Does Graph Distillation See Like Vision Dataset Counterpart?

Beining Yang[1,2]* **Kai Wang**[3]* **Qingyun Sun**[1,2]† **Cheng Ji**[1,2], **Xingcheng Fu**[1,2],
Hao Tang[4], **Yang You**[3], **Jianxin Li**[1,2]‡
[1]School of Computer Science and Engineering, Beihang University
[2]Advanced Innovation Center for Big Data and Brain Computing, Beihang University
[3]National University of Singapore [4]Carnegie Mellon University

Abstract

Training on large-scale graphs has achieved remarkable results in graph representation learning, but its cost and storage have attracted increasing concerns. Existing graph condensation methods primarily focus on optimizing the feature matrices of condensed graphs while overlooking the impact of the structure information from the original graphs. To investigate the impact of the structure information, we conduct analysis from the spectral domain and empirically identify substantial Laplacian Energy Distribution (LED) shifts in previous works. Such shifts lead to poor performance in cross-architecture generalization and specific tasks, including anomaly detection and link prediction. In this paper, we propose a novel **S**tructure-broadcasting **G**raph **D**ataset **D**istillation (**SGDD**) scheme for broadcasting the original structure information to the generation of the synthetic one, which explicitly prevents overlooking the original structure information. Theoretically, the synthetic graphs by SGDD are expected to have smaller LED shifts than previous works, leading to superior performance in both cross-architecture settings and specific tasks. We validate the proposed SGDD across 9 datasets and achieve state-of-the-art results on all of them: for example, on the YelpChi dataset, our approach maintains 98.6% test accuracy of training on the original graph dataset with 1,000 times saving on the scale of the graph. Moreover, we empirically evaluate there exist 17.6% $\sim$ 31.4% reductions in LED shift crossing 9 datasets. Extensive experiments and analysis verify the effectiveness and necessity of the proposed designs. The code is available in the https://github.com/RingBDStack/SGDD.

1 Introduction

Graphs have been applied in many research areas and achieved remarkable results, including social networks [27, 67, 73, 79], physical [16, 3, 57, 10], and chemical interactions [2, 93, 111, 80]. Graph neural networks (GNNs), a classical and wide-studied graph representation learning method [40, 84, 98, 63], is proposed to extract information via modeling the features and structures from the given graph. Nevertheless, the computational and memory costs are extremely heavy when training on a large graph [48, 89, 99]. One of the most straightforward ideas is to reduce the redundancy of the large graph. For example, graph sparsification [66, 77] and coarsening [53, 52, 46, 75] are proposed to achieve this goal by dropping redundant edges and grouping similar nodes. These methods have shown promising results in reducing the size and complexity of large graphs while preserving their essential properties.

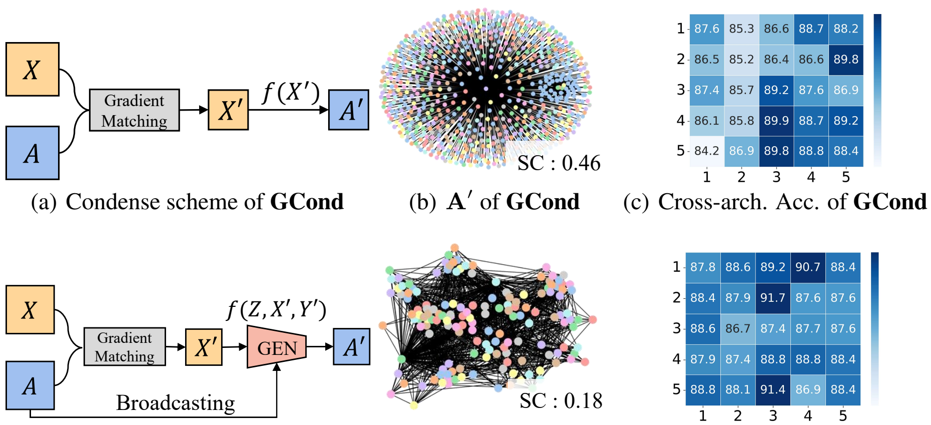
*Equal contribution (yangbeining@buaa.edu.cn, kai.wang@comp.nus.edu.sg).

†Project lead sunqy@buaa.edu.cn

‡Corresponding author lijx@buaa.edu.cn.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

https://doi.org/10.52202/075280-2315



(a) Condense scheme of **GCond** (b) $\mathbf{A}'$ of **GCond** (c) Cross-arch. Acc. of **GCond**

(d) Condense scheme of **SGDD** (e) $\mathbf{A}'$ of **SGDD** (f) Cross-arch. Acc. of **SGDD**

Figure 1: (a) and (d) illustrate that the pipelines of SGDD and GCond [37]. One can find that SGDD broadcast $\mathbf{A}$ from the original graph to the generation of $\mathbf{A}'$ while GCond synthesizes $\mathbf{A}'$ via the operation $f(\cdot)$ (*e.g.*, pair-wise feature similarity [74]) on $\mathbf{X}'$. We also show the condensed graph and its shift coefficient (SC) of GCond and SGDD in (b) and (e), respectively. Note that we introduce SC as an approximation of the LED shift. We show the cross-architecture performance of GCond and SGDD in (c) and (f), 1, 2, 3, 4, and 5 denote APPNP [42], Cheby [13], GCN [41], SAGE [26], and SGC [92], more cross-architecture results can be found in Tab. 2.

However, graph sparsification and coarsening heavily rely on heuristics [5] (*e.g.*, the largest principle eigenvalues [78], pairwise distances [66, 81]), which may lead to poor generalization on various architectures or tasks and sub-optimal for the downstream GNNs training [37]. Most recently, as shown in Fig. 1(a), GCond [37] follows the vision dataset distillation methods [88, 105, 24, 103] and proposes to condense the original graph by the gradient matching strategy. The feature of the synthetic graph dataset is optimized by minimizing the gradient differences between training on original and synthetic graph datasets. Then, the synthetic feature is fed into the function $f(\mathbf{X}')$ (*e.g.*, pair-wise feature similarity [74]) to obtain the structure of the graph. The synthetic graph dataset (including feature and structure) is expected to preserve the task-relevant information and achieve comparable results as training on the original graph dataset at an extremely small cost.

Although the gradient matching strategy has achieved significant success in graph dataset condensation, most of these works [37, 36, 50, 109] follow the previous vision dataset distillation methods to synthesize the condensed graphs, which results in the following limitations: 1) In Fig. 1(b), we can find that the original graph structure information is not well preserved in the condensed graph. That's because they build the structure ($\mathbf{A}'$) upon the learned feature ($\mathbf{X}'$) (Fig. 1(a)), which may cause the loss of original structure information. 2) In vision dataset distillation, applying the gradient matching strategy may result in an entanglement between the synthetic dataset and the architecture that is used for condensing [106, 104, 39]. This can detrimentally impact their performance, especially when it comes to generalizing to unseen architectures [22]. This issue is compounded when dealing with graph data that comprise both features and structures. As shown in Fig. 1(c) and Tab. 2, GCond [37] explicitly displays inferior performance, demonstrating that generating structure only based on the feature degrades the generalization of the condensed graph.

To investigate the relation between the structure of the condensed graph and the performance of training on it, we follow previous works [82, 1, 13, 78] to analyze it from a spectral view. Specifically, we explore the relationship between the Laplacian Energy Distribution (LED) shift [82, 25, 12] and the generalization performance of the condensed graph. We empirically find a positive correlation between LED shift and the performance in cross-architecture settings. To address these issues, we introduce a novel Structure-broadcasting Graph Dataset Distillation (SGDD) scheme to condense the original graph dataset by broadcasting the original adjacency matrix to the generation of the synthetic one, named SGDD, which explicitly prevents overlooking the original structure information. As shown in Fig. 1(d), we propose the graphon approximation to broadcast the original structure $\mathbf{A}$ as supervision to the generation of condensed graph structure $\mathbf{A}'$. Then, we optimize $\mathbf{A}'$ by minimizing

https://doi.org/10.52202/075280-2315

the optimal transport distance of these two structures. For $\mathbf{X}'$, we follow the [37] to synthesize the feature of the condensed graph. The $\mathbf{A}'$ and $\mathbf{X}'$ are jointly optimized in a bi-level loop. Both theoretical analysis (Sec. 3.2) and empirical studies (Fig. 1(c) and 1(f)) consistently show that our method reduces the LED shift significantly (0.46 → 0.18).

We further evaluate the impact of LED shifts and conduct experiments in the cross-architecture setting. Comparing Fig. 1(f), 1(c), and Tab. 2, the improvement of SGDD is up to 11.3%. To evaluate the generalization of our method, we extend SGDD to node classification, anomaly detection, and link prediction tasks. SGDD achieves state-of-the-art results on these tasks in most cases. Notably, we obtain new state-of-the-art results on YelpChi and Amazon with 9.6% and 7.7% improvements, respectively. Our main contributions can be summarized as follows:

- Based on the analysis of the difference between graph and vision dataset distillation, we introduce SGDD, a novel framework for graph dataset distillation via broadcasting the original structure information to the generation of a condensed graph.
- In SGDD, the graphon approximation provides a novel scheme to broadcast the original structure information to condensed graph generation, and the optimal transport is proposed to minimize the LED shifts between original and condensed graph structures.
- SGDD effectively reduces the LED shift between the original and the condensed graphs, consistently surpassing the performance of current state-of-the-art methods across a variety of datasets.

2 Related Work

Dataset Distillation & Dataset Condensation. Dataset distillation (DD) [88, 106, 24, 51, 110, 23, 86, 49, 87, 6, 7] aims to distill a large dataset into a smaller but informative synthetic one. The proposed method imposes constraints on the synthetic samples by minimizing the training loss difference, while ensuring that the samples remain informative. This technique is useful for applications such as continual learning [105, 106, 39, 44, 104], as well as neural architecture search [61, 62, 96]. Recently, GCond [37] is proposed to reduce a large-scale graph to a smaller one for node classification using the gradient matching scheme in DC [105]. Unlike GCond [37], who build the structure only through the learned feature, we explicitly broadcast the original graph structure to the generations to prevent the overlooking of the original graph structure information.

Graph Coarsening & Graph Sparsification. Graph Coarsening [53, 52, 14] follows the intuition that nodes in the original graph can naturally group the similiar node to a super-nodes. Graph Sparsification [66, 38, 78] is aimed at reducing the edges in the original graph, as there are many redundant relations in the graphs. We can simply sum up both two methods by trying to reduce the "useless" component in the graph. Nevertheless, these methods primarily rely on unsupervised techniques such as the largest k eigenvalue [66] or the multilevel incomplete LU factorization [14], which cannot guarantee the behavior of the synthetic graph in downstream tasks. Moreover, they fail to reduce the graph size to an extremely small degree (*e.g.*, reduce the number of nodes to 0.1% of the original). In contrast, graph condensation can aggregate information into a smaller yet informative graph in a supervised way, thereby overcoming the aforementioned limitations.

3 Preliminary and Analysis

3.1 Formulation of graph condensation

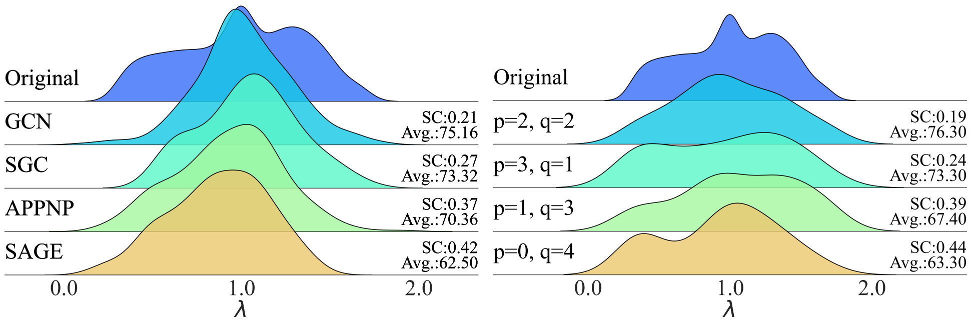
Consider a graph dataset $\mathcal{G} = \{\mathbf{A}, \mathbf{X}, \mathbf{Y}\}$, where $\mathbf{A} \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix, $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the feature, and $\mathbf{Y} \in \mathbb{R}^{N \times 1}$ represents the labels. $\mathcal{S} = \{\mathbf{A}', \mathbf{X}', \mathbf{Y}'\}$ is defined as the synthetic dataset, the first dimension of $\mathbf{A}'$, $\mathbf{X}'$, and $\mathbf{Y}'$ are N' ($N' \ll N$).

