# GC4NC: A BENCHMARK FRAMEWORK FOR GRAPH CONDENSATION ON NODE CLASSIFICATION WITH NEW INSIGHTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graph condensation (GC) is an emerging technique designed to learn a significantly smaller graph that retains the essential information of the original graph. This condensed graph has shown promise in accelerating graph neural networks while preserving performance comparable to those achieved with the original, larger graphs. Additionally, this technique facilitates downstream applications like neural architecture search and deepens our understanding of redundancies in large graphs. Despite the rapid development of GC methods, particularly for node classification, a unified evaluation framework is still lacking to systematically compare different GC methods or clarify key design choices for improving their effectiveness. To bridge these gaps, we introduce **GC4NC**, a comprehensive framework for evaluating diverse GC methods on node classification across multiple dimensions including performance, efficiency, privacy preservation, denoising ability, NAS effectiveness, and transferability. Our systematic evaluation offers novel insights into how condensed graphs behave and the critical design choices that drive their success. These findings pave the way for future advancements in GC methods, enhancing both performance and expanding their real-world applications. The code is available at https://anonymous.4open.science/r/GC4NC-1620/.

## 1 INTRODUCTION

Graphs are ubiquitous data structures describing relations of entities and have found applications in various domains such as chemistry (Reiser et al., 2022; Guo et al., 2023), bioinformatics (Wang et al., 2021), epidemiology (Liu et al., 2024a), e-commerce (Wang et al., 2023a; Ding et al., 2023) and so on. To harness the wealth of information in graphs, graph neural networks (GNN) have emerged as powerful tools for exploiting structural information to handle diverse graph-related tasks (Kipf and Welling, 2016; Veličković et al., 2018; Wu et al., 2019a; Wang et al., 2023a; Zhou et al., 2021). However, the proliferation of large-scale graph datasets in practical applications introduce significant computational difficulties for GNN utilization (Hamilton et al., 2017; Jin et al., 2022a; Zhang et al., 2023). These large datasets complicate GNN training, as time complexity escalates with the increase of nodes and edges. Furthermore, the extensive sizes of these graphs also strain GPU memory, disk storage, and network communication bandwidth (Zhang et al., 2023).

Inspired by dataset distillation (or dataset condensation) (Wang et al., 2018; Yu et al., 2023; Cui et al., 2022) in the image domain, graph condensation (GC) (Jin et al., 2022a; Hashemi et al., 2024; Gao et al., 2024; Xu et al., 2024) has been proposed to learn a significantly smaller (e.g., 1,000× smaller number of nodes) graph that retains essential information of the original large graph. This condensed graph is expected to train downstream GNNs in a highly efficient manner with minimal performance degradation. As a data-centric technique, GC is considered to be orthogonal to existing model-centric efforts on GNN acceleration (Wu et al., 2019b; Frasca et al., 2020), since using condensed graph datasets as input can further speed up existing models. Remarkably, GC not only excels at compressing graph data but also shows promise for various other applications, such as federated learning (Pan et al., 2023) and neural architecture search (NAS) (Ding et al., 2022).

Despite the rapid advancements in this field, the lack of a unified and comprehensive evaluation protocol for GC significantly hinders progress in evaluating, understanding and improving these methods. ***First***, existing GC methods adopt different approaches to select the best condensed graphs, including variations in validation models, reliance on test set results rather than validation ones, and conducting overly frequent intermediate validations, which could introduce unfairness in evaluation. *Second*, while most GC methods are evaluated primarily on performance and transferability, they often neglect critical aspects such as the effectiveness of NAS. Furthermore, intuitive benefits of GC like privacy preservation and denoising ability are frequently mentioned but remain under-explored (Sachdeva and McAuley, 2023; Hashemi et al., 2024). *Third*, the impact of design choices during the condensation process including the condensation objectives, how condensed graphs are initialized, whether to generate a condensed graph structure, and which graph properties to preserve, are still poorly understood. By systematically addressing these limitations, we aim to shed light on the successes and pitfalls in current GC research and guide future directions in this evolving area. Given that most GC methods are developed for node classification (NC), we will focus on this task and propose a new benchmark framework, GC4NC, with the following contributions:

- **A Fair Evaluation Protocol.** We establish a graph condensation benchmark by introducing a fair and consistent evaluation protocol that facilitates comparison across methods. This unified evaluation approach properly utilizes validation data to select the most effective condensed graphs. In addition, we provide an open-source, well-structured, and user-friendly codebase specifically designed to facilitate easy integration and evaluation of different GC approaches.
- **Comprehensive Comparison through Multiple Dimensions.** Using the fair evaluation protocol, we conduct comprehensive comparisons of various GC methods across multiple dimensions including (a) performance and scalability, (b) privacy preservation, (c) denoising ability, (d) NAS effectiveness, and (e) transferability. To our knowledge, we are the first to systematically benchmark privacy preservation and denoising ability across various GC methods.
- **In-Depth Analysis of Design Choices.** We further conduct a thorough analysis of how key design choices impact condensation performance, including data initialization, structure-free vs. structure-based methods, and graph property preservation. Our results provide valuable guidance for optimizing and exploring these critical choices in future research.
- **Novel Insights.** Through a comprehensive comparison of these methods, our experimental results provide key insights into the behavior of graph condensation such as:
  (a) Among varied condensation objectives, methods based on trajectory matching generally deliver the best condensation performance but fall short in efficiency. Furthermore, graph condensation achieves better performance than image dataset condensation at the same reduction rates, but it struggles to scale to larger reduction rates.
  (b) Certain GC methods can **preserve privacy** by reducing the success of membership inference attacks while still maintaining high condensation performance.
  (c) GC methods exhibit **a certain level of denoising ability** against structural noise (both adversarial and random noise), yet they are less effective against node feature noise.
  (d) Trajectory matching or inner optimization through gradient matching is essential for reliable NAS performance and enhanced transferability.
  (e) Compared to *structure-based* methods, *structure-free* methods exhibit strong condensation performance and favorable efficiency but poorer denoising ability.

Note that two concurrent works (Liu et al., 2024b; Sun et al., 2024) on GC benchmarks have emerged alongside this paper. While all studies contribute uniquely to the field of graph condensation, **GC4NC** stands out by offering deeper insights. First, it covers a wider range of GC methods for NC. Second, it pioneers the exploration of GC methods in terms of privacy preservation and denoising ability. Third, it provides a more in-depth analysis of graph property preservation to enhance the understanding of GC methods. For further details, please refer to the Appendix A.1.

## 2 Related Work

### 2.1 Graph Condensation

Graph condensation (GC) is an emerging technique designed to create a significantly smaller graph that preserves the maximum amount of information from the original graph (Jin et al., 2022a; Hashemi et al., 2024; Jin et al., 2022b; Zhang et al., 2024a; Gao and Wu, 2023; Yang et al., 2024). The goal is to ensure that GNNs trained on this condensed graph exhibit comparable performance to those trained on the original one. Based on their specific condensation objectives, existing GC methods employ the following matching strategies to bridge the gap between condensed and real graphs:

**Gradient Matching (GM). GCond** (Jin et al., 2022a) matches the gradients of the original graph $\mathcal{T}$

and condensed graphs $\mathcal{S}$ by: $\min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_{t=0}^{T-1} D\left(\nabla_\theta \mathcal{L}_{\mathcal{T}}, \nabla_\theta \mathcal{L}_{\mathcal{S}}\right)\right]$, where $D(\cdot,\cdot)$ denotes a distance function. During this process, it also updates $\theta$ by training the GNN for several epochs on the condensed graph $\mathcal{S}$, referred to as **inner optimization**. However, this nested optimization significantly hinders efficiency and scalability. To address this, **DosCond** (Jin et al., 2022b) only matches the gradients of the first epoch. To avoid generating dense graphs while producing diverse structures, **MSGC** (Gao and Wu, 2023) utilizes multiple sparse graphs to enhance the capture of neighborhood information. To explicitly incorporate the information of original structure, **SGDD** (Yang et al., 2024) broadcasts the original structure into the synthetic graph by optimal transport.

**Trajectory Matching (TM).** Inspired by (Cazenavette et al., 2022) in image domain, **SFGC** (Zheng et al., 2024) learns node features by matching the GNN training trajectories with the guidance of the offline expert parameter distribution: $\min_{\mathcal{S}} \mathcal{L} = \| \hat{\theta}_{t+N} - \theta^*_{t+M} \|_2^2$, where $\hat{\theta}$ is the student parameters optimized on condensed graph and $\theta^*$ is the expert parameters. **GEOM** (Zhang et al., 2024a) utilizes an expanding window technique that adjusts the matching range for nodes of varying difficulty during the process of matching training trajectories.

**Others.** *Distribution Matching (DM)*, originally developed for the image domain (Liu et al., 2023a), has been adapted to the graph domain as **GCDM** (Liu et al., 2022). They match the distances between the average embedding outputs of each graph convolution layer in the condensed graph and those in the original graph. We adopt its structure-free variant, **GCDMX**, in our experiments as it performs better in the original paper. To address the issue of higher computational consumption in the inner optimization of GM, **GCSNTK** (Wang et al., 2023b) replaces it with Graph Neural Tangent Kernel (GNTK) (Du et al., 2019) in the Kernel Ridge Regression (KRR) paradigm, which can efficiently synthesize a smaller graph: $\mathcal{L}_{\mathrm{KRR}} = \frac{1}{2} \| \mathbf{y}_{\mathcal{T}} - \mathbf{K}_{\mathcal{T}\mathcal{S}} \left(\mathbf{K}_{\mathcal{S}\mathcal{S}} + \epsilon \mathrm{I}\right)^{-1} \mathbf{y}_{\mathcal{S}} \|^2$, where $\mathbf{K}$ is the kernel matrix and $\mathbf{y}$ is concatenated graph labels. This method is called *meta-model matching (MM)* in Sachdeva and McAuley (2023). **GDEM** (Liu et al., 2023b) employs the *eigenbasis matching (EM)* which is derived from GM but avoids the biases inherent in condensation models. All methods except GDEM are presented in main experiments, while GDEM's results are included in Appendix A.4.

### 2.2 Coreset Selection and Graph Coarsening

We emphasize the necessity of exploring a broader spectrum of graph reduction methods beyond GC. **First**, recent years have seen the development of many coreset selection (Ding et al., 2024) and coarsening methods (Cao et al., 2024), which show high potential in preserving GNN performance. Thus, these methods are indispensable baselines for comparison with GC methods. **Second**, these methods can all serve as data initialization strategies for GC as we will explore in Section 4.7. Thus, it can be limited to study GC in isolation without considering other graph reduction methods.

**Coreset.** Coreset selection (Har-Peled and Kushal, 2005) identifies the most representative samples based on specific criteria. In graph domain, it typically selects nodes or edges and then utilizes selected nodes or edges to induce a small graph. We choose the following coreset methods as our baselines: **Random**, which randomly selects nodes. **KCenter** (Har-Peled and Kushal, 2005; Sener and Savarese, 2017) selects nodes in a way that minimizes the maximum distance of any node's embedding to the nearest chosen center, thereby effectively covering the feature space. **Herding** (Welling, 2009) selects nodes by iteratively minimizing the difference between the mean embedding and the sum of the embeddings of the selected nodes. More selection methods are explored in Appendix A.4.

**Graph Coarsening.** To preserve all node information, graph coarsening methods group nodes and aggregate them to supernodes. The following graph coarsening methods are chosen as baselines — **Averaging**, a data initialization strategy in MSGC (Gao and Wu, 2023), creates supernodes by averaging the features of training set nodes within each class. **Virtual Node Graph (VNG)** (Si et al., 2022) minimizes the forward propagation error by applying weighted k-means to obtain a mapping matrix, which maps each node to a supernode. VNG obtains the adjacency matrix by solving an optimization problem. **Variation Neighbors (VN)** (Loukas, 2019; Huang et al., 2021) is a classic coarsening method which contracts nodes that share the most similar neighborhoods. *We do not put its performance in main content as its reduction rate is uncontrollable.*

## 3 Benchmark Design

### 3.1 Evaluation Protocol

**A Unified Evaluation Approach.** Existing GC methods vary in their evaluation strategies to identify optimal condensed graphs throughout the condensation process. **First**, some approaches utilize the GNTK as the validation model, while others employ GNNs. **Second**, some select graphs based on the best test results rather than validation results. **Third**, some assess the condensed graph at every condensation epoch, whereas others opt for periodic evaluations to conserve computational resources. Thus, a unified evaluation approach is crucial for ensuring a fair comparison. We achieve this by unifying the validation model and restricting the validation frequency, as detailed in Section 4.1.

**Multi-Dimensional Evaluation.** Many methods overlook critical evaluation dimensions such as scalability, privacy preservation, NAS performance, and transferability. Our benchmark aims to address this gap by enabling a comprehensive comparison of GC methods across these key aspects.

*(a) Performance and Scalability.* We first attempt to reproduce and measure the basic results of all graph reduction methods within our scope. In addition to evaluating the performance of GCN in node classification, we assess their efficiency and highlight the trade-off between performance and efficiency to assist users in selecting the appropriate method based on their hardware resources. Our efficiency reports include preprocessing time, running time per epoch, total running time, peak memory, GPU memory and disk memory usage. By examining the resource consumption across various dataset sizes and reduction rates, we can also illustrate the scalability of different methods. Additionally, we also examine the condensation performance across broader reduction rates. *Summary: A good GC method should achieve good performance while also ensure high efficiency.*

*(b) Privacy Preservation.* As the downstream model is trained on a synthetic graph that differs from the original, GC may preserve a certain level of privacy by obscuring sensitive information. To evaluate this capability, we assess the resilience of GC against privacy attacks. Specifically, we apply the method from (Duddu et al., 2020) to measure privacy leakage across different GC techniques. This approach employs Membership Inference Attack (MIA) to assess privacy risks, where MIA accuracy reflects the probability that an adversary can correctly identify whether a node belongs to the training or test set. For a detailed explanation of why MIA is chosen over other attack methods, please refer to Appendix A.5. *Summary:* We anticipate that the condensed graph will mitigate the exposure of sensitive training information, such as membership, thereby reducing privacy risks.

*(c) Denoising ability.* Since GC preserves the essential information of the original graph, it can potentially reduce noise present in the original graph, even though it is not specifically designed for this purpose. We hypothesize that this capability may provide GC with denoising ability against various types of noise. To study this, we inject three types of noise to the original graph before feeding it into the GC algorithms: (1) **Feature noise**, which randomly changes features for all nodes, (2) **Structural noise**, which randomly modifies edges, and (3) **Adversarial structural noise**, which learns corrupt graph structure to degrade the performance of the GNN model. Furthermore, to examine the denoising ability of GC in two settings, transductive and inductive, we apply poisoning plus evasion corruption (i.e., corrupting both the training and test graphs) on transductive datasets, and poisoning corruption (i.e., only corrupting the training graph) on inductive datasets. *Summary:* We expect GC process can mitigate noise without specific denoising design.

