Robustness, Heterogeneity and Structure Capturing for Graph Representation Learning and its Application

SUN, ZHONGTIAN

Robustness, Heterogeneity and Structure Capturing for Graph Representation Learning and its Application

Zhongtian Sun

Supervisor: Prof. Alexandra I. Cristea

A Thesis presented for the degree of

Doctor of Philosophy

Department of Computer Science
Durham University
United Kingdom
Jan 2024

Abstract

Graph neural networks (GNNs) are potent methods for graph representation learning (GRL), which extract knowledge from complicated (graph) structured data in various real-world scenarios. However, GRL still faces many challenges. Firstly GNN-based node classification may deteriorate substantially by overlooking the possibility of noisy data in graph structures, as models wrongly process the relation among nodes in the input graphs as the ground truth. Secondly, nodes and edges have different types in the real-world and it is essential to capture this heterogeneity in graph representation learning. Next, relations among nodes are not restricted to pairwise relations and it is necessary to capture the complex relations accordingly. Finally, the absence of structural encodings, such as positional information, deteriorates the performance of GNNs. This thesis proposes novel methods to address the aforementioned problems:

1. Bayesian Graph Attention Network (BGAT): Developed for situations with scarce data, this method addresses the influence of spurious edges. Incorporating Bayesian principles into the graph attention mechanism enhances robustness, leading to competitive performance against benchmarks (Chapter 3).

2. Neighbour Contrastive Heterogeneous Graph Attention Network (NC-HGAT): By enhancing a cutting-edge self-supervised heterogeneous graph neural network model (HGAT) with neighbour contrastive learning, this method addresses heterogeneity and uncertainty simultaneously. Extra attention to edge relations in heterogeneous graphs also aids in subsequent classification tasks (Chapter 4).

3. A novel ensemble learning framework is introduced for predicting stock price

movements. It adeptly captures both group-level and pairwise relations, leading to notable advancements over the existing state-of-the-art. The integration of hypergraph and graph models, coupled with the utilisation of auxiliary data via GNNs before recurrent neural network (RNN), provides a deeper understanding of long-term dependencies between similar entities in multivariate time series analysis (Chapter 5).

4. A novel framework for graph structure learning is introduced, segmenting graphs into distinct patches. By harnessing the capabilities of transformers and integrating other position encoding techniques, this approach robustly capture intricate structural information within a graph. This results in a more comprehensive understanding of its underlying patterns (Chapter 6).

Declaration

The work in this thesis is based on research carried out at the Department of Computer Science, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

I am profoundly grateful for the opportunity to express my deepest gratitude and appreciation to all those who have supported, encouraged, and inspired me throughout my challenging yet rewarding journey towards obtaining my PhD in Computer Science. This achievement would not have been possible without the unwavering support of numerous individuals.

First and foremost, I would like to thank Prof. Alexandra I. Cristea for giving me the opportunity to pursue a PhD under her supervision. I am tremendously grateful for her continuous support, countless inspiring discussions, for giving me the freedom to pursue a variety of research directions, and for frequently reminding me to "think different" in the past four years. Many thanks also go to Dr Jingyun Wang for co-advising me in the later stage of my PhD.

I want to express my sincere appreciation to my family, especially my amazing parents, Dr Xiaowei Sun and Dr Zhili Zhang, for their unwavering love, selfless support, and strong belief in my abilities. Despite my previous background in finance, they encouraged me to pursue my passion for computer science, understanding the challenges of this transition. Their dedication and love have been integral to my success.

To my dear friends Miss Anoushka Harit, Dr Jialin Yu, Dr Neelanjan Bhowmik, Dr Ahmed Alamri, Dr Jack Barker, Mr Brian Kostadinov Shalon Isaac-Medina, Dr Yona Falinie Abd Gaus, Miss Yuxuan Song and Dr Olanrewaju Tahir Aduragba, thank you all for being an incredible support system during the past four years. The COVID-19 pandemic brought about unforeseen challenges, and I am grateful for the camaraderie and resilience we have demonstrated together in overcoming them. Your assistance and encouragement, both academically and personally, have made this journey more bearable.

I would like to express my gratitude to my collaborators and advisors, Miss

Anoushka Harit, Dr Jialin Yu, Dr Ahmed Alamri, Dr Noura AI Moubayed, Prof. Pietro Lio, Dr Jingyun Wang, Prof. Tingwei Chen, Dr Lei Shi, Dr Sunčica Hadžidedić and Prof. Alexandra I. Cristea, who have contributed to my intellectual and personal growth throughout my time here. Special thanks to my girlfriend, Miss Anoushka Harit, whose unwavering support and encouragement throughout this journey have been extraordinary.

I would like to thank my thesis committee members, Dr Sunčica Hadžidedić and Prof. Steven Schockaert for their insightful feedback and constructive criticism, which significantly enhanced the quality of my research.

I am also grateful for being awarded the scholarship by the Great Britain-China Educational Trust (GBCET, charity number: 269944), which helped me through financial distress time.

Lastly, I want to acknowledge the countless others who have directly or indirectly contributed to my success. Your contributions, no matter how small, have left a lasting impact on my journey, and I am truly grateful.

This thesis is dedicated to all of you as a testament to the power of perseverance, support, and community during challenging times.

Contents

List of Figures

List of Tables

Nomenclature

σ	Activation function
A	Adjacency matrix
a	Attention score
D	Degree matrix
E	A set of edges
F	Functions
G	Graph
H	Embedding matrix
K	K-hop
l	Layer of neural network

N	Number
V	A set of nodes
X	An input matrix
x	An input vector
y	An output vector

CHAPTER 1

Introduction

1.1 Representation Learning on Graphs

Representation learning aims to process and present data in a form that's more interpretable for machines, enabling them to better tackle downstream tasks such as question answering, recommendation systems and knowledge graphs. Previously, researchers proposed statistical approaches [4] combined with engineering features of input data for such representation learning tasks. However, engineering features require domain expertise and hard to generalise. Additionally, this type of manual rules setting is not cost-efficient. Therefore, automatic feature learning from raw input is an important topic. Then, representation learning has been further boosted by advances in deep learning in the multi-layered hierarchy of feature extraction

across various domains, including acoustics, natural language processing, and images [5]. However, the generalisation ability of deep learning in representation learning relational and complex reasoning still needs to be improved.

The world can be understood and modelled in terms of concepts and relations. While images can be characterized using localised convolutional filters, graphs present a more complex scenario. Unlike the structured nature of images, graphs can be irregular, arbitrary, non-Euclidean in structure and contain rich values [6]. This inherent complexity and richness make graphs particularly adept at representing knowledge (entities and relationships) in the real world [5], as depicted in Figure 1.1.

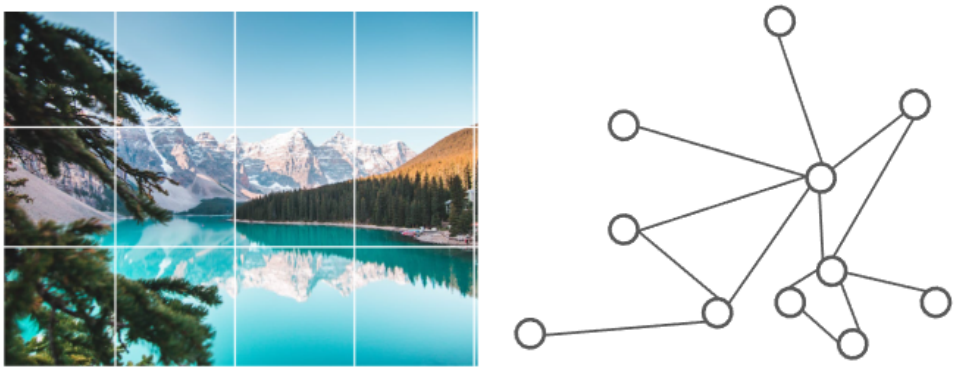


Figure 1.1: Image in Euclidean space and graph in non-Euclidean space

A graph is a type of structure defined as $G = (V, E)$ where V is a set of nodes representing entities and E is a set of edges indicating relationships among entities. Graph representation learning is usually based on graph topological structures and graph features. There are several methods for graph representation learning, including random walking, deep neural networks, and mtrix factorisation. According to [6], models like graph neural networks (GNNs) and their variants, including graph

convolutional networks (GCNs), message passing neural networks (MPNNs), graph attention networks (GATs), graph autoencoders (GAEs), graph recurrent neural networks (GraphRNN), graph adversarial models and graph transformer models are developed extensively for tasks including node classification, graph classification as well as link prediction and attract increasing research attention.

Existing graph representation learning models rely on aggregating neighbourhood information or convolutional operators to learn the representations of nodes, which describe the graph's structure [7]. The information pass, aggregation and update process is known as message passing, which will be explained in Chapter 2.2.2. Despite the great success of GNNs across different domain tasks, including natural language processing (NLP), computer vision (CV), social recommendation and many other applications, some **significant challenges remain** to be solved as follows [8–10]:

Robustness This reflects the extent to which generated graphs can withstand adversarial attacks, which is particularly applicable when graphs are incomplete or when some properties are missing. According to [9, 11], graph neural networks are highly sensitive to adversarial attacks. The problem could be integrated with semi-supervised classification when given information is limited, and researchers are required to infer the labels of unknown nodes based on existing knowledge.

