Efficient Representation Learning With Graph Neural Networks

YONGCHENG JING

Supervisor: Prof. Dacheng Tao

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

August 2023

To my parents and grandparents.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Yongcheng Jing
School of Computer Science
Faculty of Engineering
The University of Sydney

Authorship Attribution Statement

This thesis was conducted at the University of Sydney, under the supervision of Prof. Dacheng Tao, between 2020 and 2023. This thesis contains several chapters that were previously published by Yongcheng Jing as the first author in the following publications:

(1) **Yongcheng Jing**, Chongbin Yuan, Li Ju, Yiding Yang, Xinchao Wang, and Dacheng Tao. Deep Graph Reprogramming. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 24345–24354. Presented in Chapter 3. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

(2) **Yongcheng Jing**, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao. Amalgamating Knowledge from Heterogeneous Graph Neural Networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15709–15718. Presented in Chapter 4. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

(3) **Yongcheng Jing**, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao. Meta-Aggregator: Learning to Aggregate for 1-bit Graph Neural Networks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 5301–5310. Presented in Chapter 5. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

(4) **Yongcheng Jing**, Yining Mao, Yiding Yang, Yibing Zhan, Mingli Song, Xinchao Wang, and Dacheng Tao. Learning Graph Neural Networks for Image Style Transfer. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022, pp. 111–128. Presented in Chapter 6. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Yongcheng Jing	Date

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Dacheng Tao	Date

Acknowledgements

I am immensely grateful to Prof. Dacheng Tao, my supervisor, for guiding me from being a junior student to becoming a researcher. I feel incredibly fortunate to have had Prof. Tao by my side throughout this PhD journey, providing invaluable support and patient supervision. Prof. Tao is not only as my mentor but has also been an endless source of inspiration to me, with his profound research passion and unwavering commitment to academic excellence. I am truly honored to learn from him and will cherish his teachings as a priceless treasure throughout my life.

I am also filled with profound gratitude to express my deepest appreciation for my co-supervisor, Prof. Xinchao Wang, whose exceptional kindness, unwavering support, and enlightening discussions have been an invaluable asset throughout my academic journey. The profound impact that Prof. Wang has had on me, particularly in the realms of academic writing and cultivating innovative research ideas, cannot be overstated. The guidance and unwavering backing provided by Prof. Wang have played an indispensable role in shaping my scholarly path, and for that, I am forever grateful.

I also want to express my heartfelt gratitude to my colleagues and friends, Dr. Yiding Yang, Zhihao Cheng, Yajing Kong, Dr. Zhen Wang, Dr. Fengxiang He, Dr. Yu Cao, Dr. Sen Zhang, Dr. Hao Guan, Dr. Jiayan Qiu, Chongbin Yuan, Li Ju, Yining Mao, Shiye Lei, Xiaofei Liu, Haibo Qiu, Dr. Zhi Hou, Xu Zhang, Yufei Xu, Qiming Zhang, Wen Wang, Dr. Wei Zhai, Dr. Shanshan Zhao, Dr. Chaoyue Wang, Dr. Jing Zhang, Dr. Liang Ding, Dr. Baosheng Yu, Qi Zheng, Dr. Yibing Zhan, Haimei Zhao, Yang Qian, Xingyi Yang, Songhua Liu, Jingwen Ye, Gongfan Fang, Erdun Gao, Dr. Jie Song, Dr. Zunlei Feng, Cheng Wen, Sihan Ma, and Greg Ryan. Throughout my PhD journey, the unwavering assistance and encouragement I have received from them has been an invaluable asset, benefiting both my research endeavors and daily life in immeasurable ways.

Abstract

Graph neural networks (GNNs) have emerged as the dominant paradigm for graph representation learning, igniting widespread interest in utilizing sophisticated GNNs for diverse computer vision tasks in various domains, including visual SLAM, 3D object recognition and segmentation, as well as visual perception with event cameras. However, the applications of these GNNs often rely on cumbersome GNN architectures for favorable performance, posing challenges for real-time interaction, particularly in edge computing scenarios. This is particularly relevant in cases such as autonomous driving, where timely responses are crucial for handling complex traffic conditions. The objective of this thesis is to contribute to the advancement of learning efficient representations using lightweight GNNs, enabling their effective deployment in resource-constrained environments. To achieve this goal, the thesis explores various efficient learning schemes, focusing on four key aspects: the data side, the model side, the data-model side, and the application side. In terms of data-driven efficient learning, the thesis proposes an adaptive data modification scheme that allows a pre-trained model to be repurposed for multiple designated downstream tasks in a resource-efficient manner, without the need for re-training or fine-tuning. For model-centric efficiency, the thesis introduces a multi-talented and lightweight architecture, without accessing human annotations, that can integrate the expertise of the pre-trained complex GNNs specializing in different tasks. Furthermore, the thesis explores a dedicated binarization scheme on the data-model side that converts both input data and model parameters into 1-bit representations, resulting in lightweight 1-bit architectures. Finally, the thesis investigates an application-specific efficient learning scheme that models the style transfer process as message passing in GNNs, enabling efficient semi-parametric stylization.

Contents

List of Abbreviations

SLAM	Simultaneous Localization and Mapping 1
CNNs	Convolutional Neural Networks 1
GNNs	Graph Neural Networks 1
RNNs	Recurrent Neural Networks 8
DNNs	Deep Neural Networks 8
GCN	Graph Convolutional Network 9
GAT	Graph Attention Network 9
GIN	Graph Isomorphism Network 9
GatedGCN	Gated Graph Convolutional Network 9
GARE	Deep Graph Reprogramming 17
DARE	Data Reprogramming 17
MERE	Model Reprogramming 17
MetaFP	Meta-FeatPadding 17
EdgSlim	Edge-Slimming 18
MetaGP	Meta-GraPadding 18
KD	Knowledge Distillation 19
TAM	Topological Attribution Map 49
GNA	Greedy Gumbel Neighborhood Aggregator 77
ANA	Adaptable Hybrid Neighborhood Aggregator 77

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Background and Motivation

In recent years, there has been a remarkable progress in the field of computer vision, marked by an increasing number of diverse tasks and their inspiring applications. One prominent example is the application of computer vision in autonomous driving, which involves tasks such as visual *Simultaneous Localization and Mapping (SLAM)* for simultaneous pose estimation and map generation in unknown environments [151], visual perception using novel sensors like event cameras [38], and 3D object recognition and segmentation using different 3D data representations [45]. *Convolutional Neural Networks (CNNs)* have played a pivotal role in achieving success in these applications.

More recently, there has been a growing interest in leveraging the merits of *Graph Neural Networks (GNNs)* to further enhance computer vision tasks, including visual SLAM [139], point cloud processing [167, 88, 129, 127], object detection [55, 43], multi-person pose estimation and tracking [190], image classification [49] and super-resolution [223], as well as visual perception based on event cameras [140]. Notably, in the domain of visual SLAM, Sarlin *et al*. [139] introduced an attention-based GNN that addresses the optimal transport problem in local feature matching, achieving state-of-the-art performance in indoor and outdoor pose estimation. Similarly, Wang *et al*. [167] investigated a dynamic graph convolutional model for point cloud classification and semantic segmentation, combining the strengths of PointNet [126] and GNNs [82].

However, the favorable performance achieved by these applications is often attributed to cumbersome GNN architectures, which comes at the cost of significant computational requirements and high memory loads. For example, SuperGlue [139], a GNN-based local feature matching architecture in visual SLAM, necessitates 12 million network parameters to achieve leading performance. DeepGCNs, developed by Li *et al.* [90], leverage a 56-layer GCN architecture to alleviate the over-smoothing issue and achieve state-of-the-art performance in point cloud semantic segmentation.

The resulting computational burden stemming from these cumbersome architectures gives rise to three major challenges:

(1) **Challenge of deployment in resource-constrained environments:** The computational burden imposed by cumbersome architectures presents a significant challenge for the deployment of GNNs in resource-constrained environments, such as edge computing, where only limited computational resources are available;
(2) **Infeasibility for time-sensitive applications:** The complex and cumbersome nature of these architectures hinders their deployment in time-sensitive applications that require strict real-time interaction. For instance, in autonomous driving, GNN-based visual SLAM algorithms must maintain fast and timely responses to effectively handle sophisticated traffic conditions;
(3) **Scalability issues with large-scale graphs:** Existing cumbersome GNNs face challenges in processing increasingly large-scale graphs encountered in real-world scenarios, often involving millions of nodes and edges. The computational and memory resources required to handle such large-scale graphs are tremendous, making it a significant challenge for current complex GNN architectures.

Motivated by the challenges outlined above, this thesis aims to contribute to the field of ultra lightweight graph inference under limited memory and computational resources, achieved by developing a series of techniques for enhanced efficiency of graph representation learning with GNNs.

1.2 Problem Statement

To tackle the challenges posed by existing cumbersome GNN architectures, the thesis starts by subdividing the task of learning efficient representations with GNNs into four specific learning subproblems:

(1) **Data-driven efficient learning:** This subproblem focuses on adaptively *modifying the input graph data* while keeping the model parameters unchanged, enabling a single pre-trained model to be seamlessly adapted to various downstream tasks across different task levels and domains. For example, the pre-trained model can be initially trained for node classification, and then easily applied to graph classification and regression tasks without requiring *re-training* or *fine-tuning*;

(2) **Model-driven efficient learning:** In this subproblem, the goal is to extract knowledge from multiple pre-trained models with diverse architectures and *train a multi-talented yet lightweight network* that can effectively leverage the expertise of these pre-trained models, without accessing labels. The resulting architecture should integrate the capabilities of different pre-trained models, allowing it to simultaneously handle diverse tasks, such as point cloud segmentation and classification;

(3) **Data-model-driven efficient learning:** The aim of this subproblem is to explore a universal scheme that considers both data and model efficiency. The focus is not only on developing lightweight models but also on lightweight input data representations, thereby contributing to overall efficiency gains by jointly considering both aspects in graph inference tasks;

(4) **Application-driven efficient learning:** This subproblem investigates the feasibility of customized application-specific efficiency. It involves incorporating the principles of general efficient learning schemes while explicitly considering the specific characteristics of the designated task through customized algorithmic design. The objective is to enhance efficiency in a targeted application by tailoring the representation learning process to its unique requirements.

To address these four subproblems pertaining to data-driven, model-driven, data-model-driven, and application-driven efficiencies, this thesis discusses four complementary works. Each work effectively tackles one of the four perspectives, collectively contributing to the advancement of efficient representation learning with GNNs.

1.3 Contributions

The contributions of this thesis can be summarized as follows:

- **Thoroughly examining the challenges associated with efficient representation learning using GNNs and identifying four crucial problems**, including data-driven efficient learning, model-driven efficient learning, joint data-model-driven efficient learning, and application-driven efficient learning;
- **Development of an elaborate deep graph reprogramming scheme to tackle the problem of data-driven efficiency.** This scheme keeps the pre-trained model parameters unchanged while modifying input graphs through *feature padding*, *edge slimming*, and *graph padding*. It enables the adaptation of the pre-trained model to multiple downstream tasks without re-training or fine-tuning any part of the model;
- **Introduction of an innovative knowledge amalgamation scheme tailored for GNNs to address the issue of model-driven efficiency.** This scheme amalgamates knowledge from multiple pre-trained teacher models with heterogeneous architectures and expertise in different tasks, without requiring human-labeled annotations. The resulting student model maintains a compact architecture while integrating the expertise of the teachers, achieved by utilizing a *slimmable graph convolutional operation* to accommodate varying-dimensional features from the pre-trained teachers, as well as a *topological attribution map* scheme for learning the teachers' topological semantics;
- **Investigation of a novel binarization framework to simultaneously consider data- and model-side efficiency.** This framework adaptively binarizes both the full-precision 32-bit input graph data and the model parameters into more compact

1-bit representations for lightweight graph inference, while retaining competitive performance. It leverages two elaborated *meta neighborhood aggregators* to enhance the topological discriminative ability of the 1-bit GNNs;

- **Exploration of a customized efficient learning scheme specific to the application of neural style transfer.** This scheme models the style transfer procedure as attention-based *heterogeneous message passing* between style and content patches in a learnable manner, improving the efficiency of previous non-parametric neural style transfer methods that depend on greedy one-to-one patch matching.

1.4 Thesis Outline

The remainder of this thesis is organized into six chapters, with the main content of each chapter summarized as follows:

Chapter 2: Literature Review

This chapter presents an overview of the relevant literature pertaining to this thesis. It covers various aspects such as different architectures of GNNs and their applications, model reusing techniques, universal models, adversarial reprogramming approaches, CNN-based network binarization techniques, multi-task learning methods, and neural style transfer techniques.

Chapter 3: Data-Driven Efficient Learning with Deep Graph Reprogramming

In this chapter, a novel deep graph reprogramming paradigm is introduced for data-driven efficient learning. This paradigm enables the adaptation of a pre-trained GNN to multiple cross-level downstream tasks without the need for re-training or fine-tuning, only through the modification of the input data. The proposed approach combines three complementary techniques, namely *Meta-FeatPadding*, *Edge-Slimming*, and *Meta-GraPadding*, which effectively handle graphs with varying dimensions in transductive and inductive scenarios. Additionally, an elaborated *Reprogrammable Aggregating* method is employed to enhance the model capacity. The experimental results on fourteen benchmarks across node and graph classification, graph regression, point cloud classification, and action recognition, demonstrate

the competence of the pre-trained GNN with deep graph reprogramming in handling a wide range of downstream tasks.

Chapter 4: Model-Driven Efficient Learning with Knowledge Amalgamation

In the pursuit of model-driven efficient learning, this chapter presents a novel approach that leverages GNN-based knowledge amalgamation to train a versatile student model capable of encompassing the expertise of heterogeneous-task teachers, all without the need for human annotations. The proposed method employs a *slimmable graph convolutional operation* to accommodate varying-dimension features from the teachers. Additionally, a *topological attribution map* scheme is introduced to capture the topological semantics of the teachers. The effectiveness of this approach is evaluated on diverse tasks spanning different domains, including single- and multi-label node classifications, 3D object recognition, and part segmentation. The experimental results demonstrate that the learned student GNN model excels in handling these various tasks, sometimes even surpassing the performance of the individual teachers, while significantly reducing computational costs.

Chapter 5: Data-Model-Driven Efficient Learning with Meta-Aggregator

In this chapter, a novel GNN-customized binarization framework is investigated, aiming to achieve joint data and model efficiency. The framework enables the generation of a lightweight 1-bit GNN model while maintaining competitive performance, thus enabling its applicability in resource-constrained scenarios like edge computing. The framework leverages an adaptive *meta aggregation* scheme to effectively handle the challenges associated with quantized graph features. Extensive evaluations are conducted on multiple large-scale benchmarks across diverse domains and graph tasks, including graph regression, node classification, and 3D object recognition. The experimental results demonstrate the superiority of the proposed meta aggregators compared to state-of-the-art methods, showcasing their ability to outperform both the devised 1-bit binarized GNN models and general full-precision models. These findings highlight the effectiveness of the proposed framework in achieving efficient and high-performing GNN models, further validating its potential for various applications.

Chapter 6: Application-Driven Efficient Learning with Semi-parametric Style Transfer

This chapter focuses on application-driven efficient learning, presenting an efficient *semi-parametric* arbitrary stylization scheme designed specifically for image style transfer. The scheme enables the efficient generation of both global and local style patterns. This is specifically achieved by modeling the neural style transfer process as the information propagation between the content and style nodes in a stylization graph. As such, by formulating the style transfer procedure within the GNN framework, the proposed method achieves efficient arbitrary style transfer while taking advantage of the inherent structural information present in the images. Extensive comparative results involving six methods demonstrate that the proposed method efficiently produces high-quality stylized images, showcasing its efficiency and effectiveness in the field of image style transfer.

Chapter 7: Conclusions

This chapter concludes the thesis, summarizing the contributions and implications of the research conducted as well as the future directions.

CHAPTER 2

Literature Review

This chapter briefly reviews here several topics that are related to this thesis, including graph neural network (GNN) with its applications, model reusing, multi-task learning, prior CNN-based network binarization techniques, as well as neural style transfer methods.

2.1 Graph Neural Network

Deep Neural Networks (DNNs) have achieved remarkable success in handling regular data in the Euclidean space, including images, audios, and videos, across various domains spanning from image and video processing to natural language understanding. However, an increasing number of applications involve data samples that exhibit irregular graph structures in the non-Euclidean domain [177, 221]. For instance, in social networks, individual entities are represented as nodes, while the relationships between them are depicted as edges in a graph. Similarly, in chemical analysis, molecules can be represented as graphs, where nodes correspond to atoms and edges symbolize chemical bonds.

In comparison to Euclidean data, non-Euclidean graph samples are characterized by their irregularity, with variable structures and unordered nodes. Moreover, graph analysis encompasses a broader range of task levels and settings, including graph-, node-, and edge-level learning, as well as transductive and inductive scenarios. These topological diversities in the non-Euclidean domain present significant challenges for existing end-to-end DNN paradigms, such as *Convolutional Neural Networks (CNNs)* and *Recurrent Neural Networks (RNNs)*, when applied to graph-related applications.

Consequently, there has been a wealth of research dedicated to extending DNNs for analyzing topological graph data, leading to the development of GNNs [82, 32, 177, 221]. GNNs explicitly incorporate topological information within graphs for feature aggregation and have demonstrated promising performance in various graph-related tasks.

In recent years, GNNs have made significant advancements [82, 68, 221, 32, 189, 183, 92, 107, 158, 58, 91, 189, 119, 99, 177] and have emerged as the dominant learning paradigm for dealing with non-Euclidean graph data, which contains rich topological relational information. The literature has seen the introduction of numerous advanced and effective GNN architectures, including *Graph Convolutional Network (GCN)* [82], *Graph Attention Network (GAT)* [153], *Graph Isomorphism Network (GIN)* [180], *Gated Graph Convolutional Network (GatedGCN)* [7], and *GraphSAGE* [47].

In particular, the seminal work of Kipf and Welling [82] proposes GCNs, which successfully generalizes CNNs to deal with graph-structured data, by utilizing neighborhood aggregation functions to recursively capture high-level features from both the node and its neighbors. Also, GAT [153] introduces a novel attention mechanism for efficient graph processing. GraphSAGE [47], on the other hand, addresses the scalability issues on large-scale graphs by sampling and aggregating feature representations from local neighborhoods. Also, Huang *et al*. [58] propose an adaptive sampling strategy to improve the efficiency in training. Wang *et al*. [167] further devise a dynamic graph convolutional model where the topological connections among different nodes are adaptively changed in a learnable manner. Furthermore, to alleviate the oversmoothing problem in GCN [91, 119], Zhao *et al*. [219] and Rong *et al*. [136] propose a novel PairNorm layer and a DropEdge strategy, respectively. Moreover, the emerging transformers can also be treated as generalizations of GNNs with a fully connected structure [133, 203, 213, 214, 182].

The research on GNNs leads to increasing interest in deploying GNN models in various graph-based tasks, where the input data can be naturally represented as graphs [221]. For example, in social networks, the mutual relations among different individuals can be modeled as edges in a graph [125]. In biological domains, graphs can be readily used to represent molecule

structures [40, 67]. For the applications in recommendation systems, user interactions can also be easily modeled as the graph topological connections [172, 199].

Furthermore, the success of GNNs has also boosted the applications of graph networks in a wider range of problem domains, extending beyond traditional graph analysis [221], including semantic segmentation [167, 88, 129, 127], object detection [55, 43], pose estimation [190], interaction detection [128, 64], image classification [49] and super-resolution [223], visual perception with event cameras [140], and visual SLAM [139], etc. Specifically, Wang *et al*. [167] propose a dynamic graph convolutional model for point cloud classification and semantic segmentation, which combines the advantages of the PointNet [126] and graph convolutional network [82]. For the task of visual SLAM, Sarlin *et al*. [139] introduce an attention-based GNN to solve the optimal transport problem in local feature matching, which achieves the state-of-the-art performance on the tasks of indoor and outdoor pose estimation.

Despite the encouraging performance, there is a lack of research on compressing cumbersome GNN models, which is critical for deployment in resource-constrained environments like on the mobile-terminal side.

2.2 Model Reusing

Reusing pre-trained models has become increasingly prevalent in recent years. The seminal work of Hinton *et al*. [53] proposes the concept of knowledge distillation, where the soft labels obtained from a cumbersome teacher model are used for training a compact student model. Following this pioneering teacher-student framework, plenty of algorithms are proposed to fully utilize the knowledge concealed in the pre-trained teachers [138, 135, 205, 37, 225, 124, 144]. In particular, Rusu *et al*. [138] propose a novel progressive neural network to learn useful features from multiple teachers. Parisotto *et al*. [124], on the other hand, propose an Actor-Mimic scheme to reuse several teacher models specializing in diversified tasks. Also, a series of works in the literature propose to reuse multiple trained teacher CNNs [142, 143, 106, 196], working on different tasks, to learn a versatile student model, but built upon a strong assumption that the teacher models share the same CNN architecture.

Previously, reusing pre-trained models for downstream tasks was typically studied in the domain of convolutional neural networks (CNNs) [53, 138, 135, 205, 37, 225, 124]. In recent years, with the increasing number of pre-trained GNNs that have been generously released online for reproducibility, the vision community [16, 208, 164, 209, 185, 207] has witnessed a growing interest in reusing GNNs to enhance performance, alleviate training efforts, and improve inference speed [36, 76, 101, 102, 35, 165, 202, 163]. The seminal work is performed by Yang *et al*. [188], where a dedicated knowledge distillation (KD) method, tailored for GNNs, is proposed to obtain a lightweight GNN from a teacher. Deng *et al*. [26] further polish the work of Yang *et al*. [188] with a more challenging setting of graph-free KD. Moreover, Joshi *et al*. [76] improve the method of Yang *et al*. [188] with the prevalent contrastive learning scheme.

Model reusing is also related to *adversarial reprogramming*. Unlike previous adversarial attacks that aim to degrade the model performance, adversarial reprogramming seeks to utilize a class-agnostic perturbation to reuse and repurpose a target pre-trained model to perform the specific different task designated by the attacker. Adversarial reprogramming has recently been studied in various areas, including image classification [33, 34, 220, 84, 18] , text classification [120, 121], and language understanding [46].

However, the existence of adversarial reprogramming has not yet been validated in the non-Euclidean graph domain. This thesis is the first work that explores adversarial reprogramming in GNNs, and further innovatively *turns* such adversarial manner into guards to derive a novel task for resource-efficient model reusing.

2.3 Multi-task Learning

The proposed task of graph knowledge amalgamation in this thesis is also related to multi-task learning. Multi-task learning aims to leverage task relatedness to jointly learn a group of tasks with shared architectures [2, 23, 31, 42, 85, 216]. The setting of multi-task learning is similar to that of transfer learning, yet with a significant difference in objectives [216]. Specifically, in multi-task learning, the goal is to improve the performance of all the given

tasks equally. By contrast, transfer learning only focuses on the target task, with assistance from the source ones. In the past few years, multi-task learning has been widely studied in various areas, such as bioinformatics [51, 93], ubiquitous computing [178, 179], natural language processing [23, 176] and computer vision [105, 222, 74, 69]. Specifically, He *et al*. [52] develop a multi-task framework that combines object detection and segmentation. Also, Zhang *et al*. [217] devise a convolutional neural network architecture for joint face detection, pose estimation, and landmark localization. Chu *et al*. [22] construct a multi-task recurrent neural network, of which the output layer has multiple units to simultaneously estimate the relative distance, interactions, and standing orientations. A more recent work of Luo *et al*. [105] further proposes a multi-task collaborative network that achieves joint learning of referring expression comprehension and segmentation.

Multi-task learning is also related to the problem of deriving a *universal* model that is applicable to various-domain tasks. Such universal models have been previously studied in the image and language domains [5, 148, 132, 201, 113], such as the recently emerged BERT [27], GPT-3 [8], GPT-4 [123], and *Segment Anything* (SA) model [83].

While substantial advancements have been made in visual foundation models within the Euclidean domain, there remains a dearth of research focusing on universal models in the non-Euclidean graph domain, leaving an untapped area of investigation [73]. This thesis performs a pilot study on developing universal models in the non-Euclidean domain, thereby making one step further towards artificial general intelligence (AGI) [114].

2.4 Network Binarization

In the field of model compression [204, 142, 145, 15, 143], network binarization techniques aim to save memory occupancy and accelerate the network inference by binarizing network parameters and then utilizing bitwise operations [61, 60, 9]. In recent years, various CNN binarization methods have been proposed, which can be categorized into direct binarization [25, 61, 60, 79] and optimization-based binarization [131, 9, 111]. Specifically, direct binarization quantizes the weights and activations to 1 bit with a pre-defined binarization

function. In contrast, optimization-based binarization introduces scaling factors for the binarized parameters to improve the representation ability, but inevitably leading to inferior efficiency.

Driven by the success of the aforementioned binarization techniques in the CNN domain, in this chapter, we propose a GNN-specific binarization method. Specifically, we primarily focus on GNN-based direct binarization, since our goal is to develop super lightweight GNN models. We also notice three concurrent works [157, 159, 3] that also aim to accelerate the forward process for GNN models. However, two of them directly apply CNN-based binarization techniques without considering the characteristics of GNNs, which in fact will serve as the baseline method in our experiments [157, 159]. The other work only focuses on improving the efficiency of dynamic graph convolutional model [167], by speeding up the dynamic construction of k-nearest-neighbor graphs in the Hamming space [3].

Unlike the existing works [157, 159, 3], this thesis aims to devise a more general GNN-specific binarization framework that is applicable to most existing GNN models.

2.5 Neural Style Transfer

Driven by the power of *convolutional neural networks (CNNs)* [210, 218, 211, 87], Gatys *et al*. propose to leverage CNNs to capture and recombine the content of a given photo and the style of an artwork [39]. Following the CNN-based paradigm introduced by Gatys *et al*. [39], numerous subsequent studies have been conducted, giving rise to an emerging field known as neural style transfer (NST) within the computer vision community [210, 211, 218, 87]. The objective of the NST task is to automatically transfer the artistic style from a source style image to a given content image. To achieve this goal, existing NST approaches can be broadly divided into parametric and non-parametric NST methods, according to the way to capture the style information. Specifically, parametric NST approaches leverage the global representations to transfer the target artistic style, which are obtained by computing the summary statistics in either an image-optimization-based online manner [39, 94, 134, 104], or model-optimization-based offline manner [75, 212, 95, 162, 1, 100, 13, 96, 59, 12,

69]. On the other hand, non-parametric methods exploit the local feature patches to represent the image style [89, 147, 11, 19, 115, 97], inspired by the conventional patch-based texture modeling approaches with Markov random fields. The idea is to search the most similar neural patches from the style image that match the semantic local structure of the content one [89, 147, 11, 19, 115, 97]. For instance, Chen and Schmidt [19] introduced a style swap algorithm that associates each content patch with its most similar style patch and performs a swap operation between them. In a different approach, Sheng *et al*. [147] proposed a method to whiten the textures of feature patches before conducting the patch matching process. In summary, parametric neural style transfer methods excel in preserving global style patterns, whereas non-parametric neural methods demonstrate greater effectiveness when the content and style images possess similar structures.

This thesis aims to seek a balance between parametric and non-parametric NST methods and improve the efficiency of non-parametric ones by incorporating the use of GNNs.

CHAPTER 3

Data-Driven Efficient Learning with Deep Graph Reprogramming

This chapter investigates a novel deep graph reprogramming scheme for data-driven efficient learning with GNNs. The goal of deep graph reprogramming is to reprogram a pre-trained GNN, without amending raw node features nor model parameters, to handle a bunch of cross-level downstream tasks in various domains. To this end, this chapter proposes an innovative *Data Reprogramming* paradigm alongside a *Model Reprogramming* paradigm. The former one aims to address the challenge of diversified graph feature dimensions for various tasks on the input side, while the latter alleviates the dilemma of fixed per-task-per-model behavior on the model side. For data reprogramming, we specifically devise an elaborated *Meta-FeatPadding* method to deal with heterogeneous input dimensions, and also develop a transductive *Edge-Slimming* as well as an inductive *Meta-GraPadding* approach for diverse homogenous samples. Meanwhile, for model reprogramming, we propose a novel task-adaptive *Reprogrammable-Aggregator*, to endow the frozen model with larger expressive capacities in handling cross-domain tasks. Experiments on fourteen datasets across node/graph classification/regression, 3D object recognition, and distributed action recognition, demonstrate that the proposed methods yield gratifying results, on par with those by re-training from scratch.

3.1 Introduction

With the explosive growth of graph data, graph neural networks (GNNs) have been deployed across increasingly wider areas [191, 189, 71, 70, 190], such as recommendation system

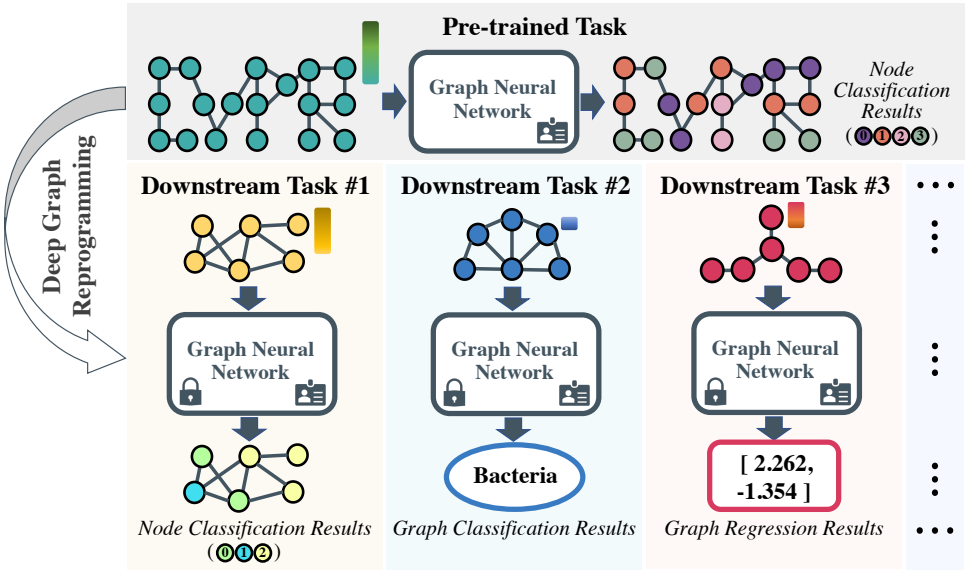


FIGURE 3.1: Illustrations of the proposed task of *deep graph reprogramming* (GARE) that aims to reuse pre-trained GNNs to handle plenty of cross-level tasks with heterogeneous graph feature dimensions, without changing model architectures nor parameters.

[172] and autonomous driving [139, 166, 171]. However, the favorable performance for such applications generally comes at the expense of tremendous training efforts and high memory loads, precluding the deployment of GNNs on the edge side. As such, reusing pre-trained GNNs to alleviate training costs has recently emerged as a trending research topic [36, 188, 187, 154, 186, 26, 68, 224, 76].

Pioneered by the work of [188] that generalize knowledge distillation [53, 192, 135, 194, 149, 195] to the non-Euclidean domain, almost all existing approaches on reusing GNNs are achieved by following the distillation pipeline in [188]. Despite the encouraging results, the distilling-based scheme is limited to the *per-task-per-distillation* setting, where a distilled model can only tackle the same task as the teacher can, leading to considerable storage and computation burdens, especially for the deployment of multiple tasks.

Meanwhile, the distillation mechanism rests upon the hypothesis that abundant pre-trained models are available in the target domains, which indeed holds for image-based areas that always take data in the regular RGB form, thereby readily allowing for *per-model-multiple-dataset* reusing. However, such an assumption is typically *not* satisfied in the non-Euclidean

domain: on the input side, irregular graph samples have heterogeneous feature dimensions, as shown with the color bars in Fig. 3.1; on the task side, graph analysis takes various task levels and settings, such as graph-, node-, and edge-level learning, as well as transductive and inductive scenarios. Such nature of topological diversities leads to inadequate pre-trained GNNs that fit the target downstream tasks.

In this chapter, we strive to take one step towards generalized and resource-efficient GNN reusing, by studying a novel *Deep Graph Reprogramming (GARE)* task. Our goal is to reuse a *single* pre-trained GNN across *multiple* task levels and domains, for example the pre-trained one working on node classification and the downstream ones on graph classification and regression, as shown in Fig. 3.1. We further impose two constraints to both data and model, where raw features and parameters are frozen in handling downstream tasks. As such, unlike distillation that essentially leverages a pre-trained teacher to guide the *re-training* of a student, the proposed task of GARE, without re-training nor fine-tuning, can thereby be considered to *reprogram* a pre-trained GNN to perform formerly unseen tasks.

Nonetheless, such an ambitious goal is accomplished with challenges: diversified graph feature dimensions and limited model capacities with a single frozen GNN. Driven by this observation, we accordingly reformulate GARE into two dedicated paradigms on data and model sides, respectively, termed as *Data Reprogramming (DARE)* and *Model Reprogramming (MERE)*. The goal of DARE is to handle downstream graph samples with both the heterogeneous and homogenous dimensions, without amending pre-trained architectures. Meanwhile, MERE aims to strengthen the expressive power of frozen GNNs by dynamically changing model behaviors depending on various tasks.

Towards this end, we propose a universal *Meta-FeatPadding (MetaFP)* approach for heterogeneous DARE that allows the pre-trained GNN to manipulate heterogeneous-dimension graphs, by accommodating pre-trained feature dimensions via adaptive feature padding in a task-aware manner. The rationale behind the proposed *MetaFP*, paradoxically, is derived from *adversarial reprogramming examples* [33] that are conventionally treated as attacks to learning systems, where attackers secretly repurpose the use of a target model without informing model providers, by inserting perturbations to input images. Here we turn the role

of the adversarial reprogramming example on its head, by padding around graph perturbations for generalized cross-task model reusing.

Complementary to the dedicated *MetaFP* that is tailored for heterogeneous-DARE, we also devise a transductive Edge-Slimming (EdgSlim) and an inductive Meta-GraPadding (MetaGP) methods for homogenous-DARE, that handle the downstream graphs with homogenous dimensions under transductive and inductive task settings, respectively, by adaptively eliminating node connections or inserting a tiny task-specific graph, with only, for example, ten vertices, to the raw input sample. Furthermore, we perform a pilot study on MERE, exploring the pre-trained model capacity for various downstream tasks, by only reprogramming the pre-trained aggregation behavior (*ReAgg*) upon the well-established *Gumbel-Max* trick.

In sum, our contribution is a novel GNN-based model reusing paradigm that allows for the adaption of a pre-trained GNN to multiple cross-level downstream tasks, and meanwhile requires no re-training nor fine-tuning. This is typically achieved through a series of complementary approaches entitled *MetaFP*, *EdgSlim*, and *MetaGP*, that tackle the heterogeneous- and homogenous-dimension graphs within the transductive and inductive scenarios, respectively, together with an elaborated *ReAgg* method to enhance the model capacity. Experimental results on fourteen benchmarks demonstrate that a pre-trained GNN with GARE is competent to handle all sorts of downstream tasks.

3.2 Motivation and Pre-analysis

In this section, we start by giving a detailed analysis on the dilemma of the prevalent reusing scheme of knowledge distillation, and accordingly propose the novel task of *deep graph reprogramming (*GARE*)*, which leads to resource-efficient and generalized GNN reusing. Then, we uncover the two key challenges of GARE and introduce the proposed paradigms of DARE and MERE with the elaborated rationales on how to address the two challenges.

3.2.1 Task Motivation and Definition

Prevalent Distillation-based Reusing. In the literature, almost all existing methods for reusing GNNs are achieved by *Knowledge Distillation (KD)* elaborated in Task 3.2.1.

Task 3.2.1 (Reusing GNNs via Knowledge Distillation). *The goal of knowledge distillation is to re-train a compact student model from scratch, that masters the expertise of the pre-trained teacher, via extracting and transferring the knowledge from the pre-trained cumbersome teacher model.*

Such a *KD* manner is inevitably limited by two issues:
- 1. *KD* is built upon an ideal condition that for any downstream task, sufficient pre-trained teacher models are always available for reusing. Such an assumption indeed holds for most cases of image analysis, where the input data is always RGB-pattern. As such, the publicly available model trained on large-scale datasets, such as *ImageNet*, is readily reusable for downstream classification tasks. However, graph data instead has highly diversified input dimensions, feature types (*e.g.*, node and edge features), as well as various task levels (*e.g.*, node- and graph-level analysis), making it challenging to reuse online-released GNNs, as image-domain does, for such diversified graph scenarios;
- 2. *KD* is resource-*inefficient*. The distilled model from *KD* only handles exactly the same task as the teacher does. In other words, every student is always unique to a single task, leading to model redundancy for multi-task scenarios.

Proposed Novel Deep Graph Reprogramming (GARE). Driven by the challenges of the *KD*-based model reusing scheme, we develop in this chapter a novel paradigm of *deep graph reprogramming* (GARE) for more generalized and resource-efficient model reusing, that explicitly considers the topological uniqueness of graph data:

Task 3.2.2 (Reusing GNNs via Deep Graph Reprogramming). *Deep graph reprogramming aims to reuse a pre-trained model, without changing any architecture nor parameter, for*

a bunch of various-domain and cross-level downstream tasks, via reprogramming graph data or model behaviors.

As such, the proposed GARE is ideally superior to the ubiquitous *KD* with the following merits:

+ 1. GARE allows for the reuse of a *single* pre-trained GNN for *multiple* cross-level/domain downstream tasks and datasets, as shown in Fig. 3.1, thereby getting rid of the *KD* restriction on well-provided pertinent pre-trained models;

+ 2. GARE is free of re-training or fine-tuning, unlike *KD* that substantially re-trains a student model from scratch, thereby making it possible for deployment in resource-constrained environments such as edge computing;

+ 3. GARE is memory-efficient, where a pre-trained model with GARE is anticipated to be versatile and multi-talented that integrates the expertise of multiple tasks.

3.2.2 Challenges Towards GARE

The ambitious goal of GARE in Task 3.2.2 is primarily accomplished with the two key challenges:

※ **Data Side:** The first issue to be tackled regards handling various-dimension downstream features, considering that the pre-trained GNN in GARE is frozen without auxiliary transforming layers nor fine-tuning. For example, every node in the *Cora* citation network has 1433 input features, whereas that in *Amazon Co-purchase* graphs has 767 ones;

※ **Model Side:** The second challenge towards GARE lies in the insufficient model capacity under the per-GNN-multiple-task scenario of GARE, especially for cross-domain downstream tasks as shown in Fig. 3.1.

To tackle the data and model dilemmas of GARE, we devise a couple of data and model reprogramming paradigms, respectively, as will be elaborated in the following sections.

3.2.3 Reprogramming Paradigms for GARE

3.2.3.1 Data Reprogramming (DARE)

Rationale Behind DARE. To tackle the problem of diversified features on the input side, a naïve idea is rearranging graph representations to adapt varying-dimension downstream features to accommodate to the pre-trained GNN. As such, the challenge instead comes to be how to adapt the target downstream features.

To address this challenge of feature adaption, we paradoxically resort to a special type of adversarial attack [66, 215, 108, 110, 168, 169], termed as *adversarial reprogramming attack* [33]. In essence, the adversarial reprogramming attack demonstrates a security vulnerability of CNNs, where an attacker can easily redirect a model with perturbations to perform the selected task without letting the model providers know, thereby leading to ethical concerns, such as repurposing housekeeping robots to criminal activities.

Our idea here is to flip the role of adversarial reprogramming attacks, by turning the attackers that mean to perturb the model usage, into guards that aim to repurpose a pre-trained GNN to perform the intended downstream tasks.

However, adversarial reprogramming attack is formerly merely studied in the CNN domain. As such, it remains unknown in the machine learning community whether GNNs are also vulnerable to adversarial reprogramming attacks, which is a *prerequisite* for the success in applying the idea of adversarial reprogramming to DARE.

To this end, we as attackers perform in Tab. 3.1 an evasion attack on graph data, that tries to repurpose a pre-trained GNN designated for *product category prediction* to the new tasks of *molecule classification* and *molecule property regression*, through simply *adding* the generated adversarial perturbations to the raw node features [150]. We employ here the datasets of *AmazonCoBuy* [112, 161], *ogbg-molbbbp* [174], and *ogbg-molesol* [174] as examples. Surprisingly, with such a vanilla manner, the attacked pre-trained GNNs are extraordinarily competent to handle the unseen tasks, which have a significant domain gap with the former ones, leading to the observation as follows:

TABLE 3.1: Results of adversarial reprogramming attacks on graphs.

Roles	**Model Provider**	**Adversarial Attacker**	**Model Provider**	**Adversarial Attacker**
Datasets	`Computers`	`ogbg-molbbbp`	`Photo`	`ogbg-molesol`
Task Types	Computer-Category Prediction	Molecule Classification	Photo-Category Prediction	Molecule Regression
Before Attack	*Accuary*: 0.9485	–	*Accuary*: 0.9561	–
After Attack	–	*ROC-AUC*: 0.6132	–	*RMSE*: 2.7479
Re-training	–	*ROC-AUC*: **0.6709**	–	*RMSE*: **1.3000**

Remark 3.2.1 (Adversarial Reprogramming Attacks on Graph Data). *Graph neural networks are susceptible to adversarial reprogramming attacks, where an adversarial perturbation on graph data can readily repurpose a graph neural network to perform a task chosen by the adversary, without notifying the model provider.*

Motivations of Heter-DARE-*MetaFP* and Homo-DARE-*EdgSlim+MetaGP* Methods. We turn our role back from attackers to reputable citizens that would like to reuse a pre-trained GNN to alleviate the training efforts for downstream tasks. Remark 3.2.1 thereby illustrates that:

► **1.** It is technically feasible to convert adversarial reprogramming attack to effective DARE on graph data, which **motivates** us to devise a universal *Meta-FeatPadding (MetaFP)* approach, upon adversarial node feature perturbations, for heterogeneous-DARE (Sect. 3.3.2);

► **2.** Except for node-level perturbations, other adversarial example types tailored for graph data should also be effective for DARE, such as edge-level perturbations and structure-level perturbations [150], **motivating** us to develop a *transductive Edge-Slimming (EdgSlim)* (Sect. 3.3.3) and an *inductive Meta-GraPadding (MetaGP)* (Sect. 3.3.4) approaches, respectively, for homogenous-DARE.

