Similarity-Guided Structural Matching Learning for Graph Dataset Condensation

Yiyang Zhang[1] , **Yutong Ye**[1] , **Yingbo Zhou**[2] , **Nan Zhang**[1] , **Xiang Lian**[3] , **Mingsong Chen**[1]
[1]MoE Eng. Research Center of SW/HW Co-Design Tech. and App., East China Normal University
[2]College of Computer Science and Artificial Intelligence, Fudan University
[3]Department of Computer Science, Kent State University

Abstract

As graph repositories grow in scale and diversity, training Graph Neural Networks (GNNs) becomes computationally demanding. However, existing graph condensation methods often fail to retain the intrinsic structural patterns of the original graphs, which are essential in graph-based learning. Therefore, these methods suffer from limited performance and poor generalization in downstream tasks due to the loss of structural information. To address this, we propose Similarity-guided Structural Matching Learning for Graph Dataset Condensation (SSGDC), which efficiently reduces repository size while maintaining both task performance and structural information. Our approach introduces a similarity-based graph selector to identify high-quality subsets for condensation. The condensed graphs are optimized with a dual-objective loss that combines gradient matching for task alignment with a metric-learning loss for structural preservation within the selected subset. This ensures that the condensed dataset retains both task-relevant information and the essential relational topology that supports GNN training and enhances generalization. Experiments demonstrate that our method achieves higher accuracy and better structure retention across varying condensation ratios, highlighting the critical role of structural preservation in graph dataset condensation.

1 Introduction

Graph-structured data have become ubiquitous in modern machine learning applications [Gilmer *et al.*, 2017], covering diverse fields such as social network analysis [Liu *et al.*, 2021], bioinformatics [Fout *et al.*, 2017; Huang *et al.*, 2023], compound modeling, and recommendation systems [Fan *et al.*, 2019; Ying *et al.*, 2018]. These datasets encode complex relational patterns that are critical to advancing scientific and industrial applications. At the same time, as artificial intelligence advances, neural networks exhibit strong expressive power for graph data [Hamilton *et al.*, 2017]. Therefore, *Graph Neural Networks* (GNNs) [Kipf and Welling, 2017; Velickovic *et al.*, 2018] have been proposed and rapidly developed. However, training GNNs isincreasingly challenging as dataset sizes have grown significantly in recent years [Wu *et al.*, 2021; Bojchevski *et al.*, 2020]. For example, the ogbg-ppa dataset [Hu *et al.*, 2020] has 158,100 graphs, 243.4 nodes, and 2,266.1 edges on average.

Training high-performance GNNs [Yang *et al.*, 2024] on such large real-world graphs requires substantial computational resources and time, which are unacceptable to many researchers. As a result, many graph condensation methods have been proposed, such as trajectory matching [Jin *et al.*, 2022a; Gao *et al.*, 2024], distribution matching [Zhao and Bilen, 2023], and kernel-based distillation [Xu *et al.*, 2023]. They address this challenge by synthesizing a significantly smaller, highly informative set of graphs that preserves the essential structural and semantic information of the original large graph repository [Hashemi *et al.*, 2024; Dai *et al.*, 2019]. The ultimate goal is to enable GNNs trained on this small set of synthetic graphs to achieve performance comparable to that of GNNs trained on large-scale data [Lei and Tao, 2024; Khoshraftar and An, 2024].

Although existing condensation methods [Sun *et al.*, 2024] have shown promise in reducing training costs, they often incur high memory costs in the condensing phase and fail to preserve the structural information in the original dataset. Figure 1 shows the graphs generated by the current method and our method. The triplets in the figure represent the average degree, average number of triangles, and average clustering coefficient of each graph repository, respectively. We can see that the graph compressed by the current method retains almost no structural information. They primarily focus on optimizing the synthetic dataset by aligning the train-

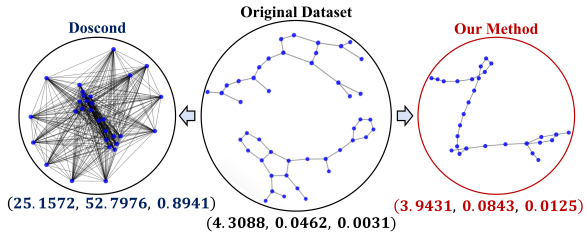

Figure 1: Condensed graphs by existing methods(left) and samples for original dataset (right) in NCI1.

ing gradients between the original large graph repository and the condensed graph set within each class [Jin *et al.*, 2022a]. Although effective in preserving task-related characteristics, this core optimization strategy suffers from a critical limitation. It largely neglects crucial structural information during gradient matching, resulting in insufficient discrimination in the condensed graph set [Gao *et al.*, 2025]. Therefore, these methods fail to explicitly encourage the condensed graph to learn representations from its structure that can effectively distinguish among different classes. This leads to insufficient discriminative power in the synthesized graph, hindering optimal performance on downstream tasks. This shortcoming becomes particularly pronounced when dealing with complex or large-scale graph repositories [Ma *et al.*, 2025]. Therefore, *how to condensate a large graph repository to higher quality while preserving structural information* is becoming a major challenge in graph dataset condensation.

In this work, we propose a novel approach, Similarity-guided Structural Matching Learning for Graph Dataset Condensation (SSGDC), that addresses these limitations. First, we develop a similarity-based mechanism that intelligently selects high-quality graph subsets from the original repository, identifying the most representative samples for structure matching. This selective approach ensures that the condensation process focuses on informative graphs rather than treating all samples equally. Second, we formulate a dual-objective optimization framework that combines gradient matching with structural preservation via Fused Gromov-Wasserstein (FGW) alignment. This co-optimization strategy ensures that the condensed repository maintains both the training dynamics and structural characteristics of the original dataset. Moreover, our approach leverages GNNs for feature extraction and employs latent-space optimization to address the discrete nature of graph data. By optimizing in continuous latent spaces and decoding back to graph structures, our method overcomes the inherent challenges of directly optimizing the graph's discrete structure matrix while fully preserving structural integrity. This paper makes the following four major contributions:

- We introduce the novel framework for graph repository condensation that simultaneously addresses structural preservation and feature distillation through bi-level optimization in the latent space.
- We propose a novel similarity-based pruning mechanism that identifies high-quality graph subsets for targeting structure matching, improving the efficiency and effectiveness of the condensation process.
- We develop a dual-loss optimization strategy that combines gradient matching with FGW-based structural alignment, ensuring that condensed graphs preserve both task-relevant information and structural relationships.
- We conduct extensive experiments across multiple real-world graph repositories, demonstrating that our approach outperforms other methods across a range of condensation ratios and preserves structural information.

