Rethinking and Accelerating Graph Condensation: A Training-Free Approach with Class Partition

Anonymous Author(s)

Abstract

The increasing prevalence of large-scale graphs poses a significant challenge for graph neural network training, attributed to their substantial computational requirements. In response, graph condensation (GC) emerges as a promising data-centric solution aiming to substitute the large graph with a small yet informative condensed graph to facilitate data-efficient GNN training. However, existing GC methods suffer from intricate optimization processes, necessitating excessive computing resources and training time. In this paper, we revisit existing GC optimization strategies and identify two pervasive issues therein: (1) various GC optimization strategies converge to coarse-grained class-level node feature matching between the original and condensed graphs; (2) existing GC methods rely on a Siamese graph network architecture that requires time-consuming bi-level optimization with iterative gradient computations. To overcome these issues, we propose a training-free GC framework termed Class-partitioned Graph Condensation (CGC), which refines the node distribution matching from the class-to-class paradigm into a novel class-to-node paradigm, transforming the GC optimization into a class partition problem which can be efficiently solved by any clustering methods. Moreover, CGC incorporates a pre-defined graph structure to enable a closed-form solution for condensed node features, eliminating the need for back-and-forth gradient descent in existing GC approaches. Extensive experiments demonstrate that CGC achieves an exceedingly efficient condensation process with advanced accuracy. Compared with the state-of-the-art GC methods, CGC condenses the Ogbn-products graph within 30 seconds, achieving a speedup ranging from $10^2\times$ to $10^4\times$ and increasing accuracy by up to 4.2%.

1 Introduction

Graph neural networks (GNNs) [7, 47, 51, 67] have garnered significant attention for their exceptional representation capabilities for complex graph data and have been utilized in a wide range of real-world applications, including chemical molecules [42], social networks [43], and recommender systems [57]. However, the increasing prevalence of large-scale graphs within these real-world applications poses formidable challenges in training GNN models. Most GNNs follow the message-passing paradigm [17], which is formulated as convolutions over the entire graph and aggregating information from multi-hop neighboring nodes. This process leads to exponential growth in neighbor size [18, 59] when applied to large-scale graphs, necessitating considerable training computations. In response to the urgent demand for processing large-scale graphs, a few studies borrow the idea of dataset distillation [28, 46] from computer vision (CV) and introduce graph condensation (GC) [24] to generate a compact yet informative graph to accelerate the GNN training. By capturing essential characteristics of the original large graph, GNNs trained on these small condensed graphs can achieve comparable performance to those trained on the original graphs. This efficacy enables GC to be applied to a variety of applications rapidly, e.g., graph continual learning [32], inference acceleration [14], and hyper-parameter search [8].

Despite the effectiveness of expediting GNN training, existing GC practices still suffer from complex optimization and intensive computation during the condensation process. As depicted in Figure 1 (a), to bridge the original and condensed graph, existing GC methods employ the Siamese network architecture to encode both graphs through a relay model, and the condensed graph is optimized to simulate the class distributions of the original graph. This framework necessitates a bi-level optimization procedure, with the inner loop refining the relay model on the condensed graph and the outer loop subsequently optimizing the condensed graph, ensuring optimal model performance on both graphs. However, two main issues persist in this framework: (1) Existing optimization strategies in GC [24, 30, 31, 45, 53, 66] manifest a single, unified optimization objective for all condensed nodes within the same class, resulting in the coarse-grained optimization target for condensed nodes. (2) Bi-level optimization involves iterative and intensive gradient computations for both the relay model and condensed graph. Recent efforts aim to expedite this process by simplifying outer or inner loop optimizations. Techniques like distribution matching [30] and the structure-free approach [66] respectively eliminate model gradient calculations and adjacency matrix optimizations in the outer loop. Meanwhile, approaches such as one-step matching [23], kernel ridge regression (KRR) [45] and pre-trained model [53] simplify the relay model updates in the inner loop. Despite these advancements, the optimization of the condensed graph still involves back-and-forth gradient calculations and updates, resulting in a time-consuming condensation procedure.

To address these issues, we investigate the foundational objective of existing optimization strategies and design an exceedingly efficient GC approach with a training-free framework termed Class-partitioned Graph Condensation (CGC). As illustrated in Figure 1(b), CGC refines the distribution matching from the class-to-class paradigm to a delicate class-to-node distribution matching paradigm. Notably, this refinement further simplifies the distribution matching objective as a class partition problem, which can be efficiently optimized using any clustering method (e.g., K-Means).

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

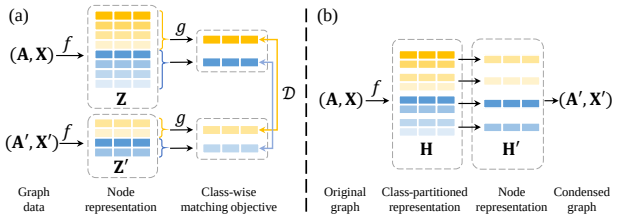

Figure 1: (a) The class-to-class matching paradigm in existing GC methods. (b) Our proposed class-to-node matching paradigm. f denotes the relay model, $\mathcal{D}$ represents the distance function, and g measures the matching objective (refer to Table 1 for details).

Moreover, CGC utilizes the pre-defined graph structure with the Dirichlet energy constraint [25] to derive a closed-form solution for the condensed node features. Equipped with these non-parametric modules, CGC eliminates gradient-based optimization in existing GC methods, enabling the condensation process to be executed on CPUs only. This enhances both the efficiency and effectiveness of the GC process, significantly broadening the utility of GC in real-world applications.

The main contributions of this paper are three-fold:

- **New observations and insights.** We theoretically demonstrate that existing GC optimization strategies converge to the class-level distribution matching paradigm, and subsequently simplify this process into a class partition problem, eliminating the sophisticated bi-level optimization and enabling efficient resolution through any clustering method.
- **New GC framework.** We present CGC, the first training-free GC framework characterized by a fine-grained class-to-node matching paradigm and closed-form feature generation, facilitating both precise and efficient GC procedures. Furthermore, CGC demonstrates considerable versatility, and any component within the framework can be replaced with a variety of alternative methods, such as distinct propagation and partition techniques.
- **State-of-the-art performance.** Extensive experiments demonstrate that CGC achieves advanced accuracy with an extremely fast condensation procedure. For instance, it condenses the Ogbn-products dataset within 30 seconds, which is 1,038× faster than GCond [24] and 148× faster than the most efficient GC baseline SimGC [53]. Our code is available at: https://anonymous.4open.science/r/CGC-condensation/.

2 Rethinking Existing Graph Condensation Methods

In this section, we first formally formulate the graph condensation and then revisit the existing GC optimization strategies along with their mutual connections. Subsequently, we formulate these strategies within a unified framework and demonstrate their adherence to a common class-level distribution matching paradigm. Finally, we simplify this paradigm into a class partition problem, enabling efficient optimization through clustering methods.

2.1 Problem Formulation

We consider a large-scale original graph $\mathcal{T} = \{\mathbf{A}, \mathbf{X}\}$ with N nodes, where $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix and $\mathbf{X} \in \mathbb{R}^{N\times d}$ denotes the d-dimensional node feature matrix. Each node belongs to one of c classes $\{C_1, \cdots, C_c\}$, translating into numeric labels $\mathbf{y} \in \{1, ..., c\}^N$ and one-hot labels $\mathbf{Y} \in \mathbb{R}^{N\times c}$. Graph condensation [24] aims to generate a small condensed graph $\mathcal{S} = \{\mathbf{A}', \mathbf{X}'\}$, such that GNNs trained on $\mathcal{S}$ yield performance comparable to those trained on $\mathcal{T}$. Specifically, $\mathbf{A}' \in \mathbb{R}^{N'\times N'}$, $\mathbf{X}' \in \mathbb{R}^{N'\times d}$ and $c \le N' \ll N$. Similarly, each node in $\mathcal{S}$ belongs to one of c classes $\{C'_1, \cdots, C'_c\}$, and labels are denoted as $\mathbf{y}' \in \{1, ..., c\}^{N'}$ or $\mathbf{Y}' \in \mathbb{R}^{N'\times c}$. We follow GCond to pre-define the condensed node labels, which preserve the same class proportion as the original node labels. To facilitate the expression, we assume all nodes in the original and condensed graphs are organized in ascending order according to labels.

Notice that the generation of $\mathbf{A}'$ is optional in existing GC methods [24, 63, 66]. If $\mathbf{A}'$ is opted out, the identity matrix $\mathbf{I}$ is used instead in GNN training, and this approach is termed the graphless GC variant (a.k.a. structure-free GC [66]).

2.2 Class-level Matching Paradigm in GC

To achieve the GC objective, existing methods use a Siamese network architecture with a relay model f to encode both graphs as shown in Figure 1 (a) and employ three kinds of optimization strategies [19, 54, 58]: parameter matching, performance matching, and distribution matching.

Parameter matching posits that the parameters of the GNN classifier should possess high consistency whenever it is trained on $\mathcal{S}$ or $\mathcal{T}$. To this end, GCond [24] tries to match model parameters at each training step and simplifies the objective to facilitate that gradients generated by $\mathcal{S}$ match those from the same class in $\mathcal{T}$:

$$\mathcal{L}_{GM} = \mathop{\mathbb{E}}_{\boldsymbol{\Theta}\sim\Phi}\left[\sum_{i=1}^{c} \mathcal{D}(\nabla_{\boldsymbol{\Theta}}\mathcal{L}_i^{\mathcal{S}}, \nabla_{\boldsymbol{\Theta}}\mathcal{L}_i^{\mathcal{T}})\right], \tag{1}$$

where Φ is the distribution of the relay model (i.e., the GNN classifier) parameter $\boldsymbol{\Theta}$, and we omit relay model update in the inner loop for simplicity. $\mathcal{D}$ indicates the distance function. $\mathcal{L}_i^{\mathcal{S}}$ and $\mathcal{L}_i^{\mathcal{T}}$ are classification losses (e.g., cross-entropy loss) for class i w.r.t $\mathcal{S}$ and $\mathcal{T}$, respectively. However, gradient matching [64] may accumulate errors when the relay model is iteratively updated on $\mathcal{S}$ over multiple steps. To mitigate this problem, SFGC [66] introduces trajectory matching [3] to align the long-term training trajectories of classification models. Nonetheless, to avoid overfitting one initialization of the relay model, it requires training hundreds of GNNs on $\mathcal{T}$ to obtain the trajectories, resulting in heavy condensation computations.

Performance matching aligns the performance of models trained on $\mathcal{S}$ and $\mathcal{T}$ by ensuring that the model trained on $\mathcal{S}$ achieves minimal loss on $\mathcal{T}$ [58]. To obtain the optimal model on $\mathcal{S}$, KiDD [55] and GC-SNTK [45] substitute the classification task with the regression and incorporate KRR [38] in the GC procedure for a closed-form solution of the relay model. The objective is formulated as:

$$\mathcal{L}_{PM} = \left\|\mathbf{Y} - \mathbf{Z}\mathbf{Z}'^{\top}\left(\mathbf{Z}'\mathbf{Z}'^{\top} + \lambda\mathbf{I}\right)^{-1}\mathbf{Y}'\right\|^2, \tag{2}$$

Table 1: The comparison of different GC optimization strategies.