3.2.3.2 Model Reprogramming (MERE)

Rationale Behind MERE. Backed by the theory of adversarial reprogramming attacks (Remark 3.2.1), in most cases, a pre-trained model equipped with DARE in Sect. 3.2.3.1 can already achieve encouraging results in tackling various downstream tasks. Despite its gratifying performance, we empirically observe that the downstream performance by only using DARE is prone to a bottleneck especially for some tasks that have considerable domain

gaps with the pre-trained one, for example the pre-trained task on paper analysis and the other one on e-commerce prediction.

We conjecture that such a bottleneck is due to the frozen GNN parameters and architectures, leading to insufficient expressive capabilities in modeling cross-domain topological properties. Motivated by the above observation, we further develop a MERE paradigm to strengthen the model capacities, acting as a complement to DARE under the scenarios of tremendous-domain-gap GNN reusing.

To this end, a vanilla possible solution for model enhancement will be resorting to dynamic networks [50] that are well-studied in the CNN domain. Plenty of dynamic inference schemes that are designated for CNNs, in fact, are equally feasible to the non-Euclidean domain of GNNs, such as early exiting, layer skipping, and dynamic routing. Moreover, almost all these dynamic strategies require no changes to original model parameters, thus readily acting as a specific implementation of MERE.

Nevertheless, instead of simply using CNN-based dynamic network schemes, we perform in this chapter a pilot study of MERE by explicitly considering the most critical characteristic that is unique to GNNs, namely *message aggregation*, and leaving the explorations of other dynamic paradigms in MERE for future works.

In the literature, message aggregation schemes have already been identified as one of the most crucial components in graph analysis, both empirically and theoretically [24]. However, the significance of aggregation behaviors in model reusing has not yet been explored, which is a precondition for the success of aggregation-based MERE.

To this end, we explore the MERE paradigm by firstly performing a prior study with *Cora* dataset in Fig. 3.2, that attempts to observe the diverse performance of reusing a fixed GNN pre-trained for classifying the node categories of {Case-Based, Genetic-Algorithm, Neural-Network, Probabilistic-Method}, to directly handle the separate downstream classes of {Reinforcement-Learning, Rule-Learning, Theory}, by only replacing the pre-trained aggregator with other aggregation methods. Remarkably, different aggregators in Fig. 3.2 lead to distinct downstream performance, which can be summarized as:

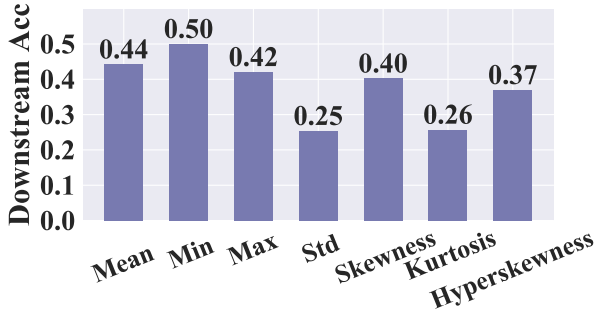


FIGURE 3.2: Reusing with various aggregators.

Remark 3.2.2 (**Aggregation Matters for Reusing**). *Various aggregators lead to diversified downstream task performance with the same model. There exists an optimal aggregation method tailored for each pair of downstream tasks and pre-trained models.*

► **Motivated** by Remark 3.2.2, we accordingly derive a *reprogrammable aggregating (ReAgg)* method as a specific implementation of the MERE paradigm, that aims to dynamically change the aggregation behaviors under various downstream scenarios, which will be elaborated in Sect. 3.3.5.

3.3 Proposed Methods: Implementing DARE and MERE Paradigms

In this section, we instantiate the proposed paradigms of DARE (Sect. 3.2.3.1) and MERE (Sect. 3.2.3.2), by elaborating three DARE methods and one MERE approach, tailored for various scenarios of the GARE-based model reusing.

3.3.1 Overview and Case Discussions

As analyzed in Sect. 3.2.3, a vanilla method to achieve DARE is generating an adversarial feature perturbation as an addition to raw features. However, such a naïve addition manner is prone to a heavy computational burden, especially for high-dimensional-feature scenarios, where we have to optimize an equally high-dimensional perturbation for downstream tasks.

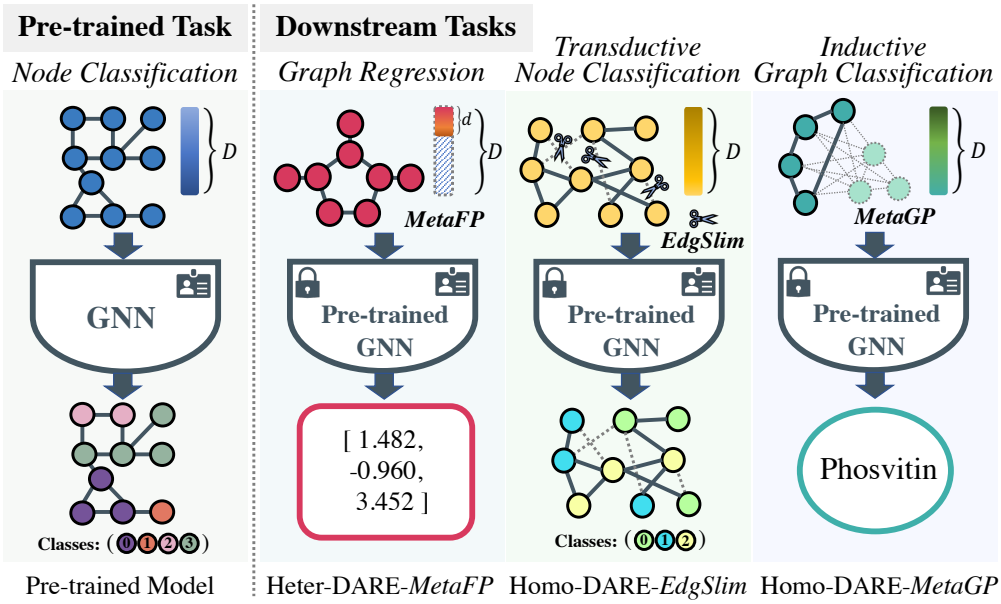


FIGURE 3.3: Illustration of the proposed approaches of *MetaFP*, *EdgSlim*, and *MetaGP* for transductive and inductive DARE with heterogeneous and homogenous input dimensions.

Also, such perturbation addition manner completely changes all raw inputs, thereby intrinsically can be interpreted as transforming downstream data to pre-trained one for model reusing, leading to performance bottleneck when the data gap is too significant for transformation.

Motivated by this observation, we resort to generating lower-dimensional perturbations as *paddings* around raw features, *never* amending any raw input feature. As such, the issues of both computational costs and troublesome transformation are simultaneously alleviated.

Despite its merits, such a perturbation padding manner can only be applicable to the scenario where pre-trained and downstream features have heterogeneous dimensions. Driven by this consideration, we propose to divide the GARE scenarios into *three cases*, and explore them separately to devise the corresponding best-suited methods for more resource-efficient model reusing:

• **Case #1. Universal-Heter-DARE:** Heterogeneous dimensions between pre-trained and downstream features under both transductive and inductive settings, addressed by *MetaFeatPadding* in Sect. 3.3.2;

• **Case #2. Transductive-Homo-DARE:** Homogenous pre-trained and downstream dimensions for transductive tasks, solved by *Edge-Slimming* in Sect. 3.3.3;

• **Case #3. Inductive-Homo-DARE:** Homogenous input dimensions for inductive tasks, tackled by *Meta-GraPadding* in Sect. 3.3.4.

Furthermore, we provide in Sect. 3.3.5 an examplar implementation of the MERE paradigm, by proposing *reprogrammable aggregation (ReAgg)* that aims to complement DARE on the model side for challenging downstream tasks.

3.3.2 Universal Meta-FeatPadding for Heter-DARE

The proposed *Meta-FeatPadding (MetaFP)* aims to accommodate the diverse downstream feature dimensions, by padding around the raw features, supported by the theory of *node-level* adversarial perturbations [150].

Given a pre-trained model termed $\texttt{GNN}_{\text{pre-trained}}$, the process of generating the padded features for downstream tasks with *MetaFP* can be formulated as follows:

$$\min_{\boldsymbol{\delta}_{\text{padding}}} E_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}_{\text{downstream}}\left(\texttt{GNN}_{\text{pre-trained}}[x||\boldsymbol{\delta}_{\text{padding}}], y\right)\right], \tag{3.1}$$

where $\mathcal{D}$ is the downstream data distribution, with (x, y) denoting the downstream graph features and the associated labels, respectively. Also, we use $||$ to represent the concatenation operation, which, in fact, performs feature padding that combines the optimized padding features $\boldsymbol{\delta}_{\text{padding}}$ with the raw input features x.

As such, the task-specific $\boldsymbol{\delta}_{\text{padding}}$ not only accommodates the feature dimensions of the downstream tasks to those of the pre-trained one with the frozen model of $\texttt{GNN}_{\text{pre-trained}}$, but also benefits the downstream performance by reducing the loss derived from the downstream loss function of $\mathcal{L}_{\text{downstream}}$. Although $\boldsymbol{\delta}_{\text{padding}}$ needs to be learned through gradient descent, the optimization process of $\boldsymbol{\delta}_{\text{padding}}$ is empirically very fast for most cases, where only one or several epochs are typically sufficient for converged results, which is much faster than model re-training.

During inference, the optimized universal $\boldsymbol{\delta}_{\text{padding}}$ on training downstream data is padded around all the testing downstream samples to obtain the prediction results. Furthermore, for the case where the output downstream dimensions are not aligned with the pre-trained ones, we simply use the corresponding part of the pre-trained neurons at the final linear layer, which is a common avenue in dynamic networks.

3.3.3 Transductive Edge-Slimming for Homo-DARE

The devised *MetaFP* in Sect. 3.3.2 is competent to tackle the heterogeneous-dimension case of GARE-based model reusing. Despite its encouraging performance, *MetaFP* is not effective in handling the downstream graph samples that have homogenous feature dimensions to the pre-trained ones, since it is no longer necessary to conduct meta padding for dimension accommodation. As such, it remains challenging in such homogenous-dimension cases, on performing DARE to adapt the pre-trained model to new tasks.

Driven by this challenge, we turn from *node-level* perturbations to another type of adversarial graph examples, namely *adversarial edge-level perturbations* [150], that aim to attack the model by manipulating edges. Here, we flip again the attacker role of *adversarial edge-level perturbations* to achieve resource-efficient model reusing, by modifying the node connections in the downstream graph data, meanwhile without changing raw node features, leading to the proposed *Edge-Slimming (EdgSlim)* DARE approach.

To this end, we formulate the algorithmic process of *EdgSlim* as a combinatorial optimization problem:

$$\begin{aligned} \min_{\{u_i,v_i\}_{i=1}^m} & \sum_{i=1}^{m} \left| \frac{\partial \mathcal{L}_{\text{downstream}}}{\partial \alpha_{u_i,v_i}} \right| \\ \textit{s.t. } \tilde{\mathcal{G}} &= \text{Modify}\left(\mathcal{G}, \{\alpha_{u_i,v_i}\}_{i=1}^m\right) \\ &= \left(\mathcal{G} \setminus \{u_i, v_i\}\right), \text{ if } \frac{\partial \mathcal{L}_{\text{downstream}}}{\partial \alpha_{u_i,v_i}} > 0, \end{aligned} \tag{3.2}$$

where $\{u, v\}$ denotes the connection between the node u and v, with m representing the total number of edges in the input graph $\mathcal{G}$. $\mathcal{L}_{\text{downstream}}$ is the loss function for the downstream task. α_{u_i,v_i} is our constructed unary edge feature, such that we can compute the derivative of

$\mathcal{L}_{\text{downstream}}$ with respect to the adjacency matrix of $\mathcal{G}$, with $\alpha_{u,v} = \mathbb{I}(u \in \mathcal{N}(v))$ where $\mathcal{N}(v)$ denotes the set of neighbors for the node v. $\setminus$ represents the edge deletion operation.

As such, Eq. 3.2 indicates that the proposed *EdgSlim* sequentially slims the connections in the downstream graph of which the corresponding edge gradients are greater than 0, starting from the edge with the largest gradients. The downstream loss can thereby be reduced by simply optimizing the connections. Notably, similar to *MetaFP*, the optimization with *EdgSlim* converges very fast, typically with only several epochs, and meanwhile occupies limited resources.

3.3.4 Inductive Meta-GraPadding for Homo-DARE

In spite of the gratifying results of *EdgSlim*, Eq. 3.2 is not applicable to the inductive task setting, where plenty of graphs are received as inputs. In this case, the edge slimming operation can only be performed on training graphs, not capable of transferring to the testing ones. To alleviate this dilemma, we propose a *Meta-GraPadding (MetaGP)* method to tackle the inductive GARE scenarios, where the downstream features have the same dimensions as the pre-trained ones, as illustrated in Fig. 3.3.

Our design of *MetaGP* is driven by the *structure-level perturbation* in adversarial examples [150]. In particular, instead of padding the generated perturbations around raw node features, the proposed *MetaGP* yields a tiny subgraph, with only, for example, ten nodes, which is then padded around every downstream graph, of which each meta node connects the downstream graph nodes in a fully-connected manner. The features in the introduced meta graph are generated in the same way as that of yielding padded features in Eq. 3.1. At the inference stage, the learned meta-graph is padded around all the testing graphs, leading the pre-trained GNN to perform the target downstream inductive task.

The process of generating the meta-graph in *MetaGP* is computation-efficient, where a meta-graph with only ten nodes is typically sufficient for most tasks. Moreover, the feature generation procedure is also lightweight, given the property of inductive graph learning tasks where the input features are generally low-dimension, *e.g.*, the *QM7b* dataset having only

TABLE 3.2: Results of reusing a pre-trained model on *Citeseer* to simultaneously handle four unseen tasks with heterogeneous dimensions and objectives, averaged with 20 independent runs. "Re-training" indicates whether the pre-trained parameters are changed. Notably, the 8$^{\text{th}}$ line shows that *ReAgg* is more competent for the large-domain-gap scenarios (2.3% improvement averagely), but slightly falls behind for *similar*-domain tasks, such as {*Cora*, *Citeseer*}, both of which classify computer science papers. Also, our *MetaFP* yields stable results that only slightly vary with padding initializations, with standard deviations of {0.0030, 0.0023, 0.0006, 0.0008} for the four downstream tasks.

Methods	**Model Re-training?**	**Model Parameter Sizes**	**Pre-trained Task**	**Downstream Heterogeneous Tasks**			
			Citeseer	Cora	Pubmed	Computers	Photo
Pre-trained Model [153]	×	474.89K	0.7950	N/A	N/A	N/A	N/A
Training from Scratch [153]	√	919.10K	0.7950	0.9144	0.8530	0.9475	0.9555
Reusing via Fine-tuning [57]	√	474.89K	0.7950	0.8710	0.8860	0.9542	0.9555
Multi-task Learning [10] + SlimGNN [68]	√	477.62K	0.7880	0.8780	0.8450	0.9108	0.9317
Vanilla Reusing [153] + SlimGNN [68]	×	474.89K	0.7950	0.1571	0.3250	0.5037	0.2183
Ours (*MetaFP*)	×	**474.89K**	**0.7950**	**0.8335**	**0.7790**	**0.9085**	**0.8909**
Ours (*MetaFP + ReAgg*)	×	**474.89K**	**0.7950**	**0.8312**	**0.8030**	**0.9229**	**0.9213**

1-dimension features, as well as the *ogbg-molbace*, *ogbg-molbbbp*, and *ogbg-molesol* datasets with an input feature dimension of nine.

3.3.5 Reprogrammable Aggregating for MERE

With the three elaborated DARE methods demonstrated in the preceding sections, a pre-trained GNN can already achieve empirically encouraging results in various downstream tasks and settings. To further improve the reusing performance especially under the large-domain-gap scenarios, we propose here a *reprogrammable aggregating (ReAgg)* method as a pilot study of the MERE paradigm.

Driven by Remark 3.2.2, the goal of the proposed *ReAgg* is to adaptively determine the optimal aggregation behaviors conditioned on different downstream tasks, without changing model parameters, thereby strengthening the model capacities. However, such an ambitious goal comes with the challenge of the undifferentiable discrete decisions of aggregators. To address this challenge, one possible solution is resorting to *reinforcement learning (RL)*. However, it is a known issue that *RL* is prone to a high computation burden, due to its Monte Carlo search process. Another solution is to use the *improved SemHash* technique [77] for

discrete optimization. However, we empirically observe that *improved SemHash* for MERE is likely to cause the collapse issue, where a specific aggregator is always or never picked up.

Motivated by the above analysis, we propose to leverage Gumbel-Max trick [152] for *ReAgg*, which is a more prevalent strategy for optimizing discrete variables than *improved SemHash* in dynamic neural networks [50]. In particular, to alleviate the dilemma of model collapse, we propose incorporating stochasticity into the aggregator decision process with the well-studied Gumbel sampling [109, 152]. We then propagate the gradients via the continuous form of the Gumbel-Max trick [65]. Specifically, despite the capability in parameterizing discrete distributions, the Gumbel-Max trick is, in fact, dependent on the *argmax* operation, which is non-differentiable. To address this issue, we thereby employ its continuous relaxation form of the Gumbel-softmax estimator that replaces *argmax* with a *softmax* function.

The detailed process of determining the optimal aggregation manner for each downstream task can be formulated as: $\text{Aggregator}_{\mathbf{k}} = \text{softmax}\big((\mathcal{F}(\mathcal{G}) + G)/\tau\big)$, where $\mathbf{k}$ denotes the k-th downstream task. Also, G denotes the sampled Gumbel random noise, which introduces stochasticity to avoid the collapse problem. $\mathcal{F}$ represents the intermediate features with $\mathcal{G}$ as inputs. τ is a constant denoting the softmax temperature. We clarify that for superior performance, $\mathcal{F}$ can be generated by feeding $\mathcal{G}$ into a transformation layer, which lies out of the pre-trained model and does not directly participate in the inference process as a part of the Gumbel-softmax estimator. In this way, the proposed *ReAgg* adaptively determines the optimal aggregator conditioned on each task, also without changing any pre-trained parameters, thereby enhancing the model capability.

3.4 Experiments

We evaluate the performance of a series of DARE and MERE approaches on fourteen publicly available benchmarks. Here we clarify that our goal in the experiments is *not* to achieve the state-of-the-art performance, but rather reusing a pre-trained GNN to yield favorable results for as many downstream tasks as possible under limited computational resources.

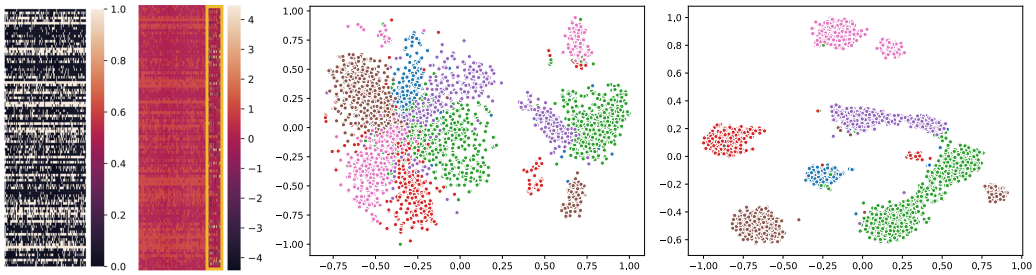
TABLE 3.3: Ablation studies of diverse padding sizes/positions and various pre-trained/downstream tasks. Notably, our *MetaFP* is effective even with tiny sizes and random positions.

Set of Heterogeneous Tasks {Pre-trained, Downstream}	**Padding Size**	**Padding Positions**			
		Front	Center	End	Random
{Computers, Photo}	22	0.9183	0.9161	0.9212	0.9165
{Photo, Pubmed}	245	0.8420	0.8430	0.8420	0.8440
{Computers, Pubmed}	267	0.8300	0.8320	0.8370	0.8330
{Cora, Computers}	666	0.8585	0.8792	0.8561	0.8910
{Cora, Photo}	688	0.8337	0.8785	0.8402	0.8915
{Cora, Pubmed}	933	0.8180	0.8210	0.8200	0.8240
{Citeseer, Cora}	2270	0.8370	0.8335	0.8417	0.8535
{Citeseer, Pubmed}	3203	0.7790	0.7750	0.7740	0.8010

3.4.1 Experimental Settings

Implementation Details. Detailed dataset statistics can be found in the next section of additional results and details. In particular, we follow [56, 68] to split *Amazon Computers*, *Amazon Photo*, and the OGB datasets, whereas for *Cora*, *Citeseer*, and *Pubmed* datasets, we use the splitting protocol in the supervised scenario for more stable results, as also done in [17]. Task-by-task architectures and hyperparameter settings can be found in the next section of additional results and details. All the experiments are performed using a single NVIDIA GeForce RTX 2080 Ti GPU.

Comparison Methods. Given our novel GARE setting, there are few existing methods in the literature for a fair comparison, either with distinct task settings or inconsistent objectives. As such, we derive two possible solutions that partly match our task. Specifically, we derive a *reusing via fine-tuning* approach that reuses a GNN via fine-tuning the parameters for downstream tasks. Moreover, a *multi-task-learning+SlimGNN* pipeline is also developed, that trains a slimmable graph convolution [68] from scratch to accommodate the pre-trained and downstream feature dimensions.



(a) (b) (c) (d)

FIGURE 3.4: Feature/t-SNE visualizations of (a, c) before padding and (b, d) after padding, with the yellow frame indicating the paddings.

3.4.2 Reprogramming in Heterogeneous Domains

Heterogeneous Node Property Prediction. We show in Tab. 3.2 the results of reprogramming a pre-trained GNN for a bunch of cross-domain node classification tasks, and further give in Tab. 3.3 the ablation studies of diverse padding sizes and positions as well as different pre-trained and downstream tasks. The 6$^{\text{th}}$ line of Tab. 3.2 shows that *MetaFP* makes it possible for cross-domain GNN reusing. Also, the proposed *ReAgg* for MERE further improves the downstream performance by about 2.3% on average (the 7$^{\text{th}}$ line of Tab. 3.2). Moreover, the visualization results before and after applying *MetaFP* are demonstrated in Fig. 3.4.

TABLE 3.4: Results of reusing a single model of a node-level task to directly tackle graph regression and graph classification tasks.

Pre-trained Task **Heterogeneous** Task Type	Photo ***Product Category Prediction***			
Pre-trained Results	*Acc*: 0.9561			
Downstream Tasks **Heterogeneous** Task Types	QM7b ***Molecule Regression***		PROTEINS ***Protein Prediction***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*MAE*: 24.18	*MAE*: **2.3093**	*Acc*: 0.3304	*Acc*: **0.6071**
Re-training from Scratch	*MAE*: 0.7264		*Acc*: 0.6964	
Pre-trained Task **Heterogeneous** Task Type	Cora ***Publication Classification***			
Pre-trained Results	*Acc*: 0.9121			
Downstream Tasks **Heterogeneous** Task Types	QM7b ***Molecule Regression***		PROTEINS ***Protein Prediction***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*MAE*: 13.04	*MAE*: **0.8889**	*Acc*: 0.4018	*Acc*: **0.5893**
Re-training from Scratch	*MAE*: 0.7264		*Acc*: 0.6964	

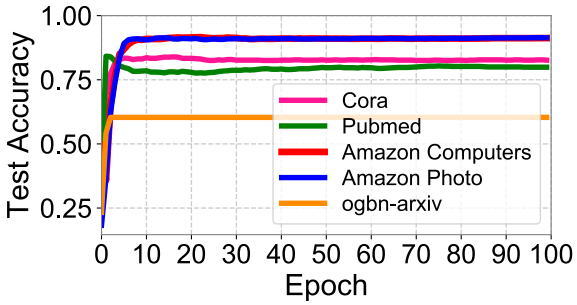


FIGURE 3.5: Convergence speed of the proposed method.

TABLE 3.5: Results of Homo-DARE that adapts a pre-trained node property prediction model (*ogbn-arxiv-s1*) to handle *20* unseen *homogenous* categories (*ogbn-arxiv-s2*) in ogbn-arxiv dataset [56].

Homogenous Multi-class	**Re-train?**	**Params**	**Pre-trained Task** `ogbn-arxiv-s1`	**Downstream Task** `ogbn-arxiv-s2`
Number of Classes	-	-	20	20
Pre-trained Model [153]	×	35.75K	**0.7884**	N/A
Training from Scratch [153]	√	35.75K	N/A	**0.8115**
Reusing via Fine-tuning [57]	√	35.75K	N/A	0.8112
Multi-task Learning [10]	√	38.35K	0.6387	0.6776
Vanilla Reusing [153]	×	35.75K	N/A	0.2334
Ours (*EdgSlim*)	×	35.75K	**0.7884**	0.6034

Furthermore, we'd also like to highlight in Fig. 3.5 that our method reaches convergence with only a few epochs, making it possible for deployment in resource-constrained environments.

Heterogeneous Graph Classification and Regression. Tab. 3.4 shows the results of reusing a GNN for the more challenging cross-level tasks, indicating our proficiency in such cross-level scenarios.

3.4.3 Reprogramming in Homogenous Domains

Homogenous Node Property Prediction. We perform in Tab. 3.5 extensive experiments of reusing a GNN for homogenous downstream tasks in a class-incremental setting. The proposed *EdgSlim*, as shown in Tab. 3.5, achieves competitive performance at a low computational cost (Fig. 3.5).

TABLE 3.6: Results of homogenous cross-domain graph-level tasks.

Pre-trained Task **Homogenous** Task Type	`ogbg-molbace` ***Molecular Classification***			
Pre-trained Results	*ROC-AUC*: 0.7734			
Downstream Tasks **Homogenous** Task Types	`ogbg-molbbbp` ***Molecular Classification***		`ogbg-molesol` ***Molecular Regression***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*ROC-AUC*: 0.5136	*ROC-AUC*: **0.6691**	*RMSE*: 6.950	*RMSE*: **2.050**
Re-training from Scratch	*ROC-AUC*: 0.6709		*RMSE*: 1.300	

Homogenous Graph Classification and Regression. We show in Tab. 3.6 the cross-domain results for homogenous graph-level tasks, where our *MetaGP* is proficient in reusing a graph classification model for the task of graph regression.

3D Object Recognition. Tab. 3.7 shows the results of reusing a pre-trained DGCNN tailored for `ModelNet40` [175], to tackle distinct downstream classes in `ShapeNet` [197]. Remarkably, the proposed *MetaGP* makes it possible for such cross-domain model reusing with large-scale 3D datasets. We also illustrate in Fig. 3.6 the structure of the feature space at the intermediate layer, showing that ours leads to semantically similar structures to those of re-training from scratch.

Distributed Action Recognition. We construct temporally growing graphs from `WARD` [184, 155] and convert the problem of distributed action recognition into that of subgraph classification, as is also done in [156]. The results are shown in Tab. 3.8, demonstrating the effectiveness of our method.

TABLE 3.7: Results of 3D object recognition tasks with DGCNN [167].

Types	**Tasks**	**# Classes**	***Pre-trained Performance***	***Reusing Performance***	
				Vanilla	**Ours**
Pre-trained Acc	`ModelNet40`	40	0.9327	N/A	N/A
Downstream Acc	`ShapeNet`	16	N/A	0.1545	**0.6090**

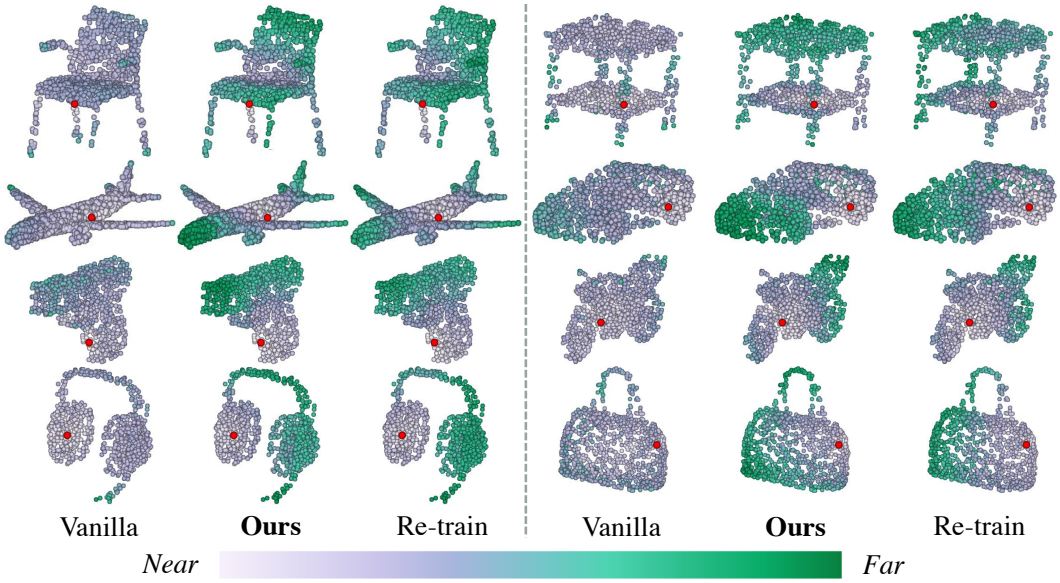


FIGURE 3.6: Visualization results of feature space structures, depicted as the distance between the red point and the rest of the others.

TABLE 3.8: Results of distributed action recognition with incremental time-series data streams and categories as downstream tasks.

Tasks	**Pre-trained Action Categories**								
	Up	ReLi	WaLe	TuLe	Down	Jog	Push	ReSt	**Acc**
Pre-trained Acc	0.9721	0.8563	0.9704	0.9731	0.9265	0.9875	0.9522	0.9229	0.9366

Tasks	**Downstream Action Categories**					
	ReSi	WaFo	TuRi	WaRi	Jump	**Acc**
Downstream Acc	0.7337	0.9574	0.6419	0.6893	0.8081	**0.7871**

3.5 Additional Details and Results

This section provides more details of the method pre-analysis, more implementation details of the proposed approaches, and various ablation studies for the experiments.

3.5.1 More Details of Method Pre-analysis

This chapter provides here more details and discussions on the method pre-analysis section, including the validation of adversarial reprogramming attacks on graph data, and the rationale of using task-adaptive aggregators in model reusing.

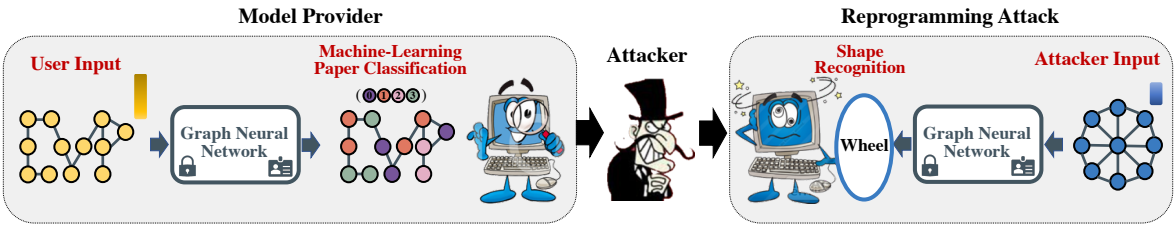


FIGURE 3.7: Illustrations of adversarial reprogramming attacks on graph data.

3.5.1.1 Adversarial Reprogramming Attacks on Graph Data

In this section, we further explain and validate Remark 3.2.1, which indicates the existence of adversarial reprogramming attacks on topological graphs:

To further illustrate Remark 3.2.1, we demonstrate in Fig. 3.7 an example adversarial reprogramming attack with graph data as inputs, where an attack redirects a machine-learning paper classification model to handle the intended topological shape recognition task, without informing the model provider.

Here, in Tab. 3.9, we show the results of the example attack in Fig. 3.7 with detailed explanations in the captions. Specifically, to obtain the pre-trained node property prediction model in Tab. 3.9, we use the `Cora` dataset [141] in training, with the detailed architectures provided in Tab. 3.10. For the attacker's designated task of topological shape recognition, we instead use a generated `MiniGC-Dataset` provided in the deep graph library [161]. The method of generating adversarial examples here is the node-level perturbation mentioned in [150], where we specifically add the generated adversarial perturbations to the raw node features.

From Tab. 3.9, it is noticeable that the original model provided by the model provider is easily repurposed to tackle the unseen graph-level task of shape recognition, whereas the original model function is to handle a node-level classification task. The results on shape recognition

TABLE 3.9: Results of adversarial reprogramming attacks on graph data, where the adversary repurposes a node classification model from the model provider to perform the adversary's designated shape recognition task.

Designated Tasks	**Before Attack**	**After Attack**	**Re-train from Scratch**
Node Property Prediction (*Model Provider*)	Acc: 87.69%	-	Acc: 87.69%
Shape Recognition (***Adversarial Attacker***)	-	**Acc: 80.00%**	Acc: 87.00%

TABLE 3.10: Detailed architectures used in Tab. 3.9.

Architecture	Layers	Attention Heads	Input	Hidden	Output
Pre-trained Model {Cora}	2	{8, 1}	1433	8	7

TABLE 3.11: Detailed architectures used in the section of "Rationale Behind MERE".

Architecture	Layers	Attention Heads	Input	Hidden	Output
Pre-trained Model {Cora-subset}	2	{8, 1}	1433	8	4

are also promising, with an accuracy of 80%, which is on par with that of re-training from scratch shown in the last column of Tab. 3.9. We also perform the experiment of directly feeding the raw data of shape recognition, without adding adversarial perturbation, to the pre-trained model. Expectedly, the accuracy of such a vanilla manner is only 0.16%, which demonstrates the effectiveness of the adversarial perturbations.

In aggregate, the observation in Tab. 3.9 validates that adversarial reprogramming attacks not only exist in the Euclidean image domain, but are also effective in the non-Euclidean graph domain. This motivates our idea that flips the role of adversarial reprogramming attack on its head, by paradoxically converting their function as threats to machine learning systems to resource-efficient model reusing where only limited pre-trained models are available.

3.5.1.2 Aggregation Matters for Reusing

We provide in this section more explanations and discussions on Remark 3.2.2, which suggest the importance of adaptive aggregation methods in reusing GNNs.

To validate Remark 3.2.2, we perform a pilot experiment, by dividing the Cora dataset into two subsets, with the first subset containing four classes and the second one including three node categories. We pre-train a model on the first Cora subset and obtain a frozen GNN that can predict the first four classes in Cora. Then, we aim to reuse this pre-trained model to handle the task of node classification with the last three separate and unseen categories. The network architecture of the pre-trained model on the first subset of Cora is demonstrated in Tab. 3.11.

TABLE 3.12: Vanilla GNN reusing results on the `Cora-subset` dataset with various aggregation behaviors. The detailed network architecture for producing the results can be found in Tab. 3.11.

Various Aggregation Methods	Mean	Max	Min	Std	Var	Skewness	Kurtosis	Hyperskewness
Downstream Performance (Acc)	0.4420	0.4203	0.5000	0.2536	0.2826	0.4022	0.2572	0.3696

TABLE 3.13: Summary of the fourteen datasets. Additional dataset statistics for point cloud classification are shown in Tab. 3.14.

Names	**Task Descriptions**	**Feature Dimensions**	**Nodes**	**Edges**	**# Graphs**
1. `Cora` [141]	Machine-Learning Paper Classification	1,433	2,708	5,429	1
2. `Citeseer` [141]	Computer-Science Paper Classification	3,703	3,327	4,732	1
3. `Pubmed` [141]	Diabete-related Publication Classification	500	19,717	44,338	1
4. `ogbn-arxiv` [160, 56]	Subject Area Prediction of arXiv Papers	128	169,343	1,166,243	1
5. `Amazon Computers` [112]	Computer-Product Category Prediction	767	13,752	574,418	1
6. `Amazon Photo` [112]	Photo-Product Category Prediction	745	7,650	287,326	1
7. `QM7b` [117]	Molecule Property Regression	1	111,180	1,766,366	7,211
8. `ogbg-molesol` [174, 56]	Molecule Property Regression	9	14,991	30,856	1,128
9. `PROTEINS` [6]	Protein Property Prediction	3	43,471	205,559	1,113
10. `ogbg-molbace` [174, 56]	Molecule Property Classification	9	51,577	111,539	1,513
11. `ogbg-molbbbp` [174, 56]	Molecule Property Classification	9	49,068	105,842	2,039
12. `WARD` [184, 155]	Distributed Human Action Recognition	125	3,521,550	15,846,975	35,2155
13. `ModelNet40` [175]	3D Object Recognition	3	12,603,392	252,067,840	12,311
14. `ShapeNet` [197]	3D Object Recognition	3	17,286,144	345,722,880	16,881

Here, our goal is to validate the influence of aggregators in model reusing. As such, we adopt the simplest vanilla reusing method, by just using the pre-trained model to directly handle the novel three downstream categories, but changing the aggregation behaviors. The corresponding results of various aggregation methods are shown in Tab. 3.12, where we specifically use eight prevalent aggregation methods as examples as mentioned in [24].

Notably, the results in Tab. 3.12 show that various aggregation manners can lead to totally distinct model reusing results, where the min aggregation method is optimal for our task of `Cora-subset`. Such observation leads to our idea of using the task-adaptive aggregation method to enhance the model capability in different downstream tasks.

3.5.2 Dataset Statistics and Descriptions

We provide in Tab. 3.13 the statistics of several graph benchmarks.

Specifically, the first three datasets, *i.e.*, Cora, Citeseer and Pubmed [141], are all citation network datasets, for the purpose of single-label node classification. Besides, Amazon

TABLE 3.14: Detailed dataset statistics of the ShapeNet part dataset [197].

	Total	Aero	Bag	Cap	Car	Chair	Earphone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skateboard	Table
# shapes	16,881	2,690	76	55	898	3,758	69	787	392	1,547	451	202	184	283	66	152	5,271
# shape labels	16	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Computers and Amazon Photo datasets are, in fact, the segments of the Amazon co-purchase graphs introduced in [112]. Moreover, ogbn-arxiv [160, 56] contains a single directed graph that represents the citation network among all the computer science papers posted in arXiv. Furthermore, the adopted ogbg-molesol, ogbg-molbace, and ogbg-molbbbp [174, 56] are molecular property prediction datasets. Also, QM7b [117] aims at molecular property regression, containing 7,211 molecules with totally 14 regression targets. The PROTEINS dataset [6], on the other hand, focuses on protein classification, such as enzymes or non-enzymes.

We also adopt a distributed human action recognition dataset with wearable motion sensor networks, termed as WARD [184, 155], to validate the proposed methods. There are five sensors that are exploited to capture the data in WARD. Every sensor specifically yields 5 data streams and in total 5×5 data streams are generated.

For the task of point cloud classification, we leverage the ModelNet40 dataset [175] as well as the ShapeNet part dataset [197]. Specifically, ModelNet40 has 12,311 CAD models with 40 man-made object categories, of which 9,843 CAD models are used for training and 2,468 models are for testing. For each CAD model, we specifically sample 1,024 3D points, as also done in [167]. Also, the ShapeNet part dataset involves 16,881 3D shapes from 16 categories. For each 3D shape, we also sample 1,024 points. Detailed class-by-class statistics of ShapeNet part dataset are provided in Tab. 3.14.

3.5.3 Additional Results on Heterogeneous Node Property Prediction

In this section, we provide additional results of using heterogeneous data reprogramming to tackle heterogeneous node property prediction.

Implementation Details. In total, we use five datasets, `Citeseer`, `Cora`, `Pubmed`, `Amazon Computers`, and `Amazon Photo` as examples to validate the effectiveness

of the proposed approaches. In particular, we'd like to clarify that the different pre-trained models on various datasets have a maximum number of output dimensions, meaning that the number of classes in the downstream datasets cannot exceed that of the pre-trained one. So, since we use `Citeseer` with six categories as the pre-trained task, we have to use the corresponding six categories for `Cora`, `Amazon Computers`, and `Amazon Photo`.

For the sake of the consistency, when we use the datasets of `Cora`, `Amazon Computers`, and `Amazon Photo` everywhere in this work, we consistently only use the first six target classes of the corresponding datasets, as shown in Tab. 3.15. For the task of `Pubmed` with only three classes, we predict the full target three classes, by simply using the corresponding three output neurons in the pre-trained `Citeseer` model for the final prediction. Nevertheless, it is indeed possible to leverage the technique of adaptive prototype learning to alleviate the dilemma of such output limits. However, due to the page limit, we have to elaborate on this part in our future work.

For the dataset splittings of `Citeseer`, `Cora`, `Pubmed`, we use the splitting protocol in [17]. Specifically, for Cora, we use 1208 samples for training, 500 samples for validation, and 1000 samples for testing; for Citeseer, we use 1827 samples for training, 500 samples for validation, and 1000 samples for testing; for Pubmed, 18217 samples are used for training, 500 samples are used for validation, and 1000 samples are used for testing. For `Amazon Computers` and `Amazon Photo`, since there is no standard splitting protocol, in our experiment, we randomly split these two datasets with the ratio of TrainingSet : ValidationSet : TestingSet $= 6:2:2$.

At the pre-training stage, the learning rate is set to 0.005. At the reusing stage, the ascent step size for optimizing the padded features is set to 0.0001 with a weight decay of 5×10^{-4} by default. We use the Adam optimizer for both stages. The results are obtained by computing the average of 20 independent runs. The network architectures are given in Tab. 3.15, which follow the official architecture design in deep graph library [161] without specific modifications.

Ablation Studies. In Tab. 3.16 and Tab. 3.17, we demonstrate the results of the ablation studies with various pre-trained GNNs, corresponding to Tab. 3.2. In particular, we use `Cora`

TABLE 3.15: Detailed network architectures for producing the results in Tab. 3.2 and also those in Tabs. 3.16 and 3.17.

Pre-trained Architectures	**Layers**	**Attention Heads**	**Input**	**Hidden**	**Output**	**Parameter Sizes**
`Citeseer`	2	{8, 1}	3703	8	6	474,892
`Cora`	2	{8, 1}	1433	8	6	184,332
`Amazon Computers`	2	{8, 1}	767	8	6	99,084
`Amazon Photo`	2	{8, 1}	745	8	6	96,268

TABLE 3.16: Ablation studies of using different pre-trained GNNs, corresponding to Tab. 3.2. Here, we reuse a pre-trained node classification model on `Cora` to handle the tasks of `Amazon Computers` and `Amazon Computers` with heterogeneous feature dimensions.

