Dual-Kernel Graph Community Contrastive Learning (Appendix)

Xiang Chen[1,2], **Kun Yue**[1,2], **Wenjie Liu**[1,2], **Zhenyu Zhang**[3], **Liang Duan**[1,2*]

[1]School of Information Science and Engineering, Yunnan University, Kunming, China
[2]Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, China
[3]College of Artificial Intelligence, Tianjin University of Science and Technology, Tianjin, China
chenx@stu.ynu.edu.cn, duanl@ynu.edu.cn

A. Detailed Proofs

A.1 Proof of Proposition 1

Proposition 1. *Let the feature dimension of the community-level feature space $\mathcal{X}_{\mathcal{P}}$ be d^P. Then, the expected number of distinct partitioned substructures generated by the Dropout$(\cdot)$ operation for each partition P_j is:*

$$\mathbb{E}[|P_j^s|s=1,\cdots,d^P|] = d^P\left(1-(1-p)^{|P_j|}\right) \tag{17}$$

where P_j^s is a substructure of P_j on the feature dimension s, and p is the dropout probability.

Proof. We consider each dimension of community-level features as an aggregation of one-dimensional node features within a partitioned substructure. Let C_j^s denote the indicator random variable for P_j^s:

$$C_j^s = \begin{cases} 0, & \text{if } P_j^s = P_j \\ 1, & \text{otherwise} \end{cases} \tag{18}$$

For the partitioned substructure P_j^s to be identical to the original partition P_j, all nodes within the partition must aggregate their one-dimensional features towards the community centroid. Given that the probability of any node in the partitioned substructure aggregating its features to the community centroid is $1-p$, then in this scenario, we have:

$$P(C_j^s) = \begin{cases} (1-p)^{|P_j|}, & C_j^s = 0 \\ 1-(1-p)^{|P_j|}, & C_j^s = 1 \end{cases} \tag{19}$$

Then, the expected value of C_j^s is

$$\begin{aligned} \mathbb{E}[C_j^s] &= 0 \cdot P(C_j^s = 0) + 1 \cdot P(C_j^s = 1) \\ &= 1-(1-p)^{|P_j|} \end{aligned} \tag{20}$$

According to the linearity of expectation, we have:

$$\begin{aligned} \mathbb{E}[|P_j^s|s=1,\cdots,d^P|] &= \mathbb{E}[\sum\nolimits_k^{d^P} C_j^s] = \sum\nolimits_k^{d^P} \mathbb{E}[C_j^s] \\ &= d^P\left(1-(1-p)^{|P_j|}\right) \end{aligned} \tag{21}$$

To this end, we can deduce Proposition 1. □

*Corresponding author.

A.2 Proof of Proposition 2

Proposition 2. *Let $\bar{\mathbf{A}}$ be the normalized adjacency matrix constructed from positive node pairs in the contrastive loss $\mathcal{L}_{\mathcal{P}}$ and $y(v)$ denote the label of v. Then, the bound of the smoothness between node embeddings is:*

$$\left\|\mathbf{V}-\bar{\mathbf{A}}\mathbf{V}\right\|_F \leq \sqrt{2}L \sum_{v_i\in\mathcal{V}} \frac{1}{1+\frac{\varepsilon(v_i)\lambda_{v_i}}{(1-\varepsilon(v_i))\gamma_{v_i}}} \tag{22}$$

where $\lambda_{v_i} = \mathbb{E}_{v_t\in\mathcal{N}(v_i),y(v_i)=y(v_t)}\kappa_B(\{\mathbf{v}_i,\mathbf{c}_j\},\{\mathbf{v}_t,\mathbf{c}_k\})$ and $\gamma_{v_i} = \mathbb{E}_{v_t\in\mathcal{N}(v_i),y(v_i)\neq y(v_t)}\kappa_B(\{\mathbf{v}_i,\mathbf{c}_j\},\{\mathbf{v}_t,\mathbf{c}_k\})$. L is the Lipschitz constant, and $\varepsilon(v_i)$ is the one-hop homophily score of node v_i in $\bar{\mathbf{A}}$, defined as:

$$\varepsilon(v_i) = \frac{1}{|\mathcal{N}(v_i)|}\sum\nolimits_{v_t\in\mathcal{N}(v_i)} \mathbb{1}[y(v_i)=y(v_t)] \tag{23}$$

where $\mathbb{1}[\cdot]$ is the indicator function.

Proof. Let the positive sample score $a_{v_iv_t}$ in the contrastive loss serve as the weight between node v_i and node v_t in the adjacency matrix $\bar{\mathbf{A}}$, we have:

$$\left\|\mathbf{V}-\bar{\mathbf{A}}\mathbf{V}\right\|_F \leq \sum_{v_i\in\mathcal{V}} ||\mathbf{v}_i - \sum_{v_t\in\mathcal{N}(v_i)} a_{v_iv_t}\mathbf{v}_t||_2 \tag{24}$$

Since $\bar{\mathbf{A}}$ is row-normalized, we have:

$$\begin{aligned} \left\|\mathbf{V}-\bar{\mathbf{A}}\mathbf{V}\right\|_F &\leq \sum_{v_i\in\mathcal{V}} || \sum_{v_t\in\mathcal{N}(v_i)} (\mathbf{v}_i - a_{v_iv_t}\mathbf{v}_t)||_2 \\ &\leq \sum_{v_i\in\mathcal{V}}\sum_{v_t\in\mathcal{N}(v_i)} a_{v_iv_t}\left\|\mathbf{v}_i-\mathbf{v}_t\right\|_2 \end{aligned} \tag{25}$$

Assume that the linear mapping function from node features to labels is L-Lipschitz continuous, we have:

$$\begin{aligned} &\left\|\mathbf{V}-\bar{\mathbf{A}}\mathbf{V}\right\|_F \leq L\sum_{v_i\in\mathcal{V}}\sum_{v_t\in\mathcal{N}(v_i)} a_{v_iv_t}\left\|y(v_i)-y(v_t)\right\|_2 \\ &= L\sum_{v_i\in\mathcal{V}}\sum_{v_t\in\mathcal{N}(v_i),y(v_i)\neq y(v_t)} a_{v_iv_t}\left\|y(v_i)-y(v_t)\right\|_2 \\ &= \sqrt{2}L\sum_{v_i\in\mathcal{V}}\sum_{v_t\in\mathcal{N}(v_i),y(v_i)\neq y(v_t)} a_{v_iv_t} \\ &= \sqrt{2}L\sum_{v_i\in\mathcal{V}}\frac{\sum_{v_t\in\mathcal{N}(v_i),y(v_i)\neq y(v_t)}\kappa_B(\{\mathbf{v}_i,\mathbf{c}_j\},\{\mathbf{v}_t,\mathbf{c}_k\})}{\sum_{v_t\in\mathcal{N}(v_i)}\kappa_B(\{\mathbf{v}_i,\mathbf{c}_j\},\{\mathbf{v}_t,\mathbf{c}_k\})} \end{aligned} \tag{26}$$

Let $\lambda_{v_i} = \mathbb{E}_{v_t \in \mathcal{N}(v_i), y(v_i)=y(v_t)} \kappa_B(\{\mathbf{v}_i, \mathbf{c}_j\}, \{\mathbf{v}_t, \mathbf{c}_k\})$, $\gamma_{v_i} = \mathbb{E}_{v_t \in \mathcal{N}(v_i), y(v_i)\neq y(v_t)} \kappa_B(\{\mathbf{v}_i, \mathbf{c}_j\}, \{\mathbf{v}_t, \mathbf{c}_k\})$, and $\varepsilon(v_i)$ is the one-hop homophily score of node v_i. Then, we can obtain:

$$\begin{aligned}
&\left\|\mathbf{V} - \bar{\mathbf{A}}\mathbf{V}\right\|_F \\
&\leq \sqrt{2}L \sum_{v_i \in \mathcal{V}} \frac{|\mathcal{N}(v_i)|(1-\varepsilon(v_i))\gamma_{v_i}}{|\mathcal{N}(v_i)|(1-\varepsilon(v_i))\gamma_{v_i} + |\mathcal{N}(v_i)|\varepsilon(v_i)\lambda_{v_i}} \\
&= \sqrt{2}L \sum_{v_i \in \mathcal{V}} \frac{1}{1 + \frac{\varepsilon(v_i)\lambda_{v_i}}{(1-\varepsilon(v_i))\gamma_{v_i}}}
\end{aligned} \tag{27}$$

Here, we complete the proof of Proposition 2. □

A.3 Proof of Proposition 3

To prove Proposition 3, we first introduce a lemma that shows the contrastive loss on the original graph is close to the sum of the coarsened contrastive loss and the low-rank approximation gap (Zhang et al. 2024).