2 Related Work

Graph Condensation. For the condensation of large single graph datasets [Jin *et al.*, 2022b], existing methods can be divided into three categories: trajectory matching, distribution matching, and gradient matching. Trajectory matching methods match the entire training trajectory of models trained on original and condensed data [Zheng *et al.*, 2023; Zhang *et al.*, 2024]. The distribution matching methods [Liu *et al.*, 2022] align the feature distributions of the condensed and original data in a latent space [Liu *et al.*, 2023]. Gradient matching methods [Yin *et al.*, 2025; Fang *et al.*, 2024] align training gradients between the original and condensed data, ensuring similar optimization behaviors [Jin *et al.*, 2022b; Jin *et al.*, 2022a]. However, these graph condensation methods, designed for single large networks, are often inadequate for multi-graph datasets, as their independent application to each graph fails to account for the critical dimension of inter-graph relationships.

Condensation for the Graph Repository. Existing approaches can be broadly categorized into sampling-based methods [Peleg and Schäffer, 1989; Karger, 1994] and condensation-based [Loukas, 2019; Loukas and Vandergheynst, 2018] methods. They employ distinct strategies to reduce dataset size while preserving essential information for downstream tasks. Sampling-based approaches [Sener and Savarese, 2018a] represent the most straightforward compression strategy, selecting representative subsets from the original graph repository. Random sampling [Ribeiro and Towsley, 2010] serves as the simplest baseline, uniformly selecting graphs without considering their structural or feature characteristics. Herding methods [Welling, 2009] offer a more principled alternative by sequentially selecting samples that minimize the distance to the mean of all samples in the feature space. These approaches [Sener and Savarese, 2018b] fail to capture the diversity and representativeness of the original repository. Condensation-based methods [Liu *et al.*, 2022; Zheng *et al.*, 2023] represent a more sophisticated approach that generates synthetic graphs rather than simply selecting existing ones. DosCond [Jin *et al.*, 2022a] was the first to apply gradient matching to graph dataset condensation, thereby extending the dataset condensation framework to graph-structured data repositories. KiDD [Xu *et al.*, 2023] employs a kernel-based approach using graph neural tangent kernels (GNTK) to approximate the training behavior of GNNs. Mirage [Gupta *et al.*, 2024; Liu *et al.*, 2023] introduces a model-agnostic graph distillation framework based on frequent computational tree patterns. Their condensed graphs perform well on downstream tasks. However, they have critical limitations for graph repository condensation [Wang *et al.*, 2024; Xiao *et al.*, 2025] because they do not consider the structural information of the original graphs. Therefore, it remains difficult to improve the quality of compressed graphs.

3 Our Method

Figure 2 details the workflow of our proposed SSGDC method. First, we apply graph data pruning based on similarity to make the sizes of the original and compressed datasets

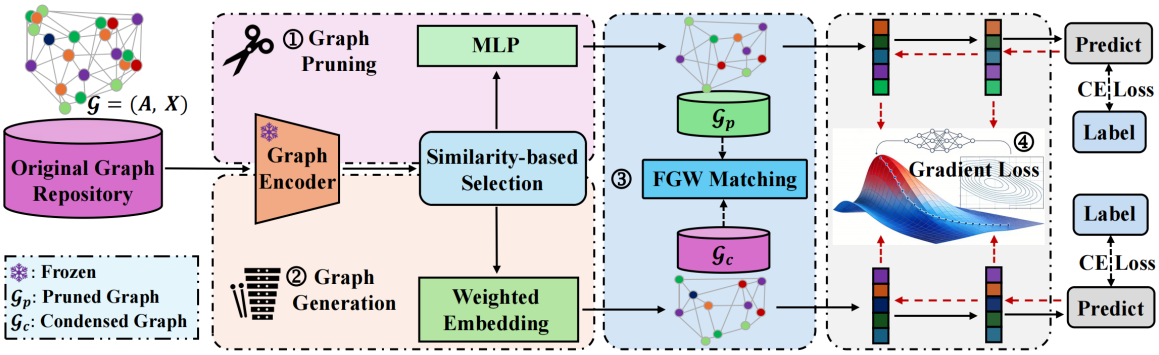

Figure 2: Workflow of our SSGDC approach

more similar for subsequent structural information matching. Next, we introduce the module for generating condensed graphs. By weighting graph-embedding vectors with similarity scores, we obtain more informative graph representations, which are then reconstructed into graph structures via a decoder. Furthermore, we introduce structure information-matching and gradient-matching modules, enabling the training process to effectively preserve both structural properties and task-discriminative information.

3.1 Similarity-based Graph Pruning

Prior to the formal training phase, we perform a pruning step to extract a representative subset from the original dataset. The size of this subset is designed to match the scale of the subsequently condensed dataset, thereby providing a properly scaled and information-rich sample foundation for structure matching. Because not all graphs contribute equally to the learning objective, some may contain redundant information, whereas others may serve as crucial exemplars of particular structural patterns or feature distributions. By leveraging the expressive power of multi-head linear layers, our method captures diverse aspects of graph importance across multiple representation subspaces, enabling a more effective pruning process than conventional approaches.

