

Graph Data Condensation via Self-expressive Graph Structure Reconstruction

Zhanyu Liu*
Shanghai Jiao Tong University
Shanghai, China
zhyliu00@sjtu.edu.cn

Chaolv Zeng*
Shanghai Jiao Tong University
Shanghai, China
zclzcl@sjtu.edu.cn

Guanjie Zheng†
Shanghai Jiao Tong University
Shanghai, China
gjzheng@sjtu.edu.cn

Abstract

With the increasing demands of training graph neural networks (GNNs) on large-scale graphs, graph data condensation has emerged as a critical technique to relieve the storage and time costs during the training phase. It aims to condense the original large-scale graph to a much smaller synthetic graph while preserving the essential information necessary for efficiently training a downstream GNN. However, existing methods concentrate either on optimizing node features exclusively or endeavor to independently learn node features and the graph structure generator. They could not explicitly leverage the information of the original graph structure and failed to construct an interpretable graph structure for the synthetic dataset. To address these issues, we introduce a novel framework named Graph Data Condensation via Self-expressive Graph Structure Reconstruction (GCSR). Our method stands out by (1) explicitly incorporating the original graph structure into the condensing process and (2) capturing the nuanced interdependencies between the condensed nodes by reconstructing an interpretable self-expressive graph structure. Extensive experiments and comprehensive analysis validate the efficacy of the proposed method across diverse GNN models and datasets. Our code is available at <https://github.com/zclzcl0223/GCSR>.

CCS Concepts

• Information systems → Data mining.

Keywords

Graph Data Condensation; Graph Neural Network; Node Classification

ACM Reference Format:

Zhanyu Liu, Chaolv Zeng, and Guanjie Zheng. 2024. Graph Data Condensation via Self-expressive Graph Structure Reconstruction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, August 25–29, 2024, Barcelona, Spain. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3637528.3671710>

*Both authors contributed equally to this work.

†Corresponding Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD '24, August 25–29, 2024, Barcelona, Spain

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0490-1/24/08

<https://doi.org/10.1145/3637528.3671710>

1 Introduction

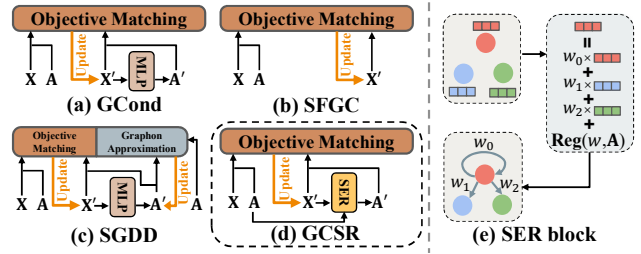


Figure 1: Learning pipelines of different graph condensation methods. X, A denotes the original full dataset and X', A' denotes condensed dataset. *SER* is our proposed self-expressive graph structure reconstruction module and *Reg* is the regularization term for reconstruction.

In recent years, the significant increase in the volume of graph data in different fields, such as social networks [29, 49], recommendation systems [11, 65], traffic networks [34, 40–42], and knowledge graphs [62, 74], has presented challenges for practitioners and researchers. Consequently, storing, processing, analyzing, and transmitting large-scale data has become a burden in real-world applications. Moreover, when considering deep learning tasks such as neural architecture search [39, 55] and continual learning [20, 33], directly utilizing the full dataset for training a graph neural network (GNN) has the potential to result in poor efficiency.

To address this issue, data-centric graph size reduction methods, such as graph coarsening [2, 25], graph sparsification [1, 4], coreset selection [33, 64], and graph sampling [5, 73], aim to reduce the redundancy of the graph data and output a small dataset. However, these methods rely on heuristics such as eigenvalues [1] and the accuracy loss [33], which could lead to unsatisfactory generalization between different network architectures and suboptimal performance on downstream tasks such as node classification [23, 71]. In recent studies [23, 24, 70, 71, 80], the concept of *Graph Data Condensation* has emerged, intending to train a compact synthetic dataset that exhibits comparable performance to the full dataset when utilized to train the same GNN model. This approach achieves good results by matching surrogate objectives between the synthetic dataset and the original dataset.

However, existing graph data condensation methods fail to effectively and efficiently preserve the valuable information embedded within the original graph structure and capture the inter-node correlations within the condensed dataset. Fig. 1 illustrates the learning pipeline of various graph data condensation methods focused on node classification including GCond [24], SGDD [71], SFGC [80],

and our proposed GCSR. GCond aims to model the graph structure by utilizing node features as input to a multilayer perceptron (MLP), while SFGC proposes a graph-free framework. These two methods cannot build explicit and interpretable inter-node correlations, and ignore the rich information contained in the original graph structure. SGDD employs the original graph structure to guide the generation of the synthetic graph structure through graphon approximation, but the application of bi-loop optimization may result in unstable performance and unsatisfactory efficiency.

To effectively and efficiently construct an explicit and interpretable graph structure for the synthetic condensed data, and integrate the information from the original graph structure, we propose a novel framework named **Graph Data Condensation via Self-expressive Graph Structure Reconstruction (GCSR)**. Overall, GCSR contains three novel modules. The first module is the Initialization Module, where we utilize the k -order node feature for node initialization and the probabilistic adjacency matrix derived from the original graph structure to initialize the regularization term used in the subsequent reconstruction process. The second module is the Self-expressive Reconstruction Module. Leveraging the self-expressiveness property of graph data [26, 68], which indicates that nodes within the same feature subspace can represent each other, we reconstruct an explicit and interpretable graph structure using a closed-form expression. The last module is the Update Module. In this module, the node feature is updated through multi-step gradient matching with the training trajectories of the full dataset. The regularization term of the reconstruction is updated through bootstrapping, ensuring adaptability to the training dynamics and preventing overfitting of the learned graph structure to the small condensed dataset. In general, our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to attempt to construct the explicit and interpretable graph structure for the synthetic condensed data in the graph condensation task. The graph structure is constructed based on the self-expressive nature of graph data and effectively integrates the extensive information present in the original full graph.
- We propose a novel framework for the graph condensation task via self-expressive graph structure reconstruction, abbreviated as GCSR. This framework encompasses three key modules, including initialization, self-expressive reconstruction, and update. These modules work in tandem to generate condensed data that effectively captures the essence of the original graph dataset.
- We conduct extensive experiments on five real-world graph datasets. The results demonstrate that our proposed framework achieves superior performance and has many characteristics such as maintaining the inter-class similarity of the original graph.

2 Related Work

Graph Neural Networks. Taking both node features and their structure information as the input, graph neural networks (GNNs) [7, 10, 16, 19, 28, 60, 66, 69, 72] have emerged as powerful tools in graph machine learning. The versatility of graph-structured data and the strong graph representation capabilities of GNNs have led to their

widespread adoption in a range of real-world applications, including recommender systems [13], natural language processing [67], and computer vision [50].

Dataset Condensation. Dataset condensation (DC), also known as dataset distillation [3, 37, 52, 57, 63, 77–79], aims to generate a smaller synthetic dataset from a large training dataset, such that it can effectively substitute the original dataset for downstream training tasks. To obtain such a synthetic dataset, various works employ different optimization objectives. Some methods [8, 37, 43, 51, 52, 63, 81] learn the synthetic dataset by using a meta-learning style framework to optimize the performance on the real full data. Some methods [3, 6, 9, 27, 31, 32, 75, 77, 79] optimize the synthetic dataset by matching the gradients of the neural networks trained on the synthetic dataset and the original dataset respectively. Some methods [30, 57, 61, 78] optimize the synthetic dataset by matching the distribution between the synthetic dataset and the original dataset. While initially applied to image data, DC has recently found wide application in graph-structured data for both node-level tasks [24, 71, 80] and graph-level tasks [23, 70]. GCond [24] first tries to condense graph data via gradient matching and generate the adjacency matrix from the synthetic node features using a trained multilayer perceptron. SGDD [71] enhances GCond by broadcasting the original graph structure to the generation of the synthetic graph structure through laplacian energy distribution matching. SFGC [80] proposes to generate structure-free graphs by matching training trajectories and selecting optimal synthetic graphs using graph neural tangent kernel (GNTK) [10]. However, these methods could not effectively and efficiently capture an explicit and interpretable graph structure for the condensed graph and neglect the explicit information of the original graph structure, which results in suboptimal performance.