Lemma 1. *Assuming the original features $\mathbf{X}$ is bounded by $S_{\mathbf{X}} := \max_i ||\mathbf{X}_i||_2$. Then, the contrastive loss of the k-step diffusion graph $\mathbf{A}^k$, denoted as $\mathcal{L}_G$, can be approximated by the coarsened contrastive loss $\mathcal{L}_S$ of StructComp.*

$$|\mathcal{L}_G - \mathcal{L}_S| \leq L||\mathbf{A}^k - \mathbf{P}\mathbf{P}^{\mathrm{T}}||_F S_{\mathbf{X}} ||\mathbf{W}_P||_2 \tag{28}$$

Intuitively, this lemma shows that the community-level kernel can approximate the contrastive loss of the k-step diffusion graph $\mathbf{A}^k$.

Proposition 3. *Assuming the original features $\mathbf{X}$ and the mapped features $\phi(\mathbf{X})$ are bounded by $S_{\mathbf{X}} := \max_i ||\mathbf{X}_i||_2$ and $S_{\phi(\mathbf{V})} := \max_i ||\phi(\mathbf{V}_i)||_2^2$, respectively. Then, the original contrastive loss of the k-step diffusion graph $\mathbf{A}^k$, denoted as $\mathcal{L}_G$, can be approximated by the dual-kernel community contrastive loss, $\mathcal{L}_{\mathcal{P}}^{lc}$, without considering the influence of combination coefficients:*

$$|\mathcal{L}_G - \mathcal{L}_{\mathcal{P}}^{lc}| \leq L||\mathbf{A}^k - \mathbf{P}\mathbf{P}^{\mathrm{T}}||_F S_{\mathbf{X}} ||\mathbf{W}_P||_2 + S_{\phi(\mathbf{V})}$$

Proof. According to the triangle inequality for absolute values, we have:

$$\begin{aligned}
|\mathcal{L}_G - \mathcal{L}_{\mathcal{P}}^{lc}| &= |\mathcal{L}_G - \mathcal{L}_S + \mathcal{L}_S - \mathcal{L}_{\mathcal{P}}^{lc}| \\
&\leq |\mathcal{L}_G - \mathcal{L}_S| + |\mathcal{L}_S - \mathcal{L}_{\mathcal{P}}^{lc}|
\end{aligned} \tag{29}$$

We denote the node-level and community-level positive pairs in the dual-kernel GCCL loss with the linear combination method $\ell_{lc}(v_i, P_j)$ as $\ell_{lc}^+(v_i)$ and $\ell_{lc}^+(P_j)$, respectively. Then, for the term $|\mathcal{L}_S - \mathcal{L}_{\mathcal{P}}^{lc}|$, we can obtain:

$$\begin{aligned}
|\mathcal{L}_S - \mathcal{L}_{\mathcal{P}}^{lc}| &= |\frac{1}{n}\sum_{v_i \in \mathcal{V}} \ell_{lc}^+(P_j) - \frac{1}{n}\sum_{v_i \in \mathcal{V}} \left(\ell_{lc}^+(P_j) + \ell_{lc}^+(v_i)\right)| \\
&= |\frac{1}{n}\sum_{v_i \in \mathcal{V}} \left(\ell_{lc}^+(v_i)\right)| \\
&= |\frac{1}{n}\sum_{v_i \in \mathcal{V}} \sum_{P_k \in \mathcal{N}(P_j)} \sum_{v_t \in P_k} \phi(\mathbf{v}_i)^{\mathrm{T}}\phi(\mathbf{v}_t)| \\
&\leq |\frac{1}{n}\sum_{v_i \in \mathcal{V}} \sum_{v_t \in \mathcal{V}} \phi(\mathbf{v}_i)^{\mathrm{T}}\phi(\mathbf{v}_t)|
\end{aligned} \tag{30}$$

Let $\phi(\mathbf{v}_{max})$ be the upper bound of feature map $\phi(\mathbf{v}_i)$, where $i \in n$, we can obtain:

$$\begin{aligned}
|\mathcal{L}_S - \mathcal{L}_{\mathcal{P}}^{lc}| &\leq |\frac{1}{n}\sum_{v_i \in \mathcal{V}} \sum_{v_t \in \mathcal{V}} \phi(\mathbf{v}_i)^{\mathrm{T}}\phi(\mathbf{v}_t)| \\
&= \frac{1}{n}\sum_{v_i \in \mathcal{V}} \sum_{v_t \in \mathcal{V}} |\phi(\mathbf{v}_i)^{\mathrm{T}}\phi(\mathbf{v}_t)| \\
&\leq \frac{1}{n}\sum_{v_i \in \mathcal{V}} \sum_{v_t \in \mathcal{V}} |\phi(\mathbf{v}_{max})^{\mathrm{T}}\phi(\mathbf{v}_{max})| \\
&= \frac{1}{n}\sum_{v_i \in \mathcal{V}} \sum_{v_t \in \mathcal{V}} ||\phi(\mathbf{v}_{max})||_2^2 \\
&\overset{\mathrm{c}}{=} S_{\phi(\mathbf{V})}
\end{aligned} \tag{31}$$

where $\overset{\mathrm{c}}{=}$ denotes that minimizing $\frac{1}{n}\sum_{v_i \in \mathcal{V}} \sum_{v_t \in \mathcal{V}} ||\phi(\mathbf{v}_{max})||_2^2$ is equivalent to minimizing $S_{\phi(\mathbf{V})}$.

Combining Eq. 28, Eq. 29 and Eq. 31, we can derive:

$$|\mathcal{L}_G - \mathcal{L}_{\mathcal{P}}^{lc}| \leq L||\mathbf{A}^k - \mathbf{P}\mathbf{P}^{\mathrm{T}}||_F S_{\mathbf{X}} ||\mathbf{W}_P||_2 + S_{\phi(\mathbf{V})}$$

□

A.4 Proof of Proposition 4

Proposition 4. *Let G be a graph with B classes and the classes are balanced. Then, there exists a linear function $g(\cdot) : \mathcal{X}_G \to \mathbb{R}^B$ such that the error upper bound is*

$$\mathbb{E}_v[||y(v) - g(\mathbf{v})||_2^2] \leq 1 + B^2 \sum\nolimits_v (1 + \mathcal{L}_{\mathcal{P}}(v) - \varepsilon(v)) \tag{32}$$

Proof. Let the one-hot label corresponding to $y(v) \in \mathbb{R}^{1\times B}$ be $\mathbf{Y}_v$, and assume there exists a linear mapping matrix $\mathbf{W} \in \mathbb{R}^{d^G \times B}$ that maps node features to labels. For any class i, the number of nodes in that class is $b_i = \frac{n}{B}$, since we assume an ideal class-balanced setting. Then we have:

$$\begin{aligned}
\mathbb{E}_v[||y(v) - g(\mathbf{v})||_2^2] &= \frac{1}{n}||\mathbf{Y} - \mathbf{V}\mathbf{W}||_2^2 \\
&= \frac{1}{n}||\mathbf{Y} - \bar{\mathbf{A}}\mathbf{B} + \bar{\mathbf{A}}\mathbf{B} - \mathbf{V}\mathbf{W}||_2^2 \\
&\leq \frac{1}{n}||\mathbf{Y} - \bar{\mathbf{A}}\mathbf{B}||_2^2 + \frac{1}{n}||\bar{\mathbf{A}}\mathbf{B} - \mathbf{V}\mathbf{W}||_2^2
\end{aligned} \tag{33}$$

where $\mathbf{B}_{v,i} = b_i^{-1}\mathbb{1}[y_v = i]$. For the first term $\frac{1}{n}||\mathbf{Y} - \bar{\mathbf{A}}\mathbf{B}||_2^2$, we can obtain:

