UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2022-36
The Faculty of Sciences, Technology and Medicine

DISSERTATION

Defence held on 04/04/2022 in Esch-sur-Alzette
to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Zhiqiang ZHONG
Born on 19 July 1993 in Suzhou, (China)

LEVERAGING GRAPH MACHINE LEARNING FOR SOCIAL NETWORK ANALYSIS

Dissertation defence committee

Dr. Jun Pang, dissertation supervisor
Senior Researcher, Université du Luxembourg

Dr. Sjouke Mauw, Chairman
Professor, Université du Luxembourg

Dr. Martin Theobald, Vice Chairman
Professor, Université du Luxembourg

Dr. Cheng-Te Li
Associate Professor, National Cheng Kung University

Dr. Davide Mottin
Assistant Professor, Aarhus University

Abstract

Leveraging Graph Machine Learning for Social Network Analysis

by Zhiqiang Zhong

As a ubiquitous complex system in quotidian life around everyone, *online social networks* (OSNs) provide a rich source of information about billions of users worldwide. To some extent, OSNs have mirrored our real society: people perform a multitude of different activities in OSNs as they do in the offline world, such as establishing social relations, sharing life moments, and expressing opinions about various topics. Therefore, understanding OSNs is of immense importance. One key characteristic of human social behaviour in OSNs is their inter-relational nature, which can be represented as graphs. Due to sparsity and complex structure, analysing these graphs is quite challenging and expensive.

Over the past several decades, many expert-designed approaches to graphs have been proposed with elegant theoretical properties and successfully addressed numerous practical problems. Nevertheless, most of them are either not data-driven or do not benefit from the rapidly growing scale of data. Recently, in the light of remarkable achievements of artificial intelligence, especially deep neural networks techniques, *graph machine learning* (GML) has emerged to provide us with novel perspectives to understanding and analysing graphs. However, the current efforts of GML are relatively immature and lack attention to specific scenarios and characteristics of OSNs. Based on the pros and cons of GML, this thesis discusses several aspects of how to build advanced approaches to better simplify and ameliorate OSN analytic tasks. Specifically:

1. *Overcoming flat message-passing graph neural networks.* One of the most widely pursued branches in GML research, *graph neural networks* (GNNs), follows a similar *flat* message-passing principle for representation learning. Precisely, information is iteratively passed between adjacent nodes along observed edges via non-linear transformation and aggregation functions. Its effectiveness has been widely proved; however, two limitations need to be tackled: *(i)* they are costly in encoding long-range information spanning the graph structure; *(ii)* they are failing to encode features in the high-order neighbourhood in the graphs as they only perform information aggregation across the observed edges in the original graph. To fill up the gap, we propose a novel *hierarchical message-passing* framework to facilitate the existing GNN mechanism. Following this idea, we design two practical implementations, i.e., HC-GNN and AdamGNN, to demonstrate the framework's superiority.

2. *Extending graph machine learning to heterophilous graphs.* The existing GML approaches implicitly hold a *homophily* assumption that nodes of the same class tend to be connected. However, previous expert studies have shown the enormous importance of addressing the *heterophily* scenario, where "opposites attract", is essential for network analysis and fairness study. We demonstrate the possibility of extending GML to heterophilous graphs by simplifying supervised node classification models on heterophilous graphs (CLP) and designing an unsupervised heterophilous graph representation learning model (Selene).

3. *Online social network analysis with graph machine learning.* As GML approaches have demonstrated significant effectiveness over general graph analytic tasks, we perform two practical OSN analysis projects to illustrate the possibility of employing GML in practice. Specifically, we propose a *semantic image graph embedding* (SiGraph) to improve OSN image recognition task with the associated hashtags semantics and a simple GNN-based *neural link prediction* framework (NeuLP) to boost the performance with tiny change.

Keywords: Graph machine learning, Social network analysis, Graph neural networks, Hierarchical structure, Homophily/Heterophily graphs, Link prediction, Online image content understanding.

To the memory of my father.

Acknowledgements

This thesis represents not only my work of the four-year PhD study but also a milestone in my research career and my life. In addition to the produced papers and thousands of lines of code, it is more important than the journey over the past four years has taught me how to think, investigate, write and present, and have taught me to be curious, optimistic and collaborative, and have made me realise my ordinary and extraordinary. I am deeply thankful to the following people, without the help and support of whom I would not have managed to complete this journey of a lifetime.

First and foremost, I would like to thank Jun Pang for giving me the opportunity to pursue a PhD under his supervision. I am tremendously grateful for his continuous support, for countless inspiring discussions, for giving me the freedom to pursue a variety of research directions, and for frequently encouraging me to explore the unknown in the past four years.

Being part of SaToSS group was a wonderful and genuinely growing experience. Thanks to Sjouke Mauw for his encouragement, advice and kind help. It has been a pleasure to have him as one of my CET (comité d'encadrement de thèse) committees. Thanks to Olga Gadyatskaya, Aleksandr Pilgun and Stanislav Dashevskyi, who helped me a lot during the first teaching assistant experience of the first semester. They gave me great courage and confidence. I have fond memories of sharing an office with Marco Crepaldi and Semen Yurkov; I could not have hoped for a more joyful environment for working, especially enjoyed the "afternoon tea break" (with Aleksandr Pilgun too). I am also pleased to have had the opportunity to collaborate with Yang Zhang, Ninghan Chen and Xihui Chen, and thanks for the inspiring discussions and encouragement. To all my other colleagues in SaToSS, thanks for making the past four years an unforgettable experience.

Thanks to Martin Theobald for his continuous guidance and insightful discussion. I also count myself lucky for having him as one of my CET committees and encouraging me to perform in-depth research.

I am also very happy to have had the opportunity to collaborate with fantastic researchers outside of the University of Luxembourg: Cheng-Te Li, Sergey Ivanov, Guadalupe Gonzalez and Daniele Grattarola. I have had the chance to work with very talented MSc students: HaoCheng Ho and Yi-Wei Cheng.

Many special thanks go to my friends and colleagues, including Cui Su, Ninghan Chen, Marco Crepaldi and Semen Yurkov, for all the wonderful moments with them in Luxembourg.

Finally, I would like to thank my family. My mother, not only gave me endless love, but I would not be where I am today as a person without having her in my life. Thank you for enduring my absence in the past few years. To my fiancee, Zhe Xie: thank you for your steadfast support in every way I could imagine with encouragement and tolerance.

Sincerely,
Zhiqiang Zhong

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Human Social Activities and Social Networks

Millions[1] of years of evolution have made humanity stand out from millions of species on this planet[2], and also reshaped the way we live and our perception and cognition. Meanwhile, human interaction also evolved from cooperative hunting to collective farming to novel forms after the *industrial revolution*[3]. Predominantly driven by the *fourth industrial revolution*, humans have taken perception to new extremes to create the *internet*. Humans can perform social activities in virtual scenes - the *online social networks* (OSNs).

OSNs have become a vital part of modern human life. They contain a wealth of useful information about human online social activity, including establishing social relations, sharing life moments, pursuing social interactions and expressing opinions about various topics. To some extent, OSNs have mirrored our real social society and have become an indispensable tool for understanding and solving offline real-world issues, such as social healthcare analysis [TY12, DYKC18], academic network analysis [GN02, TZY+08], information and trust propagation [GKRT04, JE10], natural disaster mitigation [NSJ12, ÖE12] and viral marketing [DTSS10, ML10].

How can we formalise and investigate online social networks? - A heuristic question is naturally raised in our minds. To answer this question, the researchers proposed formally defining OSNs as large-scale ubiquitous *complex systems* composed of a collective of components (individual or human groups) that may interact with each other (social interactions). In order to simplify the processing of OSNs, researchers further proposed a definition - *graph* - which provides a unified way to represent information in a complex system. A graph is a data structure describing a set of entities, represented as *nodes*, and their pairwise interactions represented as *edges*. Additional rich properties corresponding to each node and edge are summarised as node and edge *attributes*.

With the assistance of the concept of the graph, we can conveniently represent OSNs. However, understanding and analysing them is still challenging due to the involuted dependencies, transformation, and relationships.

[1] Or billions, when considering the beginning of biology

[2] "Sapiens: A Brief History of Humankind", a book by Yuval Noah Harari

[3] `https://en.wikipedia.org/wiki/Industrial_Revolution`

1.2 Artificial Intelligence and Graph Machine Learning

"Those who can imagine anything, can create the impossible. "

Alan Turing

The desire to understand *online social networks* (OSNs) has spawned a variety of scientific disciplines. Theoretical physics, graph theory, graph signal processing and the study of *artificial intelligence* (AI) are the most prominent examples of scientific fields. The past decade has observed impressive achievements of AI in terms of theoretical understanding and practical applications [RN20].

The work in this thesis is situated in the field of *machine learning*, which is one of the most widely pursued branches in AI research [JM15]. The principal objective of machine learning is to answer the question of if/how we can build intelligent systems and design agile algorithms that learn from data and experience. Such a novel schema contrasts the traditional approaches in computer science, where systems are explicitly programmed to follow a precisely outlined sequence of instructions [DCG+89].

The problem of machine learning is accustomed approached by fitting a well-designed model to data with the goal that this learned model can generalise to new data. Traditionally, machine learning models are built on top of designed or processed feature sets, extracted from the raw data format using a pre-defined procedure. For instance, the pixel distribution in image data and the word occurrence statistics in text documents are treated as processed features. After, we can train the learning model upon the obtained features by tuning the model's parameters to memory knowledge from the data. The memorised knowledge will be employed on new experiences.

Nevertheless, the process of developing practicable features - also referred to as *feature engineering* - plays a decisive role in the model performance and requires lots of human effort. Moreover, the developed feature extractor are often not transferable that we need to design for each dataset and task.

Deep learning, another widely pursued branch in AI Research, has enjoyed the immense spotlight in recent years [LBH15]. Instead of manually developing learning models with handcrafted features extractors, it directly learns representations from raw data. This is achieved by stacking multiple *layers* of differentiable linear and non-linear transformations and by training such a model with the raw data in an *end-to-end* fashion. The derived class of models that follow the deep learning paradigm is referred to *deep neural networks*. Such unified and strong models have demonstrated superior power in many areas, including computer vision [LBD+89, KSH12], nature language processing [LBBH98, BDVJ03], automaton controlling [MKS+15, SSS+17], etc [LKB+17, PSY+19].

Despite recent soaring successes of deep learning in many areas, those aforementioned deep neural networks are still falling short of general ability for complex systems that take the form of graphs. So how can we build systems and algorithms that work well with an arbitrary structure? For example, the prestigious

deep neural network architecture, *convolutional neural networks* [LBBH98], comprises trainable grid-shaped parameter sets that are tied or shared across image locations, requiring grid-structured data with a fixed size. Another reputable architecture, *recurrent neural networks* [Hoc98], shares parameter sets over time steps and hence generalise more favourably on sequential data. But it still cannot cope with more complex graph structure data.

In this thesis, we focus on a different machine learning category, referred to as *graph machine learning* (GML) [CWPZ19, ZCZ20, XSY+21], that is designed as a predominant approach to finding effective data representations from complex systems that take the form of graphs. The principal target of GML is to extract the desired features of a graph as informative representations that can be easily used by downstream tasks such as node-level, edge-level and graph-level, pattern discovery, analysis, classification and regression tasks. One way to achieve this is by designing handcrafted feature extractors and explicit algorithms that learn from graph-structured data and experience, which is similar to traditional machine learning approaches and suffers from parallel deficiencies. Another way is by structuring the representations and computations in a deep neural network in the form of a graph, leading to a class of models named *graph neural networks* (GNNs) [GMS05, SGT+09, KW17, WPC+21], which is of central importance in this thesis. Precisely, we reflect on the mechanism of GNNs based on two typical online human social activity scenarios and propose our solutions. Moreover, two interesting social network analysis projects are further presented to illustrate the effectiveness of GML.

1.3 Scope and Research Questions

This thesis is structured in three parts: Part I first demonstrates the *flatness* of vanilla message-passing *graph neural networks* (GNNs) and then introduces two advanced graph neural network models to overcome the *flat* message-passing mechanism. Part II discusses the availability of applying *graph machine learning* (GML) models on *heterophilous* graphs, i.e., nodes of different classes tend to be connected ("opposites attract") and presents two GML approaches that work well on both homophily and heterophily. Part III presents two fascinating example projects that demonstrate utilising GML methods for *online social network* (OSN) analysis.

The contributions of this thesis are guided by the following questions:

Research Question 1: *Can we develop graph machine learning models that overcome the limitations of flat message-passing mechanism to learn more comprehensive representations?*

Our main contribution to address this research question is designing a novel *hierarchical* message-passing idea that allows information aggregation through observed graph structure and implicit hierarchical relations from the graph. We name the novel designed framework as *hierarchical message-passing graph neural networks*. Following the proposed hierarchical message-passing idea, in Chapter 3, we

propose practical and effective implementations, i.e., *hierarchical community-aware graph neural network* (HC-GNN), to adopt several hierarchical community detection algorithms to reveal the implicit hierarchical structure and build up message-passing pipelines follow this structure. Theoretical complexity analysis confirms HC-GNN's efficiency, and extensive experimental results demonstrate its effectiveness on node-wise and edge-wise tasks on inductive and transductive settings.

Research Question 2: *Can we realise the hierarchical message-passing idea without any manual preprocessing?*

After introducing the hierarchical community-aware graph neural network that implements the hierarchical message-passing idea with hierarchical community detection algorithms in Chapter 3. We further argue that the predefined hierarchical structure may not be optimised for every downstream task on graphs with different characteristics, and such a framework violates the end-to-end training mechanism of deep learning models. Therefore, we introduce another ameliorated model, namely *adaptive multi-grained graph neural networks* (AdamGNN), which can adaptively construct the hierarchical structure and enhance the node representation with the learned multi-grained semantics. We demonstrate applications of AdamGNN to a multitude of different downstream tasks in Chapter 4.

Having introduced advanced GNNs for overcoming the *flat* message-passing mechanism in Part I of this thesis (Chapters 3 and 4), Part II investigates whether/how GML models can be developed and efficiently applied to graphs with *heterophily*, i.e., linked nodes are likely to be from different classes.

Research Question 3: *Can graph machine learning models be efficiently applied on heterophilous graphs?*

Classic GML models implicitly assume graphs have homophily property, that said nodes of the same class tend to be connected. However, heterophilous graphs widely exist in the real world and addressing the heterophilous scenario is essential for OSN analysis and fairness study. Therefore, we perform a large-scale empirical study to investigate the performance of existing node representation learning models and propose a simple but strong node classification framework, termed *compatible label propagation* (CLP), in Chapter 5.

Research Question 4: *Can graph machine learning models be utilised for unsupervised node representation learning on heterophilous graphs?*

To address this challenging task, that learning useful node representation *without* supervision from graphs where linked nodes are likely to be from different classes, we first analyse the performance and embedding mechanisms of existing unsupervised graph representation learning models. We find that the homophily ratio can significantly affect the performance of graph representation learning models that

rely on the graph structure. There are three challenges to designing valid unsupervised graph representation learning models that work well with homophily and heterophily. To address these three research challenges, we introduce our solution: the *self-supervised network embedding* (Selene) in Chapter 6. This model demonstrates outstanding performance on challenging node clustering task that does not utilise any supervisor information on homophilous and heterophilous graphs.

Aside from the above-listed research questions that were investigated in this thesis, Part III presents two interesting practical OSN analysis projects.

Research Question 5: *Can online social networks be used for image classification?*

Image is one of the most popular information shared in OSNs, and large-scale image classification is a classic and challenging task. Chapter 7 proposes to utilise OSN languages, i.e., image associated hashtags, to organise images into a graph. We utilise graph representation learning methods to learn image representations that involve the relationship among images' contents, which can significantly improve image classification performance.

Research Question 6: *Can graph neural networks be utilised for link prediction in online social networks?*

Link prediction, which aims to estimate the likelihood of the existence of an edge between two nodes, is one of the key problems in OSN mining and has been widely applied for many business scenes. In Chapter 8, we propose a unified end-to-end deep learning model, namely *neural link prediction* (NeuLP). The model integrates the linearity and non-linearity user interactions to overcome the limitation of GNNs in capturing long-range interactions. Experimental evaluation demonstrates NeuLP's significant improvements over several baseline models. And NeuLP achieves a reliable link prediction given two users' different types of attributes.

After presenting the main body of our work in Parts I, II and III, we conclude and outline interesting directions for future work in Chapter 9.

1.4 List of Papers

The main content (Parts I-III) of this thesis is based on the following papers:

- **Zhiqiang Zhong**, Cheng-Te Li and Jun Pang. "Hierarchical Message-Passing Graph Neural Networks" [ZLP20b], Under review of *Journal Track of European Conference on Machine Learning and Data Mining*. Chapter 3 is mainly based on this paper.

- **Zhiqiang Zhong**, Cheng-Te Li and Jun Pang. "Multi-grained Semantics-aware Graph Neural Networks" [ZLP20a], Under review of *IEEE Transactions on Knowledge and Data Engineering*. Chapter 4 is based on this paper.
- **Zhiqiang Zhong**, Sergey Ivanov and Jun Pang. "Simplifying Node Classification on Heterophilous Graphs with Compatible Label Propagation", Under review of *2022 International Conference on Knowledge Discovery and Data Mining*. Chapter 5 is based on this paper.
- **Zhiqiang Zhong**, Guadalupe Gonzalez, Daniele Grattarola and Jun Pang. "Unsupervised Heterophilous Network Embedding via *r*-Ego Network Discrimination", Under review of *IEEE Transactions on Neural Networks and Learning Systems*. Chapter 6 is based on this paper.
- **Zhiqiang Zhong**, Yang Zhang and Jun Pang. "A Graph-Based Approach to Explore Relationship Between Hashtags and Images" [ZZP19], In *2019 International Conference on Web Information Systems Engineering*. Chapter 7 is based on this paper.
- **Zhiqiang Zhong**, Yang Zhang and Jun Pang. "NeuLP: An End-to-End Deep-Learning Model for Link Prediction" [ZZP22], In *2020 International Conference on Web Information Systems Engineering*. Chapter 8 is based on this paper.

The author has further contributed to the following papers:

- **Zhiqiang Zhong**, Cheng-Te Li and Jun Pang. "Personalised Meta-path Generation for Heterogeneous Graph Neural Networks" [ZLP22], In *2022 Journal Track of European Conference on Machine Learning and Data Mining*.
- Ninghan Chen, **Zhiqiang Zhong** and Jun Pang "An Exploratory Study of COVID-19 Information on Twitter in the Greater Region" [CZP21], In *Journal of Big Data and Cognitive Computing*.
- Yi-Wei Cheng, **Zhiqiang Zhong**, Jun Pang and Cheng-Te Li. "Hierarchical Bipartite Graph Convolutional Network for Recommendation", Under review of *ACM Transactions on Intelligent Systems and Technology*.
- Ninghan Chen, Xihui Chen, **Zhiqiang Zhong**, Jun Pang. "From #Jobsearch to #Mask: Improving COVID-19 Cascade Prediction with Spillover Effects" [CCZP21], In *2021 International Conference on Advances in Social Networks Analysis and Mining*. Another extended version "Exploring Spillover Effects for COVID-19 Cascade Prediction" [CCZP22], In *Journal of Entropy*.
- Ninghan Chen, Xihui Chen, **Zhiqiang Zhong** and Jun Pang. ""Double vaccinated, 5G boosted!": Learning Attitudes towards COVID-19 Vaccination From Social Media", Under review of *2022 International Conference on Knowledge Discovery and Data Mining*.

Chapter 2

Background

In this chapter, we provide a brief introduction to several important background topics and notations that are extensively used throughout this thesis. Additional background for each specific topic will be introduced where necessary in later chapters. In what follows, we give an introduction to *graph representation learning* in Section 2.2, to *graph neural networks* in Section 2.3, to *node representations for social network analysis* in Section 2.4 and lastly to *self-supervised learning* in Section 2.5.

2.1 Notations

This section provides a reference for the most commonly used notations throughout this thesis. Individual chapters introduce additional notations where necessary.

Table 2.1: Summary of notations

Notation	Description
$\mathcal{G}$	Graphical representation of the data with n nodes
$\mathcal{V}$	Set of nodes in the graph, $\lvert\mathcal{V}\rvert = n$
$\mathcal{E}$	Set of edges in the graph, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
$\mathbf{A}$	Adjacency matrix that summarises $\mathcal{V}$ and $\mathcal{E}$, $\mathbf{A} \in \{0,1\}^{n \times n}$
$\mathbf{X}$	Node attribute matrix, $\mathbf{X} \in \mathbb{R}^{n \times \pi}$
$\mathbf{x}_i$	Node attribute of node v_i, $\mathbf{x}_i \in \mathbb{R}^{\pi}$
$\mathcal{Y}$	Set of class labels for all $v \in \mathcal{V}$
y_i	Class label of node v_i
$\mathbf{Y}$	Set of one-hot class labels for all $v \in \mathcal{V}$
$\mathbf{y}_i$	One-hot class label of node v_i
$f(\mathbf{x}), f(\mathbf{X})$	If $f(\cdot)$ is a function defined on scalars, then $f(\mathbf{x})$ and $f(\mathbf{X})$ are to be understood as an element-wise application of $f(\cdot)$ on the elements of the vector $\mathbf{x}$ or matrix $\mathbf{X}$
$f_\theta(\cdot), f_\phi(\cdot)$	Parameter dependency of functions is typically made explicit with a greek alphabet θ or ϕ.
$\mathbf{Z}$	Node representation matrix, $\mathbf{Z} \in \mathbb{R}^{n \times d}$
$\mathbf{z}_i$	Node representation vector of node v_i, $\mathbf{z}_i \in \mathbb{R}^{d}$
$\mathcal{L}$	Loss function

2.2 Graph Representation Learning

Graphs are a ubiquitous data structure that models objects and their relationships, such as social networks, biological protein-protein networks, recommendation systems, etc [HYL17a]. However, how to deal with the complex and diverse non-Euclidean data structure is a challenging and essential task. Over the last decade, in the light of the outstanding developments of artificial intelligence, graph representation learning has become a promising approach to machine learning with graphs. The goal of graph representation learning is to learn low-dimensional representations for all entities of a graph [ZCZ20], which can be applied for a multitude of different applications that involve node-wise [KW17, XHLJ19], edge-wise [KW16, ZC18] and graph-wise tasks [ZLLW19, BWS+20]. We start with necessary definitions and preliminaries.

Definition 1 (Graph)**.** *An attributed unweighted graph with n nodes can be formally represented as* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$*, where* $\mathcal{V}$ *is the set of nodes and* $|\mathcal{V}| = n$*,* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ *denotes the set of edges, and* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$*, with each* $\mathbf{x}_i \in \mathbb{R}^\pi$*, represents node features (*π *is the dimensionality of node features).* $\mathcal{Y}$ *stands for a set with class labels for all* $v \in \mathcal{V}$*. For subsequent discussion, we summarise* $\mathcal{V}$ *and* $\mathcal{E}$ *into an adjacency matrix* $\mathbf{A} \in \{0,1\}^{n \times n}$*.*

To process the graph-structured data, traditional approaches mainly rely on handcrafted features, including graph statistics [BKT13], kernel functions [VSKB10] and experts designed features [LK07]. However, these conventional approaches often suffer from practical limits on large-scale graphs with rich node and edge attributes. Recently, *graph representation learning* emerged to be a promising direction.

Definition 2 (Graph representation learning)**.** *For node-wise and edge-wise tasks, given a graph* $\mathcal{G}$*, the task of graph representation learning (or termed as graph embedding in some papers) is to learn a mapping function* $f_n : \mathcal{G} \rightarrow \mathbf{Z}$*, where* $\mathbf{Z} \in \mathbb{R}^d$*, and each row* $\mathbf{z}_i \in \mathbf{Z}$ *corresponds to node* v_i*'s representation. For graph-wise tasks, similarly it aims to learn a mapping* $f_g : \mathcal{D} \rightarrow \mathbf{Z}$*, where* $\mathcal{D} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots\}$ *is a set of graphs, each row* $\mathbf{z}_i \in \mathbf{Z}$ *corresponds to the graph* $\mathcal{G}_i$*'s representation. The mapping function's effectiveness is evaluated by applying* $\mathbf{Z}$ *to different downstream tasks.*

Depending on the *graph representation learning* (GRL) model's inherent architecture, existing GRL methods can be categorised into "shallow" or "deep" groups. Shallow GRL methods comprise an embedding lookup table that directly encodes each node as a vector and is optimised during training. Within this group, several Skip-Gram [MSC+13]-based GRL methods have been proposed, such as DeepWalk [PARS14] and node2vec [GL16] as well as their matrix factorisation interpretation NetMF [QDM+18], LINE [TQW+15] and PTE [TQM15]. To better capture the structural identity of nodes independent of network position and neighbourhood's labels, struc2vec [RSF17] constructs a hierarchy to encode structural node similarities at different scales.

The deep GRL methods - *graph neural networks* (GNNs) - have recently shown promising results in modelling structural and relational data [WPC+21]. GNNs

related GRL models and applications are of central importance to the topics covered in this thesis, we give a formal and more comprehensive introduction in the next section.

2.3 Graph Neural Networks

"*All models are wrong, but some are useful.*"

George E.P. Box

Graph neural networks (GNNs) are a class of neural network models suitable for processing graph-structured data. They use the graph structure ($\mathbf{A}$) and node features ($\mathbf{X}$) to learn a representation vector of a node ($\mathbf{z}_v$), or the entire graph ($\mathbf{z}_\mathcal{G}$). Modern GNNs follow a common idea of a recursive neighbourhood aggregation or message-passing scheme, where we iteratively update the representation of a node by aggregating representations of its neighbouring nodes. After ℓ iterations of aggregation or message-passing, a node's representation captures the graph structural information within ℓ-hop neighbourhood.

Let $\hat{\mathbf{A}} = (\hat{\mathbf{A}}_{uv})_{u,v\in\mathcal{V}}$, where $\hat{\mathbf{A}}_{uv}$ is a normalised value of $\mathbf{A}_{uv}$. Thus, we can formally define ℓ-th layer of a GNN as:

$$\begin{aligned}
\mathbf{m}_a^{(\ell)} &= \textsc{Aggregate}^N(\{\hat{\mathbf{A}}_{uv}, \mathbf{h}_u^{(\ell-1)} \mid u \in \mathcal{N}(v)\}),\\
\mathbf{m}_v^{(\ell)} &= \textsc{Aggregate}^I(\{\hat{\mathbf{A}}_{uv} \mid u \in \mathcal{N}(v)\})\, \mathbf{h}_v^{(\ell-1)},\\
\mathbf{h}_v^{(\ell)} &= \textsc{Combine}(\mathbf{m}_a^{(\ell)}, \mathbf{m}_v^{(\ell)})
\end{aligned} \tag{2.1}$$

where $\textsc{Aggregate}^N(\cdot)$ and $\textsc{Aggregate}^I(\cdot)$ are two possibly differential parameterised functions. $\mathbf{m}_a^{(\ell)}$ is aggregated message from node v's neighbourhood nodes ($\mathcal{N}(v)$) with their structural coefficients, and $\mathbf{m}_v^{(\ell)}$ is the residual message from node v after performing an adjustment operation to account for structural effects from its neighbourhood nodes. After, $\mathbf{h}_v^{(\ell)}$ is the learned as representation vector of node v with combining $\mathbf{m}_a^{(\ell)}$ and $\mathbf{m}_v^{(\ell)}$, termed as $\textsc{Combine}(\cdot)$, at the ℓ-th iteration/layer. Note that, we initialise $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ and the final learned representation vector after L iterations/layers $\mathbf{z}_v = \mathbf{h}_v^{(L)}$. In addition, in terms of the representation of an entire graph ($\mathbf{z}_\mathcal{G}$), we can apply a READOUT function to aggregate node representations of all nodes of the graph $\mathcal{G}$, as

$$\mathbf{z}_\mathcal{G} = \text{READOUT}(\{\mathbf{z}_v \mid \forall \mathbf{z}_v \in \mathcal{V}\}) \tag{2.2}$$

where READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function.

Following the general structure of GNNs as defined in Equation 2.1, we can further generalise the existing GNNs as variants of it. For instance, several classic and popular GNNs can be summarised as Table 2.2.

The complete formations of these GNN models are represented as:

Table 2.2: Define different GNN variants according to Equation 2.1.

GNN Model	$\textsc{Aggregate}^N(\cdot)$	$\textsc{Aggregate}^I(\cdot)$	$\textsc{Combine}(\cdot)$
GCN	$\sum_{u \in \mathcal{N}(v)} \frac{\mathbf{W}^{(\ell)}\mathbf{h}_u^{(\ell-1)}}{\sqrt{\lvert\mathcal{N}(u)\rvert\lvert\mathcal{N}(v)\rvert}}$	$\frac{\mathbf{W}^{(\ell)}\mathbf{h}_v^{\ell-1}}{\sqrt{\lvert\mathcal{N}(v)\rvert\lvert\mathcal{N}(v)\rvert}}$	$\sigma(\textsc{Sum}(\mathbf{m}_a^{(\ell)}, \mathbf{m}_v^{(\ell)}))$
GraphSAGE	$\textsc{Agg}(\{\mathbf{h}_u^{(\ell-1)} \mid u \in \mathcal{N}(v)\})$	$\mathbf{h}_v^{(\ell-1)}$	$\sigma(\mathbf{W}^{(\ell)} \cdot \textsc{Concat}(\mathbf{m}_a^{(\ell)}, \mathbf{m}_v^{(\ell)}))$
GAT	$\sum_{u \in \mathcal{N}(v)} \alpha_{uv}\mathbf{W}^{(\ell)}\mathbf{h}_u^{(\ell-1)}$	$\alpha_{vv}\mathbf{W}^{(\ell)}\mathbf{h}_v^{(\ell-1)}$	$\sigma(\textsc{Sum}(\mathbf{m}_a^{(\ell)}, \mathbf{m}_v^{(\ell)}))$
GIN	$\sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(\ell-1)}$	$(1+\epsilon)\mathbf{h}_v^{(\ell-1)}$	$\text{MLP}_\theta(\textsc{Sum}(\mathbf{m}_a^{(\ell)}, \mathbf{m}_v^{(\ell)}))$

Graph convolutional network (GCN) [KW17] is the classic modern GNN which applies two normalised mean aggregations to aggregate feature vectors node v's neighbourhood nodes $\mathcal{N}(v)$ and combine with itself:

$$\mathbf{h}_v^{(\ell)} = \sigma\Big(\sum_{u \in \mathcal{N}(v)} \frac{\mathbf{W}^{(\ell)}\mathbf{h}_u^{(\ell-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} + \frac{\mathbf{W}^{(\ell)}\mathbf{h}_v^{(\ell-1)}}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(v)|}}\Big) \tag{2.3}$$

where $\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}$ is a constant normalisation coefficient for the edge $\mathcal{E}_{uv}$, which is calculated from the normalised adjacent matrix $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-\mathbf{1/2}}$. $\mathbf{D}$ is the diagonal node degree matrix of $\mathbf{A}$. $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n \times d}$ is a trainable weight matrix of layer ℓ and σ is a non-linear activation (ReLU).

GraphSAGE [HYL17b] learns aggregation functions to inductively generate node representations by sampling and aggregating node feature vectors from node v's neighbourhood nodes:

$$\mathbf{h}_v^{(\ell)} = \sigma(\mathbf{W}^{(\ell)} \cdot \textsc{Concat}(\textsc{Agg}(\{\mathbf{h}_u^{(\ell-1)} \mid u \in \mathcal{N}(v)\}), \mathbf{h}_v^{(\ell-1)})) \tag{2.4}$$

where $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n \times d}$ is a trainable weight matrix of layer ℓ and σ is a non-linear activation (ReLU). GraphSAGE considered three different aggregation functions (Agg), such as mean aggregator, LSTM aggregator [HS97] and pooling aggregator.

Graph attention network (GAT) [VCC+18] adaptively learns a normalisation coefficient α_{uv} for edge $\mathcal{E}_{uv}$ to aggregate feature vectors from node v's neighbourhood nodes $\mathcal{N}(v)$ and combine with itself:

$$\begin{aligned} \mathbf{h}_v^{(\ell)} &= \sigma\Big(\sum_{(u,v) \in \mathcal{E}} \alpha_{uv}^{(\ell)}\mathbf{W}^{(\ell)}\mathbf{h}_u^{(\ell-1)}\Big) \\ \alpha_{uv}^{(\ell)} &= \text{Softmax}(\text{LeakReLU}(\mathbf{a}[\mathbf{W}\mathbf{h}_u^{(\ell-1)} \parallel \mathbf{W}\mathbf{h}_v^{(\ell-1)}])) \end{aligned} \tag{2.5}$$

where $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n \times d}$ is a trainable weight matrix of layer ℓ, σ is a non-linear activation (ReLU) and $\parallel$ is the concatenation operator.

Graph isomorphism network (GIN) [XHLJ19] takes the sum aggregation over neighbourhood nodes of a node, followed by the *multi-layer perceptron* (MLP) [Ros61]:

$$\mathbf{h}_v^{(\ell)} = \text{MLP}^{(\ell)}((1+\epsilon^{(\ell)})\mathbf{h}_v^{(\ell-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(\ell-1)}) \tag{2.6}$$

where $\epsilon^{(\ell)}$ is a learnable parameter or fixed scalar.

We will review additional prior and concurrent work on GRL and GNNs related to our contributions where necessary. For an overview of recent variants and applications of GRL and GNNs, we recommend the comprehensive review articles [CZC18, CWPZ19, ZCZ20, WPC+21].

2.4 Graph Representation Learning for Social Network

With the advent of *graph representation learning* (GRL), social network analysis has been evolved into a new schema of classification and prediction models on low-dimensional latent representations. Typical example applications include node role identification [BCM11], personalised recommendation [ZZP22], social healthcare [BM21], academic network analysis [TZY+08], graph classification [ZCNC18] and epidemic trend study [CZP21]. Next, we formally represent several example social network analysis tasks that are of great importance to the topics covered in this thesis.

Node classification. In social networks ($\mathcal{G}$), users (nodes) ($\mathcal{V}$) are often associated with semantic labels ($\mathcal{Y}$) relevant to certain about them, such as gender, age, affiliation, interest etc. However, these graphs are often partially (or sparsely) labelled or even unlabelled due to the high cost of node selection and labelling. With partially labelled nodes (supervised settings) ($\mathcal{Y}_{Train}$), the node classification aims to identify labels for the rest of unlabelled nodes ($\mathcal{Y}_{Test}$) by leveraging connectivity patterns (homophily or heterophily) of labelled ones extracted from the graph structure. On the other hand, if there is no given labelling information about the graph's nodes, which are termed unsupervised settings, we name this task as *node clustering*. The main goal of node clustering is to predict labels for all nodes ($\mathcal{Y}_{\mathcal{V}}$) by identifying nodes with similar structural information and node attributes. For example, Twitter attempts to identity fraud accounts to protest other users from being scammed.

Link prediction. The graph structures ($\mathbf{A}$) of social networks ($\mathcal{G}$) are not always complete as partial friendship links between users can be temporally missing. Link prediction aims to infer the presence of emerging links ($\mathbf{A}^{+}$) in the future based on node attributes and observed graph structure and evolution mechanism. Since social networks evolve fast and continuously, link prediction is crucial to know what could happen soon and can help recover a whole social graph when we only have an incomplete view of it. For instance, this technique has been widely used for personalised recommendations on Facebook and Amazon.

Graph classification. In social networks ($\mathcal{G}$), users (nodes) ($\mathcal{V}$) tend to be often interacted by nodes with similar interests or background so that different social networks (graph set) or user groups (sub-graph set) ($\mathcal{D} = \{\mathcal{G}_1, \mathcal{G}_2, \dots\}$) are formed. By finding commonalities among users of the same group and differences between different user groups, we can categorise them as different categories. For example, LinkedIn identifies different career concerns of different professional groups.

2.5 Self-Supervised Learning

Self-supervised learning (SSL) provides a promising learning paradigm that reduces the dependence on manual labels [LZH+21]. In SSL, models are learned by solving a series of handcrafted auxiliary tasks (so-called pre-given tasks), in which the supervision signals are acquired from data itself automatically without the need for manual annotation. With the help of well-designed pre-given tasks, SSL enables the model to learn more informative representations from unlabelled data to achieve better performance, generalisation and robustness on various downstream tasks [JT21]. This approach naturally fits graph-structured data, as relations are given by the edges in the graph [LPJ+21]. We can cast this problem in with a mathematical formulation of *graph self-supervised learning* (GSSL).

Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. We construct an encoder-decoder framework to formalise GSSL. The encoder f_θ (parameterised by θ) aims to learn a low-dimensional representations $\mathbf{Z}$ for all nodes $\mathcal{V}$ of $\mathcal{G}$. In general, the encoder f_θ can be any GNNs as introduced in Section 2.3. The pre-given decoder f_ϕ (parameterised by ϕ) takes $\mathbf{Z}$ as its input for the pre-given tasks. The architecture of f_ϕ depends on specific downstream tasks.

Under this framework, GSSL can be formulated as:

$$\theta^*, \phi^* = \underset{\theta,\phi}{\operatorname{argmin}} \mathcal{L}_{gssl}(f_\theta, f_\phi, D) \tag{2.7}$$

where D denotes the graph data distribution that satisfies $(\mathcal{V}, \mathcal{E}) \sim D$ in an unlabelled graph $\mathcal{G}$, and $\mathcal{L}_{gssl}$ is the GSSL loss function that regularises the output of pre-given decoder according to the specific crafted tasks. In the following, we present a taxonomy of GSSL whose category is classified into (*i*) generation-based methods, (*ii*) auxiliary property-based methods, (*iii*) contrast-based methods and (*iv*) hybrid methods.

Generation-based methods. They form the pretext task as the graph data reconstruction from two perspectives: node/edge attributes and graph structure. Specifically, they focus on the node/edge features or/and graph adjacency reconstructions. In such a case, Equation 2.7 can be further derived as:

$$\theta^*, \phi^* = \underset{\theta,\phi}{\operatorname{argmin}} \mathcal{L}_{gssl}(f_\phi(f_\theta(\widehat{\mathcal{G}})), \mathcal{G}) \tag{2.8}$$

where $\widehat{\mathcal{G}}$ denotes the graph data with perturbed node/edge features or/and adjacency matrix. For most of the generation-based approaches, the self-supervised objective function $\mathcal{L}_{gssl}$ is typically defined to measure the difference between the reconstructed and the original graph data.

Auxiliary property-based methods. They enrich the supervision signals by capitalising on a larger set of attributive and topological graph properties. In particular, for different crafted auxiliary properties, we further categorise these methods into two

types: regression- and classification-based. Formally, they can be formulated as:

$$\theta^*, \phi^* = \underset{\theta,\phi}{\operatorname{argmin}} \mathcal{L}_{gssl}(f_\phi(f_\theta(\widehat{\mathcal{G}})), c) \tag{2.9}$$

where c denotes the specific crafted auxiliary properties. For regression-based approaches, c can be localised or global graph properties, such as the node degree or distance to clusters within $\mathcal{G}$. For classification-based methods, on the other hand, the auxiliary properties are typically constructed as pseudo labels, such as the graph partition or cluster indices.

Contrast-based methods. They are usually developed based on the concept of *mutual information* (MI) maximisation, where the estimated MI between augmented instances of the same object (e.g., node, subgraph, and graph) is maximised. For contrastive-based GSSL, Equation 2.7 is reformulated as:

$$\theta^*, \phi^* = \underset{\theta,\phi}{\operatorname{argmin}} \mathcal{L}_{gssl}(f_\phi(f_\theta(\mathcal{G}^{(1)}, \mathcal{G}^{(2)}))) \tag{2.10}$$

where $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ are two differently augmented instances of $\mathcal{G}$. In these methods, the pretext decoder f_ϕ indicates the discriminator that estimates the agreement between two instances and minimise the MI between negative samples, which is implicitly included in $\mathcal{L}_{gssl}$.

Hybrid methods. They take advantage of previous categories and are consist of more than one pretext decoder and/or training objective. We formulate this branch of methods as the weighted or unweighted combination of two or more GSSL schemes based on formulas from Equations 2.8 to 2.10.

We will make use of GSSL for unsupervised representation learning on heterophilous graphs and node clustering in Chapter 6 using an objective-based on a contrast-based method (Equation 6.8).

Part I

Overcoming *flat* Message-passing Graph Neural Networks

Motivation and Summary

Graphs are a ubiquitous data structure, and *graph neural networks* (GNNs) have been proved as a useful approach for learning node representations from large scale graphs for a wide variety of graph-related tasks. However, GNNs mainly follow a *flat* message-passing mechanism, which may lead to the following two limitations: (*i*) it is very costly to encode global information on the graph topology and (*ii*) GNNs may fail to model meso- and macro-level semantics hidden in the graph.

This part of the thesis explores how we can overcome *flat* message-passing GNNs to learn more effective node and graph representations.

In Chapter 3, we introduce the *hierarchical message-passing graph neural networks* (HMGNNs) framework, which performs message-passing in a hierarchical manner. In addition, we propose a practical implementation model following the idea of the HMGNNs framework, *hierarchical community-aware graph neural network* (HC-GNN), which exploits well-known hierarchical community detection algorithms to build up the hierarchical structure. We theoretically confirm HC-GNN's significant ability in capturing long-range interactions without introducing heavy computation complexity and empirically demonstrate the efficiency and effectiveness of HC-GNN on node- and edge-level tasks under different settings.

Chapter 4 introduces the *adaptive multi-grained graph neural networks* (AdamGNN) model, which follows the hierarchical message-passing idea but does not rely on an external algorithm to build the hierarchical structure. That said, AdamGNN can adaptively construct a proper hierarchical structure of one graph for different downstream tasks. The practical evaluation confirms the effectiveness of AdamGNN on node- edge- and graph-level tasks.

Chapter 3

Hierarchical Message-passing Graph Neural Networks

3.1 Introduction

Graphs are a ubiquitous data structure that models objects and their relationships within complex systems, such as social networks, biological networks, recommendation systems, etc [WPC+21]. Learning node representation from a large graph has been proved as a useful approach for a wide variety of network analysis tasks, including link prediction [ZC18], node classification [ZAL18] and community detection [CLB19].

Graph neural networks (GNNs) are currently one of the most promising paradigms to learn and exploit node representations due to their effective ability to encode node features and graph topology in transductive, inductive, and few-shot settings [ZCZ20]. Many existing GNN models follow a similar *flat* message-passing principle where information is iteratively passed between adjacent nodes along observed edges. Such a paradigm is able to incorporate local information surrounded by each node [GSR+17]. However, it has been proven to suffer from several drawbacks [XHLJ19, MWW20, LWWL20].

Among these deficiencies of flat message-passing GNNs, the limited ability for information aggregation over long-range has attracted significant attention [LHW18], since most graph-related tasks require the interactions between nodes that are not directly connected [AY21]. That said, flat message-passing GNNs struggle in capturing dependencies between distant node pairs. Inspired by the outstanding effectiveness of very deep neural network models that have been demonstrated in computer vision and natural language processing domains [LBH15], a natural solution is stacking lots of GNN layers together to directly increase the receptive field of each node. Consequently, deeper models have been proposed by simplifying the aggregation design of GNNs and accompanied by well-designed normalisation units [CWH+20]. Nevertheless, Alon and Yahav have theoretically shown that flat GNNs are susceptible to being a *bottleneck* when aggregating messages across a long path and lead to severe *over-squashing* issues [AY21].

On the other hand, in this chapter, we further argue another crucial deficiency of flat message-passing GNNs is that they rely on only aggregating messages across

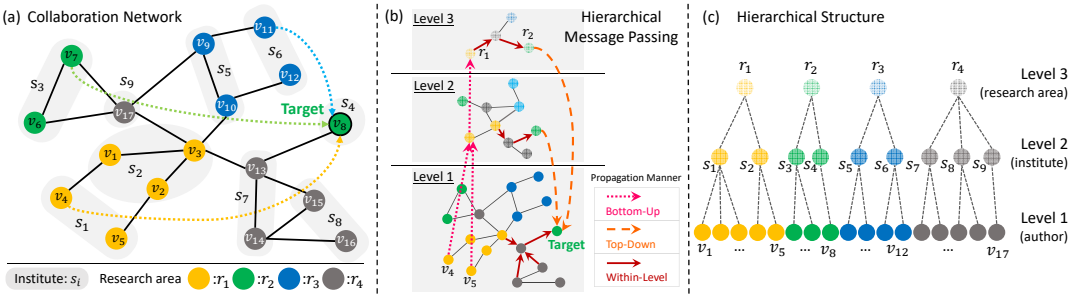


Figure 3.1: Elaboration of the proposed hierarchical message passing: (a) a collaboration network, (b) an illustration of hierarchical message-passing mechanism based on (a) and (c), and (c) an example of the identified hierarchical structure.

the observed topological structure. The hierarchical semantics behind the graph structure provides useful information and should be incorporated into the learning of node representations. Taking the collaboration network in Figure 3.1-(a) as an example, author nodes highlighted in light yellow come from the same institutes, and nodes filled with different colours indicate authors in various research areas. In order to generate the node representation of a given author, existing GNNs mainly capture the co-author level information depending on the explicit graph structure. However, information hidden at *meso* and *macro* levels is neglected. In the example of Figure 3.1, meso-level information means authors belong to the same institutes and their connections to adjacent institutes. Macro-level information refers to authors of the same research areas and their relationship with related research areas. Both meso- and macro-level knowledge cannot be directly modelled through flat message passing via observed edges.

In this chapter, we investigate the idea of a hierarchical message-passing mechanism to enhance the information aggregation pipeline of GNNs. The ultimate goal is to make the node representation learning process aware of both long-range interactive information and implicit multi-resolution semantics within the graph.