Heterogeneous Contexts Most existing studies focus on homogeneous rather than heterogeneous graphs, where nodes and edges are composed of different types and must be tackled specifically [8]. A typical example of heterogeneous graphs is social network graphs [12]. In contrast to homogeneous graph neural network meth-

ods, the proximity between items in a heterogeneous information network (HIN) should not just be confined to distance but also semantics.

Structural Learning Structural learning is another important topic for graph representation learning, which has received increasing focus recently [13, 14]. This work focuses on capturing relations, geometry and position information comprehensively. High-order relations, prevalent in real-world applications such as social networks, drug discovery, and transport, have been demonstrated to be effective in graph representation learning [15]. The effectiveness of high-order relation learning in financial investment is thus investigated.

Recently, without canonical positional information of nodes, the expressive power of GNNs will decrease [16]. One tendency is to combine transformers with GNNs to avoid strict structural inductive biases by overcoming the limitation of local neighborhood aggregation and over-squashing in existing message passing mechanisms [17]. Over-squashing means information is overly compressed due to exponentially increased neighbours of nodes, and standard message passing GNNs do not benefit from more than a few layers [18]. However, there is no guarantee that existing transformers in GNNs can distinguish structural similarity between entities properly [2], and they have quadratic complexity, which is not scalable to massive graphs. Therefore, it is necessary to develop novel models to consider structural information comprehensively with less complexity and enhance the representation power of GNNs.

In this thesis, we are **addressing the robustness, heterogeneity and structural learning problems** stated above, and **propose corresponding new mod-**

els to tackle them - illustrated in Figure 1.2, which also shows the corresponding chapters where these problems are addressed.

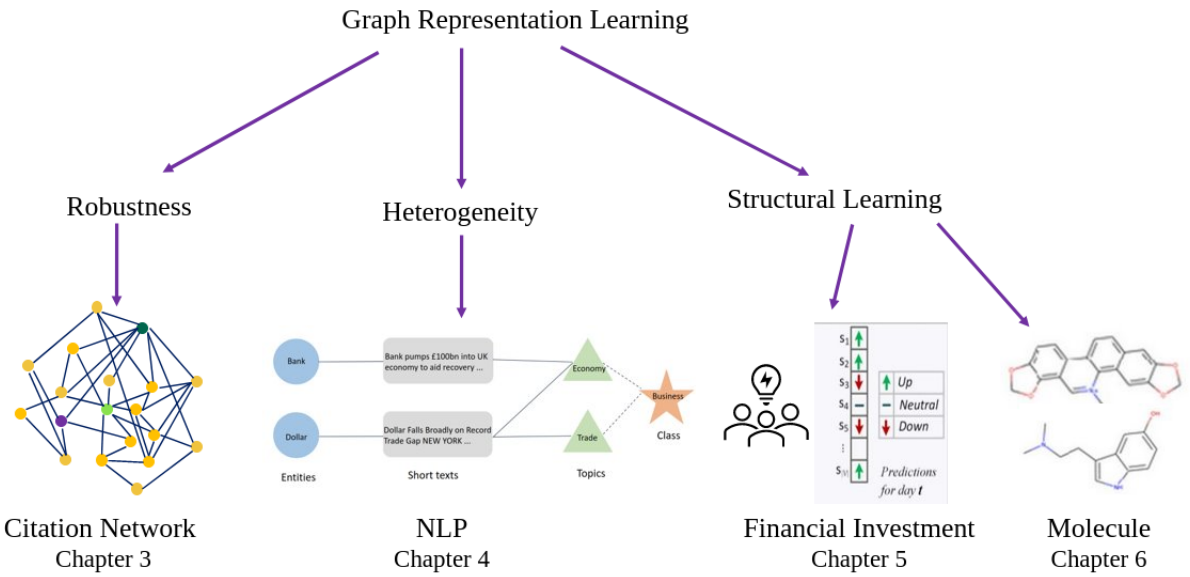


Figure 1.2: My research in graph representation focuses on 3 key problems

1.2 Research Questions

As stated, this study addresses the problems of robustness, heterogeneity and structural learning to enhance the models' learning ability in graph representation learning. Most existing GNNs are based on iterative message passing to update features of neighboring nodes. Performance deteriorates when limited training labels are available, making it challenging to learn discriminative node embeddings [19]. This issue is referred to as the robustness problem, leading to the first research question:

RQ1 ***How to increase robustness regarding uncertainty (few labels available during training or noisy input) in representation learning based on graph neural networks without negatively affecting the***

performance? (RQ1 is answered in Chapter 3)

We proposed a novel and effective method by extending a GNN method with a parametric random graph model for this problem on the citation network datasets in Chapter 3.

However, this chapter mainly focused on the homogeneous graph, in which all nodes and edges belong to the same type, while in practice, entities primarily interact with other different types of objects in complex systems [20], known as heterogeneous graphs. Compared with homogeneous graphs, heterogeneous graphs are crucial to reflect the actual semantics of complex graphs, and it is necessary to consider different types of nodes and edges simultaneously [21].

RQ2 ***How to account for the heterogeneity of graphs, which contain diverse types of edges, features and attributes, in graph neural networks?***(RQ2 is answered in Chapter 4).

We addressed the heterogeneity problem by incorporating contrastive learning and types of nodes and edges in text classification tasks in Chapter 4. We also look into heterogeneity from the point of view of its impact on robustness, effectively combining the first two research questions. This is also addressed in Chapter 4.

However, this study did not consider the structural information of graphs, which is crucial for guaranteeing the expressive power of GNNs. Therefore, the last research question is to examine how to conduct structural learning by capturing high-order relations and comprehensive positional information more effectively in graph representation learning:

RQ3 ***How to improve the structural learning ability, e.g. relation extraction and positional encoding for existing graph representation learning?*** (RQ3 is answered in Chapters 5 and 6).

Overall we aim to investigate how to improve the expressive power of models for graph representation learning by answering all three research questions. We further discuss how these three axes can be combined in section 7.2, starting with the combination of the first two axes as an illustration in section 4.7.

1.3 Research Contributions

Addressing the above questions, this thesis contributes to the field of graph representation learning and the main contributions of this thesis can be summarised as follows:

Investigating and proposing novel methods to enhance the representation learning ability of GNNs from three aspects: *robustness*, *heterogeneity* and *structural learning*. Further we illustrate these advances with specific applications in citation network, NLP, finance and molecule learning to showcase their diversity of applicability. Specifically, the following contributions have been achieved during my PhD:

1. Demonstrated the benefits of **incorporating uncertainty in graph structure via a parametric random graph model** (in Chapter 3) as a solution to the *robustness* problem. Unlike existing studies, this work introduced for the first time *attention to capturing the fact that different nodes have different influences in random graph generation*; the proposed novel Bayesian graph

deep learning model outperformed earlier state-of-the-art. This is the ***first* research on integrating Bayesian deep learning with the graph attention method to solve the data-scarcity problem**.

2. Proposed a **novel neighbouring contrastive learning method** (see Chapter 4) by identifying the limitations and challenges that need to be solved for the heterogeneity area. Instead of directly passing a message relying on the adjacency matrix, this neighbouring contrastive method leverages graph structure to learn the semantics of the different types of nodes and also improves the robustness of representation learning. This is the ***first study* on how contrastive learning can be used with heterogeneous graph neural networks for text classification tasks**.

3. Explored how to leverage graphs to model complex relations in financial investment stock prediction (Chapter 5). This study validated the merits of hypergraphs and simple graphs for capturing group-level and pairwise interactions of nodes and is the first one to prove the necessity of considering them together when complex relations exist. This is also the ***first study* to demonstrate the effectiveness of integrating auxiliary information via GNNs before using RNNs for temporal studies** - so that the long-term dependencies of similar entities can be learnt by RNNs later (in this task, the auxiliary information refers to industry information, and entities refer to companies).

4. Optimised the strategy of capturing comprehensive structural information of nodes and alleviated the over-squashing problem in existing message passing

mechanisms (Chapter 6). This research showed the potential of considering positional information and geometry information learning by converting graphs into patches. This is the ***first study*** **on how Ricci curvature can be considered in contrastive learning to alleviate the over-squashing problem in the graph patch learning task**.

1.4 Thesis Outline

In Chapter 1, an overview of graph representation learning, its progression, and the persisting, unresolved challenges are introduced. Then the motivation driving this thesis, the proposed research questions and the objectives are explained. Moreover, an outline of this thesis is presented, accompanied by a list of all publications to date in this chapter.

In Chapter 2, fundamental concepts of GNN (graph neural network), such as the Laplacian matrix, Fourier transform, and adjacency matrix, are thoroughly detailed and explained. Additionally, two prevalent GNN architectures - spectral and spatial methods, which serve as the backbones of the proposed models in this thesis are described. Then, a brief overview of contrastive learning is provided. Next, the vanilla transformer architecture, attention mechanism, and methods of position encoding in representation learning are introduced. Lastly, the datasets utilised in this thesis are summarised.

Rather than presenting the literature review as a standalone chapter, this thesis weaves it throughout each chapter. This structure allows for a closer connection between the existing literature and the specific research questions addressed in each

chapter.

We develop a generative graph model in Chapter 3, which is particularly suited for structured representation under data-scarce situations to solve the robustness problem in graph representation learning.

Then, we present neighbour contrastive learning with a heterogeneous graph attention model targeting heterogeneity problems in Chapter 4. In addition to the published work, we explore the influence of different types of edges in text classification tasks and also demonstrate that the proposed model performs against early state-of-the-art with few labels available.