$$\begin{aligned}
\frac{1}{n}||\mathbf{Y} - \bar{\mathbf{A}}\mathbf{B}||_2^2 &= \frac{1}{n}||\sum_{v \in \mathcal{V}} \left(\mathbf{Y}_v - \bar{\mathbf{A}}\mathbf{B}\right)||_2^2 \\
&\leq \frac{1}{n}\sum_{v \in \mathcal{V}} ||\mathbf{Y}_v - \bar{\mathbf{A}}\mathbf{B}||_2^2 \\
&= \frac{1}{n}\sum_{v \in \mathcal{V}} \sum_{i=1}^{B} \left(\mathbf{Y}_{v,i} - (\bar{\mathbf{A}}\mathbf{B})_{v,i}\right)^2 \\
&= \frac{1}{n}\sum_{v \in \mathcal{V}} [(1 - \sum_{u \in \mathcal{N}(v)} \frac{1}{b_i}\mathbb{1}[y_v = i])^2 \\
&\quad + (\sum_{u \in \mathcal{N}(v)} \sum_{i=1}^{B} \mathbb{1}[y_v \neq i]\frac{1}{b_i}\mathbb{1}[y_u = i])^2]
\end{aligned} \tag{34}$$

By simplifying the above formula, we can obtain:

$$\begin{aligned}\frac{1}{n}||\mathbf{Y}-\bar{\mathbf{A}}\mathbf{B}||_2^2 &\leq \frac{1}{n}\sum_{v\in\mathcal{V}}[1+\frac{B^2}{n^2}(\sum_{u\in\mathcal{N}(v)}\mathbb{1}[y_v\neq y_u])^2] \\ &\leq \frac{1}{n}\sum_{v\in\mathcal{V}}[1+\frac{B^2}{n^2}\sum_{u\in\mathcal{N}(v)}\mathbb{1}[y_v\neq y_u]] \\ &\leq \frac{1}{n}\sum_{v\in\mathcal{V}}[1+\frac{B^2}{n^2}\frac{n}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)}\mathbb{1}[y_v\neq y_u]] \\ &\leq \frac{1}{n}\sum_{v\in\mathcal{V}}[1+\frac{B^2}{n}(1-\varepsilon(v))]\end{aligned} \tag{35}$$

For the second term $\frac{1}{n}||\bar{\mathbf{A}}\mathbf{B}-\mathbf{V}\mathbf{W}||_2^2$ in Eq. 33, we have:

$$\begin{aligned}\frac{1}{n}||\bar{\mathbf{A}}\mathbf{B}-\mathbf{V}\mathbf{W}||_2^2 &= \frac{1}{n}||\bar{\mathbf{A}}\mathbf{B}-\mathbf{V}\mathbf{V}^{\mathrm{T}}\mathbf{B}+\mathbf{V}\mathbf{V}^{\mathrm{T}}\mathbf{B}-\mathbf{V}\mathbf{W}||_2^2 \\ &= \frac{1}{n}||(\bar{\mathbf{A}}-\mathbf{V}\mathbf{V}^{\mathrm{T}})\mathbf{B}+\mathbf{V}(\mathbf{V}^{\mathrm{T}}\mathbf{B}-\mathbf{W})||_2^2 \\ &\leq \frac{1}{n}||(\bar{\mathbf{A}}-\mathbf{V}\mathbf{V}^{\mathrm{T}})\mathbf{B}||_2^2+\frac{1}{n}||\mathbf{V}(\mathbf{V}^{\mathrm{T}}\mathbf{B}-\mathbf{W})||_2^2\end{aligned} \tag{36}$$

Since the i-th row of $\mathbf{B}$ is the average representation of nodes in class i, we have:

$$\begin{aligned}\frac{1}{n}||\bar{\mathbf{A}}\mathbf{B}-\mathbf{V}\mathbf{W}||_2^2 &\leq \frac{1}{n}||(\bar{\mathbf{A}}-\mathbf{V}\mathbf{V}^{\mathrm{T}})\mathbf{B}||_2^2 \\ &\leq \frac{1}{n}||(\bar{\mathbf{A}}-\mathbf{V}\mathbf{V}^{\mathrm{T}})||_2^2||\mathbf{B}||_2^2 \\ &= \frac{1}{n}\frac{B^2}{n^2}||(\bar{\mathbf{A}}-\mathbf{V}\mathbf{V}^{\mathrm{T}})||_2^2\end{aligned} \tag{37}$$

Recent work has shown that finding the global optimal value of the contrastive loss is equivalent to solving a matrix factorization problem, i.e., $\min||\bar{\mathbf{A}}-\mathbf{V}\mathbf{V}^{\mathrm{T}}||_2^2$ (Balestriero and LeCun 2022). Therefore, combining Proposition 3, our dual-kernel graph community contrastive loss can approximate the contrastive loss of the original graph, and we have:

$$\begin{aligned}\frac{1}{n}||\bar{\mathbf{A}}\mathbf{B}-\mathbf{V}\mathbf{W}||_2^2 &\leq \frac{1}{n}\frac{B^2}{n^2}||(\bar{\mathbf{A}}-\mathbf{V}\mathbf{V}^{\mathrm{T}})||_2^2 \\ &\overset{c}{=} \frac{1}{n}\frac{B^2}{n^2}\mathcal{L}_{\mathcal{P}}(v)\end{aligned} \tag{38}$$

Then, combining Eq. 33, Eq. 35 and Eq. 38, we can derive:

$$\begin{aligned}&\mathbb{E}_v[||y(v)-g(\mathbf{v})||_2^2] \leq \\ &\quad\frac{1}{n}\sum_{v\in\mathcal{V}}[1+\frac{B^2}{n}(1-\varepsilon(v))]+\frac{1}{n}\frac{B^2}{n^2}\mathcal{L}_{\mathcal{P}} \\ &\quad\leq 1+\sum_{v\in\mathcal{V}}[B^2(1-\varepsilon(v)+B^2\mathcal{L}_{\mathcal{P}}(v)] \\ &\quad= 1+B^2\sum_{v\in\mathcal{V}}(1+\mathcal{L}_{\mathcal{P}}(v)-\varepsilon(v))\end{aligned} \tag{39}$$

To this end, we complete the proof of Proposition 4. □

A.5 Proof of Proposition 5

Proposition 5. *Minimizing the distillation loss $\mathcal{L}_D$ is equivalent to maximizing the mutual information between the representation $\mathbf{V}$ and the K-hop pattern Y:*

$$\mathcal{L}_D \geq H(V|Y)-H(V) = -I(Y;V) \tag{40}$$

where V is the random variable corresponding to $\mathbf{V}$.

Proof. Let the distilled node representation $\mathbf{V} = \mathrm{MLP}(\mathbf{X}\mathbf{W}_G)$ and the local neighborhood representation $\mathbf{Z} = 1/K\sum_{k=1}^{K}\widetilde{\mathbf{A}}^k\mathbf{X}\mathbf{W}_G$. Based on the graph homophily assumption, nodes of the same semantic class typically share similar neighborhood representations. Thus, the local neighborhood representation $\mathbf{Z}$ can be viewed as sampled from a standard Gaussian distribution centered at $\mathbf{Z}_Y$, i.e., $Z|Y\sim N(\mathbf{Z}_Y, I)$, where Y denotes the latent semantic class of the K-hop patterns and Z is the random variable corresponding to $\mathbf{Z}$. Then, following (Boudiaf et al. 2020), we can interpret $\mathcal{L}_D$ as the conditional cross-entropy between V and Z, given the pseudo labels Y under the K-hop pattern:

$$\begin{aligned}\mathcal{L}_D &= \frac{1}{n}\sum_{v\in\mathcal{V}}||\mathbf{V}_v-\mathbf{Z}_v||_2^2 \\ &\overset{c}{=} H(V;Z|Y) \\ &= H(V|Y)+\mathcal{D}_{KL}(V||Z|Y) \\ &\geq H(V|Y)\end{aligned} \tag{41}$$

where $\mathcal{D}_{KL}(\cdot|\cdot)$ is the KL divergence. The above equality holds because the KL divergence is non-negative. According to the definition of mutual information, we have:

$$\begin{aligned}\mathcal{L}_D &\geq H(V|Y) \\ &\geq H(V|Y)-H(V) \\ &= -I(Y;V)\end{aligned} \tag{42}$$