We note that a few graph pooling approaches have recently delivered various attempts to use the hierarchical structure idea [GJ19, YYM+18, HLL+19, RST20]. g-U-Net [GJ19] employs a bottom-up and top-down pooling operation; however, it does not allow long-range message-passing. DiffPool [YYM+18], AttPool [HLL+19] and ASAP [RST20] target at graph classification tasks instead of enabling node representations to capture long-range dependencies and multi-grained semantics of one graph. Moreover, P-GNNs [YYL19] create a different information aggregation mechanism that utilises sampled anchor nodes to impose topological position information into learning node representations. While P-GNNs can capture global information, the hierarchical semantics mentioned above is still overlooked, and the global message-passing is not realised. Besides, the anchor-set sampling process is time-consuming for large graphs, and it cannot work well under the inductive setting.

Specifically, we present a novel framework, *hierarchical message-passing graph neural networks* (HMGNNs), elaborated in Figure 3.1. In detail, HMGNNs can be organised into the following four phases.

(*i*) Hierarchical structure generation. To overcome long-distance obstacles in the process of GNN message-passing, we propose to use a hierarchical structure to reduce the size of graph $\mathcal{G}$ gradually, where nodes at each level t are integrated into different super nodes $(s_1^{t+1}, \ldots, s_n^{t+1})$ at each level $t+1$.

(*ii*) t-level super graph construction. In order to allow the message passing among generated same-level super nodes, we construct a super graph $\mathcal{G}_t$ based on the connections between nodes at its lower level $t-1$.

(*iii*) Hierarchical message propagation. With the generated hierarchical structure for a given graph, we develop three propagation manners, including bottom-up, within-level and top-down.

(*iv*) Model learning. Last, we leverage task-specific loss functions and a gradient descent procedure to train the model.

Designing a feasible hierarchical structure is crucial for HMGNNs, as the hierarchical structure determines how messages can be passed through different levels and what kind of meso- and macro-level information to be encoded in node representations. In this chapter, we consider (but are not restricted to) *network communities*. As a natural graph property, the community has been proved very useful for many graph mining tasks [WPL14, WCW$^+$17]. Lots of community detection methods can generate hierarchical community structures. Here, we propose an implementation model for the proposed framework, *hierarchical community-aware graph neural network* (HC-GNN). HC-GNN exploits a well-known hierarchical community detection method, i.e., the *Louvain* method [BGLL08] to build up the hierarchical structure, which is then used for the hierarchical message-passing mechanism.

The theoretical analysis illustrates HC-GNN's remarkable capacity in capturing long-range information without introducing heavy additional computation complexity. Extensive empirical experiments are conducted on 9 graph datasets to reveal the performance of HC-GNN on a variety of tasks, i.e., link prediction, node classification, and community detection, under transductive, inductive and few-shot settings. The results show that HC-GNN consistently outperforms a set of state-of-the-art approaches for link prediction and node classification. In the few-shot learning setting, where only 5 samples of each label are used to train the model, HC-GNN achieves a significant performance improvement, up to 16.4%. We also deliver a few empirical insights: (a) the lowest level contributes most to node representations; (b) how to generate the hierarchical structure has a significant impact on the quality of node representations; (c) HC-GNN maintains an outstanding performance for graphs with different levels of sparsity perturbation; (d) HC-GNN possess significant flexibility in incorporating different GNN encoders, which means HC-GNN can achieve superior performance with advanced flat GNN encoders.

3.2 Additional Related Work

Flat message-passing GNNs. They perform graph convolution, directly aggregate node features from neighbours in the given graph, and stack multiple GNN layers to capture long-range node dependencies [KW17, HYL17b, VCC+18, XHLJ19]. However, they were observed *not* to benefit from more than a few layers, and recent studies have theoretically shown to be *over-smoothing*. On the other hand, GraphRNA [HSLH19] presents graph recurrent networks to capture interactions between far-away nodes. Still, we can not apply it to inductive learning settings, and the recurrent aggregations introduce high computation costs. P-GNNs [YYL19] incorporate a novel global information aggregation mechanism based on the distance of a given target node to each anchor set. However, P-GNNs sacrifice the ability to exist GNNs on inductive node-wise tasks and the anchor-set sampling operation brings a high computational cost for large-size graphs. Recently, deeper *flat* GNNs have been proposed by simplifying the aggregation design of GNNs and accompanied with well-designed normalisation units [CWH+20]. Nevertheless, it [AY21] has theoretically shown that flat GNNs are susceptible to being a *bottleneck* when aggregating messages across a long path and lead to severe *over-squashing* issues. Moreover, we will theoretically discuss the advantages of our method compared with flat GNNs in Section 3.4.3, in terms of long-range interactive capability and complexity.

Hierarchical representation GNNs. In recent years, some studies generalise the pooling mechanism of computer vision [RFB15] to GNNs for graph representation learning [YYM+18, HLL+19, GJ19, RST20]. However, most of them, such as DiffPool [YYM+18], AttPool [HLL+19] and ASAP [RST20], are designed for graph classification tasks rather than learning node representations to capture long-range dependencies and multi-resolution semantics. Thus they cannot be directly applied to node-level tasks. g-U-Net [GJ19] defines a similarity-based pooling operator construct the hierarchical structure and implements bottom-up and top-down operations. Despite the success of g-U-Net on producing graph-level representations, they cannot model the multi-grained semantics and realise long-range message-passing. HARP [CPHS18] and LouvainNE [BMD+20] are two unsupervised network representation approaches that adopt a hierarchical structure, but they do not support the supervised training paradigm to optimise for specific tasks, and they cannot be applied with inductive settings.

Table 3.1 summarises the critical advantages of the proposed HC-GNN and compares it with a number of state-of-the-art methods published recently. We are the first to present the hierarchical message passing to efficiently model long-range informative interaction and multi-grained semantics. In addition, our HC-GNN can utilise the community structures and be applied for transductive, inductive and few-shot inferences.

Table 3.1: Model comparison in aspects of Node-wise Task (NT), SUPervised training paradigm (SUP), Transductive Inference (TI), Inductive Inference (II), Long-range Information (LI), and Hierarchical Semantics for Node Representations (HSNR).

	NT	**SUP**	**TI**	**II**	**LI**	**HSNR**
GCN [KW17]	√	√	√	√		
GraphSAGE [HYL17b]	√	√	√	√		
GAT [VCC+18]	√	√	√	√		
GIN [XHLJ19]	√	√	√	√		
P-GNNs [YYL19]	√	√	√		√	
GCNII [CWH+20]	√	√	√	√	√	
DiffPool [YYM+18]		√	√	√		
g-U-Net [GJ19]	√	√	√	√		√
AttPool [HLL+19]		√	√	√		
ASAP [RST20]		√	√	√		
GraphRNA [HSLH19]	√	√	√			
HARP [CPHS18]	√		√			
LouvainNE [BMD+20]	√		√			
HC-GNN	√	√	√	√	√	√

3.3 Problem Statement

Problem setup. Given a graph $\mathcal{G}$ and a pre-defined representation dimension d, the goal is to learn a mapping function $f_\theta : \mathcal{G} \to \mathbf{Z}$, where $\mathbf{Z} \in \mathbb{R}^{n \times d}$ and each row $\mathbf{z}_i \in \mathbf{Z}$ corresponds to the node v_i's representation. The effectiveness of f_θ is evaluated by applying $\mathbf{Z}$ to different tasks, including node classification, link prediction, and community detection. Table 3.2 lists the additional mathematical notation used in this chapter.

Table 3.2: Summary of additional notations.

Notation	**Description**
T	Number of hierarchy level
$\mathcal{G}_t$	Super graph at level t
s_n^t	n-th super node of $\mathcal{G}_t$ at level t
$\mathcal{H}$	Set of constructed super graphs
$\mathcal{N}(v)$	Set of neighbour nodes of node v

Flat node representation learning. Prior to introducing the hierarchical message-passing mechanism, we first give a general review of existing *graph neural networks* (GNNs) with *flat* message-passing. Let $\hat{\mathbf{A}} = (\hat{\mathbf{A}}_{uv})_{u,v \in \mathcal{V}}$, where $\hat{\mathbf{A}}_{uv}$ is a normalised value of $\mathbf{A}_{uv}$. Thus, we can formally define ℓ-th layer of a flat GNN as:

$$\begin{aligned}
\mathbf{m}_a^{(\ell)} &= \textsc{Aggregate}^N(\{\hat{\mathbf{A}}_{uv}, \mathbf{h}_u^{(\ell-1)} \mid u \in \mathcal{N}(v)\}), \\
\mathbf{m}_v^{(\ell)} &= \textsc{Aggregate}^I(\{\hat{\mathbf{A}}_{uv} \mid u \in \mathcal{N}(v)\})\, \mathbf{h}_v^{(\ell-1)}, \\
\mathbf{h}_v^{(\ell)} &= \textsc{Combine}(\mathbf{m}_a^{(\ell)}, \mathbf{m}_v^{(\ell)})
\end{aligned} \tag{3.1}$$

where $\textsc{Aggregate}^N(\cdot)$ and $\textsc{Aggregate}^I(\cdot)$ are two possibly differential parameterised functions. $\mathbf{m}_a^{(\ell)}$ is aggregated message from node v's neighbourhood nodes ($\mathcal{N}(v)$) with their structural coefficients, and $\mathbf{m}_v^{(\ell)}$ is the residual message from node v after performing an adjustment operation to account for structural effects from its neighbourhood nodes. After, $\mathbf{h}_v^{(\ell)}$ is the learned as representation vector of node v by with combining $\mathbf{m}_a^{(\ell)}$ and $\mathbf{m}_v^{(\ell)}$, termed as $\textsc{Combine}(\cdot)$, at the ℓ-th iteration/layer. Note that, we initialise $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ and the final learned representation vector after L iterations/layers $\mathbf{z}_v = \mathbf{h}_v^{(L)}$.

Take the classic *graph convolutional network* (GCN) [KW17] as an example, which applies two normalised mean aggregations to aggregate feature vectors node v's neighbourhood nodes $\mathcal{N}(v)$ and combine with itself:

$$\mathbf{h}_v^{(\ell)} = \text{ReLU}\Big(\sum_{u \in \mathcal{N}(v)} \frac{\mathbf{W}^{(\ell)}\mathbf{h}_u^{(\ell-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} + \frac{\mathbf{W}^{(\ell)}\mathbf{h}_v^{(\ell-1)}}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(v)|}}\Big) \tag{3.2}$$

where $\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}$ is a constant normalisation coefficient for the edge $\mathcal{E}_{uv}$, which is calculated from the normalised adjacent matrix $\hat{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-\mathbf{1/2}}$. $\mathbf{D}$ is the diagonal node degree matrix of $\mathbf{A}$. $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n \times d}$ is a trainable weight matrix of layer ℓ. From Equation 3.1 and Equation 3.2, we can find that existing GNNs iteratively pass messages between adjacent nodes along observed edges, which will lead to two significant limitations: (a) the limited ability for information aggregation over long-range. They need to stack k layers to capture interactions within k steps for each node. (b) they are infeasible in encoding meso- and macro-level graph semantics.

3.4 Proposed Approach

We propose a framework, *hierarchical message-passing graph neural networks* (HMGNNs), whose core idea is to use a hierarchical message-passing structure to enable node representations to receive long-range messages and multi-grained semantics from different levels. Figure 3.2 provides an overview of the proposed framework, consisting of four components. First, we create a hierarchical structure to coarsen the input graph $\mathcal{G}$ gradually. Nodes at each level t of the hierarchy are grouped into different super nodes $(s_1^t, \ldots, s_n^t)$. Second, we further organise level t generated super nodes into a super graph $\mathcal{G}_{t+1}$ at level $t+1$ based on the connections between nodes at level t, in order to enable message-passing that encodes the interactions between generated super nodes. Third, we develop three different propagation schemes to propagate messages among nodes within the same level and across different levels. At last, after obtaining node representations, we use the task-specific loss function and a gradient descent procedure to train the model.

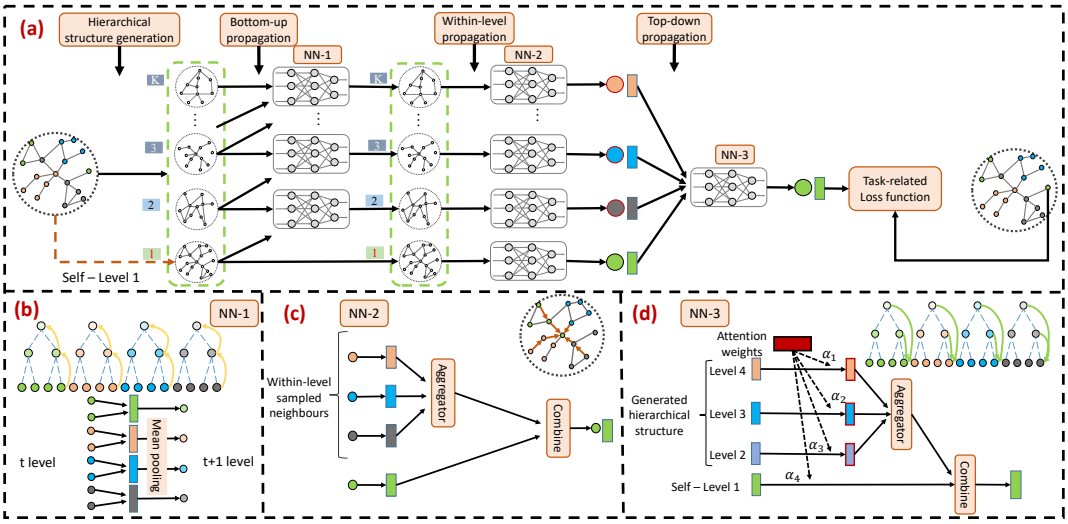


Figure 3.2: (a) The architecture of HMGNNs: we first generate a hierarchical structure, in which each level is formed as a super graph, use the level t graph to update nodes of level $t+1$ graph (bottom-up propagation), apply the typical neighbour aggregation on each level's graph (within-level propagation), use the generated node representations from level $2 \leq t \leq T$ to update node representations at the level 1 (top-down propagation), and optimises the model via a task-specific loss. (b) NN-1: bottom-up propagation. (c) NN-2: within-level propagation. (d) NN-3: top-down propagation.

3.4.1 Hierarchical Message-passing GNNs

I. Hierarchical structure generation. A graph $\mathcal{G}$ can be naturally organised by super node structures, in which densely inter-connected nodes are grouped. For example in Figure 3.1-(a), authors $\{v_1, v_2, \ldots, v_{17}\}$ can be grouped into different super nodes $\{s_1, s_2, \ldots, s_9\}$ based on their institutes. Institutes also can be grouped into higher-level super nodes $\{r_1, r_2, \ldots, r_4\}$ according to research areas. Meanwhile, there is a relationship between nodes at different levels, as indicated by dashed lines in Figure 3.1-(c). Hence, we can generate a hierarchical structure to depict the inter- and intra-relationships among authors, institutes, and research areas. We will discuss how to implement the hierarchical structure generation in Section 3.4.2.

II. t-Level super graph construction. The level t super graph $\mathcal{G}_t$ is constructed based on level $t-1$ graph $\mathcal{G}_{t-1}$ ($t \geq 2$), where $\mathcal{G}_1$ represents the original graph $\mathcal{G}$. Given all nodes at level $t-1$, i.e., $\{s_1^{t-1}, \ldots, s_m^{t-1}\}$, we consider every node s_i^{t-1} belonging to the same super node as a super node in the super graph $\mathcal{G}_t$, and create an edge between super nodes s_i^{t-1} and s_j^{t-1} if there exist more than γ edges in $\mathcal{G}_{t-1}$ connecting elements in s_i^{t-1} and elements in s_j^{t-1}, where γ is a hyper-parameter and $\gamma = 1$ by default. In this way, we represent the hierarchical structure $\mathcal{H}$ as a list of graphs $\mathcal{H} = \{\mathcal{G}_1, \ldots, \mathcal{G}_T\}$, where $\mathcal{G}_1 = \mathcal{G}$. In which inter-level edges are created to depict the relationships between super nodes at levels t and $t-1$, if a level $t-1$ node has a corresponding super node at level t, as shown in Figure 3.1-(c). We initialise the feature vectors of generated super nodes to be zero vectors with the same length as the original node feature vector $\mathbf{x}_i$. Take the collaboration network in Figure 3.1 as an example. At the micro-level (level 1), we have authors and their co-authorship relations. At the meso-level (level 2), we organise authors according to

their affiliations and establish relations between institutes. At the macro-level (level 3), institutes are further grouped according to their research areas, and we have the relations among the research areas. In addition, inter-level links are also created to depict the relationships between authors and institutes and between institutes and research areas.

III. Hierarchical message propagation. The hierarchical message-passing mechanism works as a supplementary process to enhance the node representations with long-range interactions and multi-grained semantics. Thus it does not change the flat node representation learning process as described in Section 3.3, to ensure the local information is well maintained. And we adopt the classic GCN, as described in Equation 3.2, as our default flat GNN encoder throughout this chapter. Particularly, the hierarchical message-passing mechanism consists of ℓ-th layer consisting of 3 steps.

1. *Bottom-up propagation.* After obtaining node representations ($\mathbf{h}_{s^{t-1}}^{(\ell)}$) of $\mathcal{G}_{t-1}$ with ℓ-th flat information aggregation, we perform bottom-up propagation, i.e., NN-1 in Figure 3.2-(b), using node representations in $\mathcal{G}_{t-1}$ to update node representations in $\mathcal{G}_t$ ($t \geq 2$) in the hierarchy $\mathcal{H}$, as follows:

$$\mathbf{a}_{s_i^t}^{(\ell)} = \frac{1}{|s_i^t| + 1} \left(\sum_{s^{t-1} \in s_i^t} \mathbf{h}_{s^{t-1}}^{(\ell)} + \mathbf{h}_{s_i^t}^{(\ell-1)} \right) \tag{3.3}$$

   where s_i^t is a super node in $\mathcal{G}_t$, and s^{t-1} is a node in $\mathcal{G}_{t-1}$ that belongs to s_i^t in $\mathcal{G}_t$. $\mathbf{h}_{s_i^t}^{(\ell-1)}$ is the node representation of s_i^t that generated by layer $\ell-1$ in graph $\mathcal{G}_t$, $|s_i^t|$ is the number of nodes of level $t-1$ that belonging to super node s_i^t, and $\mathbf{a}_{s_i^t}^{(\ell)}$ is the updated representation of s_i^t.

2. *Within-level propagation.* We explore the typical *flat* GNN encoders [KW17, HYL17b, VCC+18, XHLJ19, CWH+20] to propagate information within each level's graph $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T\}$, i.e., NN-2 in Figure 3.2-(c). The aim is to aggregate neighbours' information and update within-level node representations. Specifically, the information aggregation at level t is depicted as follows:

$$\begin{aligned} \mathbf{m}_a^{(\ell)} &= \textsc{Aggregate}^N(\{\hat{\mathbf{A}}_{uv}^t, \mathbf{a}_u^{(\ell)} \mid u \in \mathcal{N}^t(v)\}), \\ \mathbf{m}_v^{(\ell)} &= \textsc{Aggregate}^I(\{\hat{\mathbf{A}}_{uv}^t \mid u \in \mathcal{N}^t(v)\})\, \mathbf{a}_v^{(\ell)}, \\ \mathbf{b}_v^{(\ell)} &= \textsc{Combine}(\mathbf{m}_a^{(\ell)}, \mathbf{m}_v^{(\ell)}) \end{aligned} \tag{3.4}$$

   where $\mathbf{a}_u^{(\ell)}$ is the node representation of u after bottom-up propagation at the ℓ-th layer, $\mathcal{N}^t(v)$ is a set of nodes adjacent to v at level t, and $\mathbf{b}_v^{(\ell)}$ is the aggregated node representation of v based on local neighbourhood information. Note that we adopt the classic GCN, as described in Equation 3.2, as our default GNN encoder throughout this chapter. We will discuss the possibility of incorporating with other advanced GNN encoders in Section 3.5.3.

3. *Top-down propagation.* The top-down propagation is illustrated by NN-3 in Figure 3.2-(d). We use node representations in $\{\mathcal{G}_2, \ldots, \mathcal{G}_T\}$ to update the representations of original nodes in $\mathcal{G}$. The importance of messages at different levels can be different for other tasks. Hence, we adopt the attention mechanism [VCC+18] to adaptively learn the contribution weights of different levels during top-down integration, given by:

$$\mathbf{h}_v^{(\ell)} = \text{ReLU}(\mathbf{W} \cdot \text{MEAN}\{\alpha_{uv}\mathbf{b}_u^{(\ell)}\}), \forall u \in \mathcal{C}(v) \cup \{v\} \tag{3.5}$$

where α_{uv} is a trainable normalised attention coefficient between node v to super node u or itself, MEAN is an element-wise mean operation, $\mathcal{C}(v)$ denotes the set of different-level super nodes from level $\{2, \ldots, K\}$ that node v belongs to ($|\mathcal{C}(v)| = K - 1$), and ReLU is the activation function. $\mathbf{H}^{(\ell)}$ is the generated node representation of layer ℓ with $\mathbf{h}_v^{(\ell)} \in \mathbf{H}^{(\ell)}$. We generate the output node representations of the last layer (L) via:

$$\mathbf{z}_v = \sigma(\mathbf{W} \cdot \text{MEAN}\{\alpha_{uv}\mathbf{b}_u^{(L)}\}), \forall u \in \mathcal{C}(v) \cup \{v\} \tag{3.6}$$

where σ is the Euclidean normalisation function to reshape values into $[0, 1]$. $\mathbf{Z} \in \mathbb{R}^{n \times d}$ is the final generated node representation with each row vector $\mathbf{z}_v \in \mathbf{Z}$.

IV. Model learning. The proposed HMGNNs could be trained in unsupervised, semi-supervised, or supervised settings. Here, we only discuss the supervised setting used for node classification in our experiments. We define the loss function based on cross entropy, as follows:

$$\mathcal{L} = -\sum_{v \in \mathcal{V}} \mathbf{y}_v^\top \log(\text{Softmax}(\mathbf{z}_v)) \tag{3.7}$$

where $\mathbf{y}_v$ is a one-hot vector denoting the label of node v. We allow $\mathcal{L}$ to be customised for other task-specific objective functions, e.g., the negative log-likelihood loss [VCC+18].

We summarise the process of *hierarchical message-passing graph neural networks* in Algorithm 1. Given a graph $\mathcal{G}$, we first generate the hierarchical structure and combine it with the original graph $\mathcal{G}$, to obtain $\mathcal{H} = \{\mathcal{G}_t \mid t = 1, 2, \ldots, T\}$, where $\mathcal{G}_1 = \mathcal{G}$ (line 2). For each node, including original and generated super nodes, in each NN layer, we perform three primary operations in order: (1) bottom-up propagation (line 6), (2) within-level propagation (line 7), and (3) top-down propagation (line $9-15$). After getting the representation vector of each node that is enhanced with informative long-range interactions and multi-grained semantics, and we train the model with the loss function $\mathcal{L}$ in Equation 3.7.

3.4.2 Hierarchical Community-aware GNN

Identifying hierarchical super nodes for the proposed HMGNNs is the most crucial step as it determines how the information will be propagated within and between

Algorithm 1: *Hierarchical Message-passing Graph Neural Networks*

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$.
Output: Node representations $\mathbf{Z} \in \mathbb{R}^{n \times d}$.

$\mathbf{h}_v^{(0)} \leftarrow \mathbf{x}_v$;
Generate hierarchical structure: $\mathcal{H} = \{\mathcal{G}_t | t = 1, 2, \ldots, T\}$;
for $\ell \leftarrow \{1, 2, \ldots, L\}$ **do**
  $\mathbf{h}_v^{(\ell)} = \text{ReLU}(\sum_{u \in \mathcal{N}(v)} \frac{\mathbf{W}^{(\ell)} \mathbf{h}_u^{(\ell-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} + \frac{\mathbf{W}^{(\ell)} \mathbf{h}_v^{(\ell-1)}}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(v)|}}), \forall v \in \mathcal{G}$;
  for $t \leftarrow 2$ **to** T **do**
    $\mathbf{a}_{s_i^t}^{(\ell)} = \frac{1}{|s_i^t|+1} \left(\sum_{s^{t-1} \in s_i^t} \mathbf{h}_{s^{t-1}}^{(\ell)} + \mathbf{h}_{s_i^t}^{(\ell-1)} \right), \forall s_i^t \in \mathcal{G}_t$;
    $\mathbf{b}_v^{(\ell)} = \text{ReLU}(\sum_{u \in \mathcal{N}(v)} \frac{\mathbf{W}^{(\ell)} \mathbf{a}_u^{(\ell)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} + \frac{\mathbf{W}^{(\ell)} \mathbf{a}_v^{(\ell)}}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(v)|}}), \forall v \in \mathcal{G}_t$;
  end
  for $v \in \mathcal{G}$ **do**
    if $\ell < L$ **then**
      $\mathbf{h}_v^{(\ell)} = \text{ReLU}(\mathbf{W} \cdot \text{MEAN}\{\alpha_{uv} \mathbf{b}_u^{(\ell)}\}), \forall u \in \mathcal{C}(v) \cup \{v\}$;
    else
      $\mathbf{z}_v = \sigma(\mathbf{W} \cdot \text{MEAN}\{\alpha_{uv} \mathbf{b}_u^{(L)}\}), \forall u \in \mathcal{C}(v) \cup \{v\}$;
    end
  end
end

levels. We consider *hierarchical network communities* to construct the hierarchy. The network community has been proved helpful for assisting typical network analysis tasks, including node classification [WPL14, WCW+17] and link prediction [SH12, RGP+15]. Taking the algorithm efficiency into account, we adopt the well-known *Louvain* algorithm [BGLL08] to build the first implementation of HMGNNs, termed as *hierarchical community-aware graph neural network* (HC-GNN). The *Louvain* algorithm returns us a hierarchical structure as described in Section 3.4.1, based on which we can learn node representations involving long-range interactive information and multi-grained semantics.

3.4.3 Theoretical Analysis and Model Comparison

Long-range interactive capability. We now theoretically analyse the asymptotic complexity of different GNN models to capture long-range interaction. We first analyse flat GNN models, that they need to stack $\mathcal{O}(\text{diam}(\mathcal{G}))$ layers to ensure the communication between any pair of nodes in $\mathcal{G}$. For HMGNNs, let us assume that $|\mathcal{V}_{t+1}|/|\mathcal{V}_t| = \lambda$, that λ is the pooling ratio. Thus, the potentially total number of nodes in HMGNNs over $\mathcal{G}$ with n nodes is $\sum_{t=1}^{\infty} n\lambda^t = \mathcal{O}(n)$, while the number of possible levels is $\log_{\lambda^{-1}} n = \mathcal{O}(\log n)$. That said, the shortest path between any two nodes of $\mathcal{G}$ is upper-bounded by $\mathcal{O}(\log n)$. Compared to $\mathcal{O}(\text{diam}(\mathcal{G}))$ with flat GNNs, HMGNNs lead to significant improvement over the capability in capturing long-range interactions.

Model complexity. For the vanilla flat GNN model, i.e., GCN, its computational complexity of one layer is $\mathcal{O}(n^3)$ [KW17], and the computational complexity of a GCN model contains ℓ is $\mathcal{O}(\ell n^3)$. For GAT, except for the same convolutional operation as GCN, the additional masked attention over all nodes requires $\mathcal{O}(\ell n^2)$ computational complexity [VCC+18]. Thus, overall it takes $\mathcal{O}(\ell(n^3 + n^2))$ complexity. For the hierarchical representation model, g-U-Net, its computational complexity of one hierarchy is $\mathcal{O}(2\ell n^3)$ [GJ19], because its unpooling operation introduces another $\mathcal{O}(\ell n^3)$ complexity, in addition to the convolutional operations as GCN. Thus the complexity of g-U-Net with T levels is $\sum_{t=1}^{T} 2\ell(n\lambda^{t-1})^3 = \mathcal{O}(2\ell n^3)$, since the pooled graphs are supposed have much smaller number of nodes than $\mathcal{G}$. For HC-GNN, take GCN as an example GNN encoder and the *Louvain* algorithm as an example hierarchical structure construction method, which has optimal $O(n \log c)$ computational complexity [Tra15], where c is the average degree. The top-down propagation allows each node of $\mathcal{G}$ to receive T different messages from T levels with different weights, this introduces $\mathcal{O}(Tn)$ computational complexity, where T is the number of levels, and we assume $T \ll n$. Altogether, the complexity of HC-GNN is $\sum_{t=1}^{T} \ell(n\lambda^{t-1})^3 + \mathcal{O}(n \log c + Tn) = \mathcal{O}(\ell n^3 + n \log c + Tn)$, which is more efficient than GAT and g-U-Net.

3.5 Experiments

We conduct extensive experiments to answer 6 research questions (RQ):

- **RQ1:** How does HC-GNN performs *vs.* state-of-the-art methods for node classification (**RQ1-1**), community detection **(RQ1-2)**, and link prediction (**RQ1-3**)?
- **RQ2:** Can HC-GNN leads to satisfying performance under settings of transductive, inductive, and few-shot learning?
- **RQ3:** How do different levels in the hierarchical structure contribute to the effectiveness of node representations?
- **RQ4:** How do various hierarchical structure generation methods affect the performance of HC-GNN?
- **RQ5:** Does HC-GNN survive from low sparsity of graphs?
- **RQ6:** Does HC-GNN available with different encoders?

3.5.1 Evaluation Setup

Datasets. We perform experiments on both synthetic and real-world datasets. For the link prediction task, we adopt 3 datasets:

- Grid [YYL19]. A synthetic 2D grid graph representing a 20×20 grid with $|\mathcal{V}| = 400$ and no node features.

Table 3.3: Summary of dataset statistics. LP: Link Prediction, NC: Node Classification, CD: Community Detection, N.A. means a dataset does not contain node features or node labels.

Dataset	Task	#Nodes	#Edges	#Features	#Classes
Grid	LP	400	760	N.A.	N.A.
Cora	LP&NC	2,708	5,278	1,433	7
Power	LP	4,941	6,594	N.A.	N.A.
Citeseer	NC	3,312	4,660	3,703	6
Pubmed	NC	19,717	44,327	500	3
Emails	CD	799	10,182	N.A.	18
PPI	NC	56,658	818,435	50	121
Protein	NC	42,576	79,482	29	3
Ogbn-arxiv	NC	169,343	1,166,243	128	40

- Cora [SNB+08]. A citation network consists of 2,708 scientific publications and 5,429 links. A 1,433 dimensional word vector describes each publication as a node feature.
- Power [WS98]. An electrical grid of western US with 4,941 nodes and 6,594 edges and no node features.

For node classification, we use 6 datasets: including Cora, Citeseer [KW17] and Pubmed [KW17] and a large-scale benchmark dataset Ogbn-arxiv [HFZ+20] for transductive settings, and 2 protein interaction networks Protein and PPI [YYM+18] for inductive settings.

- Cora. The same above-mentioned Cora dataset contains 7 classes of nodes. A citation network consists of 3,312 scientific publications classified into one of 6 classes, and the dataset contains 4,660 edges. Each node is labelled with the class it belongs to.
- Citeseer [SNB+08]. Each node comes with 3,703-dimensional node features.
- Pubmed [NLGH12]. A dataset consists of 19,717 scientific publications from PubMed database about diabetes classified into one of 3 classes. Each node is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words.
- PPI [ZL17]. 24 protein-protein interaction networks and nodes of each graph comes with 50 dimensional feature vector.
- Protein [BOS+05]. 1113 protein graphs and nodes of each graph comes with 29 dimensional feature vector. Each node is labelled with a functional role of the protein.
- Ogbn-arxiv [HFZ+20]. A large-scale citation graph between 169,343 computer science arXiv papers. Each node is an arXiv paper, and each directed edge indicates that one paper cites another one. Each paper comes with a 128-dimensional feature vector obtained by averaging the embeddings of words in its title and abstract. The task is to predict the 40 subject areas of these papers.

For node community detection, we use an email communication dataset:

- Emails [LK14]. 7 real-world email communication graphs from SNAP with no node features. Each graph has 6 communities, and each node is labelled with the community it belongs to.

The data statistics of datasets is summarised in Table ,3.3 and they are available for download with our published code.

Experimental settings. We evaluate HC-GNN under the settings of transductive and inductive learning. For node classification, we additionally conduct experiments with the few-shot setting.

- *Transductive learning.* For link prediction, we follow the experimental settings of [YYL19] to use 10% existing links and an equal number of non-existent links as validation and test sets. The remaining 80% existing links and a dual number of non-existent links are used as the training set. For node classification, we follow the semi-supervised settings of [KW17]: if there are enough nodes, for each class, we randomly sample 20 nodes for training, 500 nodes for validation, and 1000 nodes for testing. For the Emails dataset, we follow the supervised learning settings of [HSLH19] to randomly select 80% nodes as the training set, and use the two halves of remaining as the validation and test set, respectively. We report the test performance when the best validation performance is achieved.
- *Inductive learning.* This aims at examining a model's ability to transfer the learned knowledge from existing nodes to future ones that are newly connected to existing nodes in a graph. Hence, we hide the validation and testing graphs during training. We conduct the experiments for inductive learning using PPI and Protein datasets. We train models on 80% graphs to learn an embedding function f and apply it on the remaining 20% graphs to generate the representation of new-coming nodes.
- *Few-shot learning.* Since the cost of collecting massive labelled datasets is high, having a few-shot learning model would be pretty valuable for practical applications. Few-shot learning can also be considered as an indicator to evaluate the robustness of a deep learning model. We perform few-shot node classification, in which only 5 samples of each class are used for training. The sampling strategies for testing and validation sets follow those in transductive learning.

Evaluation metrics. We adopt AUC to measure the performance of link prediction. For node classification, we use micro- and macro-average F1 scores and accuracy. NMI score is utilised for community detection evaluation.

Competing methods. To validate the effectiveness of HC-GNN, we compare it with 9 competing methods which include 6 flat message-passing GNN models, 2 hierarchical GNN models and another state-of-the-art model.

- GCN[1] [KW17] is the first deep learning model which generalises the convolutional operation on graph data and introduces the semi-supervised train paradigm.
- GraphSAGE[2] [HYL17b] extends the convolutional operation of GCN to mean/ max/ LSTM convolutions and introduces a sampling strategy before employing convolutional operations on neighbour nodes.
- GAT[3] [VCC+18] employs trainable attention weight during message aggregation from neighbours, which makes the information received by each node different and provide interpretable results.
- GIN[4] [XHLJ19] summarises previous existing GNN layers as two components, AGGREGATE and COMBINE, and models injective multiset functions for the neighbour aggregation.
- HARP[5] [CPHS18] is a hierarchical structure by various collapsing methods for unsupervised node representation learning.
- P-GNNs[6] [YYL19] introduce anchor-set sampling to generate node representation with global position-aware.
- g-U-Net[7] [GJ19] generalises the U-nets architecture of convolutional neural networks for graph data to get better node representation. It constructs a hierarchical structure with the help of pooling and unpooling operators.
- GraphRNA[8] [HSLH19] proposes using recurrent neural networks to capture the long-range node dependencies to assist GNN to obtain better node representation.
- GCNII[9] [CWH+20] simplifies the aggregation design of flat GNNs and joined with well-designed normalisation units to get much deeper GNN models.

Reproducibility. For fair comparison, all methods adopt the same representation dimension ($d = 32$), learning rate ($= 1e-3$), Adam optimiser and the number of iterations ($= 200$) with early stop (50). In terms of the neural network layers, we report the one with better performance of GCNII with better performance among $\{8, 16, 32, 64, 128\}$; for other models, we report the one with better performance between $2-4$; For all models with hierarchical structure (including g-U-Net and HC-GNN), we use GCN as the default GNN encoder for fair comparision. Note that for the strong competitor, P-GNNs, since its representation dimension is related to the number of nodes in a graph, we add a linear regression layer at the end of

[1] https://github.com/tkipf/pygcn
[2] https://github.com/williamleif/GraphSAGE
[3] https://github.com/PetarV-/GAT
[4] https://github.com/weihua916/powerful-gnns
[5] https://github.com/GTmac/HARP
[6] https://github.com/JiaxuanYou/P-GNN
[7] https://github.com/HongyangGao/Graph-U-Nets
[8] https://github.com/xhuang31/GraphRNA_KDD19
[9] https://github.com/chennnM/GCNII

P-GNNs for node classification tasks to ensure its end-to-end structure is the same as other models [HSLH19]. For HC-GNN, the number of HC-GNN layers is varied and denoted as 1L, 2L or 3L. In Section 3.5.3, HC-GNN adopts the number of layers leading to the best performance for model analysis i.e., 2L for the Cora dataset, 1L for the Citeseer and Pubmed datasets. For *Louvain* community detection, we use the implementation of a given package[10], which does not require any hyper-parameters. We use PyTorch Geometric to implement all models mentioned in this chapter. More details are referred to our code file[11]. The experiments are repeated 10 times, and average results are reported. Note that we use only node features with unique one-hot identifiers to differentiate different nodes if there are no given node features from the datasets and use the original node features if they are available. We employ Pytorch and PyTorch Geometric to implement all models. Experiments were conducted with GPU (NVIDIA Tesla V100) machines.

3.5.2 Experimental Results

Table 3.4: Results in Micro-F1 and Macro-F1 for transductive semi-supervised node classification task. Results in Acc for node classification of Ogbn-arxiv follows the default settings of OGB dataset [HFZ+20], and results in NMI for community detection (i.e., on the Emails data in the last column). Standard deviation errors are given. ‡ indicates the results from OGB leaderboard [HFZ+20]. OOM: out-of-memory.

	Cora		**Citeseer**		**Pubmed**		**Emails**	**Ogbn-arxiv**
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	NMI	Acc (%)
GCN	0.802±0.019	0.786±0.020	0.648±0.019	0.612±0.012	0.779±0.027	0.777±0.026	0.944±0.010	71.74±0.29‡
GraphSAGE	0.805±0.013	0.792±0.009	0.650±0.027	0.611±0.020	0.768±0.031	0.763±0.030	0.925±0.014	71.49±0.27‡
GAT	0.772±0.019	0.761±0.023	0.620±0.024	0.594±0.015	0.775±0.036	0.770±0.022	0.947±0.009	72.06±0.31‡
GIN	0.762±0.020	0.759±0.018	0.615±0.023	0.591±0.020	0.744±0.036	0.733±0.041	0.640±0.047	71.76±0.33‡
P-GNNs	0.438±0.044	0.431±0.040	0.331±0.019	0.314±0.018	0.558±0.033	0.551±0.036	0.598±0.020	OOM
GCNII	0.823±0.017	0.801±0.022	0.722±0.011	0.677±0.010	0.791±0.009	0.790±0.016	0.947±0.010	72.74±0.16
HARP	0.363±0.020	0.350±0.021	0.343±0.023	0.317±0.017	0.441±0.024	0.329±0.019	0.371±0.014	OOM
GraphRNA	0.354±0.070	0.244±0.040	0.352±0.050	0.259±0.047	0.476±0.054	0.355±0.089	0.434±0.047	OOM
g-U-Net	0.805±0.017	0.796±0.018	0.673±0.015	0.628±0.012	0.782±0.018	0.781±0.019	0.939±0.015	71.78±0.37
HC-GNN-1L	0.819±0.002	**0.816**±0.005	**0.728**±0.005	**0.686**±0.003	**0.812**±0.009	**0.806**±0.009	**0.961**±0.005	72.69±0.25
HC-GNN-2L	**0.834**±0.007	**0.816**±0.006	0.696±0.002	0.652±0.006	**0.809**±0.004	**0.804**±0.005	**0.962**±0.005	**72.79**±0.31
HC-GNN-3L	0.813±0.008	**0.806**±0.006	0.686±0.006	0.633±0.008	**0.804**±0.004	0.780±0.020	0.935±0.014	72.58±0.27

Transductive node classification (RQ1-1&RQ2). We present the results of transductive node classification in Table 3.4. We can see that HC-GNN consistently outperforms all of the competing methods in the 5 datasets, and even the shallow HC-GNN model with only one layer may lead to better results. We think the outstanding performance of HC-GNN results from two aspects: (a) the hierarchical structure allows the model to capture informative long-range interactions of graphs, i.e., propagating messages from and to distant nodes in the graph; and (b) the meso- and macro-level semantics reflected by the hierarchy is encoded through bottom-up, within-level, and top-down propagations. On the other hand, P-GNNs, HARP, and GraphRNA perform worse in semi-supervised node classification. The possible reason is they need more training samples, such as using 80% of existing

[10] `https://python-louvain.readthedocs.io/en/latest/api.html`

[11] Code and data are available at `https://github.com/zhiqiangzhongddu/HC-GNN`

nodes as the training set, as described in their papers [YYL19, HSLH19], but we have only 20 nodes for training in the semi-supervised setting.

Table 3.5: Micro-F1 results for inductive node classification. Standard deviation errors are given.

	PPI	**Protein**
GCN	0.444±0.004	0.542±0.018
GraphSAGE	0.409±0.014	<u>0.637</u>±0.018
GAT	0.469±0.062	0.608±0.077
GIN	<u>0.571</u>±0.008	0.631±0.016
GCNII	0.507±0.008	0.614±0.011
g-U-Net	0.433±0.012	0.547±0.011
HC-GNN-1L	0.48±0.091	**0.638**±0.027
HC-GNN-2L	**0.584**±0.087	0.622±0.031
HC-GNN-3L	**0.584**±0.002	0.582±0.025

Inductive node classification (RQ1-1&RQ2). The results are reported in Table 3.5[12]. We can find that HC-GNN is still able to show some performance improvement over existing GNN models. But the improvement gain is not so significant and inconsistent in different layers of HC-GNN compared to the results in transductive learning. The possible reason is that different graphs may have other hierarchical community structures. Nevertheless, the results lead to one observation: the effect of transferring hierarchical semantics between graphs for inductive node classification is somewhat limited. Therefore, exploring an ameliorated model that can adaptively exploit hierarchical structure for different graphs for different tasks would be interesting. We further discuss it in Section 3.6 as one concluding remark.

Table 3.6: Micro-F1 results for few-shot node classification. Standard deviation errors are given.

	Cora	**Citeseer**	**Pubmed**
GCN	0.695±0.049	0.561±0.054	0.699±0.059
GraphSAGE	<u>0.719</u>±0.024	0.559±0.049	0.707±0.051
GAT	0.630±0.030	0.520±0.054	0.664±0.046
GIN	0.691±0.038	0.509±0.060	0.714±0.036
P-GNNs	0.316±0.040	0.332±0.011	0.547±0.037
GCNII	0.701±0.022	0.564±0.015	<u>0.717</u>±0.047
HARP	0.224±0.033	0.260±0.035	0.415±0.039
GraphRNA	0.274±0.063	0.206±0.019	0.429±0.042
g-U-Net	0.706±0.054	<u>0.567</u>±0.044	0.693±0.036
HC-GNN-1L	0.681±0.023	**0.639**±0.019	0.704±0.043
HC-GNN-2L	**0.759**±0.015	**0.660**±0.024	**0.724**±0.052
HC-GNN-3L	**0.752**±0.017	**0.642**±0.016	**0.742**±0.045

Few-shot node classification (RQ1-1&RQ2). We exhibit the results in Table 3.6. HC-GNN demonstrates better performance in few-shot learning than all competing

[12]Since HARP, P-GNNs and GraphRNA cannot be applied in the inductive setting, we do not present their results in Table 3.5.

methods across 3 datasets. Such results indicate that the hierarchical message passing is able to transfer supervised information through inter- and intra-level propagations. In addition, the hierarchical message-passing pipeline further enlarges the influence range of supervision information from a small number of training samples. With effective and efficient pathways to broadcast information, HC-GNN is proven to be quite promising in few-shot learning.

Community detection (RQ1-2). The community detection results conducted on the Emails dataset are also shown in Table 3.4. It can be seen that HC-GNN again outperforms all competing methods. We believe this is because the communities identified by Louvain are further exploited by learning their hierarchical interactions in HC-GNN. In other words, HC-GNN is able to reinforce the intra- and inter-community effect and encode it into node representations.

Table 3.7: Results in AUC for link prediction. Cora-Feat means node features are used in the Cora dataset, and conversely, Cora-NoFeat means node features are not used. Standard deviation errors are given.

	Grid	**Cora-Feat**	**Cora-NoFeat**	**Power**
GCN	0.763±0.036	0.869±0.006	0.785±0.007	0.624±0.013
GraphSAGE	0.775±0.018	0.870±0.006	0.741±0.017	0.569±0.012
GAT	0.782±0.028	0.874±0.010	0.789±0.012	0.621±0.013
GIN	0.756±0.025	0.862±0.009	0.782±0.010	0.620±0.011
P-GNNs	<u>0.867</u>±0.034	0.818±0.013	<u>0.792</u>±0.012	<u>0.704</u>±0.006
GCNII	0.807±0.024	0.889±0.019	0.770±0.011	0.695±0.014
HARP	0.687±0.021	0.837±0.033	0.721±0.017	0.529±0.004
g-U-Net	0.701±0.032	<u>0.909</u>±0.006	0.772±0.007	0.628±0.024
HC-GNN-1L	0.823±0.035	0.884±0.006	**0.795**±0.012	0.682±0.016
HC-GNN-2L	**0.913**±0.011	0.895±0.007	**0.837**±0.006	**0.767**±0.020
HC-GNN-3L	**0.914**±0.011	0.891±0.007	**0.839**±0.004	**0.784**±0.017

Link prediction (RQ1-3). Here, we motivate our idea by considering pairwise relation prediction between nodes. Suppose a pair of nodes u, v are labelled with label y, and our goal is to predict y for unseen pairs. From the perspective of representation learning, we can solve the problem via learning an embedding function f that computes the node representation $\mathbf{z}_v$, where the objective is to maximise the likelihood of distribution $p(y|\mathbf{z}_u, \mathbf{z}_v)$. The results in Table 3.7 indicate that the HC-GNN leads to competitive performance compared to all competing methods, with up to 11.4% AUC improvement, demonstrating its effectiveness on link prediction tasks. To examine whether HC-GNN cannot better work on link prediction without node features, we conduct the same experiment on Cora without using node features (i.e., Cora-NoAtt), and we find HC-GNN leads to the best results. Because HC-GNN can better model graph topology and hierarchical semantics to capture the underlying relation between nodes.