Given an original graph dataset $\mathcal{G} = \{g_1, g_2, \ldots, g_N\}$ where each graph $g_i = (\mathbf{A}_i, \mathbf{X}_i)$ consists of an adjacency matrix $\mathbf{A}_i \in R^{n_i \times n_i}$ and node feature matrix $\mathbf{X}_i \in R^{n_i \times d}$, we first obtain graph-level representations through a graph neural network encoder:

$$\mathbf{h}_i = f_{\text{encoder}}(\mathbf{A}_i, \mathbf{X}_i), \tag{1}$$

where $\mathbf{h}_i \in R^{d_h}$ is the graph embedding for G_i. The encoding process involves multiple layers of feature transformation and neighborhood aggregation, allowing the model to capture both local substructure patterns and global graph properties. We then apply a global pooling operation to aggregate node-level representations into a single graph-level embedding vector. These final graph embeddings serve as the input features for our similarity-based graph pruning module.

The core of our pruning module uses multi-head linear layers to compute graph similarity scores. For each head $m \in 1, 2, \ldots, M$, we first project the input graph embeddings into two distinct spaces. These projections are achieved through separate linear transformations, each with its own set of learnable parameters $\mathbf{W}$:

$$[\mathbf{H_1}_i^m,\ \mathbf{H_2}_i^m] = h_i[W_1^m,\ W_2^m], \tag{2}$$

where $\mathbf{W}_1^m, \mathbf{W}_2^m \in R^{d_h \times d_k}$ are learnable projection matrices for head m, and $d_k = d_h/M$ is the dimension per head. The vector $\mathbf{H_1}$ represents the current graphs for which we want to compute importance, while the vector $\mathbf{H_2}$ represents all graphs in the dataset that can be attended to. The similarity scores between graph i and graph j in head m can be computed as a scaled dot product of their representations:

$$e_{ij}^m = \frac{(\mathbf{H_1}_i^m)^\top \mathbf{H_2}_j^m}{\sqrt{d_k}}, \tag{3}$$

where the similarity computation begins by measuring the compatibility between each vector pair through a scaled dot-product operation. This produces raw similarity scores indicating how much focus each graph should place on each other graph in the dataset, from the perspective of the current similarity head. These raw scores are then normalized using the softmax function to create proper probability distributions that sum to one. We then obtain the similarity weights using softmax normalization, which can be expressed as:

$$\alpha_{ij}^m = \frac{\exp(e_{ij}^m)}{\sum_{k=1}^N \exp(e_{ik}^m)}, \tag{4}$$

The resulting similarity weights determine how much information from each graph's value vector to incorporate into the output representation for each target graph. The output for graph i from head m is computed as:

$$\mathbf{z}_i^m = \sum_{j=1}^N \alpha_{ij}^m \mathbf{h}_j^m. \tag{5}$$

Following the independent similarity computations across all heads, we aggregate the outputs of the multiple similarity heads to form a comprehensive representation for each graph. The standard approach concatenates the output vectors from all heads and projects them through a final linear transformation. This aggregation step combines the diverse perspectives captured by different linear heads into a unified representation

that encapsulates multiple facets of each graph's characteristics and its relationship to other graphs in the dataset:

$$\mathbf{z}_i = \text{Concat}(\mathbf{z}_i^1, \mathbf{z}_i^2, \ldots, \mathbf{z}_i^M)\mathbf{W}_O + \mathbf{b}_O, \tag{6}$$

where $\mathbf{W}_O \in R^{Md_k \times d_h}$ is the output projection matrix. The aggregated representations are subsequently fed into a scoring network that computes a scalar importance score for each graph. This scoring network consists of several fully connected layers with non-linear activation functions, followed by a final projection to a single-dimensional output. The importance scores quantitatively reflect the estimated value of each graph for the compression objective, with higher scores indicating that a graph contains more essential information to preserve during condensation:

$$s_i = \sigma(\mathbf{w}_s^\top \text{ReLU}(\mathbf{W}_s \mathbf{z}_i + \mathbf{b}_s) + b_s), \tag{7}$$

where $\mathbf{W}_s \in R^{d_s \times d_h}$, $\mathbf{w}_s \in R^{d_s}$, and σ is the sigmoid function. Then we need to select graphs with a higher score. We can clearly extract the graphs with higher similarity. Traditional top-k pruning operations are inherently non-differentiable, which would prevent gradient flow from the compression objectives back to the similarity parameters. We address this challenge by implementing a relaxed pruning mechanism that approximates the discrete top-k operation with a continuous, differentiable function. The pruning probability for each graph is defined as:

$$p_i = -\frac{\exp(s_i/\tau)}{\sum_{j=1}^{N} \exp(s_j/\tau)}, \tag{8}$$

where τ is a temperature parameter that controls the smoothness of pruning. The final selected subset $\mathcal{G}_p$ is obtained by:

$$\mathcal{G}_p = \text{TopK}(p_1, p_2, \ldots, p_N, k). \tag{9}$$

The pruning module outputs both the indices of selected graphs and their corresponding pruning weights, which can be used to weight each selected graph's influence during the subsequent gradient-matching phase. This differentiable formulation enables optimization of the similarity parameters with respect to the final compression quality, ensuring that the pruning criteria align with the overall learning objective.

3.2 Similarity-Guided Graph Aggregation and Generation

After obtaining the similarity matrix, we apply the derived similarity scores to weight the graph embedding vectors. Since it is not feasible to perform gradient descent directly on the adjacency matrix, we employ a GNN model f_{decoder} that captures the underlying structural distribution of the original graph. This approach ensures that the generated graph adheres to the distribution of real graph structures.

For each selected graph g_k ($k \in \{1, 2, \ldots, K\}$), we extract the corresponding similarity vector $\mathbf{s}_k \in R^N$ from the k-th row of the similarity matrix $\mathbf{S}$. This similarity vector encodes the relative importance of each original graph with respect to the current selected graph g_k. The aggregated embedding for the compressed graph corresponding to selected graph g_k is computed by Equation 5. This weighted aggregation ensures that the compressed graph embedding $\mathbf{z}_k$ captures the most relevant information from the entire dataset as determined by the similarity mechanism.

