EFFICIENT GRAPH CONDENSATION: A UNIFIED FRAMEWORK FROM KERNEL METHODS TO GAUSSIAN PROCESSES

WANG LIN

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University
Department of Computing

Efficient Graph Condensation: A Unified Framework from Kernel Methods to Gaussian Processes

WANG Lin

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
August 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: WANG Lin

Abstract

The rapid development of Internet technology has given rise to a vast amount of graph-structured data. Graph Neural Networks (GNNs), as an effective method for various graph mining tasks, incur substantial computational resource costs when dealing with large-scale graph data. This computational bottleneck significantly limits the practical applications of GNNs in real-world scenarios where efficiency and scalability are crucial. To address this challenge, graph condensation has emerged as a promising data-centric solution that aims to condense large graph datasets into smaller, representative subsets without sacrificing the predictive performance of GNNs. However, existing graph condensation methods often rely on computationally intensive bi-level optimization architectures, which themselves suffer from massive computation costs and limited scalability.

In this thesis, we propose a comprehensive framework that addresses the computational challenges of graph condensation through three interconnected contributions:

1. **Graph Condensation with Structure-based Neural Tangent Kernel (GC-SNTK)**: We reformulate the graph condensation problem as a Kernel Ridge Regression (KRR) task instead of iteratively training GNNs in the inner loop of bi-level optimization. This approach utilizes a Structure-based Neural Tangent Kernel (SNTK) to capture the topology of graphs and serves as the kernel function in the KRR paradigm, significantly accelerating graph condensation while maintaining high prediction performance.

2. **Simplified Graph Neural Tangent Kernel (SGTK) and Simplified Graph Neural Kernel (SGNK)**: We address the computational limitations of existing Graph Neural Tangent Kernel (GNTK) methods, which suffer from redundant computations due to layer-stacking strategies. SGTK replaces the traditional multi-layer stacking mechanism with a continuous K-step aggregation operation, streamlining the iterative kernel computation process while preserving expressiveness. SGNK models infinitely wide Graph Neural Networks as Gaussian Processes, allowing kernel values to be directly determined from the expected outputs of activation functions in the infinite-width regime, further reducing computational complexity.

3. **Graph Condensation via Gaussian Process (GCGP)**: We demonstrate the synergistic potential of our kernel methods by proposing GCGP, which leverages the computational efficiency of our simplified kernels to achieve even faster graph condensation. GCGP utilizes a Gaussian Process with the condensed graph serving as observations to estimate the posterior distribution of predictions, eliminating the need for iterative and resource-intensive GNN training. We derive a specialized covariance function that incorporates structural information through local neighborhood aggregation and utilize Concrete random variables to approximate binary adjacency matrices in continuous counterparts, enabling gradient-based optimization for discrete graph structures.

Our comprehensive experimental resultsdemonstrate that this integrated approach achieves superior computational efficiency while maintaining high prediction performance across various graph mining tasks. The proposed methods collectively address the scalability challenges of Graph Neural Networks through a unified framework that combines the benefits of kernel methods with graph condensation techniques.

Keywords: Graph Neural Networks, Graph Condensation, Neural Tangent Kernel, Gaussian Process, Kernel Methods, Computational Efficiency, Graph Mining

Publications Arising from the Thesis

1. Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, Qing Li, "Fast Graph Condensation with Structure-based Neural Tangent Kernel", in *Proceedings of the ACM Web Conference 2024*.

2. Lin Wang, Shijie Wang, Sirui Huang, and Qing Li, "Simplifying Graph Neural Kernels: from Stacking Layers to Collapsed Structure", *The Joint Conference of The 24th International Conference on Web-Based Learning and The 10th International Symposium on Emerging Technologies for Education (ICWL-SETE 2025)*.

3. Lin Wang, and Qing Li, "Efficient Graph Condensation via Gaussian Process", manuscript submitted to *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2025.

Acknowledgments

The completion of this thesis marks the culmination of a transformative journey that has shaped both my academic career and personal growth. This work represents the collaborative effort of many individuals whose guidance, support, and encouragement have been invaluable throughout this process.

The path to a PhD is never a solitary one. It is built upon countless conversations, shared insights, technical assistance, emotional support, and the collective wisdom of mentors, colleagues, family, and friends. As I look back on this journey, I am deeply moved by the generosity of spirit and passion for knowledge that accompanied me throughout.

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Professor Li Qing, for his invaluable guidance, unwavering support, and understanding throughout my PhD studies. His patient mentorship and encouraging words during difficult times played a crucial role in my academic and personal development. I would also like to thank my co-supervisor, Dr. Fan Wenqi, for his kind guidance and helpful advice at the early stage of my research journey.

I am also deeply thankful to Professor Chua Tat-Seng, who supervised me during my visiting period at NUS. During the six months I spent at NExT++, under his leadership, we had a highly collaborative and joyful research experience. His kindness and approachability—whether sharing meals or going for runs with us—made a lasting impression. That time significantly broadened my research perspective and enriched

my academic journey.

I would like to sincerely thank my peers and research partners at PolyU. The spirit of collaboration and shared enthusiasm for research created an inspiring academic environment. Special thanks go to Wang Shijie, Chen Jingfan, Liu Yunqing, Liu Chengyi, Su Jingran, Li Jiatong, Huang Sirui, Sun Xueyao, Ning Liangbo, Qu Haohao, and Zhao Zihuai for their technical support, insightful discussions, and emotional encouragement during both successful and challenging times. I am especially grateful to Zheng Changmeng, Ding Yujun, Bin Yi, Ren Da, Chen Xiao, and Wu Jiahao—my senior peers at PolyU—who guided me both academically and personally during my years at the university.

I also want to express my sincere thanks to my colleagues and research partners at NExT++. The stimulating discussions, collaborative spirit, and shared passion for research created a truly inspiring environment. Special thanks to Zhang Yang, Wu Bin, Lyu Jiayi, Li Haoxuan, Yu Xiaoyan, Zhang Liang, Zhao Xiaoyan, Lin Xinyu, Li Moxin, Zhang Ao, Xu Yiyan, Shi Tianhao, Liu Jiahong, Dang Jisheng, Liu Xiaohao, Shui Ruihao, Lu Kangkang, and Zhou Zhenlin for their support in research and beyond.

To my fellow graduate school friends, especially Ma Zhenyu, Xie Fangyuan, and Wang Hongmei—thank you for staying in touch and offering continued support in both academic and personal matters, even after I graduated from NWPU.

I am also grateful to my close friends, especially Zhao Mingyue and Yin Chenchen, for encouraging me during difficult times. A special thanks to Xu Kunguang, my long-time friend—although we don't talk often, your consistent support has always meant a lot to me.

To my friends in Shenzhen—Zhu Tao, Xiao Xuanji, Tan Wei, Chen Yunyu, Shi Mingyang, Xu Jiyuan, Quan Hongnan, Ren Zihao, and Liu Shizhang—thank you for taking such good care of me. Your help, both in life and academics, has been

invaluable and selfless.

To all my friends, inside and outside academia—thank you for your companionship, encouragement, and for reminding me that there is life beyond research. Your support helped me stay grounded and maintain perspective throughout this journey.

I owe my deepest gratitude to my family for their unconditional love, understanding, and unwavering support. Thank you to my parents and my sister—for teaching me the value of perseverance and hard work. Your belief in me has always been my greatest source of strength. To my girlfriend, Song Xiaoqing—thank you for your patience, understanding, and emotional support during the long process of research and writing. Your trust and encouragement were my anchor in difficult times.

I also want to thank PolyU for the financial support that made this research possible. The scholarship not only supported my studies and daily life but also gave me opportunities to attend international conferences and collaborate with researchers around the world.

My sincere appreciation also goes to the administrative staff in the Department of Computing, especially Jason and Vivian, for their efficient and kind support in handling various administrative matters, which made my academic life much easier.

Finally, I want to thank all the scholars and researchers whose work has inspired and enriched my own. The shared pursuit of knowledge in the academic community has always motivated me and pushed me to contribute meaningfully to the field of graph learning.

This thesis is not only the outcome of my personal efforts but also a reflection of the support, guidance, and encouragement I have received from many people. I am truly grateful for this incredible journey and the opportunity to contribute, even in a small way, to the advancement of knowledge.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides a comprehensive introduction to the research background, motivations, and contributions of this thesis. We begin by establishing the fundamental context of dataset distillation and graph condensation, highlighting the computational challenges faced by Graph Neural Networks (GNNs) when processing large-scale graph data. Following this background, we present the motivations and technical contributions of our three interconnected research works: (1) a fast graph condensation framework based on kernel ridge regression, (2) simplified methods for graph neural kernels, and (3) an efficient graph condensation framework based on Gaussian processes. These contributions collectively address the scalability and efficiency challenges in graph condensation through a unified kernel-based approach. Finally, we outline the structure of the thesis to guide readers through the subsequent chapters.

1.1 Background: Graph Data and Dataset Distillation

1.1.1 Graph Data: Structure, Types, and Applications

Graph-structured data represents a fundamental data structure that captures complex relationships and interactions between entities. Formally, a graph $G = (V, E)$ consists of a set of nodes (vertices) $V = \{v_1, v_2, ..., v_n\}$ and a set of edges $E = \{(v_i, v_j)|v_i, v_j \in V\}$ that connect pairs of nodes. Each node v_i is typically associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$ representing its attributes, while edges may carry additional information such as weights, directions, or types. This flexible representation enables graphs to model a wide variety of real-world phenomena.

The scale of graph data in real-world applications varies dramatically, from small networks with hundreds of nodes to massive graphs with billions of vertices and edges. For instance, social networks like Facebook contain over 2.8 billion active users, while the World Wide Web consists of trillions of interconnected web pages. Biological networks, such as protein-protein interaction networks, can contain millions of proteins and their interactions. These large-scale graphs pose significant computational challenges for traditional graph analysis methods.

Graph data finds extensive applications across diverse domains. In **social network analysis**, graphs model user relationships, enabling community detection, influence analysis, and recommendation systems. **Bioinformatics** utilizes graphs to represent molecular structures, protein interactions, and gene regulatory networks, facilitating drug discovery and disease understanding. **Computer vision** employs graphs to model spatial relationships in images and videos, supporting scene understanding and object detection. **Natural language processing** uses graphs to represent semantic relationships, knowledge graphs, and dependency structures. **Recommendation systems** leverage user-item interaction graphs to provide personalized recommenda-

tions. **Transportation networks** model road systems, public transit, and traffic flow patterns for urban planning and navigation.

1.1.2 Graph Neural Networks: Principles and Challenges

Graph Neural Networks (GNNs) have emerged as a powerful paradigm for learning representations from graph-structured data. Unlike traditional neural networks designed for Euclidean data (e.g., images, text), GNNs operate directly on the non-Euclidean structure of graphs, leveraging both node features and topological information. The core principle of GNNs is the message-passing mechanism, where each node aggregates information from its neighboring nodes to update its representation.

The fundamental operation in GNNs can be expressed as:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{AGGREGATE}^{(l)}\left(\{\mathbf{h}_j^{(l)} : j \in \mathcal{N}(i)\}\right)\right) \tag{1.1}$$

where $\mathbf{h}_i^{(l)}$ is the representation of node i at layer l, $\mathcal{N}(i)$ denotes the neighborhood of node i, AGGREGATE is a permutation-invariant function (e.g., sum, mean, max), $\mathbf{W}^{(l)}$ is a learnable weight matrix, and σ is a non-linear activation function.

Various GNN architectures have been developed to capture different aspects of graph structure. **Graph Convolutional Networks (GCNs)** [43] employ a simple mean aggregation with symmetric normalization. **Graph Attention Networks (GATs)** [83] introduce attention mechanisms to weight neighbor contributions dynamically. **GraphSAGE** [28] samples fixed-size neighborhoods to enable inductive learning on large graphs. **Graph Transformer** models adapt transformer architectures to graph data, capturing long-range dependencies through self-attention mechanisms.

Despite their effectiveness, GNNs face several fundamental challenges when dealing with large-scale graphs. **Computational complexity** scales with the number of nodes and edges, making training on billion-scale graphs prohibitively expensive. **Memory requirements** grow quadratically with graph size, limiting the ability to

process large graphs on standard hardware. **Over-smoothing** occurs when node representations become indistinguishable after multiple message-passing iterations, degrading performance on deep networks. **Over-squashing** refers to the compression of exponentially growing receptive fields into fixed-size node representations, limiting the model's ability to capture long-range dependencies.

1.1.3 Dataset Distillation: Principles and Progress

Dataset distillation, also known as dataset condensation, represents a data-centric approach to address the computational challenges of training deep neural networks on large datasets. The fundamental idea is to synthesize a small, representative subset of the original dataset that preserves the essential information needed for model training, thereby reducing computational requirements while maintaining performance.

Formally, given a large dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ with N samples, dataset condensation aims to create a synthetic dataset $\mathcal{S} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{M}$ with $M \ll N$ samples, such that models trained on $\mathcal{S}$ achieve comparable performance to those trained on $\mathcal{D}$. This is typically formulated as a bi-level optimization problem:

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathcal{L}_{\text{outer}}(\theta^*(\mathcal{S}), \mathcal{D}_{\text{val}}) \tag{1.2}$$

where $\theta^*(\mathcal{S}) = \arg\min_\theta \mathcal{L}_{\text{inner}}(\theta, \mathcal{S})$ represents the optimal model parameters obtained by training on the synthetic dataset $\mathcal{S}$.

Dataset condensation has achieved remarkable success in the image domain. For example, **DC** [124] synthesizes images by matching gradients between models trained on real and synthetic data. **DM** [123] employs distribution matching to ensure synthetic data follows the same distribution as real data. These methods have demonstrated that as few as 10 synthetic images per class can achieve 90%+ performance compared to training on the full dataset.

The application of dataset condensation to graph-structured data presents unique

challenges and opportunities. Unlike images, graphs contain both node features and structural information, requiring condensation methods to preserve both aspects. **Graph condensation** aims to create a small synthetic graph that maintains the essential structural and feature information of the original graph, enabling efficient training of GNNs.

Recent work has explored various approaches to graph condensation. **GCond** [40] formulates graph condensation as a bi-level optimization problem, where the inner loop trains a GNN on the synthetic graph, and the outer loop updates the synthetic graph to match the performance of models trained on the original graph. **DOSCOND** [39] extends this approach to handle heterogeneous graphs and multiple GNN architectures. **SFGC** [125] introduces a more efficient single-level optimization framework.

However, existing graph condensation methods face significant limitations. The bi-level optimization framework requires extensive GNN training in the inner loop, leading to high computational costs that can exceed those of training on the original dataset. Multiple parameter initializations are often necessary to ensure generalization across different GNN architectures, further increasing computational overhead. The iterative nature of these methods results in slow convergence and unstable training, limiting their practical applicability.

1.2 Motivations and Contributions

The computational challenges associated with training GNNs on large-scale graph data have become increasingly critical as the size and complexity of graph datasets continue to grow exponentially. Traditional approaches that rely on processing entire datasets often face severe limitations in terms of computational resources, training time, and memory requirements. This bottleneck is particularly pronounced in real-world applications where efficiency and scalability are paramount, such as social

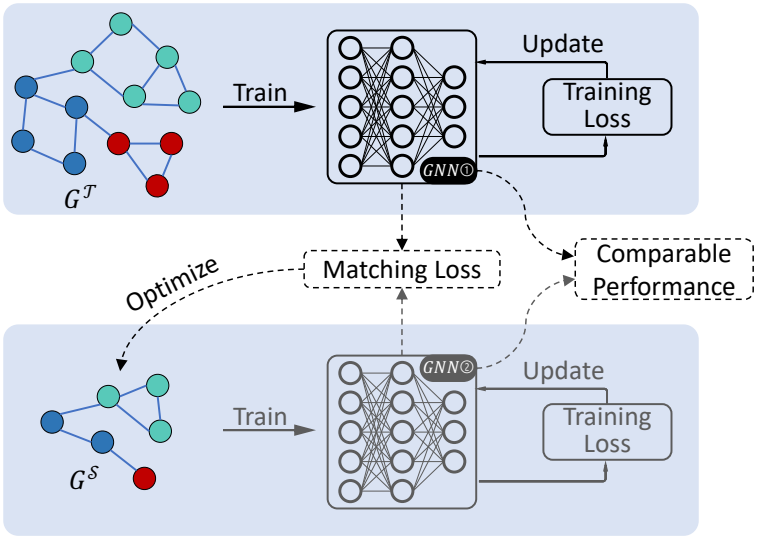


Figure 1.1: Graph condensation aims to condense graph data to a smaller but informative version. In general, two GNN classifiers (i.e., GNN① and GNN②) are trained on $G^{\mathcal{T}}$ and $G^{\mathcal{S}}$ simultaneously. Meanwhile, various matching objectives on two GNNs are conducted to synthesize $G^{\mathcal{S}}$.

network analysis, molecular property prediction, and recommendation systems.

Graph condensation emerges as a promising solution to address these challenges by creating compact, representative subsets of large graph datasets. However, existing graph condensation methods, particularly those based on bi-level optimization, introduce their own computational overhead that can undermine the very efficiency gains they aim to achieve. The iterative nature of these methods, combined with the need for extensive GNN training and multiple parameter initializations, often results in computational costs that are comparable to or even exceed those of training on the original dataset.

This thesis addresses these fundamental limitations through a systematic approach that leverages the computational efficiency of kernel methods while maintaining the expressive power of GNNs. Our research is motivated by three key observations: (1) the need for more efficient graph condensation methods that avoid the computa-

tional overhead of bi-level optimization, (2) the potential of kernel methods to provide closed-form solutions for graph learning tasks, and (3) the opportunity to combine the strengths of both approaches to achieve superior computational efficiency and performance.

1.2.1 A Fast Graph Condensation Framework Based on Kernel Ridge Regression

To tackle the problems of bi-level optimization-based graph condensation, we propose to reformulate the graph condensation as a Kernel Ridge Regression (KRR) task in a closed-form solution [65, 15], instead of training GNNs (in the inner loop) with multiple iterations to learn synthetic graph data via bi-level optimization. However, leveraging KRR for graph condensation faces tremendous challenges. The KRR paradigm, which can be treated as a classification task, largely relies on the design of kernel functions to calculate the similarity matrix among instances. The vanilla kernel function, such as dot product and polynomial kernel, might hinder the expressiveness of modeling complex relationships among nodes. Therefore, the first challenge is how to design an appropriate kernel function for optimizing graph condensation in the KRR paradigm. Moreover, unlike images or text data, graph-structured data lies in non-Euclidean space with complex intrinsic dependencies among nodes. In other words, it is imperative to capture graph topological signals in non-linear kernel space. Thus, the second challenge is how to effectively take advantage of topological structure in graphs to enhance the design of kernel function in KRR for graph condensation.

In this work, we introduce a novel dataset condensation framework for graph-structured data in the node classification task, named **G**raph **C**ondensation with **S**tructure-Based **N**eural **T**angent **K**ernel (**GC-SNTK**). More specifically, a novel graph condensation framework is developed by harnessing the power of the estimated neural kernel into the KRR paradigm, which can be considered as training infinite-width

neural networks through infinite steps of SGD. What's more, to capture the topological structure of graphs, a **S**tructure-based **N**eural **T**angent **K**ernel (**SNTK**) is introduced to execute neighborhood aggregation of nodes for generating the high-quality condensed graph.

1.2.2 A Simplifying Methods for Graph Neural Kernels

While the Graph Neural Tangent Kernel (GNTK) [16] successfully bridges the gap between kernel methods and Graph Neural Networks (GNNs), addressing key challenges such as the difficulty of training deep networks and the limitations of traditional kernel methods, it exhibits certain inherent limitations. Specifically, the GNTK employs a layer-stacking strategy to compute the kernel value, as illustrated in Figure 4.1. In this strategy, each layer begins with an aggregation step, where information from neighboring nodes is collected, followed by a transformation step that applies non-linear operations to the aggregated data [36]. These steps are performed recursively across layers to iteratively update the kernel values. While this recursive stacking mechanism enhances the expressiveness of the kernel by incorporating information from multiple layers, it also leads to an increase in computational complexity, which grows linearly with the number of layers stacked.

Moreover, as the depth of the network increases, the transformations performed in the deeper layers contribute progressively less to the overall performance of the model [104]. This diminishing contribution limits the utility of excessively deep networks. As a result, while the GNTK is a powerful tool, its high computational cost and reduced benefits at greater depths may hinder its scalability and applicability in certain cases.

To address these issues, we investigate the structure of the GNTK method and identify computational redundancies. Specifically, as the number of stacked layers increases, the impact of the non-linear transformations in deeper layers diminishes. Thus, re-

taining only aggregations and a limited number of Neural Tangent Kernel [36] iterations within these layers suffices. This approach reduces computation time while maintaining the kernel function's effectiveness.

Building on this observation, we propose the **S**implified **G**raph Neural **T**angent **K**ernel (**SGTK**) method. This method involves performing K consecutive steps of aggregation operations, followed by a single update of the kernel function. SGTK reduces computational redundancy and enhances the efficiency of kernel function computation. Furthermore, we define an infinitely wide, simple graph neural model, deriving a Gaussian process (GP) [103] kernel function with reduced computational complexity, termed the **S**implified **G**raph **N**eural **K**ernel (**SGNK**). This method calculates the expectations of the post-activation output of the infinitely wide graph model to determine the kernel value. Compared to SGTK, SGNK further streamlines the process and reduces computational complexity while maintaining effectiveness in downstream tasks.

1.2.3 An Efficient Graph Condensation Framework Based on Gaussian Process

Current graph condensation methods are commonly framed as bi-level optimization problems [40, 120], where the inner loop trains the GNN using condensed data, and the outer loop updates the condensed data. While these approaches have achieved notable progress, they face significant computational inefficiency [92] due to extensive iterative GNN training. The nested loops require iterative updates for both GNN parameters and condensed data, resulting in slow convergence. Moreover, to ensure generalization across different GNN initializations, the inner loop often involves thousands of parameter reinitializations, making the computational cost comparable to or even exceeding that of training on the original dataset.

Gaussian Process (GP) [103, 49], a non-parametric method, 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.

However, employing GPs 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 the adjacency matrix, which is non-differentiable and incompatible with gradient-based optimization methods. This limitation hinders the 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 [49, 104]. This approach systematically improves GP predictive performance. For optimizing structural information, we relax the binary adjacency matrix [39] into a differentiable form using concrete random variables [61]. As the temperature approaches zero, the relaxed adjacency matrix converges to a binary form. This relaxation enables gradient-based optimization to refine the graph

structure effectively.

In this work, 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 serve 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, particularly in optimizing the synthetic graph's structure.

1.3 Problem Setting and Theoretical Assumptions

The proposed kernel-based graph condensation frameworks are established under several fundamental assumptions that define the theoretical boundaries of their applicability.

Static Graph Assumption. We assume that the original graph is static during the condensation process. That is, the nodes set, edge structure, and feature matrix remain fixed and are not subject to temporal or streaming changes. This assumption ensures that the structural similarities and kernel relationships computed between the original and synthetic graphs remain well-defined and stable throughout optimization. The current framework does not address dynamic or evolving graphs, where temporal dependencies would require additional modeling beyond the static kernel formulation.

i.i.d. Embedding Assumption. After graph convolution or message passing, the resulting node embeddings are assumed to be independently and identically dis-

tributed (i.i.d.) samples drawn from an underlying embedding distribution. This assumption simplifies the kernel-based analysis and allows us to interpret condensation as a distribution matching problem between embedding spaces. Although true independence may not hold strictly in real graphs due to connectivity patterns, this approximation is a standard practice for theoretical tractability in kernel methods and graph representation analysis.

1.4 Outline of the Thesis

This thesis presents a framework for efficient graph condensation through three interconnected contributions that systematically address the computational challenges in graph machine learning. The remainder of this thesis is organized as follows:

- **Chapter 2: Related Work** provides a comprehensive review of the literature on dataset condensation, graph neural networks, and kernel methods. We examine the evolution of dataset condensation from image domains to graph-structured data, discuss the computational challenges of existing approaches, and establish the theoretical foundation for our kernel-based solutions.
- **Chapter 3: Graph Condensation with Structure-based Neural Tangent Kernel (GC-SNTK)** presents our first contribution, which reformulates graph condensation as a Kernel Ridge Regression task. We introduce the Structure-based Neural Tangent Kernel (SNTK) that captures graph topological information and demonstrate how this approach significantly accelerates graph condensation while maintaining high prediction performance. The chapter includes theoretical analysis, algorithmic details, and extensive experimental validation across multiple datasets.
- **Chapter 4: Simplified Graph Neural Kernels (SGTK and SGNK)** addresses the computational limitations of existing Graph Neural Tangent Kernel

methods. We propose two complementary approaches: the Simplified Graph Neural Tangent Kernel (SGTK) that replaces multi-layer stacking with continuous K-step aggregation, and the Simplified Graph Neural Kernel (SGNK) that models infinitely wide GNNs as Gaussian Processes. Both methods reduce computational complexity while preserving kernel expressiveness.