Optimization strategy	Representative methods	f	g
Parameter matching	GCond [24], SFGC [66], GEOM [62], GCSR [34]	GNN	Classification model parameter
Performance matching	GC-SNTK [45], KiDD [55]	Graph kernel	Regression model parameter
Distribution matching	GCDM [30], GCEM [31], SimGC [53]	GNN	Class prototype

where $\mathbf{Z}' \in \mathbb{R}^{N' \times d}$ and $\mathbf{Z} \in \mathbb{R}^{N \times d}$ are node embeddings for $\mathcal{S}$ and $\mathcal{T}$, respectively. λ is a small constant weight of the regularization term for numerical stability. $\|\cdot\|$ denotes the ℓ_2 norm. However, graph kernels used in KRR are computationally intensive and memory-consuming, limiting their scalability in large graphs.

Distribution matching directly aligns the class distributions of the original and condensed graphs [63]. GCDM [30] and CaT [32] first introduce this strategy in GC by minimizing the discrepancy between class prototypes as:

$$\mathcal{L}_{DM} = \mathop{\mathbb{E}}_{\mathbf{\Theta} \sim \Phi} \left[\left\| \mathbf{P}'\mathbf{Z}' - \mathbf{P}\mathbf{Z} \right\|^2 \right], \tag{3}$$

where $\mathbf{P}' \in \mathbb{R}^{c \times N'}$ and $\mathbf{P} \in \mathbb{R}^{c \times N}$ are linear aggregation matrices to construct c class-level features, i.e., class prototypes, for $\mathcal{S}$ and $\mathcal{T}$, respectively. Specifically, $\mathbf{P}'_{i,j} = \frac{1}{|C'_i|}$ if $\mathbf{y}'_j = i$, and $\mathbf{P}'_{i,j} = 0$ otherwise. $\mathbf{P}_{i,j} = \frac{1}{|C_i|}$ if $\mathbf{y}_j = i$, and $\mathbf{P}_{i,j} = 0$ otherwise. Here, $|C'_i|$ and $|C_i|$ represent the sizes of class i in $\mathcal{S}$ and $\mathcal{T}$. By eliminating the need to calculate model gradients and parameters, distribution matching achieves an efficient and flexible condensation process, making it prevalent in recent GC studies, e.g., eigenbasis matching [31] and pre-trained model-based distribution matching [52, 53].

Analysis. Despite variations in format, existing optimization strategies can be uniformly formulated in a matching paradigm as:

$$\mathcal{S} = \arg\min_{\mathcal{S}} \mathcal{D} \left[g\left(f\left(\mathcal{S}\right), \mathbf{Y}'\right), g\left(f\left(\mathcal{T}\right), \mathbf{Y}\right) \right], \tag{4}$$

where f is the relay model. The function g measures the matching objective for distance function $\mathcal{D}$, which varies in format for diverse optimization strategies as shown in Table 1. Besides adopting the same framework, these three optimization strategies are inherently interconnected. Inspired by the study in CV [58], we investigate their relationship and put forward three propositions as follows:

Proposition 1. *The performance matching objective is equivalent to the optimal parameter matching objective.*

Proposition 2. *The distribution matching objective represents a simplified formulation of the performance matching, omitting feature correlation considerations.*

Proposition 3. *The distribution matching objective with a feature correlation constraint provides an upper bound for the parameter matching objective.*

The proofs of propositions are deferred to the Appendix A.1, A.2 and A.3, respectively.

Remark 1. *In light of Propositions 1-3, various optimization strategies converge to class-level feature matching between the original and condensed graphs, i.e., distribution matching in Eq. (3).*

However, on top of the efficiency issue discussed earlier, this matching paradigm only emphasizes on simulating the class distribution of the original graph by assigning a unified objective for all condensed nodes in the same class, resulting in the coarse-grained optimization target for each node. To mitigate these issues, in the following subsection, we investigate the distribution matching objective and refine the feature matching from the class-to-class paradigm into the class-to-node paradigm, thereby providing each condensed node with an explicit and efficient optimization target.

2.3 Simplifying Distribution Matching

For the sake of simplicity, we first follow existing GC methods [14, 24, 30, 32, 53] to specify the relay model f as widely used SGC [49], which decouples the propagation layer and transformation layer in GNN for efficient graph encoding as:

$$\begin{aligned} \mathbf{Z}' &= \mathbf{H}'\mathbf{\Theta} = \hat{\mathbf{A}}'^{K}\mathbf{X}'\mathbf{\Theta}, \\ \mathbf{Z} &= \mathbf{H}\mathbf{\Theta} = \hat{\mathbf{A}}^{K}\mathbf{X}\mathbf{\Theta}, \end{aligned} \tag{5}$$

where $\mathbf{\Theta}$ is the learnable weight matrix to transform the K-th order propagated features $\mathbf{H}'$ and $\mathbf{H}$. $\hat{\mathbf{A}}' = \tilde{\mathbf{D}}'^{\frac{1}{2}}\tilde{\mathbf{A}}'\tilde{\mathbf{D}}'^{\frac{1}{2}}$ and $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{\frac{1}{2}}$ represent the symmetric normalized adjacency matrices, where $\tilde{\mathbf{A}}'$ and $\tilde{\mathbf{A}}$ are adjacency matrices with self-loops. $\tilde{\mathbf{D}}'$ and $\tilde{\mathbf{D}}$ are degree matrices for $\tilde{\mathbf{A}}'$ and $\tilde{\mathbf{A}}$, respectively. Consequently, the distribution matching objective in Eq. (3) is formulated as:

$$\arg\min_{\mathcal{S}} \mathop{\mathbb{E}}_{\mathbf{\Theta} \sim \Phi} \left[\left\| \mathbf{P}'\mathbf{H}'\mathbf{\Theta} - \mathbf{P}\mathbf{H}\mathbf{\Theta} \right\|^2 \right]. \tag{6}$$

To facilitate the *class-to-node distribution matching paradigm*, we enhance the aggregation matrix $\mathbf{P}$ with two objectives:

- The number of aggregated features is expanded from c to N', ensuring that each aggregated feature in the original graph corresponds to a distinct condensed node;
- Aggregations are performed within classes to preserve the class semantics of aggregated features.

Consequently, it is expected that $|C_i|$ original nodes in class i will be aggregated into $|C'_i|$ features to match with condensed nodes in C'_i. This aggregation procedure analogizes to a class partition problem, defined as follows:

Definition 1. **Class partition.** *The class partition divides $|C_i|$ nodes in class i into $|C'_i|$ non-overlapping sub-classes $\{S^i_1, ..., S^i_{|C'_i|}\}$, with each sub-class characterized by a centroid aggregated by constituent nodes. The node mapping function of this partition is defined as $\pi^i : \{1, ..., |C_i|\} \to \{1, ..., |C'_i|\}$ and the class-wise aggregation matrix $\mathbf{R}^i \in \mathbb{R}^{|C'_i| \times |C_i|}$ is formulated as:*

$$\mathbf{R}^i_{j,k} = \begin{cases} \frac{1}{\left|S^i_j\right|} & \text{if } \pi^i(k) = j \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where $\mathbf{R}^i_{j,k}$ denotes the aggregation weight for node k in class i, and $\pi^i(k)$ indicates the subclass index for node k. $\left|S^i_j\right|$ represents the size of sub-class S^i_j.

Table 2: The comparison of simplified distribution matching with the conventional objective. Speedup ratios compared to GCDM-X are indicated in brackets (r: condensation ratio, model: GCN).

Dataset	r	Accuracy (%)			Condensation time (s)		
		GCDM	GCDM-X	SimDM	GCDM	GCDM-X	SimDM
Cora	2.60%	77.2±0.4	81.4±0.1	80.1±0.7	27.3	24.0	0.3 (80×)
Citeseer	1.80%	69.5±1.1	71.9±0.5	70.9±0.6	62.0	51.0	0.7 (73×)
Ogbn-arxiv	0.25%	59.6±0.4	61.2±0.1	66.1±0.4	690.0	469.0	7.1 (66×)
Flickr	0.50%	46.8±0.1	45.6±0.1	45.8±0.2	209.5	86.2	2.1 (41×)
Reddit	0.10%	89.7±0.2	87.2±0.1	90.6±0.1	921.9	534.8	8.2 (65×)

With the defined class partition, the aggregation matrix $\mathbf{P} \in \mathbb{R}^{c\times N}$ can be updated to $\hat{\mathbf{P}} \in \mathbb{R}^{N'\times N}$, which is constructed by organizing all class-wise aggregation matrices along the diagonal as $\hat{\mathbf{P}} = \text{diag}(\mathbf{R}^1, ..., \mathbf{R}^c)$. Assuming that condensed nodes are organized in ascending order based on their labels, $\mathbf{P}'$ degrades into the identity matrix $\mathbf{I}$, and Eq. (6) can be reformulated as:

$$\arg\min_{\mathcal{S},\hat{\mathbf{P}}} \mathop{\mathbb{E}}_{\boldsymbol{\Theta}\sim\Phi}\left[\left\|\mathbf{H}'\boldsymbol{\Theta} - \hat{\mathbf{P}}\mathbf{H}\boldsymbol{\Theta}\right\|^2\right], \tag{8}$$

which is upper-bounded by:

$$\mathop{\mathbb{E}}_{\boldsymbol{\Theta}\sim\Phi}\left[\left\|\mathbf{H}'\boldsymbol{\Theta} - \hat{\mathbf{P}}\mathbf{H}\boldsymbol{\Theta}\right\|^2\right] \leq \mathop{\mathbb{E}}_{\boldsymbol{\Theta}\sim\Phi}\left[\left\|\mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}\right\|^2 \|\boldsymbol{\Theta}\|^2\right]. \tag{9}$$

Given that $\boldsymbol{\Theta}$ is independent to $\mathcal{S}$ and $\hat{\mathbf{P}}$, we can minimise the upper-bound by achieving:

$$\arg\min_{\mathcal{S},\hat{\mathbf{P}}} \left\|\mathbf{H}' - \hat{\mathbf{P}}\mathbf{H}\right\|^2. \tag{10}$$

This objective can be further simplified under the graphless GC variant, where $\mathcal{S} = \{\mathbf{I}, \mathbf{X}'\}$ and $\mathbf{H}' = \hat{\mathbf{A}}'^K\mathbf{X}' = \mathbf{I}^K\mathbf{X}' = \mathbf{X}'$. Consequently, the objective is reformulated to:

$$\arg\min_{\mathbf{X}',\hat{\mathbf{P}}} \left\|\mathbf{X}' - \hat{\mathbf{P}}\mathbf{H}\right\|^2. \tag{11}$$

Remark 2. *The objective in Eq. (11) indicates that the condensed node feature $\mathbf{X}'$ in graphless GC can be obtained by performing class partition on the propagated features $\mathbf{H}$, eliminating the gradient-based distribution matching optimization. This $\mathbf{X}'$ can be directly used to train GNNs with the identity matrix $\mathbf{I}$.*

Specifically, we can efficiently obtain the solution by applying any Expectation-Maximization (EM) based clustering algorithms (e.g., K-means) to each class, iteratively updating the cluster centroid $\mathbf{X}'$ and the aggregation matrix $\hat{\mathbf{P}}$ until convergence. Note that another intuitive case arises when the size of the condensed graph matches the number of classes, i.e., $N' = c$. In this scenario, $\mathbf{P}' = \mathbf{I}$ and the conventional distribution matching objective in Eq. (6) degrades to calculating the class prototypes with the pre-defined $\mathbf{P}$, which is consistent with Eq. (11).