The aggregated embeddings $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K\}$ serve as the latent representations for the compressed graphs. Then we employ a graph decoder $f_{\text{decoder}} : R^d \to \mathcal{G}$ that maps the compressed embeddings back to the graph domain. The reconstruction process involves generating both the adjacency structure and node features for each compressed graph. For a compressed graph embedding $\mathbf{z}_k$, the decoding operation is formalized as follows:

$$\hat{G}_k = f_{\text{decoder}}(\mathbf{z}_k) = (\hat{\mathbf{A}}_k, \hat{\mathbf{X}}_k), \tag{10}$$

where $\hat{\mathbf{A}}_k \in R^{n_c \times n_c}$ is the reconstructed adjacency matrix and $\hat{\mathbf{X}}_k \in R^{n_c \times d_f}$ represents the reconstructed node feature matrix for the compressed graph. Here, n_c denotes the number of nodes in the compressed graph (typically smaller than the original graph), and d_f is the feature dimension. Since the decoder f_{decoder} has been trained to generate graphs that conform to the distribution of the original dataset, its parameter remains fixed throughout the process, ensuring that the reconstructed graph retains intra-graph information.

3.3 FGW Matching for Graph Repositories

After initializing the condensed dataset, we perform structure matching between it and a subset of the original dataset. To provide a foundational account of graph similarity, we adopt the structured data formalism proposed by Vayer et al. In this framework, a graph $\mathcal{G}$ is not only a set of nodes and edges but also a structured object that explicitly decouples feature and structural information. Formally, we define a graph G with N nodes as a triplet (A, X, h). $X \in R^{N*d}$ is the matrix of all features, and $A \in R^{N*N}$ is a matrix that encodes a notion of similarity between nodes. $h = (h_1, ..., h_N)$ is a histogram that assigns an importance weight to each node. Without prior knowledge, uniform weights can be chosen so that h = $\frac{1}{N}1_N$. This representation provides a unified lens for viewing graphs as probability distributions over coupled feature-structure spaces, enabling comparison via optimal transport. The Wasserstein distance is well-suited for comparing features. Let $G_1 = (A_1, X_1, h_1)$ and $G_2 = (A_2, X_2, h_2)$ be two structured graphs. The Wasserstein distance can be formulated as follows:

$$E_W(\pi) = \sum_{i,j} d(X_1^i, X_2^j)^q \pi_{i,j}, \tag{11}$$

where the coupling $\pi \in \Pi(h_1, h_2)$ is a joint distribution, where $\pi_{i,j}$ indicates how much node i in G_1 is matched to node j in G_2. The coupling π that achieves the infimum is called the *optimal transport plan*. The Gromov-Wasserstein distance is well-suited for comparing structures. Let $C_1 \in R^{n \times n}$, $C_2 \in R^{m \times m}$ be the intra-graph structural distance matrices where $C_1(i,k) = d_A(A_1^i, A_1^k)$ and $C_2(j,l) = d_A(A_2^j, A_2^l)$. The Gromov-Wasserstein distance can be formulated as follows:

$$E_{GW}(\pi) = \sum_{i,j,k,l} |C_1(i,k) - C_2(j,l)|^q \pi_{i,j} \pi_{k,l}. \tag{12}$$

The structure cost $|C_1(i,k) - C_2(j,l)|$ punishes matching pairs of nodes (i,k) and (j,l) if the internal structural distance within G_1 ($C_1(i,k)$) differs from that within G_2 ($C_2(j,l)$). This ensures that, if two nodes are matched, their relational contexts within their respective graphs are similar.

While GW provides a powerful framework for comparing structures, it entirely ignores node-level features and attributes that may be present in the data. Therefore, in this case, we use the Fused Gromov-Wasserstein (FGW) distance to unify them, enabling a joint comparison.The FGW distance of order p, with trade-off parameter $\alpha \in [0,1]$ and internal cost exponent $q \geq 1$, is defined as:

$$\begin{aligned} D_{FGW}(G_1, G_2) &= \left(\min_{\pi \in \Pi(h_1, h_2)} E_{p,q,\alpha}(\pi) \right)^{\frac{1}{p}}, \\ E_{p,q,\alpha}(\pi) &= (1-\alpha) \cdot E_W(\pi) + \alpha \cdot E_{GW}(\pi), \end{aligned} \tag{13}$$

where the parameter α plays a key role: it balances the trade-off between feature similarity and structural consistency. When $\alpha = 0$, FGW reduces to the Wasserstein distance on features; when $\alpha = 1$, it reduces to the Gromov-Wasserstein distance on structures. Then the optimal transport plan π^* provides a soft, semantic matching between the nodes of the two graphs, aligning them based on both their attributes and their relative positions within the graph topology.

In our compression task, we aim to ensure that the set of condensed graphs $\hat{\mathcal{G}}$ is structurally and feature-wise representative of a high-quality subset $\mathcal{G}_p$ selected from the original data via a similarity mechanism. We define the FGW Structural Matching Loss as follows:

$$\mathcal{L}_{FGW} = \frac{1}{|\mathcal{G}_p|} \sum_{g_s \in \mathcal{G}_p} \min_{\hat{g_c} \in \{\hat{\mathcal{G}}\}} D_{FGW,\alpha,p,q}(g_s, \hat{g_c}). \tag{14}$$

For each high-quality original graph g_s in the attended subset $\mathcal{G}_p$, we find its closest counterpart in the condensed set $\hat{\mathcal{G}}$ based on the FGW distance. We then compute the average of these minimum distances over the entire subset $\mathcal{G}_p$. This equation forces every condensed graph to represent a region of the original graph space, and ensures that every graph in $\mathcal{G}_p$ has a close representative in the condensed set. The use of FGW ensures that this representation is faithful with respect to both features and structure.

