MONASH University

Automated Graph Machine Learning Operations (MLOps) Workflow

Xin Zheng

Doctor of Philosophy

A Thesis Submitted for the Degree of Doctor of Philosophy at

Monash University in 2024

Faculty of Information Technology

Department of Software Systems & Cybersecurity

Copyright notice

Abstract

Graph machine learning (ML) with graph neural networks (GNNs) has achieved great success in modeling and exploiting graph structural data for many real-world scenarios. Despite the development of extensive GNN models designed for various graph ML tasks from a research standpoint, existing efforts have yet to effectively solve the challenge of developing and deploying GNN models that are ready for industrial applications. Real-world applicable graph ML should have a systematic, automated, and operationalized workflow, encompassing both data-level and model-level components to support a wide range of applications and tasks related to learning from graph data.

This thesis delves into the systematic workflow of graph machine learning operations (MLOps) in an automated paradigm, *i.e.*, **Auto-GMLOps**. It concentrates on three key phases in the automated graph MLOps workflow: (1) graph data engineering, (2) automated GNN model design, and (3) GNN model deployment. The goal is to tackle the challenge of developing and deploying production-ready GNNs for real-world graph learning scenarios.

First, for graph data engineering, this thesis explores two aspects, graph data types and graph data scales. It begins by exploring and understanding the characteristics of complex graph data types, with a particular focus on multi-relational graphs and heterophilic graphs. Then, it discovers the effects of graph data scales on GNN model development and proposes to condense large-scale graphs into smaller-scale graph substitutes for more efficient graph learning.

Second, for automated GNN model design, this thesis explores the graph neural architecture search (NAS) technique for automated model development, on multi-relational and heterophilic graphs respectively. It proposes to customize GNN models driven by specific graph data characteristics and graph learning tasks, by effectively designing graph NAS search spaces and search strategies.

Finally, for GNN model deployment, this thesis focuses on gaining a thorough understanding and conducting accurate performance evaluations of existing well-trained GNNs, when they are deployed in real-world online graph ML services. The main challenge lies in the difficulty in achieving effective GNN generalization, especially when there exist potential graph data distribution shifts between the training and test graphs. It proposes a GNN model evaluator to precisely estimate the well-trained GNN performance for online service, offering valuable insights for deploying these models on real-world unseen test graphs with distribution shifts.

All the phases covered in this thesis play important roles in building a comprehensive and systematic automated graph MLOps workflow, facilitating the analysis and understanding of diverse and complex graph data, as well as benefiting the development and deployment of expressive automated GNNs in a wide range of real-world graph learning scenarios.

Publications during enrolment

Publications included in this thesis:

(1) **Zheng, X.**, Zhang, M., Chen, C., Molaei, S., Zhou, C., & Pan, S. (2023). GNNEvaluator: Evaluating GNN Performance On Unseen Graphs Without Labels. Advances in Neural Information Processing Systems (NeurIPS), 2023. [CORE ranked A*]

(2) **Zheng, X.**, Zhang, M., Chen, C., Nguyen, Q. V. H., Zhu, X., & Pan, S. (2023). Structure-free Graph Condensation: From Large-scale Graphs to Condensed Graph-free Data. Advances in Neural Information Processing Systems (NeurIPS), 2023. [CORE ranked A*]

(3) **Zheng, X.**, Zhang, M., Chen, C., Zhang, Q., Zhou, C., & Pan, S. (2023). Auto-HeG: Automated Graph Neural Network on Heterophilic Graphs. Proceedings of the ACM Web Conference (WWW), 2023. [CORE ranked A*]

(4) **Zheng, X.**, Zhang, M., Chen, C., Li, C., Zhou, C., & Pan, S. (2022). Multi-Relational Graph Neural Architecture Search with Fine-grained Message Passing. IEEE International Conference on Data Mining (ICDM), 2022. [CORE ranked A*]

Other publications not included in this thesis:

(5) **Zheng, X.**, Song, D., Wen, Q., Du, B., & Pan, S. (2024). Online GNN Evaluation Under Test-time Graph Distribution Shifts. International Conference on Learning Representations (ICLR), 2024. [CORE ranked A*]

(6) **Zheng, X.**, Liu, Y., Bao, Z., Fang, M., Hu, X., Liew, A. W. C., & Pan, S. (2023). Towards Data-centric Graph Machine Learning: Review and Outlook. arXiv preprint arXiv:2309.10979.

(7) Wu, M., **Zheng, X.**, Zhang, Q., Shen, X., Luo, X., Zhu, X., & Pan, S. (2024). Graph Learning under Distribution Shifts: A Comprehensive Survey on Domain Adaptation, Out-of-distribution, and Continual Learning. arXiv preprint arXiv:2402.16374.

(8) Uddamvathanak, R., **Zheng, X.**, & Pan, S. (2022). Joint Graph-Sequence Learning for Molecular Property Prediction. In IEEE International Joint Conference on Neural Networks (IJCNN), 2022. [Core B]

Thesis including published works declaration

I hereby declare that this thesis contains no material that has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis. This is a general declaration that Chapter 3 contains the student's original work (submitted but not yet accepted: **Zheng, X.**, Wang, Y., Liu, Y., Li, M., Zhang, M., Jin, D., & Yu, P. S., Pan, S., (2022). Graph Neural Networks for Graphs with Heterophily: A Survey. Submitted to the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2024. [JCR Q1]) to provide a background narrative closely related to the core theme of the thesis.

This thesis includes 4 original papers published in top-tier conferences. The core theme of the thesis is 'Automated Graph Machine Learning Operations (MLOps) Workflow'. The ideas, development, and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the Faculty of Information Technology, Department of Software Systems & Cybersecurity, at Monash University, Clayton Campus, under the supervision of Prof. Chunyang Chen and Prof. Shirui Pan. In the case of Chapter 4, Chapter 5, Chapter 6, and Chapter 7, my contributions to the work involved the following:

Thesis Chapter	**Publication Title**	**Status**	**Nature and % of Student Contribution**	**Co-author Names Nature and % of Student Contribution**	**Co-authors Monash Student**
4	Structure-free Graph Condensation: From Large-scale Graphs to Condensed Graph-free Data	Accepted	65%. Concept, methodology design, implementation, experiment & analysis, investigation, and writing the manuscript	1) Zhang, M., Conceptualisation and supervision 4% 2) Chen, C., Supervision 3% 3) Nguyen, Q. V. H., Supervision 3% 4) Zhu, X., Conceptualisation, supervision, and revising the manuscript 12% 5) Pan, S., Conceptualisation, supervision, and revising the manuscript 13%	1) No 2) No 3) No 4) No 5) No

5	Auto-HeG: Automated Graph Neural Network on Heterophilic Graphs	Accepted	65%. Concept, methodology design, implementation, experiment & analysis, investigation, and writing the manuscript	1) Zhang, M., Conceptualisation, domain knowledge, supervision, and revising the manuscript 15% 2) Chen, C., Conceptualisation and supervision 5% 3) Zhang, Q., Conceptualisation and supervision 5% 4) Zhou, C., Supervision 5% 5) Pan, S., Conceptualisation, supervision, and domain knowledge 5%	1) No 2) No 3) No 4) No 5) No
6	Multi-Relational Graph Neural Architecture Search with Fine-grained Message Passing	Accepted	65%. Concept, methodology design, implementation, experiment & analysis, investigation, and writing the manuscript	1) Zhang, M., Conceptualisation, domain knowledge, supervision, and revising the manuscript 15% 2) Chen, C., Conceptualisation and supervision 5% 3) Li, C., Supervision 5% 4) Zhou, C., Supervision 5% 5) Pan, S., Conceptualisation, supervision and domain knowledge 5%	1) No 2) No 3) No 4) No 5) No

7	GNNEvaluator: Evaluating GNN Performance On Unseen Graphs Without Labels	Accepted	65%. Concept, methodology design, implementation, experiment & analysis, investigation, and writing the manuscript	1) Zhang, M., Conceptualisation and supervision 5% 2) Chen, C., Conceptualisation and supervision 5% 3) Molaei, S., Supervision 5% 4) Zhou, C., Supervision 5% 5) Pan, S., Conceptualisation, supervision, domain knowledge, and revising the manuscript 15%	1) No 2) No 3) No 4) No 5) No

I have renumbered sections of submitted or published papers in order to generate a consistent presentation within the thesis.

Student name: Xin Zheng

Student signature: Xin Zheng Date: 22 Aug 2024

I hereby certify that the above declaration correctly reflects the nature and extent of the student's and co-authors' contributions to this work. In instances where I am not the responsible author I have consulted with the responsible author to agree on the respective contributions of the authors.

Main Supervisor name: Shirui Pan

Main Supervisor signature: **Date: 23 Aug 2024**

Shirui Pan

Acknowledgements Sincere thanks to my supervisors Prof. Shirui Pan, and Prof. Chunyang Chen, for your mentorship in the expansive realm of academia and research.

Warm thanks to all my collaborators, and team members in Shirui's GRAND research group, for your excellent domain expertise and valuable suggestions for all my research works.

Grateful appreciation to all my lifelong friends, Xin Lei, Yu Pang, Ruiyan Xu, Ziqi Zhao, Jiaxin Ju, Tina Wu, Tira Wei, He Zhang, Yicheng Wu, and Han Wang, for supporting me throughout my Ph.D. journey.

Heartfelt thanks to Jinx, for joining me in the adventure of finding myself and creating our own life.

Earnest thanks to my Grandma and my brother Fangyu, for always encouraging me to be true to myself.

Deepest thanks to Mom and Dad, for the best idea ever in the world of bringing me to our home.

To the beloved cats, Kele and Wangzai, and the adorable dog, Qixi, you are the most cherished companions in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Graph structural data, originating from non-Euclidean domains and represented by complex relationships and inter-dependencies between objects, has garnered significant attention in the realm of modern machine learning tasks [4, 5]. In contrast to images, text, and speech, which typically exhibit fixed forms akin to fixed-size grid graphs, sequences, and line graphs, most graph structural data is intricate and varied. It consists of variable-sized sets of unordered nodes and an unpredictable number of connected neighbors [6], posing considerable challenges for accurate learning and exploration. In this case, graph machine learning (ML) has emerged with great success for many real-world scenarios, such as social networks [7, 8], knowledge bases [9, 10], chemical molecules [11–13], and recommendation systems [14, 15], as shown in Fig. 1.1.

At the data level, real-world graph structural data generally comprises numerous nodes and edges, reflecting diverse node attributes and complex structural connections. And this leads to various graph structural data types with diversity and complexity, for example, single relational graphs [7, 16] and multi-relational graphs [2, 17], homophilic graphs [7, 18] and heterophilic graphs [19, 20], which pose severe challenges to effective graph-structured data learning. At the model level, various graph ML models are delicately well-designed, to model different types of graph-structured data for different application tasks.

Graph neural networks (GNNs), as prominent models in graph ML, have attained remarkable success attributed to their potent learning capabilities [5, 19, 21–25], as shown in Fig. 1.2. The problem-solving capabilities of GNNs encompass a broad spectrum of practical graph learning tasks in both industrial and academic domains. For example, in drug discovery and cheminformatics, GNNs excel at modeling molecules, which are natural graphs. They are employed for predicting molecular activities and properties through the graph classification task [12]. Within social networks, GNNs infer social

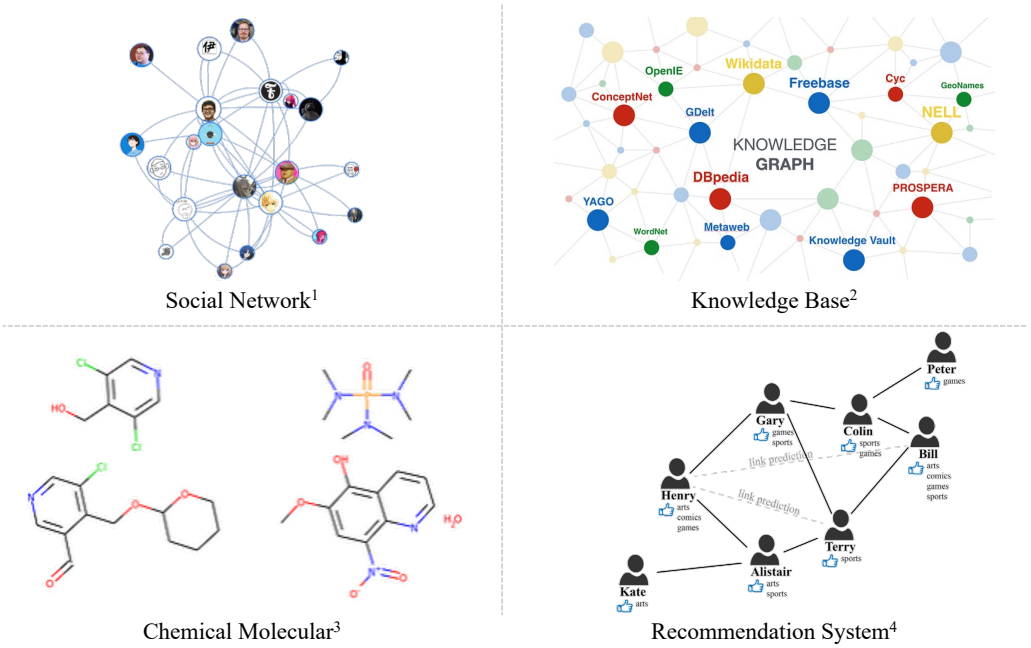

FIGURE 1.1: Real-world graph structural data in various application scenarios.
Image sources: 1-https://www.nebula-graph.io/posts/social-networks-with-graph-database-1;
2-https://www.linkedin.com/pulse/unleashing-power-knowledge-graphs-elevating-digital-marketing-masai/;
3-https://discuss.dgl.ai/t/understanding-attention-in-graph-networks-for-chemical-structures/464;
4-https://aws.amazon.com/blogs/machine-learning/graph-based-recommendation-system-with-neptune-ml-an-illustration-on-social-network-link-prediction-challenges/

interactions and suggest potential connections among users [26]. Facebook, for example, extensively depends on GNNs for such tasks. Also, in recommendation systems, GNNs have been widely used in predicting criminal associations through the link prediction task. And Google Maps uses a GNN-based architecture for traffic networks in route optimization [6]. In academia, significant efforts have been directed towards comprehensively understanding graph-level, node-level, and edge-level properties, designing and customizing GNN architectures appropriately, and exploring generative approaches for facilitating graph data diversity, knowledge graph construction, and extension.

In recent years, many researchers have focused extensively on customizing GNN model architectures with adequate expertise and efforts for well-performed GNN model design. Then, well-designed GNNs are expected to be instrumental in modeling and analyzing diverse real-world graph structural data across wide-range practical scenarios. However, despite the great success of current graph ML studies with massive productive GNNs from the research perspective, they still fail to meet the practical requirements of building production-ready graph learning models for industrial applications. Real-world applicable graph ML should have a systematic, automated, and operationalized workflow, with both data-level and model-level components for a wide range of applications and tasks for graph data learning. The reasons behind this mainly lie in the following aspects:

- From the perspective of graph data types, most existing works design GNNs for single-relational graphs and under the homophily property assumption, lacking the comprehensive modeling and analysis of various types of graph structural data with different characteristics;

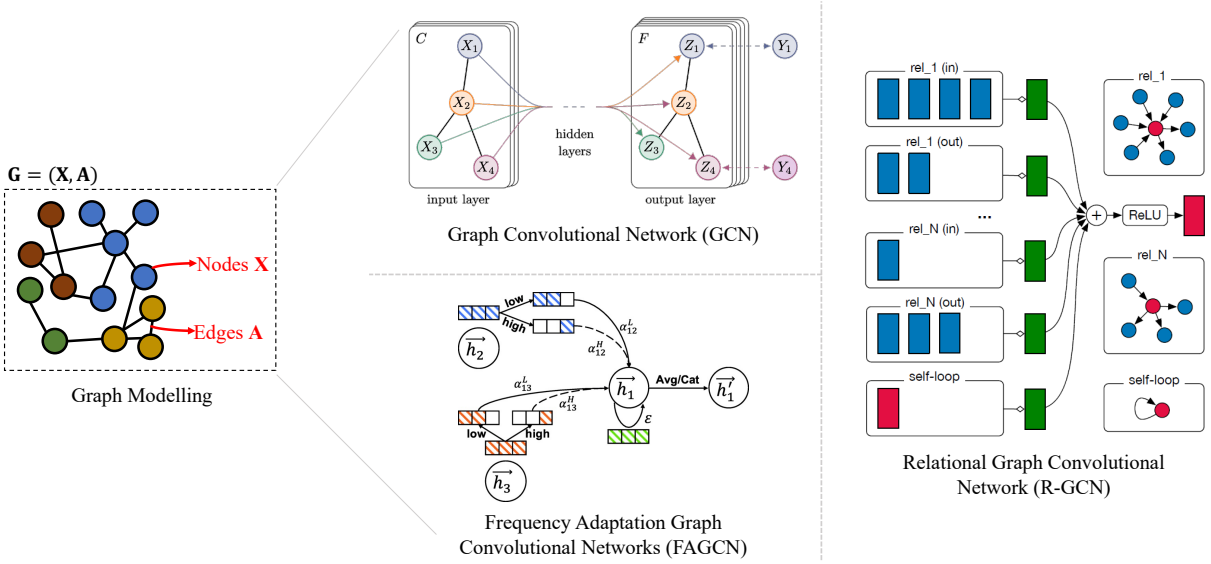

FIGURE 1.2: Modelling graph structural data with expressive graph neural networks (GNNs): GCN [1] for single-relational graphs; R-GCN [2] for multi-relational graphs; and FAGCN [3] for heterophilic graphs.

- From the perspective of graph data scales, developing well-performed GNN models is required to conduct repeat training and parameter tuning on large-scale graph training sets, incurring severe storage, computation, and memory costs;
- From the perspective of GNN model design, most existing GNN models are human-designed manually, highly relying on expert knowledge with much human effort cost, in the meantime, the constructed GNN architectures are constrained in limited design space with limited expertise, resulting in suboptimal performance for specific graph data types and graph learning tasks.
- From the perspective of GNN model deployment, existing methods mainly focus on developing new GNN models and overlook the practical applications of well-trained GNN models for serving and inferring on real-world test graphs unseen in the training stage. That means, existing works can not understand how the well-designed and well-trained GNNs would perform on real-world unseen test graphs, and estimating the GNNs' performance in such a scenario is a vital step of GNN model deployment and service.

These unexplored issues motivate this research to explore the systematic workflow of graph machine learning operations (MLOps) in an automated paradigm, *i.e.*, Auto-GMLOps. To begin with, this thesis provides a detailed analysis of the challenges of building an automated graph MLOps workflow as follows:

Challenge #1. Real-world graph data usually involves various types with diverse properties and complex structures, and at the graph data type level, comprehensively understanding the diverse and complex characteristics of various graph-structured data types is the first challenge, considering such

data characteristics play important roles in GNN model designing for effective learning and inferring on practical graph structural data.

Challenge #2. As the graph data scales increase, learning and analyzing various graph structural data with millions of nodes and edges become more difficult, when heavy storage, computation, and memory costs will hinder the development progress of GNN models on large-scale graph data. Hence, at the graph data scale level, significantly reducing the size of graph data and meanwhile preserving its representative information for training GNN models is another challenge. The GNNs trained on the small-scale graph are expected to have comparable test performance with those trained on the original large-scale graph.

Challenge #3. To automatically design expressive and powerful GNN architectures driven by different graph data types, the graph neural architecture search (NAS) technique is an effective solution, which requires a well-designed search space with a well-adapted search strategy. Concretely, the graph NAS search space contains candidate GNN architecture components targeting specific graph types, and the graph NAS search strategy selects the optimal component combinations as the architecture driven by specific graph learning tasks. In light of this, how to design fine-grained search spaces and effective search strategies through graph NAS, with expressive GNN architectures for modeling various graph data types would be another challenge.

Challenge #4. To effectively deploy powerful and expressive GNN models serving real-world graph structural data, understanding and estimating the well-trained GNNs' performance on these practical graphs would be a critical step, especially when the in-service GNN models are usually trained on the fully-observed training graphs and they are required to make inferences on unseen test graphs, which are usually unlabeled and share different distributions with the training graphs. Hence, in practical GNN deployment of the automated graph MLOps workflow, evaluating in-service GNNs' performance on unseen test graphs under graph data distribution shifts is a critical challenge for understanding and utilizing well-developed GNN models.

All these challenges further underscore the necessity for an automated Graph MLOps workflow, hence, this thesis aims at:

Goal #1. Comprehensively understand the diverse and complex characteristics of graph structural data to design effective GNN models.

Goal #2. Develop strategies to manage the scale of graph data efficiently, preserving essential information (*i.e.*, performance on original test graphs) while minimizing storage, computation, and memory costs.

Goal #3. Employ graph neural architecture search (NAS) techniques to automate the design of expressive and powerful GNN architectures tailored to specific graph data types and learning tasks (**Goal #3-1** for heterophilic graphs, and **Goal #3-2** for multi-relational graphs).

Goal #4. Evaluate and understand the performance of GNN models on unseen, real-world test graphs to ensure their effectiveness and practicality in GNN deployment scenarios.

By exploring these aspects, this thesis aims to advance the field of automated graph machine learning, contributing to the in-depth graph data understanding, analysis, and modification, as well as the development of more efficient, scalable, and effective GNN modeling and deployment strategies.

1.2 Research Questions

This thesis considers three core stages of automated graph MLOps workflow, involving (1) graph data engineering, (2) automated GNN model design, and (3) GNN model deployment. The following research questions are explored to develop the systematic pipeline of automated graph MLOps workflow, leading to product-ready GNN models with comprehensive graph data understanding and analysis in real-world graph machine learning applications.

RQ1: How to fully explore the heterophilic graph structural data type while overcoming the limitations of the homophily assumption? (Challenge/Goal #1, Completed, Survey Work Published in Arxiv'2022 and Submit to IEEE TPAMI'2024)

The homophily assumption defines that nodes with similar features or same class labels are linked together. In contrast, heterophily, the property that linked nodes have dissimilar features and different class labels, does not attract equal attention but widely exists in the real world. Since heterophily stands the opposite of the homophily assumption, existing homophilic GNNs cannot be directly applied to heterophilic graphs. Hence, it is the first step to answer the above question by comprehensively understanding the properties of graphs with heterophily, so that the heterophily property can be further incorporated into the design of GNN models.

RQ2: How to reduce large graph data scales into small-scale and effective counterparts to benefit GNN model development? (Challenge/Goal #2, Completed, Published in Conference NeurIPS'2023)

Large-scale graphs have millions of nodes and edges with more diverse and complex characteristics, and developing and training GNN models on them would incur negative effects, *e.g.*, heavy storage, computation, and memory costs. In light of these, reducing the sizes of original large-scale graphs into small-scale counterparts is a feasible solution, where the key is to preserve representative information of original large-scale graphs in the small-scale graphs. In this case, the GNNs trained on the small-scale graphs have comparable test performance with those trained on the original large-scale graphs. Hence, distilling a large-scale graph into a small-scale counterpart as its substitution for training GNNs is the focus for answering this question.

RQ3-1: How to comprehensively incorporate the heterophilic graph characteristics into automated GNN model design with graph NAS? (Challenge/Goal #3-1, Completed, Published in Conference WWW'2023)

Since most current human-designed GNN architectures are designed under the homophily assumption, these models could not perform well on heterophilic graphs with much human effort cost. Based on the full exploration and understanding of the heterophily property from RQ1, how to incorporate the heterophily property into the whole pipeline of graph NAS schema, involving heterophily-specific search space design and heterophily-aware search strategy development, is the research focus of this thesis for automatically customizing heterophilic GNN models driven by heterophilic graphs.

RQ3-2: How to comprehensively incorporate the multi-relational graph characteristics into automated GNN model design with graph NAS? (Challenge/Goal #3-2, Completed, Published in Conference ICDM'2022)

As a more general class of graphs, multi-relational graphs (*e.g.*, knowledge graphs) that involve different relation types and edge directions are not fully explored, where node and relation representations are expected to be learned jointly. However, all existing multi-relational GNN models heavily rely on human design with expertise, causing massive manual enumerations and efforts to obtain proper network architectures. The automated learning technique of graph NAS has not benefited multi-relational graphs for relieving human efforts on the model design yet. Exploring the characteristics of the multi-relational graphs and further incorporating them into the search space of automated GNN model design under graph NAS is an essential research focus.

Connection between RQ3-1 and RQ3-2. In heterophilic graphs, connected nodes often have dissimilar features and different class labels. However, in real-world scenarios, multi-relational graphs can be a more general and complex graph type. The edges in multi-relational graphs have multiple types, representing various relationships among different entities. In multi-relational graphs, connected nodes may represent different entities, where each entity can be viewed as a distinct class or similar entities can also be grouped into a meta-class. Therefore, when moving from RQ3-1 (heterophilic graphs) to RQ3-2 (multi-relational graphs), the primary challenge for graph NAS lies in the increased complexity of edge types and connections, as well as the diverse edge connections among various node classes.

RQ4: How to understand and evaluate well-trained GNNs' performance when serving on practical unseen test graphs without labels? (Challenge/Goal #4, Completed, Published in Conference NeurIPS'2023)

The in-service GNN models are usually well-trained on the fully-observed training graphs, in the practical GNN model deployment stage of the automated graph MLOps workflow, they are usually required to make inferences on unseen test graphs without labels. The unseen test graphs usually lie in different distributions with observed training graphs, and under such a graph data distribution shift

scenario, evaluating in-service GNNs' performance on unseen test graphs without labels is a vital task. Hence, this research focuses on the GNN model evaluation problem, which aims to accurately estimate the well-trained GNNs' performance on unseen and unlabeled test graphs, so that the automated graph MLOps workflow could provide users the guidance or criterion to select optimal in-service GNNs to provide accurate predictions on users' test graphs.

1.3 Thesis Contribution

In this thesis, we explore the systematic workflow of graph machine learning operations (MLOps) in an automated paradigm, as shown in Fig. 1.3, incorporating (1) graph data engineering, (2) automated GNN model design, and (3) GNN model deployment, for designing and serving productive graph ML with GNNs in real-world graph learning scenarios.

First, for graph data engineering, this thesis answers RQ1 and RQ2 with the consideration of two aspects: graph data types and graph data scales, in Chapter 3 and Chapter 4, respectively. For graph data types, this thesis considers two types of complex but common real-world graph structural data, *i.e.*, heterophilic graphs and multi-relational graphs. For graph data scales, this thesis focuses on reducing the graph data scales with the graph condensation strategy for alleviating the storage and computation costs in industrial applications.

Second, for the automated GNN model design, this thesis answers RQ3-1 and RQ3-2 with the consideration of fine-grained search space design and specific search strategy development in Chapter 5 and Chapter 6, leading to the expressive automated GNN development, specifically driven by the incorporated characteristics of heterophilic graphs and multi-relational graphs.

Third, for the GNN model deployment, this thesis answers RQ4 with the consideration of understanding and estimating well-designed and well-trained GNNs' performance in Chapter 7, when they are deployed and served in practical real-world test graphs that they never encounter in the training stage. A GNN model evaluation problem is proposed, which aims to accurately estimate the GNNs' performance on unobserved graphs without labels.

Subsequent sections will detail each study about all research questions.

1.3.1 Graphs with Heterophily in Graph Neural Networks

Chapter 3 presents an overview of the current progress made in the study of graphs with heterophily, for breaking the limitation of homophily assumption in existing GNNs. As a ubiquitous graph property in numerous real-world scenarios, heterophily, *i.e.*, nodes with different labels tend to be linked, significantly limiting the performance of existing tailor-made homophilic GNNs, under the the homophily assumption that nodes belonging to the same class are more likely to be connected.

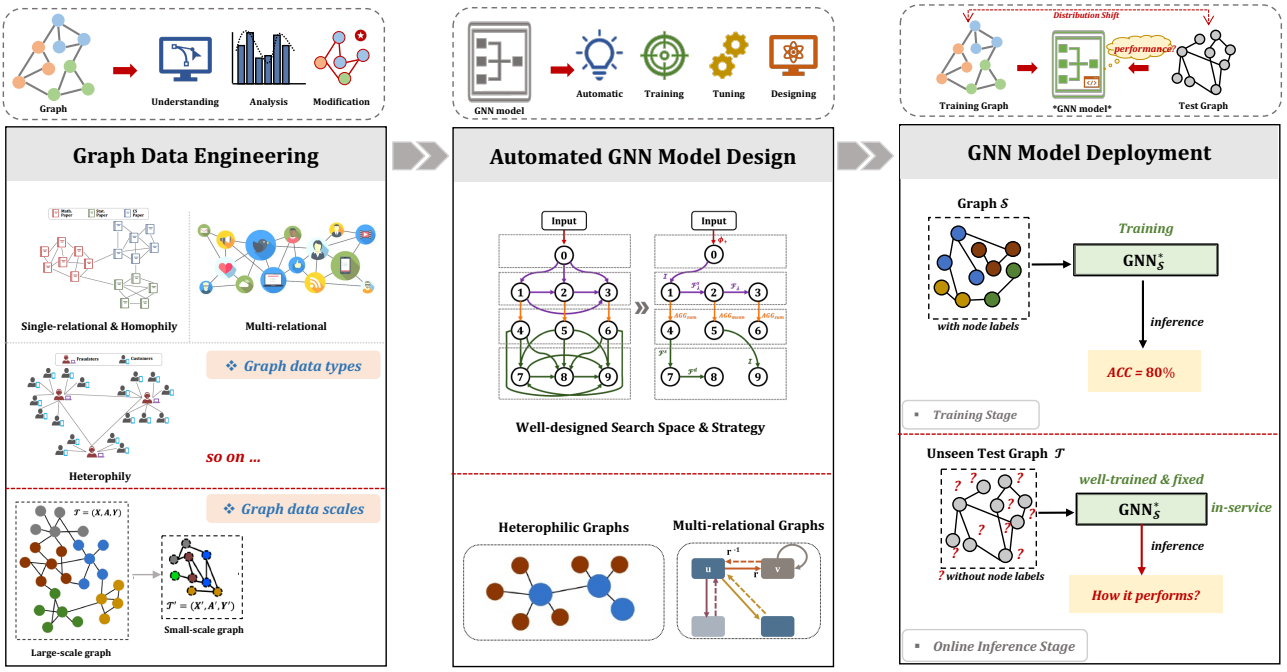

FIGURE 1.3: The overall research blueprint of automated graph MLOps workflow.

First, this thesis provides a comprehensive review of GNN model development for heterophilic graphs. Then, it proposes a systematic taxonomy that essentially governs existing heterophilic GNN models, along with a general summary and detailed analysis. Furthermore, it discusses the correlation between graph heterophily and various graph research domains. In the end, we point out the potential directions to advance and stimulate more future research and applications on heterophilic graph learning with GNNs.

1.3.2 Graph Data Scale Reduction with Graph Condensation

Chapter 4 deals with the challenges caused by large-scale graph data on GNN development. The core idea is to reduce the scale of a large-scale graph by synthesizing a small-scale condensed graph as its substitution, *i.e.*, graph condensation technique, to alleviate the effects of large-scale graph data quantity on graph learning tasks.

Given existing graph condensation methods rely on the joint optimization of nodes and structures in the condensed graph, and overlook critical issues in effectiveness and generalization ability, This thesis advocates a new Structure-Free Graph Condensation paradigm, named SFGC, to distill a large-scale graph into a small-scale graph node set without explicit graph structures, *i.e.*, graph-free data. Our idea is to implicitly encode topology structure information into the node attributes in the synthesized graph-free data, whose topology is reduced to an identity matrix. Specifically, SFGC contains two collaborative components: (1) a training trajectory meta-matching scheme for effectively synthesizing small-scale graph-free data; (2) a graph neural feature score metric for dynamically evaluating the

quality of the condensed data. Through training trajectory meta-matching, SFGC aligns the long-term GNN learning behaviors between the large-scale graph and the condensed small-scale graph-free data, ensuring a comprehensive and compact transfer of informative knowledge to the graph-free data. Afterward, the underlying condensed graph-free data would be dynamically evaluated with the graph neural feature score, which is a closed-form metric for ensuring the excellent expressiveness of the condensed graph-free data. Extensive experiments verify the superiority of SFGC across different condensation ratios.

1.3.3 Automated Graph Neural Networks on Heterophilic Graphs

Chapter 5 deals with the heterophily property in automated GNN model development, through the lens of graph NAS. Given existing graph NAS methods mainly work under the homophily assumption and overlook the heterophily graph property, this thesis proposes a novel automated graph neural network on heterophilic graphs, namely Auto-HeG, to automatically build heterophilic GNN models with expressive learning abilities.

The proposed Auto-HeG incorporates heterophily into all stages of automatic heterophilic graph learning, including search space design, supernet training, and architecture selection. Through the diverse message-passing scheme with joint micro-level and macro-level designs, we first build a comprehensive heterophilic GNN search space, enabling Auto-HeG to integrate complex and various heterophily of graphs. With a progressive supernet training strategy, we dynamically shrink the initial search space according to layer-wise variation of heterophily, resulting in a compact and efficient supernet. Taking a heterophily-aware distance criterion as the guidance, this thesis conducts heterophilic architecture selection in the leave-one-out pattern, so that specialized and expressive heterophilic GNN architectures can be derived. Extensive experiments illustrate the superiority of Auto-HeG in developing excellent heterophilic GNNs to human-designed models and graph NAS models.

1.3.4 Automated Graph Neural Networks on Multi-Relational Graphs

Chapter 6 deals with the multi-relational property in automated GNN model development, through the lens of graph NAS. Given existing graph NAS methods mainly work on single-relational graphs, and current search spaces of automated GNNs are generally coarse-grained by simply integrating typical GNN layers and hyper-parameters, resulting in severe limitations on search capacities and scopes for creating innovative GNN architectures.

This thesis tackles the limitations of single-relational setting and coarse-grained search space design in existing graph NAS, and proposes a novel framework of multi-relational graph neural architecture search, dubbed MR-GNAS, to automatically develop innovative and excellent multi-relational GNN

architectures. Specifically, to enlarge search capacities and improve search flexibility, MR-GNAS contains a fine-grained search space that embraces the full-pipe multi-relational message passing schema, enabling expressive architecture search scopes. With the well-designed fine-grained search space, MR-GNAS constructs a relation-aware supernet with a tree topology, to jointly learn discriminative node and relation representations. By searching with a gradient-based strategy in the supernet, the proposed MR-GNAS could derive excellent multi-relational GNN architectures in multi-relational graph analysis. Extensive experiments on entity classification and link prediction tasks over multi-relational graphs illustrate the effectiveness and superiority of the proposed method.

1.3.5 GNN Performance Evaluation on Unseen Graphs without Labels

Chapter 7 deals with the problem of evaluating the performance of GNN models for practical model deployment and serving. Deployed GNNs typically face significant performance uncertainty when inferring on unseen and unlabeled test graphs, due to mismatched training-test graph distributions. This thesis proposes a new problem, GNN model evaluation, that aims to assess the performance of a specific GNN model trained on labeled and observed graphs, by precisely estimating its performance (*e.g.*, node classification accuracy) on unseen graphs without labels.

Concretely, this thesis proposes a two-stage GNN model evaluation framework, including (1) DiscGraph set construction and (2) GNNEvaluator training and inference. The DiscGraph set captures wide-range and diverse graph data distribution discrepancies through a discrepancy measurement function, which exploits the outputs of GNNs related to latent node embeddings and node class predictions. Under the effective training supervision from the DiscGraph set, GNNEvaluator learns to precisely estimate node classification accuracy of the to-be-evaluated GNN model and makes an accurate inference for evaluating GNN model performance. Extensive experiments on real-world unseen and unlabeled test graphs demonstrate the effectiveness of our proposed method for GNN model evaluation.

1.4 Thesis Structure

This thesis is organized as follows. Chapter 1 introduces the background of Auto-GMLOps, the motivation of our research, the challenges and research questions, as well as the contributions of this thesis. Chapter 2 offers a comprehensive review of the literature on general research topics within GNN models, graph data, and graph learning tasks. Chapters 3 and 4 deal with the graph data engineering stage of Auto-GMLOps, with studies on the heterophily graph data property and the graph data scale reduction. Chapters 5 and 6 present automated GNN development methods on heterophilic graphs and multi-relational graphs in the automated GNN model design stage of Auto-GMLOps. Chapter 7 introduces the new GNN model evaluation problem for real-world GNN online deployment. Chapter

8 discusses possible future directions for this thesis, and Chapter 9 concludes the thesis, summarizing the study's main discoveries, implications, and contributions to the research on Auto-GMLOps.

Chapter 2

Literature Review

2.1 Graph Neural Networks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, unweighted graph where $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ is the node set and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the edge set. The neighbor set of node v is $\mathcal{N}(v) = \{u : (v, u) \in \mathcal{E}\}$, and initial node features are represented by a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_0}$ with d-dimension features, and the edges are represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. Consequently, a given graph can be also represented as $G = (\mathbf{X}, \mathbf{A})$ within the matrix notation.

For the uniform message passing scheme, we have a L-layer GNN model parameterized by $\boldsymbol{\theta}$, and the discriminative node representations are learned with $\mathbf{H}^{(l)} = \mathrm{GNN}_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{A})$ for $l = \{1, 2, \cdots, L\}$, where $\mathbf{H}^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times d_1}$. The GNN model first aggregates the messages from local neighbors, and then combines the aggregated messages with ego-node representations [27] for the update. This detailed process can be denoted as:

$$\begin{aligned} \mathbf{m}_v^{(l)} &= \mathrm{AGG}^{(l)}(\{\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}(v)\}), \\ \mathbf{h}_v^{(l)} &= \mathrm{UPDATE}^{(l)}(\mathbf{h}_v^{(l-1)}, \mathbf{m}_v^{(l)}), \end{aligned} \tag{2.1}$$

where $\mathbf{m}_v^{(l)}$ and $\mathbf{h}_v^{(l)}$ are the message vector and the representation vector of node v at the l-th layer, respectively. AGG$(\cdot)$ and UPDATE$(\cdot)$ are aggregation function and update function, respectively.

2.2 Multi-relational Graph Neural Networks

MR-GNN models jointly learn node and relation representations with various entities, as well as different relation types and edge directions. Typically, RGCN [2] proposed relation-specific filters with basis and block diagonal decomposition for constructing Graph Convolutional Network (GCN)

on relational graphs. Specifically, the directed and labeled multi-relational graph can be denoted as: $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ with the set of nodes $\mathcal{V}$, relations $\mathcal{R}$, and edges $\mathcal{E}$, each edge (u, r, v) represents that the relation r exists from source node u to target node v for $\forall u, v \in \mathcal{V}$ and $r \in \mathcal{R}$, then RGCN defines following propagation scheme as:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\mathbf{W}_r^{(l)}\mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)}\right). \tag{2.2}$$

where $\mathcal{N}_i^r$ denotes the neighbor set that node i under relation $r \in \mathcal{R}$. Different from R-GCN taking the neighborhood of each entity equally, Weighted-GCN [28] presented relation-specific scalar weights within adaptive GCN learning. Despite the satisfying performance, these methods paid more attention to the node embedding learning while the relation embedding learning was not well addressed. CompGCN [17] introduced the entity-relation embedding composition inspired by KG embedding techniques, resulting in a more generic framework and excellent performance. Nevertheless, all these models heavily rely on human design with expertise, causing massive manual enumerations and efforts for obtaining proper network architectures.

