.

The two modifications (*i.e.*, $\mathrm{Detach}(\cdot)$ and confidence discounting) lead to Eq. 7.

Furthermore, despite the randomness in initialization of $\{\mathrm{MSG}_k\}_k$, the diversity of $\{\mathcal{A}_k\}_k$ may still degrade, especially with MACD. To improve diversity, we optionally regularize $Z_u = \{z_{k,u}\}_k \in \mathbb{R}^{K \times d/K}$ via $L^{(Div)}$ in Eq. 21. Here, regularizing $\{\mathrm{MSG}_k\}_k$ and $\{c_k\}_k$ may be options, too. Specifically, with an identity matrix $I_K \in \mathbb{R}^{K \times K}$, we formalize $L^{(Div)}$ as below:

$$L^{(Div)} = \frac{1}{n} \sum_{u \in V} \mathrm{MSE}\left(Z_u \cdot Z_u^T,\ I_K\right). \tag{21}$$

Finally, incorporating the supervised loss of the teacher $L_{Sup}^{(T)}$:

$$L_{Sup}^{(T)} = \frac{1}{|V_L|} \sum_{u \in V_L} L\left(P_u^{(T)}, y_u\right), \tag{22}$$

we minimize the following comprehensive pre-training loss $L_{Pre}^{(T)}$(which reduces to Eq. 10 when $\alpha_{Div} = 0$):

$$L_{Pre}^{(T)} = L_{Sup}^{(T)} + \alpha_{Dis} \cdot L^{(Dis)} + \alpha_{Div} \cdot L^{(Div)}, \tag{23}$$

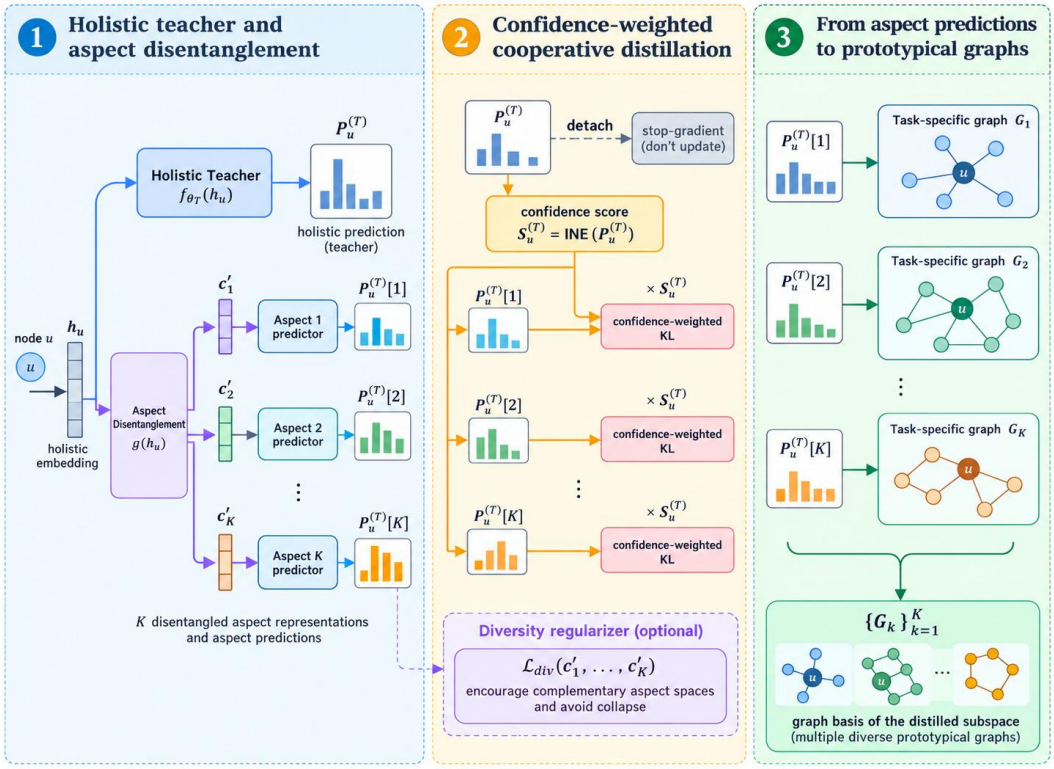


Figure 4: **Full illustration of Multi-Aspect Cooperative Distillation (MACD).** MACD distills complementary task-relevant knowledge from a holistic teacher into multiple disentangled aspects and subsequently constructs a diverse graph basis. Given the holistic embedding h_u of node u, the teacher produces a holistic prediction $P_u^{(T)}$, while an aspect-disentanglement module decomposes h_u into K aspect representations, from which the corresponding aspect predictions $\{P_u^{(T)[k]}\}_{k=1}^K$ are obtained. The holistic prediction is detached and serves as the distillation target for the aspect branches. To account for the reliability of teacher knowledge, MACD computes a confidence score $S_u^{(T)} = \text{INE}(P_u^{(T)})$ and uses it to weight the KL-based cooperative distillation objective, so uncertain teacher predictions contribute less to the transfer process. An optional diversity regularizer further encourages the disentangled aspects to capture complementary knowledge and mitigates aspect collapse. Finally, the resulting aspect-specific predictions are converted into K task-specific prototypical graphs $\{G_k\}_{k=1}^K$, which jointly constitute the graph basis of the distilled interaction space. The two-stage main-text overview in Fig. 2 shows where MACD enters Stage 1.

with two trade-off hyper-parameters $\alpha_{Dis}, \alpha_{Div} \geq 0$. Notably, we omit the students' direct supervision $L_{Sup}^{(T)[k]}$, which is typically encapsulated in $L^{(Dis)}$ [3] in most cases of our settings.

This pre-training scheme couples holistic prediction with aspect-wise guidance. The concatenated representation contributes to the teacher's supervised prediction, while each aspect learns from that prediction over both labeled and unlabeled nodes. The diversity penalty, when used, has a different role: it discourages redundant message directions rather than improving task relevance directly. Task relevance and structural complementarity should therefore be assessed separately; the AAA comparison evaluates the former.

A game-theoretical interpretation of the competition and cooperation among aspects is given in Sec. L.1.

[3] $L_{Sup}^{(T)[k]} = \frac{1}{|V_L|} \sum_{u \in V_L} L\left(P_u^{(T)[k]}, y_u\right)$. Since in our settings, the model typically well fits the training examples, *i.e.*, $L_{Sup}^{(T)} = \frac{1}{|V_L|} \sum_{u \in V_L} L(P_u^{(T)}, y_u) \approx 0$ meaning $P_u^{(T)} \approx \text{Onehot}(y_u)$ and $S_u^{(T)} \approx 1$ for u in V_L.

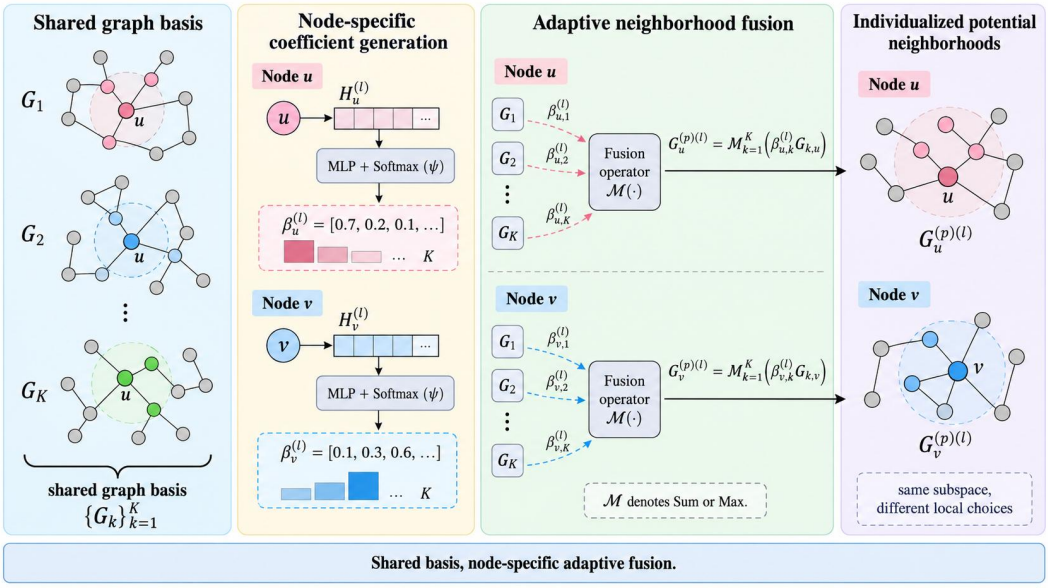


Figure 5: **Full illustration of node-wise adaptive fusion in the distilled interaction space.** Given the shared graph basis $\{G_k\}_{k=1}^K$ distilled in Stage 1, the student performs node-specific graph searching by generating adaptive fusion coefficients for each node. For node u at layer l, its semantic representation $H_u^{(l)}$ is fed into an MLP followed by a Softmax function to produce the node-specific coefficient vector $\boldsymbol{\beta}_u^{(l)} = [\beta_{u,1}^{(l)}, \ldots, \beta_{u,K}^{(l)}]$. These coefficients weight the node-centered neighborhoods $\{G_{k,u}\}_{k=1}^K$ from different prototypical graphs, which are then fused by an aggregation operator $\mathcal{M}(\cdot)$, such as element-wise summation or max pooling, to form the individualized potential neighborhood $G_u^{(p)(l)} = \mathcal{M}_{k=1}^K(\beta_{u,k}^{(l)} G_{k,u})$. The same shared graph basis is used for all nodes, while the node-specific coefficients allow different nodes to emphasize different prototypical structures, thereby enabling adaptive local graph search within a common distilled interaction space. Under the broader subspace-distillation analogy, this interaction space can be viewed as the graph-structured instance of the transferred subspace.

B.3 Accelerating Graph Construction via MIPS

Recall the edge weights in Eq. 11. Although each node retains only K' other neighbors, evaluating all pairwise scores costs $\mathcal{O}(n^2 C)$ per aspect. The prediction dimension is C, rather than the aspect embedding dimension d/K. The following proposition rewrites the ranking as an inner product and thereby permits indexed retrieval. It assumes positive prediction entries and positive endpoint confidence; zero-confidence edges have zero original weight and must be handled as boundary cases.

Proposition B.1. $\mathrm{ArgTop}_{v \in V} W^{(k)}(u, v)$
$= \mathrm{ArgTop}_{v \in V} \left\langle P_u^{(T)[k]} \,\|\, a, \;\; \tau' \cdot \log P_v^{(T)[k]} \,\|\, \log S_v^{(T)[k]} \right\rangle$.

The proof can be found in Sec. E. The reduction allows the use of Maximum Inner Product Search (MIPS) implementations such as Faiss (Johnson et al., 2019). It establishes ranking equivalence, not a universal subquadratic retrieval guarantee. For a fixed query with positive confidence, candidate-specific terms alone determine the ordering; the complete algebra is provided in the proof.

B.4 Full Illustration of Node-Wise Adaptive Fusion

Sec. 2.3.1 defines the node-wise fuser used in Stage 2. Figure 5 visualizes the full process: the same shared graph basis is reused for all nodes, while each node at each layer predicts its own fusion coefficients from the current student representation. This separation is important because the prototypes specify the admissible structural family, whereas the node-wise coefficients determine how each student state navigates that family.

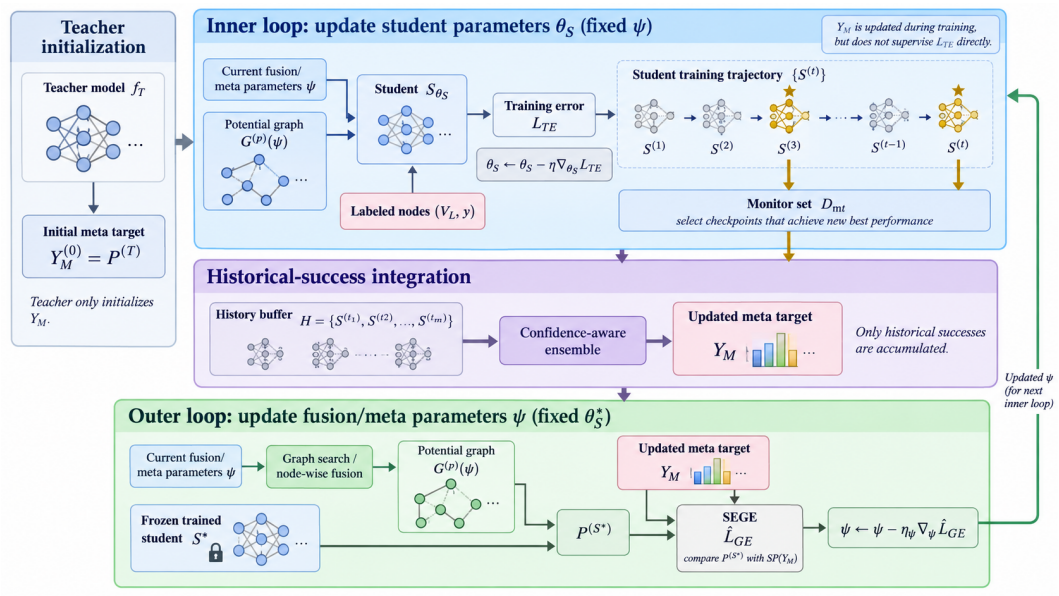


Figure 6: **Full illustration of the Meta Self-Update mechanism used in Stage 2.** The teacher first initializes the meta target $Y_M^{(0)} = P^{(T)}$. With fixed meta/fusion parameters ψ, the inner loop updates the student parameters θ_S by minimizing the training error L_{TE} on labeled nodes. As training proceeds, checkpoints that achieve new best performance on the monitor set $\mathcal{D}_{mt}$ are treated as historical successes and accumulated in a history buffer. These historical successes are integrated by a confidence-aware ensemble to refine the meta target Y_M. With the trained student fixed, the outer loop then updates ψ by minimizing the self-estimated generalization error $\widehat{L}_{GE}$ based on the updated meta target. This figure complements the compact Stage-2 overview in Fig. 2 and the complete procedure in Algorithm 5.

B.5 More Details of the Meta Self-Update Mechanism

Section 2.3.2 uses a dynamic target set to guide the fuser separately from labeled-node fitting. We now specify the monitor, its update threshold, and the confidence-weighted target ensemble. The monitor is fixed and teacher-pseudo-labeled, whereas the target set used in the outer loss is updated. Keeping these roles distinct is important: the monitor supplies a selection reference, not independent ground-truth validation labels. Figure 6 provides a complete view of how teacher initialization, inner-loop student fitting, historical-success integration, and outer-loop fuser updates interact.

Specifically, we fix $\mathcal{V}_M = V$ and dynamically estimate $\mathcal{Y}_M$. We set *Monitor* data $\mathcal{D}_{mt} = (\mathcal{V}_{mt}, \mathcal{Y}_{mt})$, initial $\mathcal{D}_M = \left(V, \{P_u^{(T)}\}\right)$, and initial agreement threshold $\gamma = \gamma_0$ with $\mathcal{Y}_{mt,u} = \arg\max P_u^{(T)}$, $\mathcal{V}_{mt} = \{u \in V | S_u^{(T)} \geq \beta_S\}$, and large hyper-parameters $\beta_S, \gamma_0 \in [0, 1]$. At each inner step, after updating $\mathcal{S}$ itself (*i.e.*, learnable parameters), if it has higher agreement on $\mathcal{D}_{mt}$ than γ, *i.e.*, $\mathrm{Acc}\left(\mathcal{S}, \mathcal{D}_{mt}\right) > \gamma$, we update $\gamma \leftarrow \mathrm{Acc}\left(\mathcal{S}, \mathcal{D}_{mt}\right)$ and $\mathcal{Y}_M \leftarrow \mathrm{Ens}\left(\mathcal{Y}_M, \{P_u^{(S)}\}_{u \in V}\right) = (\mathbf{1} - \boldsymbol{\lambda}) \odot \mathcal{Y}_M + \boldsymbol{\lambda} \odot \{P_u^{(S)}\}$ with $\boldsymbol{\lambda} \in [0, 1]^n$ defined (*e.g.*) in Eq. 24, $\forall\, u \in V$:

$$\boldsymbol{\lambda}_u = \mathrm{INE}\left(P_u^{(S)}\right)^{\xi} \Big/ \left(\mathrm{INE}\left(\mathcal{Y}_{M,u}\right)^{\xi} + \mathrm{INE}\left(P_u^{(S)}\right)^{\xi}\right). \tag{24}$$

The update resembles other approaches that optimize a tractable surrogate in place of a difficult objective, including Variational Inference (Zhang et al., 2018; Wan et al., 2020), Bound Optimization (Salakhutdinov & Roweis, 2003; Salakhutdinov et al., 2012; McMahan & Streeter, 2010), and Majorization Minimization (Breloy et al., 2021; Phan et al., 2023; Tang et al., 2023). This is an analogy about the role of a surrogate, not an application of their convergence theorems: we do not prove that $\widehat{L_{GE}}$ bounds the true risk or that the pseudo-labels improve monotonically. In particular, $\mathrm{Acc}(\mathcal{S}, \mathcal{D}_{mt})$ above measures agreement with teacher labels. Better agreement can select a useful

student state, but is not itself proof of better generalization. Section L.4 discusses fixed-teacher targets, alternative ensembling rules, and possible monitor updates in more detail.

B.6 Supporting Student Fitting and Structure Selection

The outer surrogate evaluates a potential graph through the predictions of a student trained under that graph. A poorly fitted student can therefore provide a misleading signal about the structure, even when the surrogate targets are useful. In the ideal formulation of Eq. 3, the inner solution $\{\alpha_{i,j}\}^*, \theta^*$ minimizes $L_{TE}(\cdot; G^{(p)})$, not the outer objective itself. To improve this inner fitting process, we retain two complementary refinements: a curriculum that controls how strongly attention initially follows the graph prior, and a regularizer that teaches the unmasked attention scores to recognize supported neighbors. Both are optional additions to the labeled-node loss.

(I) A Near-to-far Curriculum-driven Local Refinement. When a student gets a solution from a teacher, he may try it *directly* before *indirectly* (*i.e.*, with near-to-far modifications). Therefore, motivated by this, to facilitate better exploitation of the searched graph $G^{(p)}$, we propose a curriculum-driven local refinement to train the student with strict-to-loose guidance, *i.e.*, by encouraging $\alpha_{i,\cdot}$ to first simulate a uniform distribution on neighborhood N_i in $G^{(p)}$ (*i.e.*, strictly following the teacher $\mathcal{T}$, $G^{(agg)} = \text{RowNorm}\left(\{\alpha_{i,j}\} \odot G^{(p)}\right) \approx G^{(p)}$) and gradually evolve into a sparser one (*i.e.*, loosely following $\mathcal{T}$ and going further), which can be driven by a simple entropy regularization locally on $G^{(p)}$, *i.e.*:

$$L^{(Cur)} = -\frac{1}{nL} \sum_{l\in[1,L]} \sum_{u\in V} \sum_{v\in V: G^{(p)}_{u,v}>0} \alpha^{(l)}_{u,v} \cdot \log\left(\alpha^{(l)}_{u,v}\right), \tag{25}$$

with a negative-to-positive Cosine curriculum schedule $\text{Cos-Schedule}\,(\cdot)$, $\forall\, t \in \left[1, T^{(in)}\right]$:

$$\begin{aligned} \lambda_{Cur} &= \text{Cos-Schedule}\left(t;\ t^{[B]}, t^{[E]}, \lambda^{[B]}, \lambda^{[E]}\right), \\ &= \lambda^{[B]} + \frac{\lambda^{[E]} - \lambda^{[B]}}{2} \cdot \\ &\left(1 + \cos\left(\pi \cdot \text{Clamp}\left(1 - \frac{t - t^{[B]}}{t^{[E]} - t^{[B]}},\ 0,\ 1\right)\right)\right), \end{aligned} \tag{26}$$

where $\lambda^{[B]} \le \lambda^{[E]}$ and $1 \le t^{[B]} \le t^{[E]} \le T^{(in)}$ are hyper-parameters. $\text{Clamp}\,(t, a, b)$ clips t into the interval $[a, b]$. λ_{Cur} evolves with the inner epoch number t in $L^{(\mathcal{S})}_{TE}$ (Eq. 30). With $L^{(Cur)}$ defined as attention entropy, a negative coefficient encourages a more uniform distribution, whereas a positive coefficient favors selective weighting. The intended schedule thus begins with $\lambda^{[B]} \le 0$ and ends with $\lambda^{[E]} \ge 0$, with $t^{[E]} > t^{[B]}$ for the displayed interpolation. "Near-to-far" describes increasing deviation from the graph prior, not an increasing hop distance. Configurations that start with a positive coefficient implement a selective-attention schedule rather than this uniform-to-selective special case.