To validate the simplified objective, we compare the condensed graph $\{\mathbf{I}, \mathbf{X}'\}$, derived by objective in Eq. (11) and termed SimDM, with the conventional distribution matching-based methods (i.e., GCDM [30] and its graphless variation GCDM-X). The test accuracy and condensation time are detailed in Table 2 and experimental setting are deferred to Section 4. SimDM significantly excels in condensation time while maintaining comparable accuracy, confirming the effectiveness of our class partition-based objective.

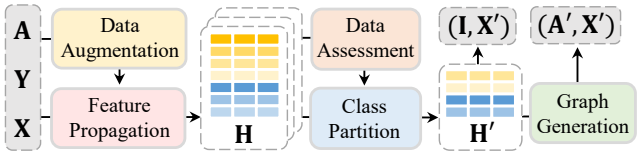

Figure 2: The pipeline of CGC and CGC-X.

3 Class-partitioned Graph Condensation (CGC)

Despite the potential to accelerate the condensation procedure, the class partition in SimDM neglects the data quality of the original graphs, a critical aspect in addressing data-centric challenges in GC. To establish a comprehensive data processing framework, we further enhance SimDM by incorporating data assessment, augmentation, and graph generation modules. Consequently, we introduce a novel GC framework, as illustrated in Figure 2, where all five involved modules are training-free, facilitating an efficient and robust condensation process.

3.1 Feature Propagation

To eliminate the gradient calculation in the condensation procedure, the non-parametric feature propagation module is deployed to smooth the node features $\mathbf{X}$ according to the original graph structure $\mathbf{A}$ and generate the node embeddings. The propagation method is replaceable with a variety of choices with diverse characteristics, e.g., SGC [49], personalized PageRank (PPR) [39] and SAGE [18], etc. Without loss of generality, we follow existing GC methods [14, 24, 30, 32, 53] and adopt the propagation method in SGC to generate the node embeddings:

$$\mathbf{H}^{(l)} = \hat{\mathbf{A}}^l\mathbf{X}, \tag{12}$$

where $0 \leq l \leq K$, and the embeddings in the last layer are denoted as $\mathbf{H} = \mathbf{H}^{(K)}$ for the final class partition.

3.2 Data Assessment

For the sake of robust class representation, we assess the node embeddings prior to the class partition process. Inspired by [65, 68], we incorporate node embeddings at various propagation depths and utilize a simple linear layer $\mathbf{W}$ as the classifier for label prediction.

$$\hat{\mathbf{Y}} = \mathbf{T}\mathbf{W} = \frac{1}{K+1}\sum_{l=0}^{K}\mathbf{H}^{(l)}\mathbf{W}. \tag{13}$$

Subsequently, the MSE loss is employed for optimization:

$$\arg\min_{\mathbf{W}} \left\|\mathbf{Y} - \hat{\mathbf{Y}}\right\|^2. \tag{14}$$

This loss facilitates an efficient and precise closed-form solution, expressed as $\hat{\mathbf{W}} = \mathbf{T}^+\mathbf{Y}$, where $\mathbf{T}^+$ denotes the pseudo inverse of $\mathbf{T}$. Afterwards, the linear classifier is utilised to evaluate node embeddings $\mathbf{H}^{(l)}$ for $0 \leq l \leq K$, and their confidence scores w.r.t the ground-truth are recorded as $\mathbf{r} \in \mathbb{R}^{N(K+1)}$. Additionally, we calculate the prediction errors for each class of $\mathbf{H}$ and represent these class prediction errors as $\mathbf{e} = [e_1, ..., e_c]$. The confidence scores $\mathbf{r}$ reflect the reliability of node embeddings in class representation, while the class prediction errors $\mathbf{e}$ highlight the difficulty associated with each class. In subsequent modules, these metrics will

inform the data augmentation strategies and calibration of condensed features, aiming to enhance the quality of the condensed graph.

3.3 Data Augmentation

Although the non-overlapping class partition in SimDM simplifies optimization, it reduces the number of original nodes matched by each condensed node, intensifying the dependency on the quantity of nodes involved in the partition process. Concurrently, the time complexity of class partition scales proportionally with the increasing number of nodes. To address these issues and achieve equilibrium, we propose augmenting underrepresented classes with additional node embeddings to ensure robust class representations.

Specifically, we utilize the node embeddings with smaller propagation depths as augmentations, i.e., $\mathbf{H}^{(l)}$, where $0 \leq l < K$. It is important to note that only training set nodes are involved, with each node paired with its respective label. For clarity, we refer to these node embeddings collectively as $\mathbb{H}$. The utility of these embeddings offers threefold benefits. Firstly, these embeddings are an intermediate product in the feature propagation process and incur no additional computational overhead. Secondly, they aggregate node features across various hops, exhibiting different levels of smoothness [61]. Lastly, different from data augmentation methods such as mixup, edge drop, or feature mask, these embeddings prevent the introduction of excessive noises during the condensation process.

Subsequently, class prediction errors $\mathbf{e}$ are used as sampling weights to randomly select $p\%$ of the embeddings from $\mathbb{H}$, where p serves as a hyper-parameter controlling the size of augmentation data. This sampling strategy prioritizes classes with higher errors, potentially enhancing the class representation quality.

Consequently, the embeddings and labels of the sampled nodes are represented as $\mathbf{H}_{aug}$ and $\mathbf{y}_{aug}$, respectively, and the augmented data for the condensation process are defined as $\mathbf{H}_{cond} = [\mathbf{H}; \mathbf{H}_{aug}]$ and $\mathbf{y}_{cond} = [\mathbf{y}; \mathbf{y}_{aug}]$.

3.4 Class Partition

With enhanced label $\mathbf{y}_{cond}$, we perform class partition on $\mathbf{H}_{cond}$ to address the optimization problem in Eq. (11). Initially, nodes labeled in $\mathbf{y}_{cond}$ are categorised into c classes $\{\hat{C}_1, \cdots, \hat{C}_c\}$ with corresponding embeddings $\{\hat{\mathbf{H}}_1, \cdots, \hat{\mathbf{H}}_c\}$. Then, $\hat{\mathbf{H}}_i$ for class i is partitioned into $|C'_i|$ sub-classes $\{S^i_1, ..., S^i_{|C'_i|}\}$ by applying any EM-based clustering method and the node mapping function is denoted by π^i.

Subsequently, rather than utilizing the class-wise aggregation matrix $\mathbf{R}^i$ as initially defined in Eq. (7), we incorporate the confidence scores $\mathbf{r}$ for each aggregated node to calibrate the condensed node embeddings. Consequently, Eq. (7) is updated as:

$$\mathbf{R}^i_{j,k} = \begin{cases} \frac{\mathbf{r}_k}{\tau} & \text{if } \pi^i(k) = j \\ 0 & \text{otherwise} \end{cases}, \tag{15}$$

where $\mathbf{R}^i_{j,k}$ denotes the aggregation weight for node k in subclass j, and $\pi^i(k)$ indicates the subclass index for node k. $\mathbf{r}_k$ is the confidence score of node k, and τ represents the global temperature to control the sensitivity for confidence scores. Following this, the row-normalized aggregation matrix $\hat{\mathbf{R}}^i = \text{norm}(\mathbf{R}^i)$ is employed to produce aggregated embeddings as: $\mathbf{H}'_i = \hat{\mathbf{R}}^i \hat{\mathbf{H}}_i$, and the condensed node embeddings are constructed by: $\mathbf{H}' = [\mathbf{H}'_1; ...; \mathbf{H}'_c]$.

3.5 Graph Generation

We follow existing GC methods to provide two parameterization methods for the condensed graph, including CGC with the graph generation and its graphless variant CGC-X.

According to Eq. (5), we expect a symmetric encoding procedure between the original and condensed graphs. Therefore, our objective is to construct $\mathbf{A}'$ and $\mathbf{X}'$ satisfying $\hat{\mathbf{A}}'^K \mathbf{X}' = \mathbf{H}'$. To this end, we utilize the pre-defined graph structure and calculate $\mathbf{X}'$ in a close-formed solution.

Specifically, we construct the condensed graph structure according to the condensed node embeddings [69] as follows:

$$\mathbf{A}'_{i,j} = \begin{cases} 1 & \text{if } \cos(\mathbf{H}'_i, \mathbf{H}'_j) > T \\ 0 & \text{otherwise} \end{cases}, \tag{16}$$

where $\cos(\cdot, \cdot)$ measures the cosine similarity and T is the hyper-parameter for graph sparsification. To ensure that generated features change smoothly between connected nodes, we introduce the Dirichlet energy constraint [25] in feature reconstruction loss and quantify the smoothness of graph signals. The optimization objective for $\mathbf{X}'$ is formulated as:

$$\mathcal{L} = \arg\min_{\mathbf{X}'} \left\| \hat{\mathbf{A}}'^K \mathbf{X}' - \mathbf{H}' \right\|^2 + \alpha \text{tr}(\mathbf{X}'^\top \mathbf{L}' \mathbf{X}'), \tag{17}$$

where α balances the losses, and $\text{tr}(\cdot)$ denotes the matrix trace. $\mathbf{L}' = \mathbf{D}' - \mathbf{A}'$ is the Laplacian matrix, where $\mathbf{D}'$ is the degree matrix. Consequently, the closed-form solution for Eq. (17) is presented as:

Proposition 4. *Assume an undirected condensed graph $\mathcal{S} = \{\mathbf{A}', \mathbf{X}'\}$, the closed-form solution of Eq. (17) takes the form: $\mathbf{X}' = (\mathbf{Q}^\top \mathbf{Q} + \alpha \mathbf{L}')^{-1} \mathbf{Q}^\top \mathbf{H}'$, where $\mathbf{Q} = \hat{\mathbf{A}}'^K$.*

The proof is deferred to the Appendix A.4. Although the closed-form solution involves an inverse operation, the target matrix is small (i.e., N'-by-N') and can be efficiently calculated.

A Graphless Variant. Based on Remark 2, the node embedding $\mathbf{H}'$ equivalents to the condensed node feature $\mathbf{X}'$ when utilizing $\mathbf{I}$ as the condensed graph structure and employing non-parametric feature propagation for graph encoding. Consequently, $\mathbf{H}'$ derived in the Class Partition module can directly serve as the condensed graph for CGC-X, i.e., $\mathcal{S} = \{\mathbf{I}, \mathbf{X}'\} = \{\mathbf{I}, \mathbf{H}'\}$.

