

Appendix

A Related Work [Extended]

A.1 Self-supervised Learning on Graphs

Graph self-supervised learning has been a promising paradigm for learning representations without labels [46, 12, 11]. Early studies on graph representation often utilize conventional network embedding techniques, such as random walks or graph reconstruction [51, 1], which over-emphasize node proximity while inevitably losing some topological information. As a result, various graph contrastive learning methods [18, 59, 53] have been proposed to perform self-supervised learning on graphs, which aims to learn representations by following the principle of maximizing similarity between positive sample pairs and have achieved promising performance. However, such methods are built upon a strong homophily assumption and perform poorly under heterophilic graphs. Until recently, people started to explore the heterophily issue in the graph self-supervised learning framework. These methods generate reasonably perturbed views by leveraging random augmentation strategies [27, 48, 56] based on node/edge dropping and feature masking or carefully designed augmentation strategies [44, 7, 6], then align the representations of the augmented positive pairs. However, their performance heavily relies on effective augmentation strategies, where perturbations to the topology can significantly alter the semantic relationships of neighbors. Differentially, instead of following the "augmentation-contrast" learning paradigm, we perform SSL by assigning distinct colors to different types of nodes, fully preserving the inherent structural properties of heterophilic graphs.

A.2 Graph Coloring

Graph coloring problem (GCP) is one of the most classical optimization problems in graph theory, with a wide range of applications across various domains, such as air traffic flow management [2], register allocation [55], and job scheduling [5]. The objective of graph coloring is to assign colors (i.e., labels) to the nodes of a graph such that no two adjacent nodes share the same color, while minimizing the total number of colors used. With the growing prevalence of Graph Neural Networks (GNNs), recent studies have explored the use of GNNs' message-passing mechanisms to tackle the GCP by incorporating penalty terms that enforce distinct color assignments for neighbor nodes. For example, Schuetz et al. [37] formulates the graph coloring task as a multi-class node classification problem, leveraging GNNs to learn node representations and adopting the Potts model from statistical physics for unsupervised learning. Similarly, Li et al. [24] introduces a boundary loss to encourage dissimilar embeddings for adjacent nodes, thereby achieving effective coloring. However, these approaches are typically designed to approximate the classical GCP directly and mainly emphasize enforcing distinct colors for adjacent nodes. They are less suited for capturing more complex structural interactions, particularly the interplay of homophilic and heterophilic relations commonly present in real-world graphs. In contrast, our work does not attempt to directly solve the classical GCP. Instead, we draw heuristic inspiration from the coloring concept and extend it to heterophilic graph representation learning. Specifically, we leverage a learnable edge evaluator to dynamically guide the coloring process, encouraging nodes to share the same or different colors according to their homophilic or heterophilic relations. This relaxation from the strict GCP constraint allows our model to simultaneously capture both affinity and disparity between nodes, thereby uncovering the underlying dependencies and semantics in heterophilic graphs.

B Algorithm

The algorithm of CoRep is elaborated in Algorithm 1. As we can see, the position encodings and the multi-hop neighbor node set can be pre-computed in the initialization phase. In each iteration, we first estimate the homophily score between node pairs in the graph using an edge evaluator. Next, we employ a graph encoder to obtain node representations, followed by the edge-aware coloring matching learning stage. In this phase, a coloring classifier is pioneered to generate coloring labels for nodes, which are then discretized using the Gumbel-Softmax technique to obtain node colors. During this training, we compute three losses: the coloring matching loss, coloring redundancy constraint, and triplet relation ranking loss, corresponding to Equations (7), (8), and (10), respectively. Furthermore, a global positive sample set is constructed based on Equation (11), and aligned using

Algorithm 1 CoRep

Input: Graph $\mathcal{G} = (\mathcal{A}, \mathcal{X})$, graph encoder f_θ , edge evaluator f_ε , coloring classifier f_φ , and projector f_v .

Output: Trained encoder f_θ and node representations $\{\vec{h}_i\}_{i \in \llbracket n \rrbracket}$.