The above equality holds because the entropy $H(\cdot)$ is non-negative. Thus, we complete the proof of Proposition 5. □

B. Additional Explanations for GCCL

B.1 Two Variants of GCCL Loss

We consider two kernel functions κ_G and κ_P defined on the node-level space $\mathcal{X}_G$ and community-level space $\mathcal{X}_P$, respectively. Let the corresponding kernel matrices be

$$\kappa_G(\mathbf{v}_i,\mathbf{v}_t) =< \phi_G(\mathbf{v}_i),\phi_G(\mathbf{v}_t) >,\ \ \mathbf{v}_i,\mathbf{v}_t\in\kappa_G \tag{43}$$

and

$$\kappa_P(\mathbf{c}_J,\mathbf{c}_k) =< \phi_P(\mathbf{c}_j),\phi_P(\mathbf{c}_k) >,\ \ \mathbf{c}_j,\mathbf{c}_k\in\kappa_P. \tag{44}$$

Tensor Product Method. We now consider one variant of the GCCL loss based on the tensor product of kernels:

$$\begin{aligned}\kappa_B(\{\mathbf{v}_i,\mathbf{c}_j\},\{\mathbf{v}_t,\mathbf{c}_k\}) &= \kappa_G(\mathbf{v}_i,\mathbf{v}_t)\cdot\kappa_P(\mathbf{c}_j,\mathbf{c}_k) \\ &=< \phi_G(\mathbf{v}_i),\phi_G(\mathbf{v}_t) > \cdot < \phi_P(\mathbf{c}_j),\phi_P(\mathbf{c}_k) > \\ &= \phi_G(\mathbf{v}_i)^{\mathrm{T}}\phi_G(\mathbf{v}_t)\cdot\phi_P(\mathbf{c}_j)^{\mathrm{T}}\phi_P(\mathbf{c}_k) \\ &= (\phi_G(\mathbf{v}_i)\otimes\phi_P(\mathbf{c}_j))^{\mathrm{T}}(\phi_G(\mathbf{v}_t)\otimes\phi_P(\mathbf{c}_k))\end{aligned} \tag{45}$$

where $\otimes$ represents kronecker product. This formulation indicates that the variant of GCCL Loss based on the tensor

product of kernels performs a outer product of the node-level and community-level feature maps, and subsequently uses the resulting product for contrastive loss computation. This method enables full-dimensional interactions across different granularity levels, providing a tight integration of node-level and community-level structural information. Empirical results suggest that this variant is particularly beneficial for node-level tasks on hterophilic graphs.

Linear Combination Method. We now consider another variant of the GCCL loss based on the linear combination of kernels:

$$\begin{aligned}\kappa_B(\{\mathbf{v}_i, \mathbf{c}_j\}, \{\mathbf{v}_t, \mathbf{c}_k\}) &= \alpha\kappa_G(\mathbf{v}_i, \mathbf{v}_t) + \beta\kappa_P(\mathbf{c}_j, \mathbf{c}_k) \\ &= \alpha < \phi_G(\mathbf{v}_i), \phi_G(\mathbf{v}_t) > + \beta < \phi_P(\mathbf{c}_j), \phi_P(\mathbf{c}_k) > \\ &= \alpha\phi_G(\mathbf{v}_i)^{\mathrm{T}}\phi_G(\mathbf{v}_t) + \beta\phi_P(\mathbf{c}_j)^{\mathrm{T}}\phi_P(\mathbf{c}_k) \\ &= \left(\sqrt{\alpha}\phi_G(\mathbf{v}_i) \oplus \sqrt{\beta}\phi_P(\mathbf{c}_j)\right)^{\mathrm{T}} \\ &\qquad \left(\sqrt{\alpha}\phi_G(\mathbf{v}_t) \oplus \sqrt{\beta}\phi_P(\mathbf{c}_k)\right)\end{aligned} \tag{46}$$

where $\oplus$ represents the weighted concatenation of the node-level and community-level feature maps. This method preserves the independence of features at different levels, allowing the model to flexibly adjust their relative importance. Experimental results demonstrate that this variant is advantageous for node-level tasks on homophilic graphs, as it enables the model to emphasize node-level features by assigning a larger value to the coefficient α.

B.2 Model Training

The overall training process of our method is divided into two stages. The first stage trains the GCL model f_Ω via the dual-kernel contrastive loss, and the second stage trains the distillation model f_Φ by minimizing the distance between the local representation and the community representation. The training procedure is provided in Algorithm 1.

Given a graph G with n nodes and m communities, suppose the dimension of the node-level feature space $\mathcal{X}_G$ is d^G and the dimension of the community-level feature space $\mathcal{X}_P$ is d^P. Then, the complexity of obtaining the bi-level pair of features is $O(nd(d^P + d^G))$. By leveraging the kernel trick to linearize the node-level contrastive loss, the complexity is reduced from $O(n^2d^G)$ to $O(nd^G)$. The computational complexity of the community-level contrastive loss is $O(m^2d^P)$. Since $m \ll n$, the dual-kernel contrastive loss has a linear complexity with respect to n.

Algorithm 1: Model Training Procedure

Input: a graph $G = (\mathbf{A}, \mathbf{X})$
Parameter: number of communities m, type of dual-kernel s, training epochs of GCL model T^g and distillation model T^d
Output: final graph representations $\mathbf{Z}^*$
Steps:

Initialize model parameters Ω and Φ.
$\mathcal{P} \leftarrow$ construct the partition of G by Metis.
$\mathbf{A}^{\mathcal{P}} \leftarrow$ construct the community-level graph by $\mathbf{P}^{\mathrm{T}}\mathbf{AP}$.
// Stage 1: Training the GCL model f_Ω.
for $i = 1$ to T^g **do**
  $\{\mathbf{v}, \mathbf{c}\} \leftarrow$ generate the bi-level features via Eq. 6.
  if s = tensor product **then**
    $\mathcal{L}_{\mathcal{P}} \leftarrow$ calculate the contrastive loss via Eq. 8.
  else
    $\mathcal{L}_{\mathcal{P}} \leftarrow$ calculate the contrastive loss via Eq. 9.
  end if
  $\Omega \leftarrow$ updates GCL model parameters with $\mathcal{L}_{\mathcal{P}}$.
end for
// Stage 2: Training the distillation model f_Φ.
for $i = 1$ to T^d **do**
  $\mathbf{V}' \leftarrow$ generate distilled graph representations.
  $\mathcal{L}_D \leftarrow$ calculate the distillation loss via Eq. 13.
  $\Phi \leftarrow$ updates distillation model parameters with $\mathcal{L}_{\mathcal{D}}$.
end for
$\mathbf{Z}^* \leftarrow$ generate final graph representations via Eq. 14.
return $\mathbf{Z}^*$

C. Comparison with Related Methods

C.1 Comparison with StructComp

Our method differs from StructComp (Zhang et al. 2024) in three key aspects.

(i) Framework Design. StructComp generates a coarsened graph by treating each community as a single node, and then computes the graph contrastive loss on this coarsened graph. In contrast, instead of using coarsening techniques to simplify each community into single node, we envision the graph as a network of node sets interconnected between communities, which allows us to preserve essential node information for model training. Figure 4 illustrates the primary difference between our method and StructComp in framework design.

(ii) View Augmentation Strategy. StructComp employs a view augmentation method called DropMember, which randomly drops a portion of the nodes within a community to re-aggregate its features. Our method, however, applies a Dropout operation on each node's features, which can be seen as generating a partitioned substructure in each dimension. Figure 5 highlights the main differences between these two augmentation strategies.

(iii) Contrastive Loss. We leverage MKL techniques to compute the graph community contrastive loss. Notably, when we employ a linear combination of kernels with the parameter α=0, StructComp becomes a special case of our method, as our approach discards node-level information under this condition, while the community-level kernel function is equivalent to computing the graph contrastive loss on this coarsened graph.