*(d) Neural Architecture Search (NAS).* NAS (Elsken et al., 2019; Ren et al., 2021) is one of the most promising applications of GC. It focuses on identifying the best-performing architecture from a vast pool of models but is computationally expensive, which requires the training of numerous architectures on the full dataset. Since the condensed graph is much smaller than the whole graph, GC methods are utilized to accelerate NAS (Ding et al., 2022). In practical situations, preserving the rank of validation results between models trained on the condensed graph and the whole graph is important because we select the best architectures based on top validation results. We argue that all the graph condensation methods should be evaluated on the NAS task because it can effectively evaluate the practical value of a condensation method. *Summary: We expect a reliable correlation in validation performance between training on the condensed graph and the whole graph to be observed.*

*(e) Transferability.* The most critical aspect of evaluating GC methods is determining whether the condensed data can be effectively used to train diverse GNNs, adhering to a data-centric perspective. Usually, condensed graphs are closely tied to the backbone GNN used during the condensation process such as GCN and SGC, potentially embedding the inductive biases of that particular GNN, which might impair their performance on other GNNs. To address this concern, we aim for condensed graphs to exhibit consistent performance across different GNNs. Some previous studies (Jin et al., 2022b; Gao and Wu, 2023) don't include experiments evaluating transferability across GNNs. Additionally, evaluations of various methods are often performed on different datasets or reduction rates, hindering

fair comparison. Thus, we assess the performance of condensed graphs on multiple widely-used GNN models with a unified evaluation setting. *Summary: A high-quality condensed graph, like a graph in the real world, should be versatile enough to train different models.*

### 3.2 Impact of design choices

Current GC methods follow similar procedural frameworks, with multiple choices available at each intermediate stage of the process. However, the effects of these internal mechanisms, such as how different configurations or choices influence the performance and effectiveness of graph condensation, remain largely underexplored. In this benchmark, we aim to go beyond just the matching strategies discussed in Section 2.1, by thoroughly investigating the following key design choices.

**Data Initialization.** As a crucial stage in the standard procedure of GC, data initialization helps accelerate convergence and enhances final results (Cui et al., 2022). Besides, the initialization of the condensed graph can naturally integrated with coreset selection and graph coarsening methods. Previous work primarily relies on random selection for data initialization, with only a few studies employing alternative methods such as KCenter and Averaging (Zhang et al., 2024a; Gao and Wu, 2023). Therefore, we aim to conduct a comprehensive study on whether different data initialization can impact the performance of GC.

**Structure-Free vs. Structure-Based Methods.** Another important choice is whether to synthesize the structure. Structure-based methods including GCond, DosCond, and MSGC, utilize separate multilayer perceptrons (MLP) to generate links between nodes based on the synthetic node features. Other structure-based methods adopt different strategies, e.g, SGDD employs a structure broadcasting strategy, while GDEM aligns the eigenbasis to recover the adjacency matrix. To assist future research in making this decision, we discuss it in Section 4.2 and 4.4, as this choice shows significant differences in these two aspects.

**Graph Property Preservation.** Graph data comprises features, structures, and labels, which can be characterized by various established metrics, also known as graph properties. We aim to explore what graph properties are preserved by condensed graphs and understand the reasons behind the success of current GC methods. We select the following metrics from different aspects of a graph: **Density** (structure), **Max Eigenvalue** of Laplacian matrix (spectra), Davies-Bouldin Index (**DBI**) (Davies and Bouldin, 1979) (feature) and **Homophily** (Zhu et al., 2020a)(structure and label) . To further incorporate structural information into DBI, we developed a new metric named **DBI-AGG** (structure and feature), which calculates DBI based on node embeddings after two rounds of GCN-like aggregation.

## 4 Empirical Studies

### 4.1 Experimental Setup

In an attempt to address unfairness in this area, we unify some of the settings in GC papers while leaving other hyperparameters as reported in their papers or source code. **First**, we restrict one set of hyperparameters for each dataset, ensuring that they do not vary across different reduction rates. For methods that do not follow this setting, we use the set of hyperparameters from the highest reduction rate. This setting is more practical because tuning for every reduction rate can be very expensive. **Second**, we set the evaluation interval to the number of epochs divided by 10 to balance the frequency of intermediate evaluations and total epochs for each method. This strategy will benefit fast-converging and stable methods while penalizing those that rely on long epochs and frequent validation. **Third**, we adopt GCN in all evaluation parts, training a 2-layer GCN with 256 hidden units on the reduced graph. We then evaluate it on the validation and test sets of the original graph, using 300 epochs without early stopping. We select condensed graphs with best validation accuracy for final evaluation. To mitigate the effect of randomness, we run each evaluation 10 times and report the average performance. The above GNN training settings are applied across intermediate, final evaluations, and all other experiments. **Additionally**, sparsification is only applied to the final evaluation, with the threshold adhering to the reported results in the original paper. Specifically, for structure-free methods, an identity matrix is used as the adjacency matrix during training stage. Then, in inference stage, the original graph is input into the trained model. To benchmark methods under both transductive and inductive settings, we use the former for Citeseer, *Cora* (Kipf and Welling, 2016), *Pubmed* (Namata et al., 2012) and *Arxiv* (Hu et al., 2021), and the latter for *Flickr*, *Reddit* (Zeng et al., 2019) and *Yelp* (Rayana and Akoglu, 2015). All data preprocessing and training/validation/test set splits follow the GCond paper (Jin et al., 2022a). For datasets not used in GCond paper, we follow the settings of SGDD paper (Yang et al., 2024). More details about datasets and implementation are in Appendix A.2 and A.3.

### 4.2 Performance, Efficiency and Scalability

We report the performance of graph reduction methods in Table 1 and the efficiency in Figure 1.

**Obs. 1: TM-based methods show the best condensation performance but not the best efficiency.** From Table 1, we observe that GC methods significantly outperform coreset selection and coarsening methods and the margin is larger at low reduction rates. Among all, TM-based methods, GEOM and SFGC, lead across most datasets and reduction rates, showing the highest performance is achieved by trajectory methods. However, when we consider the efficiency and resource consumption in Figure 2, we find that though achieving state-of-the-art performance in Table 1, both GEOM and SFGC require additional preprocess time and large disk memory to produce and store the trajectory of experts. In addition, some learning-free methods, such as Averaging, exhibit high performance on certain datasets like *Yelp*, while being more efficient than all GC methods. Finally, the performance gap between the best GC methods and whole dataset training varies across datasets. Some datasets, like *Arxiv* and *Reddit*, still exhibit significant room for improvement.

**Obs. 2: Compared to structure-based methods, structure-free methods are more efficient while still performing well.** When comparing structure-free methods to their structure-based counterparts, such as GCondX and GCond, e.g., comparing GCondX and GCond in Figure 2 & 3 and Table 1, the following key insights emerge: (1) the absence of structure synthesis negatively impacts the performance of structure-free methods. (2) structure-based methods require significantly more memory and GPU resources, especially when applied to large graphs. (3) structure-free methods exhibit superior scalability w.r.t. reduction rates, as their computational resource usage remains relatively stable, even with increasing reduction rates. The increased complexity of structure-based methods stems from the time- and resource-intensive nature of structure synthesis, which must be repeated each time the synthetic features are updated. To fully harness the benefits of structure-based approaches, a more efficient structure generation method is needed. This is crucial as the structure provides valuable information beyond the features and has the potential to enhance the denoising ability, as discussed in Section 4.4.

**Obs. 3: GC outperforms image dataset condensation at the same reduction rate but struggles to scale effectively at larger reduction rates, where image dataset condensation excels.** We adjust the reduction rate from values corresponding to only one node per class to values that cause OOM on large datasets and present the results in Figure 3. While Figure 3 generally shows a positive correlation between performance and the reduction rate, we have three unique findings that are not observed in vision dataset condensation (Cui et al., 2022): (1) GC methods can still perform well when the Instance Per Class (IPC) is as low as 1; (2) Unlike in the image domain, GC methods cannot scale to larger IPC values due to OOM issues. We foresee the need for more scalable GC techniques, particularly those structure-based ones. In addition, our results indicate some instability of structure-free GC, as shown by $r$=0.5% on *Reddit* for GEOM and $r$=1.25% on *Arxiv* for GCondX.

### 4.3 Privacy Preservation

This attack reveals which samples were used in training, leading to privacy leakage of training set. It leverages confidence scores, i.e., the probability of the true label, to identify if a sample was part of the training set. The optimal threshold is determined by analyzing all confidence scores to maximize the attack's success in distinguishing between training and non-training samples.

**Obs. 4: Certain GC methods can achieve both privacy preservation and high condensation performance.** The results in Table 2 suggest the following: (1) compared to non-protected whole dataset training, GC methods enhance membership privacy by around 5%-10% on *Cora* and *Citeseer*. Notably, GDEM achieves significant preservation performance on *Cora*, with an improvement up to 14.21%, while still maintain a good performance (Acc). Also, certain method such as GEOM achieve both lowest MIA Acc and highest Acc on *Citeseer*, highlight the nature of GC in reducing the risk of privacy leakage. These improvements stem from the fact that no real training nodes are used when we apply GC, ensuring the membership information remains protected. In addition, the gain in *Arxiv* is not as significant, and we conjecture that it's close to the lower bound of 50%, resulting

Table 1: Performance of graph reduction methods under three reduction rates. We report test accuracy (%) for all datasets, except for *Yelp*, where we use F1-macro (%). The best and the second-best results, excluding the whole graph training, are marked in **bold** and underlined. *Structure-free* and *structure-based* condensation methods are marked in blue and red, respectively.

| Dataset | Reduction rate (%) | *Coreset* | | | | | *Coarsening* | | *Condensation* | | | | | | | | Whole |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | *TM* | | *DM* | *GM* | | | | | |
| | | **Cent-D** | **Cent-P** | **Random** | **Herding** | **K-Center** | **Averaging** | **VNG** | **GEOM** | **SFGC** | **GCDM** | **GCondX** | **GCond** | **DosCond** | **MSGC** | **SGDD** | |
| *Citeseer* | 0.36 | 42.86 | 37.78 | 35.37 | 43.73 | 41.43 | 69.75 | 66.14 | 67.61 | 66.27 | 70.65 | 67.79 | 70.05 | 69.41 | 60.24 | **71.87** | 72.6 |
| | 0.90 | 58.77 | 52.83 | 50.71 | 59.24 | 51.15 | 69.59 | 66.07 | 70.70 | 70.27 | 71.27 | 69.69 | 69.15 | 70.83 | **72.08** | 70.52 | |
| | 1.80 | 62.89 | 63.37 | 62.62 | 66.66 | 59.04 | 69.50 | 65.34 | **73.03** | 72.36 | 72.08 | 68.38 | 69.35 | 72.18 | 72.21 | 69.65 | |
| *Cora* | 0.50 | 57.79 | 58.44 | 35.14 | 51.68 | 44.64 | 75.94 | 70.40 | 78.14 | 75.11 | 79.21 | 79.74 | 80.17 | **80.65** | 80.54 | 80.15 | 81.5 |
| | 1.30 | 66.45 | 66.38 | 63.63 | 68.99 | 63.28 | 75.87 | 74.48 | **82.29** | 79.55 | 80.26 | 78.67 | 80.81 | 80.85 | 80.98 | 80.29 | |
| | 2.60 | 75.79 | 75.64 | 72.24 | 73.77 | 70.55 | 75.76 | 76.03 | **82.82** | 80.54 | 80.68 | 78.60 | 80.54 | 81.15 | 80.94 | 81.04 | |
| *Pubmed* | 0.02 | 56.16 | 57.28 | 49.46 | 62.91 | 62.91 | 75.60 | 75.60 | 69.64 | 67.61 | 77.62 | 72.03 | 77.36 | 58.13 | 75.25 | **78.11** | 78.6 |
| | 0.03 | 55.61 | 62.50 | 56.10 | 69.28 | 65.59 | 75.60 | 75.72 | 76.21 | 66.89 | 76.63 | 72.05 | 78.05 | 52.70 | **78.26** | 78.07 | |
| | 0.15 | 71.95 | 73.35 | 71.84 | 75.53 | 74.00 | 75.60 | 77.53 | **78.49** | 67.61 | 77.48 | 71.97 | 76.46 | 76.45 | 78.20 | 75.95 | |
| *Arxiv* | 0.05 | 32.88 | 36.48 | 50.39 | 51.49 | 50.52 | 59.62 | 54.89 | **64.91** | 64.91 | 60.04 | 59.40 | 60.49 | 55.70 | 57.66 | 58.50 | 71.4 |
| | 0.25 | 48.85 | 47.90 | 58.92 | 58.00 | 55.28 | 59.96 | 59.66 | **68.78** | 66.58 | 60.59 | 62.46 | 63.88 | 57.39 | 64.85 | 59.18 | |
| | 0.50 | 52.01 | 55.65 | 60.19 | 57.70 | 58.66 | 59.94 | 60.93 | **69.59** | 67.03 | 60.71 | 59.93 | 64.23 | 61.06 | 65.73 | 63.76 | |
| *Flickr* | 0.10 | 40.70 | 40.97 | 42.94 | 42.80 | 43.01 | 37.93 | 44.33 | **47.15** | 46.38 | 43.75 | 46.66 | 46.75 | 45.87 | 46.21 | 46.69 | 47.4 |
| | 0.50 | 42.90 | 44.06 | 44.54 | 43.86 | 43.46 | 37.76 | 43.30 | 46.71 | 46.38 | 45.05 | 46.69 | **47.01** | 45.89 | 46.77 | 46.39 | |
| | 1.00 | 42.62 | 44.51 | 44.68 | 45.12 | 43.53 | 37.66 | 43.84 | 46.13 | 46.61 | 45.88 | 46.58 | **46.99** | 45.81 | 46.12 | 46.24 | |
| *Reddit* | 0.05 | 40.00 | 45.83 | 40.13 | 46.88 | 40.24 | 88.23 | 69.96 | **90.63** | 90.18 | 87.28 | 86.56 | 85.39 | 86.56 | 87.62 | 87.37 | 94.4 |
| | 0.10 | 50.47 | 51.22 | 55.73 | 59.34 | 48.28 | 88.32 | 76.95 | **91.33** | 89.84 | 89.96 | 88.25 | 89.82 | 88.32 | 88.15 | 88.73 | |
| | 0.20 | 55.31 | 61.56 | 58.39 | 73.46 | 56.81 | 88.33 | 81.52 | **91.03** | 90.71 | 89.08 | 88.73 | 90.42 | 88.84 | 87.03 | 90.65 | |
| *Yelp* | 0.05 | 48.67 | 46.81 | 46.08 | 46.08 | 46.07 | **55.04** | 49.24 | 52.80 | 46.20 | 50.75 | 52.44 | 52.30 | 51.10 | 52.94 | 52.02 | 58.2 |
| | 0.10 | 51.03 | 46.08 | 46.28 | 52.23 | 46.22 | **53.51** | 47.33 | 47.56 | 47.96 | 52.49 | 49.70 | 53.22 | 52.54 | 50.97 | 54.13 | |
| | 0.20 | 46.08 | 46.08 | 49.31 | 47.49 | 46.85 | 54.42 | 48.63 | 49.48 | 46.70 | **55.89** | 48.77 | 51.76 | 52.19 | 51.35 | 52.86 | |

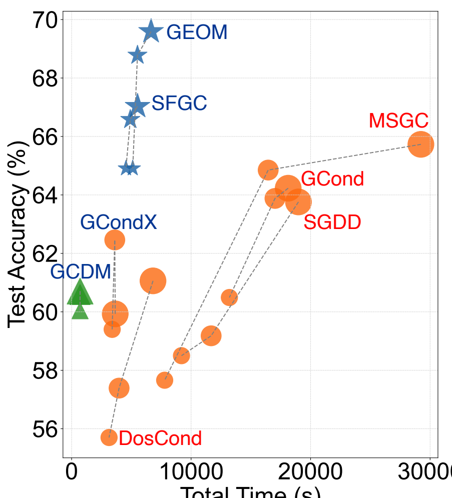



Figure 1: Test accuracy vs. total time for *structure-free* and *structure-based* condensation methods on *Arxiv*. TM is represented by ★, GM by ●, and DM by ▲. Marker sizes increase with reduction rates of 0.05%, 0.25%, and 0.50%.