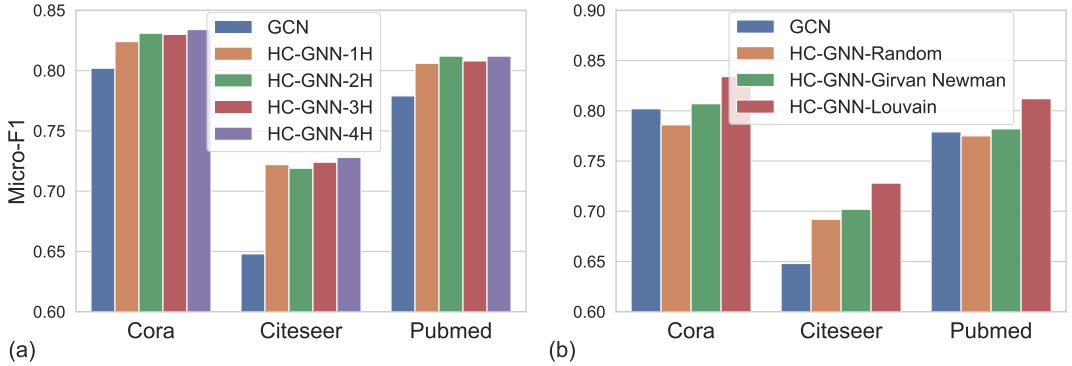


Figure 3.3: Results in Micro-F1 for semi-supervised node classification using HC-GNN by varying: (a) the number of hierarchy levels adopted for message passing, and (b) the approaches to generate the hierarchical structure.

3.5.3 Empirical Model Analysis

Contribution of different levels (RQ3). Since HC-GNN highly relies on the generated hierarchical structure, we aim to examine how different levels in the hierarchy contribute to the prediction. We report the transductive semi-supervised node classification performance by varying the number of levels (from 1 to 4). GCN is also selected for comparison because it considers no hierarchy, i.e., only within-level propagation in the original graph. The results are shown in Figure 3.3(a), in which 1H and 2H indicate only the first level and the first 2 levels are adopted, respectively. We can find that HC-GNN using more levels for hierarchy construction lead to better results. The flat message passing of GCN cannot work well. Such results provide strong evidence that GNNs can significantly benefit from the hierarchical message-passing mechanism. In addition, more hierarchical semantics can be encoded if more levels are adopted.

Influence of hierarchy generation approaches (RQ4). HC-GNN implements the proposed *hierarchical message-passing graph neural networks* based on the *Louvain* community detection algorithm, that is termed HC-GNN-*Louvain* in this paragraph. We aim to validate (A) whether the community information truly benefits the classification tasks, and (B) how different approaches to generate the hierarchical structure affect the performance. To answer (A), we construct a random hierarchical structure to generate randomised HC-GNN, termed HC-GNN-Random, in which *Louvain* detects hierarchical communities, and nodes are randomly swapped among the same-level communities. In other words, the hierarchy structure is maintained, but community memberships are perturbed. The results on semi-supervised node classification are exhibited in Figure 3.3(b). We can see that HC-GNN-Random works worse than GCN in Cora and Pudmed, and much worse than HC-GNN-*Louvain*. It implies that hierarchical communities generated from the graph topology genuinely lead to a positive effect on information propagation. To answer (B), we utilise *Girvan Newman* [GN02] to produce the hierarchical structure by following the same way described in Section 3.4.1, and have a model named HC-GNN-*Girvan Newman*. The results are shown in Figure 3.3(b). Although HC-GNN-*Girvan Newman* is not as effective as HC-GNN-*Louvain*, they still outperform GCN. Such

a result indicates that the approaches to generate the hierarchical structure will influence the capability of HC-GNN. While HC-GNN-*Louvain* leads to promising performance, one can search for a proper hierarchical community detection method to perform better on different tasks.

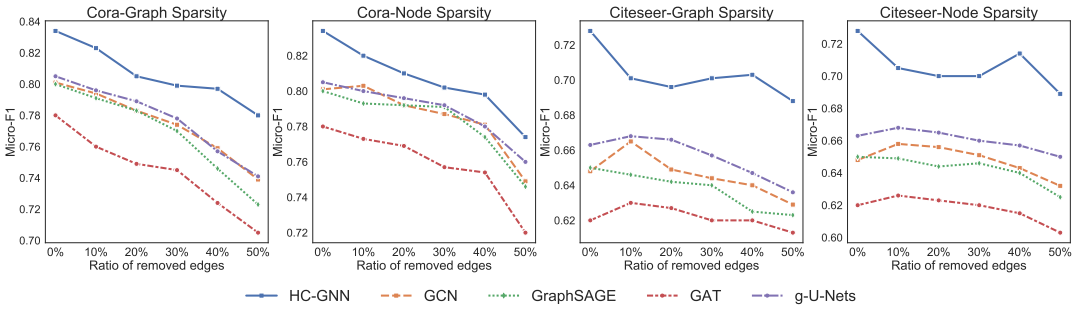


Figure 3.4: Results on semi-supervised node classification in graphs by varying the percentage of removed edges.

Influence of graph sparsity (RQ5). Since community detection algorithms are sensitive to the sparsity of the graph [NN12], we aim at studying how HC-GNN perform under graphs with low sparsity values in the task of semi-supervised node classification. We consider two kinds of sparsity. One is graph sparsity by randomly removing a percentage of edges from all edges in the graph, i.e., $10\% - 50\%$. The other is node sparsity by randomly drawing a portion of edges incident to every node in the graph. The random removal of edges can be considered that users hide partial connections due to privacy concerns. The results for Cora and Citeseer are presented in Figure 3.4. HC-GNN significantly outperforms the competing methods on graph sparsity and node sparsity under different edge-removal percentages. Such results prove that even though communities are subject to sparse graphs, our HC-GNN are more robust than other GNN models.

Table 3.8: Comparison of HC-GNN with different primary GNN encoders (*within-level propagation*), follow the transductive node classification settings. Reported results in Micro-F1.

Models	**Cora**	**Citeseer**	**Pubmed**
GCN	0.802	0.648	0.779
HC-GNN w/ GCN	**0.834**	**0.728**	**0.812**
GAT	0.772	0.629	0.775
HC-GNN w/ GAT	**0.801**	**0.712**	**0.819**
GCNII	0.823	0.722	0.791
HC-GNN w/ GCNII	**0.841**	**0.734**	**0.816**

Ablation study of different primary GNN encoders (RQ6). We adopted GCN as the default primary GNN encoder in model presentation (Section 3.4) and previous experiments. Here, we present more experimental results by endowing HC-GNN with advanced GNN encoders in Table 3.8. The table demonstrates that advanced GNN encoders can still benefit from the multi-grained semantics of HC-GNN. For instance, GCNII can stack lots of layers to capture long-range information; however, it still follows a *flat* message-passing mechanism hence naturally ignoring the

multi-grained semantics. HC-GNN further ameliorates this problem for better performance.

3.6 Conclusion and Future work

This chapter has presented a novel *hierarchical message-passing graph neural networks* (HMGNNs) framework, which deals with two critical deficiencies of the *flat* message passing mechanism in existing GNN models, i.e., the limited ability for information aggregation over long-range and infeasible in encoding meso- and macro-level graph semantic information. Following this innovative idea, we further presented the first implementation, *hierarchical community-aware graph neural network* (HC-GNN), with the assistance of a hierarchical communities detection algorithm. The theoretical analysis confirms HC-GNN's significant ability in capturing long-range interactions without introducing heavy computation complexity. Extensive experiments conducted on 9 graph datasets show that HC-GNN can consistently outperform state-of-the-art GNN models in 3 tasks, including node classification, link prediction, and community detection, under settings of transductive, inductive, and few-shot learning. Furthermore, the proposed hierarchical message-passing GNN provides model flexibility. For instance, it friendly allows different choices and customised designs of the hierarchical structure, and it incorporates well with advanced flat GNN encoders to obtain more impressive results. That said, the HMGNNs could be easily applied to work as a general practical framework to boost downstream tasks with arbitrary hierarchical structure and encoder.

The proposed hierarchical message-passing GNNs provide a good starting point for exploiting graph hierarchy with GNN models. In the future, we aim to incorporate the learning of the hierarchical structure into the model optimisation of GNNs such that a better hierarchy can be searched on the fly. Moreover, it is also interesting to extend our framework for heterogeneous networks.

Chapter 4

Multi-grained Semantics-aware Graph Neural Networks

4.1 Introduction

Having explored the problem of *flat* message-passing mechanism of *graph neural networks* (GNNs) and proposed the *hierarchical message-passing graph neural networks* (HMGNNs) framework and the practical implementation model, *hierarchical community-aware graph neural network* (HC-GNN), in the previous chapter. And as shown in the experimental results on inductive node classification, HC-GNN performs less effective than on transductive settings. We argue it is because HC-GNN's hierarchical structure is pre-defined by the hierarchical community detection algorithm and the effect of transferring hierarchical semantics between graphs for inductive setting is somewhat limited. Therefore, we now turn to explore an ameliorated model that can adaptively exploit hierarchical structure for different graphs for different tasks.

In addition, in the midst of investigation, we also found that most existing graph machine learning models solve either the node-wise task or the graph-wise task independently while they are inherently correlated. That said, node representations form graph representation, and graph representation can provide node representations with meso/macro-level semantic information in the graph. Joint modelling with node- and graph-wise tasks allows GNNs to overcome the limitation of *flat* propagation mode in capturing multi-grained semantics, and the enriched node representation could further ameliorate the graph representation. However, to the best of our knowledge, none of the existing work simultaneously exploit node- and graph-wise tasks, along with capturing multi-grained semantics hidden in the graph, to learn representations of nodes and the graph.

In this chapter, we introduce a novel model, *adaptive multi-grained graph neural networks* (AdamGNN), that follow the HMGNNs framework. It integrates graph convolution, adaptive pooling and unpooling operations into one framework to adaptively generate the hierarchical structure and generate both node and graph level representations interactively. Unlike the above-mentioned GNN models, we treat node and graph representation learning tasks in a unified framework so that they can collectively optimise each other during training. In modelling multi-grained

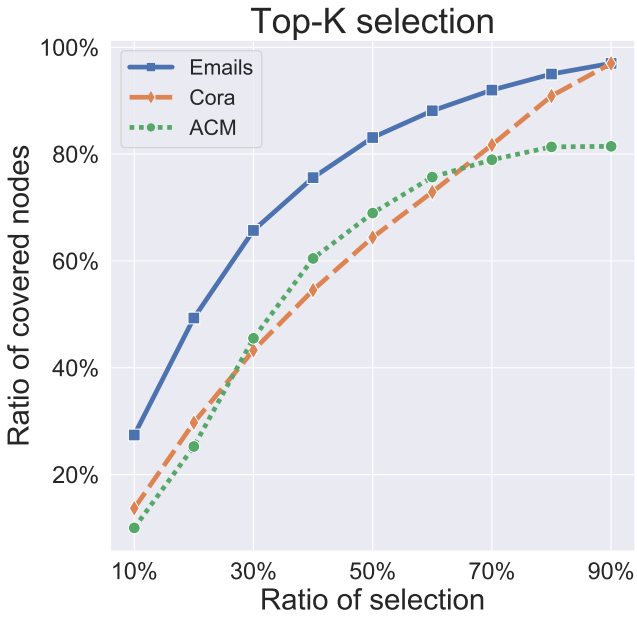


Figure 4.1: Ratio of covered nodes with various selection ratios.

semantics, the adaptive pooling and unpooling operators preserve the important node features and hierarchical structural features.

More concretely, as shown in Figure 4.2-(a), we employ *(i)* an adaptive graph pooling (AGP) operators to generate a multi-grained structure based on the derived primary node representations by a GNN layer, *(ii)* graph unpooling (GUP) operators to further distribute the explored meso- and macro-level semantics to the corresponding nodes of the original graph, and *(iii)* a *flyback* mechanism to integrate all received multi-grained semantic messages as the evolved node representations. Besides, the attention-enhanced *flyback* aggregator provides a reasonable explanation of the importance of messages from different grains. Experimental results reveal the effectiveness of AdamGNN, and the ablation and empirical studies confirm the effectiveness and flexibility of different components in AdamGNN. At last, through case studies, AdamGNN is shown to highlight variant-range node interactions in different graph datasets.

4.2 Additional Related Work

Graph pooling. Pooling operation overcomes GNN's weakness in generating graph-level representation by recursively merging sets of nodes to form super nodes in the pooled graph. DiffPool [YYM+18] is a differentiable pooling operator, which learns a soft assign matrix that maps each node to a set of clusters, treated as super nodes. Since this assignment is relatively *dense* that incurs high computation cost, it is not scalable for large graphs [CVJ+18]. Following this direction, a Top-k selection based pooling layer (g-U-Net) is proposed to select important nodes from the original graph to build a pooled graph [GJ19]. SagPool [LLK19] and ASAP [RST20] further use attention and self-attention for cluster assignment. They address the

Table 4.1: Model comparison from various aspects: Node-wise Task (NT), Graph-wise Task (GT), Pooling and/or Unpooling (P/U), Adaptive Pooling (AP), Efficient Pooling (EP), Multi-grained Explanation (ME).

	NT	GT	P/U	AP	EP	ME
GCN [KW17]	√					
GraphSAGE [HYL17b]	√					
GAT [VCC+18]	√					
GIN [XHLJ19]	√	√				
PNA [CCB+20]	√	√				
GCNII [CWH+20]	√					
GRAND [FZD+20]	√					
DiffPool [YYM+18]		√	P			
g-U-Net [GJ19]	√	√	P, U		√	
SagPool [LLK19]		√	P		√	
EigenPool [MWAT19]		√	P	√	√	
StructPool [YJ20]		√	P			
ASAP [RST20]		√	P		√	
AdamGNN	√	√	P, U	√	√	√

problem of sparsity in DiffPool, however, such a manual-defined hyper-parameter k is quite sensitive to the final performance [GJ19], thus limiting the adaptivity of these models on graphs of different sizes. In addition, as shown in Figure 4.1, different k values significantly affect the number of covered nodes in the graph, which means the important node features could get lost during the trivial pooling operation. Note that nodes covered by a super node refer to nodes involved in the super node's aggregation tree.

Recently, EigenPool [MWAT19] proposes a pooling operator based on graph Fourier, which does not rely on the Top-k selection strategy, StructPool [YJ20] designs strategies to involve both node and graph structures, and includes conditional random fields technique to ameliorate the cluster assignment. However, StructPool treats the graph assignment as a *dense* clustering problem, which gives rise to a high computation complexity as in DiffPool.

Discussion. Table 4.1 summarises the key advantages of the proposed AdamGNN and compares it with a number of state-of-the-art methods. Among the existing GNN models, g-U-Net and AdamGNN support both node- and graph-level tasks. However, *(i)* the Top-k selection strategy of g-U-Net introduces a new hyper-parameter and may lose important node features or graph structure; *(ii)* g-U-Net generates super graph only with k selected super nodes, which ruins multi-grained semantics of the original graph; *(iii)* g-U-Net does not support mini-batch because it needs to compute scores of all nodes in one big batch to select super nodes. On the contrary, AdamGNN is a unified framework that adaptively integrates multi-grained semantics into node representations and achieves a mutual optimisation between node- and graph-wise tasks. Besides, AdamGNN supports efficient mini-batch pooling and unpooling, and also provides model explanation via the multi-grained semantics. Therefore, we believe AdamGNN's framework is more general, effective and scalable than G-U-Net.

4.3 Proposed Approach

4.3.1 Preliminaries

Problem setup. For node-wise tasks, the goal is to learn a mapping function $f_n : \mathcal{G} \rightarrow \mathbf{Z}$, where $\mathbf{Z} \in \mathbb{R}^d$, and each row $\mathbf{z}_i \in \mathbf{Z}$ corresponds to node v_i's representation. For graph-wise tasks, similarly it aims to learn a mapping $f_g : \mathcal{D} \rightarrow \mathbf{Z}$, where $\mathcal{D} = \{\mathcal{G}_1, \mathcal{G}_2, \dots\}$ is a set of graphs, each row $\mathbf{z}_i \in \mathbf{Z}$ corresponds to the graph $\mathcal{G}_i$'s representation. The mapping function's effectiveness f_n and f_g is evaluated by applying $\mathbf{Z}$ to different downstream tasks. Table 4.2 lists the additional mathematical notation used in this chapter.

Table 4.2: Summary of additional notations.

Notation	Description
T	Number of hierarchy level
$\mathcal{G}_t$	Super graph at level t
$\mathcal{D}$	Set of generated super graphs
$\mathcal{C}_\lambda$	List of generated ego-networks from $\mathcal{G}$
$\mathcal{N}_i^\lambda$	Set of nodes in the ego-network of node v_i
$\mathbf{S}_t$	Assignment matrix at level t

Primary node representation. We use *graph convolution network* (GCN) [KW17] as an example primary GNN encoder to obtain the node representation and GCN (Equation 2.3) can be formally represented as:

$$\mathbf{H}^{(\ell+1)} = \text{ReLU}(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}) \tag{4.1}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\hat{\mathbf{D}} = \sum_j \hat{\mathbf{A}}_{ij}$ and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n \times d}$ is a trainable weight matrix for layer ℓ. $\mathbf{H}^{(\ell)}$ is the generated node representation of layer ℓ which is defined as the primary node representations $\mathbf{H} = \mathbf{H}^{(\ell)}$.

Node representations are generated based on each target node's local neighbours, which are aggregated via learning based on the adjacency matrix $\mathbf{A}$. GCN cannot capture meso/macro-level knowledge, even with stacking multiple layers. Hence we term such generated node representations as primary node representations.

4.3.2 Adaptive Graph Pooling for Multi-grained Structure Generation

The proposed model, AdamGNN, adaptively generates a multi-grained structure to realise the collective optimisation between the node and graph level tasks within one unified framework. The key idea is that apply an adaptive graph pooling operator to present the multi-grained semantics of $\mathcal{G}$ explicitly and improve the node representation generation with the derived meso/macro information. While AdamGNN is usually performed under multiple levels of granularity (T different grains), in this section, we present how level t's super graph is adaptively constructed based on graph of level $t-1$, i.e., $\mathcal{G}_{t-1} = (\mathcal{V}_{t-1}, \mathcal{E}_{t-1}, \mathbf{X}_{t-1})$.

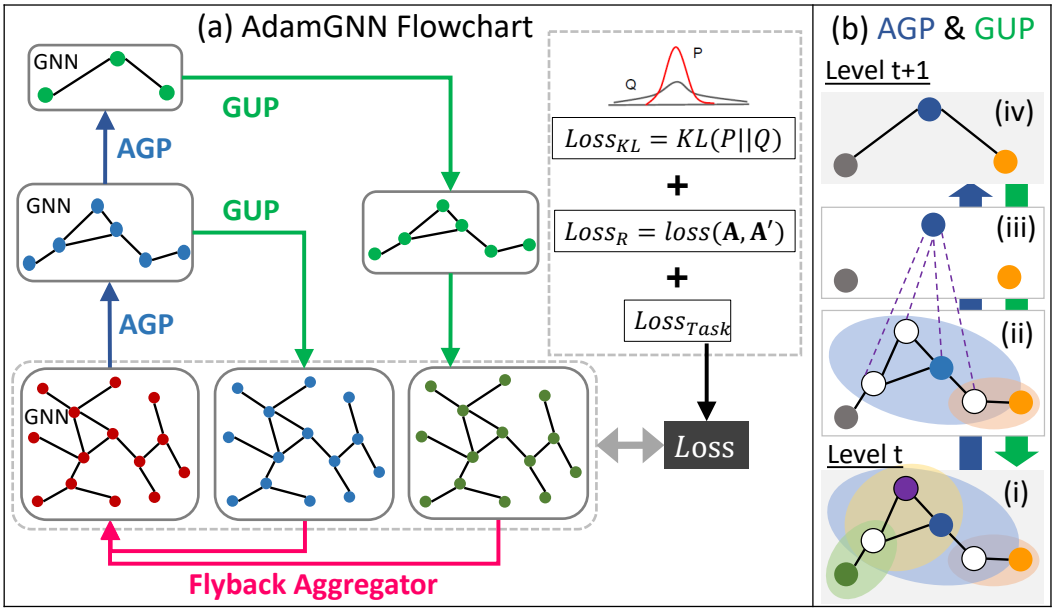


Figure 4.2: (a) An illustration of AdamGNN with 3 levels. AGP: adaptive graph pooling, GUP: graph unpooling. (b) An example of performing adaptive graph pooling on a graph: (i) ego-network formation, (ii-iii) super node generation, (iv) maintaining super graph connectivity.

Ego-network formation. We initially consider the graph pooling as an ego-network selection problem, i.e., each ego node can determine whether to aggregate its local neighbours to form a super node, resolving the *dense* issue of DiffPool by avoiding using a dense assignment matrix. As shown in Figure 4.2-(b)-(i), each ego-network c_λ contains the ego and its local neighbours $\mathcal{N}_i^\lambda$ within λ-hops., i.e., $\mathcal{N}_i^\lambda = \{v_j \mid d(v_i, v_j) \leq \lambda\}$, where $d(v_i, v_j)$ means the shortest-path length between v_i and v_j. Thus an ego-network can be formally presented as: $c_\lambda(v_i) = \{v_j \mid \forall v_j \in \mathcal{N}_i^\lambda\}$, and a set of ego-networks $\mathcal{C}_\lambda = \{c_\lambda(v_1), \ldots, c_\lambda(v_n)\}$ can be generated from $\mathcal{G}$. We investigate the impact of the ego-network size in the ablation studies of Section 4.4.2

Super node determination. Given $\mathcal{G}$ with n nodes has n ego-networks, forming a super graph with all ego-networks blurs the useful multi-grained semantics and lead to a high computation cost. We select a fraction of ego-networks from $\mathcal{C}_\lambda$ to organise the multi-grained semantics of $\mathcal{G}$. We make the selection based on a closeness score ϕ_i that evaluates the representativeness of the ego v_i to its local neighbours $v_j \in c_\lambda(v_i)$. We first create a function to calculate the closeness score ϕ_{ij} between v_i and v_j:

$$\begin{aligned} \phi_{ij} &= f_\phi^{non}(v_i, v_j) \times f_\phi^{lin}(v_i, v_j) \\ &= \text{Softmax}(\overrightarrow{\mathbf{a}}^{\mathrm{T}} \sigma(\mathbf{WH}[j] \parallel \mathbf{WH}[i])) \times \text{Sigmoid}(\mathbf{H}^{\mathrm{T}}[j] \cdot \mathbf{H}[i]) \end{aligned} \tag{4.2}$$

where $\overrightarrow{\mathbf{a}} \in \mathbb{R}^{2\pi}$ is the weight vector, $\parallel$ is the concatenation operator and σ is an activation function (LeakReLU) to avoid the vanishing gradient problem [Hoc98] during the model training process. $f_\phi^{non}(v_i, v_j) = \frac{\exp(\overrightarrow{\mathbf{a}}^{\mathrm{T}} \sigma(\mathbf{WH}[j] \| \mathbf{WH}[i]))}{\sum_{v_r \in \mathcal{N}_j^\lambda} \exp(\overrightarrow{\mathbf{a}}^{\mathrm{T}} \sigma(\mathbf{WH}[j] \| \mathbf{WH}[r]))}$ calculates one component of ϕ_{ij} considering the *non-linear* relationship between node

v_j's and ego v_i' representations, and its output lies in $(0, 1)$ as a valid probability for ego-network selection. Meanwhile, we further add another component $f_\phi^{lin}(v_i, v_j) = \text{Sigmoid}(\mathbf{H}^\mathrm{T}[j] \cdot \mathbf{H}[i])$ to supercharge ϕ_{ij} with the linearity between node v_j and ego v_i to capture more comprehensive information. Consequently, nodes with similar features and structure information to the ego contribute to higher closeness scores. In the end, we produce the closeness score of $c_\lambda(v_i)$ as:

$$\phi_i = \frac{1}{|\mathcal{N}_i^\lambda|} \sum_{v_j \in \mathcal{N}_i^\lambda} \phi_{ij} \tag{4.3}$$

where $|\mathcal{N}_i^\lambda|$ indicates the number of nodes in $\mathcal{N}_i^\lambda$.

After obtaining ego-networks' closeness scores, we propose an adaptive approach to select a fraction of ego-networks to form super nodes without pre-defined hyper-parameters (cf. the Top-k selection strategy [GJ19]). Our key intuition is that a high diameter ego-network could be composed of multiple low diameter ego-networks. Therefore, we intend first to find these low diameter ego-networks, then recursively aggregate them to form a super node that contains these ego-networks. Specifically, we form ego-networks by selecting a fraction of egos $\hat{N}_p$ as: $\hat{N}_p = \{v_i \mid \phi_i > \phi_j, \forall v_j \in \mathcal{N}_i^1\}$, where $\mathcal{N}_i^1$ means the neighbour nodes of node v_i within the first hop. Note that each node may belong to various ego-networks since they may play different roles in different groups. Therefore, we allow overlapping between different selected ego-networks and utilise $\mathcal{N}_i^1$ instead of $\mathcal{N}_i^\lambda$. If we adopt $\mathcal{N}_i^\lambda$ here, the selected ego node v_i cannot be involved in other ego-networks anymore. Following this, we can select a fraction of ego-networks to form super nodes at granularity level t.

Proposition 1. *Let $\mathcal{G}$ be a connected graph with n nodes, and a total number of n ego-networks can be formed from the graph $\mathcal{G}$, i.e., $\mathcal{C}_\lambda = \{c_\lambda(v_1), c_\lambda(v_2), \ldots, c_\lambda(v_n)\}$. Each ego-network $c_\lambda(v_i)$ is assigned with a closeness score ϕ_i. Then, there exist at least one ego-network $c_\lambda(v_i)$ that satisfies $\phi_i > \phi_j$, $\forall v_j \in \mathcal{N}_i^1$.*

Proof of Proposition 1. For $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with n nodes. n ego-networks can be generated by following the procedures, $c_\lambda(v_i) = \{j \mid \forall j \in \mathcal{N}_i^\lambda\}$, and each ego-network is given a closeness score ϕ_i as Equation 4.3. We assume that these cluster closeness scores are not all the same thus, there exists at least one maximum ϕ_{max}. Hence, the clusters with closeness score ϕ_{max} satisfy the requirements of ego-network selection requirement that:

$$\phi_{max} > \phi_j, \ \ \forall v_j \in \mathcal{N}_{max}^1 \tag{4.4}$$

where $\mathcal{N}_{max}^1 = \{v_j \mid if\, d(v_i, v_j) = 1\}$. So, for any connected $\mathcal{G}$ with n nodes, there exists at least one cluster that satisfies the requirements of our ego-network selection approach. □

Proposition 1 ensures that our super node determination method can find at least one ego-network to generate a super graph for any graph. It guarantees the generality of our strategy.

Meanwhile, we would also retain nodes that do not belong to any selected ego-networks, denoted as $\hat{N}_r$, to maintain the graph structure: $\hat{N}_r = \{v_j \mid v_j \notin c_\lambda(v_i), \forall v_i \in \hat{N}_p\}$. In this way, a super node formation matrix $\mathbf{S}_t \in \mathbb{R}^{n \times (|\hat{N}_p| + |\hat{N}_r|)}$ can be formed, where $(|\hat{N}_p| + |\hat{N}_r|)$ is number of nodes of the generated super graph, rows of $\mathbf{S}_t$ corresponds to the n nodes of $\mathcal{G}_{t-1}$, and columns of $\mathbf{S}_t$ corresponds to the selected ego-networks ($\hat{N}_p$) plus the remaining nodes ($\hat{N}_r$). We have $\mathbf{S}_t[i,j] = \phi_{ij}$ if node v_j belongs to the selected ego-network $c_\lambda(v_i)$ and $\mathbf{S}_t[i,j] = 1$ if node v_j is a remaining node corresponds to node v_i in the super graph; otherwise $\mathbf{S}_t[i,j] = 0$. The weighted super node formation matrix $\mathbf{S}_t$ can better maintain the relation between different super nodes in the pooled graph.

Maintaining super graph connectivity. After selecting the ego-networks and retaining nodes in level $t-1$, as shown in Figure 4.2-(b)-(iii-iv), we construct the new adjacent matrix $\mathbf{A}_t$ for the super graph using $\hat{\mathbf{A}}_{t-1}$ and $\mathbf{S}_t$ as follows: $\mathbf{A}_t = \mathbf{S}_t^{\mathrm{T}} \hat{\mathbf{A}}_{t-1} \mathbf{S}_t$. This formula makes any two super nodes connected if they share any common nodes or any two nodes are already neighbours in $\mathcal{G}_{t-1}$. In addition, $\mathbf{A}_t$ retains the edge weights passed by $\mathbf{S}_{t-1}$ that involves the relation weights between super nodes. Eventually, we obtain a generated super graph $\mathcal{G}_t$ at granularity level t.

Super node feature initialisation. All nodes in the super graph $\mathcal{G}_t$ need initial feature vectors to support the graph convolution operation. Recall that we have the closeness score as calculated in Equation 4.2, between node v_j to the ego v_i. However, this is not equivalent to the contribution of node v_j's feature to the super node feature, since we need to compare the relationship strength between ego v_i and v_j with the relation between other $v_r \in c_\lambda(v_i)$. Therefore, we further propose a super node feature initialisation method through a self-attention mechanism [VCC+18]. Specifically, it can be described as:

$$\mathbf{X}_t[i] = \mathbf{H}_{t-1}[i] + \sum_{v_j \in c_\lambda(v_i) \setminus v_i} \alpha_{ij} \mathbf{H}_{t-1}[j] \tag{4.5}$$

where $\mathbf{H}_{t-1}$ is the generated node representation by the $(t-1)$-th primary GNN layer, i.e., at level $t-1$ with a similar method as Equation 4.1, α_{ij} describes the importance of node v_j to the initial feature of $c_\lambda(v_i)$ at level t. And α_{ij} can be learned as follows: $\alpha_{ij} = \frac{\exp(\overrightarrow{\mathbf{a}_1}^{\mathrm{T}} \sigma(\mathbf{W}(\phi_{ij}\mathbf{H}_{t-1}[j]) \| \mathbf{H}_{t-1}[i]))}{\sum_{v_r \in c_\lambda(v_i)} \exp(\overrightarrow{\mathbf{a}_1}^{\mathrm{T}} \sigma(\mathbf{W}(\phi_{ir}\mathbf{H}_{t-1}[r]) \| \mathbf{H}_{t-1}[i]))})$ where $\overrightarrow{\mathbf{a}_1} \in \mathbb{R}^{2\pi}$ is the weight vector. For the remaining nodes $\hat{N}_r$ that do not belong to any super nodes, we keep their representations of $\mathbf{H}_{t-1}$ as initial node features.

4.3.3 Graph Unpooling

Different from existing graph pooling models [YYM+18, CVJ+18, LLK19, RST20, YJ20] which only coarsen graphs to generate graph representations, we aim to mutually utilise node-wise and graph-wise tasks to better encode multi-grained semantics into both node and graph representations under a unified framework. We design a mechanism to allow the learned multi-grained semantics to enrich the node representations of the original graph $\mathcal{G}$ as shown in Figure 4.2-(a). Vice versa, the updated node representation can further ameliorate the graph representation

in the next training iteration. A reasonable unpooling operation that passes macro-level information to original nodes has not been well studied in the literature. For instance, Gao et al. [GJ19] directly relocate the super node back into the original graph and utilise other GNN layers to spread its message to other nodes. However, these additional aggregation operations cannot allow each node to receive meaningful information since some nodes may be distant from super nodes. In such a case, these operations can exacerbate local-smoothing [LHW18].

We implement the unpooling process by devising a *top-down* message-passing mechanism, which endows GNN models with meso/macro level knowledge. Specifically, since $\mathbf{S}_t$ records how nodes of $\mathcal{G}_{t-1}$ form the super nodes of $\mathcal{G}_t$, so we utilise $\mathbf{S}_t$ to restore the generated ego-network representation at level t to that at level $t-1$ until we arrive at the original graph $\mathcal{G}$, i.e., $t \to 0$, as follows:

$$\hat{\mathbf{H}}_t = (\mathbf{H}_t^{\mathrm{T}} \mathbf{S}_t^{\mathrm{T}} \mathbf{S}_{t-1}^{\mathrm{T}} \dots \mathbf{S}_1^{\mathrm{T}})^{\mathrm{T}} \tag{4.6}$$

where $\hat{\mathbf{H}}_t \in \mathbb{R}^{n \times d}$. At the end of each iteration, nodes in the original graph $\mathcal{G}$ receive high-level semantic messages from the different levels, i.e., $\{\hat{\mathbf{H}}_1, \dots, \hat{\mathbf{H}}_T\}$. As illustrated in Figure 4.2-(b)-(iv-i), the graph unpooling process can be treated as an inverse process of the adaptive graph pooling process.

4.3.4 Flyback Aggregation

Since the super graphs at different granularity levels present multi-grained semantics and how each node utilises the received semantic information with its $flat$ representation is a challenging question. And nodes of the same graph may need different granularity levels' information. Therefore, we propose a novel attention mechanism to integrate the derived representations at different levels, given by:

$$\mathbf{Z} = \mathbf{H} + \sum_{\forall t} \boldsymbol{\beta}_t \hat{\mathbf{H}}_t \tag{4.7}$$

where the attention score $\boldsymbol{\beta}_t$ estimates the importance of the message from level t, given by: $\boldsymbol{\beta}_t[i] = \frac{\exp(\overrightarrow{\mathbf{a}_2}^{\mathrm{T}} \sigma(\mathbf{W}\hat{\mathbf{H}}_t[i] \| \mathbf{H}[i])))}{\sum_{j \in T} \exp(\overrightarrow{\mathbf{a}_2}^{\mathrm{T}} \sigma(\mathbf{W}\hat{\mathbf{H}}_j[i] \| \mathbf{H}[i])))}$ where $\overrightarrow{\mathbf{a}_2} \in \mathbb{R}^{2\pi}$ is the weight vector. We term this process as the *flyback* aggregation, which considers the attention scores of different levels and allows each node to decide whether/how to utilise semantic information from different granularity levels. We verify the effectiveness of *flyback* aggregation in the ablation study of Section 4.4.2 and discuss the explainability in Section 4.4.3.

4.3.5 Training Strategy

Till now, there are still two challenges when training the model. The first is how to highlight the difference among nodes' representations from different ego-networks. Nodes belonging to neighbouring ego-networks receive closely related messages from super nodes since their super nodes are connected in the super graph, and local smoothing makes their representation vectors similar. Representations of

proximal nodes could be further closer to each other in the representation latent space. To address this problem and enhance the discrimination capability between ego-networks, we exploit a self-optimisation strategy [XGF16], which makes nodes in different ego-networks distinguishable. Specifically, we use the Student's t-distribution (Q) as a kernel to measure the similarity between representation vectors of v_j and ego v_i: $q_{ij} = \frac{(1+\|\mathbf{Z}[j]-\mathbf{Z}[i]\|^2/\mu)^{-\frac{\mu+1}{2}}}{\sum_{i'}(1+\|\mathbf{Z}[j]-\mathbf{Z}[i']\|^2/\mu)^{-\frac{\mu+1}{2}}}$, where $v_j \in c_\lambda(v_i)$, $v_{i'}$ are other ego nodes, μ are the degrees of freedom of Student's t-distribution. Following this, q_{ij} can be integrated the probability of assigning node v_j to ego v_i. In this chapter, we set $\mu = 1$ the same as [XGF16]. After, we propose to learn better node representations by matching Q to the auxiliary target distribution (P), and we choose the proposition of [XGF16] which first raises q_i to the second power and then normalises by frequency per ego-network: $p_{ij} = \frac{q_{ij}^2/g_i}{\sum_{i'}(q_{ij}^2/g_{i'})}$, where $g_i = \sum_j q_{ij}$. Therefore, apart from the task-related loss function $\mathcal{L}_{task}$, we further define a KL divergence loss as:

$$\mathcal{L}_{KL} = KL(P \parallel Q) = \sum_{v_j}\sum_{v_i} p_{ij}\log\frac{p_{ij}}{q_{ij}} \tag{4.8}$$

The second challenge is to avoid the over-smoothing problem that nodes of a graph tend to have indistinguishable representations. GNN is proved as a special form of Laplacian smoothing [LHW18, CLL$^+$20] that naturally assimilates the nearby nodes' representations. AdamGNN further exacerbates this problem because it distributes semantic information from one super node representation to all nodes of the ego-network. Therefore, we introduce the reconstruction loss, which can alleviate the over-smoothing issue and drive the node representations to retain the structure information of $\mathcal{G}$ by differentiating non-connected nodes' representations. Specifically, the reconstruction loss is defined as:

$$\mathcal{L}_R = -\frac{1}{n}\sum(\mathbf{A}_{ij}\cdot\log(\mathbf{A}'_{ij}) + (1-\mathbf{A}_{ij})\cdot\log(1-\mathbf{A}'_{ij})) \tag{4.9}$$

where $\mathbf{A}' = \text{Sigmoid}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})$. Therefore, the overall loss function consists of the training task $\mathcal{L}_{Task}$, the self-optimising task $\mathcal{L}_{KL}$, and the reconstruction task $\mathcal{L}_R$, given by:

$$\mathcal{L} = \mathcal{L}_{Task} + \gamma\mathcal{L}_{KL} + \delta\mathcal{L}_R \tag{4.10}$$

where $\mathcal{L}_{Task}$ is a flexible task-specific loss function, and γ and δ are two hyper-parameters that we discuss in Section 4.4.1. Note that for link prediction task we have $\mathcal{L} = \mathcal{L}_R + \gamma\mathcal{L}_{KL}$, since $\mathcal{L}_{Task}$ equals to $\mathcal{L}_R$. Moreover, we demonstrate the effectiveness of each component of loss function $\mathcal{L}$ in the ablation study of Section 4.4.2.

4.3.6 Algorithm

We have presented the idea of AdamGNN and the design details of each component in Section 4.3. Here, we summarise the entire AdamGNN model in Algorithm 2 to provide a general view of our model. Given a graph $\mathcal{G}$, we first apply a primary GNN encoder to generate the primary node embedding (line 1). Then we construct

Algorithm 2: *Adaptive Multi-grained Graph Neural Networks*

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$.
Output: node representations $\mathbf{Z}$, graph representations $\mathbf{Z}_g$

1 $\mathbf{H} = \text{ReLU}(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{X}\mathbf{W})$;
2 **for** $t \leftarrow \{1, 2, \ldots, T\}$ **do**
3 **for** $v_i \leftarrow \{v_1, v_2, \ldots, v_n\}$ **do**
4 **for** $v_j \in \mathcal{N}_i^{\lambda}$ **do**
5 $\phi_{ij} = f_{\phi}^{non}(v_i, v_j) \times f_{\phi}^{lin}(v_i, v_j)$;
6 **end**
7 $\phi_i = \frac{1}{|\mathcal{N}_i^{\lambda}|} \sum_{v_j \in \mathcal{N}_i^{\lambda}} \phi_{ij}$;
8 **end**
9 **for** $v_i \leftarrow \{v_1, v_1, \ldots, v_n\}$ **do**
10 $\hat{N}_p = \{v_i \mid \phi_i > \phi_j, \forall v_j \in \mathcal{N}_i^1\}$;
11 **end**
12 $\hat{N}_r = \{v_j \mid v_j \notin c_{\lambda}(v_i), \forall v_i \in \hat{N}_c\}$;
13 Generate the super node formation matrix: $\mathbf{S}_t$;
14 **for** $v_i \in \hat{N}_r$ **do**
15 $\mathbf{X}_t[i] = \mathbf{H}_{t-1}[i]$;
16 **end**
17 **for** $v_i \in \hat{N}_p$ **do**
18 $\mathbf{X}_t[i] = \mathbf{H}_{t-1}[i] + \sum_{v_j \in c_{\lambda}(v_i) \setminus v_i} \alpha_{ij} \mathbf{H}_{t-1}[j]$;
19 **end**
20 $\mathbf{A}_t = \mathbf{S}_t^{\mathrm{T}} \hat{\mathbf{A}}_{t-1} \mathbf{S}_t$;
21 $\mathbf{H}_t = \text{ReLU}(\hat{\mathbf{D}}_t^{-\frac{1}{2}} \hat{\mathbf{A}}_t \hat{\mathbf{D}}_t^{\frac{1}{2}} \mathbf{X}_t \mathbf{W}_t)$;
22 $\hat{\mathbf{H}}_t = (\mathbf{H}_t^T \mathbf{S}_t^{\mathrm{T}} \mathbf{S}_{t-1}^{\mathrm{T}} \ldots \mathbf{S}_1^{\mathrm{T}})^{\mathrm{T}}$;
23 **end**
24 $\mathbf{Z} = \mathbf{H} + \sum_t^T \beta_t \hat{\mathbf{H}}_t$;
25 $\mathbf{Z}_g = \text{READOUT}(\{\mathbf{Z}, \hat{\mathbf{H}}_1, \ldots, \hat{\mathbf{H}}_T\}) \in \mathbb{R}^d$;

a multi-grained structure with t-th level (line 3-13) with the proposed adaptive graph pooling operator. Meanwhile, we also propose a method to define the initial features of pooled super nodes (line 14-19). The graph connectivity of the pooled graph is maintained by line 20. We apply a GNN encoder on the pooled graph to summarise the relationships between super nodes (line 21) to learn macro grained semantics of t-th granularity level. The learned multi-grained semantics are further distributed to the original graph following an unpooling operator (line 22). Last, the *flyback* aggregator generates the meso/macro level knowledge from different levels as the node representations of $\mathcal{G}$ (line 24), and an additional READOUT operators [CCB+20] produce the node representations as to the graph representation (line 25).

Model scalability. According to the design for AdamGNN framework, we can find that the primary node representation learning module of each level and the adaptive graph pooling and unpooling operators are categorised as a local network algorithm [Ten16], which only involves local exploration of the graph structure. Therefore, our design enables AdamGNN to scale to representation learning on

large-scale graphs and to be friendly to distributed computing settings [QCD+20]. We present instances that utilise a multi-GPU computing framework to accelerate the training process of AdamGNN in Section 4.4.3.

4.4 Experiments

4.4.1 Experimental Setup

We evaluate our proposed model, AdamGNN, on 14 benchmark datasets, and compare with 16 competing methods over both node- and graph-wise tasks, including node classification, link prediction and graph classification.

Table 4.3: Data statistics for node-wise tasks and the split for the semi-supervised node classification task. N.A. means a dataset does not contain node attributes or does not support semi-supervised settings.

Dataset	#Nodes	#Edges	#Features	#Classes	#Train-#Val-#Test
ACM	3,025	13,128	1,870	3	60-500-1,000
Citeseer	3,327	4,552	3,703	6	120-500-1,000
Cora	2,708	5,278	1,433	7	140-500-1,000
DBLP	4,057	3,528	334	4	80-500-1,000
Emails	799	10,182	N.A.	18	180-309-310
Pubmed	19,717	88,648	500	3	60-500-1,000
Wiki	2,405	1,2178	4973	17	481-962-962
Ogbn-arxiv	169,343	1,166,243	128	40	N.A.

Table 4.4: Data statistics for graph classification.

Dataset	#Graphs	#Nodes (avg)	#Edges (avg)	#Features	#Classes
NCI1	4,110	29.87	32.3	37	2
NCI109	4,127	29.68	32.13	38	2
D&D	1,178	284.32	715.66	89	2
MUTAG	188	17.93	19.79	7	2
Mutagenicity	4,337	30.32	30.77	14	2
PROTEINS	1,113	39.06	72.82	32	2

Datasets. To validate the effectiveness of our model on real-world applications, we adopt datasets that come from different domains with different topics and relations. We use 8 datasets for node-wise tasks (data statistics are summarised in Table 4.3). Ogbn-arxiv [HFZ+20], ACM [BWS+20], Cora [KW17], Citeseer [KW17] and Pubmed [KW17] are paper citation graph datasets. DBLP [BWS+20] is an author graph dataset from the DBLP dataset. Emails [LK14] is an email communication graph dataset. Wiki [YLZ+15] is a webpage graph dataset.

For the graph classification task, we adopt 6 bioinformatics datasets [YJ20] (data statistics are summarised in Table 4.4). D&D and PROTEINS are datasets containing proteins as graphs. NCI1 and NCI109 involve anticancer activity graphs. The MUTAG and Mutagenicity consist of chemical compounds divided into two classes

according to their mutagenic effect on a bacterium. Note that all datasets can be downloaded with our published code automatically.

Competing methods. For node-wise tasks, we adopt 9 competing methods that include 7 GNN models with flat message-passing mechanism, and one state-of-the-art method that contains a hierarchical structure: MLP [Ros61], GCN [KW17], GraphSAGE [HYL17b], GAT [VCC+18], GIN [XHLJ19], PNA [CCB+20], GCNII [CWH+20], GRAND [FZD+20] and g-U-Net [GJ19]. For the graph classification task, except for GIN, PNA and g-U-Net which support graph-wise tasks, we adopt extra 7 competing methods that involve the state-of-the-art models: 3WL-GNN [MBSL19], SortPool [ZCNC18], DiffPool [YYM+18], SagPool [LLK19], EigentPool [MWAT19], StructPool [YJ20] and ASAP [RST20]. Note that we already carefully discussed these competing methods in Section 3.2 and Section 4.2; therefore, we do not repeat the method description here. The competing model implementations can be found in our published project on Github.

