Efficient Graph Condensation via Gaussian Process

Lin Wang
The Hong Kong Polytechnic University
Hong Kong SAR.
comp-lin.wang@connect.polyu.hk

Qing Li*
The Hong Kong Polytechnic University
Hong Kong SAR.
csqli@comp.polyu.edu.hk

***Abstract*—Graph condensation reduces the size of large graphs while preserving performance, addressing the scalability challenges of Graph Neural Networks caused by computational inefficiencies on large datasets. Existing methods often rely on bi-level optimization, requiring extensive GNN training and limiting their scalability. To address these issues, this paper proposes Graph Condensation via Gaussian Process (GCGP), a novel and computationally efficient approach to graph condensation. GCGP utilizes a Gaussian Process (GP), with the condensed graph serving as observations, to estimate the posterior distribution of predictions. This approach eliminates the need for the iterative and resource-intensive training typically required by GNNs. To enhance the capability of the GCGP in capturing dependencies between function values, we derive a specialized covariance function that incorporates structural information. This covariance function broadens the receptive field of input nodes by local neighborhood aggregation, thereby facilitating the representation of intricate dependencies within the nodes. To address the challenge of optimizing binary structural information in condensed graphs, Concrete random variables are utilized to approximate the binary adjacency matrix in a continuous counterpart. This relaxation process allows the adjacency matrix to be represented in a differentiable form, enabling the application of gradient-based optimization techniques to discrete graph structures. Experimental results show that the proposed GCGP method efficiently condenses large-scale graph data while preserving predictive performance, addressing the scalability and efficiency challenges.**

***Index Terms*—graph condensation, Gaussian process, efficiency**

I. Introduction

Graph-structured data, consisting of nodes and edges, is a versatile representation used to model various real-world systems, including social networks [1], [2], transportation infrastructures [3], [4], and molecular structures [5], [6]. Its ability to capture relationships and interactions makes it a powerful framework for describing complex systems. To effectively extract meaningful insights from graph-structured data, graph neural networks (GNNs) [7], [8] have been developed as a specialized class of deep neural networks. GNNs are designed to process graph data by leveraging a message-passing mechanism [9]. Through iterative aggregation of information from neighboring nodes, GNNs expand the receptive field of nodes, enabling them to represent both local and global graph structures [10]. This capability has made GNNs highly effective in numerous graph mining tasks [11], [12]. Despite their success, the application of GNNs faces significant challenges due to the large scale of graph data [13], [14]. Training and deploying GNNs often require substantial computational resources, including extensive memory, and prolonged GPU usage. Furthermore, the requirements of training iterations, hyperparameter tuning, and neural architecture search significantly limit their practical deployment.

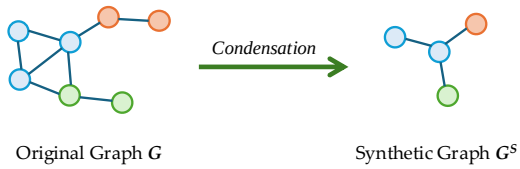


Fig. 1. Graph condensation reduces a large graph dataset G into a smaller one G^S with fewer nodes, while preserving essential information.

Graph condensation [15], [16] techniques have been introduced to address the computational challenges associated with GNN training. These methods condense the original dataset into a smaller synthetic dataset while retaining critical information. The resulting condensed dataset enables more efficient GNN training by reducing both computational cost and training time. By mitigating the resource-intensive nature of GNN training, graph condensation techniques facilitate the scalability and broader applicability of GNNs. Current graph condensation methods are commonly framed as bi-level optimization problems [15], [17], where the inner loop training the GNN using condensed data, and the outer loop updates the condensed data.

While these approaches have achieved notable progress, they face significant challenges, particularly in computational inefficiency [18] caused by the extensive iterative training of GNN. The nested loops of bi-level optimization requires iterative updates for both GNN parameters and condensed data, resulting in slow convergence. Moreover, to improve the generalization of the condensed data across GNN models with diverse initializations, the GNN of inner loop often involves thousands of parameter reinitializations. This excessive computational demand significantly undermines the efficiency of the graph condensation, to the extent that training large models directly on raw data may become more practical than using condensed data.

Gaussian Process (GP) [19], [20], a non-parametric method,

* Corresponding author.

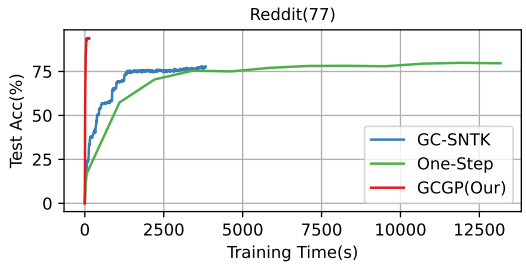


Fig. 2. Condensation time and accuracy comparison on Reddit dataset. By condensing 153,431 training nodes into a graph with only 77 synthetic nodes, the GCGP achieves a accuracy of 93.9% on test set, which is comparable to the accuracy obtained using the full training set on GCN, while also demonstrating the fastest runtime. More experimental results about the condensation efficiency please refer to Section V-C.

eliminates the need for complex iterative procedures during model training. This characteristic positions GPs as a computationally efficient alternative to GNNs in graph condensation tasks. A GP is defined by its mean function and covariance function, which together enable the posterior outputs for input data by leveraging prior knowledge and observations. Incorporating GPs into the graph condensation process eliminates the iterative training typically required by GNN models. Consequently, this approach enhances computational efficiency while simultaneously avoiding the dependency of the condensed data on parameter initialization, thereby improving the robustness and overall effectiveness of the condensation process.

Employing GP for prediction poses significant challenges. In graph-structured data, node features describe node attributes, while structural information encodes dependencies and interaction patterns through topological relationships. Combining these aspects challenges model design, as covariance functions must capture both feature similarity and graph context. Structural information is vital for modeling graph data, as it reflects both local and global topological characteristics. Neglecting it can hinder a model's ability to represent graph semantics, making the integration of node features and structural information a key challenge. A second challenge stems from the discrete nature of graph structures, such as adjacency matrix, which is non-differentiable and incompatible with gradient-based optimization methods. This limitation hinders efficient optimization of condensed graph data. To address these challenges, we propose a covariance function that integrates structural information from graph topology. This function supports neighborhood message passing, expanding node receptive fields and enhancing dependency modeling while minimizing computational costs [10], [20]. This approach systematically improves GP predictive performance. For optimizing structural information, we relax the binary adjacency matrix [16] into a differentiable form using concrete random variables [21]. As the temperature approaches zero, the relaxed adjacency matrix converges to a binary form. This relaxation enables gradient-based optimization to refine graph structure effectively.

As shown in Figure 2, our method condenses the training set of the Reddit [22] dataset to 77 nodes, representing only 0.05% of the original data. Among the evaluated algorithms, our method achieves superior condensation quality, as measured by node classification accuracy, while also demonstrating high computational efficiency. These results highlight the method's effectiveness in addressing both quality and efficiency challenges in graph data condensation.

In this paper, we propose a novel GP-based graph condensation framework for node classification tasks, termed **G**raph **C**ondensation via **G**aussian **P**rocesses (**GCGP**). The framework leverages a GP model as the predictive component, where the condensed data serves as observations. To enhance the predictive capabilities of our model, we propose a specialized covariance function that effectively captures dependencies between test inputs and observations. Furthermore, we employ concrete random variables to relax the binary adjacency matrix into a differentiable form, thereby facilitating gradient-based optimization. The discrepancy between the GP predictions and the ground truth is then leveraged to guide the optimization process, particularly in refining the structure of the synthetic graph.

The key contributions of this work are as follows:

- We propose a graph condensation method using GP with condensed graph as observations to improve graph condensation efficiency.
- We design a covariance function for graph-structured data, leveraging local neighborhood aggregation to capture structural information.
- We introduce the binary concrete relaxation to optimize the adjacency matrix, enabling gradient-based optimization of graph structures.
- We validate the proposed method through experiments on diverse datasets and baselines, demonstrating its efficiency and adaptability across GNN architectures.

II. Related Work

Gaussian Process. The Gaussian Process, a widely recognized non-parametric model, has been extensively studied and serves as a foundational tool in machine learning and statistical modeling. Williams [23] first derived the analytic forms of the covariance matrix for single-hidden-layer infinite-width neural networks, laying the groundwork for subsequent theoretical developments. Building on this, Lee et al. [20] established a connection between infinite-width deep neural networks and Gaussian processes, providing a formal probabilistic framework. Expanding this line of research, Jacot et al. [24] introduced the Neural Tangent Kernel, which offers insights into the training dynamics of infinitely wide networks. Further, Matthews et al. [25] demonstrated the emergence of Gaussian process behavior in such networks, reinforcing the theoretical underpinnings of their behavior.

Graph Condensation. Dataset condensation [26], [27] aims to condense large-scale datasets into smaller synthetic datasets while ensuring comparable model performance. The bi-level optimization framework DC [26] builds on the concept of optimizing image pixels as parameters via gradients [28]. Key

evaluation criteria for DC include gradient matching [27], [29], [30], feature alignment [31], training trajectory matching [32], and distribution matching [33]. Additionally, [34] introduced a meta-learning approach for DC, drawing on infinite-width neural network theory [24], [35]–[37].