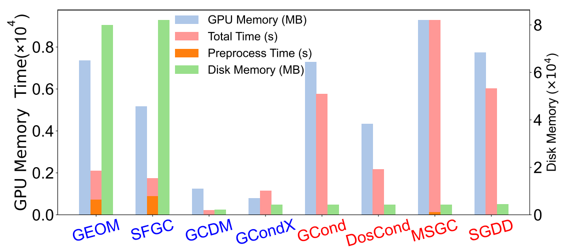



Figure 2: Comparison of GPU memory, disk memory, preprocess time, and total time on *Arxiv* ($r = 0.5\%$).

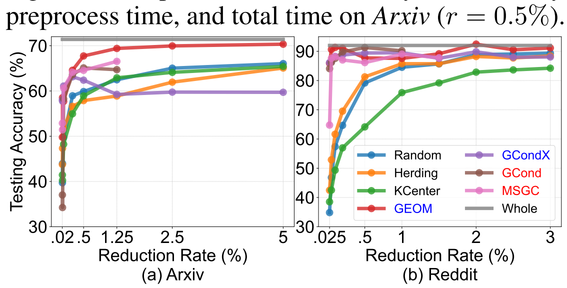



Figure 3: Varying reduction rates on Arxiv and Reddit. No mark represents OOM when the reduction rate is too large for a method.

in a smaller margin of improvement. (2) Different reduction methods vary in their effectiveness. For example, GEOM and GDEM exhibit a strong balance between mitigating MIA accuracy and maintaining model performance. This suggests the potential to design improved GC methods that do not compromise privacy. In other words, the typical tradeoff between utility and privacy preservation could potentially be eliminated through the use of GC techniques.

### 4.4 DENOISING ABILITY

To explore the denoising ability of GC methods, specifically their ability to mitigate noise from the original graph via the condensation process, we inject three types of representative noise as outlined in Section 3.1. These include: (1) **Feature Noise**: We simulate feature noise by masking node features to zero. (2) **Structural Noise**: This is introduced by randomly adding edges to the

Table 2: Privacy preservation evaluation. "MIA Acc" measures how well an attacker can infer whether a node is in the training or test set. We also report node classification accuracy ("Acc"), aiming to emphasize the balance between model performance and privacy preservation.

| Methods | *Cora, r = 2.6%* | | *Citeseer, r = 1.8%* | | *Arxiv, r = 0.5%* | |
|---|---|---|---|---|---|---|
| | MIA Acc (↓) | Acc (↑) | MIA Acc (↓) | Acc (↑) | MIA Acc (↓) | Acc (↑) |
| **Whole** | 74.87 ± 1.16 | 81.50 ± 0.50 | 81.76 ± 1.01 | 72.61 ± 0.27 | 54.26 ± 0.11 | 71.43 ± 0.11 |
| **GCond** | 72.10 ± 0.96 | 80.54 ± 0.67 | 74.11 ± 0.61 | 69.35 ± 0.82 | **53.04 ± 0.18** | 64.23 ± 0.16 |
| **GCondX** | 66.83 ± 0.81 | 78.60 ± 0.31 | 71.97 ± 0.58 | 68.38 ± 0.45 | 54.64 ± 0.17 | 59.93 ± 0.54 |
| **DosCond** | 69.70 ± 0.50 | 81.15 ± 0.50 | 74.33 ± 0.34 | 72.18 ± 0.61 | 54.04 ± 0.79 | 61.06 ± 0.59 |
| **SGDD** | 70.43 ± 1.63 | 81.04 ± 0.54 | 77.07 ± 4.32 | 69.65 ± 1.68 | 53.29 ± 0.46 | 63.76 ± 0.22 |
| **GDEM** | **60.66 ± 1.26** | 81.76 ± 0.53 | 70.01 ± 2.94 | 71.74 ± 0.90 | - | - |
| **GEOM** | 67.90 ± 0.55 | **82.82 ± 0.17** | **67.55 ± 0.62** | **73.03 ± 0.31** | 53.80 ± 0.19 | **69.59 ± 0.24** |
| **SFGC** | 67.29 ± 1.02 | 80.54 ± 0.45 | 72.12 ± 0.44 | 72.36 ± 0.53 | 54.49 ± 0.53 | 67.03 ± 0.48 |

Table 3: Denoising ability evaluation. "Perf. Drop" shows the relative loss of accuracy compared to the original results before corruption. The best results are in **bold** and results that outperform whole dataset training are underlined. *Structure-free* and *Structure-based* methods are colored blue and red.

| Dataset | Method | Feature Noise | | Structural Noise | | Adversarial Structural Noise | |
|---|---|---|---|---|---|---|---|
| | | Test Acc. ↑ | Perf. Drop ↓ | Test Acc. ↑ | Perf. Drop ↓ | Test Acc. ↑ | Perf. Drop ↓ |
| *Citeseer 1.8%* | Whole | 64.07 | 11.75% | 57.63 | 20.62% | 53.90 | 25.76% |
| | GCond | **64.06** | **7.63%** | **65.64** | **5.35%** | **66.19** | **4.55%** |
| | GCondX | 61.27 | 10.40% | 60.42 | 11.65% | 60.75 | 11.15% |
| | GEOM | 58.77 | 19.53% | 51.41 | 29.60% | 57.94 | 20.67% |
| *Cora 2.6%* | Whole | 74.77 | 8.26% | 72.13 | 11.49% | 66.63 | 18.24% |
| | GCond | 67.62 | 16.04% | 63.14 | 21.61% | 68.90 | 14.45% |
| | GCondX | **67.72** | **13.85%** | **63.95** | **18.63%** | **69.24** | **11.91%** |
| | GEOM | 49.68 | 40.01% | 53.59 | 35.29% | 66.32 | 19.93% |
| *Flickr 1%* | Whole | 46.68 | 1.51% | 42.60 | 10.13% | 44.44 | 6.24% |
| | GCond | **46.29** | **1.49%** | **46.97** | **0.04%** | 43.90 | 6.58% |
| | GCondX | 45.60 | 2.11% | 46.19 | 0.83% | 42.00 | 9.83% |
| | GEOM | 45.38 | 1.63% | 45.52 | 1.32% | **44.72** | **3.06%** |

graph. (3) **Adversarial Structural Noise**: We employ PR-BCD (Geisler et al., 2021), a scalable adversarial noise using Projected Gradient Descent (PGD). In transductive settings, we apply both poisoning and evasion corruptions, which affects both the training and test phases of the graph. The perturbation rates are set to 50% for feature and structural noise and 25% for adversarial structural noise, respectively. Each corruption is repeated three times, producing three distinct corrupted graphs. We then evaluate and report the average performance across these graphs.

**Obs. 5: GC methods exhibit a certain level of denoising ability against structural noise, with structure-based approaches offering superior denoising compared to structure-free ones.** As shown in Table 3, GC methods outperform GCN trained on the whole corrupted graph in the two structural noises, but GC does not show denoising ability against feature noise. For example, GC methods achieve the highest Test Acc. across three datasets under structural noise but fall short when dealing with feature noise. This suggests that GC methods are more effective at handling structural denoising than feature denoising. Additionally, the state-of-the-art methods GEOM and the structure-free version of GCond, GCondX show lower performance compared to GCond after being corrupted, indicating that structure-free methods lose some denoising ability if they do not synthesize the structure. While GC can mitigate some noise, it still lacks specialized denoising mechanisms to achieve stronger denoising capabilities, presenting a potential direction for future work.

### 4.5 Neural Architecture Search

As a key application of GC, we evaluate the performance of NAS using three commonly-used metrics: Top 1 test accuracy, correlation between validation set accuracies, and correlation between ranks of validation set accuracies of the condensed graph and the whole graph. We use the Pearson coefficient (Cohen et al., 2009) to quantify the correlation. We conduct NAS with APPNP, a flexible GNN model whose structure can vary by using a different number of propagation layers, residual coefficients, etc. More details are provided in the Appendix A.7.

Table 4: NAS evaluation. The best result is in **bold**. The runner-up is underlined. The **worst** is colored red.

| | Random | K-Center | GCondX | SFGC | GEOM | GCond | DosCond | MSGC | Whole |
|---|---|---|---|---|---|---|---|---|---|
| Top 1 (%) | 81.88 | 81.74 | 81.49 | **82.42** | 82.19 | 81.82 | 81.91 | 82.40 | 82.51 |
| Acc. Corr. | 0.56 | 0.47 | 0.40 | **0.72** | 0.65 | 0.70 | 0.14 | 0.71 | - |
| Rank Corr. | 0.64 | 0.60 | 0.57 | 0.71 | 0.74 | 0.66 | 0.20 | **0.78** | - |

**Obs. 6: Trajectory matching or inner optimization is essential for reliable NAS effectiveness.** The results in Table 4 demonstrate that: (1) GC methods demonstrate a strong potential to identify

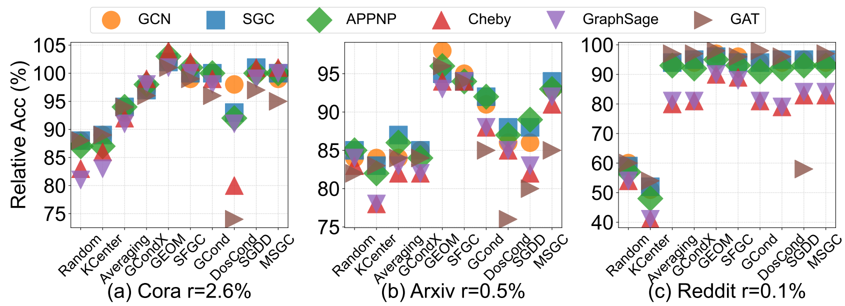



Figure 4: Condensed graph performance evaluated by different GNNs. The **relative accuracy** refers to the accuracy preserved compared to training on the whole dataset.

the best architectures, sometimes even outperforming the results obtained from the original dataset. (2) Methods utilizing trajectory matching demonstrate strong results in NAS. (3) Models without inner optimization during the condensation process, such as DosCond, yield poor NAS performance, with a Pearson correlation coefficient below 0.6. Given that methods employing trajectory matching or inner optimization tend to achieve better NAS results, we hypothesize that explicitly mimicking the training trajectory of GNNs is critical for effective NAS.

### 4.6 Transferability

We conduct extensive experiments assessing the performance of condensed graphs on six widely-used GNN models: GCN (Kipf and Welling, 2016), SGC (Wu et al., 2019b), APPNP (Gasteiger et al., 2018), Cheby (Defferrard et al., 2016), GraphSage (Hamilton et al., 2017) and GAT (Veličković et al., 2018). We tune hyperparameters for these evaluation GNN models, with the search space for hyperparameters and sensitivity analysis listed in Appendix A.6. To simplify, we fix the reduction ratios at 2.6%, 0.5%, and 0.1% for *Cora*, *Arxiv* and *Reddit*, respectively.

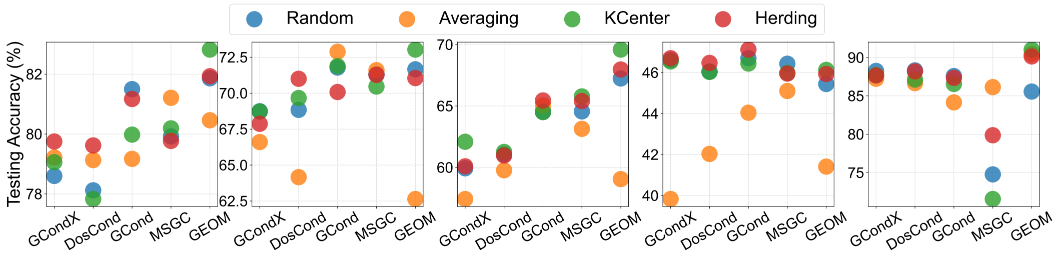
**Obs. 7: Different GC methods exhibit varying degrees of transferability across datasets, leaving considerable room for improvement in this area.** From Figure 4 we can observe that (1) there is no significant performance loss for the majority of cases when condensed graphs are transferred to various GNNs. This highlights the success of GC methods, which typically only use GCN or SGC for condensation. (2) However, for some methods such as DosCond and SGDD, GAT performs much worse than other GNNs. We conjecture this is because GAT is more structure-sensitive and can only leverage the connection information instead of the edge weights. (3) We also investigate the transferability to Graph Transformer (Wu et al., 2023) in Appendix A.6. However, the performance of Graph Transformer drops a lot, which suggests that future research should explore the transferability to non-GNN graph learning architectures.

**Obs. 8: Trajectory matching or inner optimization facilitates transferability.** GEOM and SFGC achieve significantly better performance than GCondX. Similarly, GCond outperforms DosCond. These two phenomena indicate that trajectory matching or inner optimization is key to improving transferability. We conjecture these two designs introduce additional inductive biases related to the backbone models used in the condensation process, which likely benefit all message-passing GNNs.

### 4.7 Data Initialization

To study the impact of different data initialization strategies, we equip 5 GC methods with 5 initialization strategies across all datasets. **Obs. 9: Current initialization strategies do not have a consistent impact across all datasets or GC methods.** Figure 5 illustrates that there is no single best data initialization method for every GC method or dataset. Notably, KCenter is the average best initialization method for most datasets. Averaging is a very unstable strategy, especially for large datasets, and it only works in rare cases. We conclude that GC methods do not need to be consistently good with different initialization strategies. Therefore, we recommend treating initialization strategies as hyperparameters in future studies. **Obs. 10: Better coreset selection methods do not guarantee better GC initialization.** When we compare Figure 5 with coreset and coarsening columns in Table 9, we find that the best one, Herding, is not necessarily the best data initialization method for GC. This finding cautions that future research should carefully combine different graph reduction methods, as various GC methods may not complement each other effectively.