2.3 Heterophilic Graph Neural Networks

Existing heterophilic GNNs mainly work on two aspects: non-local neighbor extension [20, 29–33] and GNN architecture refinement [3, 34–39], to first discover appropriate neighbors and then accordingly aggregate informative node representations. In detail, non-local neighbor extension methods focus on enriching the neighbor set in the original graph structure. Mixhop [20] and H2GCN [29] introduced higher-order neighbor mixing to make ego node receive informative representations not only from local one-hop neighbors but also from further k-hop neighbors, so that the heterophilic GNNs can mix latent information from neighbors at various distances. On the other hand, Geom-GCN [31] defined the split 2D Euclidean geometry locations as the geometric relationship criterion to discover potential neighbors. In terms of GNN architecture refinement, existing heterophilic GNNs fully exploit and extract neighbor information by customizing different aggregation functions and update functions in the message passing scheme, where aggregation functions integrate information from the discovered neighbors and update functions combine the learned neighbor messages with the ego information to obtain optimal node representations. FAGCN [3] worked on the aggregation function development, by introducing low-pass and high-pass filters to learn homophily and heterophily, respectively. The critical intuition behind these methods lies in that homophily corresponds to low-frequency signals learned by low-pass filters, and heterophily corresponds to high-frequency signals learned by high-pass filters. Besides, GCNII [38] and GPR-GNN [39] consider the layer-wise operations to boost the representation power of GNNs under heterophily.

2.4 Graph Neural Architecture Search

Graph NAS has greatly enlarged the design space of graph neural architecture for discovering excellent and powerful models, making it a promising research direction that attracts much attention from the research community. Generally, there are two important aspects of graph NAS research, *i.e.*, search space defines architecture candidates and search strategy explores the optimal network models. Typically, GraphNAS [40] first proposed to construct the micro-level search space with candidate components from classical GNN models and related hyper-parameters, followed by an architecture controller with the reinforcement learning (RL) strategy for searching and deriving optimal GNN architectures. Moreover, SANE [41] further considered the macro-level search by introducing inter-layer connections into the search space, followed by the prevalent gradient-based differential search strategy, *i.e.*, DARTS [42]. By involving the implicit link information in search space, Auto-GEL [43] extended gradient-based graph NAS from non-relational graphs to multi-relational graphs, and from node classification task to link prediction task. Nevertheless, current graph NAS methods are still constrained under the homophily assumption, and there should be a well-designed search space with a customized search strategy to address other complex and diverse graph structured data, such as graphs with heterophily. The inspiration could be got from the lines of NAS research on search space shrinking [44–48], supernet optimization [49, 50], and architecture selection [51].

Another area of research closely related to graph NAS is the application of the mixture of experts (MoE) on graphs [52–54]. In MoE, several expert models are utilized to capture diverse node representations and graph properties, with each expert specializing in learning different latent features [55]. Within graph machine learning, MoE can dynamically allocate computational resources to the most suitable experts when addressing various graph structures or complex tasks. Typically, an MoE system comprises two key components: (1) Expert models, which are often advanced and well-established GNN architectures, take the same input and generate their respective outputs; (2) A gating function, which assesses the relevance of each expert model for a given input graph. The final output is a weighted combination of the expert models' outputs, where the weights are determined by the gating function. In summary, MoE excels at dynamically selecting and combining expert models based on the input, and NAS focuses on the automated discovery of optimal GNN architectures tailored to specific tasks and graphs, aiming to push the boundaries of model design and performance. In this thesis, the primary focus is on the graph NAS.

2.5 Graph Data Scale Reduction

Graph size reduction aims to reduce the graph size to fewer nodes and edges for effective and efficient GNN training, including graph sampling [56, 57], graph coreset [58, 59], graph sparsification [60, 61], graph coarsening [62, 63], and recently rising graph condensation [64–66]. Concretely, graph sampling

methods [56, 57] and graph coreset methods [58, 59] sample or select the subset of nodes and edges from the whole graph, such that the information of the derived sub-graph is constrained by the whole large-scale graph, which considerably limits the expressiveness of the size-reduced graph. Moreover, graph sparsification methods [60, 61] and graph coarsening methods [62, 63] reduce the number of edges and nodes by simplifying the edge connections and grouping node representations of the large-scale graph, respectively. The core idea of both sparsification and coarsening is to preserve specific large-scale graph properties (*e.g.*, spectrum and principle eigenvalues) in the sparse and coarsen small graph. The preserved graph properties in the small-scale graph, however, might not be suitable for downstream GNN tasks.

In contrast, this thesis focuses on graph condensation, a dataset distillation technique, which synthesizes a small typical dataset that distills the most important knowledge from a given large target dataset, such that the synthesized small dataset could serve as an effective substitution of the large target dataset for various scenarios [67, 68], *e.g.*, model training and inference, architecture search, and continue learning. Most current dataset distillation/condensation works deal with image data in the computer vision domain, and graph condensation targeting graph-structured data is not fully explored. Only several works, for example, GCOND [64] and DosCond [66] extended the online gradient matching scheme used in image data to structural graph data, along with graph structure learning module to synthesize condensed edge connections.

2.6 Predicting GNN Generalization Error

Our work is relevant to the line of research on predicting model generalization error, which aims to develop a good estimator of a model's classifier performance on unlabeled data from the unknown distributions in the target domain, when the estimated models' classifier has been trained well in the source domain with labeled data [69–74]. Typically, Guillory et al. [73] proposed to estimate a classifier's performance of convolutional neural network (CNN) models on image data based on a designed criterion, named difference of confidences (DoC), that estimates and reflects model accuracy. And Garg et al. [70] proposed to learn a score-based Average Thresholded Confidence (ATC) by leveraging the softmax probability of a CNN classifier, whose accuracy is estimated as the fraction of unlabeled images that receive a score above that threshold. In contrast, Deng et al. [69, 74] directly predicted CNN classifier accuracy by deriving distribution distance features between training and test images with a linear regression model.

However, these existing methods mostly focus on evaluating CNN model classifiers on image data in computer vision, and the formulation of evaluating GNNs for graph structural data still remains under-explored in graph machine learning. Concretely, conducting the model evaluation on GNNs for graph structural data has two critical challenges: (1) different from Euclidean image data, graph structural data lies in non-Euclidean space with complex and non-independent node-edge interconnections, so

TABLE 2.1: Notations of symbols in this thesis.

Symbols	Notations	Symbols	Notations
$\mathcal{G}$	Graph	$\mathbf{H}$	Node latent representations
$\mathcal{V}$	Node set of graph	$\mathbf{h}_v$	Node v embedding
$\mathcal{E}$	Edge set of graph	$\mathbf{m}_v$	Node v message
$\mathcal{R}$	Relation set of graph	$\mathcal{H}_{node}$	Node heterophily
$\mathcal{Y}$	Label set of graph	$\mathcal{H}_{edge}$	Edge heterophily
$\mathcal{N}(v) = \{u : (v, u)\}$	Neighbor set of node v	$a_{uv}^{(l)}$	Edge-aware weights for node pair (u, v) in l-th GNN layer
$\mathcal{N}_v^r$	Neighbor set of node v under relation r	Γ_{op}	GNAS operation set
$\mathbf{X}$	Node feature matrix	$\mathcal{A}$	Graph augmentation set
$\mathbf{A}$	Adjacency matrix	$\mathcal{O}$	GNAS search space
$\mathbf{Y}$	Node label matrix	$\boldsymbol{\theta}$	GNN model parameters

that its node contexts and structural characteristics significantly vary under wide-range graph data distributions, posing severer challenges for GNN model evaluation when serving on unknown graph data distributions. (2) GNNs have entirely different convolutional architectures from those of CNNs, when GNN convolution aggregates neighbor node messages along graph structures. Such that GNNs trained on the observed training graph might well fit its graph structure, and due to the complexity and diversity of graph structures, serving well-trained GNNs on unlabeled test distributions that they have not encountered before would incur more performance uncertainty.

All general notations of symbols used in this thesis are listed in Table. 2.1.

Chapter 3

Graphs with Heterophily in Graph Neural Networks

3.1 Introduction

Graphs are pervasively structured data and have been widely used in many real-world scenarios, such as social networks [75, 76], knowledge bases [17], traffic networks [77] and recommendation systems [14, 78]. Recently, graph neural networks (GNNs) have achieved remarkable success with powerful learning ability and become prevalent models to tackle various graph analytical tasks, such as node classification, link prediction, and graph classification [1, 17, 79, 80].

While a large number of GNNs with diverse architectures have been designed [27, 81–83], the majority of them follow the homophily assumption, *i.e.*, nodes with similar features or same class labels are linked together. For example, in citation networks, a study usually cites reference papers from the same research area [18]. However, real-world graphs do not always obey the homophily assumption but show an opposite property, *i.e.*, *heterophily* that linked nodes have dissimilar features and different class labels. For instance, in online transaction networks, fraudsters are more likely to build connections with customers instead of other fraudsters [84]; in dating networks, most people prefer to date with people of the opposite gender [85]; in molecular networks, protein structures are more likely composed of different types of amino acids that are linked together [29]. The examples of homophilic and heterophilic graphs are provided in Fig. 3.1 to illustrate their difference visually. Importantly, such heterophily restricts the learning ability of existing homophilic GNNs on general graph-structural data, resulting in significant performance degradation on heterophilic graphs.

What Are the Challenges of GNNs for Heterophily? We attribute the performance degradation to the uniform message passing framework under the homophily setting. The procedure of

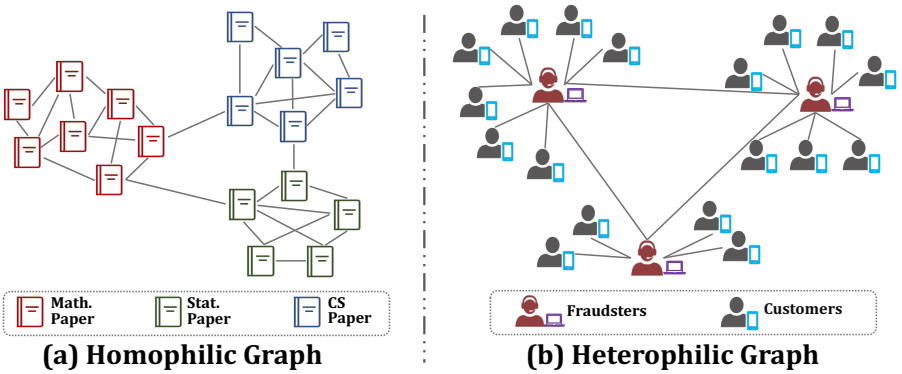

FIGURE 3.1: Examples of homophilic and heterophilic graphs (Left: (a) a citation network; Right: (b) an online transaction network).

this framework can be summarized as: first aggregating the messages extracted from local neighbor nodes, then updating the final ego node (the current central node itself) representations with aggregated neighbor messages. However, under the heterophily setting, such mechanism mainly has two thorny limitations: (1) Local neighbors are defined as proximal nodes in graph topology, which fail to capture informative nodes with long-term distances; On heterophilic graphs, nodes with high structural and semantic similarities might be farther away from each other. (2) Uniform aggregation and update ignore the difference in information between similar and dissimilar neighbors. On heterophilic graphs, discriminative node representation learning yearns for distinguishable information with diverse message passing.

In light of these challenges, recently, more and more researchers have started turning their attention to the study of GNNs with heterophily. The research focus is sufficiently broad, from heterophilic graph data exploration [36, 86, 87] to various technical algorithm development [3, 35, 39]. In fact, heterophilic graph learning is becoming an upward trending research topic due to its great potential. The main reasons behind this are from the following aspects: (1) Graph-structure data with the heterophilic property is widespread in the real world, ranging from daily-life personal relationships to scientific chemical molecular study. The corresponding applications of heterophilic graphs are bright prospects for both academia and industry; (2) Heterophilic graph analysis tasks are still open and promising research topics in development, while numerous challenges need to be tackled for designing powerful and expressive heterophilic GNN models. Thus, it is necessary and timely to provide a general overview of existing heterophilic graph learning approaches.

In this thesis, we develop a comprehensive and systematic review of GNNs for heterophilic graphs to provide a general blueprint of heterophilic graph research. To the best of our knowledge, this is the first time establishing connections and making comparisons among different heterophilic GNN methods, leading to an in-depth understanding of how different methods tackle the challenges. We are expecting that our survey will significantly inspire and facilitate the development of heterophilic graphs. The contributions of our work are summarized as follows:

- **Comprehensive Overview**: We provide a comprehensive overview of current heterophilic GNNs in terms of data, algorithms, and applications. We provide detailed descriptions of each model type, along with the necessary comparison and the gist summary.
- **Systematic Taxonomy**: We provide a systematic taxonomy of heterophilic GNNs and categorize existing methods into two classes, *i.e.*, non-local neighbor extension and GNN architecture refinement.
- **Future Directions**: We suggest promising future research directions and discuss the limitations of existing heterophilic GNNs from multiple perspectives, namely interpretability, robustness, scalability, and heterophilic graph data exploration.

3.2 Preliminary

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, unweighted graph where $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ is the node set and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the edge set. The neighbor set of node v is denoted as $\mathcal{N}(v) = \{u : (v, u) \in \mathcal{E}\}$. The node features are represented by a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where the i-th row $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector of node v_i and d is the number of features.

Problem Setting. Existing works mainly focus on the problem of *semi-supervised node classification*. In this task, assuming each node v belongs to one out of C classes, only a part of nodes are provided with labels as $y_v \in \mathcal{Y} = \{1, \cdots, C\}$ in the training node set. The goal of the task is to predict the classes of nodes whose labels are not given.

Measure of Heterophily & Homophily. In general, heterophily and homophily of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be measured by two metrics: node homophily [31] and edge homophily [29]. Concretely, the node homophily $\mathcal{H}_{node}$ is the average proportion of the neighbors with the same class of each node:

$$\mathcal{H}_{node} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{u \in \mathcal{N}(v) : y_v = y_u\}|}{|\mathcal{N}(v)|}. \tag{3.1}$$

The edge homophily $\mathcal{H}_{edge}$ is the proportion of edges connecting two nodes with the same class:

$$\mathcal{H}_{edge} = \frac{|\{(v, u) \in \mathcal{E} : y_v = y_u\}|}{|\mathcal{E}|}. \tag{3.2}$$

The range of both $\mathcal{H}_{node}$ and $\mathcal{H}_{edge}$ is $[0, 1]$. Graphs with strong homophily have higher $\mathcal{H}_{node}$ and $\mathcal{H}_{edge}$ (closer to 1); whereas graphs with strong heterophily have smaller $\mathcal{H}_{node}$ and $\mathcal{H}_{edge}$ (closer to 0).

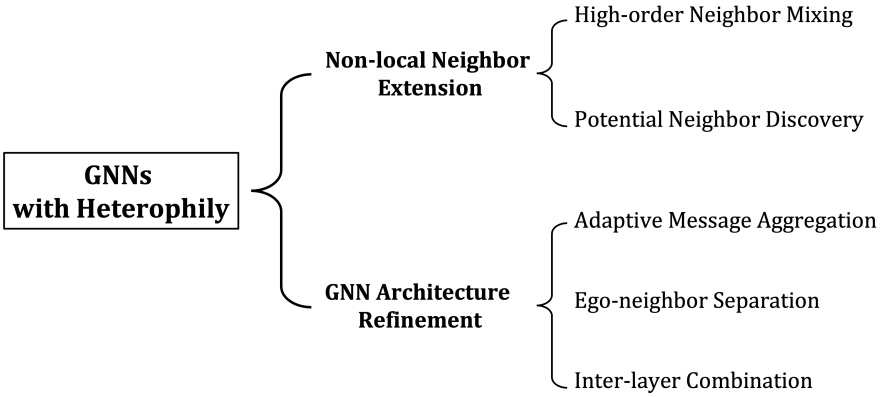

FIGURE 3.2: The algorithm taxonomy of GNNs with heterophily.

3.3 GNNs with Heterophily: Framework and Taxonomy

The proposed algorithm taxonomy is presented in Fig. 3.2. Through the lens of the message passing mechanism, existing heterophilic GNNs can be categorized into two groups, *i.e.*, (1) *Non-local neighbor extension methods*; and (2) *GNN architecture refinement methods*. These two categories correspondingly address two critical issues: (1) How to discover appropriate neighbors; and (2) How to exploit the information from the discovered neighbors. In detail, non-local neighbor extension methods can be further grouped into two subcategories, *i.e.*, *high-order neighbor mixing* and *potential neighbor discovery*, both working on modifying the definition of neighbor set $\mathcal{N}(v)$ in Eq. (2.1). On the other hand, GNN architecture refinement methods can be further divided into three subgroups, *i.e.*, *adaptive message aggregation*, *ego-neighbor separation*, and *inter-layer combination scheme*, working on the functions AGGREGATE$(\cdot)$, UPDATE$(\cdot)$, and GNN layer-wise architecture, respectively. We summarize the techniques of different GNNs with heterophily methods adopted in Table 3.1. As can be observed, some approaches combine different techniques orthogonally to acquire more powerful and expressive heterophilic GNNs.

3.3.1 Non-local Neighbor Extension

Under the uniform message passing framework for homophilic graphs, the neighborhood is usually defined as the set of all neighbors one-hop away (*e.g.*, GCN), which means only messages from proximal nodes in a graph are aggregated. However, such a local neighborhood definition might not be appropriate for heterophilic graphs, where nodes of the same class exhibit high structural similarity but can be farther away from each other. In light of these, current heterophilic GNNs attempt to extend the local neighbors to non-local ones primarily through two schemes: high-order neighbor mixing and potential neighbor discovery. As a result, the representation ability of heterophilic GNNs can be improved significantly by capturing the important features from distant but informative nodes. More details of these two schemes can be found as follows.

3.3.1.1 High-order Neighbor Mixing

Higher-order neighbor mixing allows the ego node to receive latent representations from their local one-hop neighbors and from further k-hop neighbors, so that the heterophilic GNNs can mix latent information from neighbors at various distances. Formally, the k-hop neighbor set is defined as $\mathcal{N}_k(v) = \{u : d(v, u) = k\}$, where $d(u, v)$ measures the shortest distance between nodes u and v.

MixHop [20] is a representative method that aggregates messages from multi-hop neighbors. Apart from one-hop neighbors, MixHop also considers two-hop neighbors for message propagation. After that, the messages acquired from different hops are encoded by different linear transformations and then mixed by concatenation. Similar to MixHop, H2GCN [29] also proposes to aggregate information from higher-order neighbors at each message passing step. To provide a theoretical support, H2GCN verifies that two-hop neighbors tend to involve more nodes with the same class to the ego node, when the labels of one-hop neighbors are conditionally independent to the label of the ego node. Following a similar idea, UGCN [30] utilizes the two-hop network as one of propagation graphs to perform message passing. To control the scale of the two-hop neighbor set, UGCN further restricts the two-hop neighbor set in which nodes have at least two different paths to the connected ego nodes. Besides, TDGNN [88] leverages the tree decomposition to separate neighbors at different k-hops into multiple subgraphs, and then propagates information on these subgraphs in parallel.

3.3.1.2 Potential Neighbor Discovery

Compared to high-order neighbor mixing methods directly utilizing the inherent structural information from graphs, potential neighbor discovery methods reconsider the definition of neighbors in heterophilic graphs and build innovative structural neighbors through the entire topology exploration with heterophily. Apart from the original neighbor set, these methods construct a new potential neighbor set $\mathcal{N}_p(v) = \{u : s(v, u) < \tau\}$, where $s(u, v)$ is a metric function that measures the distance between nodes u and v in a specifically defined latent space and τ is a threshold to limit the number of neighbors.

Typically, Geom-GCN [31] maps the input graph to a continuous latent space and defines the geometric relationships, *i.e.*, split 2D Euclidean geometry locations, as the criteria to discover potential neighbors. Apart from inherent neighbors in original input graphs, neighbors that conform to the defined geometric relationships also participate in the message aggregation of GCN. Furthermore, NLGNN [89] and GPNN [90] adopt the attention mechanism or leverage a pointer network to rank the potential neighbor nodes according to the attention scores or the relevant relationships to the ego node. In this way, potential neighbors that are most similar to the ego node in heterophilic graphs can be discovered and selected. HOG-GCN [32] constructs a homophily degree matrix with the label propagation technique to explore the extent to which a pair of nodes belong to the same class in the

TABLE 3.1: Summary of different heterophilic GNNs. 'Stru Sim' and 'Feat Sim' take the structure similarity and the feature similarity as the distance metric for potential neighbor discovery, respectively. 'Feat-aware' and 'Wegt-aware' work on the node feature $\mathbf{h}_u^{(l-1)}$ and the edge-related weight $a_{uv}^{(l)}$ for adaptive message aggregation, respectively.

Methods	**Neighbor Extension**		**GNN Architecture Refinement**		
	High-order Neighbors	Potential Neighbors	Message Aggregation	Ego-neighbor Separation	Inter-layer Combination
JKNet [92]	✗	✗	✗	✗	✓
MixHop [20]	2-hops	✗	✗	✗	✗
GCNII [38]	✗	✗	✗	✗	✓
GPR-GNN [39]	✗	✗	✗	✗	✓
H2GCN [29]	Multi-hops	✗	✗	✓	✓
Geom-GCN [31]	✗	Stru Sim	✗	✗	✗
UGCN [30]	2-hops	Feat Sim	✗	✗	✗
TDGNN [88]	Multi-hops	✗	✗	✗	✗
NLGNN [89]	✗	Feat Sim	Feat-aware	✗	✗
WRGNN [37]	✗	Stru Sim	Wegt-aware	✓	✗
FAGCN [3]	✗	✗	Wegt-aware	✗	✗
ACM [34]	✗	✗	Wegt-aware	✓	✗
DMP [35]	✗	✗	Wegt-aware	✗	✗
CPGNN [85]	✗	✗	Feat-aware	✗	✗
GGCN [36]	✗	✗	Feat-aware	✓	✗
SimP-GCN [91]	✗	Feat Sim	✗	✗	✗
GPNN [90]	✗	Feat Sim	Feat-aware	✗	✗
HOG-GNN [32]	✗	Stru & Feat Sim	✗	✗	✗
BM-GCN [33]	✗	Feat Sim	✗	✓	✗

entire heterophilic graph. By involving the class-aware information during the propagation process, intra-class nodes with higher heterophily (*i.e.*, lower homophily degree) would contribute more to the neighbor aggregation than underlying inter-class nodes. Also with the consideration of node similarity, UGCN [30] and SimP-GCN [91] choose top k similar node pairs in terms of feature-level cosine similarity for each ego node to construct the neighbor set through kNN algorithm. In contrast, WRGNN [37] takes the degree sequence of neighbor nodes as the metric of structural similarity between ego nodes to reconstruct a graph in which different neighbors are indicated by different relations of edges. Moreover, BM-GCN [33] builds a new network topology based on a block similarity matrix, so that it can explore block-guided neighbors and conduct classified aggregation with different aggregation rules for homophilic and heterophilic nodes.

3.3.2 GNN Architecture Refinement

General GNN architectures in Eq. (2.1) contain two essential components: the aggregation function AGGREGATE$(\cdot)$ to integrate information from the discovered neighbors, and the update function UPDATE$(\cdot)$ to combine the learned neighbor messages with the initial ego representation. Given the original local neighbors and the extended non-local neighbors on heterophilic graphs, existing GNN architecture refinement methods contribute to fully exploiting the neighbor information from the

following aspects by accordingly revising AGGREGATE(·) and UPDATE(·): (1) *Adaptive message aggregation* discriminates the messages of similar neighbors from dissimilar ones; (2) *Ego-neighbor separation* focuses on the difference between ego representations and their neighbor messages; (3) *Inter-layer combination* emphasizes the effect of different propagation ranges (*i.e.*, the number of GNN layers) on node representation learning. All these three aspects come to the same destination: boosting the expressive ability of GNNs for heterophilic graphs by encouraging distinguishable and discriminative node representations.

3.3.2.1 Adaptive Message Aggregation

Given the neighbors to be aggregated, the key of integrating beneficial messages on heterophilic graphs is distinguishing the information of similar neighbors (likely in the same class) from that of dissimilar neighbors (likely in different classes). To make node representations on heterophilic graphs more discriminative, adaptive message aggregation methods alter the aggregation operation AGGREGATE(·) by imposing adaptive edge-aware weights $a_{uv}^{(l)}$ for node pair (u, v) at the l-th layer as:

$$\mathbf{m}_v^{(l)} = \text{AGGREGATE}^{(l)}(\{a_{uv}^{(l)}\, \mathbf{h}_u^{(l-1)} : u \in \mathcal{N}(v)\}). \tag{3.3}$$

In this way, different methods develop various weight assignment schemes for $a_{uv}^{(l)}$ to model the importance of similar and dissimilar neighbors during aggregation. In the following, we provide the details of weight assignment schemes adopted by existing methods in the *spectral* domain and the *spatial* domain, respectively. To be concrete, spectral GNNs leverage the theory of graph signal processing to design graph filters, and spatial GNNs focus on the graph structural topology to develop aggregation strategies.

In spectral domain, in contrast to Laplacian smoothing [93] and low-pass filtering [94] to approximate graph Fourier transformation on homophilic graphs, spectral GNNs on heterophilic graphs involve both low-pass and high-pass filters to adaptively extract low-frequency and high-frequency graph signals. The essential intuition behind this lies in that low-pass filters mainly retain the commonality of node features, while high-pass filters capture the difference between nodes.

Typically, FAGCN [3] adopts a self-gating attention mechanism to learn the proportion of low-frequency and high-frequency signals by splitting the $a_{uv}^{(l)}$ into two components, *i.e.*, $a_{uv}^{(l,LP)}$ and $a_{uv}^{(l,HP)}$, corresponding to low-pass and high-pass filters, respectively. Through adaptive frequency signal learning, FAGCN could achieve expressive performance on different types of graphs with homophily and heterophily. Apart from low-pass and high-pass filters, ACM [34] further involves the identity filter, which is the linear combination of low-pass and high-pass filters. In this way, ACM could adaptively exploit beneficial neighbor information from different filter channels for each node; Meanwhile, its identity filter could guarantee less information loss of the input signal.

In spatial domain, heterophilic GNNs require the diverse topology-based aggregation of neighbors from the same or different classes, rather than the average aggregation in homophily GNNs. Therefore, the edge-aware weights of neighbors should be assigned according to the spatial graph topology and node labels. Taking node attributes as weak labels, DMP method [35] considers node attribute heterophily for diverse message passing and specifies every attribute propagation weight on each edge. Instead of the scalar weight $a_{uv}^{(l)}$ that aggregates all the node attributes with the same weight at the node level, DMP extends the weight to a vector $\mathbf{a}_{uv}^{(l)}$ through operating in the attribute dimension, and this vector can be calculated by either relaxing GAT [80] weights to real values or allowing an element-wise average of neighbor weights. In contrast, WRGNN [37] first transforms the original input heterophilic graph to a multi-relational graph that models the heterophily in relational edges, followed by relational aggregation with explicit link weights.

Apart from the adaptive edge-aware weight learning, there is another solution working on the neighbor representation learning by revising $\mathbf{h}_u^{(l-1)}$ in Eq. (3.3). Conventionally, the above-mentioned methods mainly utilize the contextual node representations of neighbors; In contrast, this solution transforms contextual node embeddings $\mathbf{h}_u^{(l-1)}$ to other node-level properties that reflect the heterophily. In this way, heterophilic GNNs could capture the beneficial information of heterophily for distinguishable node representation learning. Typically, instead of propagating original node feature representations, CPGNN [85] propagates a prior belief estimation based on a compatibility matrix, so that it could capture both heterophily and homophily by modeling the likelihood of connections between nodes in different classes. Besides, GGCN [36] uses the cosine similarity to send signed neighbor features under certain constraints of relative node degrees. In this way, the messages are allowed to be optionally multiplied by a negative sign or a positive sign. Intuitively, signed messages consist of the negated messages sent by neighbors of the opposing classes, and the positive messages sent by neighbors of the same class. Going beyond the uniform GCN aggregation, NLGNN [89] and GPNN [90] consider the sequential aggregation where the neighbors are ranked based on the class similarity in order. A regular 1D-convolution layer is applied to extract the affinities between the sequential nodes whether the nodes are close or distant in heterophilic graphs.

3.3.2.2 Ego-neighbor Separation

On heterophilic graphs, an ego node is likely to be dissimilar from its neighbors in terms of class labels. Hence, encoding ego-node representations separately from the aggregated representations of neighbor nodes would benefit distinguishable node embedding learning. In detail, ego-neighbor separation methods detach self-loop connections of the ego nodes in AGGREGATE$(\cdot)$. Meanwhile, they alter the UPDATE$(\cdot)$ as the non-mixing operations, *e.g.*, concatenation, instead of the mixing operation, *e.g.*, 'average' in vanilla GCN.

H2GCN [29] first proposes to exclude the self-loop connection and points out that the non-mixing operations in the update function ensure that expressive node representations would survive over multiple rounds of propagation without becoming prohibitively similar. Besides, WRGNN [37] imposes different mapping functions on the ego-node embedding and its neighbor aggregated messages, while GGCN [36] simplifies the mapping functions to learnable scalar parameters to separately learn ego-neighbor representations. Moreover, ACM [34] adopts the identity filter to separate the ego embedding and then conducts the channel-level combination with its neighbor information in the update function.

3.3.2.3 Inter-layer Combination

Different from adaptive message aggregation methods and ego-neighbor separation methods which deal with heterophilic graphs by looking deeply into each layer of GNNs, inter-layer combination methods consider the layer-wise operations to boost the representation power of GNNs under heterophily. The intuition behind this strategy is that in the shallow layers of GNNs, they collect local information, *e.g.*, one-hop neighbor locality in two-layer vanilla GCN, when the layers go deeper, GNNs gradually capture global information implicitly via multiple rounds of neighbor propagation. Under the heterophily setting, neighbors with similar information, *i.e.*, class labels, might locate in both local geometry and long-term global topology. Hence, combining intermediate representations from each layer contributes to leveraging different neighbor ranges with the consideration of both local and global structural properties, resulting in powerful heterophilic GNNs.

The prior idea first comes from JKNet [92] which flexibly captures better structure-aware representation with different neighborhood ranges under the homophily setting. Due to the more complex structural topology in heterophilic graphs, H2GCN [29] is inspired to concatenate the node representations from all previous layers and learn heterophilic node features at the final layer distinctly. Compared with H2GCN using all previous intermediate representations, GCNII [38] only integrates the first layer's node embedding at each layer with the initial residual connection. Instead of using the simple concatenation operation, GPR-GNN [39] further assigns learnable weights to combine the representations of each layer adaptively via the Generalized PageRank (GPR) technique. Hence, inter-layer combination methods are able to conduct topological feature exploration and benefit from informative multi-round propagation, making node features of heterophilic graphs distinguishable.

3.4 Future Directions

GNNs for heterophilic graphs are fast developing in the past few years. Beyond current research, there still remain several open challenges and opportunities worthy of further attention and exploration. In this section, we discuss the following directions to stimulate future research.

3.4.1 Interpretability and Robustness.

Interpretability and robustness are two crucial aspects of GNN models in risk-sensitive or privacy-related application fields, *e.g.*, healthcare and cybersecurity. Although there are several studies on the interpretability [95, 96] and robustness [97] for homophilic GNNs, how to explain the predictions of heterophilic GNNs and how to conduct adversarial learning on heterophilic graphs are still under-explored. Heterophily makes these tasks more challenging than homophily: for interpretability, since most local neighbor nodes are not in the same class as the ego nodes, extracting explainable subgraphs from highly heterophilic graph data is much harder, where both proximal and distant topological structures are required to be discovered and exploited; for robustness, simply adding/detecting the node pairs with low similarities for attacking/defending homophilic GNNs may not work for heterophilic GNNs, since the ego nodes connected by unperturbed edges inherently have dissimilar features with neighbor nodes. Hence, complex heterophilic topologies and semantics desire more targeted adversarial attack and defence techniques.

3.4.2 Scalable Heterophilic GNNs.

Current heterophilic GNNs are generally trained on relatively small graphs, which significantly limit their ability of modelling large-scale data and exploring more complicated heterophilic patterns. Although possible solutions to tackle scalability can be borrowed from the mainstream graph sampling strategies [56, 57] for homophilic GNNs, the connections and relationships of heterophilic nodes would be undermined by sampling only mini-batches, especially when similar and dissimilar neighbors contribute differently to learning the ego-node representations. LINKX [87] recently verifies that even a simple MLP-based model could outperform GNNs for mini-batch training on large-scale heterophilic graphs. Hence, addressing the scalability problem of heterophilic graphs requires exploring more relationships between ego-neighbor node features. Moreover, how to keep the inherent heterophily unchanged when conducting sampling on heterophilic graphs is still an open question.

3.4.3 Relationship between Heterophily and Over-smoothing.

Heterophily and over-smoothing are two critical aspects of limiting current GNNs' performance, where the former violates the general homophilic topological assumption, and the latter prevents most GNNs from going deeper when the performance drops with the increasing number of layers. Surprisingly, some studies show that GNNs for heterophilic graphs could be empirically utilized to address the over-smoothing problem [3, 89], and vice versa [38]. A recent work [36] makes the first attempt to explain the connections between these two aspects and points out they could be attributed to a common cause from a unified theoretical perspective. However, there is still a blank for thoroughly exploring and understanding the relationship between heterophily and over-smoothing. Hence, this

brings researchers great opportunities to deeply dive into solid underpinnings and proofs behind and further overcome them together.

3.4.4 Comprehensive Benchmarks and Metrics.

Currently, most of existing heterophilic benchmark datasets use node heterophily and edge heterophily to measure the heterophilic degree of a graph. For one thing, these commonly-used datasets are identified with certain limitations, *e.g.*, small sizes and synthetic classes, resulting in narrow application ranges of heterophilic graphs. Despite recently released large-scale heterophilic graph datasets [87], it is desirable to extract heterophilic datasets from more real-world scenarios, such as bioinformatics, protein-protein interaction, and physical process. Meanwhile, more heterophilic graph analytic tasks and applications are desired for extensive heterophilic graph data study, apart from the current mainstream semi-supervised node classification. For another thing, current heterophily metrics only target nodes and edges of heterophilic graphs, which significantly limit the discovery of other properties of heterophily, such as imbalanced class distributions [87] and non-random neighbor distributions [98]. Therefore, a future opportunity is to study the underlying properties of heterophilic graphs and come up with novel metrics to quantify such properties to further strengthen the expressive ability of GNNs.

3.5 Conclusion

In this thesis, we conducted a comprehensive overview of graph neural networks for heterophilic graphs. To that, we proposed a novel and systematic taxonomy of existing heterophilic GNNs, consisting of non-local neighbor extension methods and GNN architecture refinement methods. We provided a thorough introduction and analysis of the current research progress and challenges of heterophilic graph learning, followed by the collected heterophilic graph benchmarks for facilitating evaluations. In the end, we shared our insights into future research opportunities and directions to advance the development of heterophilic GNNs.

Chapter 4

Graph Data Scale Reduction with Graph Condensation

4.1 Introduction

As prevalent graph data learning models, graph neural networks (GNNs) have attracted much attention and achieved great success [5, 19, 23, 24, 99–101]. Various graph data in the real world comprises millions of nodes and edges[102–104], reflecting diverse node attributes and complex structural connections [105–108]. Modeling such large-scale graphs brings serious challenges in both data storage and GNN model designs, hindering the applications of GNNs in many industrial scenarios [26, 109–114]. For instance, designing GNN models usually requires repeatedly training GNNs for adjusting proper hyper-parameters and constructing optimal model architectures. When taking large-scale graphs as training data, repeated training through message passing along complex graph structures, makes it highly computation-intensive and time-consuming through try-and-error.

To address these challenges brought by the scale of graph data, a natural data-centric solution [115] is graph size reduction, which transforms the real-world large-scale graph to a small-scale graph, such as graph sampling [56, 57], graph coreset [58, 59], graph sparsification [60, 61], and graph coarsening [62, 63]. These conventional methods either extract representative nodes and edges or preserve specific graph properties from the large-scale graphs, resulting in severe limitations of the obtained small-scale graphs in the following two folds. First, the available information on derived small-scale graphs is significantly upper-bounded and limited within the range of large-scale graphs [56, 59]. Second, the preserved properties of small-scale graphs, *e.g.*, spectrum and clustering, might not always be optimal for training GNNs for downstream tasks [60, 62, 63].

In light of these limitations of conventional methods, in this chapter, we mainly focus on graph condensation [64, 66], a new rising synthetic method for graph size reduction. Concretely, graph

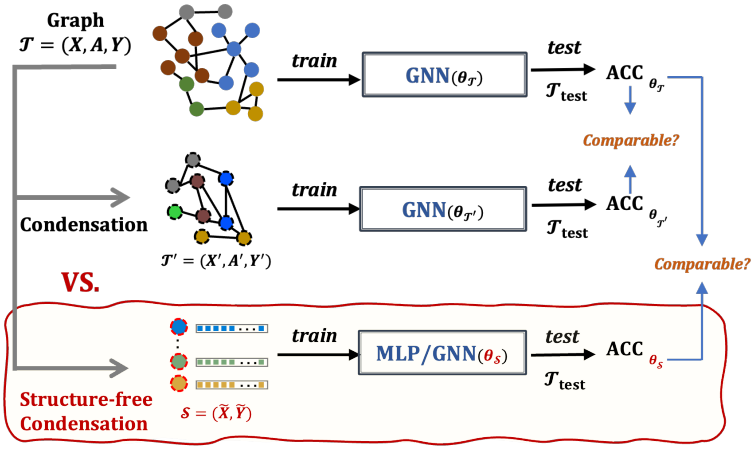

FIGURE 4.1: Comparisons of condensation *vs.* structure-free condensation on graphs.

condensation aims to directly optimize and synthesize a small-scale condensed graph, so that the small-scale condensed graph could achieve comparable test performance as the large-scale graph when training the same GNN model. Therefore, the principal goal of graph condensation is to ensure consistent test results for GNNs when taking the large-scale graph and the small-scale condensed graph as training data.

However, due to the structural characteristic of graph data, nodes and edges are tightly coupled. This makes condensing graph data a complicated task since high-quality condensed graphs are required to jointly synthesize discriminative node attributes and topology structures. Some recent works have made initial explorations of graph condensation [64, 66]. For instance, GCOND [64] proposed the online gradient matching schema between the synthesized small-scale graph and the large-scale graph, followed by a condensed graph structure learning module for synthesizing both condensed nodes and structures. However, existing methods overlook two-fold critical issues regarding **effectiveness and generalization ability**. First, graph condensation requires a triple-level optimization to jointly learn three objectives: GNN parameters, distilled node attributes, and topology structures. Such complex optimization cannot guarantee optimal solutions for both nodes and edges in the condensed graph, significantly limiting its effectiveness as the representative of the large-scale graph. Furthermore, existing online GNN gradients [64, 66] are calculated with the short-range matching, leading to the short-sight issue of failing to imitate holistic GNN learning behaviors, limiting the quality of condensed graphs. Second, existing condensed graphs generally show poor generalization ability across different GNN models [64, 66], because different GNN models vary in their convolution operations along graph structures. As a result, existing methods are vulnerable to overfiting of specific GNN architectures by distilling convolutional information into condensed graph structures.