Extending the concept of dataset condensation to graph data, graph condensation encompasses two primary tasks: node-level condensation and graph-level condensation [15], [16]. Node-level condensation synthesizes a smaller graph with fewer nodes, while graph-level condensation reduces a set of graphs into a smaller, synthetic set. Jin et al. [15] were the first to adapt dataset condensation to the graph domain by proposing a bi-level graph condensation method. Recent advancements in graph condensation have introduced diverse methodologies, targeting both node-level and graph-level tasks. Zheng et al. [38] proposed a structure-free approach that condenses a graph into node embeddings by employing training trajectory matching. Similarly, Liu et al. [39] advanced node-level graph condensation by utilizing receptive field distribution matching. Neural tangent kernels have also been explored in this domain; for instance, GC-SNTK [18] leverages the Kernel Ridge Regression framework with a structure-based neural tangent kernel to enhance condensation efficiency. Furthermore, Zhang et al. [17] introduced a lossless node-level graph condensation method grounded in trajectory matching. For graph-level condensation, probabilistic modeling of synthetic graph structures has been investigated. DosCond [16] employs a probabilistic approach to model the discrete structure of graphs. In addition, Kernel Ridge Regression-based techniques have been applied to this task. The KiDD method [40] utilizes the KRR framework to condense entire graph-level datasets effectively.

III. Preliminary

Generally, the condensed synthetic graph data $G^{\mathcal{S}}$ is expected to perform as well as the target graph G when applied to the downstream tasks (e.g., training a GNN model f_θ with parameter θ). The current approach to the graph condensation problem involves a bi-level optimization process, where two nested training loop is utilized to optimize the model parameter θ and the condensed graph $G^{\mathcal{S}}$, respectively. The the inner loop handles the training of GNN f_{θ_t} on the condensed data $G^{\mathcal{S}}$ by the training loss $\ell(f_\theta, G^{\mathcal{S}})$, while the outer loop optimize the condensed graph $G^{\mathcal{S}}$ by minimizing the matching loss $\mathcal{L}(f_{\theta_t}, G)$. Therefore, the bi-level graph condensation problem could be modeled as

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathcal{L}(f_\theta, G) \\ s.t. \quad &\theta = \arg\min_\theta \ell(f_\theta, G^{\mathcal{S}}). \end{aligned} \tag{1}$$

In general, the convergence of parameter θ is influenced by its initial values θ_0. This implies that the condensed data will only yield favorable results when the neural network model is initialized with θ_0. However, the objective of condensation is to produce data that can perform well under random initialization distribution P_{θ_0}. To address this constraint, multiple random initializations are required, and problem (1) is accordingly modified as follows,

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\mathcal{L}(f_\theta, G)\right] \\ s.t. \quad &\theta = \arg\min_\theta \ell(f_\theta, G^{\mathcal{S}}), \end{aligned} \tag{2}$$

where the loss function of the outer loop $\mathcal{L}$ is designed as matching the gradients [27] that are calculated by $G^{\mathcal{S}}$ and G on model f_{θ_t}, respectively. This can be expressed as follows:

$$\mathcal{L} = D(\nabla_{\theta_t} \ell(f_{\theta_t}, G^{\mathcal{S}}), \nabla_{\theta_t} \ell(f_{\theta_t}, G)), \tag{3}$$

where $D(\cdot, \cdot)$ is the function measuring distance between gradients $\nabla_{\theta_t} \ell(f_{\theta_t}, G^{\mathcal{S}})$ and $\nabla_{\theta_t} \ell(f_{\theta_t}, G)$. θ_t is the parameter of the inner loop model updated t times by the specific optimization algorithm $\texttt{opt-alg}_\theta$ (e.g., Stochastic Gradient Descent, SGD). Denote T as the total iteration times of outer loop. Finally, the bi-level graph condensation problem becomes:

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_{t=0}^{T} D(\nabla_{\theta_t} \ell(f_{\theta_t}, G^{\mathcal{S}}), \nabla_{\theta_t} \ell(f_{\theta_t}, G))\right] \\ &s.t. \quad \theta_t = \texttt{opt-alg}\left[\ell(f_{\theta_{t-1}}, G^{\mathcal{S}})\right]. \end{aligned} \tag{4}$$

Problem (4) offers a potential solution for graph data condensation. However, it presents a highly complex optimization challenge. Since a GNN is a parameterized model, the inner loop of this condensation framework requires iterative optimization of the parameters f_θ. Simultaneously, the outer loop optimizes the condensed data $G^{\mathcal{S}}$. To ensure the generalization capability of the condensed data under initialization distribution P_{θ_0} of the GNN parameters, an additional outermost loop repeatedly initializes the GNN parameters θ_0, typically exceeding 1,000 iterations. Consequently, this three-level nested optimization algorithm is computationally expensive and time-intensive.

IV. Methodology

This section detailed the proposed GCGP framework with a covariance function specifically designed for graph-structured data. We further adopt the concrete random variables to relax the binary adjacency matrix with a differentiable counterpart. Finally, we outline the optimization process and provide a complexity analysis to demonstrate the method's efficiency advantages.

A target graph dataset $G = \{X, A\}$ consists of n nodes, where $X \in \mathbb{R}^{n \times d}$ represents the node features of dimensionality d, and $A \in \{0, 1\}^{n \times n}$ is the adjacency matrix. Here, $A_{ij} = 1$ indicates a connection between nodes i and j, while $A_{ij} = 0$ indicates no connection. Additionally, let Y represent the associated node labels. Graph condensation is a technique designed to reduce the size of a graph dataset while retaining its critical properties for downstream tasks. Specifically, this method condenses G into a smaller synthetic graph $G^{\mathcal{S}} = \{X^{\mathcal{S}}, A^{\mathcal{S}}\}$, accompanied by a corresponding labels $Y^{\mathcal{S}}$. In the condensed graph, $X^{\mathcal{S}} \in \mathbb{R}^{m \times d}$, where the number of nodes m satisfies $m \ll n$.

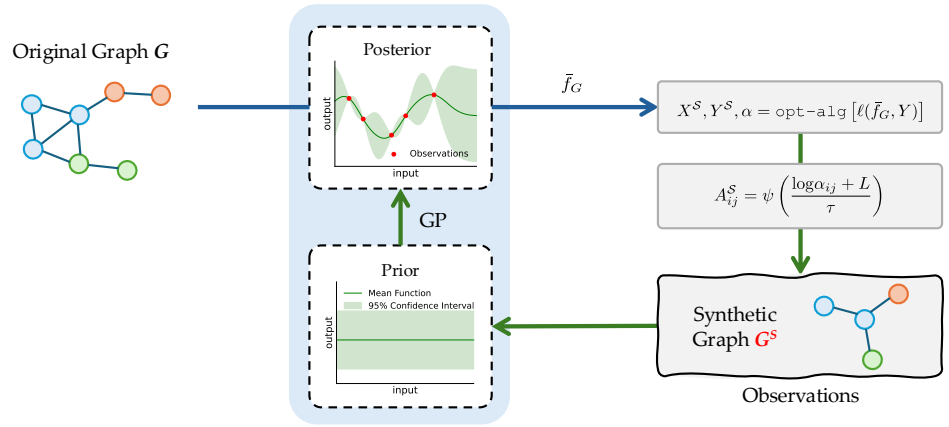


Fig. 3. The workflow of the proposed GCGP framework involves three key steps. First, the condensed synthetic graph $G^{\mathcal{S}}$ is utilized as the observations for the GP. Next, predictions are generated for the test locations, corresponding to the original graph G. Finally, the condensed graph is iteratively optimized by minimizing the discrepancy between the GP's predictions and the ground-truth labels.

A. *Efficiency Graph Condensation via Gaussian Process*

Gaussian processes (GPs) provide a probabilistic framework for modeling distributions over functions. Formally, a GP is defined as a collection of random variables, any finite subset of which follows a joint Gaussian distribution [19]. To model functions within this framework, Denoting a mapping $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, which transforms an input x into a d'-dimensional feature space. Using this mapping, a Bayesian linear regression model is defined as $f(x) = \phi(x)W$, where the weight W is assigned a Gaussian prior $W \sim \mathcal{N}(0, \Sigma_p)$. In practical scenarios, the true function values are often not directly observable. Instead, we observe noisy outputs $y = f(x) + \varepsilon$, where the noise ε is assumed to follow an independent and identically distributed Gaussian distribution, $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. This model serves as the basis for making predictions. Within this context, A GP is fully characterized by its mean function and covariance function. Usually, for notational simplicity we will take the mean function to be zero $\mu(x) = 0$. The covariance function is given by,

$$\begin{aligned} Cov(Y^{\mathcal{S}}) &= \mathbb{E}[(f(G^{\mathcal{S}}) + \varepsilon)(f(G^{\mathcal{S}}) + \varepsilon)] \\ &= \phi(G^{\mathcal{S}})^\top \Sigma_p \phi(G^{\mathcal{S}}) + \sigma_\varepsilon^2 I \\ &= \mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I, \end{aligned} \tag{5}$$

where $G^{\mathcal{S}}$ and $Y^{\mathcal{S}}$ are the observed data. I is the identity matrix.