- **Chapter 5: Graph Condensation via Gaussian Process (GCGP)** presents our most efficient contribution, which leverages the computational advantages of Gaussian Processes to achieve even faster graph condensation. We develop a specialized covariance function that incorporates structural information through local neighborhood aggregation and utilize Concrete random variables to enable gradient-based optimization of discrete graph structures. This chapter demonstrates the synergistic potential of combining kernel methods with graph condensation techniques.
- **Chapter 6: Conclusion and Future Directions** summarizes the key contributions of this thesis, discusses the broader implications of our work for the field of graph machine learning, and outlines promising directions for future research in efficient graph condensation and kernel-based graph learning.

Each chapter builds upon the previous ones, creating a coherent narrative that progresses from understanding the problem landscape to developing increasingly sophisticated and efficient solutions. The three main contributions (GC-SNTK, SGTK/SGNK, and GCGP) are designed to work synergistically, with each addressing different aspects of the computational efficiency challenge in graph condensation.

Chapter 2

Related Work

This chapter reviews the literature that forms the theoretical foundation for our research on efficient graph condensation. We examine five key areas: (1) Graph Neural Networks (GNNs) and their computational challenges, (2) Graph Neural Tangent Kernel (GNTK) methods and their limitations, (3) Gaussian Processes and their connection to neural networks, (4) dataset distillation techniques from image domains to graph-structured data, and (5) current graph condensation research and its computational bottlenecks. This review establishes the motivation for our kernel-based approaches, highlighting the computational inefficiencies in existing methods and the potential of kernel methods to address these challenges while maintaining the expressive power of GNNs.

2.1 Graph Neural Networks

GNNs have undergone significant advancements, addressing diverse aspects of graph learning through various models. For semi-supervised learning, GCNs [43] leverage spectral graph convolutions, while GATs [84] incorporate attention mechanisms to prioritize relevant neighbors. Extensions like R-GCNs [71] handle multi-relational data,

and GraphSAGE [28] introduces sampling-based aggregation for scalable inductive learning. To enhance expressiveness, GINs [111] employ sum aggregation inspired by the Weisfeiler-Lehman test. Simplifying computations, SGC [104] removes non-linear activations and collapses weight matrices, while APPNP [24] integrates personalized PageRank with neural networks for improved performance.

2.2 Graph Neural Tangent Kernel

Neural tangent kernel (NTK) [36] fundamentally transforms our understanding of training dynamics in infinitely wide neural networks by equating them to kernel methods. CNTK [2] extends NTK to convolutional neural networks. GNTK [16] and GPGC [31] are introduced for graph neural networks, demonstrating their efficacy in graph-based tasks. Krishnagopal et al. [46] investigate the training dynamics of large-graph GNNs using GNTKs and graphons. GCNTK [129] shows that for wide GCNs, the output for semi-supervised problems can be described by a simple differential equation. A quantum graph learning model, GraphQNTK [79], leverages quantum parallelism to enhance computational efficiency; however, experiments reveal that its performance lags behind that of GNTK. Jiang et al. [38] employ sketching to accelerate GNTK, but they do not experimentally demonstrate efficiency improvements. Wang et al. [92] integrate structural information with the neural tangent kernel and propose the SNTK, a method designed to address graph condensation problems. Additionally, GCKM [106] employs stacked layers to compute graph kernels, but this approach involves redundant computations, which contribute to increased computational complexity.

2.3 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 [102] first derived the analytical 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.[49] 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.[36] introduced the Neural Tangent Kernel, which offers insights into the training dynamics of infinitely wide networks. Furthermore, Matthews et al. [62] demonstrated the emergence of Gaussian process behavior in such networks, reinforcing the theoretical underpinnings of their behavior.

2.4 Dataset Distillation

Dataset Distillation or Dataset Condensation [96, 124] represents a fundamental paradigm for condensing large-scale datasets into smaller synthetic datasets while ensuring comparable model performance. This approach addresses the computational and storage challenges associated with training deep learning models on massive datasets by creating compact, representative subsets that preserve the essential characteristics of the original data. The seminal work by Wang et al. [96] introduced the bi-level optimization framework DC (Dataset Condensation), which builds on the concept of optimizing image pixels as parameters via gradients [60]. This framework established the core principle of matching model behavior between synthetic and original datasets through iterative optimization, where the synthetic dataset is treated as learnable parameters that are optimized to minimize the performance gap between models trained on synthetic and real data.

Subsequent research has explored various evaluation criteria and optimization strategies for dataset condensation, each targeting different aspects of the learning process:

Gradient matching approaches [124, 122, 51] aim to align gradients between models trained on synthetic and real data. This strategy ensures that the synthetic dataset induces similar learning dynamics as the original dataset, leading to comparable model convergence and performance. **Feature alignment** methods [91] focus on matching intermediate feature representations extracted by neural networks. By aligning the feature distributions at different layers, these approaches preserve the hierarchical representations learned by deep models. **Training trajectory matching** [8] ensures that the learning dynamics are preserved across datasets by matching the entire training trajectory of models. This comprehensive approach captures the temporal evolution of model parameters during training. **Distribution matching** techniques [123] align the statistical distributions of synthetic and real data, ensuring that the condensed dataset maintains the statistical properties of the original dataset.

Beyond the basic bi-level optimization framework, researchers have developed more sophisticated approaches. Deng et al. [13] introduce the learnable address matrix in condensation, which considers the address matrix as part of the condensed dataset, providing a novel perspective on dataset representation. Nguyen et al. [65] introduced a meta-learning approach for DC, drawing on infinite-width neural network theory [16, 36, 2, 53]. This approach leverages the theoretical insights from neural tangent kernels to design more effective condensation strategies.

2.5 Graph Condensation

Extending the concept of dataset condensation to graph-structured data introduces unique challenges due to the non-Euclidean nature of graphs and the need to preserve both node features and structural information. Graph condensation encompasses two

primary task categories, each addressing different aspects of graph data compression.

Node-level condensation focuses on condensing a large graph into a synthetic one with fewer nodes while preserving the essential structural and feature information. This approach is particularly relevant for scenarios where the goal is to reduce computational overhead while maintaining the ability to train effective GNNs on large-scale graphs. **Graph-level condensation** aims to condense numerous graphs into a smaller, synthetic set containing only a limited number of representative graphs. This task is crucial for applications involving multiple graph instances, such as molecular property prediction and social network analysis, where the goal is to create a compact dataset that preserves the discriminative characteristics of the original graph collection.

The development of node-level graph condensation began with Jin et al. [40], who were the first to adapt dataset condensation to the graph domain by proposing a bi-level graph condensation method. This pioneering work established the fundamental framework for graph condensation, demonstrating the feasibility of creating synthetic graphs that preserve the training dynamics of the original graph. Recent advancements have introduced diverse methodologies to enhance the effectiveness and efficiency of node-level condensation: **Trajectory-based approaches** have gained prominence, with Zheng et al. [125] proposing a structure-free method that condenses a graph into node embeddings by employing training trajectory matching. Similarly, Liu et al. [55] advanced node-level graph condensation by utilizing receptive field distribution matching, which captures the propagation patterns of information through the graph structure. **Efficiency-focused methods** have emerged to address the computational challenges of traditional bi-level optimization. Xiao et al. [110] propose the use of a pre-trained Simplified Graph Convolution model, achieving up to a tenfold improvement in computational efficiency while preserving performance. DisCo [109], a GNN-free graph condensation framework, decouples node and edge condensation into two independent stages, facilitating computational acceleration while achieving

accuracy comparable to GNN-based methods. **Kernel-based approaches** leverage the computational efficiency of kernel methods. GC-SNTK [92] utilizes the Kernel Ridge Regression framework with a structure-based neural tangent kernel to enhance condensation efficiency, avoiding the iterative training required by traditional bi-level optimization. **Training-free frameworks** represent a recent innovation, with Gao et al. [23] proposing a framework that simplifies graph condensation into a clustering-based class-to-node partition problem, eliminating the need for extensive training procedures. **Lossless methods** aim to preserve all essential information during condensation. Zhang et al. [120] introduce a lossless node-level graph condensation method grounded in trajectory matching, ensuring that no critical information is lost during the condensation process.

Graph-level condensation addresses the challenge of condensing multiple graph instances into a representative subset. This task is particularly relevant for applications involving graph classification, where the goal is to create a synthetic dataset that preserves the discriminative characteristics of the original graph collection. **Probabilistic modeling approaches** have been explored for graph-level condensation. DosCond [39] employs a probabilistic approach to model the discrete structure of graphs, enabling the generation of synthetic graphs that capture the statistical properties of the original dataset. **Kernel-based techniques** have also been applied to graph-level condensation. The KiDD method [112] utilizes the Kernel Ridge Regression framework to condense entire graph-level datasets effectively, demonstrating the versatility of kernel methods in graph condensation tasks.

Despite significant progress, graph condensation faces several ongoing challenges. The computational complexity of bi-level optimization remains a primary concern, particularly for large-scale graphs. The trade-off between condensation efficiency and information preservation requires careful consideration. Additionally, the generalization of condensed graphs across different GNN architectures and tasks presents an important research direction.

Future research directions include the development of more efficient optimization strategies, the exploration of novel evaluation metrics that better capture graph-specific properties, and the investigation of adaptive condensation methods that can dynamically adjust the condensation ratio based on the complexity of the underlying graph structure.

2.6 Extensions and Relations to Other Learning Paradigms

The proposed kernel-based graph condensation frameworks not only mitigate the computational bottlenecks of existing methods but also enable integrations with other advanced paradigms, such as few-shot learning and knowledge distillation. These connections extend graph condensation beyond data compression and scalability, supporting generalization under limited data and facilitating efficient model transfer.

2.6.1 Integration with Few-Shot Learning

Few-shot learning (FSL) aims to generalize from very limited data by leveraging transferable representations [75, 22, 87]. Our kernel-based condensation aligns naturally with this goal, as both emphasize information efficiency and structural preservation.

- **Condensed Graphs as Meta-Tasks.** Graphs condensed by GC-SNTK or GCGP can serve as diverse yet compact meta-training tasks, providing rich topology and feature patterns for rapid adaptation in meta-learning frameworks.
- **Low-Resource Graph Learning.** In few-label scenarios, condensed graphs can amplify learning signals by capturing relational information from unlabeled graphs, enhancing label efficiency when combined with transductive or semi-supervised settings [126, 93].

Overall, graph condensation can act as task-level pretraining for few-shot graph learning, inspiring an emerging direction of few-shot graph condensation, where synthetic graphs serve as compact priors for fast adaptation across domains.

2.6.2 Connections to Knowledge Distillation

Knowledge distillation (KD) [30] transfers knowledge from a large teacher model to a smaller student. While KD compresses models in the parameter space, graph condensation compresses data in the data space. The two can be unified within a data-model co-distillation framework:

- **Data-to-Model Transfer.** Synthetic graphs implicitly distill structural and semantic knowledge from the full dataset, enabling efficient student training with minimal data.
- **Model-to-Data Guidance.** Teacher predictions or representations can guide the condensation process, ensuring that synthetic graphs preserve key decision and relational cues, forming a teacher-student feedback loop.

Together, knowledge distillation and graph condensation form complementary compression paradigms—one in model space, the other in data space. Their integration enables graph co-distillation, achieving multi-level efficiency and generalization in graph learning.

Chapter 3

Kernel-Based Graph Condensation Framework with Structure-Aware Neural Tangent Kernels

This chapter presents our first major contribution to efficient graph condensation, introducing the Graph Condensation with Structure-based Neural Tangent Kernel (GC-SNTK) framework. Building upon the computational bottlenecks identified in Chapter 2, we reformulate the bi-level optimization problem as a Kernel Ridge Regression (KRR) task, eliminating iterative GNN training. The GC-SNTK framework establishes the theoretical foundation for kernel-based graph condensation, serving as the cornerstone for subsequent developments in Chapters 4 and 5. Through extensive experiments, we demonstrate remarkable efficiency improvements while maintaining competitive performance.

3.1 Introduction

The prevalence of internet technology has accumulated a large amount of graph-structured data, which is widely used in Web applications. For example, social networks [118, 5, 19, 20, 18], and semantic networks [127, 52], can be naturally represented as graphs consisting of nodes and edges. Due to the extensive application of graph-structured data, Graph Neural Networks (GNNs) [117, 58, 21], as one of the advanced Deep Neural Networks (DNNs), have gained significant attention for the performance of various graph mining tasks during the past few years. Notably, the remarkable achievement of most existing GNNs largely relies on large-scale datasets [24, 11]. Despite being effective, training GNNs on such large-scale datasets still presents some difficulties, as it usually requires enormous computational resources (e.g., power, memory storage, GPU runtime, etc.) caused by thousands of training iterations, hyper-parameters optimization, and neural architecture search [107, 40].

A data-centric approach to tackle these challenges is to condense the original large-scale graph datasets into synthesized smaller yet information-rich versions [96]. As one of the most advanced paradigms, Dataset Condensation (DC), also known as dataset distillation [124], has achieved remarkable performance to obtain a small-scale synthetic version of the full dataset while preserving models' prediction performance in the image domain. For instance, 10 images distilled from the whole MNIST dataset [47] with 60,000 images are sufficient to achieve a 94% accuracy, reaching 95% performance of the full dataset [96]. Recently, early efforts explore the potential of dataset condensation on graph-structured data [40, 39]. The vanilla graph condensation methods [40, 39] can be formulated as a bi-level optimization problem, so as to condense the entire graph into a small graph of a few nodes with corresponding features via various matching objectives between two GNNs models. More specifically, as illustrated in the outer synthetic graph optimization (i.e., outer loop) heavily relies on the inner GNN model trained on the synthetic graph (i.e., inner loop) [40, 39].

Despite the aforementioned success, most existing bi-level optimization-based graph condensation methods suffer from intrinsic limitations such as unstable training [67, 63], and massive computation costs [86], leading to an inferior performance on synthesized data generations. Worse still, ensuring condensed data's generalization on various GNN architectures requires multiple parameter initialization [39, 40], leading to complex three nested loops in algorithmic optimization and substantial time consumption.

To tackle the problems, in this paper, we propose to reformulate the graph condensation as a Kernel Ridge Regression (KRR) task in a closed-form solution [65, 15], instead of training GNNs (in the inner loop) with multiple iterations to learn synthetic graph data via bi-level optimization. However, leveraging KRR for graph condensation faces tremendous challenges. The KRR paradigm, which can be treated as a classification task, largely relies on the design of kernel functions to calculate the similarity matrix among instances. The vanilla kernel function, such as dot product and polynomial kernel, might hinder the expressiveness of modeling complex relationships among nodes. Therefore, the first challenge is how to design an appropriate kernel function for optimizing graph condensation in the KRR paradigm. Moreover, unlike images or text data, graph-structured data lies in non-Euclidean space with complex intrinsic dependencies among nodes. In other words, it is imperative to capture graph topological signals in non-linear kernel space. Thus, the second challenge is how to effectively take advantage of topological structure in graphs to enhance the design of kernel function in KRR for graph condensation.

In this paper, we introduce a novel dataset condensation framework for graph-structured data in the node classification task, named **G**raph **C**ondensation with **S**tructure-Based **N**eural **T**angent **K**ernel (**GC-SNTK**). More specifically, a novel graph condensation framework is developed by harnessing the power of the estimated neural kernel into the KRR paradigm, which can be considered as training infinite-width neural networks through infinite steps of SGD. What's more, to capture the topo-

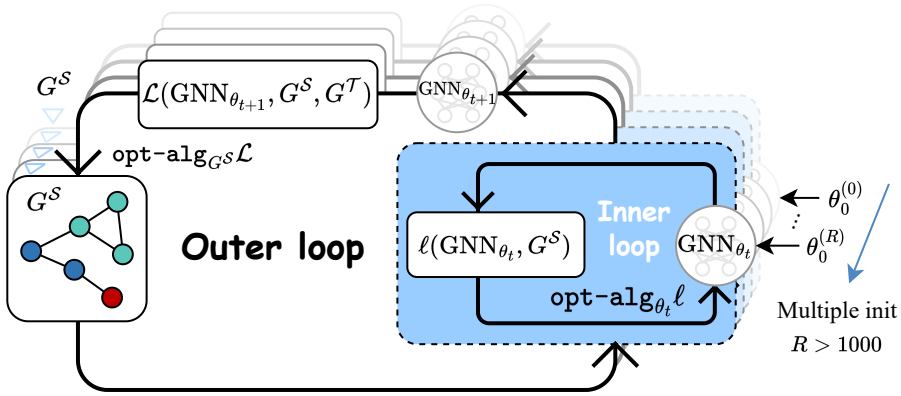
logical structure of graphs, a **S**tructure-based **N**eural **T**angent **K**ernel (**SNTK**) is introduced to execute neighborhood aggregation of nodes for generating the high-quality condensed graph.

3.2 Methodology

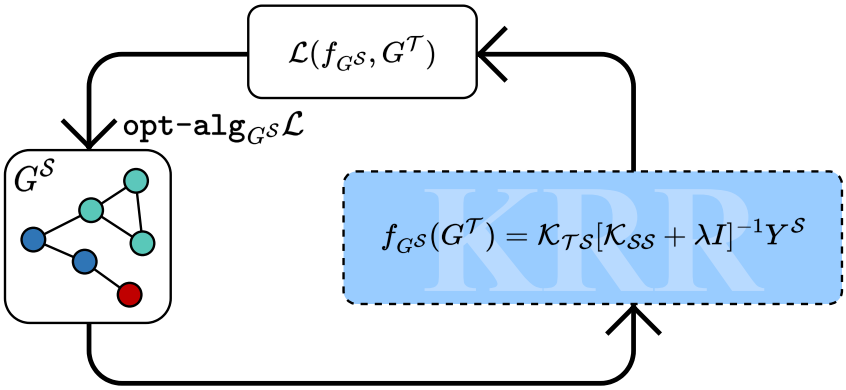
In this section, we start by introducing the bi-level graph condensation model. Next, we provide comprehensive details on the proposed fast graph condensation framework (as shown in Figure 3.1b), and illustrate a structure-based kernel method specifically designed for graph data. Finally, a theoretical analysis of the computational complexity is presented, proving that the proposed GC-SNTK is more efficient compared to previous SOTA methods.

3.2.1 Notations and Definitions

In general, a graph can be represented as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ is the set of $|\mathcal{V}|$ nodes and $\mathcal{E}$ is the edge set. We use $X \in \mathbb{R}^{|\mathcal{V}| \times d}$ to denote the node features matrix, where d is the dimensional size of node features. The structural information of the graph can be represented as an adjacency matrix $A \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, where $A_{ij} = 1$ indicates that there is a connection between node v_i and v_j, and 0 otherwise. Given a target graph dataset $G^{\mathcal{T}} = \{X^{\mathcal{T}}, A^{\mathcal{T}}, Y^{\mathcal{T}}\}$ with N nodes, where $Y^{\mathcal{T}}$ is nodes' labels. The goal of graph condensation is to condense $G^{\mathcal{T}}$ into a graph $G^{\mathcal{S}} = \{X^{\mathcal{S}}, A^{\mathcal{S}}, Y^{\mathcal{S}}\}$ with a significantly smaller nodes number M ($M \ll N$), while the model trained on $G^{\mathcal{S}}$ can achieve comparable performance to that trained on the much larger target graph dataset $G^{\mathcal{T}}$ (as shown in Figure 1.1).



(a) Bi-level graph condensation optimization



(b) Our proposed GC-SNTK

Figure 3.1: (a) Bi-level graph condensation optimization and (b) the proposed GC-SNTK. $G^{\mathcal{T}}$ and $G^{\mathcal{S}}$ denote the target and condensed graph data, respectively. GNN_θ is a graph neural network model with parameter θ. $\mathcal{L}$ and ℓ are the losses of the outer and inner loops, respectively. `opt-alg` refers to the optimization algorithm. The bi-level model entails an inner GNN training loop, an outer $G^{\mathcal{S}}$ optimization loop, and R-time initialization. In contrast, the proposed GC-SNTK only has a single $G^{\mathcal{S}}$ optimization loop.

3.2.2 Bi-level Graph Condensation

As shown in Figure 3.1a, existing solutions regards graph condensation as a bi-level optimization problem [39], which can be formulated as follows:

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_{t=0}^{T} \mathcal{L}(\mathrm{GNN}_{\theta_{t+1}}, G^{\mathcal{S}}, G^{\mathcal{T}}) \right] \\ &s.t. \quad \mathrm{GNN}_{\theta_{t+1}} = \texttt{opt-alg}_{\theta_t} \left[\ell(\mathrm{GNN}_{\theta_t}, G^{\mathcal{S}}) \right]. \end{aligned} \tag{3.1}$$

In Equation (3.1), the outer loop is responsible for optimizing the condensed graph data $G^{\mathcal{S}}$ by minimizing the matching loss $\mathcal{L}$. At the same time, the inner loop trains a model GNN_θ on condensed graph data $G^{\mathcal{S}}$ by minimizing the training loss ℓ. Besides, to ensure the robust generalization of the condensed graph data $G^{\mathcal{S}}$ on parameter initialization distribution P_{θ_0}, multiple initialization of the model parameters is required. As a result, the solving algorithm of the bi-level graph condensation actually has three nested loops, leading to huge computational costs in terms of time and GPU resources.

3.2.3 Fast Graph Condensation via Kernel Ridge Regression

To tackle the substantial computational issues, we propose to replace the GNN_θ in Equation (3.1) by the KRR paradigm. More specifically, KRR [65, 15] entails a convex optimization process with a closed-form solution, bypassing the resource-intensive GNN_θ training. Consequently, the bi-level optimization in the general graph condensation framework can be simplified into a more computational efficient single-level paradigm. Moreover, KRR leverages the condensed graph data $G^{\mathcal{S}}$ as the model parameter, eliminating the need for multiple initialization to ensure robust generalization of the condensed graph data. These distinctive characteristics contribute to a significant improvement in the overall efficiency of the graph condensation. Denote $f_{G^{\mathcal{S}}}$ as the KRR model constructed by condensed data $G^{\mathcal{S}}$. Mathematically, the KRR

model can be written as:

$$f_{G^{\mathcal{S}}}(G^{\mathcal{T}}) = \mathcal{K}_{\mathcal{TS}}(\mathcal{K}_{\mathcal{SS}} + \lambda I)^{-1} Y^{\mathcal{S}}, \tag{3.2}$$

where $\mathcal{K}_{\mathcal{TS}} : G^{\mathcal{T}} \times G^{\mathcal{S}} \to \mathbb{R}^{N \times M}$ is a node-level kernel function. $\lambda > 0$ is the hyperparameter on regularization for preventing over-fitting to mislabeled data [32]. Equation (3.2) leverages condensed graph data $G^{\mathcal{S}}$ to construct the KRR model and giving predictions of target graph data $G^{\mathcal{T}}$. To evaluate the predicting performance, the Mean Square Error (MSE) loss is adopted:

$$\mathcal{L}(f_{G^{\mathcal{S}}}, G^{\mathcal{T}}) = \frac{1}{2} ||Y^{\mathcal{T}} - f_{G^{\mathcal{S}}}(G^{\mathcal{T}})||_F^2. \tag{3.3}$$

Then $G^{\mathcal{S}}$ is updated based on optimization algorithm `opt-alg` and the condensation loss $\mathcal{L}$:

$$G^{\mathcal{S}} = \texttt{opt-alg}_{G^{\mathcal{S}}}[\mathcal{L}(f_{G^{\mathcal{S}}}, G^{\mathcal{T}})]. \tag{3.4}$$

Iteratively conducting equations from (3.2) to (3.4) until convergence, we will get the well condensed graph data $G^{\mathcal{S}}$. Finally, the proposed graph condensation framework with KRR can be modeled as the following minimization problem:

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathcal{L}(f_{G^{\mathcal{S}}}, G^{\mathcal{T}}) \\ &s.t.\ f_{G^{\mathcal{S}}} = \mathcal{K}_{\cdot\mathcal{S}}(\mathcal{K}_{\mathcal{SS}} + \lambda I)^{-1} Y^{\mathcal{S}}. \end{aligned} \tag{3.5}$$

The proposed graph condensation framework is illustrated in the Figure 3.1b, where incorporating KRR in graph condensation allows the bi-level optimization architecture streamlined into a single-level one. Compared with GNN_θ in Equation (3.1), on one hand, the convex nature of KRR allows the optimal model to be trained without iteratively training. On the other hand, KRR does not require multiple initialization of model parameters. The two aspects contribute to the noteworthy improvement of the graph condensation efficiency.

3.2.4 Structure-based Neural Tangent Kernel (SNTK)

The role of the kernel function $\mathcal{K}$ in KRR is mapping the data to a higher-dimensional space in order to capture more intricate nonlinear structures present in the data. Different kernel functions are used to capture patterns in different types of data. Therefore, the choice of the kernel function directly affects the performance of KRR predictions and, consequently, the quality of the condensed graph data.