### 4.8 Graph Property Preservation




(a) Cora r=2.6% (b) Citeseer r=1.8% (c) Arxiv r=0.5% (d) Flickr r=1% (e) Reddit r=0.1%

Figure 5: Test accuracy for different methods with different initialization.

We explore the relationship between graph property preservation and structure-based GC methods. We calculate the metrics related to different graph properties for the condensed graph. For MSGC, we calculate the average results.

Table 5: Graph properties of condensed graphs on Cora.

| Graph Property | | VNG | GCond | MSGC | SGDD | Avg. | Whole |
|---|---|---|---|---|---|---|---|
| Density% (Structure) | *Cora* | 52.17 | 82.28 | 22.00 | 100.00 | 64.11 | 0.14 |
| | Corr. | -0.81 | 0.07 | 0.55 | 0.13 | -0.02 | - |
| Max Eigenvalue (Spectra) | *Cora* | 3.73 | 34.90 | 1.69 | 14.09 | 13.60 | 169.01 |
| | Corr. | 0.85 | 0.25 | 0.95 | 0.28 | 0.58 | - |
| DBI (Label & Feature) | *Cora* | 3.69 | 1.84 | 0.70 | 4.34 | 2.64 | 9.28 |
| | Corr. | 0.81 | 0.93 | 0.94 | 0.97 | **0.91** | - |
| DBI-AGG (Label & Feat. & Stru.) | *Cora* | 3.59 | 0.38 | 0.57 | 0.18 | 1.18 | 4.67 |
| | Corr. | 0.99 | 0.93 | 0.95 | 0.89 | **0.94** | - |
| Homophily (Label & Structure) | *Cora* | 0.14 | 0.16 | 0.19 | 0.13 | 0.16 | 0.81 |
| | Corr. | -0.83 | -0.68 | -0.46 | -0.80 | -0.69 | - |

**Obs. 11: Only the properties related to node features and aggregated features, i.e., *DBI* and *DBI-AGG*, are relatively preserved in condensed graphs.** Despite examining various graph-size-agnostic graph properties, our results in Table 5 show that none of the absolute values tend to be preserved. Consequently, we resort to the *Pearson correlation* between metrics in the original and condensed graphs. From the results, we can conclude that only *DBI* and *DBI-AGG* are relatively preserved, as they have average correlation coefficients of 0.91 and 0.94. Therefore, we suggest that researchers explicitly preserve these two properties to potentially bolster performance. Notably, we observed that MSGC preserves the maximum eigenvalue up to 0.94. As further evidence, the latest method, GDEM (Liu et al., 2023b), focuses on learning to preserve eigenvectors, supporting the idea that maintaining spectral properties may be beneficial. In contrast, *Density* appears to be the least important property to preserve among these GC methods. Additionally, we observe that a homophilous graph is often condensed into a heterophilous graph while still achieving high performance. This finding suggests that the relationship between GNN performance and homophily (Zheng et al., 2022; Zhu et al., 2020b) need to be reconsidered.

# 5 Conclusion and Outlook

This paper establishes the first benchmark for GC methods with multi-dimension evaluation, providing novel insights on privacy preservation, denoising ability, and design choices of current GC methods. The findings from our experimental results inspire the following future directions:

(1) **Better performance and scalability.** Future work can focus on closing the gap between GC methods and whole dataset training, and scaling to larger datasets and higher reduction rates.

(2) **Comprehensive Privacy Preservation.** Future work can exploit the privacy preservation advantage of GC methods to synthesize graphs that safeguard additional types of privacy.

(3) **Stronger Denoising Ability.** Future work can further explore the denoising ability of graph condensation methods under diverse settings, such as feature attacks and out-of-distribution (OOD) and develop techniques to enhance their robustness. Furthermore, it would also be of interest to incorporate GNN defense methods to enhance the denoising ability of GC methods.

(4) **Leveraging coreset selection or coarsening.** Future work can combine powerful coreset selection and graph coarsening methods, making GC competitive in both efficiency and performance.

**Limitations.** We anticipate that our benchmark and insights will contribute to progress in the field and encourage the development of more practical GC methods going forward. However, GC-Bench is not without limitations and some areas of benchmarking can be further explored. These include examining the effectiveness of other privacy techniques such as Differential Privacy (Ponomareva et al., 2023), evaluating denoising ability against other types of attacks, measuring NAS effectiveness in larger architecture spaces such as Graph Design Space (You et al., 2020), examining the transferability of condensed knowledge to various domains and downstream tasks, and identifying and preserving certain graph properties to improve performance.

# REFERENCES


Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.

Zhichun Guo, Bozhao Nan, Yijun Tian, Olaf Wiest, Chuxu Zhang, and Nitesh V Chawla. Graph-based molecular representation learning. *International Joint Conference on Artificial Intelligence*, 2023.

Hao Wang, Jiaxin Yang, and Jianrong Wang. Leverage large-scale biological networks to decipher the genetic basis of human diseases using machine learning. *Artificial Neural Networks*, pages 229–248, 2021.

Zewen Liu, Guancheng Wan, B Aditya Prakash, Max SY Lau, and Wei Jin. A review of graph neural networks in epidemic modeling. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024a.

Yu Wang, Yuying Zhao, Yi Zhang, and Tyler Derr. Collaboration-aware graph convolutional network for recommender systems. In *Proceedings of the ACM Web Conference 2023*, 2023a.

Kaize Ding, Albert Jiongqian Liang, Bryan Perozzi, Ting Chen, Ruoxi Wang, Lichan Hong, Ed H Chi, Huan Liu, and Derek Zhiyuan Cheng. Hyperformer: Learning expressive sparse feature representations via hypergraph transformer. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2062–2066, 2023.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019a.

Jiajun Zhou, Zhi Chen, Min Du, Lihong Chen, Shanqing Yu, Guanrong Chen, and Qi Xuan. Robustecd: Enhancement of network structure for robust community detection. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):842–856, 2021.

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

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022a. URL `https://openreview.net/forum?id=WLEx3Jo4QaB`.

Shichang Zhang, Atefeh Sohrabizadeh, Cheng Wan, Zijie Huang, Ziniu Hu, Yewen Wang, Jason Cong, Yizhou Sun, et al. A survey on graph neural network acceleration: Algorithms, systems, and customized hardware. *arXiv preprint arXiv:2306.14052*, 2023.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Dc-bench: Dataset condensation benchmark. *Advances in Neural Information Processing Systems*, 35:810–822, 2022.

Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

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

Hongjia Xu, Liangliang Zhang, Yao Ma, Sheng Zhou, Zhuonan Zheng, and Bu Jiajun. A survey on graph condensation. *arXiv preprint arXiv:2402.02000*, 2024.

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

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

Qiying Pan, Ruofan Wu, LIU Tengfei, Tianyi Zhang, Yifei Zhu, and Weiqiang Wang. Fedgkd: Unleashing the power of collaboration in federated graph neural networks. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.

Mucong Ding, Xiaoyu Liu, Tahseen Rabbani, Teresa Ranadive, Tai-Ching Tuan, and Furong Huang. Faster hyperparameter search for gnns via calibrated dataset condensation. 2022.

Noveen Sachdeva and Julian McAuley. Data distillation: A survey. *Transactions on Machine Learning Research*, 2023.

Yilun Liu, Ruihong Qiu, and Zi Huang. Gcondenser: Benchmarking graph condensation. *arXiv preprint arXiv:2405.14246*, 2024b.

Qingyun Sun, Ziying Chen, Beining Yang, Cheng Ji, Xingcheng Fu, Sheng Zhou, Hao Peng, Jianxin Li, and Philip S Yu. Gc-bench: An open and unified benchmark for graph condensation. *arXiv preprint arXiv:2407.00615*, 2024.

Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 720–730, 2022b.

Yuchen Zhang, Tianle Zhang, Kai Wang, Ziyao Guo, Yuxuan Liang, Xavier Bresson, Wei Jin, and Yang You. Navigating complexity: Toward lossless graph condensation via expanding window matching. *arXiv preprint arXiv:2402.05011*, 2024a.

Jian Gao and Jianshe Wu. Multiple sparse graphs condensation. *Knowledge-Based Systems*, 278: 110904, 2023.

Beining Yang, Kai Wang, Qingyun Sun, Cheng Ji, Xingcheng Fu, Hao Tang, Yang You, and Jianxin Li. Does graph distillation see like vision dataset counterpart? *Advances in Neural Information Processing Systems*, 36, 2024.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4750–4759, 2022.

Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems*, 36, 2024.

Yanqing Liu, Jianyang Gu, Kai Wang, Zheng Zhu, Wei Jiang, and Yang You. Dream: Efficient dataset distillation by representative matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17314–17324, 2023a.

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

Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. Fast graph condensation with structure-based neural tangent kernel. *arXiv preprint arXiv:2310.11046*, 2023b.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

Yang Liu, Deyu Bo, and Chuan Shi. Graph condensation via eigenbasis matching. *arXiv preprint arXiv:2310.09202*, 2023b.

Mucong Ding, Yinhan He, Jundong Li, and Furong Huang. Spectral greedy coresets for graph neural networks. *arXiv preprint arXiv:2405.17404*, 2024.

Linfeng Cao, Haoran Deng, Chunping Wang, Lei Chen, and Yang Yang. Graph-skeleton:˜ 1% nodes are sufficient to represent billion-scale graph. *arXiv preprint arXiv:2402.09565*, 2024.

Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 126–134, 2005.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*, pages 1121–1128, 2009.

Si Si, Felix Yu, Ankit Singh Rawat, Cho-Jui Hsieh, and Sanjiv Kumar. Serving graph compression for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2022.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 675–684, 2021.

Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 76–85, 2020.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020a.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, page 1, 2012.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, pages 985–994, 2015.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34:7637–7649, 2021.

Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. In *Advances in Neural Information Processing Systems*, 2023.

Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S Yu, and Shirui Pan. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020b.

Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Guha Thakurta. How to dp-fy ml: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research*, 77:1113–1201, 2023.

Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Amy N Langville and Carl D Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bring order to the web. In *Proc. of the 7th International World Wide Web Conf*, 1998.

Arthur L Liestman and Thomas C Shermer. Additive graph spanners. *Networks*, 23(4):343–363, 1993.

Yilun Liu, Ruihong Qiu, and Zi Huang. Cat: Balanced continual graph learning with graph condensation. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 1157–1162. IEEE, 2023c.

Yi Zhang, Yuying Zhao, Zhaoqing Li, Xueqi Cheng, Yu Wang, Olivera Kotevska, S Yu Philip, and Tyler Derr. A survey on privacy in graph neural networks: Attacks, preservation, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2024b.

Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chee-Kong Lee, and Enhong Chen. Model inversion attacks against graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):8729–8741, 2022.

Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. *ACM Transactions on Privacy and Security (TOPS)*, 21(1):1–30, 2018.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv preprint arXiv:2109.05641*, 2021.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

Junfeng Fang, Xinglin Li, Yongduo Sui, Yuan Gao, Guibin Zhang, Kun Wang, Xiang Wang, and Xiangnan He. Exgc: Bridging efficiency and explainability in graph condensation. In *Proceedings of the ACM on Web Conference 2024*, pages 721–732, 2024.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.

# A Appendix

## A.1 Comparison with concurrent works

To better illustrate the differences of scope and details of our benchmark and others, we create the table below:

Table 6: Comparison between our GC4NC and two concurrent works. "OOM" means if the benchmark explore when the GC methods report out-of-memory error. In "Impact of Initialization", *new strategy* means the initialization is not served as one baseline methods (coreset or coarsening).

| Benchmark Scope | **GCondenser** (Liu et al., 2024b) | **GC-Bench** (Sun et al., 2024) | **GC4NC** |
|---|---|---|---|
| **Methods** | | | |
| *Coreset & Sparsification* | Random, KCenter | Random, KCenter, Herding | Cent-D, Cent-P, Random, KCenter, Herding, TSpanner |
| *Coarsening* | - | - | Averaging, VNG, Clustering, VN |
| *Condensation* ↓ | | | |
| Gradient Matching | GCond, DosCond, SGDD | GCond, DosCond, SGDD | GCond, DosCond, SGDD, MSGC |
| Trajectory Matching | SFGC | SFGC, GEOM | SFGC, GEOM |
| Others | GCDM, DM, GDEM | GCDM, DM, KiDD, Mirage | GCDM, GDEM, GCSNTK |
| **Datasets** | Cora, Citeseer, Pubmed, Arxiv, Flickr, Reddit | Cora, Citeseer, Arxiv, Flickr, Reddit, Yelp, Amazon DBLP, ACM, NCI1, DD, ogbg-molbace ogbg-molbbbp, ogbg-molhiv | Cora, Citeseer, Pubmed, Arxiv, Flickr, Reddit, Yelp |
| **Tasks** | Node classification | Node classification, link prediction, node clustering, graph classification | Node classification |
| **Evaluation Protocols** | | | |
| Performance on standard condensation rate | ✓ | ✓ | ✓ |
| Efficiency & Scalability | Time | Time, Memory, OOM | Time, Memory, Disk Space, OOM |
| Transferability | Cross-model | Cross-model (include GraphTransformer), cross-task | Cross-model (include GraphTransformer) |
| Privacy preservation | - | - | ✓ |
| Denoising Ability | - | - | ✓ |
| Neural Architecture Search | - | - | ✓ |
| Continual learning | ✓ | - | - |
| **Impact of inner mechanism** | | | |
| Impact of if synthesizing the structure | ✓ | ✓ | ✓ |
| Impact of Initialization | 2 new and 1 coreset strategies | 5 new strategies | 5 coreset and coarsening strategies |
| Impact of validators | ✓ | - | - |
| Graph properties | - | ✓ | ✓ |

From this table, our contributions are evident. **First**, we incorporate a broader range of traditional coreset and coarsening methods, along with additional condensation methods focused on node classification (NC). **Second**, we provide a more comprehensive analysis of efficiency and scalability, including disk space considerations. **Third**, we explore the application of GC methods in terms of privacy preservation and denoising effects. **Finally**, our data initialization aligns with the coreset and coarsening methods, resulting in elegant, reusable code and enabling a preliminary trial of multi-layer condensation.

Table 6 may also show some limitations of our benchmark, though most of these stem from differences in opinion and focus. (1) As our title suggests, GC4NC is primarily a benchmark for NC, since the majority (approximately 90%) of condensation papers have concentrated on this task. That's also why we have fewer datasets compared to GC-Bench. (2) We argue that the condensation model and validator can be viewed as hyperparameters, similar to how methods like GEOM approach it. Therefore, we do not study the impact of them as they are just selected by datasets. (3) With regard to another important application, Continual Learning (CL), Gcondenser (Liu et al., 2024b) points out that many existing methods, including GDEM, SFGC, and GEOM, are incompatible with graph continual learning frameworks. This somewhat lowers the priority of CL as they are most competitive ones.