To model the underlying function while accounting for noise in the observations, we incorporate the observed data into a probabilistic framework. In the context of graph condensation tasks, the condensed data $G^{\mathcal{S}} = \{X^{\mathcal{S}}, A^{\mathcal{S}}\}$ is treated as a potential input, while the corresponding labels $Y^{\mathcal{S}}$ are considered as the observed targets. We define a prior joint distribution over the observed targets $Y^{\mathcal{S}}$ and the function values f_G at the test locations of the original graph data G,

$$\begin{bmatrix} Y^{\mathcal{S}} \\ f_G \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I & \mathcal{K}(G^{\mathcal{S}}, G) \\ \mathcal{K}(G, G^{\mathcal{S}}) & \mathcal{K}(G, G) \end{bmatrix}\right). \tag{6}$$

The joint prior in Equation (6) serves as the foundation for modeling the Gaussian process. By conditioning this joint prior on observed data, as described by the Gaussian identity [19], the Gaussian process derives the posterior distribution. This process yields the predictive mean $\bar{f}_G$, representing the best estimate of the function at unobserved points, and the predictive variance $Cov(f_G)$, quantifying the uncertainty of these predictions,

$$f_G | G, G^{\mathcal{S}}, Y^{\mathcal{S}} \sim \mathcal{N}(\bar{f}_G, Cov(f_G)), \tag{7}$$

where,

$$\bar{f}_G = \mathcal{K}(G, G^{\mathcal{S}})[\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I]^{-1} Y^{\mathcal{S}}, \tag{8}$$

$$\begin{aligned} Cov(f_G) =& \mathcal{K}(G, G) - \\ & \mathcal{K}(G, G^{\mathcal{S}})[\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I]^{-1} \mathcal{K}(G^{\mathcal{S}}, G). \end{aligned} \tag{9}$$

Traditional bi-level graph condensation methods typically utilize parametric predictive models, such as GNNs, to evaluate the performance of condensed data on tasks like node classification. These methods require iterative training to optimize model parameters, which constitutes the inner-loop optimization in bi-level graph condensation. In contrast, the proposed framework leverages GP, a non-parametric probabilistic model, to make predictions without the need for iterative training. By specifying a joint Gaussian distribution, GP enables conditional inference to estimate function values at unseen data points G while simultaneously quantifying predictive uncertainty [41], [42]. This design avoids the computational overhead of iterative optimization and parameter initialization inherent in GNN-based methods. As a result, the computational efficiency of GP makes them a strong candidate for graph condensation, enabling the bi-level condensation process to be reformulated as a simpler optimization problem:

$$\min_{G^{\mathcal{S}}} \ell(\bar{f}_G, Y). \tag{10}$$

Compared to the bi-level condensation model in Equation (4), this approach simplifies the optimization problem

by reformulating it into a single-level optimization for the graph condensation objective. Replacing the GNN with a GP eliminates the iterative training step in the inner loop of traditional bi-level formulations, significantly reducing time complexity and computational costs. This reduction is particularly beneficial for large-scale graph datasets, where the cost of iterative training can become prohibitive. Consequently, the reformulated approach improves scalability and computational efficiency, making it more suitable for real-world applications involving large graphs.

B. Covariance Function for Graph Structural Data

The covariance function is a fundamental component of GPs because it defines the correlation structure that determines key properties of the posterior distribution, such as smoothness and uncertainty [43]. Its design directly influences the predictive model's fitting capability, generalization performance, and computational efficiency, as these aspects are encapsulated in the properties of the resulting covariance matrix [44].

In the context of the graph condensation task, the design of the covariance function is crucial, as it must meet task-specific requirements to effectively extract relevant information while ensuring computational efficiency. As shown in Equation (5), the covariance function $\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) = \phi(G^{\mathcal{S}})^{\top}\Sigma_p\phi(G^{\mathcal{S}})$ depends on the mapping model $\phi(\cdot)$ and the matrix Σ_p. Since the initialization of the weights is sampled from the same distribution, Σ_p remains fixed. Therefore, the selection and design of the mapping model $\phi(\cdot)$ play a pivotal role in determining the behavior of the covariance function.

To address the challenges of designing effective mappings for graph data, we propose a computationally efficient mapping model that captures structural information through the message passing. The structural information of a graph, which encodes relationships among nodes, is essential for generating meaningful node representations. Our model aggregates k-hop neighborhood information in a single step, allowing node features to encompass the k-hop receptive field [10]. This approach enhances the representation of local graph structures and mitigates the computational overhead associated with multi-step message passing. The message-passing mechanism is detailed below:

$$\hat{X} = \hat{A}^k X, \tag{11}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. $\tilde{A} = A + I$ is the self-looped adjacency matrix, and $\tilde{D} = \mathrm{diag}(\sum_j \tilde{A}_{1j}, \sum_j \tilde{A}_{2j}, \ldots, \sum_j \tilde{A}_{nj})$. In the development of models prioritizing simplicity and low computational complexity, node features are updated through a k-hop message passing mechanism. This process is followed by a single non-linear operation to improve the expressive capacity of the mapping while minimizing additional computational overhead. Formally, this mapping procedure can be represented as:

$$\phi(X) = \psi(\hat{A}^k X W) + \varepsilon, \tag{12}$$

where $\psi(x) = \frac{2}{\pi}\int_0^x e^{-t^2}dt$ represents the non-linear activation function. W is the weight matrix. Building on the mapping, the prediction could be written as:

$$Y = \psi(\hat{A}^k X W^{(0)} + \varepsilon^{(0)})W^{(1)} + \varepsilon^{(1)}, \tag{13}$$

where weight matrices $W^{(0)} \in \mathbb{R}^{d\times d_1}$ and $W^{(1)} \in \mathbb{R}^{d_1\times C}$ are initialized such that $W^{(0)}, W^{(1)} \sim \mathcal{N}(0, \sigma_w^2)$, C represents the number of possible categories of the input data. The bias terms follow $\varepsilon^{(0)}, \varepsilon^{(1)} \sim \mathcal{N}(0, \sigma_\varepsilon^2)$.

As the width of the weight matrix approaches infinity ($d_1 \to \infty$), the resulting infinite-width limit facilitates an analytical formulation of the covariance function. Assuming that the elements $\hat{x}_i$ in $\hat{X}$ are independent and identically distributed (i.i.d.), the covariance function of the mapping can be expressed in terms of the expectations of the post-activation outputs [20], [43]. Let $\beta = \sigma_\varepsilon^2$. For two graphs $G = \{X, A\}$ and $G' = \{X', A'\}$, the covariance function $\mathcal{K}$ is defined as:

$$\mathcal{K}(G, G') = \mathbb{E}_{f\sim\mathcal{GP}(0,\Lambda^{(0)})}[\psi(f(G))\psi(f(G'))] + \beta, \tag{14}$$

where,

$$\Lambda(G, G') = \begin{bmatrix}\Sigma(G, G) & \Sigma(G, G')\\ \Sigma(G', G) & \Sigma(G', G')\end{bmatrix}, \tag{15}$$

$$\Sigma(G, G') = \frac{1}{d}\hat{A}^k X (\hat{A'}^k X')^{\top} + \beta. \tag{16}$$

The elements of the covariance function $\mathcal{K}(G, G')$ can be calculated analytically as [23],

$$\mathcal{K}_{ij} = \frac{2}{\pi} sin^{-1}\frac{2\tilde{x}_i^{\top}\Sigma_w\tilde{x}_j}{\sqrt{(1 + 2\tilde{x}_i^{\top}\Sigma_w\tilde{x}_i)(1 + 2\tilde{x}_j^{\top}\Sigma_w\tilde{x}_j)}}, \tag{17}$$

where $\tilde{x}_i = (\hat{x}_i, 1)^{\top}$ denotes the augmented vector, with $\hat{x}_i$ representing the i-th element of $\hat{X}$. The covariance matrix $\Sigma_w = \mathrm{diag}(\sigma_w^2, \sigma_w^2, \cdots, \beta)$ characterizes the noise structure in the model, where σ_w^2 is typically set to 1. The Equation (17) facilitates computational efficiency by enabling element-wise calculations, which are well-suited for acceleration using GPUs. As a result, this approach provides a computationally efficient and scalable solution for reducing the prediction time of GP.

C. Learning the Discrete Synthetic Graph Structure

In the synthetic condensed graph $G^{\mathcal{S}}$, the structural information is encoded in the binary adjacency matrix $A^{\mathcal{S}} \in \{0, 1\}^{m\times m}$. However, Optimizing the adjacency matrix is challenging because it is discrete and typically cannot be directly optimized using gradient-based methods. [16], [45]. To address this issue, this work adopts concrete random variables to relax the adjacency matrix, enabling smoother optimization processes and better computational efficiency.

Particularly, we propose applying the Concrete relaxation [21], [46] technique to approximate the discrete adjacency matrix with a continuous counterpart. For the synthetic adjacency matrix, the Concrete relaxation reformulates the binary variable $A_{ij}^{\mathcal{S}}$, which originally follows a Bernoulli distribution, into a differentiable function. This reformulation

introduces concrete random variables α_{ij} and τ, such that $A_{ij}^{\mathcal{S}} \sim \text{BinConcrete}(\alpha_{ij}, \tau)$. Let $L \sim \text{Logistic}$, the $A_{ij}^{\mathcal{S}}$ is reparameterized as,

$$A_{ij}^{\mathcal{S}} = \psi\left(\frac{\log\alpha_{ij} + L}{\tau}\right), \tag{18}$$

where $\psi(x) = \frac{1}{1+\exp(-x)}$, and L is defined as $L \overset{d}{=} \log(U) - \log(1-U)$, with $U \sim \text{Uniform}(0,1)$. The parameters α_{ij} and τ are constrained to the interval $(0, \infty)$, where τ denotes the temperature parameter.

In Equation (18), the binary adjacency matrix is reformulated into a differentiable representation parameterized by α, facilitating gradient-based optimization. During this optimization process, the temperature parameter τ is gradually reduced toward zero. This systematic reduction drives the adjacency matrix toward a binary state, effectively approximating its discrete structure. The probability that a specific element of the condensed adjacency matrix, $A_{ij}^{\mathcal{S}}$, equals 1 is expressed as:

$$\mathbb{P}\left(\lim_{\tau\to 0} A_{ij}^{\mathcal{S}} = 1\right) = \frac{\alpha_{ij}}{1+\alpha_{ij}}. \tag{19}$$

To ensure that $A_{ij}^{\mathcal{S}} = 1$, we set the threshold $\mathbb{P}\left(\lim_{\tau\to 0} A_{ij}^{\mathcal{S}} = 1\right) > 0.5$.