The goal of graph condensation is achieving comparable results as training on the original graph dataset $\mathcal{G}$ via training on the synthetic one $\mathcal{S}$. The optimization process can be formulated as follow,

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathrm{M}(\theta_{\mathcal{S}}^*, \theta_{\mathcal{G}}^*) \quad \text{s.t} \quad \theta_t^* = \arg\min_{\theta} \mathrm{Loss}(\mathrm{GNN}_\theta(t)), \tag{1}$$

where $\mathrm{GNN}_\theta(t)$ denotes a GNN parameterized with θ, $\theta_{\mathcal{S}}$ and $\theta_{\mathcal{G}}$ are the parameters that are trained on $\mathcal{S}$ and $\mathcal{G}$, and $t \in \{\mathcal{S}, \mathcal{G}\}$, $\mathrm{M}(\cdot)$ denotes a matching function and $\mathrm{Loss}(\cdot)$ is the loss function.

https://doi.org/10.52202/075280-2315



(a) LEDs in various architectures

(b) LEDs in BWGNN [82]

Figure 2: (a): Illustrations of the LEDs (use the density plotting, each peak represents the LED of a graph), SC, and Avg. (average test performances on GCN, SGC, APPNP, and SAGE) in various architectures. (b): Evaluation of these three metrics in frequency-adaptive BWGNN. We empirically find that the LED shifts (*i.e.*, SC) are high-consistent with Avg. performances, thus SC could be a good indicator to evaluate the performance of the condensed graph.

3.2 Analysis of the structure of the condensed graph and its effects

In this section, we first introduce the definition of Laplacian Energy Distribution (LED), then analyze the LED in previous work, and finally explore its influence on generalization performance.

Laplacian Energy Distribution. Following the previous works [25, 12, 82], we introduce Laplacian energy distribution (LED) to analyze the structure of graph data. Given an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, the Laplacian matrix $\mathbf{L}$ is defined as $\mathbf{D} - \mathbf{A}$, where the $\mathbf{D}$ is the degree matrix of $\mathbf{A}$. We utilize the normalized Lapalacian matrix $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}$ is an identity matrix. The eigenvalues of $\tilde{\mathbf{L}}$ are then defined as $\lambda_1, \dots \lambda_n, \dots \lambda_N$ by ascending order, *i.e.*, $\lambda_1 \leq \cdots \lambda_n \leq \cdots \leq \lambda_N$, with orthonormal eigenvectors $\boldsymbol{U} = (\mathrm{u}_1, \cdots \mathrm{u}_n, \cdots \mathrm{u}_N)$ correspond to these eigenvalues.

Definition 1 (Laplacian Energy Distribution [25, 12, 82])**.** *Let* $\mathbf{X} = (x_1, \cdots x_n, \cdots x_N)^\top \in \mathbb{R}^{N \times d}$ *be the feature of graph, we have* $\hat{\mathbf{X}} = (\hat{x}_1 \cdots \hat{x}_n \cdots \hat{x}_N)^\top = \boldsymbol{U}^\top \mathbf{X}$ *as the post-Graph-Fourier-Transform of* $\mathbf{X}$*. The formulation of LED is defined as:*

$$\eta_n(\mathbf{X}, \tilde{\mathbf{L}}) = \frac{\hat{x}_n^2}{\sum_{i=1}^{N} \hat{x}_i^2} = \bar{x_n}. \tag{2}$$

LED analysis of previous work. First, we define $\eta^{\mathcal{G}}$ and $\eta^{\mathcal{S}}$ as the LED of the original graph and condensed graph, respectively. Then the LED shift of the above two graphs can be formulated as: $||\eta^{\mathcal{G}} - \eta^{\mathcal{S}}|| = ||\sum_{i=1}^{N} \eta_i(\mathbf{X}, \tilde{\mathbf{L}}) - \sum_{j=1}^{N'} \eta_j(\mathbf{X}', \tilde{\mathbf{L}}')||$. In GCond [37], the matching function M defaults as the MSE loss. Therefore, the objective of GCond can be written as $||\theta_{\mathcal{G}} - \theta_{\mathcal{S}}|| \leq \epsilon$, where ϵ is a small number as expected. Incorporating such an objective in the LED shift formula, the lower bound is shown as follows.

Proposition 1. *Refer to [1, 4, 82], the GNN can be recognized as the bandpass filter. Assume the frequency response area of GNN is* (a, b)*, where* (a, b) *is architecture-specific. The lower bound of GCond is shown in Eq.* (3)*. Detailed proof can be found in Appendix B.1.*

$$||\eta^{\mathcal{G}} - \eta^{\mathcal{S}}|| \geq \epsilon + \sum_{i=1}^{a} ||\bar{\boldsymbol{x}}_{\boldsymbol{i}}^{\mathbf{2}} - \bar{\boldsymbol{x}}_{\boldsymbol{i}}'^{\mathbf{2}}|| + \sum_{j=b}^{N'} ||\bar{\boldsymbol{x}}_{\boldsymbol{j}}^{\mathbf{2}} - \bar{\boldsymbol{x}}_{\boldsymbol{j}}'^{\mathbf{2}}||. \tag{3}$$

According to Eq. (3), we find the lower bond of LED in GCond is related to the frequency response area (a, b) (*i.e.*, specific GNN). For example, GCN [40] or SGC [92] (utilized in GCond) is a low-pass filter [1], which emphasizes the lower Laplacian energy. As shown in Fig. 1(a), the $\mathbf{A}'$ is built upon the learned "low-frequency" feature $\mathbf{X}'$, which may fail to generalize to cross-architectures (*i.e.*, high-pass filters) and specific tasks.

Exploration of the effects of LED shift on the generalization performance. Although the LED shift phenomenon occurs in GCond, quantifying the LED shift is challenging because $\mathcal{G}$ and $\mathcal{S}$ have

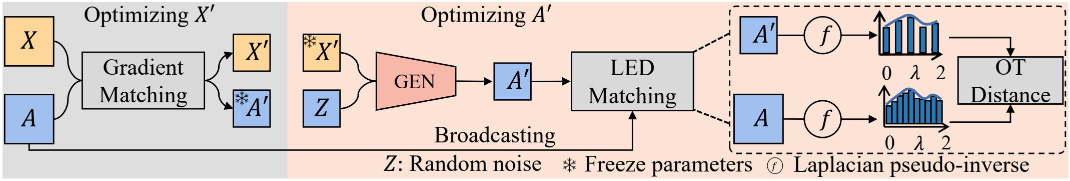


Figure 3: The Training pipeline of the SGDD (left). We first fix $\mathbf{A}'$ to optimize $\mathbf{X}'$ through the gradient matching strategy and we broadcast the supervision of $\mathbf{A}$ to the generation of the graph structure $\mathbf{A}'$. To mitigate the Laplacian Energy Distribution (LED) shift phenomenon, we propose the LED Matching strategy to optimize the $\mathbf{A}'$, which optimizes the learned structure with the optimal transport (OT) distance (right).

different numbers of nodes. To enable comparison, we follow an intuitive assumption that two nodes with similar eigenvalue distribution proportions can be aligned in comparison. Thus, we first convert LEDs of $\mathcal{G}$ and $\mathcal{S}$ into probability distributions using the Kernel Density Estimation (KDE) method [60]. Then we quantify the LED shifts as the distance of the probability distributions using the Jensen-Shannon (JS) divergence [47]. We define the LED shift coefficient (SC) in Definition 2:

Definition 2 (LED shift coefficient, SC)**.** *The LED shift coefficient between $\mathcal{G}$ and $\mathcal{S}$ is:*

$$SC = \mathrm{JS}\left(\frac{1}{|V_{\mathcal{G}}|h}\sum_{\bar{x}_i \in X_{\mathcal{G}}} K(\frac{x-\bar{x}_i}{h})\Big|\Big|\frac{1}{|V_{\mathcal{S}}|h}\sum_{\bar{x}_j \in X_{\mathcal{S}}} K(\frac{x-\bar{x}_j}{h})\right), \tag{4}$$

where the $\mathrm{JS}(\cdot||\cdot)$ denotes the JS-divergence, the $|V_{\mathcal{G}}|$ and the $|V_{\mathcal{S}}|$ is the number of the nodes to corresponding graphs, the $K(\cdot)$ represents the Gaussian kernel function with bandwidth parameter h. $SC \in [0,1]$ reflects the divergence between $\mathcal{G}$ and $\mathcal{S}$ (a smaller SC indicates more similar).

In Fig. 2, we empirically study the influences of SC in two settings: various GNNs in Fig. 2(a) and fixed BWGNN [82] with adaptive bandpass in Fig. 2(b). **Based on the results in Fig. 2, We have several observations:** (1) The entangled learning paradigm that building structure (*i.e.*, adjacency matrix) upon on feature matrix will significantly lead to the LED shift phenomenon. (2) The positive correlation exists between the LED shift and the generalization performance of the condensed graph. (3) Preserving more information about the original graph structure may alleviate the LED shift phenomenon and improve the generalization performance of the condensed graph.

4 Structure-broadcasting Graph Dataset Distillation

In this section, we first present the pipeline and overview of SGDD in Fig. 3. Then, we introduce two modules of SGDD. Finally, we summarize the training pipeline of our SGDD.

4.1 Learning graph structure via graphon approximation

To prevent overlooking the original structure $\mathbf{A}$ from $\mathcal{G}$, we broadcast $\mathbf{A}$ as supervision for the generation of $\mathbf{A}'$. Considering the different shapes between $\mathbf{A}$ and $\mathbf{A}'$ ($N' \ll N$), we introduce graphon [21, 70, 17, 33, 94] to distill the original structure information to the condensed structure $\mathbf{A}'$. Specifically, given random noise $\mathcal{Z}(N') \in \mathbb{R}^{N' \times N'}$ as the input coordinates, through the generative model, we then synthesize an adjacency matrix $\mathbf{A}'$ with N' nodes. This process can be formulated as $\mathbf{A}' = \mathrm{GEN}(\mathcal{Z}(N'); \Phi)$, where the $\mathrm{GEN}(\cdot)$ is a generative model with parameter Φ, and the optimization process is then defined as:

$$\mathcal{L}_{\textbf{structure}} = \mathrm{Distance}(\mathbf{A}, \mathrm{GEN}(\mathcal{Z}(N'); \Phi)), \tag{5}$$

where $\mathbf{A}$ is supervision and $\mathrm{Distance}(\cdot)$ is a metric that measure the difference of $\mathbf{A}$ and $\mathbf{A}'$. The details of $\mathrm{Distance}(\cdot)$ can be found in Sec. 4.2. To avoid overlooking of the inherent relation [32, 73] between $\mathbf{A}'$ and the corresponding node information (*i.e.*, $\mathbf{X}'$ and $\mathbf{Y}'$), we jointly input $\mathbf{X}'$ and $\mathbf{Y}'$ as conditions to generate $\mathbf{A}'$. Therefore, the final version of the generative model can be written as $\mathbf{A}' = \mathrm{GEN}(\mathcal{Z}(N') \oplus \mathbf{X}' \oplus \mathbf{Y}'; \Phi)$, the $\oplus$ denotes the concatenate operation.

https://doi.org/10.52202/075280-2315

To study the performance of SGDD in the above paradigm, we theoretically prove the upper bound of LED shift in SGDD by invoking graphon theory. The result can be presented as follows.

Proposition 2. *The upper bound of the LED shift on SGDD is shown as:*

$$||\eta^{\mathcal{G}} - \eta^{\mathcal{S}}|| \leq \delta_{\square}(W_{\mathbf{A}}, \mathbf{A}'), \tag{6}$$

where $\delta_{\square}$ denotes the cut distance [54] and $W_{\mathbf{A}}$ is the graphon of $\mathbf{A}$. See details in Appendix B.2.

Note minimizing the upper bound of Eq. (6) is equal to optimizing the $L_{structure}$ on Eq. (5). Compared to the lower bound in (Eq. (3)), our upper bound is not related to any frequency response of specific GNN (*i.e.*, the terms of $\sum_{i=1}^{a} ||\bar{\boldsymbol{x}}_{\boldsymbol{i}}^{\mathbf{2}} - \bar{\boldsymbol{x}}_{\boldsymbol{i}}^{\prime\mathbf{2}}|| + \sum_{i=b}^{N'} ||\bar{\boldsymbol{x}}_{\boldsymbol{i}}^{\mathbf{2}} - \bar{\boldsymbol{x}}_{\boldsymbol{i}}^{\prime\mathbf{2}}||$). As a result, SGDD may perform better than the previous work, especially in the cross-architecture setting and specific tasks.

4.2 Optimizing the graph structure via optimal transport

To mitigate the LED shift between $\mathbf{A}$ and $\mathbf{A}'$, ideally, we can directly minimize the proposed SC. However, SC requires an extremely time-consuming ($O(N^3)$) [4, 58] eigenvalue decomposition operation. Therefore, we propose the LED Matching strategy based on the *optimal transport* theory to form an efficient optimizing process.