- 1: Initialize parameters of f_θ , f_ε , f_φ , and f_v ;
 - 2: Initialize position encodings \mathbf{p}_i via Equation (1);
 - 3: Initialize \varkappa -hop neighbor node set $\mathcal{N}^{(\varkappa)}(v_i)$;
 - 4: **while** not converged **do**
 - 5: Evaluate the homophily score $\hat{\omega}_{i,j}$ by f_ε ;
 - 6: Encode \mathcal{G} by f_θ to get $\{\vec{h}_i\}_{i \in \llbracket n \rrbracket}$;
 - 7: Predict \mathcal{G} by f_φ to get $\{\pi_i\}_{i \in \llbracket n \rrbracket}$;
 - 8: Discretize $\{\pi_i\}_{i \in \llbracket n \rrbracket}$ by Gumbel-Softmax to get $\{\mathcal{C}_i^{col}\}_{i \in \llbracket n \rrbracket}$;
 - 9: Calculate the coloring matching loss and coloring redundancy constraint via Equations (7) and (8);
 - 10: Calculate the triplet relation ranking loss via Equation (10);
 - 11: Project $\{\vec{h}_i\}_{i \in \llbracket n \rrbracket}$ by f_v to get $\{\mathbf{z}_i\}_{i \in \llbracket n \rrbracket}$;
 - 12: Generate global positive sample set $\mathcal{P}_G(v_i)$ via Equation (11);
 - 13: Calculate multi-hop neighborhood contrastive loss \mathcal{L}_c via Equation (12);
 - 14: Calculate the overall loss \mathcal{L} via Equation (13) and update model's parameters by minimizing \mathcal{L} .
 - 15: **end**
 - 16: **return** Trained encoder f_θ .
-

a multi-hop neighborhood contrastive loss to enhance global structural consistency. Finally, we aggregate the above loss terms into a total loss, which is minimized to update the model's parameters.

C Complexity [Extended]

Based on Algorithm 1, we analyze the time complexity of our algorithm. Let $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of nodes and edges in the graph, respectively, and let d and $\chi_{\mathcal{G}}$ be the dimensionality of the original node features and coloring labels, respectively, where $\chi_{\mathcal{G}}$ is typically a small value (e.g., $\chi_{\mathcal{G}} \leq 20$). During preprocessing, both positional encodings and multi-hop neighborhoods can be computed and stored in advance. By leveraging sparse matrix multiplications, their complexity is $\mathcal{O}(|\mathcal{V}||\mathcal{E}|d^\sharp)$ and $\mathcal{O}(|\mathcal{V}||\mathcal{E}|\varkappa)$, where d^\sharp is the dimensionality of the positional encodings, and \varkappa is the number of hops of neighbor nodes. Considering the graphs are typically sparse (i.e., $|\mathcal{E}| \ll |\mathcal{V}|^2$), and d^\sharp and \varkappa are set to the relatively small values ($d^\sharp = 16$ and $\varkappa \leq 4$), the computational cost here is acceptable. Our algorithm consists of two key components: the edge evaluator and the graph encoder. For the edge evaluator, computing the homophily probability incurs a cost of $\mathcal{O}(|\mathcal{V}|d^\circ(d + d^\sharp) + |\mathcal{E}|d^\circ)$, where d° denotes the hidden dimension in the evaluator. The computation cost of the graph encoder is $\mathcal{O}(Ld^\dagger(|\mathcal{V}| + |\mathcal{E}|))$, where d^\dagger is the dimensionality of the final node embeddings and L is the number of layers (typically set to 2). The algorithm incorporates three core loss functions: coloring matching loss, triplet relation ranking loss, and multi-hop neighborhood contrastive loss. Their respective computational complexities are $\mathcal{O}(\chi_{\mathcal{G}}|\mathcal{E}|)$, $\mathcal{O}(\chi_{\mathcal{G}}|\mathcal{E}|)$, and $\mathcal{O}(|\mathcal{V}|^2\kappa d^\sharp)$, where κ denotes the average number of positive samples and $\kappa \ll |\mathcal{V}|$ and d^\sharp is the dimensionality of the projected node embeddings. For slightly large-scale graphs, we adopt a mini-batch strategy to reduce the computational cost of multi-hop neighborhood contrastive loss to $\mathcal{O}(|\mathcal{V}|b\kappa d^\sharp)$, where b is the batch size of contrastive loss.

D Datasets

We evaluate our algorithm on 8 homophilic graph datasets (i.e., Cora, CiteSeer, PubMed, Wiki-CS, Amazon Computers, Amazon Photo, CoAuthor CS, and CoAuthor Physics) and 6 heterophilic graph datasets (i.e., Chameleon, Squirrel, Actor, Cornell, Texas, and Wisconsin). The statistics of all datasets are summarized in Table 4. Their details are shown as follows:

Table 4: Statistics of datasets.