To deal with the above two-fold challenges, in this chapter, we propose a novel Structure-Free Graph Condensation paradigm, named SFGC, to distill large-scale real-world graphs into small-scale synthetic graph node sets without graph structures, *i.e.*, condensed graph-free data. Different from conventional graph condensation that synthesizes both nodes and structures to derive a small-scale

graph, as shown in Fig. 4.1, the proposed structure-free graph condensation only synthesizes a small-scaled node set to train a GNN/MLP, when it implicitly encodes topology structure information into the node attributes in the synthesized graph-free data, by simplifying the condensed topology to an identity matrix. Overall, the proposed SFGC contains two essential components: (1) a training trajectory meta-matching scheme for effectively synthesizing small-scale graph-free data; (2) a graph neural feature score metric for dynamically evaluating the quality of condensed graph-free data. To address the short-sight issue of existing online gradient matching, our training trajectory meta-matching scheme first trains a set of training trajectories of GNNs on the large-scale graph to acquire an expert parameter distribution, which serves as offline guidance for optimizing the condensed graph-free data. Then, the proposed SFGC conducts meta-matching to align the long-term GNN learning behaviors between the large-scale graph and condensed graph-free data by sampling from the training trajectory distribution, enabling the comprehensive and compact transfer of informative knowledge to the graph-free data. At each meta-matching step, we would obtain updated condensed graph-free data, which would be fed into the proposed graph neural feature score metric for dynamically evaluating its quality. This metric is derived based on the closed-form solutions of GNNs under the graph neural tangent kernel (GNTK) ridge regression, eliminating the iterative training of GNNs in the dynamic evaluation. Finally, the proposed SFGC selects the condensed graph-free data with the smallest score as the optimal representative of the large-scale graph. Our proposed structure-free graph condensation method could benefit many potential application scenarios, such as, *graph neural architecture search* [21, 25], *privacy protection* [116], *adversarial robustness* [117, 118], *continual learning* [115], and so on.

In summary, the contributions of this chapter are listed as follows:

- We propose a novel Structure-Free Graph Condensation paradigm to effectively distill large-scale real-world graphs to small-scale synthetic graph-free data with superior expressiveness, to the best of our knowledge, for the first time.
- To explicitly imitate the holistic GNN training process, we propose the training trajectory meta-matching scheme, which aligns the long-term GNN learning behaviors between the large-scale graph and the condensed graph-free data, with the theoretical guarantee of eliminating graph structure constraints.
- To ensure the high quality of the condensed data, we derive a GNTK-based graph neural feature score metric, which dynamically evaluates the small-scale graph-free data at each meta-matching step and selects the optimal one. Extensive experiments verify the superiority of our method.

Prior Works. Our research falls into the research topic *dataset distillation (condensation)* [67, 119], which aims to synthesize a small typical dataset that distills the most important knowledge from a given large target dataset as its effective substitution. Considering most of the works condense image data [119–123], due to the complexity of graph structural data, only a few works [64, 66] address

graph condensation, while our research designs a new structure-free graph condensation paradigm for addressing the effectiveness and generalization ability issues in existing graph condensation works. Our research also significantly differs from other general graph size reduction methods, for instance, graph coreset [58, 59], graph sparsification [60, 61] and so on.

4.2 Structure-Free Graph Condensation

4.2.1 Preliminaries

Notations. Denote a large-scale graph dataset to be condensed by $\mathcal{T} = (\mathbf{X}, \mathbf{A}, \mathbf{Y})$, where $\mathbf{X} \in \mathbb{R}^{N \times d}$ denotes N number of nodes with d-dimensional features, $\mathbf{A} \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix indicating the edge connections, and $\mathbf{Y} \in \mathbb{R}^{N \times C}$ denotes the C-classes of node labels. In general, graph condensation synthesizes a small-scale graph dataset denoted as $\mathcal{T}' = (\mathbf{X}', \mathbf{A}', \mathbf{Y}')$ with $\mathbf{X}' \in \mathbb{R}^{N' \times d}$, $\mathbf{A}' \in \mathbb{R}^{N' \times N'}$, and $\mathbf{Y}' \in \mathbb{R}^{N' \times C}$ when $N' \ll N$. In this chapter, we propose the structure-free graph condensation paradigm, which aims to synthesize a small-scale graph node set $\mathcal{S} = (\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}})$ without explicitly condensing graph structures, *i.e.*, the condensed graph-free data, as an effective substitution of the given large-scale graph. Hence, $\widetilde{\mathbf{X}}$ contains joint node context attributes and topology structure information, which is a more compact representative compared with $(\mathbf{X}', \mathbf{A}')$.

Graph Condensation. Given a GNN model parameterized by $\boldsymbol{\theta}$, graph condensation [64] is defined to solve the following triple-level optimization objective by taking $\mathcal{T} = (\mathbf{X}, \mathbf{A}, \mathbf{Y})$ as input:

$$\begin{aligned} & \min_{\mathcal{T}'} \mathcal{L}\left[\mathrm{GNN}_{\boldsymbol{\theta}_{\mathcal{T}'}}(\mathbf{X}, \mathbf{A}), \mathbf{Y}\right] \\ s.t. \quad & \boldsymbol{\theta}_{\mathcal{T}'} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left[\mathrm{GNN}_{\boldsymbol{\theta}}\left(\mathbf{X}', \mathbf{A}'\right), \mathbf{Y}'\right], \\ & \boldsymbol{\psi}_{\mathbf{A}'} = \arg\min_{\boldsymbol{\psi}} \mathcal{L}\left[\mathrm{GSL}_{\boldsymbol{\psi}}\left(\mathbf{X}'\right)\right], \end{aligned} \tag{4.1}$$

where $\mathrm{GSL}_{\boldsymbol{\psi}}$ is a submodule parameterized by $\boldsymbol{\psi}$ to synthesize the graph structure $\mathbf{A}'$. One of inner loops learns the optimal GNN parameters $\boldsymbol{\theta}_{\mathcal{T}'}$, while the other learns the optimal GSL parameters $\boldsymbol{\psi}_{\mathbf{A}'}$ to obtain the condensed $\mathbf{A}'$, and the outer loop updates the condensed nodes $\mathbf{X}'$. All these comprise the condensed small-scale graph $\mathcal{T}' = (\mathbf{X}', \mathbf{A}', \mathbf{Y}')$, where $\mathbf{Y}'$ is pre-defined based on the class distribution of the label space $\mathbf{Y}$ in the large-scale graph.

Overall, the above optimization objective needs to solve the following variables iteratively: (1) condensed $\mathbf{X}'$; (2) condensed $\mathbf{A}'$ with $\mathrm{GSL}_{\boldsymbol{\psi}_{\mathbf{A}'}}$; and (3) $\mathrm{GNN}_{\boldsymbol{\theta}_{\mathcal{T}'}}$. Jointly learning these interdependent objectives is highly challenging. It is hard to guarantee that each objective obtains the optimal and convergent solution in such a complex and nested optimization process, resulting in the limited expressiveness of the condensed graph. This dilemma motivates us to reconsider the optimization objective of graph condensation to synthesize the condensed graph more effectively.

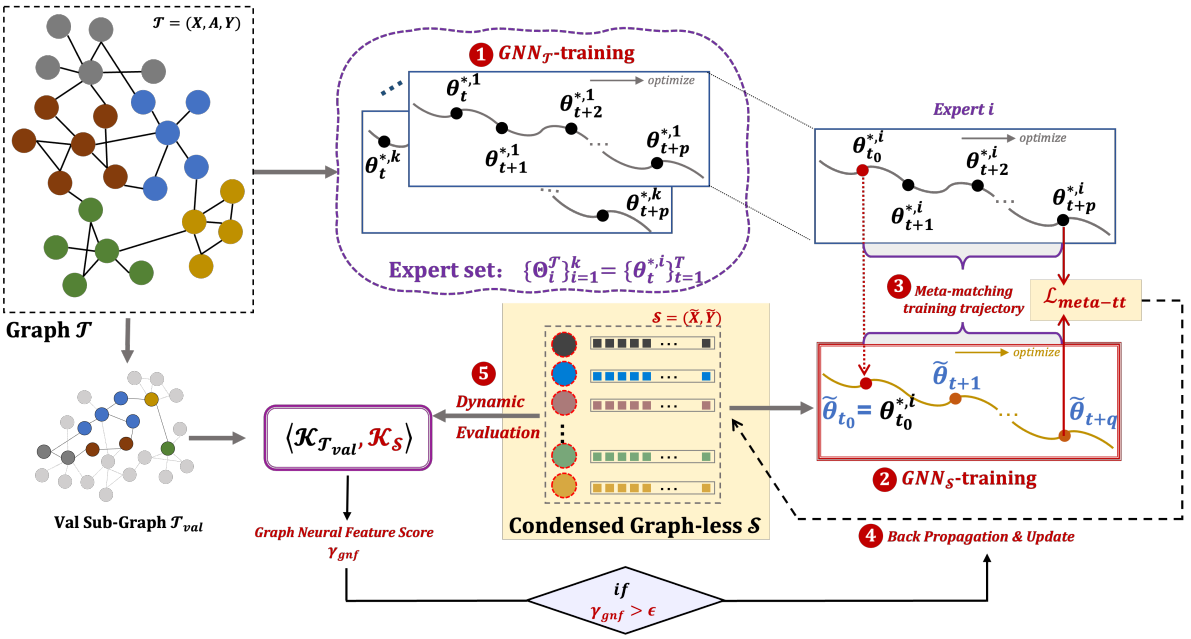

FIGURE 4.2: Overall pipeline of the proposed Structure-Free Graph Condensation (SFGC) framework.

Graph Neural Tangent Kernel (GNTK). As a new class of graph kernels, graph neural tangent kernel (GNTK) is easy to train with provable theoretical guarantees, and meanwhile, enjoys the full expressive power of GNNs [124–127]. In general, GNTK can be taken as the infinitely-wide multi-layer GNNs trained by gradient descent. It learns a class of smooth functions on graphs with close-form solutions. More specifically, let $G = (V, E)$ denote a graph with nodes V and edges E, where each node $v \in V$ within its neighbor set $\mathcal{N}(v)$. Given two graphs $G = (V, E)$ and $G' = (V', E')$ with n and n' number of nodes, their covariance matrix between input features can be denoted as $\Sigma^{(0)}(G, G') \in \mathbb{R}^{n \times n'}$. Each element in $\left[\Sigma^{(0)}(G, G')\right]_{uu'}$ is the inner product $\boldsymbol{h}_u^\top \boldsymbol{h}_{u'}$, where $\boldsymbol{h}_u$ and $\boldsymbol{h}_{u'}$ are of input features of two nodes $u \in V$ and $u' \in V'$. Then, for each GNN layer $\ell \in \{0, 1, \ldots, L\}$ that has $\mathcal{B}$ fully-connected layers with ReLU activation, GNTK calculates $\boldsymbol{\mathcal{K}}_{(\beta)}^{(\ell)} \langle G, G' \rangle$ for each $\beta \in [\mathcal{B}]$:

$$\begin{aligned} \left[\boldsymbol{\mathcal{K}}_{(\beta)}^{(\ell)} \langle G, G' \rangle\right]_{uu'} = & \left[\boldsymbol{\mathcal{K}}_{(\beta-1)}^{(\ell)} \langle G, G' \rangle\right]_{uu'} \left[\dot{\boldsymbol{\Sigma}}_{(\beta)}^{(\ell)}(G, G')\right]_{uu'} \\ & + \left[\boldsymbol{\Sigma}_{(\beta)}^{(\ell)}(G, G')\right]_{uu'}, \end{aligned} \tag{4.2}$$

where $\dot{\Sigma}^{(\ell)}$ denotes the derivative *w.r.t.* the ℓ-th GNN layer of the covariance matrix, and the $(\ell+1)$-th layer's covariance matrix aggregates neighbors along graph structures as $\left[\boldsymbol{\Sigma}_{(0)}^{(\ell+1)}(G, G')\right]_{uu'} = \sum_{v \in \mathcal{N}(u) \cup \{u\}} \sum_{v' \in \mathcal{N}(u') \cup \{u'\}} \left[\boldsymbol{\Sigma}_{(\mathcal{B})}^{(\ell)}(G, G')\right]_{vv'}$, ditto for the kernel $\left[\boldsymbol{\mathcal{K}}_{(0)}^{(\ell+1)}(G, G')\right]_{uu'}$. With the GNTK matrix $\boldsymbol{\mathcal{K}}_{(\mathcal{B})}^{(L)} \langle G, G' \rangle \in \mathbb{R}^{n \times n'}$ at the node level, we use the graph kernel method to solve the equivalent GNN model for node classification with closed-form solutions. This would significantly benefit the efficiency of condensed data evaluation by eliminating iterative GNN training.

4.2.2 Overview of SFGC Framework

The crux of achieving structure-free graph condensation is in determining discriminative node attribute contexts, which implicitly integrates topology structure information. We compare the paradigms between existing graph condensation (GC) and our new structure-free condensation SFGC as follows:

$$\begin{aligned} &\mathcal{T} = (\mathbf{X}, \mathbf{A}, \mathbf{Y}) \to \mathcal{T}' = (\mathbf{X}', \mathbf{A}', \mathbf{Y}'), \quad \text{GC}. \\ &\mathcal{T} = (\mathbf{X}, \mathbf{A}, \mathbf{Y}) \to \mathcal{S} = (\widetilde{\mathbf{X}}, \mathbf{I}, \widetilde{\mathbf{Y}}) = \mathcal{S} = (\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}), \quad \text{SFGC}. \end{aligned} \tag{4.3}$$

It is fundamentally different between existing GC paradigm with $\mathcal{T} \to \mathcal{T}'$ and our SFGC with $\mathcal{T} \to \mathcal{S}$. Our idea is to synthesize a graph-free data $\mathcal{S} = (\widetilde{\mathbf{X}}, \mathbf{I}, \widetilde{\mathbf{Y}})$ whose topology is reduced to an identity matrix $\mathbf{I}$ (*i.e.*, structure-free), instead of explicitly learning $\mathbf{A}'$as existing GC. To enforce node attribute $\widetilde{\mathbf{X}}$ encoding topology structure information as $\widetilde{\mathbf{X}} \simeq (\mathbf{X}, \mathbf{A})$, we propose a long-term imitation learning process with training trajectories, which requires a GNN model learned from $\mathcal{S}$, *i.e.*, $\text{GNN}_{\mathcal{S}}$= $\text{GNN}(\widetilde{\mathbf{X}}, \mathbf{I}, \widetilde{\mathbf{Y}})$ must imitate a GNN model from the original graph, *i.e.*, $\text{GNN}_{\mathcal{T}} = \text{GNN}(\mathbf{X}, \mathbf{A}, \mathbf{Y})$. The overall pipeline of the proposed Structure-Free Graph Condensation (SFGC) framework is shown in Fig. 4.2.

We consider two coupled stages in the proposed SFGC framework, *i.e.*, graph-free data synthesis (① ∼ ④) and condensed graph-free data evaluation (⑤), corresponding to two essential components: (1) the training trajectory meta-matching scheme and (2) the graph neural feature score metric. Concretely, as illustrated in Fig. 4.2, taking the given large-scale graph $\mathcal{T}$ as input, we first (①) train an expert set of $\text{GNN}_{\mathcal{T}}$, parameterized by $\left\{\mathbf{\Theta}_{\mathcal{T}}^{i}\right\}_{i=1}^{K} = \left\{\boldsymbol{\theta}_{t}^{*,i}\right\}_{t=1}^{T}$, denoting K numbers of expert training trajectories, each within T time steps. Then, we sample a single expert training trajectory i at t_0 step, *i.e.*, $\boldsymbol{\theta}_{t_0}^{*,i}$, and further use it to initialize the model (②) $\text{GNN}_{\mathcal{S}}$ trained by the condensed graph-free data $\mathcal{S}$. Then, after q steps of $\text{GNN}_{\mathcal{S}}$ and p steps of $\text{GNN}_{\mathcal{T}}$, we conduct the meta-matching (③) between the long-term intervals of training trajectories $\left[\widetilde{\boldsymbol{\theta}}_{t_0}, \widetilde{\boldsymbol{\theta}}_{t_0+q}\right]$ and $\left[\boldsymbol{\theta}_{t_0}^{*,i}, \boldsymbol{\theta}_{t_0+p}^{*,i}\right]$ with the proposed meta-matching loss $\mathcal{L}_{\text{meta-tt}}$. Next, the loss function back-propagates along $\text{GNN}_{\mathcal{S}}$ and the condensed graph-free data $\mathcal{S}$ is updated (④). After, the updated $\mathcal{S}$ at the current step is used to calculate the GNTK-based graph neural feature score metric γ_{gnf} for dynamic evaluation (⑤), along with the large-scale validation subgraph $\mathcal{T}_{\text{val}}$. Finally, SFGC selects the optimal condensed graph-free data with the smallest γ_{gnf}^{*} as the expressive substitution of the large-scale graph.

4.2.3 Training Trajectory Meta-matching

Different from existing graph condensation methods [64, 66] that conduct online gradient matching within the short range, *i.e.*, step-by-step or single-step matching gradients between the large-scale graph and the condensed graph, the proposed SFGC matches their long-term GNN training trajectories with the guidance of the offline expert parameter distribution. Concretely, inspired by [123], we

first train K numbers of GNN models with the same architecture denoted as $\text{GNN}_{\mathcal{T}}$ on the large-scale graph $\mathcal{T}$. Then, we save their network parameters $\left\{\boldsymbol{\Theta}_{\mathcal{T}}^{i}\right\}_{i=1}^{K}=\left\{\boldsymbol{\theta}_{t}^{*,i}\right\}_{t=1}^{T}$ at certain epoch intervals, resulting in K numbers of expert training trajectories that have comprehensive knowledge of the large-scale graph in terms of GNN's training process. Such expert training trajectories further build a parameter distribution $P_{\boldsymbol{\Theta}_{\mathcal{T}}}$ denoting the GNN learning behavior on the large-scale graph. Note that such a parameter distribution is pre-calculated and stored. Hence, this process can be offline and separated from the end-to-end graph condensation pipeline, moderately reducing the online computation costs.

By sampling from the pre-derived parameter distribution $P_{\boldsymbol{\Theta}_{\mathcal{T}}}$, we optimize the following objective for synthesizing $\mathcal{S}$ as:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_{t}^{*,i}\sim P_{\boldsymbol{\Theta}_{\mathcal{T}}}}\left[\mathcal{L}_{\text{meta-tt}}\left(\boldsymbol{\theta}_{t}^{*}|_{t=t_0}^{p},\widetilde{\boldsymbol{\theta}}_{t}|_{t=t_0}^{q}\right)\right]. \tag{4.4}$$

Note that $\boldsymbol{\theta}_{t}^{*}|_{t=t_0}^{p}$ and $\widetilde{\boldsymbol{\theta}}_{t}|_{t=t_0}^{q}$ denote the parameters of $\text{GNN}_{\mathcal{T}}$ within the range of (t_0, t_0+p) and $\text{GNN}_{\mathcal{S}}$ within (t_0, t_0+q), respectively, where $t_0 < t_0+p < T$. More specifically, $\mathcal{L}_{\text{meta-tt}}$ calculates certain parameter training intervals within $\left[\boldsymbol{\theta}_{t_0}^{*,i}, \boldsymbol{\theta}_{t_0+p}^{*,i}\right]$ and $\left[\widetilde{\boldsymbol{\theta}}_{t_0}, \widetilde{\boldsymbol{\theta}}_{t_0+q}\right]$ as

$$\mathcal{L}_{\text{meta-tt}}=\frac{\left\|\widetilde{\boldsymbol{\theta}}_{t_0+q}-\boldsymbol{\theta}_{t_0+p}^{*,i}\right\|_2^2}{\left\|\widetilde{\boldsymbol{\theta}}_{t_0}-\boldsymbol{\theta}_{t_0+p}^{*,i}\right\|_2^2}. \tag{4.5}$$

Here, we initialize the parameter of $\text{GNN}_{\mathcal{S}}$ with that of $\text{GNN}_{\mathcal{T}}$ at t_0 training step, so that we have $\boldsymbol{\theta}_{t_0}^{*,i}=\widetilde{\boldsymbol{\theta}}_{t_0}$. And we consider the expectation on $\mathcal{S}$ *w.r.t.* different initialized parameter $\boldsymbol{\theta}_{t_0}^{*,i}$ in distribution $P_{\boldsymbol{\Theta}_{\mathcal{T}}}$, which can be taken as a 'meta' way to make the distilled dataset $\mathcal{S}$ adapt different parameter initialization. That is why we call it 'meta-matching'. In this way, when initializing $\text{GNN}_{\mathcal{T}}$ and $\text{GNN}_{\mathcal{S}}$ with the same model parameters, Eq. (4.5) contributes to aligning the learning behaviors of $\text{GNN}_{\mathcal{T}}$ that experiences p-steps optimization, to $\text{GNN}_{\mathcal{S}}$ that experiences q-steps optimization. In this way, the proposed training trajectory meta-matching schema could comprehensively imitate the long-term learning behavior of GNN training. As a result, the informative knowledge of the large-scale graph $\mathcal{T}$ can be effectively transferred to the small-scale condensed graph-free data $\mathcal{S}=(\widetilde{\mathbf{X}},\widetilde{\mathbf{Y}})$ in the above outer-loop optimization objective of Eq. (4.4).

For the inner loop, we train $\text{GNN}_{\mathcal{S}}$ on the synthesized small-scale condensed graph-free data for optimizing its model parameter until the optimal $\widetilde{\boldsymbol{\theta}}_{\mathcal{S}}^{*}$. Therefore, the final optimization objective of the proposed SFGC is

$$\begin{aligned}&\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_{t}^{*,i}\sim P_{\boldsymbol{\Theta}_{\mathcal{T}}}}\left[\mathcal{L}_{\text{meta-tt}}\left(\boldsymbol{\theta}_{t}^{*}|_{t=t_0}^{p},\widetilde{\boldsymbol{\theta}}_{t}|_{t=t_0}^{q}\right)\right],\\ &s.t.\quad \widetilde{\boldsymbol{\theta}}_{\mathcal{S}}^{*}=\arg\min_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{\text{cls}}\left[\text{GNN}_{\widetilde{\boldsymbol{\theta}}}\left(\mathcal{S}\right)\right],\end{aligned} \tag{4.6}$$

where $\mathcal{L}_{\text{cls}}$ is the node classification loss calculated with the cross-entropy on graphs. Compared with the triple-level optimization in Eq. (4.1), the proposed SFGC directly replaces the learnable $\mathbf{A}'$ in Eq. (4.1) with a fixed identity matrix $\mathbf{I}$, resulting in the condensed structure-free graph data

$\mathcal{S} = (\widetilde{\mathbf{X}}, \mathbf{I}, \widetilde{\mathbf{Y}})$. Without synthesizing condensed graph structures with $\text{GSL}_{\boldsymbol{\psi}}$, the proposed SFGC refines the complex triple-level optimization to the bi-level one, ensuring the effectiveness of the condensed graph-free data.

Hence, the advances of the training trajectory meta-matching schema in the proposed SFGC can be summarized as follows: (1) compared with the online gradient calculation, SFGC's offline parameter sampling avoids dynamically computing and storing gradients of both the large and condensed small graphs, reducing computation and memory costs during the condensation process; (2) compared with short-range matching, SFGC long-term meta-matching avoids condensed data to short-sightedly fit certain optimization steps, contributing to a more holistic and comprehensive way to imitate GNN's learning behaviors.

4.2.4 Graph Neural Feature Score

For each update of the outer loop in Eq. (4.6), we would synthesize the brand-new condensed graph-free data. However, evaluating the quality of the underlying condensed graph-free data in the dynamical meta-matching condensation process is quite challenging. That is because we cannot quantity a graph dataset's performance without blending it in a GNN model. And the condensed graph-free data itself cannot be measured by convergence or decision boundary. Generally, to evaluate the condensed graph-free data, we use it to train a GNN model. If the condensed data at a certain meta-matching step achieves better GNN test performance on node classification, it indicates the higher quality of the current condensed data. That means, evaluating condensed graph-free data needs an extra process of training a GNN model from scratch, leading to much more time and computation costs.

In light of this, we aim to derive a metric to dynamically evaluate the condensed graph-free data in the meta-matching process, and utilize it to select the optimal small-scale graph-free data. Meanwhile,

Algorithm 1 Structure-Free Graph Condensation (SFGC)

Require: (1) $P_{\boldsymbol{\Theta}_\mathcal{T}}$: Pretrained a set of K numbers of GNNs on the large-scale graph $\text{GNN}_\mathcal{T}$ parameterized by $\boldsymbol{\Theta}_\mathcal{T}$; (2) T_0: numbers of meta-matching steps; (3) T_1: $\text{GNN}_\mathcal{S}$ training steps.

Ensure: A small-scale condensed graph-free data $\mathcal{S} = \left(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}\right)$

1: **while** $\eta < T_0$ **do**
2: randomly sample a pretrained training trajectory in $P_{\boldsymbol{\Theta}_\mathcal{T}}$ and calculate the $\mathcal{L}_{\text{meta-tt}}$ according to Eq. (4.5);
3: **for** $t < T_1$ **do**
4: train $\text{GNN}_\mathcal{S}$ through gradient descent by $\widetilde{\boldsymbol{\theta}}_\mathcal{S}^{t+1} \leftarrow \widetilde{\boldsymbol{\theta}}_\mathcal{S}^{t} - \zeta \nabla_{\widetilde{\boldsymbol{\theta}}} \mathcal{L}_{\text{cls}} \left[\text{GNN}_\mathcal{S}\left(\widetilde{\mathbf{X}}, \mathbf{I}\right), \widetilde{\mathbf{Y}}\right]$, where ζ is the step size;
5: **end for**
6: update the condensed graph-free data $\mathcal{S}_\eta$ according to Eq. (4.4);
7: calculate current η-th step $\gamma_{\text{gnf}}(\mathcal{S}_\eta)$ according to Eq. (4.7);
8: **end while**
9: select the optimal condensed graph-free data $\mathcal{S}_\eta^*$ with the smallest score γ_{gnf}^* as the final output.

the evaluation progress would not introduce extra GNN iterative training for saving computation and time. To achieve this goal, we first identify what characteristics such a metric should have: (1) closed-form solutions of GNNs to avoid iterative training in evaluation; (2) the ability to build strong connections between the large-scale graph and the small-scale synthesized graph-free data. In this case, the graph neural tangent kernel (GNTK) stands out, as a typical class of graph kernels, and has the full expressive power of GNNs with provable closed-form solutions. Moreover, as shown in Eq. (4.2), GNTK naturally builds connections between arbitrary two graphs even with different sizes.

Based on the graph kernel method with GNTK, we proposed a graph neural feature score metric γ_{gnf} to dynamically evaluate and select the optimal condensed graph-free data as follows:

$$\gamma_{\text{gnf}}(\mathcal{S}) = \frac{1}{2}\left\|\mathbf{Y}_{\text{val}} - \boldsymbol{\mathcal{K}}\left\langle\mathcal{T}_{\text{val}}, \mathcal{S}\right\rangle\left(\boldsymbol{\mathcal{K}}\left\langle\mathcal{S}, \mathcal{S}\right\rangle + \lambda\mathbf{I}\right)^{-1}\widetilde{\mathbf{Y}}\right\|^2, \tag{4.7}$$

where $\boldsymbol{\mathcal{K}}\left\langle\mathcal{T}_{\text{val}}, \mathcal{S}\right\rangle \in \mathbb{R}^{N_{\text{val}}\times N'}$ and $\boldsymbol{\mathcal{K}}\left\langle\mathcal{S}, \mathcal{S}\right\rangle \in \mathbb{R}^{N'\times N'}$ denote the node-level GNTK matrices derived according to Eq. (4.2). And $\mathcal{T}_{\text{val}}$ is the validation sub-graph of the large-scale graph with N_{val} numbers of nodes. Concretely, $\gamma_{\text{gnf}}(\mathcal{S})$ calculates the graph neural tangent kernel based ridge regression error. It measures that, given an infinitely-wide GNN trained on the condensed graph $\mathcal{S}$ with ridge regression, how close such GNN's prediction on $\mathcal{T}_{\text{val}}$ to its ground truth labels $\mathbf{Y}_{\text{val}}$. Note that Eq. (4.7) can be regarded as the Kernel Inducing Point (KIP) algorithm [120, 128] adapted to the GNTK kernel on GNN models.

Hence, the proposed graph neural feature score meets the above-mentioned characteristics as: (1) it calculates a closed-form GNTK-based ridge regression error for evaluation without iteratively training GNN models; (2) it strongly connects the condensed graph-free data with the large-scale validation graph; In summary, the overall algorithm of the proposed SFGC is presented in Algo. 1.

4.3 More Analysis of Structure-free Paradigm

In this section, we theoretically analyze the rationality of the proposed structure-free paradigm from the views of statistical learning and information flow, respectively.

The View of Statistical Learning. We start from the graph condensation optimization objective of synthesizing graph structures in Eq. (4.3). Considering its inner loops $\boldsymbol{\theta}_{\mathcal{T}'} = \arg\min_{\boldsymbol{\theta}}\mathcal{L}\left[\text{GNN}_{\boldsymbol{\theta}}\left(\mathbf{X}', \mathbf{A}'\right), \mathbf{Y}'\right]$ with $\mathbf{A}' = \text{GSL}_{\boldsymbol{\psi}}\left(\mathbf{X}'\right)$, it equals to learn the conditional probability $Q(\mathbf{Y}' \mid \mathcal{T}')$ given the condensed

graph $\mathcal{T}' = (\mathbf{X}', \mathbf{A}', \mathbf{Y}')$ as

$$\begin{aligned} Q(\mathbf{Y}' \mid \mathcal{T}') \approx & \sum_{\mathbf{A}' \in \boldsymbol{\psi}(\mathbf{X}')} Q(\mathbf{Y}' \mid \mathbf{X}', \mathbf{A}') Q(\mathbf{A}' \mid \mathbf{X}') \\ = & \sum_{\mathbf{A}' \in \boldsymbol{\psi}(\mathbf{X}')} Q(\mathbf{X}', \mathbf{A}', \mathbf{Y}') / Q(\mathbf{X}', \mathbf{A}') \cdot Q(\mathbf{X}', \mathbf{A}') / Q(\mathbf{X}') \\ = & \sum_{\mathbf{A}' \in \boldsymbol{\psi}(\mathbf{X}')} Q(\mathbf{X}', \mathbf{A}', \mathbf{Y}') / Q(\mathbf{X}') \\ = & Q(\mathbf{X}', \mathbf{Y}') / Q(\mathbf{X}') = Q(\mathbf{Y}' \mid \mathbf{X}'), \end{aligned} \tag{4.8}$$

where we simplify the notation of graph structure learning module $\mathrm{GSL}_{\boldsymbol{\psi}}$ as parameterized $\boldsymbol{\psi}(\mathbf{X}')$. As can be observed, when the condensed graph structures are learned from the condensed nodes as $\mathbf{A}' \in \boldsymbol{\psi}(\mathbf{X}')$, the optimization objective of the conditional probability is not changed, while its goal is still to solve the posterior probability $Q(\mathbf{Y}' \mid \mathbf{X}')$. In this way, eliminating graph structures to conduct structure-free condensation is rational from the view of statistical learning. By directly synthesizing the graph-free data, the proposed SFGC could ease the optimization process and directly transfer all the informative knowledge of the large-scale graph to the condensed graph node set without structures. Hence, the proposed SFGC conducts more compact condensation to derive the small-scale graph-free data, whose node attributes already integrate implicit topology structure information.

The View of Information Flow. For training on large-scale graphs to obtain offline parameter trajectories, we solve the node classification task on $\mathcal{T} = (\mathbf{X}, \mathbf{A}, \mathbf{Y})$ with a certain GNN model as

$$\boldsymbol{\theta}_{\mathcal{T}}^{*} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathrm{cls}} \left[\mathrm{GNN}_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{A}), \mathbf{Y} \right], \tag{4.9}$$

where $*$ denotes the optimal training parameters that build the training trajectory distribution $P_{\boldsymbol{\Theta}_{\mathcal{T}}}$. The whole graph information, *i.e.*, node attributes $\mathbf{X}$ and topology structures $\mathbf{A}$ are both embedded in the latent space of GNN network parameters. Hence, the large-scale graph information flows to GNN parameters as $(\mathbf{X}, \mathbf{A}) \Rightarrow P_{\boldsymbol{\Theta}_{\mathcal{T}}}$. In this way, by meta-sampling in the trajectory distribution, the proposed SFGC explicitly transfers learning behaviors of the large-scale graph to the parameter space $\widetilde{\boldsymbol{\theta}}_{\mathcal{S}}$ of $\mathrm{GNN}_{\mathcal{S}}$ as $P_{\boldsymbol{\Theta}_{\mathcal{T}}} \Rightarrow \widetilde{\boldsymbol{\theta}}_{\mathcal{S}}$. As a result, the informative knowledge of the large-scale graphs, *i.e.*, node attributes and topology structure information $(\mathbf{X}, \mathbf{A})$, would be comprehensively transferred as $(\mathbf{X}, \mathbf{A}) \Rightarrow P_{\boldsymbol{\Theta}_{\mathcal{T}}} \Rightarrow \widetilde{\boldsymbol{\theta}}_{\mathcal{S}}$. In this way, we could identify the critical goal of graph condensation is to further transfer the knowledge in $\widetilde{\boldsymbol{\theta}}_{\mathcal{S}}$ to the output condensed graph data as:

$$\begin{aligned} &(\mathbf{X}, \mathbf{A}) \Rightarrow P_{\boldsymbol{\Theta}_{\mathcal{T}}} \Rightarrow \boldsymbol{\Theta}_{\mathcal{S}} \Rightarrow \mathcal{T}' = (\mathbf{X}', \mathbf{A}'), \quad \text{GC.} \\ &(\mathbf{X}, \mathbf{A}) \Rightarrow P_{\boldsymbol{\Theta}_{\mathcal{T}}} \Rightarrow \boldsymbol{\Theta}_{\mathcal{S}} \Rightarrow \mathcal{S} = (\widetilde{\mathbf{X}}), \quad \text{SFGC.} \end{aligned} \tag{4.10}$$

where GC and SFGC correspond to the existing graph condensation and the proposed structure-free graph condensation, respectively.

Hence, from the view of information flow, we could observe that condensing structures would not inherit more information from the large-scale graph. Compared with GC which formulates the condensed graph into nodes and structures, the proposed SFGC directly distills all the large-scale graph knowledge into the small-scale graph node set without structures. Consequently, the proposed SFGC conducts more compact condensation to derive the small-scale graph-free data, which implicitly encodes the topology structure information into the discriminative node attributes.

4.4 Experiments

4.4.1 Experimental Settings

Datasets. Following [64], we evaluate the node classification performance of the proposed SFGC method on Cora, Citeseer [129], and Ogbn-arxiv [130] under the transductive setting, on Flickr [56] and Reddit [7] under the inductive setting. For all datasets under two settings, we use the public splits and setups for fair comparisons. We consider three condensation ratios (r) for each dataset. Concretely, r is the ratio of condensed node numbers $rN(0 < r < 1)$ to large-scale node numbers N. In the transductive setting, N represents the number of nodes in the entire large-scale graph, while in the inductive setting, N indicates the number of nodes in the training sub-graph of the whole large-scale graph. We provide the details of the original dataset statistics in Table 4.1.

TABLE 4.1: Details of dataset statistics.

Datasets	#Nodes	#Edges	#Classes	#Features	Train/Val/Test
Cora	2,708	5,429	7	1,433	140/500/1000
Citeseer	3,327	4,732	6	3,703	120/500/1000
Ogbn-arxiv	169,343	1,166,243	40	128	90,941/29,799/48,603
Flickr	89,250	899,756	7	500	44,625/22312/22313
Reddit	232,965	57,307,946	41	602	15,3932/23,699/55,334

Baselines & Implementations. We adopt the following baselines for comprehensive comparisons [64]: graph coarsening method [131], graph coreset methods, *i.e.*, Random, Herding [59], and K-Center [132], the graph-based variant DC-Graph of general dataset condensation method DC [121], which is introduced by [64], graph dataset condensation method GCOND [64] and its variant GCOND-X without utilizing the graph structure. The whole pipeline of our experimental evaluation can be divided into two stages: (1) the condensation stage: synthesizing condensed graph-free data, where we use the classical and commonly-used GCN model [129]; (2) the condensed graph-free data test stage: training a certain GNN model (default with GCN) by the obtained condensed graph-free data from the first stage and testing the GNN on the test set of the large-scale graph with repeated 10 times. We report the average transductive and inductive node classification accuracy (ACC%) with standard deviation (std). Following [64], we use the two-layer GNN with 256 hidden units as the defaulted

TABLE 4.2: Node classification performance (ACC%±std) comparison between other graph size reduction methods and our proposed SFGC under different condensation ratios. (Best results are in bold, and the second-bests are underlined.)

Datasets	Ratio (r)	Coarsening [131]	Random [59]	Herding [59]	K-Center [132]	**SFGC (ours)**	Whole Dataset
Citeseer	0.9%	52.2±0.4	54.4±4.4	57.1±1.5	52.4±2.8	**71.4**±0.5	71.7±0.1
	1.8%	59.0±0.5	64.2±1.7	66.7±1.0	64.3±1.0	**72.4**±0.4	
	3.6%	65.3±0.5	69.1±0.1	69.0±0.1	69.1±0.1	**70.6**±0.7	
Cora	1.3%	31.2±0.2	63.6±3.7	67.0±1.3	64.0±2.3	**80.1**±0.4	81.2±0.2
	2.6%	65.2±0.6	72.8±1.1	73.4±1.0	73.2±1.2	**81.7**±0.5	
	5.2%	70.6±0.1	76.8±0.1	76.8±0.1	76.7±0.1	**81.6**±0.8	
Ogbn-arxiv	0.05%	35.4±0.3	47.1±3.9	52.4±1.8	47.2±3.0	**65.5**±0.7	71.4±0.1
	0.25%	43.5±0.2	57.3±1.1	58.6±1.2	56.8±0.8	**66.1**±0.4	
	0.5%	50.4±0.1	60.0±0.9	60.4±0.8	60.3±0.4	**66.8**±0.4	
Flickr	0.1%	41.9±0.2	41.8±2.0	42.5±1.8	42.0±0.7	**46.6**±0.2	47.2±0.1
	0.5%	44.5±0.1	44.0±0.4	43.9±0.9	43.2±0.1	47.0±0.1	
	1%	44.6±0.1	44.6±0.2	44.4±0.6	44.1±0.4	**47.1**±0.1	
Reddit	0.05%	40.9±0.5	46.1±4.4	53.1±2.5	46.6±2.3	**89.7**±0.2	93.9±0.0
	0.1%	42.8±0.8	58.0±2.2	62.7±1.0	53.0±3.3	**90.0**±0.3	
	0.2%	47.4±0.9	66.3±1.9	71.0±1.6	58.5±2.1	89.9±0.4	

TABLE 4.3: Node classification performance (ACC%±std) comparison between condensation methods and our proposed SFGC under different condensation ratios. (Best results are in bold, and the second-bests are underlined.)