Evaluation settings. For the *node-wise* tasks, we follow the *supervised node classification* (Sup-NC) settings of PNA [CCB+20], i.e., using two sets of 10% labelled nodes as validation and test sets, with the remaining 80% labelled nodes used as the training set. Meanwhile, we follow the *semi-supervised node classification* (Semi-NC) settings of GCN [KW17], and the data split is shown in Table 4.3, i.e., randomly assigning 20 labelled nodes for each class for training, and 500 and 1000 nodes for validation and testing, respectively. Note that since the Email does not have a sufficient number of nodes for classic Semi-NC setting, we choose 10 labelled nodes for each class for training, and the rest data is evenly separated as validation and test sets. Wiki is imbalanced, where some classes only have very few labelled nodes, e.g., class 12 has 9 labelled nodes and class 4 has 10 labelled nodes, which cannot support Semi-NC settings. Therefore, we follow Sup-NC settings to split Wiki for the Semi-NC experimental parts but use only 20% labelled nodes for training and the remaining nodes for validation and testing, respectively. Ogbn-arxiv follows the fixed split of OGB leaderboard [HFZ+20]. For the *link prediction* (LP) task, we follow the settings of [YYL19], i.e., using two sets of 10% existing edges as validation and test sets, with the remaining 80% edges used as the training set. Note that, an equal number of non-existent links are randomly sampled and used for every set. We present the average performance of 10 times experiments with random seeds. The AUC score evaluates link prediction, and node classification tasks are evaluated by accuracy. We conduct the experiments with random parameter initialisation with 10 random seeds and report the average performance.

For the *graph-wise* task, i.e., *graph classification* (GC) task, we perform all experiments following the pooling pipeline of SagPool [LLK19]. 80% of the graphs are randomly selected as training, and the rest two 10% graphs are used for validation and testing, respectively. We conduct the experiments using 10-fold cross-validation and report the average classification accuracy on 10 random seeds.

Model configuration. For all methods, we set the embedding dimension $d = 64$ and utilise the same learning rate $= 0.01$, Adam optimiser, number of training epochs $= 1000$ with early stop (100). In terms of the neural network layers, we

report the one with better performance of GCNII with better performance among $\{8, 16, 32, 64, 128\}$; for other models, we report the one with better performance between $2-4$; For all models with hierarchical structure (including AdamGNN), we use GCN as the GNN encoder for fair comparision. In terms of the number of levels that is required by hierarchical models, we present the one with better performance, between $2-5$. On other hyper-parameter settings of competing methods, we employ the default values of each competing method, as shown in the paper's official implementation. Particularly, for AdamGNN, by tuning the hyper-parameters based on the validation set, we have $\gamma = 0.1$ and $\delta = 0.01$ for Equation 4.10 for the experiments to let loss values lie in a reasonable range, i.e., $(0, 10)$. We employ Pytorch and PyTorch Geometric to implement all models. Experiments were conducted with GPU (NVIDIA Tesla V100) machines.

4.4.2 Experimental Results and Ablation Study

Table 4.5: Results in accuracy (%) for supervised and semi-supervised node classification on eight datasets. ‡ indicates the results from OGB leaderboard [HFZ+20]. The bold numbers represent the top-2 results.

Models	ACM		Citeseer		Cora		Emails		DBLP		Pubmed		Wiki		Ogbn-arxiv
	Sup	Semi	Sup	Semi	Sup	Semi	Sup	Semi	Sup	Semi	Sup	Semi	Sup	Semi	Sup
MLP	87.08	76.14	70.87	59.10	76.12	57.82	N.A.	N.A.	79.17	66.55	83.41	72.81	20.42	17.46	55.50‡
GCN	92.25	86.14	76.13	70.53	88.90	80.41	85.03	77.32	82.68	72.22	86.04	77.33	57.36	46.19	71.74‡
GraphSAGE	92.48	87.18	76.75	70.51	**88.92**	81.07	85.80	78.19	83.20	72.20	86.01	78.43	57.24	49.21	71.49‡
GAT	91.69	87.52	**76.96**	69.83	88.33	81.40	84.67	77.35	84.04	73.24	86.21	77.54	58.07	50.27	72.06
GIN	90.66	85.98	76.39	66.87	87.74	79.46	**87.18**	79.23	82.54	73.42	87.10	77.54	66.29	49.88	71.76
PNA	**93.97**	**89.81**	72.67	61.78	88.78	71.25	81.25	73.23	**86.21**	72.40	87.63	72.90	21.67	19.58	72.37
GCNII	93.05	87.80	76.37	72.27	88.07	**82.30**	82.51	72.90	84.48	72.82	84.48	79.13	60.24	**56.18**	**72.74**‡
GRAND	93.05	86.90	76.88	**73.54**	87.82	**82.80**	52.53	71.94	85.47	70.46	86.63	**81.93**	59.58	27.94	70.97
g-U-Net	93.42	89.34	75.59	67.59	87.68	80.71	89.16	**80.48**	85.27	**73.86**	**87.67**	77.04	**71.33**	54.18	71.78
AdamGNN	**94.37**	**90.72**	**78.92**	**72.42**	**90.92**	81.37	**91.88**	**83.23**	**88.36**	**74.60**	**89.81**	**80.40**	**73.37**	**62.06**	**72.65**

Performance on node-wise tasks. We compare AdamGNN with 7 GNN models and one pooling-based model, i.e., g-U-Net, since other pooling approaches do not provide an unpooling operator and thus cannot support node-wise tasks. Results on node classification (with supervised and semi-supervised settings) are summarised in Table 4.5. They show that AdamGNN can outperform most competing methods with up to 10.47% and 5.39% improvements on semi-supervised and supervised settings, respectively. AdamGNN brings the most significant improvement in Wiki data with semi-supervised settings, and the competing method that only adopts node features, i.e., MLP, achieve terrible accuracy, 17.46%. We argue that because the node features and node labels are weakly correlated in this dataset, the multi-grained semantics provided by AdamGNN help to ameliorate the performance.

Link prediction results in Table 4.6 show that AdamGNN can significantly outperform the 7 competing methods by up to 25.3% improvement in terms of AUC. It indicates the versatility of AdamGNN on different node-wise tasks and exhibits the usefulness of modelling multi-grained semantics into node representations. Similar

Table 4.6: Results in AUC for link prediction on seven datasets.

Models	ACM	Citeseer	Cora	Emails	DBLP	Pubmed	Wiki
GCN	0.975	0.887	0.918	0.930	0.904	0.941	0.523
GraphSAGE	0.972	0.884	0.908	0.923	0.889	0.905	0.577
GAT	0.968	0.910	0.912	0.930	0.889	0.947	0.594
GIN	0.787	0.808	0.878	0.859	0.820	0.927	0.501
PNA	0.978	0.974	0.731	0.932	0.908	0.896	0.538
GCNII	0.968	0.969	0.871	0.926	0.890	0.933	0.709
g-U-Net	0.890	0.918	0.932	0.936	0.934	0.962	0.734
AdamGNN	**0.988**	**0.975**	**0.948**	**0.957**	**0.965**	**0.969**	**0.920**

to node classification task, AdamGNN again brings the most significant AUC improvement on the Wiki dataset, i.e., achieving 29.76% improvement compared with the *flat* GNN models.

Table 4.7: Results in accuracy (%) for graph classification on six datasets.

Models	NCI1	NCI109	D&D	MUTAG	Mutagenicity	PROTEINS
GIN	76.17±1.12	77.31±1.42	78.05±1.89	75.11±2.64	77.24±2.26	75.37±1.62
3WL-GNN	79.38±1.73	78.34±1.90	78.32±2.44	78.34±3.39	81.52±2.23	77.92±2.09
PNA	78.96±1.01	79.06±1.15	75.47±2.52	**81.91**±2.59	81.72±1.46	77.72±2.25
SortPool	72.25±1.33	73.21±2.21	73.31±2.43	71.47±2.31	74.65±3.35	70.49±2.37
DiffPool	76.47±0.96	76.17±1.11	76.16±1.69	73.61±3.94	76.30±0.37	71.90±2.75
g-U-Net	77.56±1.92	77.02±2.30	73.98±2.63	76.60±5.03	78.64±3.11	72.94±3.68
SagPool	75.76±1.29	73.67±2.32	76.21±1.56	75.27±1.92	77.09±1.17	75.27±0.57
EigenPool	77.54±1.82	77.20±1.81	78.25±1.78	76.21±2.74	78.60±1.24	75.19±1.95
StructPool	77.61±1.08	78.39±1.23	80.10±1.77	77.13±3.93	80.94±1.67	78.84±1.70
ASAP	78.21±1.75	78.16±1.62	79.50±1.80	80.17±1.77	81.52±1.24	**78.92**±1.45
AdamGNN	**79.77**±1.29	**79.36**±1.03	**81.51**±1.56	80.11±2.58	**82.04**±1.73	77.04±0.78

Performance on graph-wise task. Experimental results are summarised in Table 4.7. It is apparent that our AdamGNN achieves the best performance on 4 of the 6 datasets, and consistently outperforms most of competing pooling-based techniques by 1.76% improvement. For the datasets MUTAG and PROTEINS, our results are still competitive since PNA and ASAP only slightly outperform AdamGNN. Nevertheless, AdamGNN is still better than other baselines. This is because our model involves adaptive pooling and unpooling operators to update node- and graph-wise information interactively, and further enhance the representations of nodes and graphs during the training process.

Table 4.8: Comparison of AdamGNN with different loss functions on three tasks. NC task follows the supervised settings.

	DBLP (LP)	Citeseer (NC)	Mutagenicity (GC)
AdamGNN w/ $\mathcal{L}_{Task}$	0.956	76.63	79.04
AdamGNN w/ $\mathcal{L}_{Task}$+$\mathcal{L}_{KL}$	-	77.17	78.94
AdamGNN w/ $\mathcal{L}_{Task}$+$\mathcal{L}_{R}$	-	77.64	80.65
AdamGNN (Full model)	**0.965**	**78.92**	**82.04**

Ablation study of different loss functions. The loss function of our AdamGNN consists of three parts, i.e., $\mathcal{L}_{Task}$, $\mathcal{L}_R$ and $\mathcal{L}_{KL}$. We examine how each part contributes to the performance. Table 4.8 provides the results. For the link prediction task, we have $\mathcal{L} = \mathcal{L}_R + \gamma\mathcal{L}_{KL}$, since $\mathcal{L}_{Task}$ equals to $\mathcal{L}_R$. Thus, two comparison experiments are missing in link prediction. From the results, we can see that $\mathcal{L}_R$ can significantly improve the performance over three tasks. This is because it can eliminate the over-smoothing problem caused by the received messages from different granularity levels. Meanwhile, $\mathcal{L}_{KL}$ can slightly improve the results of node-wise tasks as well.

Table 4.9: Comparison of AdamGNN with and without *flyback* aggregation in terms of graph classification accuracy on NCI1, NCI109 and Mutagenicity datasets.

AdamGNN	**NCI1**	**NCI109**	**Mutagenicity**
No *flyback* aggregation	75.54	77.49	79.89
Full model	**79.77**	**79.36**	**82.04**

Ablation study of the *flyback* aggregation. Experimental results of node-wise tasks confirm that capturing multi-grained semantics in AdamGNN can help to learn more meaningful node representations. Here, we further study whether *flyback* aggregator can improve graph representations. Specifically, we aim to see how the *flyback* aggregator contributes to graph classification performance by removing and keeping it. The results are summarised in Table 4.9. It is clear that the node representations enhanced by the *flyback* aggregation can indeed improve the graph representation in the classification task.

Table 4.10: Comparison of AdamGNN with different number of granularity levels in terms of different tasks. NC task follows the supervised settings.

# Levels (T)	**DBLP**	**Wiki**	**ACM**	**Citeseer**	**Emails**	**Mutagenicity**
	LP	LP	NC	NC	NC	GC
1	0.951	0.912	92.60	77.68	86.83	78.16
2	0.958	0.913	93.38	74.67	**91.88**	**82.04**
3	0.959	0.917	**94.37**	76.15	90.61	81.58
4	**0.965**	**0.920**	90.84	**78.92**	-	81.01

Ablation study of number of granularity levels. As it has been proved that the existing GNN models have worse performance when the neural network goes deeper [LHW18], here we examine how AdamGNN can be benefited from more granularity levels. By varying the number of granularity levels, we report the performance of AdamGNN on link prediction, supervised node classification and graph classification, as summarised in Table 4.10. We can observe that increasing the number of granularity levels can improve both link prediction and node classifications. As for graph classification, 2 levels would be a proper choice.

Ablation study of different primary GNN encoders. We adopted GCN as the default primary GNN encoder in model presentation (Section 4.3) and previous experiments. Here, we present more experimental results by endowing AdamGNN with advanced GNN encoders in Table 4.11. The table demonstrates that advanced GNN encoders can still benefit from the multi-grained semantics of AdamGNN.

Table 4.11: Comparison of AdamGNN with different primary GNN encoders, follow the semi-supervised node classification settings.

Models	Cora	Citeseer	Pubmed
GCN	80.41	70.53	77.33
AdamGNN w/ GCN	**81.37**	**72.42**	**80.40**
PNA	71.25	61.78	72.90
AdamGNN w/ PNA	**76.01**	**68.12**	**76.89**
GCNII	82.30	72.27	79.13
AdamGNN w/ GCNII	**83.48**	**73.44**	**80.62**

For instance, GCNII can stack lots of layers to capture long-range information; however, it still follows a *flat* message-passing mechanism hence naturally ignoring the multi-grained semantics. AdamGNN further ameliorates this problem for better performance.

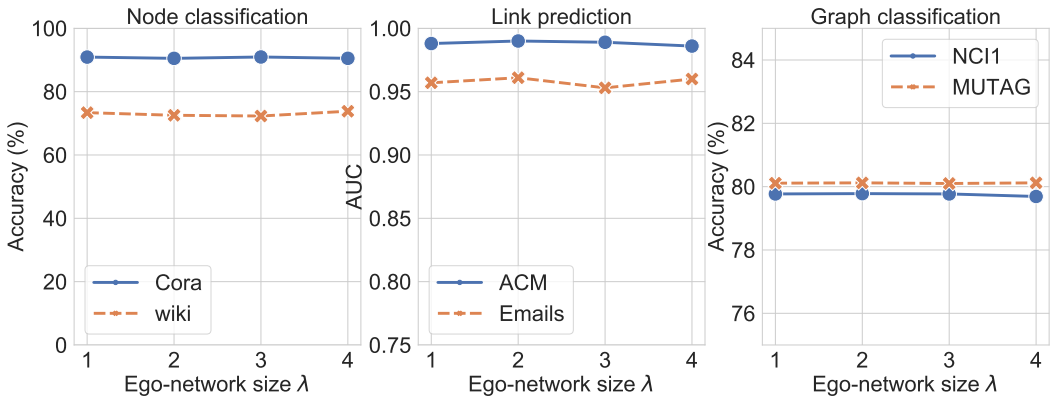


Figure 4.3: Ablation study of Ego-network size λ in terms of different tasks. NC task follows the supervised settings.

Ablation study of Ego-network size (λ). The size of an ego-network as defined in Section 4.3 is captured by λ. We present an ablation study to investigate the influence of λ on AdamGNN's performance, results are summarised in Figure 4.3. The figure indicates that λ has no significant influence on the model performance. We simply adopt $\lambda = 1$ throughout the chapter.

4.4.3 More Model Analysis

Running time comparison. We present the average epoch training time of different node and graph classification models in Table 4.12 and Table 4.13, respectively. In terms of node classification task, AdamGNN requires more training time due to the computation cost of α_{ij} and β_t, similar to any attention-mechanism enhanced models [VSP+17]. However, AdamGNN is designed as a local network algorithm, maintaining good scalability; hence it can be easily accelerated by mini-batch and multi-GPUs computing frameworks [KSO+21]. It significantly mitigates the computational issues. On the other hand, AdamGNN follows the sparse design similar to SagPool, ASAP, striking a balance between performance improvement and maintaining proper time efficiency. DiffPool and StructPool employ a *dense*

Table 4.12: Average one epoch running time (in seconds) for supervised node classification task. (×2) means accelerated by 2 GPUs [KSO+21].

Models	ACM	Citeseer	Cora
GCN	0.008	0.008	0.009
GAT	0.010	0.011	0.011
GCNII	0.045	0.040	0.041
g-U-Net	0.076	0.064	0.069
AdamGNN (×1)	0.087	0.072	0.074
AdamGNN (×2)	0.059	0.050	0.052
AdamGNN (×3)	0.050	0.047	0.048

Table 4.13: Average one epoch running time (in seconds) for graph classification task.

Models	NCI1	NCI109	PROTEINS
DiffPool	6.23	3.22	3.65
SagPool	1.95	1.55	0.45
g-U-Net	4.58	4.45	1.46
EigenPool	3.88	3.54	1.38
ASAP	2.04	1.83	1.09
StructPool	6.31	6.04	1.34
AdamGNN	3.62	3.24	1.03

mechanism that is not easily scalable to larger graphs [CVJ+18], and g-U-Net uses convolution operations, which bring additional computation cost, to distribute the received information to the graph.

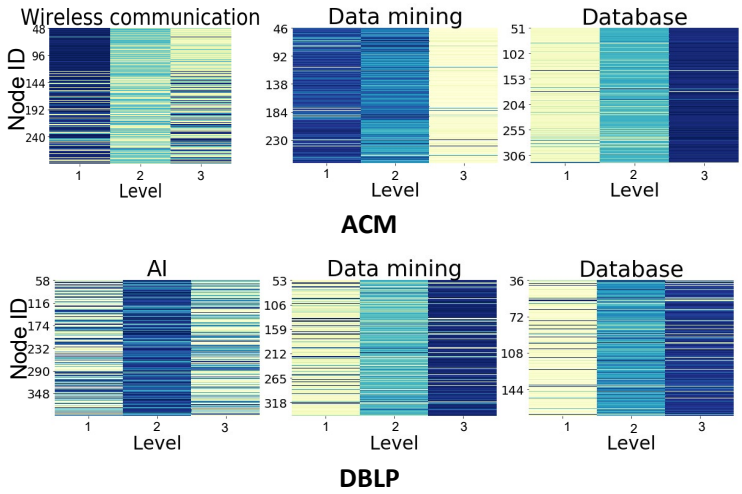


Figure 4.4: Visualisation of attention weight for messages at different granularity levels. Dark colours indicate higher weights.

Messages from different levels. We aim to figure out the importance of received messages from different granularity levels since messages from different levels contain the meso/macro-level knowledge encoded by super nodes. Here we consider the node classification on the ACM and DBLP as an example. Specifically, ACM's paper nodes are labelled with 3 topics: database (DB), data mining (DM) and wireless communication (WC); DBLP's author nodes have 4 research areas: AI, DB, DM

and computer vision. The node classification task on these two datasets is predicting the paper/scholar's research area. The attention scores of nodes that highlight the importance of different levels' messages are plotted in Figure 4.4. We can find different distributions of attention weights over different granularity levels for various areas' classifications. The relatively general topics, i.e., AI and WC, receive messages from different levels with relatively indistinguishable weights, i.e., higher attention scores of nodes are distributed across levels. The DM topic in two datasets has different attention patterns: it receives messages from level 1 with the greatest attention in ACM but receives greatest attention messages from level 3 in DBLP. This is because DM is not closely related to the other two topics of ACM dataset, Scholars of DM-related papers are less likely to collaborate with researchers from DB or WC. DM papers are close to each other in the network. Thus DM papers only need to receive level 1 granularity semantic information summarised from neighbouring nodes. In contrast, DBLP's other 3 topics are close to DM. DM-related scholars may cite any other scholars' papers, and information related to DM is scattered over author nodes in DBLP network. Therefore, DM researchers tend to be characterised by level 3 semantics from a wide range.

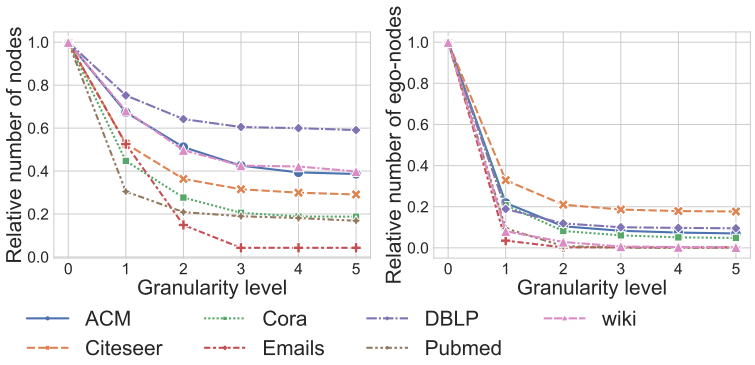


Figure 4.5: Visualisation of different granularity levels.

Visualisation of different granularity levels. To better understand the process of learning multi-grained semantics, we report the relative numbers of nodes (i.e., node ratio concerning the original graph) at different granularity levels generated by AdamGNN in 7 datasets, as shown in Figure 4.5. In particular, we set the max number of granularity levels as 5, and level 0 indicates the original graph. We train AdamGNN with semi-supervised node classification and report the relative number of nodes and selected ego-nodes at each level. In the right, we can find out that the number of ego-nodes can stay stable after $1-2$ times of our adaptive pooling, which indicates that AdamGNN can effectively find a compact structure that contains multi-grained semantics. In the left, we can see that the number of nodes of each level stabilises after $3-4$ times of our adaptive pooling, which illustrates AdamGNN maintain the graph size at a proper level to avoid a dense super graph.

Visualisation of adjacency matrices at different granularity levels. One of the fundamental limitations of existing GNNs is the inability of capturing long-range node interactions in the graph [LMQ+19]. We find that AdamGNN can provide a

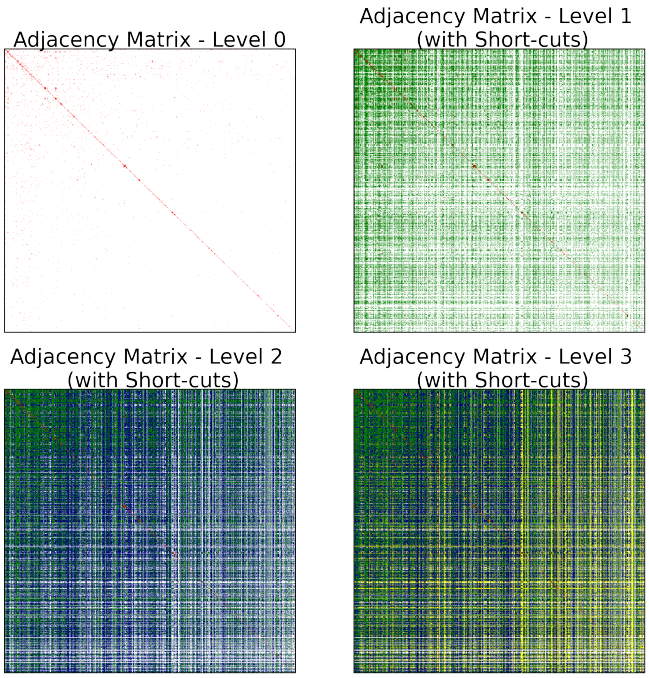


Figure 4.6: Visualisation of the adjacency matrices stacking the 1-st (green), 2-nd (blue), and 3-rd (yellow) granularity levels on the Wiki dataset.

possible solution to overcome this limitation. AdamGNN allows nodes to receive messages from far-away nodes with the support of the adaptive multi-grained structure. That said, the learned multi-grained structure can be regarded as a kind of *short-cuts* to let far-away nodes be aware of each other. We visualise these short-cut connections at different levels on the Wiki dataset in Figure 4.6. Figure-level 0 plots the original adjacency matrix, Figure-level 1 exhibits the learned short-cuts by the first pooled super graph (in green). Similarly, figure-levels 2 and 3 present the derived short-cuts by further stacking the adjacency matrices of the second (in blue) and third (in yellow) pooled graphs, respectively. We can clearly see that the original graph of the Wiki dataset is very sparse, and AdamGNN adds short-cuts between nodes with the help of the learned multi-grained structure. In this way, AdamGNN allows nodes to capture global information with few adaptively pooled graphs.

Table 4.14: Performance comparison for 1-shot-NC task on *Karate*-club dataset.

Model	1-**Layer**	2-**Layers**
GCN	65.26	69.74
AdamGNN (1-level)	88.65	97.91

Exploration of *short-cuts* of AdamGNN. To explore the *short-cuts* derived by AdamGNN, we perform another empirical analysis on one additional network, i.e., the *Karate* [Zac77] club network. We choose 1-shot NC as target task, where we randomly select one sample from each class as training set, an equal number of nodes as validation set and the rest nodes for test. Graph structure, experimental results are summarised in Table 4.14 and two adjacency matrices are shown in Figure 4.7. Short-cuts derived by the first pooled super graph are depicted in green in the level 1 adjacency

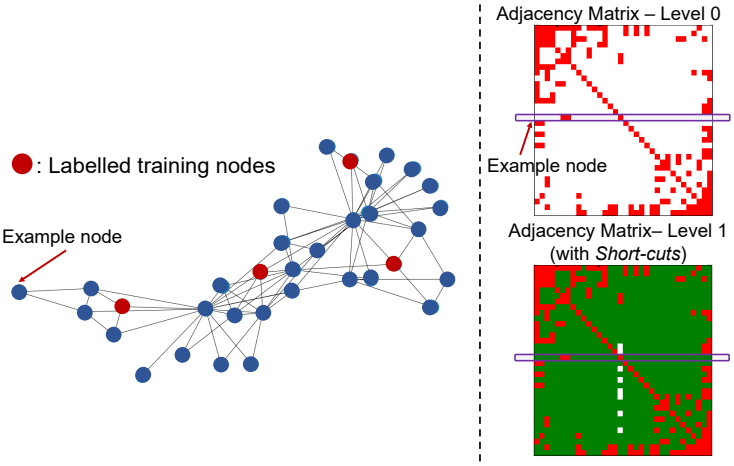


Figure 4.7: Visualisation of graph structure and adjacency matrix at different granularity levels of *Karate*-club dataset

matrix. We find that AdamGNN outperforms GCN on 1-shot NC task with up to 40.5% performance improvements. The two figures in the right part of Figure 4.7 clearly demonstrate that the *short-cuts* derived by AdamGNN make the example node aware of far-away nodes.

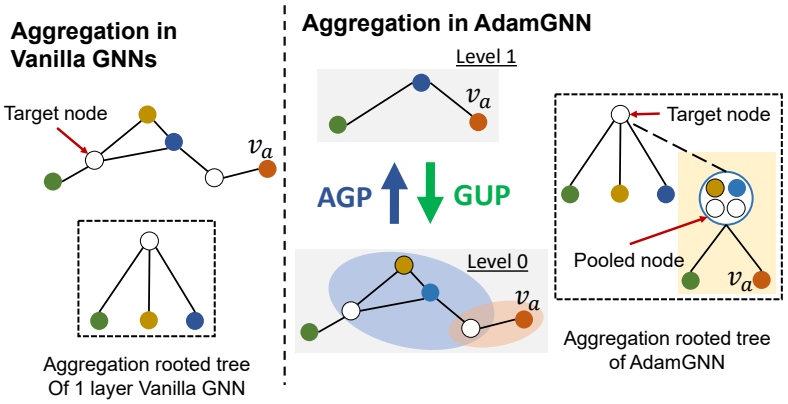


Figure 4.8: A toy example of the aggregation schemas of vanilla GNNs (left) and AdamGNN (right).

Comparing aggregation mechanisms. To demonstrate the internal aggregation mechanism of AdamGNN and figure out the reason that it leads to performance improvements as shown in Section 4.4.2, we give a toy example of the aggregation schemas in vanilla GNNs and AdamGNN. As shown in Figure 4.8, 1-layer vanilla GNN can only capture limited information as presented in the left rooted tree. Nevertheless, thanks to the adaptive hierarchical structure learned by AdamGNN, target nodes can receive multi-grained semantics as well as endowed with the ability to capture information from nodes from a long range. For instance, node v_a's message cannot be obtained by the target node with a few layers of vanilla GNN, but AdamGNN allows the target node to receive v_a's information with 1 granularity level. The level 1's graph allows the super node to receive a message from node v_a and pass it to the target nodes by *flyback* aggregator.

4.5 Conclusion and Future work

To summarise, we proposed AdamGNN, an adaptive hierarchical message-passing graph neural networks model that integrates multi-grained semantics into node representations and realises collective optimisation between node- and graph-wise tasks in one unified process. We have designed an adaptive and efficient pooling operator with a novel ego-network selection approach to encoding the multi-grained structural semantics and a training strategy to overcome the over-smoothing problem. Extensive experiments conducted on 14 real-world datasets showed the promising effectiveness of AdamGNN on node- and graph-wise downstream tasks.

One future direction is to appropriately apply the adaptive multi-grained structure on heterogeneous networks for node and graph level tasks.

Part II

Extending Graph Machine Learning to *Heterophilous* Graphs

Motivation and Summary

Following the triumph of machine learning in computer vision and natural language processing, there are more and more success stories coming from machine learning paradigms suited for graph-structured data. *Graph machine learning* (GML) has become a predominant approach for a multitude of graph analytic tasks. However, these outstanding results of GML approaches are often obtained for the graphs that exhibit only high *homophily*, i.e., the structure where nodes of the same class tend to be connected. In graphs with low homophily, known as *heterophily*, most GML approaches fall short. And addressing the heterophily scenario is essential for network analysis and fairness study.

This part of the thesis explores how we can extend GML approaches to heterophilous graphs with and without supervision.

In Chapter 5, we introduce the efficient and strong *compatible label propagation* (CLP) model for node classification *with* supervision information.

Chapter 6 focuses on more challenging settings: how to learn useful node representations on heterophilous graphs *without* supervision information? We propose a unified model, *self-supervised network embedding* (Selene), to redefine the graph representation learning as a r-ego network discrimination task and introduce a set of designs to accomplish it. We empirically demonstrate that it works well for graphs with both homophily and heterophily.

Chapter 5

Simplifying Node Classification on Heterophilous Graphs

5.1 Introduction

Following the triumph of deep learning in computer vision and natural language processing, more and more success stories are coming from *graph neural networks* (GNNs) suited for relational data such as graphs or meshes [ZCZ20, WPC+21]. The majority of modern deep learning architectures can be considered as a special case of the GNN with specific geometrical structures [BBCV21]. These models have achieved state-of-the-art performance in tasks such as node classification, common in real-world applications, and crested popular leaderboards such as Open Graph Benchmark [HFZ+20]. The landscape of GNNs is rich, and many new architectures have been recently proposed to compensate for limited expressivity [AL20, XHLJ19, DHS+19, VCC+18] or to solve specific problems such as oversmoothing, inherent to the traditional message-passing layers [YHS+21, ZA20, MWW20, LHW18]. Unfortunately, these models attain desiderata with the extra price of being more complex and less intuitive during inspection of their performance gains, therefore restricting their applicability in practice.

To address these problems, several models were proposed recently that do not use message-passing algorithm of GNNs but instead are based on well-studied algorithms that show promising results in graph problems [HHS+21, IP21, RFC+20, TZPM19]. Here, we resort to a graph algorithm called *label propagation* (LP) [ZBL+03, Zhu05] – a competitive algorithm in semi-supervised node classification setup, which was popular for more than a decade. While GNNs learn mapping functions between node features and class labels, LP algorithm directly incorporates class labels of the train nodes to make predictions on the test nodes. As traditional LP algorithm does not use node features (which may contain significant signal about the labels of the nodes), it was recently shown [HHS+21] that by making "base predictions" by a linear network on the node features and then substituting the predictions to the LP algorithm, it is possible to boost the performance up to the results of more complex GNNs. These results, however, are often obtained for the graph datasets that exhibit only high homophily, i.e. structure where neighbouring nodes are likely to have the same class labels. In graphs with low homophily, known as heterophily ("opposites attract"), LP and traditional GNNs fall short and

are often outperformed by simple methods such as multi-layer perceptron.

Motivated by this limitation, several GNN architectures were proposed to make message-passing paradigm work on heterophilous graphs [ZYZ+20, CWH+20, YHS+21, BWSS21]. These models revolve around modifications of neighbourhoods used for aggregation schemes of GNNs to enrich the diversity of labels among neighbours. For example, [ZYZ+20] uses multiple-hop neighbourhoods for the aggregation in GNNs, which in turn provides more complete information about the connectivity of different classes. While such approaches bridge the gap for traditional GNNs on heterophilous graphs, they often do so at the expense of more parameters and longer training time.

Instead, in this chapter, we modify LP algorithm to work well in semi-supervised node classification on heterophilous graphs. We start by conducting an experimental investigation over existing models' micro-level performance, i.e., evaluating the node classification accuracy for node groups with subgraphs of different homophily ratios. The investigation results (as shown in Figure 5.1) demonstrate that recent GNNs designed for heterophilous graphs do not outperform simple neural network model that only relies on node features, i.e., multi-layer perceptron, when the subgraph homophily ratio of a node is low. Inspired by this finding, we propose an efficient framework that relies on base predictions given by a simple neural network and further ameliorate the base predictions with a compatible LP algorithm. In particular, we propose a simple pipeline (CLP) with three main steps (Figure 5.2): (i) base predictions of all nodes are made by a simple neural network purely on the node features; (ii) a global compatibility matrix that computes connectivity of different labels is estimated; and (iii) smoothing of the predictions across neighbours weighted by the compatibility of the class labels is performed. Intuitively, step (i) calculates the class probabilities for the test nodes, while step (ii) defines the weights on edges with which LP algorithm at step (iii) will propagate the class probabilities for each node. While steps (i) and (iii) have been tried independently for semi-supervised classification before [KW17, IP21], it is learning the compatibility matrix at step (ii) that makes a big difference as we show in the experiments. In our theoretical analysis, we show that our approach can be computed via closed-form solution that provides necessary and sufficient conditions for convergence. Empirically, extensive experimental results on a wide variety of benchmarks show the competitive and efficient performance of CLP.

A significant boost in the performance of our method is related to learning a global compatibility matrix between classes. This idea is not new — before the rise of neural networks for semi-supervised learning several algorithms such as DCE [PLG20], ZooBP [EGF+17], LinBP [GGKF15] and FaBP [KKK+11] use compatibility matrix for belief-propagation algorithm. However, all of these methods are motivated by the regularisation framework, where the labelling function minimises some energy objective that does not depend on the node features [Gat14] and were shown to have suboptimal performance to GNNs [PLG20]. More recently, compatibility matrix was used for GNNs in the heterophily setting [ZRR+21] and showed a significant increase in performance. That being said, we find that learning a compatibility matrix from the node features significantly improves the performance of LP on

heterophilous graphs.

Overall, we generalise LP algorithm to arbitrary heterophily assumption, where the commonly used smoothness assumption (homophily) is a special case with the identity matrix acting as the compatibility matrix. In this case, LP is orders of magnitude faster than log-likelihood estimators such as GNNs, and it presents new ways to understand the performance of graph learning through the lens of diffusion-based learning [KKK+11, Gat14, ZBL+03, Zhu05]. For example, the insights of using compatibility matrix and class labels as part of the training can be incorporated into existing GNN models. As such, we hope that the ideas of LP algorithm could be fruitful for other tasks such as node regression, and LP could become a commonly used baseline of graph learning practitioners.

5.2 Additional Related Work

At first, there was rich literature on solving the oversmoothing problem [LHW18] which prevents an increasing number of layers of GNNs without loss of performance (common to deep convolutional nets). After that, with new graph datasets with high heterophily [PWC+20, APK+19] and new theory that connects the oversmoothing problem with the tendency of nodes to connect to the opposite classes [YHS+21], it has become evident that GNNs must incorporate additional knowledge to be suited for heterophilous graphs. Several GNNs were proposed to deal with heterophily setting [CPLM21, BWSS21, ZRR+21]; however, usually improved accuracy of these GNNs is traded with an extra computational cost which makes it hard to scale for large datasets, unlike LP algorithm, which is a simple graph algorithm. A recent approach CPGNN [ZYZ+20] also uses compatibility matrix with LP propagation step; however, there are several notable differences compared to our approach. First, CPGNN adjusts the weight of the message only based on the class of a sending node and compatibility matrix. In turn, we additionally consider the class of a receiving node, which significantly improves the results in our experiments. Second, we provide additional theoretical analysis of our method, giving a closed-form solution and convergence guarantees, which is not available for CPGNN model.

Label propagation for heterophily regime. Perhaps the closest work to ours is [Gat14, GGKF15], where a compatibility matrix is used in the *linearised belief propagation* (LinBP) algorithm. There, a compatibility matrix is provided or estimated via a closed-form solution to minimise a convex energy function and does not use the node features that are crucial in the estimation of the right labelling functions. Several follow-ups aimed to generalise LinBP to various types of heterophily [Pee17] or Markov Random Fields [Gat17]. It was later shown in the experiments that these methods are less effective than GNNs in graphs with node features [PLG20]. In contrast, our method combines two orthogonal sources of information – one from the labelling function learned on the node features and another from LP algorithm that uses known labels together with the graph structure.

5.3 Preliminaries

Problem setup. In this chapter, we focus on the semi-supervised node classification task on a graph $\mathcal{G}$, where $\mathcal{T}_\mathcal{V} \subset \mathcal{V}$ with known class labels y_v for all $v \in \mathcal{T}_\mathcal{V}$. We aim to infer the unknown class labels y_u for all $u \in \mathcal{V} \setminus \mathcal{T}_\mathcal{V}$. In addition, $\mathcal{T}_\mathcal{V}$ is split into two subsets: $\mathcal{T}_\mathcal{V}^t$ and $\mathcal{T}_\mathcal{V}^v$, where $\mathcal{T}_\mathcal{V}^t$ is training set and $\mathcal{T}_\mathcal{V}^v$ works as the validation set for early stopping or parameter fine-tune to prevent overfitting.

In order to give a precise description of the node label relationship of an arbitrary graph, here we introduce and formally define the homophily ratio of a graph.

Definition 3 (Homophily Ratio h)**.** *For an arbitrary graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, its homophily ratio h is determined by the relationship between node labels and graph structure encoded in the adjacency matrix. Recent work commonly use two homophily metrics: edge homophily h_{edge} [ZRR+21] and node homophily h_{node} [PWC+20]. They can be formulated as:*

$$\begin{aligned} h_{edge} &= \frac{|\{(u,v) : (u,v) \in \mathcal{E} \wedge y_u = y_v\}|}{|\mathcal{E}|} \\ h_{node} &= \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{u : u \in \mathcal{N}_v \wedge y_u = y_v\}|}{|\mathcal{N}_v|} \end{aligned} \tag{5.1}$$

where $\mathcal{N}_v$ is the set of adjacent nodes of node v and $|\cdot|$ represents the number of elements of the set. Specifically, h_{edge} evaluates the fraction of edges in a graph that connect nodes that have the same class labels; h_{node} evaluates the overall fraction of neighbouring nodes that have the same class labels. In this chapter, we focus on edge homophily and set $h = h_{edge}$ in the following sections.

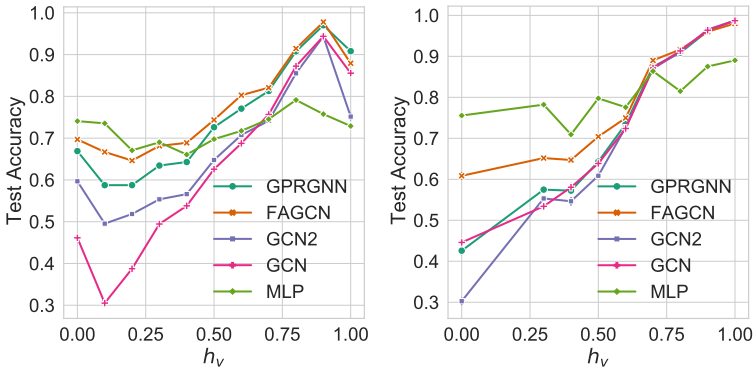
The homophily ratio h defined above is suitable for measuring the *overall* homophily level in the graph. However, the actual homophily level is not necessarily uniform within all parts of the graph. One typical case is that the homophily level varies among different pairs of classes. To measure the variability of the homophily level, we further define the *compatibility matrix* $\mathbf{H}$.

Definition 4 (Compatibility Matrix $\mathbf{H}$)**.** *Let $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{Y}|}$ where $[\mathbf{Y}]_{vj} = 1$ if $y_v = j$ and $[\mathbf{Y}]_{vj} = 0$ otherwise. Then, the compatibility matrix $\mathbf{H}$ is defined as:*

$$\mathbf{H} = (\mathbf{Y}^\top \mathbf{A} \mathbf{Y}) \oslash (\mathbf{Y}^\top \mathbf{A} \mathbf{E}) \tag{5.2}$$

where $\oslash$ is Hadamard (element-wise) division and $\mathbf{E}$ is a $|\mathcal{V}| \times |\mathcal{Y}|$ all-ones matrix.

In the semi-supervised node classification settings, compatibility matrix $\mathbf{H}$ empirically models the probability of nodes belonging to each pair of classes to connect. Modelling $\mathbf{H}$ is crucial for heterophily settings, but calculating the exact $\mathbf{H}$ would require knowledge to class labels of all nodes in the graph, which violates the semi-supervised node classification setting. Therefore it is not possible to incorporate exact $\mathbf{H}$. In Section 5.5.2, we propose an approach to estimate $\mathbf{H}$ based on a sparsely labelled graph, which is utilised after to assist the label propagation step (Section 5.5.3). An empirical study in Section 5.6.5 empirically discusses the quality of estimated $\mathbf{H}$ and its influence on the model performances.



(a) Wiki (b) ACM

Figure 5.1: Classification accuracy for different 1-hop subgraph homophily ratios on Wiki (5.1a) and ACM (5.1b) graphs.

5.4 An Experimental Investigation

In this section, we conduct an empirical study to motivate the design of our approach. The main idea of this experiment is to study how different models perform at the level of an individual node depending on the homophily ratio of the 1-hop subgraphs. We define homophily ratio of an individual node h_v as follows:

$$h_v = \frac{|\{(u,v) : (u,v) \in \mathcal{E}_v \wedge y_u = y_v\}|}{|\mathcal{E}_v|} \tag{5.3}$$

where $\mathcal{E}_v$ is the edge set of the induced 1-hop neighbourhood of v.

We take two graphs as examples: a heterophily graph Wiki with $h = 0.30$ and a homophily graph ACM with $h = 0.82$. We train different models on the training nodes of a graph and compute predictions for the test nodes. We then aggregate the accuracy of predictions for each level of homophily, $\{0, 0.1, \ldots, 0.9, 1\}$, and plot the obtained results in Figure 5.1. Global accuracy across all test nodes can be found in Table 5.3 and Figure 5.4.

Results from Table 5.3 and Figure 5.4 demonstrate that in general GNNs outperform *multi-layer perceptron* (MLP) [Ros61]. However, if we zoom in on local neighbourhoods, as shown in Figure 5.1, the results of MLP are often better than those of GNNs when the homophily ratio of a node's 1-hop subgraph is low.

In particular, we can see from Figure 5.1 that (i) vanilla GCN has superior accuracy for nodes with strong subgraph homophily ratio ($h_v \geq 0.7$) on both graphs; other advanced GNN models mainly improve the classification accuracy over nodes with low h_v, and (ii) MLP is relatively stable across different homophily ratio h_v and is a better model for nodes with low h_v compared to other GNNs. For instance, MLP achieves the best accuracy on nodes with $h_v \leq 0.3$ on Wiki graph and nodes with $h_v \leq 0.6$ on ACM graph.

This illustrates that recent GNN models designed for heterophilous graphs do not outperform MLP for nodes with a considerable fraction of neighbours with opposite class labels; instead, they have better global accuracy than MLP by having

better accuracy on nodes with high homophily ratio h_v. Based on this evidence, we propose a simple but effective approach that mainly relies on the predictions of MLP to maintain its favourable performance on nodes with low homophily and that further ameliorates the classification results by incorporating the knowledge of the graph structure.

5.5 Proposed Approach

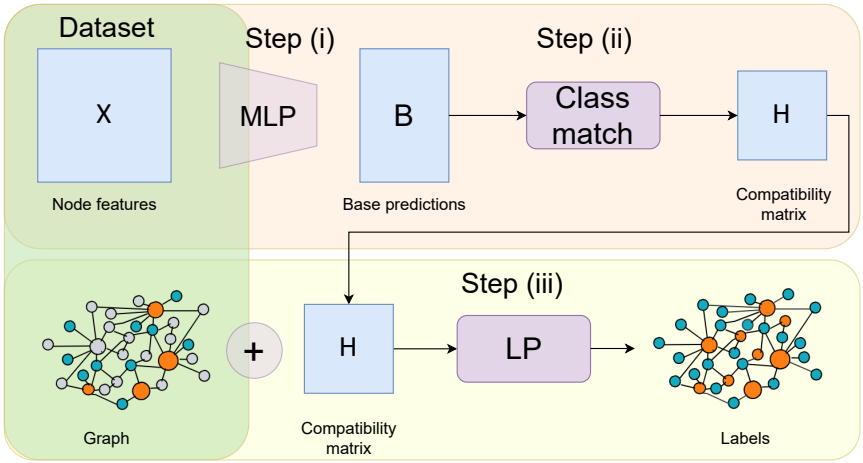


Figure 5.2: Overview of Compatible CLP model. Step (i): base predictor, MLP, makes class predictions for each node using only node features. Step (ii): global compatibility matrix between classes is computed with Equation 5.6. Step (iii): propagate class predictions with LP algorithm and get the classes for test nodes. Intuitively, compatibility matrix measures the weighted probabilities of any two target classes being connected, and as such, it defines the edge weights in LP algorithm.