Next, we propose a novel ensemble learning framework consisting of a GCN to capture pairwise information and a hypergraph convolution network for group-oriented information with adversarial training for stock price prediction, named MONEY, in Chapter 5. Further, we demonstrate that pairwise relations should not be neglected in hypergraph learning and the proposed model outperforms state-of-the-art and delivers more stable results in a bear market.

In Chapter 6, we present an effective general framework for graph representation learning. The method improves the expressive power of current graph representation learning models with comprehensive structural information and alleviates the over-squashing issues.

This thesis is summarized in Chapter 7 and the contributions made towards answering the research questions listed in 1.2 are detailed. Additionally, potential future work, which could extend from these contributions, is proposed.

1.5 Publications to Date

Note on publications included in this thesis: At the time of submission, four chapters of this thesis are heavily based on papers submitted for publication or published in conferences and journals:

- **Chapter 3: Sun, Z.**, Harit, A., Yu, J., Cristea, A. I., Al Moubayed, N. (2021). A Generative Bayesian Graph Attention Network for Semi-Supervised Classification on Scarce Data. In 2021 International Joint Conference on Neural Networks (IJCNN, Core A Ranked Conference). pp. 1-7. IEEE. [22]

- **Chapter 4: Sun, Z.**, Harit, A., Cristea, A. I., Yu, J., Shi, L., Al Moubayed, N. (2022). Contrastive Learning with Heterogeneous Graph Attention Networks on Short Text Classification. In 2022 International Joint Conference on Neural Networks (IJCNN, Core A Ranked Conference). pp. 1-6. IEEE. [23]

- **Chapter 5: Sun, Z.**, Harit, A., Cristea, A. I., Yu, J., Lei, Shi., Al Moubayed, N. MONEY: A Novel enseMble Learning: cOnvolutional Network with adVersarial hYpergraph Model for Stock Price Movement Prediction. AI Open Journal (Impact Score 30). 4, pp. 165-174. [24]

- **Chapter 6: Sun, Z.**, Harit, A., Cristea, A. I, Lio, P and Wang, J. A Ricci Curvature Contrastive PerformerMixer Framework for Graph Representation Learning. In 2023 IEEE International Conference on Big Data (Big Data).

Note on publications not included in this thesis: In addition to the above publications, the following works have been published during the period of research

for this thesis which enhanced my comprehension of work towards this thesis. However, these publications do not integrate into the overarching narrative of this thesis and thus are not included in the text.

- **Sun, Z.**, Cristea, A.I., Lio, P. and Yu, J., 2023. Adaptive Distance Message Passing From the Multi-Relational Edge View. ICLR Tiny.

- Xiao, C., Ye, Z., Hudson, G.T., **Sun, Z.**, Blunsom, P. and Al Moubayed, N., 2023. Can Text Encoders be Deceived by Length Attack?. ICLR Tiny.

- Yu, J., Cristea, A.I., Harit, A., **Sun, Z.**, Aduragba, O.T., Shi, L. and Al Moubayed, N., 2023, May. Language as a latent sequence: Deep latent variable models for semi-supervised paraphrase generation. AI Open.

- Yu, J., Cristea, A.I., Harit, A., **Sun, Z.**, Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022, July. Efficient Uncertainty Quantification for Multilabel Text Classification. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

- Yu, J., Cristea, A.I., Harit, A., **Sun, Z.**, Aduragba, O.T., Shi, L. and Al Moubayed, N., 2022, July. INTERACTION: A Generative XAI Framework for Natural Language Inference Explanations. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

- **Sun, Z.**, Harit, A., Cristea, A.I., Yu, J., Al Moubayed, N. and Shi, L., 2022, December. Is Unimodal Bias Always Bad for Visual Question Answering? A Medical Domain Study with Dynamic Attention. In 2022 IEEE International Conference on Big Data (Big Data) (pp. 5352-5360). IEEE.

- Yu, J., Alrajhi, L., Harit, A., **Sun, Z.**, Cristea, A.I. and Shi, L., 2021. Exploring bayesian deep learning for urgent instructor intervention need in mooc forums. In Intelligent Tutoring Systems: 17th International Conference, ITS 2021, Virtual Event, June 7–11, 2021, Proceedings 17 (pp. 78-90). Springer International Publishing.

- Alamri, A., **Sun, Z.**, Cristea, A. I., Stewart, C., Pereira, F. D. (2021). MOOC next week dropout prediction: weekly assessing time and learning patterns. In International Conference on Intelligent Tutoring Systems. pp. 119-130. Springer, Cham..

- Alamri, A., **Sun, Z.**, Cristea, A. I., Senthilnathan, G., Shi, L., Stewart, C. (2020). Is MOOC Learning Different for Dropouts? A Visually-Driven, Multi-granularity Explanatory ML Approach. In International Conference on Intelligent Tutoring Systems. pp. 353-363. Springer, Cham.

- Yu, J., Aduragba, O.T., **Sun, Z.**, Black, S., Stewart, C., Shi, L., Cristea, A. (2020). Temporal Sentiment Analysis of Learners: Public Versus Private Social Media Communication Channels in a Women-in-Tech Conversion Course. In International Conference on Computer Science and Education pp. 182-187.

CHAPTER 2

Background and Methodology

This chapter dedicated to the background will provide a comprehensive overview of the foundational method that forms the basis of the graph representation learning research conducted in this doctoral thesis. The goal of graph representation learning is to model the features and structures of graphs accurately and there are two types of methods: traditional graph embedding and graph neural networks (GNNs) [25]. This thesis focuses on GNNs, which are studied in Chapter 3 - 6.

In this chapter, the basic knowledge of graphs is first introduced in section 2.1. Subsequently, a brief explanation of backbone machine learning methods, including graph neural networks (GNNs), is presented in section 2.2. Next, contrastive learning is discussed in section 2.3, and the concept of transformers is explicated in section 2.4. Furthermore, an in-depth review of the literature pertinent and actual

model implementation details to each specific research question will be presented in the respective Chapters, thereby facilitating a detailed comprehension of the specific research question associated with each distinct investigation.

GNNs can be categorised into spectra-based and spatial-based, depending on the function of the graph filter. Contrastive learning is a technology to learn the unique features of inputs without labels, which is applied in Chapter 4 and 6, addressing the robustness problem. Transformer is a popular deep learning model based on self-attention, which can evaluate the importance of each input component and is investigated in Chapter 6.

2.1 Basics of a Graph

The primary focus of my thesis is Graph Representation Learning, a field that fundamentally relies on the concept of a graph. Throughout this thesis, the subsequent sections and Chapters 3 and 6 will consistently adhere to the following graph definition. Other types of graphs, including heterogeneous graphs and hypergraphs will be discussed in Chapter 4 and 5, respectively.

A graph can be denoted as $G = (V, E)$ where V refers to nodes and E represents edges [26]. Adjacency matrix A represents a graph with a number of N nodes, as shown in Figure 2.1:

$$A(i,j) = \begin{cases} 1 \text{ if node } i \text{ connects with node } j \\ \\ 0 \text{ otherwise} \end{cases}$$

Adjacency matrix encodes the relationships between nodes, serving as a founda-

tional input for propagating node features and learning node embeddings in graph representation learning [27]. Another important concept in graph representation learning is degree matrix D, which captures the number of links associated with each node, and is often utilised in the computation of graph Laplacians and normalisation procedures [28].

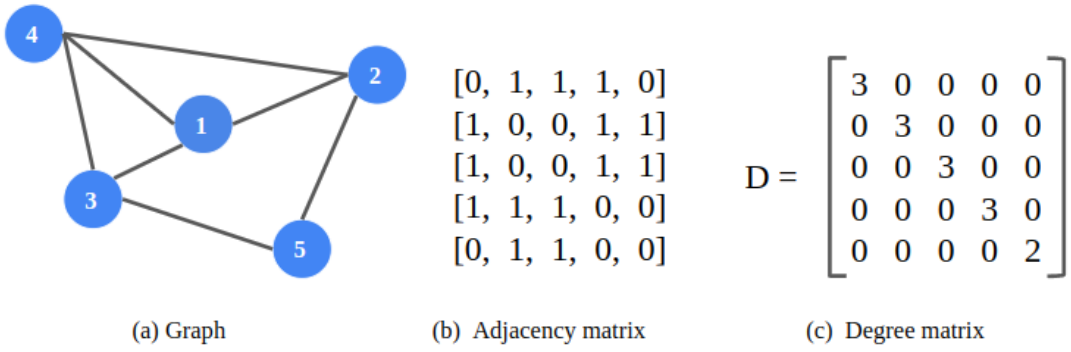
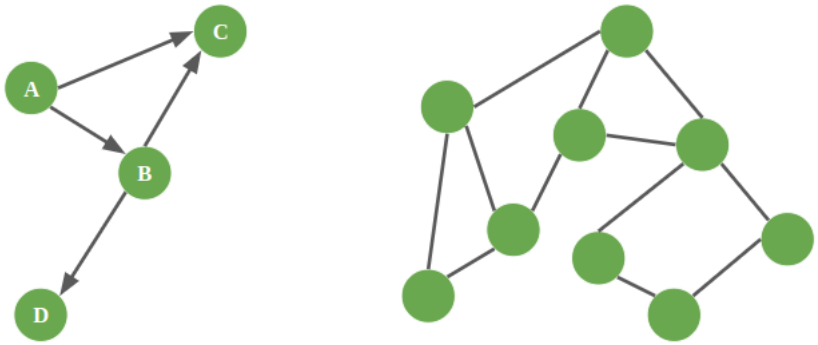


Figure 2.1: Graph, adjacency and degree matrix

A node can contain qualities like weight, size, position, and other features. An edge can also have different properties, such as weight denoting the strength of connection or direction representing relations.