(II) A Kernel Decomposition-based Implicit Sparsity Regularization. Knowledge imbalance can make the candidate neighborhoods of uncertain nodes less reliable. These nodes may need more selective attention than nodes with confident, label-consistent neighborhoods. Importantly, this distinction concerns prediction reliability rather than whether a node belongs to the training set. However, the sparsity of $G^{(p)}$ (*i.e.*, *explicit sparsity*) can only ensure sparse aggregations to reduce complexities, yet providing too weak supervision for elaborate attention learning and direct knowledge transferring (discussed in Sec. L.5). To facilitate *implicit sparsity*, we further row-wisely regularize $\mathbf{QK}^T$ ($L^{(Reg)}$ in Eq. 27) to reduce attentions outside $G^{(p)}$ with uncertainties, where nodes u with confident neighborhoods $G^{(p)}_u$ (*i.e.*, Eq. 28) are restricted more strictly, which can be formulated as:

$$\begin{aligned} L^{(Reg)} &= -\frac{1}{L} \cdot \sum_{l=1}^{L} \frac{1}{C^{(l)}} \sum_{u\in V} \Gamma^{(l)}_u \cdot \\ &\log\left(\frac{\sum_{v\in V: G^{(p)(l)}_{u,v}>0} \exp\left(\mathbf{Q}^{(l)}_u \cdot \mathbf{K}^{(l)T}_v / \sqrt{d}\right)}{\sum_{v\in V} \exp\left(\mathbf{Q}^{(l)}_u \cdot \mathbf{K}^{(l)T}_v / \sqrt{d}\right)}\right), \\ C^{(l)} &= \sum_{u\in V} \Gamma^{(l)}_u \in \mathbb{R}, \end{aligned} \tag{27}$$

where $\Gamma_u^{(l)}$ is the confidence of node u's neighborhood $G_u^{(p)(l)}$, defined below with $\beta_{u,k}^{(l)}$ in Eq. 15 and $S_u^{(\mathcal{T})[k]} = \mathrm{INE}\left(P_u^{(\mathcal{T})[k]}\right)$ (Eq. 9):

$$\Gamma_u^{(l)} = \mathrm{Confidence}\left(G_u^{(p)(l)}\right) = \sum_{k=1}^{K} \beta_{u,k}^{(l)} \cdot S_u^{(\mathcal{T})[k]} \in \mathbb{R}. \tag{28}$$

Incorporating **(I)** and **(II)** with the supervised loss $L_{Sup}^{(\mathcal{S})}$, *i.e.*:

$$L_{Sup}^{(\mathcal{S})} = \frac{1}{|V_L|} \sum_{u \in V_L} L\left(P_u^{(\mathcal{S})}, y_u\right), \tag{29}$$

we obtain the following comprehensive training loss $L_{TE}^{(\mathcal{S})}\left(\{\alpha_{i,j}\}, \theta\right)$ (Eq. 30) for the inner optimizations of the student $\mathcal{S}$ (Eq. 3):

$$L_{TE}^{(\mathcal{S})} = L_{Sup}^{(\mathcal{S})} + \lambda_{Cur} \cdot L^{(Cur)} + \lambda_{Reg} \cdot L^{(Reg)}, \tag{30}$$

where $\lambda_{Reg} > 0$ is a hyper-parameter, and λ_{Cur} is defined in Eq. 26.

Moreover, to improve the scalability, we utilize a Random Feature-based method (Likhosherstov et al., 2022) to decompose the attention kernel $\exp\left(\mathbf{a}^T\mathbf{b}\right)$ into inner products $\langle \mathbf{a}', \mathbf{b}' \rangle$ for efficient calculation (Sec. K). Please check Algo. 5 for detailed implementations of this stage.

C Experimental Results Omitted from the Main Text

C.1 Full Distillation Comparison and Ablation Studies

In addition to the observations in Sec. 3, Tab. 2 also shows: **A vs. F**: $\mathcal{S}$ can sometimes benefit from a single graph distillation but not always; **BC vs. FG**: $\mathcal{S}$ guided by $\mathcal{T}$ CANNOT even consistently surpass the vanilla teacher $\mathcal{T}$, signifying a distillation gap against ours;

Tab. 5 reports the complete ablation studies, where we:

(1) drop Multi-Aspect Cooperative Distillation (MACD) $L^{(Dis)}$ in Eq. 10; **(2)** drop confidence in weighting schema, *i.e.*, Eq. 11; **(3)** modify a node-wise fuser $\mathrm{Fuse}_\psi(\cdot)$ to a graph-wise fuser; **(4)** modify the structure-selection surrogate $\widehat{L_{GE}}$ into a training loss in Eq. 3; **(5)** modify $\widehat{L_{GE}}$ into filtered dark knowledge (*i.e.*, teacher's predictions) in Eq. 3; **(6)** drop curriculum-driven local refinement $L^{(Cur)}$ in Eq. 30; **(7)** drop implicit sparsity regularization $L^{(Reg)}$ in Eq. 30; **(8)-(9)** modify graph search into random prototypical graph selection or the input original graph, testing whether searching surpasses special single graphs (Sec. L.7).

Table 5: The complete results of ablation studies

No.	Method	Cora	Citeseer	Pubmed	Actor	Cornell	Texas	Wisconsin	ArXiv	Flickr
(1)	*w.o.* $L^{(Dis)}$	53.70	51.50	69.40	34.54	70.27	72.97	78.43	53.84	39.36
(2)	*w.o. confidence*	81.65	73.35	81.74	37.24	<u>80.00</u>	79.46	80.00	56.33	41.80
(3)	node-wise fuser ⇒ graph-wise fuser	84.04	73.38	81.50	37.04	<u>80.00</u>	78.92	77.25	56.56	42.21
(4)	$\widehat{L_{GE}}$ ⇒ training loss	83.96	73.26	81.80	<u>37.98</u>	78.92	78.38	81.18	<u>56.86</u>	42.27
(5)	$\widehat{L_{GE}}$ ⇒ teacher's predictions	83.82	73.36	81.76	37.33	78.92	78.38	81.18	56.75	42.13
(6)	*w.o.* $L^{(Cur)}$	83.95	73.34	81.66	37.43	78.38	78.92	<u>82.35</u>	56.81	41.99
(7)	*w.o.* $L^{(Reg)}$	<u>84.23</u>	<u>73.42</u>	81.83	35.99	78.38	80.00	79.22	56.76	<u>42.58</u>
(8)	graph search ⇒ random graph selection	82.00	72.20	<u>81.90</u>	35.59	70.27	<u>81.08</u>	78.43	54.48	40.84
(9)	graph search ⇒ original graph	77.60	68.32	76.60	36.25	70.27	76.76	80.39	54.38	29.46
(·)	Ours	**84.88**	**74.05**	**82.58**	**38.74**	**81.08**	**83.24**	**84.71**	**57.28**	**43.81**

C.2 Hyper-Parameter Studies

To understand the effect of the number of graphs/aspects on the comprehensive performance of our distillation framework, we study the hyper-parameter K (Sec. 2.2) on three citation networks (*i.e.*, Cora, Citeseer, and Pubmed) and plot their results in terms of average accuracies and their standard deviations in Fig. 7. We can observe from Fig. 7: (1) the performance generally exhibits ascending and then descending trends; (2) despite the distinctness of the best hyper-parameter Ks on different

benchmarks, the results are relatively stable between 4 and 6; (3) several graphs are enough for clear performance gain, meaning that the time and space bottlenecks are quite acceptable (*e.g.*, setting $K = 2$ or 4 in most cases).

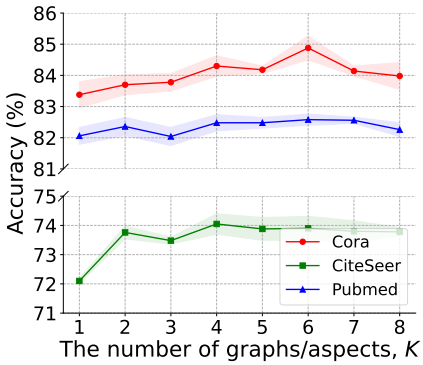


Figure 7: The effect of the number of graphs/aspects (*i.e.*, the hyper-parameter K) on our average performance.

C.3 Embedding Visualizations

Furthermore, we also visualize the node embeddings produced by ours and counterparts in Fig. 8 and Fig. 17 of Sec. S for better intuition. The more separated clusters provide a qualitative view of the learned embeddings. Such projections complement the accuracy comparisons, but do not establish classification margins or explain the gains by themselves.



(a) GCN on Cora (b) Ours on Cora

Figure 8: Visualizations of node embeddings on Cora benchmark produced by GCN and Ours.

C.4 Comparison with More Recent Graph Transformers

We further compare our framework with four recent Graph Transformers, *i.e.*, GCFormer (Chen et al., 2024), DUALFormer (Zhuo et al., 2025), Polynormer (Deng et al., 2024), and CoBFormer (Xing et al., 2024), under the same settings as Tab. 1. As shown in Tab. 6, our framework achieves the best results on Cora and Pubmed, and remains highly competitive with the strongest counterpart CoBFormer on Citeseer and ArXiv (with gaps of 0.03 and 0.23, respectively). The differences show that performance depends on the low-label protocol as well as the architecture. These accuracy values alone do not identify overfitting as the cause of a baseline's lower result.

Table 6: Comparison with more recent Graph Transformers (with the best bolded and the runner-ups underlined). TLE denotes exceeding the time limit in the sampling step.

Method	Cora	Citeseer	Pubmed	ArXiv
GCFormer	$78.50_{\pm 0.93}$	$49.70_{\pm 1.96}$	$69.30_{\pm 1.00}$	TLE
DUALFormer	$81.76_{\pm 0.93}$	$70.20_{\pm 1.06}$	$77.68_{\pm 0.84}$	$13.67_{\pm 1.04}$
Polynormer	$78.68_{\pm 0.67}$	$68.04_{\pm 0.35}$	$77.62_{\pm 0.37}$	$51.31_{\pm 0.62}$
CoBFormer	$\underline{84.01}_{\pm 1.06}$	$\mathbf{74.08}_{\pm 0.72}$	$\underline{80.80}_{\pm 0.27}$	$\mathbf{57.51}_{\pm 0.43}$
Ours	$\mathbf{84.88}_{\pm 0.38}$	$\underline{74.05}_{\pm 0.35}$	$\mathbf{82.58}_{\pm 0.18}$	$\underline{57.28}_{\pm 0.23}$

C.5 Focused Comparison with M^{3}Dphormer

M^{3}Dphormer (Xing et al., 2025) is a particularly relevant recent baseline because it also represents node interactions through multiple candidate structures: it combines local, cluster, and global hierarchical attention masks with adaptive expert routing. This makes it a useful comparison for our interaction-space perspective. The key difference is where the candidate structures come from. M^{3}Dphormer integrates three theoretically motivated hierarchical mask types defined by its architecture, whereas our Stage 1 distills task-informed prototypical graphs from a teacher and Stage 2 learns node-/layer-specific combinations in the resulting interaction space. The comparison therefore checks whether our gains persist against a strong multi-mask, adaptively routed GT rather than only against single-structure or earlier sparse-attention baselines.

Table 7 reports the comparison on Cora, Citeseer, and Pubmed using the same public splits as our main experiments: $140/500/1000$, $120/500/1000$, and $60/500/1000$, respectively. Our mean test accuracy is higher by 7.06, 6.97, and 5.12 percentage points on the three datasets. This focused result is consistent with the value of constructing a task-informed interaction space instead of relying only on a predefined collection of masks. As with any end-to-end comparison, however, the accuracy gap reflects the complete methods and does not by itself isolate the causal contribution of any single design choice.

Table 7: Focused comparison with M^{3}Dphormer on the public citation-network splits used in our main experiments. Results are test accuracy (%).

Method	Cora	Citeseer	Pubmed
M^{3}Dphormer	$77.82_{\pm 1.40}$	$67.08_{\pm 1.61}$	$77.46_{\pm 0.79}$
Ours	$\mathbf{84.88}_{\pm 0.38}$	$\mathbf{74.05}_{\pm 0.35}$	$\mathbf{82.58}_{\pm 0.18}$

C.6 Memory Cost

Tab. 8 reports the GPU memory usage of our framework and two representative GTs. Our framework uses more memory than SGFormer on the three citation networks and more than Graphormer on Cora and Citeseer. Maintaining several prototypical graphs adds storage, reflecting a trade-off between the richness of the interaction space and memory use. Graphormer runs out of memory on Pubmed under the reported setting, so no numerical memory comparison is available there. On the large-scale ArXiv, it requires 13421MB, while Graphormer runs out of memory (Tab. 1). Graph sampling techniques (*e.g.*, Cluster-GCN (Chiang et al., 2019)) offer a possible direction for reducing memory use, but a sampled implementation and its effect on prototype coverage require separate evaluation.

Table 8: GPU memory usage (MB). "–" denotes not measured.

Method	Cora	Citeseer	Pubmed	ArXiv
SGFormer	1262	1302	1416	–
Graphormer	1282	1320	OOM	OOM
Ours	2244	2522	2698	13421

C.7 Searching in the Subspace vs. Using the Best Prototypical Graph

One may directly adopt the best prototypical graph (*i.e.*, the best aspect) instead of learning combinations in the interaction space it helps define. We replace graph search with the prototypical graph of the best aspect and compare it with the variants in Tab. 5 in Tab. 9. Using the best individual prototype is worse than adaptive search on all three benchmarks. This supports the value of learning combinations in the tested setting, rather than assuming that the strongest teacher aspect supplies the best student graph. With Sum fusion, the simplex-defined family includes the individual prototypes at its boundary as well as their mixtures; the softmax fuser approximates boundary choices with concentrated coefficients. These results do not imply that every fixed graph is inferior, nor do they isolate edge-budget effects without additional matched-support comparisons.

Table 9: Searching in the distilled interaction space vs. using single graphs.

Method	Cora	Citeseer	Pubmed
graph search ⇒ original graph	77.60	68.32	76.60
graph search ⇒ random prototypical graph	82.00	72.20	81.90
graph search ⇒ best prototypical graph	82.50	72.40	82.00
Ours (graph search)	**84.88**	**74.05**	**82.58**

D Symbol Table

To provide a better understanding of the notation and symbols appearing in this paper, we give concise and clear symbol tables, *i.e.*, Tab. 10 and Tab. 11, including some important notations, their explanations, and possible cross-reference.

E Omitted Proofs

In this section, we give the omitted proof of Proposition B.1 in Sec. B.3.

E.1 The Proof of Proposition 1

Recall Proposition B.1 in Sec. B.3.

Proposition E.1 (Restatement of Proposition B.1)**.**

$$\begin{aligned} &\operatorname{ArgTop}_{v\in V} W^{(k)}(u,v) \\ =&\operatorname{ArgTop}_{v\in V} \\ &\left\langle P_u^{(T)[k]} \parallel a,\ \left(\tau' \cdot \log P_v^{(T)[k]}\right) \parallel \log S_v^{(T)[k]} \right\rangle. \end{aligned} \tag{31}$$

Proof. Fix a query node u with $S_u^{(T)[k]} > 0$, and consider candidates with positive confidence and strictly positive prediction components. These conditions permit taking logarithms. The reduction applies to the top-K' ranking with the same candidate set and tie rule. The expansion below distinguishes score identities from ranking equivalences obtained by removing query-only terms. Recall the definition of $W^{(k)}(u,v)$ in Sec. 2.2 of the main text:

$$W^{(k)}(u,v) = \left(S_u^{(T)[k]}\right)^a \cdot \left(S_v^{(T)[k]}\right)^a \cdot \tag{32}$$

$$\exp\left\{-\tau' \cdot \mathrm{KL}\left(P_u^{(T)[k]} \parallel P_v^{(T)[k]}\right)\right\} \in [0,1]. \tag{33}$$

Besides, recall the definition of Kullback–Leibler divergence:

$$\mathrm{KL}(P \parallel Q) = -\sum_{x\in\mathcal{X}} P(x)\ \log\left(\frac{Q(x)}{P(x)}\right), \tag{34}$$

Table 10: The Symbol Table Containing Notations and Their Explanations I

Notation	Explanation	Cross-Reference
$G = (V, E)$	the original input graph G, the node set V, the edge set E	Sec. I
A, X	the adjacency matrix A and the feature matrix X	Sec. I
f, d, C	feature dim, hidden dim (embedding dim), and the number of node classes	Sec. I, Sec. 2
$H, H^{(l)}, \{H_u^{(l)}\}$, $\{H_u^{(l)}\}_u$, $\{H_u^{(l)}\}_{u\in V}$	all node embeddings (of layer l)	Sec. I, Sec. 2
n, m	$n = \|V\|, m = \|E\|$	Sec. I, Sec. H
$v_i \in V$	a node v_i	Sec. I, Sec. 2
V_L	the labeled node set, also the training set	Sec. I, Sec. 2
$G^{(agg)[l]}$	the aggregation graph of GT in layer $l \in [1, L]$, *i.e.*, similar to $\hat{A}$ in GCN	Sec. 2
$G^{(p)[l]}$	the potential graph of GT in layer $l \in [1, L]$, *i.e.*, similar to relative positional encoding (*e.g.*, original G in a LGT), denoting what graph the attention mechanism is conducted on	Sec. 2
$\{\alpha_{i,j}\}^{[l]}$	the attention coefficients (matrix) in layer $l \in [1, L]$	Sec. 2
$L(\cdot)$	the Cross-Entropy loss default	Sec. 2
$\mathrm{MLP}(\cdot)$	a general MLP, typical containing 1 or 2 layers	Sec. 2
$L_{TE}, L_{GE}, \hat{L_{GE}}$	the general comprehensive training loss, real/true generalization loss/error, and the structure-selection surrogate	Sec. 2.1, Sec. 2.3
f_{θ_T}	a general MPNN teacher	Sec. 2.2
K	the number of different aspects or prototypical graphs	Sec. 2.2
$\{\mathcal{A}_k\}_{k\in[1,K]}$	K different aspects	Sec. 2.2
$\{G_k\}_{k\in[1,K]}$	K different prototypical graphs	Sec. 2.2
$\mathbb{G}$	the distilled interaction space	Sec. 2
$R_i, S_i, \forall\, i \in [1, K]$	the abstract notation of disentangled representation (matrix) and auxiliary knowledge (*e.g.*, confidence score) from aspect i, *i.e.*, $\mathcal{A}_i$	Sec. 2
$p_{k,v}^{[u]}$	how much (or a probability) node v can benefit the neighborhood of node u from $\mathcal{A}_k$	Sec. 2
$z_{k,v}, z_{k,u}, Z_u(k)$	the global messages sent from nodes v and u, and a node will send K messages shared by all neighbors	Sec. 2
N_u	the first-order neighborhood of node $u \in V$	Sec. 2
$c_k, c_k^{[u]}, c_k^{(u)}$	the cluster centroid in the $\mathcal{A}_k$ of the first-order neighborhood of the ***conditioned*** node u, note that c_k is related to u (omitting u for brevity)	Sec. 2
$c_k', c_k^{[u]\prime}, c_k^{(u)\prime}$	the specific disentangled representing of node u in the $\mathcal{A}_k$ (see Sec. 3)	Sec. 2
$\mathrm{MSG}_k(\cdot), \mathrm{UPD}(\cdot), \mathrm{AGG}(\cdot)$	the operators in MPNN, here MSG_k is viewed as a copy of MSG into $\mathcal{A}_k$	Sec. 2
$P_u^{(T)}, P_u^{(T)[k]}, P_u^{[\mathcal{T}]}, P_u^{[\mathcal{T}][k]}$	the predictions of the teacher $\mathcal{T}$ at node u (holistically from H_u or partly from $\mathcal{A}_k$, *i.e.*, c_k')	Sec. 2
$P_u^{(S)}, P_u^{(S)[k]}, P_u^{[\mathcal{S}]}, P_u^{[\mathcal{S}][k]}$	the predictions of the student $\mathcal{S}$	Sec. 2
$S_u^{(T)}, S_u^{(T)[k]}, S_u^{[\mathcal{T}]}, S_u^{[\mathcal{T}][k]}$	the confidence of predictions of the teacher $\mathcal{T}$ at node u (holistically from H_u or partly from $\mathcal{A}_k$, *i.e.*, c_k')	Sec. 2
$S_u^{(S)}, S_u^{(S)[k]}, S_u^{[\mathcal{S}]}, S_u^{[\mathcal{S}][k]}$	the confidence of predictions of the student $\mathcal{S}$	Sec. 2
$L_{Pre}^{(T)}, L_{Sup}^{(T)}, L^{(Dis)}, \cdot L^{(Div)}$	the comprehensive pre-training loss, the supervised loss of the teacher, the MACD distillation loss, the diversity regularization loss	Sec. 2

Table 11: The Symbol Table Containing Notations and Their Explanations II

Notation	Explanation	Cross-Reference
$I(\cdot;\cdot)$	the mutual information	Sec. 2
$W^{(k)}(u,v)$	the weighting formula	Sec. 2
$G_{k,u,v}$	the weights of the graph G_k	Sec. 2
$\mathrm{Fuse}_{\psi}(\cdot)$	a learnable node-wise fuser	Sec. 2
$\boldsymbol{\beta}_u^{(l)}$	the fusing coefficients for node u in the layer l	Sec. 2
$G_{k,u}$	the neighborhood of node u in G_k	Sec. 2
$\mathbb{G}_u$	the space of the neighborhood of u in $\mathbb{G}$	Sec. 2
$\mathcal{M}(\cdot)$	a merging operator, *e.g.*, Max or Sum	Sec. 2
$\mathcal{D}_M = (\mathcal{V}_M, \mathcal{Y}_M)$	the self-estimated meta-training data	Sec. 2.3
$\mathcal{D}_{mt} = (\mathcal{V}_{mt}, \mathcal{Y}_{mt})$	the monitor data	Sec. 2.3
$\mathrm{INE}(\cdot)$	the Inverted Normalized Entropy (INE) confidence score, defined in Eq. 9	Sec. 2.3
$L_{TE}^{(\mathcal{S})}, L_{Sup}^{(\mathcal{S})}, L^{(Cur)}, L^{(Reg)}$	the comprehensive training loss of the student $\mathcal{S}$, the supervised loss of the student, the curriculum term, and the implicit sparsity regularization term	Sec. 2.3
$\mathrm{Clamp}(x,\ 0,\ 1)$	change x to the nearest value in $[0,1]$	Sec. 2.3
$\mathbb{K}, \mathbb{K}^{[\mathcal{T}]}$	the abstract knowledge in knowledge distillation	Sec. J
$\mathrm{RowNorm}(B)$	normalize the sum of each row of the matrix B to one, by dividing all elements in row i via the sum of row i	Sec. 2

then:

$$\mathrm{KL}\left(P_u^{(T)[k]} \parallel P_v^{(T)[k]}\right) \tag{35}$$

$$= -\sum_{i=1}^{C} P_u^{(T)[k]}(i)\ \log\left(\frac{P_v^{(T)[k]}(i)}{P_u^{(T)[k]}(i)}\right). \tag{36}$$