Pre-trained Task **Heterogeneous** Task Type	`Cora` ***Machine-Learning Paper Classification***			
Pre-trained Results	*Accuray*: 0.9121			
Downstream Tasks **Heterogeneous** Task Types	`Amazon Computers` ***Computer-Product Category Prediction***		`Amazon Photo` ***Photo-Product Category Prediction***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*Accuray*: 0.1497	*Accuray* : **0.8792**	*Accuray*: 0.1354	*Accuray*: **0.8785**
Re-training from Scratch	*Accuray*: 0.9485		*Accuray*: 0.9561	

and `Amazon Computers` as the pre-trained tasks for Tab. 3.16 and Tab. 3.17, respectively. It is noticeable that the proposed method still achieves promising results with different pre-trained models. For example, the three results of the downstream `Amazon Photo` task with the pre-trained models `Citeseer` (Tab. 3.2), `Cora` (Tab. 3.16), and `Amazon Computers` (Tab. 3.17) are all equally encouraging, showing that the proposed *MetaFP* and *ReAgg* methods make it possible for resource-efficient model reusing under the scenarios of having only a limited number of pre-trained models.

TABLE 3.17: Ablation studies of using a pre-trained computer-product category prediction model to tackle `Amazon Computers` and `Pubmed` with various input dimensions, corresponding to Tab. 3.2.

Pre-trained Task **Heterogeneous** Task Type	`Amazon Computers` ***Computer-Product Category Prediction***			
Pre-trained Results	*Accuray*: 0.9485			
Downstream Tasks **Heterogeneous** Task Types	`Amazon Photo` ***Photo-Product Category Prediction***		`Pubmed` ***Diabete-Publication Classification***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*Accuray*: 0.2012	*Accuray*: **0.9161**	*Accuray*: 0.4270	*Accuray* : **0.8320**
Re-training from Scratch	*Accuray*: 0.9561		*Accuray*: 0.8840	

TABLE 3.18: Network architectures for heterogeneous downstream graph classification and regression tasks, corresponding to Tab. 3.4 and also Tabs. 3.19 and 3.20.

Pre-trained Architectures	Layers	Attention Heads	Input	Hidden	Output	Parameter Sizes
`Cora`	2	{8, 1}	1433	8	6	184,332
`Amazon Computers`	2	{8, 1}	767	8	6	99,084
`Amazon Photo`	2	{8, 1}	745	8	6	96,268

3.5.4 Additional Results on Heterogeneous Graph Classification and Regression

In this section, we demonstrate more results of exploiting the heterogeneous data reprogramming method of *MetaFP* to handle the heterogeneous cross-level graph classification and regression tasks.

Implementation Details. To demonstrate the effectiveness of the proposed methods for heterogeneous cross-level graph analysis, here we use node classification models as the pre-trained ones, and reuse them to handle the task of heterogeneous graph classification and regression. The network architectures of the pre-trained node property prediction models in Tab. 3.4 and Tabs. 3.19 and 3.20 are provided in Tab. 3.18. Also, to address the issue of different output dimensions of node-level and graph-level tasks, we adopt the slimmable strategy in dynamic networks, *i.e.*, simply using the part of the output neurons to generate the prediction results, and ignoring the other extra unaligned output dimensions.

The detailed dataset statistics of the various datasets used in this section can be found in Sect. 3.5.2. In the pre-training phase, the experimental settings are the same as those in Sect. 3.5.3, *i.e.*, with a learning rate of 0.005 and the Adam optimizer. During model reusing, we set the ascent step size as 0.0001 with a weight decay of 5×10^{-4}.

Ablation Studies. We show in Tab. 3.19 and Tab. 3.20 the ablation study results of various pre-trained models for heterogeneous downstream graph classification and regression tasks, corresponding to Tab. 3.4. As can be observed from Tabs. 3.19, 3.20 and Tab. 3.4, the proposed *MetaFP* approach delivers gratifying results with all these three different pre-trained tasks of `Amazon Computers`, `Amazon Photo`, and `Cora`. Such observation validates

TABLE 3.19: Ablation studies of reusing various pre-trained GNNs, corresponding to Tab. 3.4. Here, we pre-train a model on `Amazon Computers` and then reuse it to tackle the graph regression task of `QM7b` as well as the graph classification task of `PROTEINS`.

Pre-trained Task **Heterogeneous** Task Type	`Amazon Computers` ***Computer-Product Category Prediction***			
Pre-trained Results	*Accuray*: 0.9485			
Downstream Tasks **Heterogeneous** Task Types	`QM7b` ***Molecule Property Regression***		`PROTEINS` ***Protein Property Prediction***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*MAE*: 13.1409	*MAE* : **2.4634**	*Accuray*: 0.5268	*Accuray*: **0.6250**
Re-training from Scratch	*MAE*: 0.7264		*Accuray*: 0.6964	

TABLE 3.20: Ablation studies of reusing a pre-trained node classification model on `Amazon Photo` to handle the unseen graph-level regression and classification tasks of `QM7b` and `PROTEINS`, corresponding to Tab. 3.4.

Pre-trained Task **Heterogeneous** Task Type	`Amazon Photo` ***Photo-Product Category Prediction***			
Pre-trained Results	*Accuray*: 0.9561			
Downstream Tasks **Heterogeneous** Task Types	`QM7b` ***Molecule Property Regression***		`PROTEINS` ***Protein Property Prediction***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*MAE*: 24.1829	*MAE* : **2.3093**	*Accuray*: 0.3304	*Accuray*: 0.6071
Re-training from Scratch	*MAE*: 0.7264		*Accuray*: 0.6964	

the resource-efficient property of the proposed method in Sect. 3.5.3 again: getting rid of the restriction on well-provided pertinent pre-trained models.

3.5.5 Additional Results on Homogenous Node Property Prediction

In this section, we illustrate additional results of leveraging the homogenous data reprogramming method of *EdgSlim* to deal with the task of homogenous node property prediction.

Implementation Details. The architecture details for Tab. 3.5 and Tab. 3.22 are provided in Tab. 3.21. In particular, Architecture-ogbn-arxiv-V1 and Architecture-ogbn-arxiv-V2 represent the two distinct pre-trained architectures used in the ablation studies in Tab. 3.22.

TABLE 3.21: Network architectures used in Tab. 3.5 and Tab. 3.22.

Pre-trained Architectures	**Layers**	**Attention Heads**	**Input**	**Hidden**	**Output**	**Parameter Sizes**
`Architecture-ogbn-arxiv-V1`	4	{8, 8, 8, 1}	128	8	20	35.75K
`Architecture-ogbn-arxiv-V2`	3	{8, 8, 1}	128	8	20	27.43K

TABLE 3.22: Ablation studies of reusing the pre-trained node classification models on `ogbn-arxiv` with various network architectures elaborated in Tab. 3.21.

Architectures	**Types**	**Model Parameter Sizes**	***Pre-trained Performance***	***Reusing Performance***	
				Vanilla	**Ours**
`Architecture-ogbn-arxiv-V1`	Pre-trained Acc	35.75K	0.7884	N/A	N/A
`Architecture-ogbn-arxiv-V1`	Downstream Acc	35.75K	N/A	0.2334	**0.6034**
`Architecture-ogbn-arxiv-V2`	Pre-trained Acc	27.43K	0.7849	N/A	N/A
`Architecture-ogbn-arxiv-V2`	Downstream Acc	27.43K	N/A	0.2191	**0.5507**

The dataset details can be found in Tab. 3.13. Here, we divide the `ogbn-arxiv` dataset into two subsets, termed as `ogbn-arxiv-s1` and `ogbn-arxiv-s2`, where each subset contains 20 separate categories in the full `ogbn-arxiv` dataset. The pre-trained task is to predict the 20 classes in `ogbn-arxiv-s1`, whereas the downstream task is to classify the distinct 20 categories in `ogbn-arxiv-s2`. During pre-training, we use the Adam optimizer, with a learning rate of 0.005 and a weight decay of 5×10^{-4}, which are the same as other datasets without specific modifications or hyperparameter tuning.

Ablation Studies. Tab. 3.22 demonstrates the results of the ablation studies of different pre-trained architectures. As can be observed from the last column of Tab. 3.22, the proposed *EdgSlim* leads to promising downstream performance without re-training or fine-tuning and outperforms the results of vanilla reusing by at least 30%. Also, the results are obtained at a low computational cost, with only three epochs. The physical edge elimination time is even less than one second for both architectures on a single NVIDIA GeForce RTX 2080 Ti GPU.

3.5.6 Additional Results on Homogenous Graph Classification and Regression

In this section, we give more results of applying the proposed homogenous data reprogramming approach of *MetaGP* to tackle the downstream tasks of homogenous node property prediction.

Implementation Details. To demonstrate the effectiveness of the proposed *MetaGP* method under the scenarios of homogenous graph classification and regression with homogenous input dimensions, we specifically use the three datasets of `ogbg-molbace`, `ogbg-molbbbp`,

TABLE 3.23: Network architectures for producing the results in Tab. 3.6 and also those in Tab. 3.24.

Pre-trained Architectures	Layers	Attention Heads	Output Layer	Input	Hidden	Output	Parameter Sizes
ogbg-molbace	4	{1, 1, 1}	Linear	9	256	1	69.63K
ogbg-molbbbp	4	{1, 1, 1}	Linear	9	256	1	69.63K

TABLE 3.24: Ablation studies of reusing different pre-trained GNNs, corresponding to Tab. 3.6. Here, the pre-trained model is designated for ogbg-molbbbp, whereas ogbg-molbace and ogbg-molesol are considered as the two target downstream tasks.

Pre-trained Task **Homogenous** Task Type	ogbg-molbbbp ***Graph Classification***			
Pre-trained Results	*ROC-AUC*: 0.6709			
Downstream Tasks **Homogenous** Task Types	ogbg-molbace ***Graph Classification***		ogbg-molesol ***Graph Regression***	
Reusing Methods	Vanilla	Ours	Vanilla	Ours
Downstream Results	*ROC-AUC*: 0.4330	*ROC-AUC*: **0.5903**	*RMSE*: 7.979	*RMSE*: **2.8183**
Re-training from Scratch	*ROC-AUC*: 0.7734		*RMSE*: 1.300	

and ogbg-molesol that aim to classify or regress the graph properties, with more detailed statistics and descriptions in Sect. 3.5.2. The architecture details are provided in Tab. 3.23. In particular, different from the node classification task, the output layer of the graph-level tasks are linear layers. The learning rate setting is set to 0.005, with a weight decay of 5×10^{-4}, which is the same as other experiments. We use the Adam optimizer for pre-training.

Ablation Studies. We show in Tab. 3.24 the ablation studies of varying pre-trained models. In particular, instead of using ogbg-molbace as the pre-trained task as Tab. 3.6 does, here we perform extensive ablation studies by pre-training a GNN on ogbg-molbbbp and considering ogbg-molbace as the downstream tasks. The results in Tab. 3.24 demonstrate that the proposed *MetaGP* is competent for various-domain downstream tasks even with different pre-trained models. As such, our method is readily applicable to scenarios where there is a limited number of pre-trained GNNs.

TABLE 3.25: Detailed network architectures for the task of 3D object recognition on ModelNet40 and ShapeNet.

Pre-trained Models	Layers	GNN Type	Feature Map Channels	MLPs
Architecture-ModelNet40	8	EdgeConv	[64, 64, 128, 256, 1024]	[512, 256, 40]

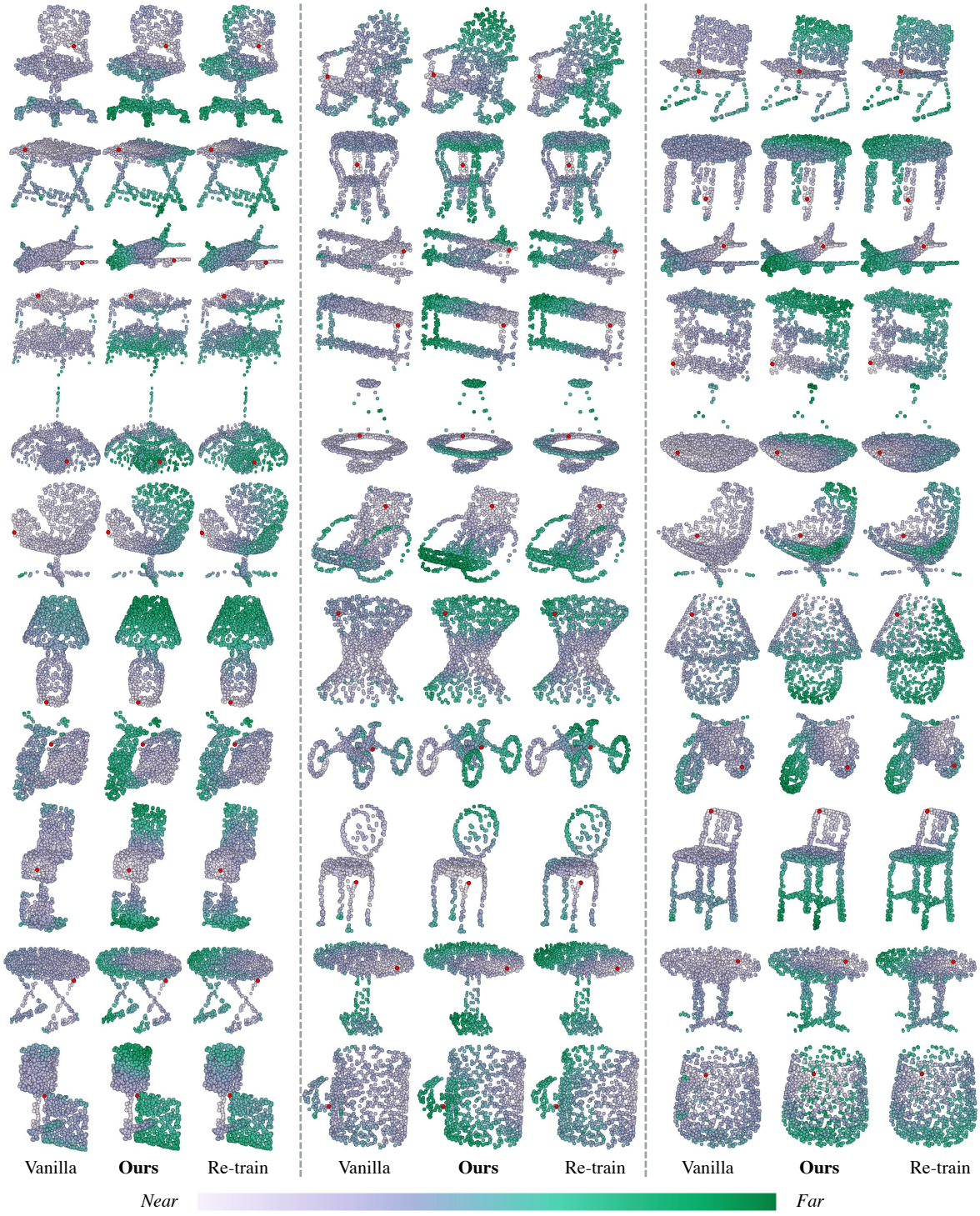


FIGURE 3.8: Visualization results of the structures of the feature space, depicted as the distance between the red point and the rest of the others. The visualized features are extracted from the intermediate layer of the models.

3.5.7 Additional Results on 3D Object Recognition

Implementation Details. In addition to node classification, graph classification, and graph regression tasks with citation networks and molecular graphs, we also conduct extensive experiments by reusing a GNN for 3D object recognition tasks. Here, we adopt two prevalent point cloud classification datasets, entitled `ModelNet40` and `ShapeNet`, of which the detailed statistics can be found in Tab. 3.13. We follow the official dataset splitting protocol in [175, 167], where 9,843 CAD models are used for training and 2,468 CAD models are for testing in pre-training. For each CAD model in both `ModelNet40` and `ShapeNet`, we sample 1,024 3D points from the mesh surfaces. We also rescale the associated point coordinates, as also done in [167]. The learning rate is set as 0.001 and the batch size is set to 16. We adopt the Adam optimizer [80]. The detailed architecture designs are summarized in Tab. 3.25. During the reusing stage, since `ModelNet40` contains 40 categories whereas `ShapeNet` has 16 classes, we simply use the first 16 output channels of the pre-trained `ModelNet40` as the output predictions for `ShapeNet`.

More Visualization Results. In Fig. 3.8, we show more qualitative results of reusing GNNs for point cloud classification, by visualizing the structures of the feature spaces, corresponding to Fig. 3.6. The column termed "Vanilla" in Fig. 3.8 contains the results of vanilla model reusing, corresponding to "Vanilla" in Tab. 3.7. Meanwhile, the columns termed "Ours" and "Re-train" in Fig. 3.8 indicate the results with the proposed *MetaGP* and those of re-training from scratch, respectively. It can be observed that the proposed method yields results that have a very similar feature structure to those of the cumbersome re-training ones, demonstrating the superiority of our approach.

3.6 Summary

This chapter introduces a novel GARE task for resource-efficient and generalized model reusing, tailored for GNNs. Our objective is to reuse a pre-trained GNN for diverse cross-level/domain downstream tasks, being rid of re-training or fine-tuning. To this end, we identified two key challenges on the data and model sides, respectively, and propose a suit

of three *data reprogramming* (DARE) and one *model reprogramming* (MERE) approaches to resolve the dilemma. Experiments on fourteen benchmarks across various domains, including node and graph classification, graph property regression, 3D object recognition, and distributed action recognition, demonstrate that the proposed methods lead to encouraging downstream performance, and meanwhile enjoy a low computational cost. In our future work, we will strive to generalize GARE to other domains.

CHAPTER 4

Model-Driven Efficient Learning with Knowledge Amalgamation

In continuation with the previous chapter's investigation on data-driven efficiency, this chapter delves into the problem of model-driven efficient learning by exploring a novel knowledge amalgamation paradigm specifically designed for GNNs. The objective of the proposed knowledge amalgamation is to train a multi-talented student GNN, without accessing human annotations, that "amalgamates" knowledge from a couple of teacher GNNs with heterogeneous architectures and handling distinct tasks. The student derived in this way is expected to integrate the expertise from both teachers while maintaining a compact architecture. To this end, this chapter proposes an innovative approach to train a slimmable GNN that enables learning from teachers with varying feature dimensions. Meanwhile, to explicitly align topological semantics between the student and teachers, this chapter introduces a *Topological Attribution Map (TAM)* to highlight the structural saliency in a graph, based on which the student imitates the teachers' ways of aggregating information from neighbors. Experiments on seven datasets across various tasks, including multi-label classification and joint segmentation-classification, demonstrate that the learned student, with a lightweight architecture, achieves gratifying results on par with and sometimes even superior to those of the teachers in their specializations.

4.1 Introduction

An increasing number of pre-trained deep neural networks (DNNs) have been generously released online for the sake of handy reproducibility [206]. As such, reusing these pre-trained

models to alleviate training effort or to enhance performance, has emerged as a trending research topic in recent years. The seminal work of Hinton *et al.* [53], for instance, first raises *Knowledge Distillation*, where a pre-trained teacher model is utilized to generate soft labels so as to learn a lightweight student model with competent performance. Following this student-teacher paradigm, many other distillation-based approaches have been applied to various domains and have demonstrated promising results [37, 138, 198, 205, 225].

Almost all existing approaches on knowledge transfer from pre-trained models have been focused on convolutional neural networks (CNNs), which take data in regular domains, like images, as input. Nevertheless, many other data samples take irregular forms and thereby resort to graph representations, calling for graph neural networks (GNNs). The work of [188], as the first attempt, generalizes knowledge distillation to GNNs, and introduces a customized approach tailored for irregular data. In spite of the improved performance, this approach is limited to the scenario where the student learns from a single teacher, and meanwhile holds a homogeneous architecture and tackles the same task as the teacher does.

In this chapter, we strive to make one step further towards knowledge transfer from pre-trained GNNs, by studying a novel *knowledge amalgamation* task. Our goal is to train a multi-talented student GNN, from a couple of pre-trained teacher GNNs with heterogeneous architectures and specializes in different tasks, for example one working on point cloud segmentation and the other on classification, as shown in Fig. 4.1. We further assume that, in the knowledge amalgamation process, no human annotations are available. The student learned in this way is anticipated to integrate both teachers' expertise yet comes with a compact size, making it competent for resource-constrained applications such as edge computing.

Nevertheless, such an ambitious goal is accompanied with challenges. The first challenge regards handling graph features with varying dimensions. Unlike CNNs that take as input grid-structured data with *fixed channel numbers*, such as RGB images, in our scenario, GNNs pre-trained on different datasets work with distinct feature dimensions. For example, nodes in the citation network dataset *Cora* have 1433 features, while those in *Citeseer* have 3703 features. The student GNN would therefore have to accommodate the diverse feature dimensions. The second challenge lies in encoding topological semantics of graphs. As

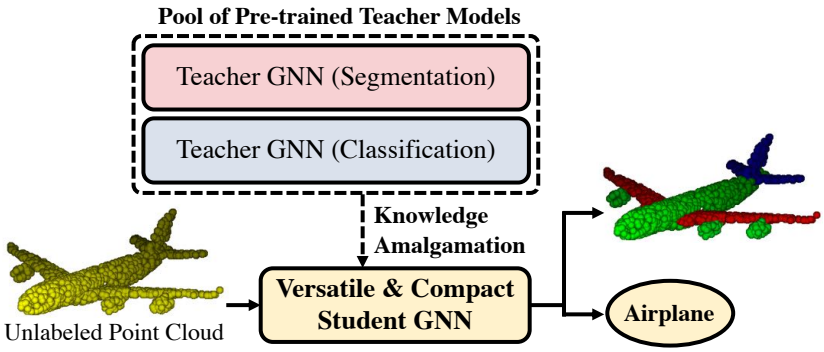


FIGURE 4.1: Illustrations of amalgamating knowledge from heterogeneous teacher GNN models. "Teacher GNN (Segmentation)" and "Teacher GNN (Classification)" are pre-trained point cloud part segmentation and classification models, respectively. Knowledge amalgamation aims to learn a multi-talented and lightweight student GNN from teacher GNNs without human annotations.

GNNs are designed to explicitly account for the topological information concealed in the graph data, aligning the topological semantics between teachers and the student emerges as a critical issue to be addressed in GNN knowledge amalgamation.

Towards this end, we propose a slimmable graph convolutional operation that enables adaptive activation or deactivation of layer channels; graph data of different input channels can therefore be simultaneously accounted for under one student model. Furthermore, we introduce *topological attribution map* (TAM), a general graph representation scheme to highlight structural saliency in terms of information propagation from neighbors. The derived student model is enforced to produce a TAM that resembles those from the teachers, in which way the student imitates the teachers' fashions of aggregating features to the center node. Notably, TAM is free of data labels and readily applied to heterogeneous GNN architectures.

Our contribution is therefore a novel GNN-based knowledge amalgamation approach to train a versatile student model that covers the specialties from heterogeneous-task teachers, without human annotations. This is typically accomplished through a slimmable graph convolutional operation to accommodate varying-dimension features from teachers, together with a TAM scheme for learning the teachers' topological semantics. We evaluate the proposed method on four different tasks across various domains, including single- and multi-label node classifications, 3D object recognition, and part segmentation. Experimental results

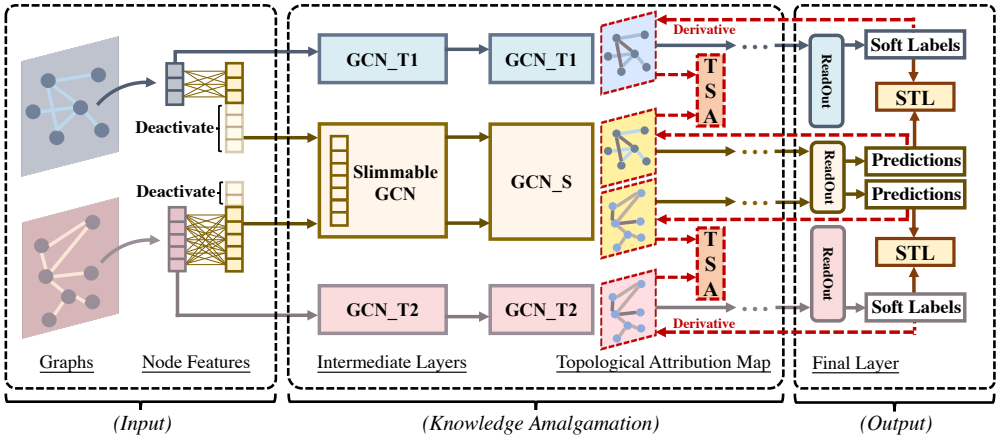


FIGURE 4.2: The overall framework of the proposed knowledge amalgamation method tailored for GNNs. For illustration, we take two pre-trained teacher GCNs as an example. On the input side, the dimensions of input node features would vary with different graph samples. **GCN_T1**, **GCN_S** and **GCN_T2** represent the graph convolutional layers from pre-trained teacher #1, lightweight student, and pre-trained teacher #2, respectively. **TSA** and **STL** denote the proposed topological semantics alignment module and the soft target learning module, respectively. The topological attribution map is obtained by computing the edge gradients of the constructed unary edge features, as explained in Sect. 4.3.3.

demonstrate that, the learned student GNN model is competent to handle all different tasks of the heterogeneous teachers, sometimes with a performance even superior to those of the teachers, and meanwhile comes at a significant reduction in computational cost.

4.2 Problem Definition

The problem we aim to address here is to learn a versatile and lightweight student GNN model, with only unlabeled graph data, that amalgamates topology-aware knowledge from multiple task-wise heterogeneous teachers. Specifically, assume that we are given N pre-trained GNN models $G = \{g_1, g_2, \cdots, g_N\}$, each of which specializes in different tasks, such as paper classifications on specific topics [141] or predictions of various protein functions [226]. We use $T(g_i)$ to represent the specific task handled by teacher g_i. The goal of knowledge amalgamation is then formulated as learning a student GNN model g_s that has the following three properties:

- The student g_s covers the expertise of all heterogeneous teachers.
- The model size of the student is smaller than the sum of teachers, preferably even smaller than a single teacher.
- Learning of g_s requires only raw graph data without human-labeled annotations.

The target student GNN model is therefore expected to be capable of simultaneously handling heterogeneous tasks, and meanwhile more portable for deployment on the mobile-terminal side.

Also, for different pre-trained teachers g_i, we impose *no constraints* on g_i's architectures being the same, meaning that g_i can have diversified layer numbers, different feature dimensions, or even distinct layer mechanisms, such as graph convolutional layers by Kipf *et al*. [82] and graph attention layers by Veličković *et al*. [153].

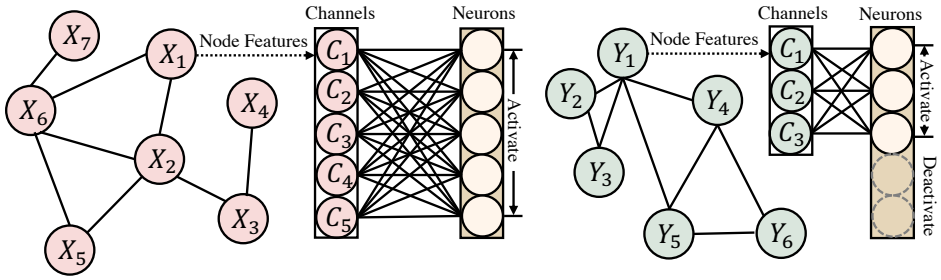
4.3 Proposed Method

Towards addressing the proposed problem of knowledge amalgamation, we introduce the proposed dedicated approach tailored for GNN models. In what follows, we start by giving an overview of the proposed method, and then detail the key modules. Finally, we propose a dedicated training strategy that trains the student GNN intertwined with teacher GNNs.

4.3.1 Overview

The overall workflow of the proposed method is shown in Fig. 4.2. The task of knowledge amalgamation imposes three major challenges, respectively on input data, intermediate features, and output labels. The challenge on the input side concerns handling multiple teacher GNNs with different feature dimensions. This dilemma is solved by equipping the student with the proposed slimmable graph convolutions (Fig. 4.2 *(Input)*).

The second challenge lies in the effective extraction and transfer of topological information from teachers. In our proposed approach, this issue is tackled by the proposed topological semantics alignment module (Fig. 4.2 *(TSA)*).



(a) Input Graph #1 (b) Input Graph #2

FIGURE 4.3: Illustrations of the proposed slimmable graph convolutional operation, where X and Y denote graph nodes. The neurons in multi-layer perceptrons (MLPs) of GNN are adaptively activated or deactivated based on the feature dimensions of the input graph data.

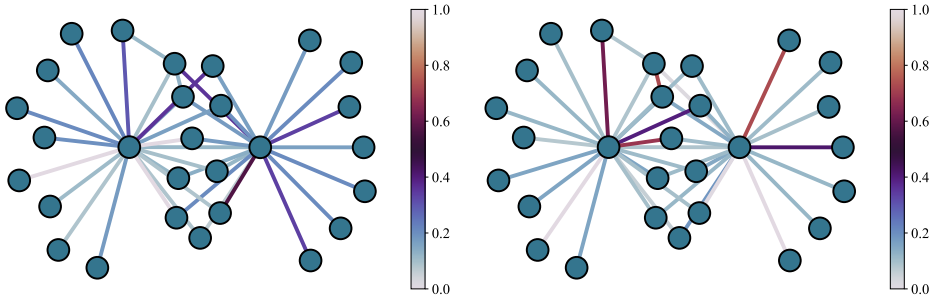
The last challenge relates to the lack of human-labeled annotations: how to obtain supervision information from unlabeled graph data. We address this issue by explicitly imitating the soft predictions of heterogeneous teachers (Fig. 4.2 *(STL)*), as is also done in CNN-based model reusing technique [53].

Therefore, in what follows, we put our emphasis upon the slimmable graph convolutional modules and the topological semantics alignment module, both of which are specific to the task of GNN model reusing.

4.3.2 Slimmable Graph Convolution

On the input side, unlike CNNs that always receive grid-like RGB images with constant channel numbers, GNN models, depending on the handled tasks, vary in the feature dimensions of input nodes. Taking the three popular paper-citation datasets, *Cora*, *Citeseer* and *Pubmed* as examples [141], all of these three datasets contain publications as graph nodes. Nevertheless, they contain distinct channel numbers for each node: 1433 for *Cora*, 3703 for *Citeseer*, and 500 for *Pubmed*. This challenge of diversified feature dimensions makes it infeasible to simply use a naive GNN architecture for the target multi-talented student model.

To solve this dilemma, we devise a dedicated slimmable graph convolutional layer, where the layer channels can be adaptively activated or deactivated depending on different input



(a) TAM of Teacher GNN #1 (b) TAM of Teacher GNN #2

FIGURE 4.4: Visualizations of the scaled topological attribution map (TAM) of two teacher GNNs given the same input graph data. As an example, two teachers here are pre-trained multi-label node classification models that handle a different set of classes. Colors encode the importance of each connection for the corresponding task of each teacher.

feature dimensions, as shown in Fig 4.3. To further illustrate the proposed slimmable graph convolutional layer, we take the task of node classification as an example.

Assume that we have separate input graph nodes X_i and Y_j from different graphs with C_i and C_j feature dimensions ($C_i \neq C_j$) to concurrently account for. Firstly, before training, we set a maximum channel number C_{max} for the proposed slimmable graph convolutions, so as to define the shape of weights in GNN layers. Then, given input nodes X_i with the node feature dimension of C_i, the slimmable graph convolution adaptively deactivates the $|C_{max} - C_i|$ neurons and uses only the C_i-channel filter to deal with X_i. For the processing of the node Y_j with C_j feature channels, a similar scheme is also applied, where the slimmable graph convolution dynamically switches to C_j-channel filter to manage the corresponding input node of Y_j.

In knowledge amalgamation, by replacing the first layer with slimmable graph convolutional layer, the student GNN can simultaneously handle graph samples with varying input feature dimensions; while also equipped with slimmable graph convolutions in the intermediate layers, the student GNN model can also trade off between accuracy and latency at runtime, by switching between models with different numbers of active layer channels, thus making it possible to adapt the learned student model across different devices with limited response time budgets.

4.3.3 Topological Semantics Alignment

Unlike conventional convolutional layers that only receive grid-structured data as input and generate high-level semantic representations, graph convolutional layers are designed to process the graph data, either in the form of grid or non-grid structures. To this end, the intrinsical mechanism of graph convolutions is to generate representations for each node by collectively aggregating its own features and its neighboring nodes' features. As a result, the generated feature maps from graph convolutional layers contain both the topological properties of the input graph and also the high-level node content information. Simply applying prior CNN-based model reusing techniques, regardless of topological connections among different nodes, for GNN-based knowledge amalgamation, will inevitably lead to lossy knowledge transfer [177].

Towards addressing this challenge, the key issue to be considered is: how to derive a structure-aware graphical representation, tailored for aligning the concealed topological information between teachers and the student. One possible solution to this issue could be using the pairwise feature distance between every two connected nodes as the potential structure-aware representation to perform alignment between the student and teacher, as is done in [188]. This solution might be feasible for topology-aware knowledge transfer from a *single teacher*.

However, this possible graphical representation does not fit our case of amalgamating multiple streams of knowledge from heterogeneous teachers. Take the amalgamation of multi-label node classification models as an example, where each teacher handles a separate set of classes. The goal of the student GNN is to concurrently deal with all the classes covered by the teachers. In this case, given the same graph as input, different teachers would have distinct aforementioned possible representations, whereas the student would derive only one single representation. As a result, simultaneously aligning these multiple distinct representations of teachers with a single student GNN will make the learning of different knowledge compete with each other, which will be validated in the experiments. This competitive situation is contradictory to our goal, where we expect that the learning of different teachers' knowledge could potentially benefit and cooperate with each other for improved performance.

Motivated by this observation, we propose a novel topological representation, termed as *topological attribution map* (TAM), for the structural semantics alignment in knowledge amalgamation from heterogeneous teacher GNNs. Specifically, the proposed TAM is derived by computing the gradients of the given GNN's output class scores with respect to the adjacency matrix, as shown in Fig. 4.4. As a result, the obtained TAM contains the structural saliency in propagating information from neighbor nodes, indicating the importance of each individual connection on the final GNN predictions. Compared with the aforementioned possible representation, the design of the proposed TAM offers two benefits in knowledge amalgamation:

- The proposed TAM can be readily applied to heterogeneous GNN architectures, including the models with distinct aggregating mechanisms like graph convolutional network (GCN) [82] and graph attention network (GAT) [153], and also those with different layer numbers and channels.
- The proposed TAM can be extracted in a teacher-aware manner, meaning that a student GNN can derive multiple TAMs, which correspond to different teachers that handle separate classes. Specifically, this is achieved by using the specific subset of class scores, corresponding to the task of each teacher, to compute the teacher-specific TAMs. This manner alleviates the aforementioned competitive dilemma in amalgamating multiple teacher GNNs.

The workflow of computing the proposed TAM is given as follows. Consider a graph represented by a tuple $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of unordered vertices and $\mathcal{E}$ represents the set of edges connecting different vertices $v \in \mathcal{V}$. Let $\mathcal{A} \in \mathbb{R}^{n \times n}$ denote the adjacency matrix, where n is the number of graph nodes. Given an input graph $\mathcal{G}_0$ and a GNN model g, the proposed TAM representation $\mathcal{F}$ can be generally computed as:

$$\mathcal{F} = \left.\frac{\partial \mathcal{P}}{\partial \mathcal{A}}\right|_{\mathcal{G}_0} \in \mathbb{R}^{n \times n}, \quad \mathcal{P} = g(\mathcal{G}_0), \tag{4.1}$$

where $\mathcal{P}$ is the predicted class scores with the input $\mathcal{G}_0$.

Algorithm 1 GNN-based Knowledge Amalgamation from Heterogeneous Teachers

Input: $\mathcal{T} = \{\mathcal{T}_i\}_{i=1}^{\mathcal{M}}$: $\mathcal{M}$ trained teacher GNNs; $\mathcal{G} = \{\mathcal{G}_k\}_{k=1}^{\mathcal{K}}$: unlabeled graph samples.
Output: $\mathcal{S}$: Target versatile and lightweight student GNN.

Set C_{max} as the maximum feature dimension in $\mathcal{G}$;
Initilize student model $\mathcal{S}$;
for $m = 1$ to $\mathcal{M}$ **do**
    // Obtain topological representation and soft labels from Teacher $\mathcal{T}_m$
    Feed $\mathcal{G}$ with matched input dimensions into $\mathcal{T}_m$;
    Compute topological representation $\mathcal{F}^{\mathcal{T}_m}$ by Eq. 4.1;
    Compute the soft labels $\mathcal{P}^{\mathcal{T}_m}$ from the output layer of teacher $\mathcal{T}_m$;
    // Obtain topological representation and output predictions from Student $\mathcal{S}$
    Feed the same $\mathcal{G}$ into $\mathcal{S}$ and process $\mathcal{G}$ with *slimmable graph convolutions* in $\mathcal{S}$;
    Compute topological representation $\mathcal{F}^{\mathcal{S}}$ by Eq. 4.1;
    Compute soft labels $\mathcal{P}^{\mathcal{S}}$ from the output layer of $\mathcal{S}$;
    // Compute two losses
    Compute $\mathcal{L}_{topology}^{\mathcal{T}_m}$ from $\mathcal{F}^{\mathcal{T}_m}$ and $\mathcal{F}^{\mathcal{S}}$ by Eq. 4.2;
    Compute $\mathcal{L}_{soft}^{\mathcal{T}_m}$ from $\mathcal{P}^{\mathcal{T}_m}$ and $\mathcal{P}^{\mathcal{S}}$;
end for
Compute total loss over $\{\mathcal{T}_i\}_{i=1}^{\mathcal{M}}$ by Eq. 4.3;
Optimize $\mathcal{S}$ with Adam for epochs.

Based on Eq. 4.1, given a set of pre-trained teacher GNNs $\{\mathcal{T}\}$, we propose a topological semantics alignment loss for knowledge amalgamation:

$$\mathcal{L}_{topology}^{\mathcal{T}_i} = \| \frac{\partial \mathcal{P}_{d_{\mathcal{S}} \cap d_{\mathcal{T}_i}}^{\mathcal{S}}}{\partial \mathcal{A}} - \frac{\partial \mathcal{P}_{d_{\mathcal{T}_i}}^{\mathcal{T}_i}}{\partial \mathcal{A}} \|, \tag{4.2}$$

where $d_{\mathcal{S}}$ and $d_{\mathcal{T}_i}$ represent the set of classes handled by the student $\mathcal{S}$ and the i-th teacher $\mathcal{T}_i$, respectively. $\mathcal{P}_{d_{\mathcal{S}} \cap d_{\mathcal{T}_i}}^{\mathcal{S}}$ represents a subset of the student's predicted class scores corresponding to those of the teacher $\mathcal{T}_i$, thus leading to a teacher-aware topological representation for knowledge amalgamation. The total topological alignment loss can then be computed as the sum of Eq. 4.2 over multiple teachers: $\mathcal{L}_{topology} = \sum_i \mathcal{L}_{topology}^{\mathcal{T}_i}$.

For implementations, there are two specific issues to be considered when using the naive computation method in Eq. 4.1 to obtain TAM. Firstly, there is a lack of a unified approach for computing the derivative of network outputs with respect to the adjacency matrix for heterogeneous GNN architectures like GCN and GAT. Different types of GNNs have different ways to incorporate the adjacency matrix in information aggregations, leading to inconsistent ways to obtain TAM.

Thus, we devise here a unified implementation method to compute TAM across various GNN architectures. Our idea is to first construct unary edges within the network based on the adjacency matrix, where the corresponding edge features are all equal to 1. The constructed unary edges are then involved in the graph computations by multiplying with the node features in aggregating features from neighbors. In this way, the proposed TAM in Eq. 4.1 can be equally obtained by directly computing the edge gradients of the constructed unary edges, of which the computation flow is shown as the red arrows in Fig. 4.2.

The other issue in implementations is related to the scale of the computed unary edge gradients. We experimentally observe that for some teacher GNNs, the obtained gradients could be large in magnitude, leading to a relatively large topological semantics alignment loss that would dominate other loss terms at the initial stage of training. As a result, the convergence speed of the student GNN would be slowed down. To address this issue, we propose to perform topological-aware edge gradient normalization before computing the topological semantics alignment loss. Specifically, we firstly compute the mean $\mu_i(\{\mathcal{F}\})$ and the standard deviation $\sigma_i(\{\mathcal{F}\})$ of the unary edge gradients around each center node v_i. The normalized unary edge gradients around v_i can then be obtained by computing $\frac{\{\mathcal{F}\}-\mu_i}{\sigma_i+\epsilon}$, where ϵ is a constant that avoids zero denominator.

4.3.4 Loss Function and Training Strategy

The total loss function for amalgamating knowledge from heterogeneous teachers can be formulated as:

$$\mathcal{L}_{total} = \mathcal{L}_{soft} + \lambda \mathcal{L}_{topology}, \tag{4.3}$$

where $\mathcal{L}_{soft}$ is the soft target loss computed as the mean squared error among the soft predictions from the student and the heterogeneous teachers, which is shown as the soft target learning (STL) module in Fig. 4.2. The definition of $\mathcal{L}_{topology}$ can be found in Sect. 4.3.3.