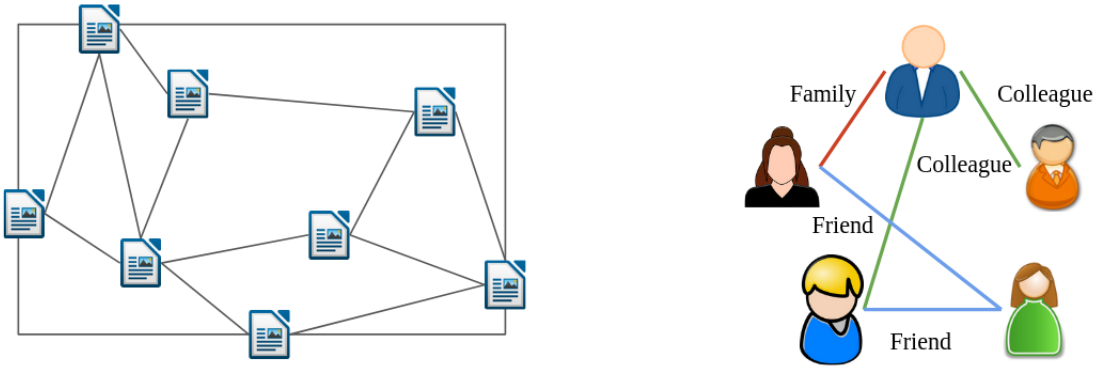
Graphs can have different types: directed, undirected, homogenous, heterogeneous, weighted, unweighted, simple and hypergraph [29]. Edges can have directions in a graph, referred to as a directed graph. Each edge in an undirected graph can also be considered two directed edges, making it a special instance of a directed graph [26]. As Figure 2.2 shows, a bidirectional graph can be considered an undirected graph.



(a) Directed Graph (b) Undirected Graph

Figure 2.2: Directed and undirected graph

In homogeneous graph, all nodes are of same type and all edges represent the same types relations [20], such as citation network Cora, Citeseer and PubMed, explained in Section 2.5. In heterogeneous graphs, nodes and edges have different properties [30], as illustrated in Figure 2.3, where the homogeneous graph consists of scientific publications and the heterogeneous graph consists of different persons with various relations.



(a) Homogeneous Graph (b) Heterogeneous Graph

Figure 2.3: Homogenous and heterogeneous graph

2.2 Graph Neural Networks

Deep learning models that act on graph-structured data are known as Graph Neural Networks (GNNs) [8]. GNNs have attracted much attention in recent years, as they can accurately depict the intricate connections between objects represented as nodes in a graph. GNNs are proposed to capture interactions in the form of learned node representations, which are utilised to represent entities and their relationships in graphs. Tasks like graph categorisation, node classification, and graph construction can all be accomplished using these representations. Graph classification is to classify a graph, considering its structure and node features. Node classification is to classify the labels of nodes based on their features where each node needs to be projected into an embedding space while preserving the intrinsic graph structure, as illustrated in Figure 2.4. Link prediction predicts the existence of edges between a pair of nodes, which is commonly seen in recommendation systems, drug discovery and knowledge graph applications.

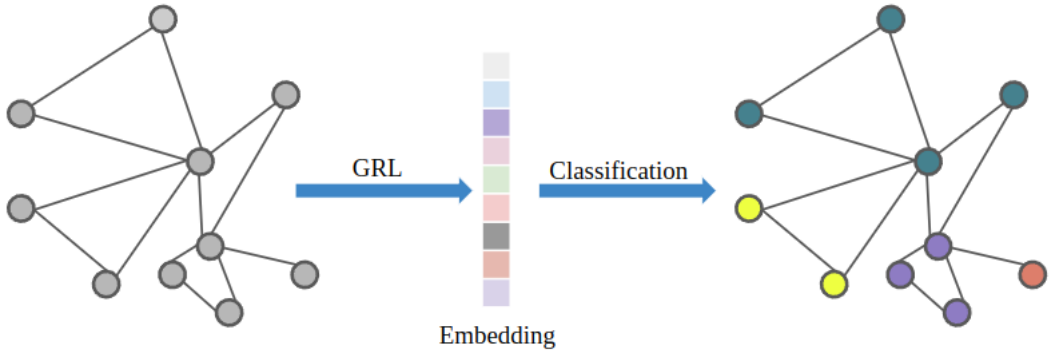


Figure 2.4: Node classification in graph representation learning

Researchers began investigating how to analyse graph-structured data using neural networks in the early 2000s when GNNs first emerged. [31] initially formulated

GNNs to update nodes' states iteratively, until reaching stable states. They represented each node in a graph as a vector of features and computed the output state by propagating discrete features from neighbouring nodes for pattern recognition. [32] elaborated it further by extending recurrent models to deal with graphs, including directed, cyclic, or mixed graphs based on information diffusion and relaxation methods [8]. However, the complicated relationships between nodes in a graph can only be partially captured by the early GNNs. Since then, GNNs have grown to various architectures and training methods. Wu et al. [33] categorised the recent models into four types based on the information propagation mechanism: spectral-based, spatial-based, attention-based and recurrent-based graph filters. This section mainly introduces the first two mechanisms. More related work and the attention-based graph filters are explained in Chapter 3 and 4. In this thesis, models grounded in recurrent-based graph filters are not developed, given that primary research focus is not dynamic graph scenarios, where such filters are predominantly applied [34].

2.2.1 Spectral-based Graph Filters

Spectral approaches are inspired by graph signal processing, which defines the convolution operator using the Fourier transform from the spectral domain. One seminal work is graph convolution network (GCN), proposed by [35], which utilised a first-order Chebyshev approximation on *spectral convolutions* for graphical representation learning. They stacked multi-convolutional layers to extract hierarchical features of spectral-represented graphs, and the design of filter kernels is based on the eigendecomposition of the graph Laplacian matrix.

A Graph Laplacian is defined as a diagonal degree matrix $D \in R^{N \times N}$ minus the adjacency matrix A:

$$L = D - A,\ L_{norm} = I - D^{-1/2} A D^{-1/2} \tag{2.1}$$

Where L_{norm} denotes the normalised graph Laplacian. N is the number of nodes and I is the identity matrix. The eigendecomposition of L is $L = U \sum U^T$, where L is the Laplacian matrix, U is the eigenvectors matrix and $\sum = diag(\lambda)$ represents the diagonal eigenvalues matrix.

Kipf and Welling [35] defined layer-wise propagation of GCN as:

$$H_{l+1} = \sigma(D^{-1/2} \hat{A} D^{-1/2} H_l W_l) \tag{2.2}$$

where $H_0 = x$ (initial feature of node i), W_l is trainable layer wised weight matrix and σ is an activation function. $\hat{A}$ denotes the new adjacency matrix, which considers self-connections of nodes by adding the identity matrix I:

$$\hat{A} = A + I \tag{2.3}$$

Due to the excellent performance of GCNs on node classification tasks, the convolution neural networks on graph-structured data have received extensive attention [36]. As a seminal graph representation learning method, GCN has been deployed as baseline models in all four Chapters 3, 4, 5 and 6, and is directly integrated into my proposed model in Chapter 5 to explore the pairwise relationships among companies for stock prediction task.

2.2.2 Spatial-based Graph Filters

Spatial methods create convolution operators based on graph topology, which satisfy the following rules [6]:

- The feature vectors of nodes are transformed by projection.
- The information aggregation function is permutation-invariant.
- The update procedure is to consider the current values of nodes and their aggregated neighbourhood representation.

Permutation-invariant means the functions can produce the same output regardless of the input order, such as sum and mean calculation. Gilmer et al. [37] summarise this procedure as a general message passing neural network (MPNN), illustrated in Figure 2.5 as:

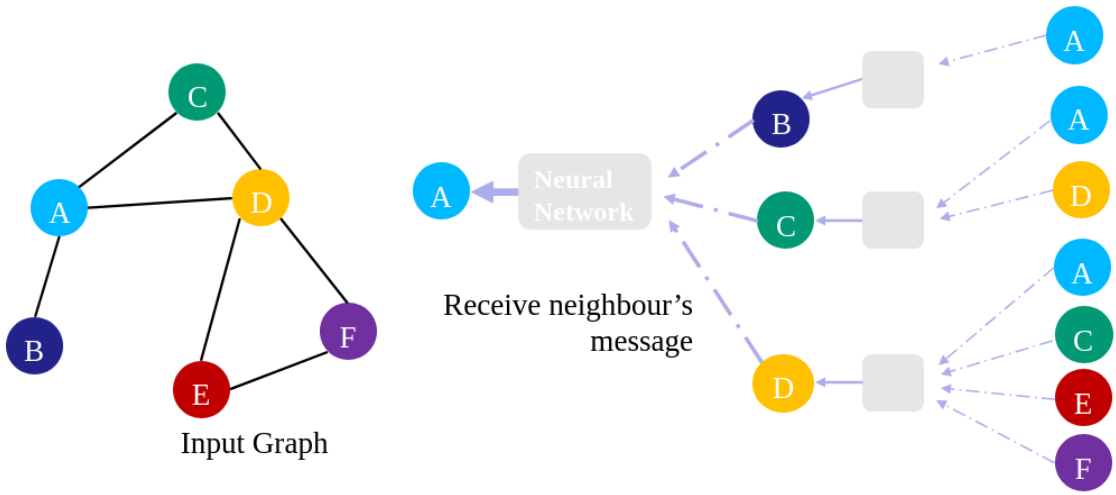


Figure 2.5: Message passing example

Initialisation The initial node features are translated into a hidden space from the feature space using a neural network Φ (in most cases, a fully-connected linear

layer):

$$H^1 = \Phi(X) = (H_1^{(1)}, ...H_n^{(1)}) \tag{2.4}$$

H_i^1 denotes the hidden representation of node i and is utilised as the initial features in subsequent iterations. Edge features can also be formulated in a similar way for later operation.