Recall in the Eq.(4) of calculating SC, we first decompose the eigenvalue, followed by aligning the node through the JS divergence, which essentially compares the distribution proportions. The key point is to decide the node mapping strategy to align such two graphs. Alternatively, assuming we know the prior distribution of the alignment of $\mathcal{S}$ (*i.e.*, we know the bijection of nodes in $\mathcal{S}$ to the $\mathcal{G}$) and denoting such alignment as $\mathcal{S}*$, we can directly measure the distance between $\mathcal{G}$ and $\mathcal{S}*$ by employing the 2-Wasserstein metric[15, 56].

$$||\eta^{\mathcal{G}} - \eta^{\mathcal{S}*}|| = W_2^2\left(\eta^{\mathcal{G}}, \eta^{\mathcal{S}*}\right) \tag{7}$$

Furthermore, following the assumption in previous work[56, 15], we have $\eta^{G} \sim \mathcal{N}\left(0, L_{\mathcal{G}}^{\dagger}\right)$ and $\eta^{S} \sim \mathcal{N}\left(0, L_{\mathcal{S}*}^{\dagger}\right)$. Then, the Eq.(7) have a closed-form expression[15] as follows:

$$||\eta^{\mathcal{G}} - \eta^{\mathcal{S}*}|| = N \operatorname{tr}\left(L_{\mathcal{G}}^{\dagger}\right) + N' \operatorname{tr}\left(L_{\mathcal{S}*}^{\dagger}\right) - 2 \operatorname{tr}\left(\sqrt{L_{\mathcal{S}*}^{\frac{\ddagger}{2}} L_{\mathcal{G}}^{\dagger} L_{\mathcal{S}*}^{\frac{\dagger}{2}}}\right), \tag{8}$$

the $L^{\dagger}$ denotes the Laplacian pseudo-inverse operation and the tr indicates the trace operation of matrix. Therefore, even though we could not know the actual mapping strategy $\mathcal{S}*$, we can use the infimum of all possible strategies as a proxy solution. Formally, following the prior work [56], we employ the function T as a transport plan in the metric space $\mathcal{X}$ to represent all feasible mapping strategies. Then, we use the $T_{\#}\eta^{\mathcal{S}}$ to represent the pushing forward process of transferring the distribution of $\eta^{\mathcal{S}}$ to the $\eta^{\mathcal{G}}$. As a result, the distance can be regarded as finding the infimum.

$$\begin{aligned} ||\eta^{\mathcal{G}} - \eta^{\mathcal{S}}|| &= \inf_{T_{\#}\eta^{\mathcal{S}} = \eta^{\mathcal{G}}} \int_{\mathcal{X}} \|x - T(x)\|^2 d\eta^{\mathcal{S}}(x) \\ &= N' \operatorname{tr}(L_{\mathcal{S}}^{\dagger}) - 2 \operatorname{tr}\left(\left((L_{\mathcal{S}}^{\dagger})^{1/2} P^T L_{\mathcal{G}}^{\dagger} P \left(L_{\mathcal{S}}^{\dagger}\right)^{1/2}\right)^{1/2}\right). \end{aligned} \tag{9}$$

Here, due to the transport plan T is impractical in optimizing, following the previous work[15], we use $P \in \mathbb{R}^{N' \times N}$ denotes as a free parameter serving as the direct mapping strategy between nodes. Thus, the $\mathrm{Distance}$ in the Eq. (5) could be directly optimized by the Eq.(9) (*i.e.*, use the P represents all possible mapping strategies, the optimizing of P is equal to choosing a more optimal mapping strategy). In the experimental setting, we use the Sinkhorn-Knopp[9] algorithm to optimize P.

The overall time complexity is reduced to the $O(N^{\omega}) \leq O(N^{2.373})$[15]. Note that the $L_{\mathcal{G}}^{\dagger}$ may be too large for computing, so we empirically sample a medium size (*e.g.*, 2,000 nodes) sub-structure in the experiment and ablate its influence in Appendix C.7.

Table 1: Comparisons to state-of-the-art methods. SGDD achieves the highest results in most cases on node classification (NC), anomaly detection (AD), and link prediction (LP) tasks. We report test accuracy (%) on NC datasets (including Citeseer, Cora, Ogbn-arxiv, Flickr, and Reddit), F1-macro (%) on AD datasets (including YelpChi and Amazon), and AUC (%) on LP datasets (including Citeseer-L and DBLP). **Bold entries** are best results, underline mark the runner-ups.

	Dataset	Ratio (r)	Random	Herding	K-Center	Coarsening	GDC	Gcond	SGDD	Whole
NC	Citeseer [41]	0.90%	$54.4_{\pm 4.4}$	$57.1_{\pm 1.5}$	$52.4_{\pm 2.8}$	$52.2_{\pm 0.4}$	$66.8_{\pm 1.5}$	$\mathbf{70.5}_{\pm 1.2}$	$\underline{69.5}_{\pm 0.4}$	$71.7_{\pm 0.1}$
		1.80%	$64.2_{\pm 1.7}$	$66.7_{\pm 1.0}$	$64.3_{\pm 1.0}$	$59.0_{\pm 0.5}$	$66.9_{\pm 0.9}$	$\mathbf{70.6}_{\pm 0.9}$	$\underline{70.2}_{\pm 0.8}$	
		3.60%	$69.1_{\pm 0.1}$	$69.0_{\pm 0.1}$	$69.1_{\pm 0.1}$	$65.3_{\pm 0.5}$	$66.3_{\pm 1.5}$	$\underline{69.8}_{\pm 1.4}$	$\mathbf{70.3}_{\pm 1.7}$	
	Cora [41]	1.30%	$63.6_{\pm 3.7}$	$67.0_{\pm 1.3}$	$64.0_{\pm 2.3}$	$31.2_{\pm 0.2}$	$67.3_{\pm 1.9}$	$\underline{79.8}_{\pm 1.3}$	$\mathbf{80.1}_{\pm 0.7}$	$81.2_{\pm 0.2}$
		2.60%	$72.8_{\pm 1.1}$	$73.4_{\pm 1.0}$	$73.2_{\pm 1.2}$	$65.2_{\pm 0.6}$	$67.6_{\pm 3.5}$	$\underline{80.1}_{\pm 0.6}$	$\mathbf{80.6}_{\pm 0.8}$	
		5.20%	$76.8_{\pm 0.1}$	$76.8_{\pm 0.1}$	$76.7_{\pm 0.1}$	$70.6_{\pm 0.1}$	$67.7_{\pm 2.2}$	$\underline{79.3}_{\pm 0.3}$	$\mathbf{80.4}_{\pm 1.6}$	
	Ogbn-arxiv [29]	0.05%	$47.1_{\pm 3.9}$	$52.4_{\pm 1.8}$	$47.2_{\pm 3.0}$	$35.4_{\pm 0.3}$	$58.6_{\pm 0.4}$	$\underline{59.2}_{\pm 1.1}$	$\mathbf{60.8}_{\pm 1.3}$	$71.4_{\pm 0.1}$
		0.25%	$57.3_{\pm 1.1}$	$58.6_{\pm 1.2}$	$56.8_{\pm 0.8}$	$43.5_{\pm 0.2}$	$59.9_{\pm 0.3}$	$\underline{63.2}_{\pm 0.3}$	$\mathbf{65.8}_{\pm 1.2}$	
		0.50%	$60.0_{\pm 0.9}$	$60.4_{\pm 0.8}$	$60.3_{\pm 0.4}$	$50.4_{\pm 0.1}$	$59.5_{\pm 0.3}$	$\underline{64.0}_{\pm 0.4}$	$\mathbf{66.3}_{\pm 0.7}$	
	Flickr [100]	0.10%	$41.8_{\pm 2.0}$	$42.5_{\pm 1.8}$	$42.0_{\pm 0.7}$	$41.9_{\pm 0.2}$	$46.3_{\pm 0.2}$	$\underline{46.5}_{\pm 0.4}$	$\mathbf{46.9}_{\pm 0.1}$	$47.2_{\pm 0.1}$
		0.50%	$44.0_{\pm 0.4}$	$43.9_{\pm 0.9}$	$43.2_{\pm 0.1}$	$44.5_{\pm 0.1}$	$45.9_{\pm 0.1}$	$\mathbf{47.1}_{\pm 0.1}$	$\underline{47.1}_{\pm 0.3}$	
		1.00%	$44.6_{\pm 0.2}$	$44.4_{\pm 0.6}$	$44.1_{\pm 0.4}$	$44.6_{\pm 0.1}$	$45.8_{\pm 0.1}$	$\underline{47.1}_{\pm 0.1}$	$\mathbf{47.1}_{\pm 0.1}$	
	Reddit [26]	0.01%	$46.1_{\pm 4.4}$	$53.1_{\pm 2.5}$	$46.6_{\pm 2.3}$	$40.9_{\pm 0.5}$	$88.2_{\pm 0.2}$	$\underline{88.0}_{\pm 1.8}$	$\mathbf{90.5}_{\pm 2.1}$	$93.9_{\pm 0.0}$
		0.05%	$58.0_{\pm 2.2}$	$62.7_{\pm 1.0}$	$53.0_{\pm 3.3}$	$42.8_{\pm 0.8}$	$89.5_{\pm 0.1}$	$\underline{89.6}_{\pm 0.7}$	$\mathbf{91.8}_{\pm 1.9}$	
		0.50%	$66.3_{\pm 1.9}$	$71.0_{\pm 1.6}$	$58.5_{\pm 2.1}$	$47.4_{\pm 0.9}$	$\underline{90.5}_{\pm 1.2}$	$90.1_{\pm 0.5}$	$\mathbf{91.6}_{\pm 1.8}$	
AD	YelpChi [68]	0.05%	$41.8_{\pm 0.3}$	$46.1_{\pm 0.9}$	$49.3_{\pm 1.1}$	$46.2_{\pm 2.1}$	$47.9_{\pm 1.1}$	$\underline{48.6}_{\pm 3.7}$	$\mathbf{56.2}_{\pm 1.8}$	$61.1_{\pm 1.8}$
		0.10%	$43.7_{\pm 1.2}$	$47.1_{\pm 1.2}$	$44.2_{\pm 1.8}$	$47.5_{\pm 1.8}$	$\underline{50.2}_{\pm 2.1}$	$49.6_{\pm 1.8}$	$\mathbf{58.1}_{\pm 2.3}$	
		0.20%	$45.9_{\pm 2.2}$	$46.4_{\pm 0.8}$	$47.5_{\pm 0.4}$	$49.1_{\pm 1.2}$	$49.7_{\pm 2.0}$	$\underline{50.1}_{\pm 2.8}$	$\mathbf{59.7}_{\pm 1.8}$	
	Amazon [102]	0.02%	$76.2_{\pm 0.8}$	$74.1_{\pm 0.9}$	$73.4_{\pm 2.1}$	$75.2_{\pm 2.9}$	$74.1_{\pm 1.9}$	$\underline{77.9}_{\pm 3.1}$	$\mathbf{83.3}_{\pm 2.6}$	$89.5_{\pm 0.9}$
		0.20%	$76.4_{\pm 1.6}$	$76.5_{\pm 2.3}$	$74.2_{\pm 1.1}$	$76.8_{\pm 1.0}$	$\underline{78.2}_{\pm 2.1}$	$78.1_{\pm 1.9}$	$\mathbf{84.8}_{\pm 1.7}$	
		2.00%	$78.2_{\pm 0.9}$	$77.2_{\pm 1.8}$	$73.8_{\pm 2.2}$	$77.8_{\pm 2.8}$	$\underline{79.3}_{\pm 3.1}$	$79.2_{\pm 2.0}$	$\mathbf{86.9}_{\pm 2.1}$	
LP	Citeseer-L [97]	0.90%	$56.8_{\pm 0.8}$	$63.1_{\pm 1.3}$	$78.3_{\pm 2.6}$	$76.9_{\pm 0.8}$	$81.5_{\pm 2.5}$	$\underline{83.4}_{\pm 1.9}$	$\mathbf{86.4}_{\pm 1.6}$	$96.8_{\pm 1.8}$
		1.80%	$61.4_{\pm 1.8}$	$63.3_{\pm 1.8}$	$79.1_{\pm 1.8}$	$77.4_{\pm 1.7}$	$83.4_{\pm 1.6}$	$\underline{83.8}_{\pm 2.1}$	$\mathbf{87.2}_{\pm 2.1}$	
		3.60%	$63.5_{\pm 0.9}$	$64.7_{\pm 1.7}$	$80.6_{\pm 2.7}$	$78.4_{\pm 0.4}$	$\underline{84.2}_{\pm 2.1}$	$83.1_{\pm 2.7}$	$\mathbf{87.1}_{\pm 1.2}$	
	DBLP [83]	0.05%	$64.1_{\pm 2.6}$	$65.8_{\pm 0.7}$	$72.4_{\pm 1.6}$	$77.8_{\pm 1.3}$	$\underline{78.9}_{\pm 2.1}$	$77.2_{\pm 2.1}$	$\mathbf{81.3}_{\pm 2.8}$	$84.2_{\pm 0.7}$
		0.25%	$68.3_{\pm 1.4}$	$71.2_{\pm 1.7}$	$74.9_{\pm 2.8}$	$76.9_{\pm 0.9}$	$77.5_{\pm 1.9}$	$\underline{78.6}_{\pm 1.2}$	$\mathbf{82.1}_{\pm 1.9}$	
		0.50%	$68.7_{\pm 2.6}$	$74.3_{\pm 0.8}$	$75.2_{\pm 0.6}$	$76.8_{\pm 2.1}$	$77.6_{\pm 2.1}$	$\underline{79.9}_{\pm 1.9}$	$\mathbf{82.1}_{\pm 1.8}$	

4.3 Training pipeline of SGDD

As illustrated in Fig. 1(d) and Fig. 3, we commence by introducing a novel graph structure learning paradigm termed "graphon approximation". This paradigm integrates both the feature X' and auxiliary information Z to generate the structure. Subsequently, the learned structure A' is forced to be closer to the original graph structure A in terms of Laplacian energy distribution.