We also propose a training strategy, tailored for the proposed approach. As a whole, the detailed process of training a student GNN model from multiple heterogeneous teacher GNNs is

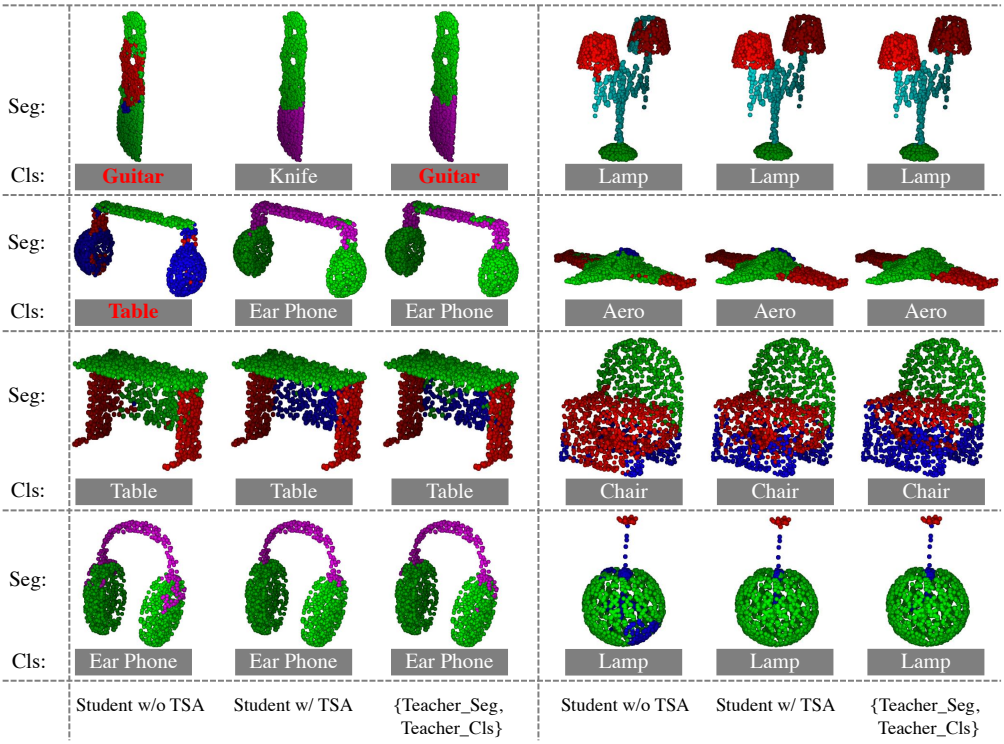


FIGURE 4.5: Visualization results of joint part segmentation (Seg) and classification (Cls). From left to right: the results of the learned student GNN without the proposed topological semantics alignment (TSA) module, those of the student with TSA, and the results of the two teacher GNNs. We use red texts to highlight the misclassified outputs. For some cases, our student GNN even achieves results superior to those of the teachers, as shown in the classification result of *Knife* and the segmentation results of *Ear Phone*.

concluded in Alg. 1. For each iteration, we accumulate the loss from all heterogeneous teachers and jointly optimize the student model, so as to make sure that the student simultaneously learns from all the teachers.

4.4 Experiments

To evaluate the performance of the proposed approach, we conduct experiments on seven publicly available benchmarks across various tasks, including node classifications, point cloud classifications and part segmentation. Here, we clarify that in the experiments, our goal is *not* to achieve the state-of-the-art performance on each benchmark, but rather transferring as much as knowledge from heterogeneous teachers.

4.4.1 Experimental Settings

Datasets and Implementation Details. We evaluate the proposed knowledge amalgamate method on seven datasets across various tasks. Specifically, for multi-label node classification, we use protein-protein interaction (PPI) dataset [226], containing biological graphs with nodes labeled with various protein functions. Each node can concurrently have several labels. We further divide PPI into two subsets, termed as PPI_Set1 and PPI_Set2 with 60 and 61 biological labels, respectively, which are used to train two corresponding teachers. The student GNN aims to amalgamate the knowledge from the two teachers, capable of predicting all 121 labels.

For the amalgamation of single-label node classification models, we adopt Amazon Computers (10 classes) and Amazon Photo (8 classes) datasets [112], where the nodes represent various goods, labeled by the corresponding product categories. We randomly split the dataset with a ratio of 2:2:6 for training, validation and testing, respectively. We also use three citation network datasets for single-label node classification, *i.e.*, Cora (7 classes), Citeseer (6 classes) and Pubmed (3 classes) [141]. The papers involved in these three datasets are all scientific publications, but with different subjects. We adjust the training/validation/testing split for the training of teachers in the supervised scenario, as is also done in [17].

TABLE 4.1: Results of amalgamating knowledge from multi-label node classifications GAT models, in terms of micro-averaged F_1 score. The obtained student achieves competitive performance compared with the teachers, yet with a moderately compact size.

Methods	Model Size	PPI_Set1	PPI_Set2
Teacher 1	11.61M	98.73	N/A
Teacher 2	11.56M	N/A	98.62
Student_{MTL+AT} [205]	14.57M	97.03	96.99
Student_{MTL+LSP} [188]	14.57M	97.27	97.22
Student_Ours (w/o TSA)	14.57M	97.95	97.98
Student_Ours (w/ TSA)	**14.57M**	**98.44**	**98.42**

TABLE 4.2: Results of amalgamating teachers with heterogeneous GNN architectures, in terms of micro-averaged F_1 score.

Type	Teacher 1 (GAT)	Teacher 2 (GAT)	Student (GAT)
Task	{PPI_1}	{PPI_2}	{PPI_1, PPI_2}
F_1 Score	98.73	98.62	**98.44 / 98.42**
Type	Teacher 1 (GCN)	Teacher 2 (GAT)	Student (GAT)
Task	{PPI_1}	{PPI_2}	{PPI_1, PPI_2}
F_1 Score	69.48	98.62	**70.01 / 98.01**
Type	Teacher 1 (GAT)	Teacher 2 (GCN)	Student (GAT)
Task	{PPI_1}	{PPI_2}	{PPI_1, PPI_2}
F_1 Score	98.73	63.62	**98.05 / 62.96**
Type	Teacher 1 (GCN)	Teacher 2 (GCN)	Student (GAT)
Task	{PPI_1}	{PPI_2}	{PPI_1, PPI_2}
F_1 Score	69.48	63.62	**69.64 / 62.51**

For knowledge amalgamation from point cloud classification and part segmentation models, we use the ShapeNet part dataset [197], containing $16,881$ shapes from 16 categories, annotated with 50 parts in total. The labeled categories and annotated parts are used to pre-train the teacher classification model and segmentation model.

For the unlabeled data sampling for the student GNN, we clarify that for a fair comparison with the pre-trained teacher GNNs, the training of the student in our experiments still uses the same training samples as those of the teachers, but without accessing ground truth labels, as explained in Sect. 4.2. Sampling more unlabeled graph samples from external datasets for training could further improve the performance of the learned student GNN.

We use heterogeneous architectures for the teachers and students in the task of node classifications, such as GCN [82] and GAT [153]. In particular, all the student GNNs are built with the proposed slimmable graph convolutional layer, so as to support graph inputs of varying feature dimensions. For the task of point cloud classification and part segmentation, we adopt the architecture of PointNet++ [127] for both the teachers and the student. The hyperparameter ϵ is set to 10^{-5}.

Comparison Methods. Since there are few existing knowledge amalgamation methods tailored for GNNs in the literature, we derive two possible solutions based on [205, 188]

TABLE 4.3: Results of amalgamating knowledge from point cloud classification and part segmentation models. The learned student GNN is even more compact than each of the teacher GNNs, yet competent to simultaneously handle all the tasks of teachers.

Method	**Model Size**	**mAcc** (Cls)	**mIoU** (Seg)	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate Board	Table
# shapes	–	–	–	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
Teacher_Cls	17.69M	97.83	N/A	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Teacher_PartSeg	17.01M	N/A	81.72	82.34	81.92	86.12	78.33	90.54	72.18	91.25	86.09	83.57	95.48	70.63	94.98	81.98	55.99	73.73	82.34
Student_{MTL+AT}	6.37M	97.06	77.58	80.40	72.50	81.84	75.58	89.66	64.24	89.92	85.02	82.29	95.39	55.94	93.20	78.20	44.13	70.48	82.52
Student_{MTL+LSP}	6.37M	97.30	77.79	81.04	74.07	79.21	75.97	89.32	59.89	90.15	86.73	82.61	95.40	55.97	93.29	78.80	47.04	72.67	82.52
Student_Ours (w/o TSA)	6.37M	97.23	77.76	80.62	73.08	83.41	76.07	89.54	60.37	90.37	85.19	81.74	95.17	55.32	91.82	79.50	46.94	72.44	82.57
Student_Ours (w/ TSA)	**6.37M**	**97.67**	**78.96**	**81.82**	**76.07**	**81.17**	**76.91**	**89.59**	**70.56**	**90.17**	**85.69**	**82.95**	**94.92**	**57.06**	**94.02**	**79.24**	**48.05**	**72.67**	**82.50**

and the multi-task learning (MTL) scheme for comparisons. Specifically, upon the idea of attention transfer method [205] and MLT scheme, we devise a "Student_{MTL+AT}" method that amalgamates knowledge by matching the attention maps with heterogeneous teacher GNNs. Furthermore, we take the local structure preserving (LSP) module from [188] and develop a "Student_{MTL+LSP}" knowledge amalgamation approach by replacing our topological semantics alignment module with LSP. Specifically, "Student_{MTL+LSP}" uses the pairwise feature distance between every two connected nodes as the structure-aware representation to perform topological alignment, as mentioned as the possible solution in Sect. 4.3.3.

TABLE 4.4: Results of amalgamating single-label node classification models, in terms of average classification accuracies (%).

	Teacher 1	Teacher 2	Teacher 3	Teacher 4	Teacher 5
Type	GCN	GCN	GAT	GAT	GAT
Task	{Computers}	{Photo}	{Cora}	{Citeseer}	{Pubmed}
Model Size	25.84K	25.06K	739.6K	1.901M	259.8K
Accuracy	89.36	92.48	87.90	79.00	85.70
	Student 1		Student 2		
Type	GCN		GAT		
Task	{Computers, Photo}		{Cora, Citeseer, Pubmed}		
Model Size	20.29K		1.450M		
Accuracy	88.81 / 91.79		87.10 / 77.30 / 83.20		

4.4.2 Results

Amalgamating Node Classification Models. Tab. 4.1 shows the results of amalgamating two pre-trained multi-label node classification model. In particular, to validate the effectiveness of the proposed TSA module, we conduct the ablation study by only using soft target learning for amalgamation, *i.e.*, setting $\lambda = 0$ in Eq. 4.3, which is termed as the method of "Student_Ours (w/o TSA)" in the table.

The student model learned with the proposed method, as shown in Tab. 4.1 (Student_Ours (w/ TSA)), achieves gratifying performance on par with that of the two teacher models, and meanwhile maintains a compact model size. Also, the results in the last two lines of Tab. 4.1 validate the effectiveness of the proposed TAM-based topological semantics alignment module, where Student_Ours (w/ TSA) outperforms Student_Ours (w/o TSA) by about 0.5 in F_1 score. The proposed knowledge amalgamation method also achieves favorable performance compared with the two derived comparison methods.

In Tab. 4.2, we also show the corresponding multi-label classification results by amalgamating knowledge from various types of GNN models. The notation "{PPI_1}" means that the teacher can only handle the task of PPI_Set1, while "{PPI_1, PPI_2}" indicates the capability of simultaneously handling the two tasks. Despite the heterogeneous types of trained teachers, the obtained student model still achieves encouraging results, sometimes even superior to those of the teacher, as shown in the sixth and the last rows of Tab. 4.2 for the specific task of PPI_1.

Tab. 4.4 shows the knowledge amalgamation results from pre-trained single-label node classification teacher GNNs. The first student model, Student 1 in Tab. 4.4, is obtained by amalgamating two teachers that handle the classification tasks of *Computers* and *Photos*, respectively. With a lightweight architecture which is even smaller than every single teacher, the obtained Student 1 still yields competitive results compared with those of the teachers. We also perform knowledge amalgamation on three teachers that deal with *Cora*, *Citeseer*, and *Pubmed*, respectively. The obtained Student 2 also delivers comparable results with those of teachers, yet maintaining a more compact size.

Amalgamating Point Cloud Classification and Segmentation Models. The results of amalgamating pre-trained classification and part segmentation teacher models are shown in Tab. 4.3. We also demonstrate in Fig. 4.5 the corresponding visualization results of the teachers and student. With the proposed TSA module, the learned versatile student gains boost by at least 0.4 in mean class accuracy and 1.2 in mean class IoU. as shown in the last two lines of Tab. 4.3. Also, as can be observed in Fig. 4.5, the learned lightweight and multi-talented student can sometimes achieve even superior performance to those of the cumbersome teachers, demonstrating that the knowledge from one teacher can potentially benefit the task of the other.

4.5 Additional Details and Results

This section presents additional results obtained from the amalgamation of heterogeneous graph neural networks. Firstly, it showcases the outcomes of amalgamating graph regression models. Subsequently, it provides further results from the amalgamation of node classification models, as well as point cloud classification and part segmentation models. Furthermore, this section conducts thorough ablation studies to validate the effectiveness of the proposed knowledge amalgamation method. These studies encompass different dataset splitting protocols, diverse network architectures, various combinations of teacher models, and a range of heterogeneous types of graph neural networks (GNNs).

4.5.1 Amalgamating Graph Regression Models

To evaluate the performance of the proposed method beyond classifications, in this section, we perform extended experiments on the task of graph-property regression.

Implementation Details. To pre-train two teacher models, we split the QM7b dataset with a ratio of $3:1:6$ for training, validation, and testing, respectively. Also, following [41], each target is normalized to have mean 0 and variance 1. Furthermore, we divide the 14 regression tasks in QM7b into two subsets. Each teacher is trained to predict 7 target properties, and the

TABLE 4.5: Teacher and student network architectures for graph regressions on QM7b dataset.

Models	Layers	Attention Heads	Hidden	Output
Teacher 1 {QM7b_Set1}	3	{1, 1, 1}	256	7
Teacher 2 {QM7b_Set2}	3	{1, 1, 1}	256	7
Student {QM7b_Set1, QM7b_Set2}	3	{1, 1, 1}	128	14

TABLE 4.6: Results of amalgamating knowledge from two graph regression models in terms of the mean absolute error (MAE). Each teacher model handles regressions of 7 different properties. Our lightweight student is able to simultaneously deal with regressions of 14 properties.

Properties	Teacher 1	Student	Properties	Teacher 2	Student
E (PBE0)	0.195	0.193	LUMO (PBE0)	0.798	0.804
α (PBE0)	0.919	0.890	LUMO (ZINDO)	0.761	0.863
α (SCS)	0.662	0.729	IP (ZINDO)	0.618	0.594
HOMO (GW)	0.814	0.836	EA (ZINDO)	0.763	0.875
HOMO (PBE0)	0.724	0.728	$E^*_{1\text{st}}$ (ZINDO)	0.630	0.603
HOMO (ZINDO)	0.660	0.666	E^*_{max} (ZINDO)	0.599	0.685
LUMO (GW)	0.804	0.856	I_{max} (ZINDO)	0.486	0.580
Model Size:	**291K**	**92K**	**Model Size:**	**291K**	**92K**

student model is therefore expected to handle all 14 targets without ground-truth labels. The network architectures of the teachers and student are shown in Tab. 4.5.

Results. Tab. 4.6 shows the experimental results of knowledge amalgamation from two regression teacher models. The obtained student model, as can be observed, achieves competitive performance in almost all target properties, as compared to the results of the teachers. Also, our student model is trained without any human annotation, yet with a more lightweight architecture than any of the teachers.

TABLE 4.7: Summary of the teacher and student network architectures for amalgamating knowledge from multi-label node classification models, corresponding to Tab. 4.1 and Tab. 4.2.

Models	Layers	Attention Heads	Input	Hidden	Output
Teacher 1 (GAT) {PPI_Set1}	3	{4, 4, 6}	50	256	61
Teacher 2 (GAT) {PPI_Set2}	3	{4, 4, 6}	50	256	60
Teacher 1 (GCN) {PPI_Set1}	3	–	50	256	61
Teacher 2 (GCN) {PPI_Set2}	3	–	50	256	60
Student V1 {PPI_Set1, PPI_Set2}	3	{4, 4, 6}	[1, 50]	256	121
Student V2 {PPI_Set1, PPI_Set2}	5	{2, 2, 2, 2, 2}	[1, 50]	68	121
Student V3 {PPI_Set1, PPI_Set2}	3	{2, 2, 2}	[1, 50]	68	121

4.5.2 Amalgamating Node Classification Models

4.5.2.1 Multi-label Node Classification

Implementation Details. We first randomly split all the labels of PPI into two sets, termed as PPI_Set1 (containing 61 labels) and PPI_Set2 (containing 60 labels). Then we pre-train two teacher GNNs on PPI_Set1 and PPI_Set2, respectively. The goal of our student model is to concurrently recognize all 121 labels covered by the two teachers. Tab. 4.7 shows the architecture details of the student and teacher models for the task of multi-label node classification on the PPI dataset. The architecture of Student V1 is used in Tab. 4.1 and Tab. 4.2. We also perform extensive ablation studies by designing additional two student architectures, termed as Student V2 and Student V3. In training, the batch size is set to 2. For Student V1, the learning rate is set to 1×10^{-4}; for Student V2 and V3, the learning rate is set to 0.005. All the student models are trained for 1500 epochs.

Ablation Studies. We show in Tab. 4.8 the results of using different student architectures and also different types of teacher GNNs, including GCN and GAT. In general, the performance of Student V1 is superior to that of Student V2 and Student V3, which is not unexpected since the model size of Student V1 is much larger than the other two architectures. By comparing the results of Student V2 and Student V3, we can observe that in the case of fewer hidden features and attention heads, the student can boost performance by a large margin through

TABLE 4.8: Ablation studies of amalgamating knowledge from multi-label node classifications models, in terms of F_1 score.

Methods	**Model Size**	**PPI_Set1**	**PPI_Set2**	**Methods**	**Model Size**	**PPI_Set1**	**PPI_Set2**
Teacher 1 (GAT)	11.61M	98.73	N/A	Teacher 1 (GCN)	379.3K	69.48	N/A
Teacher 2 (GAT)	11.56M	N/A	98.62	Teacher 2 (GCN)	378.3K	N/A	63.62
Student_V1	14.57M	98.44	98.42	Student_V1	14.57M	69.64	62.51
Student_V2	744.6K	96.16	96.17	Student_V2	744.6K	69.65	63.15
Student_V3	445.0K	90.73	90.46	Student_V3	445.0K	69.14	61.93
Teacher 1 (GAT)	11.61M	98.73	N/A	Teacher 1 (GCN)	379.3K	69.48	N/A
Teacher 2 (GCN)	378.3K	N/A	63.62	Teacher 2 (GAT)	11.56M	N/A	98.62
Student_V1	14.57M	98.05	62.96	Student_V1	14.57M	70.01	98.01
Student_V2	744.6K	94.81	65.80	Student_V2	744.6K	70.09	95.35
Student_V3	445.0K	90.06	63.32	Student_V3	445.0K	70.24	89.65

using more layers in the corresponding architecture. Also, from the results of amalgamating heterogeneous types of teacher GNNs, it can be observed that different student architectures achieve similar performance for amalgamating GCN models.

Visualizations. Fig. 4.6 shows the visualization results of the proposed topological attribute maps (TAM) for both the teacher GNNs and the student GNN, which are used in the topological semantics alignment module in knowledge amalgamation. The proposed TAM can be extracted in a teacher-aware manner, which is demonstrated as "Student {PPI_Set1}" and "Student {PPI_Set2}" in Fig. 4.6. Specifically, the TAM "Student {PPI_Set1}" is derived by only using a subset of the output class scores, corresponding to {PPI_Set1}. A similar strategy is also used to compute the TAM "Student {PPI_Set2}". The TAMs of the learned student GNN, as can be observed in the second and fourth columns of Fig. 4.6, are similar to those of the corresponding teacher GNNs, meaning that the student GNN has learned the importance of each connection for the corresponding task of the teacher GNNs. The visualizations shown in Fig. 4.6 can partially explain the results in Tab. 4.1, where the proposed topological semantics alignment module can boost the performance of the learned student model in knowledge amalgamation.

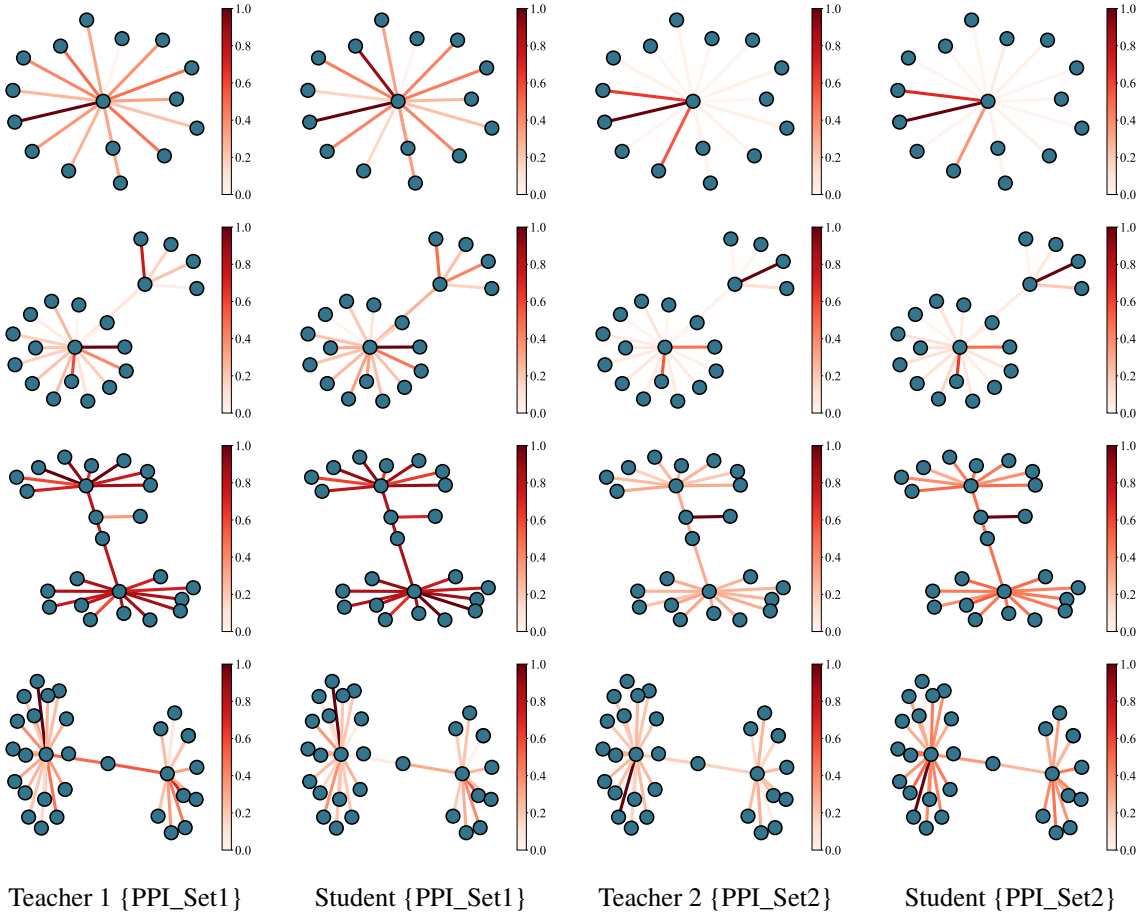


FIGURE 4.6: Visualizations of the scaled topological attribution map (TAM) of two teacher GNNs and the learned student GNN, corresponding to Tab. 4.1. The two teachers here are pre-trained multi-label node classification models that handle a different set of classes (*i.e.*, {PPI_Set1} and {PPI_Set2}). Colors encode the importance of each connection for the corresponding task of each teacher.

4.5.2.2 Single-label Node Classification

Implementation Details. For the task of amalgamating single-label node classification models, we primarily use five datasets, including Amazon Computers, Amazon Photo, Cora, Citeseer, and Pubmed datasets. For the dataset splittings, since there is no standard splitting protocol for Amazon Computers and Amazon Photo datasets, in our experiment, we design three dataset splitting methods: 1) Training Set : Validation Set : Testing Set $= 2 : 2 : 6$; 2) Training Set : Validation Set : Testing Set $= 5 : 2 : 3$; 3) Training Set : Validation Set : Testing Set $= 6 : 2 : 2$. For the dataset splittings of the Cora, Citeseer, and Pubmed datasets,

TABLE 4.9: Summary of teacher and student network architectures for the task of node classification on Amazon Computers and Amazon Photo datasets.

Models	Layers	Input	Hidden	Output
Teacher 1 {Amazon Computers}	2	767	8	10
Teacher 2 {Amazon Photo}	2	745	8	8
Student {Amazon Computers, Amazon Photo}	3	[1, 767]	6	18

we use the splitting protocol in [17]. Specifically, for Cora, we use 1208 samples for training, 500 samples for validation, and 1000 samples for testing; for Citeseer, we use 1827 samples for training, 500 samples for validation, and 1000 samples for testing; for Pubmed, 18217 samples are used for training, 500 samples are used for validation, and 1000 samples are used for testing. The network architectures of the student and teacher models on Amazon Computers and Amazon Photo are shown in Tab. 4.9, while those on Cora, Citeseer, and Pubmed are provided in Tab. 4.10.

More Results. Tab. 4.11 shows the quantitative results of different methods, including MTL+AT [205], MLT+LSP [188], the proposed method without the topological semantics alignment module, and the proposed approach with the proposed topological semantics alignment module. Also, to evaluate the performance of the proposed slimmable graph convolutional layer, we devise a possible solution to the challenge of varying input dimensions of graph data. Specifically, we first use principal components analysis (PCA) to conduct feature dimension reductions. As a result, a naive GNN architecture can simultaneously process the data with distinct feature dimensions in different datasets. However, this possible solution is prone to information loss due to feature reductions, especially for the input graphs with a large difference in feature dimensions. Thus, the obtained student model is limited in performance, which is demonstrated in the seventh row of Tab. 4.11.

Also, we show in Tab. 4.12 more results with different dataset splitting protocols. The student models learned with the proposed knowledge amalgamation method, as can be observed from Tab. 4.12, achieves gratifying results on par with those of the teachers with all three different dataset splittings. The versatile student model can sometimes even achieve performance superior to that of the teachers. For example, Student 2 and Student 3 outperform the corresponding teachers on the classification of the category set Computers.

TABLE 4.10: Summary of teacher and student network architectures for the task of single-label node classification on Cora, Citeseer, and Pubmed datasets.

Models	Layers	Attention Heads	Input	Hidden	Output
Teacher 1 {Cora}	2	{8, 1}	1433	8	7
Teacher 2 {Citeseer}	2	{8, 1}	3703	8	6
Teacher 3 {Pubmed}	2	{8, 1}	500	8	3
Student {Cora, Citeseer}	3	{8, 1}	[1, 3703]	6	13
Student {Cora, Pubmed}	3	{8, 1}	[1, 1433]	6	10
Student {Citeseer, Pubmed}	3	{8, 1}	[1, 3703]	6	9
Student {Cora, Citeseer, Pubmed}	3	{8, 1}	[1, 3703]	6	16

Tab. 4.13 shows the results of amalgamating knowledge from the teacher GNNs trained on Cora, Citeseer, and Pubmed datasets. For a more comprehensive evaluation, we show the knowledge amalgamation results of all the combinations of the three teacher GNNs, *i.e.*, {Cora, Citeseer}, {Cora, Pubmed}, {Citeseer, Pubmed}, and also {Cora, Citeseer, Pubmed}. The corresponding learned student models, as shown in the sixth row of Tab. 4.13, competent to handle all different tasks of the heterogeneous teachers.

Visualizations. In Fig. 4.7, we also visualize the features from the intermediate layers of the teachers and the student using t-SNE. The representations of the obtained student model exhibit discernible clusterings, yet with a more compact and lightweight architecture compared with the teachers.

TABLE 4.11: Results of amalgamating single-label node classification models on Amazon Computers and Amazon Photo dataset, in terms of average classification accuracies (%).

Methods	Amazon-Computers	Amazon-Photo
Teacher 1	89.36	N/A
Teacher 2	N/A	92.48
Student_{MTL+AT} [205]	88.44	89.85
Student_{MTL+LSP} [188]	87.99	91.37
Student_Ours (w/o TSA, w/ Slimmable GraphConv)	88.06	91.22
Student_Ours (w/ TSA, w/ PCA)	88.04	89.56
Student_Ours (w/ TSA, w/ Slimmable GraphConv)	**88.81**	**91.79**

TABLE 4.12: Results of amalgamating node classification models on Amazon Computers and Amazon Photo with various dataset splittings.

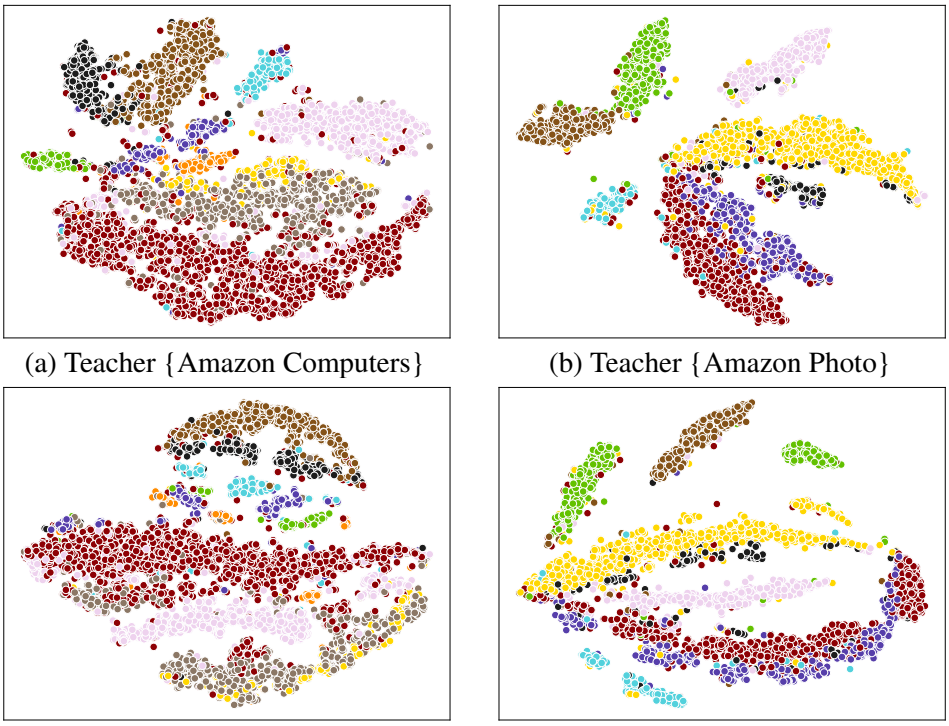
Splitting	Train : Val : Test = 2 : 2 : 6		Train : Val : Test = 5 : 2 : 3		Train : Val : Test = 6 : 2 : 2	
Type	Teacher 1	Teacher 2	Teacher 3	Teacher 4	Teacher 5	Teacher 6
Task	{Computers}	{Photo}	{Computers}	{Photo}	{Computers}	{Photo}
Accuracy	89.36	92.48	89.70	93.99	89.60	94.18
Type	Student 1		Student 2		Student 3	
Task	{Computers, Photo}		{Computers, Photo}		{Computers, Photo}	
Accuracy	88.81 / 91.79		89.84 / 92.81		90.22 / 92.35	

4.5.3 Amalgamating Point Cloud Classification and Segmentation Models

Implementation Details. In additional to amalgamating graph regression models and node classification models, we also conduct extensive experiments by amalgamating knowledge from point cloud classification model and part segmentation model. The dataset we used here is ShapeNet part dataset, where each 3D shape has the labels of both the category and the segmentation parts. For the network architecture, we exploit PointNet++ [127], specifically with single-scale grouping at each level as our backbone. In particular, based on the Euclidean distance among different points [167], the used architecture of PointNet++ starts by establishing graphs from 3D shapes. Then, PointNet++ employs graph coarsening at each layer. Notably, the graphs in PointNet++ are constructed with point coordinates. As such, the established graphs are, in fact, fixed during training. The aggregation method in PointNet++ is a max operation. Tab. 4.14 shows the detailed architectures of the teachers and

TABLE 4.13: Results of amalgamating single-label node classification models on Cora, Citeseer, and Pubmed datasets, in terms of average classification accuracies (%). We show the knowledge amalgamation results of all the combinations of the three teachers.

Type	Teacher 1	Teacher 2	Teacher 1	Teacher 3	Teacher 2	Teacher 3	Teacher 1	Teacher 2	Teacher 3
Task	{Cora}	{Citeseer}	{Cora}	{Pubmed}	{Citeseer}	{Pubmed}	{Cora}	{Citeseer}	{Pubmed}
Accuracy	87.90	79.00	87.90	85.70	79.00	85.70	87.90	79.00	85.70
Type	Student 1		Student 2		Student 3		Student 4		
Task	{Cora, Citeseer}		{Cora, Pubmed}		{Citeseer, Pubmed}		{Cora, Citeseer, Pubmed}		
Accuracy	87.20 / 77.30		86.90 / 83.40		77.00 / 83.60		87.10 / 77.30 / 83.20		



(a) Teacher {Amazon Computers} (b) Teacher {Amazon Photo}

(c) Student {Amazon Computers, Amazon Photo}

FIGURE 4.7: A t-SNE plot of the features from the first hidden layer of the teachers and the student on Amazon Computers and Amazon Photo dataset.

TABLE 4.14: Summary of teacher and student network architectures for the task of point cloud classification and part segmentation.

Models	Layers	MLPs	K
Teacher 1 {Classification}	6	{[64, 64, 128], [128, 128, 256], [256, 512, 1024]}	[32, 64, 1]
Teacher 2 {Segmentation}	8	{[64, 64, 128], [128, 128, 256], [256, 512, 1024]}	[32, 64, 1]
Student {Classification, Segmentation}	11	{[32, 32, 64], [64, 64, 128], [128, 256, 512]}	[32, 64, 1]

the student. For the sake of clarity, we only show the MLPs in the set abstraction layers. K represents the number of points that belong to the neighborhood of each central point.

Ablation Studies. We perform extensive ablation studies on the architectures of the learned student GNN. The corresponding results are shown in Tab. 4.15. Specifically, we report the results of the learned student model with more channels, more layers, and more MLPs, respectively. It can be observed that the student GNN with more channels delivers gratifying results compared with other architectures. However, the corresponding student model has a much larger size than the others, almost four times larger than the baseline model shown in the fourth row of Tab. 4.15. By contrast, adding more layers or more MLPs may not enlarge the

TABLE 4.15: Ablation studies on the task of amalgamating knowledge from point cloud classification and part segmentation models.

Method	**Model Size**	**mAcc** (Cls)	**mIoU** (Seg)	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate Board	Table
Teacher_Cls	17.69M	97.83	N/A	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Teacher_PartSeg	17.01M	N/A	81.72	82.34	81.92	86.12	78.33	90.54	72.18	91.25	86.09	83.57	95.48	70.63	94.98	81.98	55.99	73.73	82.34
Student	**6.37M**	**97.67**	**78.96**	**81.82**	**76.07**	**81.17**	**76.91**	**89.59**	**70.56**	**90.17**	**85.69**	**82.95**	**94.92**	**57.06**	**94.02**	**79.24**	**48.05**	**72.67**	**82.50**
Student (w/ more Channels)	24.96M	97.61	81.12	82.02	82.46	8712	78.23	90.18	67.67	90.92	86.36	83.42	94.99	68.08	94.43	81.02	53.19	74.72	83.14
Student (w/ more Layers)	9.34M	97.33	76.99	79.99	73.43	78.84	74.28	89.05	65.00	89.38	84.32	81.61	95.16	56.80	91.79	76.26	42.37	71.45	82.07
Student (w/ more MLPs)	7.14M	97.39	78.41	80.78	76.78	79.21	76.33	89.61	69.72	90.32	85.82	83.01	95.33	570.8	93.91	77.94	44.29	71.89	82.57

model too much. However, we can observe that more layers or more MLPs do not necessarily contribute to improved classification and segmentation performance.

More Qualitative Results. In Fig. 4.8, we show more qualitative results of joint classification and part segmentation, in correspondence to Fig. 4.5. The corresponding quantitative results are shown in Tab. 4.3. We use red texts in Fig. 4.8 to highlight the misclassified results. The learned student models can sometimes achieve results superior to those of the teachers, as can be observed from the first three rows of Fig. 4.8 where the teacher model misclassifies the input point clouds of class aero, pistol, and mug while the learned student generates the correct labels. Also, in the fifth, sixth, and seventh rows, the learned student outperforms the corresponding teacher in the part segmentation results. These observations demonstrate that the knowledge from one teacher can potentially benefit the task of the other.

To validate the effectiveness of the proposed topological semantics alignment (TSA) module, in Fig. 4.8, we also demonstrate the results of the student with TSA and those of the student without TSA. As shown in Fig. 4.8, the proposed TSA module can boost the performance of both the point cloud classification and part segmentation. For example, in the third row of Fig. 4.8, "Student w/o TSA" misclassifies "Mug" into "Lamp", whereas the student with TSA generates the correct prediction. Moreover, for most cases in Fig. 4.8, "Student w/ TSA" delivers superior results in part segmentation, compared with "Student w/o TSA".

4.6 Summary

This chapter introduces a novel model reusing task tailored for heterogeneous GNNs. Our goal is to learn a versatile and lightweight student GNN that masters the complete set

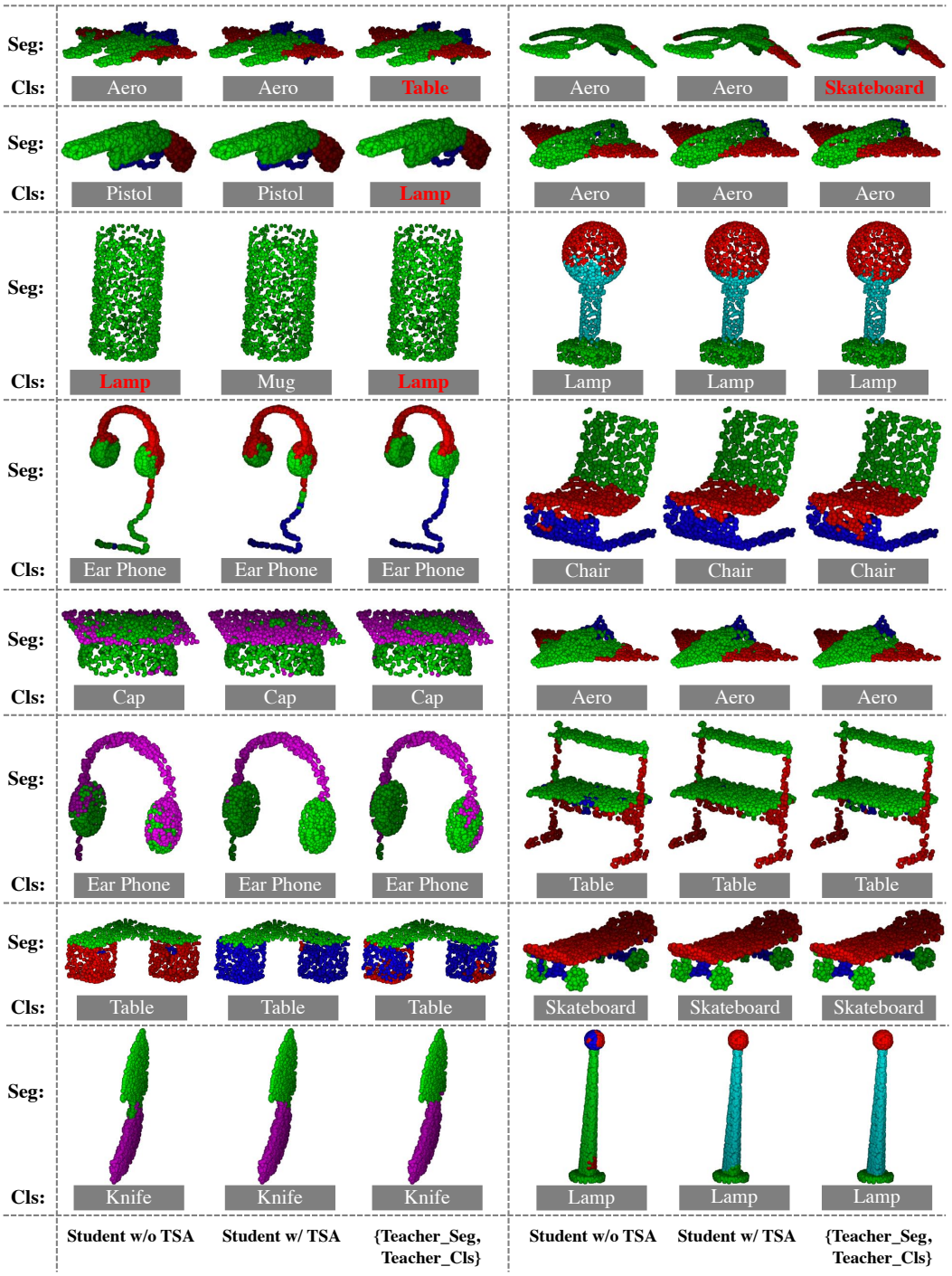


FIGURE 4.8: Visualization results of joint part segmentation (Seg) and classification (Cls). From left to right: the results of the learned student GNN without the proposed topological semantics alignment (TSA) module, those of the student with TSA, and the results of the two teacher GNNs. We use red texts to highlight the misclassified outputs.

of expertise of multiple heterogeneous teachers, yet without human-labeled annotations. Towards this end, we identified two key challenges, and propose a dedicated slimmable graph convolutional operation as well as a novel topological attribution map (TAM) to solve the dilemma. Experiments on single- and multi-label classification and point cloud segmentation-classification demonstrate that, the obtained student GNN, with a moderately compact size, achieves performances on par with or even superior to those of the individual teachers on their specialized tasks. In our future work, we will further explore knowledge amalgamation from heterogeneous teachers specializing in cross-domain tasks, like semantic segmentation and object tracking. We will also strive to generalize the proposed TAM to other tasks beyond knowledge amalgamation.

CHAPTER 5

Data-Model-Driven Efficient Learning with Meta-Aggregator

The previous two chapters delved into the realms of data-driven and model-driven efficient learning, examining them individually. In this chapter, the focus shifts towards the pursuit of joint data-model-driven efficient representation learning. To accomplish this, this chapter presents a tailored binarization framework designed specifically for GNNs. The proposed framework aims to enable the simultaneous binarization of input graph data and model parameters, facilitating lightweight inference. To achieve this goal, this chapter begins by developing a vanilla 1-bit GNN framework that binarizes both the GNN parameters and the graph features. Despite the lightweight architecture, it was observed that this vanilla framework suffered from insufficient discriminative power in distinguishing graph topologies, leading to a dramatic drop in performance. This discovery motivates us to devise meta aggregators to improve the expressive power of vanilla binarized GNNs, of which the aggregation schemes can be adaptively changed in a learnable manner based on the binarized features. Towards this end, this chapter proposes two dedicated forms of meta neighborhood aggregators, an exclusive meta aggregator termed as *Greedy Gumbel Neighborhood Aggregator (GNA)*, and a diffused meta aggregator termed as *Adaptable Hybrid Neighborhood Aggregator (ANA)*. GNA learns to exclusively pick one single optimal aggregator from a pool of candidates, while ANA learns a hybrid aggregation behavior to simultaneously retain the benefits of several individual aggregators. Furthermore, the proposed meta aggregators may readily serve as a generic plugin module into existing full-precision GNNs. Experiments across various domains demonstrate that the proposed method yields results superior to the state of the art.