Aggregation and Update Information is passed from nodes' neighbourhoods and edges to the target node. Then the representations of a node are updated based on the passed information and its previous state:

$$H_i^{l+1} = f_U(H_i^l, \sum_{j \in N(i)} f_A(H_j^l, e_{i,j})) \tag{2.5}$$

Where H_i^{l+1} is the updated embedding of node i at layer $l+1$. $j \in N(i)$ is neighbors of node i and e_{ij} denotes the edge between node i and j. The choice of aggregate f_A and update f_U functions in GNNs is various in different models.

Readout After aggregation and update, one optional process is readout, which refers to a function that generates a fixed-size representation of the complete graph G based on the input node, edge attributes and graph topology:

$$\hat{y} = F_R(H_i^T | i \in G) \tag{2.6}$$

Where F_R denotes the readout function, which must be permutation-invariant. Downstream tasks such as graph classification, regression, or clustering can be performed using the readout function's output. Among various GNNs, MPNN is considered a leading prominent paradigm for performing deep learning tasks on graphs.

However, to sustain performance, MPNN must be able to communicate across long distances (high-order interaction) by stacking more layers, which leads to the over-squashing phenomena. MPNN is deployed in Chapter 4 and 6 to investigate the effectiveness of position information learning for when partitioning graphs into patches.

The MPNNs have a limitation in their capacity to distinguish non-isomorphic graphs, which has been examined through the Weisfeiler-Leman graph isomorphism test [38] based on colour refinement. In response to this, [39] then propose a general class of k-WL-GNNs which are able to universally represent any class of k-WL graphs. However, these models come with memory and speed complexities of $O(N^k)$, where N represents the number of nodes. This k-WL test is widely employed to evaluate the expressive power of GNNs [40]. However, directly comparing the proposed neural network with the Weisfeiler-Lehman test is challenging due to the nonlocal manner in which information is transferred between layers [3]. In Chapter 6, a comparative analysis is conducted with other existing methods that were unable to pass the k-WL test to demonstrate the effectiveness of proposed approach.

2.3 Contrastive Learning

Contrastive learning is a powerful training strategy to learn discriminative representation to maximise the agreement between representations of similar samples from dissimilar ones without the requirement of labels [41]. One seminal work is SimCLR [42], which significantly outperformed state-of-the-art at the proposed time for self-supervised and semi-supervised learning in visual learning. SimCLR followed the steps:

1. Obtain a set of augmented data by pre-processing.

2. Apply a deep neural network to extract feature representations from augmented data.

3. Train the model using a contrastive loss function in a self-supervised fashion.

The key idea is that if the representations of a sample's augmented views are not similar, then the contrastive loss function will penalise the model. For a batch of N samples, the contrastive loss function L can be generalised as [42]:

$$L = -log\frac{exp(sim(z_i, z_i'/)\tau)}{\sum_{j=1}^{N} exp(sim(z_i, z_j)/\tau)} \tag{2.7}$$

Where $sim(z_i, z_i')$ is the cosine similarity between two feature representations from two augmented versions for the same sample input i, referred to as a positive sample. z_j is the feature representation of an augmented view of another input j, known as a negative sample, and N denotes the number of examples in the batch. τ is a temperature hyperparameter to control the scale of the product.

In conclusion, contrastive learning is popular and effective at learning robust input representations, even without labels. Inspired by its outstanding performance, there is increasing interest in applying it to different learning tasks. The application of contrastive learning is extensively discussed in Chapters 4 and 6. For a comprehensive review of the relevant literature and a detailed exploration of the implementation process please refer to the two Chapters.

2.4 Transformer

The vanilla transformer [43] consists of an encoder and a decoder, as illustrated in Figure 2.6:

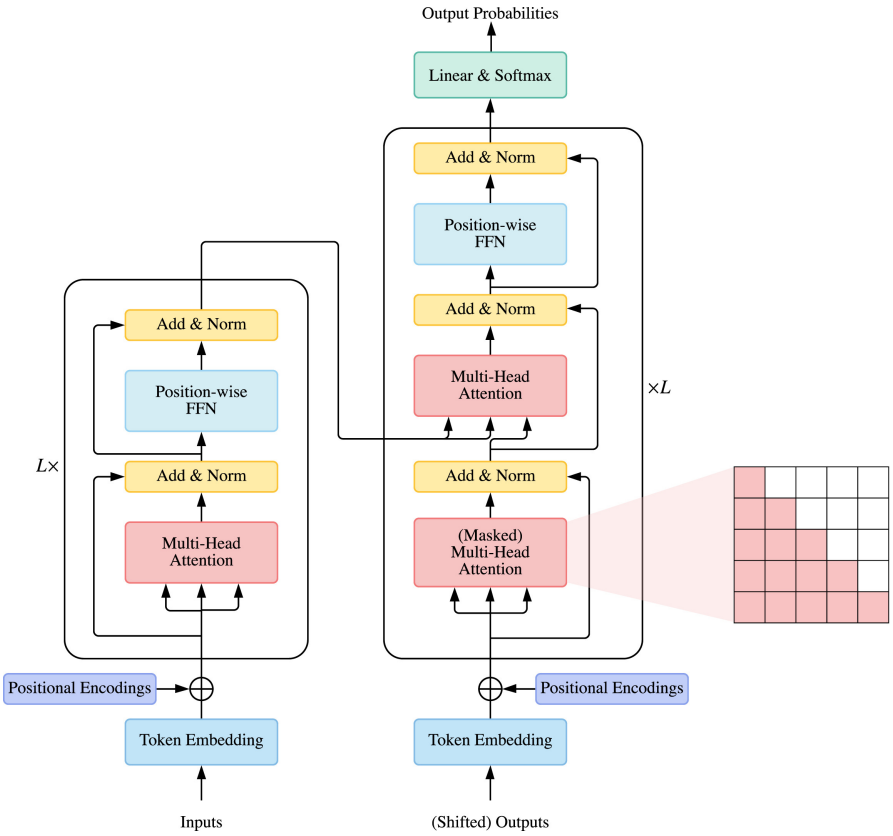


Figure 2.6: Overview of vanilla transformer, cited from [1]

The encoder comprises a feed-forward network (FFN) with position encoding and a multi-head self-attention module. It can also be extended with a residual connection [44] and layer normalisation [45]. Decoder is primarily the same as the encoder, except for a cross-attention component to consider the attention over the output of the encoder, followed by the FFN. Additionally, the self-attention layer is adjusted to rule out the later positions from the current computation so that the prediction for position i is not dependent on positions more than i. The attention score can be calculated as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.8}$$

Where Q, K, and V represent query, key and value. d_k refers to the dimensions of keys. The dot-product QK^T is divided by $\sqrt{d_k}$ to alleviate the gradient vanishing issue of the softmax operation. This attention mechanism is critical for graph representation learning and has been applied in all four Chapters 3 - 6.

The equation in 2.8 is for single head attention, and one improvement is *multi-head attention*, which concatenates all the output from equation 2.8 to weigh information at different positions from various embedding spaces jointly:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_n)W^o, \\ where head_n &= Attention(QW_n^Q, KW_n^K, VW_n^V). \end{aligned} \tag{2.9}$$

This formulation increases the expressive power of the attention model [43]. Apart from the attention mechanism, another vital component for a Transformer is Position-wise feed-forward network FFN [1]. It consists of two layers and is applied to each position identically and separately [43]:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.10}$$

Where W_1, W_2, b_1 and b_2 are trainable parameters, a ReLU activation function exists in between. For each position *pos*, positional information is calculated using

sin and *cos* function:

$$PE(pos)_i = \begin{cases} cos(w_i pos) \text{ if } i \text{ is odd} \\ \\ sin(w_i pos) \text{ if } i \text{ is even} \end{cases}$$

Where w_i is the frequency of dimension i, and this position encoding will be incorporated into token embeddings. The position representation can be added to each transformer layer's input to avoid information loss during message propagation, [46,47]. For graphs, position information can be calculated in different methods, such as Laplacian eigenvectors [48] and k-step random walk [49].

Laplacian eigenvectors in Graph Transformer [50] are calculated by the factorization of the graph Laplacian matrix:

$$L = I - D^{-1/2} A D^{-1/2} = U diag(\lambda) U^T \tag{2.11}$$

Where U and $diag(\lambda)$ are eigenvectors matrix and diagonal eigenvalues matrix. k smallest eigenvectors are picked as the unique positional encoding for nodes. This position embedding can be directly added to node features and passed to the input for the transformer in Figure 2.6.