Our approach starts with a simple base predictor on node features, which does not rely on any learning over the topological structure. Any off-the-shelf graph-agnostic model can be plugged in to become a base predictor, which enables our approach to accommodate any node features. After, we propose an approach to estimate the compatibility matrix $\mathbf{H}$ of the overall graph and apply it to calculate the relation between each pair of nodes. Finally, we use label propagation algorithm with an estimated compatibility matrix to smooth the prior prediction probabilities on the weighted graph to get the final predictions.

5.5.1 Simple Base Predictor

To start, we use a simple base predictor that does not rely on graph structure to learn prior predictions. Specifically, we train a model f_θ to minimise $\sum_{v \in \mathcal{T}_\mathcal{V}^l} \mathcal{L}(f_\theta(\mathbf{x}_v), y_v)$, where $\mathbf{x}_v$ is the contextual feature of node v and y_v is its true class label, $\mathcal{L}$ is a loss function. In this chapter, we adopt a simple *multi-layer perceptron* (MLP) as the base predictor, where ℓ-th layer can be formally formulated as following:

$$\mathbf{D}^{(\ell)} = \sigma(\mathbf{D}^{(\ell-1)}\mathbf{W}^{(\ell)} + \mathbf{b}^{(\ell)}) \tag{5.4}$$

where $\mathbf{W}^{(\ell)}$ are learnable parameters and $\mathbf{b}^{(\ell)}$ is the bias vector. σ is the activation function (e.g. ReLU), and we initialise $\mathbf{D}^{(0)} = \mathbf{X}$.

From f_θ, we get a prior prediction $\hat{\mathbf{D}} = \mathbf{D}^{(\ell)} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{Y}|}$, where each row of $\hat{\mathbf{D}}$ is a probability distribution resulting from the softmax of the last layer of base predictor. Omitting the graph for the prior predictions brings several benefits: (i) it avoids the sensitivity to homophily/heteriophily of the graph (as was shown in Figure 5.5, MLP's performance maintains good stability for graphs with different homophily levels); and (ii) it significantly reduces the number of parameters that we need to learn, thus accelerating the approach (as we shown in Figure 5.8). Next, we use MLP's predictions to estimate the weights for label propagation algorithm.

5.5.2 Estimation of Compatibility Matrix

The focal idea of compatibility matrix is summarising the relative frequencies of classes between neighbours. Under the semi-supervised node classification settings, we only know the class labels of a small fraction of nodes ($\mathcal{T}_\mathcal{V}^t$). We derive the preliminary class labels of unknown nodes ($\mathcal{V} - \mathcal{T}_\mathcal{V} + \mathcal{T}_\mathcal{V}^v$) as the base prediction $\hat{\mathbf{D}}$. Note that we treat validation set nodes as unknown nodes, which will be used to evaluate the performance of LP step and pick up the better final predictions. More specifically, denote the training mask $\mathbf{M}$ as: $[\mathbf{M}]_{i,:} = \begin{cases} \mathbf{1}, & \text{if } i \in \mathcal{T}_\mathcal{V}^t \\ \mathbf{0}, & \text{otherwise} \end{cases}$. The preliminary knowledge of class labels can be formally represented as:

$$\hat{\mathbf{B}} = \mathbf{M} \circ \mathbf{Y} + (1 - \mathbf{M}) \circ \hat{\mathbf{D}} \tag{5.5}$$

where $\circ$ is the *Hadamard* (element-wise) product.

Next, we define a compatibility matrix $\hat{\mathbf{H}}$ that calculates the probability that a node of one class is connected with a node of another class.

$$\hat{\mathbf{H}} = \mathcal{S}((\mathbf{M} \circ \mathbf{Y})^\top \mathbf{A}\hat{\mathbf{B}}) \tag{5.6}$$

where $\mathcal{S}$ is the Sinkhorn-Knopp function that ensures $\hat{\mathbf{H}}$ is doubly-stochastic [SK67, PLG20].

A compatibility matrix $\hat{\mathbf{H}}$ can be seen as a multiplication of two matrices, $(\mathbf{M} \circ \mathbf{Y})^\top$ and $\mathbf{A}\hat{\mathbf{B}}$. The matrix $(\mathbf{M} \circ \mathbf{Y})^\top$ represents one-hot encoded labels of training nodes only. In turn, the matrix $\mathbf{A}\hat{\mathbf{B}}$ computes the sum of class probabilities across all neighbours of each node. After multiplication of these two matrices, each entry (i, j) of $(\mathbf{M} \circ \mathbf{Y})^\top \mathbf{A}\hat{\mathbf{B}}$ represents a score that a class i among training nodes is connected with a node of class j estimated with prior probabilities $\hat{\mathbf{D}}$. A function $\mathcal{S}$ converts these scores back to probabilities such that each entry (i, j) of $\hat{\mathbf{H}}$ indicates a probability that a class i is connected with class j.

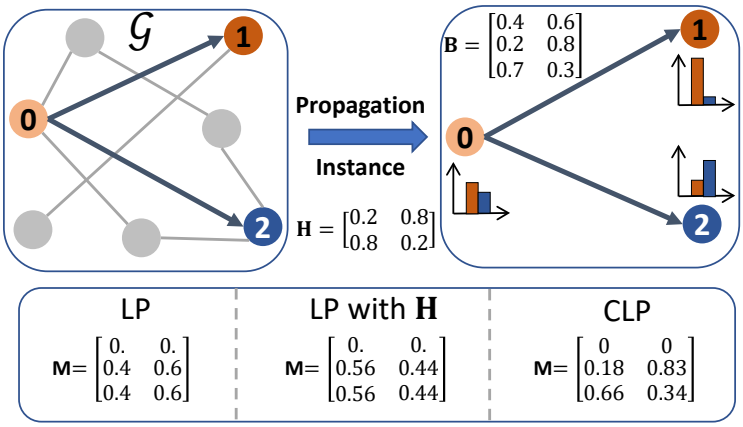


Figure 5.3: Comparison of three propagation schemes, $\mathbf{M}$ represents the received messages after one propagation iteration. In LP nodes **1** and **2** receive the same message; LP with $\mathbf{H}$ overturns prior prediction of node **1**; CLP adapts the heterophily of the graph and reassures confident prior predictions.

5.5.3 Compatible Label Propagation

After obtaining the estimation $\hat{\mathbf{H}}$, we propagate the knowledge about node class labels with the guide of $\hat{\mathbf{H}}$ over the graph. The key idea of our method is that the edge weight of a message $u \mapsto v$ in label propagation algorithm depends on both predicted classes of sending and receiving nodes. That contrasts previous works [GGKF15, ZRR+21] where edge weight depends only on the sending node class probabilities. In particular, for each edge (i, j), we define an edge weight as follows:

$$[\mathbf{F}]_{ij} = ([\hat{\mathbf{B}}]_i \hat{\mathbf{H}}) \circ [\hat{\mathbf{B}}]_j \tag{5.7}$$

Intuitively, edge weight $[\mathbf{F}]_{ij}$ depends on the probabilities that node i is connected with some class k, $([\hat{\mathbf{B}}]_i \hat{\mathbf{H}})$, and the probabilities that node j has the same class k. Naturally, we can assign the edge weights to corresponding positions of adjacent matrix to get $\mathbf{A}^{\mathbf{F}} \in \mathbb{R}^{n \times n \times |\mathcal{Y}|}$, where $[\mathbf{A}]^{\mathbf{F}}_{ij} = [\mathbf{F}]_{ij}$.

Let $\hat{\mathbf{B}}$ and $\hat{\mathbf{D}}$ be the final node class prediction matrix and the base prediction, respectively. $\mathbf{A}^{\mathbf{F}}$ is the weighted adjacent matrix. Then, the final node classifications are approximated by the equation system:

$$\hat{\mathbf{B}} = (1 - \alpha)\hat{\mathbf{D}} + \alpha\, \mathbf{A}^{\mathbf{F}} \oplus \hat{\mathbf{B}} \tag{5.8}$$

where $\mathbf{A}^{\mathbf{F}} \oplus \hat{\mathbf{B}}^{(l)} = \mathop{\Big\|}_{k \in |\mathcal{Y}|} \mathbf{A}^{\mathbf{F}}_k\, [\hat{\mathbf{B}}^{(l)}]_{:,k}$ and $\mathbf{A}^{\mathbf{F}}_k$ means the weighted adjacent matrix with k-th dimensional edge weights. α is a hyper-parameter, which defines how much update to the previous state each label propagation step makes.

Iterative updates. Notice that Equation 5.8 gives an implicit definition of the final node classification after convergence, it can also be used as iterative update equations, allowing an iterative calculation of the final node classification predictions:

$$\hat{\mathbf{B}}^{(l+1)} \leftarrow (1 - \alpha)\hat{\mathbf{D}} + \alpha \mathbf{A}^{\mathbf{F}} \oplus \hat{\mathbf{B}}^{(l)} \tag{5.9}$$

Thus, the final node classification predictions can be computed via linear matrix operations. Note that previous works [GGKF15, ZRR+21] compute the compatibility matrix $\hat{\mathbf{H}}$ for LP as follows:

$$\hat{\mathbf{B}}^{(l+1)} \leftarrow (1-\alpha)\hat{\mathbf{D}} + \alpha \mathbf{A}\hat{\mathbf{B}}^{(l)}\hat{\mathbf{H}} \tag{5.10}$$

Equation 5.10 defines an edge weight by the relation between the sending node and $\hat{\mathbf{H}}$. Hence, receiving nodes get the same message from a sending node regardless of the class of the receiving nodes. We argue that the proper weight of a message should be determined by both sending and receiving nodes (Figure 5.3).

5.5.4 Theoretical Analysis of CLP

Equation 5.9 allows solving CLP Equation 5.8 via iterative updates. Here, we show an alternative method that provides a closed-form solution, which in turn sets convergence guarantees of CLP. We start by defining vectorisation of a matrix $\mathbf{X}$, which stacks columns of $\mathbf{X}$ side-by-side.

Definition 5 (Matrix Vectorization [HVH81])**.** *Vectorization of an $m \times n$ matrix $\mathbf{X}$ is an $mn \times 1$ vector given by:*

$$vec(\mathbf{X}) = [\mathbf{x}_{11}, \ldots, \mathbf{x}_{n1}, \mathbf{x}_{12}, \ldots, \mathbf{x}_{n2}, \ldots, \mathbf{x}_{1n}, \ldots, \mathbf{x}_{nn}]^{\mathrm{T}} \tag{5.11}$$

Additionally, the Kronecker product of $\mathbf{X}$ and $\mathbf{Q}$ is the $mp \times nq$ matrix is defined by:

$$\mathbf{X} \otimes \mathbf{Q} = \begin{bmatrix} \mathbf{x}_{11}\mathbf{Q} & \mathbf{x}_{12}\mathbf{Q} & \ldots & \mathbf{x}_{1n}\mathbf{Q} \\ \mathbf{x}_{21}\mathbf{Q} & \mathbf{x}_{22}\mathbf{Q} & \ldots & \mathbf{x}_{2n}\mathbf{Q} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{m1}\mathbf{Q} & \mathbf{x}_{m2}\mathbf{Q} & \ldots & \mathbf{x}_{mn}\mathbf{Q} \end{bmatrix} \tag{5.12}$$

We are now ready to give a closed-form solution to Equation 5.8:

Proposition 2 (Closed-form CLP)**.** *The closed-form solution for CLP (Equation 5.8) is given by:*

$$vec(\hat{\mathbf{B}}) = (\mathbf{I} - \alpha\ (\mathbf{I} \otimes \mathbf{A}^{\mathbf{F}}))^{-1}(1-\alpha)\ vec(\hat{\mathbf{D}}) \tag{5.13}$$

Proof of Proposition 2. We start by considering only k-th class in Equation 5.8. In this case, we can rewrite Equation 5.8 as:

$$\begin{aligned} Equation\ 5.8 : \hat{\mathbf{B}} &= (1-\alpha)\hat{\mathbf{D}} + \alpha\ \mathbf{A}^{\mathbf{F}} \oplus \hat{\mathbf{B}} \\ Equation\ 5.8 : \hat{\mathbf{B}} &= (1-\alpha)\hat{\mathbf{D}} + \alpha \mathop{\Big\|}_{k \in |\mathcal{Y}|} \mathbf{A}^{\mathbf{F}}_k\ [\hat{\mathbf{B}}]_{:,k} \\ Equation\ 5.8 : [\hat{\mathbf{B}}]_{:,k} &= (1-\alpha)[\hat{\mathbf{D}}]_{:,k} + \alpha\ \mathbf{A}^{\mathbf{F}}_k [\hat{\mathbf{B}}]_{:,k} \end{aligned} \tag{5.14}$$

We then can apply Roth's column lemma [HVH81, GGKF15] to obtain a closed-form solution:

$$\begin{aligned}
\hat{\mathbf{B}} &= (1-\alpha)\hat{\mathbf{D}} + \alpha\mathbf{A}^{\mathbf{F}}\hat{\mathbf{B}} \\
vec(\hat{\mathbf{B}}) &= (1-\alpha)\ vec(\hat{\mathbf{D}}) + \alpha\ vec(\mathbf{A}^{\mathbf{F}}\hat{\mathbf{B}}) \\
vec(\hat{\mathbf{B}}) - \alpha\ vec(\mathbf{A}^{\mathbf{F}}\hat{\mathbf{B}}) &= (1-\alpha)\ vec(\hat{\mathbf{D}}) \\
vec(\hat{\mathbf{B}}) - \alpha\ (\mathbf{I}\otimes\mathbf{A}^{\mathbf{F}})vec(\hat{\mathbf{B}}) &= (1-\alpha)\ vec(\hat{\mathbf{D}}) \\
(\mathbf{I} - \alpha\ (\mathbf{I}\otimes\mathbf{A}^{\mathbf{F}}))vec(\hat{\mathbf{B}}) &= (1-\alpha)\ vec(\hat{\mathbf{D}}) \\
vec(\hat{\mathbf{B}}) &= (\mathbf{I} - \alpha\ (\mathbf{I}\otimes\mathbf{A}^{\mathbf{F}}))^{-1}(1-\alpha)\ vec(\hat{\mathbf{D}}) \qquad (5.15)
\end{aligned}$$

□

Therefore, instead of iterative updates Eq. 5.10, we can compute the final node predictions in a closed-form by using Eq. 5.13, as long as the inverse of the matrix exists. Based on this closed-form solution we next establish necessary and sufficient criteria of convergence.

Convergence of iterative CLP. We remind that spectral radius of a matrix $\mathbf{X}$ is the maximum eigenvalue, i.e. $\rho(\mathbf{X}) = \max(\{|\lambda_1|, \ldots, |\lambda_n|\})$. With Equation 5.13 we are now ready to establish convergence guarantees for CLP.

Proposition 3 (Convergence of CLP)**.** *CLP iterative updates Equation 5.13 converge if and only if* $\rho(\mathbf{A}^{\mathbf{F}}) < \alpha^{-1}$.

Proof of Proposition 3. First, following the Jacobi method [Saa81], we notice that the solution to Equation 5.13 can be expressed via an iterative form:

$$vec(\hat{\mathbf{B}}^{(l+1)}) \leftarrow (1-\alpha)\ vec(\hat{\mathbf{D}}) + \alpha\mathbf{A}^{\mathbf{F}}vec(\hat{\mathbf{B}}^{(l)}) \qquad (5.16)$$

These equations converge for any choice of the initial value of $\mathbf{B_0}$ if and only if the spectral radius of matrix $\alpha\mathbf{A}^{\mathbf{F}}$ is less than one. Moreover, from Proposition 2 we know that the convergence guarantees for CLP Equation 5.8 are equivalent to the closed-form solution Equation 5.13. Hence, Equation 5.16 converges if and only if the spectral radius of $\rho(\alpha\mathbf{A}^{\mathbf{F}}) = \alpha\rho(\mathbf{A}^{\mathbf{F}}) < 1 \iff \rho(\mathbf{A}^{\mathbf{F}}) < \alpha^{-1}$. □

As computing largest eigenvalue may be too expensive for large graphs, following the Gershgorin circle theorem [Wei03], one can replace the spectral norm with any sub-multiplicative norm that is faster to compute and give an upper bound to the spectral radius. For $\| \mathbf{X} \|_p = (\sum_i \sum_j |\mathbf{X}(i,j)|^p)^{1/p}$ we have $\rho(\mathbf{X}) \leq \| \mathbf{X} \|_2 \leq \| \mathbf{X} \|_1$. Hence, one can use Frobenius or 1-induced norm to obtain sufficient convergence guarantees faster. In our experiments we found that CLP converges efficiently for all datasets.

5.5.5 Summary

To review our approach, we start with a base predictor, which purely learns from node features to make node class label predictions. Next, we estimate the global

compatibility matrix $\hat{\mathbf{H}}$ based on the sparsely labelled graph and base predictions. $\hat{\mathbf{H}}$ describes the overall possibility of nodes belonging to each pair of classes to connect, which can be utilised to estimate the relationship between each pair of base prediction vectors. Finally, we perform an efficient LP step to smooth base predictions and obtain labels with the assistance of the relationship between each pair of nodes.

Compared with existing GNN models, CLP similarly benefits from both node features and graph structure, yet separates them into two processes. It is motivated by the investigation in Section 5.4 that MLP has better accuracy over other GNN models for nodes with low homophily. Hence we would like to maintain MLP's advantages and utilise graph structure to improve it to obtain final predictions. Following this way, both node features and graph structure are appropriately involved in our approach, and it only requires learning parameters specified by a base predictor.

Before showing that CLP achieves competitive performances on node classification tasks, we briefly describe another simple way of improving performance by normalising the received messages of each node: $\hat{\mathbf{B}} = (1-\alpha)\hat{\mathbf{D}} + \alpha\ \mathcal{F}(\mathbf{A}^{\mathbf{F}} \oplus \hat{\mathbf{B}})$, where $\mathcal{F}$ is a function that guarantees that each message is a probability distribution. We empirically verified that $\mathcal{F}$ can additionally boost the performance practically.

5.6 Evaluation

To validate our approach's effectiveness, we first empirically demonstrate the performance of CLP and competing models on real-world and synthetic datasets with a wide variety of settings. Second, we compare the number of required parameters, the quality of compatibility estimation, the models' execution time, and their performance on different graphs with different label rates. Third, we empirically show the advantages of our propagation method compared with the previous design. We also study the influence of different label rates on the compatibility matrix estimation and classification accuracy and show the efficiency of CLP in terms of the model size.

5.6.1 Datasets

Real-world datasets. We use a total of 19 real-world datasets (Texas, Wisconsin, Actor, Squirrel, Chameleon, USA-Airports, Brazil-Airports, Wiki, Cornell, Europe-Airports, deezer-europe, Twitch-EN, Twitch-RU, Ogbn-Proteins, WikiCS, DBLP, CS, ACM, Physics) in diverse domains (web-page, citation, co-author, flight transport, biomedical and online user relation). Note that we use ROC-AUC as the evaluation metric for the class imbalanced datasets, i.e., Twitch-EN, Twitch-RU and Ogbn-Proteins, following [LHL+21].

Synthetic datasets. We generate random synthetic graphs with various homophily levels h and node features by adopting a similar approach [APK+19, KO21] but

Table 5.1: Statistics of real-world graphs. $|\mathcal{V}|$: number of nodes; $|\mathcal{E}|$: number of edges; π: dimensionality of nodes features; OSF: nodes only have structure related features; d_{avg}: average degree; $|\mathcal{Y}|$: number of possible class labels; h: homophily ratio; Split: split approach of graph; Directed: if the graph is directed.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	π	**OSF**	$\lvert\mathcal{Y}\rvert$	d_{avg}	h	**Split**	**Directed**
Texas	183	325	1,703	False	5	1.7760	0.061	Random	True
Wisconsin	251	515	1,703	False	5	2.0518	0.170	Random	True
Actor	7,600	30,019	932	False	5	3.9499	0.216	Random	True
Squirrel	5,201	198,493	2,089	False	5	76.3288	0.223	Random	False
Chameleon	2,277	31,421	2325	False	5	27.5986	0.234	Random	False
USA-Airports	1,190	13,599	1	True	4	22.8555	0.251	Random	False
Brazil-Airports	131	2077	1	True	4	16.3969	0.286	Random	False
Wiki	2,405	12,761	4973	False	17	10.6121	0.298	Random	False
Cornell	183	298	1,703	False	5	1.6284	0.298	Random	True
Europe-Airports	399	11988	1	True	4	30.0501	0.309	Random	False
deezer-europe	28,281	185,504	31,241	False	2	6.5593	0.525	Random	False
Twitch-EN	7,126	77,774	2,514	False	2	17.2035	0.556	Random	True
Twitch-RU	4,385	37,304	2,224	False	2	16.1232	0.618	Random	True
WikiCS	11,701	291,039	300	False	10	24.8730	0.691	Random	True
DBLP	4,057	3,528	334	False	4	1.7397	0.799	Random	False
CS	18,333	163,788	6,805	False	15	8.9341	0.808	Random	False
ACM	3,025	13,128	1,870	False	3	8.6797	0.821	Random	False
Physics	34,493	495,924	8415	False	5	14.3775	0.931	Random	False
Ogbn-Proteins	132,534	39,561,252	8	False	2	6.5593	0.623 - 0.940	Fixed	False

Table 5.2: Statistics for synthetic Datasets. (Prod) means node features come from Ogbn-Products [HFZ+20], or adopt the statistic features designed by 2D Gaussians [APK+19].

Benchmark Name	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{Y}\rvert$	h	d_{avg}
Syn-(Prod)-1	$10,000$	$49,446$ to $50,352$	10	[0, 0.1, ..., 1]	4.95 to 5.02
Syn-(Prod)-2	$10,000$	$99,556$ to $99,556$	10	[0, 0.1, ..., 1]	9.96 to 10.01
Syn-(Prod)-3	$10,000$	$149,090$ to $15,1494$	10	[0, 0.1, ..., 1]	14.91 to 15.15

with some modifications. For instance, synthetic graphs [APK+19] have no available contextual node attributes. Specifically, each synthetic graph has 10 classes and $1,000$ nodes per class. Nodes are assigned random features sampled from 2D Gaussians (Syn) or features from real-world datasets [HFZ+20] (Syn-Prod). Except for homophily ratio, we also control the average degree of each graph (around 5, 10 or 15) to investigate the performance with respect to graph sparsity. Here, we give detailed descriptions of the generation process.

Graph generation. We generate synthetic graph $\mathcal{G}$ of $|\mathcal{V}|$ nodes with $|\mathcal{Y}|$ different class labels, and $\mathcal{G}$ has $|\mathcal{V}|/|\mathcal{Y}|$ nodes per class. $|\mathcal{V}|$ and $|\mathcal{Y}|$ are two prescribed numbers to determine the size of $\mathcal{G}$. Synthetic graph's homophily ratio h is mainly controlled by p_{in} and p_{out}, where p_{in} means the possibility of existing an edge between two nodes with the same label and p_{out} is the possibility of existing an edge between two nodes with different labels. Furthermore, the average degree of $\mathcal{G}$ is $d_{avg} = |\mathcal{V}|/|\mathcal{Y}| \cdot \delta$, where $\delta = p_{in} + (|\mathcal{Y}| - 1) \cdot p_{out}$. Following the described graph generation process, with given $|\mathcal{V}|$, $\mathcal{Y}$ and d_{avg}, we choose p_{in} from $\{0.0001\delta, 0.1\delta, 0.2\delta, \ldots, 0.9\delta, 0.9999\delta\}$. Note that the synthetic graph generation process requires both p_{in} and p_{out} are positive numbers, hence we use $p_{in} = 0.0001\delta$ and 0.9999δ to estimate $h = 0$ and $h = 1$ cases, respectively.

Node features generation. In order to comprehensively evaluate the performances of

different models, we assign each node with statistic features (Syn) or real-world contextual node features (Syn-Prod). For graphs with statistic node features, the feature values of nodes are sampled from 2D Gaussian [APK+19]. The mean of Gaussian can be described in polar coordinates: each means has radius 300 and angle $\frac{2\kappa}{10} \times (class\, id)$. For datasets with real-world contextual node features, we first establish a class mapping $\psi : \mathcal{Y} \to \mathcal{Y}_b$ between classes in the synthetic graph $\mathcal{Y}$ to classes of existing benchmark graph $\mathcal{Y}_b$. The only requirement for the target graph dataset is that the class size and node set size in the benchmark is larger than that of the synthetic graph, i.e., $|\mathcal{Y}|_b \leq |\mathcal{Y}|$ and $|\mathcal{V}| \leq |\mathcal{V}|_b$. In this chapter, we adopt the large-scale benchmark, Ogbn-Products [HFZ+20].

5.6.2 Experimental Setup

Competing methods. We compare our model against state-of-the-art graph neural networks and related node classification methods for all datasets under fair settings. Specifically, MLP [Ros61] is the competing model that only utilises node attributes, while LINK [ZG09] only utilises graph structure. Meanwhile, we also adopt general GNN models with underlying homophily assumption: GCN [KW17], GAT [VCC+18] and GCN2 [CWH+20]. Moreover, we adopt several models that are designed for heterophily graphs: Mixhop [APK+19], SuperGAT [KO21], GPRGNN [CPLM21], FAGCN [BWSS21], H2GCN [ZYZ+20] and CPGNN [ZRR+21]. At last, two LP-based models: LP [Zhu05] and C&S [HHS+21].

Implementation and splits. We follow the experimental setup of FAGCN and CPGNN with minor adjustments. Specifically, our experimental setup examines the semi-supervised node classification in the transductive setting. We consider four different choices for the random split into training/validation/test settings, which we call *sparse* splittings (5%/5%/90%), *medium* splitting (10%/10%/80%) and *dense* splitting (48%/32%/20%), respectively. The *sparse* splitting (5%/5%/90%) is similar to the original semi-supervised setting in [KW17], but we do not restrict each class to have the same number of training instances since it is the case closer to the real-world application. For a fair comparison, we generate 10 fixed split instances with different splitting and results are summarised after 10 runs with random seeds. Note that Ogbn-Proteins dataset adopts its default splitting settings.

5.6.3 Results on Real-world Graphs

Real-world graphs with *heterophily*. The performance of diverse methods on heterophily graphs is summarised in Table 5.3, top-2 performances of each graph are highlighted in colour. Advanced GNN models that are designed for heterophily graphs generally perform better than GNNs designed with high-homophily assumption. MLP, which only utilise node features, achieves outstanding performances in several benchmarks. Our model, CLP, inherits the advantage of MLP but also benefits from graph structure, and it achieves outstanding and stable performance on all heterophily graphs. Moreover, many competing methods lead to

Table 5.3: Summary of node classification results on *heterophily* graphs under *medium* splitting. ‡ indicates the results from [LHL+21]. The bold numbers represent the top-2 results.

Hom.R h	Texas 0.06	Wisc. 0.17	Actor 0.22	Squirrel 0.22	Cham. 0.23	USA-A. 0.25	Bra.-A. 0.29	Wiki 0.30	Cornell 0.30	Eu.-A. 0.31	deezer 0.53	Tw.-EN 0.60	Tw.-RU 0.639	O.-Proteins –
MLP	**67.94**	69.32	32.07	26.18	35.94	54.92	**59.52**	70.13	68.19	**50.41**	63.77	59.56	49.33	73.43‡
LINK	59.52	47.79	24.03	46.02	58.28	24.71	27.97	25.07	46.47	29.59	55.95	55.65	51.27	63.49‡
GCN	54.17	47.55	26.82	24.71	34.61	30.88	26.84	53.15	55.81	31.65	59.94	59.79	51.51	72.03‡
GAT	54.12	48.73	27.37	24.55	36.6	28.13	23.76	47.21	55.18	24.34	56.22	58.66	51.65	OOM‡
GCN2	55.22	47.63	27.14	25.5	36.26	36.59	27.22	60.29	53.87	35.05	62.33	59.66	51.53	74.10
Mixhop	54.62	51.63	27.46	27.81	38.14	52.68	44.41	61.74	51.29	45.55	64.16	60.38	52.54	**75.60**‡
SuperGAT	54.88	49.94	26.69	24.88	35.49	27.02	23.47	33.23	54.47	24.63	57.07	59.66	50.95	OOM
GPRGNN	55.31	50.89	27.72	25.29	34.67	41.83	24.85	68.02	55.03	31.47	62.74	59.42	51.17	OOM‡
FAGCN	60.95	63.08	32.6	24.93	36.68	**56.14**	48.19	**72.12**	62.32	48.22	**65.04**	**60.76**	50.19	OOM
H2GCN	61.29	65.67	32.27	26.95	36.93	54.24	38.95	70.57	57.26	40.56	62.82	59.06	51.22	OOM‡
CPGNN	62.95	**70.05**	**32.42**	**28.7**	**47.7**	25.21	27.51	70.18	**68.04**	34.86	64.95	57.07	**52.37**	OOM
LP	15.58	11.4	17.69	17.59	20.62	24.35	24.48	23.89	18.51	27.2	55.44	54.42	51.9	75.14‡
C&S	66.9	67.34	31.94	26.85	26.85	45.26	55.33	71.49	67.04	37.32	63.92	59.36	52.12	71.13‡
CLP (Ours)	**69.63**	**72.64**	**33.1**	**31.76**	**43.29**	**56.3**	**63.53**	**74.08**	**70.36**	**53.83**	**65.69**	**60.81**	**52.78**	**75.73**

out-of-memory (OOM) issues on the large dataset, i.e., Ogbn-Proteins, but CLP avoids this problem, demonstrating its memory efficiency.

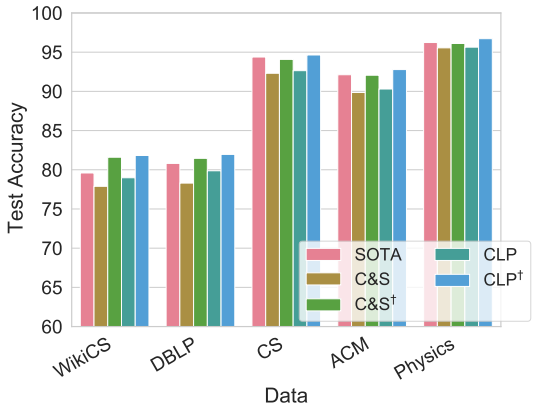


Figure 5.4: Performance comparison of C&S and CLP with best performance of GNN models (SOTA) on *homophily* graphs under *medium* splitting.

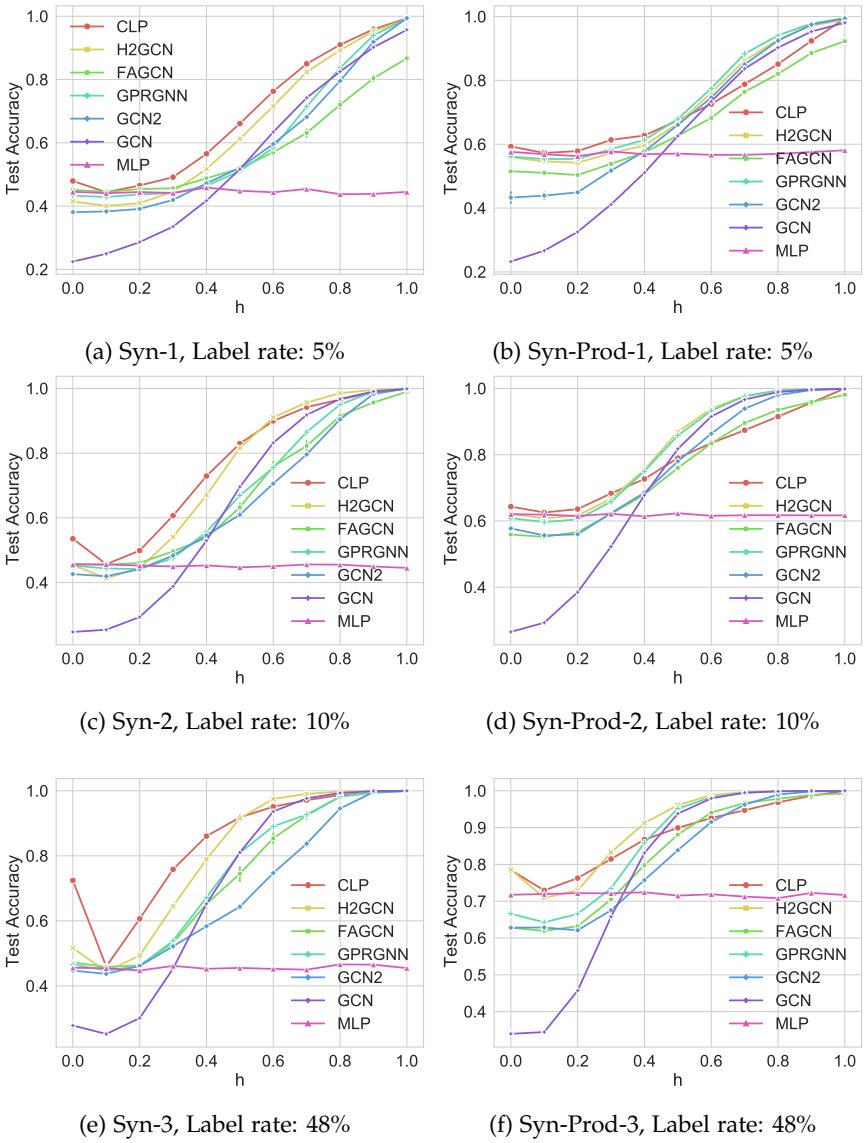
Real-world graphs with *homophily*. Performance on homophily graphs is summarised in Figure 5.4. Inspired by [HHS+21], we further adopt the spectral and diffusion features as additional raw node features to C&S and CLP and compare their performances with the best performance of competing models (SOTA). C&S† and CLP† refer to performance with additional raw features and the figure demonstrates that CLP† outperforms or matches the SOTA on *homophily* graphs.

5.6.4 Results on Synthetic Graphs

Synthetic graphs *without* node features. Most previous work [KW17, BWSS21, ZYZ+20] on semi-supervised node classification have focused only on graphs that have contextual features on the nodes. However, the vast majority of graph data does not have node-level features in practical applications, which greatly limits the utility of methods proposed in prior work. Besides, several components of our

approach depend on node features, for instance, the compatibility matrix estimation ($\hat{\mathbf{H}}$) relies on the prior predictions which are learned from node features. $\hat{\mathbf{H}}$ plays crucial role in the following LP step. Therefore, it's natural to ask how CLP performs over graphs without contextual node features compared with other competitive models?

We conduct extensive experiments on semi-supervised node classification with *sparse*, *medium* and *dense* splittings on three synthetic datasets with different average degrees. For instance, Syn-1 dataset contains 11 graphs with h from 0 to 1, and the average degree per graph is set to around 5 (4.95 to 5.02). Syn-2 and Syn-3 follow similar settings but their average degree of each graph is set to 10 and 15, respectively.



(a) Syn-1, Label rate: 5%

(b) Syn-Prod-1, Label rate: 5%

(c) Syn-2, Label rate: 10%

(d) Syn-Prod-2, Label rate: 10%

(e) Syn-3, Label rate: 48%

(f) Syn-Prod-3, Label rate: 48%

Figure 5.5: Classification accuracy of different methods with different label rates on synthetic datasets. Only competitive results are presented due to the space limit.

We present results of three synthetic datasets in Figure 5.5-(a, b, c). We observe similar trends on three figures: CLP has the best trend overall, outperforming competing methods in heterophily settings while matching with other competing methods in homophily settings. The performance of vanilla GCN and GCN2 increase with respect to the homophily level ($h \to 1$). But, while synthetic graphs have no contextual node features, MLP is more accurate than them under strong heterophily ($h \to 0$). From Figure 5.5, we can find that the classification accuracy of MLP has been stable at about 45%, a relatively low level. Yet, CLP can still achieve the overall best performance. Overall, it indicates that our approach works for graphs without contextual features.

Synthetic graphs *with* node features. We perform extensive experiments on graphs with contextual features to further validate the performance of CLP under various settings. Similar to the experiments on synthetic graphs *without* node features, there are three synthetic graphs, i.e., Syn-Prod-1, Syn-Prod-2 and Syn-Prod-3, which have the same graph structure as Syn-1, Syn-2 and Syn-3, but with contextual node features. Experimental results are presented in Figure 5.5-(d, e, f). These figures emphasise that CLP is the best model for most heterophily cases ($h \to 1$), which again confirms the effectiveness of our approach. It echoes the results of the real-world graphs (Table 5.3). Besides, GCN and GCN2, which was proposed with implicit homophily assumption, are significantly less accurate than MLP (near-flat performance curve as it is graph-agnostic) under strong heterophily ($h \leq 0.4$). Such evidence can be found in some cases for other heterophilous GNN models (H2GCN, FAGCN, GPRGNN). For instance, they perform significantly better than GCN but are outperformed by MLP on Syn-Prod-1 under $h \leq 0.3$ (Figure 5.5b). It reaffirms what we found in Section 5.4, i.e. MLP could be a better choice for making classification for strong heterophily node groups. Our approach, CLP, can always achieve better performance than MLP in graphs with any heterophily levels and sparsity levels.

5.6.5 More Analysis

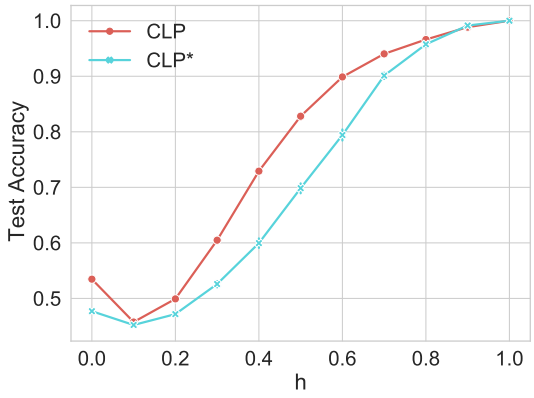


Figure 5.6: Performance comparison of CLP and CLP* on Syn-1 dataset with *medium* splitting.

Comparison between two propagation schemes. In Section 5.5.3, we explained the design of our compatible LP process and discussed its advantages over prior work [ZYZ+20]. The messages between two nodes are adaptively determined by nodes of both ends. Here, we perform extensive experiments to empirically compare the performance of LP steps with two propagation schemes. We choose one synthetic dataset with 11 graphs under various homophily (Syn-1) under the medium splitting. Other settings follow the common setup of CLP. The approach that utilises Equation 5.10 named as CLP*. Their performances are reported in Figure 5.6. We observe that CLP has the better trend overall, outperforming CLP* in most heterophily settings ($h \leq 0.9$) and matching with CLP* in other settings.

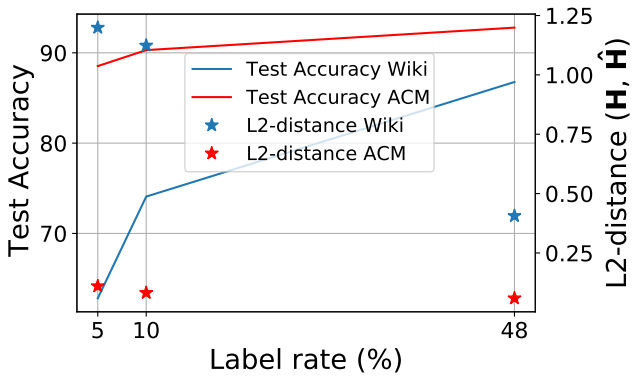


Figure 5.7: Classification accuracy and L2-distance between estimated/true compatibility matrix with different label rates.

Influence of label rate on test accuracy and quality of compatibility matrix estimation. Figure 5.7 presents the CLP's test accuracy and the quality of compatibility matrix estimation ($\hat{\mathbf{H}}$) with different splittings. Specifically, the quality of $\hat{\mathbf{H}}$ is evaluated by $dist(\mathbf{H}, \hat{\mathbf{H}}) = \sqrt{\sum_{i=1}^{|\mathcal{Y}|} \sum_{j=1}^{|\mathcal{Y}|} \left([\mathbf{H}]_{ij} - [\hat{\mathbf{H}}]_{ij}\right)^2}$. It is not surprising to find that higher label rates lead to better performance and more accurate compatibility matrix estimation. Therefore, one of the future directions is to learn better compatibility matrix estimation according to prior predictions.

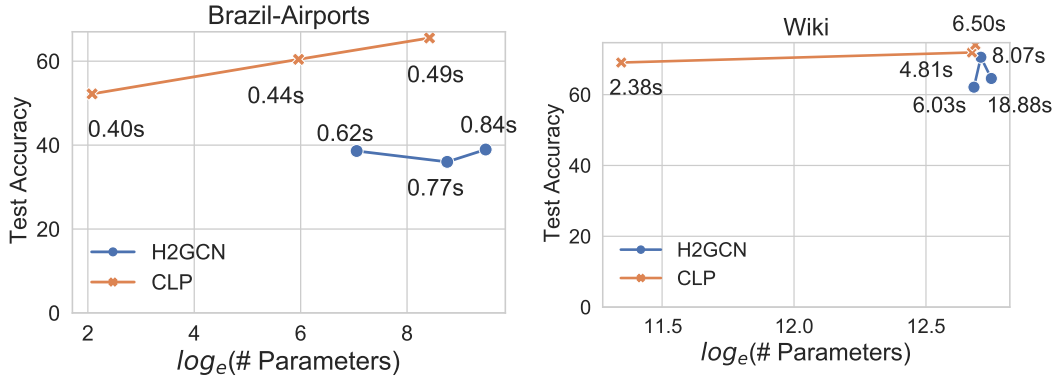


Figure 5.8: Classification accuracy and execution time of different methods with different layers on *heterophily* graphs. Execution time is marked in the plot in terms of seconds (s).

Parameter number and execution time comparison. Our approach often requires significantly fewer parameters than GNN models since only the base predictor has

parameters to train, which is less than GNN models. Moreover, another gain is faster training time because we do not use the graph structure for our prior predictions, and the LP step is time-efficient. As an example, we plot parameter numbers vs test accuracy of CLP and H2GCN of two heterophily graphs, i.e., Brazil-Airports and Wisconsin, in Figure 5.8. Note that H2GCN similarly contains an MLP component as a node feature encoder. We endow CLP and H2GCN with *Linear*, 2-layers and 3-layers MLP models as base predictors (feature encoder for H2GCN). The hidden dimensions of CLP and H2GCN are the same as the general settings. Each model's execution time (average value of 10 runs) under different settings is shown in Figure 5.8. We observe that CLP achieves much better performance with orders of magnitude fewer parameters and execution time.

5.7 Conclusion and Future work

In this chapter, we focused on the graph learning tasks with challenging heterophily settings. Motivated by an experimental investigation of existing models performance, we proposed an approach that extends LP algorithm to heterophily settings by smoothing the prior predictions across neighbours weighted by the compatibility matrix. A theoretical analysis shows that CLP has a closed-form solution with mild conditions on an appropriate matrix and we can thus give a detailed explanation of when CLP will support convergence. Comprehensive experiments demonstrate the effectiveness and efficiency of our approach on real-world and synthetic graphs.

In future work, we plan to generalise CLP approach to the heterophily setting of regression problems on graphs.

Chapter 6

Unsupervised Heterophilous Graph Embedding

6.1 Introduction

Following the triumph of machine learning in computer vision and natural language processing, there are more and more success stories coming from machine learning paradigms suited for relational data such as graphs or meshes [ZCZ20]. *Graph embedding* (GE) or *graph representation learning* (GRL) has become a predominant approach to find effective data representations from complex systems that take the form of graphs [CWPZ19]. Depending on the model's inherent architecture, existing GE methods can be categorised into "shallow" or "deep" groups. "Shallow" methods are characterised by an embedding lookup table optimised during training, which directly encodes each vertex as a vector [PARS14, TQW$^+$15, GL16, RSF17]. Despite the relative success of shallow embedding methods, they often ignore node attributes and only focus on the graph structural information. This greatly limits their ability to learn expressive representations.

Unlike shallow embedding methods, "deep" methods are often empowered by more complex encoders, usually a deep neural network, enabling the natural modelling of graph structures and node attributes [DBV16, KW17, XHLJ19]. These methods have proven effective in (semi-)supervised settings and have made remarkable breakthroughs in various application areas, such as social networks, e-commerce networks, biology networks, and traffic networks [ZCZ20]. Nevertheless, the effectiveness of deep methods in unsupervised settings, i.e. how to learn effective representations without any supervision, is relatively unexplored. In addition, recent work demonstrated that classic supervised GE methods have limited representation power on heterophilous graphs [ZYZ$^+$20, BWSS21, LHL$^+$21]. For instance, GCN [KW17] is proven to be a low-frequency filter that results in indistinguishable node representations on heterophilous graphs [BWSS21]. And moreover, addressing the heterophily scenario is essential for graph analysis and fairness study [LvdH13, KGW$^+$18]. In light of these three observations, a natural question arises **RQ1**: *how do existing GE methods perform on heterophilous graphs without supervision?*

Contribution. We focus on the effectiveness of GE methods in tackling the node

clustering task in a challenging setting: the goal of the task is to group similar nodes into the same category without any manual supervision [HW79, NJW01] (for convenience, we refer to unsupervised graph embedding as graph embedding in the remainder of the text). Moreover, unlike most prior work, which implicitly holds a *homophily* assumption that nodes of the same class tend to be connected, we focus on the more difficult *heterophily* setting where "opposites attract", i.e., linked nodes are likely to be from different classes.