Comparison with Coarsening Methods. Similar to our simplified GC objective in Eq. (11), coarsening methods develop the aggregation matrix to merge original nodes into super-nodes and transform the original graph structure into a smaller graph. However, these methods are implemented within an unsupervised paradigm, prioritizing the preservation of graph properties such as spectral [36] and cut [35] guarantees while disregarding label information [19]. In contrast, our CGC framework synthesizes nodes and connections under a supervised paradigm, thereby enhancing the utility of downstream tasks.

The detailed **algorithm** and **time complexity analysis** can be found in Appendix B and C, respectively.

Table 3: The accuracy (%) comparison between our methods (CGC and CGC-X) and baselines. OOM means out-of-memory. The best (bold) and runner-up (underlined) performances for (I, X') and (A', X') are highlighted, respectively.

Dataset (homo.)	r	(I, X')						(A', X')								Whole Dataset
		GCond-X	GCDM-X	SNTK-X	SFGC	GEOM	CGC-X	VN	A-CM	GCond	GCDM	SNTK	SimGC	GCSR	CGC	
Cora (0.81)	1.30%	75.9±1.2	81.3±0.4	82.2±0.3	80.1±0.4	80.3±1.1	**83.4±0.3**	31.2±0.2	74.6±0.1	79.8±1.3	69.4±1.3	81.7±0.7	80.8±2.3	79.9±0.7	**82.7±0.3**	81.2±0.2
	2.60%	75.7±0.9	81.4±0.1	82.4±0.5	81.7±0.5	81.5±0.8	**83.4±0.4**	65.2±0.6	72.8±0.2	80.1±0.6	77.2±0.4	81.5±0.7	80.9±2.6	80.6±0.8	**83.5±0.1**	
	5.20%	76.0±0.9	82.5±0.3	82.1±0.1	81.6±0.8	82.2±0.4	**82.8±1.0**	70.6±0.1	78.0±0.3	79.3±0.3	79.4±0.1	81.3±0.2	82.1±1.3	81.2±0.9	**82.5±0.6**	
Citeseer (0.74)	0.90%	71.4±0.8	69.0±0.5	69.9±0.4	71.4±0.5	71.1±0.2	**73.1±0.8**	52.2±0.4	65.1±0.1	70.5±1.2	62.0±0.1	66.4±1.0	73.8±2.5	70.2±1.1	**73.9±0.8**	71.7±0.1
	1.80%	69.8±1.1	71.9±0.5	69.9±0.5	72.4±0.4	71.3±0.1	**72.6±0.2**	59.0±0.5	66.0±0.2	70.6±0.9	69.5±1.1	68.4±1.1	72.2±0.5	71.7±0.9	**72.4±0.2**	
	3.60%	69.4±1.4	72.8±0.6	69.1±0.4	70.6±0.7	72.1±1.0	**72.9±0.2**	65.3±0.5	66.1±0.2	69.8±1.4	69.8±0.2	69.8±0.8	71.1±2.8	**74.0±0.4**	72.5±0.1	
Arxiv (0.65)	0.05%	61.3±0.5	61.0±0.1	63.9±0.3	65.5±0.7	64.7±0.4	**65.8±0.5**	35.4±0.3	58.0±0.1	59.2±1.1	59.3±0.3	64.4±0.2	63.6±0.8	60.6±1.1	**64.5±0.7**	71.4±0.1
	0.25%	64.2±0.4	61.2±0.1	65.5±0.1	66.1±0.4	**67.5±0.3**	66.8±0.1	43.5±0.2	60.0±0.3	63.2±0.3	59.6±0.4	65.1±0.8	66.4±0.3	65.4±0.8	**67.0±0.3**	
	0.50%	63.1±0.5	62.5±0.1	65.7±0.4	66.8±0.4	**67.6±0.2**	67.0±0.1	50.4±0.1	61.0±0.2	64.0±0.4	62.4±0.1	65.4±0.5	66.8±0.4	65.9±0.6	**67.2±0.4**	
Flickr (0.33)	0.10%	45.9±0.1	46.0±0.1	46.6±0.3	46.6±0.2	46.1±0.5	**46.7±0.2**	41.9±0.2	42.2±0.1	46.5±0.4	46.1±0.1	46.7±0.1	45.3±0.7	46.6±0.3	**46.8±0.0**	47.2±0.1
	0.50%	45.0±0.2	45.6±0.1	46.7±0.1	**47.0±0.1**	46.2±0.2	**47.0±0.1**	44.5±0.1	45.2±0.3	**47.1±0.1**	46.8±0.1	46.8±0.1	45.6±0.4	46.6±0.2	**47.1±0.1**	
	1.00%	45.0±0.1	45.4±0.3	46.6±0.2	**47.1±0.1**	46.7±0.1	47.0±0.1	44.6±0.1	45.1±0.1	**47.1±0.1**	46.7±0.1	46.5±0.2	43.8±1.5	46.8±0.2	47.0±0.1	
Reddit (0.78)	0.05%	88.4±0.4	86.5±0.2	OOM	89.7±0.2	90.1±0.2	**90.3±0.2**	40.9±0.5	72.2±1.2	88.0±1.8	89.3±0.1	OOM	89.6±0.6	90.5±0.2	**90.6±0.2**	93.9±0.0
	0.10%	89.3±0.1	87.2±0.1	OOM	90.0±0.3	90.4±0.1	**90.8±0.0**	42.8±0.8	73.5±1.0	89.6±0.7	89.7±0.2	OOM	90.6±0.3	91.2±0.2	**91.4±0.1**	
	0.20%	88.8±0.4	88.8±0.1	OOM	89.9±0.4	90.9±0.1	**91.4±0.1**	47.4±0.9	75.1±1.3	90.1±0.5	90.2±0.4	OOM	91.4±0.2	**92.2±0.1**	91.6±0.2	
Products (0.81)	0.025%	64.5±0.2	65.1±0.1	OOM	66.2±0.3	67.7±0.2	**68.0±0.0**	34.3±0.8	58.8±0.9	64.2±0.1	66.1±0.1	OOM	63.7±1.1	66.5±0.2	**68.0±0.1**	73.1±0.0
	0.050%	65.2±0.3	66.8±0.2	OOM	67.0±0.2	68.4±0.3	**68.9±0.2**	35.1±0.9	60.1±0.6	64.7±0.2	67.4±0.4	OOM	64.9±1.2	67.8±0.3	**68.9±0.3**	
	0.100%	65.5±0.2	67.2±0.1	OOM	68.8±0.3	68.7±0.5	**69.0±0.1**	37.4±0.9	62.4±0.9	65.0±0.1	68.4±0.3	OOM	67.2±1.4	68.5±0.3	**69.1±0.2**	

4 Experiments

We design comprehensive experiments to validate the efficacy of our proposed methods and explore the following research questions:

Q1: Compared to the other graph reduction methods, can the condensed graph generated by CGC and CGC-X achieve better GNN performance?

Q2: Can the CGC and CGC-X condense the graph faster than other GC approaches?

Q3: Can the condensed graph generated by CGC and CGC-X generalize well to different GNN architectures?

Q4: How do the different components, i.e., data augmentation, data assessment and class partition methods affect CGC and CGC-X?

Q5: How do the different hyper-parameters affect the CGC and CGC-X?

4.1 Experimental Setup

Datasets & Baselines. We evaluate our proposed methods on four transductive datasets (Cora, Citeseer [26], Ogbn-arxiv (Arxiv) [20] and Ogbn-products (Products) [20]), as well as two inductive datasets (Flickr and Reddit [59]), all with public splits. We compare 12 baselines, encompassing both graph coarsening and graph condensation methods with diverse optimization strategies: (1) graph coarsening methods: Variation Neighborhoods (VN) [21, 35] and A-ConvMatch (A-CM) [6]; (2) gradient matching-based GC methods: GCond and GCond-X [24]; (3) trajectory matching-based GC methods: SFGC [66], GEOM [62] and GCSR [34]; (4) KRR-based GC methods: SNTK and SNTK-X [45]; (5) distribution matching-based GC methods: GCDM, GCDM-X [30] and SimGC [53]. Notice that the suffix "-X" represents the graphless variant. More details about the datasets and baselines are provided in Appendix D.1 and D.2, respectively.

Implementations. Following GCond [24], we evaluate three condensation ratios ($r = N'/N$) for each dataset. In the transductive setting, N represents the original graph size, while in the inductive setting, N indicates the sub-graph size observed in the training stage. Two-layer GNNs with 256 hidden units are used for evaluation. We adopt the propagation method in SGC for feature propagation and spectral clustering [12] with acceleration implementation (i.e., FAISS [9]) for class partition (refer to Section 4.5 for results of alternative class partition methods). For reproducibility, other detailed implementations, hyper-parameters and computing infrastructure are summarised in Appendix D.3.

4.2 Effectiveness Comparison (Q1)

For the sake of fairness, we compare CGC-X and CGC with graphless and graph-generated GC baselines separately. The condensed graphs generated by GC methods are evaluated to train a 2-layer GCN and the test accuracies with standard deviation are reported in Table 3. In the table, "Whole Dataset" refers to the GCN performance which is trained on the original graph and we make the following observations. Firstly, CGC-X and CGC consistently outperform other baselines across all datasets. While GCSR achieves the best performance on Citeseer and Reddit under the largest condensation ratio, the performance gap between CGC and GCSR remains small. Moreover, CGC significantly outperforms GCSR across other condensation ratios. Notably, on the Arxiv dataset, CGC demonstrates substantial improvement over GCSR, highlighting the superiority of our proposed method. Furthermore, our proposed method can effectively mitigate the label sparsity issue in GC. On two datasets with sparse labels, i.e., Cora and Citeseer, CGC and CGC-X consistently achieve superior performances. This is contributed to the data augmentation module which can introduce reliable nodes for precise class distribution representation in the GC procedure.

Table 4: The condensation time (seconds) comparison of different graph reduction methods. OOM denotes out-of-memory. r is set as 2.60%, 1.80%, 0.25%, 0.50%, 0.10% and 0.05% for six datasets, respectively.