Since StructComp is a special case of our method, our method naturally inherits several of its desirable properties. For example, the contrastive loss on the coarsened graph can be viewed as introducing an additional regularization term to the vanilla InfoNCE loss, which enhances the robustness of the encoder against small perturbations (See Theorem 4.2 in StructComp paper).

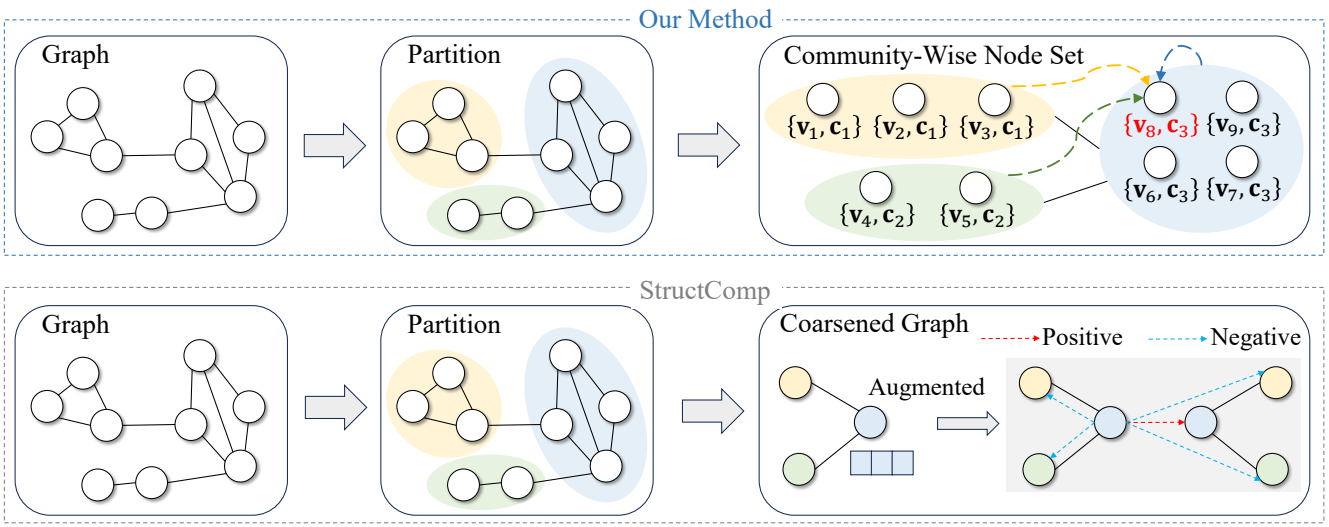

Figure 4: Comparison with StructComp.

C.2 Comparison with Other Kernel Methods

In kernel-based representation learning, $ELU() + 1$ (Qin et al. 2022) and $ReLU()$ (Katharopoulos et al. 2020) are two commonly used kernel functions. They perform well in graph classification tasks, where a key feature of such datasets is their extremely small node scale (e.g., only tens of nodes). However, when applied to large-scale graphs with millions of nodes, these kernel functions often lead to training collapse.

Specifically, in our experiments on Ogbn-Products, replacing the node-level feature map $\phi(v)$ in our graph community contrastive loss with either $ReLU()$ or $ELU() + 1$ resulted in NaN training losses. This is because both kernel functions accumulate positive values in the graph representations across the 2 million nodes, causing the denominator of the contrastive loss to exceed the numerical limits of 32-bit floating-point representation, which ultimately leads to NaN values and unstable training. To mitigate this issue, we adopt $Sigmoid()$ as the kernel function of node-level feature map, which maps the graph representations into a bounded range between 0 and 1. This effectively prevents numerical overflow and ensures stable training even on large-scale graphs.

C.3 Comparison with N2C-Attn

Our work is inspired by N2C-Attn (Huang et al. 2024). However, there are several key differences between the two approaches. N2C-Attn focuses on supervised graph-level tasks, specifically graph classification based on graph Transformer architectures. It employs Multiple Kernel Learning (MKL) to compute attention scores and outputs community-level representations, which are inherently suited only for graph-level tasks. In contrast, our method is designed for unsupervised node-level tasks, where MKL is used to compute the contrastive loss, while also addressing the inference efficiency bottleneck. Furthermore, N2C-Attn adopts $ReLU()$ and $ELU() + 1$ as kernel functions. As discussed in Section C.2, these kernels are unsuitable for large-scale graphs with millions of nodes due to numerical instability and training collapse issues. Therefore, our approach differs fundamentally from N2C-Attn in problem formulation, MKL design, and training objectives.

D. Experimental Study

D.1 Dataset Statistics

Datasets. We evaluate our method on 16 benchmark datasets with different scales and homogeneity levels. including: (i) homophilic graphs: Cora, CiteSeer, PubMed, Wiki-CS, Amazon-Photo, Coauthor-CS, and Coauthor-Physics (Sen et al. 2008; Mernyei and Cangea 2020; Shchur et al. 2018). (ii) 7 heterophilic graphs: Cornell, Texas, Wisconsin, Actor, Crocodile, Amazon-Ratings, and Questions (Pei et al. 2020; Rozemberczki, Allen, and Sarkar 2021; Platonov et al. 2023). (iii) 2 large-scale graphs: Ogbn-Arxiv and Ogbn-Products (Hu et al. 2020). The summary statistics of the graphs are shown in Table 3.

- **Cora**, **CiteSeer** and **PubMed** are three citation network datasets where nodes represent papers, edges represent citation relationships between papers, features consist of bag-of-words representations of papers, and labels correspond to the research topics of the papers.
- **Wiki-CS** is a reference network extracted from Wikipedia, where nodes represent articles on computer science, edges represent hyperlinks between articles, features are average bag-of-words embeddings of the corresponding article contexts, and labels are the specific fields of each article.
- **Amazon-Photo** and **Amazon-Ratings** are two co-purchase networks from Amazon, where nodes represent products, edges represent co-purchase relationships (i.e., two products are frequently bought together), features are bag-of-words representations of product reviews, and labels are product categories.

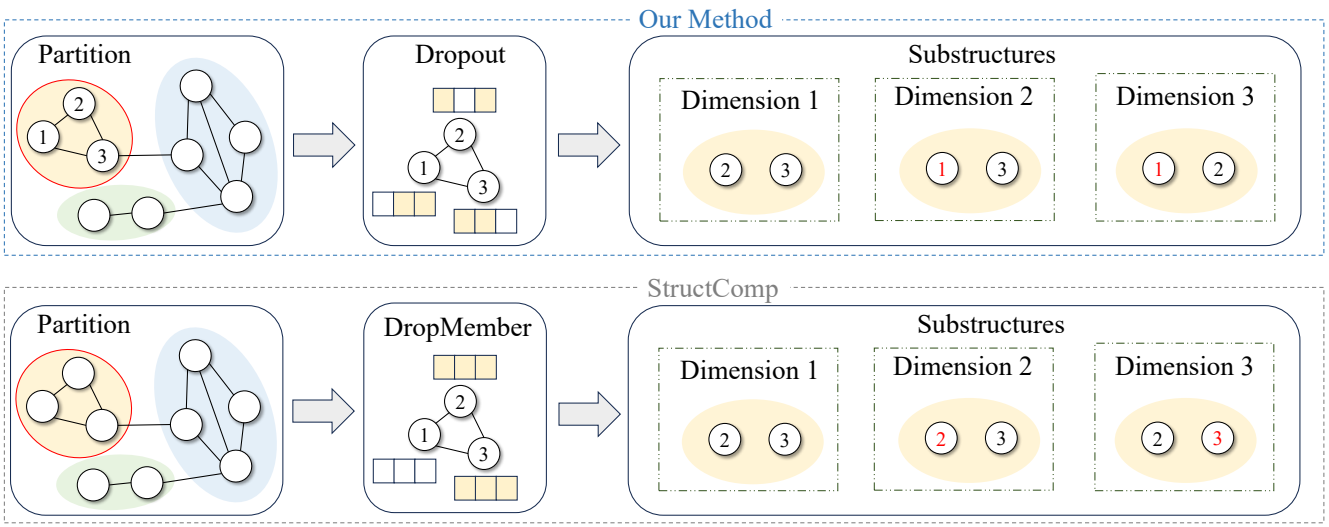

Figure 5: Augmentation strategy comparison with StructComp. Our method can be regarded as constructing a random substructure along each feature dimension, where such diversity of substructures helps enhance generalization ability. In contrast, DropMember directly discards all feature dimensions of certain nodes, which only produces a single substructure.