## A.2 Datasets

We evaluate all the methods on four transductive datasets: *Cora*, *Citeseer*, *Pubmed* and *Arxiv*, and three inductive datasets: *Flickr*, *Reddit* and *Yelp*. The **reduction rate** is calculated by ( number of nodes in condensed graph) / (number of nodes in training graph). Specifically, the training graph is defined as the whole graph in transductive datasets, and only the training set for inductive datasets. Dataset statistics are shown in Table 7.

For the choices of reduction rate $r$, we divide the discussion into two parts: for transductive datasets (i.e. *Citeseer*, *Cora* and *Arxiv*), their training graph is the whole graph. For *Citeseer* and *Cora*, since their labeling rates of training graphs are very small (3.6% and 5.2%, respectively), we choose $r$ to

Table 7: Datasets Statistics

| Dataset | #Nodes | #Edges | #Classes | #Features | #Training/Validation/Test |
|---|---|---|---|---|---|
| *Citeseer* | 3,327 | 4,732 | 6 | 3,703 | 120/500/1000 |
| *Cora* | 2,708 | 5,429 | 7 | 1,433 | 140/500/1000 |
| *Pubmed* | 19,717 | 88,648 | 3 | 500 | 60/500/1000 |
| Arxiv | 169,343 | 1,166,243 | 40 | 128 | 90,941/29,799/48,603 |
| Flickr | 89,250 | 899,756 | 7 | 500 | 44,625/22,312/22,313 |
| Reddit | 232,965 | 57,307,946 | 210 | 602 | 15,3932/23,699/55,334 |
| Yelp | 45,954 | 3,846,979 | 2 | 32 | 36,762/4,596/4,596 |

be $\{10\%, 25\%, 50\%\}$ of the labeling rate. For *Arxiv*, the labeling rate is 53% and we choose $r$ to be $\{1\%, 5\%, 10\%\}$ of the labeling rate; for inductive datasets (i.e. *Flickr*, *Reddit* and *Yelp*), the nodes of their training graphs are all labeled (labeling rate is 100%). Thus, the fraction of labeling rate is equal to the final reduction rate $r$. The labeling rate, fraction of labeling rate and final reduction rate $r$ of each dataset are shown in Table 8.

Table 8: Explanation of Reduction Rate under transductive and inductive settings

| Dataset | Labeling Rate | Reduction Rate of Labeled Nodes | Reduction Rate $r$ |
|---|---|---|---|
| *Citeseer* | 3.6% | 10% | 0.36% |
| | | 25% | 0.9% |
| | | 50% | 1.8% |
| *Cora* | 5.2% | 10% | 0.5% |
| | | 25% | 1.3% |
| | | 50% | 2.6% |
| *Pubmed* | 0.3% | 1% | 0.3% |
| | | 10% | 3% |
| | | 50% | 15% |
| *Arxiv* | 53% | 1% | 0.05% |
| | | 5% | 0.25% |
| | | 10% | 0.5% |
| *Flickr* | 100% | 0.1% | 0.1% |
| | | 0.5% | 0.5% |
| | | 1% | 1% |
| *Reddit* | 100% | 0.05% | 0.05% |
| | | 0.1% | 0.1% |
| | | 0.2% | 0.2% |
| Yelp | 100% | 0.05% | 0.05% |
| | | 0.1% | 0.1% |
| | | 0.2% | 0.2% |

### A.3 Implementation Details

Since the node selection of Random, KCenter, and Herding varies too much in each random seed, we run these three methods three times, and all the results in Table 1 represent the average performance. We conduct all the experiments on a cluster mixed with NVIDIA A100, V100, K80 and RTX3090 GPUs. Notably, GDEM can only be reproduced by RTX3090 with their provided eigendecomposition. We use Pytorch (modified BSD license) and PyG (Fey and Lenssen, 2019) (MIT license) to reproduce all those methods in a user-friendly and unified way.

### A.4 PERFORMANCE AND SCALABILITY

Table 9 provides the complete average accuracy with the standard deviation of 10 runs results. We also append two coreset selection baselines first introduced by Cao et al. (2024): **Cent-D** selects nodes based on their degree, prioritizing those with the highest connectivity. **Cent-P** (Langville and Meyer, 2004) selects nodes with high PageRank (Page et al., 1998) values, prioritizing those that are more central and influential in the graph structure. We also explore the potential of one traditional sparsification method called **TSpanner** (Liestman and Shermer, 1993) which only reduces the number of edges and preserves the shortest distance property. Note that due to the reproducibility challenges of GDEM on larger datasets in our experiments, we have focused on its performance with the three small datasets and have not included it in the main content.

Table 9: Test accuracy and standard error of each graph reduction method across different datasets and three representative reduction rates for each dataset. The best and second-best results, excluding the whole graph training results, are marked in **bold** and underlined, respectively.

| Dataset | Reduction rate (%) | *Coreset & Sparsification* | | | | | | *Coarsen* | | | *Condensation* | | | | | | | | | | Whole |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | *Structure-free* | | | | | *Structure-based* | | | | | |
| | | **Cent-D** | **Cent-P** | **Random** | **Herding** | **K-Center** | **TSpanner** | **Averaging** | **VN** | **VNG** | **GEOM** | **SFGC** | **GCSNTK** | **GCDMX** | **GCondX** | **GCond** | **DosCond** | **MSGC** | **SGDD** | **GDEM** | |
| *Citeseer* | 0.36 | 42.86 ± 2.7 | 37.78 ± 1.3 | 35.37 ± 2.8 | 43.73 ± 1.6 | 41.43 ± 1.4 | 71.83 ± 0.3 | 69.75 ± 0.6 | 34.32 ± 5.9 | 66.14 ± 0.3 | 67.61 ± 0.7 | 66.27 ± 0.8 | 63.51 ± 1.9 | 70.65 ± 0.5 | 67.79 ± 0.7 | 70.05 ± 2.1 | 69.41 ± 0.8 | 60.24 ± 6.0 | **71.87 ± 0.6** | 67.88 ± 1.8 | |
| | 0.90 | 58.77 ± 0.5 | 52.83 ± 0.4 | 50.71 ± 0.8 | 59.24 ± 0.4 | 51.15 ± 1.1 | 71.62 ± 0.4 | 69.59 ± 0.5 | 40.14 ± 5.3 | 66.07 ± 0.4 | 70.70 ± 0.5 | 70.27 ± 0.7 | 62.91 ± 0.8 | 71.27 ± 0.6 | 69.69 ± 0.5 | 69.15 ± 1.2 | 70.83 ± 0.4 | **72.08 ± 0.7** | 70.52 ± 0.6 | 70.13 ± 1.1 | 72.6 |
| | 1.80 | 62.89 ± 0.4 | 63.37 ± 0.4 | 62.62 ± 0.6 | 66.66 ± 0.5 | 59.04 ± 0.9 | 71.60 ± 0.4 | 69.50 ± 0.6 | 41.98 ± 7.0 | 65.34 ± 0.6 | **73.03 ± 0.3** | 72.36 ± 0.5 | 63.90 ± 3.4 | 72.08 ± 0.2 | 68.38 ± 0.5 | 69.35 ± 0.8 | 72.18 ± 0.6 | 72.21 ± 0.4 | 69.65 ± 1.7 | 71.74 ± 0.9 | |
| *Cora* | 0.50 | 57.79 ± 1.7 | 58.44 ± 1.7 | 35.14 ± 2.5 | 51.68 ± 2.1 | 44.64 ± 4.4 | 79.79 ± 0.4 | 75.94 ± 0.7 | 24.62 ± 5.7 | 70.40 ± 0.6 | 78.14 ± 0.5 | 75.11 ± 2.2 | 71.58 ± 0.9 | 79.21 ± 0.4 | 79.74 ± 0.5 | 80.17 ± 0.8 | **80.65 ± 0.6** | 80.54 ± 0.3 | 80.15 ± 0.5 | 54.76 ± 4.5 | |
| | 1.30 | 66.45 ± 2.2 | 66.38 ± 1.7 | 63.63 ± 1.3 | 68.99 ± 0.7 | 63.28 ± 1.4 | 80.84 ± 0.3 | 75.87 ± 0.6 | 51.07 ± 5.8 | 74.48 ± 0.5 | **82.29 ± 0.6** | 79.55 ± 0.3 | 71.22 ± 2.6 | 80.26 ± 0.3 | 78.67 ± 0.4 | 80.81 ± 0.5 | 80.85 ± 0.4 | 80.98 ± 0.5 | 80.29 ± 0.8 | 72.87 ± 1.8 | 81.5 |
| | 2.60 | 75.79 ± 0.7 | 75.64 ± 1.6 | 72.24 ± 0.6 | 73.77 ± 0.9 | 70.55 ± 1.4 | 80.41 ± 0.3 | 75.76 ± 1.1 | 56.75 ± 5.4 | 76.03 ± 0.4 | **82.82 ± 0.2** | 80.54 ± 0.5 | 73.34 ± 0.6 | 80.68 ± 0.3 | 78.60 ± 0.3 | 80.54 ± 0.7 | 81.15 ± 0.5 | 80.94 ± 0.4 | 81.04 ± 0.5 | 81.76 ± 0.5 | |
| *Pubmed* | 0.02 | 56.16 ± 2.6 | 57.28 ± 1.2 | 49.46 ± 1.6 | 62.91 ± 1.5 | **79.18 ± 0.2** | 62.91 ± 1.5 | 74.09 ± 0.6 | 75.60 ± 0.4 | 75.60 ± 0.4 | 69.64 ± 1.4 | 67.61 ± 2.0 | 29.45 ± 10.9 | 77.62 ± 0.2 | 72.03 ± 1.6 | 77.36 ± 0.7 | 58.13 ± 2.2 | 75.25 ± 0.7 | 78.11 ± 0.3 | 77.52 ± 0.7 | |
| | 0.03 | 55.61 ± 1.6 | 62.50 ± 1.0 | 56.10 ± 1.8 | 69.28 ± 1.6 | 65.59 ± 2.4 | **79.39 ± 0.3** | 75.60 ± 0.4 | 74.09 ± 0.6 | 75.72 ± 0.3 | 76.21 ± 0.7 | 66.89 ± 3.3 | 68.37 ± 3.0 | 76.63 ± 1.2 | 72.05 ± 1.6 | 78.05 ± 0.3 | 52.70 ± 0.3 | 78.26 ± 0.3 | 78.07 ± 0.3 | 78.05 ± 1.3 | 78.6 |
| | 0.15 | 71.95 ± 0.5 | 73.35 ± 0.4 | 71.84 ± 0.7 | 75.53 ± 0.4 | 74.00 ± 0.2 | 78.39 ± 0.2 | 75.60 ± 0.4 | 73.68 ± 1.6 | 77.53 ± 0.5 | 78.49 ± 0.2 | 67.61 ± 4.1 | 69.89 ± 2.2 | 77.48 ± 0.5 | 71.97 ± 0.5 | 76.46 ± 0.5 | 76.45 ± 0.1 | 78.20 ± 0.2 | 75.95 ± 0.3 | 78.76 ± 1.1 | |
| *Arxiv* | 0.05 | 32.88 ± 2.7 | 36.48 ± 2.0 | 50.39 ± 1.4 | 51.49 ± 0.7 | 50.52 ± 0.5 | - | 59.62 ± 0.4 | OOM | 54.89 ± 0.3 | **64.91 ± 0.4** | 64.91 ± 0.5 | 58.21 ± 1.7 | 60.04 ± 0.4 | 59.40 ± 0.5 | 60.49 ± 0.4 | 55.70 ± 0.3 | 57.66 ± 0.4 | 58.50 ± 0.2 | - | |
| | 0.25 | 48.85 ± 1.1 | 47.90 ± 0.9 | 58.92 ± 0.8 | 58.00 ± 0.5 | 55.28 ± 0.6 | - | 59.96 ± 0.3 | OOM | 59.66 ± 0.2 | **68.78 ± 0.1** | 66.58 ± 0.3 | 59.98 ± 1.7 | 60.59 ± 0.4 | 62.46 ± 0.3 | 63.88 ± 0.2 | 57.39 ± 0.2 | 64.85 ± 0.3 | 59.18 ± 0.2 | - | 71.4 |
| | 0.50 | 52.01 ± 0.5 | 55.65 ± 0.5 | 60.19 ± 0.5 | 57.70 ± 0.2 | 58.66 ± 0.4 | - | 59.94 ± 0.3 | OOM | 60.93 ± 0.2 | **69.59 ± 0.2** | 67.03 ± 0.5 | 54.73 ± 5.0 | 60.71 ± 0.7 | 59.93 ± 0.5 | 64.23 ± 0.2 | 61.06 ± 0.6 | 65.73 ± 0.2 | 63.76 ± 0.2 | - | |
| *Flickr* | 0.10 | 40.70 ± 0.4 | 40.97 ± 0.9 | 42.94 ± 0.3 | 42.80 ± 0.1 | 43.01 ± 0.5 | - | 37.93 ± 0.3 | 32.77 ± 5.7 | 44.33 ± 0.3 | **47.15 ± 0.1** | 46.38 ± 0.2 | 41.85 ± 3.1 | 43.75 ± 0.3 | 46.66 ± 0.1 | 46.75 ± 0.1 | 45.87 ± 0.3 | 46.21 ± 0.1 | 46.69 ± 0.1 | - | |
| | 0.50 | 42.90 ± 0.3 | 44.06 ± 0.3 | 44.54 ± 0.5 | 43.86 ± 0.5 | 43.46 ± 0.8 | - | 37.76 ± 0.4 | 33.79 ± 5.2 | 43.30 ± 0.6 | 46.71 ± 0.2 | 46.38 ± 0.2 | 33.39 ± 6.0 | 45.05 ± 0.3 | 46.69 ± 0.1 | **47.01 ± 0.2** | 45.89 ± 0.3 | 46.77 ± 0.1 | 46.39 ± 0.2 | - | 47.4 |
| | 1.00 | 42.62 ± 0.2 | 44.51 ± 0.3 | 44.68 ± 0.6 | 45.12 ± 0.4 | 43.53 ± 0.6 | - | 37.66 ± 0.3 | 34.39 ± 6.0 | 43.84 ± 0.8 | 46.13 ± 0.2 | 46.61 ± 0.1 | 31.12 ± 4.2 | 45.88 ± 0.1 | 46.58 ± 0.1 | **46.99 ± 0.1** | 45.81 ± 0.1 | 46.12 ± 0.2 | 46.24 ± 0.3 | - | |
| *Reddit* | 0.05 | 40.00 ± 1.1 | 45.83 ± 1.7 | 40.13 ± 0.9 | 46.88 ± 0.4 | 40.24 ± 0.8 | - | 88.23 ± 0.1 | OOM | 69.96 ± 0.5 | **90.63 ± 0.2** | 90.18 ± 0.2 | OOM | 87.28 ± 0.2 | 86.56 ± 0.2 | 85.39 ± 0.2 | 86.56 ± 0.4 | 87.62 ± 0.1 | 87.37 ± 0.2 | - | |
| | 0.10 | 50.47 ± 1.4 | 51.22 ± 1.4 | 55.73 ± 0.5 | 59.34 ± 0.7 | 48.28 ± 0.7 | - | 88.32 ± 0.1 | OOM | 76.95 ± 0.2 | **91.33 ± 0.1** | 89.84 ± 0.3 | OOM | 89.96 ± 0.1 | 88.25 ± 0.3 | 89.82 ± 0.1 | 88.32 ± 0.2 | 88.15 ± 0.1 | 88.73 ± 0.3 | - | 94.4 |
| | 0.20 | 55.31 ± 1.8 | 61.56 ± 0.2 | 58.39 ± 2.3 | 73.46 ± 0.5 | 56.81 ± 1.7 | - | 88.33 ± 0.1 | OOM | 81.52 ± 0.6 | **91.03 ± 0.3** | 90.71 ± 0.1 | OOM | 89.08 ± 0.1 | 88.73 ± 0.2 | 90.42 ± 0.1 | 88.84 ± 0.2 | 87.03 ± 0.1 | 90.65 ± 0.1 | - | |
| *Yelp* | 0.05 | 48.67 ± 0.3 | 46.81 ± 0.1 | 46.08 ± 0.0 | 46.08 ± 0.0 | 46.07 ± 0.0 | - | **55.04 ± 0.1** | 51.52 ± 1.6 | 49.24 ± 0.1 | 52.80 ± 2.2 | 46.20 ± 0.1 | OOM | 50.75 ± 0.4 | 52.44 ± 0.4 | 52.30 ± 0.1 | 51.10 ± 0.3 | 52.94 ± 0.2 | 52.02 ± 0.2 | - | |
| | 0.10 | 51.03 ± 0.1 | 46.08 ± 0.0 | 46.28 ± 0.1 | 52.23 ± 0.3 | 46.22 ± 0.0 | - | **53.51 ± 0.8** | 51.68 ± 1.0 | 47.33 ± 0.5 | 47.56 ± 0.2 | 47.96 ± 0.0 | OOM | 52.49 ± 0.1 | 49.70 ± 1.5 | 53.22 ± 0.1 | 52.54 ± 0.1 | 50.97 ± 0.8 | 54.13 ± 0.2 | - | 58.2 |
| | 0.20 | 46.08 ± 0.0 | 46.08 ± 0.0 | 49.31 ± 0.4 | 47.49 ± 0.1 | 46.85 ± 0.2 | - | 54.42 ± 0.3 | 52.63 ± 1.1 | 48.63 ± 0.4 | 49.48 ± 0.7 | 46.70 ± 0.1 | OOM | **55.89 ± 0.2** | 48.77 ± 1.3 | 51.76 ± 0.2 | 52.19 ± 0.5 | 51.35 ± 0.5 | 52.86 ± 0.1 | - | |