Dataset	VN	A-CM	GCond	GCond-X	GCDM	GCDM-X	SNTK	SNTK-X	SimGC	SFGC	GEOM	GCSR	CGC	CGC-X
Cora	2.7	7.6	498.9	80.2	27.3	24.0	30.8	20.3	210.6	2,524.4	3,302.2	850.4	1.4	0.4
Citeseer	3.5	141.4	479.3	70.6	62.0	51.0	19.4	18.7	227.6	3,877.0	4,230.3	488.6	1.5	0.8
Arxiv	412.6	179.9	14,876.0	13,516.7	690.0	469.0	9,353.6	9,079.9	252.4	86,288.5	108,962.5	5,283.5	8.8	7.9
Flickr	243.8	160.1	1,338.3	1,054.7	209.5	86.2	562.6	510.0	345.8	52,513.0	54,601.9	1,202.6	7.5	6.8
Reddit	464.1	192.1	21,426.5	22,154.3	921.9	534.8	OOM	OOM	475.8	246,997.1	248,837.4	1,747.3	18.5	17
Products	59,074.1	6,254.6	26,789.3	24,145.4	7,897.9	5,790.6	OOM	OOM	3,824.3	283,068.2	284,941.5	18,901.3	31.5	25.8

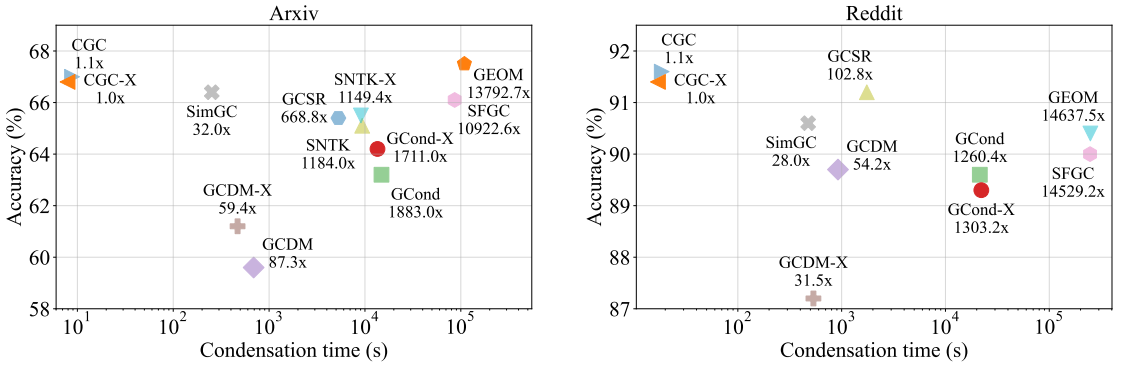

Figure 3: The accuracy and condensation time comparison of GC methods on Arxiv (r = 0.25%) and Reddit (r = 0.10%). SNTK and SNTK-X are out-of-memory on Reddit dataset.

4.3 Efficiency Comparison (Q2)

We now report the condensation time of the proposed method and baselines on different datasets. For each method, we repeated the experiments 5 times and the averaged condensation time is reported in Table 4. Instead of adopting the Siamese network architecture as other existing GC methods, CGC and CGC-X eliminate the gradient-based optimization and achieve extremely efficient condensation procedures. All datasets can be condensed within 1 minute, which is multiple orders of magnitude improvement compared to other GC methods. In contrast, graph coarsening methods use a hierarchical node aggregation paradigm to iteratively reduce the graph size, which leads to high latency on large-scale datasets. To facilitate a clearer comparison of the methods, Figure 3 shows the accuracy of GC methods over their condensation time on the transductive dataset (Arxiv) and inductive dataset (Reddit). The relative condensation time to the fastest CGC-X is marked. Our proposed CGC achieves the highest test accuracy, and CGC-X is 32.0× and 28.0× faster than the most efficient baseline SimGC.

4.4 Generalizability Comparison (Q3)

To compare the generalizability across different GNN architectures, we assess the performance of GC methods under different GNN models, including GCN, SGC, SAGE [18], APPNP [16], Cheby [5] and GAT [44]. The detailed accuracies of graph-generated GC methods are shown in Table 5. The results of graphless GC methods can be found in Appendix E. We could observe that all GNNs trained on the condensed graph generated by CGC exhibit similar levels of performance and CGC achieves a significant improvement over other compared baselines. This indicates the effectiveness of the pre-defined graph structure in our proposed method, which captures the relationship among aggregated features and encourages smoothness among connected condensed nodes.

Table 5: The generalizability of GC methods with graph generation. SNTK is out-of-memory on Reddit. AVG indicates the average value. The best performances are highlighted.

Dataset (r)	Method	SGC	GCN	SAGE	APPNP	Cheby	GAT	AVG
Arxiv (0.25%)	GCond	63.7	63.2	62.6	63.4	54.9	60.0	61.3
	GCDM	61.2	59.6	61.1	62.8	55.4	61.2	60.2
	SNTK	62.7	65.1	62.9	62.6	55.1	61.8	61.7
	SimGC	64.3	66.4	60.4	61.5	54.7	61.1	61.4
	GCSR	**65.6**	65.4	65.4	**64.4**	58.9	63.5	63.9
	CGC	64.9	**67.0**	**65.7**	63.7	**60.5**	**65.0**	**64.5**
Flickr (0.50%)	GCond	46.1	47.1	46.2	45.9	42.8	40.1	44.7
	GCDM	44.3	46.8	45.8	45.2	41.8	41.9	44.3
	SNTK	45.7	46.8	45.9	45.3	41.3	41.4	44.4
	SimGC	43.4	45.6	44.4	44.8	42.8	41.2	43.7
	GCSR	46.3	46.6	**46.6**	46.3	44.9	45.6	46.1
	CGC	**47.3**	**47.1**	**46.6**	**46.9**	**45.7**	**46.1**	**46.6**
Reddit (0.10%)	GCond	89.6	89.6	89.1	87.8	75.5	60.2	82.0
	GCDM	88.0	89.7	89.3	88.9	74.9	69.3	83.3
	SimGC	90.8	90.6	86.2	88.6	76.2	65.1	82.9
	GCSR	91.0	91.2	**91.0**	**88.9**	80.4	86.4	88.2
	CGC	**91.3**	**91.4**	90.2	88.7	**81.7**	**89.1**	**88.7**

4.5 Ablation Study (Q4)

Data Augmentation. To validate the impact of data augmentation, CGC and CGC-X are evaluated by disabling the augmentation component (termed "w/o AUG") and results are shown in Table 6.

Table 6: The ablation study. r is set as 0.25%, 0.50% and 0.10% for evaluated datasets, respectively.

Method	Arxiv	Flickr	Reddit
CGC-X w/o AUG	66.5+0.1	46.6+0.2	90.7+0.1
CGC-X w/o CAL	66.3+0.2	46.2+0.2	90.6+0.1
CGC-X w K-means	66.7+0.1	46.9+0.1	90.7+0.1
CGC-X	**66.8+0.1**	**47.0+0.1**	**90.8+0.0**
CGC w/o AUG	66.4+0.1	46.7+0.1	90.9+0.1
CGC w/o CAL	66.5+0.2	46.3+0.1	91.0+0.1
CGC w K-means	**67.2+0.2**	46.9+0.2	91.3+0.2
CGC	67.0+0.3	**47.1+0.1**	**91.4+0.1**

Table 7: The performance comparison of different propagation methods. r is set as 2.60%, 1.80%, 0.25%, 0.50% and 0.10% for five datasets, respectively. OOM indicates the out-of-memory. The best performances are highlighted.

Dataset	CGC-X			CGC		
	SGC	PPR	SAGE	SGC	PPR	SAGE
Cora	**83.4+0.4**	81.0+0.5	81.1±0.5	**83.5+0.1**	80.9+0.7	80.0±1.8
Citeseer	72.6+0.2	70.5+2.0	**73.0±0.2**	72.4+0.2	70.5+2.0	**72.9±0.3**
Arxiv	**66.8+0.1**	66.3±0.2	65.4+0.6	**67.0+0.3**	65.7+0.1	65.9+0.3
Flickr	**47.0+0.1**	**47.0+0.1**	46.7+0.2	**47.1+0.1**	47.0+0.1	46.5+0.0
Reddit	90.8+0.0	**90.9+0.1**	88.7±0.7	**91.4+0.1**	91.0+0.1	89.5±0.3

Due to the data augmentation, our proposed methods can introduce more labeled nodes in GC and facilitate a more precise class representation, leading to better performance on all the datasets.

Data Assessment. We compare the results of CGC and CGC-X with those obtained by removing the calibration of the condensed node embeddings (referred to as "w/o CAL"), as shown in Table 6. When the class-wise aggregation matrix is replaced by Eq. (7), the confidence score for each node is disregarded, leading to equal aggregation of all nodes within the clusters. This modification results in a performance drop across all datasets and verifies the effectiveness of our data assessment module.

Partition Method. In addition to spectral clustering, we also tested K-means for class partitioning, with results presented in Table 6. The similar performance levels across methods indicate that our proposed method is insensitive to the choice of partition method.

Propagation Method. We now represent the effect of different propagation methods in our proposed method. Besides the propagation method in SGC, we further evaluate two propagation methods, including personalised PageRank (PPR) [39] and SAGE [18]. The test accuracies are shown in Table 7. We can observe that performance varies for different propagation methods on different datasets. For instance, SAGE achieves better performance on Citeseer. PPR outperforms other methods on Reddit dataset. These results verify the generalization of our proposed method and the proper propagation method should be selected for different datasets.

4.6 Hyper-parameter Sensitivity Analysis (Q5)

Due to the training-free nature of our proposed methods, the number of hyper-parameters is significantly reduced. CGC contains four hyper-parameters: the constraint weight α, adjacency matrix threshold T, temperature τ, and augmentation ratio p. Since T and α are specific to the condensed graph generation, CGC-X contains only two hyper-parameters, making it the method with the fewest hyper-parameters among existing graph condensation approaches [54].

We examine the impact of these hyper-parameters on our method's performance, with the results presented in Figure 4. A higher T generally improves performance, indicating that a sparser adjacency matrix enhances node representation, in line with findings from [24]. The optimal value for α should be selected to balance node smoothness with feature reconstruction. Additionally, τ controls the contribution of nodes in the aggregation, and more complex datasets like Flickr benefit from a smaller τ, which emphasizes the reliability of node scores. The augmentation ratio p determines the number of augmented nodes, with increased augmentations leading to better results. However, an excessive number of augmented nodes can degrade performance and slow down the condensation process.

It is worth noting that the rapid condensation process of our method could significantly simplify hyper-parameter tuning, emphasizing the practical utility and superior effectiveness of our proposed method.

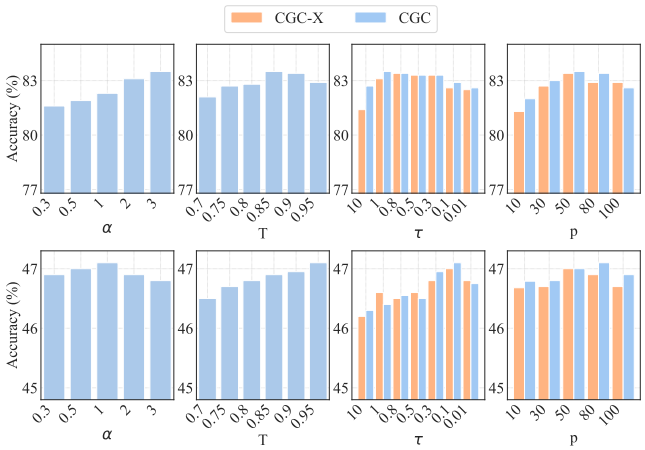

Figure 4: Test accuracy across varying hyper-parameters: the first row shows results for Cora (r = 2.60%), and the second row for Flickr (r = 0.50%).