Datasets	Ratio (r)	DC-Graph [121]	GCOND-X [64]	GCOND [64]	**SFGC (ours)**	Whole Dataset
Citeseer	0.9%	66.8±1.5	71.4±0.8	70.5±1.2	**71.4**±0.5	71.7±0.1
	1.8%	66.9±0.9	69.8±1.1	70.6±0.9	**72.4**±0.4	
	3.6%	66.3±1.5	69.4±1.4	69.8±1.4	**70.6**±0.7	
Cora	1.3%	67.3±1.9	75.9±1.2	79.8±1.3	**80.1**±0.4	81.2±0.2
	2.6%	67.6±3.5	75.7±0.9	80.1±0.6	**81.7**±0.5	
	5.2%	67.7±2.2	76.0±0.9	79.3±0.3	**81.6**±0.8	
Ogbn-arxiv	0.05%	58.6±0.4	61.3±0.5	59.2±1.1	**65.5**±0.7	71.4±0.1
	0.25%	59.9±0.3	64.2±0.4	63.2±0.3	**66.1**±0.4	
	0.5%	59.5±0.3	63.1±0.5	64.0±0.4	**66.8**±0.4	
Flickr	0.1%	46.3±0.2	45.9±0.1	46.5±0.4	**46.6**±0.2	47.2±0.1
	0.5%	45.9±0.1	45.0±0.2	**47.1**±0.1	47.0±0.1	
	1%	45.8±0.1	45.0±0.1	**47.1**±0.1	**47.1**±0.1	
Reddit	0.05%	88.2±0.2	88.4±0.4	88.0±1.8	**89.7**±0.2	93.9±0.0
	0.1%	89.5±0.1	89.3±0.1	89.6±0.7	**90.0**±0.3	
	0.2%	**90.5**±1.2	88.8±0.4	90.1±0.5	89.9±0.4	

setting. Besides, we adopt the K-center [132] features to initialize our condensed node attributes for stabilizing the training process.

4.4.2 Experimental Results

Performance of SFGC on Node Classification. We compare the node classification performance between SFGC and other graph size reduction methods, especially the graph condensation methods. The overall performance comparisons are listed in Table 4.2 and Table 4.3. Generally, SFGC

achieves the best performance on the node classification task with 13 of 15 cases (five datasets and three condensation ratios for each of them), compared with all other baseline methods, illustrating the high quality and expressiveness of the condensed graph-free data synthesized by our SFGC. More specifically, the better performance of SFGC than GCOND and its structure-free variant GCOND-X experimentally verifies the superiority of the proposed method. We attribute such superiority to the following two aspects regarding the condensation stage. First, the long-term parameter distribution matching of our SFGC works better than the short-term gradient matching in GCOND and GCOND-X. That means capturing the long-range GNN learning behaviors facilitates holistically imitating GNN's training process, leading to comprehensive knowledge transfer from the large-scale graph to the small-scale condensed graph-free data. Second, the structure-free paradigm of our SFGC enables more compact small-scale graph-free data. For one thing, it liberates the optimization process from triple-level objectives, alleviating the complexity and difficulty of condensation. For another thing, the obtained optimal condensed graph-free data compactly integrates node attribute contexts and topology structure information. Furthermore, on Cora and Citeseer, SFGC synthesizes better condensed graph-free data that even exceeds the whole large-scale graph dataset. These results confirm that SFGC can break the information limitation under the large-scale graph and effectively synthesize new, small-scale graph-free data as an optimal representation of the large-scale graph.

Effectiveness of Structure-free Paradigm in SFGC. The proposed SFGC introduces the structure-free paradigm without condensing graph structures in graph condensation. To verify the effectiveness of the structure-free paradigm, we compare the proposed SFGC with its variants, which synthesize graph structures in the condensation process. Specifically, we evaluate the following three different methods of synthesizing graph structures with five variants of SFGC: (1) discrete k-nearest neighbor (kNN) structures calculated by condensed node features under $k = (1, 2, 5)$, corresponding to the variants SFGC-d1, SFGC-d2, and SFGC-d5; (2) cosine similarity based continuous graph structures calculated by condensed node features, corresponding to the variant SFGC-c; (3) parameterized graph structure learning module with condensed node features adapted by [64], corresponding to the variant SFGC-p. We conduct experiments on three transductive datasets under nine condensation ratios for each graph structure synthesis variant and the proposed SFGC. The results are presented in Fig. 4.3. In general, the proposed SFGC achieves the best performance over various graph structure

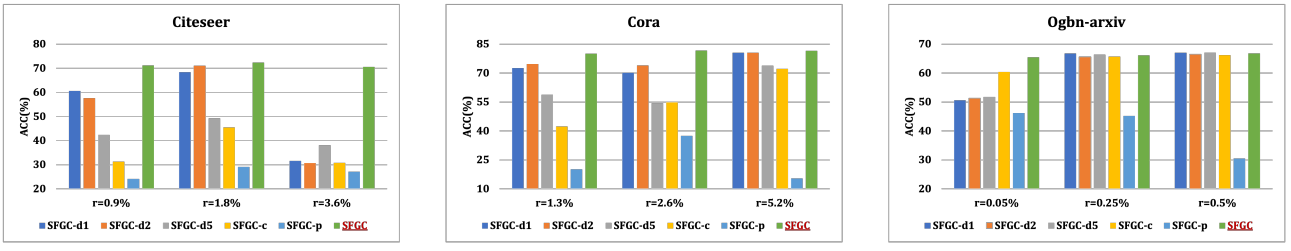

FIGURE 4.3: Comparisons between five variants of synthesizing graph structures *vs.* our structure-free SFGC (discrete k-nearest neighbor (kNN) structure variants: SFGC-d1 ($k = 1$), SFGC-d2 ($k = 2$), and SFGC-d5 ($k = 5$), continuous graph structure variant: SFGC-c, parameterized graph structure variant: SFGC-p).

TABLE 4.4: Performance across different GNN architectures.

Datasets (Ratio)	Methods	MLP	GAT [133]	APPNP [134]	Cheby [135]	GCN [129]	SAGE [7]	SGC [136]	Avg.
Citeseer ($r = 1.8\%$)	DC-Graph [121]	66.2	-	66.4	64.9	66.2	65.9	69.6	66.6
	GCOND-X [64]	69.6	-	69.7	70.6	69.7	69.2	71.6	70.2
	GCOND [64]	63.9	55.4	69.6	68.3	70.5	66.2	70.3	69.0
	SFGC (ours)	**71.3**	**72.1**	**70.5**	**71.8**	**71.6**	**71.7**	**71.8**	**71.5**
Cora ($r = 2.6\%$)	DC-Graph [121]	67.2	-	67.1	67.7	67.9	66.2	72.8	68.3
	GCOND-X [64]	76.0	-	77.0	74.1	75.3	76.0	76.1	75.7
	GCOND [64]	73.1	66.2	78.5	76.0	80.1	78.2	79.3	78.4
	SFGC (ours)	**81.1**	**80.8**	**78.8**	**79.0**	**81.1**	**81.9**	**79.1**	**80.3**
Ogbn-arxiv ($r = 0.25\%$)	DC-Graph [121]	59.9	-	60.0	55.7	59.8	60.0	60.4	59.2
	GCOND-X [64]	64.1	-	61.5	59.5	64.2	64.4	64.7	62.9
	GCOND [64]	62.2	60.0	63.4	54.9	63.2	62.6	63.7	61.6
	SFGC (ours)	**65.1**	**65.7**	**63.9**	**60.7**	**65.1**	**64.8**	**64.8**	**64.3**
Flickr ($r = 0.5\%$)	DC-Graph [121]	43.1	-	45.7	43.8	45.9	45.8	45.6	45.4
	GCOND-X [64]	42.1	-	44.6	42.3	45.0	44.7	44.4	44.2
	GCOND [64]	44.8	40.1	**45.9**	42.8	**47.1**	46.2	**46.1**	**45.6**
	SFGC (ours)	**47.1**	**45.3**	40.7	**45.4**	**47.1**	**47.0**	42.5	45.0
Reddit ($r = 0.1\%$)	DC-Graph [121]	50.3	-	81.2	77.5	89.5	89.7	90.5	85.7
	GCOND-X [64]	40.1	-	78.7	74.0	89.3	89.3	**91.0**	84.5
	GCOND [64]	42.5	60.2	87.8	75.5	89.4	89.1	89.6	86.3
	SFGC (ours)	**89.5**	**87.1**	**88.3**	**82.8**	**89.7**	**90.3**	89.5	**88.2**

synthesis methods, and these results empirically verify the effectiveness of the proposed structure-free condensation paradigm. More specifically, for discrete k-nearest neighbor (kNN) structure variants, different datasets adapt different numbers of k-nearest neighbors under different condensation ratios, which means predefining the value of k can be very challenging. For example, Citeseer dataset has better performance with $k = 1$ in SFGC-d1 under $r = 0.9\%$ than SFGC-d2 and SFGC-d5, but under $r = 1.8\%$, $k = 2$ in SFGC-d2 performs better than others two. Besides, for continuous graph structure variant SFGC-c, it generally cannot exceed the discrete graph structure variants, except for Ogbn-arxiv dataset under $r = 0.05\%$. And the parameterized variant SFGC-p almost fails to synthesize satisfied condensed graphs under the training trajectory meta-matching scheme. The superior performance of SFGC to all structure-based methods demonstrates the effectiveness of its structure-free paradigm.

Effectiveness of Graph Neural Feature Score in SFGC. We compare the learning time between GNN iterative training *vs.*. our proposed GNTK-based closed-form solutions of γ_{gnf}. Note that the iterative training evaluation strategy mandates the complete training of a GNN model from scratch at each meta-matching step, hence, we calculate its time that covers all training epochs under the best test performance for fair comparisons. Typically, for Flickr dataset ($r = 0.1\%$), our proposed γ_{gnf} based GNTK closed-form solutions takes only 0.015s for dynamic evaluation, which significantly outperforms the iterative training evaluation with 0.845s. The superior performance can also be observed in Ogbn-arxiv dataset ($r = 0.05\%$) with 0.042s of our γ_{gnf}, compared with 4.264s of iterative training, illustrating our SFGC's high dynamic evaluation efficiency.

Generalization Ability of SFGC across Different GNNs. We evaluate and compare the generalization ability of the proposed SFGC and other graph condensation methods. Concretely, we test the node classification performance of our synthesized graph-free data (condensed on GCN) with seven different GNN architectures: MLP, GAT [133], APPNP [134], Cheby [135], GCN [129], SAGE [7], and SGC [136]. It can be generally observed that the proposed SFGC achieves outstanding performance over all tested GNN architectures, reflecting its excellent generalization ability. This is because our method reduces the graph structure to the identity matrix, so that the condensed graph node set can no longer be influenced by different convolution operations of GNNs along graph structures, enabling it consistent and good performance with various GNNs.

4.5 Potential Application Scenarios

We would like to highlight the significance of graph condensation task to various application scenarios within the research field of dataset distillation/condensation, while comprehensive overviews can be found in survey works [67, 119]. Specifically, we present several potential scenarios where our proposed structure-free graph condensation method could bring benefits:

Graph Neural Architecture Search. Graph neural architecture search (GraphNAS) aims to develop potential and expressive GNN architectures beyond existing human-designed GNNs. By automatically searching in a space containing various candidate GNN architecture components, GraphNAS could derive powerful and creative GNNs with superior performance on specific graph datasets for specific tasks [25, 26, 40, 41, 137]. Hence, GraphNAS needs to repeatedly train different potential GNN architectures on the specific graph dataset, and ultimately selects the optimal one. When in the large-scale graph, this would incur severe computation and memory costs. In this case, searching on our developed small-scale condensed graph-free data, a representative substitution of the large-scale graph, could significantly benefit for saving many computation costs and accelerating new GNN architecture development in GraphNAS research field.

Privacy Protection. Considering the outsourcing scenario of graph learning tasks, the original large-scale graph data is not allowed to release due to privacy, for example, patients expect to use GNNs for medical diagnosis without their personal medical profiles being leaked [138, 139]. In this case, as a compact and representative substitution, the synthesized small-scale condensed graph could be used to train GNN models, so that the private information of the original graph data can be protected. Besides, considering the scenario that over-parameterized GNNs might easily memorize training data, inferring the well-trained models could cause potential privacy leakage issue. In this case, we could release a GNN model trained by the synthesized small-scale condensed graph, so that the model avoids explicitly training on the original large-scale graph and consequently helps protect its data privacy.

Adversarial Robustness. In practical applications, GNNs might be attacked with disrupted performance, when attackers impose adversarial perturbations to the original graph data [5], for instance, poisoning attacks on graph data [140–142], where attackers attempt to alter the edges and nodes of training graphs of a target GNN. Training on poisoned graph data could significantly damage GNNs' performance. In this case, given a poisoned original training graph, graph condensation could synthesize a new condensed graph from it, which we use to train the target GNN would achieve comparable test performance with that trained by the original training graph before being poisoned. Hence, the new condensed graph could eliminate adversarial samples in the original poisoned graph data with great adversarial robustness, so that using it to train a GNN would not damage its performance for inferring test graphs.

Continual learning. Continual learning (CL) aims to progressively accumulate knowledge over a continuous data stream to support future learning while maintaining previously learned information [143–145]. One of key challenges of CL is catastrophic forgetting [146, 147], where knowledge extracted and learned from old data/tasks are easily forgotten when new information from new data/-tasks are learned. Some works have studied that data distillation/condensation is an effective solution to alleviate catastrophic forgetting [148–151], where the distilled and condensed data is taken as representative summary stored in a replay buffer that is continually updated to instruct the training of subsequent data/tasks.

To summarize, graph condensation task holds great promise and is expected to bring significant benefits to various graph learning tasks and applications. By producing compact, high-quality, small-scale condensed graph data, graph condensation has the potential to enhance the efficiency and effectiveness of future graph machine learning works.

4.6 Conclusion

This chapter proposes a novel Structure-Free Graph Condensation paradigm, named SFGC, to distill the large-scale graph into the small-scale graph-free node set without graph structures. Under the structure-free learning paradigm, the training trajectory meta-matching scheme and the graph neural feature score measured dynamic evaluation work collaboratively to synthesize small-scale graph-free data with superior effectiveness and good generalization ability. Extensive experimental results and analysis under large condensation ratios confirm the superiority of the proposed SFGC method in synthesizing excellent small-scale graph-free data. It can be anticipated that our work would bridge the gap between academic GNNs and industrial MLPs by synthesizing small-scale, graph-free data to address graph data scalability, while retaining the expressive performance of graph learning. Our method works on condensing the number of nodes in a single graph at the node level, and we will explore extending it to condense the number of graphs in a graph set at the graph level in the future.

We will also explore the potential of unifying graphs and large language models [152] for the graph condensation task.

Chapter 5

Automated Graph Neural Networks on Heterophilic Graphs

5.1 Introduction

Current mainstream research on graph neural networks (GNNs) principally builds upon the following two foundations: the homophily assumption of graph-structured data and the expertise dependency of GNN architecture design [5, 14, 17, 24, 75–78]. At first, the homophily assumption defines that nodes with similar features or same class labels are linked together. For instance, research articles within the same research area would be linked more likely by citing each other. In contrast, *heterophily*, the property that linked nodes have dissimilar features and different class labels, does not attract equal attention but widely exists in various web-related real-world applications [19, 21, 29, 84, 85]. For instance, in online transaction networks, fraudsters are more likely to build connections with customers instead of other fraudsters [84]. At this point, heterophily of graphs plays an important role in web socio-economic system, and modeling the heterophily explicitly would benefit the development of web techniques and infrastructure. Due to the differences of structure and property between homophilic and heterophilic graphs, existing GNNs, which target for modelling homophily, cannot be directly applied to heterophilic graphs, and the main reasons lie in the following two aspects [21]: (1) local *vs.*. non-local neighbors: homophilic GNNs concentrate more on addressing local neighbor nodes within the same class, while on heterophilic graphs, informative same-class neighbors usually are non-local in graph topology; (2) uniform *vs.*. diverse aggregation: homophilic GNNs uniformly aggregate information from similar local neighbors and then accordingly update central (ego) nodes, while heterophilic graphs expect discriminative node representation learning to diversely extract information from similar and dissimilar neighbors.

Very recently, a few researchers divert their attention to develop heterophilic GNNs, typically by introducing higher-order neighbors [29] or modelling homophily and heterophily separately in the

message passing scheme [3]. Nevertheless, current heterophilic GNNs highly rely on the knowledge of experts to manually design models for graphs with different degrees of heterophily. For one thing, this process would cost many human efforts and the performance of derived GNNs would be limited by expertise. For another thing, real-world heterophilic graphs generally show the significant complexity and variety in terms of graph structure. It would be harder to artificially customize GNNs to make them adapt to various heterophilic graphs.

In light of this, automated graph neural network learning via neural architecture search (NAS) [26, 42, 44–48, 51], a line of research for saving human efforts in designing effective GNNs, would be a feasible way of tackling the dilemma of heterophilic GNN development. Through learning in well-designed search spaces with efficient search strategies, automated GNNs via NAS have achieved promising progress on various graph data analysis tasks [40, 41, 43, 153–155]. However, existing graph NAS studies, *e.g.*, GraphNAS [40] and SANE [41], are all constrained under the homophily assumption. To the best of our knowledge, there is still a research blank in developing heterophilic graph NAS for automated heterophilic graph learning. The most straightforward way to implement heterophilic graph NAS might be replacing the homophilic operations in existing homophilic search spaces with heterophilic GNN operations, followed by universal gradient-based search strategies. But this direct solution would incur some issues: First, it is difficult to determine what heterophilic operations are beneficial and whether certain homophilic operations should be kept for facilitating heterophilic graph learning, since the heterophilic graphs might be various and complex with different degrees of heterophily. Second, simply making existing universal search strategies adapt to new-defined heterophilic search spaces would limit the searching performance, resulting in suboptimal heterophilic GNN architectures. The heterophily should be integrated into the searching process for architecture optimization as well.

To tackle all the above-mentioned challenges of heterophilic graph NAS, in this chapter, we propose a novel automated graph neural network on heterophilic graphs, namely Auto-HeG, to effectively learn heterophilic node representations via heterophily-aware graph neural architecture search. To the best of our knowledge, this is the first automated heterophilic graph learning method, and our theme is to explicitly incorporate heterophily into all stages of automatic heterophily graph learning, containing search space design, supernet training, and architecture selection.

For search space design: By integrating joint micro-level and macro-level designs, Auto-HeG first builds a comprehensive heterophilic search space through the diverse message-passing scheme, enabling it to incorporate the complexity and diversity of heterophily better. At the micro-level, Auto-HeG conducts non-local neighbor extension, ego-neighbor separation, and diverse message passing; At the macro-level, it introduces adaptive layer-wise combination operations.

For supernet training: To narrow the scope of candidate operations in the proposed heterophilic search space, Auto-HeG presents a progressive supernet training strategy to dynamically shrink the

initial search space according to layer-wise variation of heterophily, resulting in a compact and efficient supernet.

For architecture selection: Taking heterophily as the specific guidance, Auto-HeG derives a novel heterophily-aware distance as the criterion to select effective operations in the leave-one-out pattern, leading to specialized and expressive heterophilic GNN architectures. Extensive experiments on the node classification task illustrate the superior performance of the proposed Auto-HeG to human-designed models and graph NAS models. In summary, our contributions are listed as follows:

- We propose a novel automated graph neural network on heterophilic graphs by means of heterophily-aware graph neural architecture search, namely **Auto-HeG**, to the best of our knowledge, for the first time.
- To integrate the complex and various heterophily explicitly, we build a comprehensive heterophilic GNN search space that incorporates non-local neighbor extension, ego-neighbor separation, diverse message passing, and layer-wise combination at the micro-level and the macro-level.
- To learn a compact and effective heterophilic supernet, we introduce a progressive supernet training strategy to dynamically shrink the initial search space, enabling the narrowed searching scope with layer-wise heterophily variation.
- To select optimal GNN architectures specifically instructed by heterophily, we derive a heterophily-aware distance criterion to develop powerful heterophilic GNNs in the leave-one-out manner and extensive experiments verify the superiority of Auto-HeG.

5.2 Related work

5.2.1 Graph Neural Networks with Heterophily.

Existing heterophilic GNNs mainly work on two aspects [21]: non-local neighbor extension [20, 29–33] and GNN architecture refinement [3, 34–39]. In detail, non-local neighbor extension methods focus on exploring the informative neighbor set beyond local topology. In contrast, GNN architecture refinement methods design heterophily-specific message-passing models to learn discriminative node representations. Typically, Mixhop [20] and H2GCN [29] introduced higher-order neighbor mixing of local one-hop neighbors and non-local K-hop neighbors to learn discriminative central node representations. And Geom-GCN [31] defined the split 2D Euclidean geometry locations as the geometric relationships with different degrees of heterophily, enabling it to discover potential neighbors for further effective aggregation. In contrast, FAGCN [3] developed diverse aggregation functions by introducing low-pass and high-pass filters, corresponding to learning homophily and heterophily, respectively. Besides, GCNII [38] and GPR-GNN [39] considered the layer-wise representation integration to boost the performance of GNNs with heterophily. Despite the promising development, these

heterophilic GNNs highly rely on the expertise to manually design GNNs for tackling the complex and diverse heterophily. For one thing, it would cost exhausted human efforts. For another thing, the design scopes and flexibility would be constrained by expert knowledge, leading to limited model performance. In light of this, we are the first to propose an automated graph neural architecture search framework for heterophilic graphs, to significantly save human efforts and automatically derive powerful heterophilic GNN architectures with excellent learning abilities.

5.2.2 Graph Neural Architecture Search.

Graph NAS has greatly enlarged the design picture of automated GNN development for discovering excellent and powerful models [40, 41, 43, 153–157]. Generally, the development of graph NAS methods focuses on two crucial research aspects: search space and search strategy. The former defines architecture and functional operation candidates in a set space, and the latter explores the powerful model components in the defined search space. Typically, GraphNAS [40] and AGNN [156] constructed the micro-level search space containing classical GNN components and related hyper-parameters, followed by architecture controllers based reinforcement learning (RL) search strategy. And SNAG [157] further simplified GraphNAS at the micro-level and introduced the macro-level inter-layer architecture connections. Based on this search space, SANE [41] implemented DARTS [42], a gradient-based search strategy, to automatically derive effective GNN architectures on graphs in a differential way. Nevertheless, current graph NAS methods are still constrained under the homophily assumption and cannot learn explicitly and effectively on graphs with complex and diverse heterophily. Therefore, we draw inspiration from the lines of NAS research on search space shrinking [44–48], supernet optimization [49, 50], and architecture selection [51], and importantly extend them to the development of automated heterophilic GNNs. Our critical goal is to build a well-designed search space with a customized search strategy via graph NAS, to explicitly integrate the heterophily into full stages of automated heterophilic GNN development.

5.3 Auto-HeG: Automated graph neural network on heterophilic graphs

5.3.1 Preliminary.

Uniform Message Passing. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, unweighted graph where $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ is the node set and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the edge set. The neighbor set of node v is $\mathcal{N}(v) = \{u : (v, u) \in \mathcal{E}\}$, and initial node features are represented by $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_0}$ with d_0-dimension features. For the uniform message passing scheme under the homophily assumption, node representations of GNNs are learned by first aggregating the messages from local neighbors, and then updating the

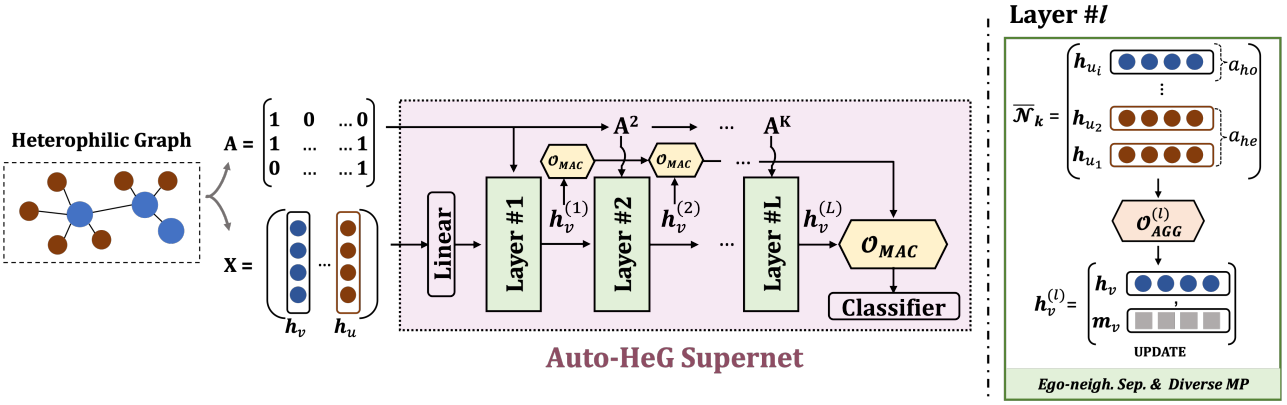

FIGURE 5.1: Overall supernet of the proposed Auto-HeG. $\mathbf{A}$ denotes the adjacency matrix and $\mathbf{A}^k$ denotes the k-th hop neighbors for $k = \{1, 2, \cdots, K\}$, and the right part shows the details of l-th layer with the layer-wise heterophily-specific operation space $\mathcal{O}_{AGG}^{(l)}$ via progressive supernet training.

ego-node representations by combining the aggregated messages with themselves [27]. This process can be denoted as:

$$\begin{aligned} \mathbf{m}_v^{(l)} &= \text{AGG}^{(l)}(\{\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}(v)\}), \\ \mathbf{h}_v^{(l)} &= \text{UPDATE}^{(l)}(\mathbf{h}_v^{(l-1)}, \mathbf{m}_v^{(l)}), \end{aligned} \tag{5.1}$$

where $\mathbf{m}_v^{(l)}$ and $\mathbf{h}_v^{(l)}$ are the message vector and the representation vector of node v at the l-th layer, respectively. AGG$(\cdot)$ and UPDATE$(\cdot)$ are the aggregation function and update function, respectively. Given the input of the first layer, the learned node representations with d_1 dimensions at each layer of L-layer GNN can be denoted as $\mathbf{H}^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times d_1}$ for $l = \{1, 2, \cdots, L\}$.

Measure of Heterophily & Homophily. In general, heterophily and homophily of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be measured by node homophily [31]. Concretely, the node homophily γ_{node} is the average proportion of the neighbors with the same class of each node as: $\gamma_{node} = 1/|\mathcal{V}| \sum_{v \in \mathcal{V}} (|\{u \in \mathcal{N}(v) : y_v = y_u\}|/|\mathcal{N}(v)|)$.

The range of γ_{node} is $[0, 1]$. Graphs with strong homophily have higher γ_{node} (closer to 1); Whereas graphs with strong heterophily have smaller γ_{node} (closer to 0).

5.3.2 Heterophilic Search Space Design

To incorporate the complexity and variety of heterophily on graphs, we design a comprehensive heterophilic search space in the proposed Auto-HeG, involving joint micro-level and macro-level candidate operations. At the micro-level, the proposed search space contains three important components: non-local neighbor extension, ego-neighbor separation, and diverse message passing; While the macro-level consists of intermediate layer combination functions and skip connections for integrating layer-wise node representations. The well-designed search space is the basis for building a supernet for search effective architectures on heterophilic graphs as shown in Fig. 5.1, and we will discuss this further in

TABLE 5.1: Heterophilic search space details of the proposed Auto-HeG. 'homo.' and 'hete.' indicate homophily-related and heterophily-related aggregation functions, respectively.

Search Space	Modules		Operations
Micro-level	Neighbors		$\{A, A^2, \cdots, A^K\}$
	$\mathcal{O}_{AGG}$	homo.	{ `SAGE`, `SAGE_SUM`, `SAGE_MAX`, `GCN`, `GIN`, `GAT`, `GAT_SYM`, `GAT_COS`, `GAT_LIN`, `GAT_GEN_LIN`, `GeniePATH`}
		hete.	{ `GCNII`, `FAGCN`, `GPRGNN`, `SUPERGAT`, `GCN_CHEB`, `APPNP`, `SGC`}
Macro-level	$\mathcal{O}_{MAC}$		*l_skip*, *l_zero*, *l_concat*, *l_max*, *l_lstm*

Sec. 5.3.3. In the following, we mainly give detailed descriptions of the candidate operations in the proposed heterophilic search space, where its summary is listed in Table 5.1.

5.3.2.1 Micro-level Design

Non-local Neighbor Extension. Homophilic graphs generally take local nodes from one hop away as neighbors of the ego node for message aggregation, since the same-class nodes mainly locate in their proximal topology. On the contrary, on heterophilic graphs, nodes within the same class could be far away from each other. That means only considering the one-hop-away neighbors would be insufficient to capture the heterophily on graphs explicitly. Hence, to break the local topology limitation under the homophily assumption, we extend local one-hop neighbors to non-local K-hop neighbors to incorporate heterophilic neighbor information. Concretely, we modify the uniform message passing scheme in Eq. (5.1) in terms of K-hop neighbors as follows:

$$\mathbf{m}_v^{(l)} = \mathrm{AGG}^{(l)}(\{\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}_k(v)\}),\ k = \{1, 2, \cdots, K\}, \tag{5.2}$$

where $\mathcal{N}_k(v)$ denotes the k-th hop neighbor set of node v. In this way, the derived heterophilic GNNs could mix latent information from neighbors within the same class at various distances of graph topology. Specifically, to avoid multi-hop neighbors bringing the exponential explosion of graph scale, we restrict the k-hop neighbor set as neighbors that are connected by at least k different paths to the ego nodes. For instance, a two-hop neighbor set contains neighbors that are two hops away from ego nodes and have at least two different paths to reach the ego nodes. Note that, different from existing heterophilic GNNs (*e.g.*, Mixhop [20] and H2GCN [29]) introducing multi-hop neighbors at each layer and combining them in parallel, we relate the number of neighbor hops to the number of GNN layers correspondingly in the proposed Auto-HeG. The intuition behind this is to alleviate the complexity of the search space but keep its effectiveness at the same time.

Ego-neighbor Separation. Despite capturing the messages from non-local neighbors, some local neighbors still have dissimilar class labels with the ego nodes. Hence, it is necessary to separate the ego node representations from their neighbor representations, as verified by the work [29]. In this way, heterophilic GNNs could learn the ego node features by discriminatively combining the aggregated

neighbor information with themselves at the following update step. At this point, we further modify the uniform message passing scheme at the aggregation step in Eq. (5.2) as

$$\mathbf{m}_v^{(l)} = \mathrm{AGG}^{(l)}(\{\mathbf{h}_u^{(l-1)} : u \in \bar{\mathcal{N}}_k(v)\}),\ k = \{1, 2, \cdots, K\}, \tag{5.3}$$

where $\bar{\mathcal{N}}_k(v)$ denotes the neighbor set that excludes the ego node v, *i.e.*, removing self-loops in the original graph structure.

Diverse Message Passing. Given the extended non-local neighbor set, diverse message-passing scheme is the core of tackling the heterophily on graphs. By distinguishing the information of similar neighbors (likely in the same class) from that of dissimilar neighbors (likely in different classes), heterophilic GNNs could aggregate discriminative messages from diverse neighbors. Typically, we decompose the edge-aware weights $\mathbf{a}_{uv}$ into homophily-component a_{uv}^{ho} and heterophily-component a_{uv}^{he} on different neighbor representations $\mathbf{h}_u$, respectively. Hence, we can obtain $\mathbf{a}_{uv} = [a_{uv}^{ho}; a_{uv}^{he}]$. Moreover, due to the complexity and variety of heterophily, there might be no adequate prior knowledge of the extent of heterophily on graphs. Certain homophilic aggregation functions, *e.g.*, GAT [80], also have the ability to impose discriminative weights on different neighbors. Hence, we first introduce ample heterophilic and homophilic aggregation functions for the comprehensiveness of the initial search space design. Adaptively learning a more compact search space is postponed to the latter supernet training stage. Specifically, we introduce 18 homophilic and heterophilic aggregation functions, denoting as the diverse aggregation function set $\mathcal{O}_{AGG}$ shown in Table 5.1. In this way, the heterophilic message-passing scheme can be denoted as:

$$\mathbf{m}_v^{(l)} = \mathcal{O}_{AGG}^{(l)}(\{\mathbf{a}_{uv}^{l-1}\mathbf{h}_u^{(l-1)} : u \in \bar{\mathcal{N}}_k(v)\}),\ k = \{1, 2, ..., K\}. \tag{5.4}$$

In summary, the micro-level design of the proposed heterophilic search space mainly works on diverse message passing with extended local and non-local neighbors, along with separated ego and neighbor node representation learning. And such design encourages heterophilic GNNs to adaptively incorporate the heterophily of graphs in each phase of message passing.

5.3.2.2 Macro-level Design

To integrate local and global information from different layers, combining intermediate layer representations and introducing skip connections have been verified as beneficial layer-wise operations in human-designed GNNs [29, 92, 158] and automated GNNs [41, 43, 159]. In light of this, to enable the flexible GNN architecture design on heterophilic graphs, we introduce the inter-layer connections and combinations into the macro-level search space. In this way, the automated heterophilic GNNs could capture hierarchical information at different layers of network architectures. In detail, our macro-level space contains 5 candidate operations as $\mathcal{O}_{MAC} = \{l\_skip, l\_zero, l\_concat, l\_max, l\_lstm\}$. So that

the final output of a heterophilic GNN can be denoted as:

$$\mathbf{h}_{out} = \mathcal{O}_{MAC}[\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \cdots, \mathbf{h}^{(L)}]. \tag{5.5}$$

With the joint micro-level and macro-level candidate operations, the proposed heterophilic search space significantly enlarges the design scopes and the flexibility in developing GNN architectures on graphs with heterophily.

5.3.3 Progressive Heterophilic Supernet Training

Based on the proposed heterophilic search space, we build a one-shot heterophilic supernet as shown in Fig. 5.1. Given the heterophilic graph with input feature $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_0}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, the proposed Auto-HeG supernet first implements a linear layer, *i.e.*, Multi-layer Perceptron (MLP), to obtain the initial node embedding $\mathbf{H}^{(0)} \in \mathbb{R}^{|\mathcal{V}| \times d_1}$. Then, the node embedding $\mathbf{H}^{(0)}$ and the adjacency matrix $\mathbf{A}$ would be fed into the first layer containing micro-level candidate operations with the $\mathcal{O}_{AGG}$ set, leading to the output node representation $\mathbf{H}^{(1)}$. Next, $\mathbf{H}^{(1)}$ is fed into the macro-level $\mathcal{O}_{MAC}$ for the layer-wise connection. Meanwhile, $\mathbf{H}^{(1)}$ and the two-hop adjacency matrix $\mathbf{A}^2$ would be the inputs of the next micro-level layer. The above process implements repeatedly layer by layer before the final node representations experience the classifier to obtain the predictions of node classes.

Even though the proposed search space contains as many candidate operations as possible, real-world heterophilic graphs usually show significant differences in complexity and diversity, and we still lack prior knowledge on what heterophilic and homophilic aggregations are beneficial for heterophilic graph learning. The proposed initial search space might be comprehensive but severely challenge the effectiveness of supernet training to automatically derive expressive GNN architectures, especially when heterophily varies on different graphs. In light of this, a possible solution comes: why not let the proposed supernet adaptively keep the beneficial operations and drop the irrelevant counterparts as the process of iterative learning goes on? This would contribute to a more compact supernet with relevant candidate operations specifically driven by current heterophilic graphs and tasks, leading to more effective heterophilic GNN architecture development. Moreover, considering some operations will never be selected by certain layers in the final architectures mentioned by [48], this solution ensures different layers flexibly customize their own layer-wise candidate operation sets according to heterophily variation, rather than sharing the entire fixed and large search space with all layers.

Therefore, to derive a more compact heterophilic one-shot supernet for efficient architecture design, we present a progressive training strategy to dynamically shrink the initial search space and adaptively design the candidate operation space in each layer. Specifically, let $\mathcal{S}_0 = (\mathcal{E}_0^{\mathcal{S}}, \mathcal{N}_0^{\mathcal{S}})$ denotes the initial heterophilic supernet constructed based on the proposed entire search space $\mathcal{O}$, and $\mathcal{E}_0^{\mathcal{S}}$ and $\mathcal{N}_0^{\mathcal{S}}$ are the edge set and the node set, respectively. We first train $\mathcal{S}_0(\boldsymbol{\alpha}, \boldsymbol{w})$ several steps for stochastic

differentiable search based on the bi-level optimization:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \mathcal{L}_{val}\left(\boldsymbol{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}\right), \\ s.t. \quad & \boldsymbol{w}^*(\boldsymbol{\alpha}) = \operatorname{argmin}_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}), \end{aligned} \tag{5.6}$$

which conducts iterative learning of the architecture weight $\boldsymbol{\alpha}$ and the model parameter $\boldsymbol{w}$. Let $o \in \mathcal{O}$ denotes a certain operation in heterophilic search space $\mathcal{O}$, and a node pair $(\mathbf{x}_i, \mathbf{x}_j)$ denotes the latent vectors from i-th node to j-th node of the supernet, the learned architecture weights $\boldsymbol{\alpha} = \left\{\alpha_o^{(i,j)} | o \in \mathcal{O}, (i,j) \in \mathcal{N}_0^{\mathcal{S}}\right\}$, and the operation specific weight vector $\alpha_o^{(i,j)}$ can be derived as:

$$\bar{o}^{(i,j)}(\mathbf{x}) = \sum_{o \in \mathcal{O}} \frac{\exp\left[\left(\log \alpha_o^{(i,j)} + u_o^{(i,j)}\right)/\tau\right]}{\sum_{o' \in \mathcal{O}} \exp\left[\left(\log \alpha_{o'}^{(i,j)} + u_{o'}^{(i,j)}\right)/\tau\right]} \cdot o(\mathbf{x}) \tag{5.7}$$

where $\bar{o}^{(i,j)}$ is the mixed operation of edges between (i,j), and $u_o^{(i,j)} = -\log\left(-\log\left(U\right)\right)$ where $U \sim \text{Uniform}(0,1)$. τ is the temperature factor that controls the extent of the continuous relaxation. When τ is closer to 0, the weights would be closer to discrete one-hot values.

To further exploit the relevance and importance of candidate operations to heterophily, we rank the obtained $\boldsymbol{\alpha}$ layer by layer based on their magnitudes and drop the last C irrelevant operations, *i.e.*, cutting C number of edges in each layer at the current supernet training stage. By repeating the above process T iterations, we gradually and dynamically shrink the initial search space and compress the supernet, leading to a compact and effective supernet $\mathcal{S}_c$ by customizing layer-wise heterophilic candidate operation set. The overall process of the proposed progressive supernet training strategy is summarized in Algo. 2.

5.3.4 Heterophily-guided Architecture Selection

With the magnitudes of the architecture weights as the metric, Auto-HeG progressively learns a compact supernet with layer-wise search spaces driven by the heterophily variation. Considering an extreme case: the supernet shrinks gradually till every edge of it keeps only one operation. Basically,

Algorithm 2 Progressive Heterophilic Supernet Training

Require: Initial heterophilic supernet $\mathcal{S}_0$, number of shrinking iterations T, number of candidate operations C to be dropped per iteration t.
Ensure: Compact heterophilic supernet $\mathcal{S}_c$.

```
Let S_c ← S_0;
while t < T do
    Training S_c for several epochs as Eq. (5.6) and (5.7);
    Ranking the magnitudes of the architecture α;
    Dropping C operations from S_c with the smallest C architecture weights;
end while
```

that is the general strategy used by current graph NAS methods, *i.e.*, argmax-based architecture selection scheme. That means only the operations corresponding to the maximum edge weights in the architecture supernet could be preserved by such an architecture selection scheme to build the ultimate GNNs. However, very recent research [51] has verified the architecture magnitude is not effective enough to indicate the operation strength in the final GNN architecture selection. That is why we only use this strategy as the beginning to generally narrow the scope of candidate operations in the proposed search space.