First, we conduct an empirical study on 20 synthetic graphs with a variety of homophily ratios (h) to investigate whether h influences the node clustering performance of representative GE methods (RQ1). Our experimental results, summarised in Figure 6.2, show that (i) the performance of GE methods that rely on graph structure information decreases significantly when $h \to 0$; (ii) the performance of GE methods that rely on node attributes is not affected by changes of h, and their performance has a significant advantage compared to other methods when $h < 0.5$. These two findings directly answer RQ1, and meanwhile, raise another interesting and challenging question **RQ2:** *could we design a GE framework that adapts well to both homophily and heterophily settings?*

Motivated by the limitations mentioned above, we design the GE task as an r-ego network discrimination problem and propose a *self-supervised network embedding* (Selene) framework. Conceptually, we propose three solutions to avoid the homophily's influence on the GE performance and leverage the idea of negative-sample-free *self-supervised learning* (SSL) to design an optimisation objective for GE. Specifically, we propose a dual-channel features embedding pipeline that integrates node attributes and graph structure information. Next, we revisit representative GE mechanisms and propose using r-ego network sampling and anonymisation to break the inherent homophily assumption of GE and introduce graph structure features to enhance the framework's ability to capture structural information. Lastly, we employ a negative-sample-free SSL objective function to optimise the framework.

6.2 Additional Related Work

Graph embedding before GNNs. GE techniques aim at embedding rich node attributes and structural information in complex systems that take the form of graphs into low-dimensional node representations [CWPZ19]. Depending on the model architecture, GE methods can be naturally categorised into two groups: "shallow" and "deep" methods [DHW+20]. Shallow methods comprise an embedding lookup table that directly encodes each node as a vector and is optimised during training. Within this group, several Skip-Gram [MSC+13]-based GE methods have been proposed, such as DeepWalk [PARS14] and node2vec [GL16] as well as their matrix factorisation interpretation NetMF [QDM+18], LINE [TQW+15] and PTE [TQM15]. DeepWalk generates walk sequences for each node on a graph using truncated random walks and learns node representations by maximising the similarity of representations for nodes that occur in the same walks, thus preserving neighbourhood

structures. Node2vec increases the expressivity of DeepWalk by defining a flexible notion of a node's graph neighbourhood and designing a second order random walk strategy to sample the neighbourhood nodes; LINE is a special case of DeepWalk when the size of node's context is set to one; PTE can be viewed as the joint factorisation of multiple graphs' Laplacians [QDM+18]. To better capture the structural identity of nodes independent of graph position and neighbourhood's labels, struc2vec [RSF17] constructs a hierarchy to encode structural node similarities at different scales. Despite the relative success of shallow embedding methods, they often ignore the richness of node attributes and only focus on the graph structural information, which hugely limits their performance.

Graph embedding with GNNs. Recently, *graph neural networks* (GNNs) have shown promising results in modelling structural and relational data [WPC+21]. The common idea of capturing nodes' neighbourhoods to measure nodes' similarities used by shallow methods can be intuitively generalised with GNN models that follow a recursive neighbourhood aggregation or message-passing scheme. Additionally, GNNs are often powered by more complex encoders, usually a deep neural network, enabling more expressive modelling of graph structure and node attributes. Existing GNN models can be generally categorised into spectral [DBV16, LMBB17, XSC+19] and spatial approaches [KW17, HYL17b] and can be trained to fit node labels or to reconstruct graph structure. Within this field, GE is equivalent to an optimisation problem that encodes graph nodes into latent vectors by means of an encoding function, with the objective of ensuring that results decoded from vectors preserve graph properties of interest. Several approaches leveraging GNNs, such as ChebNet [DBV16], GCN [KW17], GraphSAGE [HYL17b], CayleyNets [LMBB17], GWNN [XSC+19] and GIN [XHLJ19], fuse node features and neighbourhood structures to compute embeddings, have allowed remarkable breakthroughs in numerous fields under (semi-)supervised settings. However, their effectiveness in GE without manual supervision is relatively unexplored. Recently proposed methods leveraging GNNs with self-supervised learning, such as DGI [VFH+19], GMI [PHL+20], SDCN [BWS+20], and GBT [BKC21] have been introduced for the task of unsupervised GE, primarily being evaluated on homophilous graphs.

Heterophilous graph embedding. Recent works focused on the GE on heterophilous graphs where nodes of different class labels are often connected [ZYZ+20, BWSS21, CPLM21, LHL+21] and have shown that the representation power of GNNs that follow (semi-)supervised settings is greatly limited on heterophilous graphs. Nevertheless, how do unsupervised GE methods perform on heterophilous graphs is still unexplored.

6.3 Preliminaries

Problem setup. In this chapter, we focus on the unsupervised node clustering task on a graph $\mathcal{G}$. We firstly aim to learn precise node representation $\mathbf{H}_v \in \mathbb{R}^d$, with d being the dimensionality of node representations, for all $v \in \mathcal{V}$ by leveraging node attributes and local structure context. Then, we infer the unknown class labels y_v

for all $v \in \mathcal{V}$ according to the learned node representation $\mathbf{H}_v$ with a clustering algorithm (we utilise the well-known algorithm K-means [HW79] in this chapter). Note that, for convenience, we refer to unsupervised graph embedding as graph embedding in the subsequent discussion. Table 6.1 lists the additional mathematical notation used in this chapter.

Table 6.1: Summary of additional notations.

Notation	Description
h	Homophily ratio of graph $\mathcal{G}$
$\mathcal{N}_v$	Neighbourhood around node v
$\mathcal{G}_r(v)$	r-ego network of v
$\mathcal{G}_r^{(1)}$, $\mathcal{G}_r^{(2)}$	Two distorted graphs of $\mathcal{G}_r(v)$
$\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$	Two distorted node attribute matrix of $\mathbf{X}$

Definition 6 (Neighbourhood $\mathcal{N}_v$). *We denote a general neighbourhood around ego node v, excluding v (in case $\mathcal{G}$ has self-loops), as $\mathcal{N}_v$; and the corresponding neighbourhood including the ego node v as $\widetilde{\mathcal{N}}_v$. The neighbouring nodes of ego node v within r hops are denoted by $\mathcal{N}_v^r = \{v : d(u,v) \leq r\}$, where $d(u,v)$ is the shortest path instance between u and v. For example, for the graph shown in Figure 6.1-(a), $\mathcal{N}_{v_0}^1 = \{v_0, v_1, v_2, v_4, v_7\}$.*

Definition 7 (r-ego Network $\mathcal{G}_r(v)$). *[ML12, QCD+20] For an ego node v of $\mathcal{G}$, its r-ego neighbours are $\mathcal{N}_v^r \subseteq \mathcal{V}$. The corresponding r-ego network is an induced subnetwork of $\mathcal{G}$, which is defined as $\mathcal{G}_r(v) = \{\mathcal{N}_v^r, \mathcal{E}_v^r, \mathbf{X}_v^r\}$, where $\mathcal{E}_v^r := ((\mathcal{N}_v^r \times \mathcal{N}_v^r) \cap \mathcal{E})$.*

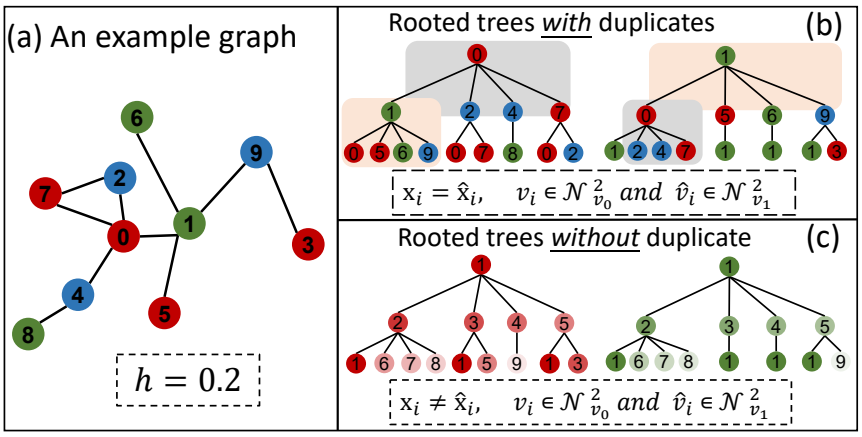


Figure 6.1: A toy example of the rooted aggregation trees on an example heterophilous graph ($h = 0.2$). Different colours represent different node classes. (a) An example graph; (b) Two rooted 2-layers aggregation trees of two connected nodes, i.e., v_0 and v_1. Duplicated components of two rooted trees are marked with the same shadow colour. (c) After ego network sampling and anonymity, two rooted aggregation trees have no duplicates.

Definition 8 (Homophily Ratio h). *For an arbitrary graph $\mathcal{G}$, its homophily ratio h is determined by the relationship between node labels and graph structure encoded in the adjacency matrix ($\mathbf{A}$). Recent work commonly use two homophily metrics: edge homophily*

(h_{edge}) [ZRR+21] and node homophily (h_{node}) [PWC+20]. They can be formulated as:

$$h_{edge} = \frac{|\{(u,v) : (u,v) \in \mathcal{E} \wedge y_u = y_v\}|}{|\mathcal{E}|}$$
$$h_{node} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{u : u \in \mathcal{N}_v \wedge y_u = y_v\}|}{|\mathcal{N}_v|} \tag{6.1}$$

Specifically, h_{edge} evaluates the fraction of edges in a graph that connect nodes that have the same class labels; h_{node} evaluates the overall fraction of neighbouring nodes that have the same class labels.

In this chapter, we focus on edge homophily and set $h = h_{edge}$ in the following sections. Figure 6.1-(a) demonstrates an example graph with $h = 0.2$.

6.4 An Experimental Investigation

In this section, we empirically analyse the performance of GE methods on 20 synthetic graphs with different homophily ratios (h) and node attributes ($\mathbf{X}$). The main goal is to investigate (**RQ1**): *how do existing GE methods perform on heterophilous graphs?* Specifically, we quantify their performance on the node clustering task on two sets of synthetic graphs, i.e., Synthetic and Synthetic-Products, with $h \in [0, 0.1, \ldots, 0.9]$. A detailed description of the synthetic datasets generation process can be found in Section 6.6.1; and we refer the reader to Section 6.6.2 for details on the experimental settings.

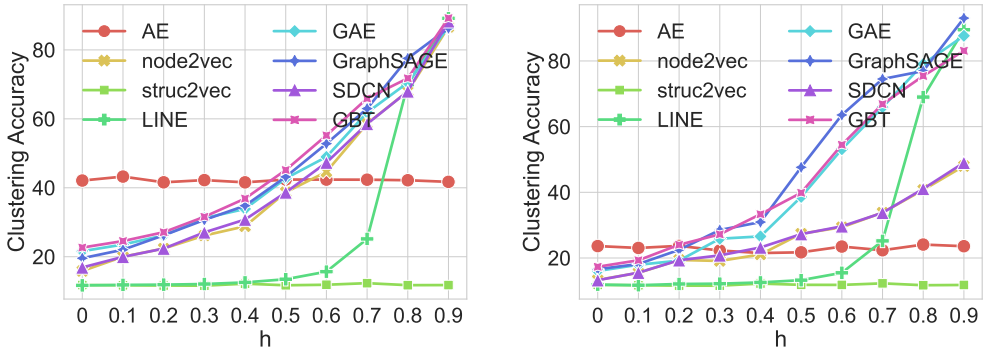


Figure 6.2: Node clustering accuracy of representative GE methods on synthetic datasets, i.e., Synthetic (a) and Synthetic-Products (b).

Figure 6.2 illustrates that with the decrease of h, the clustering accuracy of representative GE methods that rely on graph structure, i.e., node2vec[GL16], LINE[TQW+15], GAE[KW16], GraphSAGE[HYL17b], SDCN[BWS+20] and GBT[BKC21], decreases significantly. The accuracy of AE[HS06], which only relies on raw node attributes, remains stable for different values of h and demonstrates apparent advantage on Synthetic (with $h < 0.5$). The reason why existing GE methods using graph structure only fail when $h \to 0$ is that they all implicitly follow a homophily assumption. For instance, objective functions of node2vec, LINE and GraphSAGE, guide nodes

at close distance to have similar representations and nodes far away to have different ones. The inherent aggregation mechanism of GAE, GraphSAGE, SDCN and GBT naturally assumes local smoothing[CLL+20, MWW20] (which is mainly caused by the duplicated aggregation tree as shown in Figure 6.1-(b)), which translates into neighbouring nodes having similar representations. But as shown in Figure 6.2, GE methods leveraging graph structure show outstanding performance with $h \to 1$. Lastly, it is worth noting that struc2vec[RSF17], which relies on graph structure too, performs worse on the 20 synthetic graphs. The reason is that struc2vec identifies the structure of ego networks by counting the degree of nodes of different hops. The synthetic graph generation process assigns node degree to each node with a normal distribution, i.e., nodes from different classes can have the same degree. Therefore, it is not able to identify the ego network's local structure. This suggests that neural network-based models have better usability in capturing ego network structure patterns than manual defined statistical methods.

6.5 Proposed Approach

In this section, we formalise the main challenges of GE on heterophilous graphs. To address these challenges, we present the *self-supervised network embedding* (Selene) framework. Figure 6.3 shows the overall view of Selene.

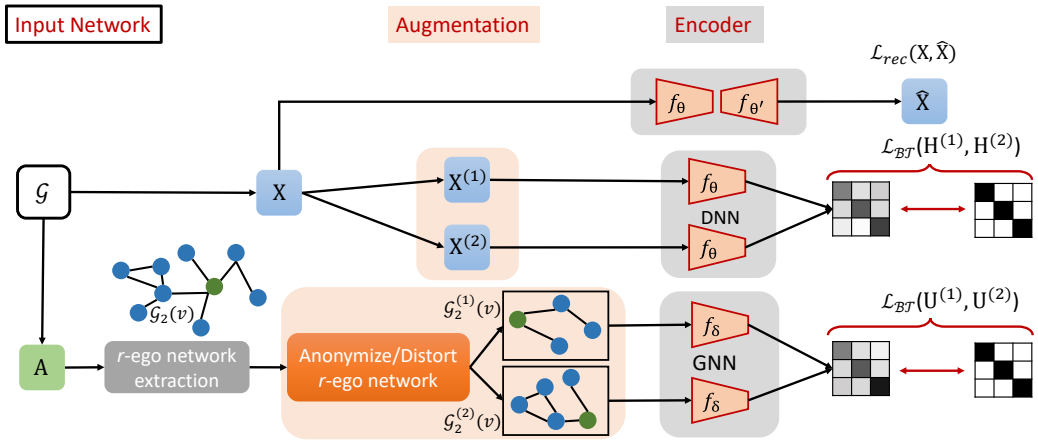


Figure 6.3: An illustration of our proposed framework Selene. $\mathbf{X}$ and $\mathbf{A}$ are raw node attributes and adjacency matrix of the input graph $\mathcal{G}$, respectively. $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ are two distorted node attribute matrix, $\widehat{\mathbf{X}}$ is the reconstructed node attribute matrix. Extracted r-ego network ($\mathcal{G}_r(v)$) of ego node v is anonymised to break its connection to neighbour nodes, and then distorted to two ego networks, i.e., $\mathcal{G}_r^{(1)}(v)$ and $\mathcal{G}_r^{(2)}(v)$. Node attribute encoder is optimised by the reconstruction loss $\mathcal{L}_{rec}(\mathbf{X}, \widehat{\mathbf{X}})$ and attribute Barlow-Twins loss $\mathcal{L}_{BT}(\mathbf{H}^{(1)}, \mathbf{H}^{(2)})$; graph structure encoder is optimised by the graph Barlow-Twins loss $\mathcal{L}_{BT}(\mathbf{U}^{(1)}, \mathbf{U}^{(2)})$. The final node representation is obtained by applying Combine to the generated node attribute representation $\mathbf{H}$ and graph structure representation $\mathbf{U}$.

Challenges. As discussed in Section 6.4, it is crucial to involve both node attributes and graph structure information in the GE model to learn expressive node representations by identifying and distinguishing differences among different nodes. Existing methods address this challenge by reconstructing the graph structure [KW16],

node attributes [HS06, BWS+20] or by assuming that nodes close to each other tend to have similar node representations [PARS14, GL16, HYL17b]. However, these solutions are not suitable for heterophilous graphs as shown in Section 6.4. Therefore, the design of an appropriate learning objective is a key challenge to address in this chapter. In the remainder of this section, we address the following research challenges:

RC1 How to leverage node attributes and graph structure for graph embedding?

RC2 How to break the inherent homophily assumptions of traditional graph embedding mechanisms?

RC3 How to define an appropriate objective function to optimise the embedding learning process?

6.5.1 Dual-channel Feature Embedding (RC1)

The empirical analysis of Section 6.4 highlighted that node attributes play a major role in the performance of GE methods, especially on heterophilous graphs. Therefore, we propose a dual-channel feature embedding pipeline to learn node representation from node attributes and graph structure separately, as shown in Figure 6.3. Such a design brings two main benefits: (i) both sources of information can be well utilised without interfering with each other, and (ii) the inherent homophily assumptions of GE methods can be greatly alleviated, an issue we will address in the next subsection.

Node attribute encoder module. As previously mentioned, learning effective node attribute representations is of great importance for NE. There are several alternative methods to learn representations for different types of data, including Autoencoder [HS06] and its derived variants [MMCS11, MSJG15, MRA+16]. In this chapter, we employ the basic Autoencoder to learn representations of raw node attributes, which can be replaced by more sophisticated encoders to obtain higher performance. We assume an L-layer Autoencoder, with the formulation of the ℓ-th encoding layer being:

$$\mathbf{H}_e^{(\ell)} = \phi(\mathbf{W}_e^{(\ell)}\mathbf{H}_e^{(\ell-1)} + \mathbf{b}_e^{(\ell)}) \tag{6.2}$$

where ϕ is a non-linear activation function such as ReLU [NH10] or PReLU [HZRS15]. $\mathbf{H}_e^{(\ell-1)} \in \mathbb{R}^{n \times d_{\ell-1}}$ is the hidden node attribute representations in layer $\ell - 1$, with $d_{\ell-1}$ being the dimensionality of this layer's hidden representation. $\mathbf{W}_e^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ and $\mathbf{b}_e^{(\ell)} \in \mathbb{R}^{d_\ell}$ are trainable weight matrix and bias of the ℓ-th layer in the encoder. Node representations $\mathbf{H} = \mathbf{H}_e^{(L)}$ are obtained after successive application of L encoding layers.

Following the encoder, the decoder reconstructs input node attributes from the computed node representations $\mathbf{H}$. Typically, a decoder has the same structure as the encoder by reversing the order of layers. Its ℓ-th fully connected layer can be formally represented:

$$\mathbf{H}_d^{(\ell)} = \phi(\mathbf{W}_d^{(\ell)}\mathbf{H}_d^{(\ell-1)} + \mathbf{b}_d^{(\ell)}) \tag{6.3}$$

where $\mathbf{W}_d^{(\ell)}$ and $\mathbf{b}_d^{(\ell)}$ are trainable weight matrix and bias of ℓ-th layer in the decoder, respectively. Reconstructed node attributes $\widehat{\mathbf{X}} = \mathbf{H}_d^{(L)}$ are obtained after successive applications of L decoding layers. We optimise the autoencoder parameters by minimising the difference between raw node attributes $\mathbf{X}$ and reconstructed node attributes $\widehat{\mathbf{X}}$ with:

$$\mathcal{L}_{rec}(\mathbf{X}, \widehat{\mathbf{X}}) = \frac{1}{2|\mathcal{V}|}||\mathbf{X} - \widehat{\mathbf{X}}||_F^2 \tag{6.4}$$

6.5.2 r-ego Network Sampling and Anonymity (RC1)

The success of GNN-related GE frameworks largely relies on their information aggregation mechanism. Nevertheless, such a design leads to low-quality GE on heterophilous graphs due to the implicit homophily assumptions. To address this issue, we modify the aggregation mechanism by cutting off connections among aggregation trees and highlighting each node's local structure.

Specifically, we first define a subnetwork instance of a certain (ego) node v to be its r-ego network $\mathcal{G}_r(v)$. For example, in Figure 6.3, $\mathcal{G}_2(v)$ represents a 2-ego network instance of $\mathcal{G}$. We note that given that $\mathcal{G}_r(v)$ captures the local structure of v, it is sufficient to compute a structural representation of v [QCD+20].

Anonymisation. However, as shown in Section 6.4, the rooted aggregation trees of two connected nodes largely overlap, which can lead to learning indistinguishable representations for connected nodes. This property is useful in homophilous settings but leads to low-quality GE performance on heterophilous graphs. To address this, we propose node anonymisation to reduce the overlap of ego networks of connected nodes. Specifically, we anonymise the sampled ego network $\mathcal{G}_r(v)$ by relabelling its nodes to $\{1, 2, \ldots, |\mathcal{N}_u^r|\}$, in an arbitrary order, and erasing raw node attributes. Note that node order does not influence the representation quality because most GNNs are invariant to permutations of their inputs [BHB+18]. This step ensures that v's representation is only influenced by its local structure, as illustrated in Figure 6.1-(c).

graph structure features. Despite the significant success of GNNs in a variety of graph-related tasks, their representation power in graph structural representation learning is limited [XHLJ19]. In order to obtain invariant node structural representations so that nodes with different ego network structures are assigned different representations, we employ the structural features proposed in [LWWL20]. In this chapter, we adopt variants of shortest-path distance (SPD) as node structural features ($\widetilde{\mathbf{X}}$).

graph structure encoder module. Over the past few years, numerous GNNs have been proposed to learn node representations from graph-structured data, including spectral GNNs [DBV16, LMBB17, XSC+19] and spatial GNNs [HYL17b, XHLJ19]. For the sake of simplicity, we adopt a simple GNN variant, i.e., GCN [KW17], as the building block of the graph structure encoder. The ℓ-th layer of a GCN

(Equation 2.3) can be formally defined as:

$$\mathbf{U}^{(\ell)} = \sigma(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{U}^{(\ell-1)}\mathbf{W}^{(\ell)}) \tag{6.5}$$

with $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, and $\widehat{\mathbf{D}}$ is the diagonal node degree matrix of $\widehat{\mathbf{A}}$. $\mathbf{U}^{(\ell-1)} \in \mathbb{R}^{n \times d_{\ell-1}}$ is the hidden representation of nodes in layer $\ell - 1$, with $d_{\ell-1}$ being the dimensionality of this layer's representation, and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ is a trainable parameter matrix. σ is a non-linear activation function such as ReLU or Sigmoid [HM95] function. Structural representations $\mathbf{U} = \mathbf{U}^{(L)}$ are obtained after successive applications of L layers.

Final node representations. Representations capturing node attributes ($\mathbf{H}$) and structural context ($\mathbf{U}$) are combined to obtain expressive and powerful representations as:

$$\mathbf{Z} = \textsc{Combine}(\mathbf{H}, \mathbf{U}) \tag{6.6}$$

where $\textsc{Combine}(\cdot)$ can be any commonly used aggregation operation in GNNs [XHLJ19], such as *mean*, *max*, *sum* and *concat*. We utilise *concat* in all our experiments which allows for an independent integration of representations learnt by the dual-channel architecture. In the next subsection, we will introduce the strategy to optimise the entire framework.

6.5.3 Non-negative Self-supervised Learning (RC3)

The objective function plays a major role in GE tasks. Several objective functions have been proposed for GE, such as graph reconstruction loss [KW16], distribution approximating loss [PARS14] and node distance approximating loss [HYL17b]. Nevertheless, none of them is suitable for GE optimisation on heterophilous graphs because of the homophily assumptions used to determine (dis)similar pairs. In heterophilous graphs, node distance on graph alone does not determine (dis)similarity, i.e., connected nodes are not necessarily similar, and nodes far apart are not necessarily dissimilar. This removes the need for connected nodes to be close in the embedding space and for disconnected nodes to be far apart in the embedding space.

Inspired by the latest success of non-negative self-supervised learning (SSL), as discussed in Chapter 2, we adopt the Barlow-Twins [ZJM+21] as our overall optimisation objective. Overall, Barlow-Twins is a method (originally proposed in the computer vision domain) that learns data representations using a symmetric graph architecture and an empirical cross-correlation based loss function. Specifically, it measures the cross-correlation matrix between the output of two identical graphs fed with distorted versions of a sample and makes it as close to the identity matrix as possible. This causes representations of distorted instances of a sample to be similar, increasing the robustness of representations (invariance to small perturbations) and minimising the redundancy between the components of these representations.

Augmentation. As mentioned previously, we propose a dual-channel feature embedding pipeline to adaptively preserve the useful information from raw node attributes and graph structure. Therefore, there are two types of model inputs, i.e., node attributes matrix $\mathbf{X}$ and anonymised ego network $\mathcal{G}_r(v)$. Following the widely adopted SSL training strategy [YCS+20, BKC21], we employ two data augmentation methods (f_{aug}), i.e., node attribute masking and edge masking, for distorted instances generation. As shown in Figure 6.3, with a pair of given masking ratios, i.e., node attribute masking ratio p_x and edge masking ratio p_e, two pairs of distorted instances can be generated:

$$\begin{aligned} f^{x}_{aug}(\mathbf{X}, p_x) &= (\mathbf{X}^{(1)}, \mathbf{X}^{(2)}) \\ f^{g}_{aug}(\mathcal{G}_r(v), p_x, p_e) &= (\mathcal{G}_r^{(1)}(v), \mathcal{G}_r^{(2)}(v)) \end{aligned} \tag{6.7}$$

Barlow-Twins loss function. Two pairs of node representations follow from the pairs of distorted instances, computed with the node attribute encoder and graph structure encoder: node attribute representations $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$, and graph structure representations $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$. We utilise each pair of node representations to compute a cross-correlation matrix $\mathcal{C} \in \mathbb{R}^{d \times d}$ with d being the dimensionality of input representations. Using node attributes $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ as example, results in the following objective function:

$$\begin{aligned} \mathcal{L}_{BT}(\mathbf{H}^{(1)}, \mathbf{H}^{(2)}) &= \sum_{i}^{|\mathcal{V}|}(1 - \mathcal{C}_{ii})^2 + \lambda \sum_{i}^{|\mathcal{V}|}\sum_{j \neq i}^{|\mathcal{V}|} \mathcal{C}_{ij}^2 \\ with \qquad \mathcal{C}_{ij} &= \frac{\sum_b \mathbf{h}^{(1)}_{b,i} \mathbf{h}^{(2)}_{b,j}}{\sqrt{\sum_b (\mathbf{h}^{(1)}_{b,j})^2} \sqrt{\sum_b (\mathbf{h}^{(2)}_{b,j})^2}} \end{aligned} \tag{6.8}$$

where $\lambda > 0$ defines the trade-off between the invariance and redundancy reduction terms, we adopt the default settings as [ZJM+21]. b is the batch indexes and i, j index the vector dimension of the input representation vectors.

Empowered with Barlow-Twins, we can optimise the framework's encoders under heterophilous settings. The node attribute encoder is optimised with $\mathcal{L}_{BT}$ (Equation 6.8) and $\mathcal{L}_{rec}$ (Equation 6.4), and the graph structure encoder is optimised with $\mathcal{L}_{BT}$(Equation 6.8). The overall loss function is $\mathcal{L} = \mathcal{L}_{BT}(\mathbf{U}^{(1)}, \mathbf{U}^{(2)}) + \mathcal{L}_{BT}(\mathbf{H}^{(1)}, \mathbf{H}^{(2)}) + \mathcal{L}_{rec}(\mathbf{X}, \widehat{\mathbf{X}})$. Final node representations obtained by Equation 6.6 contain information from node attributes and structural context, which can be used in downstream tasks.

6.5.4 Model Scalability

According to the design for Selene's framework, we can find that the graph structure representation learning module is categorised as a local network algorithm [Ten16], which only involves local exploration of the graph structure. On the other hand, the node attribute representation learning module (Autoencoder) naturally support the

Table 6.2: Statistics of real-world datasets. $|\mathcal{V}|$: number of nodes; $|\mathcal{E}|$: number of edges; π: dimensionality of nodes features; OSF: nodes only have structure related features; d_{avg}: average degree; $|\mathcal{Y}|$: number of possible class labels; h: homophily ratio;

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	π	**OSF**	$\lvert\mathcal{Y}\rvert$	d_{avg}	h
Texas	183	325	1,703	False	5	1.8	0.108
Wisconsin	251	515	1,703	False	5	2.1	0.196
Actor	7,600	30,019	932	False	5	3.9	0.219
Chameleon	2,277	31,421	2,325	False	5	27.6	0.233
USA-Airports	1,190	13,599	1	True	4	22.9	0.251
Cornell	183	298	1,703	False	5	1.6	0.305
Europe-Airports	399	11,988	1	True	4	30.1	0.309
Brazil-Airports	131	2,077	1	True	4	16.4	0.311
Deezer-Europe	28,281	185,504	31,241	False	2	6.6	0.525
Citeseer	3,327	4,552	3,703	False	6	2.7	0.736
DBLP	4,057	3,528	334	False	4	1.7	0.799
Pubmed	19,717	88,648	500	False	3	4.5	0.802

mini-batch mechanism. Therefore, our design enables Selene to scale to representation learning on large-scale graphs and to be friendly to distributed computing settings [QCD+20].

6.6 Evaluation

We evaluate our proposed framework, Selene, on benchmark real-world and synthetic datasets and compare with eleven competing methods over node clustering tasks. Note that different from the experimental settings of relevant work,[VFH+19, PHL+20, BKC21] which utilise node class labels to train a classifier after obtaining the node representations to predict test nodes, we do not employ any node class label supervision to strictly adhere to the unsupervised learning requirements.

6.6.1 Datasets

Real-world datasets. We use a total of 12 real-world datasets (Texas[PWC+20], Wisconsin[PWC+20], Actor[PWC+20], Chameleon[RAS21], USA-Airports[RSF17], Cornell[PWC+20], Europe-Airports[RSF17], Brazil-Airports[RSF17], Deezer-Europe[RS20], Citeseer[KW17], DBLP[FZMK20], Pubmed[KW17]) in diverse domains (web-page, citation, co-author, flight transport and online user relation). All real-datasets are available online[1]. Statistics information is summarised in Table 6.2.

Synthetic datasets. We generate random synthetic graphs with various homophily levels h and node features by adopting a similar approach to[APK+19, KO21] but with some modifications. For instance, synthetic graphs of [APK+19] have no available contextual node attributes. Specifically, each synthetic graph has 10 classes and 500 nodes per class. Nodes are assigned random features sampled from 2D

[1] `https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html`

Gaussians (Synthetic-h) or features from real-world dataset[HFZ+20] (Synthetic-Products-h). Each dataset, i.e., Synthetic and Synthetic-Products, has 10 graphs with $h \in [0, 0.1, 0.2, \ldots, 0.9]$. The generation process is similar to the Synthetic datasets generation of the previous chapter.

6.6.2 Experimental Setup

Competing methods. We compare our framework Selene with 11 competing GE methods. We adopt 3 different shallow competing embeddings methods, including node2vec[GL16], struc2vec[RSF17] and LINE[TQW+15]. We adopt 8 additional deep competing embedding methods, including AE[HS06], GAE & VGAE[KW16], GraphSAGE[HYL17b], DGI[VFH+19], SDCN[BWS+20], GMI[PHL+20] GBT[BKC21] and FAGCN*[BWSS21]. Note that GBT is a self-supervised learning approach that migrates Barlow-Twins approach to graph structure data. However, they do not modify the model structure of GNNs but just provide a new model training approach, hence still maintaining a homophily assumption. FAGCN* is a state-of-the-art heterophilous GNN for supervised settings, here we train it with the same mechanism as GBT to adapt it to the unsupervised setting.

Evaluation metrics. We employ three node clustering evaluation metrics: accuracy (ACC), normalised mutual information (NMI) and average rand index (ARI). For each evaluation metric, a significant value means better node clustering performance.

Model implementation. For shallow embedding methods, we set the embedding dimension to 128, the number of random walks of each node to 10 and the walk length to 80. For node2vec, we additionally select p, q over $\{0.25, 0.5, 1, 2\}$ with best clustering performance. We train embedding methods, including AE, GAE, VGAE and SDCN, with the same settings as[BWS+20]. Specifically, we train the models end-to-end using all nodes and edges with 30 epochs and a learning rate of 1×10^{-3}. For AE, we set the representation dimensions to $\{\pi - 500 - 500 - 200 - 10\}$, where π is the dimensionality of raw node attributes. For GNN-related methods, including GAE, VGAE, GraphSAGE, DGI, GBT, GMI, FAGCN* and Selene, we set their representation dimensions to $\{\pi - 256 - 16\}$. The representation dimensions of SDCN's GCN module are the same as AE. [BWS+20] proposes to pretrain the SDCN's AE component to boost its performance, thus we report the best performance of SDCN with/without pre-trained AE. For DGI, GMI, GBT and Selene, we assign the same GCN[KW17] encoder and follow the optimisation protocol as[BKC21]. We set $r = 3$ for ego network extraction following[LWWL20] and the batch size = 512. For shallow competing methods, we utilise the integrated implementations from GraphEmbedding[2]. For GAE, VGAE, GraphSAGE, DGI and FAGCN* we use the implementation from Pytorch-Geometric[3]; for AE & SDCN

[2] https://github.com/shenweichen/GraphEmbedding

[3] https://pytorch-geometric.readthedocs.io/en/latest/

and GMI and GBT, we use the implementation from the published code of SDCN[4], GMI[5] and GBT[6], respectively,

Note that, all experiments are conducted on a single Tesla V100 GPU. For all NN-based methods, we initialise them 10 times with random seeds and select the best solution to follow the similar setting as[BWS+20]. After obtaining node representations with each model, we feed the learned representations into a *K*-means clustering model[HW79] to get the final clustering prediction. The final clustering section is thus repeated 10 times, and we report the mean/std performance.

6.6.3 Experimental Results

Table 6.3: Node clustering results on heterophilous datasets. The bold numbers represent the top-2 results. OOM: out-of-memory. SAGE: GraphSAGE.

Dataset	Metrics	AE	node2vec	struc2vec	LINE	GAE	VGAE	SAGE	SDCN	DGI	GMI	GBT	FAGCN*	Ours	↑ (%)
Wisc.	ACC	58.61	41.39	43.03	39.4	37.81	40.0	46.29	38.25	44.58	36.97	48.01	**61.91**	**71.35**	15.25
	NMI	**30.92**	4.23	11.23	9.7	9.19	9.87	10.16	8.46	10.72	11.68	7.55	27.35	**39.31**	27.13
	ARI	28.53	-0.48	11.5	8.13	5.2	7.97	6.06	3.67	10.31	3.74	3.85	**31.56**	**43.26**	37.07
Texas	ACC	50.49	48.8	49.73	49.4	42.02	50.27	56.83	44.04	55.74	35.19	55.46	**57.92**	**64.48**	11.33
	NMI	16.63	2.58	18.61	16.9	8.49	11.73	16.97	14.24	8.73	7.72	10.17	**23.35**	**25.22**	8.01
	ARI	14.6	-1.62	20.97	18.08	10.83	21.51	**23.5**	10.65	8.25	2.96	12.1	22.54	**34.19**	45.49
Wisc.	ACC	58.61	41.39	43.03	39.4	37.81	40.0	46.29	38.25	44.58	36.97	48.01	**61.91**	**71.35**	15.25
	NMI	**30.92**	4.23	11.23	9.7	9.19	9.87	10.16	8.46	10.72	11.68	7.55	27.35	**39.31**	27.13
	ARI	28.53	-0.48	11.5	8.13	5.2	7.97	6.06	3.67	10.31	3.74	3.85	**31.56**	**43.26**	37.07
Actor	ACC	24.19	25.02	22.49	22.7	23.45	23.3	23.08	23.67	24.26	26.18	24.68	**25.61**	**28.22**	10.19
	NMI	0.97	0.09	0.04	0.09	0.18	0.21	0.58	0.08	1.38	0.2	0.74	**3.22**	**4.69**	45.65
	ARI	**0.5**	0.06	-0.05	0.11	-0.04	0.34	0.22	-0.01	0.07	0.41	-0.57	0.34	**1.82**	264.00
Chamel.	ACC	**35.68**	21.31	26.34	31.97	32.76	30.65	31.04	33.5	27.77	25.73	32.21	31.33	**38.49**	7.88
	NMI	10.38	0.34	3.55	10.78	11.6	6.86	10.55	9.57	4.42	2.5	10.56	**14.71**	**20.05**	36.3
	ARI	5.8	0.02	1.82	6.04	4.65	4.4	6.16	5.86	1.85	0.52	**7.01**	5.16	**15.89**	126.68
USA-Air.	ACC	**55.24**	26.29	27.58	27.17	30.84	28.71	32.96	33.52	33.36	28.69	34.96	38.82	**57.39**	3.49
	NMI	**30.13**	0.25	0.44	0.23	2.71	0.55	2.67	5.21	5.52	0.6	5.27	12.3	**29.25**	0.96
	ARI	**24.2**	-0.05	0.09	-0.08	2.67	0.28	2.52	1.93	4.95	0.29	3.42	9.33	**24.69**	2.27
Cornell	ACC	52.19	50.98	32.68	34.1	43.72	43.39	44.7	36.94	44.1	33.55	52.19	**56.23**	**57.81**	2.81
	NMI	17.08	5.84	1.54	2.85	5.11	5.46	4.33	6.6	5.79	5.26	5.94	**17.08**	**17.1**	0.12
	ARI	17.41	0.18	-2.2	-1.54	6.51	3.97	5.64	3.38	4.87	3.05	0.63	**19.88**	**22.85**	14.94
Eu.-Air.	ACC	55.36	30.78	36.89	34.06	34.84	34.51	31.75	37.37	35.59	35.34	39.75	42.11	**57.47**	3.81
	NMI	**32.44**	3.69	6.15	4.77	10.15	3.68	2.1	8.45	10.77	11.08	9.44	16.81	**33.74**	4.01
	ARI	**24.24**	0.83	4.49	2.89	7.37	3.02	1.16	5.31	8.44	8.18	7.87	11.98	**25.14**	3.71
Bra.-Air.	ACC	**71.68**	30.38	38.93	33.74	36.64	32.82	37.02	38.7	37.1	38.93	40.92	44.2	**78.77**	9.89
	NMI	**49.26**	2.5	10.23	2.75	10.96	3.7	6.89	14.05	10.64	12.62	12.16	22.67	**55.6**	12.87
	ARI	**42.93**	-0.22	5.45	0.19	6.56	0.6	4.18	7.27	7.02	9.11	8.31	14.4	**52.87**	23.15
Deezer.	ACC	55.88	52.97	OOM	52.56	51.51	51.27	51.06	54.76	53.16	OOM	OOM	**56.81**	**59.75**	5.18
	NMI	**0.28**	0.0	OOM	0.08	0.13	0.12	0.16	0.17	0.05	OOM	OOM	0.27	**0.31**	10.71
	ARI	0.81	0.02	OOM	0.23	0.07	0.04	-0.02	0.61	-0.23	OOM	OOM	**0.82**	**0.89**	8.54

Real-world datasets. Node clustering results on homophilous and heterophilous real-world datasets are summarised in Table 6.4 and Table 6.3, respectively. In Table 6.3, we see that Selene is the best-performing method in all heterophilous datasets. In particular, compared to the best results of competing models, our framework achieves a significant improvement of up to 15.25% on ACC, 45.65% on NMI and 264% on ARI. Such outstanding performance demonstrates that Selene successfully integrates the important node attributes and graph structure information into node representations. An interesting case is the comparison of Selene, FAGCN* and GBT, given that FAGCN* and GBT also utilise the Barlow-Twins

[4] `https://github.com/bdy9527/SDCN`

[5] `https://github.com/zpeng27/GMI`

[6] `https://github.com/pbielak/graph-barlow-twins`

objective function to optimise a GNN model, and the major difference between FAGCN*, GBT and Selene is the dual-channel features embedding pipeline. Selene has a superior performance in all heterophilous datasets, indicating the necessity to propose an unsupervised heterophilous graph embedding model. Simply porting supervised models to unsupervised scenarios is not appropriate. And the dual-channel features embedding pipeline plays a crucial role in improving the representation learning in the heterophily scenario.

Table 6.4: Node clustering results on homophilous datasets. The bold numbers represent the top-2 results. OOM: out-of-memory. SAGE: GraphSAGE.

Dataset	Metrics	AE	node2vec	struc2vec	LINE	GAE	VGAE	SAGE	SDCN	DGI	GMI	GBT	FAGCN*	Ours	↑ (%)
Citeseer	ACC	58.79	20.76	21.22	28.42	48.37	55.67	49.28	**59.86**	58.94	59.04	57.21	47.42	**59.24**	-1.04
	NMI	30.91	0.35	1.18	8.49	24.59	32.45	22.97	30.37	**32.6**	**32.11**	31.9	20.18	29.91	-8.25
	ARI	30.29	-0.01	0.17	3.54	19.5	28.34	19.21	29.7	**33.16**	33.09	**33.17**	17.93	29.35	-11.52
DBLP	ACC	48.5	29.19	31.65	35.34	57.81	46.0	48.68	61.94	58.22	63.28	**73.1**	41.25	**75.14**	2.79
	NMI	18.98	0.14	1.33	3.22	28.94	11.57	16.46	27.13	29.98	33.91	**42.21**	10.6	**43.96**	4.15
	ARI	15.15	-0.04	1.39	2.05	18.78	11.68	13.38	27.77	26.81	28.77	**42.57**	4.32	**46.08**	8.25
Pubmed	ACC	65.34	39.32	37.39	50.53	42.08	34.38	**67.66**	61.9	65.47	OOM	OOM	56.88	**66.27**	-2.05
	NMI	26.89	0.02	0.07	18.62	1.28	0.03	**30.71**	19.71	28.05	OOM	OOM	16.91	**28.7**	-6.55
	ARI	25.98	0.09	0.06	8.17	0.15	0.0	**29.1**	18.63	27.25	OOM	OOM	16.27	**28.21**	-3.06

Results of node clustering on homophilous graphs, contained in Table 6.4, show that Selene achieves competing performance compared to the best-performing methods, consistently performing in the top-2, in all 3 datasets on ACC. This shows Selene's suitability for homophilous graphs and proves the flexibility of the dual-channel feature embedding pipeline.

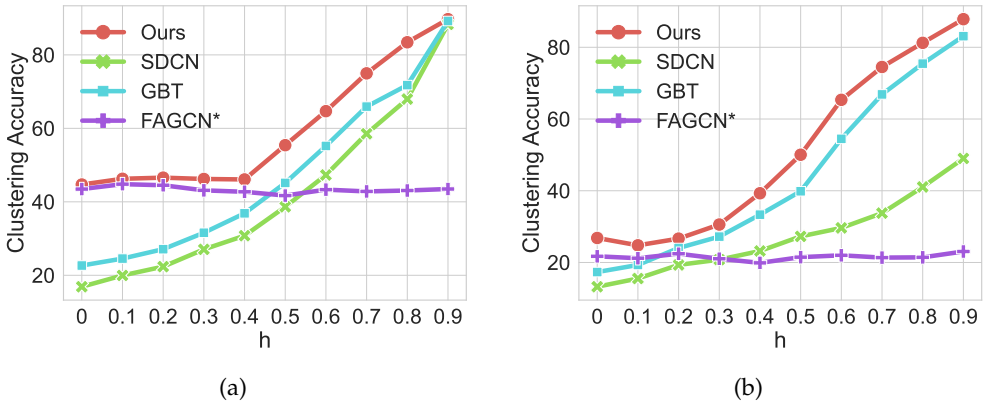


Figure 6.4: Clustering accuracy comparison of Selene vs SDCN vs GBT vs FAGCN* on Synthetic (a) and Synthetic-Products (b).

Synthetic graphs. We present Selene vs SDCN (state-of-the-art node clustering model) vs GBT (state-of-the-art graph contrastive learning model) vs FAGCN* (state-of-the-art heterophilous GNN model) clustering accuracy in Figure 6.4. Selene achieves the best performance on the Synthetic dataset and has competitive performance on the Synthetic-Products dataset. This shows that Selene adapts well to homophily/heterophily scenarios with(out) contextually raw node attributes. Moreover, FAGCN* performs worse on 20 synthetic graphs, which indicates that the heterophilous GNNs models designed for supervised settings do not adapt well to unsupervised settings because they need supervision information to train the more complex aggregation mechanism.

6.6.4 Analysis

Model scalability. Table 6.3 and Table 6.4 illustrate that two competing methods, i.e., struc2vec, GMI and GBT, have out-of-memory issues on large datasets, i.e., Pubmed and Deezer., an issue which did not arise with Selene. This shows Selene's advantage in handling large scale graphs due to its local network algorithm characteristic. Note that we only use one GPU in experiments, and such an advantage would be more evident in multi-GPU computing scenarios.

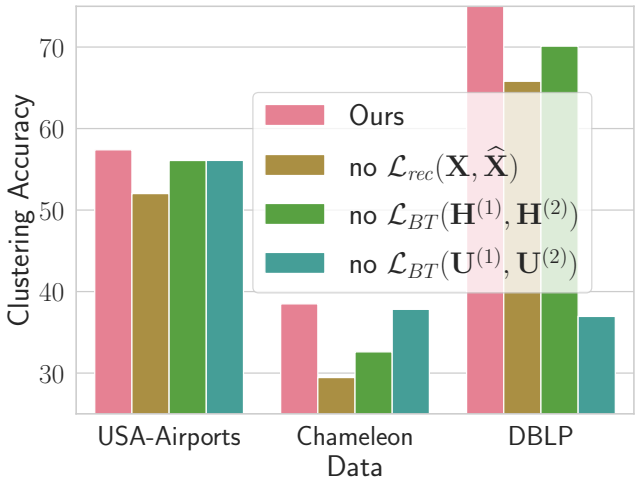


Figure 6.5: Loss function exploration. We ablate constituent of our loss function and report Selene's performance.