5.1 Introduction

Graph neural networks (GNNs) have recently emerged as the dominant paradigm for learning and analyzing non-Euclidean data, which contain rich node content information as well as topological relational information [32, 56, 177]. As such, a massive number of GNN architectures have been developed [82, 153, 180, 191, 221]. The success of GNNs also triggers a great surge of interest in applying elaborated graph networks to various tasks across many domains, such as object detection [55, 43], pose estimation [190], point cloud processing [88, 167, 129], and visual SLAM [139]. These GNN-based applications, in general, rely on cumbersome graph architectures to deliver gratifying results. For example, SuperGlue, a GNN-based feature matching approach, requires 12M network parameters to achieve the state-of-the-art performance [139].

In practice, however, such applications typically require a compact and lightweight architecture for real-time interaction, especially in resource-constrained environments. In the case of autonomous driving [116], for example, it is critical to maintain fast and timely responses for GNN-based SLAM algorithms to handle complex traffic conditions, thereby leading to the urgent need of compressing cumbersome GNN models. The work of [188], as the first attempt, leverages knowledge distillation to learn a compact student GNN with fewer parameters. In spite of the improved efficiency, this approach still relies on the expensive floating-point operations, let alone a well-performed teacher model pre-trained in the first place.

In this chapter, we strive to make one step further towards ultra lightweight GNNs. Our goal is to train a customized 1-bit GNN, as shown in Fig. 5.1, that allows for favorable memory efficiency and meanwhile enjoys competitive performance. We start with developing a naïve GNN binarization framework, achieved through converting 32-bit features and parameters into 1-bit ones, followed by leveraging straight-through estimator to optimize the binarized model. The derived vanilla binarized GNN enjoys favorable memory efficiency; however, its performance is not encouraging as expected. Through parsing its underlying process, we identified that the binarization yields limited expressive power, making the model incapable to distinguish different graph topologies. An illustrating example is shown in Fig. 5.2(a),

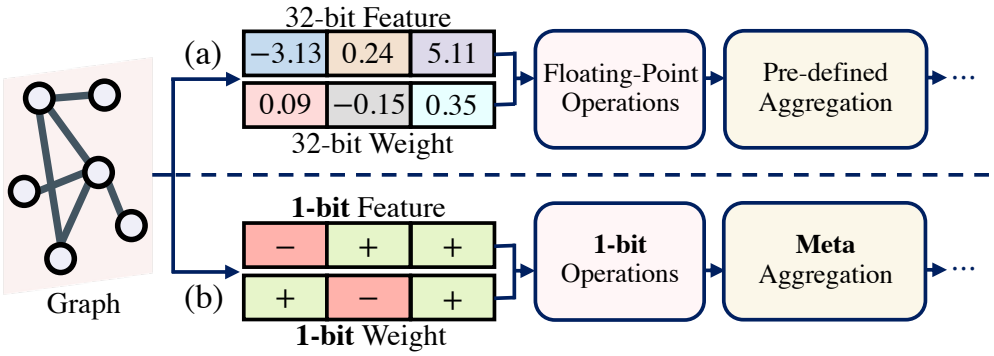


FIGURE 5.1: Illustrations of the computational workflow in (a) conventional full-precision GNNs and (b) the proposed 1-bit GNNs. In particular, we devise two meta aggregators for the proposed model, termed as *Greedy Gumbel Aggregator* (GNA) and *Adaptable Hybrid Aggregator* (ANA), that learn to perform adaptive aggregation in a graph-aware and layer-aware manner.

where a *mean* aggregator, which is commonly adopted by full-precision GNNs, produce identical aggregation results for two diversified graph topologies with binarized features, thereby leading to inferior performances.

Inspired by this discovery, we introduce to the proposed GNN binarization framework a learnable and adaptive neighborhood aggregator, so as to alleviate the aforementioned dilemma and enhance the distinguishability of 1-bit graphs. Unlike existing GNNs that rely on a pre-defined and fixed aggregator, our elaborate meta neighborhood aggregators enables dynamically *selecting* (Fig. 5.2(b)) or *generating* (Fig. 5.2(c)) customized input- and layer-specific aggregation

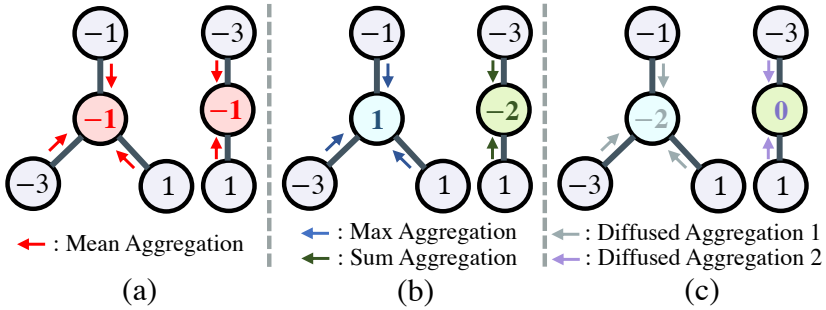


FIGURE 5.2: Example aggregation results of the two graphs with different topological structures for (a) the conventional pre-defined and fixed aggregator, (b) the proposed exclusive form of meta aggregators GNA, and (c) the proposed diffused form of meta aggregators ANA.

schemes. As such, we explicitly account for the customized characteristics of binarized graph features, and further strengthen the discriminative power for handling topological structures.

Towards this end, we propose two variants of meta aggregators: an exclusive meta aggregator, termed as *Greedy Gumbel Neighborhood Aggregator* (GNA), that adaptively *selects* an optimal aggregator in a learnable manner, as well as a diffused meta aggregator, termed as *Adaptable Hybrid Neighborhood Aggregator* (ANA), that either *approximates* a single aggregator or dynamically *generates* a hybrid aggregation behavior. Specifically, GNA incorporates the discrete decisions from the candidate aggregators, conditioned on the individual graph features, into the gradient descent process by leveraging *Greedy Gumbel Sampling*. Inevitably, the performance of GNA is bottlenecked by the individual aggregators in the candidate pool. Thus, we further devise ANA that enables generating a hybrid aggregator dynamically based on the input 1-bit graphs. ANA simultaneously preserves the strengths of multiple individual aggregators, leading to favorable competence to handle the challenging 1-bit graph features. Moreover, the proposed GNA and ANA can be readily extended as portable modules into the general full-precision GNN models to enhance the expressive capability.

In sum, our contribution is a novel GNN-customized binarization framework that generates a 1-bit lightweight GNN model with competitive performance, making it competent for resource-constrained applications such as edge computing. This is specifically achieved through an adaptive meta aggregation scheme to accommodate the challenging quantized graph features. We evaluate the proposed customized framework on several large-scale benchmarks across different domains and graph tasks. Experimental results demonstrate that the proposed meta aggregators achieve results superior to the state-of-the-art, not only on the devised 1-bit binarized GNN models, but also on the general full-precision models.

5.2 Vanilla Binary GNN and Pre-analysis

In this section, we first develop a vanilla binary GNN framework by simply binarizing model parameters and activations. We then show the limitations of this vanilla binary GNN by looking into the internal message aggregation process and accordingly develop two possible

solutions to address these limitations. Eventually, built upon the possible solutions, we introduce the idea of the proposed customized GNN binarization framework with the meta aggregators.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a directed/undirected graph with nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$, where $\{v_j\}$ is the set of neighboring nodes of v_i. Each node has an associated node feature $\mathcal{X} = [x_1\ x_2\ ...\ x_n]$. For example, in the task of 3D object classification, x can be set as the 3D coordinates.

Existing GNNs follow an iterative neighborhood aggregation scheme at each GNN layer, where each node v_i iteratively gathers features from its neighboring nodes $\{v_j\}$ to capture the structural information [90, 180]. Let $\mathcal{X}_i^\ell$ denote the feature vector of the node v_i at layer ℓ. The corresponding updated feature vector $\mathcal{X}_i^{\ell+1}$ in a GNN can then be formulated as:

$$\mathcal{X}_i^{\ell+1} = f\left(\mathcal{X}_i^\ell, \{\mathcal{X}_j^\ell : (j,i) \in \mathcal{E}\}\right), \tag{5.1}$$

where $\mathcal{X}_j^\ell$ represents the feature associated with the neighboring nodes. f is a mapping function that takes $\mathcal{X}_i^\ell$ as well as $\mathcal{X}_j^\ell$ as inputs. The choice of the mapping f corresponds to different architectures of GNNs.

For the sake of simplicity, we take here graph convolutional network (GCN) proposed by Kipf and Welling [82] as an example GNN architecture for the following discussions. We denote Mean as the mean aggregator that computes an average of the incoming messages and $\mathcal{W}$ as the learnable weight matrix for feature transformation. The general GNN form in Eq. 5.1 can then be instantiated for GCN as: $\mathcal{X}_i^{\ell+1} = \text{ReLU}\left(\mathcal{W}^l\ \text{Mean}_{(j,i)\in\mathcal{E}}\mathcal{X}_j^\ell\right)$ or $\mathcal{X}_i^{\ell+1} = \text{ReLU}\left(\text{Mean}_{(j,i)\in\mathcal{E}}\mathcal{W}^l\mathcal{X}_j^\ell\right)$, which respectively correspond to the case where aggregation comes first or comes after the feature transformation step [161].

Vanilla 1-bit GNN Models. We develop a naïve binarized GNN framework to compress cumbersome GNN models, by directly binarizing 32-bit input features and learnable weights in the feature transformation step into 1-bit ones.

Specifically, for the case of vanilla binary GCN, the forward propagation process can be modeled as:

$$\text{Net Forward:} \quad w_b = \text{sign}(w) = \begin{cases} +1, & w \geq 0 \\ -1, & w < 0 \end{cases}, \tag{5.2}$$

where w represents the element in the learnable weight matrix $\mathcal{W}$. We also binarize the graph features $\mathcal{X}$ in the same manner, by replacing w in Eq. 5.2 with the feature element x.

During the backward propagation, it is not feasible to simply exploit *Backward Propagation (BP)* algorithm [137], as most full-precision models do, to optimize binarized graph networks, due to the undifferentiable binarization function, *i.e.*, sign in Eq. 5.2. The derivative part of the sign function will lead to 0 gradients almost everywhere, thereby resulting in the vanishing gradient problem. To alleviate this dilemma, we leverage the *Straight-through Estimator (STE)* [4] for the backward propagation process in the binarized graph nets, formulated as:

$$\text{Net Backward:} \quad \frac{\partial \mathcal{L}}{\partial w} = \begin{cases} \frac{\partial \mathcal{L}}{\partial w_b}, & w \in (-1, 1) \\ 0, & \text{otherwise} \end{cases}, \tag{5.3}$$

where $\mathcal{L}$ represents the loss function. Essentially, Eq. 5.3 can be considered as propogagting the gradient through *hard tanh* function, defined as: $\text{Htanh}(x) = \text{Clip}(x, -1, 1)$.

We illustrate in Fig. 5.3 the computational workflow at an example binarized GCN layer for the case where the aggregation comes after the feature transformation. A similar scheme can be observed for the GCN model where the aggregation happens first. With compact node features and net weights, binarized GCN only relies on 1-bit XNOR and bit-count operations for graph-based processing, leading to an efficient and lightweight graph model that is competent for edge computing.

Despite the compact binarized parameters and features, we empirically observed that the results of the developed vanilla GNN were not promising as expected. Specifically, we conduct a preliminary experiment on the ZINC dataset [63] with the GCN architecture in [32]. Averaged over 25 independent runs, the full-precision GCN model achieves the performance of 0.407 ± 0.018 in terms of the mean absolute error (MAE), whereas the vanilla binarized

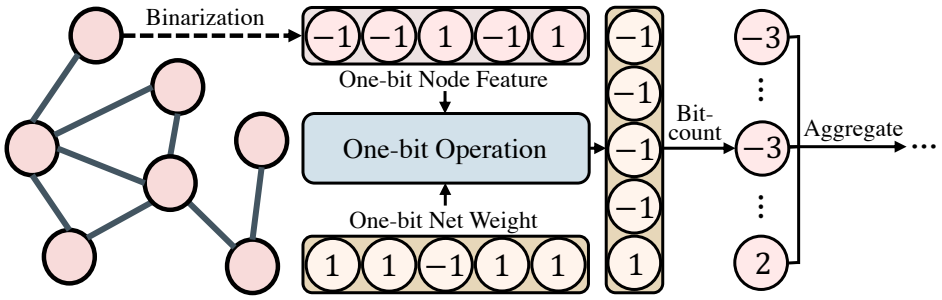


FIGURE 5.3: Illustrations of the computational workflow at an example binarized GNN layer. Despite the efficient 1-bit operations, the output features are less distinguishable between each other, leading to the challenge in the aggregation step shown in Fig. 5.4.

GCN yields the result of 0.669 ± 0.070 in MAE, which is far behind that of the full-precision one.

We explore the reason behind this challenge of implausible performance, by looking into the internal computational process in binarized GNNs. Specifically, we look back on Fig. 5.3, which shows the example workflow at a binarized GCN layer where the feature transformation is performed before the aggregation step. It is noticeable that the result of 1-bit operations lies in the discrete integer domain. The resulted feature space is thereby much smaller than that of the 32-bit floating-point operations. In other words, the outputs of 1-bit operations are less distinguishable from each other. This property, when appearing in the graph domain, leads to difficulties to extract and discriminate graph topologies in the neighborhood aggregation process, which in fact is the key to the success of graph networks.

To further illustrate this dilemma, we demonstrate a couple of examples in Fig. 5.4, including both max and mean aggregation schemes that are commonly leveraged in GNNs. Fig. 5.4(a) shows the aggregation results of the 32-bit GNN layer, where both of max and mean aggregators successfully distinguish the two different topological structures, respectively. However, for the aggregation results of discrete integer features in binarized GNNs (Fig. 5.4(b)), neither max nor mean aggregators can discriminate the corresponding two graph structures. Moreover, the situation will be more challenging for the case where the aggregation happens before the transformation, since the features fed into the aggregator are limited to only 1 or -1.

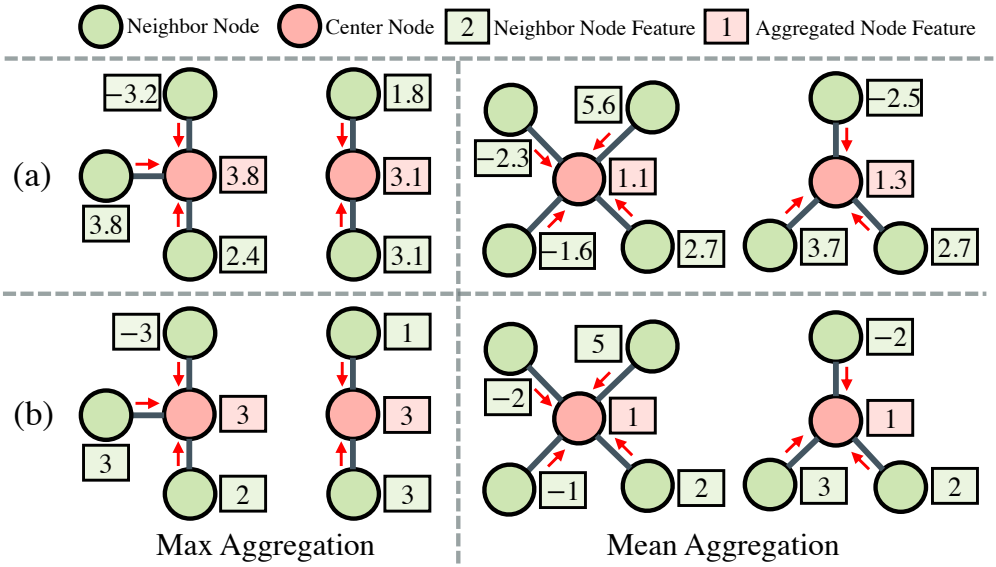


FIGURE 5.4: Example aggregation results of (a) conventional 32-bit GNN layer and (b) binarized GNN layer, corresponding to Fig. 5.3. For (a), both mean and max aggregators can distinguish the two graph structures; however, for binarized GNN (b), max and mean aggregators fail to differentiate between two topologies.

Nevertheless, from Fig. 5.4(b), we also found that, by combining different aggregation schemes, various graph topologies could in fact become distinguishable. This observation motivates us to develop possible solutions to alleviate the aforementioned dilemma in vanilla binarized GNNs. Specifically, we propose a couple of straightforward mixed multi-aggregators that combine the benefits of various aggregation schemes in two different ways. The first one conducts multiple times of message aggregation with several different aggregators and then computes the sum over the aggregation results, leading to the performance of 0.599 ± 0.017 in MAE with five standard aggregators. The second one, on the other hand, concatenates the results from several independent aggregators, achieving the average result of 0.614 ± 0.045 over 25 runs.

In spite of the improved performance, the devised possible solutions need to perform multiple times of feature aggregations at each GNN layer, resulting in heavy computational burdens. Motivated by this limitation, we introduce the proposed meta neighborhood aggregators, which aim to enhance the discriminative capability of topological structures and meanwhile enjoy efficient computations.

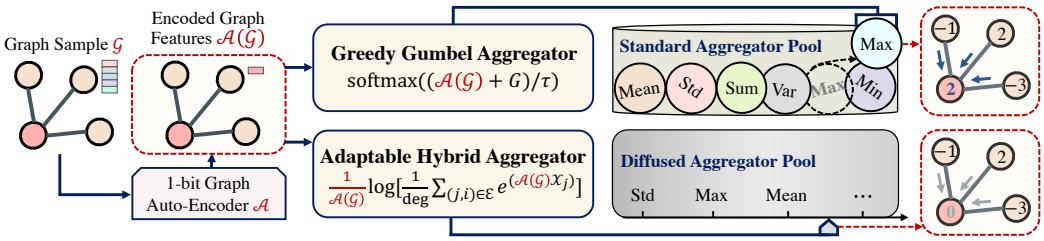


FIGURE 5.5: The overall framework of the proposed meta neighborhood aggregation methods. The upper row illustrates the workflow of the exclusive meta aggregator GNA, which receives the encoded graph features from the binarized graph auto-encoder $\mathcal{A}$ (*i.e.*, the pink trapezoid) and exclusively determines a single optimal layer-wise and node-wise aggregator from a candidate aggregator pool. The lower row, on the other hand, demonstrates the diffused meta aggregator ANA, which amalgamates various aggregation behaviors.

5.3 Meta Neighborhood Aggregation

5.3.1 Overview

Towards addressing the aforementioned limitations of the devised mixed multi-aggregators, we introduce in this section the proposed concept of the *Meta Aggregator*, which aims to adaptively and efficiently adjust the way to aggregate information in a learnable manner. Towards this end, we propose a couple of specific forms of meta aggregators, *i.e.*, the *exclusive* meta aggregation method and the *diffused* meta aggregation method, as illustrated in Fig. 5.5.

The exclusive form, termed as *Greedy Gumbel Neighborhood Aggregator* (GNA), learns to determine a single optimal aggregation scheme from a pool of candidate aggregators, according to the individual characteristics of the quantized graph features, as shown in the upper part of Fig. 5.5. The diffused meta form, on the other hand, adaptively learns a customized aggregation formulation that can potentially incorporate the properties of several independent aggregators, thereby termed as *Adaptable Hybrid Neighborhood Aggregator* (ANA) shown in the lower part of Fig. 5.5.

In what follows, we detail the devised two forms of meta neighborhood aggregation methods, *i.e.*, GNA and ANA, and also the associated training strategy.

Algorithm 2 Training a lightweight 1-bit GNN model with the proposed meta neighborhood aggregators.

Input: L: the number of layers; $\mathcal{W}$: the GNN model weight; $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$: input graph data with nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$; $\mathcal{X}$: the input binarized node feature vector; $\mathcal{A}$: the graph auto-encoder; *Meta-Aggre.*$\in$\{GNA, ANA\}: the choice of meta neighborhood aggregators.
Output: $\mathcal{M}_b$: Target 1-bit binarized GNN model.

for $\ell = 1$ to L **do**
  Feed the graph sample $\mathcal{G}$ into the GNN layer ℓ;
  Binarize the GNN weight $\mathcal{W}^\ell$ into $\mathcal{W}_b^\ell$ by Eq. 5.2;
  Perform 1-bit transformation with $\mathcal{X}$ and $\mathcal{W}_b^\ell$;
  Binarize the weight $\mathcal{W}^{\mathcal{A}^\ell}$ of $\mathcal{A}^\ell$ into $\mathcal{W}_b^{\mathcal{A}^\ell}$ by Eq. 5.2;
  Obtain the encoded features $\mathcal{A}^\ell(\mathcal{G})$ with $\mathcal{W}_b^{\mathcal{A}^\ell}$;
  // Identify the choice from the two meta aggregators
  if *Meta-Aggre.* is GNA **then**
    // Exclusively decide an optimal aggregator
    Feed $\mathcal{A}^\ell(\mathcal{G})$ into the GNA module.
    Obtain the decision GNA_i^ℓ for node v_i by Eq. 5.4;
    Perform aggregations with the obtained GNA_i^ℓ;
  else if *Meta-Aggre.* is ANA **then**
    // Generate a diffused aggregator
    Feed $\mathcal{A}^\ell(\mathcal{G})$ into the ANA module;
    Obtain the diffused aggregator ANA_i^ℓ by Eq. 5.5;
    Perform aggregations with the obtained ANA_i^ℓ;
  end if
end for
Optimize the binarized GNN $\mathcal{M}_b$ for epochs by Eq. 5.3.

5.3.2 Greedy Gumbel Aggregator

Motivated by the observation from Fig. 5.4, where different single aggregators work for a corresponding set of cases as explained in Sect. 5.2, we propose the idea of adaptively determining the optimal aggregator depending on the specific input graphs, as depicted in the upper part of Fig. 5.5.

To this end, there are a few challenges to be addressed. First, the aggregation selector should understand the underlying characteristics of various input graphs without introducing much additional computational cost. To address this issue, we propose to leverage a 1-bit graph auto-encoder to extract meaningful information from input graphs, which is then exploited to guide the decision of different aggregation methods.

TABLE 5.1: Results on the ZINC dataset with different architectures, in terms of the mean absolute error (MAE). From left to right: the results of the full-precision GNNs (Full), those of the 1-bit GNNs without the proposed meta aggregators (Vanilla), and the results of the 1-bit GNNs with GNA and ANA. We also provide the p-value of the paired t-test to demonstrate the statistically meaningful improvements by the proposed GNA and ANA.

Methods	Full (GAT) [153]	Vanilla (GAT) [61]	**GNA (GAT)**	**ANA (GAT)**	Full (GCN) [82]	Vanilla (GCN) [61]	**GNA (GCN)**	**ANA (GCN)**
Bit-width	**32/32**	1/1	1/1	1/1	**32/32**	1/1	1/1	1/1
Param Size	**399.941KB**	81.7070KB	82.0610KB	81.8799KB	**402.645KB**	82.2002KB	82.5566KB	82.3740KB
Test MAE±SD	**0.476±0.006**	0.670±0.064	0.592±0.013	0.566±0.012	**0.407±0.018**	0.669±0.070	0.608±0.024	0.607±0.020
Train MAE±SD	**0.300±0.024**	0.610±0.066	0.531±0.013	0.453±0.019	**0.303±0.026**	0.624±0.069	0.558±0.027	0.564±0.021
***p*-value**	GNA vs. Vanilla: 3.010×10^{-7} / ANA vs. Vanilla: 2.359×10^{-10}				GNA vs. Vanilla: 1.597×10^{-4} / ANA vs. Vanilla: 9.787×10^{-5}			

The second challenge is how to incorporate the discrete selections into the gradient descent process in training GNNs. One straightforward solution would be to model the discrete determination process as a state classification problem and to consider the various aggregators in the candidate pool as different labels. However, this naïve attempt does not account for the uncertainty of the selector, which is likely to cause the model collapse problem where the output choice is independent of the input graphs, such as always or never picking up a specific aggregator.

To alleviate this dilemma, we propose to impose stochasticity in the aggregator decision process with greedy Gumbel sampling [109, 152] and propagate gradients through stochastic neurons through the continuous form of Gumbel-Max trick [65]. Specifically, we introduce such stochasticity by greedily sampling noise from the Gumbel distribution, due to its property of Gumbel-Max trick [44]. In terms of Gumbel random variables, the Gumbel-Max trick can be utilized to parameterize discrete distributions. However, there is a argmax operation in the Gumbel-Max trick, which is not differentiable. We thereby resort to its continuous relaxation form, termed as Gumbel-softmax estimator, to address this issue, which uses a softmax function to replace the undifferentiable argmax function.

With the aforementioned graph auto-encoder and also the Gumbel-softmax estimator to address the two challenges, respectively, the proposed greedy Gumbel aggregator (GNA) for node v_i can then be formulated as:

$$\text{GNA}_i^{\ell} = \text{softmax}\Big(\big(\mathcal{A}^{\ell}(\mathcal{G}) + G\big)/\tau\Big), \tag{5.4}$$

where $\mathcal{A}^{\ell}$ represents the binarized graph auto-encoder at layer ℓ that extracts principal and meaningful information, and G denotes the sampled Gumbel random noise. $\mathcal{G}$ is the input subgraph with one centered node v_i and a set of its neighboring nodes v_j where the connection $(v_i, v_j) \in \mathcal{E}$. τ is a constant that denotes the temperature of the softmax. GNA_i^{ℓ} is the output one-hot vector that indicates the aggregator decision at node v_i and layer ℓ from a pool of candidate aggregators like $\{\text{max}, \text{min}, \text{std}, \text{var}, \ldots, \text{mean}\}$.

In this way, the proposed greedy Gumbel aggregator adaptively decides the optimal aggregator conditioned on each specific node and layer in a learnable manner, which can significantly improve the topological discriminative capability of the vanilla binary GNN model.

5.3.3 Adaptable Hybrid Aggregator

Despite the improved representational ability, the performance of the greedy Gumbel aggregator is bottlenecked by that of the existing standard aggregators, which leaves room for further improvement. Motivated by this observation, we further devise an adaptable hybrid neighborhood aggregator (ANA) that can generate a hybrid form of the several standard aggregators in a learnable manner, thereby simultaneously retaining the advantages of different aggregators. The overall computational pipeline of ANA is demonstrated in the lower part of Fig. 5.5.

We start by giving the developed graph-based mathematical formulation for diffused message aggregation, defined as follows:

$$\text{ANA}_i^{\ell} = \frac{1}{\mathcal{A}^{\ell}(\mathcal{G})} \log \left[\frac{1}{\text{deg}_i} \sum_{(j,i)\in\mathcal{E}} \exp(\mathcal{A}^{\ell}(\mathcal{G})\, \mathcal{X}_j^{\ell}) \right], \tag{5.5}$$

where deg_i is the in-degree of the node v_i and $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is the graph sample with edges $(v_i, v_j) \in \mathcal{E}$. We use $\mathcal{A}^{\ell}$ to denote the 1-bit graph auto-encoder at layer ℓ, as is also used in Eq. 5.4. $\mathcal{X}_j^{\ell}$ represents the feature vector of the neighboring node v_j at layer ℓ, whereas ANA_i^{ℓ} is the obtained diffused aggregator.

Eq. 5.5 can essentially approximate the max and mean functions, depending on the output of graph auto-encoder $\mathcal{A}^{\ell}(\mathcal{G})$. Specifically, higher $\mathcal{A}^{\ell}(\mathcal{G})$ will lead to a behavior similar to

that of the max aggregator, while smaller values of $\mathcal{A}^{\ell}(\mathcal{G})$ generate an effect of the mean neighborhood aggregation. Detailed mathematical proof is provided in the next section of additional results and details.

By slightly changing the form of Eq. 5.5, we can also approximate other aggregators. For example, by simply adding a minus to the input graph features, Eq. 5.5 can approach the behavior of the min aggregation. Also, by utilizing the fact $\text{Var}(\mathcal{X}) = \text{mean}(\mathcal{X}^2) - \big(\text{mean}(\mathcal{X})\big)^2$, the variance aggregator can be approximated by adding the square operations to Eq. 5.5. More detailed derivations and mathematical proofs can be found in the next section of additional results and details.

Furthermore, it is also possible to simultaneously combine the benefits of all these approximated aggregators, by summing multiple terms in Eq. 5.5 with graph-based learnable weighting factors that adaptively control the diffused degree of various aggregator approximations. We illustrate the corresponding sophisticated formulation and also more detailed explanations in the next section of additional results and details.

5.3.4 Training Strategy

We also propose a training strategy, tailored for the proposed method. As a whole, the principal operations of training a 1-bit GNN model with the proposed meta neighborhood aggregation approaches is concluded in Alg. 2. For the sake of clarity, we omit the bias terms in our illustration, which have similar behavior to that of the GNN weight $\mathcal{W}$. Also, we take the case where the feature transformation happens before the aggregation step as an example to illustrate the overall workflow.

As can be observed from Alg. 2, at each layer, the input graph is fed into the lightweight 1-bit graph auto-encoder $\mathcal{A}$ to extract useful information that is beneficial to the following meta aggregators. Followed by this graph encoding process, the meta neighborhood aggregation module receives the encoded features and exclusively determines an optimal aggregator, or produces a diffused aggregator that amalgamates the behaviors of several independent

TABLE 5.2: Results of the proposed meta aggregation methods and other approaches for 32-bit full-precision models on the ZINC dataset, in terms of MAE. The results are averaged over 25 independent runs with 25 different random seeds.

Methods	**Param Size**	**Test MAE±SD**	**Train MAE±SD**
GatedGCN [7]	413.027KB	0.426±0.012	0.272±0.023
GraphSage [47]	371.004KB	0.475±0.007	0.296±0.030
GIN [180]	402.652KB	0.387±0.019	0.319±0.020
MoNet [118]	414.070KB	0.386±0.009	0.299±0.016
GCN [82]	402.645KB	0.407±0.018	0.303±0.026
GAT [153]	399.941KB	0.476±0.006	0.300±0.024
GNA (Ours)	411.270KB	0.337±0.021	0.160±0.026
ANA (Ours)	**404.504KB**	**0.325±0.015**	**0.109±0.014**

aggregators. The desired 1-bit GNN model can eventually be obtained by optimizing the model for epochs with the straight-through estimator, as explained in Sect. 5.2.

5.4 Experiments

In this section, we perform extensive experiments on three publicly available benchmarks across diversified problem domains, including graph regression, node classification, and 3D object recognition. Followed by the evaluations, we further provide detailed discussions regarding the strengths and weaknesses of the devised meta aggregators.

5.4.1 Experimental Settings

Datasets. We validate the effectiveness of the proposed meta aggregation methods on three different datasets, each of which specializes in a distinct task. Specifically, for the task of graph regression, we use the ZINC dataset [63], which is one of the most popular real-world molecular datasets [32]. The goal of ZINC is to regress a specific molecular property, *i.e.*the constrained solubility, which is a critical property for developing GNNs for molecules [200].

Also, for the node classification task, we adopt the protein-protein interaction (PPI) dataset [226], which is a multi-label dataset with 24 graphs corresponding to different human tissues.

TABLE 5.3: Results on the PPI dataset for the task of node classification, in terms of micro-averaged F_1 score. Detailed network architectures can be found in the next section of additional results and details.

Methods	Bit-width	Param Size	F_1 Score
Full Prec. [153]	**32/32**	**43.7712MB**	**98.70**
Vanilla [61]	1/1	28.2560MB	92.68
GNA (Ours)	1/1	28.2572MB	97.52
ANA (Ours)	1/1	28.2565MB	97.71

Each node in the PPI dataset is labeled with various protein functions. The objective of PPI is thereby to predict the 121 protein functions from the interactions of human tissue proteins. Furthermore, we utilize ModelNet40 [175] for the evaluation on the task of 3D object classification. ModelNet40 is a popular dataset for 3D object analysis [126, 127], containing 12,311 meshed CAD models from 40 shape categories in total. Each object comprises a set of 3D points, with the 3D coordinates as the features. The goal is to predict the category of each 3D shape.

For other settings such as learning rates and batch size, we follow those in the works of [32], [153], and [167] for the tasks of graph regression, node classification, and point cloud classification, respectively.

In particular, for more convincing evaluations, we report the results on the ZINC dataset over 25 independent runs with 25 different random seeds. Also, as done in the field of CNN binarization [130], we keep the first and the last GNN layer full-precision and binarize the other GNN layers for all the comparison methods. More detailed task-by-task architecture designs as well as the hyperparameter settings can be found in the next section of additional results and details.

5.4.2 Results

Graph Regression. Tab. 5.1 shows the ablation results of the vanilla 1-bit GNN models and those of GNNs with the proposed meta neighborhood aggregators GAN and ANA. Specifically, we report the results on two GNN architectures, *i.e.*, GCN [82] and GAT [153], by averaging over 25 independent runs with 25 seeds.

The proposed GNA and ANA, as shown in Tab. 5.1, achieves gratifying performance in terms of both test and train MAE, and at the same time maintains a compact model size. Moreover, we provide in the last row of Tab. 5.1 the p-value of the paired t-test between the 1-bit GNNs with a fixed aggregator (Vanilla) and those with the proposed learnable meta aggregators. The corresponding results statistically validate the effectiveness of the proposed method.

Furthermore, we show in Tab. 5.2 the results of extending the proposed meta aggregators to full-precision GNNs and compare them with those of the state-of-the-art approaches [7, 47, 180, 118, 82, 153]. Specifically, the results in the last two rows of Tab. 5.2 are obtained by simply replacing the pre-defined aggregator in GAT with the proposed GNA and ANA. As can be observed from Tab. 5.2, the proposed method outperforms other approaches by a large margin, and meanwhile introduces few additional parameters.

The proposed GNA and ANA, as shown in Tab. 5.3, yield results on par with those of the 32-bit full-precision models, but comes with a more lightweight architecture. The proposed method also outperforms the vanilla 1-bit GNN model that relies on a fixed aggregation scheme.

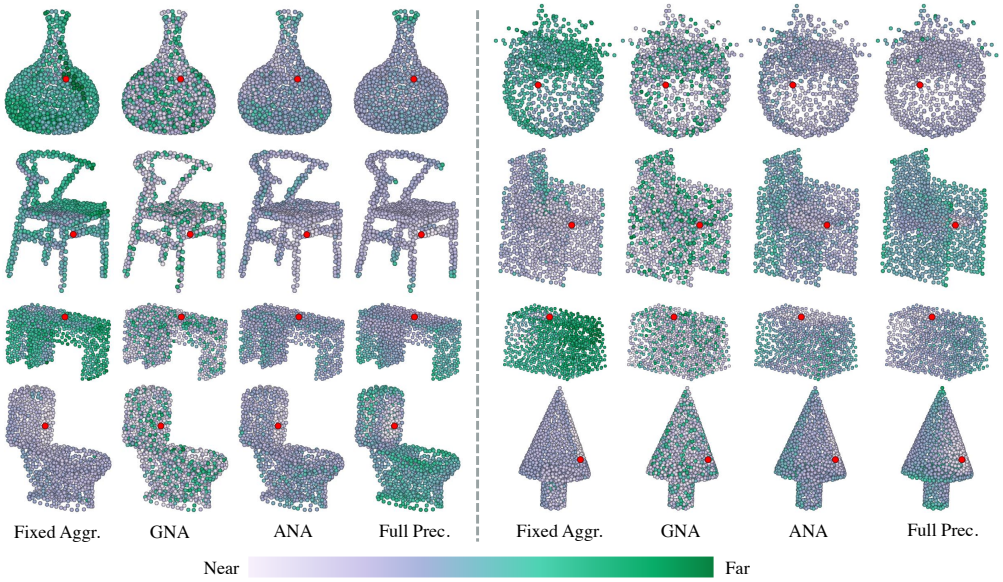


FIGURE 5.6: Visualization results of the learned feature space, depicted as the distance between the red point and the rest of the others. The visualized features are extracted from the intermediate layer of the models. More results can be found in the next section of additional results and details.

TABLE 5.4: Results on the ModelNet40 dataset for 3D object recognition, in terms of the overall accuracy (Acc) and the mean class accuracy (mAcc).

Methods	Bit-width	Param Size	Acc (%)	mAcc (%)
Full Prec. [167]	**32/32**	**1681.66KB**	**92.42**	**89.51**
Vanilla [61]	1/1	1091.20KB	74.19	65.95
GNA (Ours)	1/1	1091.30KB	78.36	71.67
ANA (Ours)	1/1	1091.30KB	84.64	78.89

We build our network here based on the architecture designed in [188]. We also demonstrate in Fig. 5.6 the corresponding visualization results of different approaches, where the column termed as "Fixed Aggr." in Fig. 5.6 corresponds to the "Vanilla" model in Tab. 5.4. With the proposed meta aggregation schemes, the 1-bit GNN model gains a boost by more than 10% in both the overall accuracy and the mean class accuracy. This improvement is also illustrated in Fig. 5.6, where the proposed meta aggregators help the 1-bit GNN learn a closer structure to that of the full-precision GNN model. Nevertheless, it's worth noting that the complexity of the point cloud analysis task leads to a considerable performance drop in comparison to the full-precision model. This issue will be a key focus for our future endeavors.

5.4.3 Discussions

We provide here a detailed account of the strengths and weaknesses of the proposed two meta aggregators GNA and ANA. For the exclusive meta form GNA, the performance can potentially be further enhanced with the advance of novel aggregation schemes. In other words, the results of GNA depend on those of every single aggregator in the candidate aggregation pool, which at the same time is a weakness of GNA since its performance is bottlenecked by that of the single aggregator. The diffused form ANA, on the other hand, may simultaneously retain the benefits of several popular aggregators. However, the mathematical form in Eq. 5.5 limits the type of aggregators that ANA can potentially approximate, meaning that ANA may not have much room for further improvement even with the emergence of novel and prevailing aggregators in the future.

5.5 Theoretical Analysis

In this section, we provide the propositions and the corresponding theoretical proofs on how and why the proposed *Adaptable Hybrid Neighborhood Aggregator (ANA)* can approximate various existing aggregation methods, such as max, mean, and variance.

We start by showing the mathematical form of ANA again, based on the *Log-Sum-Exp* function in convex optimization:

$$f(\mathcal{G},\mathcal{X}) = \frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log\left[\frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right], \tag{5.6}$$

where $\mathcal{A}^{\ell}$ denotes the 1-bit graph auto-encoder at layer ℓ. $\deg_i$ is the in-degree of the node v_i, and $\mathcal{G} = \{\mathcal{V},\mathcal{E}\}$ is the graph sample with edges $(v_i, v_j) \in \mathcal{E}$. $\mathcal{X}_j^{\ell}$ represents the feature vector of the neighboring node v_j at layer ℓ, whereas $f(\mathcal{G},\mathcal{X})$ denotes the obtained diffused aggregator.

Based on Eq. 5.6, we provide the following propositions and the corresponding proofs:

Proposition 5.5.1 (Mean)**.** *ANA, as defined in Eq. 5.6 as $f(\mathcal{G},\mathcal{X})$, can approximate the mean aggregator when $\mathcal{A}^{\ell}(\mathcal{G}) \to 0$.*

PROOF. We prove Proposition 3.2.1 primarily based on the inequality of arithmetic and geometric, defined as:

$$\frac{1}{n}\sum_{i=1}^{n} x_i \geq \Big(\prod_{i=1}^{n} x_i\Big)^{\frac{1}{n}}, \tag{5.7}$$

where the equality holds when $x_1 = x_2 = \cdots = x_n$.
By combining Eq. 5.6 and Eq. 5.7, we can derive the following inequation:

$$\begin{aligned} f(\mathcal{G},\mathcal{X}) &= \log\left[\frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} \\ &\geq \log\left[\prod_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\deg_i \mathcal{A}^{\ell}(\mathcal{G})}} = \log\left[\prod_{(j,i)\in\mathcal{E}} e^{\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\deg_i}}. \end{aligned} \tag{5.8}$$

The equality in Eq. 5.8 holds when $\mathcal{A}^{\ell}(\mathcal{G}) \to 0$, *i.e.*,

$$f(\mathcal{G},\mathcal{X}) = \log\left[\prod_{(j,i)\in\mathcal{E}} e^{\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\deg_i}} = \frac{1}{\deg_i}\log\left[\prod_{(j,i)\in\mathcal{E}} e^{\mathcal{X}_j^{\ell}}\right] = \frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{X}_j^{\ell}}. \tag{5.9}$$

Eq. 5.9 is, in fact, the formulation of the mean aggregation. Thus, the proposed AN can approximate the mean aggregator. □

Proposition 5.5.2 (Max)**.** *ANA defined in Eq. 5.6 can approximate the max aggregator when* $\mathcal{A}^{\ell}(\mathcal{G}) \to \infty$.