Self-expressiveness. The self-expressiveness property indicates that each data sample can be represented by a linear combination of other data points [45, 48]. Such a property is first explored in subspace segmentation [44], which aims to segment data drawn from multiple linear subspaces as data in the same subspace could represent each other. Later, researchers begin to apply this property to graph learning [54, 68], aiming to learn a denoised and interpretable graph structure for downstream tasks such as graph clustering [26, 48]. They add different regularizations and constraints to ensure that the learned graph structure can fully reflect the connection relationships between nodes.

3 Preliminary

Graph Condensation. A graph dataset \mathcal{T} is denoted as $\mathcal{T} = \{\mathbf{A}, \mathbf{X}, \mathbf{Y}\}$, where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the node feature, and $\mathbf{Y} \in \mathbb{R}^{N \times 1}$ is the label. Generally, graph data condensation aims to learn a downsized synthetic graph $\mathcal{S} = \{\mathbf{A}', \mathbf{X}', \mathbf{Y}'\}$ with $\mathbf{A}' \in \mathbb{R}^{N' \times N'}$, $\mathbf{X}' \in \mathbb{R}^{N' \times d}$, and $\mathbf{Y}' \in \mathbb{R}^{N' \times 1}$ from the original training graph \mathcal{T} ($N' \ll N$). Both \mathbf{Y}_i and \mathbf{Y}'_j belong to the class set $\mathcal{C} = \{0, 1, \dots, C-1\}$. Formally, graph condensation is defined by solving the problem as follows:

$$\begin{aligned} & \min \mathcal{L}(\text{GNN}_{\theta_S}(\mathbf{A}, \mathbf{X}), \mathbf{Y}) \\ \text{s.t. } & \theta_S = \arg \min_{\theta} \mathcal{L}(\text{GNN}_{\theta}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}'). \end{aligned} \quad (1)$$

Here, GNN_θ represents the GNN model parameterized with θ , θ_S denotes the parameters of the model trained on the synthetic graph $S = \{\mathbf{A}', \mathbf{X}', \mathbf{Y}'\}$, and \mathcal{L} is the error of node classification. For the graph data condensation task, only the variables \mathbf{A}' and \mathbf{X}' are optimized, while the node labels \mathbf{Y}' are fixed, maintaining the same class distribution as the original labels \mathbf{Y} .

4 Method

In this section, we present our proposed framework GCSR, which consists of three essential modules. The diagram of the framework is depicted in Fig. 2, and the detailed pseudo code is shown in Alg. 1. The first module, termed the Initialization Module, initializes the node feature and graph regularizer for the downstream synthetic dataset. Then, the Self-expressive Reconstruction Module exploits the inherent self-expressive property of the graph data and employs a closed-form solution to derive an interpretable graph structure that captures the self-representation relationships among the nodes. Finally, the Update Module uses the derived adjacency matrix to update the graph regularizer in a bootstrapping style and the multi-step gradient matching loss to update the node feature respectively.

4.1 Initialization

4.1.1 Node Initialization.

Goal: The initial feature of the condensed dataset is crucial to the final performance. Previous methods utilize random noise [8, 61] or random samples from the original dataset [3, 24, 71, 78]. However, these methods overlook the valuable graph structure information during the node feature initialization stage. In fact, in the field of graph learning, it has been observed that node features that incorporate message passing exhibit higher expressiveness and yield better performance in downstream tasks, such as node classification, due to the fusion of graph structure information [66, 72]. Building upon this insight, we propose an approach to initialize the node feature after the message passing process. By doing so, we ensure that the framework is initialized with better node features, leading to enhanced performance in subsequent tasks.

Message Passing Initialization: Given the original graph dataset $\mathcal{T} = \{\mathbf{A}, \mathbf{X}, \mathbf{Y}\}$, we first normalize it following previous work [26, 28, 48]. The symmetrically normalized adjacency matrix can be defined as:

$$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}, \quad (2)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-connections added, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ is the degree matrix, and \mathbf{I} is the identity matrix. Under the framework of message passing mechanisms [17, 48], each node of \mathcal{T} aggregates the information from its k -order neighborhood:

$$\hat{\mathbf{X}} = \hat{\mathbf{A}}^k \mathbf{X}, \quad (3)$$

where $\hat{\mathbf{A}}^k$ indicates the k -order adjacency matrix.

After Eq. 2 and Eq. 3, we obtain a node representation $\hat{\mathbf{X}}$ that aggregates neighborhood features as well as graph structure information. We then initialize the synthetic nodes \mathbf{X}' by randomly sampling the node feature from $\hat{\mathbf{X}}$,

$$\mathbf{X}' = \text{Random_Sample}(\hat{\mathbf{X}}). \quad (4)$$

Moreover, to make the synthetic dataset conform to the distribution of the original dataset and avoid creating an unbalanced dataset, we keep the class distribution the same as the original graph. Formally, the following equation holds for every node class, i.e., $c \in \{0, 1, \dots, C-1\}$.

$$\sum_{i=1}^{N'} \frac{\mathbb{1}(\mathbf{Y}'_i == c)}{N'} = \sum_{k=1}^N \frac{\mathbb{1}(\mathbf{Y}_k == c)}{N}. \quad (5)$$

Here, \mathbf{Y}'_i and \mathbf{Y}_k are the class of node in the synthetic dataset and original dataset respectively, and $\mathbb{1}$ is the indicator function.

4.1.2 Regularizer Initialization.

Goal: The downstream self-expressive reconstruction task needs a graph structure regularizer to avoid the solution falling into a trivial solution [26, 68]. So, we aim to initialize a graph structure regularizer by incorporating the information of the original graph structure. However, considering that the dimensions of adjacency matrices of the synthetic graph and the original graph are inconsistent, it is difficult to directly transfer the structure information of the original graph to the synthetic graph. Consequently, we propose to generate a probabilistic adjacency matrix [15, 23] based on the information from the original adjacency matrix as the initial graph regularizer. To guarantee the stability of the training process, we replace the learnable parameter and use the frequency of edge types as the probability instead, which reflects the correlation between classes.

Graph Regularizer Initialization: Given the original adjacency matrix \mathbf{A}_{ij} , we aim to first calculate the frequency of edge types and construct a class-wise correlation matrix as follows:

$$\bar{\mathbf{A}}_{cc'} = \frac{\sum_{i,j=1}^N \mathbb{1}(\mathbf{Y}_i = c) \mathbb{1}(\mathbf{Y}_j = c') \mathbf{A}_{ij}}{\sum_{i,j=1}^N \mathbb{1}(\mathbf{Y}_i = c) \mathbf{A}_{ij}}, \quad (6)$$

where $\bar{\mathbf{A}} \in \mathbb{R}^{|C| \times |C|}$ represents the adjacency matrix generated from the original graph, \mathbf{Y}_i is the class of node i of the original dataset, and $\mathbb{1}$ is the indicator function. The value of $\bar{\mathbf{A}}_{cc'}$ implies the frequency that an edge from a node in class c to a node in class c' exists. With $\bar{\mathbf{A}}$, we can generate an adjacency matrix that leverages the structure information of the original graph:

$$\mathbf{P}_{ij} = \bar{\mathbf{A}}_{\mathbf{Y}'_i \mathbf{Y}'_j}, \quad (7)$$

where $\mathbf{P} \in \mathbb{R}^{N' \times N'}$ and \mathbf{Y}'_i denotes the label of synthetic node i . Consequently, \mathbf{P} captures the inter-class correlations and the correlations offer valuable information for the downstream self-expressive reconstruction. By utilizing \mathbf{P} as the graph structure regularization term, the framework could avoid generating trivial solutions in the optimization process.