Additionally, our proposed methodology, SGDD, implements a bi-loop optimization schema. Within this framework, we concurrently optimize the parameters X' and A'. More specifically, the refinement of X' is achieved through a gradient matching strategy, whereas the A' is enhanced using the LED matching technique. During each step, the other component is frozen to ensure effective refinement.

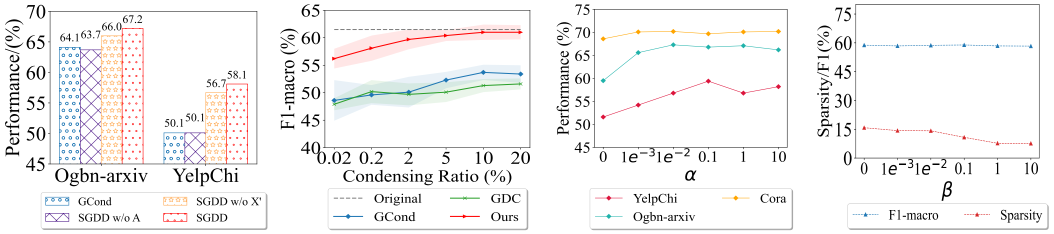
Our overall training loss function can be summarized as $\mathcal{L} = \mathcal{L}_{feature} + \alpha\mathcal{L}_{structure} + \beta||\mathbf{A}||_2$, where the $||\mathbf{A}||_2$ is proposed as a sparsity regularization term and $\mathcal{L}_{feature}$ denotes the gradient matching strategy [37]. α and β are trade-off parameters, we study their sensitiveness in the Sec. 5.3. The algorithm can be found in Appendix C.3.

5 Experiments

5.1 Datasets and implementation details

Datasets. We evaluate SGDD on five node classification datasets: Cora [41], Citeseer [41], Ogbn-arxiv [29], Flickr [100], Reddit [26], two anomaly detection datasets: YelpChi [68], Amazon [102], and two link prediction datasets: Citeseer-L [97], DBLP [83]. For the node classification and anomaly detection tasks, we follow the public settings [18] of train and test. To make a fair comparison, we also follow the previous setting [101, 83], we randomly split 80% nodes for training, 10% nodes for validation, and the remaining 10% for testing. To avoid data leakage, we only utilize 80% training samples for condensation. More details of each dataset can be found in Appendix C.1.

https://doi.org/10.52202/075280-2315



(a) Ablation of $\mathbf{X}'$ and $\mathbf{A}$. (b) Evaluation of $r(\%)$ (c) Evaluation of α (d) Evaluation of β

Figure 4: (a) Ablation of components in SGDD. (b) Evaluation of the scalability of SGDD. (c) and (d): the trade-off parameters analysis on α and β.

Implementation details. Without specific designation, **in the condense stage**, we adopt the 2-layer GCN with 128 hidden units as the backbone, and we adopt the settings on [94], which use 2-layer MLP to represent the structure generative model (*i.e.*, GEN). The learning rates for structure and feature are set to 0.001 (0.0001 for Ogbn-arxiv and Reddit) and 0.0001, respectively. We set α to 0.1, and β to 0.1. **In the evaluation stage**, we train the same network for 1,000 epochs on the condensed graph with a learning rate of 0.001. Following the settings in [37], we repeat all experiments ten times and report average performance and variance. More details[4] can be found in Appendix C.2.

5.2 Comparison with state-of-the-art methods

We compare our proposed SGDD with six baselines: Random, which randomly selected nodes to form the original graph, corset methods (Herding [90] and K-Center [72]), graph coarsening methods (Corasening [30]), and the state-of-the-art graph condensation methods (GCond [37]). GDC is proposed as a baseline in [37], where cosine similarity is added as a constraint to generate the structure of the condensed graph. In Table 1, we present the performance metrics including accuracy, F1-macro, and AUC. For clarity and simplicity, percentages are represented without the $\%$ symbol, and variance values are provided. Based on the results, we have the following observations: 1) Our proposed SGDD achieves the highest results in most settings, which shows the superiority and generalization of our method. 2) On anomaly detection datasets, the improvements are more significant than other tasks, *i.e.*, improving GCond with 9.6% and 7.7% on YelpChi and Amazon datasets, which can be explained that our method captures the structure information from the original graph dataset more efficiently.

Table 2: Results of cross-architecture setting, we test condensed graphs in APPNP, Cheby, GCN, GraphSAGE, and SGC. Avg. and Std. : the average performance and the standard deviation of the results, the $\Delta(\%)$ denotes the improvements upon the GDC. We mark the best performance by **bold**.

Datasets	Methods	Architectures							Statistics		
		MLP [11]	GAT [84]	APPNP [42]	Cheby [13]	GCN [41]	SAGE [26]	SGC [92]	Avg.	Std.	$\Delta(\%)$
Reddit [26]	GDC	50.3	54.8	81.2	77.5	89.5	89.7	90.5	76.2	16.9	-
	GCond	42.5	60.2	87.8	75.5	89.4	89.1	89.6	76.3	18.5	↑ 0.1
	Ours	56.1	74.4	89.2	78.4	89.4	89.4	89.4	**80.9**	**12.6**	↑ **4.7**
Cora [41]	GDC	67.2	64.2	67.1	67.7	67.9	66.2	72.8	67.6	2.6	-
	GCond	73.1	66.2	78.5	76	80.1	78.2	79.3	75.9	4.9	↑ 8.3
	Ours	76.7	75.8	78.4	78.5	79.8	80.4	78.5	**78.3**	**1.6**	↑ **10.7**
DBLP [83]	GDC	74.4	76.8	77.4	76.7	78.9	74.8	78.4	76.8	1.7	-
	GCond	75.3	77.6	78.9	76.1	79.6	77.4	79.9	77.8	1.7	↑ 1.1
	Ours	78.4	79.6	80.1	80.6	82.1	80.7	81.4	**80.4**	**1.2**	↑ **3.6**
YelpChi [68]	GDC	30.7	36.4	43.7	41.5	49.6	47.4	50.1	42.8	7.2	-
	GCond	48.9	31.8	46.7	48.6	50.1	42.5	48.7	45.3	6.5	↑ 2.6
	Ours	54.2	56.4	58.2	56.8	59.7	54.1	56.7	**56.6**	**2.0**	↑ **13.8**

5.3 Ablation Study

Cross-architecture generalization analysis. To evaluate the generalization ability of SGDD on unseen architectures, we conduct experiments that train on the condensed graph with different

[4] The code is available in https://github.com/RingBDStack/SGDD, based on PyTorch[65] and MindSpore[31].

Table 3: Comparison of the cross-architecture generalization performance between GCond and SGDD on Ogbn-arxiv. **Bold entries** are the best results. ↑/↓: our method show increase or decrease performance.

C\T	APPNP GCond / SGDD	Cheby GCond / SGDD	GCN GCond / SGDD	SAGE GCond / SGDD	SGC GCond / SGDD
APPNP	**60.3** / 60.2↓	51.8 / **53.2**↑	59.9 / **62.4**↑	59.0 / **60.2**↑	**61.2** / 60.4↓
Cheby	57.4 / **58.5**↑	53.5 / **55.8**↑	57.4 / **65.3**↑	**57.1** / 57.0↓	58.2 / **60.2**↑
GCN	59.3 / **60.1**↑	51.8 / **53.7**↑	60.3 / **64.2**↑	60.2 / **61.2**↑	59.2 / **59.8**↑
SAGE	57.6 / **58.9**↑	53.9 / **53.8**↑	58.1 / **63.8**↑	57.8 / **61.8**↑	59.0 / **61.1**↑
SGC	59.7 / **60.0**↑	49.5 / **52.3**↑	59.2 / **62.2**↑	58.9 / **62.9**↑	60.5 / **61.5**↑

Table 4: Neural Architecture Search. Methods are compared in validation accuracy correlation and test accuracy on obtained architecture.

Dataset	Pearson Correlation / Performance (%)			Whole Per. (%)
	Random	GCond	SGDD	
Ogbn-arxiv	0.63 / 71.1	0.64 / 71.2	**0.67 / 71.6**	71.9
YelpChi	0.43 / 56.7	0.48 / 58.4	**0.56 / 60.6**	61.1
DBLP	0.58 / 81.2	0.62 / 83.8	**0.68 / 84.0**	84.2

architectures and report their performances in Tab. 2. Here, the condensed graph is obtained by optimizing SGDD with SGC [92]. We test its cross-architecture generalization performances on 2-layer-MLP [11], GAT [84], APPNP [42], Cheby [13], GCN [41], and SAGE [26].

To better understand, we also show several statistics metrics, including Avg., Std., and Δ. SGDD and GCond improves GDC significantly, which indicates there exists a large difference between graph dataset condensation and vision dataset distillation, *i.e.*, the structure information should be specially considered. Compared to GCond, the improvement of our method is up to 11.3%, demonstrating the effectiveness of broadcasting the original structure to the condensed graph structure generation. More experiments on other datasets can be found in Appendix C.6.

Versatility of SGDD. Following the setting of GCond [37], we also study whether our proposed SGDD is robust on various architectures. We first condense the Ogbn-arxiv graph dataset with five architectures, including APPNP [42], Cheby [13], GCN [41], SAGE [26], and SGC [92], respectively. Then, we evaluate these condensed graphs on the above five architectures and report their performances in Tab. 3. The experiment results show that SGDD achieves non-trivial improvements than GCond in most cases, which demonstrates the strong versatility of our method.

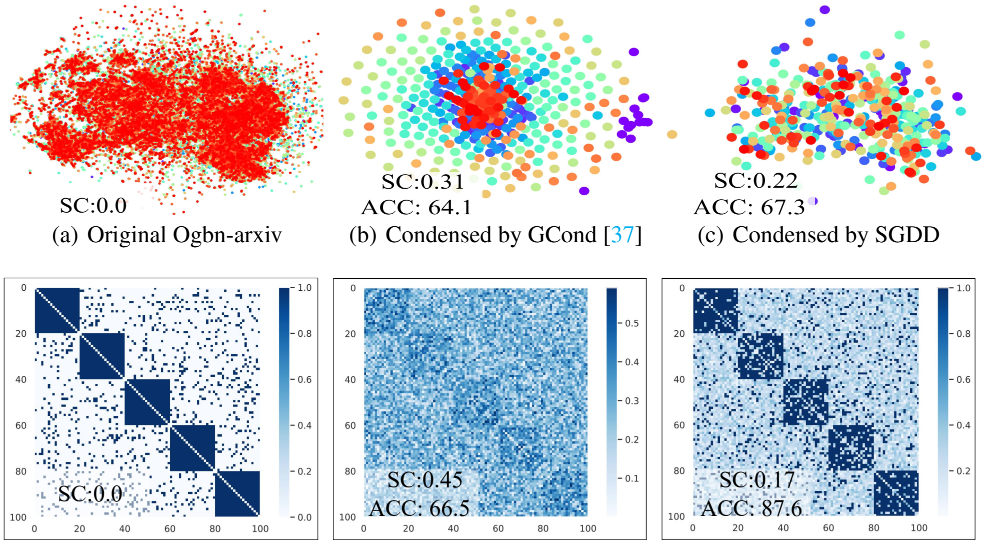
Evaluation on neural architecture search. Similar to vision dataset distillation, graph dataset distillation is also expected to reduce the high cost of neural architecture search (NAS). In order to make a fair comparison, we follow the experimental setting in [37]: searching architectures on condensed Obgn-arxiv, YelpChi, and DBLP datasets with 0.25%, 0.2%, and 0.5% condensing ratios. We report Pearson correlation [45] and performance of random, GCond, and SGDD in Tab. 4. Our SGDD consistently achieves the highest Pearson correlations as well as performances, which indicates the architectures searched by our method are efficient for the whole graph dataset training.