5 Conclusion

In this paper, we present CGC, a training-free GC framework designed for efficient condensed graph generalization. CGC transforms the class-level distribution matching paradigm identified in existing GC methods into a class partition problem, enabling the EM-based clustering solution for complex condensation optimization. Moreover, CGC incorporates the pre-defined graph structure and closed-form feature solution, facilitating efficient condensed graph generation. Despite achieving a fast condensation procedure, this work primarily focuses on simple attribute graphs. Future work could extend the class partition framework to more practical graphs, such as heterophilic graphs, digraphs, and dynamic graphs, thereby broadening the horizons of GC applications.

References

[1] Gecia Bravo Hermsdorff and Lee Gunderson. 2019. A unifying framework for spectrum-preserving graph sparsification and coarsening. *Advances in Neural Information Processing Systems* 32 (2019).

[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] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*.

[6] Charles Dickens, Edward Huang, Aishwarya Reganti, Jiong Zhu, Karthik Subbian, and Danai Koutra. 2024. Graph coarsening via convolution matching for scalable graph neural network training. In *Companion Proceedings of the ACM on Web Conference 2024*. 1502–1510.

[7] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter* 24, 2 (2022), 61–77.

[8] Mucong Ding, Xiaoyu Liu, Tahseen Rabbani, Teresa Ranadive, Tai-Ching Tuan, and Furong Huang. 2022. Faster Hyperparameter Search for GNNs via Calibrated Dataset Condensation. *arXiv* (2022).

[9] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. *arXiv preprint arXiv:2401.08281* (2024).

[10] Junfeng Fang, Xinglin Li, Yongduo Sui, Yuan Gao, Guibin Zhang, Kun Wang, Xiang Wang, and Xiangnan He. 2024. Exgc: Bridging efficiency and explainability in graph condensation. *arXiv preprint arXiv:2402.05962* (2024).

[11] Qizhang Feng, Zhimeng Jiang, Ruiquan Li, Yicheng Wang, Na Zou, Jiang Bian, and Xia Hu. 2023. Fair Graph Distillation. In *NeurIPS*.

[12] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2023. Scalable Attributed-Graph Subspace Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37.

[13] Jian Gao and Jianshe Wu. 2023. Multiple sparse graphs condensation. *Knowledge-Based Systems* 278 (2023), 110904.

[14] Xinyi Gao, Tong Chen, Yilong Zang, Wentao Zhang, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. 2024. Graph Condensation for Inductive Node Representation Learning. In *ICDE*.

[15] Xinyi Gao, Junliang Yu, Wei Jiang, Tong Chen, Wentao Zhang, and Hongzhi Yin. 2024. Graph condensation: A survey. *arXiv preprint arXiv:2401.11720* (2024).

[16] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations (ICLR)*.

[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] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 1024–1034.

[19] Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B Aditya Prakash, and Wei Jin. 2024. A Comprehensive Survey on Graph Reduction: Sparsification, Coarsening, and Condensation. *IJCAI* (2024).

[20] 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.

[21] 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.

[22] Bo Hui, Da Yan, Xiaolong Ma, and Wei-Shinn Ku. 2023. Rethinking Graph Lottery Tickets: Graph Sparsity Matters. *International Conference on Learning Representations (ICLR)* (2023).

[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. 2022. Graph Condensation for Graph Neural Networks. In *International Conference on Learning Representations*.

[25] Vassilis Kalofolias. 2016. How to learn a graph from smooth signals. In *Artificial intelligence and statistics*. PMLR, 920–929.

[26] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[27] Manoj Kumar, Anurag Sharma, Shashwat Saxena, and Sandeep Kumar. 2023. Featured graph coarsening with similarity guarantees. In *International Conference on Machine Learning*. PMLR, 17953–17975.

[28] Shiye Lei and Dacheng Tao. 2024. A Comprehensive Survey of Dataset Distillation. *TPAMI* (2024).

[29] Xinglin Li, Kun Wang, Hanhui Deng, Yuxuan Liang, and Di Wu. 2023. Attend who is weak: Enhancing graph condensation via cross-free adversarial training. *arXiv preprint arXiv:2311.15772* (2023).

[30] Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. 2022. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697* (2022).

[31] Yang Liu, Deyu Bo, and Chuan Shi. 2024. Graph Condensation via Eigenbasis Matching. *ICML* (2024).

[32] Yilun Liu, Ruihong Qiu, and Zi Huang. 2023. CaT: Balanced Continual Graph Learning with Graph Condensation. In *ICDM*.

[33] Yezi Liu and Yanning Shen. 2024. TinyGraph: Joint Feature and Node Condensation for Graph Neural Networks. *arXiv preprint arXiv:2407.08064* (2024).

[34] Zhanyu Liu, Chaolv Zeng, and Guanjie Zheng. 2024. Graph Data Condensation via Self-expressive Graph Structure Reconstruction. *SIGKDD* (2024).

[35] Andreas Loukas. 2019. Graph Reduction with Spectral and Cut Guarantees. *J. Mach. Learn. Res.* 116 (2019).

[36] Andreas Loukas and Pierre Vandergheynst. 2018. Spectrally Approximating Large Graphs with Smaller Graphs. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research)*.

[37] Runze Mao, Wenqi Fan, and Qing Li. 2023. Gcare: Mitigating subgroup unfairness in graph condensation through adversarial regularization. *Applied Sciences* 13, 16 (2023), 9166.

[38] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. 2021. Dataset Meta-Learning from Kernel Ridge-Regression. In *ICLR*.

[39] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[40] Arash Rasti-Meymandi, Ahmad Sajedi, Zhaopan Xu, and Konstantinos N Plataniotis. 2024. GSTAM: Efficient Graph Distillation with Structural Attention-Matching. *arXiv preprint arXiv:2408.16871* (2024).

[41] Si Si, Felix Yu, Ankit Singh Rawat, Cho-Jui Hsieh, and Sanjiv Kumar. 2022. Serving Graph Compression for Graph Neural Networks. In *The Eleventh International Conference on Learning Representations*.

[42] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2019).

[43] Xiangguo Sun, Hong Cheng, Bo Liu, Jia Li, Hongyang Chen, Guandong Xu, and Hongzhi Yin. 2023. Self-supervised hypergraph representation learning for sociological analysis. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[44] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*.

[45] Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. 2024. Fast graph condensation with structure-based neural tangent kernel. *WWW* (2024).

[46] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. 2018. Dataset distillation. *ArXiv preprint* (2018).

[47] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and S Yu Philip. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data* 9, 2 (2022), 415–436.

[48] Yuxiang Wang, Xiao Yan, Shiyu Jin, Hao Huang, Quanqing Xu, Qingchen Zhang, Bo Du, and Jiawei Jiang. 2024. Self-Supervised Learning for Graph Dataset Condensation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3289–3298.

[49] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. 6861–6871.

[50] Jiahao Wu, Ning Lu, Zeiyu Dai, Wenqi Fan, Shengcai Liu, Qing Li, and Ke Tang. 2024. Backdoor Graph Condensation. *arXiv preprint arXiv:2407.11025* (2024).

[51] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* (2020).

[52] Zhenbang Xiao, Shunyu Liu, Yu Wang, Tongya Zheng, and Mingli Song. 2024. Disentangled condensation for large-scale graphs. *arXiv preprint arXiv:2401.12231* (2024).

[53] Zhenbang Xiao, Yu Wang, Shunyu Liu, Huiqiong Wang, Mingli Song, and Tongya Zheng. 2024. Simple Graph Condensation. *Machine Learning and Knowledge*

Discovery in Databases (ECML PKDD) (2024).
[54] Hongjia Xu, Liangliang Zhang, Yao Ma, Sheng Zhou, Zhuonan Zheng, and Bu Jiajun. 2024. A Survey on Graph Condensation. *arXiv preprint arXiv:2402.02000* (2024).
[55] 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.
[56] 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?. In *NeurIPS*.
[57] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2023. Self-supervised learning for recommender systems: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).
[58] Ruonan Yu, Songhua Liu, and Xinchao Wang. 2023. Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).
[59] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
[60] Tianle Zhang, Yuchen Zhang, Kun Wang, Kai Wang, Beining Yang, Kaipeng Zhang, Wenqi Shao, Ping Liu, Joey Tianyi Zhou, and Yang You. 2024. Two Trades is not Baffled: Condense Graph via Crafting Rational Gradient Matching. *arXiv preprint arXiv:2402.04924* (2024).
[61] Wentao Zhang, Mingyu Yang, Zeang Sheng, Yang Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2021. Node Dependent Local Smoothing for Scalable Graph Learning. *Advances in Neural Information Processing Systems* 34 (2021).
[62] Yuchen Zhang, Tianle Zhang, Kai Wang, Ziyao Guo, Yuxuan Liang, Xavier Bresson, Wei Jin, and Yang You. 2024. Navigating Complexity: Toward Lossless Graph Condensation via Expanding Window Matching. *ICML* (2024).
[63] 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.
[64] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2021. Dataset Condensation with Gradient Matching. In *ICLR*.
[65] Jianan Zhao, Hesham Mostafa, Michael Galkin, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2024. GraphAny: A Foundation Model for Node Classification on Any Graph. *arXiv preprint arXiv:2405.20445* (2024).
[66] 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. In *NeurIPS*.
[67] Yizhen Zheng, Shirui Pan, Vincent Lee, Yu Zheng, and Philip S Yu. 2022. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. *Advances in Neural Information Processing Systems* 35 (2022), 10809–10820.
[68] Hao Zhu and Piotr Koniusz. 2020. Simple spectral graph convolution. In *International Conference on Learning Representations*.
[69] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2021. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036* 14 (2021), 1–1.

Appendix

Contents

A Proof of Propositions 10
A.1 Proof of Proposition 1 10
A.2 Proof of Proposition 2 11
A.3 Proof of Proposition 3 11
A.4 Proof of Proposition 4 11
B Algorithm 11
C Time Complexity Analysis 12
D Experimental Setup Details 12
D.1 Dataset Statistics 12
D.2 Baselines 12
D.3 Implementations 12
E Generalizability Comparison of Graphless Methods 13
F Related Work 13

A Proof of Propositions

A.1 Proof of Proposition 1

Proposition 1. *The performance matching objective is equivalent to the optimal parameter matching objective.*

The proof of Proposition 1 builds on the proposition from the dataset distillation survey in CV [58]. To ensure self-containment, we detail and expand propositions within the context of graph theory.