4.2 Self-expressive Reconstruction

4.2.1 Reconstruction. In recent years, it has been witnessed the great success of graph structure learning [26, 48, 54] via self-expressive graph structure reconstruction. Such a strategy exploits the self-expressiveness property of graph nodes to reconstruct the adjacency matrix in structure-free graphs. Mathematically, it can be formulated as the following expression:

$$\min_{\mathbf{Z}} \|\mathbf{X}'^T - \mathbf{X}'^T \mathbf{Z}\|_F^2. \quad (8)$$

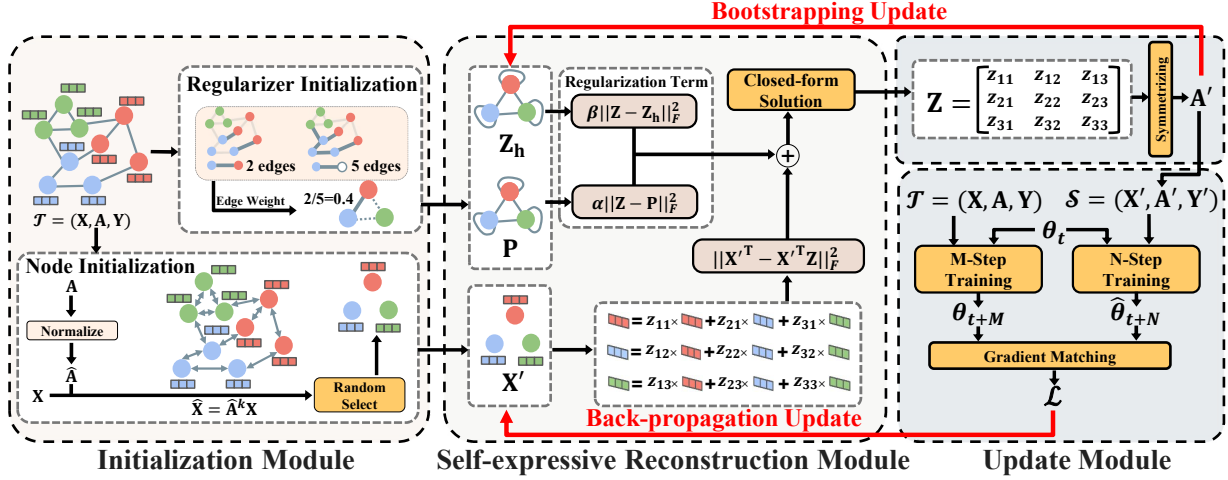


Figure 2: Overview of Graph Data Condensation via Self-expressive Graph Structure Reconstruction (GCSR).

Algorithm 1: GCSR for Graph Data Condensation

Input: $\mathcal{T} = \{A, X, Y\}$: training data
Input: Y' : pre-defined synthetic labels
Input: $\{\theta\}$: set of expert GNN parameters trained on \mathcal{T}
Require: Training epochs K , synthetic steps N , expert steps M , learning rate η_1, η_2 , regularization weight α, β , update rate τ, γ

- 1: Initialize X' according to Eq. 3 and Eq. 4
- 2: Initialize P according to Eq. 6 and Eq. 7
- 3: Initialize Z_h as identity matrix I
- 4: **for** $k = 1 \rightarrow K$ **do**
- 5: Sample expert GNN parameters: $\theta_t \sim P_\theta$;
- 6: Initialize synthetic GNN parameters: $\hat{\theta}_t = \theta_t$;
- 7: Compute A' according to Eq. 13 and Eq. 14;
- 8: **for** $i = 1 \rightarrow N$ **do**
- 9: Update synthetic GNN w.r.t. classification loss:

$$\hat{\theta}_{t+i} = \hat{\theta}_{t+i-1} - \eta_1 \nabla l(S, \hat{\theta}_{t+i-1});$$
- 10: **end for**
- 11: Compute loss between synthetic and expert params:

$$\mathcal{L} = \|\hat{\theta}_{t+N} - \theta_{t+M}\|_F^2 / \|\theta_{t+M} - \theta_t\|_F^2;$$
- 12: Update $X' \leftarrow X' - \eta_2 \nabla_{X'} \mathcal{L}(S, \mathcal{T})$;
- 13: Update $P \leftarrow \tau P + (1 - \tau) A'$;
- 14: Update $Z_h \leftarrow \gamma Z_h + (1 - \gamma) A'$;
- 15: **end for**
 // Computing the final adjacency matrix A'
- 16: $Z = (X' X'^T + \alpha I + \beta I)^{-1} (X' X'^T + \alpha P + \beta Z_h)$
- 17: $A' = (|Z| + |Z|^T) / 2$

Output: $S = \{A', X', Y'\}$

Here, $X' \in \mathbb{R}^{N' \times d}$ is the node features, and $Z \in \mathbb{R}^{N' \times N'}$ is the self-expressive matrix that measures the similarity between nodes with interpretability. However, solving such an equation could lead to trivial solutions such as identity matrix I . Consequently, existing methods [12, 26, 35, 36, 68] impose various regularizations

such as least-square and low rankness. For example, Least Square Regression (LSR) could be represented as:

$$\min_Z \|X'^T - X'^T Z\|_F^2 + \alpha \|Z\|_F^2, \quad (9)$$

where $\alpha > 0$ is a trade-off parameter and $\|\cdot\|_F$ is Frobenius norm. Nonetheless, the structure information of the original graph is only implicitly used when initializing X' via message passing. To explicitly incorporate the original graph structure and the synthetic graph structure information to Z , we add a regularization term and propose our self-expressive reconstruction model:

$$\min_Z \|X'^T - X'^T Z\|_F^2 + \alpha \|Z - P\|_F^2, \quad (10)$$

where P is the synthetic adjacency matrix generated from the original graph. By incorporating P into the regularization term, the similarity matrix could learn from the original graph structure. However, since the synthetic node features X' is updated in each epoch and X' has a significant impact on the optimizing object, the learned similarity matrix Z could experience drastic changes, resulting in unstable performance. To address this issue and mitigate the fluctuations, we propose to introduce the historical similarity matrix Z_h into the regularization term. By including Z_h , which captures the past information of the similarity matrix, we ensure a more stable learning process and enhance the overall performance of the model.

$$\min_Z \|X'^T - X'^T Z\|_F^2 + \alpha \|Z - P\|_F^2 + \beta \|Z - Z_h\|_F^2. \quad (11)$$

Here, Z_h is initialized as identity matrix I and is updated through the training process of the graph condensation task.

4.2.2 Closed-form Solution. Directly optimizing Eq. 11 is time-consuming. However, Eq. 11 can be easily solved by setting its first-order derivative w.r.t. Z to zero:

$$-2X'(X'^T - X'^T Z) + 2\alpha(Z - P) + 2\beta(Z - Z_h) = 0. \quad (12)$$

Eq. 12 could be reduced to:

$$Z = (X' X'^T + \alpha I + \beta I)^{-1} (X' X'^T + \alpha P + \beta Z_h). \quad (13)$$

Here, \mathbf{Z} not only takes the correlations between nodes into account (i.e., $\mathbf{X}'\mathbf{X}'^T$), but also retains the structure information of the original graph (i.e., \mathbf{P}). Furthermore, the learned similarity matrix \mathbf{Z} possesses interpretability as it explicitly indicates the weight to which the node feature of one node is represented by the other nodes in the synthetic graph dataset. This interpretability allows for a clear understanding of dependencies among the nodes of the synthetic graph dataset, leading to enhanced transparency in the proposed framework and a comprehensive comprehension of the graph data condensation task for the first time.

Considering the symmetry and non-negativity of the real-world adjacency matrix, we symmetrize \mathbf{Z} [26, 76] and calculate the final synthetic adjacency matrix as follows:

$$\mathbf{A}' = \frac{1}{2}(|\mathbf{Z}| + |\mathbf{Z}|^T), \quad (14)$$

where $|\cdot|$ is the elementwise absolute operation. $\mathbf{A}' \in \mathbb{R}^{N' \times N'}$ is the synthetic adjacency matrix.