Incorporating Eq. 35 into Eq. 32, we have:

$$W^{(k)}(u,v) \tag{37}$$

$$= \left(S_u^{(T)[k]}\right)^a \cdot \left(S_v^{(T)[k]}\right)^a \cdot \tag{38}$$

$$\exp\left\{\tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i)\ \log\left(\frac{P_v^{(T)[k]}(i)}{P_u^{(T)[k]}(i)}\right)\right\} \tag{39}$$

$$= \left(S_u^{(T)[k]}\right)^a \cdot \left(S_v^{(T)[k]}\right)^a \cdot \tag{40}$$

$$\exp\left\{\tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \left(\log\left(\ P_v^{(T)[k]}(i)\ \right) - \log\left(\ P_u^{(T)[k]}(i)\right)\right)\right\} \tag{41}$$

$$= \left(S_u^{(T)[k]}\right)^a \cdot \left(S_v^{(T)[k]}\right)^a \cdot \tag{42}$$

$$\exp\left\{\ \tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \log\left(\ P_v^{(T)[k]}(i)\ \right)\right. \tag{43}$$

$$\left. -\tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \log\left(\ P_u^{(T)[k]}(i)\right)\right\}. \tag{44}$$

Because the logarithm is strictly increasing on positive scores, $\mathrm{ArgTop}_v\, W^{(k)}(u,v) = \mathrm{ArgTop}_v \log W^{(k)}(u,v)$. In the next display, $\equiv_v$ denotes equality of rankings over v: terms depending only on the fixed query u may be dropped without changing the selected candidates. We obtain:

$$\mathrm{ArgTop}_{v \in V}: \tag{45}$$

$$\log\left(W^{(k)}(u,v)\right) \tag{46}$$

$$= a \cdot \log\left(S_u^{(T)[k]}\right) + a \cdot \log\left(S_v^{(T)[k]}\right) + \tag{47}$$

$$\left\{ \tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \log\left(P_v^{(T)[k]}(i) \right) \right. \tag{48}$$

$$\left. -\tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \log\left(P_u^{(T)[k]}(i) \right) \right\} \tag{49}$$

$$\equiv_v a \cdot \log\left(S_v^{(T)[k]}\right) + \tag{50}$$

$$\left\{ \tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \log\left(P_v^{(T)[k]}(i) \right) \right. \tag{51}$$

$$\left. -\tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \log\left(P_u^{(T)[k]}(i) \right) \right\} \tag{52}$$

$$\equiv_v a \cdot \log\left(S_v^{(T)[k]}\right) + \tag{53}$$

$$\tau' \cdot \sum_{i=1}^{C} P_u^{(T)[k]}(i) \cdot \log\left(P_v^{(T)[k]}(i) \right). \tag{54}$$

Therefore, we can reformultate Eq. 53 into a form of the following inner product:

$$\mathrm{ArgTop}_{v \in V} \ \log\left(W^{(k)}(u,v)\right) \tag{55}$$

$$= \left\langle a,\ \log\left(S_v^{(T)[k]}\right) \right\rangle + \left\langle P_u^{(T)[k]},\ \tau' \cdot \log\left(P_v^{(T)[k]} \right) \right\rangle \tag{56}$$

$$= \left\langle a \parallel P_u^{(T)[k]},\quad \log S_v^{(T)[k]} \parallel \tau' \cdot \log P_v^{(T)[k]} \right\rangle. \tag{57}$$

Thus, we have:

$$\mathrm{ArgTop}_{v \in V} W^{(k)}(u,v) \tag{58}$$

$$= \mathrm{ArgTop}_{v \in V} \left\langle P_u^{(T)[k]} \parallel a, \right. \tag{59}$$

$$\left. \left(\tau' \cdot \log P_v^{(T)[k]}\right) \parallel \log S_v^{(T)[k]} \right\rangle, \tag{60}$$

which concludes the proof for positive scores. If a candidate has zero confidence, its original edge weight is zero; it can be excluded before taking logarithms when enough positive candidates exist. If the query confidence is zero, all original scores are tied at zero and the logarithmic ranking formula is not defined. A numerical implementation must handle these boundary cases explicitly rather than evaluate $\log 0$. □

F Implementation Details

F.1 The Details of Attention Mechanisms

In this section, we briefly introduce the specific implementation of GTs. Typically, GTs adopt similar attention mechanisms in standard Transformers, considering the following cross-attention and self-attention formulations.

The mutual attention $Attn(X,Y) \in \mathbb{R}^{n \times d}$ with inputs $X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^{m \times d}$ can be extended to a kernel form:

$$\forall i \in [1,n]:$$

$$Attn(X,Y)_i = \sum_{j=1}^{m} \frac{\kappa(X_i, Y_j)}{\sum_{k=1}^{m} \kappa(X_i, Y_k)} \cdot Y_j \cdot \mathbf{W}^{(V)} \in \mathbb{R}^d, \tag{61}$$

$$Attn(X,Y,\mathcal{S})_i = \sum_{j \in \mathcal{S}_i} \frac{\kappa(X_i, Y_j)}{\sum_{k \in \mathcal{S}_i} \kappa(X_i, Y_k)} \cdot Y_j \cdot \mathbf{W}^{(V)} \in \mathbb{R}^d, \tag{62}$$

with

$$\kappa(x, x') = \exp\left(\mathbf{Q}_x^T \mathbf{K}_{x'} / \sqrt{d}\right) \tag{63}$$

$$= \exp\left(\left(x\mathbf{W}^{(Q)}\right)^{T} x'\mathbf{W}^{(K)}/\sqrt{d}\right), \tag{64}$$

and trainable $\mathbf{W}^{(Q)}, \mathbf{W}^{(K)}, \mathbf{W}^{(V)} \in \mathbb{R}^{d\times d}$. And Y contains m rows $\{Y_i\}$. Specially, the self attention $Attn\left(X\right) = Attn\left(X, X\right)$. Furthermore, Eq. 62 defines a sparse version where $\mathcal{S}_i$ denotes a set of indexes selected by X_i from $\{Y_j\}$, reducing the complexity from $O(nm)$ to $O\left(\sum_{i=1}^{n}|\mathcal{S}_i|\right)$.

We often call $Attn\left(X, Y, \mathcal{S}\right)$ Local Graph Transformers (*i.e.*, LGTs), since they conduct attention mechanisms locally on some graph, *e.g.*, vanilla LGT on the original graph. Notably, Eq. 61 and Eq. 62 are general in a sense that any sophisticated state-of-the-art attention calculation method can be incorporated into the kernel $\kappa\left(\cdot,\cdot\right)$

After the attention operations, Transformers typically require additional Feed Forward Networks (FFN) implemented via MLPs (*e.g.*, 2-layer MLPs with an intermediate $\mathrm{ReLU}\left(\cdot\right)$ activation). Besides, Transformers (Vaswani et al., 2017; Zhang et al., 2020) typically add some positional or structural encodings (either absolute or relative encodings) to incorporate some information about time, space, position, local information, or structure, etc. Furthermore, some normalization layers, *e.g.*, Layer Normalizations (*i.e.*, $\mathrm{LayerNorm}\left(\cdot\right)$), are still necessary for specific architectures of Transformer variants.

F.2 The Details of GTs and Ours

The architectures adopted by our method and many other GTs generally utilize the same attention mechanism as the vanilla version in Transformer (Vaswani et al., 2017), despite the generality of further extensions.

Yet, different from vanilla Transformer and some other GTs, we modify some basic modules as follows:

- We omit the FFN module in the reported student architecture. In our preliminary configurations, adding this module did not consistently improve performance under limited labels, while it introduced additional parameters. This is an implementation choice for the present experiments, not a claim that FFNs are unnecessary in graph Transformers generally.
- We add a nonlinear activation, *e.g.*, $\mathrm{ReLU}(\cdot)$, between consecutive attention layers. This supplies an explicit pointwise feature transformation in the absence of an FFN. Attention with input-dependent weights is already nonlinear; removing this activation would therefore not, by itself, make the model linear.
- We do not utilize any extra hand-craft or heuristics-based positional encoding, *e.g.*, distance or diffusion-based ones. The weights of our potential graphs provide a teacher-derived relative interaction prior. As discussed in Sec. L.3, this can be related to a pairwise attention bias, but is not an assertion that it reproduces every positional encoding.

Notably, our method can construct GTs different from Global Graph Transformers (*i.e.*, GGTs), which can be viewed as generalized LGTs, thus keeping sparsity and avoiding significant space bottlenecks.

F.3 The Details of Graph Construction in Our Stage 1

In stage 1 (Sec. 2.2), we build those prototypical graphs based on a proposed confidence-aware weighting schema and an efficient Maximum Inner Product Search (MIPS) tool, *i.e.*, *Faiss* (Johnson et al., 2019).

The main text introduces the ranking reduction but leaves the retrieval procedure implicit. We give the MIPS problem explicitly here and explain what acceleration does, and does not, follow from using Faiss. This distinction matters because sparse output graphs do not by themselves guarantee inexpensive graph construction.

Maximum inner-product search (**MIPS**) is to search some target data items, attempting to maximize the inner product between a query (vector or other abstract data item) and the data items to be retrieved. MIPS algorithms have already been widely applied to big data applications, including

recommendation and Computer Vision. Formally, it solves the following searching problem:

$$\mathrm{MIPS}\left(q;S\right)=\arg\max_{x\in S}\langle x,q\rangle, \tag{65}$$

where q is a query vector, and S is the data item set we consider. The problem can be generalized to the top-K selection, *i.e.*, generalizing $\arg\max\left(\cdot\right)$ to $\mathrm{ArgTop\text{-}K}\left(\cdot\right)$ (Eq. B.1).

Faiss supports efficient similarity search and selection on large vector collections (Johnson et al., 2019). In our reduction, each aspect supplies query vectors $[P_u^{(T)[k]}\|a]$ and database vectors $[\tau'\log P_v^{(T)[k]}\|\log S_v^{(T)[k]}]$. The retrieval step selects candidate node indices, after which Eq. 11 gives their actual weights and Eq. 12 adds the self-connection. Exact MIPS preserves the stated ranking; an approximate index trades retrieval accuracy for cost. The reduction itself does not specify an index-dependent complexity bound or establish its recall.

Furthermore, please check the survey paper (Matsui et al., 2018) for more related techniques.

F.4 Other Implementation Details

In our technical framework, We adopt N_H attention heads. We set both the search space of dropout rate (*i.e.*, *dropout*) and the attention dropout (*i.e.*, *attn dropout*) from $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$. We adopt learning rate lr_1 for inner optimization loop from $\{1e-2, 5e-3, 1e-3\}$ and learning rate lr_2 for outer optimization loop $\{1e-2, 5e-3, 1e-3, 5e-4\}$. Besides, we adopt the weight decay ratio (*i.e.*, *weight decay*) from $\{5e-3, 5e-4, 5e-5\}$. See also Sec. P.3 of Appx. for more details of their tuning and specific configurations. Additional values used by the selected configurations are listed separately in Tab. 16; the base ranges alone do not enumerate every reported setting.

G Algorithms and Pseudo-codes

G.1 Message Passing-based DRL

Recall the Message Passing-based DRL (MP-DRL) of Stage 1 in Sec. 2.2 of the main text. For clearer understanding and better reproducibility, here we give a specific implementation in Algorithm 3, which is the basic technique for the whole Stage 1.

G.2 Stage 1 (Disentangling Stage): Interaction-Space Distillation

Recall the detailed description of Stage 1 in Sec. 2.2 of the main text. To facilitate better reproducibility and understanding, here we give a detailed implementation in Algorithm 4, which is the key stage for graph interaction-space distillation.

G.3 Stage 2 (Fusing Stage): Adaptive Graph Searching

Recall the detailed description of Stage 2 in Sec. 2.3 of the main text. To facilitate better reproducibility and understanding, here we give a detailed implementation in Algorithm 5, which is the key stage for adaptive graph search via learnable prototypical graph fusion guided by Meta Self-Update mechanism.

Reading the alternating updates. Algorithm 5 specifies the rounds, student resets, inner fitting, monitor updates, and outer optimization. Its fusion assignment denotes the layer-wise operation in Eqs. 14–15; the fuser input is the representation used for the corresponding student forward pass. Fixing ψ does not mathematically fix the fused graph if that representation changes. Conversely, a cached-graph implementation must specify which representation is cached and when it is refreshed. The displayed outer gradient is a partial derivative with student parameters held fixed, rather than a full derivative through the inner training trajectory. These distinctions are relevant when matching the abstract pseudocode to an implementation.

G.4 Possible Stage 3: Recursively Reflective Surpassing

To encourage the student to further surpass its teacher after becoming mature. We give a detailed description of a possible high-level Stage 3 in Sec. J and some specific reasonable implementation.

Algorithm 3 Message Passing-based DRL (MP-DRL)

Input: adjacency matrix $A \in \{0,1\}^{n\times n}$ (or equivalently the first-order neighborhoods $\{N_u\}_{u\in V}$), node embeddings of layer $l \in [0, L)$, *i.e.*, $H^{(l)} \in \mathbb{R}^{n\times d}$ (with n the node num and d the embedding dim); a general MPNN f_{θ_T} containing MSG, UPD, and AGG.

Output: node embeddings of layer $l+1$, *i.e.*, $H^{(l+1)} \in \mathbb{R}^{n\times d}$

Hyper-Parameters:
the number of disentangled aspects: $K \in \mathbb{N}^+$,
the number of iterations in routing: T_0

Trainable-Parameters: $\mathrm{MSG}_k(\cdot)$, $\forall\, k \in [1, K]$; maybe $\mathrm{UPD}(\cdot)$ and $\mathrm{AGG}(\cdot)$

Note:
1) If we substitute the final $\mathrm{Concatenation}(\cdot)$ with $\mathrm{Sum}(\cdot)$, then it will support more number of aspects and avoid that each aspect would have too low dimension to encode node knowledge;
2) There is a trick to well adapt it to heterophilous graphs, *e.g.*, by adding one additional aspect $K+1$ or $\mathrm{Softmax}$ dim $K+1$ to absorb noisy components of probabilities $P^{[u]}_{k+1,v}$.

for each $u \in V$ **do**
  for each $k \in [1, K]$ **do**
    $z'_{k,u} \leftarrow \mathrm{MSG}_k\left(H_u^{(l)}\right) \in \mathbb{R}^{d/K}$ ▷ each node $u \in V$ sends a message $z'_{k,u}$ to each aspect space $\mathbb{S}_k$
    $z_{k,u} \leftarrow z'_{k,u} \Big/ \|z'_{k,u}\|_2$ ▷ optionally normalize the messages to unit norm
  end for
end for
for each $u \in V$ **do**
  for each $k \in [1, K]$ **do**
    $c_k^{[u]} \leftarrow z_{k,u} \in \mathbb{R}^{d/K}$
    ▷ for each center node u and its fixed first-order neighborhood N_u, initialize the cluster centroids $c_k^{[u]}$ of each $\mathbb{S}_k$. $c_k^{[u]}$ is c_k in the main text, since it is for fixed u therein
  end for
end for
for each $u \in V$ **do**
  for each $t \in [1, T_0]$ **do**
    for each $v \in N_u$ **do**
      $\left\{P^{[u]}_{k,v}\right\}_{k\in[1,K]} \leftarrow \mathrm{Softmax}_{k\in[1,K]}\left(z_{k,v}^T \cdot c_k^{[u]} \Big/ \tau\right)$
    end for
    for each $k \in [1, K]$ **do**
      $c_k^{[u]'} \leftarrow z_{k,u} + \sum_{v\in N_u} p^{([u])}_{k,v} \cdot z_{k,v}, \quad c_k^{[u]} \leftarrow c_k^{[u]'} \Big/ \|c_k^{[u]'}\|_2$ ▷ update the centroids
    end for
  end for
  for each $k \in [1, K]$ **do**
    $c_k^{[u]'} \leftarrow \mathrm{UPD}\left(z_{k,u}, \mathrm{AGG}\left(\{z_{k,v}\}_{v\in N_u}; \{p^{[u]}_{k,v}\}_{v\in N_u}\right)\right)$
    ▷ incorporating message-passing operators to improve the power of information aggregation in each $\mathbb{S}_k$
  end for
end for
return $H^{(l+1)} \leftarrow \mathrm{Concat}\left(\left\{c_k^{[u]'}\right\}_{u,k}\right) \in \mathbb{R}^{n\times d}$

Algorithm 4 Stage 1 (Disentangling Stage): Interaction-Space Distillation

Input: adjacency matrix $A \in \{0,1\}^{n\times n}$ (or equivalently the first-order neighborhoods $\{N_u\}_{u\in V}$), node features, *i.e.*, $X \in \mathbb{R}^{n\times f}$ (with n the node num and f the feature dim); a general MPNN f_{θ_T} containing MSG, UPD, and AGG.

Output: an interaction space $\mathbb{G}$ spanned by several prototypical graphs $\{G_k\}_{k\in[1,K]}$

Hyper-Parameters:

the number of disentangled aspects: $K \in \mathbb{N}^+$,

the number of iterations in routing: T_0,

$\alpha_{Dis}, \alpha_{Div} \geq 0$ are the trade-off hyper-parameters in the pre-training loss (Eq. 10),

$K', a, \tau' > 0$ are hyper-parameters in graph construction (Eq. 11 and Eq. 12) where K' denotes how many neighbors we select via MIPS. a and τ' are confidence discounting and numerical distribution controlling hyper-parameters, respectively,

the number of pre-training epoches $T^{(pre-training)}$,

the number of layers in f_{θ_T}, *i.e.*, L.

Trainable-Parameters: the MPNN teacher f_{θ_T} and $K+1$ learnable MLPs.

Note: DRL-based methods are not unique solutions for interaction-space distillation. Please check Sec. M for some alternative ideas, which can be taken as backbone techniques, too.

for each $t \in [1, T^{(pre-training)}]$ **do**

$\left\{c_k^{[u]\prime(L)}\right\}_{k\in[1,K]}, \left\{H_u^{(L)}\right\}_{u\in V}, \{z_{k,u}\}_{k\in[1,K]} \leftarrow L$-layer MP-DRL $(A, X, f_{\theta_T}; K, T_0)$

▷ obtain the last-layer disentangled embedding of each node in each aspect $\mathcal{A}_k$

▷ denote the messages by $Z_u = \{z_{k,u}\}_k \in \mathbb{R}^{K\times d/K}$

  for each $u \in V$ **do**

    $P_u^{(T)} \leftarrow \text{Softmax}\left(\text{MLP}\left(H_u^{(L)}\right)\right) \in \mathbb{R}^C$

    $S_u^{(T)} \leftarrow \text{Confidence}\left(P_u^{(T)}\right) = 1 - H\left(P_u^{(T)}\right) / \log C \in [0,1]$

    for each $k \in [1, K]$ **do**

      $P_u^{(T)[k]} \leftarrow \text{Softmax}\left(\text{MLP}\left(c_k^{[u]\prime(L)}\right)\right) \in \mathbb{R}^C$

    end for

  end for

  $L_{Sup}^{(T)} \leftarrow \frac{1}{|V_L|}\sum_{u\in V_L} L\left(P_u^{(T)}, y_u\right)$

  $L^{(Div)} \leftarrow \frac{1}{n}\sum_{u\in V} \text{MSE}\left(Z_u \cdot Z_u^T,\ I_K\right)$

  $S_{total}^{(T)} \leftarrow \sum_{u\in V} S_u^{(T)}$

  $L^{(Dis)} \leftarrow \frac{1}{K\cdot S_{total}^{(T)}} \cdot \sum_{k=1}^{K} \sum_{u\in V} S_u^{(T)}\ \text{KL}\left(\text{Detach}\left(P_u^{(T)}\right) \parallel P_u^{(T)[k]}\right)$

  $L_{Pre}^{(T)} \leftarrow L_{Sup}^{(T)} + \alpha_{Dis}\cdot L^{(Dis)} + \alpha_{Div}\cdot L^{(Div)}$

  Update all learnable parameters, *i.e.*, f_{θ_T} and MLPs via the gradients from minimizing $L_{Pre}^{(T)}$

end for

$\nu_u^{[k]} \leftarrow \left(P_u^{(T)[k]} \parallel a\right), \quad \mu_v^{[k]} \leftarrow \left(\tau' \cdot \log P_v^{(T)[k]} \parallel \log S_v^{(T)[k]}\right)$

Compute ArgTop-K$'_{v\in V}\, W^{(k)}(u,v) =$ ArgTop-K$'_{v\in V}\langle \nu_u^{[k]}, \mu_v^{[k]}\rangle$ for each $u \in V, k \in [1,K]$

Build $\{G_k\}_k$ according to Eq. 11 and Eq. 12

return $\{G_k\}_{k\in[1,K]}$

Algorithm 5 Stage 2 (Fusing Stage): Adaptive Graph Searching

Input: adjacency matrix $A \in \{0,1\}^{n\times n}$, node features, *i.e.*, $X \in \mathbb{R}^{n\times f}$; $\{P_u^{(T)}\}_u$ and $\{S_u^{(T)}\}_u$ from a teacher $\mathcal{T}$; the distilled interaction space $\mathbb{G} = \text{Span}\left(\{G_k\}_{k\in[1,K]}\right)$ from Stage 1.