Evaluation of components in SGDD. To explore the effect of the conditions (mentioned in Sec. 4.1) in the condensed graph generation, we design the ablation study of $\mathbf{X}'$ and $\mathbf{A}$. As shown in Fig 4(a), $\mathbf{X}'$ and $\mathbf{A}$ are complementary with each other. SGDD w/o $\mathbf{A}$ performs poorly on Obgn-arxiv and YelpChi datasets, which demonstrates the effect of original graph structure information. Jointly using $\mathbf{X}'$ and $\mathbf{A}$ achieves the highest performances on both datasets, improves GCond with 3.1% on Ogbn-arxiv, and 8.0% on YelpChi.

Evaluation of the scalability of SGDD. To investigate the scalability of SGDD, we evaluate the SGDD on various condensing ratios with $r \in \{0.02, 0.2, 2, 5, 10, 20\}$. As shown in Fig. 4(b), the performance of our method continuously increase as the condensing ratio rises, which indicates the strong scalability of our method. GCond obtains marginal improvements than GDC at all ratios while our SGDD outperforms them significantly. More important, SGDD achieves lossless performance as training on the original graph data when the condensing ratio is 10%.

Exploring the sensitivity of α and β. We conduct experiments to test such two parameters sensitivity on YelpChi, Cora, and Ogbn-arxiv. As shown in Fig. 4(c), we empirically find that the performance of our SGDD is not sensitive to the α. Specifically, compared to the case that α is zero, we have a significant improvement, which proves the effectiveness of our method. Another finding is that the α should be set higher on anomaly detection than on node classification tasks. It could be explained by the original graph structure information being more important on the complex task (such as anomaly detection). We define β as a regularization coefficient to control the sparsity of the condensed graph. As shown in Fig. 4(d), we evaluate the β from 0 to 10 on the YelpChi dataset. The results illustrate that the performance is not sensitive with β and achieve the highest result (F1-macro) when β is set to our default value ($\beta = 0.1$). More experiments can be found in Appendix C.5.

https://doi.org/10.52202/075280-2315



(a) Original Ogbn-arxiv (b) Condensed by GCond [37] (c) Condensed by SGDD

(d) Original synthetic graph (e) Condensed by GCond [37] (f) Condensed by SGDD

Figure 5: Visualizations of the real dataset (a), synthetic dataset (d), and the corresponding condensed graph obtained by GCond (b, e) and SGDD (c, f). The condensing ratio r is set to 0.5%.

5.4 Visualizations

To better understand the effectiveness of our SGDD, we visualize the condensed graphs of GCond and SGDD that are synthesized from real and synthetic graph datasets. For the synthetic graph dataset, we use the Stochastic Block Model (SBM) [28] to synthesize graphs with 5 community (Fig. 5(d)). As shown in Fig. 5, one can find that our method consistently achieves better performances and SC. SGDD reduces 29.0% (comparing 5(b) and 5(c)) and 62.2% (comparing 5(e) and 5(f)) SC on Ogbn-arxiv and synthetic datasets. Visually, the condensed graphs of our method preserve the original graph structure information obviously better than GCond (see the second row of Fig. 5), which proves SGDD is a powerful graph dataset distillation method.

6 Conclusion

We present SGDD, a novel framework for graph dataset distillation via broadcasting the original structure information to the generation of the synthetic one. SGDD shows its robustness on various tasks and datasets, achieving state-of-the-art results on YelpChi, Amazon, Ogbn-arxiv, and DBLP. SGDD reduces the scale of the Yelpchi dataset by 1,000 times while maintaining 98.6% as training on the original data. We provide sufficient experiments and theoretical analysis in this paper and hope it can help the following research in this area. **Limitations and future work:** Although broadcasting the original information to the generated graph shows remarkable success, some informative properties (*e.g.*, the heterogeneity) may lose during the current condense process, which results in sub-optimal performance in the downstream tasks. We are going to explore a more general method in the future.

Acknowledgement

The corresponding author is Jianxin Li. This work was supported by the NSFC through grant No.62225202 and No.62302023. This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-PhD-2021-08-008). Yang You's research group is being sponsored by NUS startup grant (Presidential Young Professorship), Singapore MOE Tier-1 grant, ByteDance grant, ARCTIC grant, SMI grant, and Alibaba grant. We gratefully acknowledge the support of MindSpore, CANN(Compute Architecture for Neural Networks) and Ascend AI Processor used for this research.

https://doi.org/10.52202/075280-2315

References

[1] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *9th International Conference on Learning Representations, Virtual Event, Austria, May 3-7, 2021*, 2021.

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *ArXiv preprint*, 2018.

[3] Daniel Bear, Chaofei Fan, Damian Mrowca, Yunzhu Li, Seth Alter, Aran Nayebi, Jeremy Schwartz, Li Fei-Fei, Jiajun Wu, Josh Tenenbaum, and Daniel L. Yamins. Learning physical graph representations from visual scenes. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual*, 2020.

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[5] Chen Cai, Dingkang Wang, and Yusu Wang. Graph coarsening with neural networks. In *9th International Conference on Learning Representations, Virtual Event, Austria, May 3-7, 2021*, 2021.

[6] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022*, pages 4749–4758, 2022.

[7] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Generalizing dataset distillation via deep generative prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3739–3748, 2023.

[8] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual*, 2020.

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2292–2300, 2013.

[10] Évariste Daller, Sébastien Bougleux, Luc Brun, and Olivier Lézoray. Local patterns and supergraph for chemical graph classification with convolutional networks. In *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2018, Beijing, China, August 17-19, 2018, Proceedings*, volume 11004, pages 97–106, 2018.

[11] Tuan Van Dao, Hiroshi Sato, and Masao Kubo. Mlp-mixer-autoencoder: A lightweight ensemble architecture for malware classification. *Inf.*, 14(3):167, 2023.

[12] Kinkar Ch. Das, Seyed Ahmad Mojallal, and Vilmar Trevisan. Distribution of laplacian eigenvalues of graphs. *Linear Algebra and its Applications*, 508:48–61, 2016.

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information*

https://doi.org/10.52202/075280-2315

Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 3837–3845, 2016.

[14] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[15] Yihe Dong and Will Sawin. COPT: coordinated optimal transport on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual*, 2020.

[16] Holger Ebel and Stefan Bornholdt. Coevolutionary games on networks. *Physical Review E*, 66(5):056118, 2002.

[17] Justin Eldridge, Mikhail Belkin, and Yusu Wang. Graphons, mergeons, and so on! In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2307–2315, 2016.

[18] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *ArXiv preprint*, 2019.

[19] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1972–1982, 2019.

[20] Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[21] Shuang Gao and Peter E Caines. Graphon control of large-scale networks of linear systems. *IEEE Transactions on Automatic Control*, 2019.

[22] Jiahui Geng, Zongxiong Chen, Yuandou Wang, Herbert Woisetschlaeger, Sonja Schimmler, Ruben Mayer, Zhiming Zhao, and Chunming Rong. A survey on dataset distillation: Approaches, applications and future directions. *ArXiv preprint*, abs/2305.01975, 2023.

[23] Jianyang Gu, Kai Wang, Wei Jiang, and Yang You. Summarizing stream data for memory-restricted online continual learning. *ArXiv preprint*, abs/2305.16645, 2023.

[24] Ziyao Guo, Kai Wang, George Cazenavette, Hui Li, Kaipeng Zhang, and Yang You. Towards lossless dataset distillation via difficulty-aligned trajectory matching. *ArXiv preprint*, abs/2310.05773, 2023.

[25] Ivan Gutman and Bo Zhou. Laplacian energy of a graph. *Linear Algebra and its Applications*, 414(1):29–37, 2006.

[26] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[27] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

[28] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

https://doi.org/10.52202/075280-2315

[29] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual*, 2020.

[30] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21)*, 2021.

[31] Huawei. Mindspore. https://www.mindspore.cn/, 2020.

[32] Joseph J. Pfeiffer III, Sebastián Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher. Attributed graph models: modeling network structure with correlated attributes. In Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 831–842, 2014.

[33] Svante Janson. Graphons, cut norm and distance, couplings and rearrangements. *New York journal of mathematics*, 2013.

[34] Svante Janson and Persi Diaconis. Graph limits and exchangeable random graphs. *Rendiconti di Matematica e delle sue Applicazioni. Serie VII*, pages 33–61, 2008.

[35] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 66–74, 2020.

[36] 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*, 2022.

[37] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[38] David R. Karger. Random sampling in cut, flow, and network design problems. *Math. Oper. Res.*, 24(2):383–413, 1999.

[39] Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 11102–11118, 2022.

[40] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[41] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[42] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[43] Eric D. Kolaczyk. *Statistical Analysis of Network Data: Methods and Models*. IEEE, 2009.

https://doi.org/10.52202/075280-2315

[44] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 12352–12364, 2022.

[45] Guanzhi Li, Aining Zhang, Qizhi Zhang, Di Wu, and Choujun Zhan. Pearson correlation coefficient-based performance enhancement of broad learning system for stock price prediction. *IEEE Trans. Circuits Syst. II Express Briefs*, 69(5):2413–2417, 2022.

[46] Jianxin Li, Qingyun Sun, Hao Peng, Beining Yang, Jia Wu, and S Yu Phillp. Adaptive subgraph neural network with reinforced critical structure mining. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[47] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theory*, 37(1):145–151, 1991.

[48] Zhiqi Lin, Cheng Li, Youshan Miao, Yunxin Liu, and Yinlong Xu. Pagraph: Scaling gnn training on large graphs via computation-aware caching. In *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020.

[49] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[50] Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. Graph condensation via receptive field distribution matching. *ArXiv preprint*, abs/2206.13697, 2022.

[51] Yanqing Liu, Jianyang Gu, Kai Wang, Zheng Zhu, Wei Jiang, and Yang You. Dream: Efficient dataset distillation by representative matching. *ArXiv preprint*, abs/2302.14416, 2023.

[52] Andreas Loukas. Graph reduction with spectral and cut guarantees. *J. Mach. Learn. Res.*, 20:116:1–116:42, 2019.

[53] Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3243–3252, 2018.

[54] László Lovász. *Large networks and graph limits*, volume 60. IEEE, 2012.

[55] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *WSDM*, pages 779–787, 2021.

[56] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. GOT: an optimal transport framework for graph comparison. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13876–13887, 2019.

[57] Brendan D. McKay, Mehmet Aziz Yirik, and Christoph Steinbeck. Surge: a fast open-source chemical graph generator. *J. Cheminformatics*, 2022.

[58] Facundo Mémoli. Spectral Gromov-Wasserstein distances for shape matching. In *ICCV Workshops*, pages 256–263, 2009.

[59] Matthew W. Morency and Geert Leus. Graphon filters: Graph signal processing in the limit. *IEEE Trans. Signal Process.*, 69:1740–1754, 2021.

https://doi.org/10.52202/075280-2315

[60] Bilal Nehme, Olivier Strauss, and Kevin Loquin. Estimating the variance of a kernel density estimation. In Christian Borgelt, Gil González-Rodríguez, Wolfgang Trutschnig, María Asunción Lubiano, María Ángeles Gil, Przemyslaw Grzegorzewski, and Olgierd Hryniewicz, editors, *Combining Soft Computing and Statistical Methods in Data Analysis, SMPS 2010, Oviedo, Spain, September 29 - October 1, 2010*, volume 77 of *Advances in Intelligent and Soft Computing*, pages 483–490, 2010.

[61] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *9th International Conference on Learning Representations, Virtual Event, Austria, May 3-7, 2021*, 2021.

[62] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, December 6-14, 2021, virtual*, pages 5186–5198, 2021.

[63] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2609–2615, 2018.

[64] Francesca Parise and Asuman Ozdaglar. Graphon games: A statistical framework for network games and interventions. *ArXiv preprint*, abs/1802.00080, 2018.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[66] David Peleg and Alejandro A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989.

[67] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710, 2014.

[68] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams, editors, *Proceedings of the 21th ACM International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 985–994, 2015.

[69] Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. The graphon fourier transform. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, Barcelona, Spain, May 4-8, 2020*, pages 5660–5664, 2020.

[70] Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual*, 2020.

[71] Luana Ruiz, Luiz FO Chamon, and Alejandro Ribeiro. Graphon signal processing. *IEEE Transactions on Signal Processing*, 69:4961–4976, 2021.

https://doi.org/10.52202/075280-2315

[72] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[73] Cosma Rohilla Shalizi and Andrew C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *CoRR*, abs/1004.4704, 2010.