Dataset	Nodes	Edges	Classes	Features	Homophily Ratio	Train / Valid / Test
Cora	2,708	10,556	7	1,433	0.77	140 /500 / 1,000
CiteSeer	3,327	9,104	6	3,703	0.63	120 / 500 / 1,000
Pubmed	19,717	88,648	3	500	0.66	60 / 500 / 1,000
Wiki-CS	11,701	431,206	10	300	0.57	1,170 / 1,171 / 9,360
Amazon-Photo	7,650	238,162	8	745	0.77	765 / 765 / 6,120
Coauthor-CS	18,333	163,788	15	6,805	0.76	1,833 / 1,834 / 14,666
Coauthor-Physics	34,493	495,924	5	841	0.85	3,449 / 3,450 / 27,594
Cornell	183	295	5	1,703	0.0311	87 /59 /37
Texas	183	309	5	1,703	0.0013	87 /59 / 37
Wisconsin	251	499	5	1,703	0.0941	120 /80 / 51
Actor	7,600	29,926	5	932	0.0110	3,634 /2,432 / 1,520
Crocodile	11,631	360,040	5	2,089	0.0842	6,978 /2,327 / 2,326
Amazon-Ratings	24,492	186,100	5	300	0.1266	12,246 /6,123 / 6,123
Questions	48,921	307,080	2	301	0.0722	24,460 /12,230 / 12,231
Ogbn-Arxiv	169,343	1,166,243	40	128	0.416	90,941 /29,799 / 48,603
Ogbn-Products	2,449,029	61,859,140	47	100	0.459	196,615 /39,323 / 2,213,091

Table 3: The detailed dataset statistics.

- **Coauthor-CS** and **Coauthor-Physics** are two co-author networks extracted from the Microsoft Academic Graph in the KDDCup 2016 challenge, where nodes represent authors, edges represent collaborative relationships, features are bag-of-words representations of paper keywords, and labels are the research fields of the authors.
- **Cornell**, **Texas** and **Wisconsin** are three networks of web pages from different computer science departments, where nodes represent web pages, edges represent hyperlinks between web pages, features are bag-of-words representations of pages, and labels are types of web pages.
- **Actor** is an actor co-occurrence network, where nodes represent actors, edges indicate co-occurrence relationships between two actors in the same film, features are extracted from keywords on Wikipedia pages, and labels are the categories of the corresponding actors.
- **Crocodile** is a Wikipedia network, where nodes represent web pages, edges represent hyperlinks between web pages, features are extracted from page keywords, and labels are the daily traffic of the pages.
- **Questions** is based on data from the question-answering website Yandex Q, where nodes represent users, edges indicate that two users answered the same question within a year, features are descriptions of users, and la-

Dataset	Lr	Epoch	Partition Rate	$k-$Hop	d	α	p	τ
Cora	0.005	15	0.09	3	1,024	0.6	0.1	0.09
CiteSeer	0.05	15	0.07	3	2,048	0.5	0.15	0.04
PubMed	0.0005	75	0.01	2	512	0.5	0.2	0.08
Wiki-CS	0.0005	20	0.02	3	1,024	0.7	0.15	0.08
Amazon-Photo	0.01	25	0.03	5	1,024	0.6	0.15	0.05
Coauthor-CS	0.005	50	0.09	1	1,024	0.8	0.1	0.08
Coauthor-Physics	0.0005	20	0.04	1	2,048	0.7	0.3	0.10
Cornell	0.0005	20	0.2	0	8,192	-	0.1	0.03
Texas	0.0001	20	0.05	0	8,192	-	0.5	0.04
Wisconsin	0.005	50	0.09	0	4,096	-	0.55	0.06
Actor	0.01	5	0.09	0	2,048	-	0.55	0.03
Crocodile	0.05	5	0.02	0	8,192	-	0.5	0.09
Amazon-Ratings	0.001	50	0.06	2	8,192	-	0.55	0.09
Questions	0.005	10	0.007	5	8,192	-	0.55	0.05
Ogbn-Arxiv	0.0005	25	0.007	10	800	0.9	0.1	0.03
Ogbn-Products	0.001	25	0.0001	10	128	0.9	0.05	0.06

Table 4: Details of the hyper-parameters of our method.

bels are the activity levels of users.

- **Ogbn-Arxiv** and **Ogbn-Products** are two large-scale datasets. Ogbn-Arxiv is a citation network, where nodes represent papers, edges represent citation relationships between papers, features are extracted from titles and abstracts, and labels correspond to the research topics of the papers. Ogbn-Products is a co-purchase network, where nodes represent products, edges represent co-purchase relationships, features are bag-of-words representations of product reviews, and labels are product categories

Splitting Strategies. For the Cora, CiteSeer and PubMed datasets, we randomly select 20 nodes per class for training, 500 nodes for validation, and 1,000 nodes for testing (Veličković et al. 2019). For the other 4 homophilic datasets, we follow previous works and adopt the public $10\%/10\%/80\%$ training/validation/testing split (Liu et al. 2023). For the heterophilic and large-scale datasets, we use the standard splits provided by PyTorch Geometric (Sun et al. 2024; Chen, Lei, and Wei 2024).

D.2 Baselines

GCL exhibits excellent capability in learning graph representations without task-specific labels, with its core idea being to leverage contrastive loss based on mutual information (MI) maximization to distinguish between positive and negative node pairs, thereby training GNNs.

- **DGI** is a foundational GCL method that maximizes MI between node representations and graph summary.
- **GCA** enhances GCL by incorporating adaptive augmentation based on rich topological and semantic priors.
- **gCooL** utilizes community information to construct positive and negative node pairs required for GCL.
- **CSGCL** adjusts the weight of contrastive samples based on community strength.
- **SP-GCL** exploits the centralized nature of node representation, eliminating the need for graph augmentation.
- **GraphECL** improves inference efficiency based on the coupling model of MLP and GNN.
- **SGRL** enhances the diversity of graph representation through a center-away strategy.

Recent studies improve the scalability by simplifying the steps of view encoding or loss calculation in GCL.

- **BGRL** is a GCL method that learns by predicting alternative augmentations of the input.
- **SUGRL** removes widely used data augmentation and discriminator from previous GCL methods.
- **GGD** adopts a binary cross-entropy loss to distinguish between the two groups of node samples
- **SGCL** utilizes the outputs from two consecutive iterations as positive pairs, eliminating the negative samples.
- **StructComp** performs contrastive learning on the constructed coarsened graph to improve scalability.
- **E2Neg** leverages a small number of representative samples to learn discriminative graph representations.

There are also some methods that explore the potential of GCL on heterophilic graphs.

- **HGRL** learns node representations by preserving original features and capturing informative distant neighbors.
- **L-GCL** samples positive examples from the neighborhood and adopts kernelized loss to reduce training time.
- **DSSL** uses latent variable modeling to decouple different neighborhood contexts without data augmentation.
- **GREET**earns node representations by distinguishing homophilic and heterophilic edges.
- **GraphACL** captures two-hop monophily similarities without relying on homophily assumptions.

- **PolyGCL** leverages polynomial filters to generate low-pass and high-pass spectral augmented views,
- **M3P-GCL** uses the macro-micro message passing to improve performance on heterophilic graphs.

D.3 Parameter Settings

We adopt the officially released implementations provided by the authors as baselines and use the hyperparameters specified in their original papers. To ensure fair comparisons, for baselines without reported settings on specific datasets, we perform grid search to carefully tune their hyperparameters. Each dataset is evaluated over 10 different random splits to ensure robustness. All experiments are conducted on a Windows 11 machine equipped with an Intel i9-10900X CPU, 128GB RAM, and an NVIDIA 3090 GPU (24GB memory).