The details of the Laplacian matrix and related operation have been discussed before. Laplacian eigenvectors represent a valid local coordinate and encode graph structure information [51] and have recently been deployed to improve the expressive power of GNNs [48, 52], defined as:

$$p_i^{lap} = [U_{i1}, U_{i2}, ..., U_{ik}] \in R^k \tag{2.12}$$

Where p_i^{lap} is Laplacian position encoding and U is the eigenvector. However, the sign of the eigenvector U is ambiguous and requires further sign flipping during training, normally random sampled between 2^k possibilities when selecting k eigenvectors [49]. Another method is the k-steps random walk, which has been proven to be more effective [16]:

$$p_i = [RW_{ii}, RW_{ii}^2, ..., RW_{ii}^k] \in R^k \tag{2.13}$$

Where p_i is the position encoding and $RW = AD^{-}1$ denotes the operator for a random walk (RW). This method only considers the probability of a node i to itself to reduce complexity. This random work can offer a distinctive node representation by ensuring that each node possesses a unique k-hop topological neighborhood, when k is sufficiently large [16]. Both position encoding methods are deployed in Chapter 6 to capture more structural information about graph patches.

2.5 Overview of Datasets

To provide clarity on the datasets used in this thesis, the table below presents an overview of the prominent datasets leveraged in prior research. It details their contextual applications and reference studies where they have been employed. By presenting this, we aim to underscore the rationale behind selecting specific datasets for the various research question tackled in this thesis.

Table 2.1: Overview of Datasets in Prior Research

Datasets	Study	Domain	Task
Cora	[35, 53, 54]	Citation Network	Node Classification
CiteSeer	[35, 53, 54]	Citation Network	Node Classification
Pubmed	[35, 53, 54]	Citation Network	Node Classification
MR	[55–58]	Heterogeneous graph neural network	Node Classification (Sentiment Analysis)
Twitter	[56–58]	Heterogeneous graph neural network	Node Classification (Sentiment Analysis)
AGNews	[56, 57]	Heterogeneous graph neural network	Node Classification (Text Classification)
Ohsumed	[55–58]	Heterogeneous graph neural network	Node Classification (Text Classification)
China A Share	[59]	Hypergraph graph neural network	Node Classification (Price Movement Prediction)
ZINC	[3, 48, 60]	Molecule Graphs	Graph Regression
Peptides-func	[3, 61, 62]	Molecule Graphs	Graph Classification
MolHIV	[3, 63, 64]	Molecule Graphs	Graph Classification
CIFAR 10	[3, 48]	Images	Graph Classification
CSL	[3, 48, 64, 65]	Synthetic Dataset	Test expressivity of GNNS
SR25	[3, 66]	Synthetic Dataset	Test Expressivity of GNNS
EXP	[3, 66]	Synthetic Dataset	Test expressivity of GNNS

There are five types of datasets used to answer the three research questions, accordingly, which were posed in Chapter 1.2 in this thesis:

- Citation networks datasets including Cora, CiteSeer and Pubmed [67] in Chapter 3 for addressing the robustness problem in node classification, with only a small portion available labelled data.

- Short text datasets to study the issue of heterogeneity and robustness in Chapter4, including AGNews [68], Ohsumed [55], where labels are the topic of the news or bibliographic document, MR [69] and Twitter (both are for sentiment classification) provided by NLTK[1].

- Finance dataset (China A-share market stock price from 2013 to 2019), provided by [36] to capture more complex relations among companies in different industries and held by various shareholders in Chapter 5. Each node repre-

[1] `https://www.nltk.org/`

sents a company and two nodes are connected if they are in the same industry or invested by the same shareholders.

- Molecule/Chemistry datasets including Zinc [48], Molhiv [70], Peptides-func [62] and image dataset CIFAR10 [48] for graph structure learning (positional information, long-range dependencies, over-squash) in Chapter 6.

- Simulated/synthetic datasets including CSL [71], SR25 [72] and EXP [73] to demonstrate the enhance expressive power of my proposed framework in Chapter 6.

2.6 Summary

This chapter presents the core concepts of graph representation learning that underpin our advanced methods. We discuss essential graph theories, the central role of message passing mechanism in graph neural networks and the potential of transformers and contrastive learning. Additionally, we provide a succinct overview of datasets employed to tackle specific research questions. The next chapter will address the first research question, regarding robustness in data-scarce scenarios.

CHAPTER 3

Robustness problem with graph representation learning

In this chapter, we introduce a novel graph generative model, named Bayesian Graph Attention Network (BGAT) to solve the first research question mentioned in Section 1.2:

How to increase robustness regarding uncertainty (few labels available during training or noisy input) in representation learning based on graph neural networks, without inferior performance?

We apply the proposed BGAT model for node classification tasks under data-scarce contexts. Our model performs against earlier state-of-the-art and can be used as a future baseline for graph generative models. This chapter is based on the following publication as listed in Chapter 1.5:

***Sun, Z.**, Harit, A., Yu, J., Cristea, A. I., Al Moubayed, N. (2021). A Genera-*

tive Bayesian Graph Attention Network for Semi-Supervised Classification on Scarce Data. In 2021 International Joint Conference on Neural Networks (IJCNN, Core A Ranked Conference). pp. 1-7. IEEE.

3.1 Introduction

Graph representation learning has recently drawn the attention of researchers from across various domains, including computer vision, natural language processing, knowledge graphs, and social networks [5], as discussed in Chapter 2. Despite the proven learning abilities of existing GNN models, their inference performance is compromised for semi-supervised tasks when only limited labelled data is available [54]. Additionally, most existing studies [35, 74, 75] process input graphs as the ground truth but neglect the fact that noise or spurious edges generated from model assumptions may be included and thus lack robustness.

Addressing the above problems, [54] proposed a generative graph model to infer the joint posterior distribution of weights of a Graph Convolutional Network (GCN) and the graph structure of the input. However, their posterior inference of the graph is mainly conditioned on the structure of the observed graph and neglects the information from the node features. As the data may be correlated with the actual graph structure, the information of features is missing which resulted in a moderate performance [76], particularly under data-scarce situations.

In this chapter, we introduce a generative model that considers the neighboring nodes' features and labels using an attention mechanism. We propose a Bayesian graph attention network (BGAT) model, to simultaneously boost the model's per-

formance and robustness when solving semi-supervised classification.

In summary, the main contributions of this work are:

1. We propose a novel BGAT model combining GAT and the Bayesian method, which allows *accounting for uncertain information*, such as spurious and missing edges between nodes in a graph, by viewing the observed graphs, as generated from a parametric random graph family.

2. We demonstrate our model's improved performance in classification tasks under data-scarce (under-labeled data) situations.

3.2 Related Work

As addressed in Chapter 2, graph neural network (GNN) was initially introduced by [31] as updating nodes' states iteratively, until reaching stable states, by propagating discrete features from neighboring nodes. Scarselli et al. [32] elaborated it further, by extending recurrent models to deal with graphs, including directed, cyclic, or mixed graphs based on information diffusion and relaxation methods [8]. Later, [37] developed a new framework for GNNs, which contained a message passing and readout phases. It improved the performance at learning hierarchical graphical representations by using sub-graphs. GNNs are effective at learning node representations via feature propagation but generally struggle to model the dependencies between various node labels. These methods usually assume a central node classification is conditionally independent of the features, but seldomly [76] model the joint distribution of the node labels and graph to extract more relationships among nodes.

Most existing studies consider the input graph a fixed observation and assume it represents ground-truth information. In such cases, neural networks do not consider the uncertainty of information, including spurious edges and missing edges in graphs. A few studies [54, 76, 77] have understood the problem and proposed to solve it by using generative models. Ng et al. [77] introduced a Gaussian process-based approach in semi-supervised learning and [54] incorporated the stochastic block model and used Monte Carlo dropout [78] to represent model uncertainty in deep learning. Recently, [76] introduced a graph-based generative framework for semi-supervised learning, but they did not consider the data-scarce situations in active learning, which is the problem of interest in this Chapter.

Our study builds on the work by [54], as their idea of considering the uncertain graph information based on GCN is practical and effective for semi-supervised learning, particularly under data-scarce situations. However, as addressed above, they did not consider the specific weights of highly correlated features and the interaction between those features and graph structures. By learning the influence among nodes, the attention mechanism could mitigate the problem of including spurious edges as addressed by [53]. We are interested in combining Bayesian methods with GAT to learn the joint distribution of labels, features and graphs under data-scarce situations for semi-supervised node classification tasks.

3.3 Model Building - Preliminaries

3.3.1 Graph Attention Mechanism

The Graph Attention Network (GAT) was proposed by [53] and its power consisted in it considering the relative influence between neighboring nodes and central nodes, instead of the fixed weights used in GCN, based on attention mechanisms. It can be outlined as:

$$h_i^l = \sigma(\sum \alpha_{ij}^l W^t h_j^{l-1}) \tag{3.1}$$

Where α_{ij}^t is the attention score between node i and j and can be calculated as:

$$\alpha_{ij}^t = softmax(\sigma(a[Wh_i^{l-1}||Wh_j^{l-1}])) \tag{3.2}$$

σ is the LeakyReLU activation, Wh are the weights of node i and j in layer $l-1$. Operation $||$ is a concatenation and the softmax function is used to sum up all neighbors of node i.

Using multiple attention layers enables a model to attend to information differently, yet it is often assumed that each attention head has equal importance [53].

More recently, [79] proposed a Gated Attention Network (GaAN) model, to further take into account the importance of each attention head, separately. The model performed well for both inductive node classification and traffic speed forecasting tasks, and the authors argued that it could also be possibly extended to integrate edge features for massive graphs.

3.3.2 Bayesian Neural Network

According to Seedat and Kanan [80], Bayesian neural networks represent uncertainty by formulating a network's parameters in the manner of a probabilistic distribution. The weight matrix can be modelled as random variables that $p(W_l)$ in layer l can be defined as $W_l \sim N(0, \mathrm{I})$, by introducing the standard matrix Gaussian prior distribution with bias vector b_l.