Proof. Performance matching introduce KRR in optimization and objectives for $\mathcal{T}$ and $\mathcal{S}$ are formulated as:

$$\begin{aligned} &\arg\min_{\Theta} \|\mathbf{Z}\Theta - \mathbf{Y}\|^2 + \lambda \|\Theta\|^2, \\ &\arg\min_{\Theta} \left\|\mathbf{Z}'\Theta - \mathbf{Y}'\right\|^2 + \lambda \|\Theta\|^2, \end{aligned} \tag{18}$$

where $\mathbf{Z}$ and $\mathbf{Z}'$ are node embeddings for the original graph and the condensed graph, respectively. Θ is the learnable weight matrix and λ is a small constant weight of the regularization term for numerical stability. Their closed-form optimal solutions are:

$$\begin{aligned} \Theta^* &= \mathbf{Z}^\top \left(\mathbf{Z}\mathbf{Z}^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{Y}, \\ \Theta'^* &= \mathbf{Z}'^\top \left(\mathbf{Z}'\mathbf{Z}'^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{Y}'. \end{aligned} \tag{19}$$

We simplifies the regularization term in KRR model and assume λ=0. Then, the optimal parameter matching objective, expressed in the form of a least squares function, is formulated as:

$$\begin{aligned} \mathcal{L}_{PM} &= \arg\min_{\mathbf{Z}'} \left\|\Theta^* - \Theta'^*\right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \left\|\mathbf{Z}^\top \left(\mathbf{Z}\mathbf{Z}^\top\right)^{-1} \mathbf{Y} - \mathbf{Z}'^\top \left(\mathbf{Z}'\mathbf{Z}'^\top\right)^{-1} \mathbf{Y}'\right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \left\|\left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1} \mathbf{Z}^\top\mathbf{Y} - \mathbf{Z}'^\top \left(\mathbf{Z}'\mathbf{Z}'^\top\right)^{-1} \mathbf{Y}'\right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \left\|\left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1} \mathbf{Z}^\top\mathbf{Y} - \left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1} \mathbf{Z}^\top\mathbf{Z}\mathbf{Z}'^\top \left(\mathbf{Z}'\mathbf{Z}'^\top\right)^{-1} \mathbf{Y}'\right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \left\|\left(\left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1} \mathbf{Z}^\top\right) \left(\mathbf{Y} - \mathbf{Z}\mathbf{Z}'^\top \left(\mathbf{Z}'\mathbf{Z}'^\top\right)^{-1} \mathbf{Y}'\right)\right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \left\|\mathbf{Y} - \mathbf{Z}\mathbf{Z}'^\top \left(\mathbf{Z}'\mathbf{Z}'^\top\right)^{-1} \mathbf{Y}'\right\|^2. \end{aligned} \tag{20}$$

The final step is justified by the independence of $(\mathbf{Z}^\top\mathbf{Z})^{-1}\mathbf{Z}^\top$ from $\mathbf{Z}'$. Then, the r.h.s of Eq. (20) is the performance matching objective. □

A.2 Proof of Proposition 2

Proposition 2. *The distribution matching objective represents a simplified formulation of the performance matching, omitting feature correlation considerations.*

Proof. According to Proposition 1, the objective of performance matching can be formulated as:

$$\begin{aligned}\mathcal{L}_{PM} &= \arg\min_{\mathbf{Z}'} \left\| (\mathbf{Z}^\top\mathbf{Z})^{-1}\mathbf{Z}^\top\mathbf{Y} - (\mathbf{Z}'^\top\mathbf{Z}')^{-1}\mathbf{Z}'^\top\mathbf{Y}' \right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \sum_{i=1}^{c} \left\| (\mathbf{Z}_i^\top\mathbf{Z}_i)^{-1}\mathbf{Z}_i^\top\mathbf{Y}_i - (\mathbf{Z}_i'^\top\mathbf{Z}_i')^{-1}\mathbf{Z}_i'^\top\mathbf{Y}_i' \right\|^2,\end{aligned} \tag{21}$$

where subscript i denotes the class, with $\mathbf{Z}_i$ and $\mathbf{Z}'_i$ representing original and condensed node embeddings in class i. $\mathbf{Y}_i$ and $\mathbf{Y}'_i$ representing one-hot labels for nodes in class i. The distribution matching objective is calculated for class prototypes separately and formulated as:

$$\begin{aligned}\mathcal{L}_{DM} &= \arg\min_{\mathbf{Z}'} \left\| \mathbf{P}'\mathbf{Z}' - \mathbf{P}\mathbf{Z} \right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\mathbf{Y}_i^\top\mathbf{Z}_i - \frac{1}{|C'_i|}\mathbf{Y}_i'^\top\mathbf{Z}_i' \right\|^2, \\ &= \arg\min_{\mathbf{Z}'} \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\mathbf{Z}_i^\top\mathbf{Y}_i - \frac{1}{|C'_i|}\mathbf{Z}_i'^\top\mathbf{Y}_i' \right\|^2,\end{aligned} \tag{22}$$

where $|C'_i|$ and $|C_i|$ represent the sizes of class i in $\mathcal{S}$ and $\mathcal{T}$, respectively. By comparing Eq. (21) and Eq. (22), it can be deduced that the performance matching objective can be transformed into distribution matching by excluding feature correlations and incorporating class size normalization.

□

A.3 Proof of Proposition 3

Proposition 3. *The distribution matching objective with a feature correlation constraint provides an upper bound for the parameter matching objective.*

The proof of Proposition 3 is derived by extending the proposition in [58].

Proof. Without loss of generality, we take the representative gradient matching objective utilized in GCond for illustration, where SGC is utilized as the relay model. Therefore, the node embeddings are represented by:

$$\mathbf{Z}' = \mathbf{H}'\mathbf{\Theta}, \qquad \mathbf{Z} = \mathbf{H}\mathbf{\Theta}, \tag{23}$$

where $\mathbf{\Theta}$ is the learnable weight matrix to transform propagated features $\mathbf{H}'$ and $\mathbf{H}$. Due to that the objective of gradient matching is calculated for each class separately, we define the losses for $\mathcal{T}$ and $\mathcal{S}$ as:

$$\begin{aligned}\mathcal{L}_i^{\mathcal{T}} &= \frac{1}{2}\left\| \mathbf{H}_i\mathbf{\Theta} - \mathbf{Y}_i \right\|^2, \\ \mathcal{L}_i^{\mathcal{S}} &= \frac{1}{2}\left\| \mathbf{H}'_i\mathbf{\Theta} - \mathbf{Y}'_i \right\|^2,\end{aligned} \tag{24}$$

where $\mathbf{H}_i$ and $\mathbf{H}'_i$ represent original and condensed propagated features in class i. $\mathbf{Y}_i$ and $\mathbf{Y}'_i$ represent one-hot labels for nodes in class i. For simplicity, we specify the distance function $\mathcal{D}$ in the gradient matching objective as the normalized least squares function. Consequently, the objective function of gradient matching is formulated as

$$\begin{aligned}\mathcal{L}_{GM} &= \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\nabla_{\mathbf{\Theta}}\mathcal{L}_i^{\mathcal{T}} - \frac{1}{|C'_i|}\nabla_{\mathbf{\Theta}}\mathcal{L}_i^{\mathcal{S}} \right\|^2, \\ &= \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}(\mathbf{H}_i^\top\mathbf{H}_i\mathbf{\Theta} - \mathbf{H}_i^\top\mathbf{Y}_i) - \frac{1}{|C'_i|}(\mathbf{H}_i'^\top\mathbf{H}'_i\mathbf{\Theta} - \mathbf{H}_i'^\top\mathbf{Y}'_i) \right\|^2, \\ &= \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\mathbf{H}_i^\top\mathbf{H}_i\mathbf{\Theta} - \frac{1}{|C'_i|}\mathbf{H}_i'^\top\mathbf{H}'_i\mathbf{\Theta} - \frac{1}{|C_i|}\mathbf{H}_i^\top\mathbf{Y}_i + \frac{1}{|C'_i|}\mathbf{H}_i'^\top\mathbf{Y}'_i \right\|^2, \\ &\le \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\mathbf{H}_i^\top\mathbf{Y}_i - \frac{1}{|C'_i|}\mathbf{H}_i'^\top\mathbf{Y}'_i \right\|^2 + \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\mathbf{H}_i^\top\mathbf{H}_i - \frac{1}{|C'_i|}\mathbf{H}_i'^\top\mathbf{H}'_i \right\|^2 \|\mathbf{\Theta}\|^2, \\ &= \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\mathbf{Y}_i^\top\mathbf{H}_i - \frac{1}{|C'_i|}\mathbf{Y}'^\top_i\mathbf{H}'_i \right\|^2 + \sum_{i=1}^{c} \left\| \frac{1}{|C_i|}\mathbf{H}_i^\top\mathbf{H}_i - \frac{1}{|C'_i|}\mathbf{H}_i'^\top\mathbf{H}'_i \right\|^2 \|\mathbf{\Theta}\|^2.\end{aligned} \tag{25}$$

The first term in r.h.s is the distribution matching objective and the second term quantifies differences in class-wise feature correlations. □

A.4 Proof of Proposition 4

Proposition 4. *Assume an undirected condensed graph $\mathcal{S} = \{\mathbf{A}', \mathbf{X}'\}$, the closed-form solution of Eq. (17) take the form: $\mathbf{X}' = (\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}')^{-1}\mathbf{Q}^\top\mathbf{H}'$, where $\mathbf{Q} = \hat{\mathbf{A}}'^K$.*

The proof of Proposition 4 is derived by extending the proposition in [27].

Proof. Given an undirected graph $\mathcal{S} = \{\mathbf{X}', \mathbf{A}'\}$, $\mathbf{L}' \in \mathbb{R}^{N'\times N'}$ is defined as the Laplacian matrix of $\mathcal{S}$ by $\mathbf{L}' = \mathbf{D}' - \mathbf{A}'$, where $\mathbf{D}'$ is the degree matrix. To solve the optimization problem in Eq. (17), we first calculate $\nabla\mathcal{L}(\mathbf{X}')$ and $\nabla^2\mathcal{L}(\mathbf{X}')$ as:

$$\begin{aligned}\nabla\mathcal{L}(\mathbf{X}') &= 2\mathbf{Q}^\top(\mathbf{Q}\mathbf{X}' - \mathbf{H}') + \alpha(\mathbf{L}' + \mathbf{L}'^\top)\mathbf{X}' \\ &= 2(\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}')\mathbf{X}' - 2\mathbf{Q}^\top\mathbf{H}'.\end{aligned} \tag{26}$$

$$\nabla^2\mathcal{L}(\mathbf{X}') = 2(\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}'). \tag{27}$$

According to the definition of $\mathbf{A}'$ in Eq. (16), $\mathbf{L}'$ and $\mathbf{Q}$ are the positive semi-definite matrices. Therefore, $\nabla^2\mathcal{L}(\mathbf{X}')$ is the positive semi-definite matrix and the optimization problem is a convex optimization problem. We can get the closed-form solution by calculate $\nabla\mathcal{L}(\mathbf{X}') = 0$ and $\mathbf{X}' = (\mathbf{Q}^\top\mathbf{Q} + \alpha\mathbf{L}')^{-1}\mathbf{Q}^\top\mathbf{H}'$.