[74] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *9th International Conference on Learning Representations, Virtual Event, Austria, May 3-7, 2021*, 2021.

[75] Mingjia Shi, Yuhao Zhou, Kai Wang, Huaizheng Zhang, Shudong Huang, Qing Ye, and Jiancheng Lv. PRIOR: Personalized prior for reactivating the information overlooked in federated learning. In *Proceedings of the 37th NeurIPS*, 2023.

[76] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual*, 2020.

[77] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

[78] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.

[79] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and Philip S. Yu. Graph structure learning with variational information bottleneck. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, Virtual Event, February 22 - March 1, 2022*, pages 4165–4174, 2022.

[80] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S Yu, and Lifang He. Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *The International Conference of World Wide Web 2021*, pages 2081–2091, 2021.

[81] Qingyun Sun, Jianxin Li, Haonan Yuan, Xingcheng Fu, Hao Peng, Cheng Ji, Qian Li, and Philip S Yu. Position-aware structure learning for graph topology-imbalance by relieving underreaching and over-squashing. In *International Conference on Information and Knowledge Management*, pages 1848–1857, 2022.

[82] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 21076–21089, 2022.

[83] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 990–998, New York, NY, USA, 2008.

[84] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[85] Renato Vizuete, Federica Garin, and Paolo Frasca. The laplacian spectrum of large graphs sampled from graphons. *IEEE Transactions on Network Science and Engineering*, page 1711–1721, Mar 2021.

[86] Kai Wang, Jianyang Gu, Daquan Zhou, Zheng Zhu, Wei Jiang, and Yang You. Dim: Distilling dataset into generative model. *ArXiv preprint*, abs/2303.04707, 2023.

https://doi.org/10.52202/075280-2315

[87] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *CVPR*, 2022.

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

[89] Yili Wang, Kaixiong Zhou, Rui Miao, Ninghao Liu, and Xin Wang. Adagcl: Adaptive subgraph contrastive learning to generalize large-scale graph training.

[90] Max Welling. Herding dynamical weights to learn. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 1121–1128, 2009.

[91] Gert W. Wolf. Facility location: concepts, models, algorithms and case studies. series: Contributions to management science. *Int. J. Geogr. Inf. Sci.*, 25(2):331–333, 2011.

[92] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871, 2019.

[93] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *ArXiv preprint*, 2019.

[94] Xinyue Xia, Gal Mishne, and Yusu Wang. Implicit graphon neural representation. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics,*, 2023.

[95] Hongteng Xu, Dixin Luo, Lawrence Carin, and Hongyuan Zha. Learning graphons via structured gromov-wasserstein barycenters. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, Virtual Event, February 2-9, 2021*, pages 10505–10513, 2021.

[96] Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence. *ArXiv preprint*, abs/2205.09329, 2022.

[97] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 40–48, 2016.

[98] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining, London, UK, August 19-23, 2018*, pages 974–983, 2018.

[99] Haiyang Yu, Limei Wang, Bokun Wang, Meng Liu, Tianbao Yang, and Shuiwang Ji. Graphfm: Improving large-scale GNN training via feature momentum. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 25684–25701, 2022.

[100] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. In *8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[101] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, December 3-8, 2018, Montréal, Canada*, pages 5171–5181, 2018.

https://doi.org/10.52202/075280-2315

[102] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM conference on research and development in Information Retrieval, Virtual Event, China, July 25-30, 2020*, pages 689–698, 2020.

[103] Yifan Zhang, Daquan Zhou, Bryan Hooi, Kai Wang, and Jiashi Feng. Expanding small-scale datasets with guided imagination. *ArXiv preprint*, abs/2211.13976, 2022.

[104] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *ArXiv preprint*, abs/2110.04181, 2021.

[105] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *9th International Conference on Learning Representations, Virtual Event, Austria, May 3-7, 2021*, 2021.

[106] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *9th International Conference on Learning Representations, Virtual Event, Austria, May 3-7, 2021*, 2021.

[107] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, Virtual Event, February 2-9, 2021*, pages 11015–11023, 2021.

[108] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *Proceedings of the 37th International Conference on Machine Learning, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11458–11468, 2020.

[109] 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. In *Advances in Neural Information Processing Systems*, 2023.

[110] Daquan Zhou, Kai Wang, Jianyang Gu, Xiangyu Peng, Dongze Lian, Yifan Zhang, Yang You, and Jiashi Feng. Dataset quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17205–17216, 2023.

[111] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *ArXiv preprint*, 2018.

https://doi.org/10.52202/075280-2315

A More Preliminary and Related Work

A.1 More preliminary

Graphon. A graphon [43, 34, 54] is a bounded, symmetric, and Lebesgue measurable function, denote as $W : \Omega^2 \rightarrow [0, 1]$, where Ω is a probability measure space. By randomly selecting points $\{v_0, v_1, \ldots, v_n\}$ from Ω, we can create a graph of any size by connecting each point with an edge weight $e_{ij} = W(v_i, v_j)$. Formally, we can follow the random sampling process to get arbitrarily sized (n) graphs :

$$\begin{aligned} v_i &\sim \text{Uniform}(\Omega), \text{ for } i = 1, \cdots, n, \\ e_{ij} &\sim \text{Bernoulli}(W(v_i, v_j)), \text{ for } i, j = 1, \cdots, n. \end{aligned} \tag{10}$$

Conventionally, we use a deterministic setting to select the node, *e.g.*, the fixed grid $v_i = \frac{i-1}{n}$. Graphs derived from the same graphon share several important properties for statistical analysis: such as density [21], clustering coefficient [70], and degree distribution [71], which motivate us to broadcast the original graph structure information to the condensed graph via the graphon approximating.

A.2 More related work

Graph structure learning. Graph structure learning [19, 35, 8, 108, 55] also aims at jointly optimizing the graph structure and the corresponding node features. Most of these methods learn the new structure based on the certain constraints (*e.g.*, low-rank [91], sparsity [8], feature smoothness [35], and homophily [108]). However, these methods are unable to learn a new structure with a significantly reduced node size. In contrast, our method can synthesize a smaller graph structure while simultaneously constructing new node features. The obtained small but informative graph can reduce the training cost in the downstream tasks.

B Proofs

B.1 Proof of Proposition 1

Proof. First, the gradient matching objective in GCond [37] can be shown as:

$$\text{Match}(\nabla_{\boldsymbol{\theta}} \mathcal{L}_{cls} \left(\text{GNN}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X}), \mathbf{Y} \right)), \quad \nabla_{\boldsymbol{\theta}} \mathcal{L}_{cls} \left(\text{GNN}_{\boldsymbol{\theta}}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}' \right)), \tag{11}$$

where $\text{Match}(\cdot)$ is the matching function that measures the distance of the two gradients. θ represents the parameters of the GNN, the ∇ denotes the gradient in the backpropagation process, and the $\mathcal{L}_{cls}$ represents the loss function that is used in the supervised tasks, *i.e.*, cross-entropy loss. Following the discussion in the [4, 1, 82], the GNN can be viewed as a bandpass filter in the spectral domain. Expanding the GNN in the spectral domain, we can rewrite the objective as:

$$\Big|\Big| \int_{i=0}^{N} \mathbf{U_A} \, \text{diag}(\text{T}_i(\boldsymbol{\lambda}_\mathbf{A})) \mathbf{U}_\mathbf{A}^\top \mathbf{X} - \int_{j=1}^{N'} \mathbf{U}_{\mathbf{A}'} \, \text{diag} \left(\text{T}_j(\boldsymbol{\lambda}_{\mathbf{A}'}) \right) \mathbf{U}_{\mathbf{A}'}^\top \mathbf{X}' \Big|\Big| \leq \epsilon, \tag{12}$$

where the $\text{T}(\boldsymbol{\lambda})$ denotes the frequency response of the given GNN, the $\boldsymbol{\lambda}$ and the $\mathbf{U}$ is the eigenvalue and eigenvector of the corresponding graph, respectively. Note we drop the ∇, the $\mathcal{L}_{cls}$, and the $\mathbf{Y}$ $(\mathbf{Y}')$ terms because they can be viewed as the intermediate process [105, 107] in approximating to Eq.(12). We use the MSE [37] as the matching function $\text{Match}(\cdot)$ for simplicity.

To simplify $\text{T}(\boldsymbol{\lambda})$, without loss of generality, we assume the bandwidth of the specific GNN's frequency response is (a, b), which intuitively indicates that only the signal of frequency in (a, b) can pass through the filter on the graph. Then the Eq. (12) can be written as:

$$\Big|\Big| \int_{i=0}^{N} \int_{k=a}^{b} \mathbf{U_A} \, \text{diag}({\boldsymbol{\lambda}_\mathbf{A}}^k) \mathbf{U}_\mathbf{A}^\top \mathbf{X} - \int_{j=1}^{N'} \int_{k=a}^{b} \mathbf{U}_{\mathbf{A}'} \, \text{diag}({\boldsymbol{\lambda}_{\mathbf{A}'}}^k) \mathbf{U}_{\mathbf{A}'}^\top \mathbf{X}' \Big|\Big| \leq \epsilon, \tag{13}$$

we further combine the first integration, the objective can thus be summarized as:

$$\Big|\Big| \int_{k=a}^{b} \left(\mathbf{U_A} \, \text{diag}({\boldsymbol{\lambda}_\mathbf{A}}^k) \mathbf{U}_\mathbf{A}^\top \mathbf{X} - \mathbf{U_A} \, \text{diag}({\boldsymbol{\lambda}_{\mathbf{A}'}}^k) \mathbf{U}_{\mathbf{A}'}^\top \mathbf{X}' \right) \Big|\Big| \leq \epsilon. \tag{14}$$

https://doi.org/10.52202/075280-2315

Then following the transformation in the spectral domain [1], we can briefly summarized:

$$\int_{i=a}^{b} ||\bar{x}_i^2 - \bar{x}_i^{'2}|| \leq \epsilon. \tag{15}$$

Take the Eq. (15) into the $\left\|\eta^{\mathcal{G}} - \eta^{\mathcal{S}}\right\|$:

$$\begin{aligned} ||\eta^{\mathcal{G}} - \eta^{\mathcal{S}}|| =& || \sum_{i=1}^{N} \bar{x}_i^2 - \sum_{i=j}^{N'} \bar{x}_i^2 || \\ \geq& \epsilon + || \sum_{i=1}^{a} \bar{x}_i^2 - \sum_{j=1}^{a} \bar{x}_j^2 || + || \sum_{i=b}^{N} \bar{x}_i^2 - \sum_{j=b}^{N'} \bar{x}_j^2 || \\ \geq& \epsilon + \sum_{i=1}^{a} \left\| \bar{x}_i^2 - \bar{x}_i'^2 \right\| + \sum_{i=b}^{N'} \left\| \bar{x}_i^2 - \bar{x}_i'^2 \right\| . \end{aligned} \tag{16}$$

Note the second inequality is under the assumption: intuitively, the overall distance of $\eta^{\mathcal{G}}$ and $\eta^{\mathcal{S}}$ should be larger than the condition that two graphs have the same size (*i.e.*, $N = N'$). □

B.2 Proof of Proposition 2

To prove Proposition 2, we first introduce the following notations and theorems in graphon theory.

The cut norm [20, 54] is defined as:

$$\|W\|_{\square} := \sup_{\mathcal{X}, \mathcal{Y} \subset \Omega} \left| \int_{\mathcal{X} \times \mathcal{Y}} W(x, y) \mathrm{d}x \mathrm{d}y \right|, \tag{17}$$

where the Ω is the probability space of a graphon W, the supremum is taken over all measurable subsets $\mathcal{X}$ and $\mathcal{Y}$ [95]. Then the cut distance between W_1, W_2 [54] can be defined as:

$$\delta_{\square}(W_1, W_2) := \inf_{\phi \in \mathcal{S}_\Omega} \left\| W_1 - W_2^{\phi} \right\|_{\square}, \tag{18}$$

where the $\mathcal{S}_\Omega$ is the set of measure-preserving mappings from Ω to Ω. When two graphons W_1, W_2 have $\delta_{\square}(W_1, W_2) = 0$, they can be seen equivalent, denoted as $W_1 \cong W_2$.

To effectively approximate W in the real world graphs, works [54, 33, 17] introduce an approximation named step function. Let $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_K)$ be a partition of Ω into K measurable sets. The step function $W_{\mathcal{P}} : \Omega^2 \mapsto [0, 1]$ is defined as:

$$W_{\mathcal{P}}(x, y) = \sum_{k, k'=1}^{K} w_{kk'} 1_{\mathcal{P}_k \times \mathcal{P}_{k'}}(x, y), \tag{19}$$

where the $w_{kk'} \in [0, 1]$ and the indicator function $1_{\mathcal{P}_k \times \mathcal{P}_{k'}}(x, y)$ is 1 if $(x, y) \in \mathcal{P}_k \times \mathcal{P}_{k'}$, or it is 0.