In recent years, Neural Tangent Kernel (NTK), which serves as a bridge between deep neural networks and kernel methods, has obtained extensive attention along with theoretical studies across enormous successful applications [36, 25, 81]. Specifically, NTK approximates behaviors of infinite-width neural networks, demonstrating the tremendous capabilities in modeling highly complex relationships among instances [25]. More importantly, harnessing the power of estimated NTK into KRR can be considered as training infinitely-wide deep neural networks in an infinite SGD training steps [2, 36], leading to efficient training without sacrificing model's expressivity. Given these advantages, the NTK provides a great opportunity to improve graph condensation that is typically based on the bi-level optimization method.

A straightforward approach using NTK [36] in graph data is to compute the kernel matrix between nodes solely based on *their features* X. For two nodes v_i and v_j with features x_i and x_j, respectively, denote $\mathbf{\Theta}_{ij}^{(0)} = \mathbf{\Sigma}_{ij}^{(0)} = x_i \cdot x_j$ as initialization. The recursive NTK workflow is given by:

$$\mathbf{\Theta}_{ij}^{(L)} = \mathbf{\Theta}_{ij}^{(L-1)} \dot{\mathbf{\Sigma}}_{ij}^{(L)} + \mathbf{\Sigma}_{ij}^{(L)}, \tag{3.6}$$

in which

$$\mathbf{\Sigma}_{ij}^{(L)} = \alpha \mathbb{E}_{(a,b)\sim\mathcal{N}(0,\Lambda_{ij}^{(L)})}[\sigma(a)\sigma(b)], \tag{3.7}$$

$$\dot{\mathbf{\Sigma}}_{ij}^{(L)} = \alpha \mathbb{E}_{(a,b)\sim\mathcal{N}(0,\Lambda_{ij}^{(L)})}[\dot{\sigma}(a)\dot{\sigma}(b)], \tag{3.8}$$

$$\mathbf{\Lambda}_{ij}^{(L)} = \begin{bmatrix} \mathbf{\Sigma}_{ii}^{(L-1)} & \mathbf{\Sigma}_{ij}^{(L-1)} \\ \mathbf{\Sigma}_{ji}^{(L-1)} & \mathbf{\Sigma}_{jj}^{(L-1)} \end{bmatrix}, \tag{3.9}$$

where $\dot{\sigma}(a)$ is the derivative with respect to activation $\sigma(a)$. α is the coefficient related to activation σ. $\mathbf{\Theta}_{ij}^{(L)}$ is the kernel value after L iterations.

However, calculating the kernel value only based on node features might easily lead to low-quality condensed graphs. Because the structural information in graphs can provide crucial insights into the relationships, dependencies, and interactions among nodes. By disregarding structural information, important contextual information and patterns in the graphs may be overlooked. Hence, considering the structural information in NTK for graph condensation is essential.

To tackle this issue, we introduce a novel kernel method, namely Structure-based Neural Tangent Kernel (SNTK), to capture the structure of graphs by connecting local neighborhood aggregation and the NTK method. Based on the homophily assumption [57], SNTK involves integrating information from neighboring nodes into enhancing node representation learning, aiming to capture nodes' context and relationships. More formally, the node neighborhood aggregation of SNTK can be defined as follows:

$$h_i = c_i \sum_{p \in \mathcal{N}\{i\} \cup \{i\}} x_p, \tag{3.10}$$

where $\mathcal{N}\{i\} \cup \{i\}$ indicates the set consisting of node v_i and its neighbors $\mathcal{N}\{i\}$. To avoid imbalanced information propagation, the aggregation coefficient is set to $c_i = (\| \sum_{p \in \mathcal{N}\{i\} \cup \{i\}} x_p \|_2)^{-1}$. With local neighborhood aggregation, the SNTK kernel can be formulated as:

$$\hat{\mathbf{\Sigma}}_{ij}^{(0)} = h_i \cdot h_j = c_i c_j \sum_{p \in \mathcal{N}\{i\} \cup \{i\}} \sum_{q \in \mathcal{N}\{j\} \cup \{j\}} \mathbf{\Sigma}_{pq}^{(0)}, \tag{3.11}$$

$$\hat{\mathbf{\Theta}}_{ij}^{(0)} = h_i \cdot h_j = c_i c_j \sum_{p \in \mathcal{N}\{i\} \cup \{i\}} \sum_{q \in \mathcal{N}\{j\} \cup \{j\}} \mathbf{\Theta}_{pq}^{(0)}, \tag{3.12}$$

$$\hat{\mathbf{\Theta}}_{ij}^{(L)} = \hat{\mathbf{\Theta}}_{ij}^{(L-1)} \dot{\mathbf{\Sigma}}_{ij}^{(L)} + \hat{\mathbf{\Sigma}}_{ij}^{(L)}. \tag{3.13}$$

Real-world scenarios often necessitate K ($K > 1$) rounds of neighborhood aggregation to capture information from nodes' K-hop neighbors. Therefore, the aggregation

followed by L iterations of kernel matrix will be recursively iterated K times.

3.2.5 Matrix Form of Structure-based Neural Tangent Kernel (SNTK)

To enable efficient acceleration computations on GPUs, we formulate the SNTK computational process in the matrix format. Given two different graphs $G = \{X, A, Y\}$ and $G' = \{X', A', Y'\}$, the initialization of SNTK is set as: $\mathbf{\Theta}^{(0)} = \mathbf{\Sigma}^{(0)} = X(X')^\top$. Denote the matrix form aggregation as $\hat{\mathbf{\Sigma}} = Aggr(\mathbf{\Sigma})$, the matrix form for Equation (3.11) and (3.12) is given by:

$$\hat{\mathbf{\Sigma}}^{(0)} = \hat{A}(C \odot \mathbf{\Sigma}^{(0)})(\hat{A}')^\top, \tag{3.14}$$

$$\hat{\mathbf{\Theta}}^{(0)} = \hat{A}(C \odot \mathbf{\Theta}^{(0)})(\hat{A}')^\top, \tag{3.15}$$

where $\hat{A}$ and $\hat{A}'$ are the self-looped adjacency matrix of graphs G and G'. C is the aggregation coefficient matrix, where $C_{ij} = c_i c_j$. $(\cdot)^\top$ indicates the transpose operation, $\odot$ is the element-wise product. Then according to [2], the matrix form of the recursive iteration in Equation (3.13) is:

$$\dot{\mathbf{\Sigma}}^{(L)} = \frac{1}{2\pi}\left[\pi - arccos(\hat{\mathbf{\Sigma}}^{(L-1)})\right], \tag{3.16}$$

$$\hat{\mathbf{\Sigma}}^{(L)} = \frac{1}{2\pi}\left[\pi - arccos(\hat{\mathbf{\Sigma}}^{(L-1)}) + \sqrt{1 - (\hat{\mathbf{\Sigma}}^{(L-1)})^2}\right], \tag{3.17}$$

$$\hat{\mathbf{\Theta}}^{(L)} = \hat{\mathbf{\Theta}}^{(L-1)} \odot \dot{\mathbf{\Sigma}}^{(L)} + \hat{\mathbf{\Sigma}}^{(L)}, \tag{3.18}$$

where $arccos(\hat{\mathbf{\Sigma}}^{(L-1)})$ and $(\hat{\mathbf{\Sigma}}^{(L-1)})^2$ indicate element-wise arc-cosine and squaring operation. To sum up, for K times neighborhood aggregation, where each aggregation followed by L iterations of Equation (3.18), the workflow for the SNTK is illustrated in **Algorithm 1**.

Finally, the algorithm for proposed GC-SNTK, which incorporates KRR paradigm and SNTK, is summarised in **Algorithm 2**.

Algorithm 1 Structure-based Neural Tangent Kernel (SNTK)

Input: Graphs $G = \{X, A, Y\}$, $G' = \{X', A', Y'\}$.
Output: Kernel matrix $\hat{\mathbf{\Theta}}^{(L)}_{(K)}$.
Initialize: $\mathbf{\Theta}^{(0)}_{(0)} = \mathbf{\Sigma}^{(0)}_{(0)} = X(X')^\top$, K, L.
for $k = 1, 2, ..., K$ **do**
    if $k = 1$ **then**
        $\hat{\mathbf{\Sigma}}^{(0)}_{(k)} = Aggr(\mathbf{\Sigma}^{(0)}_{(0)})$, $\hat{\mathbf{\Theta}}^{(0)}_{(k)} = Aggr(\mathbf{\Theta}^{(0)}_{(0)})$.
    else
        $\hat{\mathbf{\Sigma}}^{(0)}_{(k)} = Aggr(\hat{\mathbf{\Sigma}}^{(L)}_{(k-1)})$, $\hat{\mathbf{\Theta}}^{(0)}_{(k)} = Aggr(\hat{\mathbf{\Theta}}^{(L)}_{(k-1)})$.
    end if
    for $l = 1, 2, ..., L$ **do**
        Update $\dot{\mathbf{\Sigma}}^{(l)}_{(k)} \leftarrow \hat{\mathbf{\Sigma}}^{(l-1)}_{(k)}$; $\hat{\mathbf{\Sigma}}^{(l)}_{(k)} \leftarrow \hat{\mathbf{\Sigma}}^{(l-1)}_{(k)}$; $\hat{\mathbf{\Theta}}^{(l)}_{(k)} \leftarrow \hat{\mathbf{\Theta}}^{(l-1)}_{(k)}$, $\dot{\mathbf{\Sigma}}^{(l)}_{(k)}$, $\hat{\mathbf{\Sigma}}^{(l)}_{(k)}$ by Equation (3.16) - (3.18).
    end for
end for

Algorithm 2 GC-SNTK

1: **Input:** Target graph data $G^{\mathcal{T}}$ with N nodes.

2: **Output:** Condensed graph data $G^{\mathcal{S}}$ with $M(M \ll N)$ nodes.

3: **Initialize:** Random initialize $G^{\mathcal{S}}$.

4: **while** not converge **do**

5: Calculate $\mathcal{K}_{\mathcal{TS}}$ and $\mathcal{K}_{\mathcal{SS}}$ by **Algorithm** 1.

6: Calculate loss function $\mathcal{L}(f_{G^{\mathcal{S}}}, G^{\mathcal{T}})$ by Equation (3.3):

$$\mathcal{L}(f_{G^{\mathcal{S}}}, G^{\mathcal{T}}) = \frac{1}{2}||Y^{\mathcal{T}} - f_{G^{\mathcal{S}}}(G^{\mathcal{T}})||_F^2.$$

7: Update $G^{\mathcal{S}}$ by Equation (3.4):

$$G^{\mathcal{S}} = \texttt{opt-alg}_{G^{\mathcal{S}}}\left[\mathcal{L}(f_{G^{\mathcal{S}}}, G^{\mathcal{T}})\right].$$

8: **end while**

3.2.6 Computational Complexity Analysis

To theoretically demonstrate that the computational complexity of our proposed GC-SNTK method is lower than that of bi-level method, i.e., GCond, we conducte a comprehensive computational complexity analysis in this part. The notations used in this part are as follows: N and M represents the number of nodes in the original dataset and the condensed data, d stands for the node feature dimensionality, w signifies the GCN hidden layer width, and k represents the average number of neighbors per node (in the GCond method, $k = M$ since the adjacency relationship between condensed nodes is constructed as a fully connected weighted graph). Additionally, t_{in} and t_{out} denote the number of iterations in the inner loop and outer loop, respectively, and R corresponds to the number of training epochs, which is the initialization number of model parameters in GCond.

The computational complexity of GCond method during the inner loop of GCN training is $O(t_{in}(M^2d + Mdw))$. The outer loop of GCond consists of two parts: 1) op-

timizing the node features $X^{\mathcal{S}}$ or optimizing the MLP model used to update $A^{\mathcal{S}}$; 2) updating $A^{\mathcal{S}}$ according to the updated node features $X^{\mathcal{S}}$. Let the number of iterations for optimizing node features $X^{\mathcal{S}}$ be t_X and the number of iterations for optimizing MLP be t_A ($t_X + t_A = t_{out}$). Then the computational complexity of the outer loop is: $O(t_X(Nkd + M^2dw) + t_A(M^2dw))$. Therefore, the total computational complexity of the **GCond** method is $O(Rt_{out}t_{in}(M^2d + Mdw) + Rt_X(Nkd) + Rt_{out}(M^2dw))$.

On the other hand, the computational complexity of proposed GC-SNTK mainly consists of two parts: kernel matrix calculation and KRR model construction. For the former, in practical experiments, parameters K and L are usually set to be relatively small (e.g., in the ogbn-arxiv dataset, $K = L = 1$). So, these parameters linearly affect the computational complexity and that is $O(MNk^2 + MN)$. For the latter, it is $O(NM^2)$. Therefore, the total computational complexity of the **GC-SNTK** method is $O(RMNk^2 + RNM^2)$. Although the time complexity of both GCond and GC-SNTK are linearly increases with the target graph nodes number N, GC-SNTK proves advantageous due to its single loop and rapid convergence, leading to a faster execution speed compared to GCond.

3.3 Experiment

In this section, a performance comparison is conducted for node classification models trained on condensed data to evaluate the expressive ability of the condensed graph data. Subsequently, we conduct experiments on extremely small condensation sizes to measure the effectiveness of the condensation methods. Additionally, the efficiency of various condensation methods is assessed. The experimental results serve to validate the conclusions drawn in the computational analysis in Section 3.2.6. Moreover, an examination of the generalization capabilities of the condensed data is presented, along with an ablation study of the kernel method. Lastly, a sensitivity analysis of the parameters is conducted. All the experiments are conducted on an NVIDIA RTX

3090 GPU.

3.3.1 Experimental Settings

Datasets. Four different scale benchmark datasets in the graph domain are adopted, including Cora, Pubmed [114], Ogbn-arxiv [34], and Flickr [115]. These datasets vary in size, ranging from thousands of nodes to hundreds of thousands of nodes. Additionally, these datasets are categorized into two different settings: transductive (Cora, Pubmed, and Ogbn-arxiv) and inductive (Flickr). The details of the datasets are shown in Table 3.1. To ensure the transductive setting during experiments, we apply the graph convolution on the entire graph data of Cora, Pubmed, and Ogbn-arxiv datasets to facilitate information propagation between different nodes. Graph convolution can be modeled as $\tilde{X} = \tilde{A}X$, where X is the node feature matrix. $\tilde{A} = \tilde{D}^{-\frac{1}{2}}\hat{A}\tilde{D}^{-\frac{1}{2}}$, $\hat{A} = A+I$, A is the adjacency matrix. $\tilde{D} = diag(\sum_j \hat{A}_{1j}, \sum_j \hat{A}_{2j}, ..., \sum_j \hat{A}_{nj})$.

Table 3.1: Details of the datasets.

Dataset		Transductive			Inductive
		Cora	Pubmed	Ogbn-arxiv	Flickr
#Nodes		2,708	19,717	169,343	89,250
#Edges		5,429	44,338	1,166,243	899,756
#Classes		7	3	40	7
#Features		1,433	500	128	500
Split	Train	140	60	90,941	44,625
	Val.	500	500	29,799	22,312
	Test	1,000	1,000	48,603	22,313

Baselines. To evaluate the effectiveness of GC-SNTK, we compare it against various graph condensation methods serving as baselines, including three core-set methods Random, Herding [99], and K-center [72], as well as a simplified GCond method

named as One-Step [39]. Additionally, we consider two different versions of the previous state-of-the-art (SOTA) graph condensation method, denoted as GCond (X) and GCond (X, A) [40]. The former condense the graph data into the version only including node features, while the latter including both node features and structural information. The GCond framework offers flexibility in utilizing different GNNs for the condensation and testing stages. Hence, there are various possible combinations available for this approach. For the sake of generality, in the experimental section, unless otherwise specified, we default to using the best-performing combination of SGC-GCN for all GCond methods. Specifically, the SGC [104] is employed for condensation, whereas the GCN [43] is utilized for testing.

Parameter Settings. For SNTK, we tune the number of K and L in a range of $\{1, 2, 3, 4, 5\}$. The hyper-parameter λ of KRR is tuned within the range from 10^{-6} to 10^{6}. The values for learning rate are $\{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$. The experimental parameter settings for GC-SNTK is given in Table 3.2. Regarding the GCond and One-Step methods, we follow the settings described in the original papers [40, 39].

Table 3.2: Details of the parameters setting in GC-SNTK.

Dataset	Learning Rate	Ridge (λ)	K	L
Cora	0.01	1	2	2
Pubmed	0.01	0.001	2	2
Ogbn-arxiv	0.001	0.00001	1	1
Flickr	0.001	0.00001	1	1

Table 3.3: Performance evaluation of the condensed data. The last column displays the classification accuracy obtained by the Graph Convolutional Neural Network (GCN) model trained on the complete training set.

Dataset	Ratio (Size)	Random	Herding	K-Center	One-Step	Gcond		**GC-SNTK (Our)**		Full (GCN)
						X	X, A	X	X, A	
Cora	1.30% (35)	63.6±3.7	67.0±1.3	64.0±2.3	80.2±0.73	75.9±1.2	81.2±0.7	**82.2±0.3**	81.7±0.7	81.1±0.5
	2.60% (70)	72.8±1.1	73.4±1.0	73.2±1.2	80.4±1.77	75.7±0.9	81.0±0.6	**82.4±0.5**	81.5±0.7	
	5.20% (140)	76.8±0.1	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	
Pubmed	0.08% (15)	69.5±0.5	73.0±0.7	69.0±0.6	77.7±0.12	59.4±0.7	78.3±0.2	**78.9±0.7**	71.8±6.8	77.1±0.3
	0.15% (30)	73.8±0.8	75.4±0.7	73.7±0.8	77.8±0.17	51.7±0.4	77.1±0.3	**79.3±0.3**	74.0±4.9	
	0.30% (60)	77.9±0.4	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	
Ogbn-arxiv	0.05% (90)	47.1±3.9	52.4±1.8	47.2±3.0	59.2±0.03	61.3±0.5	59.2±1.1	63.9±0.3	**64.4±0.2**	71.4±0.1
	0.25% (454)	57.3±1.1	58.6±1.2	56.8±0.8	60.1±0.75	64.2±0.4	63.2±0.3	**65.5±0.1**	65.1±0.8	
	0.50% (909)	60.0±0.9	60.4±0.8	60.3±0.4	60.0±0.12	63.1±0.5	64.0±0.4	**65.7±0.4**	65.4±0.5	
Flickr	0.10% (44)	41.8±2.0	42.5±1.8	42.0±0.7	45.8±0.47	45.9±0.1	46.5±0.4	46.6±0.3	**46.7±0.1**	47.2±0.1
	0.50% (223)	44.0±0.4	43.9±0.9	43.2±0.1	46.6±0.13	45.0±0.2	**47.1±0.1**	46.7±0.1	46.8±0.1	
	1.00% (446)	44.6±0.2	44.4±0.6	44.1±0.4	45.4±0.31	45.0±0.1	**47.1±0.1**	46.6±0.2	46.5±0.2	

3.3.2 Performance Comparison of Condensed Graph Data

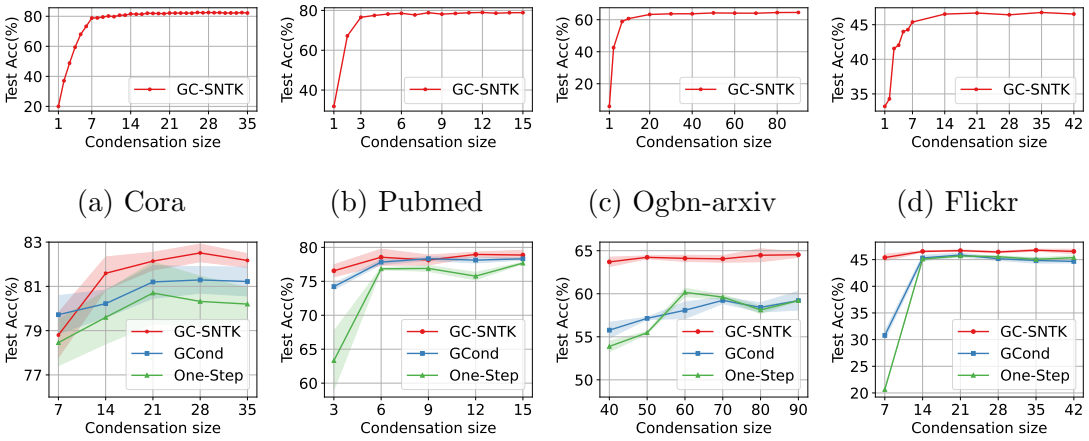
This part aims to assess the representation capability of the condensed graph data in three different condensation scales. We train nodes classification models on the condensed data to predict the labels of the test set. The classification accuracy serves as the measure for evaluating the representation capability of condensed graph data. For each dataset, we opt for three distinct condensation ratios. In the transductive scenario, $Ratio = \frac{M}{N}$. In the inductive scenario, $Ratio = \frac{M}{|training\ set|}$.

As shown in Table 3.3, the proposed GC-SNTK method demonstrates its remarkable graph condensation capability. Even at condensation ratios of 0.1% (Flickr) and 0.05% (Ogbn-arxiv), GC-SNTK achieves **99%** and **89%** of the performance of the GCN model trained on the full training set, respectively. Moreover, GC-SNTK even outperforms the GCN model on Cora and Pubmed datasets. Specifically, with condensation ratios of 1.3% (Cora) and 0.08% (Pubmed), GC-SNTK achieves performance levels of **101%** and **102%** compared to that of the GCN model trained on the full training data.

The results on Cora and Pubmed datasets show that GC-SNTK can improve efficiency, eliminate redundancies, and retain the most representative information in the original data. In other words, the proposed method can reduce datasets scale and enhance the entities' representation capability without scarifying the predictive performance.

3.3.3 Performance with Extremely Small Condensation Size

This part explores the variation in the representational performance of condensed graph data as the condensation scale decreases. We continuously reduce the number of nodes in the condensed data and observe how their performance change. The experimental results are presented in Figure 3.2. Due to the limitations of the GCond



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

(e) Cora (f) Pubmed (g) Ogbn-arxiv (h) Flickr

Figure 3.2: The comparison of node classification accuracy. (a)-(d) illustrate the performance variation of GC-SNTK as the condensation size decreases to a single node. (e)-(f) represent the performance comparison of GC-SNTK, GCond, and One-Step methods at extremely small condensation sizes.

algorithm, it is unable to condense graph data into node counts smaller than the number of node categories. Therefore, when the number of nodes in the condensed data is smaller than the number of classes in the target dataset (Figure 3.2a - 3.2d), only GC-SNTK is involved in the experiments.

The proposed GC-SNTK method maintains strong performance even the scale of synthetic graphs is extremely small. Predictive performance of the model trained on condensed data only significantly drops when the number of nodes in the condensed data is smaller than the number of node categories. Additionally, on the Ogbn-arxiv dataset, our method achieves 63% accuracy (Figure 3.2c) even when the condensed data contains only 20 nodes (with 40 categories).

The experimental results in Figure 3.2e - 3.2h show the performance comparison of GC-SNTK, with GCond and One-Step methods at smaller condensation scales. Overall, GC-SNTK exhibits outstanding performance across the majority of condensation scales on all four datasets. Particularly on the Ogbn-arxiv dataset, GC-SNTK

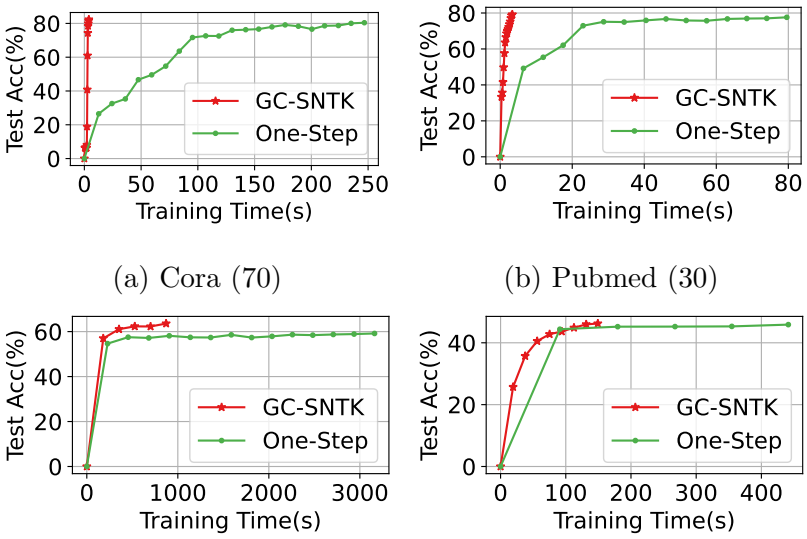
is significantly better than the other two methods, achieving a higher accuracy of **3%**. In contrast, GCond performs slightly worse than GC-SNTK, while the simplified optimization process of the One-Step method leads to the poorest performance.