PROOF. To prove Proposition 3.2.2, we begin by reformulating Eq. 5.6 into:

$$\begin{aligned} f(\mathcal{G},\mathcal{X}) &= \frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log\left[\frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right] \\ &= \log\left[\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} - \frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log(\deg_i). \end{aligned} \tag{5.10}$$

Meanwhile, in our implementation, we keep $\mathcal{A}^{\ell}(\mathcal{G}) > 0$ by using an absolute operation. As such, we can also obtain the following inequation:

$$\left[\max_{(j,i)\in\mathcal{E}}\left(e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right)\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} \le \left[\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} \le \left[\deg_i \cdot \max_{(j,i)\in\mathcal{E}}\left(e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right)\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}}. \tag{5.11}$$

By combining Eq. 5.10 and Eq. 5.11, we can obtain:

$$\begin{aligned} \log\left[\max_{(j,i)\in\mathcal{E}}\left(e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right)\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} &\le \log\left[\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} \\ &\le \frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log(\deg_i) + \log\left[\max_{(j,i)\in\mathcal{E}}\left(e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right)\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}}. \end{aligned} \tag{5.12}$$

When $\mathcal{A}^{\ell}(\mathcal{G}) \to \infty$, we have: $\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log(\deg_i) \to 0$. As such, by replacing $\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log(\deg_i)$ with 0 in Eq. 5.12, we can obtain the following equation:

$$\log\left[\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} = \log\left[\max_{(j,i)\in\mathcal{E}}\left(e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right)\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}}. \tag{5.13}$$

By combing Eq. 5.13 and Eq. 5.10, and meanwhile replacing $\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log(\deg_i)$ in Eq. 5.10 with 0, we can derive the following equation:

$$f(\mathcal{G},\mathcal{X}) = \log\left[\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}} = \log\left[\max_{(j,i)\in\mathcal{E}}\left(e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right)\right]^{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}}. \tag{5.14}$$

The right part of Eq. 5.14 is, in fact, the mathematical form of the max aggregation method, which indicates that the proposed ANA can approximate the max aggregator when $\mathcal{A}^{\ell}(\mathcal{G}) \to \infty$. □

Proposition 5.5.3 (Variance)**.** *The variant of ANA, defined as $h(\mathcal{G},\mathcal{X})$ in Eq. 5.15, can approximate the variance aggregator when $\mathcal{A}^{\ell}(\mathcal{G}) \to 0$.*

$$\begin{aligned} h(\mathcal{G},\mathcal{X}) = \frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log\left[\frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,(\mathcal{X}_j^{\ell})^2}\right] - \\ \left\{\frac{1}{\mathcal{A}^{\ell}(\mathcal{G})}\log\left[\frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{A}^{\ell}(\mathcal{G})\,\mathcal{X}_j^{\ell}}\right]\right\}^2. \end{aligned} \tag{5.15}$$

PROOF. By combining Eq. 5.6 and Eq. 5.15, we can obtain: $h(\mathcal{G},\mathcal{X}) = f(\mathcal{G},\mathcal{X}^2) - [f(\mathcal{G},\mathcal{X})]^2$. When $\mathcal{A}^{\ell}(\mathcal{G}) \to 0$, we have:

$$h(\mathcal{G},\mathcal{X}) = \frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{(\mathcal{X}_j^{\ell})^2} - \left[\frac{1}{\deg_i}\sum_{(j,i)\in\mathcal{E}} e^{\mathcal{X}_j^{\ell}}\right]^2, \tag{5.16}$$

which is, in fact, the formulation of the variance aggregator. □

The final form of the proposed ANA consists of one term in the form of Eq. 5.6 and also one term in the form of Eq. 5.15, with the two corresponding graph auto-encoders $\mathcal{A}^\ell$ as well as two learnable weighting factors that determine the portion of each approximated aggregator. As such, the proposed ANA can potentially approximate various aggregators at the same time, and generate a hybrid behavior of different aggregators by controlling the weighting factors.

5.6 Additional Results

This section provides more results of the 1-bit graph neural networks (GNNs) with the proposed meta aggregators on the tasks of graph regression and multi-label node classification, as well as additional results of point cloud classification models.

5.6.1 Additional Results on Graph Regression Task

In this section, we provide additional results on the ZINC dataset for the task of graph regression. Specifically, we conduct extensive ablation studies on ZINC to validate the effectiveness of the proposed method.

Implementation Details. We use the ZINC dataset for the task of graph regression [63]. ZINC is a large-scale molecular dataset. The objective of ZINC is to regress a specific molecular property. The node features in every molecular graph represent the heavy atom types. The corresponding edge features denote the bond types between them. For the dataset splittings, we follow the standard splitting protocol in [32]. Specifically, 10,000 molecular graphs in ZINC are used for training, 1,000 graphs are for validation, and the remaining 1,000 ones are used for testing. In training, we use the Adam optimizer [80]. The batch size is set to 128. For the learning rate, we set the initial value as 10^{-3}, which is reduced by half if there is no improvement in the validation loss after 10 epochs. The training process is stopped when the learning rate reaches 10^{-3}. We measure the performance using the mean absolute error (MAE) between the predicted property and the ground-truth one. Detailed network architectures are summarized in Tab. 5.5.

TABLE 5.5: Detailed network architectures for the task of graph regression on the ZINC dataset, where Architecture-ZINC-Main-GAT and Architecture-ZINC-Main-GCN represent the architectures of the two models shown in Tab. 5.1 and Tab. 5.2. Architecture-ZINC-Supp denotes the architectures that will be used for ablation studies in this section.

Models	**Layers**	**Attention Heads**	**Hidden**	**Output**
Architecture-ZINC-Main-GAT	6	{8, 8, 8, 8}	18	144
Architecture-ZINC-Main-GCN	6	–	145	145
Architecture-ZINC-Supp-V1	5	{8, 8, 8}	18	144
Architecture-ZINC-Supp-V2	5	{8, 8, 8}	22	176
Architecture-ZINC-Supp-V3	8	{8, 8, 8, 8, 8, 8}	22	176

TABLE 5.6: Results on the ZINC dataset for the task of graph-property regression, in terms of the mean absolute error (MAE). The detailed network architectures of Architecture-ZINC-Supp-V1 and Architecture-ZINC-Supp-V2 are shown in Tab. 5.5.

Architecture	**Architecture-ZINC-Supp-V1**				**Architecture-ZINC-Supp-V2**			
Methods	Full Prec. [153]	Vanilla [61]	**GNA (Ours)**	**ANA (Ours)**	Full Prec. [153]	Vanilla [61]	**GNA (Ours)**	**ANA (Ours)**
Bit-width	32/32	1/1	**1/1**	**1/1**	32/32	1/1	**1/1**	**1/1**
Param Size	316.691KB	78.0156KB	**78.2811KB**	**78.1226KB**	466.816KB	111.164KB	**111.488KB**	**111.294KB**
Test MAE±SD	0.495±0.008	0.647±0.064	**0.598±0.022**	**0.576±0.031**	0.496±0.006	0.687±0.081	**0.590±0.020**	**0.566±0.015**
Train MAE±SD	0.372±0.017	0.588±0.065	**0.536±0.024**	**0.471±0.035**	0.362±0.013	0.629±0.083	**0.523±0.022**	**0.444±0.024**
***p*-value**	GNA vs. Vanilla: 6.316×10^{-4}/ANA vs. Vanilla: 7.101×10^{-6}				GNA vs. Vanilla: 3.869×10^{-7}/ANA vs. Vanilla: 1.768×10^{-9}			

Ablation Studies. We show in Tab. 5.6 the regression results of the 1-bit GNNs with different network architectures. Specifically, from left to right, Tab. 5.6 shows the results of the full-precision GNNs (Full Prec.), those of the 1-bit GNNs without the proposed meta aggregators (Vanilla), and the results of the 1-bit GNNs with GNA and ANA. In the last line of Tab. 5.6, we also provide the p-value of the paired t-test between the proposed meta aggregator and the vanilla one, so as to demonstrate the statistically meaningful improvements by the proposed GNA and ANA. It is noticeable that both GNA and ANA achieve performance superior to that of the vanilla one that depends on a single fixed and pre-defined aggregator.

Furthermore, we provide in Tab. 5.7 the results of the 32-bit models with the proposed GNA and ANA, corresponding to Tab. 5.2 but with a different additional network architecture. The proposed meta aggregators, as shown in Tab. 5.7, also achieve results superior to the state-of-the-art on the full-precision models.

TABLE 5.7: Results of the proposed GNA and ANA as well as other methods for 32-bit full-precision models on the ZINC dataset, in terms of MAE. The detailed network architectures of the proposed methods are shown as Architecture-ZINC-Supp-V3 in Tab. 5.5. For the architectures of the comparison methods [7, 47, 180, 118, 82, 153], we follow the network architecture designs in [32].

Methods	Param Size	Test MAE±SD	Train MAE±SD	Methods	Param Size	Test MAE±SD	Train MAE±SD
GraphSage [47]	1973.99KB	0.398±0.002	0.081±0.009	RingGNN [20]	2059.70KB	0.353±0.019	0.236±0.019
GIN [180]	1990.43KB	0.526±0.051	0.444±0.039	MoNet [118]	1968.80KB	0.292±0.006	0.093±0.014
GCN [82]	1972.96KB	0.367±0.011	0.128±0.019	GAT [153]	2075.57KB	0.384±0.007	0.067±0.004
GNA (Ours)	**858.809KB**	**0.295±0.013**	**0.088±0.016**	**ANA (Ours)**	**846.410KB**	**0.294±0.010**	**0.079±0.018**

TABLE 5.8: Summary of the detailed network architectures for the task of multi-label node classification on the PPI dataset.

Models	Layers	Attention Heads	Hidden
Architecture-PPI-Main	3	{4, 4, 6}	512
Architecture-PPI-Supp-V1	3	{4, 4, 6}	256
Architecture-PPI-Supp-V2	5	{2, 2, 2, 2, 2}	128
Architecture-PPI-Supp-V3	5	{2, 2, 2, 2, 2}	64

5.6.2 Additional Results on Multi-label Node Classification Task

In this section, we show more results on the PPI dataset for the task of multi-label node classification. Specifically, we provide here additional results with three newly designed network architectures.

Implementation Details. We use the protein-protein interaction (PPI) dataset for the task of multi-label node classification, containing biological graphs with the nodes that are labeled with different protein functions [226]. In particular, each node can simultaneously have several labels. In training, the batch size is set to 1. The learning rate is 0.005 for each model. In total, we optimize all the models for 500 epochs and report the corresponding results with the best validation accuracies. The detailed network architectures are demonstrated in Tab. 5.8. Specifically, the 2nd row of Tab. 5.8 shows the architecture used in Tab. 5.3. The 3rd, 4th, and 5th rows, on the other hand, correspond to three newly-designed architectures that are used in the following extensive ablation studies.

Ablation Studies. We perform here ablation studies on various network architectures for the task of multi-label node classification. The corresponding results are shown in Tab. 5.9. The proposed GNA and ANA, as can be seen from Tab. 5.9, delivers competitive results as compared with those of the full-precision-based approach across all the four distinct architectures, yet maintaining a compact model size. Also, with a similar lightweight architecture, the 1-bit GNNs with the proposed meta aggregators achieve superior performance to that of the model with a pre-defined aggregator, demonstrating the effectiveness of the proposed learnable aggregation schemes.

5.6.3 Additional Results on 3D Object Recognition Task

We provide in this section more results on the ModelNet40 dataset for the task of 3D object classification. Specifically, we devise two additional architectures and conduct extensive ablation studies accordingly.

Implementation Details. We follow the official dataset splitting protocol in [175, 167]. We set the learning rate as 0.001 and use a batch size of 16. We adopt the Adam optimizer [80] and all the models are optimized for 1000 epochs for full convergence. The detailed architecture designs are summarized in Tab. 5.10, where the 2nd row corresponds to the architecture used

TABLE 5.9: Results on the PPI dataset for the task of node classification, in terms of micro-averaged F_1 score. Detailed network architectures of Architecture-PPI-Main as well as Architecture-PPI-Supp-V1, V2, and V3 can be found in Tab. 5.8.

Architecture	**Architecture-PPI-Main**			**Architecture-PPI-Supp-V1**		
Methods	Bit-width	Param Size	F_1 Score	Bit-width	Param Size	F_1 Score
Full Prec. [82]	32/32	43.7712MB	98.70	32/32	13.8884MB	98.67
Vanilla [61]	1/1	28.2560MB	92.68	1/1	10.0058MB	93.27
GNA (Ours)	**1/1**	**28.2572MB**	**97.52**	**1/1**	**10.0064MB**	**96.79**
ANA (Ours)	**1/1**	**28.2565MB**	**97.71**	**1/1**	**10.0060MB**	**97.02**
Architecture	**Architecture-PPI-Supp-V2**			**Architecture-PPI-Supp-V3**		
Methods	Bit-width	Param Size	F_1 Score	Bit-width	Param Size	F_1 Score
Full Prec. [82]	32/32	2.0311MB	98.21	32/32	0.64150MB	94.80
Vanilla [61]	1/1	1.2989MB	48.54	1/1	0.45702MB	45.88
GNA (Ours)	**1/1**	**1.2994MB**	**53.52**	**1/1**	**0.45725MB**	**53.94**
ANA (Ours)	**1/1**	**1.2991MB**	**69.24**	**1/1**	**0.45711MB**	**72.29**

FIGURE 5.7: Comparative visualization results. Node color encodes the distance between the red dot and node of interest. All the visualized features are extracted from the intermediate layer of the models.

in Tab. 5.4. The 3rd and 4th rows, on the other hand, demonstrate the architectures that are used in this section for extensive ablation studies.

TABLE 5.10: Summary of the detailed network architectures for the task of 3D object recognition on ModelNet40.

Models	Layers	Feature Map Channels	MLPs
Architecture-ModelNet40-Main	6	[64, 64, 128, 512]	[256, 40]
Architecture-ModelNet40-Supp-V1	6	[32, 32, 64, 128]	[256, 40]
Architecture-ModelNet40-Supp-V2	8	[64, 64, 128, 256, 1024]	[512, 256, 40]

TABLE 5.11: Results on the ModelNet40 dataset for the task of 3D object recognition, in terms of the overall accuracy (Acc) and the mean class accuracy (mAcc). The details of Architecture-ModelNet40-Supp-V1 and Architecture-ModelNet40-Supp-V2 are shown in Tab. 5.10.

Architecture	Architecture-ModelNet40-Supp-V1				Architecture-ModelNet40-Supp-V2			
Methods	Bit-width	Param Size	Acc	mAcc	Bit-width	Param Size	Acc	mAcc
Full Prec. [167]	32/32	388.906KB	92.30%	89.43%	32/32	7068.66KB	93.03%	89.70%
Vanilla [61]	1/1	302.930KB	55.79%	46.28%	1/1	4742.20KB	81.12%	73.88%
GNA (Ours)	**1/1**	**303.023KB**	**60.66%**	**49.74%**	**1/1**	**4742.39KB**	**81.65%**	**75.23%**
ANA (Ours)	**1/1**	**302.962KB**	**74.27%**	**65.96%**	**1/1**	**4742.33KB**	**84.81%**	**78.97%**

Ablation Studies. We provide in Tab. 5.11 the results of different approaches with the two newly-designed architectures. The quantitative results in Tab. 5.11 indicate that both of the proposed GNA and ANA can boost the performance of 1-bit GNNs by a large margin across various network architectures, as compared with the vanilla fixed aggregation method [61].

More Qualitative Results. We also show in Fig. 5.7 more qualitative results of different approaches, by visualizing the structures of the learned feature spaces. It can be observed that the proposed GNA and ANA can facilitate the 1-bit GNNs to learn a more similar feature structure to that of the cumbersome full-precision ones.

5.7 Summary

This chapter explores a couple of learnable aggregation schemes for 1-bit compact GNNs. The goal of the proposed method is to enhance the topological discriminative ability of the 1-bit GNNs. This is achieved by adaptively selecting a single aggregator, or generating a hybrid aggregation form that can simultaneously maintain the strengths of several aggregators. Moreover, the proposed meta aggregation schemes can be readily extended to the full-precision

GNN models. Experiments across various domains demonstrate that, with the proposed meta aggregators, the 1-bit GNN yields results on par with those of the cumbersome full-precision ones. In our future work, we will strive to generalize the proposed aggregator to compact and lightweight visual transformers.

CHAPTER 6

Application-Driven Efficient Learning with Semi-parametric Style Transfer

---

While the previous three chapters primarily focused on the development of *universal* efficient learning schemes, this chapter takes a different approach by introducing an alternative application-driven perspective. The primary objective of this chapter is to investigate a *customized* efficient representation learning paradigm that is specifically tailored for the application of image style transfer. In particular, this chapter studies a novel semi-parametric neural style transfer framework that alleviates the deficiency of both parametric and non-parametric stylization. The core idea of the proposed approach is to efficiently establish accurate and fine-grained content-style correspondences using graph neural networks. To this end, this chapter develops an elaborated GNN model with content and style local patches as the graph vertices. The style transfer procedure is then modeled as the attention-based heterogeneous message passing between the style and content nodes in a learnable manner, leading to adaptive many-to-one style-content correlations at the local patch level. In addition, an elaborated deformable graph convolutional operation is introduced for cross-scale style-content matching. Experimental results demonstrate that the proposed semi-parametric image stylization approach efficiently yields encouraging results on the challenging style patterns, preserving both global appearance and exquisite details.

6.1 Introduction

Image style transfer aims to automatically transfer the artistic style from a source style image to a given content one, and has been studied for a long time in the computer vision community.

Conventionally, image style transfer is generally cast as the problem of non-photorealistic rendering in the domain of computer graphics. Inspired by the success of deep learning [29, 146, 30, 193, 28], Gatys *et al*. [39] pioneer the paradigm that leverages the feature activations from deep *convolutional neural networks (CNNs)* to extract and match the target content and style, leading to the benefits of no explicit restrictions on style types and no requirements of ground-truth training data. As such, various CNN-based style transfer methods are developed in the literature [78, 86, 14, 181, 173, 62, 54, 100, 103], establishing a novel field of *neural style transfer (NST)* [72].

State-of-the-art NST algorithms can be categorized into two streams of methods, parametric and non-parametric ones, depending on the style representation mechanisms. In particular, parametric NST approaches rely on the global summary statistics over the entire feature map from pre-trained deep CNNs to extract and match the target artistic style [39, 75, 59]. Non-parametric neural methods, also known as patch-based NST methods [19, 147], leverage the local feature patches to represent the style information, inspired by the conventional patch-based texture modeling approaches with Markov random fields. The idea is to swap the content neural patches with the most similar style ones, through a greedy one-to-one patch matching strategy.

Both parametric and non-parametric methods, unfortunately, have their own limitations, as demonstrated in Fig. 6.1. Parametric stylization methods achieve good performance in transferring the overall appearance of the style images, but are incompetent in generating fine-grained local style patterns. By contrast, non-parametric style transfer algorithms allow for locally-aligned stylization; however, such patch-based methods are typically accomplished with the undesired artifacts due to content-style mismatching.

In this chapter, we present a semi-parametric style transfer scheme, towards alleviating the dilemmas of existing parametric and non-parametric methods. On the one hand, our semi-parametric approach allows for the establishment of more accurate many-to-one correspondences between different content and style regions in a learnable manner. As such, our approach explicitly tackles the issue of content-style mismatching in non-parametric NST algorithms, thereby largely alleviating the deficiency of unplausible artifacts. On the other

(a) Content (b) Huang *et al.* [59] (c) An *et al.* [1] (d) Li *et al.* [96]

(e) Style (f) Chen *et al.* [19] (g) Sheng *et al.* [147] (h) Ours

FIGURE 6.1: Existing parametric *(b,c,d)* and non-parametric *(f,g)* NST methods either barely transfer the global style appearance to the target *(f)*, or produce distorted local style patterns *(b,c,d)* and undesired artifacts *(g)*. By contrast, the proposed GNN-based approach *(h)* achieves superior stylization performance in the transfers of both global stroke arrangement and local fine-grained patterns.

hand, the proposed semi-parametric method adaptively divides content and style features into tiny and cross-scale feature patches for stylization, thus addressing the dilemma of lacking local details in prior parametric schemes.

Towards this end, we introduce to the proposed semi-parametric NST a dedicated learning mechanism, *graph neural networks* (GNNs), to enable adaptive local patch-level interplay between the content and style. As a well-established learning paradigm for handling non-Euclidean data, GNNs are designed to explicitly account for structural relations and interdependency between nodes. Moreover, GNNs are equipped with efficacious strategies for aggregating information from multiple neighbors to a center node. Such competences make GNN an ideal tool for tackling the intricate content-style region matching challenge in style transfer, especially the many-to-one mapping between each content patch and multiple potentially-matching style patches. We therefore exploit GNNs to adaptively set up the

faithful topological correspondences among the very different content and style, such that every content region is rendered with the optimal style strokes.

Specifically, we start by building a heterogeneous NST graph, with content and style feature patches as the vertices. The multi-patch parametric aggregation in semi-parametric NST can thereby be modeled as the message passing procedure among different patch nodes in the constructed stylization graph. By employing the prevalent GNN mechanisms such as the graph attention network, the k most similar patches can be aggregated in an attention-based parametric manner. The aggregated patches are then composed back into the image features, which are further aligned with the target global statistics to obtain the final stylized result. Also, a deformable graph convolutional operation is devised, making it possible for cross-scale style-content matching with spatially-varying stroke sizes in a single stylized image. Furthermore, our GNN-based NST can readily perform diversified patch-based stylization, by simply changing the number of connections during inference.

In sum, our contribution is a novel semi-parametric arbitrary stylization scheme that allows for the effective generation of both the global and local style patterns, backed by a dedicated deformable graph convolutional design. This is specifically achieved through modeling the NST process as the message passing between content and style under the framework of GNNs. Experimental results demonstrate that the proposed GNN-based stylization method yields results superior to the state of the art.

6.2 Proposed Method

Towards addressing the limitations of existing parametric and non-parametric NST methods, we introduce the proposed semi-parametric style transfer framework with GNNs. In what follows, we begin by providing an overview of the proposed GNN-based approach, and then elaborating several key components, including the construction of the topological NST graph, the dedicated deformable graph convolutional operation customized for the established NST graph, and the detailed 2-hop heterogeneous message passing process for stylization.

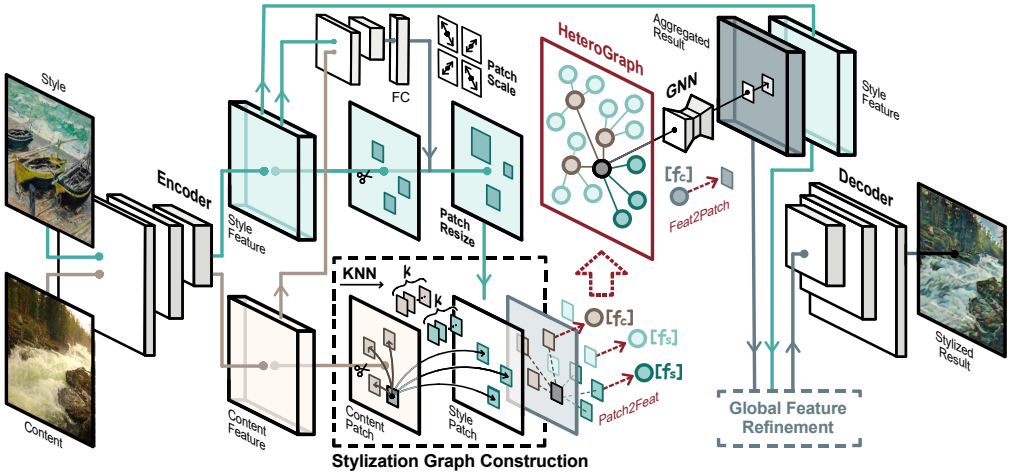


FIGURE 6.2: Network architecture of the proposed semi-parametric style transfer network with GNNs. From left to right, the corresponding stylization pipeline comprises four subprocesses, *i.e.*, image encoding with the encoder, local patch-based manipulation based on heterogeneous GNNs, global feature refinement, and the feature decoding procedure. The symbols of scissors represent the process to divide the feature maps into feature patches. HeteroGraph denotes the established heterogeneous stylization graph with two types of content-style inter-domain connections and content-content intra-domain connections.

Finally, we illustrate the cascaded patch-to-image training pipeline, tailored for the proposed GNN-based stylization system.

6.2.1 Network Overview

The overall workflow of the proposed semi-parametric NST framework is shown in Fig. 6.2. There are primarily four modules in the whole pipeline, termed as *image encoding*, *local patch-based manipulation*, *global feature refinement*, and *feature decoding*. At the heart of the proposed framework is the *local patch-based manipulation* module, which will be further detailed in the following sections.

Image Encoding Module. The proposed semi-parametric stylization starts by receiving style and content images as inputs and encoding these images into meaningful feature maps (the green and yellow blocks in Fig. 6.2), by exploiting the first few layers of the pre-trained VGG network. In particular, unlike the existing work [59] that uses the layers before `relu4_1`,

we leverage the VGG layers up to `relu3_1`, for the sake of more valid feature patches that can be exploited by the following local patch-based feature transformation stage.

Local Patch-based Manipulation Module. With the embeded content and style features as inputs, the local patch-based manipulation module extracts the corresponding content and style feature patches with the stride of s and the sliding window size of $p \times p$, represented as the scissor symbol in Fig. 6.2. We then build a heterogeneous stylization graph (the red frame in Fig. 6.2) with the obtained feature patches as graph nodes and perform the dedicated deformable graph convolution to generate the locally-stylized features, which will be further detailed in the succeeding Sect. 6.2.2 and Sect. 6.2.3.

Global Feature Refinement Module. The produced style-transferred results from the stage of patch-based manipulation are effective at preserving fine-grained local style patterns; however, the global style appearance is likely to be less similar to the target style image, due to the lack of global constraint on the stroke arrangement. To alleviate this dilemma, we propose a hierarchical patch-to-image stylization scheme to yield both the exquisite brushstroke and large-scale texture patterns. This is achieved by refining the feature representations at a global level, subsequent to the local patch-based manipulation. For the specific refinement method, since there already exist several effective global feature decorated strategies in the field of NST (*e.g.*, adaptive instance normalization (AdaIN) [59] and zero-phase component analysis (ZCA) [96]), here we directly utilize AdaIN as our refinement scheme, considering its high efficiency.

Feature Decoding Module. The last stage of our semi-parametric style transfer pipeline, termed as feature decoding, aims to decode the obtained feature representations from the preceding global feature refinement module into the final stylized image. The decoder module specifically comprises a sequence of convolutional and bilinear upsampling layers with the ReLU nonlinearities.

In the following sections, we will explain more details regarding the key module of *Local Patch-based Manipulation* with GNNs, including the graph construction procedure and the deformable graph convolutional process.

6.2.2 Stylization Graph Construction

At the stage of local patch-based manipulation, the first challenge towards the adaptive patch-level interactions between content and style with GNNs is the establishment of topological graphs. Unlike conventional GNN-based applications where the inputs can be naturally modeled as graphs (*e.g.*, biological molecules and social networks), there is no such natural topological structure for our task of semi-parametric image style transfer. To address this issue, we develop a dedicated graph construction technique, tailored for image stylization.

We start by giving the mathematical model of general graph-structured data as: $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{G}$ represents a directed or undirected graph. $\mathcal{V}$ denotes the set of vertices with nodes $v_i \in \mathcal{V}$. $\mathcal{E}$ represents the edge set with $(v_i, v_j) \in \mathcal{E}$, where $\{v_j\}$ is the set of neighboring nodes of v_i. Each vertex has an associated node feature $\mathcal{X} = [x_1\ x_2\ ...\ x_n]$. For example, x can be defined as the 3D coordinates in the task of point cloud classification.

As can be observed from the above formulation of prevalent graph data, the key elements in a graph are the vertices with the corresponding node features as well as the edges, which are thereby identified as our target objects to instantiate in the domain of style transfer as follows:

Heterogeneous Patch Vertices. To leverage GNNs to benefit the local-level stylization, we model in our framework the content and style patches as the graph nodes. Specifically, we exploit the content and style feature activations from the pre-trained VGG encoder, shown as the green and yellow blocks in Fig. 6.2, respectively, to capture the corresponding feature patches with a sliding window (*i.e.*, the scissor symbol in Fig. 6.2), in a similar manner as what is done when performing convolutions. We set the stride as 1 by default, meaning that there exist overlaps among our extracted activation patches. Such a manner of overlapped patch generation allows for smooth transitions among different stylized regions. In particular, to achieve cross-scale patch matching, we perform multi-scale patch division, which will be demonstrated in detail as a part of the deformable convolution in Sect. 6.2.3.

For the definition of the associated features for each patch vertex, we use a `Patch2Feat` operation, depicted as the red fonts in Fig. 6.2, to produce the desired format of node features

for the use of the subsequent GNN layers, as also done in [223]. The designed `Patch2Feat` operation specifically amalgamates the c-dimensional features at each position of the $p \times p$ activation patch into a 1-dimensional feature vector, which is then considered as the node feature at every patch vertex. The derived content and style node features are shown as $[f_c]$ and $[f_s]$ in Fig. 6.2, respectively, for the use of the latter GNN layers.

Inter- and Intra-KNN Edges. Another critical issue in building the stylization graph is the establishment of connections among different patch vertices. Customized for the task of style transfer, we formulate two types of edges, termed as *content-style inter-domain edges* and *content-content intra-domain edges*, leading to a special kind of heterogeneous graph.

In particular, the inter-domain connections between heterogeneous style and content nodes aim to attain more accurate many-to-one style-content matching for patch-based stylization. More specifically, for each content query patch $\phi_i(\mathcal{F}_c)$ with $\mathcal{F}_c$ representing the whole content feature map, we search the corresponding k-nearest ones in the set of style feature patches $\{\phi(\mathcal{F}_s)\}$, which are identified as the neighbors coupled with inter-domain edges. This process of k-nearest neighbor search (KNN) is shown in the black dotted frame in Fig. 6.2. We employ the distance metric of normalized cross-correlation (NCC) for pair-wise KNN, by scoring the cosine distance between a couple of content and style patches. Given a specific content patch $\phi_i(\mathcal{F}_c)$ as the query, our KNN procedure based on NCC can be specifically formulated as:

$$\mathrm{KNN}(\phi_i(\mathcal{F}_c), \{\phi(\mathcal{F}_s)\}) = \underset{j \in \{1,\ldots,N_s\}}{\arg\max_{\mathrm{k}}} \frac{\langle \phi_i(\mathcal{F}_c), \phi_j(\mathcal{F}_s) \rangle}{\|\phi_i(\mathcal{F}_c)\| \|\phi_j(\mathcal{F}_s)\|}, i \in \{1, \ldots, N_c\}, \tag{6.1}$$

where N_c and N_s denote the cardinalities of the corresponding content and style patch sets, respectively. $\max_{\mathrm{k}}$ returns the k largest elements from the set of the computed pair-wise NCCs. $\mathrm{KNN}(\phi_i(\mathcal{F}_c))$ represents the target k nearest-neighboring style vertices for the content patch $\phi_i(\mathcal{F}_c)$.

We also introduce the intra-domain connections within the set of content activation patches in our stylization graph, shown as the brown arrows in the black dotted rectangle in Fig. 6.2. The goal of such content-to-content edges is to unify the transferred styles across different content patches. In other words, we utilize the devised intra-domain connections to make sure

that the semantically-similar content regions will also be rendered with homogeneous style patterns. This is specifically accomplished by linking the query content patch $\phi_i(\mathcal{F}_c)$ with the top-k most similar patches $\{\phi_j(\mathcal{F}_c)\}$ where $j \in \{1, \ldots, N_c\}$, by NCC-based KNN search in a similar manner with that in Eq. 6.1.

The ultimate heterogeneous stylization graph, with the two node types of content and style vertices and also the two edge types of inter- and intra-domain connections, is demonstrated as the red rectangle in Fig. 6.2. The relationship between the involved nodes is defined as the NCC-based patch similarity.

6.2.3 Deformable Graph Convolution

With the constructed stylization graph, we are then ready to apply GNN layers to perform heterogeneous message passing along the content-style inter-domain edges and also content-content intra-domain edges. A naïve way will be simply performing existing graph convolutions on the heterogeneous stylization graph to aggregate messages from the content and style vertices.

However, this vanilla approach is not optimal for the task of style transfer, due to a lack of considerations in feature scales. Specifically, in the process of image stylization, the proper feature scale is directly correlated with the stroke scale in the eventual output [74], which is a vital geometric primitive to characterize an artwork. The objective stylized results should have various scales of style strokes across the whole image, depending on the semantics of different content regions.

Towards this end, we propose a dedicated deformable graph convolutional network that explicitly accounts for the scale information in message passing. The devised deformable graph convolutional network comprises two components. Specifically, the first component is an elaborated *deformable scale prediction module*, with a fully-connected (FC) layer in the end, that aims to generate the optimal scale of each patch in a learnable manner before conducting message aggregation, as also done in [21]. In particular, the scale predictor receives

both the content and style features as inputs, considering the potential scale mismatching between the content and style, as shown in the upper left part of Fig. 6.2.

As such, by adaptively performing scale adjustment according to both content and style inputs, the proposed deformable graph conventional network makes it possible for cross-scale style-content matching with spatially-varying stroke sizes across the whole image. We clarify that we only incorporate one-single predictor in our deformable graph convolutional network that produces the style scales, for the sake of computational efficiency. There is no need to also augment another predictor for content scale prediction, which is, in fact, equivalent to fixing the content scale and only changing the style one.

The second component of the proposed deformable graph convolutional network is the *general feature aggregation module* that learns to aggregate the useful features from the neighboring heterogeneous content and style nodes. Various existing message passing mechanisms can, in fact, readily be applied at this stage for message propagation. Here, we leverage the graph attention scheme to demonstrate the message flow along with the two types of stylization edges, which empirically leads to superior stylization performance thanks to its property of anisotropy.

Specifically, given an established stylization graph, our dedicated heterogeneous aggregation process is composed of two key stages, termed as *style-to-content message passing stage* and *content-to-content messing passing stage*:

Style-to-Content Message Passing. The first style-to-content stage aims to gather the useful style features from the k neighboring style vertices. For the specific message gathering method, one vanilla way is to treat the information from every style vertex equally, meaning that the aggregated result would be simply the sum of all the neighboring style node features. However, the results of such naïve approach are likely to be affected by the noisy style vertices, resulting in undesired artifacts.

To tackle this challenge, we apply an attention coefficient for each style vertex during message passing, which is learned in a data-driven manner. Given a centering content node v_c and its neighboring style nodes $\{v_s\}$ with the cardinality of k, the learned attention coefficients

$w(v_c, v_s^j)$ between v_c and a specific neighbor v_s^j can be computed as:

$$w(v_c, v_s^j) = \frac{\exp\left(\text{LeakyReLU}\left(W_a[W_b\mathcal{F}_c \| W_b\mathcal{F}_s^j]\right)\right)}{\sum_{m=1}^{k}\exp\left(\text{LeakyReLU}\left(W_a[W_b\mathcal{F}_c \| W_b\mathcal{F}_s^m]\right)\right)}, \tag{6.2}$$

where W represents the learnable matrix in linear transformation. $\|$ is the concatenation operation.

With such an attention-based aggregation manner, our stylization GNN can adaptively collect more significant information from the best-matching style patches, and meanwhile reduce the features from the less-matching noisy ones. Furthermore, we also apply a multi-headed architecture that generates the multi-head attention, so as to stabilize the attention learning process.

Content-to-Content Message Passing. With the updated node features at the content vertices from the preceding style-to-content message passing process, we also perform a second-phase information propagation among different content nodes. The rationale behind our content-to-content message passing is to perform global patch-based adjustment upon the results of the style-to-content stage, by considering the inter-relationship between the stylized patches at different locations. As such, the global coherence can be maintained, where the content objects that share similar semantics are more likely to resemble each other in stylization, which will be further validated in the experiments.

This proposed intra-content propagation also delivers the benefit of alleviating the artifacts resulting from potential style-content patch mismatching, by combining the features from the correctly-matching results. The detailed content-to-content message passing procedure is analogous to that in style-to-content message passing, but replacing the style vertices in Eq. 6.2 with the neighboring content vertices with the associated updated node features.

The eventual aggregation results from the proposed inter- and intra-domain message passing are then converted back into the feature patches by a `Feat2Patch` operation, which is an inverse operation of `Patch2Feat`. The obtained patches are further transformed into the feature map for the use of the subsequent global feature alignment module and feature decoding module.

Algorithm 3 Training a GNN-based stylization model that can transfer arbitrary styles in a semi-parametric manner.

Input: $\mathcal{I}_c$: the content image; $\mathcal{I}_s$: the style image; `VGG`: the pre-trained loss network.
Output: $\mathcal{I}_o$: Target stylized image that simultaneously preserves the appearance of $\mathcal{I}_s$ and the semantics of $\mathcal{I}_c$.

Perform initializations on the image encoder $\texttt{Enc}(\cdot)$, the scale predictor $\texttt{Prec}(\cdot)$, GNN parameters W_a and W_b, and the feature decoder $\texttt{Dec}(\cdot)$.
for $i = 1$ to $\mathcal{T}$ iterations **do**
    Feed $\mathcal{I}_s$ and $\mathcal{I}_c$ into $\texttt{Enc}(\cdot)$ and obtain the style and content features $\mathcal{F}_s$ and $\mathcal{F}_c$;
    Divide $\mathcal{F}_c$ into equal-size content patches $\{\phi(\mathcal{F}_c)\}$ by using a sliding window;
    Feed $\{\mathcal{F}_s, \mathcal{F}_c\}$ into $\texttt{Prec}(\cdot)$ and obtain the optimal scales $\{\alpha\}$ for style patches;
    Divide $\mathcal{F}_s$ into varying-size style patches $\{\phi(\mathcal{F}_s)\}$ with the obtained scales $\{\alpha\}$;
    Resize $\{\phi(\mathcal{F}_s)\}$ according to the size of the content patches $\{\phi(\mathcal{F}_c)\}$;
    Construct inter- and intra-domain edges by Eq. 6.1;
    Transform $\{\phi(\mathcal{F}_s)\}$ and $\{\phi(\mathcal{F}_c)\}$ into the node features by using `Patch2Feat`;
    Establish the heterogeneous graph $\mathcal{G}_{NST}$ and feed $\mathcal{G}_{NST}$ into the GNN layers;
    Perform heterogeneous message passing over $\mathcal{G}_{NST}$ by Eq. 6.2 and obtain $\texttt{f}_\texttt{c}$;
    Convert the aggregation results $\texttt{f}_\texttt{c}$ into feature map $\mathcal{F}_o$ by `Feat2Patch`;
    Feed the obtained features $\mathcal{F}_o$ into the global feature refiner and obtain $\mathcal{F}'_o$;
    Feed $\mathcal{F}'_o$ into the decoder $\texttt{Dec}(\cdot)$ to obtain the target stylized image $\mathcal{I}_o$;
    Feed $\{\mathcal{I}_o, \mathcal{I}_c, \mathcal{I}_s\}$ into `VGG` and compute $\mathcal{L}_c$ and $\mathcal{L}_s$ by Eq. 6.3 and Eq. 6.4;
    Optimize $\texttt{Enc}(\cdot)$, $\texttt{Prec}(\cdot)$, W_a, W_b, and $\texttt{Dec}(\cdot)$ with the Adam optimizer;
end for

6.2.4 Loss Function and Training Strategy

To align the semantic content, our content loss $\mathcal{L}_c$ is defined as the perceptual loss over the features from layer $\{\texttt{relu4\_1}\}$ of the pre-trained VGG network Φ:

$$\mathcal{L}_c = \|\Phi^{\texttt{relu4\_1}}(\mathcal{I}_c) - \Phi^{\texttt{relu4\_1}}(\mathcal{I}_o)\|_2, \tag{6.3}$$

where $\mathcal{I}_c$ and $\mathcal{I}_o$ represent the content and the output stylized images, respectively. For the style loss, we use the BN-statistic loss to extrat and transfer the style information, computed at layer $\{\texttt{relu1\_1}, \texttt{relu2\_1}, \texttt{relu3\_1}, \texttt{relu4\_1}\}$ of the VGG network Φ:

$$\mathcal{L}_s(h) = \sum_{\ell=1}^{4} \Big(\Big\| h\left(\Phi^{\texttt{relu}\ell\texttt{\_1}}\left(\mathcal{I}_s\right)\right) - h\left(\Phi^{\texttt{relu}\ell\texttt{\_1}}\left(\mathcal{I}_o\right)\right) \Big\|_2 \Big), \tag{6.4}$$

where $h(\cdot)$ denotes the mapping of computing the BN statistics over the feature maps. The style loss can then be defined as: $\mathcal{L}_s = \mathcal{L}_s(\mu) + \mathcal{L}_s(\sigma)$, with $\mu(\cdot)$ and $\sigma(\cdot)$ denoting mean and standard standard deviation, respectively.

Our total loss is thereby a weighted sum of the content and style loss, formulated as: $\mathcal{L} = \mathcal{L}_{content} + \lambda \mathcal{L}_{style}$ with λ as the weighting factor that balances the content and style portions.

We also derive an elaborated training pipeline, tailored for the proposed GNN-based semi-parametric style transfer framework. As a whole, the detailed process of training a GNN-based semi-parametric arbitrary stylization model with the proposed algorithm is concluded in Alg. 3.

6.3 Experiments

6.3.1 Experimental Settings

We demonstrate here the implementation details as per the stage of the proposed semi-parametric pipeline. For the stylization graph construction stage, we set k as 5 by default for the NCC-based KNN search. The stride s for the sliding window is set to 1, whereas the kernel size is set to 5×5. At the stage of deformable graph convolution, we primarily use the graph attention network (GAT) [153] for the GNN layers to validate the effectiveness of the proposed semi-parametric NST scheme. During training, we adopt the Adam optimizer [81] to optimize the whole GNN-based network. The learning rate is 1×10^{-4} with a weight decay of 5×10^{-5}. The batch size is set to 8. The weighting factor λ is set to 10. We employ a pre-trained VGG-19 as our loss network, as also done in [39, 59]. The network is trained on the Microsoft COCO dataset [98] and the WikiArt [122] dataset. Our code is based on Deep Graph Library (DGL) [161]. The training takes roughly two days on an NVIDIA Tesla A100 GPU.