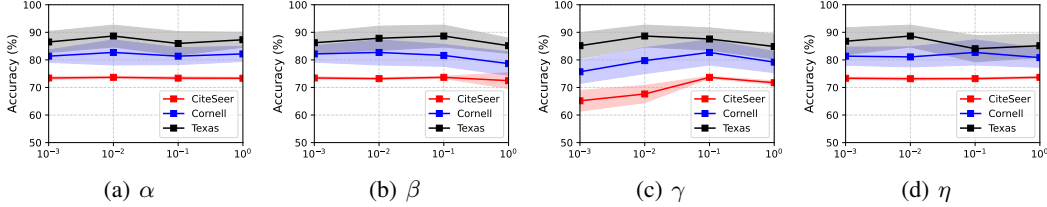
Dataset	#Nodes	#Edges	#Classes	#Features	#Homophily
Cora	2,708	10,556	7	1,433	0.810
CiteSeer	3,327	9,104	6	3,703	0.736
PubMed	19,717	88,648	3	500	0.802
Wiki-CS	11,701	431,206	10	300	0.654
Amazon Computers	13,752	491,722	10	767	0.777
Amazon Photo	7,650	238,162	8	745	0.827
CoAuthor CS	18,333	163,788	15	6,805	0.808
CoAuthor Physics	34,493	495,924	5	8,415	0.931
Chameleon	2,277	36,051	5	2,325	0.234
Squirrel	5,201	216,933	5	2,089	0.223
Actor	7,600	29,926	5	932	0.216
Cornell	183	295	5	1,703	0.298
Texas	183	309	5	1,703	0.061
Wisconsin	251	499	5	1,703	0.170

- **Cora**, **CiteSeer**, and **PubMed** [38] are 3 widely used citation network datasets, where nodes denote papers and edges denote citation relationships between papers. Each paper is represented as a feature vector in the form of a bag-of-words, and its corresponding labels are the research topic of this paper.
- **Wiki-CS** [29] is a reference network derived from Wikipedia, where nodes indicate articles about computer science and edges indicate the hyperlinks between two articles. The features of each node are obtained by averaging the GloVe word vectors of the content in the corresponding article, while labels are different fields of each article.
- **Amazon Computers** and **Amazon Photo** [39] are 2 co-purchased networks from the Amazon platform. In both graphs, each node represents a good, and the edges indicate that the two goods are frequently bought together. The features of each node are constructed using a bag-of-words representation of reviews, and the labels are the categories of goods.
- **CoAuthor CS** and **CoAuthor Physics** [39] are 2 co-authorship networks originating from Microsoft Academic Graph in the KDD Cup 2016 challenge. In these graph structures, nodes correspond to authors and edges indicate the co-authorship relationship between authors. The features are generated through bag-of-words embeddings of keywords from published papers, and the labels are the authors' research fields.
- **Chameleon** and **Squirrel** [31] are 2 Wikipedia networks, where nodes indicate web pages in Wikipedia and edges indicate linked relationships between two pages. The node features contain the informative nouns in the pages, while the labels correspond to the average traffic of the web pages.
- **Actor** [31] is an actor co-occurrence network, where nodes represent actors and edges represent the relationship between two actors co-occurring in the same movie. The node features are derived from the keyword information in the Wikipedia pages, and labels are the words of the corresponding actors.
- **Cornell**, **Texas**, and **Wisconsin** [31] are 3 web page networks from computer science departments of different universities, where nodes denote web pages and edges denote hyperlinks between two web pages. The node features are represented by the bag-of-words vectors of the corresponding web pages, and the labels are types of web pages.

For Cora, CiteSeer, and PubMed datasets, we adopt the public splits with 20 labeled nodes per class for training, 500 nodes for validation, and 1000 nodes for testing [52, 27]. For Wiki-CS, Amazon Computers, Amazon Photo, CoAuthor CS, and CoAuthor Physics datasets, we adopt the 10%/10%/80% training/validation/testing splits following previous work [60, 27]. For heterophilic graph datasets, we adopt the commonly used 48%/32%/20% training/validation/testing splits [31].

Table 5: Ablation experiments on more datasets.