Output: a trained student GT $\mathcal{S}$ and the searched potential graphs $\{G^{(p)}\}$

Hyper-Parameters:

$\beta_S, \gamma_0 \in [0,1]$, where γ_0 is initial agreement threshold controlling sensitivity in updating, and β_S is a confidence filtering threshold to set up monitoring data; $\eta,\ \xi \geq 0$ control the effects of the confidence of the supervision and that of the ensembling;

$\lambda_{Reg} > 0$, $\lambda^{[B]} \leq \lambda^{[E]}$, and $1 \leq t^{[B]} \leq t^{[E]} \leq T^{(in)}$ are hyper-parameters to control the trade-off parameters; $T^{(in)}$, $T^{(out)}$, and $T^{(r)}$ represent the number of inner steps, outer steps, and the alternative rounds.

Trainable-Parameters: the node-wise fusers $\left\{\text{Fuse}_\psi^{(l)}(\cdot)\right\}_{l\in[1,L]}$, the learnable parameters of the student GT (*e.g.*, some linear layers, attention mechanism $\mathbf{W}^{(Q)}, \mathbf{W}^{(K)}, \mathbf{W}^{(V)}$, etc).

Note: For brevity, here we aim to present the idea, and some specific formulations can be found in the main text (Sec. 2.3).

$\mathcal{D}_M = (\mathcal{V}_M, \mathcal{Y}_M) \leftarrow \left(V, \{P_u^{(T)}\}\right)$

$\mathcal{D}_{mt} = (\mathcal{V}_{mt}, \mathcal{Y}_{mt})$ with $\mathcal{Y}_{mt,u} \leftarrow \arg\max P_u^{(T)}$, $\mathcal{V}_{mt} \leftarrow \{u \in V | S_u^{(T)} \geq \beta_S\}$

$\gamma \leftarrow \gamma_0$

▷ initialize Monitor Data $\mathcal{D}_{mt}$, the estimated Meta Data $\mathcal{D}_M$, and the dynamic performance threshold γ

Initialize $\text{Fuse}_\psi^{(l)}(\cdot)$, $\forall\, l \in [1,L]$

for each $r \in [1, T^{(r)}]$ **do**

  Reinitialize or reset the student $\mathcal{S}$

  $\{G^{(p)}\} \leftarrow \text{Fuse}_\psi\left(\{G_k\}_k\right)$

  ▷ We omit the superscript l for simplicity.

  for each $t^{(in)} \in [1, T^{(in)}]$ **do**

    $\left\{P_u^{(S)}\right\}_u \leftarrow \mathcal{S}\left(A, X; \left\{G^{(p)}\right\}\right)$

    ▷ the forward pass of the student $\mathcal{S}$ for node predictions

    $\lambda_{Cur} \leftarrow \text{Cos-Schedule}\left(t^{(in)};\ t^{[B]}, t^{[E]}, \lambda^{[B]}, \lambda^{[E]}\right)$

    ▷ the varying curriculum coefficient in Eq. 26

    $L_{TE}^{(\mathcal{S})} \leftarrow L_{Sup}^{(\mathcal{S})} + \lambda_{Cur} \cdot L^{(Cur)} + \lambda_{Reg} \cdot L^{(Reg)}$

    ▷ check the components of the inner training loss $L_{TE}^{(\mathcal{S})}$ in Eq. 30, Eq. 27, Eq. 28

    Update the student $\mathcal{S}$ via the gradient of $L_{TE}^{(\mathcal{S})}$, *i.e.*, $\frac{\partial L_{TE}^{(\mathcal{S})}}{\partial \mathcal{S}}$

    if $\text{Acc}(\mathcal{S}, \mathcal{D}_{mt}) > \gamma$ **then**

      $\gamma \leftarrow \text{Acc}(\mathcal{S}, \mathcal{D}_{mt})$

      $\mathcal{Y}_M \leftarrow \text{Ens}\left(\mathcal{Y}_M, \{P_u^{(S)}\}_{u\in V}\right) = (\mathbf{1} - \boldsymbol{\lambda}) \odot \mathcal{Y}_M + \boldsymbol{\lambda} \odot \{P_u^{(S)}\}$

      ▷ check Eq. 24 for the ensembling weights $\boldsymbol{\lambda}$

    end if

  end for

  for each $t^{(out)} \in [1, T^{(out)}]$ **do**

    Forward pass and calculate $\widehat{L_{GE}}\left(\mathcal{S};\ \{G^{(p)}\}\right)$ ▷ check Eq. 16 in Sec. 2.3 for $\widehat{L_{GE}}$

    Update the fusers $\left\{\text{Fuse}_\psi^{(l)}(\cdot)\right\}_{l\in[1,L]}$ via the gradient of $\widehat{L_{GE}}$, *i.e.*, $\frac{\partial \widehat{L_{GE}}}{\partial \left\{\text{Fuse}_\psi^{(l)}\right\}} = \frac{\partial \widehat{L_{GE}}}{\partial \{G^{(p)}\}} \cdot \frac{\partial \{G^{(p)}\}}{\partial \left\{\text{Fuse}_\psi^{(l)}\right\}}$

  end for

end for

return S and $\{G^{(p)}\}$

Algorithm 6 Possible high-level Stage 3 (Surpassing Stage): Recursively Reflective Surpassing

Input: a teacher $\mathcal{T}$

Output: a good student $\mathcal{S}^{\dagger}$ surpassing its teacher (with an explicit methodology)

Hyper-Parameters:

$T^{(Substage)}$, the number of contained sub-stages

Trainable-Parameters: the student $\mathcal{S}$

Note: For brevity, here we aim to present the *high-level idea*, and some *specific implementations* can be found in Sec. J. Some RNN-related improvements can be applied among those sub-stages, *e.g.*, introduce some additional differential mechanisms.

$\mathbb{K}_0 = \mathbb{K}^{[\mathcal{T}]} = \text{Distill}\left(\mathcal{T}\right)$

for each $i \in \left[0, T^{(Substage)}\right)$ **do**

  $\mathcal{S}_{i+1} \leftarrow \text{Guide}^{(i)}\left(\mathcal{S}_i;\ \mathbb{K}_i\right)$

  $\mathbb{K}_{i+1} \leftarrow \text{Reflect}^{(i)}\left(\mathbb{K}_i;\ \mathcal{S}_{i+1}\right)$

end for

$\mathcal{S}^{\dagger} \leftarrow \mathcal{S}_{T^{(Substage)}}$

return $\mathcal{S}^{\dagger}$

To facilitate a better understanding, here we present Algorithm 6, *i.e.*, the Recursively Reflective Surpassing schema.

H Complexity Analysis

We distinguish teacher pre-training, one-time prototype construction, and student training, since these stages have different costs. Let n and m denote the numbers of nodes and observed edges, K the number of aspects, K' the number of retrieved neighbors per aspect, and N_H the number of attention heads. We use the equal-width notation d/K for aspect messages and a conservative width-d bound for feature transformations. The analysis below concerns arithmetic and storage orders; reported per-epoch times are not measurements of the complete pipeline.

Teacher pre-training. A teacher layer transforms node features and performs T_0 routing iterations over the observed neighborhoods. Retaining the separate feature and routing terms gives the following upper bound:

$$\begin{aligned} &\mathcal{O}\big(L\big(nd^2 + T_0\big((K+1)md + K(m+n)d\big)\big)\big) \\ &= \mathcal{O}\big(Lnd^2 + LT_0K(m+n)d\big) \\ &= \mathcal{O}(Lmd(d + T_0K)) \quad \text{when } m \geq n. \end{aligned} \tag{66}$$

The last expression reduces to $\mathcal{O}(Lmd^2)$ only if $T_0K = \mathcal{O}(d)$. It should not be used without this qualification when the number of aspects or routing iterations grows. Storing the layer representations requires $\mathcal{O}(Lnd)$, with additional storage for routing assignments, teacher parameters, and the graph. For the aspect-specific transformations, the equal-width parameter term can scale as Ld^2/K; other message-passing components retain their own parameter costs.

Prototype construction. Each aspect ranks pairs of C-dimensional predictions, then retains at most K' other nodes and a self-connection per query. Exhaustive scoring and linear-time top-K' selection therefore give

$$\begin{aligned} \mathcal{T}_{\text{build}} &= \mathcal{O}(Kn^2C + KnK'), \\ \mathcal{M}_{\text{prototypes}} &= \mathcal{O}\big(Kn(K'+1)\big). \end{aligned} \tag{67}$$

MIPS permits accelerated retrieval without storing the full $n \times n$ score matrix. Its actual construction and query costs depend on the index, approximation level, and device; the ranking identity alone does not imply an $n \log n$ guarantee. Prototype storage remains linear in n when K and K' are fixed.

Student fitting and fuser updates. Let $E_\cup = |\bigcup_k E_k|$, counting directed candidate entries. Since fusion does not add edges, $E_\cup \le Kn(K'+1)$. Sparse attention then has an interaction cost $\mathcal{O}(LN_H E_\cup d)$, in addition to feature projections and node-wise fuser evaluation. For width-d projections, these can be bounded by $\mathcal{O}(Lnd^2 + LnKd)$. Graph storage is $\mathcal{O}(E_\cup)$, while activations and head-specific edge scores add their corresponding layer and head factors. The fused neighborhood must not be counted as only K' neighbors unless an additional pruning operation is applied.

The implicit sparsity loss also compares supported neighbors with a global attention normalizer. Its random-feature evaluation is described in Sec. K: once features of width r are available, the normalizer costs $\mathcal{O}(nr)$ per head, rather than materializing all n^2 pairs. Producing these features and evaluating the supported edges have separate costs. Finally, the complete training budget includes $T^{(pre\text{-}training)}$ teacher epochs, one graph-construction pass, and $T^{(r)}(T^{(in)} + T^{(out)})$ alternating updates, together with monitor evaluations. The possible Stage 3 adds its own retraining and graph-update costs and is not included in the reported two-stage analysis.

I Preliminaries and More Related Work

This section collects the preliminaries moved from the main text and supplements the literature review needed to place our interaction-space perspective in context.

I.1 Problem Setting and Notation

A graph $G = (V, E)$ contains n nodes $V = \{v_i\}$ and m edges $E = \{e_{i,j}\}$ with feature and adjacency matrices $X \in \mathbb{R}^{n\times f}$ and $A \in \{0,1\}^{n\times n}$, where $x_i = X_{i,:} \in \mathbb{R}^f$ is the feature of v_i and $A_{ij} = 1$ iff $(v_i, v_j) \in E$. Semi-supervised node classification predicts the labels of unlabeled nodes from a limited labeled set $V_L \subset V$, where $y_i \in [1, C]$ denotes the label of node v_i.

I.2 Graph Neural Networks

An L-layer GCN (Kipf & Welling, 2017) stacks L aggregations, linear maps, and nonlinear activations $\sigma(\cdot)$ (*e.g.*, ReLU or Softmax):

$$H^{(0)} = X, \qquad H^{(l+1)} = \sigma\Big(\hat{A}H^{(l)}W^{(l)}\Big) \in \mathbb{R}^{n\times d}, \quad l \in [0, L).$$

where $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, $\tilde{A} = A + I$, and $\tilde{D}_{i,i} = \sum_{j=1}^{n} \tilde{A}_{i,j}$. $H^{(l)} \in \mathbb{R}^{n\times d}$ and $W^{(l)} \in \mathbb{R}^{d\times d}$ denote the layer-l node representations and trainable parameters. GAT (Veličković et al., 2018) further weighs observed neighbors by attention. Their information flow follows the observed graph, so distant information requires repeated propagation or augmented neighborhoods; deep GNNs can extend the receptive field but may encounter over-smoothing or over-squashing (Chen et al., 2020; Topping et al., 2022).

SGC (Wu et al., 2019) drops nonlinearities to simplify GCN. FastGCN (Chen et al., 2018a) and Cluster-GCN (Chiang et al., 2019) scale to large graphs via sampling strategies. GIN (Xu et al., 2019) uses an ϵ coefficient to distinguish the central node from its neighbors, supporting expressiveness analyses through the 1-WL framework. The message-passing formulation (Gilmer et al., 2017) describes how nodes exchange and aggregate information. Architectures such as GCN, SGC, GAT, GIN, and GraphSAGE differ in their aggregation and feature transformations (Kipf & Welling, 2017; Wu et al., 2019; Veličković et al., 2018; Xu et al., 2019; Hamilton et al., 2017). Their expressive properties have been studied from several perspectives (Sato, 2020; Geerts & Reutter, 2022). Extending propagation through depth can improve access to distant information (Chen et al., 2022c; Li et al., 2019), while also making over-smoothing and over-squashing relevant concerns (Chen et al., 2020; Topping et al., 2022). This background motivates more flexible interactions, but does not imply that every GT is more expressive or more accurate than every message-passing GNN.

I.3 Graph Transformers

GTs adapt the attention mechanism of Transformers (Vaswani et al., 2017; Kenton & Toutanova, 2019) to graph-structured data (Min et al., 2022). Their interaction structures range from local neighborhoods to global or learned connections. GraphBert (Zhang et al., 2020) uses diffusion-based

sampling, while Gophormer (Zhao et al., 2021) restricts attention to ego-graphs. ANS-GT (Zhang et al., 2022) combines KNN, PPR, and one-/two-hop neighborhoods using a multi-armed bandit. NodeFormer (Wu et al., 2022) learns latent interactions through kernel-based differentiable sampling, and DET (Guo et al., 2022) learns a similarity metric by contrasting neighbors with sampled distant nodes. Scalable alternatives include SGFormer, Exphormer, and Polynormer (Wu et al., 2024; Shirzad et al., 2023; Deng et al., 2024); CoBFormer studies the over-globalizing problem (Xing et al., 2024). CAD (Huang et al., 2024) studies attention distillation, and M^{3}Dphormer (Xing et al., 2025) integrates hierarchical masks. Our focus is on distilling the candidate family itself from task-aligned teacher views, rather than directly matching predictions, embeddings, or a fixed relation target (Yang et al., 2020; Joshi et al., 2022; Liu et al., 2022).

GT (Dwivedi & Bresson, 2020) and HGT (Hu et al., 2020b) conduct only neighbor interactions while EGT (Hussain et al., 2022), SAT (Chen et al., 2022a), and Graphormer (Ying et al., 2021) prioritize global interactions with sophisticated absolute or relative positional encodings (PEs, *e.g.*, Laplacian-, distance-, diffusion-based encodings). Some works combine local and global information via, *e.g.*, adding (Rampášek et al., 2022), stacking (GraphTrans (Wu et al., 2021), Grover (Rong et al., 2020)), attention trade-offs (SAN (Kreuzer et al., 2021)), and GNN-based PE designs (GiT (Mialon et al., 2021)). Kernel approximations such as Performers (Choromanski et al., 2021) provide another route to efficient global attention, complementing explicit restrictions on interaction neighborhoods.

SGFormer (Wu et al., 2024) reduces computational overhead and avoids complex pre-processing for large-graph representation learning. Exphormer (Shirzad et al., 2023) uses sparse attention with global nodes and expander graphs to achieve linear computational cost. GOAT (Kong et al., 2023) is a scalable global graph transformer for large-scale node classification on homophilous and heterophilous graphs. GTs have also been applied to molecules (Jiang et al., 2023; Young et al., 2024), text (Wang et al., 2020b; 2023), and images (Nguyen et al., 2021; Yuan et al., 2023), alongside graph-level prediction (Xie et al., 2022). Their architectural development parallels attention-based models in vision (Wu et al., 2020a; Han et al.; Liu et al., 2021). For node classification, attention guidance from a local teacher is studied by CAD (Huang et al., 2024), while M^{3}Dphormer (Xing et al., 2025) learns to integrate hierarchical masks. These are relevant precedents for teacher guidance and multiple interaction patterns. Our distinction concerns constructing the candidate family from task-aligned teacher views and learning a student-specific realization within it, rather than introducing adaptive masks or distillation in isolation. Sec. L.6 provides a focused comparison of representative adaptive-structure methods and clarifies this distinction further.

I.4 Knowledge Distillation (KD) and Curriculum Learning (CL)

KD (Hinton et al., 2015; Gou et al., 2021) typically transfers knowledge from large teachers to small students, facilitating model compression. Recent works have explored it in the graph domain. As a pioneer, LSP (Yang et al., 2020) preserves structural knowledge in GNNs by aligning predictions of teachers and students, while G-CRD (Joshi et al., 2022) aligns node embeddings. For heterogeneous graphs, HIRE (Liu et al., 2022) preserves relational knowledge by distilling correlations between a node prediction and all node types. Rather than only aligning numerical targets, we use task-aligned teacher views to define a constrained family of interaction structures. This permits *weak-to-strong* guidance: the student can use the teacher's structural priors while learning its own candidate, without treating teacher performance as an upper bound. Motivated by human learning, CL (Wang et al., 2022b; Soviany et al., 2022) expects a model to gradually learn simple-to-complex knowledge by presenting examples (Bengio et al., 2009) or designing tasks (Caubrière et al., 2019) with easy-to-hard orders, applied in many domains (Liu et al., 2020a; Sachan & Xing, 2016; Xu et al., 2020; Wang et al., 2020a; Narvekar et al., 2020; Zhou et al., 2020). For graph classifications, CurGraph (Wang et al., 2021) scores the difficulties of graphs via neural density estimation. CuCo (Chu et al., 2021) enhances graph contrastive learning by introducing similarity-based increasingly more challenging negative samples. We still improve the generalizability of students via CL, yet with strict-to-loose constraints from teachers, facilitating near-to-far local refinement for a target graph candidate.

I.5 Disentangled Representation Learning

Disentangled Representation Learning (Wang et al., 2022a) has received considerable attention, and has demonstrated rich effects in the field. The method in Retrieval-based disentangled representation

learning with natural language supervision (Zhou et al., 2023) uses natural language proxies to enhance disentangled representation learning, showing significant performance improvements in retrieval tasks across multiple datasets. In IB-GAN (an Information Bottleneck-based Generative Adversarial Network) (Jeon et al., 2021) for unsupervised separated representation learning by constraining the mutual information between the inputs and the generated outputs, the separation and interpretability of the latent space is achieved.

Variational Autoencoders (VAEs) have been extensively studied in the context of disentangled representation learning. The β-VAE (Higgins et al., 2017) introduced a modified objective function that balances reconstruction accuracy and latent space disentanglement by adjusting the weight of the KL-divergence term. This model has been foundational in exploring the trade-offs between reconstruction fidelity and the quality of the disentangled representations. Follow-up works such as FactorVAE (Kim & Mnih, 2018) and β-TCVAE (Chen et al., 2018b) have further refined these ideas by introducing additional constraints and modifications to the VAE framework to enhance disentanglement. These methods typically focus on unsupervised learning and aim to improve the interpretability and usability of the learned latent spaces in various generative and analytical tasks.

InfoGAN (Chen et al., 2016) introduced an information-theoretic extension to GANs that encourages disentanglement by maximizing the mutual information between a subset of latent variables and the observation. Additionally, the JointVAE (Dupont, 2018) model combines continuous and discrete latent variables to learn disentangled representations, which is particularly useful for tasks that involve both types of underlying factors. Another notable approach is the TC-VAE (Chen et al., 2018b), which incorporates a total correlation term in the VAE objective to promote independence among the learned factors. Furthermore, the GroupVAE (Hosoya, 2018), which enforces a group-wise disentanglement by leveraging group theory, shows promising results in learning structured and interpretable representations.

I.6 Meta-Learning Techniques

Meta-learning, also known as learning to learn, has gained significant attention in recent years due to its potential to improve the efficiency and performance of machine learning models by leveraging prior experiences. One of the pioneering works in meta-learning is the Model-Agnostic Meta-Learning (MAML) algorithm proposed by Finn et al. (2017). MAML aims to find a good initialization for model parameters such that a small number of gradient steps with a small amount of data from a new task will lead to good generalization performance. This model-agnostic approach has been widely adopted and extended in various applications. Few-shot learning, a subfield of meta-learning, focuses on the ability to learn from a limited number of examples. Vinyals et al. (2016) introduced Matching Networks, which use an attention mechanism to compare a small support set of labeled examples to an unlabeled example. Prototypical Networks, introduced by Snell et al. (2017), further improved few-shot learning by learning a metric space where classification can be performed by computing distances to prototype representations of each class.

Meta-learning has also been successfully applied to reinforcement learning (RL). The RL^2 algorithm proposed by Duan et al. (2016) learns a reinforcement learning algorithm that can adapt to new tasks through experience. Similarly, Wang et al. (2016) introduced a method to learn an RL algorithm using recurrent neural networks, demonstrating improved adaptation to new tasks. Gradient-based meta-learning methods have shown significant promise in various domains. Ravi & Larochelle (2016) proposed an optimization-based framework for few-shot learning, where the learning algorithm itself is learned. This approach uses an LSTM-based meta-learner to train the base learner's optimization process.