D. Objective and Optimization

This paper introduces GCGP, a novel method that employs GP to integrate prior information with the condensed graph $G^{\mathcal{S}}$ for predictions. The approach is computationally efficient, avoiding additional training loops. The resulting loss function is:

$$\ell(\bar{f}_G, Y) = ||\bar{f}_G - Y||_F^2. \tag{20}$$

The optimization process in GCGP involves three key parameters: $X^{\mathcal{S}}$, $Y^{\mathcal{S}}$, and $A^{\mathcal{S}}$. Since $A^{\mathcal{S}}$ is a differentiable function of α, the optimization primarily focuses on $X^{\mathcal{S}}$, $Y^{\mathcal{S}}$, and α, utilizing algorithms such as stochastic gradient descent. The overall procedure is outlined as follows.

First, the initialization of $X^{\mathcal{S}}$, $Y^{\mathcal{S}}$, α, and the adjacency matrix $A^{\mathcal{S}}$ is performed, where $A^{\mathcal{S}}$ is sampled using Equation (18). Subsequently, GP predictions $\bar{f}_G$ for the original graph G are computed as described in Equation (8). These predictions are then compared with the original labels Y to calculate the condensation loss. This loss function guides the iterative optimization process, during which $X^{\mathcal{S}}$, $Y^{\mathcal{S}}$, and α are updated.

To facilitate the optimization process, the temperature parameter τ is gradually reduced during training. As τ approaches zero, the sampled adjacency matrix $A^{\mathcal{S}}$ converges to a binary form, thereby enabling discretization. This progressive reduction in τ ensures a smooth transition from a probabilistic to a deterministic representation of the adjacency matrix. The complete training procedure is detailed in Algorithm 1.

Algorithm 1 GPGC

Input: Target graph data $G = \{X, A\}$ with n nodes and labels Y.
Output: Condensed graph data $G^{\mathcal{S}} = \{X^{\mathcal{S}}, A^{\mathcal{S}}\}$ with $m (m \ll n)$ nodes and labels $Y^{\mathcal{S}}$.
Initialize: Random initialize $X^{\mathcal{S}}$, $Y^{\mathcal{S}}$, and α.
while not converge **do**
  Update $A^{\mathcal{S}}$ by:

$$A_{ij}^{\mathcal{S}} = \psi\left(\frac{\log\alpha_{ij} + L}{\tau}\right)$$

  Calculate $\mathcal{K}(G, G^{\mathcal{S}})$ and $\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}})$ by

$$\frac{2}{\pi} sin^{-1} \frac{2\tilde{X}^\top \Sigma_w \tilde{X}'}{\sqrt{(1+2\tilde{X}^\top \Sigma_w \tilde{X})(1+2\tilde{X}'^\top \Sigma_w \tilde{X}')}}$$

  Calculate loss function:

$$\ell(\bar{f}_G, Y) = ||\mathcal{K}(G, G^{\mathcal{S}})[\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \beta I]^{-1} Y^{\mathcal{S}} - Y||_F^2$$

  Update $X^{\mathcal{S}}, Y^{\mathcal{S}}, \alpha$:

$$X^{\mathcal{S}}, Y^{\mathcal{S}}, \alpha = \texttt{opt-alg}\left[\ell(\bar{f}_G, Y)\right]$$

  Update $\tau \to 0$
end while
if $\frac{\alpha_{ij}}{1+\alpha_{ij}} > 0.5$ **then**
  $A_{ij}^{\mathcal{S}} = 1$
else
  $A_{ij}^{\mathcal{S}} = 0$
end if

E. Complexity Analysis

To demonstrate that the proposed GCGP method has lower computational complexity than the bi-level method GCond, we perform a detailed complexity analysis. The notations are defined as follows: n and m denote the number of nodes in the original and condensed datasets, respectively, d is the dimensionality of node features, and $|E|$ represents the number of edges in the original graph. Additionally, t_{in} and t_{out} are the number of iterations in the inner and outer loops, respectively, while t_{init} denotes the number of times the GNN model is re-initialized in GCond.

The computational complexity of GCond is derived by analyzing its inner and outer loops. The inner loop, involving GNN training, has a complexity of $O(t_{in}(md^2))$. For the outer loop, the primary cost is forward propagation on the original dataset, with a complexity of $O(nd^2 + |E|d)$. Considering t_{out} optimization iterations, the total outer loop complexity becomes $O(t_{out}(nd^2 + |E|d))$. Combining these, the overall complexity of GCond is:$O(t_{init}(t_{out}(nd^2+|E|d+t_{in}(md^2))))$.

The proposed GCGP method involves two main steps: covariance function computation and GP prediction. The kernel computation has a complexity of $O(mnd)$, while GP prediction has $O(m^3 + mnd)$. Thus, the total complexity of GCGP is: $O(tm^3 + tmnd)$. This analysis highlights that

TABLE I
DATASET DETAILS

Dataset	Nodes	Edges	Classes	Features	Train/Validation/Test
Cora	2,708	5,429	7	1,433	140/500/1,000
Citeseer	3,327	9,104	6	3,703	120/500/1,000
Pubmed	19,717	44,338	3	500	60/500/1,000
Photo	7,650	238,162	8	745	160/500/800
Computers	13,752	491,722	10	767	200/500/1,000
Ogbn-arxiv	169,343	1,166,243	40	128	90,941/29,799/48,603
Reddit	232,965	114,615,892	41	602	153,431/23,831/55,703

TABLE II
CONFIGURATION OF THE EXPERIMENTS

Dataset	Size	β	k	Learn A	Dataset	Size	β	k	Learn A
Cora	35	0.01	3	0	Computers	50	0.01	1	0
	35	0.5	2	1		50	0.01	1	1
	70	0.5	4	0		100	0.01	2	0
	70	1	2	1		100	0.01	1	1
	140	0.5	4	0		200	0.01	2	0
	140	0.01	2	1		200	0.001	1	1
Citeseer	30	10	5	0	Ogbn-arxiv	90	5	2	0
	30	1	2	1		90	0.5	2	1
	60	0.5	2	0		454	0.001	2	0
	60	5	1	1		454	0.5	2	1
	120	5	4	0		909	10	2	0
	120	5	1	1		909	0.5	2	1
Pubmed	15	0.001	2	0	Reddit	77	0.1	2	0
	15	0.5	2	1		77	0.01	2	1
	30	0.01	2	0		153	0.1	2	0
	30	0.5	2	1		153	0.1	1	1
	60	0.1	5	0		307	1	2	0
	60	0.5	2	1		307	0.1	2	1
Photo	40	0.1	2	0	Photo	80	0.01	1	1
	40	0.01	1	1		160	0.001	2	0
	80	0.1	2	0		160	0.001	1	1

the GCGP method achieves significantly lower computational complexity than GCond, particularly for large-scale datasets where n and $|E|$ dominate the cost.

V. EXPERIMENTS

In this section, we evaluate the condensation effectiveness of our method across various settings. Extensive comparative experiments demonstrate that our method consistently achieves the fastest performance on both small and large datasets. To assess generalization, we train different neural network models using the condensed data and analyze their performance. A sensitivity analysis of parameters is also conducted. Finally, ablation experiments confirm the method's advantages in efficiency and condensation quality. All experiments are executed on an NVIDIA RTX 3090 GPU with 24GB RAM.

A. Experimental Settings

1) Datasets: Seven different scale benchmark datasets in the graph domain are adopted, including,

- **Cora** [47] dataset consists of 2,708 scientific publications across seven classes, connected by 5,429 citation links. Each publication is represented by a binary word vector indicating the presence of 1,433 unique words.
- **Citeseer** [47] dataset consists of 3,312 publications grouped into six classes, linked by 4,732 citations. Publications are represented as binary word vectors based on 3,703 unique words.
- **Pubmed** [47] dataset contains 19,717 publications categorized into three classes and connected by 44,338 citations. Each publication is described by a TF/IDF-weighted word vector [48] derived from 500 unique words.
- **Photo** [49] dataset, a subset of the Amazon co-purchase graph, consists of 7,650 products (nodes) and 238,162 co-purchase relationships (edges). Node features are bag-of-words [50] encoded product reviews, and class labels correspond to product categories.
- **Computers** [49] dataset, another subset of the Amazon co-purchase graph, consists of 13,752 products (nodes) and 491,722 co-purchase relationships (edges).
- **Ogbn-arxiv** [51] dataset represents a citation graph of Computer Science arXiv papers. Nodes correspond to papers, with edges indicating citations. Node features are 128-dimensional vectors obtained by averaging word embeddings from paper titles and abstracts.
- **Reddit** [22] dataset models a social network with 232,965 posts (nodes) and 114,615,892 user interactions (edges). Nodes are assigned to 41 communities representing post topics.

The details of the datasets are shown in Table I.

2) Baseline Methods: To evaluate the proposed GCGP method comprehensively, we compare it against various graph condensation methods serving as baselines, including,

- **Random** selects nodes randomly to form the condensed dataset.
- **K-center** [52] selects k centroids as the condensed dataset based on distance metrics.
- **GCond** [15] is a bi-level graph condensation method that employs a GNN as the prediction model.
- **One-step** [16] simplifies the GCond method by performing the inner and outer optimization steps only once, thereby accelerating the condensation process.
- **GC-SNTK** [18] is a graph condensation method based on kernel ridge regression, utilizing the structure-based neural tangent kernel to quantify similarity among nodes.

