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 $W : \Omega^2 \to [0, 1]$, 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:

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

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:

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

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

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

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 $\mathcal{O}(N'^2d^2)$. 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 $\mathcal{O}(N'^2k + NN'^2)$. The overall complexity of SGDD can be represented as $\mathcal{O}(N'^2d^2)+\mathcal{O}(m^L N d^2)+\mathcal{O}(LN'd)+\mathcal{O}(|\theta||X'|)+\mathcal{O}(N'^2k+N'^2N)$. 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:

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

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),

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

Here, $\text{Match}(\cdot)$ 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

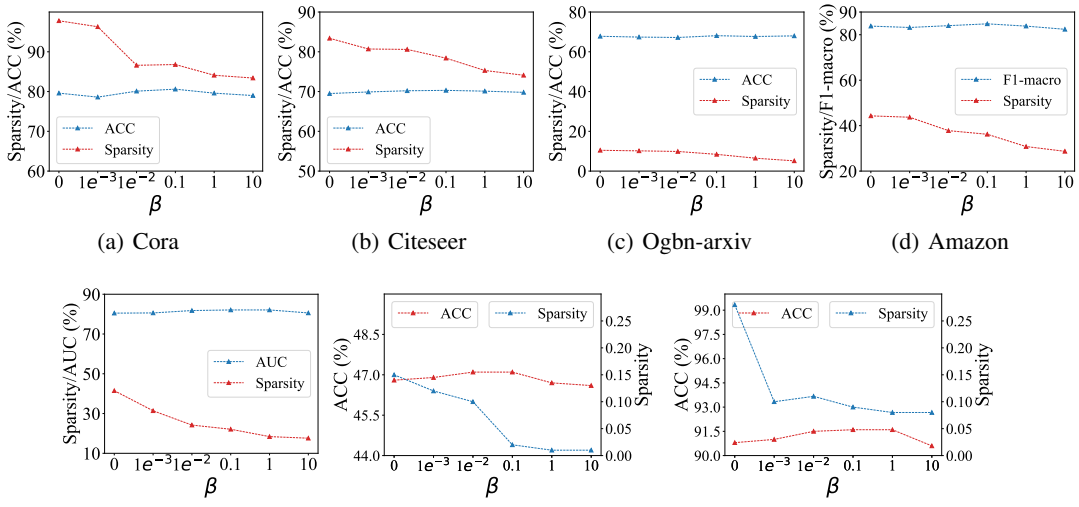
In Sec. 5.4, we generate a synthetic graph dataset with different community structures using the Stochastic Block Model (SBM) (N, C, p, q) [28]. Here, we show the parameters settings, we set the number of nodes N to 100, while setting the number of communities C to 5. The parameter p represents the edge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

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_θ
  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.

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.

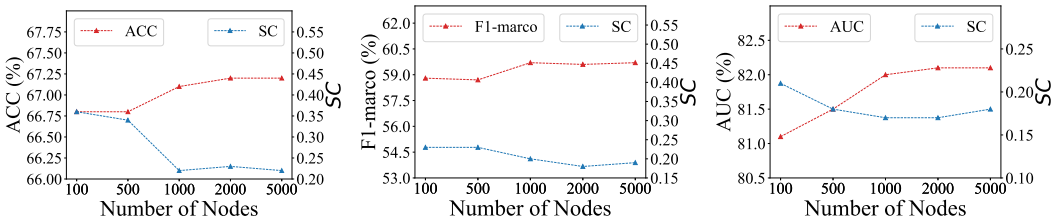
C/T	**APPNP**	**Cheby**	**GCN**	**SAGE**	**SGC**
	GCond / SGDD	GCond / SGDD	GCond / SGDD	GCond / SGDD	GCond / SGDD
APPNP	48.1 / **55.1**↑	46.5 / **57.4**↑	50.1 / **58.6**↑	46.7 / **57.1**↑	49.6 / **57.6**↑
Cheby	48.0 / **56.2**↑	45.9 / **56.8**↑	49.8 / **58.7**↑	46.8 / **58.3**↑	49.8 / **58.4**↑
GCN	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.

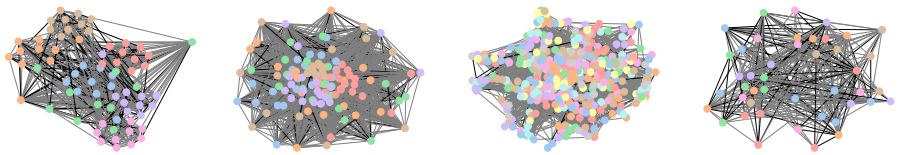
We next show the comparison of condensed graphs and original graphs. As shown in Tab. 8, the condensed graphs obviously have fewer nodes and edges, while maintaining comparable accuracy, which shows the effectiveness of our SGDD . We also represent the visualizations of some condensed graphs. In Fig. 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

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.