3.3.4 Condensation Efficiency

This part focuses on evaluating the time efficiency of GC-SNTK and One-Step (faster than GCond) from an empirical perspective. As the optimization of condensed data progresses, the performance of node classification models trained on compressed data also changes over time. Therefore, we evaluate GC-SNTK, in terms of its time efficiency by observing the correlation between model performance and the time spent on optimizing condensed data. This correlation is illustrated by the curves in Figure 3.3, where the y-axis represents model performance and the x-axis represents the time consumption. Each curve stops when it reaches the corresponding performance indicated in Table 3.3.

As shown in Figure 3.3, GC-SNTK exhibits remarkable time efficiency across all four datasets. For instance, GC-SNTK can condense the Cora and Pubmed datasets to 70 and 30 nodes in 4.01 and 3.37 seconds, which is significantly (**61.5** and **23.6** times) faster than One-Step method (Cora: 246.7s, Pubmed: 79.6s), and achieving a higher accuracy performance. Moreover, our method maintains advantages on large datasets like Ogbn-arxiv and Flickr. By condensing these two datasets to 90 and 44 nodes, GC-SNTK (Ogbn-arxiv: 871.5s, Flickr: 163.0s) outperforms the One-Step method (Ogbn-arxiv: 3161.1s, Flickr: 441.3s) by **3.6** and **2.7** times faster, respectively.

Compared to One-Step, a bi-level optimization approach simplified to improve the efficiency of condensation by performing each loop only once, GC-SNTK both ensures its performance of condensed graph data and significantly speeds up the condensation process.



(a) Cora (70)

(b) Pubmed (30)

(c) Ogbn-arxiv (90)

(d) Flickr (44)

Figure 3.3: Condensation efficiency consumption on the four datasets (the number after the name of dataset is the nodes size of the condensed data).

3.3.5 Generalization of Condensed Data

This part investigates the generalization of the condensed data across different graph node classification models. Specifically, we use the condensed data to train various models, including GCN [43, 116, 71], SGC [104, 11], APPNP [24, 58], GraphSAGE (SAGE) [28], and KRR [89, 100, 119] to assess their generalization performance. For the GCond method, we use four different GNN models (GCN, SGC, APPNP, SAGE) in the condensation process and evaluate the condensed graph data on all five models. The experimental results are presented in Table 3.4.

Even though GC-SNTK is a non-neural network model, the data condensed by it still performs well when training neural network models. For instance, the average performance of the prediction models trained on the condensed data by GC-SNTK from the Cora and Pubmed datasets achieve accuracies of 78.0% and 75.7%, respectively. The GCond method can achieve better data condensation quality and generalization

performance when using SGC as the condensation model. In contrast, the performance of the condensed data on the KRR model is poor. As a result, the overall performance of the GCond is inferior to the GC-SNTK method.

Table 3.4: The generalization capacity of the condensed data. We utilise various models to condense (C) graph data and test (T) the performance of models trained on the condensed data.

Dataset	C/T	GCN	SGC	APPNP	SAGE	KRR	Avg.
Cora (70)	**GCN**	70.6	68.7	69.8	60.2	38.9	61.6
	SGC	80.1	79.3	78.5	78.2	73.0	77.8
	APPNP	73.5	73.1	72.1	72.3	63.7	70.9
	SAGE	77.0	77.7	77.1	76.1	23.7	66.3
	GC-SNTK	78.0	76.9	75.7	77.0	82.4	**78.0**
Pubmed (30)	GCN	50.4	50.5	54.8	51.3	31.2	47.6
	SGC	77.1	76.0	77.1	77.1	45.2	70.5
	APPNP	68.0	60.7	77.5	73.7	65.7	69.1
	SAGE	51.9	58.0	69.9	65.3	41.3	57.3
	GC-SNTK	76.1	73.2	75.6	74.1	79.3	**75.7**

3.3.6 Ablation Study

In this section, we choose widely used dot product kernel function, as well as NTK for comparison, to test the impact of different kernel functions on the quality of condensed graph data. Experimental results demonstrate that our proposed structure-based neural tangent kernel method can better capture the information of graph data and improve the quality of condensed graph data.

As shown in Table 3.5, the utilization of SNTK exhibits a more significant impact on the results across all datasets. SNTK surpasses the performance of the other two

kernel functions, thereby providing evidence of its influential role in measuring the similarities among graph nodes and improving the quality of condensed graph data.

Table 3.5: Ablation study of the kernels on different datasets.

Dataset	Kernels		
	Dot Product	NTK	SNTK
Cora (70)	78.9±0.3	80.9±1.4	**82.4±0.5**
Pubmed (30)	77.4±1.3	78.8±0.5	**79.3±0.3**
Ogbn-arxiv (90)	63.8±0.1	63.9±0.1	**64.4±0.2**
Flickr (44)	41.9±0.1	43.1±1.8	**46.6±0.3**

3.4 Conclusion

This paper proposes a novel dataset condensation framework (GC-SNTK) for condensing graph-structured data. Specifically, GC-SNTK transforms the bi-level condensation problem into a single-level optimization task as the KRR paradigm. In addition, a structured-based kernel function (SNTK) is introduced to enhance the quality of the condensed data in KRR. Based on NTK and neighborhood aggregation, SNTK can simultaneously leverage the node features and structural information of graphs to capture the complex dependencies among nodes. The experimental results demonstrate the superiority of our proposed GC-SNTK method in efficiency and efficacy. Furthermore, the proposed GC-SNTK performs promising generalization capability across various GNN architectures on graph-structured data.

Chapter 4

Simplifying Graph Neural Kernels for Efficient Computation

This chapter presents our second major contribution to efficient graph analysis, introducing the Simplified Graph Neural Tangent Kernel (SGTK) and Simplified Graph Neural Kernel (SGNK) frameworks. We address the computational inefficiencies in traditional GNTK methods by eliminating redundant computations in layer-stacking. The SGNK framework bridges theoretical foundations and practical applications, establishing the algorithmic framework for kernel simplification that will be further developed in Chapter 5. Through theoretical analysis and extensive experiments, we demonstrate substantial efficiency improvements while maintaining competitive accuracy.

4.1 Introduction

Graphs are collections of nodes and edges that represents relationships between entities in a structured format, widely used to model complex networks, such as social networks [10, 80], recommender systems [29, 94, 105], molecular analysis [97, 70],

and transportation networks [82, 14]. Recently, the widespread application of graph-structured data has attracted significant attention to Graph Neural Networks (GNNs) and Graph Kernels (GKs) [95, 98, 41].

Typically, GNNs [90, 111] learn node or graph representations through recursive message passing and feature extraction. In this process, information is propagated from a node's neighbors, while linear transformations, combined with non-linear activations, extract node features. By stacking these operations, GNNs effectively capture higher-order information from neighboring nodes. In contrast, GKs embed graph data into a high-dimensional feature space to measure similarities based on structural information and node features [88, 66]. Unlike GNNs, GKs do not require training and are advantageous for small-scale graphs, making them suitable for scenarios with limited computational resources [88]. Despite their effectiveness, each method has distinct limitations. GNNs, as parameterized methods, require extensive training, making them sensitive to initialization and reliant on resource-intensive parameter tuning, which affects efficiency and scalability [128, 107]. In contrast, GKs are non-parametric methods that leverage node features and structural information for similarity measurement. While GKs avoid intensive training, their representation capabilities are less robust than those of GNNs, which may lead to sub-optimal performance [74, 108, 6].

To harness the advantages of both GNNs and GKs, [16] introduced the Graph Neural Tangent Kernel (GNTK), a novel kernel method that effectively bridges the gap between these two approaches. Specifically, the GNTK models the training dynamics of infinite-width GNNs, thereby integrating the powerful representational capabilities of GNNs with the computational efficiency of GKs. By capturing the strengths of both paradigms, the GNTK has demonstrated competitive—and, in some cases, superior—performance compared to GNNs across a variety of graph mining tasks [65, 92].

In spite of its demonstrated effectiveness, the GNTK exhibits certain inherent limitations. Specifically, the GNTK employs a layer-stacking strategy to compute the

kernel value, as illustrated in Figure 4.1. In this strategy, each layer begins with an aggregation step, where information from neighboring nodes is collected, followed by a transformation step that applies non-linear operations to the aggregated data [36]. These steps are performed recursively across layers to iteratively update the kernel values. While this recursive stacking mechanism enhances the expressiveness of the kernel by incorporating information from multiple layers, it also leads to an increase in computational complexity, which grows linearly with the number of layers stacked.

Moreover, as the depth of the network increases, the transformations performed in the deeper layers contribute progressively less to the overall performance of the model [104]. This diminishing contribution limits the utility of excessively deep networks. As a result, while the GNTK is a powerful tool, its high computational cost and reduced benefits at greater depths may hinder its scalability and applicability in certain cases.

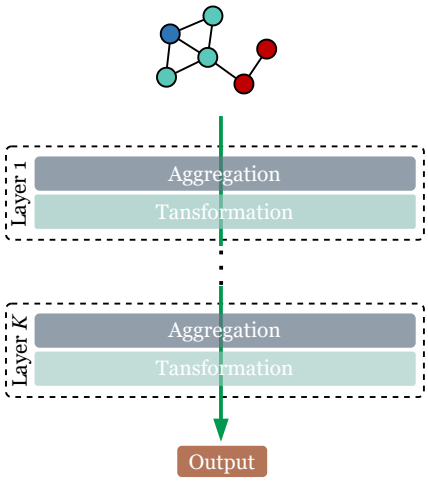


Figure 4.1: The layer stacking structure.

In this paper, we investigate the structure of the GNTK method and identify computational redundancies. Specifically, as the number of stacked layers increases, the impact of the non-linear transformations in deeper layers diminishes. Thus, retaining only aggregations and a limited number of Neural Tangent Kernel [36] iterations

within these layers suffices. This approach reduces computation time while maintaining the kernel function's effectiveness.

Building on this observation, we propose the **S**implified **G**raph Neural **T**angent **K**ernel (**SGTK**) method. This method involves performing K consecutive steps of aggregation operations, followed by a single update of the kernel function. SGTK reduces computational redundancy and enhances the efficiency of kernel function computation. Furthermore, we define an infinitely wide, simple graph neural model, deriving a Gaussian process (GP) [103] kernel function with reduced computational complexity, termed the **S**implified **G**raph **N**eural **K**ernel (**SGNK**). This method calculates the expectations of the post-activation output of the infinitely wide graph model to determine the kernel value. Compared to SGTK, SGNK further streamlines the process and reduces computational complexity while maintaining effectiveness in downstream tasks.

4.2 Preliminary

In this section, we first present the GPs. Then, the dynamical equation of infinitely wide neural networks during training and the NTK [36] are introduced, which converge to a deterministic kernel and remain unchanged throughout the training process.

4.2.1 Gaussian Process

GPs [64] are Bayesian approaches for modeling and inferring unknown functions by leveraging correlations between data points. Specifically, they employ a kernel (or covariance) function to quantify the similarity between data points. GPs allow for predictions on new input data while providing a measure of uncertainty in these predictions based on the observed data. Consequently, GPs can flexibly capture complex patterns in the data without requiring an explicit specification of the functional

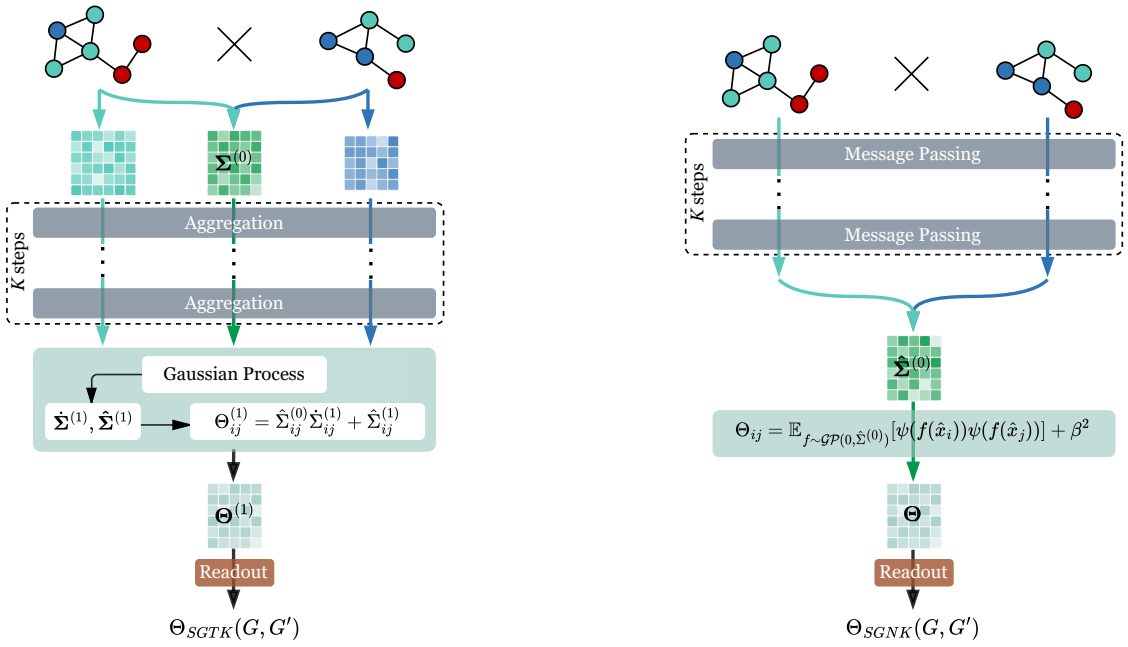


Figure 4.2: The kernel structure includes SGTK and SGNK. SGTK improves computational efficiency by using continuous K-step aggregation. SGNK models the infinite-width simple graph neural network as a GP, requiring only the expectation of the post-activation output to determine kernel values.

form [103].

A straightforward example of a GP can be derived from a Bayesian linear regression model defined as $f(x) = W^\top \phi(x)$, where the weights follow a prior distribution $W \sim \mathcal{N}(0, \sigma_w^2)$. In practical scenarios, we often observe noisy observations $y = f(x) + b$, where b is additive Gaussian noise, distributed as $b \sim \mathcal{N}(0, \sigma_b^2)$. Given the observed data $\mathcal{X}$ and the kernel (or covariance) function $\mathcal{K}$, the joint prior distribution of the training outputs $\mathcal{Y}$ and the test outputs f_t under the prior can be expressed as:

$$\begin{bmatrix} \mathcal{Y} \\ f_t \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathcal{K}(\mathcal{X},\mathcal{X}) + \sigma_b^2 \boldsymbol{I} & \mathcal{K}(\mathcal{X}, x_t) \\ \mathcal{K}(x_t, \mathcal{X}) & \mathcal{K}(x_t, x_t) \end{bmatrix}\right). \tag{4.1}$$

The predictive equation for GP regression is

$$f_t | x_t, \mathcal{X}, \mathcal{Y} \sim \mathcal{N}(\mu, Cov(f_t)), \tag{4.2}$$

where,

$$\mu = \mathcal{K}(x_t, \mathcal{X})[\mathcal{K}(\mathcal{X}, \mathcal{X}) + \sigma_b^2 \boldsymbol{I}]^{-1}\mathcal{Y}, \tag{4.3}$$

$$Cov(f_t) = \mathcal{K}(x_t, x_t) - \mathcal{K}(x_t, \mathcal{X})[\mathcal{K}(\mathcal{X}, \mathcal{X}) + \sigma_b^2 \boldsymbol{I}]^{-1}\mathcal{K}(\mathcal{X}, x_t). \tag{4.4}$$

4.2.2 Neural Tangent Kernel

Neural Tangent Kernel (NTK) is a kernel method derived from infinite-width neural networks, used to understand and analyze the training dynamics of neural networks [36, 32, 50]. Given the regularized mean-squared loss of the infinite-width neural network model f_θ, formulated as follows:

$$\mathcal{L} = \frac{1}{2}||f_\theta(\mathcal{X}) - \mathcal{Y}||_F^2 + \frac{\lambda}{2}||\theta||_F^2, \tag{4.5}$$

where $\lambda > 0$ is the regularization parameter, and $\mathcal{X}$ and $\mathcal{Y}$ represent the training data and the corresponding labels, respectively. Let $\phi(x) = \nabla_\theta f_\theta(x)$. Using the Stochastic Gradient Descent (SGD) algorithm, the optimization of the model parameter θ can be expressed as an ordinary differential dynamic equation when the learning rate is sufficiently small [50]:

$$\begin{aligned} \dot{\theta} &= -\nabla_\theta \mathcal{L} = -\frac{\partial f_\theta(\mathcal{X})}{\partial \theta} \cdot \frac{\partial \mathcal{L}}{\partial f_\theta} \\ &= -\phi(\mathcal{X})(f_\theta(\mathcal{X}) - \mathcal{Y}) - \lambda(\theta) \\ &= -\phi(\mathcal{X})\phi(\mathcal{X})^T\theta + \phi(\mathcal{X})\mathcal{Y} - \lambda\theta, \end{aligned} \tag{4.6}$$

where $\mathcal{K} = \phi(\mathcal{X})^T\phi(\mathcal{X})$ is the Neural Tangent Kernel.

4.3 Methodology

In this section, we begin with a brief review of the GNTK to establish the necessary background. Thereafter, we introduce the proposed SGTK and SGNK, as illustrated

in Figure 4.2. Subsequently, we present the implementation details of the proposed methods, outlining the key steps and considerations. Lastly, we provide a theoretical analysis of computational complexity, showing that the proposed kernel methods are more efficient than previous GNTK approaches.

4.3.1 Notations and Definitions

In general, a graph can be represented as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$ denotes the set of $|\mathcal{V}|$ nodes, and $\mathcal{E}$ represents the edge set. We use $X = [x_1, x_2, \cdots, x_{|\mathcal{V}|}]^\top \in \mathbb{R}^{|\mathcal{V}| \times d}$ to denote the node feature matrix, where d is the dimensional size of the node features. The structural information of the graph is captured by the adjacency matrix $A \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, where $A_{ij} = 1$ indicates a connection between nodes v_i and v_j, and $A_{ij} = 0$ otherwise. In this study, we examine both graph-level and node-level downstream tasks. Given two graphs $G_1 = \{X_1, A_1\}$ and $G_2 = \{X_2, A_2\}$ with n and m nodes, respectively, the graph kernel is defined as $\mathcal{K}(G_1, G_2) : \mathbb{R}^{n \times d} \times \mathbb{R}^{m \times d} \to \mathbb{R}$. Conversely, the node-level kernel is defined as $\mathcal{K}(v_i, v_j) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

4.3.2 Brief Review of Graph Neural Tangent Kernel

The GNTK [16] is a kernel function derived from infinite-width graph neural networks. It serves as a tool for analyzing and understanding the training behavior of GNNs. Specifically, we consider a two-layer GCN [43], which is defined as follows:

$$f_{\text{GCN}}(X, A) = \text{softmax}\{\hat{A}\ \text{ReLU}\{\hat{A}XW^{(0)}\}W^{(1)}\}, \tag{4.7}$$

where

$$\begin{aligned}
&\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \quad \text{is the spectral filter,} \\
&\tilde{A} = A + I, \\
&\tilde{D} = \text{diag}\left(\sum_j \hat{A}_{1j}, \sum_j \hat{A}_{2j}, \ldots, \sum_j \hat{A}_{nj}\right).
\end{aligned}$$

The GNTK follows a structure similar to that of the GCN. It first uses adjacency information to aggregate the covariance matrix of the node features. This matrix is then input into the NTK iterations. Repeating these steps K times yields the kernel value for an infinite-width GCN with K layers, as illustrated on the left in Figure 4.2. For two graphs G and G', the GNTK kernel matrix is computed as follows:

$$\mathbf{\Theta} = \mathrm{NTK}\{\hat{A}_1\ \mathrm{NTK}\{\hat{A}_1\mathbf{\Sigma}^{(0)}\hat{A}_2^\top\}\hat{A}_2^\top\}, \tag{4.8}$$

where $\Sigma_{ij}^{(0)} = c_i c_j x_i \cdot x_j$, and $c_i = (\| \sum_{p\in\mathcal{N}(i)\cup\{i\}} x_p\|_2)^{-1}$ serves as a scaling coefficient that facilitates subsequent NTK computations. $x_i \in G_1$ and $x_j \in G_2$. By conducting a readout operation on the kernel matrix in Equation (4.8), the similarity value between graphs G_1 and G_2 can be derived.

The performance and efficiency of GNTK greatly depend on its architecture, including the number of layers and NTK iterations. Although iterative stacking of layers enables GNTKs to match GCNs of arbitrary depths, minimally contributing NTK iterations in deeper layers introduce computational redundancy. This results in computational overhead, limiting their practical applicability [104, 16].

4.3.3 Simplified Graph Neural Tangent Kernel

GNTK originates from the infinitely wide GCN. To simplify the redundant computations in GNTK, it is essential to first examine the computational complexity inherent in the infinitely wide GCN.

We propose streamlining the GCN structure by retaining K rounds of message passing to enhance the receptive field of node representations while reducing the nonlinear transformations to collapse the weight matrix. Specifically, we consider the infinite-width graph neural network model with K-step message passing.

$$\hat{X} = \hat{A}^K X, \tag{4.9}$$

where $\hat{X} \in \mathbb{R}^{n \times d}$, with n representing the number of samples and d the input dimensionality.

To enhance the receptive field of the node's features, a two-layer neural network is employed, where the width of the hidden layer approaches infinity. Let the infinite-width neural network take $\hat{X}$ as its input. Each layer consists of a biased linear transformation $f^{(l)} : \mathbb{R}^{n \times d^{(l-1)}} \to \mathbb{R}^{n \times d^{(l)}}$, where $d^{(l)}$ is the width of the l-th layer. The parameters are initialized as $W^{(l)} \sim \mathcal{N}(0, \sigma_w^2)$ and $b^{(l)} \sim \mathcal{N}(0, \sigma_b^2)$. Let $\hat{X}^{(0)} = \hat{X}$. The output of the l-th layer of the network can be expressed as follows:

$$f^{(l)}(\hat{X}^{(l)}) = \sigma_w \hat{X}^{(l)} W^{(l)} + \beta b^{(l)}, \tag{4.10}$$

$$\hat{X}^{(l+1)} = \psi(f^{(l)}(\hat{X}^{(l)})), \tag{4.11}$$

where β is a hyperparameter adjusting the effect of the bias on training. Let $\sigma_b^2 = 1$. Under LeCun initialization [48], the variance of the weights $W^{(l)}$ is denoted as $\sigma_w^2 = \frac{1}{d^{(l)}}$. As the width of a neural network approaches infinity, its training dynamics can be represented by an explicit limiting kernel [36], which we term the Simplified Graph Neural Tangent Kernel (SGTK).

Specifically, for two graphs G_1 and G_2, with node features $X_1 = \{x_i\}_{i=1}^n$ and $X_2 = \{x_j\}_{j=1}^m$, respectively, the kernel preprocessing involves the covariance matrix of the enhanced node features:

$$\hat{\Sigma}_{ij}^{(0)} = \frac{1}{d} \hat{x_i} \hat{x_j}^\top + \beta^2. \tag{4.12}$$

Suppose the enhanced node features $\{\hat{x}_i\}_{i=1}^n$ and $\{\hat{x}_j\}_{j=1}^m$ are independently and identically distributed (i.i.d.). In the limit as the layer width tends to infinity, the node-level SGTK kernel value between node v_i of G_1 and node v_j of G_2 is defined by:

$$\Theta_{ij}^{(1)} = \hat{\mathbf{\Sigma}}_{ij}^{(0)} \dot{\mathbf{\Sigma}}_{ij}^{(1)} + \hat{\mathbf{\Sigma}}_{ij}^{(1)}, \tag{4.13}$$

where the intermediate calculation variables of the covariance matrix are defined as:

$$\hat{\mathbf{\Sigma}}_{ij}^{(1)} = \mathbb{E}_{f^{(0)}\sim\mathcal{GP}(0,\hat{\mathbf{\Sigma}}^{(0)})}[\psi(f^{(0)}(\hat{x}_i))\psi(f^{(0)}(\hat{x}_j))] + \beta^2, \tag{4.14}$$

$$\dot{\mathbf{\Sigma}}_{ij}^{(1)} = \mathbb{E}_{f^{(0)}\sim\mathcal{GP}(0,\hat{\mathbf{\Sigma}}^{(0)})}[\dot{\psi}(f^{(0)}(\hat{x}_i))\dot{\psi}(f^{(0)}(\hat{x}_j))], \tag{4.15}$$

where $\dot{\psi}(\cdot)$ is the derivative of the activation function. As illustrated in Figure 4.2, compared to GNTK, which requires iterative stacking of message passing and NTK iterations, the proposed SGTK enhances the receptive field of node features through K-step aggregation in a concentrated manner. This is followed by a single kernel update, effectively collapsing the layers to reduce computational redundancy.