4.2.3 Complexity Analysis. For Eq. 13, the time complexity of computing $\mathbf{X}'\mathbf{X}'^T$ is $O(N'^2d)$. The time complexity of computing $(\mathbf{X}'\mathbf{X}'^T + \alpha\mathbf{I} + \beta\mathbf{I})^{-1}$ is $O(N'^3)$ due to the matrix inverse operation. The time complexity of matrix multiplication between the two main terms is $O(N'^3)$. Consequently, the overall time complexity for calculating Eq. 13 is $O(N'^3 + N'^2d)$. In practice, since the matrix multiplication could be highly parallelized, the main time-consuming part of Eq. 13 is the matrix inverse operation with a complexity of $O(N'^3)$. It is worth noting that this time complexity is acceptable since the condensed dataset is small, with $N' \approx 100$.

4.3 Update

4.3.1 Node Feature Update. In order to update and refine the synthetic data \mathcal{S} to ensure its performance similarity to the original data \mathcal{T} when training downstream models, it is essential to enable the synthetic data \mathcal{S} to learn the training dynamics observed in models trained with the real data \mathcal{T} . This learning process facilitates the alignment of the synthetic data with the underlying patterns and characteristics present in the original data, thereby enhancing its suitability for downstream tasks such as node classification. Here, we adopt multi-step gradient matching [3, 80], which essentially matches the multi-step gradient of a model trained by the synthetic data \mathcal{S} and the real data \mathcal{T} . Formally, the matching objective could be formulated as follows:

$$\begin{aligned} D(\mathcal{S}, \mathcal{T}; \theta_t) &= \frac{\|(\widehat{\theta}_{t+N} - \theta_t) - (\theta_{t+M} - \theta_t)\|_F^2}{\|\theta_{t+M} - \theta_t\|_F^2} \\ &= \frac{\|\widehat{\theta}_{t+N} - \theta_{t+M}\|_F^2}{\|\theta_{t+M} - \theta_t\|_F^2}, \end{aligned} \quad (15)$$

where θ_t is the model parameter sampled from the training trajectory of \mathcal{T} . Based on θ_t , we optimized it with dataset \mathcal{S} for N epochs to generate the trained parameter $\widehat{\theta}_{t+N}$ and optimized it with dataset \mathcal{T} for M epochs to generate θ_{t+M} .

Subsequently, the matching loss is the expectation of the $D(\mathcal{S}, \mathcal{T}; \theta_t)$ w.r.t. θ_t , which could be formulated as follows:

$$\mathcal{L}(\mathcal{S}, \mathcal{T}) = \mathbb{E}_{\theta_t \sim P_\theta} [D(\mathcal{S}, \mathcal{T}; \theta_t)], \quad (16)$$

Table 1: Details of dataset statistics.

Datasets	#Nodes	#Edges	#Classes	#Features	Sparsity	Homophily
Citeseer	3,327	4,732	6	3,703	0.09%	0.74
Cora	2,708	5,429	7	1,433	0.15%	0.81
Ogbn-arxiv	169,343	1,166,243	40	128	0.01%	0.65
Flickr	89,250	899,756	7	500	0.02%	0.33
Reddit	232,965	57,307,946	41	602	0.09%	0.78

where P_θ is the distribution of parameter trajectories trained by real dataset \mathcal{T} . By optimizing $\mathcal{L}(\mathcal{S}, \mathcal{T})$, the synthetic graph dataset \mathcal{S} would be guided by the training trajectories of the original graph \mathcal{T} . Consequently, the model trained by \mathcal{S} could have similar parameters to the model trained by \mathcal{T} and the performance on downstream tasks is expected to be comparable.

4.3.2 Graph Regularizer Update. The graph regularizer \mathbf{P} plays a significant role in determining the resulting graph structure \mathbf{A}' . However, maintaining \mathbf{P} unchanged throughout the condensing process can lead to undesirable outcomes, such as the synthetic graph \mathbf{A}' overfitting to the specific characteristics of \mathbf{P} . Furthermore, real-world graphs are commonly subject to noise and incompleteness [15, 38], and these imperfections could be inherited by the graph regularizer \mathbf{P} . To address these issues, we update \mathbf{P} with bootstrapping algorithms [18, 38] as follows:

$$\mathbf{P} \leftarrow \tau\mathbf{P} + (1 - \tau)\mathbf{A}'. \quad (17)$$

Similarly, we update the historical similarity matrix \mathbf{Z}_h as follows:

$$\mathbf{Z}_h \leftarrow \gamma\mathbf{Z}_h + (1 - \gamma)\mathbf{A}', \quad (18)$$

where $\tau \in [0, 1]$ and $\gamma \in [0, 1]$. $\tau \geq 0.9$ provides a very slow update to preserve as much of the original structure information as possible. $\gamma \geq 0.5$ provides a faster update to ensure that the newest structure information can be retained and benefit the downstream processes.

5 Experiment

5.1 Experimental Settings

Datasets: In line with previous research [24, 71, 80], we evaluate our proposed framework on node classification task. We use three transductive datasets: Citeseer [28], Cora [28], and Ogbn-arxiv [21], as well as two inductive datasets: Flickr [73] and Reddit [19]. The details of the original dataset statistics are shown in Table 1, and we follow the public splits for each of them.

Settings: We condense each dataset to three different condensation ratios (r), which stands for the ratio of synthetic node number rN ($0 < r < 1$) to original node number N . In line with GCond [24], the condensation process involves two stages: (1) a learning stage, where a 2-layer SGC with 256 hidden units is used to generate synthetic graphs, (2) a test stage, where a 2-layer GCN with 256 hidden units is trained on the obtained synthetic graph from the first stage, and then the trained model is tested on the original test set. For each setting, we generate 5 synthetic graphs, test each synthetic graph 10 times, and report the average node classification accuracy with standard deviation.

Hyper-parameters: At the learning stage, We perform a grid search to select hyper-parameters on the following searching space: synthetic training steps N is tuned amongst $\{5, 20, 50, 100\}$; expert

Table 2: Overall node classification accuracy on the test split of datasets. For Citeseer, Cora, and Ogbn-arxiv, we report their transductive performance. For Flickr and Reddit, we report their inductive performance. Full indicates the performance with the original graph. The best results are highlighted in bold and grey, and the second-best results are underlined. Note that * indicates that we re-implement these methods to guarantee the comparison is fair since they have different GNN models for condensation in the original paper and thus the results are different.