Recent work has also explored meta-learning for neural architecture search (NAS). Zoph & Le (2016) introduced an approach where a recurrent neural network is trained to generate neural network architectures, significantly reducing the computational cost of NAS. Meta-learning has found applications in various domains, including natural language processing, computer vision, and healthcare. For instance, Chen et al. (2019) provided a comprehensive analysis of meta-learning methods for few-shot image classification, identifying key challenges and proposing solutions to improve performance.

J A POSSIBLE IMPLEMENTATION OF STAGE 3 (SURPASSING STAGE)

When gradually maturing, are students likely to surpass rather than mimic their teachers by, *e.g.*, solving some essential issues? To alleviate intrinsic knowledge imbalance in KD, via a *reflective* mechanism, Stage 3 (**Surpassing Stage**) encourages mature students to complement or update the incomplete and inaccurate teacher knowledge with confident self-knowledge, therefore obtaining a better potential graph beyond the initial interaction space. By a further *recursive reflection* schema, it theoretically generalizes both typical KD and iterative self-distillation (Tejankar et al., 2021), providing an idea for further development of sophisticated KD techniques.

J.1 POSSIBLE STAGE 3 (SURPASSING STAGE): RECURSIVELY REFLECTIVE SURPASSING

In this supplementary stage, we encourage a mature student $\mathcal{S}$ to further surpass its teacher $\mathcal{T}$ rather than simply mimic. We first present a general abstract reflective knowledge distillation schema where $\mathcal{S}$ reflects on and updates the given knowledge $\mathbb{K}^{[\mathcal{T}]}$ distilled from $\mathcal{T}$, then provide a recursive multi-substage extension, and finally show how it can be technically incorporated into our framework. A well-known typical knowledge distillation schema is conducted as follows:

$$\mathcal{S}^{\dagger} = \text{Guide}\left(\mathcal{S}_0;\ \mathbb{K}^{[\mathcal{T}]}\right), \tag{68}$$

$$\mathbb{K}^{[\mathcal{T}]} = \text{Distill}\left(\mathcal{T}\right), \tag{69}$$

where knowledge $\mathbb{K}^{[\mathcal{T}]}$ is first distilled from $\mathcal{T}$, and then guides the training or designs of the initial student $\mathcal{S}_0$ to develop a final satisfactory student $\mathcal{S}^{\dagger}$. If $\mathbb{K}^{[\mathcal{T}]}$ has some unignorable deficiency, then $\mathcal{S}^{\dagger}$ would unavoidably suffer from some issues (*e.g.*, *knowledge imbalance* here). Existing techniques (Hinton et al., 2015) mainly focus on some hand-craft filtration or discounting, which cannot thoroughly solve those issues and may intensify knowledge imbalance. To develop a sophisticated model-adaptive strategy, we encourage students to reflect and update $\mathbb{K}^{[\mathcal{T}]}$ by themselves via their dynamical confident self-knowledge during the guidance. To present the intrinsic motivation, we discuss the insights in some common KD cases. In cases **(A)** (*i.e.*, *weak-to-strong* distillation), where the students are smarter than teachers yet degrade due to possible task-inadeptness, they would gradually become mature, show their potential under the guidance of $\mathbb{K}^{\mathcal{T}}$, and thus self-correct or self-supplement some parts therein. In cases **(B)** (*i.e.*, *strong-to-weak* distillation), where the students may be much smaller than teachers facilitating model compression, the knowledge $\mathbb{K}^{\mathcal{T}}$ from $\mathcal{T}$ might not be suitable or easy-understand for $\mathcal{S}$, students may need further self-adaptation or self-simplification via, *e.g.*, a self-curriculum, self-filtering, or self-smoothing strategy, which can encapsulate some dynamic temperature-based KD techniques (Li et al., 2023; Wei & Bai, 2024; Li et al., 2021; Jafari et al., 2021). Formally, this idea can be high-levelly stated as follows:

$$\mathbb{K}^{[\mathcal{T}]} = \text{Distill}\left(\mathcal{T}\right), \tag{70}$$

$$\mathcal{S}_{mature} = \text{Guide}\left(\mathcal{S}_0;\ \mathbb{K}^{[\mathcal{T}]}\right), \tag{71}$$

$$\mathbb{K} = \text{Reflect}\left(\mathbb{K}^{[\mathcal{T}]};\ \mathcal{S}_{mature}\right), \tag{72}$$

$$\mathcal{S}^{\dagger} = \text{Guide}\left(\mathcal{S}_{mature};\ \mathbb{K}\right), \tag{73}$$

where under the guidance of $\mathbb{K}^{[\mathcal{T}]}$, the student $\mathcal{S}_0$ becomes mature into $\mathcal{S}_{mature}$, and then reflect on $\mathbb{K}^{[\mathcal{T}]}$ to obtain new knowledge $\mathbb{K}$, which can further guide the student to develop into a final $\mathcal{S}^{\dagger}$. One can naturally generalize it into a multi-substage recursive version (called *Recursively Reflective Distillation*, *i.e.*, *RRD*):

$$\mathbb{K}_0 = \mathbb{K}^{[\mathcal{T}]} = \text{Distill}\left(\mathcal{T}\right) \tag{74}$$

$$\forall\, i\ \in \left[0, T^{(Substage)}\right):\tag{75}$$

$$\mathcal{S}_{i+1} = \text{Guide}^{(i)}\left(\mathcal{S}_i;\ \mathbb{K}_i\right), \tag{76}$$

$$\mathbb{K}_{i+1} = \text{Reflect}^{(i)}\left(\mathbb{K}_i;\ \mathcal{S}_{i+1}\right), \tag{77}$$

$$\mathcal{S}^{\dagger} = \mathcal{S}_{T^{(Substage)}}, \tag{78}$$

where a student $\mathcal{S}$ recursively update the knowledge $\mathbb{K}_i$ initialized at $\mathbb{K}^{[\mathcal{T}]}$ and $\mathcal{S}_i$ itself. Please check Algo. 6 in Appx. for details.

Notably, this standard recursive schema is quite similar to RNNs (*i.e.*, Recurrent Neural Networks), which means many extensions or improvements from the RNN domain would be transferred to further refine it, *e.g.*, by introducing differentially updating strategies, injecting substage-wise inputs, incorporating memory-based mechanisms, and alleviating catastrophic forgetting issues (Kirkpatrick et al., 2017; Draxler et al., 2018; Hayes et al., 2020). For example, in some latter substages, students can recognize what knowledge or which form they further need to improve themselves more effectively, which may benefit the dynamic modification of the $\mathrm{Distill}\left(\cdot\right)$ operator via a differentially jointly learning mechanism.

To provide a practical technical implementation, we treat $\mathcal{S}_0$, $\mathbb{K}_0$, and $\mathcal{S}_1$ as a randomly initialized student, the distilled interaction space, and the optimized student in Stage 2, respectively. Then, we set $\mathbb{K}_1$ as the searched potential graph $G^{(p)}$ merged with a K-Nearest Neighbor (KNN) graph built from $\mathcal{S}_1$'s understandings (*e.g.*, its node embeddings, attention coefficients, or aggregating weights $G^{(agg)} = \mathrm{RowNorm}\left(G^{(p)} \odot \{\alpha_{i,j}\}_{i,j}\right)$). After that, we assume that all operators $\mathrm{Reflect}^{(i)}$ $(i \geq 2)$, updating $\mathbb{K}_i$ into $\mathbb{K}_{i+1}$, retain the graph structure of $\mathbb{K}_i$ intact and only modify the graph weights by injecting attention coefficients of $\mathcal{S}_{i+1}$ into $\mathbb{K}_i$ [4] with row-wise normalization [5]. All operators $\mathrm{Guide}^{(i)}$ $(i \geq 2)$, updating $\mathcal{S}_i$ into $\mathcal{S}_{i+1}$, reinitialize the attention-related learnable parameters (*i.e.*, $\mathbf{W}^{(Q)}, \mathbf{W}^{(K)}$) [6] of $\mathcal{S}_i$ and re-train the student to obtain $\mathcal{S}_{i+1}$.

We will detailedly investigate this stage in future work and explore more effective technical implementations.

K Details of the Implicit Sparsity Regularization

Recall the definition of $L^{(Reg)}$ in Eq. 27:

$$L^{(Reg)} = -\frac{1}{L} \cdot \sum_{l=1}^{L} \frac{1}{C^{(l)}} \sum_{u \in V} \Gamma_u^{(l)} \cdot \log \left(\frac{\sum_{v \in V: G_{u,v}^{(p)(l)} > 0} \exp\left(\mathbf{Q}_u^{(l)} \cdot \mathbf{K}_v^{(l)T} / \sqrt{d}\right)}{\sum_{v \in V} \exp\left(\mathbf{Q}_u^{(l)} \cdot \mathbf{K}_v^{(l)T} / \sqrt{d}\right)} \right). \tag{79}$$

Note that to speed up and save space for the calculation of Eq. 79, we need to efficiently obtain the row sums of the unnormalized attention matrix, *i.e.*:

$$\forall u \in V, \quad \mathbf{S}_u = \sum_{v \in V} \exp\left(\mathbf{Q}_u^{(l)} \cdot \mathbf{K}_v^{(l)T} / \sqrt{d}\right). \tag{80}$$

One natural idea to achieve this end is to disentangle the variable u and v, and reformulate the $\exp\left(x^T y\right)$ into some linear binary operation $\circ$ that:

$$\exp\left(x^T y\right) \approx x' \circ y', \tag{81}$$

which means that the sum operation can be applied to the variable y only (and thus can be pre-processed efficiently) since the $\mathrm{Sum}\left(\cdot\right)$ and $\circ$ are exchangeable due to the linearity of $\circ$. An inner product in Euclidean space provides a suitable bilinear operation for this purpose. Therefore, our key challenge is to disentangle the $\exp\left(x^T \cdot y\right)$ kernel into:

$$\exp\left(x^T \cdot y\right) \approx \langle x', y' \rangle, \quad x', y' \in \mathbb{R}^r. \tag{82}$$

Random-feature methods (Likhosherstov et al., 2022; 2024) provide such an approximation by constructing feature maps from shared random samples. They represent the following $\exp\left(\cdot\right)$ kernel functions:

$$K^{(\alpha)}\left(x, y\right) = \exp\left(\alpha \cdot \|x\|_2^2 + x^T \cdot y + \alpha \cdot \|y\|_2^2\right), \tag{83}$$

[4] I.e., $\mathbb{K}_i$ is the sparse relative positional encodings $G^{(p)}$.

[5] We let $G^{(p)} \leftarrow \text{Keep-Row-Sum}\left(G^{(p)} \odot \{\alpha_{i,j}\}\right)$ and $\text{Keep-Row-Sum}\left(\cdot\right)$ is to divide constants to keep the sum of each row intact to avoid numerical issues.

[6] We keep all other learnable parameters unchanged to keep the transition smooth to carefully avoid performance degradation.

where letting $\alpha = 0$ is exactly what we want. It is based on the following formulation:

$$K^{(\alpha)}(x, y) = \mathbb{E}_\nu \left[f^{(1)}(\omega, x) \cdot f^{(2)}(\omega, y) \right]. \tag{84}$$

A finite average over shared samples gives Eq. 82 approximately. The expectation identity is exact, whereas a finite feature map introduces sampling error. We use r for the number of random features, keeping it distinct from the input width d. Following Likhosherstov et al. (2022), the sampling distribution and feature functions take the form:

$$\Omega_{GE} = \mathbb{R}^d, \tag{85}$$

$$\nu_{GE} = \mathcal{N}(0, 1)^d, \tag{86}$$

$$\begin{aligned} f^{(1)}_{GE}(\omega, \mathbf{x}) &= f^{(2)}_{GE}(\omega, \mathbf{x}) \\ &= D \exp(A\|\omega\|^2 + B\omega^\top \mathbf{x} + C\|\mathbf{x}\|^2/2), \end{aligned} \tag{87}$$

where ω is sampled from Ω_{GE} with distribution ν_{GE}, and $x'_{i,j} = r^{-1/2} f^{(1)}_{GE}(\omega^{(j)}, x_i)$ and $y'_{i,j} = r^{-1/2} f^{(2)}_{GE}(\omega^{(j)}, y_i)$ for $j \in [1, r]$. Besides, the parameters A, B, C, D satisfy the following construction:

$$1 - 8A > 0, \quad B = \sqrt{1 - 4A}, \tag{88}$$

$$C = -1, \quad D = (1 - 4A)^{d/4}, \tag{89}$$

Here the construction is specialized to $\alpha = 0$. With the written exponent $C\|x\|^2/2$, the required coefficient is $C = -1$: Gaussian integration contributes $\|x + y\|^2/2$, and the feature factors cancel its two self-norm terms. For the general kernel $K^{(\alpha)}$, the coefficient becomes $C = 2\alpha - 1$. The letters A, B, C, D in this feature map are scalar coefficients and are unrelated to the graph adjacency or the number of classes. For a fixed input pair, the variance of Eq. 84 can be minimized with respect to A using the following expression:

$$\mathrm{Var}_{\nu_{GE}} f^{(1)}_{GE}(\omega, \mathbf{x}) f^{(2)}_{GE}(\omega, \mathbf{y}) = e^{\mathcal{L}_{GE}(A, \mathbf{x}, \mathbf{y})} - K^{(0)}(\mathbf{x}, \mathbf{y})^2, \tag{90}$$

$$\begin{aligned} \mathcal{L}_{GE}(A, \mathbf{x}, \mathbf{y}) = d \log \left(\frac{1 - 4A}{\sqrt{1 - 8A}} \right) \\ + \frac{2(1 - 4A)}{1 - 8A} \|\mathbf{x} + \mathbf{y}\|^2 - \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2. \end{aligned} \tag{91}$$

In other words, minimizing $\mathrm{Var}_{\nu_{GE}} f^{(1)}_{GE}(\omega, \mathbf{x}) f^{(2)}_{GE}(\omega, \mathbf{y})$ is equivalent to minimizing $\mathcal{L}_{GE}$ in Eq. 91. Likhosherstov et al. (2022) derives the optimal solution A^* of the optimization:

$$A^* = \frac{1 - \frac{1}{\rho^*}}{8}, \tag{92}$$

$$\rho^* = \frac{\sqrt{(2\|x + y\|_2^2 + d)^2 + 8d\|x + y\|_2^2} - 2\|x + y\|_2^2 - d}{4\|x + y\|_2^2}. \tag{93}$$

For a fixed pair (x, y) with $\|x + y\|_2^2 > 0$, these expressions determine A^* and hence B^*, C^*, D^*. The zero-norm case is interpreted by its limit, giving $\rho^* = 1$ and $A^* = 0$. This is a pairwise variance calculation. The efficient global summation below instead requires a shared feature map, so its coefficients and random samples must be common to all nodes; a separately optimized A^* for every pair cannot be inserted into that factorization. The pairwise result describes the variance dependence, not an independently implementable per-edge optimization in the linear-time normalizer.

Finally, after obtaining $x'_u = x' \left(\mathbf{Q}_u^{(l)} / \sqrt[4]{d} \right), y'_v = y' \left(\mathbf{K}_v^{(l)T} / \sqrt[4]{d} \right)$ in Eq. 82 using r shared random samples, incorporating them into Eq. 80, we can obtain:

$$\forall u \in V, \quad \mathbf{S}_u = \sum_{v \in V} \exp \left(\mathbf{Q}_u^{(l)} \cdot \mathbf{K}_v^{(l)T} / \sqrt{d} \right) \tag{94}$$

$$\approx \sum_{v \in V} \langle x'_u, y'_v \rangle \tag{95}$$

$$= \langle x'_u, \sum_{v \in V} y'_v \rangle. \tag{96}$$

Once the random features have been formed, the shared sum $\sum_v y'_v$ and all node normalizers require $\mathcal{O}(nr)$ time and storage per head. Forming dense random features generally adds $\mathcal{O}(ndr)$ work, and the supported-edge numerator in $L^{(Reg)}$ must also be evaluated. Thus the factorization avoids a dense all-pair normalizer, but $\mathcal{O}(nr)$ is the cost of this summation step, not automatically the cost of the entire regularizer. The first transition introduces the random-feature approximation; the second follows from distributivity of the inner product over the shared sum.

L More Discussion

L.1 A Game-Theoretical Interpretation of Stage 1

The optional diversity constraint encourages different aspects $\{\mathcal{A}_k\}$ to avoid redundant message directions. This resembles the separation of components in Principal Component Analysis (PCA) (Dunteman, 1989; Kherif & Latypova, 2020), for which game-theoretic formulations have been developed (Gemp et al., 2020). The analogy helps explain the two forces in Stage 1: aspects contribute jointly to a holistic task predictor, while the diversity term encourages them to preserve differences.

There is also an important distinction. PCA seeks a representation space $\mathbb{U} = \mathrm{Span}(\{U_k\})$ that preserves variation in X, whereas we construct $\mathbb{G} = \mathrm{Span}(\{G_k\})$ to support prediction of Y. In information-theoretic language, the intended goals concern $I(\mathbb{G}; Y)$ and the individual $I(G_j; Y)$, rather than only information about the input itself. These expressions describe the motivation; our loss is the supervised and distillation objective, not a direct mutual-information estimator. Likewise, rotations of principal components (Richman, 1986) offer an analogy for redistributing information across aspects, not an equivalence to the actual routing or MACD updates.

A game interpretation could model the holistic objective as cooperation and the diversity penalty as competition. Establishing such a model would require explicit player utilities, feasible sets, and an equilibrium analysis. We do not claim that these have been derived here, or that an equilibrium guarantees useful prototypes. The discussion instead clarifies why task relevance and diversity can pull the aspect representations in different directions, and why improving AAA alone does not prove structural complementarity.

L.2 The Roles of Curriculum and Regularization in Stage 2

We provide two additional improvements for our Stage 2 in Sec. 2.3, *i.e.*, Curriculum-based local refinement and implicit sparsity regularization. Although the ablation studies in Sec. 3 empirically show their positive contributions, we explain their complementary roles in fitting a student that can make effective use of the selected structure. The comprehensive aim of Stage 2 is to adaptively search a graph with some reasonable training mechanism. As described in our basic motivation (Sec. 2.1), we aim to reduce detrimental structure–attention co-adaptation by using a separate, generalization-oriented surrogate for structural selection.

Therefore, the above process contains ***two key parts***:

1. defining a *search* operation (**how to search**);
2. defining the *goodness* of a graph (**how good a graph is**).

The proposed *node-wise fuser* is to solve challenge **(1)** and we still need to solve challenge **(2)**, *i.e.*, training the fuser.

The Meta Self-Update mechanism presented in Sec. 2.3 has defined a self-estimated generalization loss $\widehat{L_{GE}}$. It is a valuable measure; however, it cannot be directly utilized to evaluate the graph quality in the way, *e.g.*, $\widehat{L_{GE}}\left(G^{(p)}\right)$, since it also depends on the $\mathcal{S}^*$, *i.e.*, the student produced by inner optimization.

Recall the reformulation in Eq. 3 and Eq. 16:

$$\widehat{L_{GE}}\left(\mathcal{S}^* = (\{\alpha_{i,j}\}^*, \theta^*);\ G^{(p)}\right). \tag{97}$$

If we know the specific formula of $\widehat{L_{GE}}$ without knowing $\mathcal{S}^*$, we may obtain an unfair evaluation of different $G^{(p)}$s. **Therefore, fitting $\mathcal{S}^*$ under a given structural choice remains important. In the**

ideal bi-level formulation this is the minimizer of the inner training loss; it is not defined as a minimizer of the outer surrogate. The practical alternating procedure uses a finitely trained student in its place.

Directing training GTs in semi-supervised learning settings is typically challenging. And in our situation, it may also suffer from the *knowledge imbalance* issue (Sec. 2.1, and Sec. 2.3). Thus, we really need some tricks to improve the training of inner optimization for students. The effects of the introduced two improvements can be understood as follows:

1. Curriculum-based local refinement: can encourage the student to *first directly follow* the solution (searched from the solution space distilled from the teacher) *and then gradually modify it* from near to far, which significantly mimics the behavior of a real-world student who hopes to solve a difficult problem. **Its effect is to make full use of the teacher-proved solution. He may modify it, but he must first consider it carefully.**
2. Confidence-aware implicit sparsity regularization: prevent the student from learning an arbitrary attention distribution on the training set (to reduce the training loss) due to knowledge imbalance. **It can facilitate an ability to discriminate which neighbors the teacher encourages and which ones the teacher disagrees with. And thus, it alleviates the *knowledge imbalance* issue by providing a way to transfer the discriminating ability to those non-training nodes** (*i.e.*, nodes outside the training set). Without it, the student risks suffering from degradation due to laziness of attention learning, neighbor discriminating, and knowledge transference.

Both of them are designed based on our challenges, discussing ***how to utilize teachers' guidance*** and ***how to prevent knowledge imbalance in distillation***, which are two important challenges in knowledge distillation.

Thus, the two refinements support inner training in different ways, consistent with their ablation results. They are not prerequisites for defining an interaction space, nor do they establish that the outer surrogate equals graph quality or true generalization error. Their intended benefit is to obtain a student that can use the candidate prior and still learn meaningful attention differences within it.