3.4 Gradient Matching for Graph Repository

Finally, we need to make the networks trained on the two repositories closer. Let $\mathcal{G}_p = \{g_1, g_2, ..., g_K\}$ be the high-quality subset selected by our similarity mechanism from the original graph collection, and let $\hat{\mathcal{G}} = \{\hat{g}_1, \hat{g}_2, ..., \hat{g}_M\}$ be our compressed graph set with $M \ll K$. For a given parameter initialization θ_0, we define the gradient matching progress as:

$$\begin{aligned} \hat{\mathcal{G}} &= \min_{\hat{\mathcal{G}}} \sum_{t=0}^{T-1} Dis(\theta^t_{\hat{\mathcal{G}}}, \theta^t_{\mathcal{G}_p}) \text{ s.t.}, \\ \theta^{t+1}_{\hat{\mathcal{G}}} &= \Delta_\theta(L(GNN_{\theta^t_{\hat{\mathcal{G}}}}(\hat{\mathcal{G}}), Y_{\hat{\mathcal{G}}})) \text{ and} \\ \theta^{t+1}_{\mathcal{G}_p} &= \Delta_\theta(L(GNN_{\theta^t_O}(\mathcal{G}_p), Y_{\mathcal{G}_p})), \end{aligned} \tag{15}$$

where Δ refers to the update step in gradient descent. The network trained by the synthetic graph repository can also have similar effects on the large repository. However, the overhead of directly matching network parameters is very high. Our approach aims to match the network parameters between large-real and small-synthetic training data by matching their gradients at each training step:

$$\nabla_\theta L(GNN_{\theta^t}(D), Y) = (\theta^{t+1} - \theta^t)/\mu, \tag{16}$$

where ∇ is the gradient of the network's descent at the corresponding step, and μ is the learning rate. In this way, the training trajectory on the small synthetic graph repository D_S can mimic that on the large real graph repository D_O. The gradients matching process for GNN can be modeled as:

$$\min_{D_S} \sum_{t=0}^{T-1} Dis(\nabla^t_{D_S}, \nabla^t_{D_O}), \tag{17}$$

where $\nabla^t_{D_S}$ represents the gradient of the repository D_S at the t-th step of the network, and Dis is a function that calculates the distance between two gradients. This phase is designed to preserve critical inter-graph relational information. To achieve this, we use gradient matching to ensure that synthetic data produces model update directions equivalent to those of the original data in downstream task training.

4 Experiment

To evaluate the effectiveness of our approach, we implemented our framework on top of Pytorch (version 2.5.0). All experiments were carried out on a workstation with an Ubuntu operating system, an Intel i9-12900K CPU, 128GB of memory, and a NVIDIA GeForce RTX4090 GPU. In this section, we present comprehensive experiments designed to address the following four Research Questions (RQs).

RQ1 (Superiority) What benefits does SSGDC offer in comparison to state-of-the-art (SOTA) methods?

RQ2 (Effectiveness) Can SSGDC effectively condense a graph repository so that the condensed dataset exerts a similar impact on various GNNs as the original dataset?

RQ3 (Necessity) Do the separate modules in SSGDC each play distinct roles in improving the effectiveness of graph dataset condensation?

4.1 Experimental Settings

Datasets . We investigated five real-world graph classification datasets: NCI1 and DD from TUDataset [Morris *et al.*, 2020], as well as ogbg-molhiv, ogbg-molbbbp, and ogbg-molbace from the Open Graph Benchmarks [Hu *et al.*, 2020]. The statistics for these datasets are summarized in Table 1. For each dataset, the data are randomly split into training, validation, and test sets at 80/10/10%.

Dataset	Type	Number of			Avg. Number of	
		Graphs	Attr.	Classes	Nodes	Edges
NCI1	Chemical	4110	37	2	110	64
DD	Bioinformatics	1179	89	2	284	14322
ogbg-molhiv	Bioinformatics	41127	9	2	25.5	27.5
ogbg-molbbbp	Bioinformatics	2039	9	2	24	26
ogbg-molbace	Bioinformatics	1513	9	2	34.1	36.9

Table 1: Statistics of the tested real-world graph repositories.