Dataset	ratio (%)	Random	Herding	K-Center	Coarsening	DCG	GCond	SFGC*	SGDD*	MTT	GCSR	Full
Citeseer	0.9	54.4±4.4	57.1±1.5	52.4±2.8	52.2±0.4	66.8±1.5	<u>70.5±1.2</u>	66.3±2.4	71.5±0.9	66.1±3.0	70.2±1.1	71.7±0.4
	1.8	64.2±1.7	66.7±1.0	64.3±1.0	59.0±0.5	66.9±0.9	70.6±0.9	69.0±1.1	<u>71.2±0.7</u>	69.2±1.2	71.7±0.9	
	3.6	69.1±0.1	69.0±0.1	69.1±0.1	65.3±0.5	66.3±1.5	69.8±1.4	70.8±0.4	70.9±1.2	<u>71.0±0.6</u>	74.0±0.4	
Cora	1.3	63.6±3.7	67.0±1.3	64.0±2.3	31.2±0.2	67.3±1.9	<u>79.8±1.3</u>	77.7±1.8	79.1±1.3	78.4±1.4	79.9±0.7	81.4±0.6
	2.6	72.8±1.1	73.4±1.0	73.2±1.2	65.2±0.6	67.6±3.5	<u>80.1±0.6</u>	79.3±0.8	79.0±1.9	79.7±0.9	80.6±0.8	
	5.2	76.8±0.1	76.8±0.1	76.7±0.1	70.6±0.1	67.7±2.2	79.3±0.3	79.4±0.5	80.2±0.8	<u>80.5±0.6</u>	81.2±0.9	
Ogbn-arxiv	0.05	47.1±3.9	52.4±1.8	47.2±3.0	35.4±0.3	58.6±0.4	59.2±1.1	59.0±1.8	59.6±0.5	58.7±1.7	60.6±1.1	71.3±0.1
	0.25	57.3±1.1	58.6±1.2	56.8±0.8	43.5±0.2	59.9±0.3	63.2±0.3	<u>64.6±0.3</u>	61.7±0.3	64.2±0.5	65.4±0.8	
	0.5	60.0±0.9	60.4±0.8	60.3±0.4	50.4±0.1	59.5±0.3	64.0±0.4	<u>65.2±0.8</u>	58.7±0.6	65.1±0.7	65.9±0.6	
Flickr	0.1	41.8±2.0	42.5±1.8	42.0±0.7	41.9±0.2	46.3±0.2	<u>46.5±0.4</u>	45.5±0.8	46.1±0.3	45.4±0.4	46.6±0.3	47.1±0.1
	0.5	44.0±0.4	43.9±0.9	43.2±0.1	44.5±0.1	45.9±0.1	47.1±0.1	46.0±0.4	45.9±0.4	46.0±0.4	<u>46.6±0.2</u>	
	1	44.6±0.2	44.4±0.6	44.1±0.4	44.6±0.1	45.8±0.1	47.1±0.1	46.1±0.3	46.4±0.2	46.2±0.4	<u>46.8±0.2</u>	
Reddit	0.05	46.1±4.4	53.1±2.5	46.6±2.3	40.9±0.5	<u>88.2±0.2</u>	88.0±1.8	80.0±3.1	84.2±0.7	81.6±1.8	90.5±0.2	94.1±0.0
	0.1	58.0±2.2	62.7±1.0	53.0±3.3	42.8±0.8	89.5±0.1	<u>89.6±0.7</u>	84.6±1.6	80.6±0.4	85.2±1.3	91.2±0.2	
	0.2	66.3±1.9	71.0±1.6	58.5±2.1	47.4±0.9	<u>90.5±1.2</u>	90.1±0.5	87.9±1.2	84.1±0.3	87.7±1.1	92.2±0.1	

steps M is searched in $\{1, 2\}$; the learning rate of Adam optimizer for updating synthetic node features η_2 is selected from $\{0.00001, 0.0001, 0.001, 0.01\}$; the regularization coefficient α and β is chosen from the log space between 0.1 to 1000; the update rate τ is tuned from 0.9 to 1. The update rate γ is fixed to 0.5. The learning rate for updating synthetic networks η_1 is fixed to 0.01. At the test stage, we follow the setting of GCond [24]. We fix the learning rate to 0.01, the weight decay rate to 0.0005, and train networks for 600 epochs.

Baselines: We compare our framework with the following baselines: graph coreset methods (*Random*, *Herding* [64], and *K-Center* [14, 58]), graph coarsening method [22], graph-based variant of dataset condensation method (DCG [79], MTT [3]), and graph condensation methods (GCond [24], SFGC [80], and SGDD [71]).

5.2 Overall Performance

For dataset condensation baselines, all the synthetic graphs are generated by SGC. We reuse most of the results in [24] since the experiment setting is exactly the same. We re-implement SFGC and SGDD to guarantee a fair comparison since they use GCN to generate the condensed graph dataset. The comparison of all the baselines is shown in Table 2. From the table, we can observe that: (1) overall, GCSR achieves superior performance compared to the baselines. The condensed data is comparable to the original dataset for training a GNN for node classification. (2) Compared to baselines, GCSR exhibits an improvement in accuracy with an increase in the condensed graph size, which contradicts the observations reported in prior studies and adheres to the intuition that more data leads to better performance. This further validates that our method is effective and successfully solves the graph dataset condensation task. (3) Moreover, compared with baselines that solely update

Table 3: The accuracy (%) of different GNNs used for condensation and the condensed dataset tested with GCN. The experiment is conducted on Cora with a 2.6% ratio.

Methods	Models				
	APPNP	Cheby	GCN	SAGE	SGC
GCOND	73.5±2.4	76.8±2.1	70.6±3.7	77.0±0.7	80.1±0.6
SFGC	78.1±0.8	77.1±0.8	79.1±0.7	78.5±0.8	79.3±0.8
SGDD	74.6±1.6	75.9±0.2	73.3±0.2	74.3±1.8	79.0±1.9
GCSR	80.3±1.0	80.2±0.9	80.1±0.9	80.2±1.1	80.6±0.8

synthetic node features such as DCG, SFGC, and MTT, our approach outperforms them by a large margin. We attribute the improved performance to the effectively generated adjacency matrix that captures both the self-expressiveness property of node features and the information of the original graph structure.

5.3 Cross Architecture Performance

Consistent with state-of-the-art benchmarks [24, 71, 80], we evaluate the generalizability of the graph condensation framework.

Different GNN for Condensation: Here, We choose APPNP [16], Cheby [7], GCN [28], GraphSAGE [19], and SGC [66] to serve as the models used in the condensation phase and test the condensed dataset with GCN. The experiment is conducted on Cora with a ratio of 2.6%. The performance is reported in Table 3. The results reveal that: (1) our method is model-insensitive and performs smoothly across different GNN architectures for condensation. (2) Among all these architectures, SGC is the simplest one but surpasses others in all methods. This indicates that a more complex GNN model in condensation does not necessarily bring better optimization results since it complicates the parameter matching process.

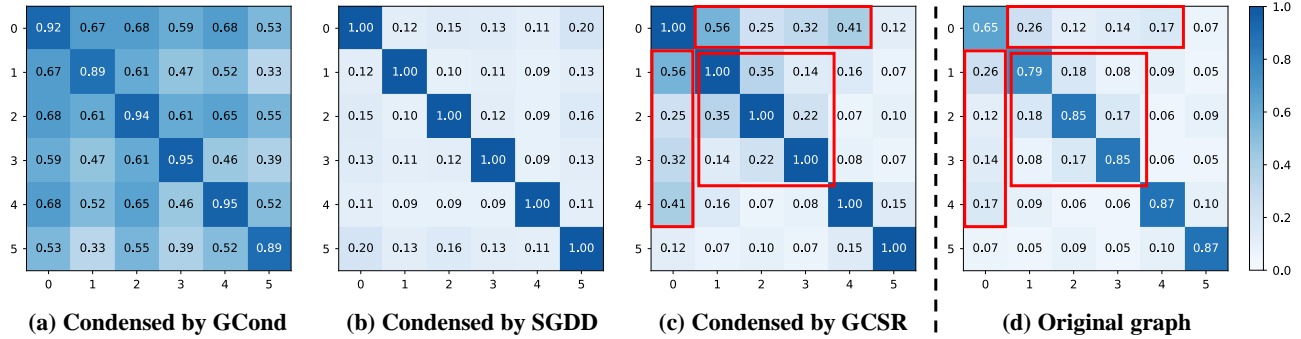


Figure 3: Cross-Class Neighborhood Similarity (CCNS) of Citeseer generated from the synthetic graph with a 3.6% condensation ratio from (a) GCond, (b) SGDD, and (c) GCSR, as well as (d) the original graph. The axes represent the classes.

Table 4: The performance of SGC used for condensation and the condensed dataset tested with different GNNs. The average accuracy (%) is reported. Avg. stands for the average accuracy of models except for MLP.