To explore the relationship between step function and the W, we introduce the Weak Regularity Lemma as follows.

Theorem 1 (Weak Regularity Lemma [54]). *For every graphon $W \in \mathcal{W}$ and $K \leq 1$, there always exists a step function $W_{\mathcal{P}}$ with $|\mathcal{P}| = K$ steps such that*

$$\|W - W_{\mathcal{P}}\|_{\square} \leq \frac{2}{\sqrt{\log K}} \|W\|_{L_2}. \tag{20}$$

We can further obtain the corollary that $\delta_{\square}(W, W_{\mathcal{P}}) \leq \frac{2}{\sqrt{\log K}} \|W\|_{L_2}$ as the $\delta_{\square}(W, W_{\mathcal{P}}) \leq \|W - W_{\mathcal{P}}\|_{\square}$. Intuitively, we can use any step function to approximate the ideal W of real graphs.

Then to investigate the properties of the graphon in the spectral domain, following [71, 85, 59, 69], we have the conclusion that:

$$||S_W - S_{W_{\mathcal{P}}}|| \leq \|W - W_{\mathcal{P}}\|_{\square}, \tag{21}$$

where the S denotes the signal spectrum of the graphs [85] (*i.e.*, the post-Graph-Fourier-Transform $\mathbf{U}^{\top}\mathbf{X}$), Intuitively, the Eq. (21) shows that when the step function $W_{\mathcal{P}}$ is approximated to the W, they have similar signal spectrum.

Proof. By combining the Theorem 1 and Eq.(21), the Proposition 2 replace the $W_{\mathcal{P}}$ with the $W'_{\mathbf{A}}$, that's because we use the generative model (*i.e.*, $\mathrm{GEN}(\cdot)$) to approximate the step function $W_{\mathcal{P}}$. As we aim to learn the graphon of the original structure $\mathbf{A}$, the graphon W can be rewrite as $W_{\mathbf{A}}$,

$$\left|\left|S_{W_{\mathbf{A}}} - S_{W'_{\mathbf{A}}}\right|\right| \leq \|W'_{\mathbf{A}} - W_{\mathbf{A}}\|_{\square} \,. \tag{22}$$

We notice that the S here is related to the Laplacian energy distribution (LED), where the LED represents the probability distribution of the S, then we use the $\sum_{i=1}^{N} \bar{x}_i$ and $\sum_{j=1}^{N'} \bar{x}_j$ to represent the summation of the original signal spectrum and the condensed one, respectively, we have:

$$\begin{aligned} ||\eta^{\mathcal{G}} - \eta^{\mathcal{S}}|| &= ||\frac{S_{W_{\mathbf{A}}}}{\sum_{i=1}^{N} \bar{x}_i} - \frac{S_{W_{\mathbf{A}'}}}{\sum_{j=1}^{N'} \bar{x}_j}|| \\ &\leq \max(\frac{1}{\sum_{i=1}^{N} \bar{x}_i}, \frac{1}{\sum_{j=1}^{N'} \bar{x}_j}) \left|\left|S_{W_{\mathbf{A}}} - S_{W'_{\mathbf{A}}}\right|\right| \\ &\leq \|W'_{\mathbf{A}} - W_{\mathbf{A}}\|_{\square} \,, \end{aligned} \tag{23}$$

where in the second inequality, we drop the $\max(\cdot)$ term since it always lower than 1. As the $W_{\mathbf{A}'}$ here represents the graphon of the synthetic $\mathbf{A}'$, we can use the $\mathbf{A}'$ directly in the $\delta_{\square}$ to form a compact upper bound.

$$||\eta^{\mathcal{G}} - \eta^{\mathcal{S}}|| \leq \delta_{\square}(\mathbf{A}', W_{\mathbf{A}}). \tag{24}$$

□

Note that minimizing the upper bound has been proven to be equivalent to minimizing the optimal transport distance between the two graphs [95].

B.3 Time complexity analysis and running time

Time complexity. For simplicity, let the number of MLP layers in $\mathrm{GEN}(\cdot)$ be L, and all the hidden units are d. *In the forward process*, we have three steps: first, we calculate the A' by the $\mathrm{GEN}(\cdot)$, which have the complexity of $\mathcal{O}(N'^2 d^2)$. Second, the forward process of GNN on the original graph has a complexity of $\mathcal{O}(m^L N d^2)$, where the m denotes the sampled size per node in training. Third, the complexity of training on the condensed graph is $\mathcal{O}(LN'd)$. *In the backward process*, the complexity of gradient matching strategy (*i.e.*, $\mathcal{L}_{feature}$) is $\mathcal{O}(|\theta||\mathbf{X}'|)$ [37]. For the structure optimization term (*i.e.*, $\mathcal{L}_{structure}$), the complexity is $\mathcal{O}(N'^2 k + NN'^2)$. The overall complexity of SGDD can be represented as $\mathcal{O}(N'^2 d^2) + \mathcal{O}(m^L N d^2) + \mathcal{O}(LN'd) + \mathcal{O}(|\theta||X'|) + \mathcal{O}(N'^2 k + N'^2 N)$. Note $N' \ll N$, we can drop the terms that only involve N' and constants (*e.g.*, the number of L and m). The final complexity can be simplified as $\mathcal{O}(m^L N d^2) + \mathcal{O}(N'^2 N)$, thus the complexity of SGDD still be linear to the number of nodes in the original graph.

Running time. We report the running time of the SGDD in the two datasets: Ogbn-arxiv, and YelpChi. We vary the condensing ratio r in the range of {0.05%, 0.25%, 0.50%} for Ogbn-arxiv and {0.05%, 0.10%, 0.20%} for YelpChi. All experiments are conducted five times on one single A100-SXM4 GPU. We also compare our results to those obtained using GCond [37] under the same settings. As shown in the Tab. 5, our approach achieves a similar running time to GCond when the condensing ratio was low (*i.e.*, $r = 0.05\%$ for both datasets), and is 10% faster when the condensing ratio increased. The difference can be explained by the efficient generative model we employed for generating structure, which prevents the consumption of time-complex operations such as calculating the pair-wised feature similarity ($\mathcal{O}(N'^2)$).

Table 5: Runing time on Ogbn-arxiv and YelpChi for 50 epochs.

Dataset	r	GCond	SGDD	Dataset	r	GCond	SGDD
Ogbn-arxiv	0.05%	315±1.8s	308±1.6s	YelpChi	0.05%	67±2.6s	47±2.8s
	0.25%	413±2.6s	374±3.2s		0.10%	96±2.8s	74±1.7s
	0.50%	527±2.7s	467±2.1s		0.20%	110±0.8s	93±2.6s

C Experimental Details and More Experiments

C.1 Dataset statistics

We evaluate the proposed SGDD on nine datasets, including five node classification datasets: Cora [41], Citeseer [41], Ogbn-arxiv [29], Flickr [100], and Reddit [26]; two anomaly detection datasets: YelpChi [68] and Amazon [102]; two link prediction datasets Citeseer-L [97] and DBLP [83]. We report the dataset statistics in Tab. 6.

Table 6: Dataset statics, including five node classification datasets, two anomaly detection datasets, and two link prediction datasets.

	Datasets	#Nodes	#Edges	#Classes	#Features
ND	Cora [41]	2,708	5,429	7	1,433
	Citeseer [41]	3,327	4,732	6	3,703
	Ogbn-arxiv [29]	169,343	1,166,243	40	128
	Flickr [100]	89,250	899,756	7	500
	Reddit [26]	232,965	57,307,946	210	602
AD	YelpChi [68]	45,954	3,846,979	2	32
	Amazon [102]	11,944	4,398,392	2	25
LP	Citeseer-L [41]	3,327	4,732	2	3,703
	DBLP [83]	26,128	105,734	2	4,057

Citeseer-L: We use the Citeseer in the link prediction setting, named Citeseer-L. We randomly sample 80% nodes in training, 10% nodes in validation, and the remaining 10% nodes for testing. The classes here denote "have edge" and "do not have edge".

DBLP: We treat the original graph as a homogeneous graph here.

C.2 Implementation details

Structure in GDC. We utilize the DC [107] as our baseline, and to incorporate the structure information, we add the constraint to produce a graph structure, named Graph DC (GDC). Specifically, we use the cosine similarity function [8] (formally, $\mathbf{A}'_{ij} = \cos(\mathbf{X}'_i, \mathbf{X}'_j)$) to generate structure, where $\mathbf{X}'_i$ and $\mathbf{X}'_j$ are the learned features obtained through the vanilla gradient matching strategy.

$\mathrm{GEN}(\cdot)$ **in SGDD.** We introduce the $\mathrm{GEN}(\cdot)$ as our generative model in Sec. 4.1. Here, we show the implementation details of this module.

To start, we aim to find a method to broadcast the original graph structure $\mathbf{A}$ to condensed graph $\mathbf{A}'$. Motivated by that all graphs with the same graphon W will exhibit similar properties, we can simply calculate the graphon W of $\mathbf{A}$ and leverage it in the condensing process. Nevertheless, directly calculating W of $\mathbf{A}$ is not feasible since conventional graphon learning methods should summarize graphon from a set of graphs [21, 33, 64] (We only have one graph $\mathbf{A}$). Recent advancements of IGNR [94] demonstrate the potential of utilizing the generative approach to approximate the W. Specifically, given that the graphon W is defined as $\Omega^2 \to [0,1]$ (as described in Appendix A.1), we can similarly construct the function f as follows.

$$f : \mathbb{R}^2 \to [0,1], \tag{25}$$

the continuous space Ω^2 is defined by $\mathbb{R}^2$. For computational convenience, the input space can further be limited to $[0,1]^2$ [94], then the Eq. (25) is transformed to sample points to reconstruct data, following IGNR[94], we use SIREN[76] as f,

$$\begin{aligned}
\mathbf{h}_0 &= \mathrm{PostionalEncoding}(\mathcal{Z}(N')), \\
\mathbf{h}_i &= \mathrm{Sin}(\mathbf{W}_i(\mathbf{h}_{i-1}) + \mathbf{b}_i),\ i = 1, \cdots, l-1, \\
\mathbf{h}_l &= \mathrm{Sigmoid}(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l),
\end{aligned} \tag{26}$$

where the $\mathcal{Z}(N') \in \mathbb{R}^{N' \times N'}$ is a random noise that plays as the coordinates, and the learnable weights $\Phi = \{\mathbf{W}_i \in \mathbb{R}^{l_i \times l_{i+1}}, \mathbf{b}_i \in \mathbb{R}^{l_i}, \quad \text{for } i = 1, \cdots, l\ \}$ map the pair of points to the edge probability. $\mathrm{GEN}(\mathcal{Z}(N'); \Phi) \in \mathbb{R}^{N' \times N'}$ is equal to represent the adjacency matrix $\mathbf{A}' \in \mathbb{R}^{N' \times N'}$

after transformation, where each entry represents a probability that each node pair should be connected. To incorporate the node information into the structure generation, we adopt them as conditional information that leads the generation process. We can then rewrite Eq. (25) to:

$$f : \mathbb{R}^d \times \mathbb{R}^2 \to [0, 1], \tag{27}$$

where the $\mathbb{R}^d$ here is to present the conditional information, thus the learned model considers both the node's coordinates message along with the node's specific information. The basic implements can be:

$$\mathbf{h}_i = \mathrm{MLP}_i(\mathbf{X}' \ \oplus \mathbf{Y}') \ \oplus \mathrm{Sin}(\mathbf{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i), \ \ i = 1, \cdots, l, \tag{28}$$

where the $\oplus$ denotes the concatenate operation, the $\mathbf{Y}'$ is treated as a one-hot vector for dimensionality fit through a multilayer perceptron (MLP). This method allows for the incorporation of significant node information into the resulting synthetic graph.

Condensation stage. For GCond [37], we use the 2-layer SGC [92] to serve as the condensing architecture with 256 units, and tune the number of epochs in a range of {400, 500, 6000, 1000, 2000}. For GDC [107, 105], we tune the number of hidden layers in the range of {1, 2, 3} and the number of hidden units in the range of {128, 256}. For SGDD, we use the GCN [40] as the default condensing architecture and tune the number of hidden layers in a range of {1, 2, 3}. We further tune the number of epochs in a range {400, 500, 600, 1000, 2000}, and tune the learning rate in a range of {0.1, 0.01, 0.001, 0.0001}.

Evaluation stage. We set the training epoch to 1000 with an early stopping strategy for evaluating GNNs and set the dropout rate to 0 with the learning rate of 0.1.