L.3 Relation with Graph Transformers with Positional Encodings

Many GTs use positional or structural encodings to incorporate information about distance, local connectivity, or communities (Chen et al., 2022a; Kreuzer et al., 2021; Rampášek et al., 2022; Ying et al., 2021). Some obtain these encodings through GNNs rather than only fixed graph statistics (Mialon et al., 2021). These approaches also influence which interactions are favored, making them relevant to the potential-graph perspective in Sec. 2.1.

For a relative pairwise bias b_{uv} added to an attention logit s_{uv}, the unnormalized score satisfies $\exp(s_{uv} + b_{uv}) = \exp(s_{uv})\exp(b_{uv})$. Thus a nonnegative potential weight $G^{(p)}_{uv} = \exp(b_{uv})$ provides a direct multiplicative interpretation, with a zero potential weight representing a masked interaction. Row normalization then yields the weighted aggregation used in our model. This explains the connection between a weighted potential graph and relative attention bias without identifying every positional encoding with an independent graph.

Absolute encodings require more care. When node and position embeddings are concatenated or added, expanding the query–key product produces feature, position, and cross terms. Some of these terms can be interpreted as pairwise biases, but they generally depend on the same trainable projections as the remaining attention score. Context-dependent position encodings (Golovneva et al., 2024) can introduce further dependencies. Therefore, the algebraic decomposition does not by itself imply a separately parameterized potential graph or a distinct outer optimization. Similarly, self-induced interaction modules may share representations with attention and be optimized by the same loss.

Our framework makes the structural family explicit: the teacher supplies prototypes, and the fuser selects their combinations using a separate surrogate objective. Positional encodings and this structural prior need not be mutually exclusive, although their joint implementation would require specifying how their contributions enter the attention weights. The present experiments use the architecture described in Sec. F, without additional handcrafted positional encodings.

L.4 Discussion on More Aspects of Meta Self-Update in Stage 2

In this subsection, we discuss more on some aspects of the proposed Meta Self-Update in Stage 2 (Sec. 2.3 of the main text), providing a better perspective to understand it and including some possible solutions to further improving it.

L.4.1 Generalizing Dark Knowledge-based Meta-Learning Techniques

To see this, one can directly set and hold $\mathcal{Y}_M$ (in $\mathcal{D}_M$) as the teacher's (filtered) predictions $\left\{P_u^{(T)}\right\}_u$. More specifically, we implement $\mathrm{Ens}\,(\cdot,\cdot)$ as:

$$\mathcal{Y}_M \leftarrow \mathrm{Ens}\left(\mathcal{Y}_M, \{P_u^{(S)}\}_{u\in V}\right) = \mathcal{Y}_M, \tag{98}$$

or equivalently set $\boldsymbol{\lambda} = \mathbf{0}$ in the formula $(\mathbf{1}-\boldsymbol{\lambda})\odot\mathcal{Y}_M + \boldsymbol{\lambda}\odot\{P_u^{(S)}\}$, since initially $\mathcal{Y}_M = \left\{P_u^{(T)}\right\}_u$ is just dark knowledge of the teacher. This way degrades our Meta Self-Update into a specific dark knowledge-based meta-learning technique. Therefore, the proposed module can be viewed as one of its generalizations.

L.4.2 Possible Advanced Implementation of Ensembling Operator $\mathrm{Ens}(\cdot,\cdot)$

The operator $\mathrm{Ens}\,(A,B)$ ensembles information B into A (or updates B into A). The update rule selects candidate Bs using the monitor, but their true predictive quality need not increase monotonically. Therefore, we implement $\mathrm{Ens}\,(A,B)$ as an exponential moving weighted sum via $\boldsymbol{\lambda}$ by default.

With a little bit more time and space, we can easily maintain an *expert committee* containing k experts, since different experts may provide more high-quality predictions (*i.e.*, pseudo-labels) for different nodes (*i.e.*, they may be generalizable at different nodes). To release its potential, we can utilize this expert committee to provide some node-wise generalization performance estimation. For example, given a node u, every expert has a prediction for it and we collect k predictions. We can select experts for node u with top k' ***expert confidence scores***, which can be calculated via either individual entropy-based method (*e.g.*, the INE metric introduced in Eq. 9) of the main text or the consistency between those experts (*e.g.*, the estimated density of some expert's prediction in a fitted distribution among all experts conditioned on node $u \in V$, or just kernel-based average similarity score). One can incorporate any general weighting method in ensembling learning when necessary to give *expert confidence scores*, which has the potential to reduce noise and better utilize node-wise flexibility. Furthermore, we can model the global confidence score of the whole *expert committee* with selected experts for a specific node $u \in V$ (called ***committee confidence scores***), which can be implemented via, *e.g.*, the variance of the fitted distribution among all experts. It denotes whether the committee can provide a good estimation of $\hat{L_{GE}}$.

Incorporating *expert confidence scores* and *committee confidence scores* simultaneously naturally generalizes Eq. 16, since it just models global confiduce score for node $u \in V$ (*i.e.*, $S_u^{(\mathcal{Y}_M)}$).

L.4.3 Possible Solution to Updating the Monitor Data $\mathcal{D}_{mt}$

To keep the simplicity of our framework, we keep the Monitor Data $\mathcal{D}_{mt}$ unchanged during the whole learning process of Stage 2. In fact, it can be updated in similar ways as $\mathcal{D}_M$ via, *e.g.*, the ensembling operator $\mathrm{Ens}\,(A,B)$. However, the potentially beneficial Bs for updating $\mathcal{D}_{mt}$ should not be the same as those for updating $\mathcal{D}_M$. Using exactly the same predictions for both roles risks making the monitor merely confirm the targets it helped construct. This is a possible source of feedback, rather than a proof that the surrogate becomes numerically identical to the training loss. Therefore, when we utilize $\mathcal{S}$ to update something, we can update $\mathcal{D}_M$ with probability $p_{\mathcal{D}_M}$ and update $\mathcal{D}_{mt}$ with probability $1-p_{\mathcal{D}_M}$, where the hyper-parameter $p_{\mathcal{D}_M} \in [0,1]$ is a relatively large probability. Such a split would separate some update events for the monitor and targets, but its effect on reliability would need to be tested. The method evaluated in the main text keeps the monitor fixed.

L.5 Knowledge Imbalance, Explicit Sparsity, and Implicit Sparsity

The *knowledge imbalance* issue presented in this paper typically exists in general knowledge distillation techniques, especially when dark knowledge distillation is utilized. The teacher can provide

unevenly reliable predictions across the graph. Training nodes may be fitted particularly well, while some unlabeled nodes near a decision boundary remain uncertain. However, labeled membership is not equivalent to being easy, and unlabeled membership is not equivalent to being difficult; the distinction relevant to our method is the reliability of the transferred prediction. Therefore, the (dark) knowledge distilled from the teacher has *knowledge imbalance* issue, where students may be misled by the possibly noisy or unreliable knowledge of those difficult nodes.

In our framework, the neighborhoods of difficult nodes may be relatively more noisy than easy nodes, and thus they potentially require more sophisticated attention coefficients to reduce the negative impact of noisy connections. However, *explicit sparsity* means attention mechanisms are conducted in a sparse graph and thus reduce the computational complexities. It only restricts the attention scopes for all nodes, yet provides limited direct supervision or guidance for the training of attention matrix W_Q, W_K, W_V. To facilitate more mature attention matrices, we introduce a concept *implicit sparsity* to denote the essential sparsity of attention, *i.e.*, the sparsity degree if attention mechanisms are conducted in a complete graph. In other words, *implicit sparsity* can indicate whether attention can identify truly beneficial distant nodes via itself without the help of the scope restrictions.

Intuitively, consider an extreme case: training nodes have perfect homophilous (sparse) neighborhoods, while non-training nodes have very poor (sparse) ones. The training of attention matrices with only *explicit sparsity* can be easily finished (*e.g.*, finding something similar to Uniform distribution attention coefficients for training nodes), which generally cannot be transferred to non-training nodes and degrades the node embeddings of them (*i.e.*, the rule encoded in attention matrices are not generalizable enough when suffering *knowledge imbalance* issue). To prevent the degradation, we can incorporate some *implicit sparsity* regularization to facilitate the encoding of essential rules that can discriminate valuable or noisy information during attention learning.

L.6 Positioning Interaction-Space Distillation Among Adaptive Structure Methods

Recent GTs and distillation methods have explored several ways to inject structural bias or adapt interaction patterns. This comparison is useful because a reviewer may otherwise conflate three different ideas: *(i)* learning a latent graph directly from data, *(ii)* adaptively selecting among heuristic or architecturally predefined interaction patterns, and *(iii)* our goal of distilling a *task-informed candidate structural family* (that is, an interaction space / admissible structural hypothesis space) and then letting the student adaptively search within it. Table 12 makes these distinctions explicit.

This comparison clarifies the broader value of our problem formulation. The challenge is not merely to design another adaptive graph transformer, but to decide *how a model should search for interaction structure when labels are too scarce to supervise an unrestricted search*. Existing approaches often choose between two extremes: predefined structural bias or highly flexible latent graph learning. Interaction-Space Distillation targets the middle ground. It transfers enough structural information to eliminate many implausible hypotheses, yet still preserves substantial student-side freedom through node- and layer-wise fusion. In this sense, the framework contributes a different kind of distillation object: not a fixed prediction, representation, or attention target, but a task-informed *feasible region* for structure learning. Figure 10 summarizes the companion conceptual point: adaptive selection alone is not the main issue; the more decisive question is whether the candidate interaction family itself is heuristic or task-informed.

This positioning also helps clarify our relation to *fixed heuristic constructions*, such as the original graph, PPR graphs, KNN graphs, or fixed one-/two-hop neighborhoods. These heuristics can be useful, but they are not task-derived and they commit the model to a hand-crafted candidate structure family before learning begins. We provide two complementary pieces of evidence that our task-informed family is preferable in the tested setting. First, our own ablations show that replacing adaptive search with a fixed choice such as the original graph is consistently worse (Tabs. 3 and 5), and even the best individual prototype remains inferior to adaptive fusion (Tab. 9). Second, ANS-GT (Zhang et al., 2022) is a particularly informative baseline because it already *adaptively selects among heuristic structures*, including KNN, PPR, and one-/two-hop neighborhoods. Our method still outperforms ANS-GT on all nine benchmarks in Tab. 1. This suggests that the key advantage does not come from adaptivity alone, but from *what is being adapted over*: a teacher-induced, task-informed interaction family rather than a manually specified heuristic menu.

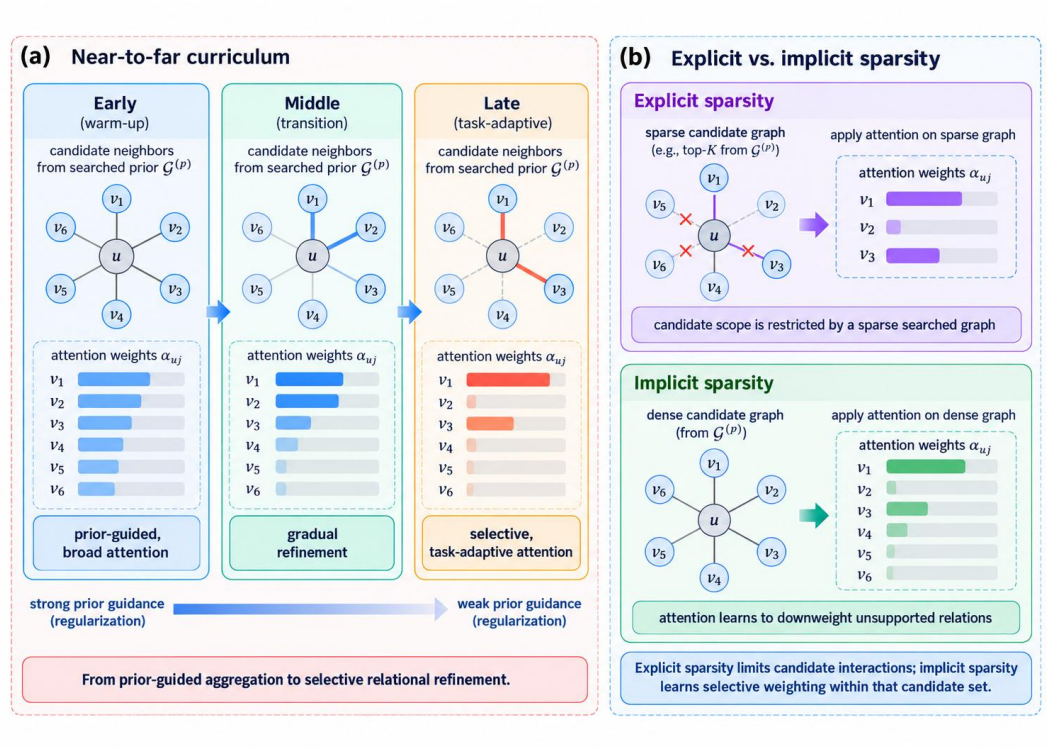


Figure 9: **Near-to-far curriculum and explicit/implicit sparsity.** **(a)** Near-to-far curriculum gradually relaxes the reliance on the searched prior graph $G^{(p)}$. In the early stage, attention is strongly regularized by the searched prior, yielding broad and prior-guided aggregation over candidate neighbors. In the middle stage, this guidance is gradually weakened, allowing the model to refine its attention pattern. In the late stage, the model becomes more task-adaptive and selectively emphasizes the most useful relations. Thus, training evolves from prior-guided aggregation to selective relational refinement. **(b)** Explicit sparsity and implicit sparsity play different roles. Explicit sparsity restricts the candidate interaction set by applying attention only on a sparse searched graph, *e.g.*, the top-K neighbors selected from $G^{(p)}$. Implicit sparsity, in contrast, applies attention on a denser candidate graph but learns to downweight unsupported relations through the attention scores themselves. Therefore, explicit sparsity limits which interactions are considered, while implicit sparsity learns selective weighting within that candidate set.

L.7 Interaction-Space Distillation vs. A Single Graph

This subsection clarifies why we distill an interaction space rather than committing the student to one graph. The key point is not that a larger candidate family is always better; rather, we want a task-informed middle ground between an overly rigid single graph and an overly unconstrained graph search. In this paper, we incorporate the following studies to show our high-quality interaction-space distillation surpasses a single graph distillation:

- *global perspective*: interaction-space distillation is better than a single holistic graph distillation, which is based on the teacher's holistic predictions;
- *local perspective*: adaptively searching in the distilled interaction space is better than random prototypical graph selection;
- *vanilla perspective*: adaptively searching in the distilled interaction space is better than directly adopting the original graph.
- *best-prototype perspective*: Tab. 9 compares adaptive fusion with the best individual aspect graph on three citation benchmarks, rather than relying only on a random prototype.

Table 12: Positioning our framework among representative methods that adapt or guide interaction structure. The table is designed to answer a specific potential confusion: our contribution is *not* merely that the final interaction pattern is adaptive. The key question is *what candidate interaction family the model is allowed to adapt over*. Interaction-Space Distillation differs by transferring a *task-informed interaction space*—equivalently, a teacher-induced candidate structural family / admissible structural hypothesis space—under label scarcity.

Method	**What candidate interaction family is provided?**	**How is the final structure adapted?**	**Why this is still different from ours**
NodeFormer (Wu et al., 2022)	No teacher-induced candidate family is transferred; the student learns latent interactions directly from data.	High; latent structure is learned jointly with the student.	The flexibility comes from end-to-end latent graph learning, not from transferring a task-informed interaction space. Under scarce labels, the student must discover candidate structure, fit representations, and predict labels at the same time.
Curriculum-Enhanced Attention Distillation (CAD) (Huang et al., 2024)	A teacher provides one attention / interaction guidance signal to regularize the student.	Moderate; the student can gradually deviate from the transferred guidance.	It mainly transfers or regularizes *one teacher interaction pattern* or attention behavior. Our framework instead transfers a *teacher-constructed family of candidates* and asks the student to search within that family.
CoBFormer (Xing et al., 2024)	Structural bias is built into the architecture to balance local and global interactions.	Architecture-level; adaptivity is induced by the model design rather than by a learned candidate family.	Its goal is to improve local/global balance, not to distill a task-specific interaction space from a teacher.
DUALFormer (Zhuo et al., 2025)	The candidate interactions come from pre-specified local-GNN and global-attention branches.	Architecture-level; the model learns how to balance these predefined branches.	The candidate family is fixed by architectural design. Our question is different: how should a teacher define *which structures are worth searching* before the student adapts?
M^3Dphormer (Xing et al., 2025)	Multiple *human-designed* hierarchical masks (local / cluster / global) are supplied in advance.	High; adaptive routing selects or mixes among these masks.	This baseline already shows that "multiple candidates + adaptive routing" is not by itself our novelty. Our added ingredient is that the *candidate masks / prototype graphs themselves are task-informed and teacher-distilled*, rather than manually specified.
Ours	**A teacher distills multiple prototypical graphs whose admissible combinations define a task-informed interaction space.**	**High; node-, layer-, and student-state-conditioned search within the shared prototype basis.**	**We transfer the *admissible structural hypothesis space* itself, rather than one fixed structure or a manually designed heuristic menu. This explicitly separates *where candidate structures come from* from *how the student adapts within them*.**

Geometry of the candidate family. For Sum fusion, $\mathbb{G}_u$ is the convex hull of the prototype rows $\{G_{k,u}\}_k$. The full row-wise family is a Cartesian product of these neighborhood sets, not their tensor product. Individual prototype graphs are boundary members of the simplex-defined family, while a fuser with finite softmax logits generally approaches those boundary choices rather than reaching them exactly. Max fusion is also continuous in the coefficients, but need not define a linear or convex family. If all relevant prototype rows coincide, varying the coefficients does not necessarily produce distinct graphs. Otherwise, the continuous parameterization can represent infinitely many weighted candidates without creating new edges outside the finite prototype union.

The MLP fuser further shares parameters across nodes, so the realizable family may be smaller than the row-independent construction. This distinguishes three objects: the outer admissible family, the fuser's parameterized candidates, and the graph produced in a particular forward pass. Our experiments evaluate the latter two; they do not prove exhaustive search of the first. Likewise, the factorization of aggregation into a potential graph and attention is a useful modeling choice rather than a unique decomposition. Accordingly, the notation Span follows the graph-basis analogy used in the main text: it denotes the structured candidate space generated by admissible prototype fusions, rather than requiring unrestricted signed linear combinations, linearly independent prototypes, or a global-optimum guarantee.

Constraint–coverage trade-off. Restricting the student to a distilled interaction space is an intentional inductive bias rather than a claim that the space contains every useful graph. Compared with unrestricted graph search, the prototype family removes many unsupported structural hypotheses and keeps the search computationally manageable; compared with one fixed graph, the continuous row-wise fusion still leaves substantially more room for student adaptation. The trade-off is that Stage 2 cannot introduce an edge that is absent from the union of all prototypes. We therefore view

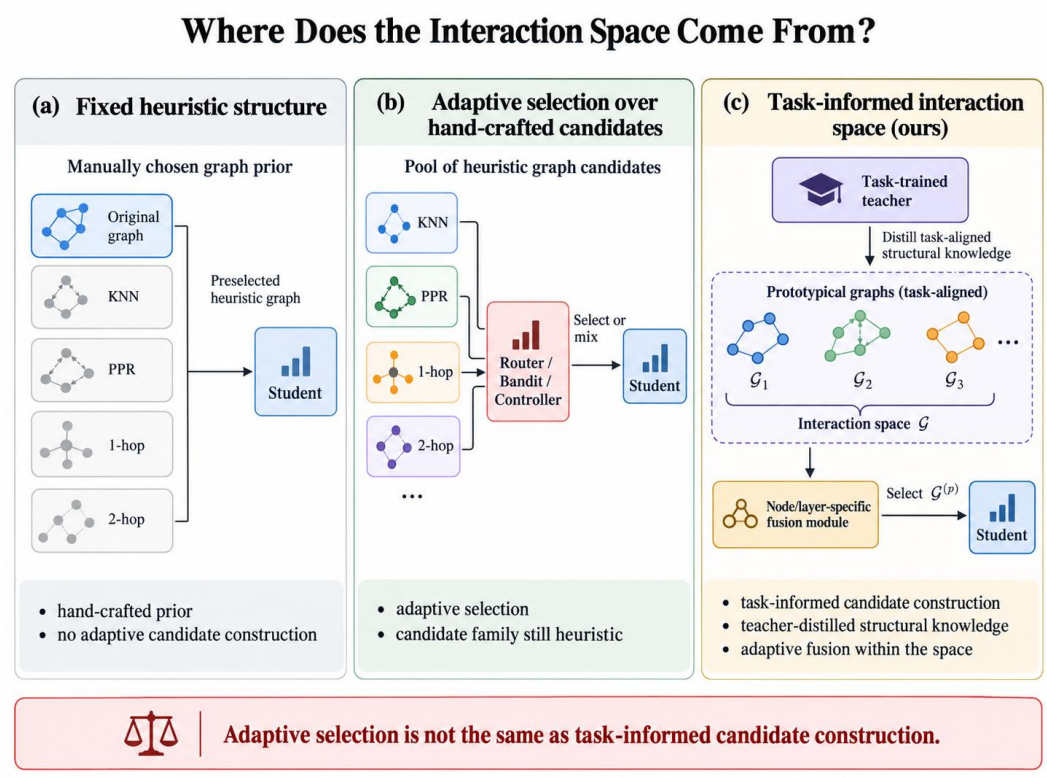


Figure 10: **Where does the interaction space come from? (a)** A fixed heuristic structure directly adopts a manually chosen graph prior, such as the original graph or a predefined heuristic graph, and does not perform adaptive candidate construction. **(b)** Adaptive selection over hand-crafted candidates improves flexibility by selecting or mixing from a pool of heuristic graph candidates, but the candidate family itself remains heuristic. **(c)** Our method constructs a task-informed interaction space from a task-trained teacher. The teacher distills task-aligned structural knowledge into multiple prototypical graphs $\{G_k\}_{k=1}^{K}$, which together form the interaction space $\mathcal{G}$. A node-/layer-specific fusion module then searches within this space to obtain a suitable potential graph for the student. Therefore, adaptive selection is not the same as task-informed candidate construction: our method contributes both a task-informed candidate family and adaptive search within that family.

interaction-space quality from several complementary diagnostics: aspect task relevance (AAA), the label-related quality of the resulting prototype graphs, the best-prototype comparison in Tab. 9, and the final student accuracy. These diagnostics do not prove that no useful relation is omitted. Rather, they test whether the retained space is sufficiently informative for the downstream task. Figure 11 summarizes this middle-ground interpretation visually. Sec. J describes one possible extension in which a later-stage student can expand or update teacher-provided structural knowledge.