4.3.4 Simplified Graph Neural Kernel

SGTK streamlines the computation process of GNTK by reducing redundant calculations. However, the computation of SGTK relies on the characteristics of infinite-width neural networks and the Gaussian process kernel. The computational efficiency of the kernel method is limited by its reliance on the Gaussian process kernel.

We further propose a kernel method with lower computational complexity than SGTK, termed the Simplified Graph Neural Kernel (SGNK). This approach directly models the defined infinite-width simple graph neural network as a Gaussian process, which is determined by the expectations of the post-activation outputs of the Gaussian process [64, 49]. These expectations can be computed analytically without the need for iteration. Consequently, this direct kernel method further reduces computational complexity. For the node features x_i and x_j, the Gaussian process kernel is defined recursively by the following functions:

$$\Sigma^{(0)}(x_i, x_j) = \frac{1}{d} x_i x_j^\top + \beta^2, \tag{4.16}$$

$$\Sigma^{(l+1)}(x_i, x_j) = \mathbb{E}_{f^{(l)}\sim\mathcal{GP}(0,\mathbf{\Sigma}^{(l)})}\left[\psi(f^{(l)}(x_i))\psi(f^{(l)}(x_j))\right] + \beta^2. \tag{4.17}$$

For an infinitely wide simple graph neural network with K-step message passing,

followed by a two-layer neural network, the node-level Simplified Graph Neural Kernel (SGNK) for the node features x_i and x_j is defined as follows:

$$\Theta_{ij} = \mathbb{E}_{f^{(0)}\sim\mathcal{GP}(0,\hat{\mathbf{\Sigma}}^{(0)})}[\psi(f^{(0)}(\hat{x}_i))\psi(f^{(0)}(\hat{x}_j))] + \beta^2. \tag{4.18}$$

Let the neural network adopt the Error Function as the activation function, defined as $\psi(x) = \frac{2}{\pi}\int_0^x e^{-t^2}dt$, then the SGNK can be calculated analytically [102],

$$\Theta_{ij} = \frac{2}{\pi} sin^{-1} \frac{2\tilde{x}_i^\top \Sigma_w \tilde{x}_j}{\sqrt{(1+2\tilde{x}_i^\top \Sigma_w \tilde{x}_i)(1+2\tilde{x}_j^\top \Sigma_w \tilde{x}_j)}}, \tag{4.19}$$

where $\tilde{x} = (\hat{x}, 1)^\top$ is the augmented vector, and the variance matrix $\Sigma_w = \text{diag}(1, 1, \cdots, \sigma_b^2)$. In matrix form, the calculation process can be expressed as follows:

$$\mathbf{\Theta} = \frac{2}{\pi} sin^{-1} \frac{2\tilde{X}_1 \Sigma_w \tilde{X}_2^\top}{\sqrt{[\mathbf{1} + 2diag(\tilde{X}_1 \Sigma_w \tilde{X}_1^\top)]^\top [\mathbf{1} + 2diag(\tilde{X}_2 \Sigma_w \tilde{X}_2^\top)]}}. \tag{4.20}$$

Given that $\Sigma_w = diag(1, 1, \cdots, \sigma_b^2)$, we have,

$$\mathbf{\Theta} = \frac{2}{\pi} sin^{-1} \frac{2\tilde{X}_1 \tilde{X}_2^\top}{\sqrt{[\mathbf{1} + 2diag(\tilde{X}_1 \tilde{X}_1^\top)]^\top [\mathbf{1} + 2diag(\tilde{X}_2 \tilde{X}_2^\top)]}} \tag{4.21}$$

Finally, for the graph-level tasks, the readout operation yields the kernel value between G and G', which is:

$$\Theta(G, G') = \sum_{i\in G, j\in G'} \Theta_{ij}. \tag{4.22}$$

The detailed execution processes of our proposed SGTK and SGNK are shown in the Algorithm 3 and 4.

4.3.5 Computational Complexity Analysis

To demonstrate the reduced computational complexity of our proposed simplified methods compared to the GNTK, we present a comprehensive computational complexity analysis. The notations used are as follows: n and m represent the number

Algorithm 3 Simplified Graph Neural Tangent Kernel (SGTK)

1: **Input:** Graphs $G = \{X, A\}$, $G' = \{X', A'\}$.

2: **Output:** Node-level kernel matrix $\mathbf{\Theta}^{(1)}$ and graph-level kernel value $\Theta(G, G')$.

3: **Initialize:** $\mathbf{\Sigma}^{(0)} = X(X')^\top$, $\mathbf{\Sigma}_X^{(0)} = X(X)^\top$, $\mathbf{\Sigma}_{X'}^{(0)} = X'(X')^\top$, $\hat{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$, $\hat{A}' = D^{-\frac{1}{2}}(A' + I)D^{-\frac{1}{2}}$, K.

4: K-step aggregation on covariance matrix.

$$\begin{aligned}
\hat{\mathbf{\Sigma}}^{(0)} &= \frac{1}{d}\hat{A}^K \mathbf{\Sigma}^{(0)} (\hat{A'}^K)^\top + \beta^2, \\
\hat{\mathbf{\Sigma}}_X^{(0)} &= \frac{1}{d}\hat{A}^K \mathbf{\Sigma}_X^{(0)} (\hat{A}^K)^\top + \beta^2, \\
\hat{\mathbf{\Sigma}}_{X'}^{(0)} &= \frac{1}{d}\hat{A'}^K \mathbf{\Sigma}_{X'}^{(0)} (\hat{A'}^K)^\top + \beta^2.
\end{aligned}$$

5: Update $\hat{\mathbf{\Sigma}}_X^{(1)}$ and $\hat{\mathbf{\Sigma}}_{X'}^{(1)}$ by Eq. (4.14).

6: Update $\hat{\mathbf{\Sigma}}^{(1)}$ and $\dot{\mathbf{\Sigma}}^{(1)}$ by Eq. (4.14), (4.15), $\hat{\mathbf{\Sigma}}^{(0)}$, $\hat{\mathbf{\Sigma}}_X^{(1)}$, and $\hat{\mathbf{\Sigma}}_{X'}^{(1)}$.

7: Calculate the node-level kernel value by Eq. (4.13),

$$\Theta_{ij}^{(1)} = \hat{\mathbf{\Sigma}}_{ij}^{(0)} \dot{\mathbf{\Sigma}}_{ij}^{(1)} + \hat{\mathbf{\Sigma}}_{ij}^{(1)}.$$

8: Readout, $\Theta(G, G') = \sum_{ij} \Theta_{ij}^{(1)}$.

of nodes in two different graphs, d stands for the node feature dimensionality, k indicates the average degree per node, and K represents the number of message-passing iterations.

In each layer, the computational complexity of GNTK comprises two parts: message passing and NTK iterations. The complexity of message passing is $O(nk^2)$, while the NTK iterations have a complexity of $O(n^2d + nmd + m^2d)$. Consequently, for the GNTK with K layers, the total computational complexity is $O(K(nk^2 + n^2d + m^2d + nmd))$.

In contrast, the proposed SGTK performs K steps of aggregation and eliminates multi-layer stacking, achieving an overall complexity of $O(Knk^2 + n^2d + m^2d + nmd)$.

Algorithm 4 Simplified Graph Neural Kernel (SGNK)

1: **Input:** Graphs $G = \{X, A\}$, $G' = \{X', A'\}$.

2: **Output:** Node-level kernel matrix $\mathbf{\Theta}$ and graph-level kernel value $\Theta(G, G')$.

3: **Initialize:** $\hat{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$, $\hat{A}' = D^{-\frac{1}{2}}(A' + I)D^{-\frac{1}{2}}$, K.

4: K-step message passing on nodes features.

$$\hat{X} = \hat{A}^K X, \quad \hat{X}' = \hat{A}'^K X',$$

5: Calculate the node-level kernel matrix by Eq. (4.21),

$$\mathbf{\Theta} = \frac{2}{\pi} sin^{-1} \frac{2\tilde{X}_1 \tilde{X}_2^\top}{\sqrt{[\mathbf{1} + 2diag(\tilde{X}_1 \tilde{X}_1^\top)]^\top [\mathbf{1} + 2diag(\tilde{X}_2 \tilde{X}_2^\top)]}} \tag{4.23}$$

6: Readout, $\Theta(G, G') = \sum_{ij} \Theta_{ij}$.

Furthermore, the SGNK reduces this complexity to $O(Knk^2 + nmd)$. This indicates that the complexity of our proposed kernels is lower than that of GNTK.

4.4 Experiment

Table 4.1: Graph-level dataset details.

Dataset	# Graphs	Avg. Nodes	Avg. Edges	# Class	# Node feature
IMDB-BINARY	1,000	19.8	96.5	2	-
IMDB-MULTI	1,500	13	65.9	3	-
PTC_MR	344	14.3	14.7	2	-
MUTAG	188	17.9	19.8	2	-

This section evaluates the effectiveness of the proposed kernel methods through extensive experiments on both graph and node classification tasks. We compare the performance and efficiency of our methods with neural network approaches and other graph kernel methods. Our experimental findings demonstrate the robust performance of

Table 4.2: Node-level dataset details.

Dataset	# Nodes	# Edges	# Class	# Node feature
Cora	2,708	5,429	7	1,433
Citeseer	3,327	9,104	6	3,703
Pubmed	19,717	44,338	3	500
Photo	7,650	238,162	8	745
Computers	13,752	491,722	10	767

our approach in classification tasks. Furthermore, they highlight computational efficiency improvements compared to neural networks and GNTK. The experiments were conducted on an Apple M1 Pro CPU with 16GB of RAM.

4.4.1 Experimental Settings

Datasets. Five widely-used node-level benchmark datasets are selected to evaluate the effectiveness of our methods: Cora, CiteSeer, PubMed [114], Photo, and Computers [73]. Additionally, four commonly used graph-level datasets are selected to evaluate the graph similarity modeling capability of the proposed kernel method: IMDB-BINARY, IMDB-MULTI [113], PTC_MR [44] and MUTAG [45]. The details of these datasets are presented in Table 4.2 and Table 4.1. For node classifications, we utilize publicly available splits for the Cora, CiteSeer, and PubMed datasets. For the Computers dataset, the first 20 instances from each category are selected as the training set, while the last 100 instances of each class are chosen as the testing set. For graph datasets lacking explicit node features, node degrees are converted into one-hot encoded representations.

Baselines. To evaluate the effectiveness of the proposed SGTK and SGNK, we compare them against various SOTA graph neural networks and kernel methods. The baseline graph neural networks include GCN [43], GAT [84], SGC [104], GraphSAGE [28], APPNP [24], and GIN [111]. Additionally, we consider graph kernels such

as the WL subtree [74], RetGK [121], GraphQNTK [79], GNTK [16], SNTK [92], and GCKM [106].

Parameter Settings. We tune the hyperparameter K over the range $\{1, 2, 3, 4, 5\}$ for GCKM, SNTK, GNTK, SGTK, and SGNK. For classification using SVM, its hyperparameter is optimized over the range $[10^{-2}, 10^{4}]$. Similarly, for KRR, the regularization parameter λ is tuned over the range $[10^{-2}, 10^{2}]$.

4.4.2 Node Classification Performance Comparison

This subsection explores the application of kernel functions to node classification tasks. Specifically, we apply the proposed kernel methods to node classification datasets, utilizing an SVM classifier to evaluate their performance. For GNN methods, we report the mean and standard deviation of the results from 10 runs and present the classification accuracy of the kernel methods using fixed training and testing set splits.

As shown in Table 4.3, the proposed methods, SGTK and SGNK, consistently demonstrate strong performance across all five node classification datasets. These methods achieve the highest classification accuracy in each dataset, outperforming both traditional kernel methods and GNNs. This consistent superiority underscores the effectiveness of the proposed kernel methods in capturing complex graph structures and node relationships.

The results obtained from the node classification datasets further emphasize the effectiveness of the proposed methods in addressing node classification tasks. Specifically, the capability of SGTK and SGNK to maintain high accuracy across diverse datasets highlights their robustness and generalizability. These findings suggest that the proposed methods are well-suited for graph-based learning tasks, contributing to advances in graph analysis techniques.

Table 4.3: Node classification accuracy (%). The top three performance methods are highlighted in varying shades of green, with darker shades representing superior performance.

Methods	Cora	Citeseer	Pubmed	Photo	Computers
GCN	81.1±0.5	71.7±0.1	77.1±0.3	91.4±0.8	84.2±0.3
GAT	79.5±1.0	68.8±0.2	75.7±0.9	90.1±0.6	83.3±0.7
SGC	79.0±0.3	67.4±0.2	76.9±0.1	91.0±0.5	84.1±0.2
APPNP	79.3±0.2	67.1±1.0	76.0±0.5	91.7±0.3	86.3±0.2
GraphSAGE	79.3±0.4	67.5±0.3	73.9±0.6	90.0±0.1	82.8±0.6
GIN	77.2±1.8	67.4±0.2	78.1±0.1	43.8±5.5	30.0±5.2
GCKM	82.4	72.3	79.8	91.30	84.20
SNTK	80.70	69.20	78.40	91.00	84.70
GNTK	80.50	69.20	77.20	91.00	84.70
SGTK(Our)	82.60	71.10	80.00	91.75	86.50
SGNK(Our)	82.80	72.60	79.90	90.88	85.50

To comprehensively evaluate the classification performance of the proposed kernel methods, experiments were also conducted using Kernel Ridge Regression (KRR) [89], as detailed in Table 4.4. KRR, a widely used technique for regression tasks, ensures reliable performance and demonstrates the compatibility of the proposed methods with various classifiers. Given the set of training nodes $\mathcal{V}$ and their corresponding labels $\mathcal{Y}$, KRR models the optimal predictions for test nodes v_t by leveraging the kernel function to capture relationships between nodes. Furthermore, the proposed methods are compatible with Support Vector Machines (SVMs), showcasing their adaptability across different classification frameworks. The prediction for a test node v_t is computed as follows:

$$f(x_t) = \mathcal{K}(v_t, \mathcal{V})[\mathcal{K}(\mathcal{V}, \mathcal{V}) + \lambda I]^{-1}\mathcal{Y}, \tag{4.24}$$

where the kernel function $\mathcal{K}(v, v') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ quantifies the similarity between nodes v and v', and $\lambda > 0$ is the regularization parameter that controls the trade-off

between fitting the training data and maintaining model generalizability. By incorporating the kernel function $\mathcal{K}$, KRR effectively captures the complex relationships among nodes in the graph, making it a suitable choice for evaluating the proposed kernel methods.

The experimental results, summarizing the performance of the kernel methods under the KRR framework, are presented in Table 4.4. These results provide a comprehensive comparison and highlight the advantages of the proposed methods over existing approaches.

Table 4.4: Kernel method on kernel ridge regression results.

Methods	Cora	Citeseer	Pubmed	Photo	Computers
GCKM	82.50	71.70	79.90	93.25	86.80
SNTK	81.40	70.70	80.00	90.13	84.20
GNTK	81.30	71.10	78.60	89.63	83.80
SGTK(Our)	83.50	72.00	80.00	92.87	87.90
SGNK(Our)	82.60	72.40	80.00	92.12	86.50

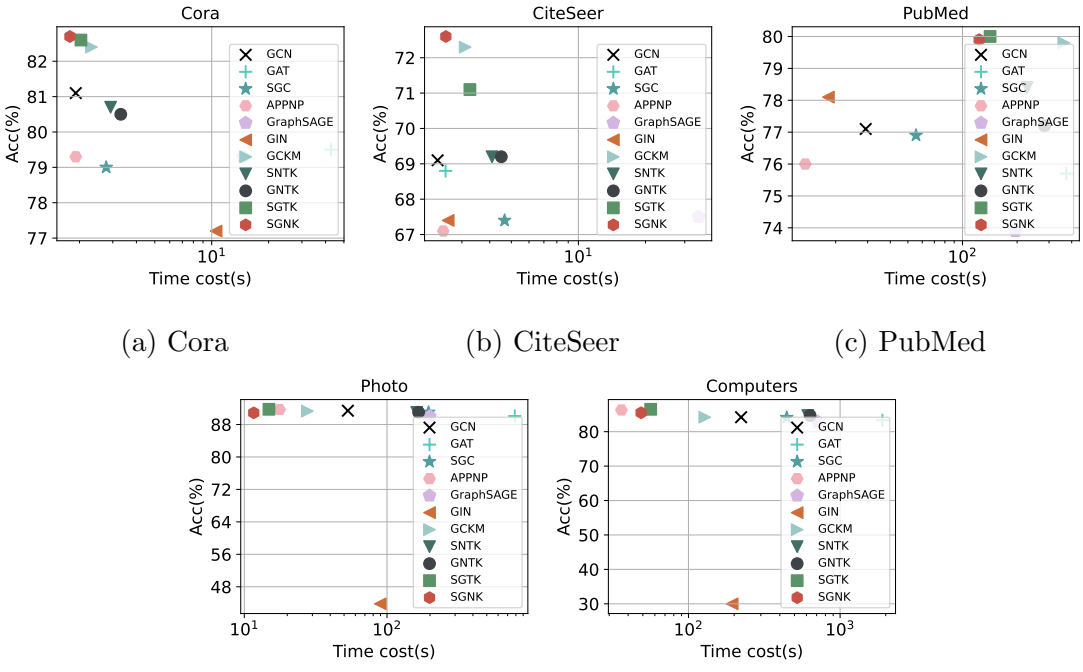
4.4.3 Node Classification Efficiency

To validate the efficiency of the proposed methods in node classification tasks, we investigated the impact of the parameter K on the time consumption of GCKM, SNTK, GNTK, SGTK, and SGNK when calculating the kernel matrix. The experimental results, as shown in Figure 4.4, indicate that GCKM, GNTK, and SNTK exhibit a linear increase in time consumption with an increase in K. In contrast, our methods remain largely unaffected by variations in the message-passing times, maintaining consistent efficiency in kernel matrix computations.

Additionally, we present the time taken by each method to achieve the classification accuracy detailed in Table 4.3. This assessment allows us to evaluate the time efficiency of each method. The results of time consumption are presented in Figure 4.3.

SGNK demonstrates low time consumption across all five datasets, followed closely by SGTK. The proposed methods exhibit time efficiency at least three times faster than GNTK, with SGNK (0.015s) outperforming GNTK (0.827s) by a factor of 53.7 on the Computers dataset. In contrast, GNN methods incur significantly higher time costs due to the need for iterative training over multiple epochs and the uncertainty of model convergence.

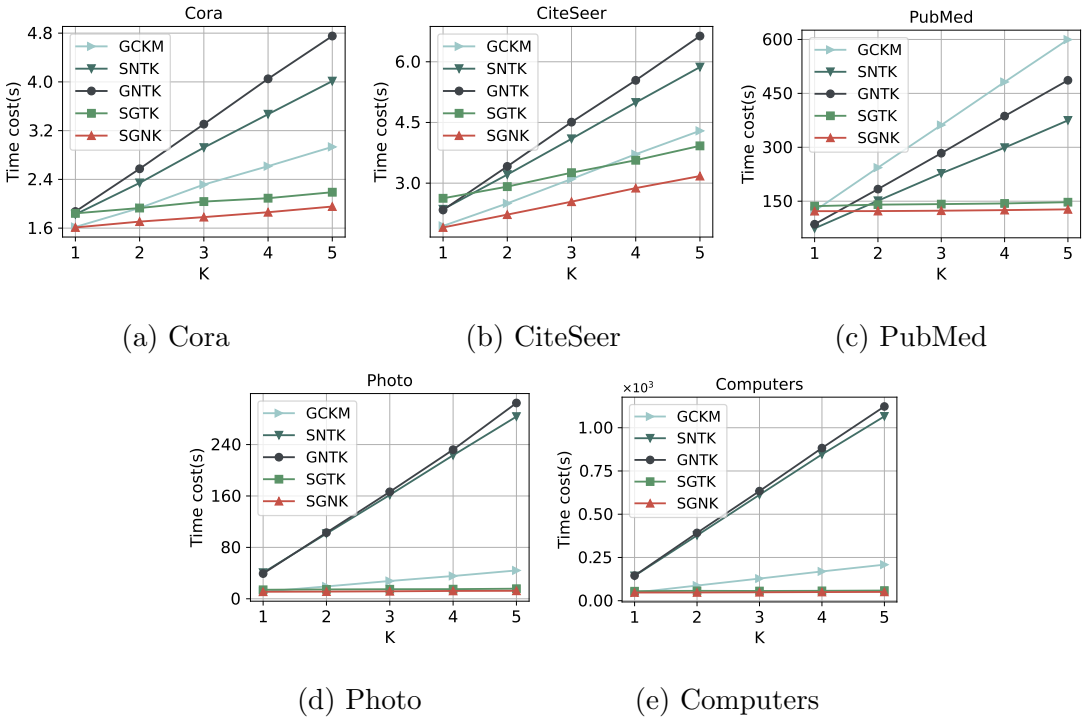
In conclusion, the proposed methods not only outperform the baselines in node classification performance but also exhibit consistent time efficiency. This consistency corroborates the theoretical analysis, providing strong evidence for the enhancement in time efficiency achieved by the proposed methods.



(a) Cora (b) CiteSeer (c) PubMed

(d) Photo (e) Computers

Figure 4.3: Node classification time consumption and accuracy comparison.



(a) Cora (b) CiteSeer (c) PubMed

(d) Photo (e) Computers

Figure 4.4: The changes of node classification time consuming with respect to K.

4.4.4 Graph Classification Performance Comparison

This subsection evaluates the proposed kernel methods for modeling similarity between graphs. To this end, we conduct experiments on four graph-level datasets, which include social network and molecular graph datasets. We employ the SVM classifier with 10-fold cross-validation to assess the performance of the proposed graph kernel methods. The evaluation metrics include the average classification accuracy and standard deviation, which are reported to provide a quantitative measure of the methods' efficacy.

As shown in Table 4.5, the proposed SGTK method achieves competitive classification performance on three out of the four datasets: IMDB-BINARY, IMDB-MULTI, and PTC_MR. Compared to other graph kernel methods, SGTK demonstrates consistent improvements in accuracy on these datasets. It is worth noting that the proposed method does not involve a learning or training phase, yet it achieves results com-

parable to—and, in some cases, better than—GNN methods that require extensive training.

The classification accuracy on graph datasets demonstrates the proposed method's effectiveness in measuring graph similarity. Despite their lower computational complexity, the proposed kernel methods deliver competitive performance, making them a practical choice for applications where computational efficiency is critical.

Table 4.5: Graph classification accuracy (%) and standard derivation. The top three performance methods are highlighted in varying shades of green, with darker shades representing superior performance.

Methods	IMDB-B	IMDB-M	PTC	MUTAG
GCN	74.0±3.4	51.9±3.8	64.2±4.3	85.6±5.8
GAT	75.8±3.4	52.0±3.5	64.3±7.5	81.1±7.2
SGC	69.6±4.9	47.4±2.0	58.7±6.5	66.5±9.6
APPNP	75.2±3.4	52.1±3.1	62.0±7.9	78.3±6.5
GraphSAGE	72.3±5.3	50.9±2.2	63.9±7.7	85.1±7.6
GIN	75.1±5.1	52.3±2.8	64.6±7.0	89.4±5.6
WL subtree	73.8±3.9	50.9±3.8	59.9±4.3	90.4±5.7
RetGK	71.9±1.0	47.7±0.3	62.5±1.6	90.3±1.1
GraphQNTK	73.3±3.6	48.1±4.3	62.9±5.0	88.4±6.5
GCKM	75.4±2.4	53.9±2.8	67.7±5.4	88.7±7.6
SNTK	75.3±2.4	52.9±4.5	65.6±5.1	85.6±6.3
GNTK	76.5±2.8	52.8±4.2	67.7±5.7	90.0±8.4
SGTK(Ours)	76.6±3.5	52.9±3.8	66.5±8.5	85.7±6.6
SGNK(Ours)	75.5±3.8	52.3±4.5	63.5±7.6	86.7±6.2

4.4.5 Graph Classification Efficiency

The subsection empirically evaluates the computational efficiency of the proposed methods on graph datasets, as outlined in the theoretical complexity analysis. In the

experiments, we vary the parameter K within the range $1, 2, 3, 4, 5$ and measure the time required to compute kernel matrices across five graph-level datasets.

Figure 4.5 presents the computation time for the GCKM, SNTK, GNTK, SGTK, and SGNK methods, corresponding to the classification accuracies reported in Table 4.5. The results align with the theoretical complexity analysis: SGNK is the most efficient method, while GNTK requires the longest computation time. Notably, the proposed methods, SGTK and SGNK, achieve at least a fourfold reduction in computation time compared to GNTK.

To investigate the impact of the K parameter on computation time, we analyze the relationship between kernel matrix computation time and K. Table 4.6 summarizes the time consumption data for all methods, and Figure 4.6 illustrates the results for comparison. As shown in Figure 4.6, the computation time of GNTK increases linearly with K. In contrast, the proposed methods, SGTK and SGNK, which employ continuous K-fold aggregation, exhibit stable computation times and achieve the lowest time costs across all tested K values.