We evaluate graph condensation using three methods: GCond, GC-SNTK, and the proposed GCGP, under two distinct scenarios. In the first scenario, the condensed graph contains only node features X, while its adjacency matrix $A^{\mathcal{S}}$ is fixed as the identity matrix I. In the second scenario, both the node features X and the graph structure $A^{\mathcal{S}}$, are optimized during condensation. These scenarios allow for a comprehensive comparison of the methods' performance under varying levels of structural information.

The GCond framework supports various combinations of GNNs for the condensation and testing stages. By default, the

TABLE III

THE MODEL'S AVERAGE ACCURACY AND STANDARD DEVIATION ARE ASSESSED BASED ON THE CONDENSED GRAPH DATA. GREEN SHADING HIGHLIGHTS THE TOP THREE CONDENSATION METHODS IN EACH ROW, WITH DARKER SHADES DENOTING SUPERIOR PERFORMANCE.

Dataset	Ratio (Size)	Random	K-Center	One-step	GCond		GC-SNTK		GCGP (Our)		Full (GCN)
					X	X,A	X	X,A	X	X,A	
Cora	1.30% (35)	63.6±3.7	64.0±2.3	80.2±0.73	75.9±1.2	81.2±0.7	82.2±0.3	81.7±0.7	82.4±0.3	78.8±1.0	81.1±0.5
	2.60% (70)	72.8±1.1	73.2±1.2	80.4±1.77	75.7±0.9	81.0±0.6	82.4±0.5	81.5±0.7	82.5±0.7	80.5±0.9	
	5.20% (140)	76.8±0.1	76.7±0.1	79.8±0.64	76.0±0.9	81.1±0.5	82.1±0.1	81.3±0.2	82.6±0.7	80.9±1.0	
Citeseer	0.90%(30)	54.4±4.4	52.4±2.8	70.8±0.3	71.6±0.8	71.8±1.2	65.7±0.3	64.8±0.7	73.1±1.5	70.3±1.5	71.7±0.1
	1.80%(60)	64.2±1.7	64.3±1.0	71.7±0.6	71.2±0.1	72.6±0.9	67.0±0.3	65.9±0.2	72.8±0.6	71.3±1.0	
	3.61%(120)	69.1±0.1	69.1±0.1	70.1±0.2	71.4±0.6	72.5±0.4	67.3±0.3	66.3±0.5	72.3±0.5	71.1±0.5	
Pubmed	0.08% (15)	69.5±0.5	69.0±0.6	77.7±0.12	59.4±0.7	78.3±0.2	78.9±0.7	71.8±6.8	79.2±0.5	74.2±0.7	77.1±0.3
	0.15% (30)	73.8±0.8	73.7±0.8	77.8±0.17	51.7±0.4	77.1±0.3	79.3±0.3	74.0±4.9	79.2±0.7	76.8±0.9	
	0.30% (60)	77.9±0.4	77.8±0.5	77.1±0.44	60.8±1.7	78.4±0.3	79.4±0.3	76.4±2.8	79.2±0.8	77.5±0.5	
Photo	0.5% (40)	43.4±3.7	58.6±0.9	85.7±0.4	86.6±0.6	85.5±0.2	88.6±0.1	82.0±2.5	92.1±0.4	90.2±0.9	91.4±0.8
	1.0% (80)	52.4±3.3	58.0±3.4	86.2±0.2	87.3±0.4	86.5±0.5	88.5±0.2	83.7±1.0	92.0±0.1	90.6±0.5	
	2.1% (160)	72.3±0.8	71.9±1.7	86.8±0.6	87.7±0.3	86.6±0.6	88.6±0.6	83.8±3.0	92.1±0.3	91.4±0.6	
Computers	0.4% (50)	22.4±3.6	44.7±0.6	82.1±0.3	82.4±0.6	82.7±0.3	83.5±0.3	77.2±5.2	86.1±0.3	85.1±0.5	84.2±0.3
	0.7% (100)	40.2±1.3	51.1±1.5	82.4±0.5	82.2±0.5	82.5±0.2	83.1±0.6	77.3±3.5	86.4±0.3	85.4±0.4	
	1.5% (200)	51.5±1.7	55.1±2.0	82.3±0.6	82.7±0.3	82.8±0.4	83.0±0.7	77.7±5.0	86.5±0.4	86.0±0.5	
Ogbn-arxiv	0.05% (90)	47.1±3.9	47.2±3.0	59.2±0.1	61.3±0.5	59.2±1.1	63.5±0.3	64.2±0.2	65.7±0.3	65.0±0.4	71.4±0.1
	0.25% (454)	57.3±1.1	56.8±0.8	60.1±0.8	64.2±0.4	63.2±0.3	64.5±0.1	65.1±0.8	65.3±0.1	65.4±0.1	
	0.5% (909)	60.0±0.9	60.3±0.4	60.0±0.1	63.1±0.5	64.0±0.4	65.7±0.4	65.4±0.5	65.7±0.1	65.3±0.3	
Reddit	0.05% (77)	46.1±4.4	46.6±2.3	80.7±0.2	88.4±0.4	88.0±1.8	77.9±0.9	74.3±0.5	93.9±0.1	92.7±0.1	93.9±0.0
	0.1%(153)	58.0±2.2	53.0±3.3	81.1±0.4	89.3±0.1	89.6±0.7	78.7±0.6	74.8±0.7	93.8±0.1	92.6±0.1	
	0.2% (307)	66.3±1.9	58.5±2.1	82.3±0.7	88.8±0.4	90.1±0.5	78.9±0.1	85.2±1.2	94.0±0.1	92.6±0.4	

experimental section adopts the best-performing configuration: SGC [10] for condensation and GCN [53] for testing, unless otherwise specified.

3) Hyperparameter Settings: The hyperparameters analyzed in this study include β in the GP and the power k of the adjacency matrix A used in the covariance function computation. To ensure consistency across datasets, β is assigned the values $10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 5, 10$ for most datasets, except for Reddit, where it is set to $10^{-3}, 10^{-2}, 10^{-1}, 1, 10$. Similarly, the parameter k is evaluated with $k = 1, 2, 3, 4, 5$ for the Cora, Citeseer, Pubmed, Photo, and Computer datasets, while $k = 1, 2, 3, 4$ is used for the Ogbn-arxiv and Reddit datasets. The detailed hyperparameter configurations and the corresponding optimal values for each dataset are summarized in Table II.

B. Condensation Quality Evaluation

To assess the quality of synthetic graphs condensed by the proposed GCGP method, we perform node classification experiments across seven datasets, each with varying condensation scales. For the Cora, Citeseer, Pubmed, Photo, and Computers datasets, three condensation scales of 5, 10, and 20 nodes per class are selected. For the Ogbn-arxiv dataset, we use 0.05% (90 nodes), 0.25% (454 nodes), and 0.5% (909 nodes) of the training set as the condensation scales. Similarly, for the Reddit dataset, the condensation scales are set to 0.05% (77 nodes), 0.1% (153 nodes), and 0.2% (307 nodes). The results of these experiments are summarized in Table III. Overall, GCGP achieves the best performance across most datasets and condensation sizes. Its advantages are particularly evident under high condensation ratios, where the amount of nodes is significantly reduced.

For instance, on the Citeseer dataset at a low condensation ratio of 0.9%, GCGP achieves a test accuracy of 73.1%, outperforming other methods. On the Amazon Photo and Amazon Computers datasets, GCGP consistently demonstrates superior performance across all condensation ratios. Specifically, on the Amazon Photo dataset at a 0.5% condensation ratio, GCGP achieves a classification accuracy of 92.1%, compared to 88.6% by GC-SNTK. Similarly, on the Amazon Computers dataset, GCGP achieves 86.1% accuracy at a 0.4% condensation ratio.

On large-scale datasets such as Ogbn-arxiv and Reddit, GCGP also delivers competitive results. For the Ogbn-arxiv dataset, GCGP achieves a test accuracy of 65.7% using only 90 condensed nodes. On the Reddit dataset, GCGP achieves the highest test accuracy of 93.9% at a 0.05% condensation ratio, using only 77 synthetic nodes. This result is comparable to the performance of training on the original 153,431 nodes with GCN.

In summary, GCGP outperforms existing baseline methods in graph data condensation, particularly under high condensation ratios and on large-scale datasets. Notably, for six out of the seven datasets (excluding Ogbn-arxiv), the results obtained with condensed data surpass those achieved by training on the full dataset with GCN. Future work explores its scalability to even larger datasets and its potential applications in other tasks, such as node classification and link prediction.

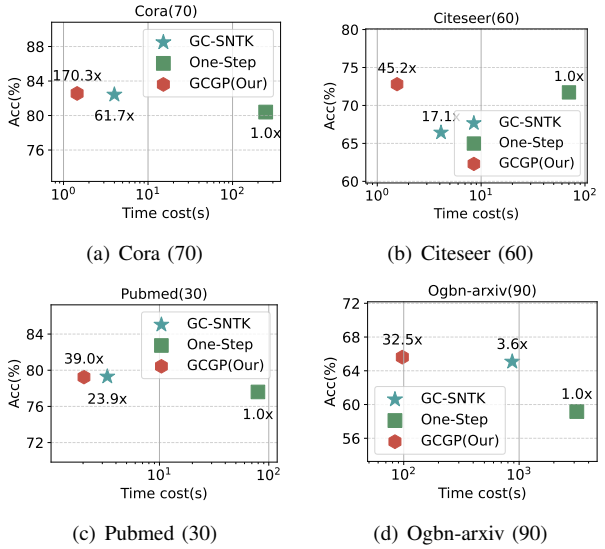
C. Efficiency Evaluation

To evaluate the efficiency of the proposed GCGP method, we examine its graph condensation time across graphs of varying condensation scales. The assessment involves a comparison with baseline methods, focusing on computational performance for both small-scale and large-scale graphs. This analysis aims to highlight the scalability and practical applicability of the proposed approach.