L.8 From the Interaction Space to the Aggregation Graph

A separate source of confusion is that several structural objects appear in Stage 2, but they play different roles. The interaction space is the teacher-distilled candidate family; the potential graph is one concrete student-conditioned realization searched from that family; the attention weights further reweight the permitted interactions; and the aggregation graph is the final weighted graph actually used for message passing. Figure 12 visualizes this hierarchy in one place.

L.9 Teacher Role, Strength, and Selection

The teacher in our framework is a source of task-informed structural constraints, not a performance ceiling for the student. This distinction is important because the default Stage 1 teacher is structurally

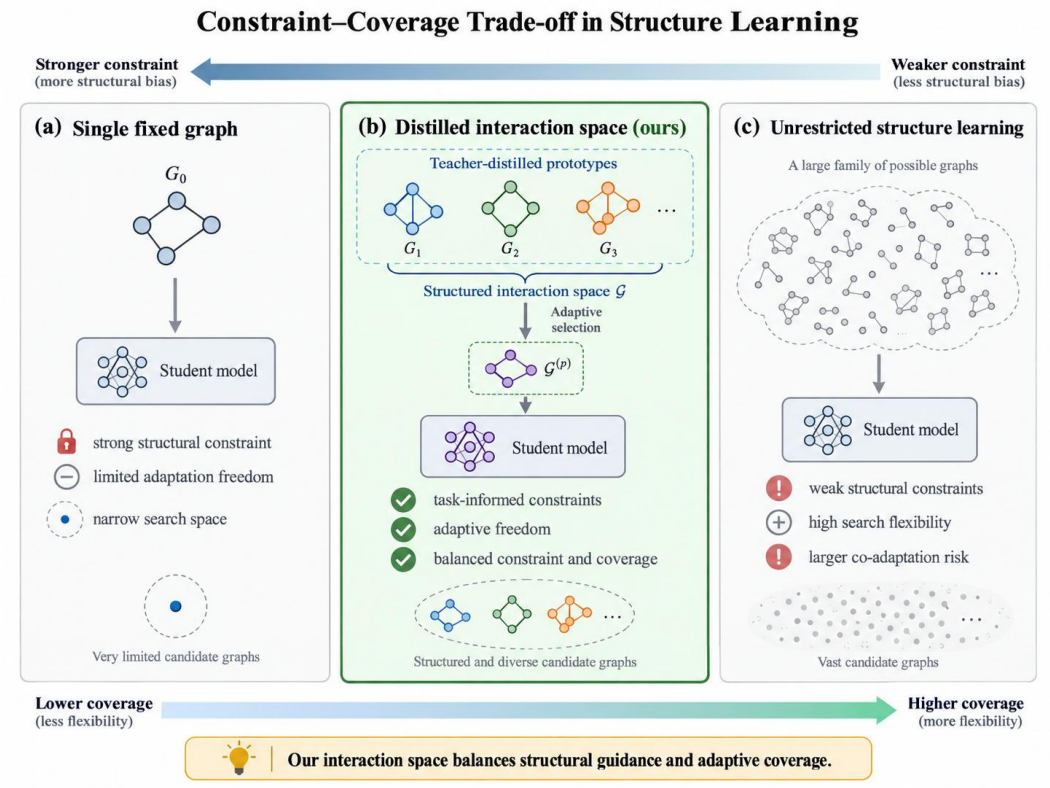


Figure 11: **Constraint–coverage trade-off in structure learning.** **(a)** A single fixed graph imposes strong structural constraints and provides only a narrow search space, but offers little adaptive freedom to the student. **(b)** Our distilled interaction space provides a structured family of candidate graphs induced by teacher-distilled prototypes. It preserves task-informed structural constraints while still allowing the student to adaptively search for a suitable potential graph $G^{(p)}$. In this way, it balances structural guidance and adaptive coverage. **(c)** Unrestricted structure learning offers a much larger search space and higher flexibility, but the structural constraints are weak and the risk of harmful co-adaptation is correspondingly higher. Overall, our interaction space occupies a desirable middle ground between overly rigid fixed-graph learning and overly unconstrained structure search.

simpler than the GT student and is not required to be the strongest predictor. Its local message-passing bias narrows the range of interactions it can fit, which makes its task predictions a useful starting point for constructing prototypes under scarce labels. The student is then allowed to search within the resulting interaction space and can therefore outperform the teacher, as observed in Tab. 2. In this sense, the framework is compatible with a weak-to-strong view of distillation: teacher knowledge restricts implausible search directions without fixing the final solution.

The framework is also not tied to one particular teacher architecture. To examine this point, we performed supplementary teacher-choice checks on the three citation benchmarks. A vanilla GCN was adapted to provide multiple views by partitioning its embedding channels and applying MACD; this deliberately simple teacher still led to competitive student accuracy. We also replaced the default DisenGCN-based teacher with IPGDN (Liu et al., 2020b), another disentangled GNN that explicitly promotes independence among latent factors. The resulting student accuracies are summarized in Tab. 13. The alternative teachers change Stage 1 only; Stage 2 keeps the same student-side search mechanism.

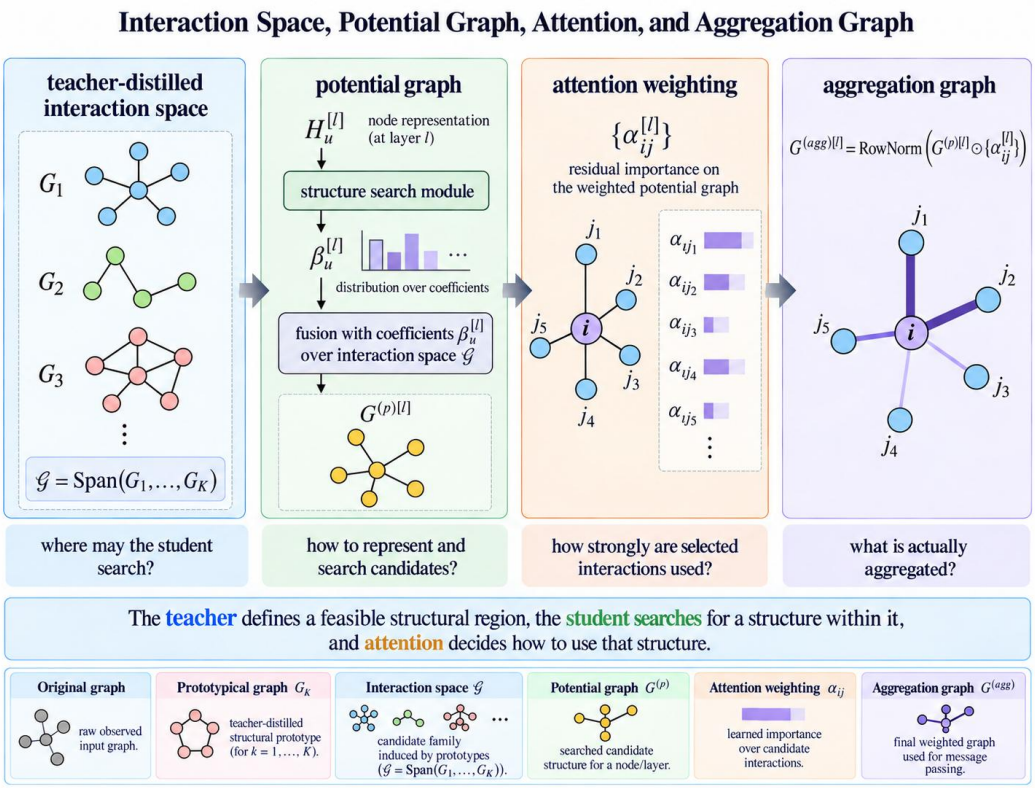


Figure 12: **Conceptual hierarchy from the interaction space to the aggregation graph.** This figure distinguishes six structural objects that play different roles in our framework and are easy to conflate. The *original graph* $G = (V, E)$ is the raw observed input structure; it is neither the distilled interaction space nor the final structure used by the student. A *prototypical graph* G_k is a teacher-distilled, task-aligned structural prototype associated with one aspect of the teacher, and the collection $\{G_k\}_{k=1}^{K}$ serves as the graph basis constructed in Stage 1. These prototypes induce the *interaction space* $\mathcal{G} = \mathrm{Span}(G_1, \ldots, G_K)$, which should be understood as the constrained family of admissible candidates generated by the prescribed prototype-fusion rule, rather than an unrestricted linear space of arbitrary graphs. In Stage 2, the current student representation $H_u^{[l]}$ is mapped by the structure-search module to node- and layer-specific coefficients $\boldsymbol{\beta}_u^{[l]}$ (implemented by an MLP followed by Softmax). The coefficient vector can be viewed as a distribution over the shared prototypes and determines how their corresponding neighborhoods are fused. Applying these coefficients within $\mathcal{G}$ yields a concrete *potential graph* $G^{(p)[l]}$ for the current student layer: this graph specifies which interactions are available and their structural prior weights before attention is applied. The subsequent *attention weighting* $\{\alpha_{ij}^{[l]}\}$ does not define the candidate family itself; instead, it learns the residual importance of candidate interactions on top of the weighted potential graph, deciding how strongly the student should use the structurally permitted relations. Finally, the *aggregation graph* $G^{(\mathrm{agg})[l]} = \mathrm{RowNorm}\left(G^{(p)[l]} \odot \{\alpha_{ij}^{[l]}\}\right)$ is the actual weighted graph used for message passing at layer l. Hence, the objects form a clear hierarchy: the teacher distills a feasible structural region $\mathcal{G}$; the student *searches* within this region to instantiate a node-/layer-specific potential graph $G^{(p)}$; attention then refines the importance of the candidate interactions; and the resulting aggregation graph determines the information that is actually propagated. This separation is central to our formulation because it distinguishes *where the student may search*, *how candidate structures are represented and searched*, *how strongly the selected interactions are used*, and *what is ultimately aggregated*.

Table 13: Supplementary teacher-choice study on the public citation-network splits. The numbers report the final student test accuracy (%). The GCN teacher is adapted by channel partitioning followed by MACD; the default and IPGDN teachers use disentangled views before MACD.

Stage 1 teacher	Cora	Citeseer	Pubmed
GCN-based teacher	83.78	73.40	82.34
DisenGCN-based teacher (default)	84.88	74.05	82.58
IPGDN-based teacher	**85.36**	**74.56**	**82.88**

These results suggest two practical points. First, a strong teacher is not a prerequisite: even the simpler GCN-based variant remains close to the default configuration, supporting the intended role of the teacher as a structural guide rather than an oracle. Second, a more suitable teacher can improve the quality of the distilled space and the final student, as the IPGDN-based variant improves all three citation results over the default teacher in this check. Thus, teacher selection presents a practical trade-off between implementation cost and the quality of the available structural views. When a specialized disentangled teacher is available, it can be beneficial; otherwise, a simple message-passing teacher can still provide useful task guidance after constructing multiple task-aligned views.

M Alternative Ideas and Other Improvements

Recall that we design an interaction-space distillation method based on a message-passing disentangled representation learning technique (Sec. 2.2). In this section, we expect to point out that it is not the unique or indispensable component in our high-level knowledge distillation framework. We selected this kind of technique since they are the simplest and the most natural solution to produce several embeddings for one node in the graph, which well matches our motivation and expectations. Yet, some other alternative ideas are also suitable for our situations, which are still under-explored. Here, we provide two additional kinds of solutions to motivate the readers with some specific challenges we have seen.

The first one is based on hierarchy (called hierarchy-based methods). One can first build a deep model with some residual connections (*e.g.*, deep GNNs), and then incorporate some tricks to alleviate the over-smoothing issues. We can take the node embeddings from different layers as the disentangled representations. Multi-exit neural networks (Han et al., 2023) may provide some techniques to solve the task relevance issue, since some shallow layers have limited knowledge about the target label [7]. This idea can be generalized to some infinite-layer GNNs, *e.g.*, graph neural ODE models (Poli et al., 2019), by choosing layers at different depths or times. Another option is to choose some models with different degrees of constraints. For instance, we can vary the Information Bottleneck (Wu et al., 2020b) to obtain different GNNs with different *bottleneck width*. From some perspective, it can give diverse representations. However, how to ensure the task relevance of some noisy versions of them remains challenging.

The second one is based on *dropout-related methods*. They can be formulated as some generalized form of dropout. They can be efficient and simple, and natural to keep diversity. However, how to improve them to ensure task relevance is their challenge, since larger dropout rates would harm task relevance while smaller ones would risk reducing diversity. Also, it can be generalized to more general data augmentation with the *augmenting-then-ensembling* schema.

N Motivation Experiments

In this section, we provide some experimental results to support our motivations in Sec. 2.1 and Sec. 2.2, which motivates our constrained bi-level optimization and a high-level knowledge distillation framework.

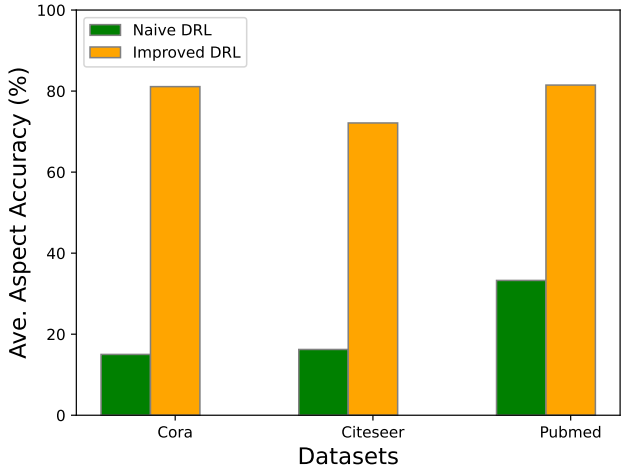
N.1 The Motivation of improving DRL

Despite the comprehensive performance of the DRL methods, however, we find the quality of the aspect predictions it produced is not ensured and typically sub-optimal. This significantly hinders the *task relevance*, which prevents a high-quality graph interaction-space distillation.

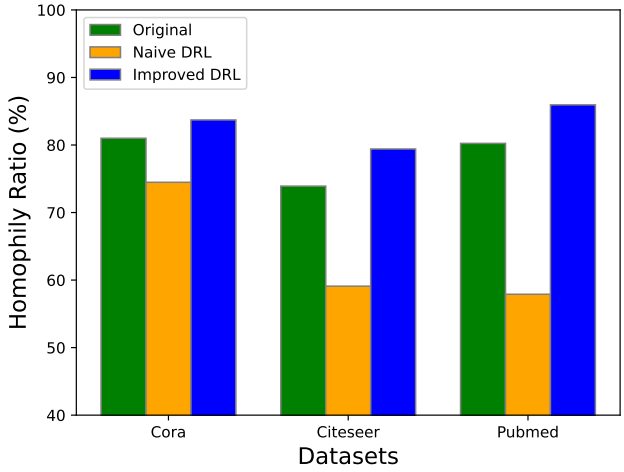
We study the average performance of those aspect predictions produced by DisenGCN as well as the variant improved by the proposed techniques in Sec. 2.2.

We further explore the quality of the graphs built via the same method in Sec. 2.2 and calculate their average homophily score, *i.e.*, same-label neighbor pairs among all neighbor pairs.

[7] It is hard to infer the category of an image with only some information about the line, background, color, etc.



(Upper) The average performance comparison of aspect predictions of naive DRL and our improved DRL



(Bottom) The average homophily score comparison of aspect graphs distilled from naive DRL and our improved DRL as well as the original graph

Figure 13: The Motivation of improving DRL

We show these results in Fig. 13, where the upper one contains results of aspect predictions and the bottom one contains the quality of built graphs. From Fig. 13, we can clearly see the aspect predictions before improvement are relatively low-quality, and the quality of the built graphs before improvement is typically sub-optimal.

N.2 Studies on the quality of GGT-induced graphs

To study whether GGTs can learn a high-quality graph by themselves (by their attention matrix itself), we pre-train a GGT and extract an attention-induced graph (by filtering the attention values that are quite small). We evaluate the quality of the induced graph from the homophily score and the sparsity. Figure 14 reports the diagnostic for the tested GGT and sparsity-regularized variant. Their induced graphs have lower homophily than the original citation graphs in these experiments. This motivates additional task-informed guidance, but does not establish that all global GTs learn poor relations or that homophily alone determines their predictive value.

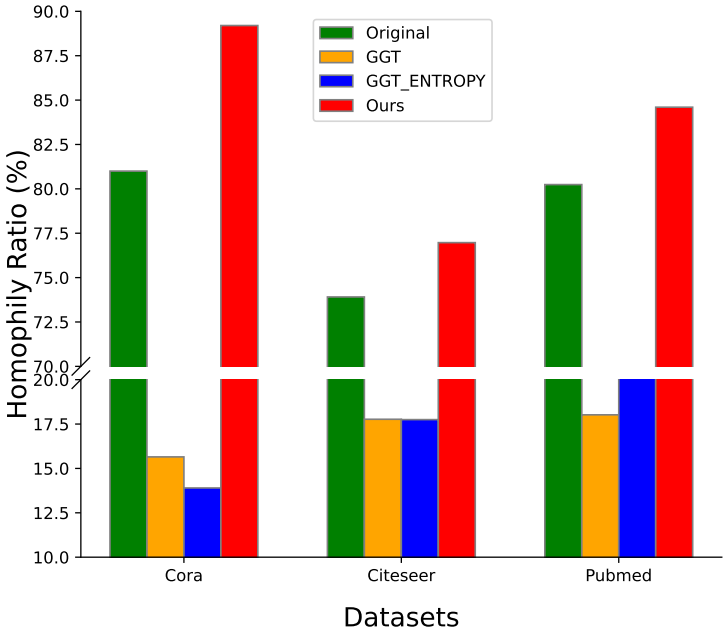


Figure 14: Studies on the quality of GGT-induced graphs.

N.3 The Motivation of Cooperative Collapse

Recall the Cooperative Collapse issue described in Sec. 1 and Sec. 2.1. We expect to disentangle the potential graph $G^{(p)}$ and attention coefficients $\{\alpha_{i,j}\}$. One method to explicitly disentangle them is to introduce a learnable module to generate a potential graph $G^{(p)}$ and jointly train it and the attention mechanism in GGT. We evaluate and compare the performance of a naive GGT and a GGT with the aid of learnable $G^{(p)}$. We differentially model $G^{(p)}$ as a MLP applied to the node embeddings and inner product-based link prediction with a $\mathrm{Sigmoid}\,(\cdot)$ activation. It will be combined with the separately learned attention via a Hadamard product and a row-sum normalization, *i.e.*:

$$\{\alpha_{i,j}\}' = \mathrm{RowNorm}\left(\{\alpha_{i,j}\} \odot G^{(p)}\right). \tag{99}$$

Our experimental results in Fig. 15 clearly show large margins between their performance on different graph benchmarks, where the dashed lines denote the performance of naive GGT and the real lines denote the performance of a GGT with a jointly learning $G^{(p)}$, which reveals that introducing a learnable $G^{(p)}$ with the same joint training objective may not alleviate the performance degradation of GGTs and even further exacerbate the severe issue. Figure 16 summarizes the underlying intuition and contrasts this joint optimization risk with our constrained bi-level alternative.

O Omitted or Incomplete Tables

The supplementary tables are placed near their explanations rather than collected without context. Section C contains the full ablation study, additional GT comparisons (including the focused M^{3}Dphormer comparison), memory measurements, and single-prototype comparison. The symbol tables are in Sec. D; dataset statistics and hyperparameter tables are in Sec. P; complete timing measurements are in Sec. Q. This index preserves the appendix organization while directing the reader to each table's accompanying discussion.

P More Experimental Details

In this section, more details of experiments are provided here to complement the experiments in the main text.