Configurations. We conduct all experiments with:

- Operating System: Ubuntu 20.04 LTS.
- CPU: Intel(R) Xeon(R) Platinum 8358 CPU@2.60GHz with 1TB DDR4 of Memory.
- GPU: NVIDIA Tesla A100 SMX4 with 40GB of Memory.
- Software: CUDA 10.1, Python 3.8.12, PyTorch [65] 1.7.0.

C.3 Objective loss function and training algorithm

In this subsection, we present the objective function and provide a detailed training algorithm. Our objective is to jointly learn $\mathbf{X}'$ and $\mathbf{A}'$. We follow GCond[37] to optimize $\mathbf{X}'$ as a free parameter using the gradient matching strategy[104], the loss can be expressed by Eq. (29),

$$\mathcal{L}_{feature} = \mathrm{Match}(\nabla_{\boldsymbol{\theta}} \mathcal{L}_{cls}\left(\mathrm{GNN}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X}), \mathbf{Y}\right), \quad \nabla_{\boldsymbol{\theta}} \mathcal{L}_{cls}\left(\mathrm{GNN}_{\boldsymbol{\theta}}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}'\right)). \tag{29}$$

Here, $\mathrm{Match}(\cdot)$ is the matching function that measures the distance of the two gradients, we use the MSE [37] in practice. θ represents the parameters of the specific backbone GNN, the ∇ denotes the gradient in the backpropagation process, and the $\mathcal{L}_{cls}$ represents the loss function that is used in the supervised tasks, *i.e.*, cross-entropy loss. Our objective loss function can be written as:

$$\mathcal{L} = \mathcal{L}_{feature} + \alpha \ \mathcal{L}_{strcuture} + \beta \ ||\mathbf{A}'||_2, \tag{30}$$

where the α controls the contribution of the $L_{strcuture}$ term. To model the low-rank properties of real-world graphs, we use the $||\mathbf{A}'||_2$ as a regularity to control the sparsification of $\mathbf{A}'$ with β.

We summarize our pipeline in Algorithm. 1.

C.4 Stochastic Block Model experments setting

In Sec. 5.4, we generate a synthetic graph dataset with different community structures using the Stochastic Block Model (SBM) (N, C, p, q) [28]. Here, we show the parameters settings, we set the number of nodes N to 100, while setting the number of communities C to 5. The parameter p represents the edge probability within the same community and q represents the edge probability between communities, we set them to 0.8 and 0.1 in practice, respectively.

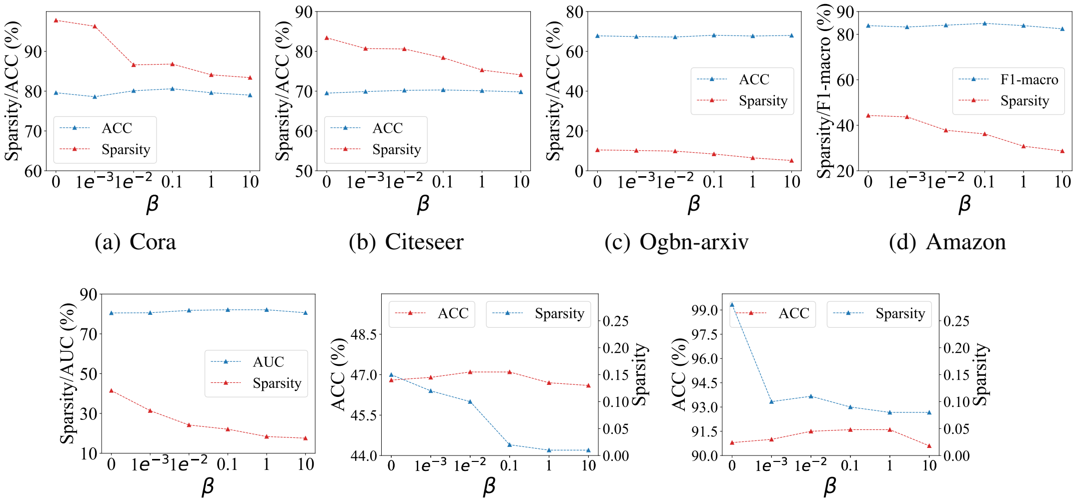
https://doi.org/10.52202/075280-2315

Algorithm 1: SGDD for Graph Condensation

```
Input: Training data $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$, pre-defined condensed labels $\mathbf{Y}'$
Initialize GEN as the structure learning model
Initialize $\mathbf{X}'$ by randomly selecet node feature from each class
for $k = 0, \ldots, K-1$ do
    Randomly initialize GNN$_\theta$
    for $t = 0, \ldots, T-1$ do
        $D' = 0$
        for $c = 0, \ldots, C-1$ do
            Initialize $\mathcal{Z}(N')$
            Compute $\mathbf{A}' = \text{GEN}(\mathcal{Z}(N') \oplus \mathbf{X}' \oplus \mathbf{Y}'; \Phi)$ then $\mathcal{S} = \{\mathbf{A}', \mathbf{X}', \mathbf{Y}'\}$
            Sample $(\mathbf{A}_c, \mathbf{X}_c, \mathbf{Y}_c) \sim \mathcal{G}$ and $(\mathbf{A}'_c, \mathbf{X}'_c, \mathbf{Y}'_c) \sim \mathcal{S}$
            Compute $\mathcal{L}_{structure}$                    ▷ detailed in Eq. (9)
            Compute $\mathcal{L}_{feature}$                     ▷ detailed in Eq. (29)
            $D' \leftarrow D' + \mathcal{L}_{feature} + \alpha\mathcal{L}_{structure} + \beta||A'||_2$
        if $t\%(\tau_1 + \tau_2) < \tau_1$ then
            Update $\mathbf{X}' \leftarrow \mathbf{X}' - \eta_1 \nabla_{\mathbf{X}'} D'$
        else
            Update $\Phi \leftarrow \Phi - \eta_2 \nabla_\Phi D'$
        Update $\boldsymbol{\theta}_{t+1} \leftarrow \text{opt}_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t, \mathcal{S}, \tau_{\boldsymbol{\theta}})$          ▷ $\tau_{\boldsymbol{\theta}}$ is the number of steps for updating $\boldsymbol{\theta}$
$\mathbf{A}' = \text{GEN}(\mathcal{Z}(N') \oplus \mathbf{X}' \oplus \mathbf{Y}'; \Phi)$
$\mathbf{A}'_{ij} = \mathbf{A}'_{ij}$ if $\mathbf{A}'_{ij} > 0.5$, otherwise 0
Return: $(\mathbf{A}', \mathbf{X}', \mathbf{Y}')$
```

C.5 More explorations of the sensitivity of β

In Fig. 4(d), we demonstrate that SGDD is not sensitive to the coefficient β on the YelpChi dataset. Here, we provide more experiments on the other datasets. In Fig. 6, we can see that increasing β leads to the more sparsity of the condensed graph, and the corresponding performance does not have severe drop. These observations demonstrate the effectiveness of the regularity term in our objective.



(a) Cora (b) Citeseer (c) Ogbn-arxiv (d) Amazon

(e) DBLP (f) Flickr (g) Reddit

Figure 6: Evaluations of β in seven datasets. For the Flickr and Reddit datasets, due to the datasets' low sparsity, we use the shared x-axis form for the Flickr and Reddit figures.

https://doi.org/10.52202/075280-2315

Table 7: Comparison of the cross-architecture generalization performance between GCond and SGDD on YelpChi. **Bold entries** are the best results. ↑/↓: our method show increase or decrease performance.

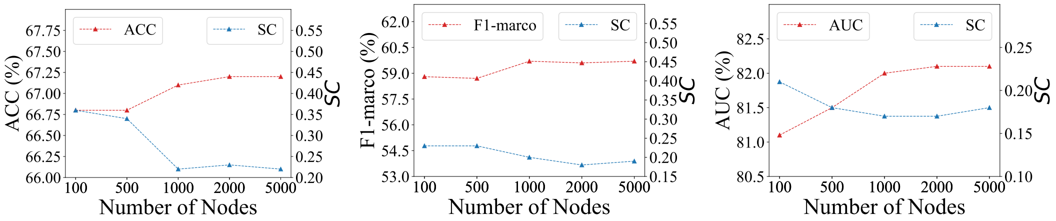
C/T	APPNP	Cheby	GCN	SAGE	SGC
	GCond / SGDD	GCond / SGDD	GCond / SGDD	GCond / SGDD	GCond / SGDD
APPNP	48.1 / **55.1**↑	46.5 / **57.4**↑	50.1 / **58.6**↑	46.7 / **57.1**↑	49.6 / **57.6**↑
Cheby	48.0 / **56.2**↑	45.9 / **56.8**↑	49.8 / **58.7**↑	46.8 / **58.3**↑	49.8 / **58.4**↑
GCN	47.6 / **56.5**↑	46.6 / **56.8**↑	48.6 / **59.7**↑	47.4 / **57.6**↑	50.1 / **57.4**↑
SAGE	46.7 / **57.6**↑	46.8 / **57.5**↑	48.9 / **58.7**↑	48.6 / **58.6**↑	48.9 / **58.6**↑
SGC	47.6 / **57.6**↑	47.7 / **57.2**↑	48.6 / **57.8**↑	47.4 / **59.0**↑	48.7 / **57.6**↑

C.6 More cross-architecture experiments

In this subsection, we further report the results of cross-architecture experiments on the YelpChi. As shown in Tab. 7, our SGDD improves the performance in all cases, the average improvements compared to the GCond [37] is 9.6%, which indicates the strong generalization performance of the condensed graph by SGDD.

C.7 Ablation of the sampling operation in OT distance

To further reduce condensing time and memory usage, we follow GCond [37] to sample from original graph $\mathbf{A}$ in the condensing stage (line 11 in the Algorithm 1). We evaluate the SGDD on various sample sizes, *i.e.*, {100, 500, 1000, 2000, 5000}, and report the corresponding performance and SC. As shown in Fig. 7, with the increase in the number of nodes, the performance obtains marginal improvements while the SC is decreasing. However, larger sample sizes are not always beneficial to performance. In this study, the highest results are obtained when the sample size is 2000. This result empirically demonstrates the scalability of the structure learning module. Specifically, the module enables efficient condensing on large graphs.



(a) Ogbn-arxiv (b) YelpChi (c) DBLP

Figure 7: Evaluation of the number of sampled nodes to the performance.

C.8 More discussion of the condensed graphs.

We next show the comparison of condensed graphs and original graphs. As shown in Tab. 8, the condensed graphs obviously have fewer nodes and edges, while maintaining comparable accuracy, which shows the effectiveness of our SGDD . We also represent the visualizations of some condensed graphs. In Fig. 8, the black lines denote that edge weights are larger than 0.5 and the gray lines represent as smaller than 0.5. We can observe that there are some patterns in the condensed graph, *e.g.*, the homophily patterns on Cora and Citeseer. However, for the remaining datasets, the visualizations are inadequate in revealing the properties of the condensed graph, which proves the superiority of analyzing structure in spectral view.

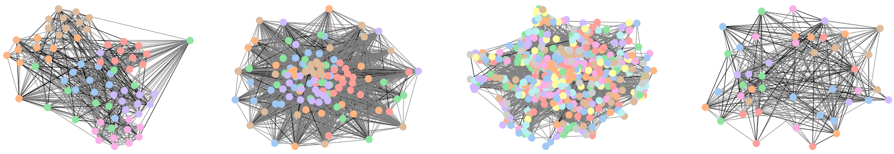
C.9 More discussion of the large graphs.

We conduct experiments on the Ogbn-mag datasets and show the details as follows: (note we transform the original heterogeneous graph to the homogeneous by ignoring difference of

https://doi.org/10.52202/075280-2315

Table 8: The comparison between condensed graphs and original graphs.

	Citeseer, r=0.9%		Cora, r=1.3%		Ogbn-arxiv, r=0.5%		Flickr, r=0.1%		Reddit, r=0.1%	
	Whole	SGDD	Whole	SGDD	Whole	SGDD	Whole	SGDD	Whole	SGDD
Accuracy	70.7	69.5	81.5	79.6	71.4	65.3	47.1	47.1	94.1	90.5
#Nodes	3k3	60	2k7	70	169k	454	44k	44	153k	153
#Edges	4k7	1k	5k4	2k	1,166k	8k6	218k	331	10,753k	3k
Storage(MB)	47.1	0.8	14.9	0.4	100.4	1.0	86.8	0.1	435.5	0.7



(a) Cora (b) Citeseer (c) Ogbn-arxiv (d) Flickr

Figure 8: Visualizations of condensed graphs, the different colors on the graphs denote the classes.

https://doi.org/10.52202/075280-2315