Specifically, we compare the GCGP method with GC-SNTK and the One-step method. The GCond method is excluded from the comparison due to its excessive time consumption. Instead, we include the simplified and faster version of GCond, the One-step method, and perform experiments on datasets including Cora, Citeseer, Pubmed, and Reddit. The results, shown in Figure 4, demonstrate that the proposed GCGP method significantly improves condensation efficiency while maintaining the quality of the condensed graphs compared to GC-SNTK and the One-step method.

In Figure 4, the x-axis represents the time required for the condensation process, while the y-axis indicates the classification accuracy of the condensed data. Each marker in the figure corresponds to the time required by a method to achieve a specific accuracy. Using the One-step method as the baseline, we annotate the relative speedup of each method next to its marker. For instance, on the Cora dataset, when condensing the dataset to 70 nodes, GC-SNTK achieves a speedup of 61.7 times compared to the One-step method, while the proposed GCGP method achieves a remarkable speedup of 170.3 times. Similarly, on the Citeseer dataset, GCGP is 45.2 times faster than the One-step method. Even on the large-scale Ogbn-arxiv dataset, GCGP achieves a speedup of 32.5 times. These results highlight the practical significance of such speed improvements, as the primary goal of graph condensation is to accelerate the training process. If the condensation process itself is excessively time-consuming, it undermines its purpose, as training directly on the original dataset would be faster. Therefore, an efficient condensation method like GCGP greatly enhances the applicability of graph condensation by minimizing the time required for this step.

To further evaluate the efficiency of the proposed method, we conduct a detailed comparison between GCGP and the existing method, GC-SNTK. The results, presented in Figure 5, provide a more comprehensive view of the efficiency improvements. Specifically, experiments are conducted on five datasets—Cora, Citeseer, Pubmed, Photo, and Computers—under three different condensation scales, resulting in 15 experimental settings. These results record the variation in classification accuracy of the condensed data with the training time consumed during the condensation process for both GCGP and GC-SNTK. In Figure 5, the x-axis denotes the training time required for the models, while the y-axis represents the classification accuracy achieved. This figure provides a comparative analysis of the performance of the proposed GCGP method and the baseline GC-SNTK method. As illustrated, the GCGP method demonstrates a



(a) Cora (70) (b) Citeseer (60)

(c) Pubmed (30) (d) Ogbn-arxiv (90)

Fig. 4. The runtime efficiency of GCGP and GC-SNTK is assessed using the One-step method as the baseline. To quantify relative performance, a speedup factor is calculated by dividing the runtime of the One-step method by the runtimes of the other methods. This factor, representing the relative acceleration of each method, is presented alongside the markers in the figure.

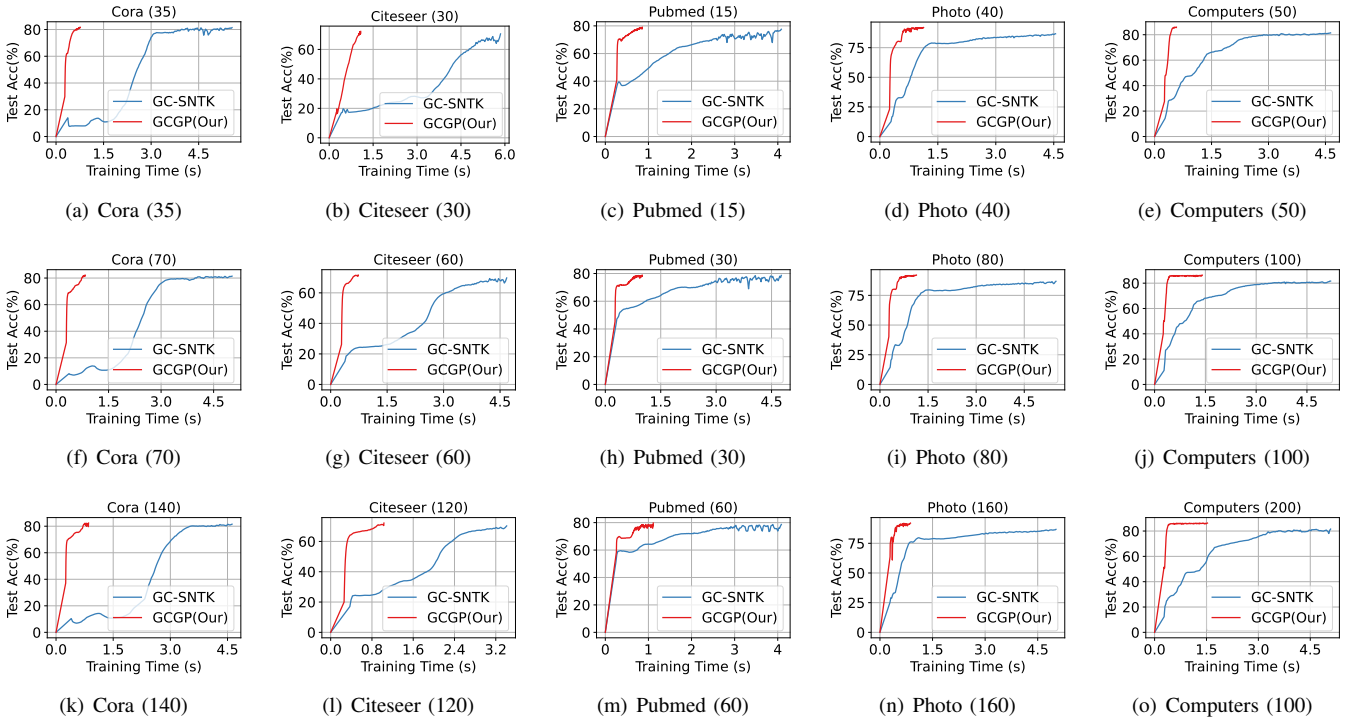
more efficient condensation process, requiring less training time to achieve higher levels of accuracy. Furthermore, across most evaluated datasets, the GCGP method achieves superior final classification accuracy compared to GC-SNTK. These results highlight the effectiveness of the proposed approach in both accelerating the condensation process and improving classification performance.

Additionally, Table IV summarizes the time consumed by the two methods across the 15 experimental settings, along with the speedup achieved by GCGP relative to GC-SNTK. The data clearly show that the proposed method is consistently more than three times faster than GC-SNTK while maintaining the quality of the condensed graphs. This stability and efficiency further underscore the robustness of the GCGP method.

In summary, the experimental results across all datasets confirm the significant efficiency improvements achieved by the proposed GCGP method. It consistently outperforms the fastest existing method, GC-SNTK, by more than three times in speed, thereby enhancing its scalability and broadening its potential application scenarios.

D. Parameter Sensitivity Analysis

To analyze the sensitivity of the proposed method to hyperparameters, we conduct experiments by varying two key parameters: β in GP and the power k of the adjacency matrix A in the calculation of covariance function. These experiments are performed on multiple datasets, including Cora, Citeseer, Pubmed, Photo, Computer, and Ogbn-arxiv, with β set to $10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 5, 10$. For the Reddit dataset, β is adjusted to $10^{-3}, 10^{-2}, 10^{-1}, 1, 10$. The parameter k is tested with values $1, 2, 3, 4, 5$ for the Cora, Citeseer, Pubmed, Photo, and Computer datasets, and $1, 2, 3, 4$ for Ogbn-arxiv and



(a) Cora (35) (b) Citeseer (30) (c) Pubmed (15) (d) Photo (40) (e) Computers (50)

(f) Cora (70) (g) Citeseer (60) (h) Pubmed (30) (i) Photo (80) (j) Computers (100)

(k) Cora (140) (l) Citeseer (120) (m) Pubmed (60) (n) Photo (160) (o) Computers (100)

Fig. 5. Condensation efficiency consumption between the proposed GCGP method and GC-SNTK is conducted on five datasets. The x-axis represents training time, while the y-axis indicates test accuracy. The results show that GCGP consistently achieves faster training times than GC-SNTK across all condensation scales on these datasets.

TABLE IV
CONDESNATION TIME COMPARISON OF THE GC-SNTK AND THE GCGP

Dataset	Size	GC-SNTK (s)	GCGP (s)	Speedup
Cora	35	5.5843	0.7672	7.3×
	70	5.0616	0.8383	6.0×
	140	4.6553	0.8595	5.4×
Citeseer	30	5.8787	1.0806	5.4×
	60	4.7093	0.7389	6.4×
	120	3.4389	1.0386	3.3×
Pubmed	15	4.1034	0.8610	4.8×
	30	4.7844	1.0022	4.8×
	60	4.1030	1.1172	3.7×
Photo	40	4.5685	1.1389	4.0×
	80	5.5139	1.1430	4.8×
	160	5.0429	0.8767	5.8×
Computers	50	4.6563	0.5787	8.0×
	100	5.2212	1.4125	3.7×
	200	5.1126	1.5352	3.3×

Reddit. The condensation sizes for the datasets are fixed as follows: 70 for Cora, 60 for Citeseer, 30 for Pubmed, 80 for Photo, 100 for Computer, 90 for Ogbn-arxiv, and 77 for Reddit. All combinations of β and k are evaluated, and the average classification accuracy is computed over multiple iterations. The results, illustrated in Figure 6 and Figure 7, show the classification accuracy achieved on the condensed graph for each parameter setting. In these figures, darker bar colors correspond to higher classification accuracy.