Therefore, to select powerful GNN architectures specifically instructed by heterophily, we derive a heterophily-aware distance as the criterion to guide heterophilic GNN architecture selection. And inspired by the perturbation-based architecture selection scheme in [51], we implement a leave-one-out manner to directly evaluate the contribution of each candidate operation to the compact supernet performance. Specifically, the proposed heterophily-aware distance D_{hete} can be defined with the Euclidean distance as $D_{hete} = ||\hat{\mathcal{H}} - \mathcal{H}||^2$, where $\mathcal{H}$ and $\hat{\mathcal{H}}$ are the heterophilic matrices denoted as:

$$\mathcal{H} = \left(\mathbf{Y}^T\mathbf{A}\mathbf{Y}\right) \oslash \left(\mathbf{Y}^T\mathbf{A}\mathbf{E}\right), \hat{\mathcal{H}} = \left(\hat{\mathbf{Y}}^T\mathbf{A}\hat{\mathbf{Y}}\right) \oslash \left(\hat{\mathbf{Y}}^T\mathbf{A}\mathbf{E}\right), \tag{5.8}$$

where $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}|\times p}$ is the ground-truth label matrix of the heterophilic graph with p classes of nodes, $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times p}$ is an all-ones matrix, and $\oslash$ denotes the Hadamard division operation. Concretely, $\mathcal{H}_{i,j}$ indicates the connection probability of nodes [33, 85] between the i-th class and the j-th class with the ground truth labels, and $\hat{\mathcal{H}}_{i,j}$ works in the same way but with the predicted labels $\hat{\mathbf{Y}} \in \mathbb{R}^{|\mathcal{V}|\times p}$, *i.e.*, the output of the compact heterophilic supernet.

The proposed heterophily-aware distance D_{hete} explicitly restricts the connection possibility of nodes in any two classes predicted by the supernet, to be close to that of the ground truth. And smaller D_{hete} indicates the better discriminative ability of a certain candidate operation to heterophilic node classes. Taking D_{hete} as the guidance, we select the optimal heterophilic GNN architecture from the pretrained compact supernet $\mathcal{S}_c$, and the whole process is illustrated in Algo. 3. In summary, Auto-HeG first builds a comprehensive heterophilic search space, and then progressively shrinks the initial search space layer by layer, leading to a more compact supernet with Algo. 2. Finally, Auto-HeG selects the ultimate heterophilic GNN architecture with the guidance of the heterophily-aware distance in

Algorithm 3 Heterophily Guided Architecture Selection

Require: Pretrained heterophilic supernet $\mathcal{S}_c$ from Algo. 2, edge set $\mathcal{E}_c^{\mathcal{S}}$ and node set $\mathcal{N}_c^{\mathcal{S}}$ of $\mathcal{S}_c$.
Ensure: A heterophilic GNN model with the set of selected operations $\{o_e^*|e \in \mathcal{E}_c^{\mathcal{S}}\}$.

```
while |E_c^S| > 0 do
    randomly remove an edge e_i ∈ E_c^S from E_c^S;
    for all operations o ∈ O on edge e_i do
        calculate D_hete(\o) when o is removed;
    end for
    select the best operation for e_i: o_e^* ← arg min_o D_hete(\o);
end while
```

TABLE 5.2: Performance (ACC%±std) of the proposed Auto-HeG compared with human-designed and graph NAS models on high-heterophily datasets. The best results are in bold and the second-best results are underline. Superscript * represents the officially reported results with the same dataset splits, where Geom-GCN and GCNII do not provide the std; And the remains are our reproduced results if official methods do not test under the same dataset splits.

Methods	Datasets	Cornell	Texas	Wisconsin	Actor
Human-designed models	H2GCN-1*	<u>82.16±4.80</u>	<u>84.86±6.77</u>	<u>86.67±4.69</u>	<u>35.86±1.03</u>
	H2GCN-2*	82.16±6.00	82.16±5.28	85.88±4.22	35.62±1.30
	MixHop*	73.51±6.34	77.84±7.73	75.88±4.90	32.22±2.34
	GPR-GNN	81.89±5.93	83.24±4.95	84.12±3.45	35.27±1.04
	GCNII*	76.49	77.84	81.57	-
	Geom-GCN-I*	56.76	57.58	58.24	29.09
	Geom-GCN-P*	60.81	67.57	64.12	31.63
	Geom-GCN-S*	55.68	59.73	56.67	30.30
	FAGCN	81.35±5.05	84.32±6.02	83.33±2.01	35.74±0.62
Graph NAS models	GraphNAS	58.11±3.87	54.86±6.98	56.67±2.99	25.47±1.32
	SNAG	57.03±3.48	62.70±5.52	62.16±4.63	27.84±1.29
	SANE	56.76±6.51	66.22±10.62	86.67±5.02	33.41±1.41
	SANE-hete	77.84±5.51	77.84±7.81	83.92±4.28	35.88±1.30
	Auto-HeG (ours)	**83.51**±6.56	**86.76**±4.60	**87.84**±3.59	**37.43**±1.37

TABLE 5.3: Performance (ACC%±std) of the proposed Auto-HeG compared with human-designed and graph NAS models on low-heterophily datasets.

Methods	Datasets	Cora	Citeseer	Pubmed
Human-designed models	GCN	85.69±1.80	75.38±1.75	86.08±0.64
	GAT	86.52±1.41	75.51±1.85	84.75±0.51
	GraphSAGE	80.60±3.63	67.18±5.46	81.18±1.12
	SGC	85.88±3.61	73.86±1.73	84.87±2.81
	GCNII*	88.01	77.13	90.30
	Geom-GCN-I*	85.19	77.99	90.05
	Geom-GCN-P*	84.93	75.14	88.09
	Geom-GCN-S*	85.27	74.71	84.75
Graph NAS models	GraphNAS	84.10±0.79	68.83±2.09	82.28±0.64
	SNAG	81.01±1.31	70.14±2.40	83.24±0.84
	SANE	84.25±1.82	74.33±1.54	87.82±0.57
	SANE-hete	85.05±0.90	74.46±1.59	88.99±0.42
	Auto-HeG (ours)	86.88±**1.10**	75.81±**1.52**	89.29±**0.27**

Algo. 3. Consequently, Auto-HeG could automatically derive powerful and expressive GNN models for learning discriminative node representations on heterophilic graphs effectively.

5.4 Experiments

In this section, we first provide the experimental setting details. Then, we compare the proposed Auto-HeG with state-of-the-art human-designed and graph NAS models on the node classification task. Finally, we conduct ablation studies to evaluate the effectiveness of each component in Auto-HeG, including heterophilic search space, progressive heterophilic supernet training, and heterophily-guided architecture selection.

5.4.1 Experimental Setting

Datasets and Baselines. We conduct experiments on seven real-world datasets with different degrees of heterophily (γ_{node}) denoted in brackets, where Cornell (0.11), Texas (0.06), and Wisconsin (0.16) are three WebKB web-page dataset [160], and Actor (0.24) is an actor co-occurrence network [75]. These four datasets are with high heterophily from [31]. Furthermore, Cora (0.83) [161], Citeseer (0.71) [162], and Pubmed (0.79) [163] are three citation network datasets with low heterophily. For comparisons on node classification, we take H2GCN-1 and H2GCN-2 [29], MixHop [20], GPR-GNN [39], GCNII [38], Geom-GCN-I, Geom-GCN-P, and Geom-GCN-S [31], FAGCN [3], as well as classical GCN [1], GAT [80], GraphSAGE [79], and SGC [94] as human-designed model baselines for high-heterophily and low-heterophily datasets, respectively. Furthermore, we take GraphNAS [40], SNAG [157], and SANE [41] as graph NAS model baselines. To test the performance of simply modifying existing graph NAS search spaces, we develop 'SANE-hete' as another baseline by directly injecting heterophilic aggregation functions into the search space of SANE [41].

Implementation. All experiments run with Pytorch platform on Quadro RTX 6000 GPUs and the core code is built based on PyG library [164]. Following the setting in [41], at the search stage, we search with different random seeds and select the best architecture according to the performance on validation data. At the train from scratch stage, we finetune hyper-parameters on validation data within fixed ranges. For all datasets, we use the same splits as that in Geom-GCN [31] and evaluate the performance of all models on the test sets over provided 10 splits for fair comparisons. We report the mean classification accuracy with standard deviations as the performance metric of node classification.

5.4.2 Experimental Results on Node Classification

The node classification results of the proposed Auto-HeG and comparison methods on high-heterophily and low-heterophily datasets are shown in Table 5.2 and Table 5.3, respectively. As shown in Table 5.2, it can be generally observed that the proposed Auto-HeG achieves the best performance when learning on high-heterophily graphs compared with human-designed and graph NAS models. Specifically, Auto-HeG excesses the second-best model, *i.e.*, human-designed H2GCN, with 1.35%, 1.90%, 1.80%, and 1.57% performance improvement on Cornell, Texas, Wisconsin, and Actor datasets, respectively. Compared with graph NAS models, first, Auto-HeG greatly outperforms all existing homophilic graph NAS models over all datasets. We attribute this to the elaborate incorporation of heterophily in our Auto-HeG, especially for graphs with high heterophily. Moreover, the superiority of Auto-HeG to SANE-hete verifies that simply injecting the heterophilic aggregation functions into the existing homophilic search space is not an effective solution. This further illustrates the effectiveness of the proposed progressive heterophilic supernet training and heterophily-guided architecture selection in our Auto-HeG. And incorporating the heterophily into all stages of automatic heterophilic GNN design indeed benefits learning on graphs with heterophily.

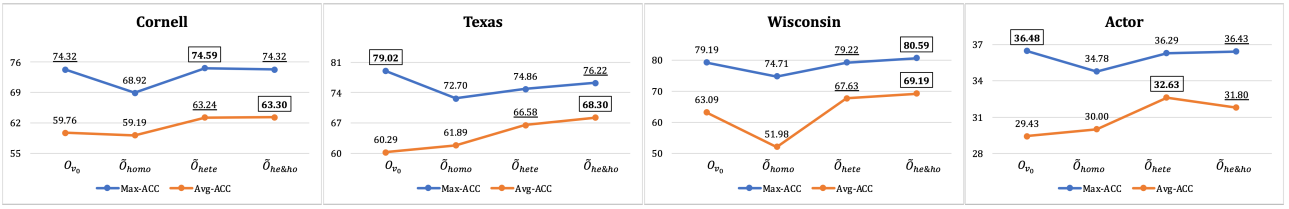

FIGURE 5.2: Illustration of the effectiveness of the proposed search space by evaluating search space variants.

For comparisons on low-heterophily datasets in Table 5.3, our Auto-HeG outperforms most human-designed classical GNN models and graph NAS models, further illustrating its ability to analyze graphs even with low heterophily. We attribute these results to two sub-modules in our Auto-HeG: the proposed search space which keeps many homophilic aggregation functions, and the derived progressive supernet training strategy which customizes layer-wise aggregation functions. These components enable the flexible construction of GNN architectures, leading to superior performance even with low-heterophily. Therefore, we could conclude that even lacking the prior information on heterophily degrees, the proposed Auto-HeG could progressively and adaptively select appropriate candidate operations along with the variation of heterophily, leading to consistently impressive performance on graphs with high-heterophily and low-heterophily. This reflects the excellent ability of our Auto-HeG to deal with complex and various heterophily.

5.4.3 Ablation Study

5.4.3.1 Effectiveness of heterophilic search space.

We derive three variants of the proposed heterophilic search space to verify its effectiveness. Denoting the overall heterophilic search space as $\mathcal{O}_{v_0}$, we consider the following subsets: (1) $\widetilde{\mathcal{O}}_{homo}$: only keep homophilic aggregation functions and remove all heterophilic ones from $\mathcal{O}_{v_0}$, to verify the importance of integrating heterophilic operations; (2) $\widetilde{\mathcal{O}}_{hete}$: only keep heterophilic aggregation functions and remove all homophilic ones from $\mathcal{O}_{v_0}$, to observe the performance when the search space only contains heterophilic operations; (3) $\widetilde{\mathcal{O}}_{he\&ho}$: randomly remove several heterophilic and homophilic aggregation functions $\mathcal{O}_{v_0}$, to illustrate the effectiveness of simultaneously involving heterophilic and homophilic operations.

Specifically, we randomly sample 20 architectures from each subset and report the max performance and average performance among them in Fig. 5.2, respectively. Following observations can be obtained: (1) No best results are achieved in $\widetilde{\mathcal{O}}_{homo}$ on all datasets in both max and average cases, which naturally verifies the importance of heterophilic operations to heterophilic graph learning; (2) $\widetilde{\mathcal{O}}_{hete}$ and $\widetilde{\mathcal{O}}_{he\&ho}$ generally have better performance compared to other search space sets, which illustrates the necessity of involving appropriate homophilic operations. Moreover, this shows that merely involving heterophilic operations is not enough for complex and diverse heterophily learning;

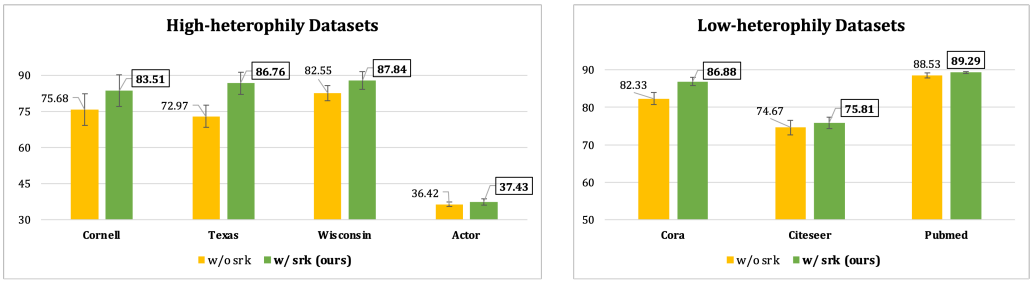

FIGURE 5.3: Performance w/ and w/o shrinking (srk) for the proposed progressive supernet training.

TABLE 5.4: Comparison of the proposed heterophily-guided architecture selection scheme (Heter. Arch. Select.) with other architecture selection methods.

Arch. Select. Methods	*High-heterophily Datasets*				*Low-heterophily Datasets*		
	Cornell	Texas	Wisconsin	Actor	Cora	Citeseer	Pubmed
Argmax Arch. Select.	79.19±8.38	79.46±3.67	86.27±4.02	37.12±1.12	85.33±1.46	75.43±2.24	88.52±0.41
Val. Loss Arch. Select.	57.84±3.67	71.35±5.30	80.59±7.09	36.96±1.08	84.67±1.64	73.86±1.11	88.23±0.53
Heter. Arch. Select. (ours)	**83.51**±6.56	**86.76**±4.60	**87.84**±3.59	**37.43**±1.37	**86.88**±1.10	**75.81**±1.52	**89.29**±0.27

(3) The overall search space $\mathcal{O}_{v_0}$ performs the best only on Texas and Actor datasets in the max case. This presents that $\mathcal{O}_{v_0}$ could not achieve consistently best performance under the random sampling scenario, even with the largest number of candidate operations. Moreover, this result also verifies that developing a more compact heterophilic search space is necessary and crucial. At this point, it could be concluded that the proposed progressive supernet training strategy is effective, and we could attribute this to its adaptive and dynamic layer-wise search space shrinking schema.

5.4.3.2 Effectiveness of progressive supernet training.

The comparison results with and without the layer-wise search space shrinking in the proposed progressive supernet training strategy are listed in Fig. 5.3. It can be generally observed that, on both high-heterophily and low-heterophily datasets, the performance of the compact supernet consistently achieves better performance with the proposed supernet training strategy. For example, the compact supernet with progressive training significantly raises the performance of Cornell from 75.68% to 83.51% and Texas from 72.97% to 86.76%. We attribute such impressive improvements to the excellent ability of our Auto-HeG in building the effective and adaptive supernet. Without the progressive training, the initial search space would keep a large number of operations that might be irrelevant and redundant to specific graphs with heterophily, which brings serious challenges to the search strategy for optimizing in such a large supernet. At this point, Auto-HeG progressively shrinks the search space layer-wisely and dynamically narrows the scope of relevant candidate operations, resulting in a more compact and effective supernet for deriving powerful heterophilic GNNs.

5.4.3.3 Effectiveness of heterophily-guided architecture selection.

We compare the proposed heterophily-guided architecture selection with the other two types of architecture selection methods, *i.e.*, architecture weight magnitude based argmax strategy, which is the most commonly used method in existing gradient-based graph NAS [41, 43], and perturbation-based architecture selection method with the validation loss criterion proposed by [51]. The comparison results are listed in Table 5.4. Generally, our proposed heterophily-guided scheme with the heterophily-aware distance criterion gains the best classification performance on all datasets consistently, illustrating its effectiveness in expressive architecture selection. Furthermore, we can observe that the architecture weight magnitude based argmax strategy, *i.e.*, 'Argmax Arch. Select.', performs better than that of perturbation-based architecture selection method with the validation loss criterion, *i.e.*, 'Val. Loss Arch. Select.'. Even the work [51] has verified the effectiveness of 'Val. Loss Arch. Select.' over CNN-based NAS in the leave-one-out manner, more importantly, we find that it might not well adapt to GNN-based NAS by implementing the validation loss based criterion straightforwardly. This fact further illustrates the effectiveness of the proposed heterophily-aware distance criterion in the leave-one-out-manner for GNN-based NAS with heterophily.

5.5 Conclusion

In this chapter, we propose a novel automated graph neural network on heterophilic graphs via heterophily-aware graph neural architecture search, namely Auto-HeG, which is the first work of automatic heterophilic graph learning. By explicitly incorporating heterophily into all stages, *i.e.*, search space design, supernet training, and architecture selection, Auto-HeG could develop powerful heterophilic GNNs to deal with the complexity and variety of heterophily effectively. To build a comprehensive heterophilic GNN search space, Auto-HeG includes non-local neighbor extension, ego-neighbor separation, diverse message passing, and layer-wise combination at both micro-level and macro-level. To develop a compact and effective heterophilic supernet based on the initial search space, Auto-HeG conducts the progressive supernet training strategy to dynamically shrink the scope of candidate operations according to layer-wise heterophily variation. In the end, taking the heterophily-aware distance criterion as the guidance, Auto-HeG selects excellent heterophilic GNN architectures by directly evaluating the contribution of each operation in the leave-one-out pattern. Extensive experiments verify the superiority of the proposed Auto-HeG on learning graphs with heterophily to both human-designed models and graph NAS models.

Chapter 6

Automated Graph Neural Networks on Multi-relational Graphs

6.1 Introduction

Graphs are pervasive structured data and have been widely used in many real-world scenarios, such as social networks [7], knowledge bases [9], and recommendation systems [14]. Recently, graph neural networks (GNNs) have become prevalent models with excellent learning abilities for analyzing various graph-structure data [2, 5, 16, 17, 24, 81, 83, 106, 165, 166]. Despite the remarkable success, existing GNN models are usually designed manually by experts for various network architectures on different graphs. Under the variety and complexity of tasks and structures on graphs, such manual designs would cost much human effort with heavy reliance on expert knowledge. Moreover, the design scopes of model architectures would be limited by human understanding, resulting in limited performance on graph learning. Hence, it is necessary to build tailored GNN architectures driven by specific graph tasks and data to relieve human effort.

In light of this, a line of automated machine learning research, *i.e.*, neural architecture search (NAS) [42, 44, 45, 48, 113, 114], is introduced into GNN development for automatically discovering and creating excellent network architectures. With well-designed search spaces and well-customized search strategies, graph NAS methods have achieved promising progress in automated graph learning [40, 41, 43, 153–155], yielding superior GNN architectures for specific tasks and data than handcrafted models. However, there are two critical challenges remaining with existing graph NAS methods, from the perspectives of graph data modelling and search space design, respectively.

Concretely, current automated GNNs mainly serve on the single-relational graph setting [40, 41], where there exists at most one edge between arbitrary nodes and each edge indicates a binary relation, *i.e.*, connected and unconnected, as shown in Fig. 6.1(a). Such a setting significantly limits the applications

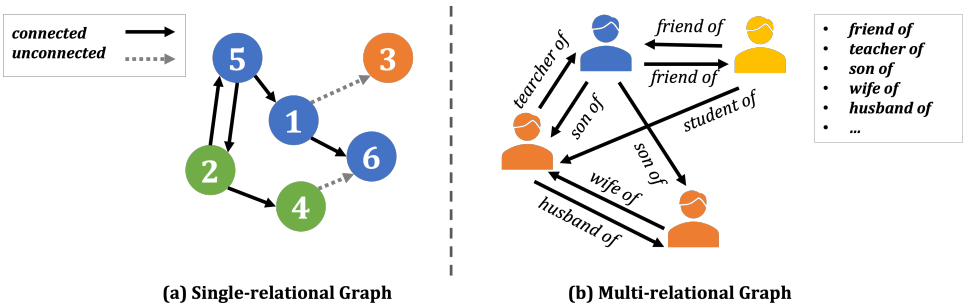

FIGURE 6.1: Difference between single-relational graphs and multi-relational graphs.

of existing automated GNNs, while real-world graphs generally contain multiple relation types, *e.g.*, knowledge graphs [167], the typical multi-relational graphs with different relation types and edge directions, as shown in Fig. 6.1(b). Furthermore, existing graph NAS models merely focus on the node representation learning, while the relation representation learning is not well explored. On multi-relational graphs, node and relation representations are required to be learned jointly. From the perspective of search space design, current search spaces of automated GNNs generally integrate typical GNN layers and hyper-parameters as candidate operations. To some extent, such a coarse-grained search space design could be regarded as a modular ensemble of existing GNN layers, and the principle GNN architectures are not changed and innovated. This would significantly limit the capacity and scope of new GNN architecture development, leading to the damaged flexibility and reasoning ability of automated GNNs.

To tackle the two challenges mentioned above, we propose a novel framework of multi-relational graph neural architecture search, named MR-GNAS, to automatically develop powerful multi-relational GNN models with outstanding abilities in analyzing multi-relational graphs. Specifically, to enlarge the search capacities and improve the search flexibility, we propose a fine-grained GNN search space with functional candidate operations to integrate the critical components of the multi-relational message passing schema. In this way, the proposed MR-GNAS could build innovative multi-relational GNN architectures with excellent learning abilities. To jointly learn with the diverse entity and relation types, MR-GNAS constructs a relation-aware supernet with a tree topology, in which the informative messages would be effectively learned, resulting in discriminative node and relation representations for better analyzing multi-relational graphs.

Concretely, based on the fine-grained search space, the relation-aware supernet is composed of four sequentially stacked cells, embracing the overall pipeline of multi-relational message passing. Given a multi-relational graph, it first inputs (1) an entity-relation integration cell, so that the semantics and contexts of nodes and edges would be well captured and integrated according to their interactions, leading to informative messages of diverse entity and relation types. Then, the obtained informative messages flow to (2) a relation-aware message filtering cell to further learn effective messages with sparse and dense gating mechanisms, so that the most relevant information of relation-aware message passing could be preserved. Further, the filtered messages would be aggregated with (3) a neighbor

aggregation cell according to the structure information on multi-relational graphs, leading to informative node representations. Finally, (4) an entity-aware embedding filtering cell outputs the ultimate node representations with entity-level sparse and dense feature selection operations. For relation representation learning, it is not only incorporated into the first entity-relation integration cell, but also implements a linear combination and mapping for yielding the ultimate relation embeddings.

Benefiting from the relation-aware supernet with full-pipe multi-relational message passing, the proposed MR-GNAS could incorporate different relation types and edge directions with joint node and relation representation learning. And the functional candidate operations in the fine-grained search space further encourage adequate exploration of multi-relational graph structures and contexts. Through a gradient-based search strategy, MR-GNAS can automatically derive expressive multi-relational GNN models driven by various multi-relational graph data and tasks, leading to wide architecture design scopes, flexible model architectures, and outstanding multi-relational graph analysis abilities.

In summary, the contributions of this chapter are as follows:

- To overcome the limitation of single-relational setting in graph NAS, we propose a novel framework of multi-relational graph neural architecture search, dubbed MR-GNAS, enabling innovative multi-relational GNN architecture design with excellent learning abilities.
- To enlarge the search capacities and improve the search flexibility, we design a fine-grained search space with functional candidate operations to embrace the multi-relational message passing schema, leading to outstanding architecture search abilities.
- To jointly learn with diverse entity and relation types, we construct a relation-aware supernet with four sequentially stacked cells in the tree topology, resulting in discriminative node and relation representations for benefiting multi-relational graph analysis.
- Through a gradient-based search strategy, the proposed MR-GNAS could derive expressive multi-relational GNN architectures and extensive experiments on entity classification and link prediction tasks illustrate its superiority.

6.2 Preliminary

General Message Passing Schema. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, unweighted graph where $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ is the node set and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the edge set. The neighbor set of node v is $\mathcal{N}(v) = \{u : (v, u) \in \mathcal{E}\}$, and initial node features are represented by $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_0}$ with d_0-dimension features. For the uniform message passing scheme of general GNNs, node representations are learned

by first aggregating the messages from local neighbors, and then combining the aggregated messages with ego-node representations [27] for update. This process is denoted as:

$$\begin{aligned} \mathbf{m}_v^{(l)} &= \mathrm{AGG}^{(l)}(\{\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}(v)\}), \\ \mathbf{h}_v^{(l)} &= \mathrm{UPDATE}^{(l)}(\mathbf{h}_v^{(l-1)}, \mathbf{m}_v^{(l)}), \end{aligned} \tag{6.1}$$

where $\mathbf{m}_v^{(l)}$ and $\mathbf{h}_v^{(l)}$ are the message vector and the representation vector of node v at the l-th layer, respectively. $\mathrm{AGG}(\cdot)$ and $\mathrm{UPDATE}(\cdot)$ are the aggregation function and the update function, respectively.

Based on the general message passing schema, existing automated GNNs on single-relational graphs usually design their search spaces on $\mathrm{AGG}(\cdot)$ and $\mathrm{UPDATE}(\cdot)$ as $\mathcal{O}_{AGG}$ and $\mathcal{O}_{UPD}$, respectively. Concretely, $\mathcal{O}_{AGG}$ usually contains typical GNN layers as candidate operations, *i.e.*, GCN [1], GAT [80], and GraphSAGE [7]. And $\mathcal{O}_{UPD}$ generally contains combination and update functions to integrate neighbor information and central node information, *i.e.*, Multi-Layer Perceptron (MLP) and concatenation operations [41, 43]. Moreover, existing search spaces might contain intra-layer operations, *i.e.*, skip connection, to benefit multi-layer message passing. Furthermore, some of current automated GNNs also involve certain hyper-parameters in their search spaces, *i.e.*, activation functions and the number of multi-head attentions [40, 156].

Multi-relational Message Passing Schema. Given a multi-relational graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ with the set of nodes $\mathcal{V}$, relations $\mathcal{R}$, and edges $\mathcal{E}$, each edge (u, r, v) represents that the relation r exists from source node u to target node v for $\forall u, v \in \mathcal{V}$ and $r \in \mathcal{R}$. Typically, RGCN [2] first proposed multi-relational message passing schema by incorporating edge-specific weights into $AGG(\cdot)$ as:

$$\mathbf{m}_v^{(l)} = \mathrm{AGG}^{(l)}(\{\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l-1)} : u \in \mathcal{N}_r(v)\}). \tag{6.2}$$

To further address relation representation learning, CompGCN [17] integrated the relation embedding $\mathbf{z}_r \in \mathbb{R}^d$ into the multi-relational message passing with $\phi(\cdot)$ operator for combining the entity-relation representations as:

$$\mathbf{m}_v^{(l)} = \mathrm{AGG}^{(l)}(\{\mathbf{W}_{\lambda(r)}^{(l)}\phi(\mathbf{h}_u^{(l-1)}, \mathbf{z}_r^{(l-1)}) : u \in \mathcal{N}_{\lambda(r)}(v)\}), \tag{6.3}$$

Note that $\mathcal{N}_{\lambda(r)}(v)$ indicates the neighbor set with different relation types and edge directions as $\lambda(r) = \{r \in \mathcal{R} \cup r^{-1} \in \mathcal{R}^{-1} \cup r_l = \top\}$, where $r_l = \top$ and $r^{-1} \in \mathcal{R}^{-1}$ denote the self-loop connection with $(u, \top, u)$ and the inverse relation with (v, r^{-1}, u), respectively. To enlarge the search capacities for flexible automated GNN construction, in this chapter, we build upon the multi-relational message passing schema by designing a fine-grained search space, with $\phi(\cdot)$ operator, $\mathrm{AGG}(\mathbf{W}_{\lambda(r)})$, and $\mathrm{UPDATE}^{(l)}(\cdot)$ in different component cells, composing of a relation-aware supernet with a tree topology.

6.3 The Proposed Method

Through the lens of multi-relational message passing schema, we propose a novel framework of multi-relational graph neural architecture search, named MR-GNAS. Specifically, MR-GNAS contains a fine-grained search space with functional candidate operations. Based on the well-designed search space, MR-GNAS constructs a relation-aware supernet with four sequentially stacked cells in the tree topology, and the overall pipeline is illustrated in Fig. 6.2. Importantly, such multi-relational message passing schema ensures the proposed MR-GNAS to better incorporate multi-relational graph structures into the learning process of tree-topology supernet. Based on this, the delicately tailored search space with adequate fine-grained candidates make MR-GNAS enjoy the advantages of different functional operations, and then, the expressive architectures would be selected to automatically build powerful and innovative MR-GNN models.

Specifically, given a multi-relational graph, MR-GNAS first takes the d-dimensional entity embeddings $\mathbf{h}_u, \mathbf{h}_v \in \mathbb{R}^d$ and relation embedding $\mathbf{z}_r \in \mathbb{R}^d$ as the initial inputs. Then, the entity and relation inputs would experience four-component cells sequentially : C_1: entity-relation integration cell, C_2: relation-aware message filtering cell, C_3: neighbor aggregation, and C_4: entity-aware embedding filtering cell.

- C_1: Entity-relation Integration Cell: capturing and integrating the semantics and contexts of nodes and edges according to their interactions, leading to informative messages of diverse entity and relation types.
- C_2: Relation-aware Message Filtering Cell: learning effective messages with sparse and dense gating mechanisms, preserving the most relevant information of relation-aware message passing.
- C_3: Neighbor Aggregation Cell: merging filtered messages according to multi-relational graph structures, leading to informative node representations.
- C_4: Entity-aware Embedding Filtering Cell: conducting entity-level sparse and dense feature selections for obtaining the ultimate node representations.

In the meantime, the relation embeddings are further updated with linear combination and mapping to generate representative relation features. Hence, the proposed MR-GNAS encases an overall pipe of multi-relational message passing with fine-grained functional operations, enabling it to jointly learn with diverse entity and relation types. In this way, discriminative node and relation representations could be generated for benefiting multi-relational graph analysis. More details of the proposed search space for each cell are as follows.

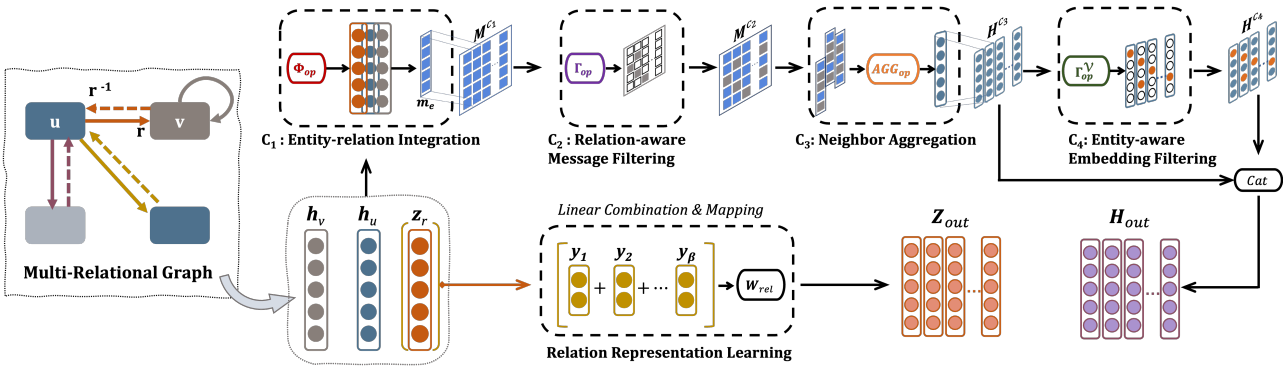

FIGURE 6.2: The overall framework of multi-relational graph neural architecture search with fine-grained message passing.

6.3.1 Search Space

C_1: Entity-relation Integration Cell. To capture the interactions between entity nodes and relation-type specific edges, we introduce three types of entity-relation integration operations to capture and compose the semantics and contexts of entities and relations, leading to informative and beneficial messages.

Specifically, we extend the $\phi(\cdot)$ operator in Eq. (6.3) to the set of integration operator with following candidate operations: (1) $\phi_+(\mathbf{h}_u, \mathbf{z}_r) = \mathbf{h}_u + \mathbf{z}_r$; (2) $\phi_-(\mathbf{h}_u, \mathbf{z}_r) = \mathbf{h}_u - \mathbf{z}_r$; (3) $\phi_*(\mathbf{h}_u, \mathbf{z}_r) = \mathbf{h}_u * \mathbf{z}_r$. Then, we can obtain the integrated entity-relation message $\mathbf{m}_e^{c_1} \in \mathbb{R}^d$ for each edge $e \in \mathcal{E}$ as:

$$\mathbf{m}_e^{c_1} = \Phi_{op}(\mathbf{h}_u, \mathbf{z}_r), \tag{6.4}$$

where $\Phi_{op} = \{\phi_+, \phi_-, \phi_*\}$ is the candidate set of the composition operators.

C_2: Relation-aware Message Filtering Cell. The key challenge of message passing on multi-relational graphs is to preserve the most relevant messages and filter the redundant information for specific tasks. In light of this, we develop a set of message gating mechanisms inspired by [168] in the message filtering cell. Different from [168] merely considering the node representation filtering with node-level messages, we introduce relation-aware message filtering, which incorporates diverse relation types and edge directions. Hence, the proposed MR-GNAS could learn informative entity and relation interaction messages, leading to better adaption and learning ability for multi-relational graph analysis.

Taking the relation-aware message matrix $\mathbf{M}^{c_1} = [\mathbf{m}_1^{c_1}, \cdots, \mathbf{m}_e^{c_1}] \in \mathbb{R}^{|\mathcal{E}| \times d}, e \in \mathcal{E}$ from C_1 as the input, C_2 considers three types of filters with five candidate operations, by extending $AGG(\mathbf{W}_{\lambda(r)})$ in Eq. (6.3) to Γ_{op} searching set, encouraging effective multi-relational message passing via

$$\mathbf{M}^{c_2} = \Gamma_{op}(\mathbf{M}^{c_1}), \tag{6.5}$$

TABLE 6.1: Summary of different types of relation-aware message filters. '$\cdot$' denotes the matrix product and '$\odot$' is the Hardmard product.

Types	Candidates	Mappings	Weights	Operators
Vanilla	$\mathcal{F}_\lambda$	-	$\mathbf{W}_\lambda \in \mathbb{R}^{3\times d\times d'}$	$\cdot$
Sparse	$\mathcal{F}^s$	φ^s	$\mathbf{W}^s \in \mathbb{R}^{\lvert\mathcal{E}\rvert\times\lvert\mathcal{E}\rvert}$	$\cdot$
	$\mathcal{F}^s_\lambda$	$\varphi^s_\lambda = [\varphi^s_r; \varphi^s_{r^{-1}}; \varphi^s_\top]$	$\mathbf{W}^s_\lambda \in \mathbb{R}^{\lvert\mathcal{E}\rvert\times\lvert\mathcal{E}\rvert}$	$\cdot$
Dense	$\mathcal{F}^d$	φ^d	$\mathbf{W}^d \in \mathbb{R}^{\lvert\mathcal{E}\rvert\times d'}$	$\odot$
	$\mathcal{F}^d_\lambda$	$\varphi^d_\lambda = [\varphi^d_r; \varphi^d_{r^{-1}}; \varphi^d_\top]$	$\mathbf{W}^d_\lambda \in \mathbb{R}^{\lvert\mathcal{E}\rvert\times d'}$	$\odot$

where $\Gamma_{op} = \{\mathcal{F}_\lambda, \mathcal{F}^s, \mathcal{F}^s_\lambda, \mathcal{F}^d, \mathcal{F}^d_\lambda\}$, *i.e.*, vanilla filter $\mathcal{F}_\lambda$, sparse filters $\mathcal{F}^s$ and $\mathcal{F}^s_\lambda$, as well as dense filters $\mathcal{F}^d$ and $\mathcal{F}^d_\lambda$. Note that $\mathcal{F}_\lambda$ could be taken as the basic $AGG(\mathbf{W}_{\lambda(r)})$ in Eq. (6.3). Moreover, we include the identity operation $\mathcal{I}(\mathbf{M}^{c_1}) = \mathbf{M}^{c_1}$ in this cell, which can be taken as the skip-connection to preserve the original beneficial messages. Different types of message filters learn and select specific messages at different scales of relation types and edge directions, leading to the full exploration of data-driven and task-driven messages of entity-relation interactions.

We summarize the main differences of different relation-aware message filters in Γ_{op} in Table 6.1, from the perspectives of three essential components, *i.e.*, mapping functions, weight matrices, and operators. For convenience, we simplify the relation-type set $\lambda(r)$ as λ and more details of each filter can be found as follows.

(1) Vanilla Filter $\mathcal{F}_\lambda$ typically weights the beneficial messages according to the directions of edges and the types of relations. In detail, it learns the re-scaling weight parameter matrix $\mathbf{W}_\lambda = [\mathbf{W}_r; \mathbf{W}_{r^{-1}}; \mathbf{W}_\top] \in \mathbb{R}^{3\times d\times d'}$, where each of $\mathbf{W}_r$, $\mathbf{W}_{r^{-1}}$, and $\mathbf{W}_\top$ learns a $\mathbb{R}^d \to \mathbb{R}^{d'}$ mapping with the original relations r, inverse relations r^{-1}, and self-loop relations $\top$, respectively. Hence, the vanilla filter is calculated by

$$\begin{aligned}\mathcal{F}_\lambda(\mathbf{M}^{c_1}) &= \mathbf{W}_{\lambda(r)} \cdot \mathbf{M}^{c_1} \\ &= [\mathbf{M}^{c_1}_r \mathbf{W}_r;\ \mathbf{M}^{c_1}_{r^{-1}} \mathbf{W}_{r^{-1}};\ \mathbf{M}^{c_1}_\top \mathbf{W}_\top],\end{aligned} \tag{6.6}$$

where $\mathbf{M}^{c_1}_\lambda$ denotes corresponding messages with specific relation types, *e.g.*, $\mathbf{M}^{c_1}_r \in \mathbb{R}^{\lvert\mathcal{E}_r\rvert\times d}$, and $\lvert\mathcal{E}_r\rvert$ denotes the number of edges with the original relations r in $\mathcal{G}$.