Dataset	GPC	Performance								
		Random	Herding	K-Center	DosCond	KiDD	Mirage	SGDC	SSGDC	Complete
NCI1 (ACC (%))	1	50.90±2.10	51.90±1.60	51.90±1.60	49.20±1.10	60.40±0.50	50.80±2.20	61.26±1.91	**63.74±1.28**	80.0±1.82
	5	52.10±1.00	60.50±2.40	47.00±1.10	51.10±0.80	63.20±0.20	51.30±1.10	62.32±1.62	**65.55±0.83**	
	10	55.60±1.90	61.80±1.50	49.40±1.80	50.30±1.30	64.20±0.10	51.70±1.40	62.75±1.47	**68.72±1.47**	
	20	58.70±1.40	60.90±1.90	55.20±1.60	50.30±1.30	60.90±0.70	52.10±2.20	62.68±1.73	**68.96±2.21**	
	50	61.10±1.20	59.00±1.50	62.70±1.50	50.30±1.30	65.40±0.60	52.40±2.70	62.79±1.98	**69.09±1.16**	
DD (ACC (%))	1	49.70±11.30	58.80±6.10	58.80±6.10	46.30±8.50	71.30±1.50	**74.00±0.40**	69.65±1.56	73.50±1.62	76.9±2.25
	5	40.80±4.30	58.70±5.80	51.30±5.30	57.50±5.60	70.90±1.10	-	69.43±1.61	**72.88±1.56**	
	10	63.10±5.20	64.10±5.80	53.40±3.10	46.30±8.50	71.50±0.50	-	69.62±1.24	**73.72±0.94**	
	20	56.40±4.30	67.00±2.60	58.50±5.70	40.70±0.00	71.20±0.90	-	70.29±1.75	**74.33±0.57**	
	50	58.90±6.30	68.40±4.00	62.30±2.50	44.00±6.70	71.80±1.00	-	70.76±2.14	**74.28±0.79**	
ogbg-molhiv (ROC-AUC)	1	0.366±.087	0.462±.072	0.462±.072	0.674±.131	0.664±.016	0.710±.009	0.689±.012	**0.713±.024**	0.701±.028
	5	0.501±.051	0.496±.044	0.519±.096	0.369±.175	0.657±.005	0.703±.012	0.699±.014	**0.714±.012**	
	10	0.554±.031	0.458±.058	0.471±.054	0.457±.214	0.632±.000	0.513±.055	**0.716±.025**	0.712±.009	
	20	0.621±.022	0.582±.027	0.627±.050	0.281±.007	0.648±.025	0.633±.048	0.716±.019	**0.718±.014**	
	50	0.625±.062	0.600±.034	0.680±.049	0.455±.214	0.587±.038	0.588±.067	0.714±.014	**0.715±.022**	
ogbg_molbace (ROC-AUC)	1	0.468±.045	0.486±.035	0.486±.035	0.512±.092	0.706±.000	0.590±.004	0.708±.021	**0.711±.019**	0.763±.020
	5	0.312±.019	0.470±.042	0.553±.024	0.555±.079	0.562±.000	0.419±.010	0.706±.014	**0.713±.011**	
	10	0.442±.028	0.532±.031	0.594±.019	0.536±.072	0.594±.000	0.419±.010	0.708±.017	**0.711±.023**	
	20	0.510±.023	0.509±.052	0.512±.031	0.484±.080	0.640±.011	0.423±.011	0.712±.016	**0.714±.008**	
	50	0.486±.020	0.625±.026	0.595±.026	0.503±.084	**0.723±.011**	-	0.723±.008	0.715±.009	
ogbg_molbbbp (ROC-AUC)	1	0.510±.013	0.532±.015	0.532±.015	0.546±.026	0.616±.000	0.592±.004	0.638±.005	**0.642±.006**	0.635±.017
	5	0.522±.014	0.546±.020	0.581±.022	0.519±.041	0.607±.005	0.431±.013	0.642±.007	**0.659±.009**	
	10	0.508±.018	0.578±.017	0.619±.027	0.505±.028	0.663±.000	0.465±.036	0.653±.015	**0.665±.006**	
	20	0.567±.010	0.533±.009	0.546±.012	0.493±.031	0.677±.001	0.610±.022	0.655±.023	**0.682±.007**	
	50	0.595±.014	0.552±.018	0.594±.016	0.509±.015	**0.684±.009**	0.590±.031	0.653±.018	0.681±.009	

Table 2: Performance comparison on different repositories, where the best performance is **highlighted** in bold.

Baselines . To comprehensively evaluate the performance of our condensation method on the graph repository using the multiple-graphs approach, we compared it against a diverse set of baselines, including state-of-the-art methods in graph dataset condensation and sampling. The baselines are categorized as follows. For the condensation methods, we selected representative methods from each of the classic compression types. DosCond [Jin *et al.*, 2022a] is a graph dataset condensation method based on gradient matching. KiDD [Xu *et al.*, 2023] is based on kernel ridge regression. Mirage [Gupta *et al.*, 2024] extracts frequent computational tree patterns. We consider several straightforward baselines: full-data training, random subsampling, and k-center.

Hyperparameter Settings . We configured the hyperparameters of the proposed condensation method on the graph repository using both the multiple-graphs strategy and the baseline methods. The main hyperparameter choices for our experiments are outlined below. For the condensation method, the number of condensed graphs per class was set to 1, 5, 10, 20, and 50. The learning rates for the structure and feature components were fixed at 0.001 and 0.0001, respectively, and we employed the Adam optimizer with a weight decay of 5×10^{-4}. During evaluation, we trained the same network for 1,000 epochs on the condensed graph using a learning rate of 0.001.

4.2 Comparison with State-of-the-Arts (RQ1)

To highlight the effectiveness of our approach, we performed extensive comparative experiments on graph classification tasks, benchmarking our method against existing graph condensation approaches and sampling strategies. The detailed results are summarized in Table 2. This table includes five commonly used graph classification datasets, ensuring a broad and diverse evaluation. For each dataset, we report the condensation rate, representing the fraction of the original data preserved after compression. Given identical condensation rates, we present the test accuracies obtained by six different graph condensation methods. For reference, we provide the test accuracy on the original, uncompressed datasets, which serves as an upper bound for performance comparison.

As presented in Table 2, our proposed method consistently surpasses all competing approaches in most experimental settings. On the NCI1 dataset, it yields substantial improvements over the strongest baseline (i.e., KiDD), with performance gains ranging from 3.34% to 5.46% across various compression ratios. On the Molecular Property Prediction datasets, our approach achieves the highest ROC-AUC scores across most compression ratios, with especially pronounced gains at GPC=50, where it clearly outperforms K-Center and other baselines. In contrast to methods whose performance deteriorates at certain compression levels, our technique exhibits robust, stable behavior across the full range from 1 to 50 graphs per class. Moreover, although a performance gap relative to training on the full dataset persists, our method substantially reduces this gap compared to prior approaches, achieving 85-95% of full-dataset performance at different compression ratios. For the overhead, our FGW matching is strictly constrained to the interaction between the heavily pruned subset (size K) and the condensed graph set (size M_c), bounding the time complexity to $O(K \cdot M_c \cdot V_c^2 \cdot t_{FGW})$. Peak GPU memory usage is similarly capped by processing matching pairs in mini-batches. Because our similarity mech-

DS	Consumption	Methods			
		DosCond	KiDD	Ours (Grad-Only)	SSGDC (Full)
NCI1	Time (s/round)	2.28	7.89	1.06	**1.68**
	GPU Memory (MB)	175.62	763.93	**94.15**	138.50
ogbg-molhiv	Time (s/round)	2.53	18.72	1.01	**1.75**
	GPU Memory (MB)	389.48	748.82	**289.48**	342.10

Table 3: Comparison of computational costs. SSGDC (Full) includes the overhead of the FGW structural matching.

anism ensures that $K \ll N$ (original total graphs) and M_c are very small, the optimal transport problem remains highly tractable.