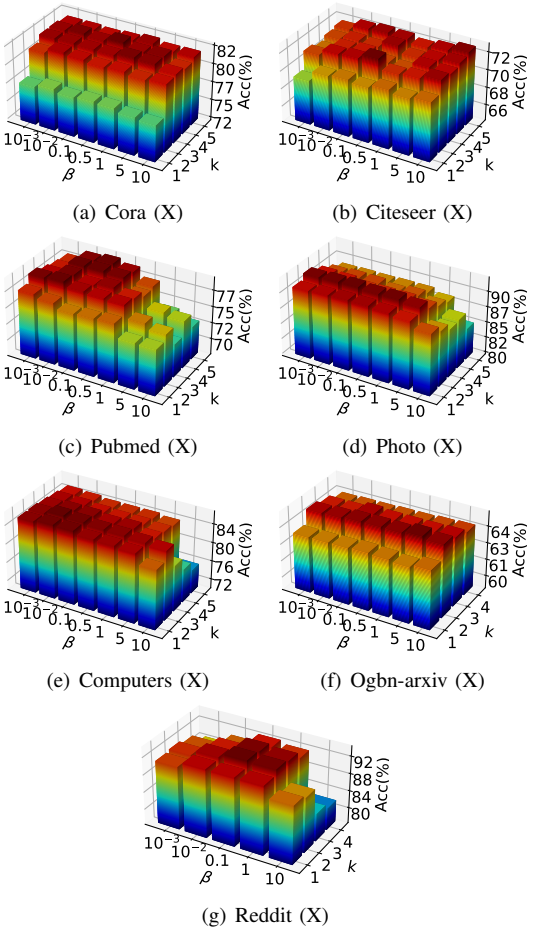
As shown in Figure 6, which depicts the results for experiments condensing only node features, the accuracy is generally insensitive to parameter changes. For the Cora dataset, performance improves significantly and stabilizes when $k > 1$. The Citeseer dataset maintains high classification accuracy even for smaller k values. The Pubmed dataset achieves favorable and stable results when β is small, while the Photo dataset performs best and remains stable for smaller k values. Among these datasets, the Computers dataset exhibits the most stable performance, with only minor variations when β is large. For large-scale datasets, Ogbn-arxiv shows minimal sensitivity to both parameters, whereas Reddit performs poorly when β and k are large.

Figure 7 illustrates the results when both structural and feature information are condensed. A consistent trend can be observed: performance degrades significantly when both k and β are large. Conversely, the method achieves better results when both k and β are small.

In summary, the proposed method demonstrates varying degrees of sensitivity to hyperparameters across datasets. While some datasets exhibit stable performance regardless of parameter changes, others show noticeable performance differences depending on the parameter settings. These findings provide valuable insights into the selection of hyperparameters for different datasets.

E. Generalize to Other Models

To assess whether the graph data condensed using the GCGP method preserves the structural and semantic infor-



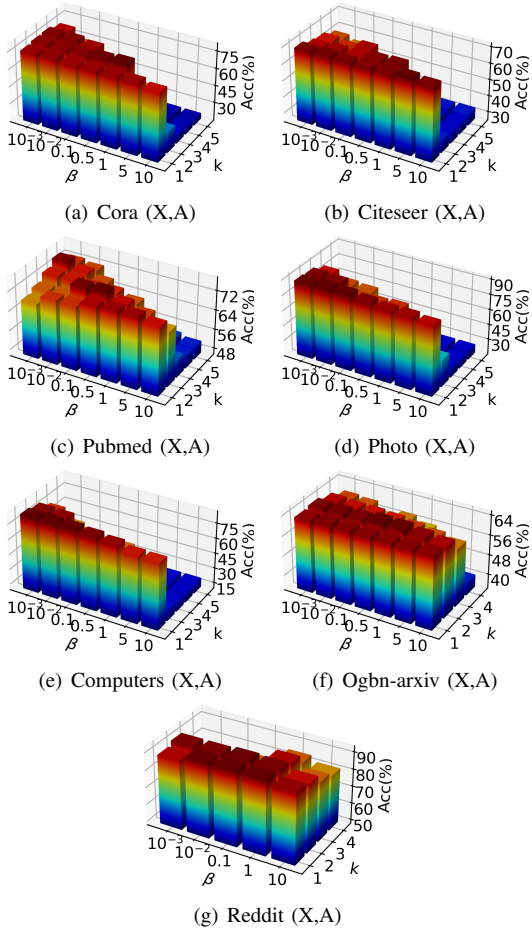
(a) Cora (X) (b) Citeseer (X)

(c) Pubmed (X) (d) Photo (X)

(e) Computers (X) (f) Ogbn-arxiv (X)

(g) Reddit (X)

Fig. 6. Parameter sensitive analysis with condense only node features $X^{\mathcal{S}}$. The height of each bar represents the average classification accuracy on the condensed graph under the corresponding parameter settings. Darker bar colors indicate higher classification accuracy.



(a) Cora (X,A) (b) Citeseer (X,A)

(c) Pubmed (X,A) (d) Photo (X,A)

(e) Computers (X,A) (f) Ogbn-arxiv (X,A)

(g) Reddit (X,A)

Fig. 7. Parameter sensitivity analysis condenses both node features $X^{\mathcal{S}}$ and structural information $A^{\mathcal{S}}$.

mation necessary for downstream tasks, we conduct a series of experiments. These experiments involve applying the condensed data to various models to evaluate its adaptability and effectiveness in contexts beyond the initial condensation process. The results demonstrate the robustness and applicability of the GCGP method, offering a basis for further exploration of its utility across diverse scenarios.

Table V summarizes the generalization performance of graph data condensed by various methods—GCond, GC-SNTK, and the proposed GCGP—on multiple GNN models, including GCN [53], SGC [10], APPNP [54], GraphSAGE [22], and KRR [55], across the Cora, Citeseer, and Pubmed datasets. Notably, the proposed GCGP method demonstrates competitive performance, achieving the highest or near-highest average accuracy across a majority of datasets and models. On the Cora dataset, GCGP achieves the highest average accuracy of 78.7%, marginally surpassing the performance of GC-SNTK, which records an average accuracy of 78.0%. It demonstrates strong generalization across architectures, achieving an accuracy of 82.6% with KRR and 79.0% with SGC. In contrast, baseline methods such as GCN-based condensation and APPNP-based condensation show significantly lower average accuracies of 30.1% and 53.7%, respectively. These results underscore the robustness of GCGP compared to alternative approaches.

On the Citeseer dataset, GCGP achieves the highest average accuracy of 68.4%, outperforming GC-SNTK, which achieves 42.2%, and matching the performance of SGC-based condensation, which records 68.2%. Its performance is particularly strong with KRR, achieving an accuracy of 72.3%, and with SGC, achieving 68.6%. These results demonstrate the stable performance of GCGP across different models. In contrast, GC-SNTK exhibits limited effectiveness on this dataset, highlighting the adaptability of GCGP to varying graph structures.

On the Pubmed dataset, GCGP achieves an average accuracy of 71.4%, which is competitive with GC-SNTK, achieving 75.7%, and SGC-based condensation, achieving 76.2%. The method performs strongly with APPNP and KRR, achieving accuracies of 77.0% and 79.2%, respectively. However, its accuracy with SGC is comparatively lower at 50.9%. Despite this variation, GCGP demonstrates consistent performance across models, reflecting its reliability in diverse scenarios.

In conclusion, GCGP demonstrates robust generalization performance across all datasets and models. Its strong results

TABLE V
GENERALIZATION OF THE CONDENSED GRAPH DATA

Dataset	Method		GCN	SGC	APPNP	SAGE	KRR	Avg.
Cora (140)	Gcond	GCN	31.9	34.8	33.5	32.6	17.5	30.1
		SGC	81.1	80.6	80.5	71.0	78.8	78.4
		APPNP	31.9	50.3	75.7	51.0	59.8	53.7
		SAGE	63.6	41.2	68.9	53.5	26.5	50.7
	GC-SNTK		77.8	77.9	74.6	78.0	82.1	78.1
	GCGP		79.7	79.0	74.1	78.1	82.6	78.7
Citeseer (120)	GCond	GCN	27.6	26.9	33.3	24.4	14.8	25.4
		SGC	72.6	65.4	71.5	62.2	69.5	68.2
		APPNP	52.6	56.8	71.2	45.1	54.1	56.0
		SAGE	38.6	23.6	64.1	44.3	43.4	42.8
	GC-SNTK		20.9	22.1	52.2	48.4	67.3	42.2
	GCGP		68.5	68.6	66.6	66.2	72.3	68.4
Pubmed (60)	GCond	GCN	43.7	62.6	61.6	70.3	66.1	60.9
		SGC	78.4	74.9	77.6	76.0	74.1	76.2
		APPNP	67.7	57.8	77.2	64.1	59.9	65.3
		SAGE	71.2	58.8	74.2	69.0	52.5	65.1
	GC-SNTK		73.4	74.5	76.6	73.2	79.4	75.4
	GCGP		75.2	50.9	77.0	74.7	79.2	71.4

with KRR suggest its effectiveness in preserving critical graph structures and features. Compared to baseline methods, GCGP consistently delivers superior outcomes, making it a reliable approach for graph data condensation in diverse downstream tasks.

F. Ablation Study

To evaluate the capacity of the proposed covariance function to model node similarity, we conduct experiments comparing it against various covariance functions, including the dot product, the neural tangent kernel (NTK) [24], and the structure-based neural tangent kernel (SNTK) [18]. The results of these experiments are presented in Table VI, providing a comparative evaluation of their performance.

Table VI summarizes the performance of various covariance functions across multiple datasets, demonstrating the effectiveness of the proposed GCGP covariance function. The experiments compare four covariance functions: Dot Product, NTK, SNTK, and GCGP, with results showing that GCGP consistently outperforms the alternatives in most scenarios.