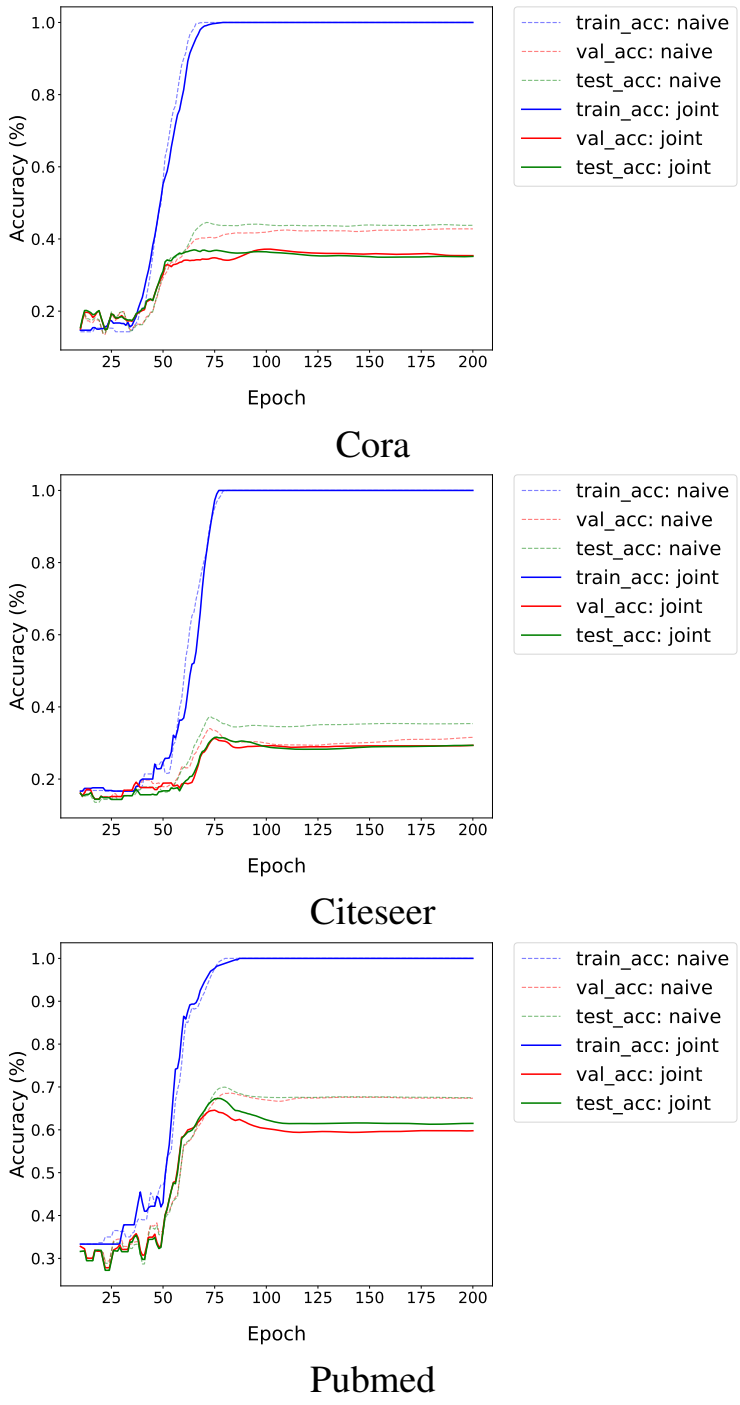


Figure 15: Motivation for separating structure selection from attention fitting. Solid curves show the tested GGT with a jointly learned potential graph; dashed curves show the naive GGT. Both fit the training nodes, while the joint variant has weaker validation/test performance on these citation benchmarks. This illustrates the co-adaptation risk termed Cooperative Collapse, rather than a universal failure of joint graph learning.

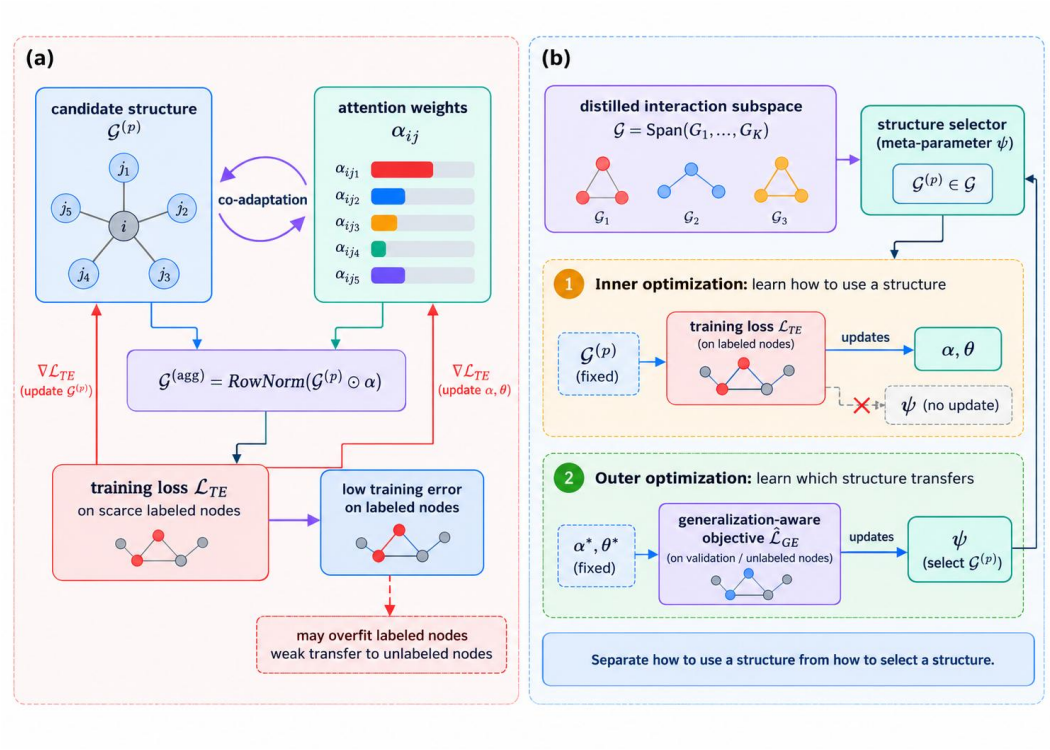


Figure 16: **Cooperative Collapse vs. constrained bi-level optimization. (a)** In joint optimization, the potential graph $G^{(p)}$ and attention weights α_{ij} are both driven by the same training loss L_{TE} on scarce labeled nodes. As a result, structure selection and attention fitting may co-adapt to reduce training error, which can overfit the labeled nodes and weaken transfer to unlabeled ones; we refer to this risk as Cooperative Collapse. **(b)** Our framework separates these two roles by constrained bi-level optimization. The potential graph is restricted to a distilled interaction space $\mathcal{G} = \text{Span}(G_1, \ldots, G_K)$. In the inner optimization, the searched structure $G^{(p)}$ is fixed and the student parameters are updated to learn how to use that structure by minimizing the training loss L_{TE}. In the outer optimization, the trained student is fixed and the structure selector (meta-parameter ψ) is updated by minimizing the generalization-aware objective $\widehat{L}_{GE}$, thereby learning which structure transfers better. This decoupling separates how to use a structure from how to select a structure.

P.1 Experimental Environments, Platforms, and Tools

We implement it via Pytorch (Paszke et al., 2019) and optimize it with Adam Optimizer (Kingma & Ba, 2015). All experiments are performed on an Ubuntu system with a single NVIDIA RTX $A6000$ GPU (48GB Memory) and 32 AMD EPYC 7543 CPUs.

Table 14: Dataset statistics of nine real-world benchmarks with their splits

Dataset	Nodes	Edges	Ave.Degree	Features	Classes	train/val/test
Cora	2,708	5,429	4.0	1,433	7	140/500/1000
Citeseer	3,327	4,732	2.84	3,703	6	120/500/1000
PubMed	19,717	44,338	4.5	500	3	60/500/1000
OGBN-ArXiv	169,343	1,166,243	13.77	128	40	800/800/167743
Texas	183	309	3.38	1,703	5	87/59/37
Wisconsin	251	499	5.45	1,703	5	120/80/51
Cornell	183	295	3.22	1,703	5	87/59/37
Actor	7,600	33,544	8.83	931	5	3648/2432/1520
Flickr	89,250	899,756	10.08	500	7	140/140/88970

Table 15: Hyper-parameters Configurations of our model *Ours* for semi-supervised node classification tasks on nine public graph benchmarks.

Dataset	Hyper-parameters Configurations
Cora	$d = 256$, $L = 2$, $N_H = 4$, $K = 6$, $K' = 10$, $\lambda^{[B]} = -0.01$, $\lambda^{[E]} = 0.04$, $\lambda_{Reg} = 0.06$, $a = 0.9$, $\tau = 1.2$, $\gamma_0 = 0.9$, $t = 0.9$, $\eta = 0.7$, $T^{(in)} = 300$, $T^{(out)} = 1$, $T^{(r)} = 2$, $lr1 = 1e-3$, $lr2 = 1e-3$, weight decay $= 0.005$, dropout $= 0.8$, attn dropout $= 0.7$
Citeseer	$d = 256$, $L = 2$, $N_H = 4$, $K = 4$, $K' = 10$, $\lambda^{[B]} = -0.1$, $\lambda^{[E]} = 0.01$, $\lambda_{Reg} = 0.06$, $a = 0.9$, $\tau = 1.2$, $\gamma_0 = 0.9$, $t = 0.9$, $\eta = 0.7$, $T^{(in)} = 300$, $T^{(out)} = 1$, $T^{(r)} = 2$, $lr1 = 1e-3$, $lr2 = 1e-2$, weight decay $= 0.005$, dropout $= 0.8$, attn dropout $= 0.7$
Pubmed	$d = 128$, $L = 1$, $N_H = 2$, $K = 6$, $K' = 6$, $\lambda^{[B]} = -0.1$, $\lambda^{[E]} = 0.1$, $\lambda_{Reg} = 0.01$, $a = 0.8$, $\tau = 1.1$, $\gamma_0 = 0.95$, $t = 0.85$, $\eta = 0.7$, $T^{(in)} = 150$, $T^{(out)} = 1$, $T^{(r)} = 2$, $lr1 = 1e-3$, $lr2 = 5e-2$, weight decay $= 5e-5$, dropout $= 0.45$, attn dropout $= 0.5$
Actor	$d = 128$, $L = 1$, $N_H = 2$, $K = 2$, $K' = 20$, $\lambda^{[B]} = -0.1$, $\lambda^{[E]} = 0.01$, $\lambda_{Reg} = 10$, $a = 1$, $\tau = 0.6$, $\gamma_0 = 0.9$, $t = 0.5$, $\eta = 10$, $T^{(in)} = 100$, $T^{(out)} = 1$, $T^{(r)} = 3$, $lr1 = 0.01$, $lr2 = 0.005$, weight decay $= 0.00005$, dropout $= 0.2$, attn dropout $= 0.1$
Cornell	$d = 64$, $L = 1$, $N_H = 1$, $K = 2$, $K' = 3$, $\lambda^{[B]} = -0.1$, $\lambda^{[E]} = 0.1$, $\lambda_{Reg} = 0.1$, $a = 0.6$, $\tau = 0.8$, $\gamma_0 = 0.95$, $t = 5$, $\eta = 0.5$, $T^{(in)} = 50$, $T^{(out)} = 1$, $T^{(r)} = 5$, $lr1 = 0.01$, $lr2 = 0.001$, weight decay $= 0.0005$, dropout $= 0.2$, attn dropout $= 0.4$
Texas	$d = 64$, $L = 2$, $N_H = 1$, $K = 2$, $K' = 20$, $\lambda^{[B]} = -1$, $\lambda^{[E]} = 0.01$, $\lambda_{Reg} = 1$, $a = 0.4$, $\tau = 0.4$, $\gamma_0 = 0.95$, $t = 5$, $\eta = 1$, $T^{(in)} = 100$, $T^{(out)} = 1$, $T^{(r)} = 2$, $lr1 = 0.005$, $lr2 = 0.001$, weight decay $= 0.00005$, dropout $= 0.2$, attn dropout $= 0.$
Wisconsin	$d = 256$, $L = 1$, $N_H = 2$, $K = 4$, $K' = 15$, $\lambda^{[B]} = 0.01$, $\lambda^{[E]} = 0.1$, $\lambda_{Reg} = 10$, $a = 0.8$, $\tau = 0.2$, $\gamma_0 = 0.9$, $t = 0.5$, $\eta = 1$, $T^{(in)} = 50$, $T^{(out)} = 1$, $T^{(r)} = 4$, $lr1 = 0.001$, $lr2 = 0.001$, weight decay $= 0.$, dropout $= 0.5$, attn dropout $= 0.1$
ArXiv	$d = 128$, $L = 1$, $N_H = 2$, $K = 2$, $K' = 15$, $\lambda^{[B]} = -0.01$, $\lambda^{[E]} = 2$, $\lambda_{Reg} = 0.00001$, $a = 0.6$, $\tau = 1.5$, $\gamma_0 = 0.99$, $t = 0.85$, $\eta = 0.4$, $T^{(in)} = 100$, $T^{(out)} = 1$, $T^{(r)} = 2$, $lr1 = 0.005$, $lr2 = 0.03$, weight decay $= 0.00005$, dropout $= 0.7$, attn dropout $= 0.9$
Flickr	$d = 128$, $L = 4$, $N_H = 8$, $K = 2$, $K' = 3$, $\lambda^{[B]} = -1$, $\lambda^{[E]} = 1$, $\lambda_{Reg} = 1$, $a = 0.65$, $\tau = 1.5$, $\gamma_0 = 0.9$, $t = 0.8$, $\eta = 0.9$, $T^{(in)} = 100$, $T^{(out)} = 1$, $T^{(r)} = 2$, $lr1 = 0.001$, $lr2 = 0.03$, weight decay $= 0.00005$, dropout $= 0.2$, attn dropout $= 0.1$

P.2 Dataset Descriptions

We summarize some important details of all the graph benchmarks in Tab. 14 utilized for evaluation, including the specific splits. We use the standard citation-network splits following Kipf & Welling (2017) and the other public splits listed in Tab. 14. For ArXiv and Flickr, we sample 20 training nodes per class to obtain the low-label setting used in our comparisons. Their validation sets contain 800 and 140 nodes, respectively. These are custom splits, whereas Actor and the webpage datasets have substantially larger labeled fractions; the benchmark collection is therefore not a uniform label-scarcity protocol. Notably, in semi-supervised settings, we have only limited labeled nodes since the labels are quite expensive to obtain in the real world, which is quite significant in our world, since collecting so many data samples is already challenging enough, not alone collecting enough labels.

Table 16: Reported hyperparameter ranges and additional values appearing in the selected configurations of Tab. 15. Outer optimization uses $T^{(out)} = 1$. The last column distinguishes selected values outside the base ranges from the ranges themselves; it does not enumerate individual tuning trials.

Parameter	Recorded range	Additional selected values
d	$64, 128, 256$	–
L	$1, 2$	4
N_H	$1, 2$	$4, 8$
K	$2, 3, 4, 5, 6, 7, 8, 9, 10$	–
K'	$5, 10, 15, 20, 25, 30$	$3, 6$
$\lambda^{[B]}$	$-1, -0.1, -0.01, 0, 0.01, 0.1, 1$	–
$\lambda^{[E]}$	$0, 0.01, 0.1, 1, 2$	0.04
λ_{Reg}	$0.01, 0.1, 0.5, 1, 2, 5, 10$	$0.00001, 0.06$
a	$0.2, 0.4, 0.6, 0.8, 1$	$0.65, 0.9$
τ	$0.2, 0.4, 0.6, 0.8, 1$	$1.1, 1.2, 1.5$
γ_0	$0.9, 0.95, 0.99$	–
η	$0, 0.1, 0.5, 1, 2, 10$	$0.4, 0.7, 0.9$
$T^{(in)}$	$50, 100$	$150, 300$
$T^{(out)}$	1	–
$T^{(r)}$	$2, 3, 4, 5$	–
$lr1$	$10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}$	–
$lr2$	$10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}$	0.03
weight decay	$5\times10^{-3}, 5\times10^{-4}, 5\times10^{-5}$	0
dropout	$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7$	$0.45, 0.8$
attn dropout	$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7$	$0, 0.9$

P.3 Hyper-parameters

We tune hyper-parameters stage by stage. In every stage, we choose the hyper-parameters according to their best validation performance. Table 16 presents the recorded base search ranges and separately lists additional values appearing in the selected configurations. This presentation distinguishes the base ranges from additional selected values rather than merging them into a single search grid. The configurations specify the settings used for the reported comparisons. For every kind of experiment, we search in the space randomly for 200 times in total. Additionally, for better reproducibility, we report all hyper-parameters configurations used by *Ours* for comparative experiments in Tab. 15.

Q Efficiency Analysis

We provide the running time in Tab. 17 on some real-world graph benchmarks to show the efficiency of our method. Our real-world running time is acceptable and much more efficient than the vanilla GGT.

R Other Supplementary Experiments and Comparison

R.1 Results on Large-scale Graphs

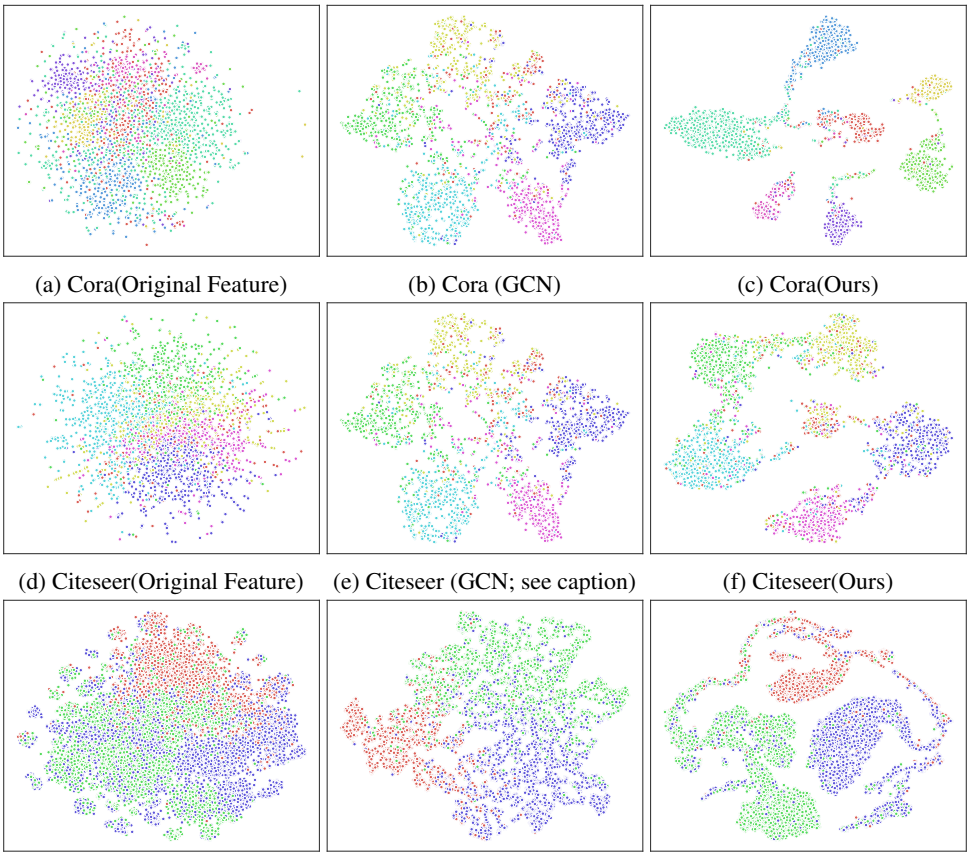
We have provided the results of our framework and sixteen counterparts on two large-scale graph benchmarks, *i.e.*, ArXiv and Flickr, in Tab. 1 of the main text.

Table 17: Running time comparison on real-world graph benchmarks (sec/epoch)

Method	Cora	Citeseer	Pubmed	Flickr	ArXiv
$\text{SuperGAT}_{\text{SD}}$	0.0291s	0.0294s	0.1239s	0.7513s	2.3145s
APPNP	0.0112s	0.0123s	0.0263s	0.0242s	0.0415s
GT	0.0107s	0.0138s	0.2543s	OOM	OOM
SAN	0.1648s	0.1542s	0.2414s	OOM	OOM
Graphormer	0.4333s	0.6385s	OOM	OOM	OOM
ANS-GT	1.4529s	2.9731s	3.4333s	7.7964s	24.5402s
Nodeformer	0.1883s	0.2168s	0.2922s	0.8376s	1.3597s
NAGphormer	0.0224s	0.0442s	0.0305s	0.8348s	1.5599s
GOAT	1.0262s	1.0453s	1.4500s	28.2812s	58.7722s
Exphormer	0.0860s	0.1753s	0.3480s	1.1123s	1.948s
SGFormer	0.0101s	0.0107s	0.0213s	0.0513s	0.1136s
DIFFormer	0.0291s	0.0302s	0.0474s	0.2966s	0.5449s
Ours	0.0416s	0.0465s	0.2583s	0.4347s	0.4608s

S VISUALIZATIONS

We plot some node embeddings produced by our method and GCN, together with the original features in Fig. 17.



(a) Cora(Original Feature) (b) Cora (GCN) (c) Cora(Ours)

(d) Citeseer(Original Feature) (e) Citeseer (GCN; see caption) (f) Citeseer(Ours)

(g) Pubmed(Original Feature) (h) Pubmed(GCN) (i) Pubmed(Ours)

Figure 17: Scatter plots of original features and embeddings from GCN and our method on Cora, Citeseer, and Pubmed. The supplied GCN visualizations for Cora and Citeseer are identical; the Citeseer GCN panel therefore requires verification against its plotting output and should not be interpreted as independent evidence.