4.3 Test with Different GNNs (RQ2)

To evaluate the effectiveness of the proposed graph condensation method, we conducted experiments with diverse GNN architectures. These experiments aimed to verify whether the condensed graphs remain compatible with various GNN models and maintain high performance across different network architectures. We further investigated the transferability of task-specific information by assessing the performance of condensed graphs on previously unseen GNN architectures. The corresponding results are summarized in Table 4, which reports the test accuracy of synthetic repositories trained with a single GNN model and evaluated on different networks. Each section of the table corresponds to a specific GNN (e.g., GCN, GAT, or GraphSAGE) used for training, followed by the accuracy achieved by other GNN architectures on the same condensed data for the same task. The results demonstrate that the condensed repositories maintain consistently high accuracy across a range of test architectures, highlighting the broad effectiveness and transferability of our method with diverse GNN models.

From Table 4, we can find that the graphs condensed by DGCNN can be used to train the remaining networks well and achieve at least 65.05% in test accuracy. And graphs condensed by other networks can achieve 64.39% accuracy in training neural networks. Furthermore, the results reveal that regardless of the GNN used for condensation, the test accuracy of a given network differs by up to 1.16%. This highlights the robustness and effectiveness of condensed graphs, which facilitate training and testing across a diverse set of GNN models, making them a valuable tool for reducing repository size without sacrificing performance.

4.4 Ablation Study (RQ3)

To validate the importance of each core component in SSGDC, we performed extensive ablation experiments on multiple architectural variants. The results are summarized in Table 5, which systematically analyzes the contributions of two key modules in our method. For the similarity-based pruning module, we designed two different variants: one removed the similarity mechanism entirely and instead used random sampling to obtain a high-quality subset from the original graph collection. The other adopted a simplified similarity mechanism based solely on graph-level features, ignoring structural information. These experiments show that even a rudimentary similarity mechanism yields better performance than random pruning but still falls far short of our full similarity module, which integrates structural features. The evidence is clear: on both datasets, the Feature-level Similarity variant consistently surpasses the Random Sampling variant, yet its performance plateaus below that of our complete model, indicating that feature-only similarity is insufficient.

Train\Test	Accuracy (%)				
	DGCNN	GIN	GAT	GraphSAGE	GCN
DGCNN	65.55±0.83	65.29±0.92	65.05±0.92	65.30±0.68	65.34±0.53
GIN	65.40±0.69	65.38±0.60	64.89±0.39	64.46±0.86	64.58±0.73
GAT	65.37±0.50	64.78±0.66	64.90±0.28	64.41±0.52	64.48±0.42
GraphSAGE	64.09±0.70	64.66±0.79	64.39±0.75	65.51±0.79	64.53±0.82
GCN	65.26±0.67	65.02±0.21	64.72±0.43	64.89±0.70	65.36±0.33

Table 4: Cross-architecture performance for the condensed dataset.

DS	GPC	Performance						
		Random Sampling	K-center Sampling	Feature-level Similarity	w/o FGW	Wasserstein Distance	Gromov-Wasserstein	SSGDC (Ours))
NCI1	1	50.85±1.15	53.96±2.10	60..74±1.44	55.47±1.67	57.81±1.13	59.76±1.82	63.74±1.28
	5	51.33±1.81	56.20±1.43	60.17±1.76	57.42±1.04	58.55±2.45	62.82±1.96	65.55±1.83
	10	55.20±1.52	54.01±1.45	61.88±1.07	60.82±1.72	61.80±1.80	63.86±1.80	68.72±1.47
	20	59.36±0.73	59.36±0.73	64.07±1.05	60.58±1.39	62.92±1.19	65.42±1.56	68.96±2.21
	50	61.55±1.71	61.55±1.71	65.50±1.36	65.47±1.44	64.81±1.53	66.73±1.39	69.09±1.16
ogbg-molhiv	1	0.535±.014	0.550±.021	0.607±.021	0.546±.007	0.572±.022	0.640±.014	0.713±.024
	5	0.561±.009	0.575±.016	0.616±.016	0.568±.022	0.606±.027	0.661±.022	0.714±.012
	10	0.615±.022	0.571±.008	0.646±.007	0.602±.013	0.644±.017	0.663±.015	0.712±.009
	20	0.630±.014	0.606±.015	0.667±.018	0.617±.019	0.643±.018	0.660±.024	0.708±.014
	50	0.629±.011	0.635±.016	0.660±.021	0.628±.022	0.645±.021	0.664±.012	0.715±.022

Table 5: Performance comparison between SSGDC variants.

For the Fused Gromov–Wasserstein structural matching, we explored two alternative setups. In the first, we substituted the FGW distance with the standard Wasserstein distance, thereby matching only the feature distributions and disregarding structural preservation. In the second, we relied solely on the Gromov–Wasserstein term, emphasizing structural consistency while neglecting explicit feature alignment. These configurations enabled us to isolate and quantify the individual and joint contributions of feature and structure preservation during the condensation process. The results clearly demonstrate that the Wasserstein Distance variant (feature-only matching) generally outperforms the feature-similarity pruning baseline (w/o FGW), highlighting the value of distributional alignment, and the Gromov-Wasserstein variant (structure-only matching) consistently surpasses the Wasserstein variant, particularly on the NC11 dataset. This underscores that structural consistency is often more critical than feature alignment alone for graph data. However, neither isolated component reaches the performance of their fusion. Our full SSGDC method, using the full FGW distance, achieves the best results across all configurations. This definitive superiority validates the necessity and synergy of jointly preserving both feature and structural information through FGW.

5 Conclusion

In this paper, we propose a novel graph condensation framework that effectively addresses the critical limitation of structural information loss inherent in existing methods. Our approach integrates a similarity-based graph selector with a dual-objective loss function that combines gradient matching and Fused Gromov-Wasserstein (FGW) divergence. This integration enables the simultaneous preservation of task-relevant informational content and the essential relational topology of the original graph repository. Extensive experiments demonstrate that our method consistently outperforms state-of-the-art baselines, achieving superior accuracy across a range of compression ratios and GNN architectures.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 62272170). Mingsong Chen is the corresponding author (mschen@sei.ecnu.edu.cn).