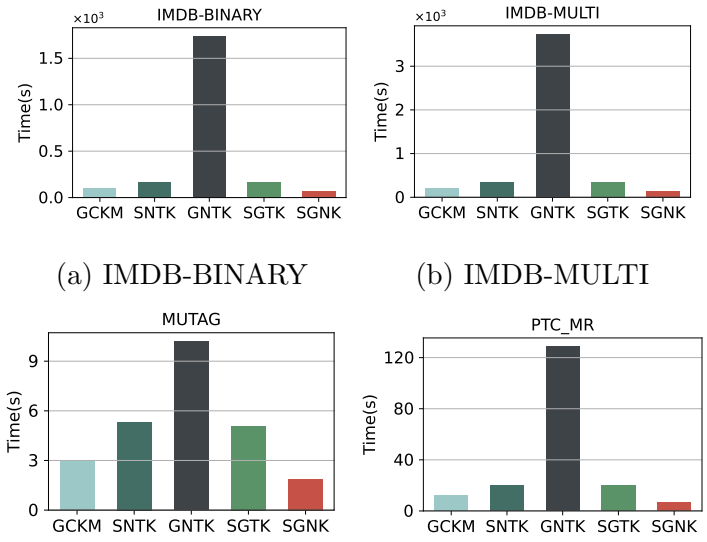
In summary, the experimental results demonstrate the computational efficiency of the proposed SGTK and SGNK methods, providing strong empirical support for the theoretical complexity analysis. These methods offer a practical advantage in scenarios where computational efficiency is critical.

4.4.6 Parameters Sensitive Analysis

This subsection investigates the impact of different K parameter values on classification accuracy in both graph and node classification tasks. We varied K within the range {1, 2, 3, 4, 5} and reported the changes in classification accuracy for SNTK, GNTK, SGTK, and SGNK on four graph classification datasets and five node classification datasets, as depicted in Figure 4.7 and Figure 4.8.

Table 4.6: Time required (s) to calculate the graph-level kernel matrix as varying K.

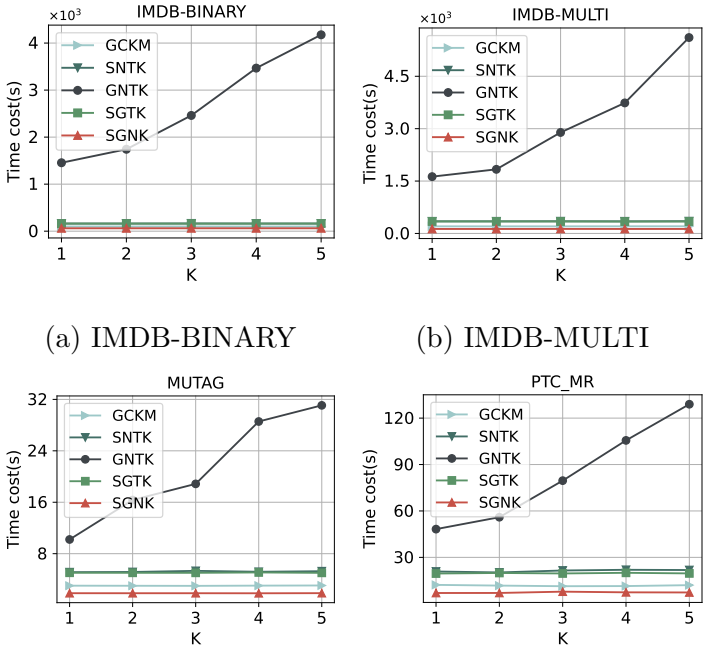
Dataset	K	GCKM	SNTK	GNTK	SGTK	SGNK
IMDB-B	1	99.1796	162.5266	1455.04	161.0031	61.5029
	2	98.7254	162.4243	1742.01	160.7982	61.5988
	3	99.4667	164.5200	2460.97	159.5090	61.8451
	4	98.1082	162.8410	3467.13	160.4228	61.8403
	5	98.6913	161.8836	4176.06	162.3239	61.8523
IMDB-M	1	203.5473	346.7882	1626.47	343.5592	128.5789
	2	202.5733	346.9627	1835.87	343.5564	127.9949
	3	203.2251	349.2096	2891.54	342.3580	129.7378
	4	204.3999	346.9047	3737.56	340.1037	128.3960
	5	204.7037	350.6760	5609.25	341.6731	128.6607
MUTAG	1	3.0228	5.1242	10.21	5.0305	1.8515
	2	3.0084	5.1546	16.35	5.0241	1.8481
	3	2.9935	5.3291	18.86	5.0208	1.8487
	4	3.0312	5.1870	28.56	5.0793	1.8402
	5	3.0546	5.2734	31.07	5.0176	1.8589
PTC	1	12.2249	20.8693	48.25	19.5684	6.9742
	2	11.7598	20.2613	55.96	19.9208	7.0001
	3	11.3650	21.5222	79.59	19.6073	7.8397
	4	11.4551	21.9708	105.60	20.0212	7.4049
	5	12.0432	21.8219	128.97	19.6431	7.3243



(a) IMDB-BINARY (b) IMDB-MULTI

(c) MUTAG (d) PTC_MR

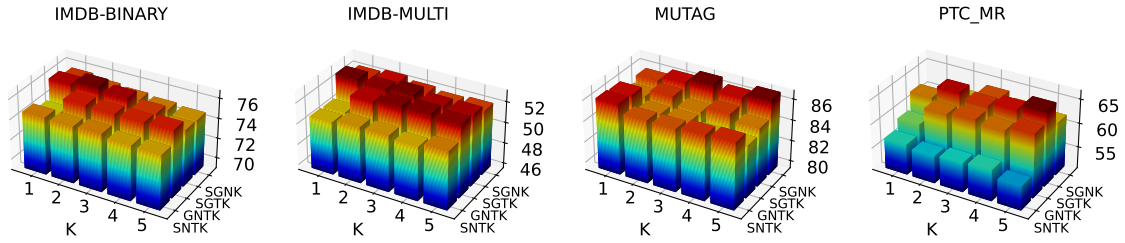
Figure 4.5: Graph classification time consuming comparison.



(a) IMDB-BINARY (b) IMDB-MULTI

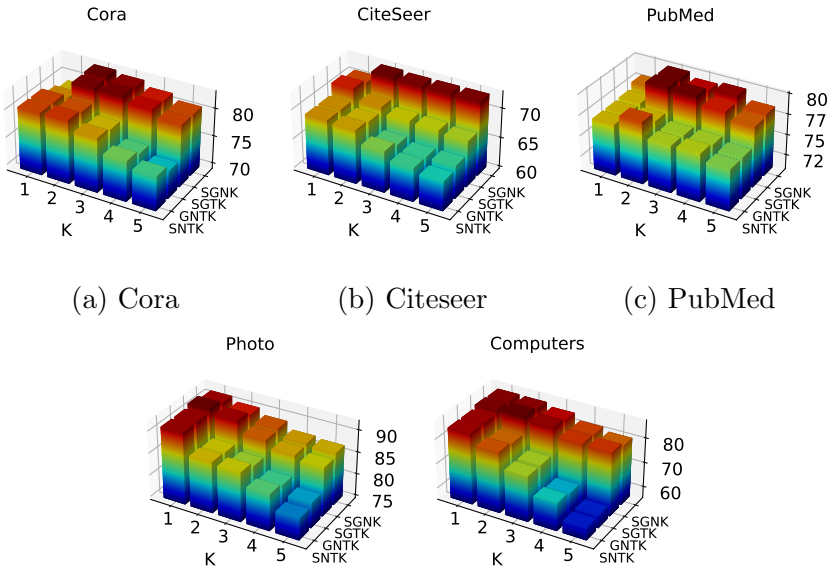
(c) MUTAG (d) PTC-MR

Figure 4.6: Variations in graph classification time consumption with changes in K.



(a) IMDB-BINARY (b) IMDB-MULTI (c) MUTAG (d) PTC_MR

Figure 4.7: The changes of graph classification accuracy with respect to K. The height of the column, representing classification accuracy, directly correlates with the color at the top, with higher columns indicated by darker colors.



(a) Cora (b) Citeseer (c) PubMed

(d) Photo (e) Computers

Figure 4.8: The changes of node classification accuracy with respect to K. The height of the column, representing classification accuracy, directly correlates with the color at the top, with higher columns indicated by darker colors.

For the graph classification task, the four methods demonstrate varying degrees of fluctuation in classification accuracy on the IMDB-MULTI and PTC_MR datasets. In contrast, the proposed methods, SGTK and SGNK, exhibit more stable classification performance across different K values on the four datasets. This observation underscores the dataset-dependent behavior of these methods. In the node classification task, SGTK and SGNK demonstrate minimal sensitivity to changes in K, maintaining stable classification accuracy across different datasets. In contrast, GNTK shows notable variability in classification accuracy when K changes, particularly on the Cora, CiteSeer, Photo, and Computers datasets. This contrast highlights the robustness of SGTK and SGNK compared to GNTK under varying K values.

These results reveal the methods' differing sensitivities to the aggregation parameter K across different datasets. This underscores the importance of parameter selection for achieving stable and high classification accuracy in both graph and node classification tasks.

4.5 Conclusion

In this paper, we propose the Simplified Graph Neural Tangent Kernel (SGTK) and its derivative, the Simplified Graph Neural Kernel (SGNK), to address the computational inefficiencies of graph kernels. By introducing continuous K-step aggregation and eliminating multi-block stacking, SGTK reduces redundancy in the Graph Neural Tangent Kernel (GNTK). SGNK further simplifies computations by interpreting an infinite-width simple graph neural network as a Gaussian process. Experimental results demonstrate that SGTK and SGNK achieve competitive classification accuracy while improving computational efficiency. These methods provide practical and scalable alternatives to traditional graph kernels and graph neural networks, particularly in resource-constrained scenarios.

Chapter 5

Efficient Graph Condensation through Gaussian Process Inference

This chapter presents our third major contribution to efficient graph condensation, introducing the Graph Condensation via Gaussian Process (GCGP) framework. Building upon Chapters 3 and 4, this work represents the culmination of our thesis contributions. We address the final bottleneck by replacing iterative GNN training with efficient Gaussian Process inference, achieving unprecedented efficiency while maintaining performance. The GCGP framework establishes the final algorithmic framework that integrates theoretical foundations with practical applications, providing a comprehensive solution for large-scale graph learning.

5.1 Introduction

Graph-structured data, consisting of nodes and edges, is a versatile representation used to model various real-world systems, including social networks[1, 85], trans-

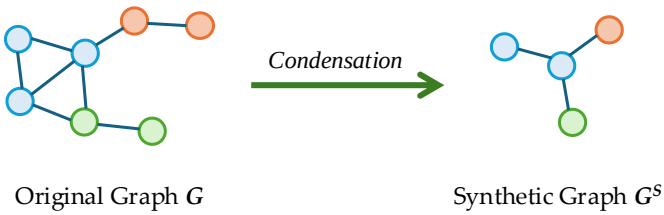


Figure 5.1: Graph condensation reduces a large graph dataset G into a smaller one $G^{\mathcal{S}}$ with fewer nodes, while preserving essential information.

portation infrastructures[14, 27], and molecular structures[101, 69]. 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, 111, 78] have been developed as a specialized class of deep neural networks. GNNs are designed to process graph data by leveraging a message-passing mechanism[107]. 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[104]. This capability has made GNNs highly effective in numerous graph mining tasks[9, 42, 77]. Despite their success, the application of GNNs faces significant challenges due to the large scale of graph data[33, 17, 76]. 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.

Graph condensation [40, 39] 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 costs and training time. By mitigating the resource-intensive nature of GNN training, graph condensation techniques facilitate the scalability and broader

applicability of GNNs [56, 26]. Current graph condensation methods are commonly framed as bi-level optimization problems [40, 120], where the inner loop trains 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 [92] caused by the extensive iterative training of GNNs. The nested loops of bi-level optimization require iterative updates for both GNN parameters and condensed data, resulting in slow convergence. Moreover, to improve the generalization of the condensed data across GNNs with diverse initializations, the GNN in the inner loop often involves thousands of parameter reinitializations. This excessive computational demand undermines the efficiency of graph condensation, to the extent that training models directly on raw data may become more practical than using condensed data.

Gaussian Process (GP) [103, 49], a non-parametric method, 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 GPs 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

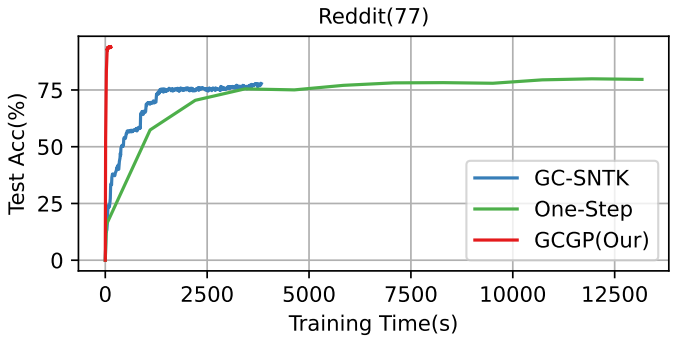


Figure 5.2: Condensation time and accuracy comparison on Reddit dataset. By condensing 153,431 training nodes into a graph with only 77 synthetic nodes, the GCGP achieves an accuracy of 93.9% on the test set, which is comparable to the accuracy obtained using the full training set on GCN, while also demonstrating the fastest runtime. For more experimental results about the condensation efficiency, please refer to Section 5.4.4.

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 the adjacency matrix, which is non-differentiable and incompatible with gradient-based optimization methods. This limitation hinders the 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 [49, 104]. This approach systematically improves GP predictive performance. For optimizing structural information, we relax the binary adjacency matrix [39] into a differentiable form using concrete random variables [61]. As the temperature approaches zero, the relaxed adjacency matrix converges to a binary form. This relaxation enables gradient-based optimization to refine the graph structure effectively.

As shown in Figure 5.2, our method condenses the training set of the Reddit [28]

dataset to 77 nodes, representing only 0.05

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 serve 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, particularly in optimizing the synthetic graph's structure.

5.2 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 loops are utilized to optimize the model parameter θ and the condensed graph $G^{\mathcal{S}}$, respectively. 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 optimizes 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

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathcal{L}(f_\theta, G) \\ s.t. \quad &\theta = \arg\min_{\theta} \ell(f_\theta, G^{\mathcal{S}}). \end{aligned} \tag{5.1}$$

In general, the convergence of the 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 a random initialization distribution P_{θ_0}. To address this constraint, multiple random initializations are required, and problem (5.1) is accordingly modified as follows:

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\mathcal{L}(f_\theta, G)\right] \\ &s.t. \quad \theta = \arg\min_{\theta} \ell(f_\theta, G^{\mathcal{S}}), \end{aligned} \tag{5.2}$$

where the loss function of the outer loop $\mathcal{L}$ is designed as matching the gradients [124] that are calculated by $G^{\mathcal{S}}$ and G on model f_{θ_t}, respectively. This can be expressed as follows:

$$\mathcal{L} = D(\nabla_{\theta_t} \ell(f_{\theta_t}, G^{\mathcal{S}}), \nabla_{\theta_t} \ell(f_{\theta_t}, G)), \tag{5.3}$$

where $D(\cdot, \cdot)$ is the function measuring the distance between the 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 a specific optimization algorithm $\texttt{opt-alg}_\theta$ (e.g., Stochastic Gradient Descent, SGD). Denote T as the total number of iterations of the outer loop. Finally, the bi-level graph condensation problem becomes:

$$\begin{aligned} &\min_{G^{\mathcal{S}}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_{t=0}^{T} D(\nabla_{\theta_t} \ell(f_{\theta_t}, G^{\mathcal{S}}), \nabla_{\theta_t} \ell(f_{\theta_t}, G))\right] \\ &s.t. \quad \theta_t = \texttt{opt-alg}\left[\ell(f_{\theta_{t-1}}, G^{\mathcal{S}})\right]. \end{aligned} \tag{5.4}$$

Problem (5.4) 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.

5.3 Methodology

This section introduces the proposed GCGP framework, which optimizes a synthesized condensed graph specifically designed to enhance the efficiency of downstream tasks. A novel covariance function tailored for graph-structured data serves as the foundation of the framework. To further refine the structural information of the condensed graph, we adopt concrete random variables, which replace the binary adjacency matrix with a differentiable counterpart. Finally, we outline the optimization process and provide a complexity analysis, highlighting the condensation efficiency advantages of the proposed GCGP.

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 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 $m \ll n$.

5.3.1 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 [103], which can be defined by a mean function and a covariance function. Usually, for notational simplicity, we will take the mean function to be zero $\mu(x) = 0$. Given the input $\mathcal{X}$ and the observed target $\mathcal{Y}$, let the test data be x_t, the function value be f_t, $\mathcal{Y}$ and f_t

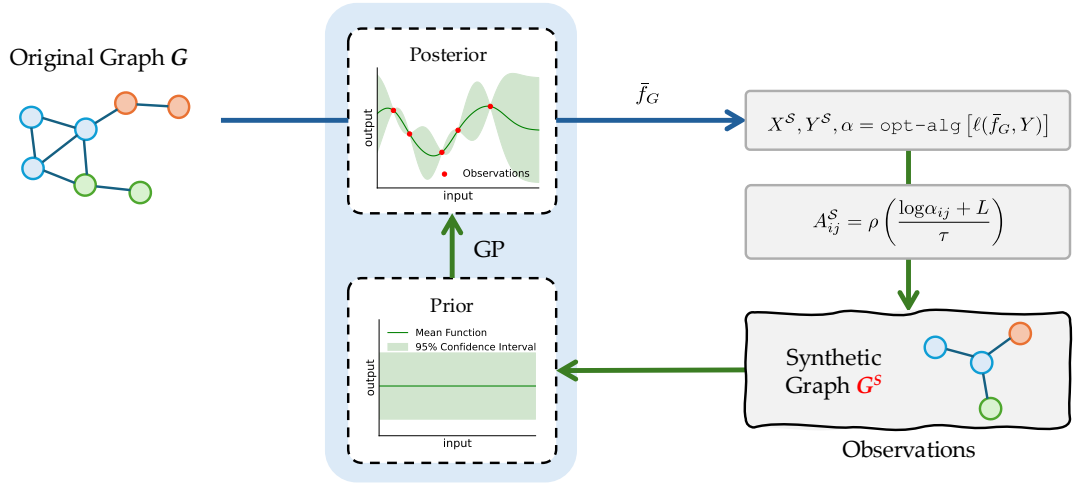


Figure 5.3: The workflow of the proposed GCGP framework involves three key steps. First, the condensed synthetic graph $G^{\mathcal{S}}$ is utilized as the observations for the GP. Next, predictions are generated for the test locations, corresponding to the original graph G. Finally, the condensed graph is iteratively optimized by minimizing the discrepancy between the GP's predictions and the ground-truth labels.

follow a joint Gaussian process,

$$\begin{bmatrix} \mathcal{Y} \\ f_t \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathcal{K}(\mathcal{X}, \mathcal{X}) + \sigma_\varepsilon^2 I & \mathcal{K}(\mathcal{X}, x_t) \\ \mathcal{K}(x_t, \mathcal{X}) & \mathcal{K}(x_t, x_t) \end{bmatrix}\right). \tag{5.5}$$

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 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.

$$\begin{bmatrix} Y^{\mathcal{S}} \\ f_G \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I & \mathcal{K}(G^{\mathcal{S}}, G) \\ \mathcal{K}(G, G^{\mathcal{S}}) & \mathcal{K}(G, G) \end{bmatrix}\right), \tag{5.6}$$

where $\mathcal{K}$ is the covariance function, modeling the dependencies between samples in a probabilistic way. In Gaussian processes, the choice of the covariance function is crucial, as it encodes prior assumptions about the function's smoothness, variability, and how closely the function values at different input points are correlated.

To better understand the origin of the covariance function, we consider a Bayesian linear regression model defined in a high-dimensional feature space. Specifically, denoting a mapping $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, which transforms an input x into a d'-dimensional feature space. Based on this mapping model, a Bayesian linear regression model can be 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 leads to the definition of a covariance function, which encodes input similarity and models dependencies in Gaussian processes,

$$\begin{aligned} Cov(Y^{\mathcal{S}}) &= \mathbb{E}[(f(G^{\mathcal{S}}) + \varepsilon)(f(G^{\mathcal{S}}) + \varepsilon)] \\ &= \phi(G^{\mathcal{S}})^\top \Sigma_p \phi(G^{\mathcal{S}}) + \sigma_\varepsilon^2 I \\ &= \mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I, \end{aligned} \tag{5.7}$$

where I is the identity matrix.

The joint prior in Equation (5.6) defines the distribution over both observed and unobserved function values in Gaussian process regression. By conditioning this prior on the observed data using the standard Gaussian conditioning identity [103], we obtain the posterior distribution over the unobserved points,

$$f_G | G, G^{\mathcal{S}}, Y^{\mathcal{S}} \sim \mathcal{N}(\bar{f}_G, Cov(f_G)), \tag{5.8}$$

where,

$$\bar{f}_G = \mathcal{K}(G, G^{\mathcal{S}})[\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I]^{-1} Y^{\mathcal{S}}, \tag{5.9}$$

$$Cov(f_G) = \mathcal{K}(G, G) - \mathcal{K}(G, G^{\mathcal{S}})[\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}}) + \sigma_\varepsilon^2 I]^{-1} \mathcal{K}(G^{\mathcal{S}}, G). \tag{5.10}$$

The predictive mean $\bar{f}_G$ represents the best estimate (in the mean-square sense) of the function at the test inputs, and the predictive variance $\mathrm{Cov}(f_G)$ quantifies the uncertainty associated with these predictions.

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 [59, 12]. 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 it a strong candidate for graph condensation, enabling the bi-level condensation process to be reformulated as a simpler optimization problem:

$$\min_{G^{\mathcal{S}}} \ell(\bar{f}_G, Y). \tag{5.11}$$

Compared to the bi-level condensation model in Equation (5.4), 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.

5.3.2 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 [64]. 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 [4].

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 (5.7), 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 pivotal roles 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 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 [104]. 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:

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

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 in Equation (5.7) can be

represented as:

$$\phi(X) = \psi(\hat{A}^k X W) + \varepsilon, \tag{5.12}$$

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:

$$\begin{aligned} Y =& \phi(X)W + \varepsilon, \\ =& \psi(\hat{A}^k X W^{(0)} + \varepsilon^{(0)})W^{(1)} + \varepsilon^{(1)}, \end{aligned} \tag{5.13}$$

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 hidden layer approaches infinity ($d_1 \to \infty$), the neural network converges to a Gaussian process, enabling 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 can be expressed as the expectations of the post-activation outputs [64, 49]. Let $\beta = \sigma_\varepsilon^2$. For two graphs $G = \{X, A\}$ and $G' = \{X', A'\}$, the covariance function $\mathcal{K}$ is defined as:

$$\mathcal{K}(G, G') = \mathbb{E}_{f\sim\mathcal{GP}(0,\Lambda^{(0)})}[\psi(f(G))\psi(f(G'))] + \beta, \tag{5.14}$$

where the covariance matrix $\Lambda(G, G')$ is constructed from a base covariance function $\Sigma(G, G')$, given by,

$$\Lambda(G, G') = \begin{bmatrix} \Sigma(G, G) & \Sigma(G, G') \\ \Sigma(G', G) & \Sigma(G', G') \end{bmatrix}, \tag{5.15}$$

$$\Sigma(G, G') = \frac{1}{d}\hat{A}^k X (\hat{A'}^k X')^\top + \beta, \tag{5.16}$$

where $\hat{A}^k$ and $\hat{A'}^k$ denote the normalized adjacency matrices raised to the k-th power, capturing the graph structural information up to k-hop neighborhoods. The term β represents the observation noise variance and is incorporated into the predictive covariance to model measurement uncertainty.

The elements of the covariance function $\mathcal{K}(G, G')$ can be calculated analytically as [102],

$$\mathcal{K}_{ij} = \frac{2}{\pi} \sin^{-1} \frac{2\tilde{x}^\top i \Sigma w \tilde{x}_j}{\sqrt{(1 + 2\tilde{x}_i^\top \Sigma_w \tilde{x}_i)(1 + 2\tilde{x}_j^\top \Sigma_w \tilde{x}_j)}}, \tag{5.17}$$

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 $\Sigma_w = \text{diag}(\sigma_w^2, \sigma_w^2, \cdots, \beta)$ characterizes the noise structure in the model, where σ_w^2 is typically set to 1. Equation (5.17) facilitates computational efficiency by enabling element-wise calculations, which are well-suited for acceleration using GPUs. Thus, this approach provides a computationally efficient and scalable solution for reducing the time cost of GP.