Effectiveness of loss function. The objective loss function of Selene contains three components and we thus sought to test the effectiveness of each component. In particular, we ablate each component and evaluate the obtained node representations on two heterophilous datasets (USA-Air., Chamel.) and one homophilous dataset (DBLP). Results shown in Figure 6.5 indicates that ablation of any component decreases the model's performance. Specifically, the ablation of $\mathcal{L}_{rec}(\mathbf{X}, \widehat{\mathbf{X}})$ causes a steeper performance degradation in heterophilous datasets, and the ablation of $\mathcal{L}_{BT}(\mathbf{U}^{(1)}, \mathbf{U}^{(2)})$ causes a steeper performance degradation in homophilous datasets, which indicates the importance of node attributes and graph structure information for heterophilous and homophilous graphs, respectively (consistent with observations in Section 6.4).

Effectiveness of dual-channel feature embedding pipeline. Selene contains a novel dual-channel features embedding pipeline to integrate node attributes and graph structure information thus we conduct an ablation study to explore the effectiveness of this pipeline. We first remove the pipeline and only use the Barlow-Twins loss function to train a vanilla GCN encoding module (such a structure is the same as GBT, hence we remark it as GBT). Next, we add the graph structure channel, which includes r-ego network extraction, anonymisation and distortion. Lastly, we add the node attribute channel to form the complete Selene framework. Experimental results are shown in Figure 6.6. Overall, we observe that the design of each channel is useful for learning better representation, with the node attribute

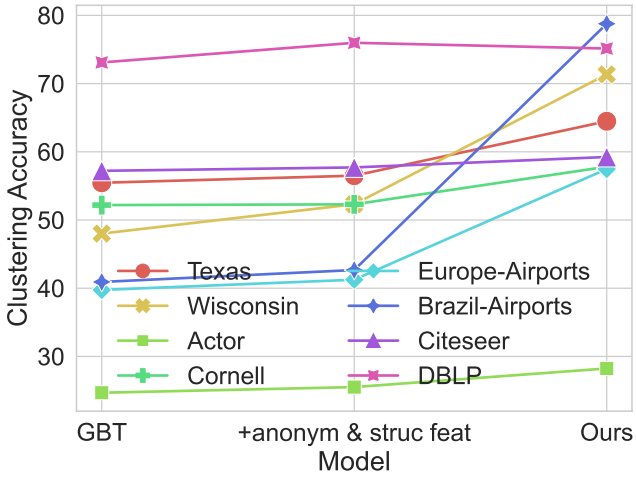


Figure 6.6: Framework component exploration. We ablate components of our dual-channel feature embedding pipeline and report Selene's performance.

channel playing a major role in the embedding on heterophilous graphs. Note that adding the node attribute channel slightly decreases the clustering accuracy for the homophilous dataset, i.e., DBLP, but it is still competitive.

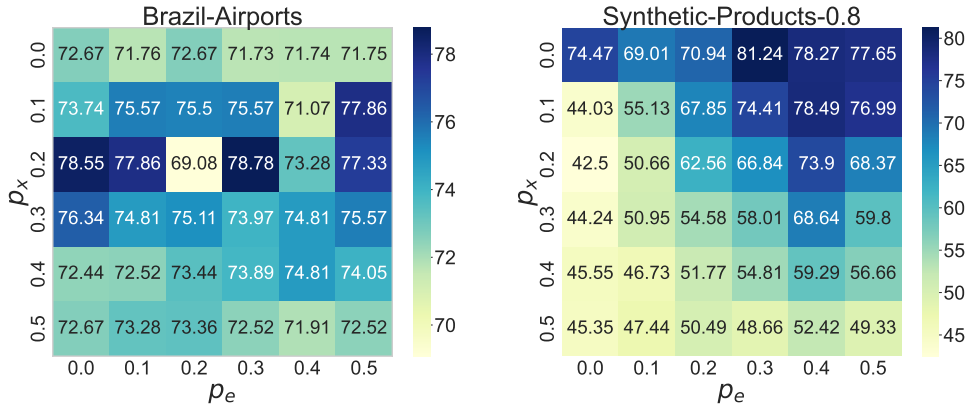


Figure 6.7: Hyper-parameter influence exploration. We present Selene's clustering accuracy on two graphs with different p_x and p_e.

Influence of p_x and p_e. We present Selene's clustering accuracy on two graphs, i.e., Brazil-Airports ($h = 0.311$) and Synthetic-Products-0.8 with different p_x and p_e in Figure 6.7. The figure indicates that hyper-parameters of augmentation methods have a significant influence on representation quality.

6.7 Conclusion and Future work

In this chapter, we focused on the graph embedding task with challenging heterophily settings and tackled two main research questions. First, we showed through an empirical investigation that the performance of existing embedding methods that rely on graph structure decreases significantly with the decrease of graph homophily ratio. Second, we proposed Selene to address the identified limitations,

which effectively fuses node attributes and graph structure information without additional supervision. Comprehensive experiments demonstrated the significant performance of Selene, and additional ablation analysis confirms the effectiveness of the components of Selene on real-world and synthetic graphs.

In future work, we propose to explore the *Information Bottleneck* [TZ15, WRLL20] of graph embedding to theoretically define the optimal representation of an arbitrary graph and optimally balance expressiveness and robustness against potential external attacks [ZAG18].

Part III

Online Social Network Analysis with Graph Machine Learning

Motivation and Summary

Online social networks (OSNs) have become a norm in our quotidian social and personal lives and provide a rich source of information about millions of users worldwide. People have been used to posting images and text to present themselves, articulate their social circles, and interact with each other in OSNs. However, analysing OSNs is very challenging and expensive due to their enormous capacity and complex structure.

This part of the thesis presents two interesting OSN analysis projects that help readers better understand what is posted in OSNs and discover potential relationships to facilitate user engagement.

In Chapter 7, we introduce a real-world dataset collected from Instagram and systematically describe the relationship between posted images and their associated hashtags. Then we propose a new graph-based approach *semantic image graph embedding* (SIGraph), to model and mine this relationship and apply it to improve the performance of the image multi-label classification task.

Chapter 8 introduces the *neural link prediction* (NeuLP) model to find the new social relationship between OSN users. Specifically, we utilise graph neural networks to mine users' local structure information and leverage neural collaborative filtering to capture long-range user interactions to obtain useful user representations for interaction prediction. Extensive experiments on real-world OSN datasets demonstrate the effectiveness of NeuLP and illustrate the availability of NeuLP on users with different attributes.

Chapter 7

A Graph-based Approach to Explore Relation between Hashtags & Images

7.1 Introduction

The last decade has witnessed the rapid development of *online social networks* (OSNs). To some extent, OSNs have mirrored our real society: people perform various activities in OSNs as they do in the offline world, such as establishing social relations, sharing life moments, and expressing opinions about various topics.

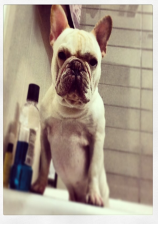
Image is one of the most popular information being shared in OSNs. For instance, 300 million photos are uploaded to Facebook on a daily base[1]. Moreover, there exist several popular OSNs dedicated to image sharing, including Instagram and Flickr. Images themselves are a rich source of information. Previously, researchers have studied images in OSNs from various perspectives [DWP+15, WYHY15, ZLK+18]. These works mainly concentrate on the contents of the images, thus adopting computer vision techniques as the main instrument. Different from images hosted on other platforms, images in OSNs are often affiliated with other types of user-shared information, such as image captions and hashtags. Such information can contribute to understanding OSN images as well. However, the relationship between the images and user-shared information has been left mostly unexplored. In this chapter, we aim to fill this gap by analysing the relationship between images and hashtags.

A hashtag is a single word or short phrase prefixed by the "#" symbol [DWP+15]; it is initially created to serve as a metadata tag for people to efficiently search for information in OSNs. Interestingly, hashtags themselves have evolved to convey far richer information than expected and provide an incredibly varied and nuanced method for describing images. Some hashtags describe precise objects in the images, e.g., #glass, #window, #building, and #sky; some are related to the feelings and intent of the users, such as #lovelyday, #whyme, and #celebrating; others refer to some event or geographic position, e.g., #paris, #rio, and #newyork [ZHR+18]. Besides, users also create many hashtags to convey meanings that previously did

[1] https://zephoria.com/top-15-valuable-facebook-statistics/

not exist in human languages, e.g., #tbt (an abbreviation for "Throw Back Thursday").

Table 7.1: Two example images from Instagram. Hashtags are generated by users, and labels are provided by Google's Cloud Vision API.

		
Hashtags	**#frenchbulldog #bulldog** #illustration **#dog** #LikeASir #Sir	**#dog** #frenchie **#frenchbulldog** **#bulldog** #puppy #igdaily
Labels	Nose, Art, Drawing, Sketch, Dog Like Mammal, Black And White, Visual Arts, Eye, Illustration, Human	Dog, Vertebrate, French Bulldog, Toy Bulldog, Non Sporting Group, Companion Dog, Puppy, Puppy Love

In this chapter, we perform an empirical study on the relationship between hashtags and image contents. Our experiments are conducted on a real-world dataset collected from Instagram. It is worth noting that as it is time-consuming to tag the image contents for all the images in our dataset manually (148,106 images), we use the image *labels* obtained from an automatic image detection tool, i.e., Google's Cloud Vision API, to represent the image contents.

Relationship verification & quantification. We first verify the relationship between hashtags and image contents (represented by their labels) using the two-sample *Kolmogorov-Smirnov* (KS) test. Experiments demonstrate that hashtags are indeed related to image contents with a significance level $\alpha = 0.001$.

Furthermore, we model the relationship between hashtags and images (i.e., their labels) as bi-directional prediction tasks, i.e., using an image's associated hashtags to predict the image's labels (H2L) and using an image's labels to predict its hashtags (L2H). The prediction performance is then used to describe the strength of the relationship between images and hashtags. For the H2L task, a straightforward approach is to use word embedding methods [MCCD13] to transform hashtags into continuous vectors, representing hashtag semantics, which are later used as features to train a machine learning classifier. A similar approach can be applied to the L2H task as well, namely, to use the obtained label vectors from word embedding methods to predict image's hashtags. However, this approach only considers the semantic meaning of hashtags (and labels), while neglecting connections among the images. As demonstrated by the example in Table 7.1, if two images share a few hashtags (i.e., #dog, #bulldog, #frenchbulldog), then their contents may have certain similarities as well (i.e., both are about pet dogs). To this end, we propose a graph-based approach, which can explore both semantic information of hashtags (and labels) and the graph structure among the images to measure the relationship

between hashtags and images.

Through extensive experiments, we show that our approach has the best prediction performance – 26.84% per-class precision (C-P) for the H2L task and 30.81% C-P for the L2H task on our Instagram dataset. Compared with the approach based on word embedding, it achieves 7.7% and 7.8% C-P gain for the two tasks, respectively.

Application. After verifying and quantifying the relationship between hashtags and images, we further explore this relationship to improve one downstream task – image multi-label classification. Experiments on the *NUS-WIDE* dataset [CTH+09] show that we can achieve a 4.0% C-P gain over a state-of-the-art method. This result further shows that hashtags can indeed help to improve online image classification.

7.2 Additional Related Work

There has been a diverse array of academic works exploring the information contained in hashtags. Tsur et al. try to explore what information is contained in hashtags based on a massive dataset from Twitter [TR12], and they view hashtags as ideas that could express users. As a result, they present the richness of information in hashtags. Furthermore, some work use hashtags to detect the topic of tweets on Twitter [WWL+11] and predict hashtags based on tweet contents [GSN+13, SC14]. These work indicate there is a strong relationship between hashtags and text contents, and it is possible to make two-way predictions between them.

Focusing on the relationship between hashtags and images, Niu et al. propose a semi-supervised Relational Topic Model (ss-RTM) to use hashtags information to recognise social media images [NHGT14]. They first organise images into a network if they share some hashtags. Then, they treat this network as a document and use a statistical model RTM, which is widely used in natural language processing tasks to extract the topic relationship among documents, to extract images' relationships into representative vectors. Compared with our work, they only use hashtags' information to build up the network but ignore their semantic meaning in the final features. Besides, due to the computational cost of RTM, they cannot involve a large number of images in one network, and there might be a strong influence from noisy hashtags. Wang et al. propose a framework (CNN-RNN) that combines hashtags and image features to perform classification [WYM+16]. CNN-RNN mainly contains two parts – a CNN model for extracting semantic representations from the images and an RNN (recurrent neural network) to model image/labels relationship and hashtags dependency. Due to the advantages of RNN, this framework can utilise the order information among hashtags, and it can predict a long sequence of labels. It achieves better performance than ss-RTM, but it neglects the connections among images. Recently, Wang et al. utilise a hashtag-related knowledge graph to improve image multi-label classification [WYG18]. They first build a large knowledge graph, which contains millions of hashtags and their semantic relationships. Then they apply the deep graph embedding methods to extract hashtags' relationship to representative vectors and use the representative vectors to assist the classification task. But it is high-cost work to build up a knowledge graph with

millions of hashtags, and they only consider the hashtags semantic information but neglect the graph structure among the images.

7.3 Image-Hashtag Relationship Verification

Instagram is one of the most popular OSNs and a major platform for hashtag- and image-sharing. Therefore, we resort to Instagram to collect our dataset relying on its public API. [2] Our data collection follows a similar strategy as the one proposed by Zhang et al. [ZHR+18]. Concretely, we sample users from New York by their geo-tagged posts. Then, for each user, we collect all her images. In total, we obtain 10,605,399 images from 25,658 users. Then, we perform some pre-processing, filtering out those images with less than 3 hashtags.

As mentioned before, we represent image contents as labels. Manual labelling can be an option but not scalable. Instead, we adopt Google's Cloud Vision API [3] to label images. The Cloud Vision API is supported by pre-trained machine learning models; it describes an image's content as a list of labels. The detected labels cover various aspects of an image ranging from the contained objects to personal feelings, e.g., happiness. It is worth noticing that this API has been already used in social media image analysis before [RT18]. Table 7.1 depicts two images labelled by Google's Cloud Vision API.

In total, we have spent 227$ on labelling 148,106 images. There are 255,298 different hashtags associated with these images. On average, each image has 6.46 hashtags, and each hashtag can appear in 4.19 images. We can see that the images with 3 hashtags have the largest count, and most images have less than 10 hashtags.

For all our images, Google's Cloud Vision API provides 6,327 different labels. On average, each image contains 8.27 labels. Google's Cloud Vision API gives at most 10 labels for one image, thus the amount of images with 10 labels is much more than the amount of images with other numbers of labels (< 10).

From the example in Table 7.1, we can confirm that the labels given by Google's Cloud Vision API can sufficiently describe the image contents. It can find the objects (e.g., Dog Like Mammal, Dog, Puppy) in the images, and give the style (e.g., Art, Drawing) and feeling (e.g., Puppy Love) of images. Besides, we also find out that some hashtags have a close relationship with image contents, e.g., #dog, #frenchbulldog, #illustration, #puppy, and some of them describe additional information, e.g., #LikeASir, #Sir. However, there are also some other hashtags that do not have too much relation with the image's contents, e.g., #igdaily.

To verify the existence of the relationship between hashtags and image labels, we perform a two-sample KS test. We construct two vectors $\mathbf{hc}_c$ and $\mathbf{hc}_d$ with an equal number of elements, where each element in $\mathbf{hc}_c$ is obtained by calculating the appear ratios of labels in images that have one specific hashtag and similarly each element in h_d is the appear ratio score of labels in images that do not have

[2] The dataset was collected in January 2016 when Instagram's API was still publicly available.
[3] `https://cloud.google.com/vision/`

this hashtag. We perform a two-sample KS test on vectors $\mathbf{hc}_c$ and $\mathbf{hc}_d$. The null hypothesis here is that the appearing ratio of labels in images with one specific hashtag does not differ from images without this hashtag, i.e., these two vectors are the same, $H_0 : \mathbf{hc}_c = \mathbf{hc}_d$. Another hypothesis is that the appearing ratio of labels in images with one specific hashtag differs from images without this hashtag. Therefore, we have the following two-sample KS test:

$$H_0 : \mathbf{hc}_c = \mathbf{hc}_d, \quad H_1 : \mathbf{hc}_c \neq \mathbf{hc}_d$$

. The two-sample KS test result suggests a strong evidence with a significance level $\alpha = 0.001$ (p-value$= 1e - 91$) to reject the null hypothesis. As a result, we confirm that there exists a relationship between hashtags and image contents.

7.4 Quantifying Image-Hashtag Relationship

In the previous section, we have demonstrated the existence of the relationship between hashtags and image contents (through examples and a statistical test). In this section, we systematically quantify this relationship. Table 7.2 lists the additional mathematical notation used in this chapter.

Table 7.2: Summary of additional notations.

Notation	Description
$\mathcal{I}$	Set of images
H_0, H_1	Hypothesis
H_i	List of hashtags associated with image i
L_i	Set of labels associated with image i
H	Set of all hashtags of all images
L	Set of labels of all images
$\mathcal{G}_1$	Graph consists of all images and their hashtags
$\mathcal{G}_2$	Graph consists of all images

Our idea for quantification is to model the relationship between hashtags and images as bi-directional prediction tasks, i.e., using an image's associated hashtags to predict the image's labels (H2L) and using an image's labels to predict its hashtags (L2H). The prediction results can be used to quantify the relationship strength – higher prediction performance indicates a stronger relationship.

In the rest of the section, we first discuss how to use word embedding methods to extract semantic meaning for hashtags and labels for our prediction tasks (Section 7.4.1). Then, we present a shallow graph embedding based approach in Section 7.4.2 and propose a new deep graph embedding based approach in Section 7.4.3. The experimental results are presented in the end (Section 7.4.4. For presentation purposes, we use H2L as an example task, similar approaches can be described for the L2H task.

7.4.1 Word Embedding based Approach

We use $\mathcal{I}$ to represent the set of images. Each image i is associated with a list of hashtags $\mathrm{H}_i = \{h_1, h_2, \ldots, h_{m_i}\}$ and a list of labels $\mathrm{L}_i = \{\ell_1, \ell_2, \ldots, \ell_{n_i}\}$. m_i and n_i denote the number of hashtags and labels in an image i, respectively. $\mathcal{H}$ represents all the hashtags.

For our H2L task, one approach is to use hashtags' semantic meaning as the features to train a machine learning classifier to predict image labels. Concretely, we rely on word embedding to transform each hashtag into a continuous vector and average the vectors of all hashtags of an image as its feature. To train hashtag embedding, we adopt the Word2vec model [MCCD13], meaning that we treat each image's associated hashtags as a "phrase", and all these phrases form a "corpus". The learning process follows the same objective function as Skip-Gram, by applying stochastic gradient descent, we can embed each hashtag into a vector space.

7.4.2 Shallow Graph Embedding based Approach

The above word embedding based approach only considers the semantic meaning of hashtags (and labels) while neglecting connections among the images. In the example depicted in Table 7.1, if two images share some hashtags, then their contents share certain similarities as well. We hypothesise that connections among images also possess a strong signal for our prediction task, thus we aim for a method to summarise this information as new features.

Our first idea of feature extraction is to organise images in a graph according to the connections among them and extract images' connection information that is represented in the graph. The first graph we construct is $\mathcal{G}_1 = (\mathcal{H}, \mathcal{I}, \mathcal{E}_{HI})$. $\mathcal{G}_1$ contains two types of nodes: hashtags ($\mathcal{H}$) and images ($\mathcal{I}$), each image node connects with its hashtags, and each hashtag node connects with images that it appears with (edges in $\mathcal{E}_{HI}$). The state-of-the-art method to extract information from a graph is graph embedding, which aims to learn a mapping that embeds nodes as points in a low-dimensional vector space [HYL17a]. Through optimising this mapping, geometric relation in this learned space reflects the attributes of the original graph.

The shallow graph embedding method we adopt is DeepWalk [PARS14], it is inspired by the idea of word embedding. We treat a graph as a "document" and sample sequence of nodes by random walk on the graph as a "phrase". Then, word embedding methods can be applied to these phrases as a traditional document task to return the feature vectors of image nodes.

7.4.3 Deep Graph Embedding based Approach

In the above discussions, Skip-Gram can use the hashtags' semantic meaning, and DeepWalk further explores connections among the images. A natural question to ask is whether we use both of these two information within one framework.

To achieve this, we propose a new graph-based approach *semantic image graph embedding* (SIGraph), which could effectively explore both the hashtags' (and labels') semantic information and the graph structure among the images. To construct the graph structure among the images, we add an edge between two images if these two images share at least n common hashtags.

To extract information from $\mathcal{G}_1$, SIGraph uses a deep graph embedding method GraphSAGE [HYL17b], which works for computing node representations in an inductive manner. This technique operates by sampling a fixed-size neighbourhood of each node and then performing a specific aggregator over it (such as the mean over all the sampled neighbours' feature vectors or the result of feeding them through a recurrent neural network). This approach has yielded impressive performance across several large-scale benchmarks [VCC+18].

7.4.4 Experiments

We evaluate the three approaches proposed in Sections 7.4.1- 7.4.3 on the bi-directional prediction tasks (H2L and L2H) on our Instagram dataset to quantify the relationship between hashtags and images.

Evaluation metrics. We adopt those evaluation metrics that are widely used in multi-label image classification fields [WYM+16], including per-class and overall precision (C-P & O-P), per-class and overall recall (C-R & O-R), per-class and overall F1 score (C-F1 & O-F1).

The precision is the number of correctly predicted labels (or hashtags) divided by the number of predicted labels (or hashtags); the recall is the number of correctly predicted labels (or hashtags) divided by the number of ground-truth labels (or hashtags); the F1 score is the geometrical average of the precision and recall scores. Per-class means the average is taken over all classes, and overall means the average is taken over all testing examples. Moreover, we only consider the top 3 predictions for both tasks in our evaluation.

Preprocessing. We adopt the following steps to prepare our dataset and build the graph ($\mathcal{G}_2$). We first convert hashtags into lowercase and delete punctuation. Second, as multiple hashtags may refer to the same underlying concept, we apply a simple process that utilises WordNet [Mil95] synsets to merge some hashtags into a single canonical form, such as "coffeehouse" and "coffeeshop" to "cafe". While building up the graph ($\mathcal{G}_2$) for the SIGraph approach, we require there are at least $n = 3$ common hashtags (or labels) between two images in order to add an edge between them. Third, for the H2L task: we first delete hashtags that appear less than 50 times in the dataset, then we select the most frequent 100 labels from the dataset and keep images with these labels. For the L2H task: we first select 5 hashtags for each image and keep 50 most frequent hashtags from the dataset; then we delete labels that appear less than 50 times in the dataset. After the preprocessing, we randomly select 13,107 and 18,372 images for H2L and L2H tasks, respectively.

Implementation details. For fairness, the default embedding dimension d in this chapter is set to 256. For the approach based on word embedding, we adopt the

Skip-Gram implementation provided by *gensim* [ŘS10], and keep the default parameters provided by the software. For the approach based on shallow graph embedding, i.e., DeepWalk, we set the length of each walk to 20 and the number of walks per image node to 5.

For our approach SIGraph, we develop a supervised multi-label GraphSAGE model[4] based on the Pytorch implementation of Hamilton et al. [HYL17b], and use the output of the last layer's encoder as the embedding. We use a 2-layer neural architecture with 256 hidden units in each layer, the sample size of the two layers is 5, and the mini-batch size is 256. Learning rate is set to 0.1 with an iteration number of 500.

In the end, we need to feed these extracted features into a logistic classifier to make predictions. In this way, we evaluate the following three approaches to our prediction tasks: Word2vec+logistic, DeepWalk+logistic, and SIGraph+logistic. All experiments are implemented with the support of GPU (K80).

Results. We can see that all the C-P scores are no less than 19% and the O-P scores are no less than 30% for all three approaches. Moreover, the results of the DeepWalk+logistic and SIGraph+logistic approaches are better than the results of the Word2vec+logistic approach (a 3.5% C-P gain and a 2.6% O-P gain for DeepWalk+logistic, and a 7.7% C-P gain and a 6.4% O-P gain for SIGraph+logistic). This indicates that the images' graph structure provides a stronger signal than using hashtag semantics only for our prediction task. The SIGraph+logistic approach achieves a significant improvement compared with the other two. When compared with the DeepWalk+logistic approach, there is a 4.2% C-P gain and a 3.8% O-P gain. This further demonstrates the superior performance of SIGraph.

We can find that all the C-P scores are more than 25% and the O-P scores are more than 32% for these three approaches. Similarly, the SIGraph+logistic approach achieves a significant improvement compared with the other two. For the C-P scores, its performance gain is 4.2% compared with the DeepWalk+logistic approach and 5.8% compared with Word2vec+logistic. For the O-P scores, its performance gain is 4.8% compared with the DeepWalk+logistic approach and 6.1% compared with Word2vec+logistic. At last, we can see that for the C-R results, the performance of the DeepWalk+logistic approach (15.47%) is worse than the Word2vec+logistic approach (15.83%), and the C-P gain (1.6%) and O-P gain (1.3%) between these two approaches are smaller than the gains for the H2L task. It indicates that for the L2H task, the information provided by the image graph structure has a similar strength as only exploring labels' semantic meaning.

The influence of n. We further evaluate the influence of the hyper-parameter n on the prediction performance of our SIGraph+logistic approach, where n is the number of common hashtags required to establish an edge between two images in $\mathcal{G}_2$. For the H2L task, we compare the different results with an equal number of images, the prediction performance in general gets better if n becomes bigger. This suggests that more hashtags can better characterise the connections among images.

[4]Our experimental code is available upon publication.

7.5 Application

After verifying and quantifying the relationship between hashtags and images, we focus on whether this relationship can be used to improve a downstream task. In particular, we aim to use hashtags' information summarised by the approach SIGraph to improve the performance of a baseline model on multi-label classification task.

For this task, we use the *NUS-WIDE* dataset, which contains human-generated labels and hashtags shared by real users. *NUS-WIDE* is a web image dataset [CTH+09], and it contains 269,648 images from Flickr. [5] It has two types of hashtags: (i) 5018 unique hashtags (5018-hashtags); (ii) 1,000 cleaner hashtags without noisy and rarely-appearing hashtags. We could see that the most frequent numbers of hashtags with images are 4, 5, or 6, and this dataset has quite some images with less than three hashtags.

The images in the dataset are also manually annotated using 81 labels by human annotators, which cover different aspects, including object classes, scenes, and attributes. The labels on each image are considered as ground truth to represent the image contents.[6] On average, each image contains 1.87 such labels. We can find that images with only one label have the largest count, and there are only a few images with more than 8 labels.

Preprocessing. To demonstrate the application of the relationship between hashtags and images to improve the performance of image multi-label classification, we use a pre-trained *convolution neural network* (CNN) as the baseline approach to extract the image features (or image embedding). This technique has been successfully used for many image-related tasks, i.e., image classification [ARW+15, WYM+16], image recognition [SKP15], etc. Then, we use the returned image embeddings to train a classifier to make predictions. Second, we use the 5018-hashtags, in this way we keep all the information provided by users. Third, while building the graph structure, we use the same settings as in Section 7.4.4, i.e., if there are at least $n = 3$ common hashtags between two images, we add an edge between them. Finally, we randomly select 17,137 images from the *NUS-WIDE* dataset.

Implementation details. For the baseline CNN, we use 16 layers VGG network [SZ15] pre-trained on ImageNet 2012 classification challenge dataset [DDS+09] using Pytorch deep learning framework. For our SIGraph approach, we use the graph structure $\mathcal{G}_2$, and the parameter setting of GraphSAGE remains the same as discussed in Section 7.4.4. The dimensions of the CNN embeddings and the SIGraph embeddings are set as the same (256). To put embeddings together, we simply concatenate them. In the end, we feed these extracted features into a logistic regression classifier to make predictions.

Results. We use the same evaluation metrics as discussed in Section 7.4.4. Table 7.3 presents the classification results of approaches using the CNN embeddings, the

[5] https://www.flickr.com/

[6] This explains why we cannot directly use our Instagram dataset as we do not have such a ground truth.

Table 7.3: Comparison of the experimental results of the top 3 image multi-label classification on the *NUS-WIDE* dataset with 5018-hashtags.

Methods	**C-P**	**C-R**	**C-F1**	**O-P**	**O-R**	**O-F1**
CNN+logistic	41.99	32.35	33.55	53.59	59.35	56.32
SIGraph+logistic	23.90	18.72	17.79	45.79	50.71	48.13
CNN+SIGraph+logistic	**45.99**	**37.28**	**37.38**	**55.34**	**61.29**	**58.16**

SIGraph embeddings and the CNN+SIGraph embeddings, respectively. From the results, we could first find that the CNN+SIGraph+logistic approach can improve the performance when only using the CNN embeddings for multi-label classification (with 4.0% C-P gain and 1.75% O-P gain). Second, the performance of the CNN+logistic approach is better than the SIGraph+logistic approach (with 18.09% C-P gain and 7.8% O-P gain). This indicates that information provided by the image itself is stronger than the relationship information extracted by our approach.

Observations. In this section, we present detailed examples to understand the different predictions given by the CNN embeddings and the CNN+SIGraph embeddings.

Table 7.4: Two example predictions by the CNN approach and the CNN+SIGraph approach on the *NUS-WIDE* dataset.

		
5018-Hashtags	#film, #army #war, #historic	#fish, #photography #underwater
Labels (ground truth)	military, person	animal, coral, fish
Prediction (CNN)	person, sky, water	animal, coral, water
Prediction (CNN+SIGraph)	military, person, sky	animal, coral, fish

In Table 7.4, there are two images with their associated labels and hashtags from the *NUS-WIDE* dataset, as well as the predictions made by the two approaches based on the CNN embeddings and the CNN+SIGraph embeddings, respectively. For the image on the left, the CNN+logistic approach gives one correct prediction ("person") and two incorrect predictions ("sky", "water"). We notice that this image is somehow unclear and over-light. Since the CNN embeddings come from the image itself, it somehow mistakes this strong light in the background as "sky" or "water". Besides, the correctly predicted label "person" is one of the most popular labels in the dataset (24.6% images contain this label), so this label could not provide precise information to clarify the image contents. On the other hand, the CNN+SIGraph+logistic approach correctly predicts the two labels "military" and "person". This indicates that this approach can capture more comprehensive information about this image itself.

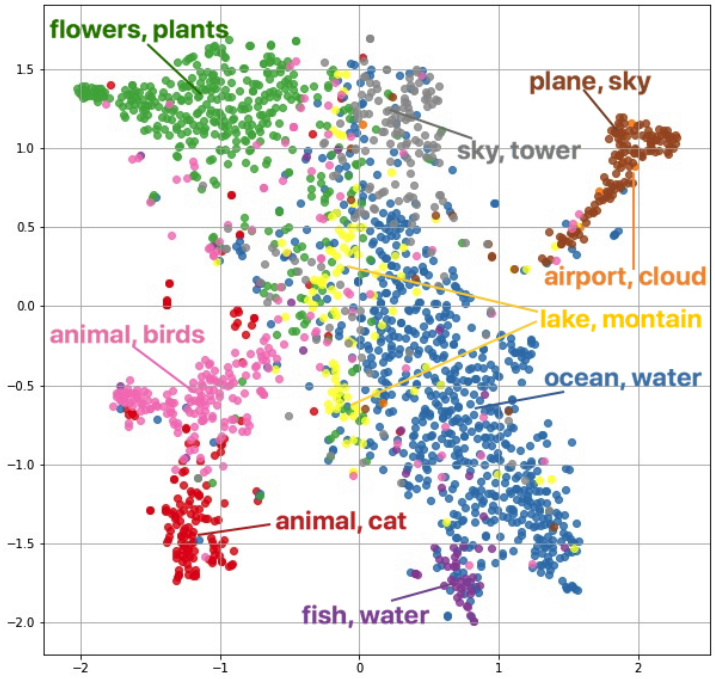


Figure 7.1: Visualisation of our SIGraph embeddings. The images information are mapped to the 2D space using the t-SNE package with learned SIGraph image embedddings as input. We select some labels: [“animal”, “cat”], [“ocean”, “water”], [“flowers”, “plants”], [“fish”, “water”], [“airport”, “clouds”], [“lake”, “mountain”], [“plane”, “sky”], [“animal”, “birds”] and [“sky”, “tower”] and collect images have these labels.

For the image on the right, the CNN+logistic approach gives two correct labels (“animal”, “coral”) and one incorrect label (“water”). However, this incorrect label is different from those two incorrect labels for the left image, as it is still relevant to the contents of the image. We could recognise that the image presents an underwater environment, so “water” is not wrong even it does not appear as one of the truth labels. The fish in the right image disguises itself in the environment. In this case, the visual CNN embeddings are not sufficient in capturing small objects (i.e., “fish") in the image. On the other hand, the CNN+SIGraph+logistic approach succeeds in predicting all three labels.

From these two example predictions on the *NUS-WIDE* dataset. We can confirm that our hashtag features through the SIGraph embeddings can provide useful information to improve the image multi-label classification even when the image quality is not good enough, or the objectives in the image are not easy to be found by visual features.

We are also interested in knowing whether the SIGraph embeddings could embed images into the correct position in the embedding space. More specifically, whether they can keep images with similar contents to be close in the space. For this aim, we select 9 groups of labels and each group to have 2 different labels and collect

images only containing one of the groups of labels. We then transform these image embeddings obtained with SIGraph into a 2D space using the dimensionality reduction algorithm t-SNE [vdMH08].

We visualise the result in Figure 7.1, and observe the existence of clustering structure in images' embeddings. These images with different groups of labels are separated into different clusters, and related clusters are close in the space. For instance, in the figure, we can find that the embeddings of images with labels ["animal", "cat"] are very close to images with labels ["animal", "birds"], as they are all animal-related images. And the embeddings of images with ["plane", "sky"] are far from the previous two types of images. Besides, we can also find that images labelled by ["lake", "mountain"] are mixed with the images labelled by ["ocean", "water"]. This is due to the contents of the two images having similar semantics.

7.6 Conclusion and Future Work

In this chapter, we have performed an empirical study on verifying and quantifying the relationship between hashtags and images based on real-world datasets collected from Instagram and Flickr, and we successfully applied the verified relationship to improve a downstream task.

We have implemented a statistical test to verify the existence of the relationship between hashtags and images. Then, we designed bi-directional prediction tasks (H2L and L2H) and used the prediction performance to quantify the relationship. In particular, we proposed a new graph-based approach to integrate both the semantic meaning of hashtags (and labels) and the graph structure of the images, which indeed help to extract more comprehensive information for hashtags (and labels). In the end, we successfully applied the extracted features of hashtags from the H2L task to improve the performance of image multi-label classification and achieved a 4% per-class precision gain compared to a state-of-the-art method.

Hashtags can be naturally organised into different categories according to their semantics and functions. In the future, we will first focus on the influence of hashtag categories, i.e., investigating the different relationship strengths between each category of hashtags and images. Second, in OSNs, different users have different habits of using hashtags, and we hypothesise that the richness of the semantic meaning contained in their hashtags could be different. How to explore these user differences is one of our future work. Third, so far, we have only applied the extracted hashtag features for an image multi-label classification task in this chapter. We want to further utilise the extracted label features (from the L2H task) to perform hashtag recommendation in OSNs.

Chapter 8

NeuLP: An End-to-end Deep-learning Model for Link Prediction

8.1 Introduction

The development of *information and communication technologies* (ICT) has changed people's lifestyle drastically over the last decades. Nowadays, people use *online social networks* (OSNs) to communicate with each other, maintain social relationships, and share life moments in various forms, such as online status, location check-ins, tweets and hashtags. We term all such user-shared information as user *attributes*, and OSNs with user attributes as *attributed social networks*. The enormous data generated by OSNs thus provide researchers with an unprecedented opportunity to gain a deeper understanding of our society. Many data mining problems have been proposed and extensively studied in the literature, such as link prediction, attribute inference, and social influence quantification.

Link prediction is one of the key problems in OSN mining, which aims to estimate the likelihood of the existence of an edge (relationship) between two nodes (users) [LK07]. It has been widely applied for friendship recommendation in OSNs, which is important to increase user engagement and leads to a more satisfactory user experience. Besides OSNs, link prediction has drawn the attention of many different application domains and researches in various fields, such as the study of protein-protein interaction networks [LR12], and identifying hidden or missing criminals in terrorist networks [CGK+11].

The existing methods for link prediction in OSNs can be classified into three categories, ranging from the early methods focusing on hand-crafted features (e.g., see methods based on user profiles [BGW11] and graph structure [LK07, LLC10]); shallow graph embedding-based methods, such as LINE [TQW+15] and node2vec [GL16]; to recently emerged *graph neural network* (GNN)-based methods [KW17, HYL17b, VCC+18]. Among these, GNNs are currently the most popular paradigm, largely owing to their efficient learning capability. In contrast, hand-crafted features methods are limited in extracting latent information in the OSNs, and shallow graph embedding-based methods cannot incorporate user attributes.

GNN-based methods follow a common scheme that explores user information from either social graph or user attributes as node embeddings and then calculates user similarity for making link prediction. However, we argue that it is insufficient since the classic GNNs cannot capture the global information for a given user pair in an OSN. Current GNNs generally follow message-passing mechanism and assume that with the use of enough GNN layers (times of interactive aggregation), the model can capture long-range dependencies. However, recent research shows that GNNs are not robust to multiple layers since they would become over-smoothing and lead to vanishing gradient during training [LHW18, LMQ+19]. Therefore, existing GNNs have shallow architecture, $2-4$ layers. It causes nodes to have indistinguishable embedding vectors if locally they are at similar positions of an OSN. To fill this gap, You et al. propose a novel GNN model (P-GNNs) that could capture the position of a given node with respect to all other nodes in the graph [YYL19]. However, the introduced anchor-set sampling brings high computational costs and performs less well on attributed graphs.

In this chapter, we propose a unified end-to-end deep learning model, namely *neural link prediction* (NeuLP), which could efficiently integrate neural collaborative filtering to overcome the above limitations, and NeuLP is flexible to handle different types of user attributes for link prediction. We first use two shallow GNNs to transform user information (their social relations and attributes) into embeddings that interactively aggregate user features from neighbours. We then add a fusion layer to merge embeddings of two users into one vector, which performs element-wise product on two user embeddings to capture the linearity interactions between two users. However, a simple element-wise product does not account for the non-linearity interactions between users, so we further apply *multilayer perceptron* (MLP) on the vector to map it to low dimensions. The MLP can explore the non-linearity interactions between two users, where each layer performs a non-linear transformation with an activation function. This architecture is inspired by the recent progress of recommendation systems [HLZ+17] which is useful for mining online user behaviours. In the end, we utilise a linear transformation layer to summarise the extracted features of a pair of users as the possibility of existing a link between them.

We conduct extensive experiments on five benchmark datasets and two Instagram datasets in various pairwise prediction tasks to evaluate the performance of NeuLP with respect to several state-of-the-art baselines, including (1) link prediction with only graph structures and (2) link prediction on attributed social network datasets with one/two type(s) of user attributes. The experimental results demonstrate our model's significant improvement over the baselines, with up to 5.8% improvement in terms of the AUC score, and its capability in leveraging user attributes and interactions when compared with GNN-based baselines. Moreover, NeuLP achieves a strong link prediction when given users' two different types of attributes. We further perform in-depth analyses on the relation between prediction performance and the graph geodesic of two users in OSNs. This demonstrates that our model successfully used user interactions to improve GNN-based baselines.

8.2 Additional Related Work

We briefly review the state-of-the-art link prediction in OSNs, including methods based on hand-crafted features, shallow graph embedding and GNNs [MO19]. The hand-crafted features-based methods extract the feature hidden inside the user attributed and edges' structures. A heuristic score is used to measure the similarity and connectivity between two users [LK07, LLC10, BGW11]. These methods are pellucid and efficient but cannot explore the latent information in OSNs.

Shallow graph embedding-based methods learn a link's formation mechanism from the graph other than assuming a particular mechanism (e.g., common neighbours), and they can be summarised into two groups, i.e., DeepWalk-based methods and matrix factorisation-based methods. DeepWalk [PARS14] pioneers graph embedding methods by considering the node paths traversed by random walks over the graph as sentences and leveraging Skip-gram model [MCCD13] for learning node representations. On the other hand, some works adopt the idea of matrix factorisation for graph embedding [QDM$^+$19, LMK$^+$19]. However, these methods cannot deal with user attributes naturally and cannot optimise parameters for a specific task.

In the past few years, many studies have adopted GNNs to aggregate node information in arbitrary graph-structured data for link prediction. Most of the existing GNN models (e.g., see [KW17, HYL17b, VCC$^+$18]) rely on a series of graph message-passing architectures that perform the convolution in the graph domain by aggregating node feature messages from its neighbours in the graph and stacked multiple GNN layers can capture the long-range node dependencies. However, existing GNNs can only adopt the shallow architecture, with $2-4$ layers, which is not enough to get global graph structure information. You et al. [YYL19] recently proposed a global position-aware information capture mechanism to overcome this limitation. However, the introduced anchor-set sampling incurs high computational costs and makes the approach perform less well on attributed datasets.

8.3 Framework

Problem setup. Given a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ or an attributed social network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, the goal of the link prediction problem is to learn a scoring function f_θ: $\mathcal{V} \times \mathcal{V} \longrightarrow \mathbb{R}$ for predicting a new link between an unlabelled pair of users in a set $\mathcal{E}_p := (\mathcal{V} \times \mathcal{V}) \setminus \mathcal{E}$.

Our proposed model NeuLP has three main components as shown in Figure 8.1: partial aggregation, information fusion and model prediction. In the partial aggregation part, we use the input layer to get the encoded user attributes and feed them into GNN layers. Then, the GNN could iteratively aggregate information from a user's local neighbourhoods in the social network and update the user presentation (embedding). After obtaining user embeddings through partial aggregation simultaneously, we add a fusion layer to merge embeddings of two users into one vector

and further apply MLP on the vector to explore the non-linearity interactions between these two users. At last, we adopt a prediction layer to linearly summarise the extracted feature of two users as the possibility of existing an edge between them.

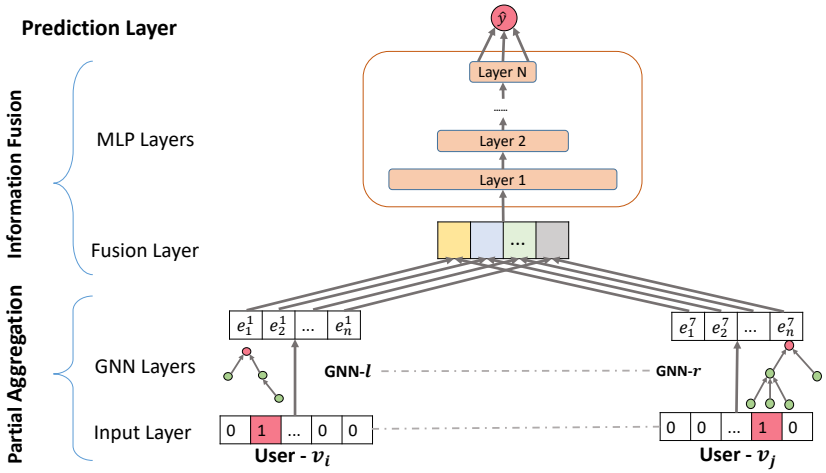


Figure 8.1: NeuLP's model architecture. The two input user feature vectors (left: v_i, right: v_j) are firstly transformed with two GNNs and further fused with multiple propagation layers, and the output is the predicted possibility of existing friendship between v_i and v_j.

8.3.1 Partial Aggregation

In this subsection, we describe the details of adopting GNNs to iteratively aggregate the partial information from a user's local network neighbourhood and update its embedding.

Input layer. The bottom input layer consists of two feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, where $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$, that describe user v_i and user v_j, respectively. They can be customised to support a wide range of user attributes, such as location check-ins and posted hashtags.

GNN encoding. We adopt the GNN to iteratively aggregate neighbour information for each user in its sub-network and update their embeddings. More specifically, we use *graph convolutional networks* (GCN) [KW17] as an example GNN, due to its widespread use and remarkable contributions. GCN operates directly on a graph and induces node feature vectors from the properties of their neighbourhoods. The model can be built by stacking multiple convolutional layers to involve information from farther neighbours.

Formally, given an attributed social network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, and its adjacent matrix $\mathbf{A}$, a GCN layer (Equation 2.3) can be formally represented as:

$$\mathbf{H}^{(\ell+1)} = \text{ReLU}(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}) \tag{8.1}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\hat{\mathbf{D}} = \sum_j \hat{\mathbf{A}}_{ij}$ and $\mathbf{W}^{(\ell)} \in \mathbb{R}^d$ is a trainable weight matrix for layer ℓ. $\mathbf{H}^{(\ell)}$ is the generated node representation of layer ℓ which is defined as the node

representation matrix $\mathbf{Z} = \mathbf{H}^{(\ell)}$. The intuition is that nodes aggregate information from their local neighbours at each layer. Therefore, we get two embedding matrices $\mathbf{Z}_1$ and $\mathbf{Z}_2$ from the left and the right GCNs, respectively.

8.3.2 Information Fusion

We continue to describe the information fusion layer and how to explore non-linearity user interactions with the MLP layers. The information fusion module is mainly inspired by the neural collaborative filtering [HLZ+17] that theoretically proves how does this module estimate classic neural collaborative filtering to learn global user interactions.