Table 10: Experiment results under **hyperparameter searching**. The search space is shown in Table 11. The best results, excluding the whole graph training results, are marked in **bold**.

| Dataset | Reduction rate (%) | *Coreset & Sparsification* | | *Coarsen* | | *Condensation* | | | | | | Whole |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | *Structure-free* | | | *Structure-based* | | | |
| | | **Random** | **K-Center** | **Averaging** | **VNG** | **GEOM** | **SFGC** | **GCondX** | **GCond** | **DosCond** | **SGDD** | |
| *Citeseer* | 0.36 | 37.67 ± 2.45 | 45.11 ± 2.19 | 69.97 ± 0.36 | 64.37 ± 1.29 | 68.90 ± 0.64 | 66.96 ± 1.47 | 68.29 ± 1.30 | **73.63 ± 0.32** | 69.53 ± 0.65 | 71.90 ± 0.24 | |
| | 0.90 | 47.13 ± 1.32 | 55.09 ± 1.14 | 69.97 ± 0.36 | 69.37 ± 0.62 | **73.20 ± 0.35** | 70.66 ± 0.23 | 69.73 ± 0.46 | 70.93 ± 0.51 | 70.97 ± 0.29 | 70.10 ± 0.73 | 72.6 |
| | 1.80 | 64.21 ± 0.72 | 62.82 ± 0.78 | 70.01 ± 0.27 | 69.35 ± 0.70 | **74.36 ± 0.30** | 72.37 ± 0.41 | 69.19 ± 0.47 | 70.69 ± 0.47 | 72.73 ± 0.35 | 70.11 ± 0.93 | |
| *Cora* | 0.50 | 47.93 ± 0.96 | 49.92 ± 3.06 | 76.55 ± 0.91 | 70.61 ± 0.64 | 79.03 ± 0.61 | 76.80 ± 2.18 | 80.04 ± 0.60 | **80.63 ± 0.48** | 80.43 ± 0.72 | 81.58 ± 0.97 | |
| | 1.30 | 69.54 ± 2.60 | 63.16 ± 1.37 | 76.99 ± 0.67 | 75.72 ± 0.21 | **83.10 ± 0.41** | 80.03 ± 0.61 | 79.22 ± 0.27 | 81.01 ± 0.50 | 81.19 ± 0.34 | 81.24 ± 0.79 | 81.81 |
| | 2.60 | 71.70 ± 1.92 | 72.02 ± 1.21 | 76.41 ± 1.47 | 77.19 ± 0.52 | **83.50 ± 0.43** | 81.64 ± 0.53 | 78.98 ± 0.31 | 81.45 ± 0.46 | 81.06 ± 0.53 | 79.80 ± 0.85 | |
| *Arxiv* | 0.05 | 50.29 ± 1.33 | 49.20 ± 0.35 | 59.59 ± 0.38 | 55.36 ± 0.45 | 64.27 ± 0.12 | **65.07 ± 0.49** | 59.63 ± 0.37 | 55.83 ± 0.68 | 56.74 ± 0.36 | 59.13 ± 0.45 | |
| | 0.25 | 59.26 ± 0.45 | 58.05 ± 0.44 | 59.94 ± 0.32 | 61.27 ± 0.19 | **68.75 ± 0.10** | 66.63 ± 0.28 | 62.43 ± 0.31 | 64.79 ± 0.27 | 57.56 ± 0.22 | 56.86 ± 0.42 | 71.22 |
| | 0.50 | 62.49 ± 0.75 | 60.77 ± 0.37 | 59.93 ± 0.29 | 64.78 ± 0.13 | **69.63 ± 0.16** | 67.43 ± 0.29 | 60.17 ± 0.54 | 64.83 ± 0.24 | 61.26 ± 0.45 | 61.15 ± 0.20 | |
| *Flickr* | 0.10 | 43.07 ± 0.56 | 42.68 ± 0.68 | 44.48 ± 0.64 | 46.14 ± 0.30 | **47.14 ± 0.11** | 46.93 ± 0.25 | 46.74 ± 0.12 | 46.63 ± 0.11 | 45.92 ± 0.19 | 46.79 ± 0.14 | |
| | 0.50 | 44.86 ± 0.16 | 44.30 ± 0.38 | 44.35 ± 0.79 | 43.23 ± 0.40 | 47.01 ± 0.17 | **47.22 ± 0.15** | 46.76 ± 0.10 | 47.13 ± 0.14 | 46.20 ± 0.18 | 46.38 ± 0.15 | 47.4 |
| | 1.00 | 45.63 ± 0.24 | 44.70 ± 0.47 | 44.38 ± 0.78 | 43.97 ± 0.52 | 46.93 ± 0.24 | **47.02 ± 0.09** | 46.63 ± 0.16 | 46.74 ± 0.15 | 46.55 ± 0.14 | 46.54 ± 0.08 | |
| *Reddit* | 0.05 | 40.32 ± 1.20 | 43.52 ± 2.04 | 88.65 ± 0.15 | 71.34 ± 0.34 | **91.42 ± 0.08** | 90.18 ± 0.14 | 86.92 ± 0.26 | 86.53 ± 0.21 | 86.66 ± 0.15 | 87.71 ± 0.20 | |
| | 0.10 | 56.37 ± 2.05 | 48.97 ± 2.72 | 88.66 ± 0.15 | 84.62 ± 0.23 | **91.57 ± 0.04** | 89.88 ± 0.19 | 88.37 ± 0.35 | 87.81 ± 0.22 | 88.44 ± 0.15 | 88.88 ± 0.25 | 93.95 |
| | 0.20 | 63.56 ± 1.08 | 56.27 ± 2.99 | 88.60 ± 0.34 | 89.03 ± 0.14 | **91.57 ± 0.09** | 90.79 ± 0.09 | 88.99 ± 0.28 | 89.80 ± 0.13 | 88.96 ± 0.23 | 90.66 ± 0.09 | |
| ***# Wins after tune*** | | 0 | 0 | 0 | 0 | 10 | 3 | 0 | 2 | 0 | 0 | |
| ***# Wins before tune*** | | 0 | 1 | 0 | 0 | 10 | 0 | 0 | 2 | 1 | 1 | |

Figure 6 and Figure 7 illustrate the scalability of *structure-free* and *structure-based* GC methods across two datasets. The number of epochs is a hyperparameter for each method. To ensure a fair comparison, we also record the epoch time for each method. **First**, structure-free GC methods are more efficient than structure-based ones, as they generally require less epoch time. **Second**, different hyperparameter settings result in varying time costs across datasets. For instance, GEOM employs soft labels to train GNNs on *Cora*, which significantly increases the time cost. **Third**, as the reduction rate increases, the performance and time costs do not necessarily rise.

### A.4.1 Details description for Test accuracy vs. total time Figure

Figure 1 compares test accuracy (y-axis) and total time (x-axis) for various graph condensation methods applied to the Arxiv dataset. The methods are distinguished by different marker shapes and colors: blue stars represent structure-free methods, red circles represent structure-based methods, and green triangles represent distribution-based methods. The size of each marker indicates the reduction rate, with smaller markers representing a reduction rate of 0.05%, medium markers 0.25%, and larger markers 0.50%. Dashed lines connect markers corresponding to the same method across different reduction rates, illustrating the method's behavior under varying levels of graph condensation. To enhance clarity, the name of each method will be positioned near the marker for its respective curve, ensuring easy identification of methods and their corresponding performance trends.

### A.4.2 Further analysis of experimental results

- **Factors Affecting Performance in Arxiv and Reddit.** We assume that the imbalanced label distributions in these two datasets are the factors for the performance. Arxiv and Reddit datasets have a larger number of classes and exhibit significant class imbalance compared to others. Consistent with most GC works, our implementation ensures at least one instance per class, guaranteeing representation for each class. However, this approach can cause distribution shifts. In contrast, datasets like Cora, Citeseer, and Pubmed have more balanced training sets, leading to more stable performance. This observation highlights the need for improved initialization methods in the GC field to effectively handle datasets with numerous and imbalanced classes.
- **Why Averaging Achieves the Best Performance on Yelp.** This performance difference can be attributed to the characteristics of the *Yelp* dataset, which is designed for anomaly detection and evaluated using the F1-macro score. Averaging methods rely only on the average representations of normal instances and anomalies, resulting in a simple decision boundary that aligns well with the dataset's requirements. In contrast, GC methods may struggle due to unbalanced class initialization, often leading to overfitted decision boundaries for anomalies.

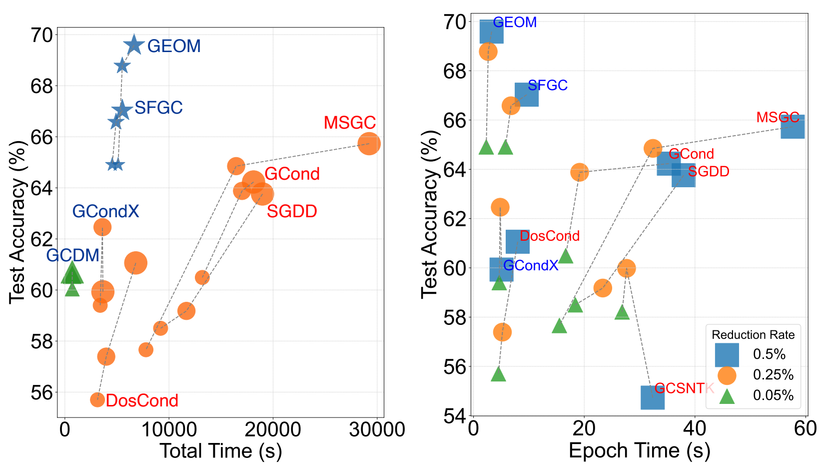



Figure 6: Performance vs. Total Time and Epoch Time on *Arxiv*.

### A.5 Privacy Preservation

We focused on a fundamental privacy attack, confidence-based membership inference attack (MIA), for the following reasons:

We are not merely benchmarking the privacy-preserving properties of existing GC methods but are also **broadening the scope of GC research** to encompass critical areas such as privacy and robustness. This expansion aims to demonstrate the potential of GC methods, inspiring **more researchers to recognize their promise and contribute to this emerging field**. Since existing applications of GC predominantly target Neural Architecture Search (NAS) (Jin et al., 2022a; Ding et al., 2022) and

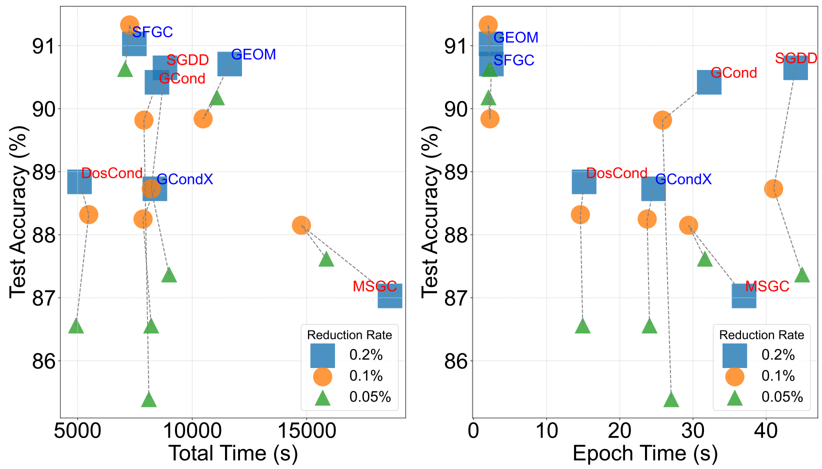



Figure 7: Performance vs. Total Time and Epoch Time on *Reddit*.

continual learning (Liu et al., 2023c), we aim to shift the conversation by highlighting their broader applicability.