References

[Bojchevski *et al.*, 2020] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2464–2473, 2020.

[Dai *et al.*, 2019] Hanjun Dai, Yujia Li, Chenglong Wang, Rishabh Singh, Po-Sen Huang, and Pushmeet Kohli. Learning transferable graph exploration. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS), 8-14*, pages 2514–2525, 2019.

[Fan *et al.*, 2019] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference (WWW)*, pages 417–426, 2019.

[Fang *et al.*, 2024] Junfeng Fang, Xinglin Li, Yongduo Sui, Yuan Gao, Guibin Zhang, Kun Wang, Xiang Wang, and Xiangnan He. EXGC: bridging efficiency and explainability in graph condensation. In *Proceedings of the ACM on Web Conference (WWW)*, pages 721–732, 2024.

[Fout *et al.*, 2017] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Processing of Annual Conference on Neural Information Processing (NeurlIPS)*, pages 6530–6539, 2017.

[Gao *et al.*, 2024] Xinyi Gao, Tong Chen, Yilong Zang, Wentao Zhang, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. Graph condensation for inductive node representation learning. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 3056–3069, 2024.

[Gao *et al.*, 2025] Xinyi Gao, Junliang Yu, Tong Chen, Guanhua Ye, Wentao Zhang, and Hongzhi Yin. Graph condensation: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 37(4):1819–1837, 2025.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 70, pages 1263–1272, 2017.

[Gupta *et al.*, 2024] Mridul Gupta, Sahil Manchanda, Hariprasad Kodamana, and Sayan Ranu. Mirage: Model-agnostic graph distillation for graph classification. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[Hamilton *et al.*, 2017] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1024–1034, 2017.

[Hashemi *et al.*, 2024] Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 8058–8066, 2024.

[Hu *et al.*, 2020] 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 *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[Huang *et al.*, 2023] Tzu-Chen Huang, Te-Cheng Hsu, Yi-Hsien Hsieh, and Che Lin. Utilizing graph neural networks for breast cancer prognosis prediction with high-dimensional genomic data. In *Proceedings of the International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1–4, 2023.

[Jin *et al.*, 2022a] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 720–730, 2022.

[Jin *et al.*, 2022b] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[Karger, 1994] David R. Karger. Random sampling in cut, flow, and network design problems. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 648–657, 1994.

[Khoshraftar and An, 2024] Shima Khoshraftar and Aijun An. A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology (TSIT)*, 15(1):19:1–19:55, 2024.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[Lei and Tao, 2024] Shiye Lei and Dacheng Tao. A comprehensive survey of dataset distillation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(1):17–32, 2024.

[Liu *et al.*, 2021] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 6837–6849, 2021.

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

[Liu *et al.*, 2023] Yilun Liu, Ruihong Qiu, and Zi Huang. Cat: Balanced continual graph learning with graph condensation. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 1157–1162, 2023.

[Loukas and Vandergheynst, 2018] Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large

graphs with smaller graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3243–3252, 2018.

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

[Ma *et al.*, 2025] Yihong Ma, Yijun Tian, Nuno Moniz, and Nitesh V. Chawla. Class-imbalanced learning on graphs: A survey. *ACM Computing Surveys*, 57(8):207:1–207:16, 2025.

[Morris *et al.*, 2020] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

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

[Ribeiro and Towsley, 2010] Bruno F. Ribeiro and Donald F. Towsley. Estimating and sampling graphs with multidimensional random walks. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 390–403, 2010.

[Sener and Savarese, 2018a] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[Sener and Savarese, 2018b] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[Sun *et al.*, 2024] Qingyun Sun, Ziying Chen, Beining Yang, Cheng Ji, Xingcheng Fu, Sheng Zhou, Hao Peng, Jianxin Li, and Philip S. Yu. Gc-bench: An open and unified benchmark for graph condensation. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[Wang *et al.*, 2024] Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. Fast graph condensation with structure-based neural tangent kernel. In *Proceedings of the ACM on Web Conference (WWW)*, pages 4439–4448, 2024.

[Welling, 2009] Max Welling. Herding dynamical weights to learn. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1121–1128, 2009.

[Wu *et al.*, 2021] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

[Xiao *et al.*, 2025] Zhenbang Xiao, Yu Wang, Shunyu Liu, Bingde Hu, Huiqiong Wang, Mingli Song, and Tongya Zheng. Disentangled condensation for large-scale graphs. In *Proceedings of the ACM on Web Conference (WWW)*, pages 4494–4506, 2025.

[Xu *et al.*, 2023] Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. Kernel ridge regression-based graph dataset distillation. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2850–2861, 2023.

[Yang *et al.*, 2024] Yuhao Yang, Lianghao Xia, Da Luo, Kangyi Lin, and Chao Huang. Graphpro: Graph pre-training and prompt learning for recommendation. In *Proceedings of the ACM on Web Conference (WWW)*, pages 3690–3699, 2024.

[Yin *et al.*, 2025] Hongzhi Yin, Xinyi Gao, Junliang Yu, Ruihong Qiu, Tong Chen, Quoc Viet Hung Nguyen, and Zi Huang. Graph condensation: Foundations, methods and prospects. In *Companion Proceedings of the ACM on Web Conference 2025 (WWW)*, pages 69–72, 2025.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 974–983, 2018.

[Zhang *et al.*, 2024] Yuchen Zhang, Tianle Zhang, Kai Wang, Ziyao Guo, Yuxuan Liang, Xavier Bresson, Wei Jin, and Yang You. Navigating complexity: Toward lossless graph condensation via expanding window matching. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[Zhao and Bilen, 2023] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6503–6512, 2023.

[Zheng *et al.*, 2023] 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 *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.