(2) Sparse Filters $\mathcal{F}^s$ and $\mathcal{F}^s_\lambda$ select and re-scale the beneficial messages through the sparse gating mechanism by fully understanding the importance of relation-aware information. Specifically, this mechanism is implemented by the sparse mapping function $\varphi^s(\cdot)$ and its relation-type specific variant $\varphi^s_\lambda(\cdot)$. We first calculate the weight matrix $\mathbf{W}^s \in \mathbb{R}^{\lvert\mathcal{E}\rvert\times\lvert\mathcal{E}\rvert}$ with the sparse mapping function $\varphi^s : \mathbb{R}^d \to \mathbb{R}^1$, then the messages can be re-scaled as:

$$\mathbf{W}^s = \mathrm{diag}\left[\varphi^s(\mathbf{M}^{c_1})\right], \quad \mathcal{F}^s(\mathbf{M}^{c_1}) = \mathbf{W}^s \cdot \mathbf{M}^{c_1}, \tag{6.7}$$

TABLE 6.2: Architecture comparison of the proposed MR-GNAS *vs.* existing multi-relational GNNs.

Models	C_1 Entity-relation Integration	C_2 Relation-aware Message Filtering	C_3 Neighbor Aggregation	C_4 Entity-aware Embedding Filtering	#Leaf Nodes of DAG
RGCN [2]	$\times$	$\approx \mathcal{F}_\lambda$	AGG_{mean}	$\times$	$[0, 1, 1, 0]$
SACN [28]	$\times$	$\approx \mathcal{F}_\lambda$	AGG_{sum}	$\times$	$[0, 1, 1, 0]$
CompGCN [17]	$\approx \Phi_{op}$	$\mathcal{F}_\lambda$	AGG_{sum}	$\times$	$[1, 1, 1, 0]$
MR-GNAS (ours)	$\Phi_{op} = \{\phi_+, \phi_-, \phi_*\}$	$\Gamma_{op} = \{\mathcal{F}_\lambda, \mathcal{F}^s, \mathcal{F}^s_\lambda, \mathcal{F}^d, \mathcal{F}^d_\lambda\}$	$AGG_{op} = \{sum, mean, max\}$	$\Gamma^{\mathcal{V}}_{op} = \{\mathcal{F}^s, \mathcal{F}^d, \mathcal{I}\}$	$[N_{c_1}, N_{c_2}, N_{c_3}, N_{c_4}]$

where diag$[\cdot]$ is adopted for converting the vector to diagonal matrix, and $\varphi^s(\cdot)$ is implemented by a two-layer MLP with sigmoid activation function.

Furthermore, as shown in Table 6.1, $\varphi^s_\lambda(\cdot)$ extends the above mapping with different relation types and edge directions by assembling them together as $\varphi^s_\lambda = [\varphi^s_r; \varphi^s_{r^{-1}}; \varphi^s_\top]$. Hence, the relation-specific sparse filter $\mathcal{F}^s_\lambda$ is natural to be derived following the same logic in Eq. (6.6) and Eq. (6.7) as $\mathcal{F}^s_\lambda(\mathbf{M}^{c_1}) = \mathbf{W}^s_\lambda \mathbf{M}^{c_1}$, where

$$\begin{aligned} \mathbf{W}^s_\lambda &= \text{diag}\left[\varphi^s_\lambda(\mathbf{M}^{c_1})\right] \\ &= \text{diag}\left[\varphi^s_r(\mathbf{M}^{c_1}_r); \varphi^s_{r^{-1}}(\mathbf{M}^{c_1}_{r^{-1}}); \varphi^s_\top(\mathbf{M}^{c_1}_\top)\right]. \end{aligned} \tag{6.8}$$

(3) Dense Filters $\mathcal{F}^d$ and $\mathcal{F}^d_\lambda$ consider a more fine-grained message gating scheme to control information flows and exploit the importance of relation-aware messages in a dense manner. Specifically, the dense mapping function $\varphi^d(\cdot)$ and its relation-type specific variant $\varphi^d_\lambda(\cdot)$ learn the $\mathbb{R}^d \to \mathbb{R}^{d'}$ mapping, yielding dense message re-scaling weights $\mathbf{W}^d$ and $\mathbf{W}^d_\lambda$ with the dimension of $\mathbb{R}^{|\mathcal{E}| \times d'}$. Then, we have:

$$\mathbf{W}^d = \varphi^d\left(\mathbf{M}^{c_1}\right), \quad \mathcal{F}^d(\mathbf{M}^{c_1}) = \mathbf{W}^d \odot \mathbf{M}^{c_1}, \tag{6.9}$$

where $\odot$ denotes the Hardmard product and $\varphi^d(\cdot)$ is a single-layer MLP with sigmoid activation function. Similarly, the relation-type specific dense filter can be obtained by $\mathcal{F}^d_\lambda(\mathbf{M}^{c_1}) = \mathbf{W}^d_\lambda \odot \mathbf{M}^{c_1}$, where

$$\mathbf{W}^d_\lambda = \varphi^d_\lambda\left(\mathbf{M}^{c_1}\right) = \left[\varphi^d_r(\mathbf{M}^{c_1}_r); \varphi^d_{r^{-1}}(\mathbf{M}^{c_1}_{r^{-1}}); \varphi^d_\top(\mathbf{M}^{c_1}_\top)\right]. \tag{6.10}$$

In summary, sparse filters calculate the weights among edges with different relation types, while dense filters go for weighting each attribute of each edge. The ensemble of dense and sparse filters would ensure comprehensive message learning at different weighting scales, leading to effective information passing.

C_3: Neighbor Aggregation Cell. Based on the multi-relational graph structures, we consider three types of aggregation functions to generate node embeddings via merging filtered relation-aware messages $\mathbf{M}^{c_2}$ from their neighbors, *i.e.*, $AGG_{op} = \{sum, mean, max\}$. As demonstrated by [27], AGG_{sum} works well for capturing comprehensive structure information, AGG_{mean} considers the statistics of input messages in the aggregating process, while AGG_{max} is robust to noise and performs better on identifying the typical and critical information. Hence, neighbor aggregation cell C_3 would yield node

embeddings $\mathbf{H}^{c_3} \in \mathbb{R}^{|\mathcal{V}| \times d'}$ as:

$$\mathbf{H}^{c_3} = AGG_{op}(\mathbf{M}^{c_2}). \tag{6.11}$$

With the ensemble of these complementary aggregation operations, the proposed MR-GNAS could benefit from the advantages of each operation so that the searched models would be more expressive compared with manually designed multi-relational GNN architectures.

C_4: Entity-aware Embedding Filtering Cell. We impose sparse and dense filters $\mathcal{F}^s$ and $\mathcal{F}^d$ on node embeddings to further capture discriminative entity representations, corresponding to coarse-grained and fine-grained feature selections, respectively. Note that different from C_2 that considers the relation-aware message gating, this cell C_4 conducts the entity-aware embedding gating. That means the former focuses on the message level with diverse relation types and edge directions, and the latter concerns the embedding level with various entity types. Thus, the candidate operation set of the entity-aware embedding filtering cell is $\Gamma^{\mathcal{V}}_{op} = \{\mathcal{F}^s, \mathcal{F}^d, \mathcal{I}\}$ and the filtered node features can be obtained by:

$$\mathbf{H}^{c_4} = \Gamma^{\mathcal{V}}_{op}(\mathbf{H}^{c_3}), \tag{6.12}$$

where the computations of sparse and dense weights in $\mathcal{F}^s$ and $\mathcal{F}^d$ following the same paradigm in Eq. (6.7) and Eq. (6.9) but working on the node embeddings. We introduce the superscript $^{\mathcal{V}}$ to make a clear distinction. At last, we concatenate the outputs of C_3 and C_4 and impose an MLP for obtaining the ultimate node representations:

$$\mathbf{H}_{out} = MLP([\mathbf{H}^{c_3}; \mathbf{H}^{c_4}]). \tag{6.13}$$

Relation Representation Learning. Considering the scalability with large-scale relation types and edges, we adopt the linear combination of a set of basis relation vectors following [17]. Then, the relation embeddings can be updated with a single fully-connected mapping with parameters $\mathbf{W}_{rel}$ as follows:

$$\mathbf{z}_r = \sum_{b=1}^{\mathcal{B}} \beta_r \mathbf{y}_b, \quad \mathbf{Z}_{out} = \mathbf{W}_{rel}[\mathbf{z}_r]_{r=1}^{\mathcal{R}}, \tag{6.14}$$

where $\mathcal{B}$ denotes the number of bases, $\{\mathbf{y}_b\}_{b=1}^{\mathcal{B}}$ is the set of learnable basis vector and $\beta_r \in \mathbb{R}$ is the learnable relation specific weight scalar of a certain basis. Moreover, the relation embeddings are also incorporated into the first entity-relation integration cell, leading to the joint learning of discriminative node and relation representations.

6.3.2 Relation-aware Supernet of MR-GNAS

Based on the proposed framework that embraces the overall pipe of multi-relational message passing schema in Fig. 6.2, we further build a multi-relational GNN supernet within a tree topology by

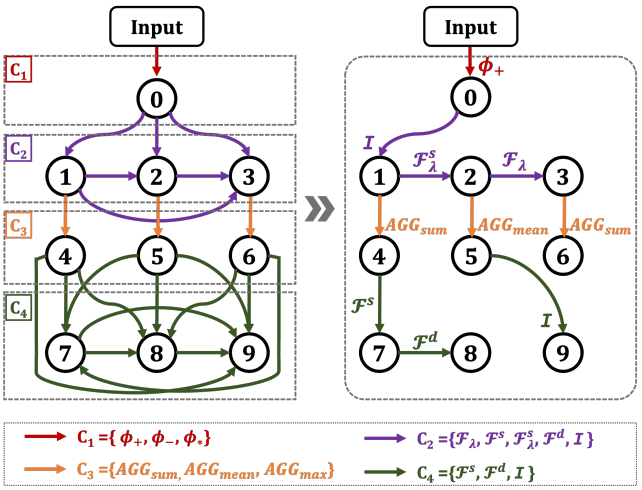

FIGURE 6.3: Supernet of MR-GNAS (left). The right shows an example of model architecture after searching.

injecting different numbers of nodes into each cell, leading to a relation-aware Directed Acyclic Graph (DAG) shown in the left of Fig. 6.3.

Specifically, in the supernet, we keep the sequential connection of four cells in the search space and expand the single-stream message passing to N leaf nodes with multiple information streams for each cell in DAG, leading to more comprehensive and adequate informative message capture and flow. Note that different from the entity nodes on multi-relational graphs, each leaf node in the supernet $x^{(i)}$ is the latent representation of the output of each cell. For example, in the first cell of entity-relation integration, $x_{c_1}^{(i)}$ denote the message matrix $\mathbf{M}^{c_1}$; While in the cell of entity-aware embedding filtering, $x_{c_4}^{(i)}$ denote the entity representation matrix $\mathbf{M}^{c_4}$ of all nodes on multi-relational graphs. In the left of Fig. 6.3, we take $N_{c_1}\{x^{(0)}\} = 1, N_{c_2}\{x^{(1,2,3)}\} = N_{c_3}\{x^{(4,5,6)}\} = N_{c_4}\{x^{(7,8,9)}\} = 3$ for illustration convenience. With the ensemble of complementary N functional candidates in each cell, the proposed relation-aware supernet of MR-GNAS could benefit from the advantages of different operations. This further enables MR-GNAS to incorporate multiple information streams in each cell and further evaluate the strengths of different candidate operations. At the end of the search stage, MR-GNAS selects appropriate candidates cell-by-cell and builds the ultimate multi-relational GNN architectures driven by specific data and tasks, as shown in the right of Fig. 6.3. In this way, the searched MR-GNNs would be more expressive and powerful due to the fine-grained candidate selection inside each cell.

To illustrate the effectiveness well-designed search space and the sufficiency of the proposed supernet, we make a detailed comparison between our MR-GNAS supernet and state-of-the-art multi-relational GNNs in Table 6.2. As can be generally observed, existing prevalent human-designed GNN models on multi-relational graphs, *i.e.*, RGCN [2], SACN [28], CompGCN [17], can be approximated as the sub-architectures in the proposed supernet of MR-GNAS. In each cell, our MR-GNAS contains more

adequate functional candidate operations to enlarge the search scope of architecture design, along with more flexible leaf node number settings for developing creative and expressive MR-GNN models.

6.3.3 Search Algorithm

Based on the proposed relation-aware supernet of MR-GNAS, we introduce the gradient-based differential search strategy following [42] to relax the categorical choices of candidate operations, resulting in the continuous optimization of the supernet. Concretely, given MR-GNAS supernet $\mathcal{S}(\mathcal{X}, \mathcal{A})$ with leaf node set $\mathcal{X}$ and the edge set $\mathcal{A}$, as well as the overall search space $\mathcal{O} = \{\Phi_{op}, \Gamma_{op}, AGG_{op}, \Gamma_{op}^{\mathcal{V}}\}$ containing multi-relational candidate operations, we first define the mixed operation $\bar{o}^{(i,j)}$ relaxed by softmax function as:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} o(x), \tag{6.15}$$

where $\alpha_o^{(i,j)}$ denotes the operation specific weight vector, *i.e.*, certain weight of one edge in $\mathcal{S}$. The leaf node pair $(x^{(i)}, x^{(j)}) \in \mathcal{X}$ is denoted as (i, j) for brevity.

In this way, the task of architecture search is to learn the network architecture $\boldsymbol{\alpha} \in \mathcal{A}$, *i.e.*, a set of continuous edge variables $\boldsymbol{\alpha} = \left\{\alpha_o^{(i,j)} | o \in \mathcal{O}\right\}$. Then, we jointly learn the architecture $\boldsymbol{\alpha}$ and weights $\boldsymbol{w}$ in the proposed MR-GNAS supernet through the bi-level optimization as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \mathcal{L}_{val}\left(\boldsymbol{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}\right), \\ s.t. \quad & \boldsymbol{w}^*(\boldsymbol{\alpha}) = \operatorname{argmin}_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}). \end{aligned} \tag{6.16}$$

At the upper level, we find $\boldsymbol{\alpha}^*$ by minimizing the validation loss $\mathcal{L}_{val}$ for learning network architectures, while at the lower level, we learn weights $\boldsymbol{w}^*$ by minimizing the training loss $\mathcal{L}_{train}$ for learning network parameters. At the end of the gradient-based search, we construct the corresponding discrete network architecture by selecting the operation with the highest architecture weights as $o^* = \operatorname{argmax}_{o \in \mathcal{O}} \alpha_o^{(i,j)}$.

6.4 Experiments

We conduct experiments within two stages following the general NAS pattern [42], *i.e.*, the search stage and the training stage. In the first search stage, we search in the relation-aware supernet of MR-GNAS for an optimal multi-relational GNN model with the bi-level optimization scheme. Then, in the second training stage, we train the searched models from scratch. Finally, we evaluate the proposed MR-GNAS on the tasks of entity classification and link prediction, respectively. The overall experiments are based on the open-source Deep Graph Library (DGL) [169] under the Pytorch framework and run with V100 and Quadro RTX 6000 GPUs.

TABLE 6.3: The statistics of evaluation datasets.

(a) Entity Classification Datasets

Datasets	Entities	Relations	Edges	Labeled	Classes
AIFB	8,285	45	29,043	176	4
MUTAG	23,644	23	74,227	340	2
BGS	333,845	103	916,199	146	2
AM	1,666,764	133	5,988,321	1,000	11

(b) Link Prediction Datasets

Datasets	Entities	Relations	Train	Validate	Test
FB15K-237	15k	237	272k	18k	20k
WN18RR	41k	11	87k	3k	3k

6.4.1 Evaluation Tasks and Datasets

Entity Classification is the task of inferring the entity types on multi-relational graphs. We evaluate our model on four datasets: AIFB, MUTAG, BGS, and AM, whose statistics are listed in Table 6.3 (a). We adopt the standard test protocol in Resource Description Framework (RDF) format from [170] and take the average classification accuracy (ACC%) as the evaluation metric. As for the baselines, we compare the proposed MR-GNAS with following human-designed models: Feat [171], WL [16], RDF2Vec [172], RGCN [2], SACN [28], and CompGCN [17], as well as automated graph NAS model GNAS [168].

Link Prediction is the task of inferring missing facts based on the known facts in knowledge graphs, where the facts are denoted by the set of triples (subject, relation, object), *i.e.*, (u, r, v). Given a knowledge graph, link prediction aims to find the correct entities to complete the facts of $(u, r, ?)$ or the $(?, r, t)$, corresponding to the tail entity prediction and the head entity prediction, respectively. The proposed MR-GNAS serves as a graph encoder and we adopt 'ConvE' [177] as the score function for simple setting. We conduct experiments on FB15K-237 [180] and WN18RR [177] datasets in Table 6.3 (b) with two commonly used evaluation metrics, *i.e.*, Mean Reciprocal Rank (MRR) and Hits@k ($k = 1, 3, 10$) under the filtered setting following [173]. Specifically, through conducting head and tail prediction on all test triples, the rank q of each target entity against the others would be

TABLE 6.4: Results of average accuracy (%) for the entity classification task. Best results are in bold, and the second best results are underlined.

Types	Models	AIFB	MUTAG	BGS	AM
Human-designed	Feat [171]	55.55	77.94	72.41	66.66
	WL [16]	80.55	80.88	86.20	87.37
	RDF2Vec [2]	88.88	67.20	<u>87.24</u>	88.33
	RGCN [2]	<u>95.83</u>	73.23	83.10	89.29
	SACN [28]	-	77.90	-	90.20
	CompGCN [17]	-	85.30	-	**90.60**
Graph NAS	GNAS [168]	88.90	<u>85.86</u>	75.90	85.35
	MR-GNAS (ours)	**100.00**	**89.70**	**89.70**	<u>89.90</u>

TABLE 6.5: MRR and Hits@k results for the link prediction task. Best results are in bold, and the second best results are underlined.

Types	Models	FB15K-237				WN18RR			
		MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
Human-designed	TransE [173]	0.294	0.465	-	-	0.226	0.501	-	-
	TransH [174]	0.233	0.401	-	-	0.186	0.451	-	-
	DisMult [175]	0.241	0.419	0.263	0.155	0.430	0.490	0.440	0.390
	ComplEx [176]	0.247	0.428	0.275	0.158	<u>0.440</u>	0.510	<u>0.460</u>	<u>0.410</u>
	ConvE [177]	<u>0.325</u>	<u>0.501</u>	0.356	<u>0.237</u>	0.430	0.520	0.440	0.400
	RGCN [2]	0.248	0.417	-	0.151	-	-	-	-
	ConvKB [178]	0.243	0.421	<u>0.371</u>	0.155	0.249	<u>0.524</u>	0.417	0.057
	VR-GCN [179]	0.248	0.432	0.272	0.159	-	-	-	-
Graph NAS	GNAS [168]	0.273	0.422	0.299	0.197	0.175	0.273	0.199	0.123
	MR-GNAS (ours)	**0.348**	**0.530**	**0.380**	**0.258**	**0.456**	**0.541**	**0.470**	**0.414**

calculated, leading to the overall test prediction ranks Q. Then, MRR evaluates the average of the inverse of the obtained ranks [181] as $MRR = 1/|Q| \sum_{q \in Q} 1/q$. And Hits@$k$ measures the ratio of predictions ranked in top $k = \{1, 2, 3\}$ as $Hit@k = |\{q \in Q : q \leq k\}|/|Q|$. The higher MRR and Hit@k indicate the better link prediction performance. We compare the proposed MR-GNAS with following human-designed methods: TransE [173], TransH [174], DistMult [175], ComplEx [176], ConvE [177], RGCN [2], ConvKB [178], and VR-GCN [179], as well as the automated graph NAS method GNAS [168]. Note that considering GNAS is proposed to serve single-relational graph learning, we reproduce the experimental results of both entity classification and link prediction tasks by adapting it to the multi-relational setting without relation representation learning.

6.4.2 Experimental Results.

Entity Classification. The experiment results of the entity classification task are reported in Table 6.4. In general, our proposed MR-GNAS achieves the best and highly competitive performance of average classification accuracy on all datasets. Specifically, MR-GNAS significantly exceeds the existing models with the performance improvement of 4.4%, 5.2%, and 2.8% on AIFB, BGS, and MUTAG datasets, respectively. Due to large-scale and complex data statistics of AM dataset, automated search in the proposed supernet of MR-GNAS would become more complicated than other datasets. It is more challenging to derive optimal multi-relational GNN models driven by large-scale graphs with the

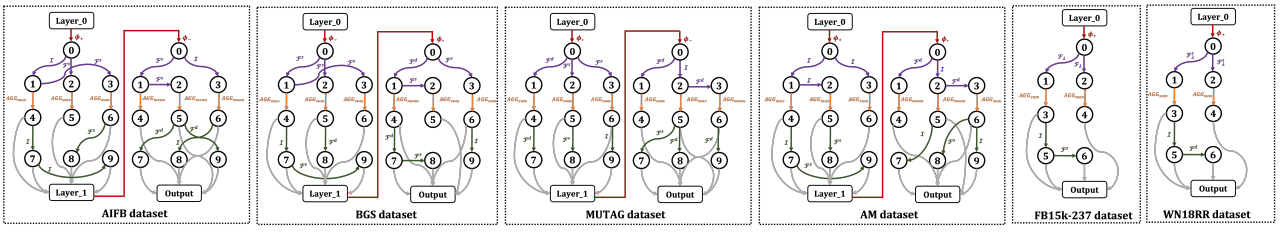

FIGURE 6.4: The searched models from the supernet of MR-GNAS on different datasets for entity classification and link prediction tasks.

computation memory limitation, causing limited entity classification performance improvement. This point could be further verified by the performance of GNAS, which is under-performed than most existing human-designed models. Even though, our MR-GNAS still achieves better classification results than automated GNAS (89.90% *vs.* 85.35%) and most human-designed models. We attribute the superiority of the proposed MR-GNAS to the well-designed and wide-scope search spaces, as well as the multi-relational message passing based supernet construction, enabling its excellent performance to automatically develop task-specific and data-driven MR-GNN architectures.

Link Prediction. The experiment results of the link prediction task is presented in Table 6.5. On the whole, our proposed MR-GNAS achieves outstanding performance on both FB15K-237 dataset and WN18RR dataset. MR-GNAS significantly surpasses the human-designed multi-relational GNN models, *i.e.*, RGCN and VR-GCN, on FB15K-237 dataset with 40.3% performance improvement for MRR from 0.248 to 0.348. Moreover, MR-GNAS still has better link prediction performance when compared with the automated GNAS model. Specifically, MR-GNAS improves MRR on FB15K-237 from 0.273 to 0.348, while on WN18RR dataset, the MRR performance improvement is greater from 0.175 to 0.456. We attribute this to the joint node and relation representation learning of the proposed MR-GNAS in the entity-relation integration cell, as well as the relation linear combination and mapping, since GNAS is unable to deal with the relation representation learning. This further reflects the effectiveness of the delicate search space design with relation-aware message passing of the proposed MR-GNAS. By enriching relation-aware candidate operations in the fine-grained manner, MR-GNAS could automatically derive expressive and powerful MR-GNNs for multi-relational graph learning.

6.4.3 Discussion and Analysis

Understanding of Searched Models We present the derived models for each task and dataset by searching in the relation-aware supernet of MR-GNAS in Fig. 6.4. We set the number of leaf nodes in the supernet as $[1, 3, 3, 3]$ with two layers and $[1, 2, 2, 2]$ with a single layer for entity classification and link prediction, respectively. In general, we can observe that different tasks and datasets have different preferences for the functional candidate operations in each cell. Therefore, resort to the well-designed search space with adequate candidates, the proposed MR-GNAS could enjoy the advantages of different functional operations for multi-relational message passing and automatically select the optimal ones to build creative and expressive MR-GNN architectures.

Scalability with Number of Relations We discuss the effects of different numbers of relation basis vectors on AM dataset for the entity classification task in Fig 6.5. As can be observed, the performance would not improve along with the increased numbers of bases, and there might be a proper basis number for better performance, *i.e.*, 80 in our searched model. Hence, the number of

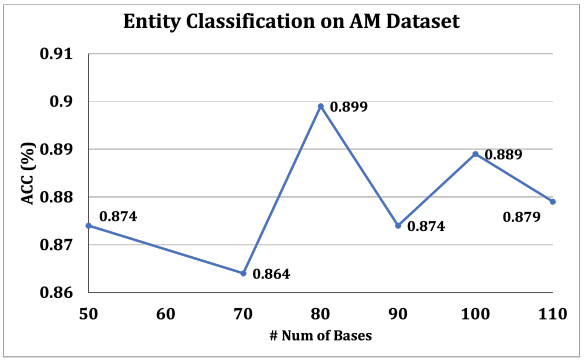

FIGURE 6.5: Effects of different numbers of relation basis vectors on AM dataset for the entity classification task.

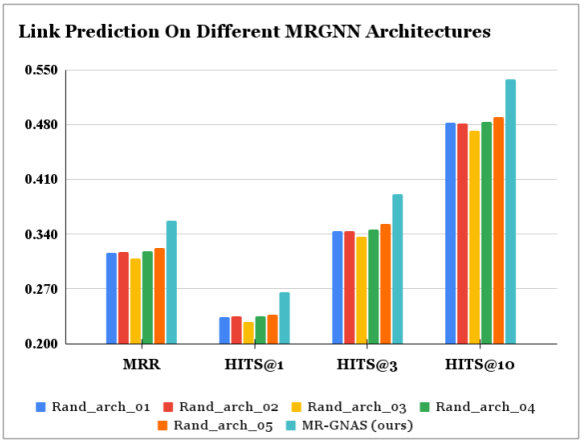

FIGURE 6.6: Comparison between randomly-selected architectures *vs.* gradient-based differential searched architectures on FB15K-237 dataset for link prediction.

relation basis vectors could be taken as a hyper-parameter of the automated MR-GNAS tuned for better multi-relational graph learning.

Effectiveness of the Search Strategy To verify the effectiveness of the search strategy on deriving optimal searched models, we randomly select five MR-GNN architectures (Rand_arch_01∼Rand_arch_05) from the proposed search space on FB15K-237 dataset for the link prediction task, and make comparison with the derived MR-GNAS based on the gradient-based differential search strategy. The experimental results are presented in Fig. 6.6. As can be observed, the proposed MR-GNAS achieves the best performance on link prediction for all metrics of MRR and Hit@k than all other randomly-built architectures, significantly verifying the effectiveness of the search strategy adopted by the proposed method. Moreover, the randomly-built MR-GNN architectures from the proposed search space are all with MRR metrics over 0.270, which still outperform existing multi-relational GNNs, *e.g.*, RGCN with 0.248 MRR in Table 6.5, further illustrating the effectiveness of the proposed search space with multi-relational message passing schema.

Search Efficiency To illustrate the search efficiency of the proposed method, we report the search time costs of 10 epochs between our MR-GNAS and graph NAS model GNAS presented in Table 6.6.

TABLE 6.6: Search time of 10 epochs (clock time in seconds) comparison.

Search time (10 epochs)		GNAS [168]	MR-GNAS (ours)
Entity Classification Dataset	AIFB	36.49	36.12
	MUTAG	69.97	69.11
	BGS	26.59	26.89
	AM	317.95	329.42
Link Prediction Dataset	FB15K-237	2.41	3.92
	WN18RR	4.23	5.91

As can be observed, even with enlarged search spaces containing more operations and relation representation learning progress, MR-GNAS does not take much extra search time compared with GNAS, illustrating the outstanding search efficiency of the proposed method.

6.5 Related Work

Multi-Relational Graph Neural Networks. Multi-relational graphs are widespread graph-structured data in the real world, reflecting complex and diverse relationships among various objects [173, 177, 179, 182]. As a typical category of multi-relational graphs, knowledge graphs (KGs) provide a formal understanding of the world based on the human knowledge [28, 167, 175, 183]. Specifically, KGs denote nodes as entities and edges as relations with labels, indicating the specific triplet facts connecting arbitrary two entities with a certain relation. Knowledge graph representation learning (KRL) is an important research line of KG analysis, where entities and relations would be embedded into low-dimensional latent spaces for capturing their semantic and context information. There are two critical components of KRL, *i.e.*, embedding models and score functions, where the former learns low-dimensional entity and relation representations and the latter measures the plausibility of triplet facts. As a prevalent type of KG embedding models, a multi-relational graph neural network (MR-GNN) utilizes the multi-relational structure connections to benefit GNN learning under message passing schema. MR-GNN models jointly learn node and relation representations of various entity and relation types with different edge directions. Typically, RGCN [2] constructed graph convolutional network (GCN) on relational graphs with relation-specific filters, where basis and block diagonal decomposition were introduced to deal with the scale issue of numerous relation types. SACN [28] developed a weighted GCN as the embedding model to capture node structures and attributes, as well as edge relation types, with adaptive relation-specific scalar weight learning. Despite the satisfying performance, these methods paid more attention to the node representation learning while the relation representation learning was not well addressed. In light of this, CompGCN [17] introduced the entity-relation embedding composition to incorporate relation learning, resulting in a more general relational GNN framework with superior performance.

Nevertheless, all these relational GNN models are human-designed, leading to heavy reliance on expert knowledge with much human effort cost, since massive manual enumerations need to be conducted for

TABLE 6.7: The comprehensive comparison between the proposed MR-GNAS and other existing graph NAS models.('Single-rel' and 'Multi-rel' denote the single-relational graphs and multi-relational graphs, respectively. 'Coarse' and 'Fine' indicate the coarse-grained and fine-grained search space characteristics, respectively. 'Dete. Differ.' and 'Stoc. Differ' denote the deterministic differential search strategy and stochastic differential search strategy, respectively.)

Methods	Graph Types	Search Spaces		Search Strategy
		Character	Models	
GraphNAS [40]	Single-rel	Coarse	GNN	RL
AGNN [156]	Single-rel	Coarse	GNN	EA+RL
SANE [41]	Single-rel	Coarse	GNN	Dete. Differ.
SANG [157]	Single-rel	Coarse	GNN	RL
Auto-SF [155]	Multi-rel	Fine	Bi-Linear	Greedy Search
GNAS [168]	Single-rel	Fine	GAP	Dete. Differ.
MR-GNAS (ours)	Multi-rel	Fine	MR-GNN	Dete. Differ.

obtaining expressive network architectures. To relieve human effort and enlarge the multi-relational GNN design prospect, we leverage the NAS technique for automated multi-relational GNN architecture design to address the above challenges. With the fine-grained search space and relation-aware supernet, the proposed MR-GNAS could automatically develop powerful multi-relational GNNs for discriminative node and relation representation learning.

Graph Neural Architecture Search. As a critical research branch of automated machine learning, NAS techniques [42, 44, 45, 48] have been introduced into automated GNN model design recently, leading to a promising research direction on graph NAS. Existing research on graph NAS has greatly enlarged the design picture of automated GNN development and relieved the human effort cost of discovering excellent and powerful GNN architectures [40, 41, 43, 153–157]. Generally, graph NAS methods have two important components: search space defines functional candidate operations and search strategy explores optimal network architectures. Typically, GraphNAS [40] and AGNN [156] constructed the micro-level search space containing classical GNN layers and related hyper-parameters, along with architecture controllers based reinforcement learning (RL) search strategy. And SNAG [157] further simplified GraphNAS at the micro-level and introduced the macro-level inter-layer connections. Based on this search space, SANE [41] implemented DARTS [42], a gradient-based differential search strategy, on graphs for deriving optimal GNN architectures efficiently. Also taking the gradient-based DARTS as the search strategy, GNAS [168] developed a Graph Neural Architecture Paradigm (GAP) composed of two types of operations, *i.e.*, feature filtering and neighbor aggregation, to explore better architectures with the optimal depth of message passing on the graphs.

Despite excellent learning abilities, these graph NAS methods mainly work in the single-relational graph setting, significantly limiting their applications of learning on multi-relational graphs. In light of these, few researchers divert their attention to the multi-relational graph setting. For example, Auto-SF [155] worked on the score function (SF) component of KRL to automatically search in a bi-linear model based SF search space. Through a unified representation of existing prevalent SFs, Auto-SF could derive creative and KG-dependent SFs with superior KG analysis abilities. Nevertheless, the

advance of automated GNN learning is not enjoyed by current KRL embedding models on multi-relational graphs. The main reason behind this lies in the complexity and difficulty of training GNNs when lacking domain-specific constraints on diverse multi-relational graph structures, as also verified by Auto-SF [155]. This further leads to instinctively less well performance of GNN embedding models than bi-linear models. However, due to the powerful abilities of GNNs on complex graph structure and representation learning, it motivates us to overcome such challenges to explore the automated multi-relational GNN architecture development. Although Auto-GEL [43] attempted to involve the implicit link information of multi-relational graphs for automated multi-relational GNN design, it still falls into the coarse-grained design of search spaces by integrating existing typical GNN layer based operations straightforwardly. That means the principle of multi-relational GNN architectures is not changed, leading to significant limitations on the capacity and scope of novel multi-relational GNN architecture development.

Therefore, instead of the simple ensemble of existing GNN layer components, we propose a fine-grained search space with functional candidate operations that embrace the multi-relational message passing schema, leading to a novel relation-aware supernet followed by a gradient-based search strategy. The comprehensive comparison between the proposed MR-GNAS and other existing graph NAS models are listed in Table 6.7. As can be observed, our MR-GNAS is the only method that works on automated multi-relational GNN architecture development with the fine-grained search space.

6.6 Conclusion

This chapter proposes a novel framework for multi-relational graph neural architecture search, dubbed MR-GNAS, to automatically develop innovative and excellent multi-relational GNN architectures. Specifically, to tackle the challenges of the single-relational setting and coarse-grained search spaces in existing graph NAS, MR-GNAS first designs a fine-grained search space that embraces the full-pipe multi-relational message passing schema, and then constructs a relation-aware supernet composed of four sequentially stacked cells within the tree topology. In this way, MR-GNAS greatly enlarges the search capacity and improves the search flexibility for enabling expressive architecture search scopes. Through searching with a gradient-based strategy in the supernet, the proposed MR-GNAS could automatically derive creative multi-relational GNN architectures and jointly learn discriminative node and relation representations, enabling excellent multi-relational graph learning abilities. Experimental results demonstrate that the proposed MR-GNAS not only achieves outstanding performance on entity classification and link prediction, but also generally provides fresh ideas for future task-specific and data-driven multi-relational GNN design.

Chapter 7

GNN Performance Evaluation on Unseen Graphs without Labels

7.1 Introduction

As prevalent graph data learning models, graph neural networks (GNNs) have attracted much attention and achieved great success for various graph structural data related applications in the real world [5, 19, 21–25, 117, 118]. In practical scenarios of GNN model deployment and serving [115, 116, 184], understanding and evaluating GNN models' performance is a vital step [185–187], where model designers need to determine if well-trained GNN models will perform well in practical serving, and users want to know how the in-service GNN models will perform when inferring on their own test graphs [73]. Conventionally, model evaluation utilizes well-annotated datasets for testing to calculate certain model performance metrics (*e.g.*, accuracy and F1-score) [102, 103, 107, 185]. However, it may fail to work well in real-world GNN model deployment and serving, where the unseen test graphs are usually not annotated, making it difficult to obtain essential model performance metrics without ground-truth labels [72, 74]. As shown in Fig. 7.1 (a-1), taking *node classification accuracy* metric as an example, it is typically calculated as the percentage of correctly predicted node labels. However, when ground-truth node class labels are unavailable, we can not verify whether the predictions of GNNs are correct, and thus cannot get the overall accuracy of the model.

In light of this, a natural question, referred to **GNN model evaluation** problem, arises: *in the absence of labels in an unseen test graph, can we estimate the performance of a well-trained GNN model?*

In this chapter, we provide a confirmed answer together with an effective solution to this question, as shown in Fig.7.1 (a-2). Given a well-trained GNN model and an unseen test graph without labels, GNN model evaluation directly outputs the overall accuracy of this GNN model. This enables users to

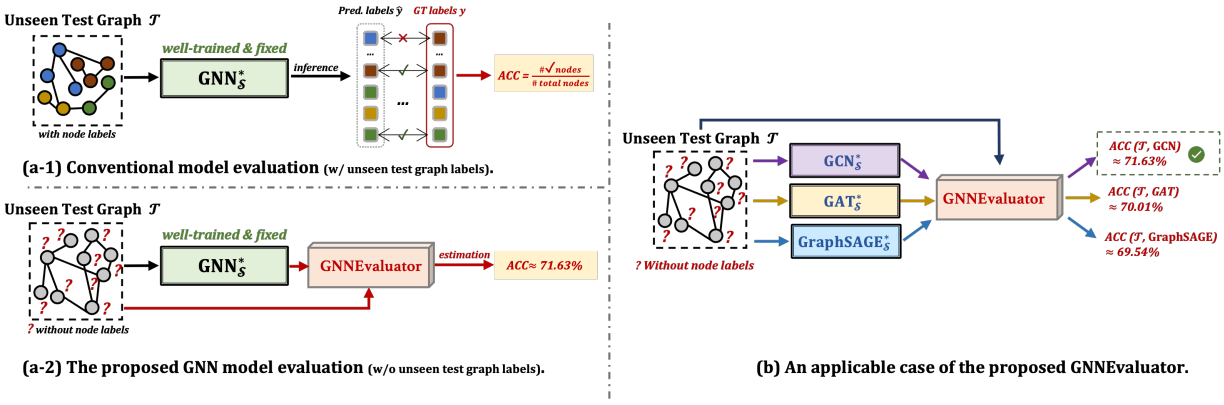

FIGURE 7.1: Illustration of conventional model evaluation *vs.* the proposed GNN model evaluation with its applicable case. The symbol $*$ indicates GNNs are well-trained and fixed during model evaluation.

understand their GNN models at hand, benefiting many GNN deployment and serving scenarios in the real world [100, 101, 104, 188–192]. For example, given a GNN model in service (*e.g.*, a GNN model trained and served with Amazon DGL and SageMaker [193]), GNN model evaluation can provide users with a confidence score based on its estimated node classification accuracy, so that users know how much faith they should place in GNN-predicted node labels on their own unlabeled test graphs. Moreover, as shown in Fig. 7.1 (b), given a set of available well-trained GNNs in service, GNN model evaluation can provide users with reliable guidance to make an informed choice among these deployed GNN models on their own unlabeled test graphs.

In this chapter, we focus on the node classification task, with accuracy as the primary GNN model evaluation metric. Developing effective GNN model evaluation methods for this task faces three-fold challenges:

Challenge-1: The distribution discrepancies between various real-world unseen test graphs and the observed training graph are usually complex and diverse, incurring significant uncertainty for GNN model evaluation.

Challenge-2: It is not allowed to re-train or fine-tune the practically in-service GNN model to be evaluated, and the only accessible model-related information is its outputs. Hence, how to fully exploit the limited GNN outputs and integrate various training-test graph distribution differences into discriminative discrepancy representations is critically important.

Challenge-3: Given the discriminative discrepancy representations of training-test graph distributions, how to develop an accurate GNN model evaluator to estimate the node classification accuracy of an in-service GNN on the unseen test graph is the key to GNN model evaluation.