Ablation	Chameleon	Photo	CS
A1 w/o Col. Mat.	62.78 \pm 7.16	91.32 \pm 2.36	93.18 \pm 0.35
A2 w/o Ed. Eva.	64.52 \pm 1.73	88.71 \pm 9.53	94.19 \pm 0.12
A3 w/o Gum. Soft.	65.18 \pm 1.21	92.24 \pm 3.28	93.27 \pm 2.44
A4 w/o \mathcal{L}_d	65.37 \pm 1.70	93.74 \pm 1.83	94.21 \pm 0.28
A5 w/o \mathcal{L}_r	65.31 \pm 2.10	93.79 \pm 1.82	94.18 \pm 0.22
A6 w/o \mathcal{L}_c	51.64 \pm 7.14	89.62 \pm 3.50	89.84 \pm 4.25
A4+A5	64.77 \pm 1.38	93.20 \pm 1.92	93.63 \pm 0.40
A4+A6	51.36 \pm 8.32	88.92 \pm 3.19	88.56 \pm 3.91
A5+A6	50.96 \pm 7.21	88.90 \pm 2.84	88.00 \pm 4.40
CoRep	65.64\pm1.39	93.84\pm1.89	94.39\pm0.31

Figure 4: Parameter sensitivity of α , β , γ , and η .

E Experimental Details

Hyperparameters. In the experiments, we use the Adam optimizer with the learning rate tuned in $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$. We experiment with weight decay, the dimensions of representation and projection, scaling hyperparameter ξ , the number of colors χ_G , the number of neighbor hops κ , and trade-off coefficients α , β , γ , and η in the sets $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, $\{64, 128, 256, 512\}$, $\{0.1, \dots, 0.5\}$, $\{5, 10, 15, 20\}$, $\{1, 2, 3, 4\}$, and $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$. We set the temperature parameters τ_m , τ_o , and τ_c to 1.0, 1.0, and 0.2, respectively. The dimensionality of the position encoding $d^{\#}$ is set to 16, and the hidden dimension of the edge evaluator d^o is set to 128.

Implementation Details. In our model, the nonlinear feature mapping layers φ_1 , φ_2 , ψ_1 , and ψ_2 , the classifier f_φ and the projector f_v are implemented by a single MLP layer. We use the linear evaluation protocol that is widely used in previous approaches [50, 6]. Specifically, in the training phase, the model is trained without any labels; then, in the evaluation phase, the learned representations are frozen and trained and tested with a logistic regression classifier. For fair comparison, the parameters of all baselines are tuned according to the parameter ranges reported by the authors in the literature to get preferable performance in most situations. All experiments are conducted on the Linux server with 8 Intel(R) Xeon(R) CPUs (E5-2667 v4 @ 3.20GHz) and 2 NVIDIA RTX A6000.

F More Experimental Results

F.1 Additional Ablation Study

To further validate the effectiveness of each component, we conduct ablation studies on more datasets. As shown in Table 5, each module positively impacts overall performance. Specifically, we observe that removing A1 leads to a performance drop across all three datasets. The results on the Chameleon and Photo datasets are particularly unstable, indicating A1’s importance in the model. Similarly, drops caused by removing A2 and A3 highlight the key roles of the edge evaluator and the Gumbel-Softmax trick. The performance degradation in A4, A5, and A6 highlights the critical role of the loss terms \mathcal{L}_d , \mathcal{L}_r , and \mathcal{L}_c in CoRep, as they are essential for maintaining intra-class compactness, edge discriminability, and global structural consistency, respectively. We also observe that the combination of losses yields better results compared to using each loss individually. The complete model (last

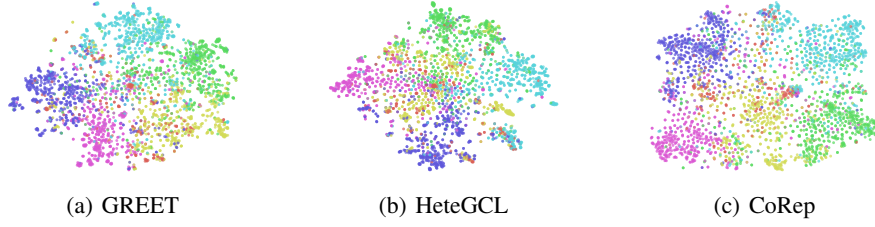


Figure 5: Visualization of node representations on the CiteSeer dataset.