We implement our method in PyTorch with Adam optimizer, with a one-layer linear layer as the encoder and a two-layer MLP as the distillation model. The learning rate lr is selected from $0.0001, 0.0005, 0.001, 0.002, 0.005, 0.1$. The number of training epochs is chosen from $5, 10, 15, 20, 25, 50, 75$. The partition rate is adjusted based on the number of nodes and classes, and setting the number of communities to tens of times the actual number of classes typically yields optimal results. The order of the diffusion matrix k is selected from $0, 1, 2, 3, 4, 5$, and for complex graphs like Ogbn-Arxiv and Ogbn-Products, k is set to 10. The node-level and community-level feature dimensions d are the same, chosen from $512, 1024, 1500, 2048, 4096, 8192$. Considering computational cost, we set $d = 800$ for Ogbn-Arxiv and $d = 128$ for Ogbn-Products. The dropout rate p ranges from 0 to 0.6. The combination coefficient α is selected from the range $[0.5, 1]$, and the temperature coefficient τ is selected from $[0.01, 0.1]$. The hyperparameters for each dataset are summarized in Table 4. More detailed settings can be found in the released code.

D.4 Additional Experimental Results

In this subsection, we provide additional experiments, including node clustering tasks, ablation studies, and analysis of parameter influences.

Exp-4: Node Clustering. We selected several methods that perform well on node classification tasks and compared them in node clustering task, where K-Means refers to clustering directly on raw node features. The results are shown in Table 5.

These results demonstrate that: (i) Our method outperforms other baselines on most datasets, which can be attributed to its ability to leverage both intra- and inter-community information. (ii) gCooL and CS-GCL also achieve strong performance in the clustering task, further highlighting the importance of community-level information in node representation learning This means that the node representations generated by our method can be extended to other node-level tasks..

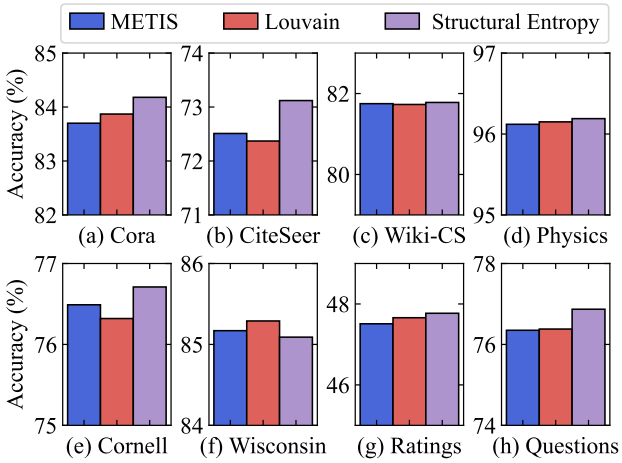

Figure 6: Impacts of partition methods.

Exp-5: Impacts of Graph Partition. We analyze the impact of graph partition on performance. First, we we compared several representative partition algorithms, including Louvain (Blondel et al. 2008), Structural Entropy (SE) (Li 2024), and Metis used in our experiments. Then, we investigate the effect of varying the number of communitys. The results are shown in Figures 6 and 7, respectively.

These results demonstrate that: (i) Our method is compatible with various graph partition algorithms. In general, more advanced algorithms tend to yield better performance (i.e., SE). Considering the complexity of partitioning, we recommend using SE for medium-scale graphs and using the more efficient algorithm Metis for large-scale graphs. (ii) The performance of our method varies with the compression ratio and exhibits a hump-shaped curve. If the number of communitys is too small, excessive compression may degrade performance, while more communitys do not bring better performance. Based on dataset statistics, we find that setting the number of communitys to tens of times the actual number of classes typically yields optimal results.

Exp-6: Ablation Studies. We conducted an ablation study to evaluate the contributions of several key components, as shown in Table 6. The specific ablation settings include: (a) Removing the dropout operator (w/o Dropout), (b) Removing the graph convolution operator (w/o GC) and (c) Removing the representation distillation operator (w/o $\mathcal{L}_D$).

These results demonstrate that: (i) All components contribute to the performance of our method. Although the representation distillation module has a relatively minor impact on performance, it is of great value in significantly improving inference efficiency. (ii) Local information is crucial for improving the accuracy of node classification on homophilic graphs, but not always effective on heterophilic graphs. (iii) Knowledge distillation techniques may have limitations on large-scale graphs. (iv) Even after removing the GC operator, our method still significantly outperforms a pure MLP, which further proves its effectiveness in capturing high-order structural information.

Methods	Cora		CiteSeer		Wiki-CS		Amz.Photo		Co.CS		Co.Physics	
	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
K-Means	8.66	4.81	22.45	20.26	25.71	15.02	25.77	14.51	60.12	40.37	48.94	27.59
gCooL	52.83	46.15	40.32	39.04	38.24	26.88	56.60	43.14	75.32	62.07	65.19	57.81
CSGCL	43.42	34.13	40.76	41.96	37.17	12.11	58.81	46.33	77.12	63.57	66.13	58.29
SP-GCL	28.29	16.62	37.67	36.12	16.33	5.81	28.54	15.17	62.37	44.12	65.43	45.97
GraphECL	52.10	42.39	25.29	22.14	34.76	19.44	49.68	29.41	74.37	61.59	63.17	60.22
SGRL	46.94	36.84	43.03	43.52	33.27	16.59	33.65	17.75	77.41	65.73	60.88	55.70
SUGRL	56.34	48.43	41.97	42.94	35.27	21.86	59.62	49.77	76.62	62.53	65.69	60.37
GREET	55.18	49.71	43.13	42.58	37.36	22.21	52.33	37.08	75.79	62.13	66.37	63.62
SGCL	54.83	48.02	39.66	39.17	39.97	17.61	52.76	38.80	59.49	52.31	**69.14**	**68.50**
E2Neg	23.21	8.63	36.09	34.69	29.65	13.69	33.75	17.43	75.23	57.52	59.15	44.83
Ours	**59.56**	**51.23**	**43.81**	**44.49**	**40.91**	**28.93**	**60.17**	**50.09**	**79.66**	**66.75**	67.25	67.34

Table 5: Node clustering results measured by NMI (%) and ARI (%).

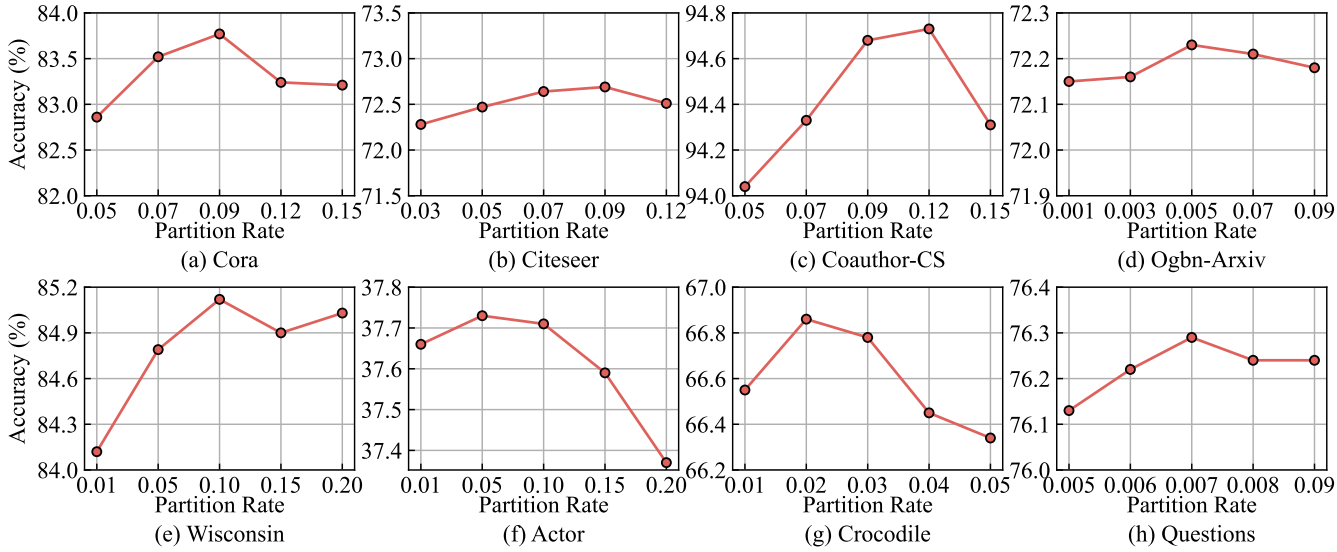

Figure 7: Impact of partition rate.