□

B Algorithm

The detailed algorithm of CGC and CGC-X is shown in Algorithm 1. In detail, we first propagate node features according to the graph structure via non-parametric propagation methods. Consequently, a linear model is generated to assess embeddings and augmented features are sampled according to evaluation results. Then, embeddings in each class are partitioned by clustering method (i.e., Clustering$(\cdot)$ in line 8) to generate aggregated embeddings $\mathbf{H}'$. $\mathbf{H}'$ can serve as the condensed node feature for CGC-X. If condensed graph structure is required, $\mathbf{A}'$ and $\mathbf{X}'$ can be generated by Eq. (16) and Eq. (17), respectively.

Algorithm 1: The framework of CGC and CGC-X.

Input: Original graph $\mathcal{T} = \{\mathbf{A}, \mathbf{X}\}$, $\mathbf{Y}$ and pre-defined condensed graph label $\mathbf{Y}'$
Output: Condensed graph $\mathcal{S}$

`/* Feature Propagation */`
Generate propagated features $\mathbf{H}^{(l)}$ according to Eq. (12).
`/* Data Assessment */`
Generate linear model and evaluate generated features.
`/* Data Augmentation */`
Sample features according to class prediction errors.
`/* Class Partition */`
Categorise $\mathbf{H}_{cond}$ into c classes $\{\hat{\mathbf{H}}_1, \cdots, \hat{\mathbf{H}}_c\}$ according to $\mathbf{y}_{cond}$.
for $i = 1, \ldots, c$ **do**
    Compute $\mathbf{H}'_i = \text{Clustering}(\hat{\mathbf{H}}_i)$.
Generate $\mathbf{H}' = [\mathbf{H}'_1; \ldots; \mathbf{H}'_c]$.
`/* Graph Generation */`
if CGC-X **then**
    Return: $\mathcal{S} = \{\mathbf{I}, \mathbf{H}'\}$
else if CGC **then**
    Generate $\mathbf{A}'$ according to Eq. (16).
    Generate $\mathbf{X}'$ by solving Eq. (17).
    Return: $\mathcal{S} = \{\mathbf{A}', \mathbf{X}'\}$

C Time Complexity Analysis

The pipeline of CGC consists of five components: feature propagation, data assessment, data augmentation, class partition and graph generalization. We analyse the time complexity for each component separately. (1) Feature propagation contains the K step graph convolution and time complexity is: $O(KEd)$, where d is the node dimension and E is the number of edges. (2) The solution of linear model involves the pseudo inverse of propagated features. In practice, this least squares problem in Eq. (13) can be effectively solved by torch.linalg.lstsq and the time complexity is $O(Nd^2)$, where N denotes the number of nodes. In addition, the time complexity of node embedding prediction is $O(dKNc)$, where c represents the number of classes. (3) By leveraging propagated nodes with smaller depth as augmentations, the data augmentation module requires no additional computations. (4) Class partition executes the EM-based clustering on each class. Take K-means as an example, the time complexity for class i is: $O\left(\left|C'_i\right| |C_i| \, dt\right)$, where t indicates the number of iteration for K-means. $\left|C'_i\right|$ and $|C_i|$ represent the size of class i in $\mathcal{S}$ and $\mathcal{T}$ respectively. (5) Graph generation contains the adjacency matrix generation and feature calculation. The former requires the pair-wise similarity calculation whose complexity is $O\left(N'^2 d\right)$. The closed-form solution of the node feature requires the inverse operation and the time complexity is $O\left(N'^3 + N'^2 d + (K+1)N'E' + dE'\right)$, where E' is the edge number of condensed graph.

In summary, the time complexity of CGC is dominated by class partition and node feature generation. However, due to the small size of the condensed graph and well-established clustering acceleration methods (e.g., FAISS [9]), CGC can be efficiently executed.

D Experimental Setup Details

D.1 Dataset Statistics

We evaluate our proposed methods on four transductive datasets, i.e., Cora, Citeseer [26], Ogbn-arxiv (Arxiv) [20] and Ogbn-products (Products) [20], as well as two inductive datasets, i.e., Flickr and Reddit [59]. We adopt public splits throughout the experiments and dataset statistics are shown in Table 8.

D.2 Baselines

We compare our proposed methods against 12 baselines, including graph coarsening method and GC methods with diverse optimization strategies:
(1) Variation Neighborhoods (VN) [21, 35]. The conventional graph coarsening method that leverages the partition matrix to construct the super-nodes and super-edges.
(2) A-ConvMatch (A-CM) [6]. A graph coarsening method hierarchically reduces the graph size while preserving the output of graph convolutions.
(3) GCond and GCond-X [24]. The first GC method that utilizes the gradient matching to align the model parameters derived from both graphs.
(4) GCDM and GCDM-X [30]. An efficient GC method that generates condensed graphs based on distribution matching by optimizing the maximum mean discrepancy between class prototypes.
(5) GC-SNTK and GC-SNTK-X [45]. An efficient GC method leverages kernel ridge regression with a structure-based neural tangent kernel to simplify the bi-level optimization process.
(6) SFGC [66]. A graphless GC method that aligns long-term model learning behaviors through trajectory matching.
(7) SimGC [53]. An efficient GC method with the graph generation that introduces the pre-trained model in distribution matching.
(8) GEOM [62]. A graphless GC method that explores the difficult samples via trajectory matching.
(9) GCSR [34]. A trajectory matching-based GC method with the self-expressive condensed graph structure.

D.3 Implementations

Condensation Ratios. We set r following GCond. For transductive datasets, r is chosen as the {25%, 50%, 100%}, {25%, 50%, 100%}, 0.1%, 0.5%, 1%} and {0.32%, 0.63%, 1.26%} of the labeled nodes for Cora, Citeseer, Arxiv and Products, respectively. For Inductive datasets, r is set as {0.1%, 0.5%, 0.1%} and {0.05%, 0.1%, 0.2%} for Flickr and Reddit.
Hyper-parameters. The hyper-parameters are determined through the grid search on the validation set. The feature proportion depth of the original graph and the condensed graph is set as $K = 2$ for all datasets. The temperature τ is optimized from the set {10, 1, 0.8, 0.5, 0.3, 0.1, 0.01} and the augmentation ratio p is searched within the range [0, 100]. For the two hyper-parameters specific to CGC, the weight α is tuned from the set {0.3, 0.5, 1, 2, 3}, and the threshold T is explored within the range [0.8, 1). We use the ADAM optimization algorithm to train all the models. The learning rate for the condensation process is determined through a search over

Table 8: The statistics of datasets used in experiments.

Dataset	#Nodes	#Edges	#Classes	#Features	Training/Validation/Test	Task type
Cora	2,708	5,429	7	1,433	140/500/1,000	Transductive
Citeseer	3,327	4,732	6	3,703	120/500/1,000	Transductive
Ogbn-arxiv	169,343	1,166,243	40	128	90,941/29,799/48,603	Transductive
Ogbn-products	2,449,029	61,859,140	47	100	196,615/39,323/2,213,091	Transductive
Flickr	89,250	899,756	7	500	44,625/22,312/22,313	Inductive
Reddit	232,965	23,213,838	41	602	153,932/23,699/55,334	Inductive

Table 9: The generalizability comparison of graphless GC methods. SNTK-X is out-of-memory on Reddit dataset. AVG indicates the average value. The best performances are highlighted.

Dataset (r)	Method	SGC	GCN	SAGE	APPNP	Cheby	AVG
Arxiv (0.25%)	GCond-X	64.7	64.2	64.4	61.5	59.5	62.9
	GCDM-X	64.4	61.2	63.4	60.5	60.2	61.9
	SNTK-X	64.0	65.5	62.4	61.8	58.7	62.5
	SFGC	64.8	66.1	64.8	**63.9**	60.7	64.1
	GEOM	65.0	67.5	64.9	62.5	60.9	64.2
	CGC-X	**66.3**	**66.8**	**65.9**	63.0	**61.3**	**64.7**
Flickr (0.50%)	GCond-X	44.4	45.0	44.7	44.6	42.3	44.2
	GCDM-X	45.8	45.6	42.6	42.4	42.4	43.8
	SNTK-X	44.0	46.7	41.9	41.0	41.4	43.0
	SFGC	47.0	**47.0**	42.5	40.7	45.4	44.5
	GEOM	46.5	46.2	42.7	42.6	46.1	44.8
	CGC-X	**47.2**	**47.0**	**46.6**	**46.8**	**46.4**	**46.8**
Reddit (0.10%)	GCond-X	91.0	89.3	89.3	78.7	74.0	84.5
	GCDM-X	90.9	87.2	89.2	79.1	75.1	84.3
	SFGC	89.5	90.0	90.3	88.3	82.8	88.2
	GEOM	89.6	90.4	90.5	89.4	82.6	88.5
	CGC-X	**91.5**	**90.8**	**90.9**	**89.0**	**82.6**	**89.0**

the set {0.01, 0.001, 0.0001}. The weight decay is 5e-4. Dropout is searched from [0, 1).

Computing Infrastructure. The codes are written in Python 3.9 and Pytorch 1.12.1. CGC and CGC-X are executed on CPUs and all GNN models are trained on GPUs. The experiments for Ogbn-products were conducted on a server equipped with Intel(R) Xeon(R) Gold 6326 CPUs at 2.90GHz and NVIDIA GeForce A40 GPUs with 48GB of memory. Other experiments were carried out on a server featuring Intel(R) Xeon(R) Gold 6128 CPUs at 3.40GHz and NVIDIA GeForce RTX 2080 Ti GPUs with 11GB of memory.

E Generalizability Comparison of Graphless Methods

We compare the GNN architecture generalizability of graphless GC methods, where $\{\mathbf{I}, \mathbf{X}'\}$ is used in GNN training. Due to the absence of condensed graph structure, we evaluate GNNs including GCN, SGC, SAGE, APPNP and Cheby. According to the results in Table 9, we could observe that our proposed CGC-X achieves the best generalizability across different datasets and architectures. Especially on the dataset Flickr, where the average improvements are 2.3%. This verifies the high quality of the condensed feature and the effectiveness of the class-to-node distribution matching strategy.

F Related Work

Graph Condensation. GC [15, 19, 48, 54] has recently garnered significant attention as a data-centric approach. Beyond the diverse optimization strategies discussed in Section 2, other research efforts have focused on enhancing these optimization strategies [29, 33, 40, 52, 60, 62] or developing superior condensed graph structures [13, 34] to improve condensed graph quality. Additionally, there is an increasing emphasis on enhancing other aspects of models trained on condensed graphs, such as generalizability [56], fairness [11, 37], security [50] and explainability [10].

Graph Sparsification & Coarsening. Early explorations into graph size reduction primarily focus on graph sparsification [4, 22] and coarsening techniques [2]. These methods aim to eliminate redundant edges and merge similar nodes while preserving essential graph characteristics, such as the largest principal eigenvalues [36] and Laplacian pseudoinverse [1]. Recently, FGC [27] incorporates node features into the coarsening process to improve node merging. However, these methods overlook node label information, resulting in sub-optimal generalization across different GNN architectures and downstream tasks. Additionally, VNG [41] generates compressed graphs by directly minimizing propagation errors. However, the compressed graph's utility is restricted to serving time and cannot be used for GNN training, limiting its applicability.