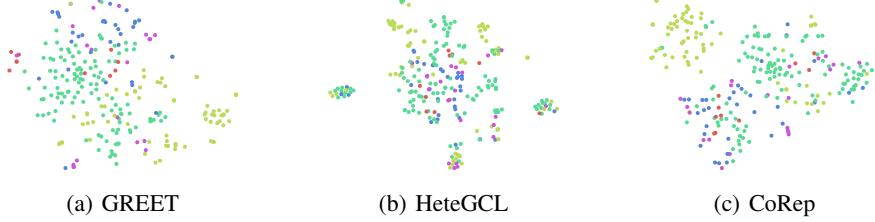


Figure 6: Visualization of node representations on the Wisconsin dataset.

row) achieves the best performance, demonstrating the complementary and synergistic effects of all components.

F.2 Additional Parameter Analysis

Effect of the Trade-off Hyperparameters α , β , γ , and η . To analyze the effect of the loss terms, we vary trade-off hyperparameters α , β , γ , and η across a wide range: $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$. The corresponding results are shown in Figure 4. We observe that the model achieves optimal accuracy when α , β , γ , and η are selected from $\{10^{-2}, 10^{-1}, 10^0\}$. We can find that the model’s performance on the Cornell and Texas datasets is relatively sensitive to changes in these hyperparameters. A possible explanation is that the heterophilic graph structures in these datasets are inherently more difficult to capture. We also find noticeable fluctuations in accuracy across all datasets as γ varies, indicating that the discriminability of node representations is relatively sensitive to the global structural consistency captured by the multi-hop neighborhood contrastive loss.

F.3 Visualization

To study which patterns CoRep learns in node representations, we map the feature representations extracted by CoRep on the CiteSeer and Wisconsin datasets to a two-dimensional space via t-SNE to visualize in Figures 5 and 6. In these projections, different colors represent different classes of nodes, allowing a clear comparison of the extent to which each method preserves the inherent class structure in the data. From Figures 5 and 6, we can see that the visualization obtained from CoRep is much more prominent, as it shows clearer class boundaries and compact intra-class structure. This result emphasizes the superior representation learning capability of our proposed CoRep approach.

F.4 Evaluating the Impact of Position Encoding

In this section, we discuss the details of Position Encoding (PE). The PE is employed to capture the global structural properties of the graph. Following [27, 14], the dimension of PE is set to $d^{\#} = 16$, balancing structural expressiveness with computational efficiency. The PE is pre-computed via a random walk-based diffusion process and kept fixed during training, serving as a stable structural prior that aids generalization. To evaluate its effectiveness, we performed an ablation study by removing PE (as shown in Table 6). The results show consistent performance drops without PE, particularly on heterophilic datasets such as Texas and Cornell, confirming that PE provides valuable global cues beyond local feature aggregation.

Table 6: An ablation study on position encoding (PE).

Ablation	Cornell	Texas	Chameleon	Photo	CS
w/o PE	81.08±2.70	85.95±5.24	64.78±1.36	93.20±2.36	94.12±0.23
CoRep	82.70±4.55	88.65±3.97	65.64±1.39	93.84±1.89	94.39±0.31

Table 7: The time and space complexities of the six methods.

Methods	Time Complexity	Space Complexity
MVGRL	$\mathcal{O}(\mathcal{V} ^2 d^\dagger + \mathcal{V} d^{\dagger^2})$	$\mathcal{O}(\mathcal{V} ^2 + \mathcal{V} d^\dagger)$
BGRL	$\mathcal{O}(\mathcal{E} d^\dagger + \mathcal{V} d^{\dagger^2})$	$\mathcal{O}(\mathcal{E} + \mathcal{V} d^\dagger + \mathcal{V} d^{\dagger^2})$
HGRL	$\mathcal{O}(\mathcal{V} ^2 d^\dagger + d^{\dagger^2})$	$\mathcal{O}(\mathcal{V} ^2 + \mathcal{V} d^\dagger)$
GREET	$\mathcal{O}(\mathcal{E} d^\dagger + \mathcal{V} b \kappa d^\sharp + \mathcal{V} d d^\dagger)$	$\mathcal{O}(\mathcal{E} + \mathcal{V} d^\sharp)$
HeteGCL	$\mathcal{O}(\mathcal{V} ^2 d^\dagger + d^{\dagger^2})$	$\mathcal{O}(\mathcal{V} ^2 + \mathcal{V} d^\dagger)$
CoRep	$\mathcal{O}(\mathcal{E} d^\dagger + \mathcal{V} b \kappa d^\sharp + \mathcal{V} d d^\dagger)$	$\mathcal{O}(\mathcal{E} + \mathcal{V} d^\sharp)$

Table 8: The average elapsed time per training epoch of the methods (in seconds). OOM indicates Out-Of-Memory.