Exp-7: Sensitivity of Parameters. We investigate the influence of the dropout rate p, embedding dimension d, and the combination coefficient α, as shown in Figure 8.

These results demonstrate that: (i) On homophilic graphs, the optimal dropout rate typically falls between 0.1 and 0.3, whereas on terophilic graphs, values of p greater than 0.3 yield better performance. This suggests that promoting substructure diversity is more effective for complex graphs, and such diversity can also reduce the training cycles (as shown in Table 4, our method requires at most 75 training epochs). (ii) A larger embedding dimension d generally improves node classification accuracy, particularly on terophilic graphs. However, on homophilic graphs, extremely large dimensions may lead to overfitting, resulting in a slight performance drop. (iii) On homophilic graphs, the combination coefficient α is typically greater than 0.5, implying that node-level information should be emphasized more heavily for node classification tasks.

References

Balestriero, R.; and LeCun, Y. 2022. Contrastive and Non-contrastive Self-supervised Learning Recover Global and Local Spectral Embedding Methods. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems* (NeurIPS), 26671–26685.

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008.

Boudiaf, M.; Rony, J.; Ziko, I. M.; Granger, E.; Pedersoli, M.; Piantanida, P.; and Ayed, I. B. 2020. A Unifying Mutual Information View of Metric Learning: Cross-Entropy vs. Pairwise Losses. In *Proceedings of the 16th Europran Conference on Computer Vision* (ECCV), 548–564.

Chen, J.; Lei, R.; and Wei, Z. 2024. PolyGCL: Graph Cntrastive Learning via Learnable Spectral Polynomial Fil-

Variants	Cora	CiteSeer	PubMed	Wiki-CS	Amz.Photo	Co.CS	Co.Physics
MLP	56.11±0.34	56.91±0.42	71.35±0.73	72.02±0.21	78.54±0.05	90.42±0.08	93.54±0.05
GCN	81.60±1.37	70.3±1.15	79.00±0.78	76.87±0.37	92.35±0.25	93.10±0.17	95.54±0.18
(w/o Do)	83.23±1.37	72.23±1.57	82.12±1.71	81.37±0.25	93.41±0.28	94.28±0.13	95.96±0.12
(w/o GC)	74.79±1.34	70.22±1.56	75.35±1.84	75.22±0.49	89.33±0.34	93.07±0.21	95.29±0.08
(w/o $\mathcal{L}_D$)	83.58±1.56	71.82±1.48	81.45±2.36	81.42±0.40	93.77±0.23	94.33±0.14	**96.14±0.16**
Ours	**83.77±1.37**	**72.68±1.19**	**82.56±1.85**	**81.75±0.36**	**93.86±0.15**	**94.68±0.14**	96.12±0.17

Variants	Cornell	Texas	Wisconsin	Actor	Crocodile	Amz.Ratings	Questions
GCN	57.03±3.30	60.00±4.80	56.47±6.55	30.83±0.77	66.72±1.24	**48.70±0.63**	76.09±1.27
(w/o Do)	76.11±3.07	84.59±4.37	84.92±4.49	37.21±0.78	66.63±0.64	46.95±0.65	76.29±1.06
(w/o GC)	-	-	-	-	-	41.15±0.61	70.56±1.01
(w/o $\mathcal{L}_D$)	-	-	-	-	-	47.02±0.73	76.17±1.08
Ours	**76.49±2.43**	**85.41±3.01**	**85.17±3.02**	**37.74±0.78**	**67.05±0.72**	47.51±0.68	**76.35±1.05**

Table 6: Ablation study on medium-scale datasets. '-' indicates that we do not use graph convolution operators.

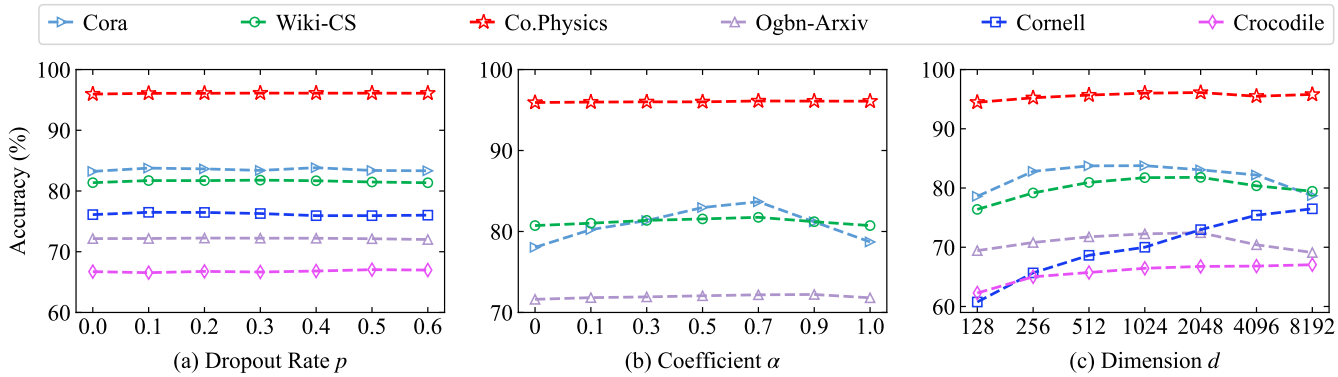

Figure 8: Impacts of dropout rate p, combination coefficient α and embedding dimension d.

ters. In *Proceedings of the 12th International Conference on Learning Representations* (ICLR).

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems* (NeurIPS), 22118–22133.

Huang, S.; Song, Y.; Zhou, J.; and Lin, Z. 2024. Cluster-wise Graph Transformer with Dual-granularity Kernelized Attention. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems* (NeurIPS), 33376–33401.

Katharopoulos, A.; Vyas, A.; Pappas, N.; and Fleuret, F. 2020. Transformers are Rnns: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning* (ICML), 5156–5165. PMLR.

Li, A. 2024. *Science of Artificial Intelligence: The mathematical principles of intelligence*. Science Press.

Liu, Y.; Zheng, Y.; Zhang, D.; Lee, V. C.; and Pan, S. 2023. Beyond Smoothing: Unsupervised Graph Representation Learning with Edge Heterophily Discriminating. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence* (AAAI), 4516–4524.

Mernyei, P.; and Cangea, C. 2020. Wiki-cs: A Wikipedia-based Benchmark for Graph Neural Networks. *arXiv preprint arXiv:2007.02901*.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *Proceedings of the 9th International Conference on Learning Representations* (ICLR).

Platonov, O.; Kuznedelev, D.; Diskin, M.; Babenko, A.; and Prokhorenkova, L. 2023. A Critical Look at the Evaluation of GNNs under Heterophily: Are We Really Making Progress? In *Proceedings of the 12th International Conference on Learning Representations* (ICLR).

Qin, Z.; Sun, W.; Deng, H.; Li, D.; Wei, Y.; Lv, B.; Yan, J.; Kong, L.; and Zhong, Y. 2022. cosFormer: Rethinking Softmax In Attention. In *Proceedings of the 10th International Conference on Learning Representations* (ICLR).

Rozemberczki, B.; Allen, C.; and Sarkar, R. 2021. Multi-

scale Attributed Node Embedding. *Journal of Complex Networks*, 9(2): cnab014.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *AI magazine*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of Graph Neural Network Evaluation. *arXiv preprint arXiv:1811.05868*.

Sun, W.; Li, J.; Chen, L.; Wu, B.; Bian, Y.; and Zheng, Z. 2024. Rethinking and Simplifying Bootstrapped Graph Latents. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (WSDM), 665–673.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *Proceedings of the 7th International Conference on Learning Representations* (ICLR).

Zhang, S.; Yang, W.; Cao, X.; Zhang, H.; and Huang, Z. 2024. StructComp: Substituting Propagation with Structural Compression in Training Graph Contrastive Learning. In *Proceedings of the 12th International Conference on Learning Representations* (ICLR).