To address the above challenges, in this chapter, we propose a two-stage GNN model evaluation framework, including (1) DiscGraph set construction and (2) GNNEvaluator training and inference. More specifically, in the first stage, we first derive a set of meta-graphs from the observed training graph,

which involves wide-range and diverse graph data distributions to simulate (ideally) any potential unseen test graphs in practice, so that the complex and diverse training-test graph data distribution discrepancies can be effectively captured and modeled (*Challenge-1*). Based on the meta-graph set, we build a set of discrepancy meta-graphs (DiscGraph) by jointly exploiting latent node embeddings and node class predictions of the GNN with a discrepancy measurement function, which comprehensively integrates GNN output discrepancies to expressive graph structural discrepancy representations involving discrepancy node attributes, graph structures, and accuracy labels (*Challenge-2*). In the second stage, we develop a GNNEvaluator composed of a typical GCN architecture and an accuracy regression layer, and we train it to precisely estimate node classification accuracy with effective supervision from the representative DiscGraph set (*Challenge-3*). During the inference, the trained GNNEvaluator could directly output the node classification accuracy of the in-service GNN on the unseen test graph without any node class labels.

In summary, the contributions of our work are listed as follows:

- **Problem.** We study a new research problem, GNN model evaluation, which opens the door for understanding and evaluating the performance of well-trained GNNs on unseen real-world graphs without labels in practical GNN model deployment and serving.
- **Solution.** We design an effective solution to simulate and capture the discrepancies of diverse graph data distributions, together with a GNNEvaluator to estimate node classification accuracy of the in-service GNNs, enabling accurate GNN model evaluation.
- **Evaluation.** We evaluate our method on real-world unseen and unlabeled test graphs, achieving a low error (*e.g.*, as small as 2.46%) compared with ground-truth accuracy, demonstrating the effectiveness of our method for GNN model evaluation.

7.2 Related Work

Our research is related to existing studies on *predicting model generalization error*, which aims to estimate a model's performance on unlabeled data from the unknown and shifted distributions [69–71, 73, 74]. However, these researches are designed for data in Euclidean space (*e.g.*, images) while our research is dedicatedly designed for graph structural data. Our research also significantly differs from others in unsupervised graph domain adaption [194–196], out-of-distribution (OOD) generalization [197, 198], and OOD detection [199, 200], in the sense that we aim to estimate well-trained GNN models' performance, rather than improve the generalization ability of new GNN models.

Predicting Model Generalization Error. Our work is relevant to the line of research on predicting model generalization error, which aims to develop a good estimator of a model's classifier performance on unlabeled data from the unknown distributions in the target domain, when the estimated models' classifier has been trained well in the source domain with labeled data [69–74]. Typically, Guillory

et al. [73] proposed to estimate a classifier's performance of convolutional neural network (CNN) models on image data based on a designed criterion, named difference of confidences (DoC), that estimates and reflects model accuracy. And Garg et al. [70] proposed to learn a score-based Average Thresholded Confidence (ATC) by leveraging the softmax probability of a CNN classifier, whose accuracy is estimated as the fraction of unlabeled images that receive a score above that threshold. In contrast, Deng et al. [69, 74] directly predicted CNN classifier accuracy by deriving distribution distance features between training and test images with a linear regression model.

However, these existing methods mostly focus on evaluating CNN model classifiers on image data in computer vision, and the formulation of evaluating GNNs for graph structural data still remains under-explored in graph machine learning. Concretely, conducting the model evaluation on GNNs for graph structural data has two critical challenges: (1) different from Euclidean image data, graph structural data lies in non-Euclidean space with complex and non-independent node-edge interconnections, so that its node contexts and structural characteristics significantly vary under wide-range graph data distributions, posing severer challenges for GNN model evaluation when serving on unknown graph data distributions. (2) GNNs have entirely different convolutional architectures from those of CNNs, when GNN convolution aggregates neighbor node messages along graph structures. Such that GNNs trained on the observed training graph might well fit its graph structure, and due to the complexity and diversity of graph structures, serving well-trained GNNs on unlabeled test distributions that they have not encountered before would incur more performance uncertainty.

Hence, in this chapter, we first investigate the GNN model evaluation problem for graph structural data with a clear problem definition and a feasible solution, taking a significant step towards understanding and evaluating GNN models for practical GNN deployment and serving. Our proposed two-stage GNN model evaluation framework directly estimates the node classification accuracy results of specific well-trained GNNs on practical unseen graphs without labels. Our method could not only facilitate model designers to better evaluate well-trained GNNs' performance in practical serving, but also provide users with confidence scores or model selection guidance, when using well-trained GNNs for inferring on their own test graphs.

Unsupervised Graph Domain Adaption. Our work is also relevant to the unsupervised graph domain adaption (UGDA) problem, whose goal is to develop a GNN model with both labeled source graphs and unlabeled target graphs for better generalization ability on target graphs. Typically, existing UGDA methods focus on mitigating the domain gap by aligning the source graph distribution with the target one. For instance, Yang et al. [194] and Shen et al. [201] optimized domain distance loss based on the statistical information of datasets, *e.g.*, maximum mean discrepancy (MMD) metric. Moreover, DANE [195] and UDAGCN [196] introduced domain adversarial methods to learn domain-invariant embeddings across the source domain and the target domain. Therefore, the critical distinction between our work and UGDA is that our work is primarily concerned with evaluating the

GNNs' performance on unseen graphs without labels, rather than improving the GNNs' generalization ability when adapting to unlabeled target graphs. Besides, different from UGDA which uses the unlabeled target graphs in their model design and training stage, the GNN model evaluation problem explored by our work is not allowed to access the unlabeled test graphs, *i.e.*, unseen in the whole GNN model evaluation process.

Graph Out-of-distribution (OOD) Generalization & Detection. Out-of-distribution (OOD) generalization [197, 198] on graphs aims to develop a GNN model given several different but related source domains, so that the developed model can generalize well to unseen target domains. Li et al. [197] categorized existing graph OOD generalization methodologies into three conceptually different branches, *i.e.*, data, model, and learning strategy, based on their positions in the graph machine learning pipeline. We would like to highlight that, even ODD generalization and our proposed GNN model evaluation both pay attention to the general graph data distribution shift issue, we have different purposes: our proposed GNN model evaluation aims to evaluate well-trained GNNs' performance on unseen test graphs, while ODD generalization aims to develop a new GNN model for improve its performance or generalization capability on unseen test graphs. Moreover, OOD detection [202], aims to detect test samples drawn from a distribution that is different from the training distribution, with the definition of distribution to be well-defined according to the application in the target domain. Hence, the primary difference between OOD detection and our GNN model evaluation is, OOD detection works on detecting OOD test samples at the data level and our GNN model evaluation works on evaluating well-trained GNN's performance on ODD data at the model level.

7.3 Problem Definition

Preliminary. Consider that we have a fully-observed training graph $\mathcal{S} = (\mathbf{X}, \mathbf{A}, \mathbf{Y})$, where $\mathbf{X} \in \mathbb{R}^{N \times d}$ denotes N nodes with d-dimensional features, $\mathbf{A} \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix indicating the edge connections, and $\mathbf{Y} \in \mathbb{R}^{N \times C}$ denotes the C-classes of node labels. Then, training a GNN model on $\mathcal{S}$ for node classification objective can be denoted as:

$$\min_{\boldsymbol{\theta}_{\mathcal{S}}} \mathcal{L}_{\text{cls}} \left(\hat{\mathbf{Y}}, \mathbf{Y} \right), \text{ where } \mathbf{Z}_{\mathcal{S}}, \hat{\mathbf{Y}} = \text{GNN}_{\boldsymbol{\theta}_{\mathcal{S}}}(\mathbf{X}, \mathbf{A}). \tag{7.1}$$

where $\boldsymbol{\theta}_{\mathcal{S}}$ denotes the parameters of GNN trained on $\mathcal{S}$, $\mathbf{Z}_{\mathcal{S}} \in \mathbb{R}^{N \times d_1}$ is the output node embedding of graph $\mathcal{S}$ from $\text{GNN}_{\boldsymbol{\theta}_{\mathcal{S}}}$, and $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times C}$ denotes GNN predicted node labels. By optimizing the node classification loss function $\mathcal{L}_{\text{cls}}$ between GNN predictions $\hat{\mathbf{Y}}$ and ground-truth node labels $\mathbf{Y}$, *i.e.*, cross-entropy loss, we could obtain a well-trained GNN with optimal weight parameters $\boldsymbol{\theta}^*_{\mathcal{S}}$, denoted as $\text{GNN}^*_{\mathcal{S}}$. Then, the node classification accuracy can be calculated to reflect GNN performance as: $\text{Acc}(\mathcal{S}) = \sum_{i=1}^{N} (\hat{y}_i == y_i)/N$, which indicates the percentage of correctly predicted node labels between the GNN predicted labels $\hat{y}_i \in \hat{\mathbf{Y}}$ and ground truths $y_i \in \mathbf{Y}$.

Given an unseen and unlabeled graph $\mathcal{T} = (\mathbf{X}', \mathbf{A}')$ including M nodes with its features $\mathbf{X}' \in \mathbb{R}^{M \times d}$ and structures $\mathbf{A}' \in \mathbb{R}^{M \times M}$, we assume the covariate shift between $\mathcal{S}$ and $\mathcal{T}$, where the distribution shift mainly lies in node numbers, node context features, and graph structures, but the label space of $\mathcal{T}$ keeps the same with $\mathcal{S}$, *i.e.*, all nodes in $\mathcal{T}$ are constrained in the same C-classes. Due to the absence of ground-truth node labels, we could NOT directly calculate the node classification accuracy to assess the performance of the well-trained $\text{GNN}^*_{\mathcal{S}}$ on $\mathcal{T}$. In light of this, we present a new research problem for GNNs as:

Definition of GNN Model Evaluation. Given the observed training graph $\mathcal{S}$, its well-trained model $\text{GNN}^*_{\mathcal{S}}$, and an unlabeled unseen graph $\mathcal{T}$ as inputs, the **goal** of GNN model evaluation aims to learn an accuracy estimation model $f_{\boldsymbol{\phi}}(\cdot)$ parameterized by ϕ as:

$$\text{Acc}(\mathcal{T}) = f_{\boldsymbol{\phi}}(\text{GNN}^*_{\mathcal{S}}, \mathcal{T}), \tag{7.2}$$

where $f_{\boldsymbol{\phi}} : (\text{GNN}^*_{\mathcal{S}}, \mathcal{T}) \rightarrow a$ and $a \in \mathbb{R}$ is a scalar denoting the overall node classification accuracy $\text{Acc}(\mathcal{T})$ for all unlabeled nodes of $\mathcal{T}$. When the context is clear, we will use $f_{\boldsymbol{\phi}}(\mathcal{T})$ for simplification.

7.4 The Proposed Method

7.4.1 Overall Framework of GNN Model Evaluation

As shown in Fig. 7.2, the proposed overall framework of GNN model evaluation contains two stages: (1) DiscGraph set construction; and (2) GNNEvaluator training and inference.

Stage 1. Given the observed training graph $\mathcal{S}$, we first extract a seed graph from it followed by augmentations, leading to a set of meta-graphs from it for simulating any potential unseen graph distributions in practice. Then, the meta-graph set $\mathcal{G}_{\text{meta}}$ and the observed training graph $\mathcal{S}$ are fed into the well-trained GNN for obtaining latent node embeddings, *i.e.*, $\mathbf{Z}^{(i,*)}_{\text{meta}}$ and $\mathbf{Z}^*_{\mathcal{S}}$, respectively. After, a discrepancy measurement function $D(\cdot)$ works on these two latent node embeddings, leading to ① discrepancy node attributes $\mathbf{X}^i_{\text{disc}}$. Meanwhile, the output node class predictions of well-trained GNN on each meta-graph are used to calculate the node classification accuracy according to its ground-truth node labels, leading to the new ② scalar accuracy labels y^i_{disc}. Along with ③ graph structures $\mathbf{A}^i_{\text{disc}}$ from each meta-graph, these three important components composite the set of discrepancy meta-graphs (DiscGraph) in expressive graph structural discrepancy representations.

Stage 2. In the second stage, the representative DiscGraph set is taken as effective supervision to train a GNNEvaluator through accuracy regression. During the inference, the trained GNNEvaluator could directly output the node classification accuracy of the in-service GNN on the unseen test graph $\mathcal{T}$ without any node class labels.

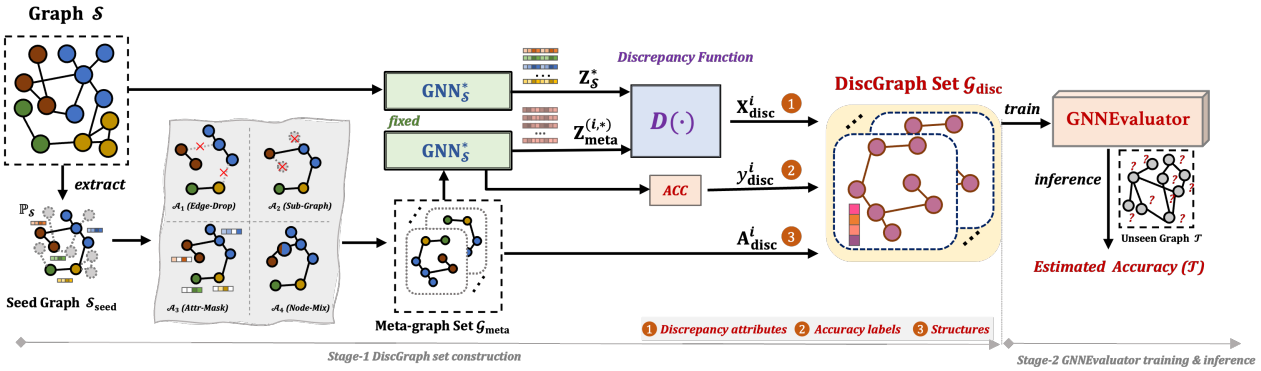

FIGURE 7.2: Overall two-stage pipeline of GNN model evaluation with (1) DiscGraph set construction and (2) GNNEvaluator training and inference.

7.4.2 Discrepancy Meta-graph Set Construction

The critical challenge in developing GNN model evaluation methods is complex and diverse graph data distribution discrepancies between various real-world unseen test graphs and the observed training graph. As the in-service GNN in practice fits the observed training graph well, making inferences on various and diverse unseen test graphs would incur significant performance uncertainty for GNN model evaluation, especially when the node classification accuracy can not be calculated due to label absence.

In light of this, we propose to construct a set of diverse graphs to simulate wide-range discrepancies of potential unseen graph distributions. Specifically, such a graph set should have the following characteristics: (1) *sufficient quantity:* it should contain a relatively sufficient number of graphs with diverse node context and graph structure distributions; (2) *represented discrepancy:* the node attributes of each graph should indicate its distribution distance gap towards the observed training graph; (3) *known accuracy:* each graph should be annotated by node classification accuracy as its label.

To address the characteristic of (1) *sufficient quantity*, we introduce a meta-graph simulation strategy to synthesize a wide variety of meta-graphs for representing any potential unseen graph distributions that may be encountered in practice. Specifically, we extract a seed sub-graph $\mathcal{S}_{\text{seed}}$ from the observed training graph $\mathcal{S}$. The principle of seed sub-graph selection strategy is that $\mathcal{S}_{\text{seed}}$ involves the least possible distribution shift within $\mathbb{P}_{\mathcal{S}}$ from the observed training graph, and shares the same label space with $\mathcal{S}$, satisfying the assumption of covariate shift. Then, $\mathcal{S}_{\text{seed}}$ is fed to a pool of graph augmentation operators as

$$\mathcal{A} = \{\mathcal{A}_1(\textsc{EdgeDrop}), \mathcal{A}_2(\textsc{Subgraph}), \mathcal{A}_3(\textsc{AttrMask}), \mathcal{A}_4(\textsc{NodeMix})\}, \tag{7.3}$$

which belongs to the distribution $\mathbb{P}_{\mathcal{A}}(p_i, |\epsilon)$ with $\{p_i\}_{i=1}^{4} \in (0,1)$ represent the augmentation ratio on $\mathcal{S}_{\text{seed}}$ for each augmentation type, and ϵ determines the probability of sampling a particular augmentation type. For instance, $p_1 = 0.5$ means dropping 50% edges in $\mathcal{S}_{\text{seed}}$. In this way, we can obtain a set of meta-graphs $\mathcal{G}_{\text{meta}} = \left\{g^i_{\text{meta}}\right\}_{i=1}^{K}$ with K numbers of graphs, and we have $\mathcal{G}_{\text{meta}} \sim \mathbb{P}_{\mathcal{A}} \times \mathbb{P}_{\mathcal{S}} : \mathcal{S}_{\text{seed}} \rightarrow g^i_{\text{meta}}$. For each meta-graph $g^i_{\text{meta}} = \left\{\mathbf{X}^i_{\text{meta}}, \mathbf{A}^i_{\text{meta}}, \mathbf{Y}^i_{\text{meta}}\right\}$, where $\mathbf{X}^i_{\text{meta}} \in \mathbb{R}^{M_i \times d}$, $\mathbf{A}^i_{\text{meta}} \in \mathbb{R}^{M_i \times M_i}$, and $\mathbf{Y}^i_{\text{meta}} \in \mathbb{R}^{M_i \times C}$ have M_i number of nodes, d-dimensional features, and known node labels from $\mathcal{S}_{\text{seed}}$ belonging to C classes.

To incorporate the characteristic of (2) *represented discrepancy*, we fully exploit the outputs of latent node embeddings and node class predictions from well-trained $\text{GNN}^*_{\mathcal{S}}$, and integrate various training-test graph distribution differences into discriminative discrepancy representations. Specifically, given the $\text{GNN}^*_{\mathcal{S}}$ learned latent node embeddings $\mathbf{Z}^*_{\mathcal{S}} = \text{GNN}^*_{\mathcal{S}}(\mathbf{X}, \mathbf{A})$, and $\mathbf{Z}^{(i,*)}_{\text{meta}}, \hat{\mathbf{Y}}^{(i,*)}_{\text{meta}} = \text{GNN}^*_{\mathcal{S}}(\mathbf{X}^i_{\text{meta}}, \mathbf{A}^i_{\text{meta}})$, where $\mathbf{Z}^*_{\mathcal{S}} \in \mathbb{R}^{N \times d_1}$ and $\mathbf{Z}^{(i,*)}_{\text{meta}} \in \mathbb{R}^{M_i \times d_1}$, we derive a distribution discrepancy measurement function $D(\cdot)$ to calculate the discrepancy meta-graph node attributes as:

$$\mathbf{X}^i_{\text{disc}} = D(\mathbf{Z}^{(i,*)}_{\text{meta}}, \mathbf{Z}^*_{\mathcal{S}}) = \frac{\mathbf{Z}^{(i,*)}_{\text{meta}} \mathbf{Z}^{*\,\text{T}}_{\mathcal{S}}}{\left\|\mathbf{Z}^{(i,*)}_{\text{meta}}\right\|_2 \cdot \left\|\mathbf{Z}^*_{\mathcal{S}}\right\|_2}, \tag{7.4}$$

where $\mathbf{X}^i_{\text{disc}} \in \mathbb{R}^{M_i \times N}$ reflects the node-level distribution discrepancy between each g^i_{meta} and $\mathcal{S}$ with the well-trained $\text{GNN}^*_{\mathcal{S}}$. Each element in $x^i_{u,v} \in \mathbf{X}^i_{\text{disc}}$ denotes the node embedding discrepancy gap between a certain node u in the i-th meta-graph and a node v in the observed training graph. Taking this representation as node attributes could effectively integrate representative node-level discrepancy produced by well-trained GNN's node embedding space.

Meanwhile, the characteristic of (3) *known accuracy* can be involved with the outputs of node class predictions produced by $\text{GNN}^*_{\mathcal{S}}$ on meta-graph $\hat{\mathbf{Y}}^{(i,*)}_{\text{meta}} = \{\hat{y}^j_{\text{meta}(i,*)}\}_{j=1}^{M_i}$. We calculate the node classification accuracy on the meta-graph given its ground-truth node class labels $\mathbf{Y}^i_{\text{meta}} = \{y^j_{\text{meta}(i)}\}_{j=1}^{M_i}$ as:

$$y^i_{\text{disc}} = \text{Acc}(g^i_{\text{meta}}) = \frac{\sum_{j=1}^{M_i} (\hat{y}^j_{\text{meta}(i,*)} == y^j_{\text{meta}(i)})}{M_i}, \tag{7.5}$$

where $y^j_{\text{meta}(i)}$ and $\hat{y}^j_{\text{meta}(i,*)}$ denote the ground truth label and $\text{GNN}^*_{\mathcal{S}}$ predicted label of j-th node in the i-th graph of the meta-graph set, respectively. Note that $y^i_{\text{disc}} \in \mathbb{R}$ is a continuous scalar denoting node classification accuracy under specific $\text{GNN}^*_{\mathcal{S}}$ within the range of $(0,1)$.

In this way, by incorporating all these characteristics with the discrepancy node attributes ($\mathbf{X}^i_{\text{disc}}$, the scalar node classification accuracy label y^i_{disc}, and the graph structure from meta-graph as $\mathbf{A}^i_{\text{disc}} = \mathbf{A}^i_{\text{meta}}$ for indicating discrepancy node interconnections, we could derive the final discrepancy meta-graph set, *i.e.*, DiscGraph set, as

$$\mathcal{G}_{\text{disc}} = \left\{g^i_{\text{disc}}\right\}_{i=1}^{K}, \text{where } g^i_{\text{disc}} = (\mathbf{X}^i_{\text{disc}}, \mathbf{A}^i_{\text{disc}}, y^i_{\text{disc}}). \tag{7.6}$$

The proposed DiscGraph set $\mathcal{G}_{\text{dist}}$ contains a sufficient number of graphs with diverse node context and graph structure distributions, where each discrepancy meta-graph contains the latent node embedding based discrepancy node attributes and the node class prediction based accuracy label, according to the well-trained GNN's outputs, along with diverse graph structures. All these make the proposed DiscGraph set a discriminative graph structural discrepancy representation for capturing wide-range graph data distribution discrepancies.

7.4.3 GNNEvaluator Training and Inference

Training. Given our constructed DiscGraph set $\mathcal{G}_{\text{disc}} = \{g^i_{\text{disc}}\}_{i=1}^K$ and $g^i_{\text{disc}} = (\mathbf{X}^i_{\text{disc}}, \mathbf{A}^i_{\text{disc}}, y^i_{\text{disc}})$ in Eq. (7.6), we use it to train a GNN regressor with $f_{\boldsymbol{\phi}} : g^i_{\text{disc}} \to a$ for evaluating well-trained GNNs, which we name as GNNEvaluator. Specifically, the proposed GNNEvaluator takes a two-layer GCN architecture as the backbone, followed by a pooling layer to average the representation of all nodes of each g^i_{disc}, and then maps the averaged embedding to a scalar node classification accuracy on the whole graph. The objective function of our GNN regressor can be written as:

$$\min_{\boldsymbol{\phi}} \sum_{i=1}^{K} \mathcal{L}_{\text{reg}}(f_{\boldsymbol{\phi}}(\mathbf{X}^i_{\text{disc}}, \mathbf{A}^i_{\text{disc}}), y^i_{\text{disc}}), \tag{7.7}$$

where $\mathcal{L}_{\text{reg}}$ is the regression loss, *i.e.*, the mean square error (MSE) loss.

Inference. During the inference in the practical GNN model evaluation, we have: (1) to-be-evaluated $\text{GNN}^*_{\mathcal{S}}$, and (2) the unseen test graph $\mathcal{T} = (\mathbf{X}', \mathbf{A}')$ without labels. The first thing is to calculate the discrepancy node attributes on the unseen test graph $\mathcal{T}$ towards the observed training graph $\mathcal{S}$ according to Eq. (7.4), so that we could obtain $\mathbf{X}^{\mathcal{T}}_{\text{disc}} = D(\mathbf{Z}^*_{\mathcal{T}}, \mathbf{Z}^*_{\mathcal{S}})$, where $\mathbf{Z}^*_{\mathcal{T}} = \text{GNN}^*_{\mathcal{S}}(\mathbf{X}', \mathbf{A}')$. Along with the unseen graph structure $\mathbf{A}^{\mathcal{T}}_{\text{disc}} = \mathbf{A}'$, the proposed GNNEvaluator could directly output the node classification accuracy of $\text{GNN}^*_{\mathcal{S}}$ on $\mathcal{T}$ as:

$$\text{Acc}(\mathcal{T}) = \hat{y}^{\mathcal{T}}_{\text{dist}} = f_{\boldsymbol{\phi}^*}(\mathbf{X}^{\mathcal{T}}_{\text{disc}}, \mathbf{A}^{\mathcal{T}}_{\text{disc}}), \tag{7.8}$$

where $\boldsymbol{\phi}^*$ denotes the optimal GNNEvaluator weight parameters trained by our constructed DiscGraph set $\mathcal{G}_{\text{dist}}$.

7.5 Experiments

This section empirically evaluates the proposed GNNEvaluator on real-world graph datasets for node classification task. In all experiments, to-be-evaluated GNN models have been well-trained on observed training graphs and keep FIXED in GNN model evaluation. We only vary the unseen test graph datasets and evaluate various different types and architectures of well-trained GNNs. Throughout our

proposed entire two-stage GNN model evaluation framework, we are unable to access the labels of unseen test graphs. We only utilize these labels for the purpose of obtaining true error estimates for experimental demonstration.

We investigate the following questions to verify the effectiveness of the proposed method. **Q1:** How does the proposed GNNEvaluator perform in evaluating well-trained GNNs' node classification accuracy (Sec. 7.5.2)? **Q2:** How does the proposed GNNEvaluator perform when conducting an ablation study regarding the DiscGraph set component? (Sec. 7.5.3) **Q3:** What characteristics does our constructed DiscGraph set have (Sec. 7.5.4)? **Q4:** How many DiscGraphs in the set are needed for the proposed GNNEvaluator to perform well (Sec. 7.5.5)?

7.5.1 Experimental Settings

Datasets. We perform experiments on three real-world graph datasets, *i.e.*, DBLPv8, ACMv9, and Citationv2, which are citation networks from different original sources (DBLP, ACM, and Microsoft Academic Graph, respectively) by following the processing of [196]. Each dataset shares the same label space with six categories of node class labels, including 'Database', 'Data mining', 'Artificial intelligent', 'Computer vision', 'Information Security', and 'High-performance computing'. We evaluate our proposed GNNEvaluator with the following six cases, *i.e.*, A→D, A→C, C→A, C→D, D→A, and D→C, where A, C, and D denote ACMv9, DBLPv8, and Citationv2, respectively. Each arrow denotes the estimation of GNNEvaluator trained by the former observed graph and tested on the latter unseen graph. Note that, in all stages of GNNEvaluator training and inference, we do not access the labels of the latter unseen graph.

GNN Models and Evaluation. We evaluate four commonly-used GNN models, including GCN [203], GraphSAGE [204] (*abbr.* SAGE), GAT [205], and GIN [206], as well as the baseline MLP model that is prevalent for graph learning. For each model, we train it on the training set of the observed graph under the transductive setting, until the model achieves the best node classification on its validation set following the standard training process, *i.e.*, the 'well-trained' GNN model. To minimize experimental variability, we train each type of GNN with five random seeds. That means, for instance, we have five well-trained GCN models on the same fully-observed graph with the same hyper-parameter space but only five different random seeds. We report the Mean Absolute Error (MAE), the average absolute difference between the ground truth accuracy (in percentage) on the unseen graph and the estimated accuracy from ours and baselines on the same unseen graph, across different random seeds for each evaluated GNN. The smaller MAE denotes better GNN model evaluation performance.

Baseline Methods. As our method is the first GNN model evaluation approach, there is no available baseline for direct comparisons. Therefore, we evaluate our proposed approach by comparing it to three convolutional neural network (CNN) model evaluation methods applied to image data. Note that existing CNN model evaluation methods are hard to be directly applied to GNNs, since GNNs

TABLE 7.1: Mean Absolute Error (MAE) performance of different GNN models across different random seeds. (GNNs are well-trained on the ACMv9 dataset and evaluated on the unseen and unlabeled Citationv2 and DBLPv8 datasets, *i.e.*, A→C and A→D, respectively. Best results are in bold.)

Methods	ACMv9→Citationv2						ACMv9→DBLPv8					
	GCN	SAGE	GAT	GIN	MLP	*Avg.*	GCN	SAGE	GAT	GIN	MLP	*Avg.*
ATC-MC [70]	4.49	8.40	4.37	18.40	34.33	14.00	21.96	24.20	30.30	24.06	26.62	25.43
ATC-MC-c [70]	**2.41**	5.74	4.67	22.00	51.41	17.25	31.15	30.55	30.18	29.71	45.81	33.48
ATC-NE [70]	3.97	8.02	4.28	17.35	38.87	14.50	22.93	24.78	30.50	23.74	31.13	26.62
ATC-NE-c [70]	4.44	6.09	**3.30**	23.95	44.62	16.48	34.42	28.31	27.02	30.28	39.28	31.86
Thres. ($\tau = 0.7$) [69]	32.64	35.81	33.63	50.76	35.28	37.63	9.59	12.14	14.30	32.67	39.72	21.68
Thres. ($\tau = 0.8$) [69]	26.30	29.60	26.18	49.25	35.87	33.44	**2.63**	7.44	14.47	32.20	40.31	19.41
Thres. ($\tau = 0.9$) [69]	17.56	21.34	16.38	46.53	36.08	27.58	8.20	7.42	16.07	31.47	40.56	20.74
AutoEval-G [69]	18.94	26.19	26.12	50.86	32.40	30.90	2.77	**2.54**	7.25	48.68	29.95	18.24
GNNEvaluator (Ours)	4.85	**4.11**	12.23	**10.14**	**22.20**	**10.71**	11.80	14.88	**6.36**	**13.78**	**17.49**	**12.86**

TABLE 7.2: Mean Absolute Error (MAE) performance of different GNN models across different random seeds. (GNNs are well-trained on the Citationv2 dataset and evaluated on the unseen and unlabeled ACMv9 and DBLPv8 datasets, *i.e.*, C→A and C→D, respectively.Best results are in bold.)

Methods	Citationv2→ACMv9						Citationv2→DBLPv8					
	GCN	SAGE	GAT	GIN	MLP	*Avg.*	GCN	SAGE	GAT	GIN	MLP	*Avg.*
ATC-MC [70]	9.50	13.40	8.28	35.51	43.40	22.02	22.57	**1.37**	21.87	29.24	35.20	22.05
ATC-MC-c [70]	6.93	11.75	**6.70**	38.93	57.43	24.35	33.67	4.92	28.23	30.89	52.59	30.06
ATC-NE [70]	8.86	13.04	7.87	34.88	47.49	22.42	23.97	1.86	23.74	28.96	39.72	23.65
ATC-NE-C [70]	7.73	13.94	7.63	41.17	62.96	26.69	37.16	4.66	29.43	31.66	58.95	32.37
Thres. ($\tau = 0.7$) [69]	37.33	36.61	31.68	58.91	34.33	39.77	10.70	23.05	12.74	34.60	38.29	23.88
Thres. ($\tau = 0.8$) [69]	29.62	28.95	22.77	57.48	34.53	34.67	5.65	15.01	7.61	34.36	38.43	20.21
Thres. ($\tau = 0.9$) [69]	19.59	19.06	11.37	55.72	34.56	28.06	10.65	8.28	8.07	34.00	38.44	19.89
AutoEval-G [69]	23.01	31.24	26.74	59.66	35.02	28.28	**2.57**	16.52	6.96	19.20	32.24	24.59
GNNEvaluator (Ours)	**5.45**	**8.53**	9.61	**29.77**	**28.52**	**16.38**	11.64	7.02	**5.58**	**6.46**	**22.87**	**10.71**

TABLE 7.3: Mean Absolute Error (MAE) performance of different GNN models across different random seeds. (GNNs are well-trained on the DBLPv8 dataset and evaluated on the unseen and unlabeled ACMv9 and Citationv2 datasets, *i.e.*, D→A and D→C, respectively. Best results are in bold.)

Methods	DBLPv8→ACMv9						DBLPv8→Citationv2					
	GCN	SAGE	GAT	GIN	MLP	*Avg.*	GCN	SAGE	GAT	GIN	MLP	*Avg.*
ATC-MC [70]	4.98	31.49	37.08	28.74	30.42	26.54	5.47	29.72	37.43	32.07	36.42	28.22
ATC-MC-C [70]	3.42	32.83	40.31	37.68	34.72	29.79	**4.66**	28.67	43.89	38.13	41.64	31.40
ATC-NE [70]	2.83	27.67	40.90	35.88	36.03	28.66	4.25	26.04	39.38	38.38	45.15	30.64
ATC-NE-C [70]	12.95	20.86	39.34	43.41	38.85	31.08	16.87	15.77	40.26	47.64	47.61	33.63
Thres. ($\tau = 0.7$) [69]	21.66	32.69	39.19	30.10	27.80	30.29	27.22	35.48	34.35	37.99	30.36	33.08
Thres. ($\tau = 0.8$) [69]	18.63	29.34	37.25	25.80	20.34	26.27	22.70	30.86	32.48	32.73	22.36	28.23
Thres. ($\tau = 0.9$) [69]	16.12	23.81	43.26	23.73	12.98	23.98	15.63	26.07	35.88	27.15	12.20	23.39
AutoEval-G [69]	7.28	**9.72**	12.05	14.17	22.07	13.06	21.13	15.11	5.65	12.33	31.90	17.22
GNNEvaluator (Ours)	**2.46**	10.27	**6.94**	**8.86**	**5.42**	**6.79**	11.68	**7.83**	**3.97**	**9.62**	**5.92**	**7.80**

have entirely different convolutional architectures working on different data types. Compared with Euclidean image data, graph structural data distributions have more complex and wider variations due to their non-independent node-edge interconnections. Therefore, we make necessary adaptations to enable these methods to work on graph-structured data for GNN model evaluation. Specifically, we compare: (1) *Average Thresholded Confidence (ATC) score* [70], which learns a threshold on CNN's

confidence to estimate the accuracy as the fraction of unlabeled images whose confidence scores exceed the threshold. We consider its four variants, denoting as ATC-NE, ATC-NE-c, ATC-MC, and ATC-MC-c, where ATC-NE and ATC-MC calculate negative entropy and maximum confidence scores, respectively, and '-c' denotes their confidence calibrated versions. In our adaptation, we use the training set of the observed graph under the transductive setting to calculate these confidence scores, and use the validation set for their calibrated versions. (2) *Threshold-based Method*, which is introduced by [69] and determines three threshold values for $\tau = \{0.7, 0.8, 0.9\}$ on the output softmax logits of CNNs and calculates the percentage of images in the entire dataset whose maximum entry of logits are greater than the threshold τ, which indicates these images are correctly classified. We make an adaptation by using the GNN softmax logits and calculating the percentage of nodes in a single graph. (3) *AutoEval Method* [69], which conducts the linear regression to learn the CNN classifier's accuracy based on the domain gap features between the observed training image data and unseen real-world unlabeled image data. We adapt it to GNN models on graph structural data by calculating the Maximum Mean Discrepancy (MMD) measurement as the graph domain gap features extracted from our meta-graph set, followed by the linear regression, which we name as AutoEval-G variant.

Following shows the details of all baseline methods.

Average Thresholded Confidence (ATC) & Its Variants. This metric [70] learns a threshold on CNN's confidence to estimate the accuracy as the fraction of unlabeled images whose confidence scores exceed the threshold as:

$$\text{ATC} = \frac{1}{M}\sum_{j=1}^{M} \mathbf{1}\left\{s\left(\text{Softmax}\left(f_{\boldsymbol{\theta}^*}\left(\mathbf{x}'_j\right)\right)\right) < t\right\}, \tag{7.9}$$

where $f_{\boldsymbol{\theta}^*}(\cdot)$ denotes the well-trained CNN's classifier with the optimal parameter $\boldsymbol{\theta}^*$, and $s(\cdot)$ denotes the score function working on the softmax prediction of $f_{\boldsymbol{\theta}^*}(\cdot)$. When the context is clear, we will use $f(\cdot)$ for simplification. We adopted two different score functions, deriving two variants as: (1) Maximum confidence variant ATC-MC with $s(f(\mathbf{x}'_j)) = \max_{k\in\mathcal{Y}} f_k(\mathbf{x}'_j)$; and (2) Negative Entropy variant ATC-NE with $s(f(\mathbf{x}'_j)) = \sum_k f_k(\mathbf{x}'_j)\log\left(f_k(\mathbf{x}'_j)\right)$, where $\mathcal{Y} = \{1, 2, \ldots, C\}$ is the label space. And for t in Eq. (7.9), it is calculated based on the validation set of the observed training set $(\mathbf{x}_u^{\text{val}}, \mathbf{y}_u^{\text{val}}) \in \mathcal{S}_{\text{val}}$:

$$\frac{1}{N_{\text{val}}}\sum_{u=1}^{N_{\text{val}}} \mathbf{1}\left\{s\left(\text{Softmax}\left(f_{\boldsymbol{\theta}^*}\left(\mathbf{x}_u^{\text{val}}\right)\right)\right) < t\right\} = \frac{1}{N_{\text{val}}}\sum_{u=1}^{N_{\text{val}}} \mathbf{1}\left\{p\left(\mathbf{x}_u^{\text{val}}; \boldsymbol{\theta}^*\right) \neq \mathbf{y}_u^{\text{val}}\right\}, \tag{7.10}$$

where $p(\cdot)$ denotes the predicted labels of samples. For all the calibration variants with '-c' in our experiments, they conduct standard Temperature Scaling [207] with following equations:

$$p_u = \max_k \text{Softmax}\left(\mathbf{z}_u/T\right)^{(k)}, \tag{7.11}$$

where $\mathbf{z}_u$ denotes the network output logits before softmax and T is the temperature scaling factor.

Threshold-based Method. This is an intuitive solution introduced by [69], which is not a learning-based method. It follows the basic assumption that a class prediction will likely be correct when it has a high softmax output score. Then, the threshold-based method would provide the estimated accuracy of a model as:

$$\text{ACC} = \frac{\sum_{i=1}^{M} \mathbf{1}\left\{\max\left(f_{\boldsymbol{\theta}^*}(\mathbf{x}'_j)\right) > \tau\right\}}{M}, \tag{7.12}$$

where τ is the pre-defined thresholds as $\tau = \{0.7, 0.8, 0.9\}$ on the output softmax logits of CNNs. This metric calculates the percentage of images in the entire dataset whose maximum entry of logits are greater than the threshold τ, which indicates these images are correctly classified.

AutoEval & Our Adaption AutoEval-G. This is a learning-based method of estimating classifier accuracy by utilizing dataset-level feature statistics [69]. AutoEval synthesizes a meta image dataset $\mathcal{D}$ (a dataset comprised of many datasets) from the observed training dataset $\mathcal{D}_{\text{ori}}$, and conducts the linear regression to learn the CNN classifier's accuracy based on the dataset-level feature statistics between the observed training image data and unseen real-world unlabeled image data. The accuracy linear regression with the dataset statistic feature f_{linear} is written as:

$$\begin{aligned} f_{\text{linear}} &= \text{FD}\left(\mathcal{D}_{\text{ori}}, \mathcal{D}\right) = \|\boldsymbol{\mu}_{\text{ori}} - \boldsymbol{\mu}\|_2^2 + \text{Tr}\left(\boldsymbol{\Sigma}_{\text{ori}} + \boldsymbol{\Sigma} - 2\left(\boldsymbol{\Sigma}_{\text{ori}}\,\boldsymbol{\Sigma}\right)^{\frac{1}{2}}\right), \\ \text{ACC} &= w_1 f_{\text{linear}} + w_0 \end{aligned} \tag{7.13}$$

where FD is the Fréchet distance [208] to measure the domain gap with the mean feature vectors $\boldsymbol{\mu}_{\text{ori}}$ and $\boldsymbol{\mu}$, the covariance matrices $\boldsymbol{\Sigma}_{\text{ori}}$ and $\boldsymbol{\Sigma}$ of $\mathcal{D}_{\text{ori}}$ and $\mathcal{D}$, respectively. And $\text{Tr}(\cdot)$ denotes the trace of the matrix, w_1 and w_0 denote the parameters of linear regression.