6.3.2 Results

Qualitative comparison. Fig. 6.3 demonstrates the results of the proposed GNN-based semi-parametric method and other arbitrary style transfer methods [96, 59, 1, 19, 147]. The results of [96] are prone to distorted patterns. By contrast, the algorithms of [59, 1] generate sharper details; however, the local style patterns in their results are not well aligned with the

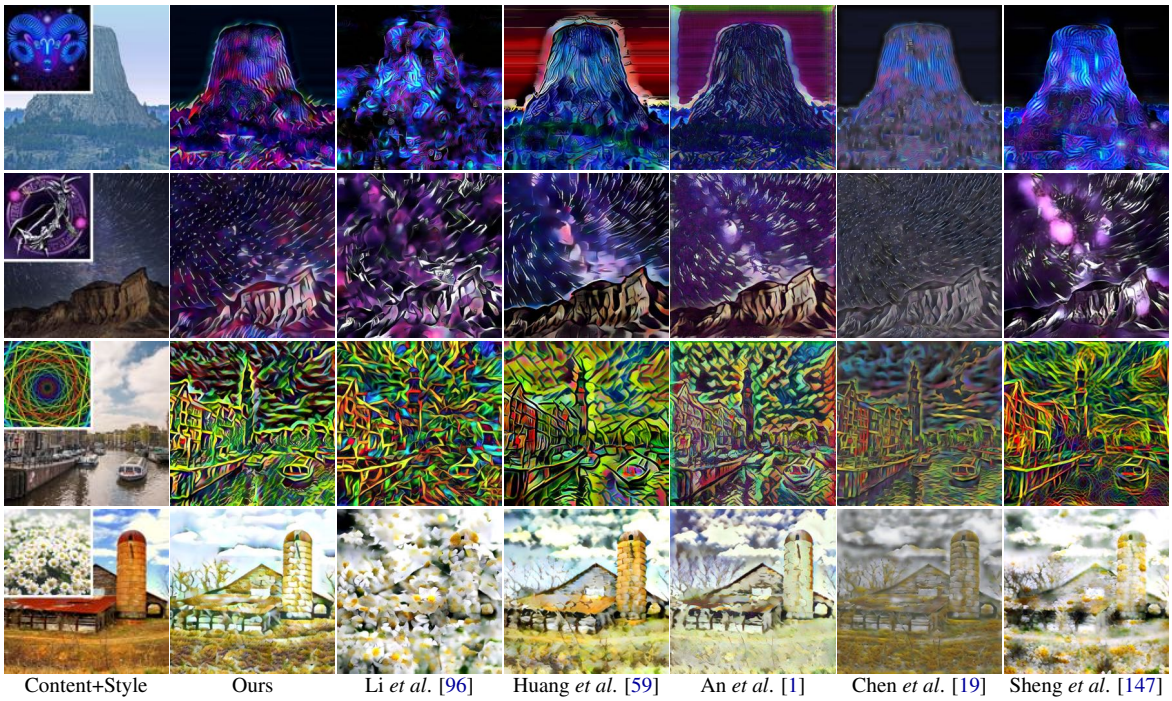


FIGURE 6.3: Qualitative results of our proposed GNN-based semi-parametric stylization algorithm and other parametric [96, 59, 1] and non-parametric [19, 147] methods.

target ones, where very few fine strokes are produced for most styles. For the non-parametric NST approaches of [19, 147], their stylized results either introduce fewer style patterns or suffer from artifacts, due to the potential issue of one-to-one patch mismatching. Compared with other approaches, our semi-parametric framework leads to few artifacts, and meanwhile preserves both the global style appearance and the local fine details, thanks to the local patch-based manipulation module with GNNs.

Efficiency analysis. In Tab. 6.1, we compare the average stylization speed of the proposed approach with other algorithms. For a fair comparison, all the methods are implemented with PyTorch. The experiments are performed over 100 equal-size content and style images of different resolutions using an NVIDIA Tesla A100 GPU. The proposed method demonstrates improved efficiency over the non-parametric method proposed by Sheng *et al.* [147], while maintaining a comparable processing speed to Chen *et al.* [19]. Moreover, it excels in generating fewer artifacts and finer-grained details than the non-parametric and parametric techniques presented by Chen *et al.* [19], Li *et al.* [96], Huang *et al.* [59], and An *et al.*

TABLE 6.1: Average speed comparison in terms of seconds per image.

Methods	Time (s)		
	256×256	384×384	512×512
Li *et al.* [96]	0.707	0.779	0.878
Huang *et al.* [59]	0.007	0.010	0.017
An *et al.* [1]	0.069	0.108	0.169
Chen *et al.* [19]	0.017	0.051	0.218
Sheng *et al.* [147]	0.412	0.536	0.630
Ours	0.094	0.198	0.464

[1], attributed to the more accurate GNN-based patch matching. Also, our speed is, in fact, bottlenecked by the KNN search process, which can be further improved with an optimized KNN implementation.

6.3.3 Ablation Studies

Heterogeneous aggregation schemes. We show in Fig. 6.4 the stylization results by using different neighborhood aggregation strategies in the local patch-based manipulation module. The results of the GAT aggregation scheme, as shown in the 3rd column of Fig. 6.4, outperform those of others in finer structures and global coherence (the areas of the sky and the human face in Fig. 6.4), thereby validating the superiority of the attention scheme in Eq. 6.2.

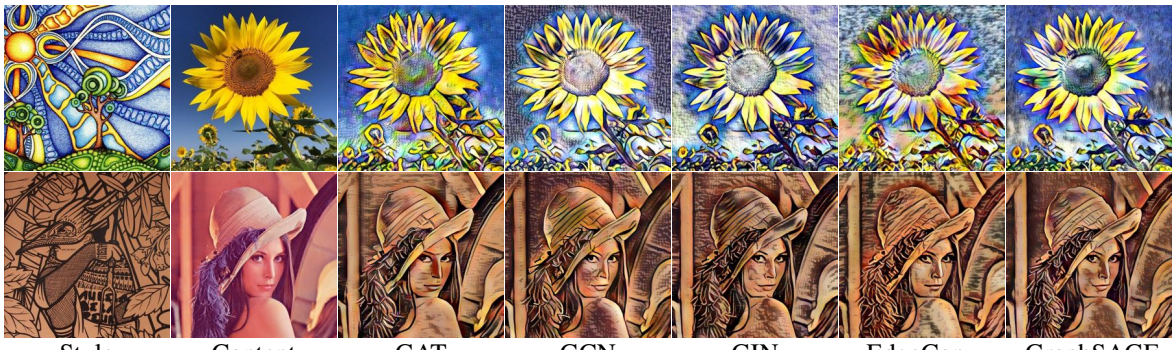


FIGURE 6.4: Comparative results of using various aggregation mechanisms for heterogeneous message passing, including graph attention network (GAT) [153], graph convolutional network (GCN) [82], graph isomorphism network (GIN) [180], dynamic graph convolution (EdgeConv) [167], and GraphSAGE [48]. The GAT mechanism generally yields superior stylization results, thanks to its attention-based aggregation scheme in Eq. 6.2.

FIGURE 6.5: Results of the equal-size patch division method and the proposed deformable one with a learnable scale predictor. Our deformable scheme allows for cross-scale style-content matching, thereby leading to spatially-adaptive multi-stroke stylization with an enhanced semantic saliency (*e.g.*, the foreground regions of the horse and squirrel).

FIGURE 6.6: Results of removing the content-to-content intra-domain edges (w/o Intra) and those with the intra-domain ones (w/ Intra). The devised intra-domain connections incorporate the inter-relationship between the stylized patches at different locations, thereby maintaining the global stylization coherence (*e.g.*, the eye regions in the figure).

Stylization w/ and w/o the deformable scheme. Fig. 6.5 demonstrates the results with the equal-size patch division method, and those with the proposed deformable patch splitting scheme. The devised deformable module makes it possible to adaptively control the strokes in different areas. As a result, the contrast information in the stylized results can be enhanced.

Graph w/ and w/o intra-domain edges. In Fig. 6.6, we validate the effectiveness of the proposed content-to-content message passing scheme, which typically leads to more consistent style patterns in semantically-similar content regions, as can be observed in the foreground human and fox eye areas, as well as the background regions of Fig. 6.6.

Euclidean distance *vs.* normalized cross-correlation. Fig. 6.7 shows the results of using the Euclidean distance and the normalized cross-correlation (NCC) as the distance metric, respectively, in the construction of the stylization graph. The adopted metric of NCC in our framework, as observed from the 4^{th} and 8^{th} columns of Fig. 6.7, leads to superior performance than the Euclidean distance (Fig. 6.7, the 3^{rd} and 7^{th} columns) in terms of both the global stroke arrangements and local details.

FIGURE 6.7: Results obtained using Euclidean distance and normalized cross-correlation (NCC) for similarity measurement during the construction of heterogeneous edges.

FIGURE 6.8: Results obtained using various patch sizes for constructing content and style vertices in the local patch-based manipulation module. By using a larger patch size, the stylized results can maintain an overall larger stroke size.

Various patch sizes. We show in Fig. 6.8 the results of diversified feature patch sizes. Larger patch sizes, as shown from the left to right in the figure, generally lead to larger strokes in the stylized results, which is especially obvious when we observe the regions of the dog and horse in Fig. 6.8.

6.3.4 Diversified Stylization Control

The proposed GNN-based arbitrary style transfer scheme, as shown in Fig. 6.9, can readily support diversified stylization with solely a single model. We also zoom in on the same regions (*i.e.*, the red frames in Fig. 6.9) to observe the details. Such diversities in Fig. 6.9 are specifically achieved by simply changing the numbers of node-specific connections for heterogeneous message passing, which provide users of various tastes with more stylization choices.

6.4 Additional Details and Results

This section presents:

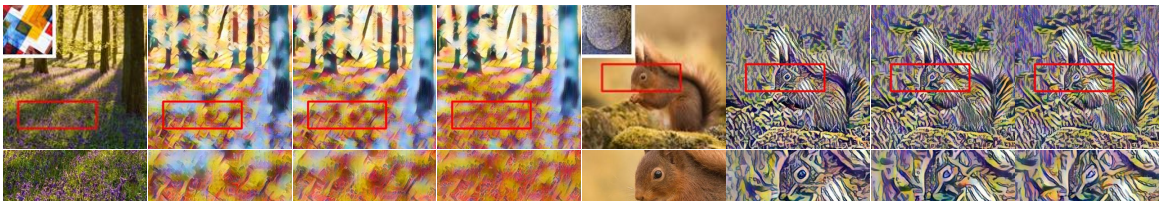


FIGURE 6.9: Flexible control of diversified patch-based arbitrary style transfer during inference. The proposed GNN-based semi-parametric stylization scheme makes it possible to generate heterogeneous style patterns with only a single trained model.

- *Two* newly-added ablation studies, including the results with the proposed local patch-based manipulation (LPM) module and those without LPM. We also validate the effectiveness of the proposed GNN-based approach by developing two possible semi-parametric solutions and demonstrate here the corresponding comparative results;
- Additional results of the *five* ablation studies, including more results of heterogeneous neighborhood aggregation schemes, different distance metrics, various content and style patch sizes, distinct patch division schemes, and the designed intra-domain connections;
- Additional results of the novel functionality, including flexible diversified arbitrary stylization and multi-style amalgamation with a single model.

For the proposed approach, this section provides here more architecture details of each module and more detailed explanations of the proposed heterogeneous content-style and content-content message passing.

6.4.1 Architecture Details

We show in Tab. 6.2 the architecture details of the proposed method. In particular, for the image encoder, we use the first few layers of VGG-19 before `relu3_1`, as also done in [19], to generate more feature patches for matching. We do not include the global feature refinement module in Tab. 6.2, since there are no involved trainable parameters during our process of global feature refinement for the sake of computational efficiency. For the deformable module,

TABLE 6.2: Detailed architectures of the image encoding module, deformable module, GNN-based local patch-based manipulation module, and feature decoding module in the proposed semi-parametric style transfer network, respectively.

	Layer	In_Chans	Out_Chans	Kernel	Stride	Activation
Image Encoding	Conv	3	3	1×1	1	-
	Conv	3	64	3×3	1	ReLU
	Conv	64	64	3×3	1	ReLU
	MaxPool	64	64	-	2	-
	Conv	64	128	3×3	1	ReLU
	Conv	128	128	3×3	1	ReLU
	MaxPool	128	128	-	2	-
	Conv	128	256	3×3	1	ReLU
Deformable Module	Conv	512	256	1×1	1	GELUS
	Conv	256	256	3×3	1	-
	Conv	256	6400	5×5	1	-
	Linear	6400	4	-	-	-
	Linear	2304	6400	-	-	-
Local Patch-based Manipulation	Feat2Patch	256	6400	-	-	-
	KNN	-	-	-	-	-
	GATConv	256	256	-	-	ReLU
	GATConv	256	256	-	-	ReLU
	Patch2Feat	6400	256	-	-	-
Feature Decoding	Conv	256	256	3×3	1	ReLU
	Conv	256	256	3×3	1	ReLU
	Conv	256	256	3×3	1	ReLU
	Conv	256	128	3×3	1	ReLU
	Upsample	128	128	1/2	-	-
	Conv	128	128	3×3	1	ReLU
	Conv	128	64	3×3	1	ReLU
	Upsample	64	64	1/2	-	-
	Conv	64	64	3×3	1	ReLU
	Conv	64	3	3×3	1	-

we would like to clarify that we omit a sampling-interpolation procedure that is hard to depict in Tab. 6.2, which addresses the fractional coordinate issue, with: $2304 = \text{in\_channels} \times \text{sampling\_window\_size}$. Also, the outputs of the scale predictor in Tab. 6.2 are the final rescaled feature patches, with $6400 = \text{in\_chans} \times \text{patch\_size}^2$.

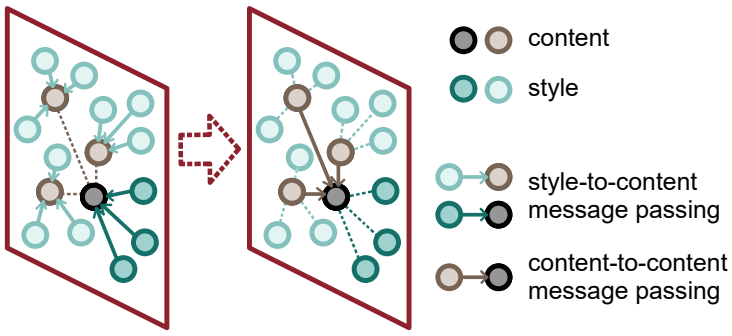


FIGURE 6.10: Illustrations of the dedicated two-stage heterogeneous aggregation process, including style-to-content message passing stage (*i.e.*, the left red block in the figure) and content-to-content messing passing stage (*i.e.*, the right red block in the figure).

6.4.2 More Illustrations of Heterogeneous Style-Content and Content-Content Message Passing

In this section, we give more explanations of the proposed path-based message passing scheme. We demonstrate the detailed style-to-content message passing and content-to-content message passing in Fig. 6.10. Specifically, the style-to-content message passing, as shown in Fig. 6.10, aims to aggregate style information from the k most similar style patches along the inter-domain edges (green arrows in Fig. 6.10). Subsequent to the style-to-content message passing, the proposed content-to-content aggregation further gathers the features from neighboring content nodes, such that the semantically-similar content regions will also be rendered with homogeneous style patterns. The effectiveness of such content-to-content message passing will be further validated in Fig. 6.15 of Sect. 6.4.4.3.

6.4.3 Newly-Added Ablation Studies

In this section, we perform extensive ablation studies to further validate the effectiveness of the proposed semi-parametric style transfer framework. In particular, we add *two new ablation studies*, including the stylization results with the local patch-based manipulation (LPM) module and those without the LPM module, and also the results of the two possible

solutions of semi-parametric stylization, including the combination of AdaIN and style swap, and also the combination of AdaIN and style decorator.

6.4.3.1 Stylization w/ and w/o Local Patch-based Manipulation Module

Fig. 6.11 shows the stylization results with the proposed local patch-based manipulation (LPM) module, and those without the LPM module. The stylized results without the proposed LPM module, as shown in the 2nd and the 5th columns of Fig. 6.11, retain the global appearance of the style images, but are prone to undesired local artifacts. In contrast, the results with the dedicated LPM are effective in producing fine-grained patterns and sharper details, as can be observed in the 3rd and the 6th columns Fig. 6.11. For example, the 3rd row, 6th column of Fig. 6.11 successfully transfers the corresponding style strokes to the petals, whereas the 5th column in Fig. 6.11 only keeps the original petal colors. Similar observations can also be obtained from the 1st row of Fig. 6.11, where the bird feathers are rendered with the corresponding best-matched style patterns.

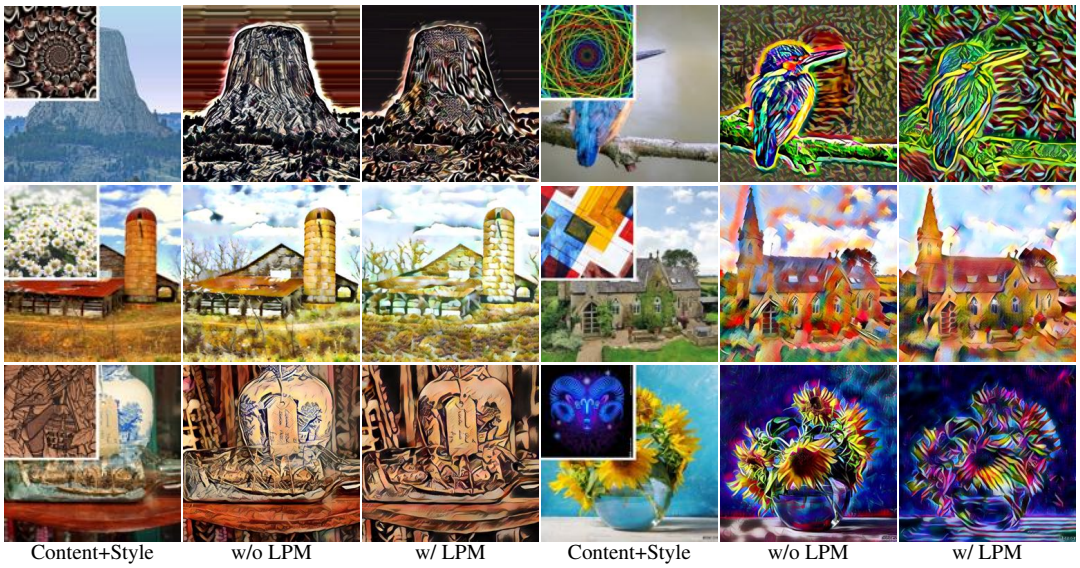


FIGURE 6.11: Comparative results without the local patch-based manipulation (LPM) module and those with the LPM module.

6.4.3.2 Ours *vs* AdaIN+Style-Swap *vs* AdaIN+Style-Decorator

To further demonstrate the superiority of the proposed local patch-based manipulation module, we develop two possible solutions for semi-parametric neural style transfer. Specifically, we combine the style swap module in [19] with the global feature refinement module (*i.e.*, AdaIN), and also combine the style decorator module in [147] with AdaIN. As such, both local manipulation and global refinement are performed, leading to the two possible semi-parametric stylization methods. The results of the developed two possible solutions and our method (*i.e.*, AdaIN+GNN) are provided in Fig. 6.12, indicating that the proposed GNN-based approach is indeed superior than others.

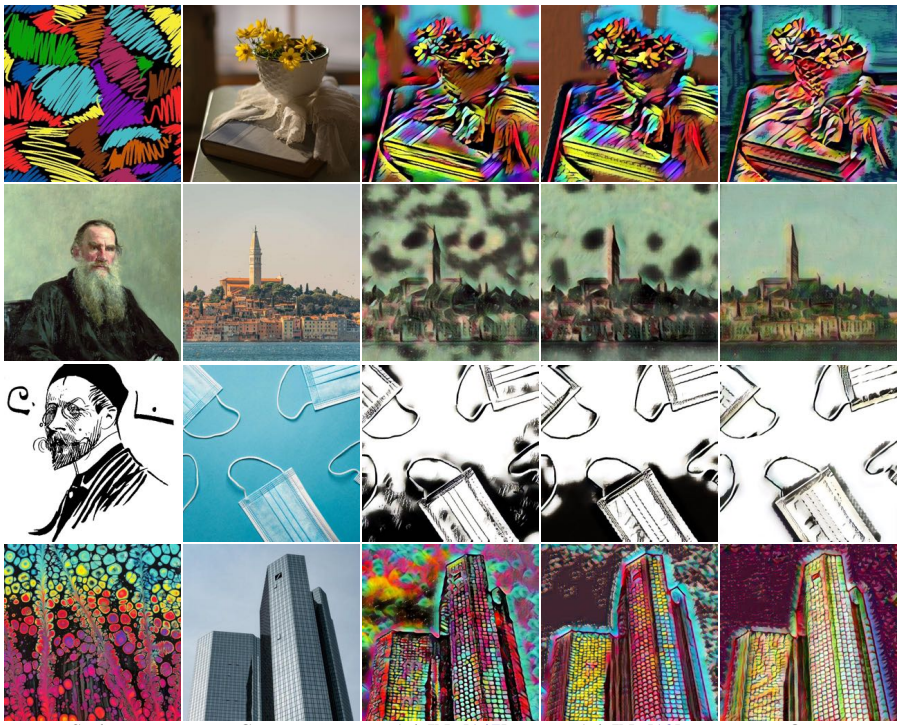


FIGURE 6.12: Comparative results of the proposed GNN-based method with two possible semi-parametric solutions of AdaIN+Style-Swap and AdaIN+Style-Decorator.

6.4.4 Additional Results of Ablation Studies

In particular, we provide additional results of the ***five ablation studies***, including the stylization results of various content/style patch sizes and heterogeneous aggregation mechanisms. We also give more results to validate the effectiveness of the proposed content-to-content message passing, the proposed deformable scheme, and the adopted similarity measurement metric of normalized cross-correlation.

6.4.4.1 Heterogeneous Aggregation Schemes

We provide in Fig. 6.13 the results of using various GNN mechanisms in the proposed local patch-based manipulation module, including graph attention network (GAT) [153], graph convolutional network (GCN) [82], dynamic graph convolution (EdgeConv) [167], GraphSAGE [48], and graph isomorphism network (GIN) [180]. In what follows, we start

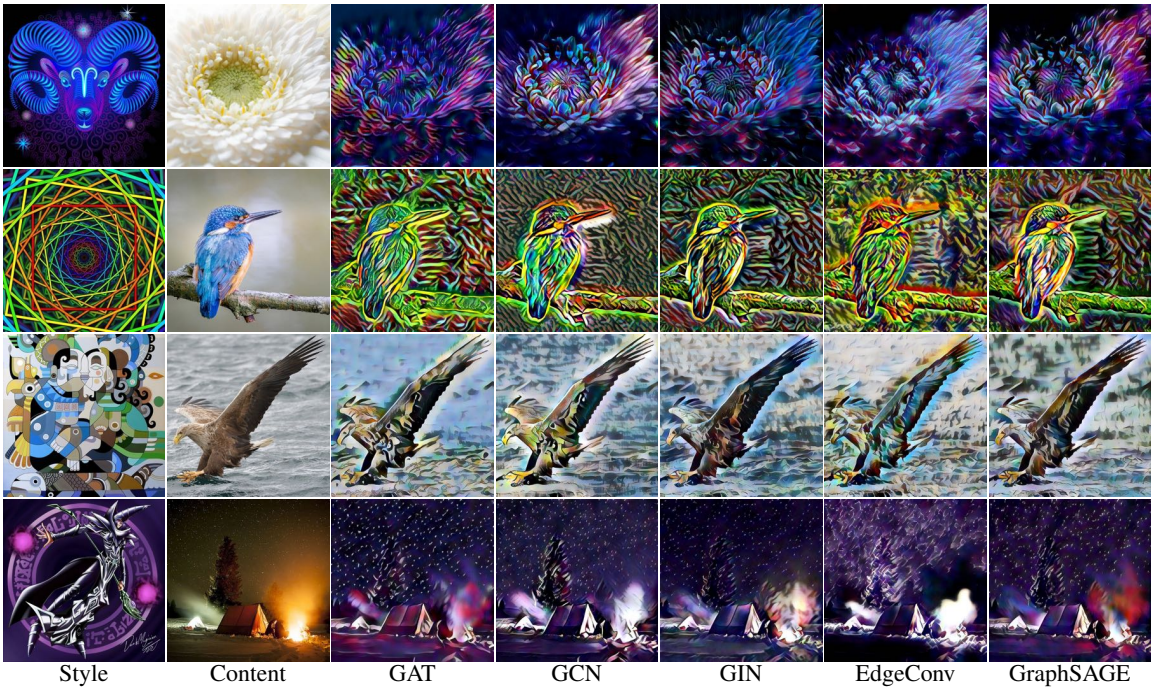


FIGURE 6.13: Comparative results of using various aggregation mechanisms for heterogeneous message passing, including graph attention network (GAT) [153], graph convolutional network (GCN) [82], graph isomorphism network (GIN) [180], dynamic graph convolution (EdgeConv) [167], and GraphSAGE [48].

by briefly introducing these different GNN schemes and then explain the corresponding comparison results.

The most straightforward GNN mechanism is GCN. The idea of GCN is to optimize the node features in an iterative manner, specifically through an isotropic averaging process over the features of the neighbor nodes: $h_i^{\ell+1} = \text{ReLU}\Big(U^\ell \ \text{Mean}_{j\in\mathcal{N}_i} \ h_j^\ell\Big)$, where $h_i^{\ell+1}$ represents the updated node features at layer $\ell+1$. U denotes the learnable transformation matrix. j represents the neighbors $\mathcal{N}_i$ of the node i. For GAT, the rationale is to improve the simplest GCN by incorporating the use of the self-attention scheme. Also, GraphSAGE improves GCN in a different way, by explicitly considering the node's own representations from the previous layer, which can be formulated as: $\hat{h}_i^{\ell+1} = \text{ReLU}\Big(U^\ell \ \text{Concat}\left(h_i^\ell, \ \text{Mean}_{j\in\mathcal{N}_i} \ h_j^\ell\right)\Big)$, where Concat represents the concatenation operation. Moreover, GIN is developed upon *Weisfeiler-Lehman Isomorphism Test* [170], whereas EdgeConv yields the edge features describing the relationships between the points and their neighbors for information propagation. More details can be found in [82, 153, 167, 48, 180, 32].

As can be observed in Fig. 6.13, the GAT mechanism generally yield superior locally-style-aligned stylized results, thanks to its attention-based scheme. For example, in the 1$^{\text{st}}$ row of Fig. 6.13, GAT yields fine-grained style elements for the petals, in contrast to other GNNs that merely transfer the global style appearance from the target style. Another observation from Fig. 6.13 is that the GraphSAGE architecture is more effective at preserving the semantics of the content images, possibly due to its property of combining the node's own features from the previous layer. Also, the results of EdgeConv are less appealing, demonstrating that the edge features are inferior to the node features for the specific task of style transfer. Similar observations can be obtained from the results of GIN, where the style patterns are sometimes not sufficiently transferred to the stylized image, as shown in the 6$^{\text{th}}$ row of Fig. 6.13.

6.4.4.2 Distinct Patch Division Schemes

We show in Fig. 6.14 additional comparative results of equal-size patch division method, and those with the proposed deformable patch splitting scheme. Our deformable scheme allows for cross-scale style-content matching, thereby leading to spatially-adaptive multi-stroke

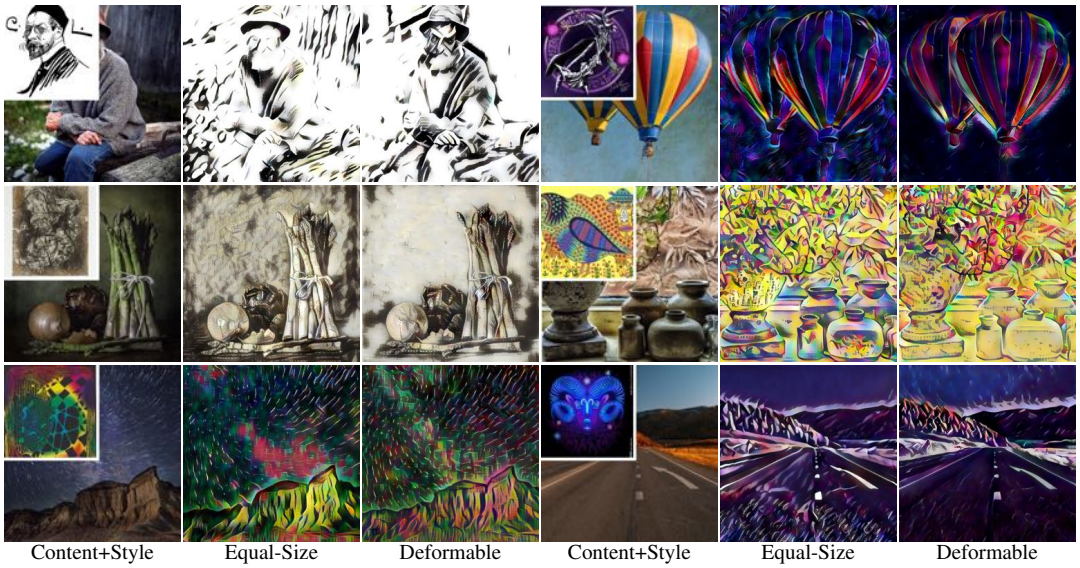


FIGURE 6.14: Additional results of the equal-size patch division method and the proposed deformable module with a learnable scale predictor.

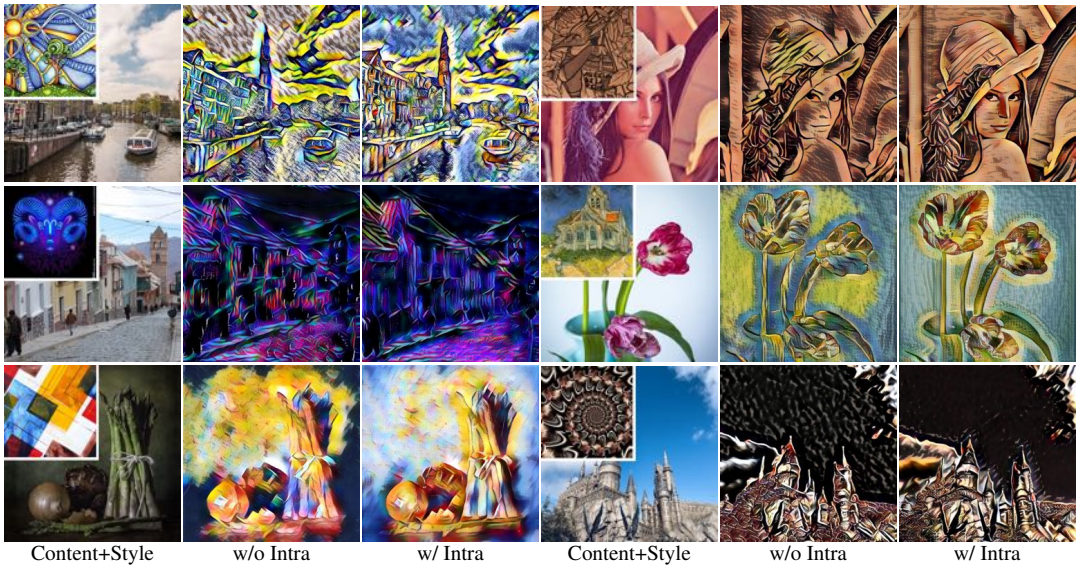


FIGURE 6.15: Stylization results of removing the content-to-content intra-domain edges and those with the intra-domain edges.

stylization with the enhanced semantic saliency. Also, the proposed deformable module reduces the undesired artifacts in the stylization results, as shown in Fig. 6.14.

FIGURE 6.16: Results obtained using Euclidean distance and normalized cross-correlation (NCC) for similarity measurement during the construction of heterogeneous content-to-style and content-to-content edges.

6.4.4.3 NST Graph w/ and w/o Intra-domain Edges

To validate the effectiveness of the proposed content-to-content message passing scheme, we perform extensive ablation studies in Fig. 6.15 by using or removing the intra-domain edges in the constructed stylization graph. Our design of intra-domain edges, as shown in the 3rd and the 6th columns of Fig. 6.15, leads to more consistent style patterns in semantically-similar content regions, which is especially obvious when we observe the human faces in the 1st row of Fig. 6.15.

6.4.4.4 Euclidean Distance *vs.* Normalized Cross-correlation

In Fig. 6.16, we compare the results of using the Euclidean distance and the normalized cross-correlation (NCC) as the similarity measurement, respectively, in the construction of the stylization graph. The adopted metric of NCC in our framework, as observed from the 3rd and the 6th columns of Fig. 6.16, leads to superior performance than the Euclidean distance (Fig. 6.16, the 2nd and 5th columns) in terms of both the global stroke arrangements and local details. We take the 3rd row of Fig. 6.16 as an example. It is evident that the stylization results with the Euclidean distance have more artifacts than those with NCC in the background of the

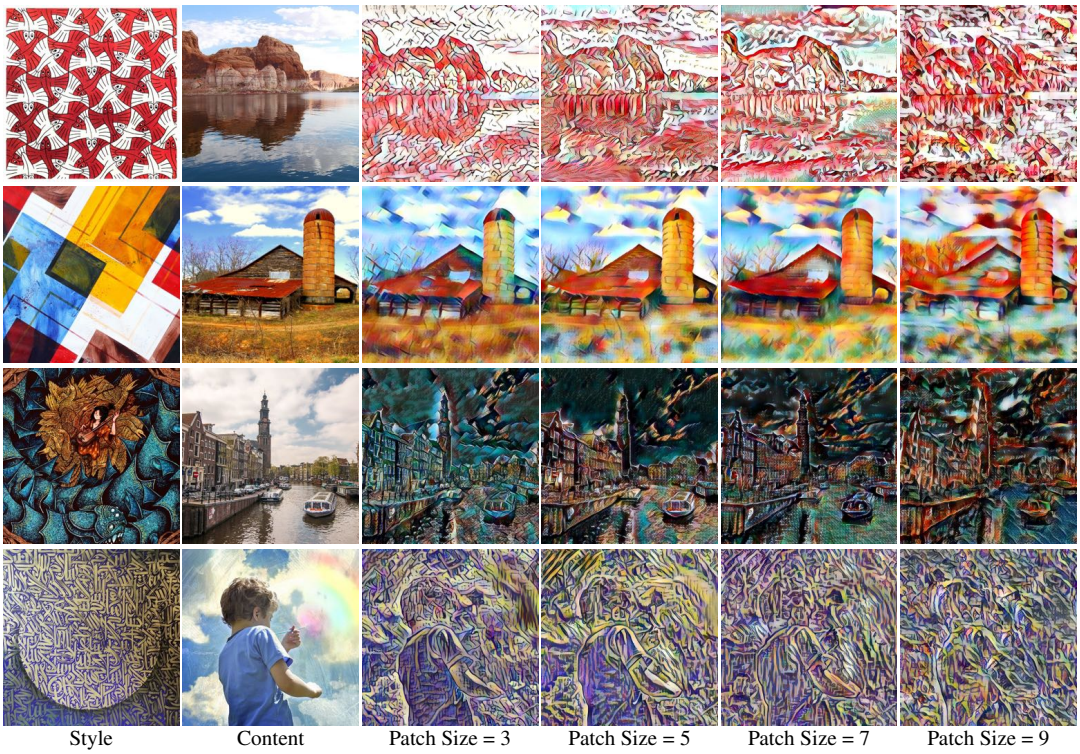


FIGURE 6.17: Results obtained using various patch sizes for constructing content and style vertices in local patch-based manipulation module.

sewing machine, demonstrating that NCC is better-suited for patch-based matching in our GNN-based framework.

6.4.4.5 Various Patch Sizes

We demonstrate in Fig. 6.17 the results of diversified feature patch sizes. Larger patch sizes, as shown in Fig. 6.17, generally lead to larger strokes in the stylized results. For example, the stylized images in the 3rd row, the 6th column of Fig. 6.17 has much larger strokes than those in the 3rd row, the 3rd column, which is especially obvious from the regions of the sky.

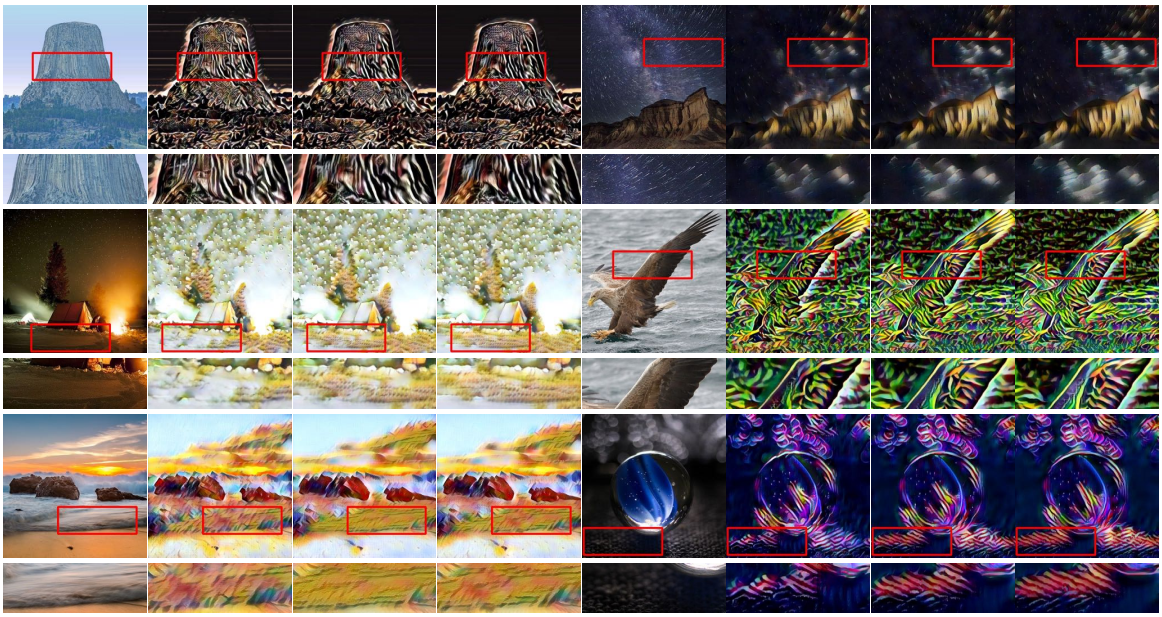


FIGURE 6.18: Additional results of diversified patch-based arbitrary style transfer with solely a single model, corresponding to Fig. 6.6. We zoom in on the same regions (*i.e.*, the red frames) to observe the details.

6.4.5 Additional Results of User Controls

6.4.5.1 Diversified Stylization Control

We provide in this section more results of the proposed diversified stylization control, corresponding to Fig. 6.9. Additional diversified results are given in Fig. 6.18, where we zoom in on the same regions in the red frames for the illustrations of local details. For example, in the last row of Fig. 6.18, it is noticeable that our diversified stylization control can yield various style patterns with different colors and strokes with only a single trained model. Such diversified user control is simply achieved by using different numbers of style-to-content connections during style-to-content message passing, leading to a limited auxiliary computational burden.

6.4.5.2 Multi-style Amalgamation

The proposed GNN-based style transfer approach also triggers the functionality of flexible multi-style transfer that combines the style patterns in multiple distinct artistic styles. We show in Fig. 6.19 the results that amalgamate four different style images as an example,

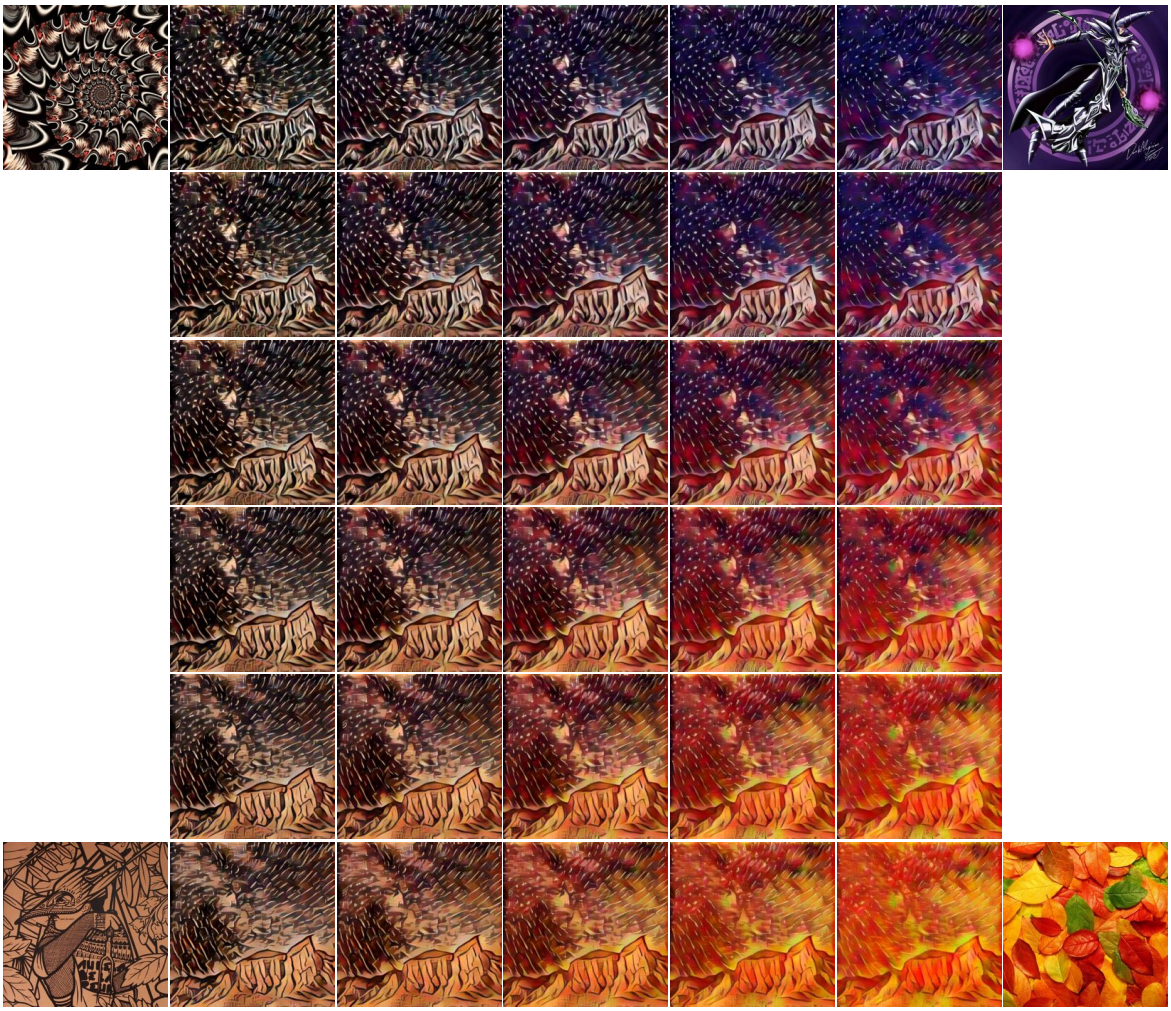


FIGURE 6.19: Multi-style transfer within a single image, by performing style interpolation among various artistic styles.

but we would like to clarify that our method readily supports arbitrary style numbers for compositions. From the algorithm level, this multi-style image stylization is specifically realized by exploiting the style feature patches from multiple style images to construct the style vertices, which are then used to establish the *multistyle*-to-content heterogeneous connections for the subsequent *multistyle* message passing.

6.5 Summary

This chapter investigates a semi-parametric arbitrary style transfer scheme for the effective transfers of challenging style patterns at the both local and global levels. Towards this goal, we identify two key challenges in existing parametric and non-parametric stylization approaches, and propose a dedicated GNN-based style transfer scheme to solve the dilemma. This is specifically accomplished by modeling the style transfers as the heterogeneous information propagation process among the constructed content and style vertices for accurate patch-based style-content correspondences. Moreover, we develop a deformable graph convolutional network for various-scale stroke generations. Experiments demonstrate that the proposed approach achieves favorable performance in both global stroke arrangement and local details. In our future work, we will investigate employing arbitrary patches based on superpixels, rather than rectangular patches, for GNN-based style transfer. We will also strive to generalize the proposed GNN-based scheme to other vision tasks.