On datasets such as Cora, Citeseer, and Pubmed, GCGP achieves higher accuracy compared to other methods, particularly outperforming simpler covariance functions like Dot Product and NTK. These results highlight the ability of advanced covariance designs to better capture graph structural properties. For Photo and Computers, GCGP demonstrates notable improvements over competing methods, further validating its effectiveness in diverse data domains.

On large-scale datasets like Ogbn-arxiv and Reddit, GCGP consistently achieves superior performance, excelling in scenarios with complex graph structures and large data volumes. These results reinforce its scalability and robustness.

In summary, the findings underscore the importance of covariance function design in model performance. GCGP consistently delivers strong results across datasets, particularly excelling in challenging and large-scale scenarios, validating its capability to effectively model graph structures and feature information.

TABLE VI
ABLATION OF THE COVARIANCE FUNCTION

Dataset	**Covariance Function**			
	Dot Product	**NTK**	**SNTK**	**GCGP**
Cora (70)	57.3±0.5	59.1±1.4	82.4±0.5	**82.5±0.7**
Citeseer (60)	60.9±1.2	62.7±0.6	67.0±0.3	**72.8±0.6**
Pubmed (30)	72.7±0.4	69.9±1.2	**79.3±0.3**	79.2±0.7
Photo (80)	87.6±1.9	82.9±0.3	88.5±0.2	**92.0±0.1**
Computers (100)	80.7±1.2	72.5±0.9	83.1±0.6	**86.4±0.3**
Ogbn-arxiv (90)	50.6±0.5	51.3±0.1	64.2±0.2	**65.7±0.3**
Reddit (77)	66.3±0.1	66.3±0.2	77.9±0.9	**93.9±0.1**

VI. CONCLUSION

In this paper, we propose a novel GP-based framework for graph data condensation, addressing computational challenges of existing bi-level methods. By treating the condensed graph as GP observations, our approach eliminates iterative GNN training, enhancing computational efficiency while maintaining predictive performance. To capture graph structure, we design a covariance function that incorporates k-hop message passing, representing local structures with minimal computational overhead. We propose using the binary concrete relaxation to optimize the adjacency matrix. This relaxes the discrete adjacency matrix into a differentiable one, enabling efficient optimization of graph structure. In summary, our contributions are: (1) integrating GP into graph condensation to eliminate iterative GNN training; (2) designing a covariance function to capture node similarities; (3) introducing the binary concrete relaxation for differentiable optimization of discrete graph structures; and (4) empirically validating the method on diverse datasets and baselines. Together, these contributions enhance the efficiency and practicality of graph condensation.

REFERENCES

[1] T. Aichner, M. Grünfelder, O. Maurer, and D. Jegeni, "Twenty-five years of social media: a review of social media applications and definitions from 1994 to 2019," *Cyberpsychology, behavior, and social networking*, vol. 24, no. 4, pp. 215–222, 2021.

[2] P. Verduyn, N. Gugushvili, K. Massar, K. Täht, and E. Kross, "Social comparison on social networking sites," *Current opinion in psychology*, vol. 36, pp. 32–37, 2020.

[3] M. Diao, H. Kong, and J. Zhao, "Impacts of transportation network companies on urban mobility," *Nature Sustainability*, vol. 4, no. 6, pp. 494–500, 2021.

[4] Y. Gu, X. Fu, Z. Liu, X. Xu, and A. Chen, "Performance of transportation network under perturbations: Reliability, vulnerability, and resilience," *Transportation Research Part E: Logistics and Transportation Review*, vol. 133, p. 101809, 2020.

[5] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, "A compact review of molecular property prediction with graph neural networks," *Drug Discovery Today: Technologies*, vol. 37, pp. 1–12, 2020.

[6] P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. van Hoesel, H. Schopmans, T. Sommer, *et al.*, "Graph neural networks for materials science and chemistry," *Communications Materials*, vol. 3, no. 1, p. 93, 2022.

[7] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," *arXiv preprint arXiv:2105.14491*, 2021.

[8] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *arXiv preprint arXiv:1810.00826*, 2018.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[10] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*, pp. 6861–6871, PMLR, 2019.

[11] D. Chakrabarti and C. Faloutsos, *Graph mining: laws, tools, and case studies*. Springer Nature, 2022.

[12] J. Kang, J. He, R. Maciejewski, and H. Tong, "Inform: Individual fairness on graph mining," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 379–389, 2020.

[13] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "Ogb-lsc: A large-scale challenge for machine learning on graphs," *arXiv preprint arXiv:2103.09430*, 2021.

[14] K. Duan, Z. Liu, P. Wang, W. Zheng, K. Zhou, T. Chen, X. Hu, and Z. Wang, "A comprehensive study on large-scale graph training: Benchmarking and rethinking," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5376–5389, 2022.

[15] W. Jin, L. Zhao, S. Zhang, Y. Liu, J. Tang, and N. Shah, "Graph condensation for graph neural networks," *arXiv preprint arXiv:2110.07580*, 2021.

[16] W. Jin, X. Tang, H. Jiang, Z. Li, D. Zhang, J. Tang, and B. Yin, "Condensing graphs via one-step gradient matching," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 720–730, 2022.

[17] Y. Zhang, T. Zhang, K. Wang, Z. Guo, Y. Liang, X. Bresson, W. Jin, and Y. You, "Navigating complexity: Toward lossless graph condensation via expanding window matching," *arXiv preprint arXiv:2402.05011*, 2024.

[18] L. Wang, W. Fan, J. Li, Y. Ma, and Q. Li, "Fast graph condensation with structure-based neural tangent kernel," in *Proceedings of the ACM on Web Conference 2024*, pp. 4439–4448, 2024.

[19] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[20] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," *arXiv preprint arXiv:1711.00165*, 2017.

[21] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[23] C. Williams, "Computing with infinite networks," *Advances in neural information processing systems*, vol. 9, 1996.

[24] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[25] A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani, "Gaussian process behaviour in wide deep neural networks," *arXiv preprint arXiv:1804.11271*, 2018.

[26] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018.

[27] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," *arXiv preprint arXiv:2006.05929*, 2020.

[28] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International conference on machine learning*, pp. 2113–2122, PMLR, 2015.

[29] B. Zhao and H. Bilen, "Dataset condensation with differentiable siamese augmentation," in *International Conference on Machine Learning*, pp. 12674–12685, PMLR, 2021.

[30] S. Lee, S. Chun, S. Jung, S. Yun, and S. Yoon, "Dataset condensation with contrastive signals," in *International Conference on Machine Learning*, pp. 12352–12364, PMLR, 2022.

[31] K. Wang, B. Zhao, X. Peng, Z. Zhu, S. Yang, S. Wang, G. Huang, H. Bilen, X. Wang, and Y. You, "Cafe: Learning to condense dataset by aligning features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12196–12205, 2022.

[32] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu, "Dataset distillation by matching training trajectories," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4750–4759, 2022.

[33] B. Zhao and H. Bilen, "Dataset condensation with distribution matching," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6514–6523, 2023.

[34] T. Nguyen, Z. Chen, and J. Lee, "Dataset meta-learning from kernel ridge-regression," *arXiv preprint arXiv:2011.00050*, 2020.

[35] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, "Graph neural tangent kernel: Fusing graph neural networks with graph kernels," *Advances in neural information processing systems*, vol. 32, 2019.

[36] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net," *Advances in neural information processing systems*, vol. 32, 2019.

[37] Z. Li, R. Wang, D. Yu, S. S. Du, W. Hu, R. Salakhutdinov, and S. Arora, "Enhanced convolutional neural tangent kernels," *arXiv preprint arXiv:1911.00809*, 2019.

[38] X. Zheng, M. Zhang, C. Chen, Q. V. H. Nguyen, X. Zhu, and S. Pan, "Structure-free graph condensation: From large-scale graphs to condensed graph-free data," *arXiv preprint arXiv:2306.02664*, 2023.

[39] M. Liu, S. Li, X. Chen, and L. Song, "Graph condensation via receptive field distribution matching," *arXiv preprint arXiv:2206.13697*, 2022.

[40] Z. Xu, Y. Chen, M. Pan, H. Chen, M. Das, H. Yang, and H. Tong, "Kernel ridge regression-based graph dataset distillation," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2850–2861, 2023.

[41] D. J. MacKay *et al.*, "Introduction to gaussian processes," *NATO ASI series F computer and systems sciences*, vol. 168, pp. 133–166, 1998.

[42] A. Damianou and N. D. Lawrence, "Deep gaussian processes," in *Artificial intelligence and statistics*, pp. 207–215, PMLR, 2013.

[43] R. M. Neal, *Bayesian learning for neural networks*, vol. 118. Springer Science & Business Media, 2012.

[44] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[45] C. Avin, "Distance graphs: From random geometric graphs to bernoulli graphs and between," in *Proceedings of the fifth international workshop on Foundations of mobile computing*, pp. 71–78, 2008.

[46] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[47] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*, pp. 40–48, PMLR, 2016.

[48] C.-z. Liu, Y.-x. Sheng, Z.-q. Wei, and Y.-Q. Yang, "Research of text classification based on improved tf-idf algorithm," in *2018 IEEE international conference of intelligent robotic and control engineering (IRCE)*, pp. 218–222, IEEE, 2018.

[49] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.

[50] W. A. Qader, M. M. Ameen, and B. I. Ahmed, "An overview of bag of words; importance, implementation, applications, and challenges," in *2019 international engineering conference (IEC)*, pp. 200–204, IEEE, 2019.

[51] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[52] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[53] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[54] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[55] V. Vovk, "Kernel ridge regression," in *Empirical inference: Festschrift in honor of vladimir n. vapnik*, pp. 105–116, Springer, 2013.