To the best of our knowledge, no prior work has empirically validated the privacy-preserving claims associated with GC. By targeting one of the most fundamental and well-studied privacy attacks, MIA, our work provides essential, empirical evidence for assessing and understanding the privacy capabilities of GC. This serves as a **preliminary yet foundational step** toward establishing a systematic and rigorous framework for evaluating the privacy guarantees of GC methods. We have chosen to omit additional privacy attacks for the following reasons:

- **Model Inversion Attack (MIvA)** (Zhang et al., 2024b): MIvA aims to reconstruct the original graph and assess attack performance via link prediction tasks. In the context of GC, the condensation process significantly reduces the number of nodes and reindexes all synthetic nodes. This reduction diminishes the granularity necessary for accurate link reconstruction, making it difficult for an attacker to determine specific node connections. Additionally, reindexing disrupts any direct correspondence between condensed and original nodes, further obfuscating the true link structure. Instead, we evaluate graph properties in Section 4.8, demonstrating that condensation alters most graph properties. This suggests that the privacy of graph properties is maintained through the condensed graph.
- **Attribute Inversion Attack (AIA)** (Zhang et al., 2022): AIA typically requires datasets with sensitive attributes, which diverges from the standard datasets in mainstream GNN studies (Zhang et al., 2022; Gong and Liu, 2018). As a benchmark requiring unifying all baseline methods and datasets, Incorporating AIA would thus fall outside the scope of our current work.

We believe that our focused approach provides an essential first step toward understanding the privacy implications of GC methods. We plan to explore additional attack scenarios in future work to further validate and extend our findings.

### A.6 Transferability

#### A.6.1 Hyperparameters Searching

For fair evaluation between different architectures, we conduct hyperparameter searching while training each architecture on the condensed graph. We select the best hyperparameter combinations based on validation results and report corresponding testing results. The search space of hyperparameters for each GNN is as follows: Number of hidden units is selected from {64, 256}, learning rate is chosen from {0.01, 0.001}, weight decay is 0 or 5e-4, dropout rate is 0 or 0.5. For GAT, since we fix the number of attention heads to 8, to avoid OOM, the number of hidden units is selected from {16, 64} and the search space of dropout rate is in {0.0, 0.5, 0.7}. Additionally, for SGC and APPNP, we also explore the number of linear layers in {1, 2}. For APPNP, we further search for alpha in {0.1, 0.2}.

### A.6.2 Hyperparameters Sensitivity Analysis

Figure 8: Hyperparameters Sensitivity Analysis on **Condensed Graphs**.

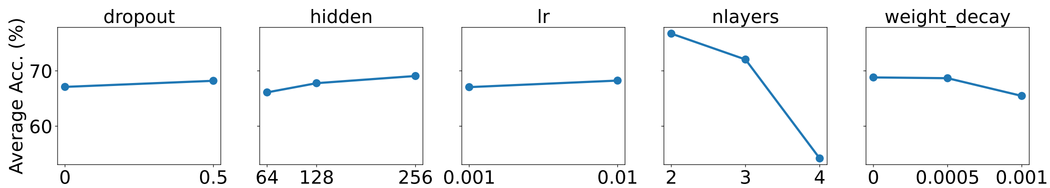



Figure 9: **Cora** Dataset.

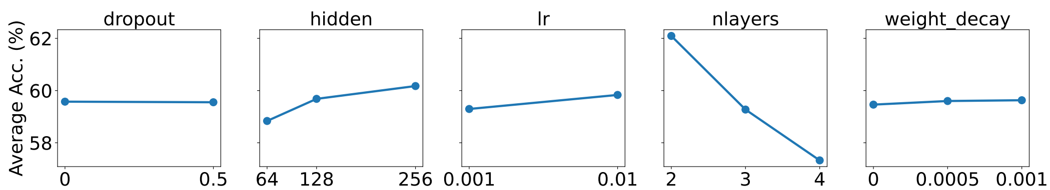



Figure 10: **Ogbn-arxiv** Dataset.

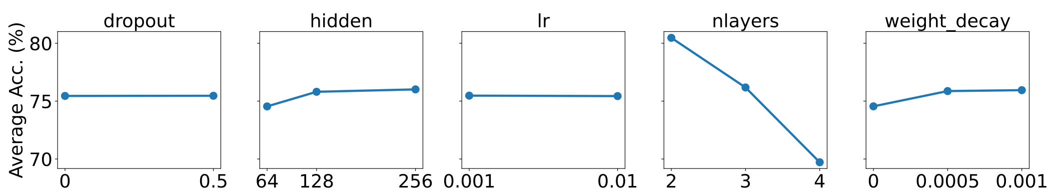



Figure 11: **Reddit** Dataset.

To provide additional insights on how varying hyperparameters affect the performance of the GNN model (e.g. GCN) trained on the whole or the condensed graphs, we further expand the search space of hyperparameters for GCN as shown in Table 11. The hyperparameter searching results for each method are shown in Table 10. We compare the winning times differences before and after tuning, which shows that GC methods that perform better in the main table generally maintain superior performance after hyperparameter tuning. Notably, methods like GEOM and GCond continue to outperform others post-tuning, reinforcing the robustness of our initial fixed hyperparameter choices.

Table 11: Hyperparameter Search Space for Sensitivity Analysis

| **Hyperparameter** | **Values** |
|---|---|
| Number of hidden units | {64, 128, 256} |
| Learning rate | {0.01, 0.001} |
| Number of layers | {2, 3, 4} |
| Weight decay | {0, 0.0005, 0.001} |
| Dropout rate | {0, 0.5} |

Figure 8 and 12 These figures show that condensed and whole graphs exhibit similar sensitivity patterns across the Cora, Ogbn-arxiv, and Reddit datasets, suggesting a consistent response to hyperparameter tuning.

Figure 12: Hyperparameters Sensitivity Analysis on **Whole Graphs**.

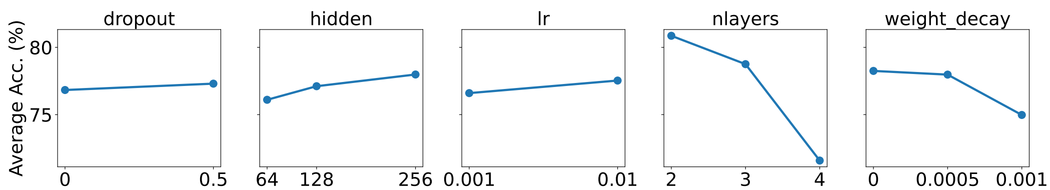



Figure 13: **Cora** Dataset.

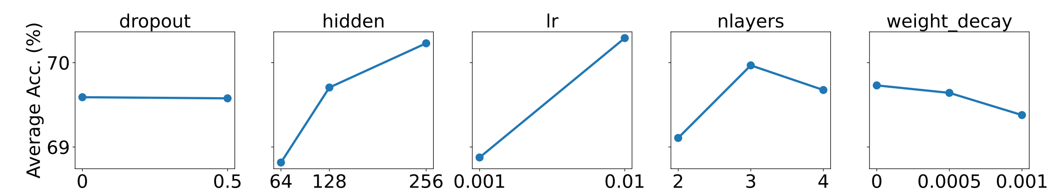



Figure 14: **Ogbn-arxiv** Dataset.

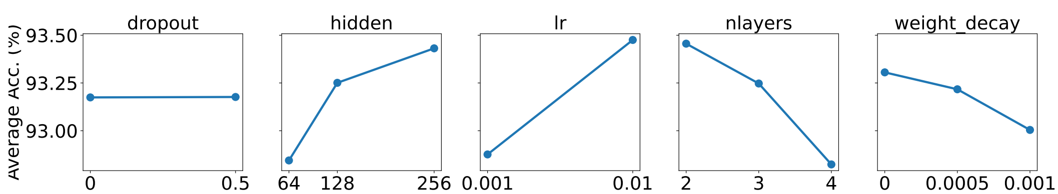



Figure 15: **Reddit** Dataset.

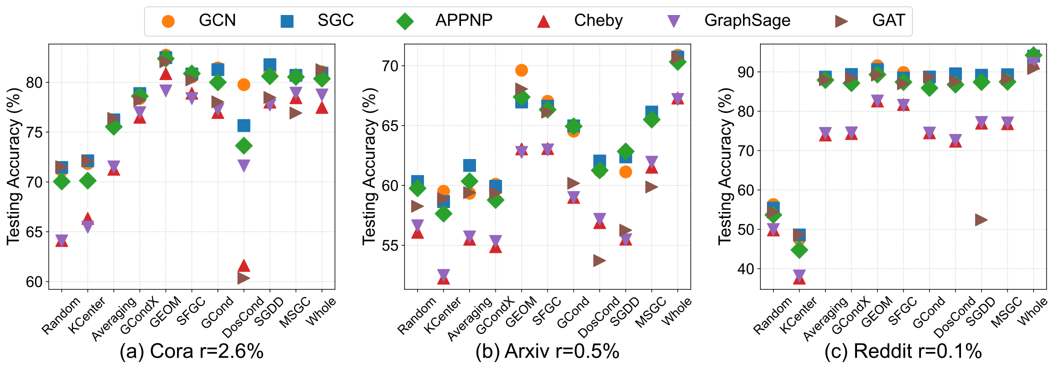



Figure 16: Performance of condensed graphs evaluated by different GNNs.

- Both condensed and whole graphs show low sensitivity to **dropout and weight decay**, with minimal variations in accuracy, indicating these hyperparameters have a limited impact on performance.
- The **hidden layer size** positively influences accuracy in both condensed and whole graphs, with larger sizes generally improving performance, highlighting the importance of hidden layer capacity in model effectiveness.
- **Learning rate** sensitivity is also comparable between condensed and whole graphs; a higher learning rate (0.01) tends to perform better in both cases, though with slight dataset-specific variation (i.e. whole graph of Ogbn-arxiv).
- Notably, the **number of layers** impacts both graph types similarly, as accuracy consistently declines with an increase in layers, suggesting that deeper architectures do not benefit either condensed or whole graphs in three datasets.

Thus, condensed and whole graphs have parallel sensitivity trends, where optimizing hidden layer size and learning rate while managing network depth is likely to enhance performance across both representations.

#### A.6.3 Relative and Absolute Accuracy

We calculate the relative accuracy by dividing the results of the model trained on the condensed graph by the results of the same model trained on the whole graph. For example, the accuracy of GCN on the GCond condensed graph is divided by the accuracy of results on the whole graph. Since Figure 4 in the main content shows the relative accuracy, we show the absolute results of each GNN here in Figure 16.

#### A.6.4 Evaluate Condensed Graph by Graph Transformer

The architectures discussed in the main content primarily utilize message-passing styles, which facilitate their transfer to each other. However, they may encounter challenges when applied to an entirely different architecture. Therefore, to conduct a more comprehensive evaluation of transferability, we assess the performance of various condensation methods using a graph transformer-based architecture **SGFormer** (Wu et al., 2023), which is totally different from those message-passing architectures. Figure **??** shows that SGFormer achieves comparable performance with other architectures on three non-GNN methods (Random, KCenter, Averaging). However, its performance significantly drops when trained on graphs condensed by GNN-involved methods. This suggests that future research should explore the transferability of other graph learning architectures.

### A.7 Neural Architecture Search

We utilize APPNP (Gasteiger et al., 2018) for NAS experiments because its architecture modules are flexible and can be easily modified. The detailed architecture search space is shown in Table 12. Following the settings in GCond (Jin et al., 2022a), we search full combinations of these choices, i.e. 480 in total for each dataset.

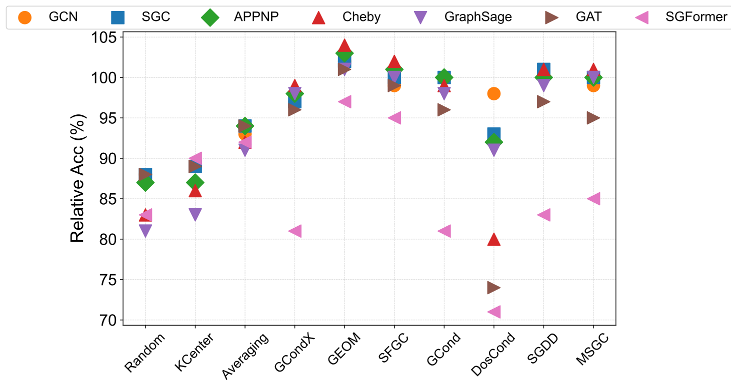



Figure 17: Condensed graph performance evaluated using different models including **SGFormer** on *Cora*.

Table 12: Architecture search space for APPNP.

| Architecture | Search Space |
|---|---|
| Number of propagation $K$ | {2, 4, 6, 8, 10} |
| Residual coefficient $\alpha$ | {0.1, 0.2} |
| Hidden dimension | {16, 32, 64, 128, 256, 512} |
| Activation function | {Sigmoid, Tanh, ReLU, Linear, Softplus, LeakyReLU, ReLU6, ELU} |

### A.8 Graph property preservation

The full results on graph property preservation are listed in Table 13. As we mention in the main content, different GC methods show totally different behavior w.r.t. property preservation. **First**, VNG and SGDD tend to produce almost complete graphs linking each node pair. That also leads to a lower homophily, as they create more proportion of inter-class connections. **Second**, VNG performs best in property preservation, however, it shows suboptimal accuracy in Table 9. This suggests that the selected graph properties are unnecessary to maintain or to preserve as much as possible. **Third**, as the only method that creates sparse graphs, MSGC is unique among these methods except in the Homophily. From this point of view, we hold that homophily is very important for future research on *structure-based* GC since all structure-based methods behave consistently. Current research mostly holds the view that the loss of homophily is harmful (Luan et al., 2021), but our benchmark may provide a contradictory perspective on this.

Notably, we observed that MSGC preserves the maximum eigenvalue up to 0.94. As further evidence, the latest method, GDEM (Liu et al., 2023b), focuses on learning to preserve eigenvectors, supporting the idea that maintaining spectral properties may be beneficial. However, upon closer examination of the properties of the graph synthesized by GDEM, as shown in Table 15, we find that these properties are not fully preserved. This is because their method only retains eigenvalues within a middle range, specifically from $K_1$ to $K_2$. This suggests that methods for accurately preserving spectral properties remain an area for further exploration.

Since only the metric DBI does not rely on structure, we also exhibit the correlation of DBI of structure-free methods across all five datasets in Table 14. From the comparison between structure-free and structure-based methods, we find that GCondX and GEOM also preserve this correlation of DBI to some extent, similar to structure-based methods.