Methods	CiteSeer	PubMed	CS	Physics	Cornell	Texas
MVGRL	0.343	0.345	2.004	OOM	0.034	0.034
BGRL	0.033	0.182	0.202	0.568	0.022	0.022
HGRL	0.058	0.836	1.770	OOM	0.031	0.037
GREET	0.202	3.225	5.195	21.891	0.025	0.025
HeteGCL	0.154	0.851	1.627	OOM	0.061	0.061
CoRep	0.047	0.156	0.164	0.491	0.022	0.023

F.5 Efficiency Analysis

To provide a clearer comparison of computational efficiency, we report the theoretical time and space complexities of our method and baseline approaches during training in Table 7, where $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of nodes and edges in the graph. d , d^\dagger , and d^\sharp are the dimensionality of the original node features, final node representations, and projected node representations. κ represents the average number of positive samples. b is the batch size of the contrastive loss. In addition, the actual training times of all methods are presented in Table 8. The results show that our method achieves comparable running speed to BGRL, one of the most efficient graph contrastive learning methods, and even outperforms it on larger datasets such as PubMed, CS, and Physics. This improvement is primarily attributed to the use of sparse computation and mini-batch processing. Furthermore, compared to other baselines, our approach demonstrates significant efficiency advantages. For example, on the Physics dataset, our method is approximately 45 times faster than GREET. These results highlight the scalability and practical efficiency of our approach.

F.6 Scalability on Large-Scale Graphs

To further evaluate the scalability of our method, we conduct experiments on two large-scale datasets: the homophilic dataset *OGBN-Arxiv* and the heterophilic dataset *Arxiv-Year*. Both datasets are large in scale, containing 169,343 nodes and 1,166,243 edges. The experimental results are reported in Table 9. As shown in the results, our method ranks second on *OGBN-Arxiv*, slightly behind BGRL, and achieves the best performance on *Arxiv-Year*. These findings demonstrate that our approach maintains strong performance and scalability on large-scale graphs with both homophilic and heterophilic structures.

G Generality Across Homophilic and Heterophilic Graphs

Although our method is motivated by the challenges of heterophily, its design is inherently general and applicable to graphs with varying degrees of homophily. At the core is the edge-aware coloring matching framework, where the coloring classifier flexibly assigns colors to nodes according to

Table 9: Results on homophilic and heterophilic large-scale graphs. OOM indicates Out-Of-Memory.

Methods	OGBN-Arxiv	Arxiv-Year
node2vec	70.07 \pm 0.13	39.69 \pm 0.09
DGI	70.19 \pm 0.73	40.60 \pm 0.21
MVGRL	OOM	OOM
BGRL	71.24\pm0.35	41.43 \pm 0.04
HGRL	OOM	OOM
GREET	OOM	OOM
CoRep	70.59 \pm 0.07	42.59\pm0.42

their edge relations. For homophilic edges, the mechanism encourages similar color assignments to connected nodes, thereby preserving community structures and label smoothness. For heterophilic edges, it enforces dissimilar color assignments, explicitly capturing cross-class dependencies. The learnable edge evaluator complements this process by dynamically scoring node-pair relations, thus guiding the classifier in making more accurate color decisions. Through this synergy, our framework seamlessly adapts to different structural patterns without architectural modifications. Extensive experiments on both homophilic and heterophilic benchmarks (Tables 1 and 2) confirm this universality, as our method consistently outperforms strong baselines across diverse graph structures.

H Limitations and Broader Impacts

Limitations. The coloring learning framework CoRep offers a novel perspective for representation learning on heterophilic graphs, but it also introduces certain limitations. In particular, the number of colors is often determined based on empirical knowledge. A common practice is to set the number of colors based on the number of classes in the dataset, typically ensuring that the number of colors is no less than the number of classes, aiming to uncover potential semantic structures. In future work, we will explore methods to obtain the number of colors directly from graph structures.

Broader Impacts. As a preliminary exploration of coloring learning in unsupervised heterophilic graph learning, CoRep has led to the following impacts:

- Revealing the goal alignment between coloring learning and heterophilic graph learning: CoRep introduces a novel coloring learning framework to explore complex heterophilic structures, providing a new perspective for future research in self-supervised heterophilic graph learning.
- Broad applicability across multiple domains: CoRep has been experimentally tested on various benchmark datasets, demonstrating the diverse potential applications of our algorithm.
- Negative social impact: No negative social impacts are foreseen at present.