Dataset	Methods	Models						Avg.
		MLP	APPNP	Cheby	GCN	SAGE	SGC	
Citeseer r=1.8%	GCOND	58.3	69.6	68.3	70.5	66.2	70.3	69.0
	SFGC	62.4	69.1	68.2	69.0	69.0	69.3	68.9
	SGDD	64.9	71.2	70.0	71.2	71.4	71.3	71.0
	GCSR	65.7	71.6	70.9	71.7	71.6	71.8	71.5
Cora r=2.6%	GCOND	61.6	78.5	76.0	80.1	78.2	79.3	78.4
	SFGC	65.4	79.2	76.4	79.3	79.4	79.3	78.7
	SGDD	65.5	77.6	77.8	79.0	79.2	77.9	78.3
	GCSR	70.1	80.6	78.3	80.6	80.6	80.8	80.2
Ogbn-arxiv r=0.25%	GCOND	45.8	63.4	54.9	63.2	62.6	63.7	61.6
	SFGC	45.3	63.9	59.2	64.6	64.7	64.7	63.4
	SGDD	41.8	59.8	51.4	61.7	61.1	61.1	59.0
	GCSR	44.7	64.4	58.9	65.4	65.4	65.6	63.9
Flickr r=0.5%	GCOND	44.8	45.9	42.8	47.1	46.2	46.1	45.6
	SFGC	44.0	46.1	43.6	46.0	46.0	46.1	45.6
	SGDD	43.0	45.3	41.7	45.9	45.8	45.7	44.9
	GCSR	45.2	46.3	44.9	46.6	46.6	46.3	46.1
Reddit r=0.1%	GCOND	42.5	87.8	75.5	89.4	89.1	89.6	86.3
	SFGC	46.3	84.3	68.4	84.6	84.6	86.6	81.7
	SGDD	41.6	80.7	69.7	80.6	81.1	67.8	76.0
	GCSR	49.5	88.9	80.4	91.2	91.0	91.0	88.5

Different GNN for Test: We optimize synthetic graphs with SGC and test their performance under different neural architectures, i.e., MLP, APPNP, Cheby, GCN, GraphSAGE, and SGC. As shown in Table 4, our method obtains the best performance under most neural architectures and shows the best average performance. It has been well studied in [47, 53, 82] that different GNN models show similar low-pass filtering behaviors and the graph topology structure is a low-pass filter that provides a means to denoise the data. Our method fully retains the original structure information which contributes to its excellent transferability.

5.4 Learned Graph Structure Visualization

5.4.1 Similarity of Synthetic Structure and Original Structure. As shown in Fig. 3, we report the cross-class neighborhood similarity (CCNS) [46] of Citeseer generated from the synthetic graph of GCond, SGDD, and GCSR, as well as the original graph. We do not involve SFGC as the synthetic graph learned by it is structure-free. The value of CCNS $s(c, c')$ measures the neighborhood patterns similarity of nodes in class c and nodes in class c' from a neighborhood label distribution perspective. It is a good metric that reflects the connection relationships between nodes in a graph. From Fig. 3, we can find that: (1) graphs learned from GCond exhibit inter-class similarity with no clear distinction because their structure is directly generated from synthetic node features via MLP. (2) SGDD tends to emphasize the intra-class similarity while ignoring the inter-class similarity. Likewise, it also exhibits no clear distinction in inter-class similarity. (3) Owing to the ability to leverage the original graph structure and the inter-node correlations within the synthetic graph, our results mimic the node connection characteristics of the original graph excellently, which leads to better performance ultimately.

5.4.2 Interpretability of Self-expressive Reconstruction. We visualize the learned synthetic adjacency matrix of Citeseer as well as the components used to generate it (i.e., $\mathbf{X}'\mathbf{X}'^T$ and \mathbf{P}). From Fig. 4, we can observe that the derived synthetic adjacency matrix tends to maintain the joint connection properties of $\mathbf{X}'\mathbf{X}'^T$ and \mathbf{P} . Connections between nodes of the same class tend to be retained, while connections between nodes of different classes tend to be eliminated. Such a phenomenon aligns with the connection characteristics of the homophilic graph like Citeseer, where nodes are prone to connect with similar others.

5.5 Learned Node Feature Visualization

To evaluate whether our method can capture more information from the original graph, we use t-SNE [59] to visualize the real and synthetic node features of Cora condensed by GCond, SFGC, SGDD, and GCSR. For a clearer display, we only show nodes from 3 out of 7 classes. We also calculate the average silhouette coefficient [56] (a metric that measures the distance between samples of different classes and a higher coefficient is better) of synthetic data generated from each method. Fig. 5 shows that: (1) data learned by GCond,

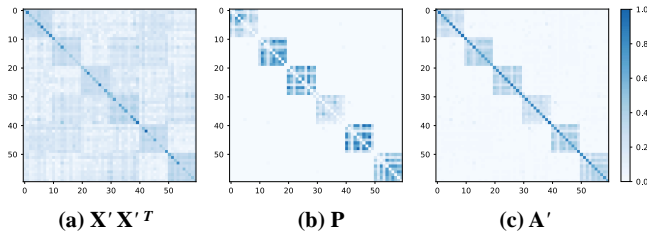


Figure 4: Visualizations of (a) the inner product of the synthetic nodes, (b) the updated probabilistic graph, and (c) the synthetic topology structure. The dataset illustrated is Cite-seer with the condensation ratio set to 1.8%.

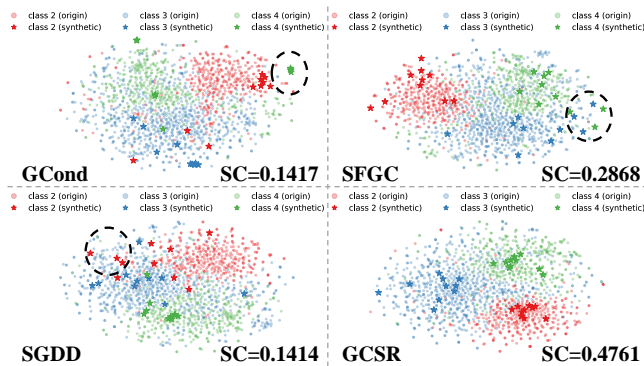


Figure 5: Distributions of the synthetic nodes condensed by four methods (GCond, SFGC, SGDD, and GCSR) on Cora under a 2.6% condensation ratio. SC represents the average silhouette coefficient of the synthetic data.

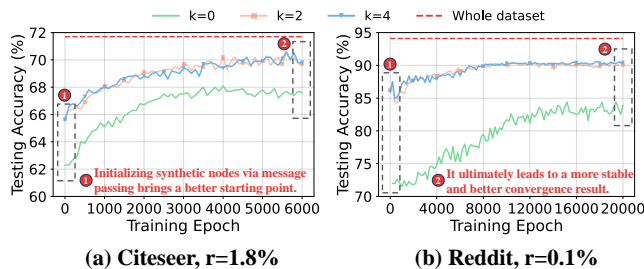


Figure 6: Training curve of different initialization methods. k stands for the hop of neighborhood being captured. $k = 0$ will have no message passing and have the same performance as a random selection from original nodes.

SFGC, and SGDD is not well conformed to the distribution of the original data, and some nodes are mixed up with nodes of other classes. (2) Data learned by GCSR obeys the distribution of the original data and exhibits a distinct separation between different classes. This could lead to better classification performance and evident distribution characteristics. (3) GCSR achieves the highest silhouette coefficient, indicating the learned node features of different classes are easy to separate.

Table 5: Evaluation of the effectiveness of each module. MPI denotes message passing initialization for node features and SER represents self-expressive reconstruction. The condensation model is SGC. The performance is shown in test accuracy (%) of GCN.

MPI	SER	Citeseer r=1.8%	Cora r=2.6%	Ogbn-arxiv r=0.25%	Flickr r=0.5%	Reddit r=0.1%
-	-	69.2±1.2	79.7±0.9	64.2±0.5	46.0±0.4	85.2±1.3
✓	-	70.1±0.8	80.4±0.8	64.2±1.1	46.4±0.4	90.4±0.7
✓	✓	71.7±0.9	80.6±0.8	65.4±0.8	46.6±0.2	91.2±0.2

5.6 Node Initialization

To evaluate the proposed message passing initialization for node features, we conduct extra experiments with one transductive dataset (Citeseer) and one inductive dataset (Reddit) under different initialization methods. To remove interference from other modules, we do not reconstruct the self-expressive adjacency matrix, i.e., we use the identity matrix as the graph structure in the synthetic dataset. Fig. 6 illustrates the node classification training curve of different initialization. We can make the following observations: (1) Initializing synthetic nodes via message passing brings a better starting point. We attribute such improvement to the fusion of the original graph structure and node features that enhance the synthetic data and the low-pass filtering properties of message passing that denoise the data. (2) Message passing initialization ultimately leads to a more stable and better convergence result. Generally, the end performance of Citeseer is about 1.3% higher and that of Reddit is about 6.1% higher. Such improvements are marvelous since there is no extra cost. (3) The results are not significantly influenced by the number of hops of the neighborhood. Empirically, setting k to 2 or 3 is a more robust and general choice.