5.3.3 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. [3, 39]. 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 [61, 37] technique to approximate the discrete adjacency matrix with a continuous counterpart. For the synthetic adjacency matrix, the Concrete relaxation reformulates the binary variable $A^{\mathcal{S}}ij$, which originally follows a Bernoulli distribution, into a differentiable function. This reformulation introduces concrete random variables αij and τ, such that $A^{\mathcal{S}}ij \sim \text{BinConcrete}(\alpha ij, \tau)$. Let $L \sim \text{Logistic}$, $A^{\mathcal{S}}ij$ is reparameterized as,

$$A^{\mathcal{S}}ij = \rho\left(\frac{\log \alpha_{ij} + L}{\tau}\right), \tag{5.18}$$

where $\rho(x) = \frac{1}{1+\exp(-x)}$, and L is defined as $L \overset{d}{=} \log(U) - \log(1-U)$, with $U \sim \mathcal{U}(0, 1)$.

The parameters α_{ij} and τ are constrained to the interval $(0, \infty)$, where τ denotes the temperature parameter.

In Equation (5.18), 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, $A^{\mathcal{S}}_{ij}$, equals 1 is:

$$\mathbb{P}\left(\lim_{\tau \to 0} A^{\mathcal{S}}_{ij} = 1\right) = \frac{\alpha_{ij}}{1 + \alpha_{ij}}. \tag{5.19}$$

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

5.3.4 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 while avoiding additional training loops. The loss function is:

$$\ell(\bar{f}G, Y) = ||\bar{f}G - Y||^2_F. \tag{5.20}$$

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 (5.18). Subsequently, GP predictions $\bar{f}_G$ for the graph G are computed as described in Equation (5.9). 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 enable a smooth transition from soft to discrete adjacency representations, the temperature parameter τ is annealed during training. As $\tau \to 0$, the sampled adjacency matrix $A^{\mathcal{S}}$ becomes increasingly binary. The annealing schedule is defined as:

$$\tau = \max\left(\text{temp}_{\text{start}} \cdot \left(\frac{\text{temp}_{\text{end}}}{\text{temp}_{\text{start}}}\right)^{\frac{\text{epoch}}{100}},\ \text{temp}_{\text{end}}\right), \tag{5.21}$$

where $\text{temp}_{\text{start}} = 0.3$ and $\text{temp}_{\text{end}} = 0.01$. The full training procedure is shown in Algorithm 5.

5.3.5 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 numbers 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 numbers 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 covariance computation has a complexity of $O(mnd)$,

Algorithm 5 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: $X_{ij}^{\mathcal{S}}, Y_{ij}^{\mathcal{S}} \sim \mathcal{U}(0,1), \quad \forall i,j$, and $\alpha_{ij} \sim \mathcal{U}(0,2), \quad \forall i,j$

while not converge **do**

  Update $A^{\mathcal{S}}$ by:

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

  Calculate $\mathcal{K}(G, G^{\mathcal{S}})$ and $\mathcal{K}(G^{\mathcal{S}}, G^{\mathcal{S}})$ by Equation (5.17)

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

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

  Update $\tau \to 0$

end while

for each (i,j) pair **do**

  if $\frac{\alpha_{ij}}{1+\alpha_{ij}} > 0.5$ **then**

    $A_{ij}^{\mathcal{S}} = 1$

  else

    $A_{ij}^{\mathcal{S}} = 0$

  end if

end for

while GP prediction has $O(m^3 + mnd)$. Thus, the total complexity of GCGP is: $O(tm^3+tmnd)$. This analysis highlights that the GCGP method achieves significantly lower computational complexity than GCond, particularly for large-scale datasets where n and $|E|$ dominate the cost.

5.4 Experiments

Table 5.1: 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

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.

Table 5.2: 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)	One-step	GCond		GC-SNTK		SFGC	SimGC	DisCo	GCGP (Our)		Full (GCN)
			X	X,A	X	X,A				X	X,A	
Cora	1.30% (35)	80.2±0.73	75.9±1.2	81.2±0.7	82.2±0.3	81.7±0.7	80.1±0.4	80.8±2.3	76.9±0.8	82.4±0.3	78.8±1.0	81.1±0.5
	2.60% (70)	80.4±1.77	75.7±0.9	81.0±0.6	82.4±0.5	81.5±0.7	81.7±0.5	80.9±2.6	78.7±0.3	82.5±0.7	80.5±0.9	
	5.20% (140)	79.8±0.64	76.0±0.9	81.1±0.5	82.1±0.1	81.3±0.2	81.6±0.8	82.1±1.3	78.8±0.5	82.6±0.7	80.9±1.0	
CiteSeer	0.90%(30)	70.8±0.3	71.6±0.8	71.8±1.2	65.7±0.3	64.8±0.7	71.4±0.5	73.8±2.5	67.9±0.7	73.1±1.5	70.3±1.5	71.7±0.1
	1.80%(60)	71.7±0.6	70.8±0.1	72.6±0.9	67.0±0.3	65.9±0.2	72.4±0.4	72.2±0.5	67.6±0.5	72.8±0.6	71.3±1.0	
	3.61%(120)	70.1±0.2	71.4±0.6	72.5±0.4	67.3±0.3	66.3±0.5	70.6±0.7	71.1±2.8	67.9±0.9	72.3±0.5	71.1±0.5	
PubMed	0.08% (15)	77.7±0.12	59.4±0.7	78.3±0.2	78.9±0.7	71.8±6.8	78.7±0.5	71.0±1.2	76.1±0.8	79.2±0.5	74.2±0.7	77.1±0.3
	0.15% (30)	77.8±0.17	51.7±0.4	77.1±0.3	79.3±0.3	74.0±4.9	78.5±0.7	70.8±2.1	77.5±0.6	79.2±0.7	76.8±0.9	
	0.30% (60)	77.1±0.44	60.8±1.7	78.4±0.3	79.4±0.3	76.4±2.8	78.8±0.8	71.0±1.8	77.7±0.4	79.2±0.8	77.5±0.5	
Amazon Photo	0.5% (40)	85.7±0.4	86.6±0.6	85.5±0.2	88.6±0.1	82.0±2.5	90.1±0.3	89.5±1.3	88.1±0.8	92.1±0.4	90.2±0.9	91.4±0.8
	1.0% (80)	86.2±0.2	87.3±0.4	86.5±0.5	88.5±0.2	83.7±1.0	90.7±0.8	89.4±0.9	89.5±0.9	92.0±0.1	90.6±0.5	
	2.1% (160)	86.8±0.6	87.7±0.3	86.6±0.6	88.6±0.6	83.8±3.0	90.6±0.6	89.6±1.1	89.3±0.6	92.1±0.3	91.4±0.6	
Amazon Computers	0.4% (50)	82.1±0.3	82.4±0.6	82.7±0.3	83.5±0.3	77.2±5.2	84.7±0.5	82.9±1.2	82.4±0.6	86.1±0.3	85.1±0.5	84.2±0.3
	0.7% (100)	82.4±0.5	82.2±0.5	82.5±0.2	83.1±0.6	77.3±3.5	85.0±0.2	82.6±1.0	82.7±0.4	86.4±0.3	85.4±0.4	
	1.5% (200)	82.3±0.6	82.7±0.3	82.8±0.4	83.0±0.7	77.7±5.0	85.5±0.7	83.9±0.9	84.0±0.7	86.5±0.4	86.0±0.5	
Ogbn-arxiv	0.05% (90)	59.2±0.1	61.3±0.5	59.2±1.1	63.5±0.3	64.2±0.2	65.5±0.7	63.6±0.8	64.0±0.7	65.7±0.3	65.0±0.4	71.4±0.1
	0.25% (454)	60.1±0.8	64.2±0.4	63.2±0.3	64.5±0.1	65.1±0.8	66.1±0.4	66.4±0.3	65.9±0.5	65.3±0.1	65.4±0.1	
	0.5% (909)	60.0±0.1	63.1±0.5	64.0±0.4	65.7±0.4	65.4±0.5	66.8±0.4	66.8±0.4	66.2±0.1	65.7±0.1	65.3±0.3	
Reddit	0.05% (77)	80.7±0.2	88.4±0.4	88.0±1.8	77.9±0.9	74.3±0.5	89.7±0.2	89.6±0.6	91.4±0.2	93.9±0.1	92.7±0.1	93.9±0.0
	0.1%(153)	81.1±0.4	89.3±0.1	89.6±0.7	78.7±0.6	74.8±0.7	90.0±0.3	92.0±0.3	91.8±0.3	93.8±0.1	92.6±0.1	
	0.2% (307)	82.3±0.7	88.8±0.4	90.1±0.5	78.9±0.1	85.2±1.2	89.9±0.4	92.6±0.1	91.7±0.3	94.0±0.1	92.6±0.4	

5.4.1 Experimental Settings

Datasets. We evaluate on seven benchmark datasets of varying scales: **Cora**, **Citeseer**, and **Pubmed** [114] are citation networks with bag-of-words or TF-IDF features [54]; **Photo** and **Computers** [73] are Amazon co-purchase graphs with product review features [68]; **Ogbn-arxiv** [35] is a citation graph of arXiv CS papers with averaged word embeddings as features; and **Reddit** [28] is a large-scale social network with post-level community labels. The details of the datasets are shown in Table 5.1.

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

- **One-step**[39] simplifies the GCond method by performing the inner and outer optimization steps only once, thereby accelerating the condensation process.
- **GCond**[40] is a bi-level graph condensation method that employs a GNN as the prediction model.
- **GC-SNTK**[92] is a graph condensation method based on kernel ridge regression, utilizing the structure-based neural tangent kernel to quantify similarity among nodes.
- **SFGC**[125] is a structure-free graph condensation method that distills large graphs into compact node sets without explicit structural information.
- **SimGC**[110] uses a pre-trained SGC model to align condensed and original graphs, achieving up to 10 times speedup with competitive performance.
- **DisCo**[109] is a GNN-free graph condensation method that decouples node and edge condensation into two stages, achieving speedup and competitive accuracy.

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 experimental section adopts the best-performing configuration: SGC [104] for condensation and GCN [43] for testing, unless otherwise specified.

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 5.3.

5.4.2 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

Table 5.3: 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

nodes). Table 5.2 presents the classification accuracy and standard deviation of various graph condensation methods across multiple datasets and condensation ratios. The top three performing methods for each setting are highlighted in green, with darker shades indicating better performance.

Overall, the proposed GCGP method consistently delivers top-tier performance across diverse datasets and condensation ratios. On Cora and Citeseer, it achieves 82.5% and 72.8% accuracy at 2.6% and 1.8% condensation rates, respectively, outperforming all baselines.

On Amazon Photo and Computers, GCGP not only surpasses other condensation methods but also outperforms full-data GCN. For example, with just 0.5% of nodes on Photo, GCGP achieves 92.1% accuracy, exceeding the 91.4% from full GCN. A similar gain is observed on Computers (86.1% vs. 84.2%). On large-scale datasets, GCGP demonstrates scalability and competitive performance. For instance, on Ogbn-

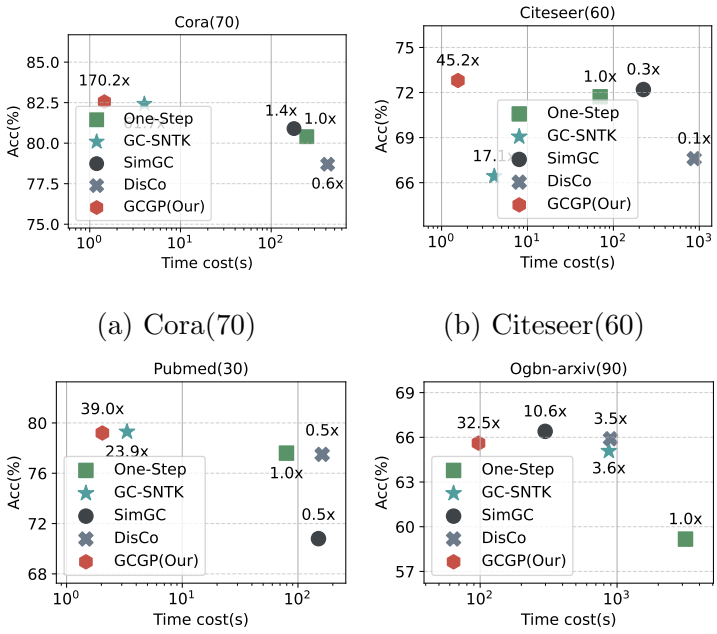
arxiv, it achieves 65.7% accuracy using only 90 condensed nodes. On Reddit, GCGP reaches 93.9% accuracy with merely 77 synthetic nodes (0.05% of the original size), closely matching the result obtained by GCN trained on all 153,431 nodes. These results highlight GCGP's effectiveness in extreme compression scenarios and its ability to generalize across diverse graph domains.

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.

5.4.3 Efficient Evaluation

To evaluate the efficiency of the proposed GCGP method, we examine its graph condensation time. 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 it with several representative baselines, including One-step, GC-SNTK, SimGC, and DisCo. GCond and SFGC are excluded from this comparison due to their prohibitive computational cost. Experiments are conducted on four benchmark datasets: Cora, Citeseer, Pubmed, and Ogbn-arxiv.

Figure 5.4 illustrates the relationship between condensation time (x-axis) and classification accuracy (y-axis), with each marker representing a method's performance. Using One-step as the baseline, we annotate the relative speedup of each method beside its marker to highlight efficiency gains. As shown in the figure, GCGP consistently demonstrates superior efficiency while maintaining high accuracy. For instance, on Cora with 70 condensed nodes, GC-SNTK achieves a 61.7× speedup over One-step, whereas GCGP further improves this to 170.3×. On Citeseer and Ogbn-arxiv, GCGP achieves speedups of 45.2× and 32.5×, respectively. These results confirm



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

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

Figure 5.4: The runtime efficiency of GC-SNTK, SimGC, DisCo, and GCGP 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 runtime of the other methods and represents the relative acceleration of each method.

that GCGP accelerates the condensation process without compromising the quality of the condensed graphs.

The results in Figure 5.4 demonstrate that the proposed GCGP method improves condensation efficiency while maintaining high classification accuracy, outperforming other graph condensation methods. This efficiency is particularly important, as the primary purpose of graph condensation is to accelerate training. If the condensation process itself is excessively time-consuming, it undermines this goal. By substantially reducing condensation time without sacrificing performance, GCGP enhances the practicality and scalability of graph condensation in real-world applications.

5.4.4 Comparison Between GCGP and Kernel-Based Method

To further assess the efficiency of the proposed method, we compare GCGP with the kernel-based baseline GC-SNTK. The results, shown in Figure 5.5, offer a comprehensive view of their performance across different datasets and condensation scales. Experiments are conducted on six datasets—Cora, Citeseer, Pubmed, Photo, Computers, and Ogbn-arxiv—under three condensation ratios, resulting in 18 settings. For each setting, we record the classification accuracy with respect to the training time required by the condensation process.

In Figure 5.5, the x-axis indicates condensation time, while the y-axis shows the corresponding test accuracy. As observed, GCGP consistently achieves higher accuracy, demonstrating both faster condensation and better performance. Across most datasets, it outperforms GC-SNTK in terms of classification accuracy, highlighting the effectiveness of GCGP in improving both efficiency and condensation quality.

Additionally, Table 5.4 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

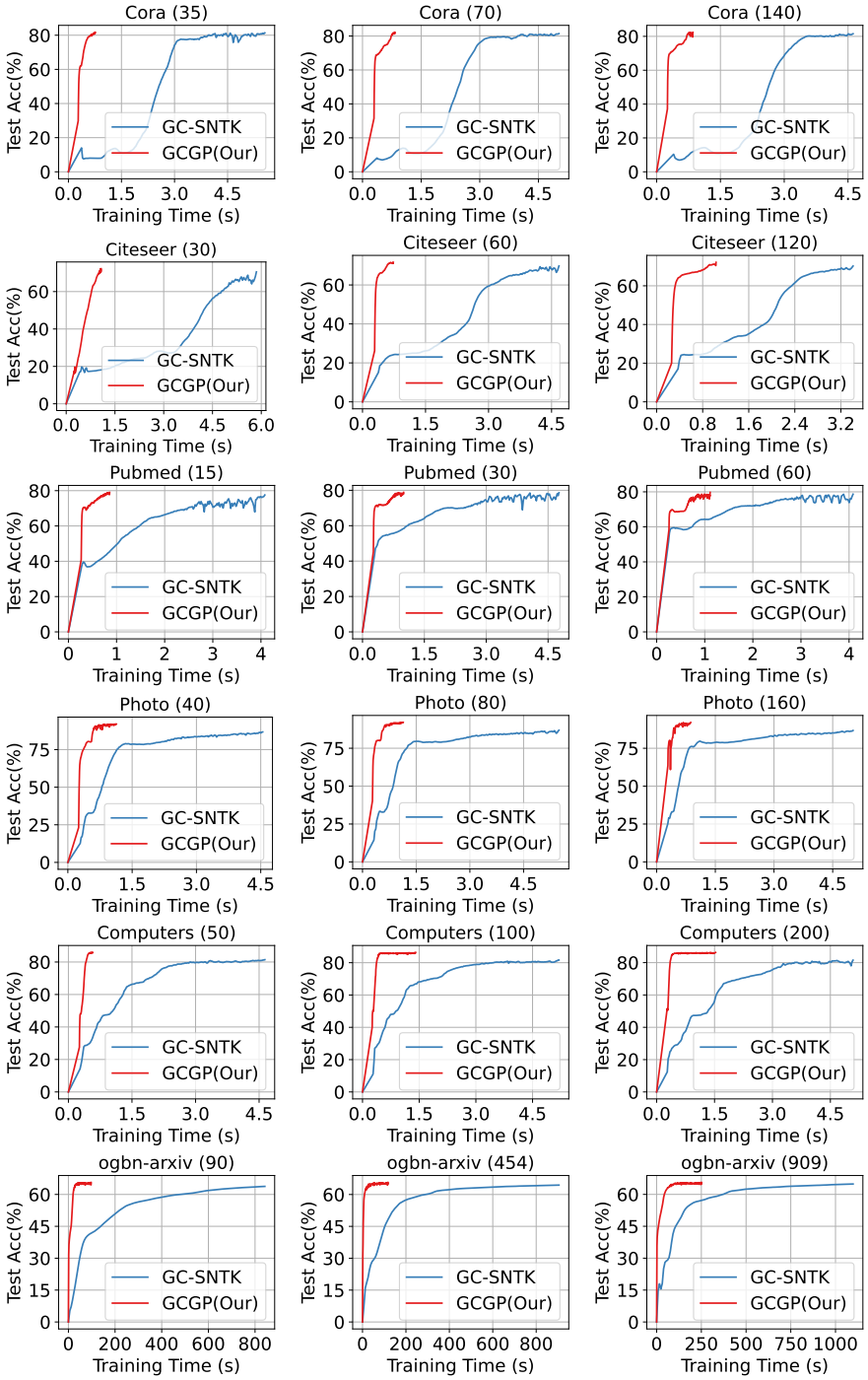


Figure 5.5: Condensation efficiency comparison between the proposed GCGP method and GC-SNTK is conducted on six datasets. Each row corresponds to a dataset (Cora, Citeseer, Pubmed, Photo, Computers, ogbn-arxiv), and each column shows results under different condensation scales (0.25, 0.5, 1.0). The x-axis represents training time, while the y-axis indicates test accuracy. GCGP consistently achieves faster training times than GC-SNTK across all scales.

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.

Table 5.4: Condensation time comparison of the GC-SNTK and the GCGP

Dataset	Size	GC-SNTK (s)	GCGP(s)	Speedup
Cora	1.30% (35)	5.5843	0.7672	7.28×
	2.60% (70)	5.0616	0.8383	6.04×
	5.20% (140)	4.6553	0.8595	5.42×
Citeseer	0.90%(30)	5.8787	1.0806	5.44×
	1.80%(60)	4.7093	0.7389	6.37×
	3.61%(120)	3.4389	1.0386	3.31×
Pubmed	0.08% (15)	4.1034	0.8610	4.77×
	0.15% (30)	4.7844	1.0022	4.77×
	0.30% (60)	4.1030	1.1172	3.67×
Amazon Photo	0.5% (40)	4.5685	1.1389	4.01×
	1.0% (80)	5.5139	1.1430	4.82×
	2.1% (160)	5.0429	0.8767	5.75×
Amazon Computers	0.4% (50)	4.6563	0.5787	8.05×
	0.7% (100)	5.2212	1.4125	3.71×
	1.5% (200)	5.1126	1.5352	3.33×
Ogbn-arxiv	0.05% (90)	842.0065	97.2541	8.66×
	0.25% (454)	904.8845	197.3606	4.58×
	0.5% (909)	1098.2102	298.4082	3.68×

5.4.5 Parameter Sensitivity Analysis

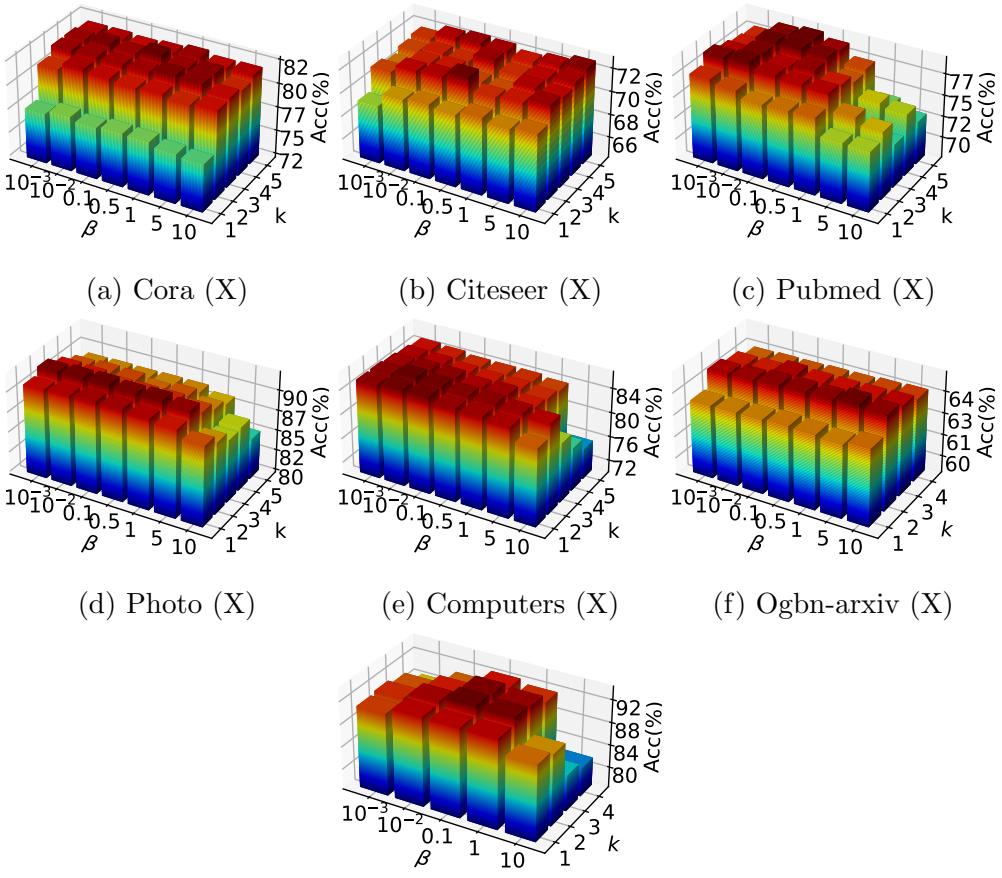
To analyze the sensitivity of the proposed method to hyperparameters, we conduct experiments by varying two key parameters: β in GP and the power k of the adjacency matrix A in the calculation of covariance function. These experiments are performed on multiple datasets, including Cora, Citeseer, Pubmed, 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 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 5.6 and Figure 5.7, 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 5.6, 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. Ogbn-arxiv is insensitive to both parameters, whereas Reddit performs poorly when β and k are large.

Figure 5.7 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 performance differences depending on the parameter settings. These findings provide valuable insights into the selection of hyperparameters for different datasets.