CHAPTER 7

Conclusions

This chapter provides a comprehensive overview of the contributions presented in the previous chapters, while also offering perspectives into potential avenues for future research.

7.1 Summary of Contributions

This thesis focuses on the research of learning efficient representations with GNN architectures for various domains, such as visual SLAM, visual perception utilizing event cameras, as well as point cloud classification and segmentation. While GNNs have shown promising results in these tasks, their increasing complexity and resource requirements pose challenges for practical deployment in real-world applications such as autonomous driving. This specifically includes the limitations in resource-constrained environments, time-sensitive applications, and processing large-scale graphs.

To address these challenges, this thesis identifies four key aspects of efficient graph representation learning: data-driven efficiency, model-driven efficiency, data-model-driven efficiency, and application-driven efficiency. To tackle these aspects, four complementary research works are introduced.

To address data-driven efficiency, a deep graph reprogramming method is proposed. It allows for the adaptation of a pre-trained GNN to multiple cross-level downstream tasks by modifying input data, without altering the model parameters. For model-driven efficiency, a novel knowledge amalgamation approach is explored. It enables the training of a compact

and versatile student model by amalgamating knowledge from heterogeneous teacher GNNs, handling different tasks without relying on human-annotated labels.

To achieve joint data-model-driven efficiency, a customized 1-bit learning framework is developed. It simultaneously binarizes input graph data and model parameters, replacing resource-intensive operations with more efficient 1-bit operations for lightweight graph inference. For application-driven efficiency, an application-specific efficient learning method is proposed for image style transfer. It employs efficient heterogeneous message passing in GNNs to model the content-style matching process, improving efficiency compared to traditional methods.

In summary, these research works contribute to advancing the field of efficient representation learning with GNNs, addressing various efficiency challenges and offering practical solutions applicable to a wide range of applications, including the field of visual SLAM.

7.2 Future Research

Efficient representation learning with GNNs is in its early stages and has yet to reach saturation. In this section, this thesis delves into several potential avenues for future research in this domain.

For data-driven efficiency, the presented *Meta-FeatPadding* technique operates under the assumption that the dimension of the pre-trained data is larger than that of the downstream data. This allows for the introduction of padding to effectively accommodate the varying downstream feature dimensions. However, this particular prerequisite imposes a constraint on the range of applications suitable for this data-driven learning approach. To overcome this limitation, one potential avenue is the integration of prototype learning, which holds promise in mitigating the aforementioned challenge. Beyond the scope of the introduced deep graph reprogramming detailed in this thesis, an alternative strategy for enhancing data-centric efficiency is graph condensation with GNNs, which involves the compression of a large graph into a more compact yet remarkably informative representation.

In the pursuit of model-driven efficient learning, the computation of the suggested topological attribution map necessitates resource-intensive gradient computations, thereby introducing some complexity into the training process. A potential avenue for future research involves the enhancement of computational efficiency in generating the proposed topological attribution map, which is a crucial component for achieving structural semantics alignment in knowledge amalgamation. Despite the encouraging results of the model-driven efficient learning approach, it exhibits suboptimal performance empirically when faced with heterogeneous teachers characterized by significant domain disparities, as seen in scenarios like segmentation and detection tasks. A prospective direction for further investigation lies in the development of more broadly applicable model-driven efficient representation learning strategies with GNNs to bridge substantial domain gaps present in diverse teacher models.

Towards joint data-model-driven efficient representation learning, the utilization of 1-bit GNNs with the proposed meta aggregators still results in a notable decline in performance when compared to the full-precision models, particularly evident in intricate tasks such as point cloud analysis. A prospective avenue for future exploration involves the development of task-specific and tailored binarization techniques to improve post-binarization performance across diverse and complex tasks. Moreover, shifting from the current 1-bit precision to half precision for both graph data and model parameters holds promise as a direction to achieve a favorable balance between performance and speed. Furthermore, the concept of meta aggregators offers potential beyond the scope of 1-bit GNNs. Exploring their application to enhance the performance of full-precision models presents a promising opportunity, especially in the context of more generalized GNN architectures.

In the context of application-driven efficiency, the proposed semi-parametric style transfer framework, built upon GNNs, adopts the strategy of segmenting images into regular rectangular patches. A potential avenue for future exploration involves the adoption of arbitrary patches based on superpixels, as opposed to confining to rectangular patches. This adaptation would offer increased flexibility and enhanced semantic precision in content-style alignment. Furthermore, this thesis stands as an innovative endeavor to leverage GNNs to enhance the manipulation of pixel-based images. A promising path for further investigation lies in the

exploration of GNN integration across a wider range of pixel-based image tasks. This encompasses domains like image deblurring, object detection and tracking, and image segmentation, each of which holds potential for reaping the benefits of GNN-powered methodologies. Additionally, the application of GNNs in more intricate video-based scenarios presents another promising trajectory for advancement, extending the potential of GNNs into more complex and dynamic visual contexts.

Bibliography

[1] Jie An, Siyu Huang, Yibing Song, Dejing Dou, Wei Liu and Jiebo Luo. 'ArtFlow: Unbiased Image Style Transfer via Reversible Neural Flows'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 862–871.

[2] Andreas Argyriou, Theodoros Evgeniou and Massimiliano Pontil. 'Convex multi-task feature learning'. In: *Machine Learning* 73.3 (2008), pp. 243–272.

[3] Mehdi Bahri, Gaétan Bahl and Stefanos Zafeiriou. 'Binary graph neural networks'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 9492–9501.

[4] Yoshua Bengio, Nicholas Léonard and Aaron Courville. 'Estimating or propagating gradients through stochastic neurons for conditional computation'. In: *arXiv preprint arXiv:1308.3432* (2013).

[5] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill et al. 'On the opportunities and risks of foundation models'. In: *arXiv preprint arXiv:2108.07258* (2021).

[6] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola and Hans-Peter Kriegel. 'Protein function prediction via graph kernels'. In: *Bioinformatics* 21.suppl_1 (2005), pp. i47–i56.

[7] Xavier Bresson and Thomas Laurent. 'Residual gated graph convnets'. In: *arXiv preprint arXiv:1711.07553* (2017).

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al. 'Language models are few-shot learners'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 1877–1901.

[9] Adrian Bulat and Georgios Tzimiropoulos. 'Xnor-net++: Improved binary neural networks'. In: *The British Machine Vision Conference (BMVC)*. 2019, p. 62.

[10] Fabio Capela, Vincent Nouchi, Ruud Van Deursen, Igor V Tetko and Guillaume Godin. 'Multitask learning on graph neural networks applied to molecular property predictions'. In: *arXiv preprint arXiv:1910.13124* (2019).

[11] Alex J Champandard. 'Semantic style transfer and turning two-bit doodles into fine artworks'. In: *arXiv preprint arXiv:1603.01768* (2016).

[12] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu and Gang Hua. 'Explicit filterbank learning for neural image style transfer and image processing'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43.7 (2020), pp. 2373–2387.

[13] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu and Gang Hua. 'StyleBank: An explicit representation for neural image style transfer'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2770–2779.

[14] Haibo Chen, Lei Zhao, Huiming Zhang, Zhizhong Wang, Zhiwen Zuo, Ailin Li, Wei Xing and Dongming Lu. 'Diverse Image Style Transfer via Invertible Cross-Space Mapping'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 14860–14869.

[15] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian and Chang Xu. 'AdderNet: Do we really need multiplications in deep learning?' In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1468–1477.

[16] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu and Wen Gao. 'Pre-trained image processing transformer'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 12299–12310.

[17] Jie Chen, Tengfei Ma and Cao Xiao. 'Fastgcn: fast learning with graph convolutional networks via importance sampling'. In: *International Conference on Learning Representations (ICLR)*. 2018.

[18] Pin-Yu Chen. 'Model reprogramming: Resource-efficient cross-domain machine learning'. In: *arXiv preprint arXiv:2202.10629* (2022).

[19] Tian Qi Chen and Mark Schmidt. 'Fast patch-based style transfer of arbitrary style'. In: *NeurIPS Workshop on Constructive Machine Learning*. 2016.

[20] Zhengdao Chen, Lei Chen, Soledad Villar and Joan Bruna. 'On the equivalence between graph isomorphism testing and function approximation with GNNs'. In: 32 (2019), pp. 15868–15876.

[21] Zhiyang Chen, Yousong Zhu, Chaoyang Zhao, Guosheng Hu, Wei Zeng, Jinqiao Wang and Ming Tang. 'Dpt: Deformable patch-based transformer for visual recognition'. In: *Proceedings of the ACM International Conference on Multimedia*. 2021, pp. 2899–2907.

[22] Xiao Chu, Wanli Ouyang, Wei Yang and Xiaogang Wang. 'Multi-task recurrent neural network for immediacy prediction'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2015, pp. 3352–3360.

[23] Ronan Collobert and Jason Weston. 'A unified architecture for natural language processing: Deep neural networks with multitask learning'. In: *International Conference on Machine Learning (ICML)*. 2008, pp. 160–167.

[24] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò and Petar Veličković. 'Principal Neighbourhood Aggregation for Graph Nets'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 13260–13271.

[25] Matthieu Courbariaux, Yoshua Bengio and Jean-Pierre David. 'BinaryConnect: training deep neural networks with binary weights during propagations'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 28. 2015, pp. 3123–3131.

[26] Xiang Deng and Zhongfei Zhang. 'Graph-free knowledge distillation for graph neural networks'. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2021, pp. 2441–2447.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *arXiv preprint arXiv:1810.04805* (2018).

[28] Liang Ding, Longyue Wang, Di Wu, Dacheng Tao and Zhaopeng Tu. 'Context-Aware Cross-Attention for Non-Autoregressive Translation'. In: *Proceedings of the International Conference on Computational Linguistics (COLING)*. 2020, pp. 4396–4402.

[29] Liang Ding, Longyue Wang and Dacheng Tao. 'Self-Attention with Cross-Lingual Position Representation'. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020, pp. 1679–1685.

[30] Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao and Zhaopeng Tu. 'Understanding and Improving Lexical Choice in Non-Autoregressive Translation'. In: *International Conference on Learning Representations (ICLR)*. 2021.

[31] Carl Doersch and Andrew Zisserman. ‘Multi-task self-supervised visual learning’. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, pp. 2051–2060.

[32] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio and Xavier Bresson. ‘Benchmarking graph neural networks’. In: *arXiv preprint arXiv:2003.00982* (2020).

[33] Gamaleldin F Elsayed, Ian Goodfellow and Jascha Sohl-Dickstein. ‘Adversarial reprogramming of neural networks’. In: *International Conference on Learning Representations (ICLR)*. 2019.

[34] Matthias Englert and Ranko Lazic. ‘Adversarial Reprogramming Revisited’. In: *arXiv preprint arXiv:2206.03466* (2022).

[35] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi and Xinchao Wang. ‘DepGraph: Towards Any Structural Pruning’. In: *arXiv preprint arXiv:2301.12900* (2023).

[36] Kaituo Feng, Changsheng Li, Ye Yuan and Guoren Wang. ‘FreeKD: Free-direction Knowledge Distillation for Graph Neural Networks’. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2022, pp. 357–366.

[37] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti and Anima Anandkumar. ‘Born Again Neural Networks’. In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 1607–1616.

[38] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis et al. ‘Event-based vision: A survey’. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 44.1 (2020), pp. 154–180.

[39] Leon A. Gatys, Alexander S. Ecker and Matthias Bethge. ‘Image style transfer using convolutional neural networks’. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423.

[40] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals and George E Dahl. ‘Neural message passing for quantum chemistry’. In: *International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 1263–1272.

[41] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals and George E Dahl. ‘Neural message passing for quantum chemistry’. In: *International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 1263–1272.

[42] Pinghua Gong, Jiayu Zhou, Wei Fan and Jieping Ye. 'Efficient multi-task feature learning with calibration'. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2014, pp. 761–770.

[43] Jiayuan Gu, Han Hu, Liwei Wang, Yichen Wei and Jifeng Dai. 'Learning region features for object detection'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 381–395.

[44] Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. Vol. 33. US Government Printing Office, 1954.

[45] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu and Mohammed Bennamoun. 'Deep learning for 3d point clouds: A survey'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43.12 (2020), pp. 4338–4364.

[46] Karen Hambardzumyan, Hrant Khachatrian and Jonathan May. 'Warp: Word-level adversarial reprogramming'. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. 2021, pp. 4921–4933.

[47] Will Hamilton, Zhitao Ying and Jure Leskovec. 'Inductive representation learning on large graphs'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, 1025—1035.

[48] William L Hamilton, Rex Ying and Jure Leskovec. 'Inductive representation learning on large graphs'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 1024–1034.

[49] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang and Enhua Wu. 'Vision gnn: An image is worth graph of nodes'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 35. 2022, pp. 8291–8303.

[50] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang and Yulin Wang. 'Dynamic neural networks: A survey'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 44.11 (2021), pp. 7436–7456.

[51] Dan He, David Kuhn and Laxmi Parida. 'Novel applications of multitask learning and multiple output regression to multiple genetic trait prediction'. In: *Bioinformatics* 32.12 (2016), pp. i37–i43.

[52] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. 'Mask r-cnn'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, pp. 2961–2969.

[53] Geoffrey Hinton, Oriol Vinyals and Jeff Dean. 'Distilling the knowledge in a neural network'. In: *NIPS Deep Learning and Representation Learning Workshop*. 2015.

[54] Kibeom Hong, Seogkyu Jeon, Huan Yang, Jianlong Fu and Hyeran Byun. 'Domain-Aware Universal Style Transfer'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 14609–14617.

[55] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai and Yichen Wei. 'Relation networks for object detection'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3588–3597.

[56] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta and Jure Leskovec. 'Open graph benchmark: Datasets for machine learning on graphs'. In: *arXiv preprint arXiv:2005.00687* (2020).

[57] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande and Jure Leskovec. 'Strategies for pre-training graph neural networks'. In: *International Conference on Learning Representations (ICLR)*. 2020.

[58] Wenbing Huang, Tong Zhang, Yu Rong and Junzhou Huang. 'Adaptive sampling towards fast graph representation learning'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31. 2018, 4563—4572.

[59] Xun Huang and Serge Belongie. 'Arbitrary style transfer in real-time with adaptive instance normalization'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, pp. 1510–1519.

[60] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv and Yoshua Bengio. 'Binarized neural networks: Training neural networks with weights and activations constrained to+ 1 or-1'. In: *arXiv preprint arXiv:1602.02830* (2016).

[61] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv and Yoshua Bengio. 'Binarized neural networks'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 29. 2016, pp. 4107–4115.

[62] Jing Huo, Shiyin Jin, Wenbin Li, Jing Wu, Yu-Kun Lai, Yinghuan Shi and Yang Gao. 'Manifold Alignment for Semantically Aligned Style Transfer'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 14861–14869.

[63] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad and Ryan G Coleman. 'ZINC: a free tool to discover chemistry for biology'. In: *Journal of Chemical Information and Modeling* 52.7 (2012), pp. 1757–1768.

[64] Ashesh Jain, Amir R Zamir, Silvio Savarese and Ashutosh Saxena. 'Structural-rnn: Deep learning on spatio-temporal graphs'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5308–5317.

[65] Eric Jang, Shixiang Gu and Ben Poole. 'Categorical reparameterization with gumbel-softmax'. In: *arXiv preprint arXiv:1611.01144* (2016).

[66] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal and Jiliang Tang. 'Adversarial attacks and defenses on graphs'. In: *ACM SIGKDD Explorations Newsletter* 22.2 (2021), pp. 19–34.

[67] Wengong Jin, Regina Barzilay and Tommi Jaakkola. 'Junction tree variational autoencoder for molecular graph generation'. In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 2323–2332.

[68] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song and Dacheng Tao. 'Amalgamating Knowledge From Heterogeneous Graph Neural Networks'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15709–15718.

[69] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song and Shilei Wen. 'Dynamic Instance Normalization for Arbitrary Style Transfer'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 34. 04. 2020, pp. 4369–4376.

[70] Yongcheng Jing, Yining Mao, Yiding Yang, Yibing Zhan, Mingli Song, Xinchao Wang and Dacheng Tao. 'Learning graph neural networks for image style transfer'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022, pp. 111–128.

[71] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song and Dacheng Tao. 'Meta-Aggregator: Learning to Aggregate for 1-bit Graph Neural Networks'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 5301–5310.

[72] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu and Mingli Song. 'Neural Style Transfer: A Review'. In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 26.11 (2020), pp. 3365–3385.

[73] Yongcheng Jing, Xinchao Wang and Dacheng Tao. 'Segment anything in non-euclidean domains: Challenges and opportunities'. In: *arXiv preprint arXiv:2304.11595* (2023).

[74] Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao and Mingli Song. 'Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 238–254.

[75] Justin Johnson, Alexandre Alahi and Li Fei-Fei. 'Perceptual losses for real-time style transfer and super-resolution'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016, pp. 694–711.

[76] Chaitanya K Joshi, Fayao Liu, Xu Xun, Jie Lin and Chuan-Sheng Foo. 'On Representation Knowledge Distillation for Graph Neural Networks'. In: *arXiv preprint arXiv:2111.04964* (2021).

[77] Łukasz Kaiser and Samy Bengio. 'Discrete autoencoders for sequence models'. In: *arXiv preprint arXiv:1801.09797* (2018).

[78] Nikolai Kalischek, Jan D Wegner and Konrad Schindler. 'In the light of feature distributions: moment matching for Neural Style Transfer'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 9382–9391.

[79] Minje Kim and Paris Smaragdis. 'Bitwise neural networks'. In: *arXiv preprint arXiv:1601.06071* (2016).

[80] Diederik P Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *International Conference on Learning Representations (ICLR)*. 2015.

[81] Diederik Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *International Conference on Learning Representations (ICLR)*. 2015.

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

[83] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo et al. 'Segment anything'. In: *arXiv preprint arXiv:2304.02643* (2023).

[84] Eliska Kloberdanz, Jin Tian and Wei Le. 'An improved (adversarial) reprogramming technique for neural networks'. In: *International Conference on Artificial Neural Networks (ICANN)*. 2021, pp. 3–15.

[85] Iasonas Kokkinos. 'Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6129–6138.

[86] Nicholas Kolkin, Jason Salavon and Gregory Shakhnarovich. 'Style transfer by relaxed optimal transport and self-similarity'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10051–10060.

[87] Yajing Kong, Liu Liu, Jun Wang and Dacheng Tao. 'Adaptive curriculum learning'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 5047–5056.

[88] Loic Landrieu and Martin Simonovsky. 'Large-scale point cloud semantic segmentation with superpoint graphs'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4558–4567.

[89] Chuan Li and Michael Wand. 'Combining markov random fields and convolutional neural networks for image synthesis'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2479–2486.

[90] 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 (ICCV)*. 2019, pp. 9267–9276.

[91] 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 (AAAI)*. Vol. 32. 1. 2018.

[92] Ruoyu Li, Sheng Wang, Feiyun Zhu and Junzhou Huang. 'Adaptive graph convolutional neural networks'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 32. 1. 2018.

[93] Yan Li, Jie Wang, Jieping Ye and Chandan K Reddy. 'A multi-task learning formulation for survival analysis'. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2016, pp. 1715–1724.

[94] Yanghao Li, Naiyan Wang, Jiaying Liu and Xiaodi Hou. 'Demystifying neural style transfer'. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 2230–2236.

[95] Yijun Li, Fang Chen, Jimei Yang, Zhaowen Wang, Xin Lu and Ming-Hsuan Yang. 'Diversified texture synthesis with feed-forward networks'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 266–274.

[96] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu and Ming-Hsuan Yang. 'Universal style transfer via feature transforms'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 386–396.

[97] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua and Sing Bing Kang. 'Visual attribute transfer through deep image analogy'. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–15.

[98] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár and C Lawrence Zitnick. 'Microsoft coco: Common objects in context'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014, pp. 740–755.

[99] Huihui Liu, Yiding Yang and Xinchao Wang. 'Overcoming Catastrophic Forgetting in Graph Neural Networks'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 35. 10. 2021, pp. 8653–8661.

[100] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li and Errui Ding. 'Adaattn: Revisit attention mechanism in arbitrary neural style transfer'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 6629–6638.

[101] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye and Xinchao Wang. 'Dataset distillation via factorization'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 35. 2022, pp. 1100–1113.

[102] Songhua Liu, Jingwen Ye, Sucheng Ren and Xinchao Wang. 'Dynast: Dynamic sparse transformer for exemplar-guided image generation'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022, pp. 72–90.

[103] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Ruifeng Deng, Xin Li, Errui Ding and Hao Wang. 'Paint transformer: Feed forward neural painting with stroke prediction'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 6598–6607.

[104] Xiao-Chang Liu, Yong-Liang Yang and Peter Hall. 'Learning To Warp for Style Transfer'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 3702–3711.

[105] Gen Luo, Yiyi Zhou, Xiaoshuai Sun, Liujuan Cao, Chenglin Wu, Cheng Deng and Rongrong Ji. 'Multi-task Collaborative Network for Joint Referring Expression Comprehension and Segmentation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 10034–10043.

[106] Sihui Luo, Wenwen Pan, Xinchao Wang, Dazhou Wang, Haihong Tang and Mingli Song. 'Collaboration by Competition: Self-coordinated Knowledge Amalgamation for Multi-talent

Student Learning'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020, pp. 631–646.

[107] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang and Wenwu Zhu. 'Disentangled graph convolutional networks'. In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 4212–4221.

[108] Jiaqi Ma, Shuangrui Ding and Qiaozhu Mei. 'Towards more practical adversarial attacks on graph neural networks'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 4756–4766.

[109] Chris J Maddison, Daniel Tarlow and Tom Minka. 'A* sampling'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 27. 2014, pp. 3086–3094.

[110] Kaleel Mahmood, Rigel Mahmood and Marten Van Dijk. 'On the robustness of vision transformers to adversarial examples'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 7838–7847.

[111] Brais Martinez, Jing Yang, Adrian Bulat and Georgios Tzimiropoulos. 'Training binary neural networks with real-to-binary convolutions'. In: *International Conference on Learning Representations (ICLR)*. 2020.

[112] Julian McAuley, Christopher Targett, Qinfeng Shi and Anton Van Den Hengel. 'Image-based recommendations on styles and substitutes'. In: *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2015, pp. 43–52.

[113] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong and Richard Socher. 'The natural language decathlon: Multitask learning as question answering'. In: *arXiv preprint arXiv:1806.08730* (2018).

[114] Scott McLean, Gemma JM Read, Jason Thompson, Chris Baber, Neville A Stanton and Paul M Salmon. 'The risks associated with Artificial General Intelligence: A systematic review'. In: *Journal of Experimental & Theoretical Artificial Intelligence (JETAI)* (2021), pp. 1–15.

[115] Roey Mechrez, Itamar Talmi and Lihi Zelnik-Manor. 'The contextual loss for image transformation with non-aligned data'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 768–783.

[116] Stefan Milz, Georg Arbeiter, Christian Witt, Bassam Abdallah and Senthil Yogamani. 'Visual slam for automated driving: Exploring the applications of deep learning'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018, pp. 247–257.

[117] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller and O Anatole Von Lilienfeld. 'Machine learning of molecular electronic properties in chemical compound space'. In: *New Journal of Physics (NJP)* 15.9 (2013), p. 095003.

[118] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda and Michael M Bronstein. 'Geometric deep learning on graphs and manifolds using mixture model cnns'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5115–5124.

[119] Hoang NT and Takanori Maehara. 'Revisiting Graph Neural Networks: All We Have is Low-Pass Filters'. In: *arXiv preprint arXiv:1905.09550* (2019).

[120] Paarth Neekhara, Shehzeen Hussain, Shlomo Dubnov and Farinaz Koushanfar. 'Adversarial reprogramming of text classification neural networks'. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019, pp. 5215–5224.

[121] Paarth Neekhara, Shehzeen Hussain, Jinglong Du, Shlomo Dubnov, Farinaz Koushanfar and Julian McAuley. 'Cross-modal Adversarial Reprogramming'. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 2427–2435.

[122] Kiri Nichol. *Painter by Numbers*. 2016. URL: https://www.kaggle.com/c/painter-by-numbers.

[123] OpenAI. 'GPT-4 Technical Report'. In: *arXiv preprint arXiv:2303.08774* (2023).

[124] Emilio Parisotto, Jimmy Lei Ba and Ruslan Salakhutdinov. 'Actor-mimic: Deep multitask and transfer reinforcement learning'. In: *arXiv preprint arXiv:1511.06342* (2015).

[125] Bryan Perozzi, Rami Al-Rfou and Steven Skiena. 'Deepwalk: Online learning of social representations'. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2014, pp. 701–710.

[126] Charles R Qi, Hao Su, Kaichun Mo and Leonidas J Guibas. 'Pointnet: Deep learning on point sets for 3d classification and segmentation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 652–660.

[127] Charles Ruizhongtai Qi, Li Yi, Hao Su and Leonidas J Guibas. 'Pointnet++: Deep hierarchical feature learning on point sets in a metric space'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, 5105—5114.

[128] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen and Song-Chun Zhu. 'Learning human-object interactions by graph parsing neural networks'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 401–417.

[129] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler and Raquel Urtasun. '3d graph neural networks for rgbd semantic segmentation'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, pp. 5199–5208.

[130] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song and Nicu Sebe. 'Binary neural networks: A survey'. In: *Pattern Recognition* 105 (2020), p. 107281.

[131] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon and Ali Farhadi. 'Xnor-net: Imagenet classification using binary convolutional neural networks'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016, pp. 525–542.

[132] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg et al. 'A generalist agent'. In: *arXiv preprint arXiv:2205.06175* (2022).

[133] Sucheng Ren, Daquan Zhou, Shengfeng He, Jiashi Feng and Xinchao Wang. 'Shunted Self-Attention via Multi-Scale Token Aggregation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 10853–10862.

[134] Eric Risser, Pierre Wilmot and Connelly Barnes. 'Stable and controllable neural texture synthesis and style transfer using histogram losses'. In: *arXiv preprint arXiv:1701.08893* (2017).

[135] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta and Yoshua Bengio. 'Fitnets: Hints for thin deep nets'. In: *arXiv preprint arXiv:1412.6550* (2014).

[136] Yu Rong, Wenbing Huang, Tingyang Xu and Junzhou Huang. 'Dropedge: Towards deep graph convolutional networks on node classification'. In: *International Conference on Learning Representations (ICLR)*. 2020.

[137] David E Rumelhart, Geoffrey E Hinton and Ronald J Williams. 'Learning representations by back-propagating errors'. In: *Nature* 323.6088 (1986), pp. 533–536.

[138] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu and Raia Hadsell. 'Progressive neural networks'. In: *arXiv preprint arXiv:1606.04671* (2016).

[139] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz and Andrew Rabinovich. 'Superglue: Learning feature matching with graph neural networks'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4938–4947.

[140] Simon Schaefer, Daniel Gehrig and Davide Scaramuzza. 'AEGNN: Asynchronous event-based graph neural networks'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 12371–12381.

[141] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher and Tina Eliassi-Rad. 'Collective classification in network data'. In: *AI Magazine* 29.3 (2008), pp. 93–93.

[142] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun and Mingli Song. 'Amalgamating knowledge towards comprehensive classification'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 33. 01. 2019, pp. 3068–3075.

[143] Chengchao Shen, Mengqi Xue, Xinchao Wang, Jie Song, Li Sun and Mingli Song. 'Customizing student networks from heterogeneous teachers via adaptive knowledge amalgamation'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 3504–3513.

[144] Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo and Mingli Song. 'Progressive Network Grafting for Few-Shot Knowledge Distillation'. In: *arXiv preprint arXiv:2012.04915* (2020).

[145] Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo and Mingli Song. 'Progressive Network Grafting for Few-Shot Knowledge Distillation'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 35. 3. 2021, pp. 2541–2549.

[146] Chengchao Shen, Youtan Yin, Xinchao Wang, Xubin Li, Jie Song and Mingli Song. 'Training generative adversarial networks in one stage'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 3350–3360.

[147] Lu Sheng, Jing Shao, Ziyi Lin, Simon Warfield and Xiaogang Wang. 'Avatar-Net: Multi-scale Zero-shot Style Transfer by Feature Decoration'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8242–8250.

[148] David Silver, Satinder Singh, Doina Precup and Richard S Sutton. 'Reward is enough'. In: *Artificial Intelligence* 299 (2021), p. 103535.

[149] Jie Song, Ying Chen, Jingwen Ye and Mingli Song. 'Spot-adaptive knowledge distillation'. In: *IEEE Transactions on Image Processing (TIP)* 31 (2022), pp. 3359–3370.

[150] 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'. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2022), pp. 1–20.

[151] Takafumi Taketomi, Hideaki Uchiyama and Sei Ikeda. 'Visual SLAM algorithms: A survey from 2010 to 2016'. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), pp. 1–11.

[152] Andreas Veit and Serge Belongie. 'Convolutional Networks with Adaptive Inference Graphs'. In: *International Journal of Computer Vision (IJCV)* 128.3 (2020).

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

[154] Can Wang, Zhe Wang, Defang Chen, Sheng Zhou, Yan Feng and Chun Chen. 'Online Adversarial Distillation for Graph Neural Networks'. In: *arXiv preprint arXiv:2112.13966* (2021).

[155] Chen Wang, Le Zhang, Lihua Xie and Junsong Yuan. 'Kernel cross-correlator'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 32. 1. 2018.

[156] Chen Wang, Yuheng Qiu, Dasong Gao and Sebastian Scherer. 'Lifelong graph learning'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 13719–13728.

[157] Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, Xiangjian He, Yiguang Lin and Xuemin Lin. 'Binarized Graph Neural Network'. In: *arXiv preprint arXiv:2004.11147* (2020).

[158] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie and Minyi Guo. 'Graphgan: Graph representation learning with generative adversarial nets'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 32. 1. 2018.

[159] Junfu Wang, Yunhong Wang, Zhen Yang, Liang Yang and Yuanfang Guo. 'Bi-GCN: Binary Graph Convolutional Network'. In: *arXiv preprint arXiv:2010.07565* (2020).

[160] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong and Anshul Kanakia. 'Microsoft academic graph: When experts are not enough'. In: *Quantitative Science Studies (QSS)* 1.1 (2020), pp. 396–413.

[161] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma et al. 'Deep graph library: Towards efficient and scalable deep learning on graphs'. In: *International Conference on Learning Representations (ICLR) Workshop*. 2019.

[162] Pei Wang, Yijun Li and Nuno Vasconcelos. ‘Rethinking and improving the robustness of image style transfer’. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 124–133.

[163] Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang and Yoon Kim. ‘Learning to Grow Pretrained Models for Efficient Transformer Training’. In: *arXiv preprint arXiv:2303.00980* (2023).

[164] Wen Wang, Yang Cao, Jing Zhang and Dacheng Tao. ‘Fp-detr: Detection transformer advanced by fully pre-training’. In: *International Conference on Learning Representations (ICLR)*. 2021.

[165] Wen Wang, Jing Zhang, Yang Cao, Yongliang Shen and Dacheng Tao. ‘Towards data-efficient detection transformers’. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022, pp. 88–105.

[166] Yue Wang and Justin M Solomon. ‘Object dgcnn: 3d object detection using dynamic graphs’. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 34. 2021, pp. 20745–20758.

[167] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein and Justin M Solomon. ‘Dynamic graph cnn for learning on point clouds’. In: *ACM Transactions on Graphics (TOG)* 38.5 (2019), pp. 1–12.

[168] Yuxuan Wang, Jiakai Wang, Zixin Yin, Ruihao Gong, Jingyi Wang, Aishan Liu and Xianglong Liu. ‘Generating transferable adversarial examples against vision transformers’. In: *Proceedings of the ACM International Conference on Multimedia*. 2022, pp. 5181–5190.

[169] Zhipeng Wei, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein and Yu-Gang Jiang. ‘Towards transferable adversarial attacks on vision transformers’. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 36. 3. 2022, pp. 2668–2676.

[170] Boris Weisfeiler and Andrei A Lehman. ‘A reduction of a graph to a canonical form and an algebra arising during this reduction’. In: *Nauchno-Technicheskaya Informatsia* 2.9 (1968), pp. 12–16.

[171] Cheng Wen, Baosheng Yu and Dacheng Tao. ‘Learning progressive point embeddings for 3d point cloud generation’. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 10266–10275.

[172] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie and Bin Cui. 'Graph neural networks in recommender systems: a survey'. In: *ACM Computing Surveys (CSUR)* 55.5 (2022), pp. 1–37.

[173] Xiaolei Wu, Zhihao Hu, Lu Sheng and Dong Xu. 'StyleFormer: Real-Time Arbitrary Style Transfer via Parametric Style Composition'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 14618–14627.

[174] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing and Vijay Pande. 'MoleculeNet: a benchmark for molecular machine learning'. In: *Chemical Science* 9.2 (2018), pp. 513–530.

[175] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang and Jianxiong Xiao. '3d shapenets: A deep representation for volumetric shapes'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920.

[176] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts and Simon King. 'Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis'. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 4460–4464.

[177] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang and S Yu Philip. 'A comprehensive survey on graph neural networks'. In: *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* 32.1 (2020), pp. 4–24.

[178] Jianpeng Xu, Pang-Ning Tan, Lifeng Luo and Jiayu Zhou. 'Gspartan: a geospatio-temporal multi-task learning framework for multi-location prediction'. In: *Proceedings of the 2016 SIAM International Conference on Data Mining (SDM)*. 2016, pp. 657–665.

[179] Jianpeng Xu, Pang-Ning Tan, Jiayu Zhou and Lifeng Luo. 'Online multi-task learning framework for ensemble forecasting'. In: *IEEE Transactions on Knowledge and Data Engineering* 29.6 (2017), pp. 1268–1280.

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

[181] Wenju Xu, Chengjiang Long, Ruisheng Wang and Guanghui Wang. 'DRB-GAN: A Dynamic ResBlock Generative Adversarial Network for Artistic Style Transfer'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 6383–6392.

[182] Yufei Xu, Qiming Zhang, Jing Zhang and Dacheng Tao. 'Vitae: Vision transformer advanced by exploring intrinsic inductive bias'. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

[183] Sijie Yan, Yuanjun Xiong and Dahua Lin. 'Spatial temporal graph convolutional networks for skeleton-based action recognition'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 32. 1. 2018.

[184] Allen Y Yang, Roozbeh Jafari, S Shankar Sastry and Ruzena Bajcsy. 'Distributed recognition of human actions using wearable motion sensor networks'. In: *Journal of Ambient Intelligence and Smart Environments* 1.2 (2009), pp. 103–115.

[185] Erkun Yang, Cheng Deng, Wei Liu, Xianglong Liu, Dacheng Tao and Xinbo Gao. 'Pairwise relationship guided deep hashing for cross-modal retrieval'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 31. 1. 2017.

[186] Xingyi Yang, Zhou Daquan, Songhua Liu, Jingwen Ye and Xinchao Wang. 'Deep model reassembly'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 35. 2022, pp. 25739–25753.

[187] Xingyi Yang, Jingwen Ye and Xinchao Wang. 'Factorizing knowledge in neural networks'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022, pp. 73–91.

[188] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao and Xinchao Wang. 'Distilling knowledge from graph convolutional networks'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 7074–7083.

[189] Yiding Yang, Zunlei Feng, Mingli Song and Xinchao Wang. 'Factorizable Graph Convolutional Networks'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 20286–20296.

[190] Yiding Yang, Zhou Ren, Haoxiang Li, Chunluan Zhou, Xinchao Wang and Gang Hua. 'Learning dynamics via graph neural networks for human pose estimation and tracking'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 8074–8084.

[191] Yiding Yang, Xinchao Wang, Mingli Song, Junsong Yuan and Dacheng Tao. 'Spagan: Shortest path graph attention network'. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2019, pp. 4099–4105.

[192] Jingwen Ye, Yixin Ji, Xinchao Wang, Xin Gao and Mingli Song. 'Data-Free Knowledge Amalgamation via Group-Stack Dual-GAN'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 12516–12525.

[193] Jingwen Ye, Yongcheng Jing, Xinchao Wang, Kairi Ou, Dacheng Tao and Mingli Song. 'Edge-sensitive human cutout with hierarchical granularity and loopy matting guidance'. In: *IEEE Transactions on Image Processing (TIP)* 29 (2019), pp. 1177–1191.

[194] Jingwen Ye, Yifang Fu, Jie Song, Xingyi Yang, Songhua Liu, Xin Jin, Mingli Song and Xinchao Wang. 'Learning with Recoverable Forgetting'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022, pp. 87–103.

[195] Jingwen Ye, Yining Mao, Jie Song, Xinchao Wang, Cheng Jin and Mingli Song. 'Safe Distillation Box'. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 36. 3. 2022, pp. 3117–3124.

[196] Jingwen Ye, Yixin Ji, Xinchao Wang, Kairi Ou, Dapeng Tao and Mingli Song. 'Student Becoming the Master: Knowledge Amalgamation for Joint Scene Parsing, Depth Estimation, and More'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2829–2838.

[197] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer and Leonidas Guibas. 'A scalable active framework for region annotation in 3d shape collections'. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–12.

[198] Junho Yim, Donggyu Joo, Jihoon Bae and Junmo Kim. 'A gift from knowledge distillation: Fast optimization, network minimization and transfer learning'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4133–4141.

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

[200] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande and Jure Leskovec. 'Graph convolutional policy network for goal-directed molecular graph generation'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31. 2018, pp. 6412–6422.

[201] Jiahui Yu and Thomas S Huang. ‘Universally slimmable networks and improved training techniques’. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1803–1811.

[202] Ruonan Yu, Songhua Liu and Xinchao Wang. ‘Dataset Distillation: A Comprehensive Review’. In: *arXiv preprint arXiv:2301.07014* (2023).

[203] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng and Shuicheng Yan. ‘Metaformer is actually what you need for vision’. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 10819–10829.

[204] Xiyu Yu, Tongliang Liu, Xinchao Wang and Dacheng Tao. ‘On compressing deep models by low rank and sparse decomposition’. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 7370–7379.

[205] Sergey Zagoruyko and Nikos Komodakis. ‘Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer’. In: *arXiv preprint arXiv:1612.03928* (2016).

[206] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik and Silvio Savarese. ‘Taskonomy: Disentangling task transfer learning’. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3712–3722.

[207] Wei Zhai, Yang Cao, Zheng-Jun Zha, HaiYong Xie and Feng Wu. ‘Deep structure-revealed network for texture recognition’. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11010–11019.

[208] Wei Zhai, Yang Cao, Jing Zhang and Zheng-Jun Zha. ‘Exploring Figure-Ground Assignment Mechanism in Perceptual Organization’. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 35. 2022, pp. 17030–17042.

[209] Wei Zhai, Hongchen Luo, Jing Zhang, Yang Cao and Dacheng Tao. ‘One-shot object affordance detection in the wild’. In: *International Journal of Computer Vision* 130.10 (2022), pp. 2472–2500.

[210] Yibing Zhan, Jun Yu, Ting Yu and Dacheng Tao. ‘Multi-task compositional network for visual relationship detection’. In: *International Journal of Computer Vision (IJCV)* 128 (2020), pp. 2146–2165.

[211] Yibing Zhan, Jun Yu, Ting Yu and Dacheng Tao. 'On exploring undetermined relationships for visual relationship detection'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5128–5137.

[212] Hang Zhang and Kristin Dana. 'Multi-style generative network for real-time transfer'. In: *arXiv preprint arXiv:1703.06953* (2017).

[213] Qiming Zhang, Yufei Xu, Jing Zhang and Dacheng Tao. 'VSA: Learning Varied-Size Window Attention in Vision Transformers'. In: *arXiv preprint arXiv:2204.08446* (2022).

[214] Qiming Zhang, Yufei Xu, Jing Zhang and Dacheng Tao. 'Vitaev2: Vision transformer advanced by exploring inductive bias for image recognition and beyond'. In: *arXiv preprint arXiv:2202.10108* (2022).

[215] Xiang Zhang and Marinka Zitnik. 'Gnnguard: Defending graph neural networks against adversarial attacks'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 9263–9275.

[216] Yu Zhang and Qiang Yang. 'An overview of multi-task learning'. In: *National Science Review* 5.1 (2018), pp. 30–43.

[217] Zhanpeng Zhang, Ping Luo, Chen Change Loy and Xiaoou Tang. 'Facial landmark detection by deep multi-task learning'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014, pp. 94–108.

[218] Haimei Zhao, Wei Bian, Bo Yuan and Dacheng Tao. 'Collaborative Learning of Depth Estimation, Visual Odometry and Camera Relocalization from Monocular Videos'. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2020, pp. 488–494.

[219] Lingxiao Zhao and Leman Akoglu. 'PairNorm: Tackling Oversmoothing in GNNs'. In: *International Conference on Learning Representations (ICLR)*. 2020.

[220] Yang Zheng, Xiaoyi Feng, Zhaoqiang Xia, Xiaoyue Jiang, Ambra Demontis, Maura Pintor, Battista Biggio and Fabio Roli. 'Why Adversarial Reprogramming Works, When It Fails, and How to Tell the Difference'. In: *arXiv preprint arXiv:2108.11673* (2021).

[221] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li and Maosong Sun. 'Graph neural networks: A review of methods and applications'. In: *arXiv preprint arXiv:1812.08434* (2018).

[222] Ling Zhou, Zhen Cui, Chunyan Xu, Zhenyu Zhang, Chaoqun Wang, Tong Zhang and Jian Yang. 'Pattern-Structure Diffusion for Multi-Task Learning'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4514–4523.

[223] Shangchen Zhou, Jiawei Zhang, Wangmeng Zuo and Chen Change Loy. 'Cross-scale internal graph neural network for image super-resolution'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 3499–3509.

[224] Sheng Zhou, Yucheng Wang, Defang Chen, Jiawei Chen, Xin Wang, Can Wang and Jiajun Bu. 'Distilling holistic knowledge with graph neural networks'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10387–10396.

[225] Xiatian Zhu, Shaogang Gong et al. 'Knowledge distillation by on-the-fly native ensemble'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 31. 2018, 7528—7538.

[226] Marinka Zitnik and Jure Leskovec. 'Predicting multicellular function through multi-layer tissue networks'. In: *Bioinformatics* 33.14 (2017), pp. i190–i198.