5.7 Ablation Study

In GCSR, two components help broadcast the original graph structure to the synthetic graph, i.e., message passing initialization and self-expressive reconstruction. We conduct an ablation study to analyze the contribution of each component. As seen in Table 5, the joint use of two components brings about state-of-the-art performance. The results support our claim that both components contribute to leveraging information from the original graph. Message passing initialization transfers the implicit information of the original graph structure to the synthetic node features. Self-expressive reconstruction generates the interpretable graph structure that combines the information of the correlation between the node embeddings and the global structure information from the original graph. It is worth mentioning that, as a novel approach to initialize synthetic nodes, message passing initialization is independent of other modules and can significantly improve the condensation results even without adjacency matrices.

6 Conclusion

In this paper, we present a novel framework GCSR which learns synthetic graphs via self-expressive graph structure reconstruction.

It fills the gap in existing methods by making full use of the original graph structure and the correlations between node features to generate an interpretable synthetic graph structure. Extensive experimental results demonstrate the superiority of our proposed method. In the future, we plan to explore a more comprehensive reconstruction strategy that broadcasts more information from the original graph to the synthetic graph (e.g., sparsity and heterogeneity) to enhance the generalizability and robustness of synthetic graphs.

Acknowledgement

This work was sponsored by National Key Research and Development Program of China under Grant No.2022YFB3904204, National Natural Science Foundation of China under Grant No. 62102246, 62272301, and Provincial Key Research and Development Program of Zhejiang under Grant No. 2021C01034. Part of the work was done when the students were doing internships at Yunqi Academy of Engineering.

References

- [1] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. 2013. Spectral sparsification of graphs: theory and algorithms. *Commun. ACM* 56, 8 (2013), 87–94.
- [2] Chen Cai, Dingkang Wang, and Yusu Wang. 2021. Graph coarsening with neural networks. *arXiv preprint arXiv:2102.01350* (2021).
- [3] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. 2022. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4750–4759.
- [4] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. 2021. A unified lottery ticket hypothesis for graph neural networks. In *International conference on machine learning*. PMLR, 1695–1706.
- [5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. 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*. 257–266.
- [6] Justin Cui, Ruochoen Wang, Si Si, and Cho-Jui Hsieh. 2022. Scaling up dataset distillation to imagenet-1k with constant memory. *arXiv preprint arXiv:2211.10586* (2022).
- [7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).
- [8] Zhiwei Deng and Olga Russakovsky. 2022. Remember the past: Distilling datasets into addressable memories for neural networks. *Advances in Neural Information Processing Systems* 35 (2022), 34391–34404.
- [9] Jiawei Du, Yidi Jiang, Vincent YF Tan, Joey Tianyi Zhou, and Haizhou Li. 2023. Minimizing the accumulated trajectory error to improve dataset distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3749–3758.
- [10] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. 2019. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems* 32 (2019).
- [11] Magdalini Eirinaki, Jerry Gao, Iraklis Varlamis, and Konstantinos Tserpes. 2018. Recommender systems for large-scale social networks: A review of challenges and solutions. , 413–418 pages.
- [12] Ehsan Elhamifar and René Vidal. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence* 35, 11 (2013), 2765–2781.
- [13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.
- [14] Reza Zanjirani Farahani and Masoud Hekmatfar. 2009. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media.
- [15] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *International conference on machine learning*. PMLR, 1972–1982.
- [16] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).
- [17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.
- [18] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems* 33 (2020), 21271–21284.
- [19] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
- [20] Yi Han, Shanika Karunasekera, and Christopher Leckie. 2020. Graph neural networks with continual learning for fake news detection from social media. *arXiv preprint arXiv:2007.03316* (2020).
- [21] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.
- [22] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. 2021. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 675–684.
- [23] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. 2022. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 720–730.
- [24] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580* (2021).
- [25] Yu Jin, Andreas Loukas, and Joseph Jaja. 2020. Graph coarsening with preserved spectral properties. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4452–4462.
- [26] Zhao Kang, Zhanyu Liu, Shirui Pan, and Ling Tian. 2022. Fine-grained attributed graph clustering. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 370–378.
- [27] Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoon Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. 2022. Dataset condensation via efficient synthetic-data parameterization. In *International Conference on Machine Learning*. PMLR, 11102–11118.
- [28] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [29] Rabindra Lamsal. 2021. Design and analysis of a large-scale COVID-19 tweets dataset. *applied intelligence* 51 (2021), 2790–2804.
- [30] Hae Beom Lee, Dong Bok Lee, and Sung Ju Hwang. 2022. Dataset condensation with latent space knowledge factorization and sharing. *arXiv preprint arXiv:2208.10494* (2022).
- [31] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoon Yun, and Sungroh Yoon. 2022. Dataset condensation with contrastive signals. In *International Conference on Machine Learning*. PMLR, 12352–12364.
- [32] Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. 2022. Dataset distillation using parameter pruning. *arXiv preprint arXiv:2209.14609* (2022).
- [33] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.
- [34] Chumeng Liang, Zherui Huang, Yicheng Liu, Zhanyu Liu, Guanjie Zheng, Hanyuan Shi, Kan Wu, Yuhao Du, Fuliang Li, and Zhenhui Jessie Li. 2023. CBLat: Supporting the Training of Large-scale Traffic Control Policies with Scalable Traffic Simulation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4449–4460.
- [35] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. 2012. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2012), 171–184.
- [36] Guangcan Liu, Zhouchen Lin, and Yong Yu. 2010. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 663–670.
- [37] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. 2022. Dataset distillation via factorization. *Advances in Neural Information Processing Systems* 35 (2022), 1100–1113.
- [38] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. 2022. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*. 1392–1403.
- [39] Zhanyu Liu, Ke Hao, Guanjie Zheng, and Yanwei Yu. 2024. Dataset Condensation for Time Series Classification via Dual Domain Matching. *arXiv preprint arXiv:2403.07245* (2024).
- [40] Zhanyu Liu, Chumeng Liang, Guanjie Zheng, and Hua Wei. 2023. FDTI: Fine-grained Deep Traffic Inference with Roadnet-enriched Graph. *arXiv preprint arXiv:2306.10945* (2023).
- [41] Zhanyu Liu, Guanjie Zheng, and Yanwei Yu. 2023. Cross-city Few-Shot Traffic Forecasting via Traffic Pattern Bank. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1451–1460.