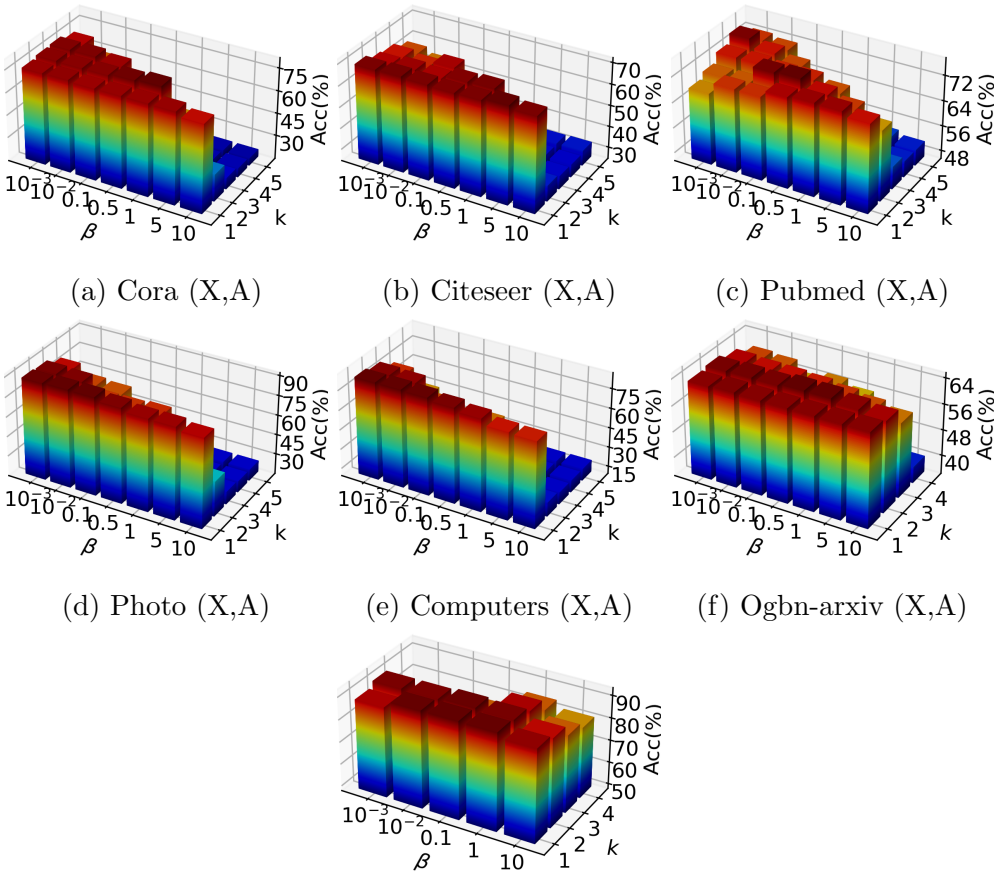


(a) Cora (X) (b) Citeseer (X) (c) Pubmed (X)

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

(g) Reddit (X)

Figure 5.6: 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)

Figure 5.7: Parameter sensitivity analysis condenses both node features $X^{\mathcal{S}}$ and structural information $A^{\mathcal{S}}$.

5.4.6 Generalization Across Different Models

To evaluate whether the graph data condensed by GCGP retains essential structural and semantic information for downstream tasks, we apply the condensed data to train a variety of GNN models. This setting allows us to assess the generalization ability of GCGP beyond the model used during condensation. The experimental results show that models trained on GCGP-condensed graphs consistently achieve competitive performance, demonstrating the robustness and transferability of the condensed data to different GNN architectures.

Table 5.5 summarizes the generalization performance of graph data condensed by various methods—GCond, GC-SNTK, and the proposed GCGP—on multiple GNN models, including GCN [43], SGC [104], APPNP [24], GraphSAGE [28], and KRR [89], 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 results 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.

5.4.7 Ablation Study

To evaluate the effectiveness of the proposed covariance function in modeling node similarity, we conduct a comprehensive ablation study comparing GCGP with several alternative covariance functions, including Dot Product, Neural Tangent Kernel (NTK) [36], Structure-based NTK (SNTK) [92], and LightGNTK [112]. The results are summarized in Table 5.6 and Table 5.7, covering both classification accuracy and condensation time across seven datasets.

As shown in Table 5.6, GCGP consistently outperforms all baseline covariance functions in terms of classification accuracy, particularly on datasets such as Cora, Citeseer, Photo, Computers, and Reddit. For example, on Photo, GCGP achieves 92.0% accuracy, significantly surpassing SNTK (88.5%) and LightGNTK (89.4%). Similarly, on Reddit, GCGP reaches 93.9%, far exceeding the next best method (77.9% by SNTK).

Table 5.5: Generalization of the condensed graph data. We evaluate the generalization of condensed graph data by training various GNNs on data derived from different methods. Model performance is assessed through test set classification accuracy.

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

In addition to accuracy, GCGP also excels in computational efficiency. As shown in Table 5.7, GCGP dramatically reduces condensation time compared to SNTK and LightGNTK. Notably, on large-scale datasets like Ogbn-arxiv and Reddit, GCGP achieves over 8× and 6× speedups respectively compared to LightGNTK, and over 30× and 28× speedups compared to SNTK. This efficiency gain is crucial for scaling to large graphs.

In summary, GCGP demonstrates clear advantages in both performance and efficiency. Its covariance function not only captures graph structure and semantics more effectively but also ensures faster condensation, making it highly practical for real-world graph learning tasks.

Table 5.6: Impact of Covariance Function Ablation on Accuracy Performance.

Dataset	**Acc ± Std.**				
	Dot Product	**NTK**	**SNTK**	**LightGNTK**	**GCGP**
Cora (70)	57.3±0.5	59.1±1.4	82.4±0.5	81.6±0.4	**82.5±0.7**
Citeseer (60)	60.9±1.2	62.7±0.6	67.0±0.3	71.1±1.1	**72.8±0.6**
Pubmed (30)	72.7±0.4	69.9±1.2	**79.3±0.3**	78.8±0.5	79.2±0.7
Photo (80)	87.6±1.9	82.9±0.3	88.5±0.2	89.4±0.8	**92.0±0.1**
Computers (100)	80.7±1.2	72.5±0.9	83.1±0.6	83.5±0.5	**86.4±0.3**
Ogbn-arxiv (90)	50.6±0.5	51.3±0.1	64.2±0.2	60.5±0.4	**65.7±0.3**
Reddit (77)	66.3±0.1	66.3±0.2	77.9±0.9	71.8±1.5	**93.9±0.1**

5.4.8 Generalization Across Different Models

To evaluate whether the graph data condensed by GCGP retains essential structural and semantic information for downstream tasks, we apply the condensed data to train a variety of GNN models. This setting allows us to assess the generalization ability of GCGP beyond the model used during condensation. The experimental results show that models trained on GCGP-condensed graphs consistently achieve competitive

Table 5.7: The Time Comparison of Different Covariance Functions.

Dataset	Time (s)		
	SNTK	**LightGNTK**	**GCGP**
Cora (70)	5.06	1.31	**0.84**
Citeseer (60)	4.71	4.56	**0.74**
Pubmed (30)	4.78	2.45	**1.00**
Photo (80)	5.51	1.52	**1.14**
Computers (100)	5.22	1.87	**1.41**
Ogbn-arxiv (90)	871.52	772.39	**97.25**
Reddit (77)	3844.36	854.96	**135.68**

performance, demonstrating the robustness and transferability of the condensed data to different GNN architectures.

Table 5.5 summarizes the generalization performance of graph data condensed by various methods—GCond, GC-SNTK, and the proposed GCGP—on multiple GNN models, including GCN [43], SGC [104], APPNP [24], GraphSAGE [28], and KRR [89], 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 results 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.

5.5 Conclusion

In this paper, we propose a novel GP-based framework for graph data condensation, addressing the 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 and represents local structures with minimal computational overhead. We propose using the binary concrete relaxation to optimize the adjacency matrix. This approach relaxes the discrete adjacency matrix into a differentiable one,

enabling the efficient optimization of the 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.

Chapter 6

Conclusion and Future Work

This thesis presents a comprehensive framework for efficient graph condensation through three interconnected contributions that address fundamental computational bottlenecks in graph neural networks. Our work demonstrates how kernel methods and probabilistic approaches can significantly accelerate graph analysis while maintaining or improving prediction performance. In this concluding chapter, we summarize our key contributions, discuss their broader implications, and outline promising directions for future research.

6.1 Summary of Contributions

Our thesis makes three major contributions that progressively build upon each other to create an efficient and scalable framework for graph condensation:

- **Chapter 3: Fast Graph Condensation with Structure-based Neural Tangent Kernels (GC-SNTK).** Our first contribution establishes the kernel-based foundation for efficient graph condensation. By reformulating bi-level optimization as Kernel Ridge Regression (KRR), we eliminated iterative GNN

training while introducing the Structure-based Neural Tangent Kernel (SNTK) to capture both node features and topological information. This approach achieved significant efficiency improvements while maintaining competitive accuracy, demonstrating the viability of kernel methods for graph condensation.

- **Chapter 4: Simplifying Graph Neural Kernels (SGNK).** Building upon the kernel foundation, our second contribution addresses computational redundancy in traditional GNTK methods. We proposed two complementary approaches: SGTK replaces iterative layer stacking with continuous K-step message passing, while SGNK models infinite-width networks as Gaussian Processes. These simplifications achieved up to 53.7x speedup while maintaining excellent classification accuracy, bridging theoretical foundations with practical applications.

- **Chapter 5: Efficient Graph Condensation via Gaussian Process (GCGP).** Our final contribution represents the culmination of our work, introducing the GCGP framework that replaces iterative training with Gaussian Process inference. By leveraging a specialized covariance function and concrete random variables, GCGP achieved unprecedented efficiency improvements (up to 170x faster) while maintaining competitive accuracy. This unified approach demonstrates the synergistic potential of combining kernel methods with probabilistic inference for large-scale graph learning.

6.2 Broader Impact and Implications

The contributions of this thesis extend beyond the specific techniques developed, offering several broader implications for the field of graph learning. Our work establishes a unified theoretical framework that connects kernel methods, neural tangent kernels, and Gaussian processes in the context of graph condensation. This frame-

work provides a principled understanding of how different computational paradigms can be leveraged for efficient graph analysis. The theoretical insights developed in this thesis can inform future research in related areas such as graph representation learning, kernel methods for graphs, and probabilistic approaches to graph analysis.

The efficiency improvements achieved by our methods have significant practical implications for real-world applications. Large-scale graph datasets are increasingly common in domains such as social networks, biological networks, and knowledge graphs. Our methods enable the analysis of such datasets with substantially reduced computational requirements, making graph machine learning more accessible to researchers and practitioners with limited computational resources.

6.3 Limitations and Future Research Directions

While our methods achieve significant improvements in efficiency and performance, several limitations and challenges remain. Despite substantial improvements, our methods still face computational challenges when applied to extremely large graphs (e.g., graphs with millions of nodes). The kernel matrix computation in GC-SNTK and SGNK scales quadratically with the number of nodes, while the covariance function evaluation in GCGP can become expensive for very large graphs. Our methods rely on several theoretical assumptions, such as the infinite-width limit for neural networks and the Gaussian process approximation. While these assumptions are well-motivated and empirically validated, their validity in all scenarios remains an open question. Additionally, our current work primarily focuses on homogeneous graphs with node classification tasks. The extension to heterogeneous graphs, multi-relational graphs, and other graph mining tasks requires further investigation.

6.3.1 Scalability and Computational Complexity

Although graph condensation significantly reduces data size, the underlying kernel computations impose nontrivial computational demands. In particular, the construction and storage of kernel matrices typically incur quadratic complexity with respect to the number of nodes ($O(n^2)$), which limits scalability to extremely large graphs such as web-scale social or biological networks. To mitigate these issues, future work could adopt several strategies: (1) leveraging low-rank kernel approximations such as the Nyström method or random Fourier features to accelerate computation; (2) designing hierarchical or subgraph-based condensation schemes that process subsets of large graphs independently before aggregation; and (3) developing distributed or GPU-parallel implementations of kernel evaluation and condensation optimization. These advances would make the proposed frameworks more amenable to real-world industrial-scale graph systems.

6.3.2 Practical Deployment Challenges and Potential Remedies

When transitioning from controlled experimental settings to real-world graph applications, several challenges may arise. Many real-world graphs are dynamic, noisy, and structurally heterogeneous, which deviates from our current assumption of static and homogeneous graphs. Extending graph condensation methods to handle temporally evolving graphs and multiple node or edge types is thus a necessary step toward wider applicability. Another challenge concerns kernel selection: the choice of kernel function significantly influences condensation quality and model expressiveness. Future work could explore adaptive or task-specific kernel learning strategies to automatically tailor kernel properties to different graph types. Furthermore, privacy and interpretability are important considerations when synthetic graphs are deployed in sensitive domains such as healthcare or finance. Integrating differential privacy tech-

niques and interpretable graph embeddings could ensure responsible and transparent use of graph condensation frameworks in practice.

6.3.3 Directions for Future Research

Based on the contributions and limitations identified in this thesis, several promising directions can be envisioned. Future work could explore approximate kernel methods, such as random feature approximations or Nyström approximations, to further reduce computational complexity. These approaches could enable the application of our methods to graphs with millions or billions of nodes. Developing distributed versions of our algorithms could enable the analysis of graphs that exceed the memory capacity of single machines.

Extending our methods to heterogeneous graphs, which contain multiple types of nodes and edges, would significantly broaden their applicability. This would require developing kernel functions and covariance functions that can handle multiple node and edge types. Many real-world graphs evolve over time, with nodes and edges being added or removed. Developing methods for efficient condensation of dynamic graphs would enable the analysis of temporal graph data. Extending our methods to multi-task learning scenarios, where multiple prediction tasks are performed simultaneously, could improve the efficiency of multi-task graph analysis.

While our methods replace traditional GNN training with kernel methods, future work could explore hybrid approaches that combine the strengths of both paradigms. This could involve using kernel methods for initial feature extraction followed by deep learning for fine-tuning. Integrating our methods with reinforcement learning could enable adaptive condensation strategies that optimize the condensed graph based on the specific requirements of downstream tasks.

Conducting experiments on larger and more diverse datasets would provide further validation of our methods' effectiveness and identify areas for improvement. Applying

our methods to real-world applications in domains such as social network analysis, biological network analysis, and knowledge graph mining would demonstrate their practical utility and identify specific challenges for future research.

6.4 Final Remarks

This thesis represents a significant step toward efficient and scalable graph machine learning. By developing a unified framework that combines kernel methods, neural tangent kernels, and Gaussian processes, we have demonstrated that substantial efficiency improvements can be achieved without sacrificing performance. The theoretical insights and practical algorithms developed in this work provide a foundation for future research in graph condensation and related areas.

The journey from traditional bi-level optimization to efficient kernel-based and probabilistic approaches illustrates the value of interdisciplinary research that draws from multiple computational paradigms. As graph data continues to grow in scale and complexity, the need for efficient analysis methods will only increase. We hope that the contributions of this thesis will inspire future research that continues to push the boundaries of what is possible in graph machine learning.

References

[1] Thomas Aichner, Matthias Grünfelder, Oswin Maurer, and Deni Jegeni. Twenty-five years of social media: a review of social media applications and definitions from 1994 to 2019. *Cyberpsychology, behavior, and social networking*, 24(4):215–222, 2021.

[2] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

[3] Chen Avin. Distance graphs: From random geometric graphs to bernoulli graphs and between. In *Proceedings of the fifth international workshop on Foundations of mobile computing*, pages 71–78, 2008.

[4] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006.

[5] Per Block, Marion Hoffman, Isabel J Raabe, Jennifer Beam Dowd, Charles Rahal, Ridhi Kashyap, and Melinda C Mills. Social network-based distancing strategies to flatten the covid-19 curve in a post-lockdown world. *Nature Human Behaviour*, 4(6):588–596, 2020.

[6] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[7] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

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

[9] Deepayan Chakrabarti and Christos Faloutsos. *Graph mining: laws, tools, and case studies*. Springer Nature, 2022.

[10] Junhan Chen, Yuan Wang, et al. Social media use for health purposes: systematic review. *Journal of medical Internet research*, 23(5):e17917, 2021.

[11] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

[12] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

[13] Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. *Advances in Neural Information Processing Systems*, 35:34391–34404, 2022.

[14] Mi Diao, Hui Kong, and Jinhua Zhao. Impacts of transportation network companies on urban mobility. *Nature Sustainability*, 4(6):494–500, 2021.

[15] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *International Conference on Machine Learning*, pages 5378–5396. PMLR, 2022.

[16] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph

neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

[17] Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. A comprehensive study on large-scale graph training: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 35:5376–5389, 2022.

[18] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. Graph trend filtering networks for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–121, 2022.

[19] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[20] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering*, 34(5):2033–2047, 2020.

[21] Wenqi Fan, Xiangyu Zhao, Xiao Chen, Jingran Su, Jingtong Gao, Lin Wang, Qidong Liu, Yiqi Wang, Han Xu, Lei Chen, et al. A comprehensive survey on trustworthy recommender systems. *arXiv preprint arXiv:2209.10117*, 2022.

[22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[23] Xinyi Gao, Guanhua Ye, Tong Chen, Wentao Zhang, Junliang Yu, and Hongzhi Yin. Rethinking and accelerating graph condensation: A training-free approach

with class partition. In *Proceedings of the ACM on Web Conference 2025*, pages 4359–4373, 2025.

[24] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[25] Eugene Golikov, Eduard Pokonechnyy, and Vladimir Korviakov. Neural tangent kernel: A survey. *arXiv preprint arXiv:2208.13614*, 2022.

[26] Shengbo Gong, Mohammad Hashemi, Juntong Ni, Carl Yang, and Wei Jin. Scalable graph condensation with evolving capabilities. *arXiv preprint arXiv:2502.17614*, 2025.

[27] Yu Gu, Xiao Fu, Zhiyuan Liu, Xiangdong Xu, and Anthony Chen. Performance of transportation network under perturbations: Reliability, vulnerability, and resilience. *Transportation Research Part E: Logistics and Transportation Review*, 133:101809, 2020.

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

[29] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[30] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[31] Jilin Hu, Jianbing Shen, Bin Yang, and Ling Shao. Infinitely wide graph convolutional networks: semi-supervised learning via gaussian processes. *arXiv preprint arXiv:2002.12168*, 2020.

[32] Wei Hu, Zhiyuan Li, and Dingli Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. *arXiv preprint arXiv:1905.11368*, 2019.

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

[34] 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. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[35] 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. *arXiv preprint arXiv:2005.00687*, 2020.

[36] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[37] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[38] Shunhua Jiang, Yunze Man, Zhao Song, Zheng Yu, and Danyang Zhuo. Fast graph neural tangent kernel via kronecker sketching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7033–7041, 2022.

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

[40] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580*, 2021.

[41] Chuanze Kang, Han Zhang, Zhuo Liu, Shenwei Huang, and Yanbin Yin. Lr-gnn: A graph neural network based on link representation for predicting molecular associations. *Briefings in Bioinformatics*, 23(1):bbab513, 2022.

[42] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. Inform: Individual fairness on graph mining. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 379–389, 2020.

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

[44] Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 291–298, 2012.

[45] Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

[46] Sanjukta Krishnagopal and Luana Ruiz. Graph neural tangent kernel: Convergence on large graphs. In *International Conference on Machine Learning*, pages 17827–17841. PMLR, 2023.

[47] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[48] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

[49] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[50] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

[51] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In *International Conference on Machine Learning*, pages 12352–12364. PMLR, 2022.

[52] Can Li, Lei Bai, Wei Liu, Lina Yao, and S Travis Waller. Graph neural network for robust public transit demand prediction. *IEEE Transactions on Intelligent Transportation Systems*, 23(5):4086–4098, 2020.

[53] Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

[54] Cai-zhi Liu, Yan-xiu Sheng, Zhi-qiang Wei, and Yong-Quan Yang. Research of text classification based on improved tf-idf algorithm. In *2018 IEEE international conference of intelligent robotic and control engineering (IRCE)*, pages 218–222. IEEE, 2018.

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

[56] Yunbo Long, Liming Xu, and Alexandra Brintrup. Efficient and privacy-preserved link prediction via condensed graphs. *arXiv preprint arXiv:2503.12156*, 2025.

[57] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2021.

[58] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1202–1211, 2021.

[59] David JC MacKay et al. Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166, 1998.

[60] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[61] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[62] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[63] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pages 4556–4565. PMLR, 2019.

[64] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[65] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050*, 2020.

[66] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021.

[67] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[68] Wisam A Qader, Musa M Ameen, and Bilal I Ahmed. An overview of bag of words; importance, implementation, applications, and challenges. In *2019 international engineering conference (IEC)*, pages 200–204. IEEE, 2019.

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

[70] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in neural information processing systems*, 33:12559–12571, 2020.

[71] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

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

[73] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[74] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[75] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[76] Xiangguo Sun, Hong Cheng, Hang Dong, Bo Qiao, Si Qin, and Qingwei Lin. Counter-empirical attacking based on adversarial reinforcement learning for time-relevant scoring system. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):5849–5859, 2023.

[77] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2131, 2023.

[78] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. Heterogeneous hypergraph embedding for graph classification. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 725–733, 2021.

[79] Yehui Tang and Junchi Yan. Graphqntk: Quantum neural tangent kernel for graph data. *Advances in Neural Information Processing Systems*, 35:6104–6118, 2022.

[80] Shu-Feng Tsao, Helen Chen, Therese Tisseverasinghe, Yang Yang, Lianghua Li, and Zahid A Butt. What social media told us in the time of covid-19: a scoping review. *The Lancet Digital Health*, 3(3):e175–e194, 2021.

[81] Nikolaos Tsilivis and Julia Kempe. What can the neural tangent kernel tell us about adversarial robustness? *Advances in Neural Information Processing Systems*, 35:18116–18130, 2022.

[82] Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023.

[83] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[84] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[85] Philippe Verduyn, Nino Gugushvili, Karlijn Massar, Karin Täht, and Ethan Kross. Social comparison on social networking sites. *Current opinion in psychology*, 36:32–37, 2020.

[86] Paul Vicol, Luke Metz, and Jascha Sohl-Dickstein. Unbiased gradient estimation in unrolled computation graphs with persistent evolution strategies. In *International Conference on Machine Learning*, pages 10553–10563. PMLR, 2021.

[87] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[88] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

[89] Vladimir Vovk. Kernel ridge regression. *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pages 105–116, 2013.

[90] Lilapati Waikhom and Ripon Patgiri. A survey of graph neural networks in various learning paradigms: methods, applications, and challenges. *Artificial Intelligence Review*, 56(7):6295–6364, 2023.

[91] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205, 2022.

[92] 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 2024*, pages 4439–4448, 2024.

[93] Ning Wang, Minnan Luo, Kaize Ding, Lingling Zhang, Jundong Li, and Qinghua Zheng. Graph few-shot learning with attribute matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1545–1554, 2020.

[94] Shijie Wang, Wenqi Fan, Xiao-Yong Wei, Xiaowei Mei, Shanru Lin, and Qing Li. Multi-agent attacks for black-box social recommendations. *ACM Transactions on Information Systems*, 43(1):1–26, 2024.

[95] Shijie Wang, Jiani Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Wenqi Fan, Hui Liu, Xiaorui Liu, Dawei Yin, et al. Graph machine learning in the era of large language models (llms). *ACM Transactions on Intelligent Systems and Technology*, 2024.

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

[97] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3):279–287, 2022.

[98] Lanning Wei, Huan Zhao, Zhiqiang He, and Quanming Yao. Neural architecture search for gnn-based graph classification. *ACM Transactions on Information Systems*, 42(1):1–29, 2023.

[99] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128, 2009.

[100] Max Welling. Kernel ridge regression. *Max Welling's classnotes in machine learning*, pages 1–3, 2013.

[101] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

[102] Christopher Williams. Computing with infinite networks. *Advances in neural information processing systems*, 9, 1996.

[103] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

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

[105] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(10):4753–4766, 2020.

[106] Zhihao Wu, Zhao Zhang, and Jicong Fan. Graph convolutional kernel machine versus graph convolutional networks. *Advances in neural information processing systems*, 36, 2024.

[107] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[108] Feng Xia, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, and Xiangjie Kong. Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2):95–107, 2019.

[109] 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 2025*, pages 4494–4506, 2025.

[110] Zhenbang Xiao, Yu Wang, Shunyu Liu, Huiqiong Wang, Mingli Song, and Tongya Zheng. Simple graph condensation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 53–71. Springer, 2024.

[111] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[112] 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 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2850–2861, 2023.

[113] Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1365–1374, 2015.

[114] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

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

[116] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[117] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in genetics*, 12:690049, 2021.

[118] Yanfu Zhang, Shangqian Gao, Jian Pei, and Heng Huang. Improving social network embedding via new second-order continuous graph neural networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 2515–2523, 2022.

[119] Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression. In *Conference on learning theory*, pages 592–617. PMLR, 2013.

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

[121] Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. Retgk: Graph kernels based on return probabilities of random walks. *Advances in Neural Information Processing Systems*, 31, 2018.

[122] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021.

[123] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023.

[124] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

[125] 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. *arXiv preprint arXiv:2306.02664*, 2023.

[126] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2357–2360, 2019.

[127] Fan Zhou, Qing Yang, Ting Zhong, Dajiang Chen, and Ning Zhang. Variational graph neural networks for road traffic prediction in intelligent transportation systems. *IEEE Transactions on Industrial Informatics*, 17(4):2802–2812, 2020.

[128] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[129] Xianchen Zhou and Hongxia Wang. On the explainability of graph convolutional network with gcn tangent kernel. *Neural Computation*, 35(1):1–26, 2023.