Considering the posterior distribution of the weight matrix W, the predictive function for a new point x* with training data X and training label Y can be formulated as:

$$p(y|x^*, X, Y) = \int p\left(y|x^*, w\right) p(w|X, Y) dw \tag{3.3}$$

As the functional form of a neural network is difficult to integrate, the exact calculation of the model posterior $p(w|X, Y)$ is generally intractable [81] and cannot usually be analysed in a close form. Therefore, we use the Monte Carlo dropout method [78] to approximate it, which accounts for model uncertainty in deep learning. We can draw samples from the approximate posterior and average the weight matrix W of the network with T stochastic forward passes:

$$p(y|x^*, X, Y) \approx \frac{1}{T} \sum_{t=1}^{T} p(y|x^*, W_1^t, ..., W_l^t) \tag{3.4}$$

3.3.3 Mixed Membership Stochastic Block Model

The mixed membership stochastic block model (MMSBM) [82] is a popular framework for community detection [83], which considers a graph $G(V, E)$ and the asso-

ciated adjacency matrix A. There are n nodes in the graph, denoted by $x_1, \ldots, x_n$. An adjacency matrix A for this graph is an n by n dimensional matrix as explained in chapter 2. If there is no connection between node x_p and node x_q, $A(p,q) = 0$, otherwise $A(p,q) = 1$. The MMSBM models the adjacency matrix A in a Bayesian hierarchical framework. According to a very recent work of [84], the absence or presence of a link between any pair of nodes (x_p, x_q) is described by a Bernoulli distribution B with a latent group membership $z_{p,q,1}$ $z_{p,q,2}$:

$$A(p,q) \mid z_{p,q,1}, z_{p,q,2}, B \sim \text{Bernoulli}\left(z_{p,q,1}^T B z_{p,q,2}\right) \tag{3.5}$$

The Bernoulli probability matrix B has a K by K dimension, which, for community detection, represents the number of communities in the data. As we focus on undirected graphs, $z_{p,q,1}$ means the distribution of node x_p is interacting with x_q in community 1 where both nodes are K-dimensional vectors, where only one element equals to one. It can be denoted as $z = [z_1, ...z_K]^T$, indicating the corresponding community the node belongs to (the rest being zero). According to [84], the joint distribution of latent group memberships of nodes Z and data X is:

$$\begin{aligned} p\left(X, Z_1, Z_2, \pi \mid \alpha, B\right) = & \prod_{p,q} p_1\left(X(p,q) \mid z_{pq1}, z_{p,q,2}, B\right) \\ & p_2\left(z_{p,q,1} \mid \pi_p\right) p_2\left(z_{p,q,2} \mid \pi_q\right) \prod_p p_3\left(\pi_p \mid \boldsymbol{\alpha}\right) \end{aligned} \tag{3.6}$$

p_1 is the Bernoulli distribution (β) which refers to the possibility of a link between two nodes. p_2, p_3 are prior of latent group membership and the prior of the former one, with Beta and Dirichlet distributions, respectively.

3.4 Methodology

3.4.1 Learning Attention Using the Bayesian Framework

We consider a semi-supervised learning problem leverage the graph attention network method. The deterministic attention weights are transformed into a distribution, making it straightforward and requiring minimal changes to the standard attention model. We can adapt a pre-trained standard attention model for variational fine-tuning. The graph structure is encoded in the attention masks so that nodes can only attend to the neighborhood's features in the graph.

3.4.2 BGAT

The inspiration for the Bayesian Graph Attention model comes from the Bayesian Graph Convolutional Neural Network (BGCN) model [54]. While the foundational Bayesian approach and the usage of the mixed membership stochastic block (MMSBM) model to generate random graphs is similar to BGCN, the key distinction lies in the application of the graph attention network (GAT) to our model, as apposed to the graph convolution network (GCN) employed in BGCN. In our design, we utilise a building block layer to construct an arbitrary graph attention network by stacking this layer, further integrating a Bayesian methodology, especially beneficial for classification in data-scarce situations.

In this chapter, the Bayesian approach views a graph as a realisation from a parametric family of random graphs based on the known labels of nodes, Y and structures from observed graphs. The joint posterior of the weights in GAT, the

parameters of the random graph and the remaining unknown node labels are the target inference. By marginalisation, the graph parameters, and posterior estimation of the labels could be inferred and obtained. Then we combine the posterior of labels and attention of nodes and implement a softmax function to obtain the final output. The posterior probability of labels is formulated as:

$$p\left(\mathbf{Z} \mid G_{obs}, \mathbf{X}, \mathbf{Y}\right) = \int p(\mathbf{Z} \mid W, G_r, \mathbf{X}) p\left(W \mid \mathbf{Y}, \mathbf{X}, G_r\right) p(G_r \mid \zeta) p\left(\zeta \mid G_{obs}\right) dW dG_r d\zeta \tag{3.7}$$

ζ is the parameter that describes a family of random graphs G_r, which can be derived from the observed graph G_{obs} using the MMSBM random graph model. W is the sampled weights of BGAT over the random graphs G_r by approximating variational inference via Monte Carlo dropout as aforementioned in 3.3.2. $\mathbf{Y}$ is the known label of nodes and later GAT will take (X, G) as input to infer the unknown labels of nodes. A softmax function will be added to the output of GAT to model $p(Z|X, Y, G_{obs})$ in a categorical distribution. Figure 3.1 provides a detailed schematic of our Bayesian GAT model.

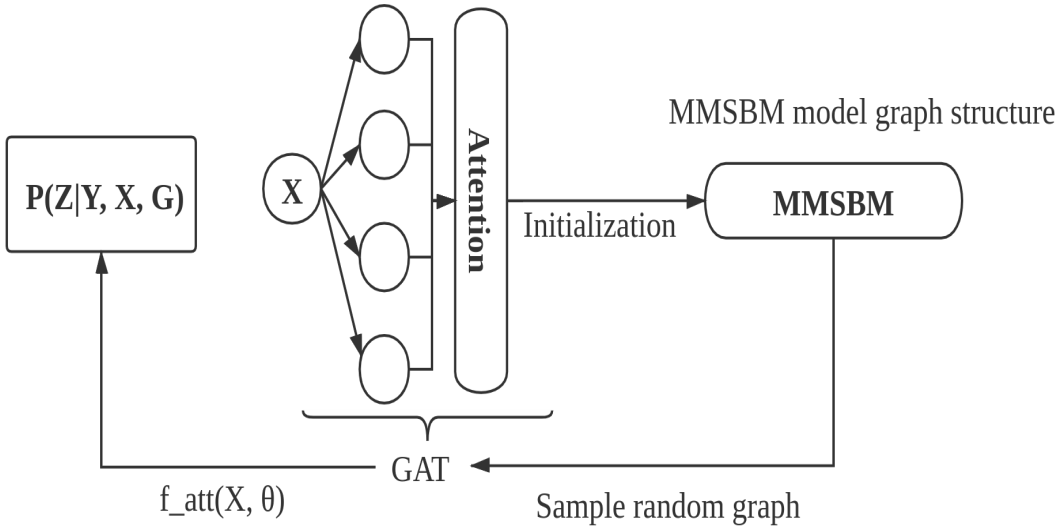


Figure 3.1: Overview of BGAT

Since the highly non-linear nature of likelihood leads to intractable computation of posterior in the equation (3.7), we can use variational inference [78, 85, 86] or MCMC [87, 88] to approximate the posterior of $p(W|Y, X, G_r)$. According to [89], averaging the weights of the network is an approximate way of Monte Carlo dropout. The weight metrices W can then be sampled from $p\left(W \mid \mathbf{Y}, \mathbf{X}, G_r\right)$ using Monte Carlo dropout given the sampled graphs generated from $p(G_r|\zeta)$. To model $p(\zeta|G_{obs})$, parametric random graph generation models, such as degree corrected block model [90] and mixed membership stochastic block model [82] could be considered. In summary, the Monte Carlo approximation of equation (3.7) is:

$$\begin{aligned} &p\left(\mathbf{Z} \mid \mathbf{Y}, \mathbf{X}, G_{\text{obs}}\right) \approx \\ &\frac{1}{V} \sum_{v}^{V} \frac{1}{N_G S} \sum_{a=1}^{N_C} \sum_{s=1}^{S} p\left(\mathbf{Z} \mid W_{s,a}, G_a, \mathbf{X}\right) \end{aligned} \tag{3.8}$$

G_a is the graph sampled from the random graphs G_r and the weight matrix

$W_{s,a}$ is sampled from $p\left(W \mid \mathbf{Y}, \mathbf{X}, G_r\right)$ based on G_a. For the Bayesian GAT, we use a similar Mixed Membership Stochastic Block Model (MMSBM) setting used in Bayesian GCN [54] for the graph and learn its parameter $\zeta = \{\beta, z\}$ using stochastic optimization to maximize the posterior of β and z based on the observed graph G_{obs}.

As addressed in section 3.3.3, the MMSBM model is used to model the random graph based on the observed graphs, which helps us establish a strong community structure between nodes and determine which community node may belong to. If any two nodes belong to the same community, meaning they have the same label, and it is highly likely to have a link between them, compared to when the two nodes belong to different communities [82]. β is to denote the possibility that there is a link between any two nodes and z is the parameter for the community membership probability distribution of nodes, and the priors of them are Beta and Dirichlet distribution, respectively.