Fusion layer. In this layer, we merge embeddings of two users ($\mathbf{z}_i \in \mathbf{Z}_1, \mathbf{z}_j \in \mathbf{Z}_2$) into one vector. In detail, inspired by the *neural matrix factorisation* model [HLZ+17], we define the fusion function as:

$$\mathbf{z}_0 = \phi(\mathbf{W}^{\mathrm{T}}\mathbf{z}_i \odot \mathbf{z}_j) \tag{8.2}$$

where $\odot$, ϕ and $\mathbf{W}$ denote the element-wise product of vectors, activation function and edge weights of the output layer, respectively. The benefit of adopting the fusion layer instead of concatenating two embeddings together is that we can extract the linearity pair-wise interaction between two users. This compensates for the insufficiency of GCNs to acquire global user interactions.

MLP layers. Since the fusion layer analogues the matrix factorisation model to capture the linear interaction between two users. However, simply capturing linearity user interaction is not enough. To address this issue, we propose to add hidden layers on the fused vector, using a standard MLP to extract the non-linear interactions for pairs of users. In this sense, we can endow the interactions between v_i and v_j, rather than the way of fusion layer that uses only a fixed element-wise product on them. More precisely, the MLP layers in our NeuLP model is defined as:

$$\begin{aligned}
\mathbf{z}_1 &= \phi_1(\mathbf{W}_1^{\mathrm{T}}\mathbf{z}_0 + \mathbf{c}_1), \\
\mathbf{z}_2 &= \phi_2(\mathbf{W}_2^{\mathrm{T}}\mathbf{z}_1 + \mathbf{c}_2), \\
&\dots \\
\mathbf{z}_m &= \phi_m(\mathbf{W}_m^{\mathrm{T}}\mathbf{z}_{m-1} + \mathbf{c}_m)
\end{aligned} \tag{8.3}$$

where $\mathbf{W}_m$, $\mathbf{c}_m$, and ϕ_m denote the weight matrix, bias vector, and activation function for the m-th layer's perceptron, respectively.

8.3.3 Model Prediction and Optimisation

After the partial aggregation and information fusion layers, we obtain a representation vector $\mathbf{r}_{ij} = \mathbf{z}_m$ for each pair of users (v_i and v_j), which presents the latent features between them. As such, in the last prediction layer, we summarise the latent features as a predicted score which represents the possibility of existing a link

between these two users:

$$\hat{y}_{ij} = \sigma(\mathbf{W}_f^{\mathrm{T}} \phi_{m+1}(\mathbf{z}_m)) \tag{8.4}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ to scale the output into $(0, 1)$. With this, we successfully sum up the comprehensive information between v_i and v_j into a similarity score $\hat{y}_{ij}$ for link prediction.

To learn model parameters, we optimise the Binary Cross Entropy Loss (BCELoss), which has been intensively used in link prediction models. We adopt the mini-batch Adam [KB15] to optimise the prediction model and update the model parameters. In particular, for a batch of randomly sampled user tuple (v_i, v_j), we compute the possibility $\hat{y}_{ij}$ of existing a link between them after partial aggregation and information fusion, and then update model parameters by using the gradient of the loss function.

8.4 Experimental Evaluation

In this section, we conducted intensive experiments in order to answer the following research questions:

- **RQ1**: How effective is NeuLP when compared with the state-of-the-art baselines focusing on graph structures?
- **RQ2**: How does NeuLP perform when compared with the state-of-the-art baselines for attributed social network datasets with only one type of user attribute?
- **RQ3**: Can we utilise NeuLP for link prediction with two different types of user attributes? How effective will it be?

8.4.1 Dataset Description

We evaluate our model on five widely used datasets for graph-related machine learning tasks: USAir, Yeast, Cora, Email and Citeseer. They are publicly accessible on the websites, together with their dataset descriptions.[1] We further use an Instagram dataset, which we collected from Instagram relying on its public API.[2] Our data collection follows a similar strategy as the one proposed by Zhang et al. [ZHR+18]. Concretely, we sample users from New York by their geotagged posts. Then, for each user, we collected all her/his posted hashtags and further performed the following preprocessing to filter out the users matching any of the following criteria: (i) users whose number of followers are above the 90th percentile (celebrities) or below the 10th percentile (bots); (ii) users without location check-ins or posted hashtags. Then we generate two datasets, Instagram-1 and Instagram-2, according to different filter conditions:

[1] https://linqs.soe.ucsc.edu/data, and http://snap.stanford.edu/data/index.html

[2] The dataset was collected in 01/2016 when Instagram's API was publicly available.

- Instagram-1: users with no less than 100 location check-ins and with no less than 10 friends.
- Instagram-2: users with no less than 100 location check-ins and with no less than 5 friends.

Table 8.1: Statistics summary of the seven datasets.

Statistics	USAir	Yeast	Cora	Email	Citeseer	Instageam-1	Instagram-2
# of Nodes ($\|\mathcal{V}\|$)	332	2,375	2,708	799	3,312	6,815	12,944
# of Edges ($\|\mathcal{E}\|$)	2,126	11,693	5,429	10,182	4,660	36,232	61,963
Node features ($\mathbf{X}$)	No	No	Yes	No	Yes	Yes	Yes

The reason for generating these two Instagram datasets is that we need datasets with different types of user attributes to discuss the possibility of applying NeuLP to attributed social networks with different types of user attributes (**RQ3**). The statistics of the seven datasets are given in Table 8.1.

For the experiments, we use two sets of 10% existing links and an equal number of nonexistent links as test and validation sets. We use the left 80% existing links and an equal number of nonexistent links as the training sets.

8.4.2 Experimental Settings

Evaluation metrics. Like existing link prediction studies [GL16], we adopt the most frequently-used metrics "Area under the receiver operating characteristic" (AUC) to measure the performance.

Baselines. To demonstrate the effectiveness of NeuLP, we compare its performance with several baseline methods: MF [ME11], NeuMF [HLZ+17], node2vec [GL16] and several state-of-the-art GNN variants, including GCN [KW17], GraphSAGE [HYL17b], GAT [VCC+18], and P-GNNs [YYL19]. Besides, we also adopt HGANE [JXZZ19], which is a collaborative graph embedding framework with a hierarchical graph attention mechanism.

In order to study the performances of NeuLP on the attributed datasets (link prediction using one same type or two different types of user attributes), we additionally compare our model with walk2friend [BHPZ17] and tag2friend [Zha19].

Implementation details. For the node2vec related approaches, i.e., node2vec, walk2friend, tag2friend, we use the default settings as in [GL16]. All the neural network-related baselines follow their original paper's code at their GitHub pages if available; otherwise, we implement it by following their original paper, e.g., HGANE. For NeuLP, we use a simple GNN model, GCN, for partial aggregation and the layer numbers of GCN and MLP are 2 and 4, respectively.[3] To make a fair comparison, all methods are set to have a similar number of parameters and hyper-parameters.

Table 8.2: Link prediction performance comparison with baseline methods on graph datasets (AUC). OOM: out of memory.

	USAir	**Yeast**	**Cora**	**Instagram-1**	**Instagram-2**
MF	0.831±0.010	0.903±0.003	0.816±0.005	0.740±0.013	0.714±0.008
NeuMF	0.801±0.015	0.907±0.005	0.686±0.009	0.744±0.001	0.727±0.001
node2vec	0.805±0.002	0.905±0.008	0.770±0.005	0.785±0.004	0.738±0.002
GCN	0.903±0.006	0.938±0.003	0.819±0.006	0.804±0.004	0.773±0.003
GraphSAGE	0.897±0.007	0.933±0.004	0.838±0.008	0.802±0.005	0.770±0.002
GAT	0.902±0.006	0.935±0.004	0.839±0.008	0.796±0.005	0.767±0.002
P-GNNs	0.911±0.018	0.940±0.006	0.852±0.008	0.734±0.007	OOM
HGANE	0.901±0.004	0.932±0.006	0.845±0.010	0.797±0.007	0.771±0.008
NeuLP	**0.952**±0.009	**0.965**±0.003	**0.894**±0.006	**0.813**±0.001	**0.790**±0.003

8.4.3 Performance Comparison

We first compare the link prediction performance of approaches on the graph datasets, which do not include any user-shared information. Here, we use the adjacent matrix **A** as the user features set. We compare the performance of NeuLP with the baselines, except for walk2friend and tag2friend, since the latter two require user-shared information.

From the results shown in Table 8.2, we first observe that our model consistently outperforms all the baselines with remarkable improvements. It indicates the effectiveness of NeuLP for link prediction with only graph structures. In particular, there is 4.9% AUC improvement on the Cora dataset. NeuLP gets 30.3% AUC improvement over NeuMF (i.e., on the Cora dataset), this indicates the observed graph plays an important role in link prediction, as NeuMF does not include the graph topology. NeuLP also outperforms P-GNNs on the Instagram-1 dataset (with 13.5% AUC improvement), which indicates the advantage of our model on real OSN datasets to capture their global information.

8.4.4 Attributed Online Social Networks

We continue to discuss the performance of various methods on attributed social networks. More specifically, we present the performance of NeuLP with one type of attribute (**RQ2**) and discuss the possibility of applying NeuLP on attributed social networks with different types of attributes (**RQ3**). We use walk2friend & tag2friend as the baselines for OSN datasets with location check-ins and hashtags, respectively, and several variant GNNs as baselines for both datasets.

Link prediction with one attribute type. Here we use the encoded user-attributes matrix as the user features set (i.e., one-hot encoding), where 1 means the user visited a location (or published a hashtag), and 0 otherwise. NeuLP* means we adopt the embeddings generated by random walk-based methods, e.g., walk2friend, tag2friend, as input user features.

[3]Code and datasets are available at https://github.com/zhiqiangzhongddu/NeuLP.

Table 8.3: Link prediction performance comparison between NeuLP and baseline methods on attributed social network datasets with one user attribute type (AUC). OOM: out of memory.

Attributes	Methods	Datasets	
		Instagram-1	Instagram-2
Location check-in	walk2friend	0.831±0.007	0.820±0.003
	GCN	0.808±0.005	0.781±0.003
	GraphSAGE	0.801±0.006	0.775±0.003
	GAT	0.784±0.006	0.758±0.008
	P-GNNs	0.686±0.012	OOM
	HGANE	0.830±0.007	0.799±0.002
	NeuLP	0.832±0.002	0.828±0.001
	NeuLP*	**0.885**±0.001	**0.880**±0.001
Hashtag	tag2friend	0.724±0.013	0.696±0.011
	GCN	0.812±0.004	0.781±0.003
	GraphSAGE	0.806±0.004	0.775±0.003
	GAT	0.774±0.012	0.763±0.009
	P-GNNs	0.717±0.010	OOM
	HGANE	0.819±0.005	0.781±0.004
	NeuLP	0.836±0.002	0.826±0.002
	NeuLP*	**0.880**±0.002	**0.875**±0.001

From Table 8.3, we can see that NeuLP outperforms walk2friend, tag2friend and other GNN models with up to 5.8% AUC improvement. Moreover, NeuLP* gets improvements over NeuLP on all three datasets with up to 6.4% AUC improvement, which means the information reflected by location check-ins or hashtags is a good supplement to the social network structure, and our model can well capture such information.

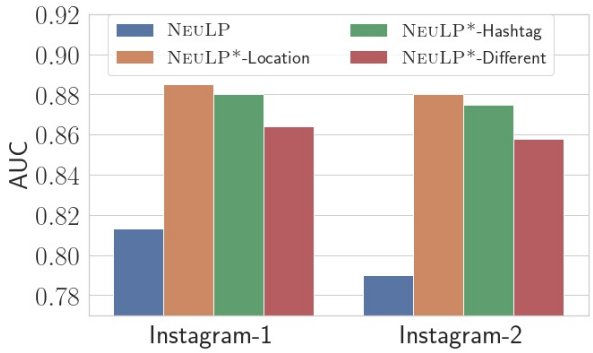
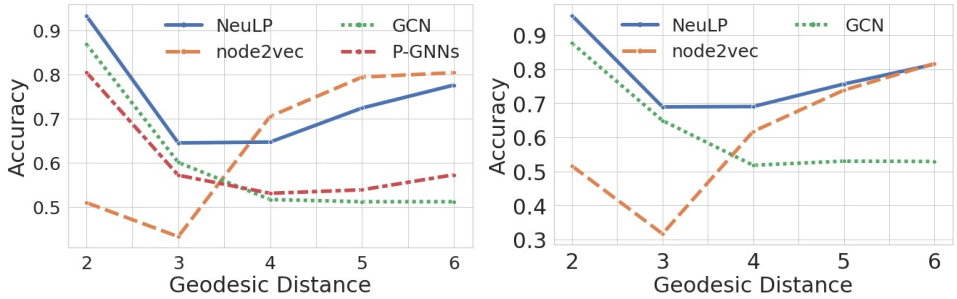


Figure 8.2: Link prediction performance comparison of NeuLP on the Instagram datasets. NeuLP: without user attributes. NeuLP*-feature: adopting random-walk related methods generated embeddings as user features.

Link prediction with two different attribute types. To answer the research question **RQ3**, we design experiments on attributed social networks with two different types of user attributes, which means performing link prediction given two user's different types of information, e.g., one user shares location check-ins, and the other user shares hashtags (see NeuLP*-Different in Figure 8.2). Following the previous settings, we randomly sample two groups of users where the first group of users

share their location check-ins and the second group sharing hashtags. These two groups account for half of all users. We organise two graphs where users with location check-ins are connected through locations and users with hashtags are connected through hashtags. Then we apply walk2friend and tag2fiend to these two graphs to generate user embeddings (model settings are as same as default settings). We use these two generated user embeddings matrix as two user features sets for left and right input, respectively.

From the experimental results are shown in Figure 8.2, we can see that NeuLP on attributed social networks with different types of user attributes gets significantly better performances compared with the results without using user attributes. NeuLP*-Different gets similar prediction performance on attributed social networks with the ones using one user attribute type, i.e., NeuLP*-Location and NeuLP*-Hashtag. It demonstrates the flexibility of our model in making link prediction, even when utilising different types of information that users share in OSNs.



(a) Instagram-1 (b) Instagram-2

Figure 8.3: Prediction accuracy of four link prediction methods on the Instagram datasets with different geodesic distances. P-GNNs lead to OOM on Instagram-2 dataset; thus not showing in Figure 8.3b.

Performance analyses with geodesic distances. In order to further investigate the prediction performances of different methods for different users pairs, we perform in-depth analyses on the relation between prediction performance and the graph geodesic of two users in OSNs. The results are drawn in Figure 8.3, which has two sub-figures to plot the prediction accuracy of the methods (node2vec, GCN, P-GNNs and NeuLP) with different user-pair geodesic distances (i.e., the number of edges in a shortest path connecting two nodes in the graph) for the two Instagram datasets. We see that the accuracy of GCN and P-GNNs significantly decreases when increasing the distance. This means GNN models cannot capture latent information for user pairs of large distances. In contrast, node2vec behaves in an opposite way: it achieves higher accuracy when increasing the distance, and it does not perform well when the distance is small. The reason might be that (long) random walks can capture long-distance information. However, NeuLP gets the most balanced performances, this demonstrates that it successfully uses both linearity and non-linearity user interactions to improve the performance of GNN-based models for link prediction.

8.5 Conclusion and Future Work

In this chapter, we have presented NeuLP as an end-to-end deep learning model for link prediction in OSNs. NeuLP incorporates linearity and non-linearity user interactions to overcome the limitations of current GNN-based link prediction methods. Compared with the state-of-the-art methods, NeuLP can not only achieve significant performance improvements but also have the flexibility of utilising different types of user attributes for link prediction. We then perform in-depth analyses to verify the advantage of our model compared with baselines formally. In future, we plan to extend NeuLP for link prediction in temporal (or time-varying) graphs [LCWL18].

Chapter 9

Concluding Remarks

9.1 Contribution

The main contribution of this thesis is the integration of advanced *graph machine learning* (GML) with *online social network* (OSN) analysis in the context of artificial intelligence. On the one hand, our focus lies in overcoming *flat* message-passing *graph neural networks* (GNNs) (Chapters 3 and 4) and extending GML to *heterophilous* graphs (Chapters 5 and 6) in the light of inspiring observations in OSNs. We further presented two practical OSN analysis projects to illustrate the possibility of employing GML in practice (Chapters 7 and 8).

Having introduced the background and necessary prior knowledge in Chapter 2, and presented the main work of this thesis in Parts I–III. Here we conclude our contributions by answering our research questions posed in Chapter 1.

Research Question 1: *Can we develop graph machine learning models that overcome the limitations of flat message-passing mechanism to learn more comprehensive representations?*

The *hierarchical message-passing graph neural networks* (HMGNNs) (Chapter 3) provide an ingenious first step towards resolving this question. HMGNNs re-formulate the message-passing pipeline of GNNs with hierarchical relations of the graph and design flexible propagation manners to realise the information aggregation over long-range and enhance node representations with meso- and macro-level semantics. Following the idea of HMGNNs, we proposed practical and effective implementations, i.e., *hierarchical community-aware graph neural network* (HC-GNN) (Chapter 3), by constructing the hierarchical structure with the assistance of hierarchical community detection algorithms. Our theoretical analysis showed the convenience of HC-GNN in capturing informative interactions between any node pairs of a graph and the acceptable overall model complexity. The following comprehensive empirical evaluation parts further demonstrated the effectiveness of HC-GNN on node-level and edge-level analysis tasks on transductive settings.

Research Question 2: *Can we realise the hierarchical message-passing idea without any manual preprocessing?*

The *adaptive multi-grained graph neural networks* (AdamGNN) (Chapter 4), proposed a differentiable pooling operator to adaptively generate a multi-grained structure that involves meso- and macro-level semantic information in the graph, that can be used for hierarchical message-passing. It provides an interactive schema to learn node and graph representations in a mutual-optimisation manner. That said, node representations are enhanced with meso/macro-level semantic information, and the ameliorated node representations further form better graph representation. Experimental results revealed the effectiveness of AdamGNN on node-level, edge-level and graph-level tasks, and the ablation and empirical studies confirmed the effectiveness and flexibility of different components in AdamGNN.

Research Question 3: *Can graph machine learning models be efficiently applied on heterophilous graphs?*

The *compatible label propagation* (CLP) was proposed as a simple and strong adaption of label propagation algorithm for supervised graph learning tasks with challenging heterophily settings. It smooths the prior predictions across neighbour nodes weighted by the compatibility matrix. On a wide variety of benchmarks, we showed that our approach achieves the top performance on graphs with various levels of homophily. Meanwhile, it has orders of magnitude fewer parameters and requires less execution time. Empirical evaluations demonstrated that simple adaptations of LP can be competitive in semi-supervised node classification in both homophily and heterophily regimes.

Research Question 4: *Can graph machine learning models be utilised for unsupervised node representation learning on heterophilous graphs?*

This is a challenging yet essential problem, and we made a first step towards resolving this problem in this thesis. In Chapter 6, we first performed a large-scale empirical study on the performance of existing unsupervised *graph representation learning* (GRL) models on graphs with different homophily ratios and revealed the deficiency that they fall short on graphs with low-homophily. Then, we introduced *self-supervised network embedding* (Selene) framework to address the challenging task by solving three research challenges. Compared with earlier unsupervised GRL models, Selene holds neither homophily nor heterophily assumptions but only learns to discriminate r-ego networks of each node. We can conclude that unsupervised GRL tasks are possible by defining similarity/dissimilarity measurement among nodes and designing a model to learn representations to distinguish them.

Research Question 5: *Can online social networks be used for image classification?*

People perform a multitude of various activities in OSNs, and they often post content associated with a special OSN language, i.e., hashtags. In Chapter 7, we verified and quantified the relationship between image contents and its associated hashtags. After, we proposed a new graph-based approach *semantic image graph embedding* (SIGraph) [ZZP19], to capture the correlation between image contents and hashtags.

We then applied the explored relationship to improve online image classification tasks.

Research Question 6: *Can graph neural networks be utilised for link prediction in online social networks?*

We have introduced the *neural link prediction* (NeuLP) model (Chapter 8 [ZZP22]) to address this question. NeuLP adopts GNN encoders to capture node's local information: iteratively aggregate neighbour information for each user in ego-network with a few layers. After, apply a neural collaborative filtering module to simulate matrix factorisation to learn global interactions among users. Following this, NeuLP well fuse local and global information to make better link predictions.

9.2 Limitations and Future Directions

Despite many *graph machine learning* (GML) techniques being developed to facilitate complex system study, i.e., social network analysis, there are still several promising directions that need further exploration.

9.2.1 Scalability

The GML approaches, particularly graph neural networks, have achieved substantial performances due to prior capacity. However, they still suffer from the problem of memory and computation efficiency due to the dense network architecture and the large-scale graphs in real-world [XSY+21]. Such a drawback significantly restrict graph machine learning models' applicability. In this thesis, we have attempted to improve this issue by introducing additional message-passing pipelines (Chapter 3), simplifying model architecture under specific cases (Chapter 5) and maintaining the local algorithm characteristics (Chapters 4 and 6). Nevertheless, scalability is still a pending issue that we have not provided a unified method to solve this problem. Thus, we argue that designing scalable GML approaches is a driving direction.

9.2.2 Interpretability

GML approaches are not only designed to be effective but also to be built for reliability and safety. Interpretability, meaning that the model can provide reasonable and reliable interpretations in terms of how and why it makes certain predictions or actions, is of great importance to achieving real credible and safe ground applications [ZCH+20]. Some work has attempted to give technical in terms of the model execution process, we further wonder if it is possible to provide vivid interpretations under quotidian scenes [YBY+19, YTHJ20].

9.2.3 GML Driven Medicine Discovery and Development

The traditional process from drug discovery and development to market costs, generally well over billions of dollars and can span more than decade years or more [GDJ+20]. Such a long and expensive rigorous scientific process can be summarised as 4 main stages: (1) target identification and hit discovery, (2) molecule identification and candidate medicine design, (3) clinical trial and (4) FDA (Food and Drug Administration) approval. One key characteristic of biomedical data that is produced and used in the drug discovery process is its interconnected nature, which can be represented as a graph. GML techniques can be naturally applied on these biomedical graph data, particularly the stages (1) and (2), to accelerate the medicine discovery and development [HFG+21]. In specific, one promising proceed is exporting human knowledge as *Knowledge Graphs* [WMWG17] to assist biologists by generating a set of interpretable candidate medicine designs.

9.3 Conclusion

This thesis was motivated by the idea of developing *graph machine learning* techniques and understanding complex human social activity systems around us, i.e., *social networks*. This theme has taken us on a journey through the jungle of graph neural networks and the fun of understanding human online behaviours. In this thesis, we studied graph machine learning and proposed several contributions for processing, understanding and analysing online social network data. However, our proposed models and approaches are hardly perfect, and we tried to remain critical in stating shortcomings in each chapter and listed several promising future directions to explore.

Bibliography

[AL20] Waïss Azizian and Marc Lelarge. Characterizing the expressive power of invariant and equivariant graph neural networks. *CoRR*, abs/2006.15646, 2020.

[APK+19] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 2019 International Conference on Machine Learning (ICML)*, pages 21–29. JMLR, 2019.

[ARW+15] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the 2015 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936. IEEE, 2015.

[AY21] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021.

[BBCV21] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021.

[BCM11] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social Network Data Analytics*, pages 115–148. Springer, 2011.

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[BGLL08] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[BGW11] Prantik Bhattacharyya, Ankush Garg, and Shyhtsun Felix Wu. Analysis of user keyword similarity in online social networks. *Social Networl Analysis Mining*, 1:143–158, 2011.

[BHB+18] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea

Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[BHPZ17] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. walk2friends: Inferring social links from mobility profiles. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1943–1957. ACM, 2017.

[BKC21] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V. Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *CoRR*, abs/2106.02466, 2021.

[BKT13] Ronald S Burt, Martin Kilduff, and Stefano Tasselli. Social network analysis: Foundations and frontiers on advantage. *Annual review of psychology*, 64:527–547, 2013.

[BM21] Debanjan Banerjee and K. S. Meena. COVID-19 as an "Infodemic" in public health: Critical role of the social media. *Frontiers in Public Health*, 9:231–238, 2021.

[BMD+20] Ayan Kumar Bhowmick, Koushik Meneni, Maximilien Danisch, Jean-Loup Guillaume, and Bivas Mitra. Louvainne: Hierarchical louvain method for high quality and scalable network embedding. In *Proceedings of the 2020 ACM International Conference on Web Search and Data Mining (WSDM)*, pages 43–51. ACM, 2020.

[BOS+05] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alexander J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[BWS+20] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *Proceedings of the 2020 International Conference on World Wide Web (WWW)*, pages 1400–1410. ACM, 2020.

[BWSS21] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the 2021 AAAI Conference on Artificial Intelligence (AAAI)*, pages 3950–3957. AAAI, 2021.

[CCB+20] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[CCZP21] Ninghan Chen, Xihui Chen, Zhiqiang Zhong, and Jun Pang. From #jobsearch to #mask: improving COVID-19 cascade prediction with spillover effects. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 455–462. ACM, 2021.

[CCZP22] Ninghan Chen, Xihui Chen, Zhiqiang Zhong, and Jun Pang. Exploring spillover effects for covid-19 cascade prediction. *Entropy*, 24(2), 2022.

[CGK+11] Alan Chia-Lung Chen, Shang Gao, Panagiotis Karampelas, Reda Alhajj, and Jon G. Rokne. Finding hidden links in terrorist networks by checking indirect links of different sub-networks. In *Counterterrorism and Open Source Intelligence*, pages 143–158. Springer, 2011.

[CLB19] Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. In *Proceedings of the 2019 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[CLL+20] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the 2020 AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2020.

[CPHS18] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. HARP: Hierarchical representation learning for networks. In *Proceedings of the 2018 AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2018.

[CPLM21] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021.

[CTH+09] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the 2009 International Conference on Image and Video Retrieval (CIVR)*. ACM, 2009.

[CVJ+18] Catalina Cangea, Petar Velickovic, Nikola Jovanovic, Thomas Kipf, and Pietro Liò. Towards sparse hierarchical graph classifiers. *CoRR*, abs/1811.01287, 2018.

[CWH+20] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 2020 International Conference on Machine Learning (ICML)*. JMLR, 2020.

[CWPZ19] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2019.

[CZC18] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[CZP21] Ninghan Chen, Zhiqiang Zhong, and Jun Pang. An exploratory study of COVID-19 information on twitter in the greater region. *Big Data Cogn. Comput.*, 5(1):5, 2021.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 2016 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3837–3845. NIPS, 2016.

[DCG+89] Peter J. Denning, Douglas Comer, David Gries, Michael C. Mulder, Allen B. Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Computer*, 22(2):63–70, 1989.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 2009 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

[DHS+19] Simon S. Du, Kangcheng Hou, Ruslan Salakhutdinov, Barnabás Póczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Proceedings of the 2019 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2019.

[DHW+20] Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun, and Jie Tang. Heterogeneous network representation learning. In *Proceedings of the 2020 International Joint Conferences on Artifical Intelligence (IJCAI)*, pages 4861–4867. IJCAI, 2020.

[DTSS10] Cora I Dăniasă, Vasile Tomiţă, Dragoş Stuparu, and Marieta Stanciu. The mechanisms of the influence of viral marketing in social media. *Economics, Management & Financial Markets*, 5(3):278–282, 2010.

[DWP+15] Emily Denton, Jason Weston, Manohar Paluri, Lubomir D. Bourdev, and Rob Fergus. User conditional hashtag prediction for images. In *Proceedings of the 2015 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1731–1740. ACM, 2015.

[DYKC18] Amandeep Dhir, Yossiri Yossatorn, Puneet Kaur, and Sufen Chen. Online social media fatigue and psychological wellbeing—a study of compulsive use, fear of missing out, fatigue, anxiety and depression. *International Journal of Information Management*, 40:141–152, 2018.

[EGF+17] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. Zoobp: Belief propagation for heterogeneous networks. *Proc. VLDB Endow.*, 10(5):625–636, 2017.

[FZD+20] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[FZMK20] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the 2020 International Conference on World Wide Web (WWW)*, pages 2331–2341. ACM, 2020.

[Gat14] Wolfgang Gatterbauer. Semi-supervised learning with heterophily. *CoRR*, abs/1412.3100, 2014.

[Gat17] Wolfgang Gatterbauer. The linearization of belief propagation on pairwise markov random fields. In *Proceedings of the 2017 AAAI Conference on Artificial Intelligence (AAAI)*, pages 3747–3753. AAAI, 2017.

[GDJ+20] Thomas Gaudelet, Ben Day, Arian R. Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy B. R. Hayter, Richard Vickers, Charles Roberts, Jian Tang, David Roblin, Tom L. Blundell, Michael M. Bronstein, and Jake P. Taylor-King. Utilising graph machine learning within drug discovery and development. *CoRR*, abs/2012.05716, 2020.

[GGKF15] Wolfgang Gatterbauer, Stephan Günnemann, Danai Koutra, and Christos Faloutsos. Linearized and single-pass belief propagation. *Proc. VLDB Endow.*, 8(5):581–592, 2015.

[GJ19] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *Proceedings of the 2019 International Conference on Machine Learning (ICML)*. JMLR, 2019.

[GKRT04] Ramanathan V. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 2004 International Conference on World Wide Web (WWW)*, pages 403–412. ACM, 2004.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 2016 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 855–864. ACM, 2016.

[GMS05] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *2005. Proceedings of the 2005 IEEE International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 729–734, 2005.

[GN02] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99:7821–7826, 2002.

[GSN+13] Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using topic models for twitter

hashtag recommendation. In *Proceedings of the 2013 International Conference on World Wide Web (WWW)*, pages 593–596. ACM, 2013.

[GSR+17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 2017 International Conference on Machine Learning (ICML)*. JMLR, 2017.

[HFG+21] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. In *Proceedings of the 2021 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2021.

[HFZ+20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[HHS+21] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. Combining label propagation and simple models outperforms graph neural networks. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021.

[HLL+19] Jingjia Huang, Zhangheng Li, Nannan Li, Shan Liu, and Ge Li. Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In *Proceedings of the 2019 IEEE International Conference on Computer Vision (ICCV)*, pages 6480–6489. IEEE, 2019.

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 2017 International Conference on World Wide Web (WWW)*, pages 173–182. ACM, 2017.

[HM95] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *Proceedings of the 1995 International Workshop on Artificial Neural Networks (IWANN)*, pages 195–201. Springer, 1995.

[Hoc98] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 6(2):107–116, 1998.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[HSLH19] Xiao Huang, Qingquan Song, Yuening Li, and Xia Hu. Graph recurrent networks with attributed random walks. In *Proceedings of the 2019 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 732–740. ACM, 2019.

[HVH81] S. R. Searle H. V. Henderson. The vec-permutation matrix, the vec operator and kronecker products: A review. *Linear and multilinear algebra*, 9(4):271–288, 1981.

[HW79] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[HYL17a] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40:52–74, 2017.

[HYL17b] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 2017 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1025–1035. NIPS, 2017.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034. IEEE, 2015.

[IP21] Sergei Ivanov and Liudmila Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021.

[JE10] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 2010 Conference on Recommender Systems (RecSys)*, pages 135–142. ACM, 2010.

[JM15] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[JT21] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4037–4058, 2021.

[JXZZ19] Yizhu Jiao, Yun Xiong, Jiawei Zhang, and Yangyong Zhu. Collective link prediction oriented network embedding with hierarchical graph attention. In *Proceedings of the 2019 ACM International Conference on Information and Knowledge Management (CIKM)*, pages 419–428. ACM, 2019.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2015.

[KGW+18] Fariba Karimi, Mathieu Génois, Claudia Wagner, Philipp Singer, and Markus Strohmaier. Homophily influences ranking of minorities in social networks. *Scientific Reports*, 8:1–12, 2018.

[KKK+11] Danai Koutra, Tai-You Ke, U Kang, Duen Horng Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Machine Learning and Knowledge Discovery in Databases - European Conference (ECMLPKDD)*, volume 6912, pages 437–452. Springer, 2011.

[KO21] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 2012 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012.

[KSO+21] Tim Kaler, Nickolas Stathas, Anne Ouyang, Alexandros-Stavros Iliopoulos, Tao B. Schardl, Charles E. Leiserson, and Jie Chen. Accelerating training and inference of graph neural networks with fast sampling and pipelining. *CoRR*, 2021. abs/2110.08450.

[KW16] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *CoRR*, abs/1611.07308, 2016.

[KW17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 2017 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBD+89] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Proceedings of the 1989 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 396–404. Morgan Kaufmann, 1989.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LCWL18] Jundong Li, Kewei Cheng, Liang Wu, and Huan Liu. Streaming link prediction on dynamic attributed networks. In *Proceedings of the 2018*

ACM International Conference on Web Search and Data Mining (WSDM), pages 369–377. ACM, 2018.

[LHL+21] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Proceedings of the 2021 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2021.

[LHW18] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 2018 AAAI Conference on Artificial Intelligence (AAAI)*, pages 3538–3545. AAAI, 2018.

[LK07] David Liben-Nowell and Jon M. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[LKB+17] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[LLC10] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 2010 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 243–252. ACM, 2010.

[LLK19] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *Proceedings of the 2019 International Conference on Machine Learning (ICML)*. JMLR, 2019.

[LMBB17] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Signal Processing Magazine*, 67(1):97–109, 2017.

[LMK+19] Xin Liu, Tsuyoshi Murata, Kyoung-Sook Kim, Chatchawan Kotarasu, and Chenyi Zhuang. A general view for network embedding as matrix factorization. In *Proceedings of the 2019 ACM International Conference on Web Search and Data Mining (WSDM)*, pages 375–383. ACM, 2019.

[LMQ+19] Guohao Li, Matthias Müller, Guocheng Qian, Itzel C. Delgadillo, Abdulellah Abualshour, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the 2019 IEEE International Conference on Computer Vision (ICCV)*, pages 9267–9276. IEEE, 2019.

[LPJ+21] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *CoRR*, 2021. abs/2103.00111.

[LR12] Chengwei Lei and Jianhua Ruan. A novel link prediction algorithm for reconstructing protein–protein interaction networks by topological similarity. *Bioinformatics*, 29(3):355–364, 2012.

[LvdH13] Nelly Litvak and Remco van der Hofstad. Uncovering disassortativity in large scale-free networks. *Physical review E*, 87(2):022801, 2013.

[LWWL20] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding - design provably more powerful graph neural networks for structural representation learning. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[LZH+21] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *CoRR*, 2021. abs/2006.08218.

[MBSL19] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Proceedings of the 2019 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2153–2164. NeurIPS, 2019.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the 2013 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2013.

[ME11] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases - European Conference (ECMLPKDD)*, pages 437–452. Springer, 2011.

[Mil95] George A. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[ML10] Rohan Miller and Natalie Lammas. Social media and its implications for viral marketing. *Asia Pacific Public Relations Journal*, 11(1):1–9, 2010.

[ML12] Julian J. McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *Proceedings of the 2012 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 548–556. NIPS, 2012.

[MMCS11] Jonathan Masci, Ueli Meier, Dan C. Ciresan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proceedings of the 2011 International Conference on Artificial Neural Networks (ICANN)*, pages 52–59. Springer, 2011.

[MO19] Ece C. Mutlu and Toktam A. Oghaz. Review on graph feature learning and feature extraction techniques for link prediction. *CoRR*, 2019. abs/1901.03425.

[MRA+16] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1607.00148, 2016.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionally. In *Proceedings of the 2013 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119. NIPS, 2013.

[MSJG15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1511.05644, 2015.

[MWAT19] Yao Ma, Suhang Wang, Charu C. Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 2019 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 723–731. ACM, 2019.

[MWW20] Yimeng Min, Frederik Wenkel, and Guy Wolf. Scattering GCN: overcoming oversmoothness in graph convolutional networks. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 2010 International Conference on Machine Learning (ICML)*, pages 807–814. JMLR, 2010.

[NHGT14] Zhenxing Niu, Gang Hua, Xinbo Gao, and Qi Tian. Semi-supervised relational topic model for weakly annotated image recognition in social media. In *Proceedings of the 2014 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4233–4240. IEEE, 2014.

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 2001 Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2001.

[NLGH12] Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *Proceedings of the 2012 International Workshop on Mining and Learning with Graphs*, page 8, 2012.

[NN12] Raj Rao Nadakuditi and M. E. J. Newman. Graph spectra and the detectability of community structure in networks. *Physical Review Letters*, 108(18):188701, 2012.

[NSJ12] Seema Nagar, Aaditeshwar Seth, and Anupam Joshi. Characterization of social media response to natural disasters. In *Proceedings of the 2012 International Conference on World Wide Web (WWW)*, pages 671–674. ACM, 2012.

[ÖE12] Fatih Özgül and Zeki Erdem. Detecting criminal networks using social similarity. In *Proceedings of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 581–585. IEEE, 2012.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 2014 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710. ACM, 2014.

[Pee17] Leto Peel. Graph-based semi-supervised learning for relational networks. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 435–443. SIAM, 2017.

[PHL+20] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of the 2020 International Conference on World Wide Web (WWW)*, pages 259–270. ACM, 2020.

[PLG20] Krishna Kumar P., Paul Langton, and Wolfgang Gatterbauer. Factorized graph representations for semi-supervised learning from sparse data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1383–1398. ACM, 2020.

[PSY+19] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria E. Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys*, 51(5):92:1–92:36, 2019.

[PWC+20] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *Proceedings of the 2020 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[QCD+20] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: graph contrastive coding for graph neural network pre-training. In *Proceedings of the 2020 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1150–1160. ACM, 2020.

[QDM+18] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the 2018 ACM International Conference on Web Search and Data Mining (WSDM)*, pages 459–467. ACM, 2018.

[QDM+19] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. Netsmf: Large-scale network embedding as sparse matrix factorization. In *Proceedings of the 2019 International Conference on World Wide Web (WWW)*, pages 1509–1520. ACM, 2019.

[RAS21] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2), 2021.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 2015 Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

[RFC+20] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael M. Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *CoRR*, abs/2004.11198, 2020.

[RGP+15] Giulio Rossetti, Riccardo Guidotti, Diego Pennacchioli, Dino Pedreschi, and Fosca Giannotti. Interaction prediction in dynamic networks exploiting community discovery. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 553–558. ACM, 2015.

[RN20] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[Ros61] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[ŘS10] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA, 2010.

[RS20] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 2020 ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1325–1334. ACM, 2020.

[RSF17] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. *struc2vec*: Learning node representations from structural identity. In *Proceedings of the 2017 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 385–394. ACM, 2017.

[RST20] Ekagra Ranjan, Soumya Sanyal, and Partha Pratim Talukdar. ASAP: adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the 2020 AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2020.

[RT18] Daniel R. Richards and Bige Tunçer. Using image recognition to automate assessment of cultural ecosystem services from social media photographs. *Ecosystem Services*, 31:318–325, 2018.

[Saa81] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 1981.

[SC14] Jieying She and Lei Chen. Tomoha: Topic model-based hashtag recommendation on twitter. In *Proceedings of the 2014 International Conference on World Wide Web (WWW)*, pages 371–372. ACM, 2014.

[SGT+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[SH12] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter*, 2012.

[SK67] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the 2015 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823. IEEE, 2015.

[SNB+08] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2015.

[Ten16] Shang-Hua Teng. Scalable algorithms for data and network analysis. *Foundations and Trends in Theoretical Computer Science*, 12(1-2):1–274, 2016.

[TQM15] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 2015 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1165–1174. ACM, 2015.

[TQW+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 2015 International Conference on World Wide Web (WWW)*, pages 1067–1077. ACM, 2015.

[TR12] Oren Tsur and Ari Rappoport. What's in a hashtag? content based prediction of the spread of ideas in microblogging communities. In *Proceedings of the 2012 ACM International Conference on Web Search and Data Mining (WSDM)*, pages 643–652. ACM, 2012.

[Tra15] V.A. Traag. Faster unfolding of communities: Speeding up the louvain algorithm. *Physical Review E*, 92(3):032801, 2015.

[TY12] Xuning Tang and Christopher C Yang. Ranking user influence in healthcare social media. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):1–21, 2012.

[TZ15] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

[TZPM19] Yu Tian, Long Zhao, Xi Peng, and Dimitris N. Metaxas. Rethinking kernel methods for node representation learning on graphs. In *Proceedings of the 2019 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 11681–11692. NeurIPS, 2019.

[TZY+08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 2008 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 990–998. ACM, 2008.

[VCC+18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proceedings of the 2018 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2018.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. In *Eurographics Conference on Visualization (EuroVis)*, pages 2579–2605. Eurographics Association, 2008.

[VFH+19] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *Proceedings of the 2019 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[VSKB10] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 2017 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008. NIPS, 2017.

[WCW+17] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proceedings of the 2017 AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2017.

[Wei03] Eric W Weisstein. Gershgorin circle theorem. *https://mathworld. wolfram. com/*, 2003.

[WMWG17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[WPC+21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[WPL14] Jun Wang, Jiaxu Peng, and Ou Liu. An approach for hesitant node classification in overlapping community detection. In *Processings of the 2014 Pacific Asia Conference on Information Systems (PACIS)*, page 47, 2014.

[WRLL20] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[WS98] D. J. Watts and Steven H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440, 1998.

[WWL+11] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. In *Proceedings of the 2011 ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1031–1040. ACM, 2011.

[WYG18] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the 2018 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6857–6866. IEEE, 2018.

[WYHY15] Jiajun Wu, Yinan Yu, Chang Huang, and Kai Yu. Deep multiple instance learning for image classification and auto-annotation. In *Proceedings of the 2015 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3460–3469. IEEE, 2015.

[WYM+16] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the 2016 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2285–2294. IEEE, 2016.

[XGF16] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 2016 International Conference on Machine Learning (ICML)*, pages 478–487. JMLR, 2016.

[XHLJ19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the 2019 International Conference on Machine Learning (ICML)*. JMLR, 2019.

[XSC+19] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. In *Proceedings of the 2019 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

[XSY+21] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021.

[YBY+19] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Proceedings of the 2019 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 9240–9251. NeurIPS, 2019.

[YCS+20] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[YHS+21] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *abs/2102.06462*, 2021.

[YJ20] Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 2020 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[YLZ+15] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. Network representation learning with rich text information. In *Proceedings of the 2015 International Joint Conferences on Artifical Intelligence (IJCAI)*, pages 2111–2117. IJCAI, 2015.

[YTHJ20] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: towards model-level explanations of graph neural networks. In *Proceedings of the 2020 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 430–438. ACM, 2020.

[YYL19] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *Proceedings of the 2019 International Conference on Machine Learning (ICML)*. JMLR, 2019.

[YYM+18] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 2018 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 4805–4815. NeurIPS, 2018.

[ZA20] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *Proceedings of the 2020 International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[Zac77] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 1977.

[ZAG18] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 2018 ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2847–2856. ACM, 2018.

[ZAL18] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[ZBL+03] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Proceedings of the 2003 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 321–328. NIPS, 2003.

[ZC18] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 2018 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5171–5181. NeurIPS, 2018.

[ZCH+20] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[ZCNC18] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the 2018 AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2018.

[ZCZ20] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[ZG09] Elena Zheleva and Lise Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 2009 International Conference on World Wide Web (WWW)*, pages 531–540. ACM, 2009.

[Zha19] Yang Zhang. Language in our time: An empirical analysis of hashtags. In *Proceedings of the 2019 International Conference on World Wide Web (WWW)*, pages 2378–2389. ACM, 2019.

[ZHR+18] Yang Zhang, Mathias Humbert, Tahleen Rahman, Cheng-Te Li, Jun Pang, and Michael Backes. Tagvisor: A privacy advisor for sharing hashtags. In *Proceedings of the 2018 International Conference on World Wide Web (WWW)*, pages 287–296. ACM, 2018.

[Zhu05] Xiaojin Zhu. Semi-supervised learning literature survey. *Technical report, University of Wisconsin-Madison Department of Computer Sciences*, 2005.

[ZJM+21] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 2021 International Conference on Machine Learning (ICML)*, pages 12310–12320. JMLR, 2021.

[ZL17] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33:i190–i198, 2017.

[ZLK+18] Bolei Zhou, Àgata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2018.

[ZLLW19] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. In *Proceedings of the 2019 International Joint Conferences on Artifical Intelligence (IJCAI)*, pages 4327–4333. IJCAI, 2019.

[ZLP20a] Zhiqiang Zhong, Cheng-Te Li, and Jun Pang. Adaptive multi-grained graph neural networks. *CoRR*, abs/2010.00238, 2020.

[ZLP20b] Zhiqiang Zhong, Cheng-Te Li, and Jun Pang. Hierarchical message-passing graph neural networks. *CoRR*, abs/2009.03717, 2020.

[ZLP22] Zhiqiang Zhong, Cheng-Te Li, and Jun Pang. Personalised meta-path generation for heterogeneous graph neural networks. *Data Min. Knowl. Discov.*, 2022.

[ZRR+21] Jiong Zhu, Ryan A. Rossi, Anup B. Rao, Tung Mai, Nedim Lipka, Nesreen K. Ahmed, and Danai Koutra. Graph neural networks with

heterophily. In *Proceedings of the 2021 AAAI Conference on Artificial Intelligence (AAAI)*, pages 11168–11176. AAAI, 2021.

[ZYZ+20] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[ZZP19] Zhiqiang Zhong, Yang Zhang, and Jun Pang. A graph-based approach to explore relationship between hashtags and images. In *Proceedings of the 2019 International Conference on Web Information Systems Engineering (WISE)*, volume 11881, pages 473–488. Springer, 2019.

[ZZP22] Zhiqiang Zhong, Yang Zhang, and Jun Pang. NeuLP: An end-to-end deep-learning model for link prediction. In *Proceedings of the 2020 International Conference on Web Information Systems Engineering (WISE)*, pages 96–108. Springer, 2022.

Curriculum Vitae

2018 – 2022	Ph.D. student, University of Luxembourg, Luxembourg
2016 – 2017	Master in Finance, Paris Dauphine University, France.
2015 – 2017	Master Degree in Data Science, EISTI, France.
2011 – 2015	Bachelor in Mathematics, China University of Petroleum, China

Born on July 19, 1993, Anhui, China.