- [42] Zhanyu Liu, Guanjie Zheng, and Yanwei Yu. 2024. Multi-scale Traffic Pattern Bank for Cross-city Few-shot Traffic Forecasting. *arXiv preprint arXiv:2402.00397* (2024).
- [43] Noel Luo, Ramin Hasani, Mathias Lechner, and Daniela Rus. 2023. Dataset Distillation with Convexified Implicit Gradients. *arXiv preprint arXiv:2302.06755* (2023).
- [44] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. 2012. Robust and efficient subspace segmentation via least squares regression. In *Computer Vision—ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012, Proceedings, Part VII 12*. Springer, 347–360.
- [45] Juncheng Lv, Zhao Kang, Xiao Lu, and Zenglin Xu. 2021. Pseudo-supervised deep subspace clustering. *IEEE Transactions on Image Processing* 30 (2021), 5252–5263.
- [46] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2021. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134* (2021).
- [47] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1202–1211.
- [48] Zhengrui Ma, Zhao Kang, Guangchun Luo, Ling Tian, and Wenyu Chen. 2020. Towards clustering-friendly representations: Subspace clustering via graph filtering. In *Proceedings of the 28th ACM international conference on multimedia*. 3081–3089.
- [49] Usman Naseem, Imran Razzak, Matloob Khushi, Peter W Eklund, and Jinman Kim. 2021. COVIDSenti: A large-scale benchmark Twitter data set for COVID-19 sentiment analysis. *IEEE transactions on computational social systems* 8, 4 (2021), 1003–1015.
- [50] Usman Nazir, He Wang, and Murtaza Taj. 2021. Survey of image based graph neural networks. *arXiv preprint arXiv:2106.06307* (2021).
- [51] Timothy Nguyen, Zhouong Chen, and Jaehoon Lee. 2020. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050* (2020).
- [52] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. 2021. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems* 34 (2021), 5186–5198.
- [53] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).
- [54] Erlin Pan and Zhao Kang. 2021. Multi-view contrastive graph clustering. *Advances in neural information processing systems* 34 (2021), 2148–2159.
- [55] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2021. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–34.
- [56] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.
- [57] Ahmad Sajedi, Samir Khaki, Ehsan Amjadian, Lucy Z Liu, Yuri A Lawryshyn, and Konstantinos N Plataniotis. 2023. Datadam: Efficient dataset distillation with attention matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17097–17107.
- [58] Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489* (2017).
- [59] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
- [60] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.
- [61] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. 2022. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12196–12205.
- [62] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. 2018. Acekg: A large-scale knowledge graph for academic data mining. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1487–1490.
- [63] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. 2018. Dataset distillation. *arXiv preprint arXiv:1811.10959* (2018).
- [64] Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1121–1128.
- [65] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3597–3606.
- [66] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.
- [67] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. 2023. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning* 16, 2 (2023), 119–328.
- [68] Jun Xu, Mengyang Yu, Ling Shao, Wangmeng Zuo, Deyu Meng, Lei Zhang, and David Zhang. 2019. Scaled simplex representation for subspace clustering. *IEEE Transactions on Cybernetics* 51, 3 (2019), 1493–1505.
- [69] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
- [70] Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Kernel Ridge Regression-Based Graph Dataset Distillation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2850–2861.
- [71] Beining Yang, Kai Wang, Qingyun Sun, Cheng Ji, Xingcheng Fu, Hao Tang, Yang You, and Jianxin Li. 2023. Does Graph Distillation See Like Vision Dataset Counterpart? *arXiv preprint arXiv:2310.09192* (2023).
- [72] Chenxiao Yang, Qitian Wu, Jiahua Wang, and Junchi Yan. 2022. Graph neural networks are inherently good generalizers: Insights by bridging gnns and mlps. *arXiv preprint arXiv:2212.09034* (2022).
- [73] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019).
- [74] Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. 2020. ASER: A large-scale eventuality knowledge graph. In *Proceedings of the web conference 2020*. 201–211.
- [75] Lei Zhang, Jie Zhang, Bowen Lei, Subhabrata Mukherjee, Xiang Pan, Bo Zhao, Caiwen Ding, Yao Li, and Dongkuan Xu. 2023. Accelerating dataset distillation via model augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11950–11959.
- [76] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. 2019. Attributed graph clustering via adaptive graph convolution. *arXiv preprint arXiv:1906.01210* (2019).
- [77] Bo Zhao and Hakan Bilen. 2021. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*. PMLR, 12674–12685.
- [78] Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 6514–6523.
- [79] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929* (2020).
- [80] Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. 2023. Structure-free Graph Condensation: From Large-scale Graphs to Condensed Graph-free Data. *arXiv preprint arXiv:2306.02664* (2023).
- [81] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. 2022. Dataset distillation using neural feature regression. *Advances in Neural Information Processing Systems* 35 (2022), 9813–9827.
- [82] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*. 1215–1226.

Table 6: The effectiveness of MPI.

Dataset	GCSR	SFGC	SFGC+MPI	GCond	GCond+MPI
Citeseer (r=1.8%)	74.0 ± 0.4	70.8 ± 0.4	71.8 ± 0.4 ↑	69.8 ± 1.4	71.3 ± 0.9 ↑
Reddit (r=0.1%)	91.2 ± 0.2	84.6 ± 1.6	90.5 ± 0.2 ↑	89.6 ± 0.7	90.2 ± 0.2 ↑

Table 7: The sensitivity analysis of k in MPI.

Dataset	0	2	4	6	8	10
Citeseer(r=1.8%)	69.2 ± 1.2	70.2 ± 0.8	70.2 ± 0.8	70.3 ± 0.7	70.4 ± 0.7	70.3 ± 0.7

Table 8: The effectiveness of graph regularizer initialization of GCSR.

Method	Citeseer (r=1.8%)	Reddit (r=0.1%)
GCSR	71.8 ± 0.9	91.2 ± 0.2
Sub-structure Init	69.5 ± 2.0	91.0 ± 0.3
Random Init	64.8 ± 2.6	72.2 ± 2.5

Table 9: The sensitive analysis of regularizer update parameter τ .

Dataset	0.5	0.9	0.99	0.999	1
Citeseer(r=1.8%)	70.2 ± 1.0	71.8 ± 0.9	70.2 ± 1.2	68.7 ± 1.7	68.1 ± 1.8

Table 10: The sensitive analysis of regularizer update parameter γ .

Dataset	0.1	0.5	0.9	1
Citeseer(r=1.8%)	71.6 ± 0.9	71.8 ± 0.9	71.6 ± 0.9	70.6 ± 0.8

Table 11: The sensitivity analysis of α .

Dataset	0	0.1	0.5	1	10
Citeseer(r=1.8%)	70.1 ± 0.8	70.2 ± 1.0	71.1 ± 0.8	71.8 ± 0.9	70.1 ± 1.0
Ogbn-arxiv (r=0.05%)	59.9 ± 1.1	60.6 ± 1.1	59.2 ± 1.3	58.5 ± 1.8	58.8 ± 0.7

Table 12: The sensitivity analysis of β .

Dataset	0	0.1	0.5	1	10
Citeseer(r=1.8%)	71.6 ± 0.9	71.8 ± 0.9	71.6 ± 1.0	71.6 ± 0.9	71.3 ± 1.0
Ogbn-arxiv (r=0.05%)	59.8 ± 1.3	60.3 ± 1.3	60.2 ± 1.3	60.6 ± 1.1	60.5 ± 1.0

A The Effectiveness of Message Passing Initialization

GCSR introduces message-passing initialization (MPI), which is a novel approach in the field of graph condensation. To further verify the effectiveness of MPI, we implement it on two other graph condensation frameworks and check whether MPI brings performance enhancement. The result is shown in Table 6. We could observe that with MPI, the performance of SFGC and GCond both improve significantly, which validates the effectiveness of our proposed MPI module. Nevertheless, our method still performs better, which illustrates the necessity of SER.

Furthermore, we evaluate the sensitivity of parameter k in the MPI module. We solely implement MPI on GCSR and get the result of different k on Table 7. Consistent with our paper, MPI could significantly improve condensation performance. However, the results are not significantly influenced by k . As illustrated in the table above, even if we set k to 8 or 10, the improvement compared to $k=2$ is very small.

B Graph Regularizer Update Analysis

We initialize the graph regularizer with a probabilistic adjacency matrix to avoid the inconsistency between the dimensions of adjacency matrices of the synthetic graph and the original graph. To further verify the effectiveness of our initialization, we compare our initialization with sub-structure initialization of DosCond [23] and random initialization. The result is shown in Table 8. We can observe that the probabilistic adjacency matrix initialization achieves the best performance.

Furthermore, update momentum ratio τ and γ play important roles in updating the graph regularizer in Eqs. 17 and 18. We evaluate their impact by setting different τ and γ as shown in Table 9 and Table 10. Here $\tau = 1$ means not update \mathbf{P} and $\gamma = 1$ means not update \mathbf{Z}_h . We could observe that these two regularizer updates contribute to the final performance. Moreover, the performance is more sensitive to τ , so it is recommended to search for more τ in the real application.

C Graph Reconstruction Analysis

In this section, we analyze the impact of hyper-parameter α and β in reconstructing the graph structure \mathbf{Z} in Eq. 13 on Citeseer and Ogbn-arxiv as shown in Table 11 and Table 12. Here, $\alpha = 0$ means we only apply regularizer \mathbf{Z}_h and $\beta = 0$ indicates we only apply regularizer \mathbf{P} . We could observe that adding each regularizer could enhance the performance and the joint use of two regularizers brings the best performance. Moreover, the hyperparameter is recommended to search in $(0,1]$.