Table 13: Graph properties in condensed graphs from different *structure-based* GC methods. The "**Corr.**" row shows the correlation of certain properties between the condensed graph and the whole graph across five datasets.

| Graph property | Dataset and $r$ | **VNG** | **GCond** | **MSGC** | **SGDD** | **Avg.** | **Whole** |
|---|---|---|---|---|---|---|---|
| Density% (Structure) | *Citeseer 1.8%* | 36.95 | 84.58 | 22.50 | 100.00 | 61.01 | 0.08 |
| | *Cora 2.6%* | 52.17 | 82.28 | 22.00 | 100.00 | 64.11 | 0.14 |
| | *Arxiv 0.5%* | 100.00 | 75.40 | 8.17 | 99.91 | 70.87 | 0.01 |
| | *Flickr 1%* | 100.00 | 100.00 | 3.44 | 99.96 | 75.85 | 0.01 |
| | *Reddit 0.1%* | 100.00 | 2.67 | 32.07 | 74.85 | 52.39 | 0.05 |
| | **Corr.** | -0.81 | 0.07 | 0.55 | 0.13 | -0.01 | - |
| Max Eigenvalue (Spectra) | *Citeseer 1.8%* | 2.98 | 22.53 | 1.67 | 10.29 | 9.37 | 100.04 |
| | *Cora 2.6%* | 3.73 | 34.90 | 1.69 | 14.09 | 13.60 | 169.01 |
| | *Arxiv 0.5%* | 2,092.99 | 163.95 | 2.33 | 79.95 | 584.81 | 13,161.87 |
| | *Flickr 1%* | 1,133.94 | 281.04 | 1.76 | 123.86 | 385.15 | 930.01 |
| | *Reddit 0.1%* | 1,120.64 | 152.00 | 2.00 | 99.84 | 343.62 | 2,503.07 |
| | **Corr.** | 0.85 | 0.25 | 0.95 | 0.28 | 0.58 | - |
| DBI (Label & Feature) | *Citeseer 1.8%* | 4.14 | 1.40 | 1.98 | 3.47 | 2.75 | 12.07 |
| | *Cora 2.6%* | 3.69 | 1.84 | 0.70 | 4.34 | 2.64 | 9.28 |
| | *Arxiv 0.5%* | 2.27 | 2.62 | 2.49 | 2.80 | 2.55 | 7.12 |
| | *Flickr 1%* | 5.60 | 7.14 | 7.33 | 13.57 | 8.41 | 31.02 |
| | *Reddit 0.1%* | 1.51 | 2.16 | 1.49 | 1.53 | 1.67 | 9.59 |
| | **Corr.** | 0.81 | 0.93 | 0.94 | 0.97 | <u>0.91</u> | - |
| DBI-AGG (Label & Feature & Structure) | *Citeseer 1.8%* | 4.11 | 0.76 | 1.75 | 0.00 | 1.66 | 8.49 |
| | *Cora 2.6%* | 3.59 | 0.38 | 0.57 | 0.18 | 1.18 | 4.67 |
| | *Arxiv 0.5%* | 2.38 | 2.86 | 2.61 | 1.77 | 2.41 | 4.40 |
| | *Flickr 1%* | 20.26 | 11.60 | 7.90 | 6.51 | 11.57 | 25.61 |
| | *Reddit 0.1%* | 1.56 | 1.90 | 1.49 | 1.37 | 1.58 | 2.48 |
| | **Corr.** | 0.99 | 0.93 | 0.95 | 0.89 | **0.94** | - |
| Homophily (Label & Structure) | *Citeseer 1.8%* | 0.18 | 0.18 | 0.23 | 0.15 | 0.18 | 0.74 |
| | *Cora 2.6%* | 0.14 | 0.16 | 0.19 | 0.13 | 0.16 | 0.81 |
| | *Arxiv 0.5%* | 0.08 | 0.07 | 0.04 | 0.07 | 0.07 | 0.65 |
| | *Flickr 1%* | 0.34 | 0.27 | 0.27 | 0.27 | 0.29 | 0.33 |
| | *Reddit 0.1%* | 0.04 | 0.04 | 0.04 | 0.07 | 0.05 | 0.78 |
| | **Corr.** | -0.83 | -0.68 | -0.46 | -0.80 | -0.69 | - |

Table 14: DBI in condensed graphs from both *structure-based* and *structure-free* GC methods, continued from Table 13.

| Datasets | **VNG** | **GCond** | **MSGC** | **SGDD** | **GCondX** | **GEOM** | **Avg.** | **Whole** |
|---|---|---|---|---|---|---|---|---|
| *Citeseer 1.8%* | 4.14 | 1.40 | 1.98 | 3.47 | 2.90 | 2.55 | 2.74 | 12.07 |
| *Cora 2.6%* | 3.69 | 1.84 | 0.70 | 4.34 | 2.18 | 3.16 | 2.65 | 9.28 |
| *Arxiv 0.5%* | 2.27 | 2.62 | 2.49 | 2.80 | 5.52 | 4.37 | 3.35 | 7.12 |
| *Flickr 1%* | 5.60 | 7.14 | 7.33 | 13.57 | 22.93 | 6.04 | 10.43 | 31.02 |
| *Reddit 0.1%* | 1.51 | 2.16 | 1.49 | 1.53 | 0.57 | 2.96 | 1.70 | 9.59 |
| **Corr.** | 0.81 | 0.93 | 0.94 | 0.97 | 0.95 | 0.78 | 0.90 | - |

Table 15: Property preservation check for GDEM, a method explicitly preserve the graph property.

| Dataset | Density % | Max Eigenvalue | DBI AGG | Homophily |
|---|---|---|---|---|
| *Cora* | 14.82 | 1.57 | 1.09 | 0.33 |
| Whole | 0.14 | 169.01 | 4.67 | 0.81 |
| *Citeseer* | 11.86 | 1.51 | 1.46 | 0.33 |
| Whole | 0.08 | 100.04 | 8.49 | 0.74 |
| *Pubmed* | 6.90 | 0.02 | 1.36 | 1.00 |
| Whole | 0.02 | 172.16 | 5.01 | 0.80 |

Table 16: Denoising effects of selected methods. "Perf. Drop" shows the relative loss of accuracy compared to the original results of each method before being corrupted. The best results are in **bold** and results that outperform whole dataset training are underlined. *Structure-free* and *structure-based* methods are colored as blue and red.

| Dataset | Method | Feature Noise | | Structural Noise | | Adversarial Structural Noise | |
|---|---|---|---|---|---|---|---|
| | | Test Acc. ↑ | Perf. Drop ↓ | Test Acc. ↑ | Perf. Drop ↓ | Test Acc. ↑ | Perf. Drop ↓ |
| *Citeseer 1.8%* (Poisoning & Evasion) | Whole | 64.07 | 11.75% | 57.63 | 20.62% | 53.90 | 25.76% |
| | Random | 56.91 | 9.11% | 61.56 | **1.69%** | 59.42 | 5.12% |
| | KCenter | 52.80 | 10.57% | 55.41 | 6.15% | 55.07 | 6.73% |
| | GCond | **64.06** | **7.63%** | **65.64** | 5.35% | **66.19** | **4.55%** |
| | GCondX | 61.27 | 10.40% | 60.42 | 11.65% | 60.75 | 11.15% |
| | GEOM | 58.77 | 19.53% | 51.41 | 29.60% | 57.94 | 20.67% |
| *Cora 2.6%* (Poisoning & Evasion) | Whole | 74.77 | 8.26% | 72.13 | 11.49% | 66.63 | 18.24% |
| | Random | 59.89 | 17.10% | 62.64 | 13.28% | 65.33 | 9.57% |
| | KCenter | 59.88 | 15.13% | 62.94 | **10.79%** | 65.51 | **7.14%** |
| | GCond | 67.62 | 16.04% | 63.14 | 21.61% | 68.90 | 14.45% |
| | GCondX | **67.72** | **13.85%** | **63.95** | 18.63% | **69.24** | 11.91% |
| | GEOM | 49.68 | 40.01% | 53.59 | 35.29% | 66.32 | 19.93% |
| *Flickr 1%* (Poisoning) | Whole | 46.68 | 1.51% | 42.60 | 10.13% | 44.44 | 6.24% |
| | Random | 44.33 | **0.78%** | 43.28 | 3.13% | 43.93 | **1.69%** |
| | KCenter | 43.15 | 0.88% | 42.36 | 2.68% | 42.21 | 3.03% |
| | GCond | **46.29** | 1.49% | **46.97** | **0.04%** | 43.90 | 6.58% |
| | GCondX | 45.60 | 2.11% | 46.19 | 0.83% | 42.00 | 9.83% |
| | GEOM | 45.38 | 1.63% | 45.52 | 1.32% | **44.72** | 3.06% |

### A.9 Denoising effects

All corruptions are implemented by a library for attack and defense methods on graphs, DeepRobust (Li et al., 2020). The full results on denoising effects are in Table 16. Apart from GC methods, we also add coreset selection methods as baselines. Results show that the simple baseline, Random, contains a certain level of denoising effects in terms of performance drop in *Citeseer* and *Flickr*. Meanwhile, KCenter exhibits the lowest performance drop in *Cora* corrupted by structural noise and adversarial structural attack. However, these phenomena do not necessarily mean they can defend the attack as the performance of these two methods before being corrupted is worse than GC methods. In contrast, the GC methods naturally outperform whole graph training in most scenarios, even though they are not specifically designed for defense.

### A.10 Code Availablity and Usage

We have developed an easy-to-use code package, which is included in the supplementary material and has been open-sourced as a PyTorch library. The package accepts graphs in the PyG (PyTorch Geometric) format as input and outputs a reduced graph that preserves the properties or performance of the original graph. Below, we provide technical details on how users can integrate new datasets, implement their own methods, propose new settings, and address potential difficulties.

#### A.10.1 Usage

```
from graphslim.dataset import *
from graphslim.evaluation import *
from graphslim.condensation import GCond
from graphslim.config import cli

args = cli(standalone_mode=False)
# Customize arguments here
args.reduction_rate = 0.5
args.device = 'cuda:0'
# Add more args.<main_args/dataset_args> as needed

graph = get_dataset('cora', args=args)
```

```
# To reproduce the benchmark, use our args and graph class
# To use your own args and graph format, ensure the args and graph class
    have the required attributes

# Create an agent for the reduction algorithm
# Add more args.<agent_args> as needed
agent = GCond(setting='trans', data=graph, args=args)

# Reduce the graph
reduced_graph = agent.reduce(graph, verbose=True)

# Create an evaluator
# Add more args.<evaluator_args> as needed
evaluator = Evaluator(args)

# Evaluate the reduced graph on a GNN model
res_mean, res_std = evaluator.evaluate(reduced_graph, model_type='GCN')
```

Listing 1: Code Example for Using the Benchmark Package

### A.10.2 Parameters Categorization

<main_args>: `dataset`, `method`, `setting`, `reduction_rate`, `seed`, `aggpreprocess`, `eval_whole`, `run_reduction`

<attack_args>: `attack`, `ptb_r`

<dataset_args>: `pre_norm`, `save_path`, `split`, `threshold`

<agent_args>: `init`, `eval_interval`, `eval_epochs`, `eval_model`, `condense_model`, `epochs`, `lr`, `weight_decay`, `outer_loop`, `inner_loop`, `nlayers`, `method`, `activation`, `dropout`, `ntrans`, `with_bn`, `no_buff`, `batch_adj`, `alpha`, `mx_size`, `dis_metric`, `lr_adj`, `lr_feat`

<evaluator_args>: `final_eval_model`, `eval_epochs`, `lr`, `weight_decay`

### A.10.3 Customization

**Adding a New Dataset:** To implement a new dataset, create a new class in `dataset/loader.py` and inherit from the `TransAndInd` class.

**Implementing a New Reduction Algorithm:** To add a new reduction algorithm, create a new class in `sparsification`, `coarsening`, or `condensation`, and inherit from the `Base` class.

**Adding a New Evaluation Metric:** To implement a new evaluation metric, create a new function in `evaluation/eval_agent.py`.

**Implementing a New GNN Model:** To add a new GNN model, create a new class in `models` and inherit from the `Base` class.

### A.10.4 Potential Difficulties

Users may encounter the following challenges:

**Disk Space Limitations:**

- Some methods store training trajectories of multiple experts, which can exceed 100 GB.
- *Solution:* Reduce the number of experts using the <method_name>.reduce() module to manage disk space.

**Memory and GPU Constraints:**

- Larger datasets might cause memory or GPU limitations during the condensation process.
- *Solution:* Load data and adjust the reduction process to run in a mini-batch manner to reduce memory usage.

**Hyperparameter Adjustment:**

- Tuning hyperparameters may be necessary for optimal performance.
- *Solution:* Modify the JSON configuration files in the `configs` folder, which contain all hyperparameters for each method.

We believe this information will help users effectively utilize, customize, and integrate our benchmark package with new datasets or algorithms. We provide comprehensive documentation and support for easy adoption and extension.

### A.11 Benefits to Graph Machine Learning Community

Our benchmark and its insights offer significant benefits to the broader graph machine learning community in the following areas:

**(a) Current Position of GC in Graph Machine Learning.** First, GC originated in the computer vision domain but has been adapted to address the unique challenges of graph data. It incorporates techniques from graph sampling and coarsening to effectively manage the complexities inherent to graph modalities while to extract essential information. Second, from the view of representation learning, GC aims to create a compact representation of the original graph, preserving essential features for training well-generalized GNNs. Third, GC is gaining traction due to its advantages in accelerating training, enhancing scalability, and improving visualization, making it a valuable tool for various graph-based applications such as NAS (Ding et al., 2022), continual learning (Liu et al., 2023c) and explainability (Fang et al., 2024).

**(b) Addressing Key Questions.**

- **When and Why Specific GC Methods Work:** Our benchmark systematically evaluates different GC methods, elucidating the conditions under which each method excels. This helps researchers and users understand the strengths and limitations of various condensation techniques.
- **Broader Applications of GC:** We demonstrate the versatility of GC beyond traditional applications like NAS and continual learning. Our benchmark highlights its potential in areas such as privacy preservation and efficient data management.
- **Key Observations and Novel Insights:** Based on our well-established benchmark, we have made several new observations and provided fresh insights in the field of GC. For instance, GC methods exhibit significant denoising capabilities against structural noise but are less effective at mitigating node feature noise. Additionally, trajectory matching and gradient-based inner optimization are crucial for achieving reliable performance in NAS and enhancing transferability. These findings highlight both the strengths and limitations of current GC techniques.

**(c) Facilitating General Graph Machine Learning Research.**

- Our benchmark provides a pioneering investigation into the practical effectiveness of GC methods in privacy preservation and their denoising effects (robustness). This highlights the potential of GC methods to serve as a novel set of baselines for comparison with existing privacy defense and robustness techniques. Furthermore, as graph condensation inherently involves modifying datasets, i.e., a data-centric approach, it can be seamlessly integrated with model-centric efforts to deliver complementary benefits in robustness and privacy preservation.
- **Observation 4:** Certain GC methods can achieve both privacy preservation and high condensation performance. This dual capability suggests the potential to break the traditional trade-off between privacy and utility in the trustworthy graph learning area by effectively synthesizing data.
- **Observation 7:** We observe that different GC methods exhibit varying degrees of transferability across datasets, indicating natural differences among GNNs including Graph Transformer. This inspires a rethinking of the similarities between current GNN models, particularly regarding the perspectives and priors they prefer to extract.
- **Observation 11:** We observed that homophilous graphs often become heterophilous after condensation while still maintaining high performance. This unexpected outcome challenges the conventional understanding of the relationship between GNN performance and homophily (Ma et al., 2021). Our findings suggest that the dependency of GNNs on homophily may need to be reevaluated, opening new avenues for research into how graph condensation affects structural properties and model performance.

Overall, our benchmark serves as a valuable resource for graph machine learning researchers by providing comprehensive evaluations, uncovering new applications of GC, and inspiring innovative methodologies. This facilitates advancements in the field, enabling the creation of more effective and adaptable graph learning models.