We make the following necessary adaptions to expand the AutoEval method to graph structural data on GNN model evaluation, deriving **AutoEval-G** by: (1) we synthesize a meta-graph set $\mathcal{G}_{\text{meta}}$ as demonstrated in Sec. 3.2 of the main manuscript; (2) we calculate the Maximum Mean Discrepancy (MMD) distance as AutoEval-G's graph dataset discrepancy representation f'_{linear}, which measures the graph distribution gap between the observed training graph $\mathcal{S}$ and each meta-graph $g_{\text{meta}} \in \mathcal{G}_{\text{meta}}$ with their node embeddings from the well-trained $\text{GNN}_{\mathcal{S}}$ as:

$$f'_{\text{linear}} = \text{MMD}(\mathcal{S}, g_{\text{meta}}) = \text{MMD}(\mathbf{Z}^*_{\mathcal{S}}, \mathbf{Z}^{(i,*)}_{\text{meta}}). \tag{7.14}$$

where $\mathbf{Z}^*_{\mathcal{S}} = \text{GNN}^*_{\mathcal{S}}(\mathbf{X}, \mathbf{A})$, and $\mathbf{Z}^{(i,*)}_{\text{meta}} = \text{GNN}^*_{\mathcal{S}}(\mathbf{X}^i_{\text{meta}}, \mathbf{A}^i_{\text{meta}})$.

7.5.2 GNN Model Evaluation Results

In Table 7.1, Table 7.2, and Table 7.3, we report MAE results of evaluating different GNN models across different random seeds. In general, we observe that our proposed GNNEvaluator significantly

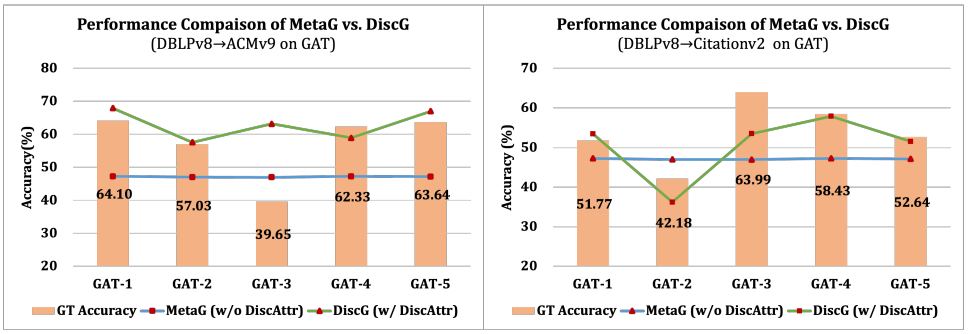

FIGURE 7.3: Ablation study of the proposed DiscGraph set with discrepancy node attributes (w/ DiscAttr) in green lines *vs.* the meta-graph set without discrepancy node attributes (w/o DiscAttr) in blue lines. And the ground-truth node classification accuracy is in histograms for reference.

outperforms the other model evaluation baseline methods in all six cases, achieving the lowest average MAE scores of all GNN model types, *i.e.*, 10.71 and 12.86 in A→C and A→D cases, 16.38 and 10.71 in C→A and C→D cases, and 6.79 and 7.80 in D→A and D→C cases, respectively.

More specifically, we observe that some baseline methods achieve the lowest MAE for a certain GNN model type on a specific case. Taking GCN as an example in Table 7.1, ATC-MC-c performs best with 2.41 MAE under A→C case. However, it correspondingly has 31.15 worst MAE under A→D case. That means, for a GCN model that is well-trained by ACMv9 dataset, ATC-MC-c provides model evaluation results with significant performance variance under different unseen graphs. Such high-variance evaluation performance would significantly limit the practical applications of ATC-MC-c, as it only performs well on certain unseen graphs, incurring more uncertainty in GNN model evaluation. Similar limitations can also be observed in Table 7.2 for ATC-MC-c on evaluating GAT model, with 6.70 lowest MAE in C→A but 28.23 high MAE in C→D. Moreover, in terms of AutoEval-G method on GraphSAGE model evaluation in Table 7.3, it achieves the lowest MAE of 9.72 in D→A case, but a high MAE of 15.11 in C→A case. In contrast, our proposed GNNEvaluator has smaller MAE variance for different unseen graphs under the same well-trained models. For instance, our GNNEvaluator achieves 4.85 MAE on GCN under A→C case, and 11.80 MAE on GCN in A→D case, which is better than ATC-MC-c with 2.41 and 31.15 on the same cases. All these results verify the effectiveness and consistently good performance of our proposed method on diverse unseen graph distributions for evaluating different GNN models.

7.5.3 Ablation Study

We conduct an ablation study to verify the effectiveness of the proposed DiscGraph set component for our proposed GNN model evaluation pipeline. Concretely, we replace the DiscGraph set with the meta-graph set for GNNEvaluator training, which does not involve discrepancy node attributes calculated based on specific well-trained GNNs' node embedding space with the discrepancy measurement function. The output node classification accuracy (%) of the proposed GNNEvaluator for evaluating

five GAT models under different seeds are shown in Fig. 7.3 with lines for the proposed DiscGraph set (w/ DiscAttr) in green and the meta-graph set (w/o DiscAttr) in blue, respectively. And we also list the ground-truth node classification accuracy in histograms for comparison.

Generally, we observe that the proposed DiscGraph set (w/ DiscAttr) trained GNNEvaluator has better performance for GAT model evaluation than that trained by the meta-graph set (w/o DiscAttr), as all green lines are closer to the ground-truth histograms than the blue line, reflecting the effectiveness of the proposed the proposed DiscGraph set, especially the discrepancy node attributes with the discrepancy measurement function. Moreover, the meta-graph set (w/o DiscAttr) trained GNNEvaluator almost produces the same accuracy on a certain GNN type, *i.e.*, its performance stays the same on all five GATs, making it fail to evaluate a certain GNN model type under different optimal model parameters. That is because, compared with the proposed DiscGraph set, the meta-graph set lacks the exploration of the latent node embedding space of well-trained GNNs and only utilizes the output node class predictions from the well-trained GNNs. In contrast, the proposed DiscGraph set fully exploits both latent node embedding space and output node class predictions of well-trained GNNs for comprehensively modeling the discrepancy between the observed training graph and unseen test graph, enabling its superior ability for GNN model evaluation.

7.5.4 Analysis of Discrepancy Meta-graph Set

We visualize the discrepancy node attributes in the proposed DiscGraph set on GCN model evaluation trained on ACMv9 and tested on unseen DBLPv8 in Fig. 7.4 (a). Taking this as a reference, we present visualizations of discrepancy node attributes derived between ACMv9 and arbitrary two meta-graphs from our created meta-graph set in Fig. 7.4 (c) and (d). Darker color close to black indicates a larger discrepancy in the output node embedding space of a well-trained GNN, and lighter color close to white indicates a smaller discrepancy. As can be observed, the discrepancy pairs of (ACMv9, unseen DBLPv8), (ACMv9, Meta-graph-1), and (ACMv9, Meta-graph-2) generally show similar heat map patterns in terms of discrepancy. This indicates the effectiveness of the proposed meta-graph set simulation strategy, which could capture potential unseen graph data distributions by synthesizing diverse graph data with various graph augmentations.

7.5.5 Hyper-parameter Analysis of GNNEvaluator

We test the effects of different numbers of discrepancy meta-graphs (K) for GNNEvaluator training in Fig. 7.5. Concretely, we utilize the number of DiscGraphs for $K = [100, 200, 300, 400]$ to train the proposed GNNEvaluator for evaluating five GCN models that are well-trained on Citationv2 dataset and make inferences on unseen ACMv9 and DBLPv8 datasets. Considering the average performance under two transfer cases reflected in the line, despite $K = 100$ having a higher MAE, the average MAE over two cases does not change much with the number K for training the proposed

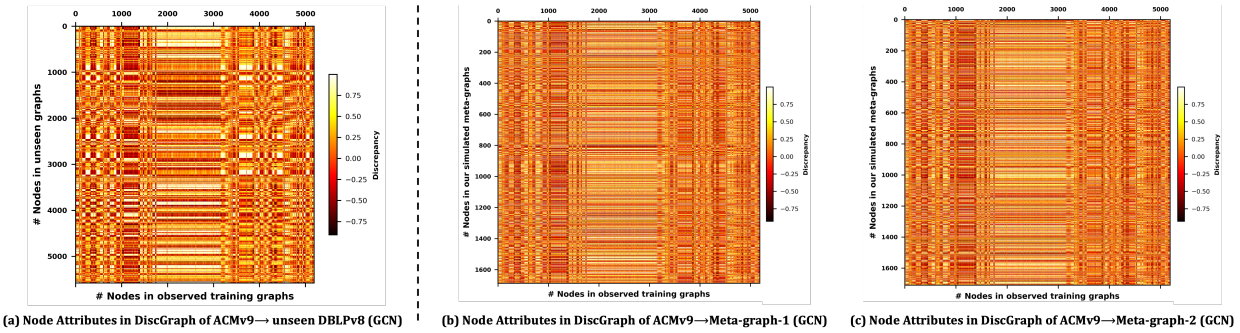

FIGURE 7.4: Visualizations on discrepancy node attributes in the proposed DiscGraph set for GCN model evaluation. Darker color denotes a larger discrepancy.

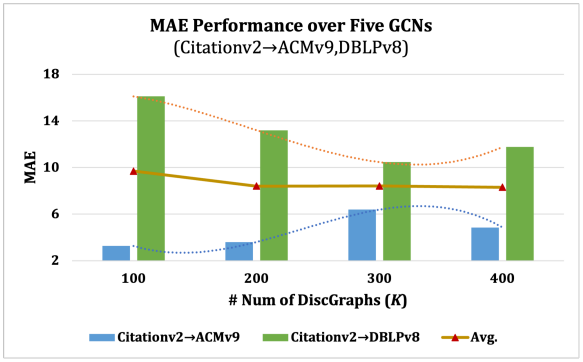

FIGURE 7.5: Effects of different numbers of DiscGraphs (K) for GNNEvaluator training. Histograms denote MAE for $C \rightarrow A$ and $C \rightarrow D$ cases over five GCNs under different K, and the line indicates the average MAE of such two evaluation cases.

GNNEvaluator. This indicates that the proposed GNNEvaluator can effectively learn to accurately predict node attributes by training on an appropriate number of DiscGraphs, without the need for a significantly excessive amount of training examples.

7.6 Conclusion

In this chapter, we have studied a new problem, GNN model evaluation, for understanding and evaluating the performance of well-trained GNNs on unseen and unlabeled graphs. A two-stage approach is proposed, which first generates a diverse meta-graph set to simulate and capture the discrepancies of different graph distributions, based on which a GNNEvaluator is trained to predict the accuracy of a well-trained GNN model on unseen graphs. Extensive experiments with real-world unseen and unlabeled graphs could verify the effectiveness of our proposed method for GNN model evaluation. Our method assumes that the class label space is unchanged across training and testing graphs though covariate shifts may exist between the two. We will look into relaxing this assumption and address a broader range of natural real-world graph data distribution shifts in the future.

Chapter 8

Future Directions

Exploring and constructing the automated graph machine learning operations (MLOps) workflow could cover a wide range of aspects for designing and serving productive graph ML with GNNs in real-world graph learning scenarios, going beyond the primary focus areas of (1) graph data engineering, (2) automated GNN model design, and (3) GNN model deployment in this research. Hence, in this chapter, we shed light on potential future research directions of the Auto-GMLOps workflow.

8.1 In-depth Exploration, Understanding, and Management of Various Graph Data Types and Characteristics

From the perspective of graph data engineering, this research advances beyond the confines of simple single-relational graph data by delving into the intricacies of multi-relational and heterophilic graph characteristics. Yet, the spectrum of real-world graph data extends even further, encompassing a vast array of diverse and complex properties such as dynamic, spatial-temporal, and heterogeneous elements. One such example is dynamic graphs, where features and structures evolve over time, giving rise to their temporal nature.

Therefore, to unlock the full potential of GNNs for modeling real-world graph data and to leverage these advancements in practical graph learning applications, it is essential to undertake a thorough exploration, understanding, and management of the extensive varieties of graph data types and their unique characteristics. Future studies should focus on the following aspects in the realm of effective graph data engineering:

- **Exploration:** (1) Handling graph data with naturally unique characteristics in terms of complexity and diversity, such as capturing and modeling temporal dynamics and heterogeneity (2) Handling graph data with characteristics that could hurdle the expressiveness of GNNs,

including over-sparse graphs, over-large-scale graphs, and partially labeled graphs with missing information or noises. (3) Incorporating multi-modal data, for instance, combining texts and images with graph data, to benefit enriched graph or multi-modal data representations, facilitating comprehensive multi-aspect data modeling and learning. .

- **Understanding:** (1) Leveraging various graph data visualization tools to transfer high-dimensional and complexly connected graph data to low-dimensional or well-presented visualization representations, for capturing latent interconnections and dependencies among nodes and their relationships. (2) Understanding the value of certain graph data in the realm of data valuation, and estimating how such graph data contributes to the final performance, facilitating the design of graph data marketplaces and explainable AI.

- **Management:** (1) Implementing privacy-preserving techniques, such as differential privacy or federated learning, for dealing with sensitive graph data. (2) Enabling graph data version control and graph database integration, to manage versions of graph datasets and learning tasks, and efficiently query and manage graph data at scale, ensuring reproducibility and traceability.

8.2 Good Generalization and Transfer Learning Abilities of Both Graph Data and Models

Real-world graph data typically has dynamic and evolving characteristics over time. For example, in financial networks, the relationships between entities (represented by graph nodes, like stocks, commodities, or institutions) change due to market trends, economic policies, or global events. As a result, a graph learning model trained on historical market data might underperform when serving on real-time new-coming financial networks. That means the distribution shift and domain discrepancy of graph data might encompass all aspects of graph data and all procedures of graph model development. Consequently, graph learning models encounter difficulties in achieving good generalization to previously unseen graph data distributions during the testing phase.

In light of this, a potential future research direction is to enable good generalization and transfer learning abilities, from both views of graph data and graph learning models.

Concretely, from the perspective of graph data, exploiting the distinct graph distribution characteristics is still an open research problem. The study on graph data distributions should span all aspects of graph components, including node features, graph structures, and label distributions, and emphasize the invariance, stability, evolution, etc.

Moreover, from the perspective of GNN models, investigating how GNNs trained on a certain graph data domain (*e.g.*, social networks) can be adapted or generalized to different other graph data domains (*e.g.*, molecular structures) is another important aspect. This process should guarantee

minimum performance degradation and effective cross-domain generalization, with explorations on transfer learning or meta-learning strategies for GNNs to improve their ability to adapt to new graph data and tasks more effectively.

8.3 Automated and Explainable GNN Models with Robustness, Fairness

Despite efforts have been made to automate GNN model design on heterophilic and multi-relational graphs in this research, how to automatically tailor GNN architectures and learning processes for specific practical application scenarios, such as drug discovery and biochemical molecules, and financial transactions. to achieve state-of-the-art performance, is a potential future research direction. Based on this, incorporating explainability robustness, and fairness as key objectives in the automated design process of GNNs to safeguard against adversarial attacks and bias, is another promising future direction, so that it can be ensured that GNN models perform reliably across varied and potentially adversarial environments.

Explainable GNNs: In certain sensitive areas, *e.g.*, healthcare, the GNN models require the explainable property, so that comprehensible model outputs could build trust and facilitate more informed decision-making.

Fairness-aware Modeling: Designing GNN architectures and training processes that explicitly account for fairness. This could involve modifying learning objectives to penalize unfair predictions or employing post-hoc adjustments to correct for detected biases.

Robustness Modeling: Investigating the robustness of GNNs against adversarial attacks, noisy and low-quality graphs, and developing techniques to defend against customized attacks and unsatisfied graph quality, could ensure the security, effectiveness, and reliability of automated GNNs in real-world applications.

8.4 Continuous Evaluation, Integration, Deployment, and Monitor of GNN Models

Designing a systematic and comprehensive continuous Auto-GMLOps workflow represents a significant challenge and opportunity in the field of graph machine learning. In this thesis, we proposes a framework comprising general stages that are integral to this workflow, emphasizing the automation and systematic of GNN development and deployment. Notably, in the stage of GNN deployment, we focus primarily on the 'GNN evaluation problem' in this research. This area is ripe for exploration, presenting fresh and important research questions for future investigation, such as:

Continuous Evaluation and Integration: (1) Developing and implementing dynamic GNN model evaluation metrics that adapt to evolving graph data characteristics and application requirements in a continuous pattern; (2) Integrating automated testing frameworks to validate the functionality and performance of GNN models upon every graph database, code, model update, ensuring new changes do not degrade the model's performance.

Continuous Deployment and Monitor: (1) Establishing automated GNN deployment pipelines that can seamlessly transition GNN models from development to production environment, covering efficient GNN model management, dependency management of different workflow stages, and deployment to scalable real-world graph machine learning tasks. (2) Continuously monitoring the performance of deployed GNN models in real-time, capturing metrics that can indicate degradation or other issues (*e.g.*, graph data distribution shifts), making sure the rapid response when GNN performance drops on certain practical uses.

Chapter 9

Conclusion

This thesis comprehensively explores the systematic workflow of graph machine learning operations (MLOps) in an automated paradigm, named Auto-GMLOps workflow. It works on three core stages for practical and production-ready GNN development and deployment, including (1) graph data engineering, (2) automated GNN model design, and (3) GNN model deployment.

For the first stage of graph data engineering, this thesis considers two aspects: graph data types and graph data scales, respectively. For one thing, this thesis explores the characteristics of various graph types, *i.e.*, heterophilic graphs and multi-relational graphs, to better facilitate automated GNN model design. For another thing, this thesis reduces the graph data scales with the graph condensation strategy for alleviating the storage and computation costs in industrial applications, to advance and accelerate the automated GNN model development progress.

For the second stage of automated GNN model design, this thesis works on fine-grained search space design and specific search strategy development, within the context of graph neural architecture search. In this way, different graph data characteristics, specifically driven by the incorporated characteristics of heterophilic graphs and multi-relational graphs, are comprehensively incorporated into the full stages of the graph NAS, leading to the expressive automated GNN development.

For the last stage of GNN model deployment, this thesis exploits and estimates well-designed and well-trained GNNs' performance, when they are deployed and served in practical real-world test graphs that they never encounter in the training stage. It fully explores the GNN generalization when there exist potential graph data distribution shifts between the training and test graphs, and proposes a GNN model evaluator to precisely estimate the well-trained GNN performance in online service.

In summary, this thesis works on all the phases of building a comprehensive and systematic automated graph MLOps workflow, facilitating the analysis and understanding of diverse and complex graph data, as well as benefiting the development and deployment of expressive automated GNNs in a wide range of real-world graph learning scenarios.

Bibliography

[1] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[2] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607. Springer, 2018.

[3] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3950–3957, 2021.

[4] 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.

[5] He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. Trustworthy graph neural networks: Aspects, methods and trends. *arXiv preprint arXiv:2205.07424*, 2022.

[6] Neptune.ai. Graph neural network and some of gnn applications, Sep 2023. URL https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications.

[7] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1025–1035, 2017.

[8] 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.

[9] Yongqi Zhang, Quanming Yao, and Lei Chen. Interstellar: Searching recurrent architecture for knowledge graph embedding. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeurIPS*, 2020.

[10] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. Relational graph neural network with hierarchical attention for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9612–9619, 2020.

[11] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pages 10362–10383. PMLR, 2022.

[12] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International conference on machine learning*, pages 7192–7203. PMLR, 2021.

[13] Huan Yee Koh, Anh TN Nguyen, Shirui Pan, Lauren T May, and Geoffrey I Webb. Physicochemical graph neural network for learning protein–ligand interaction fingerprints from sequence data. *Nature Machine Intelligence*, pages 1–15, 2024.

[14] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD*, pages 974–983, 2018.

[15] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems*, 1 (1):1–51, 2023.

[16] 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.

[17] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2019.

[18] Valerio Ciotti, Moreno Bonaventura, Vincenzo Nicosia, Pietro Panzarasa, and Vito Latora. Homophily and missing links in citation networks. *EPJ Data Science*, 5:1–14, 2016.

[19] Yixin Liu, Yizhen Zheng, Daokun Zhang, Vincent Lee, and Shirui Pan. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *Proceedings of the Association for the Advanced of Artificial Intelligence (AAAI)*, 2023.

[20] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29. PMLR, 2019.

[21] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

[22] Xin Zheng, Miao Zhang, Chunyang Chen, Chaojie Li, Chuan Zhou, and Shirui Pan. Multi-relational graph neural architecture search with fine-grained message passing. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 783–792. IEEE, 2022.

[23] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[24] Yizhen Zheng, Shirui Pan, Vincent Cs Lee, Yu Zheng, and Philip S Yu. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[25] Xin Zheng, Miao Zhang, Chunyang Chen, Qin Zhang, Chuan Zhou, and Shirui Pan. Autoheg: Automated graph neural network on heterophilic graphs. In *Proceedings of the ACM Web Conference (WWW)*, pages 611–620, 2023.

[26] Xin Zheng, Miao Zhang, Chunyang Chen, Chaojie Li, Chuan Zhou, and Shirui Pan. Multi-relational graph neural architecture search with fine-grained message passing. In *IEEE International Conference on Data Mining (ICDM)*, 2022.

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[28] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI*, volume 33, pages 3060–3067, 2019.

[29] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33, 2020.

[30] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[31] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[32] Tao Wang, Rui Wang, Di Jin, Dongxiao He, and Yuxiao Huang. Powerful graph convolutioal networks with adaptive propagation mechanism for homophily and heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[33] Dongxiao He, Chundong Liang, Huixin Liu, Mingxiang Wen, Pengfei Jiao, and Zhiyong Feng. Block modeling-guided graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[34] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv preprint arXiv:2109.05641*, 2021.

[35] Liang Yang, Mengzhe Li, Liyang Liu, Chuan Wang, Xiaochun Cao, Yuanfang Guo, et al. Diverse message passing for attribute with heterophily. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[36] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*, 2021.

[37] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1541–1551, 2021.

[38] 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.

[39] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2020.

[40] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *IJCAI*, volume 20, pages 1403–1409, 2020.

[41] ZHAO Huan, YAO Quanming, and TU Weiwei. Search to aggregate neighborhood for graph neural network. In *ICDE*, pages 552–563. IEEE, 2021.

[42] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2018.

[43] Zhili Wang, Shimin Di, and Lei Chen. Autogel: An automated graph neural network with explicit link information. *NeurIPS*, 34, 2021.

[44] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303, 2019.

[45] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 13836–13845, 2020.

[46] Minghao Chen, Jianlong Fu, and Haibin Ling. One-shot neural ensemble architecture search by diversity-guided search space shrinking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16530–16539, 2021.

[47] Yiming Hu, Yuding Liang, Zichao Guo, Ruosi Wan, Xiangyu Zhang, Yichen Wei, Qingyi Gu, and Jian Sun. Angle-based search space shrinking for neural architecture search. In *European Conference on Computer Vision*, pages 119–134. Springer, 2020.

[48] Xin Xia, Xuefeng Xiao, Xing Wang, and Min Zheng. Progressive automatic design of search space for one-shot neural architecture search. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2455–2464, 2022.

[49] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.

[50] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

[51] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representation*, 2021.

[52] Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Kompella, Zhangyang Wang, et al. Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

[53] Chonghao Chen, Fei Cai, Wanyu Chen, Jianming Zheng, Xin Zhang, and Aimin Luo. Bp-moe: Behavior pattern-aware mixture-of-experts for temporal graph representation learning. *Knowledge-Based Systems*, page 112056, 2024.

[54] Zheyuan Liu, Chunhui Zhang, Yijun Tian, Erchi Zhang, Chao Huang, Yanfang Ye, and Chuxu Zhang. Fair graph representation learning via diverse mixture-of-experts. In *Proceedings of the ACM Web Conference 2023*, pages 28–38, 2023.

[55] Pavel Rumiantsev and Mark Coates. Graph knowledge distillation to mixture of experts. *arXiv preprint arXiv:2406.11919*, 2024.

[56] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations (ICLR)*, 2019.

[57] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings*

of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 257–266, 2019.

[58] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2001–2010, 2017.

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

[60] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: Theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013.

[61] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning (ICML)*, pages 1695–1706. PMLR, 2021.

[62] Chen Cai, Dingkang Wang, and Yusu Wang. Graph coarsening with neural networks. In *International Conference on Learning Representations(ICLR)*, 2020.

[63] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 66–74, 2020.

[64] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[65] Mucong Ding, Xiaoyu Liu, Tahseen Rabbani, and Furong Huang. Faster hyperparameter search on graphs via calibrated dataset condensation. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

[66] 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 (SIGKDD)*, pages 720–730, 2022.

[67] Shiye Lei and Dacheng Tao. A comprehensive survey to dataset distillation. *arXiv preprint arXiv:2301.05603*, 2023.

[68] Noveen Sachdeva and Julian McAuley. Data distillation: A survey. *arXiv preprint arXiv:2301.04272*, 2023.

[69] Weijian Deng and Liang Zheng. Are labels always necessary for classifier accuracy evaluation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15069–15078, 2021.

[70] Saurabh Garg, Sivaraman Balakrishnan, Zachary Chase Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. In *International Conference on Learning Representations (ICLR)*, 2022.

[71] Yaodong Yu, Zitong Yang, Alexander Wei, Yi Ma, and Jacob Steinhardt. Predicting out-of-distribution error with the projection norm. In *International Conference on Machine Learning (ICML)*, pages 25721–25746. PMLR, 2022.

[72] Weijian Deng, Stephen Gould, and Liang Zheng. On the strong correlation between model invariance and generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[73] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1134–1144, 2021.

[74] Weijian Deng, Stephen Gould, and Liang Zheng. What does rotation prediction tell us about classifier accuracy under varying testing environments? In *International Conference on Machine Learning (ICML)*, pages 2579–2589. PMLR, 2021.

[75] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery & data mining*, pages 807–816, 2009.

[76] Hao Peng, Ruitong Zhang, Shaoning Li, Yuwei Cao, Shirui Pan, and Philip Yu. Reinforced, incremental and cross-lingual event detection from social messages. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[77] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 753–763, 2020.

[78] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. In *Advances in Neural Information Processing Systems*, pages 5711–5722, 2019.

[79] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

[80] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[81] 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.

[82] Shichao Zhu, Shirui Pan, Chuan Zhou, Jia Wu, Yanan Cao, and Bin Wang. Graph geometry interaction learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 7548–7558, 2020.

[83] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the 31st international conference on World Wide Web*, 2022.

[84] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210, 2007.

[85] Jiong Zhu, Ryan A Rossi, Anup B Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[86] Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. New benchmarks for learning on non-homophilous graphs. In *Workshop on Graph Learning Benchmarks, WebConf*, 2021.

[87] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[88] Yu Wang and Tyler Derr. Tree decomposed graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2040–2049, 2021.

[89] M. Liu, Z. Wang, and S. Ji. Non-local graph neural networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2021.

[90] Tianmeng Yang, Yujing Wang, Zhihan Yue, Yaming Yang, Yunhai Tong, and Jing Bai. Graph pointer neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[91] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 148–156, 2021.

[92] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[93] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[94] 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.

[95] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*, 32:9240, 2019.

[96] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in Neural Information Processing Systems*, 33, 2020.

[97] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, Lifang He, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

[98] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022.

[99] Ming Jin, Yu Zheng, Yuan-Fang Li, Siheng Chen, Bin Yang, and Shirui Pan. Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022.

[100] Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, Shirui Pan, Vincent S. Tseng, Yu Zheng, Lei Chen, and Hui Xiong. Large models for time series and spatio-temporal data: A survey and outlook. *arXiv preprint arXiv:2310.10196*, 2023.

[101] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[102] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Normalizing flow-based neural process for few-shot knowledge graph completion. In *The 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

[103] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arxiv:2310.01061*, 2023.

[104] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arxiv:2306.08302*, 2023.

[105] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022.

[106] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 33(6):2378–2392, 2021.

[107] Yixin Liu, Kaize Ding, Jianling Wang, Vincent Lee, Huan Liu, and Shirui Pan. Learning strong graph neural networks with weak information. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2023.

[108] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

[109] Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. Graph attention multi-layer perceptron. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 4560–4570, 2022.

[110] Davide Buffelli, Pietro Lio, and Fabio Vandin. Sizeshiftreg: a regularization method for improving size-generalization in graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[111] Zhe Xu, Boxin Du, and Hanghang Tong. Graph sanitation with application to node classification. In *Proceedings of the ACM Web Conference (WWW)*, pages 1136–1147, 2022.

[112] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. In *International Conference on Learning Representations (ICLR)*, 2021.

[113] Miao Zhang, Shirui Pan, Xiaojun Chang, Steven Su, Jilin Hu, Gholamreza Reza Haffari, and Bin Yang. Balenas: Differentiable architecture search via the bayesian learning rule. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11871–11880, 2022.

[114] Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. idarts: Differentiable architecture search with stochastic implicit gradients. In *International Conference on Machine Learning (ICML)*, pages 12557–12566. PMLR, 2021.

[115] Xin Zheng, Yixin Liu, Zhifeng Bao, Meng Fang, Xia Hu, Alan Wee-Chung Liew, and Shirui Pan. Towards data-centric graph machine learning: Review and outlook. *arXiv preprint arXiv:2309.10979*, 2023.

[116] He Zhang, Xingliang Yuan, Quoc Viet Hung Nguyen, and Shirui Pan. On the interaction between node fairness and edge privacy in graph neural networks. *arXiv preprint arXiv:2301.12951*, 2023.

[117] He Zhang, Bang Wu, Shuo Wang, Xiangwen Yang, Minhui Xue, Shirui Pan, and Xingliang Yuan. Demystifying uneven vulnerability of link stealing attacks against graph neural networks. In *International Conference on Machine Learning*, pages 41737–41752. PMLR, 2023.

[118] He Zhang, Xingliang Yuan, Chuan Zhou, and Shirui Pan. Projective ranking-based gnn evasion attacks. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022.

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

[120] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations (ICLR)*, 2020.

[121] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations (ICLR)*, 2020.

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

[123] 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 (CVPR)*, pages 4750–4759, 2022.

[124] 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 (NeurIPS)*, 32, 2019.

[125] Wei Huang, Yayong Li, Richard Xu, Jie Yin, Ling Chen, Miao Zhang, et al. Towards deepening graph neural networks: A gntk-based optimization perspective. In *International Conference on Learning Representations (ICLR)*, 2021.

[126] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[127] Roman Novak, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Fast finite width neural tangent kernel. In *International Conference on Machine Learning (ICML)*, pages 17018–17044. PMLR, 2022.

[128] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:5186–5198, 2021.

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

[130] 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 (NeurIPS)*, 33:22118–22133, 2020.

[131] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pages 675–684, 2021.

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

[133] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[134] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2018.

[135] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[136] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*, pages 6861–6871. PMLR, 2019.

[137] Yijian Qin, Xin Wang, Zeyang Zhang, and Wenwu Zhu. Graph differentiable architecture search with structure learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 16860–16872, 2021.

[138] Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. Pre-training of graph augmented transformers for medication recommendation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5953–5959, 2019.

[139] Yichao Du, Pengfei Luo, Xudong Hong, Tong Xu, Zhe Zhang, Chao Ren, Yi Zheng, and Enhong Chen. Inheritance-guided hierarchical assignment for clinical automatic diagnosis. In *International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 461–477. Springer, 2021.

[140] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference (WWW)*, pages 673–683, 2020.

[141] Viresh Gupta and Tanmoy Chakraborty. Viking: Adversarial attack on network embeddings via supervised network poisoning. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 103–115. Springer, 2021.

[142] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[143] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

[144] Falih Gozi Febrinanto, Feng Xia, Kristen Moore, Chandra Thapa, and Charu Aggarwal. Graph lifelong learning: A survey. *IEEE Computational Intelligence Magazine*, 18(1):32–51, 2023.

[145] Qiao Yuan, Sheng-Uei Guan, Pin Ni, Tianlun Luo, Ka Lok Man, Prudence Wong, and Victor Chang. Continual graph learning: A survey. *arXiv preprint arXiv:2301.12230*, 2023.

[146] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the Association for the Advanced of Artificial Intelligence (AAAI)*, volume 35, pages 8653–8661, 2021.

[147] Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the Association for the Advanced of Artificial Intelligence (AAAI)*, volume 35, pages 4714–4722, 2021.

[148] Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[149] Andrea Rosasco, Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Distilled replay: Overcoming forgetting through synthetic samples. In *Continual Semi-Supervised Learning: First International Workshop (CSSL)*, pages 104–117. Springer, 2022.

[150] Mattia Sangermano, Antonio Carta, Andrea Cossu, and Davide Bacciu. Sample condensation in online continual learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2022.

[151] Felix Wiewel and Bin Yang. Condensed composite memory continual learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[152] Shirui Pan, Yizhen Zheng, and Yixin Liu. Integrating graphs with large language models: Methods and prospects. *arXiv preprint arXiv:2310.05499*, 2023.

[153] Zheyi Pan, Songyu Ke, Xiaodu Yang, Yuxuan Liang, Yong Yu, Junbo Zhang, and Yu Zheng. Autostg: Neural architecture search for predictions of spatio-temporal graph. In *Proceedings of the Web Conference 2021*, pages 1846–1855, 2021.

[154] Yuhui Ding, Quanming Yao, Huan Zhao, and Tong Zhang. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 279–288, 2021.

[155] Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. Autosf: Searching scoring functions for knowledge graph embedding. In *2020 IEEE 36th International Conference on Data Engineering*, pages 433–444. IEEE, 2020.

[156] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184*, 2019.

[157] Huan Zhao, Lanning Wei, and Quanming Yao. Simplifying architecture search for graph neural network. In Stefan Conrad and Ilaria Tiddi, editors, *Proceedings of the CIKM 2020 Workshops co-located with 29th ACM International Conference on Information and Knowledge Management*, volume 2699, 2020.

[158] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019.

[159] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

[160] Alberto P García-Plaza, Víctor Fresno, Raquel Martínez Unanue, and Arkaitz Zubiaga. Using fuzzy logic to leverage html markup for web page representation. *IEEE Transactions on Fuzzy Systems*, 25(4):919–933, 2016.

[161] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[162] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[163] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, page 1, 2012.

[164] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *International Conference on Learning Representations Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[165] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *Advances in Neural Information Processing Systems*, 2022.

[166] He Zhang, Bang Wu, Xiangwen Yang, Chuan Zhou, Shuo Wang, Xingliang Yuan, and Shirui Pan. Projective ranking: A transferable evasion attack method on graph neural networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3617–3621, 2021.

[167] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE TNNLS*, 2021.

[168] Shaofei Cai, Liang Li, Jincan Deng, Beichen Zhang, Zheng-Jun Zha, Li Su, and Qingming Huang. Rethinking graph neural architecture search from message-passing. In *CVPR*, pages 6657–6666, 2021.

[169] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[170] Petar Ristoski, Gerben Klaas Dirk De Vries, and Heiko Paulheim. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *ISWC*, pages 186–194. Springer, 2016.

[171] Heiko Paulheim and Johannes Fümkranz. Unsupervised generation of data mining features from linked open data. In *WIMS*, pages 1–12, 2012.

[172] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *ISWC*, pages 498–514. Springer, 2016.

[173] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *NeurIPS*, 26, 2013.

[174] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[175] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

[176] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080. PMLR, 2016.

[177] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.

[178] Tu Dinh Nguyen Dai Quoc Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of NAACL-HLT*, pages 327–333, 2018.

[179] Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. A vectorized relational graph convolutional network for multi-relational network alignment. In *IJCAI*, pages 4135–4141, 2019.

[180] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *CVSCWorkshop*, pages 57–66, 2015.

[181] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM TKDD*, 15(2): 1–49, 2021.

[182] Xiaotian Jiang, Quan Wang, and Bin Wang. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 978–987, 2019.

[183] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2018.

[184] Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. In *Advances in Neural Information Processing Systems*, 2023.

[185] Mayee Chen, Karan Goel, Nimit S Sohoni, Fait Poms, Kayvon Fatahalian, and Christopher Ré. Mandoline: Model evaluation under distribution shift. In *International Conference on Machine Learning (ICML)*, pages 1617–1629. PMLR, 2021.

[186] Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing generalization of sgd via disagreement. In *International Conference on Learning Representations (ICLR)*, 2022.

[187] Yuzhe Lu, Zhenlin Wang, Runtian Zhai, Soheil Kolouri, Joseph Campbell, and Katia P Sycara. Predicting out-of-distribution error with confidence optimal transport. In *ICLR 2023 Workshop on Pitfalls of Limited Data and Computation for Trustworthy ML*, 2023.

[188] Real-time fraud detection with gnns. https://github.com/awslabs/realtime-fraud-detection-with-gnn-on-dgl.

[189] Lukas Galke, Iacopo Vagliano, and Ansgar Scherp. Can graph neural networks go 'online'? an analysis of pretraining and inference. In *Representation Learning on Graphs and Manifolds: ICLR 2019 Workshop*, 2019.

[190] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations (ICLR)*, 2022.

[191] Yizhen Zheng, He Zhang, Vincent Lee, Yu Zheng, Xiao Wang, and Shirui Pan. Finding the missing-half: Graph complementary learning for homophily-prone and heterophily-prone graphs. *International Conference of Machine Learning (ICML)*, 2023.

[192] Yizhen Zheng, Vincent CS Lee, Zonghan Wu, and Shirui Pan. Heterogeneous graph attention network for small and medium-sized enterprises bankruptcy prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 140–151. Springer, 2021.

[193] How AWS uses graph neural networks to meet customer needs. https://www.amazon.science/blog/how-aws-uses-graph-neural-networks-to-meet-customer-needs. Accessed: 2023-05-16.

[194] Shuwen Yang, Guojie Song, Yilun Jin, and Lun Du. Domain adaptive classification on heterogeneous information networks. In *Proceedings of International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1410–1416, 2021.

[195] Y Zhang, G Song, L Du, S Yang, and Y Jin. Dane: Domain adaptive network embedding. In *Proceedings of International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 4362–4368, 2019.

[196] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference (WWW)*, pages 1457–1467, 2020.

[197] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987*, 2022.

[198] Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:27965–27977, 2021.

[199] Yixin Liu, Kaize Ding, Huan Liu, and Shirui Pan. Good-d: On unsupervised graph out-of-distribution detection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 339–347, 2023.

[200] Vikash Sehwag, Mung Chiang, and Prateek Mittal. Ssd: A unified framework for self-supervised outlier detection. In *International Conference on Learning Representations (ICLR)*, 2021.

[201] Xiao Shen, Quanyu Dai, Sitong Mao, Fu-lai Chung, and Kup-Sze Choi. Network together: Node classification via cross-network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 32(5):1935–1948, 2020.

[202] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

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

[204] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[205] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[206] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[207] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, pages 1321–1330. PMLR, 2017.

[208] DC Dowson and BV666017 Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.