The posterior probability of labels $p(Z|Y, X, G_{obs})$ is modelled as a K-dimensional categorical distribution, where K is the number of classes/communities of the data. In GAT, a weight matrix W containing the F dimensions of features of the nodes is introduced as an initial linear transformation, and then self-attention will be applied, to compute attention coefficients, as the relative influence of node j's features on node i, defined by [53]:

$$e_{ij} = a\left(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j\right) \tag{3.9}$$

Then we can use equation 3.8, the Monte Carlo dropout method to approximate the posterior of labels and model the posterior using a categorical distribution to

establish the community membership among nodes. We then model the posterior of labels and learned attention of nodes by applying the softmax function to the output of GAT. The attention improves the model's expressiveness and the Bayesian GAT model leverages deterministic self-attention layers to process node features for graph node classification. The graph structure is encoded in adjacency matrix, and nodes can only attend to their neighborhoods' features in the graph.

3.5 Experiments

Experiments implemented to demonstrate the effectiveness of our model are described in this section.

3.5.1 Datasets

As mentioned, we perform a semi-supervised node classification task on three citation datasets: Cora, CiteSeer, and Pubmed [91] follow a random split. The details of each dataset are summarised in Table 3.1. In these datasets, each node represents a scientific document and if one paper cites another, there will be an undirected edge between them, shown in Figure 3.3 and 3.4, where Citeseer is more decentralised, compared with Cora. The decision to disregard the direction of citation aligns with the approach taken by the baselines GCN, GAT and BGCN models, and was made to simplify the network structure. In the context of node classification, our primary aim is to understand the relations between documents to derive meaningful node representation. By emphasising the mere presence of relations, instead of directionality, we reduce the potential complexities. Although it is to be noted that

considering the direction of citations might offer insights into the flow of information among documents. Each node has a sparse feature vector (keywords of the document) and the label describes the topic of the document. For instance, each node has 1433 dimensions of features attached to it, represented as 0 or 1 in Cora and the node label in the last column represents the topic/community that the document belongs to, as shown in Figure 3.2. Please note that we only have access to a few nodes per class during training to infer labels for other nodes.

A distinguishing factor is that the Pubmed dataset uses TF-IDF (Term Frequency-Inverse Document Frequency) features. In contrast, Cora and CiteSeer use binary vectors, indicating the presence or absence of terms. This differentiation in feature choice is not arbitrary, but is intrinsic to the datasets. It is crucial to note that this particular choice of features, TF-IDF for Pubmed and binary indicators for Cora and CiteSeer was preordained by the dataset curators [91]. This has been consistently adhered to by other baselines in the field, ensuring uniformity and comparability in experimental results.

Table 3.1: Dataset summary

Datasets	Cora	Citeseer	Pubmed
Nodes	2708	3327	19717
Edges	5429	4732	44338
Communities	7	6	3
Features	1433	3703	500
Features Type	Binary	Binary	TF/IDF
Average Degree	4	2	3

ID	w_0	w_1	w_2	w_3	w_4	w_5	w_1430	w_1431	w_1432	subject
31336	0	0	0	0	0	0	0	0	0	Neural_Networks
1061127	0	0	0	0	0	0	0	0	0	Rule_Learning
1106406	0	0	0	0	0	0	0	0	0	Reinforcement_Learning
13195	0	0	0	0	0	0	0	0	0	Reinforcement_Learning
37879	0	0	0	0	0	0	0	0	0	Probabilistic_Methods
...	...	...	...	...	...	...	...	...	...	...
1128975	0	0	0	0	0	0	0	0	0	Genetic_Algorithms
1128977	0	0	0	0	0	0	0	0	0	Genetic_Algorithms
1128978	0	0	0	0	0	0	0	0	0	Genetic_Algorithms
117328	0	0	0	0	1	0	0	0	0	Case_Based
24043	0	0	0	0	0	0	0	0	0	Neural_Networks

Figure 3.2: Layout of the Cora Dataset

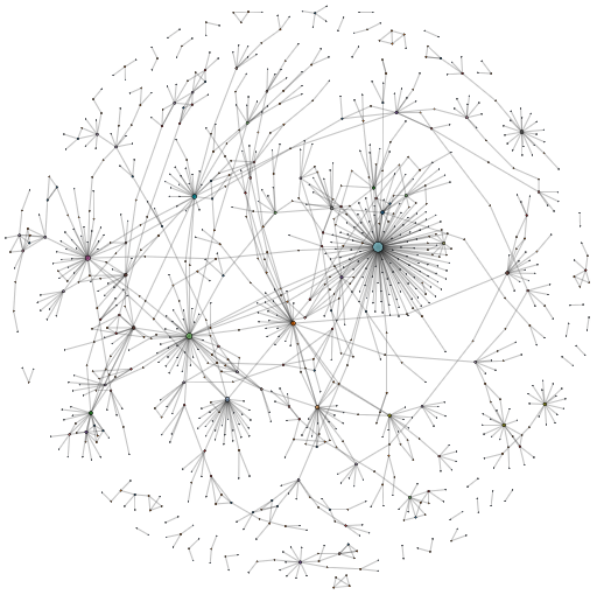


Figure 3.3: Visualisation of the Cora Dataset

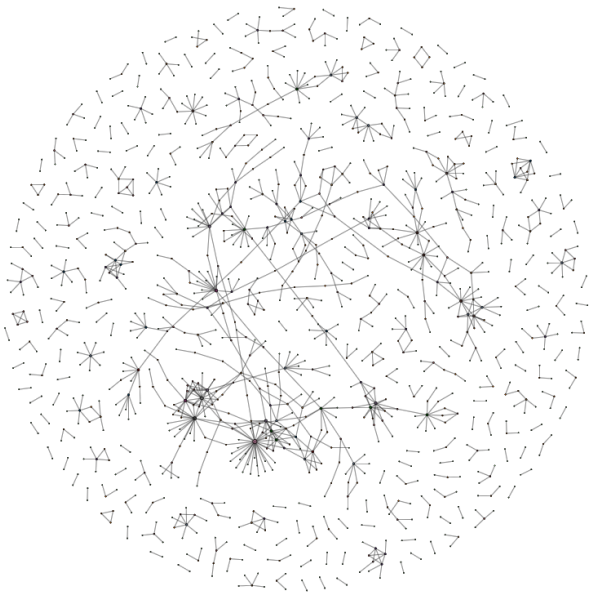


Figure 3.4: Visualisation of the Citeseer Dataset

3.5.2 Experimental Setup

Our model is implemented with Pytorch 1.4.0 with Cuda 10.2. The hyper-parameters of BGAT are borrowed from the experiments of GAT [53] and BGCN [54]. We use

two layers and the number of hidden units is 16, with a 50 percent dropout rate at each layer. The learning rate is 0.01 and the L2 regularisation parameter is 0.0005. In addition, the hyper-parameters associated with the Mixed Membership Stochastic Block Model (MMSBM) inference are: n =500, $\varepsilon_0 = 1$, $\tau = 1024$, $\eta = 1$ and $\alpha = 1$.

Three different experimental settings for a semi-supervised classification task have been considered, where 5, 10 and 20 labels per class are available in the training set, to infer labels for the others. The partitioning of the data into 20 labels per class is set the same as in [54], whereas in the other two cases, the training sets are constructed by considering the first 5 or 10 labels from the previous partition.

3.5.3 Baselines

We consider three widely applied graph learning models and one previous work on semi-supervised learning under data-scarce situations as baselines: ChebyNet, GCN, GAT and BGCN.

- **ChebyNet** A spectral CNN-based model uses Chebyshev polynomials to approximate and localize filters of graphs [92].
- **GCN** A spectral CNN-based method that deploys the first-order approximation of ChebNet and assigns a non-parametric weight of neighborhood to central nodes [35].
- **GAT** An attention mechanism-based neural network method that considers the weight of neighbor information to central nodes [53].

- **BGCN** A Bayesian GCNN framework that considers the randomness of input graphs by incorporating the Bayesian method with GCN [54].

3.6 Results and Discussion

We compare the performance of various models, including the proposed model, across different datasets. The mean and standard deviation of the proposed method's test accuracy versus the baselines are presented in the table below:

Table 3.2: Average Prediction Accuracy of Models in Datasets

	Cora			CiteSeer			Pubmed		
Model	5 labels	10 labels	20 labels	5 labels	10 labels	20 labels	5 labels	10 labels	20 labels
ChebyNet	61.7±6.8	72.5±3.4	78.8±1.6	58.5±4.8	65.8±2.8	67.5±1.9	62.7±6.9	68.6±5.0	74.3±3.0
GCN	70.0±3.7	76.0±2.2	79.8±1.8	58.5±4.7	65.4±2.6	67.8±2.3	69.7±4.5	**73.9 ± 3.4**	**77.5 ± 2.5**
GAT	70.4±3.7	76.6±2.8	79.9±1.8	56.7±5.1	64.1±3.3	67.6±2.3	68.0±4.8	72.6±3.6	76.4±3.0
BGCN	74.6±2.8	77.5±2.6	80.2±1.5	63.0±4.8	69.9±2.3	71.1±1.8	70.2±4.5	73.3±3.1	76.0±2.6
BGAT	**74.8 ± 4.5**	**78.8 ± 2.8**	**84.3 ± 1.8**	**68.6 ± 4.6**	**71.4 ± 2.6**	**74.2 ± 1.6**	**71.4 ± 4.7**	72.3±3.4	74.5±2.4

Our proposed model achieves a better performance in all bar 2 cases. For instance, the proposed model improves more than 4% and 5% of the test set accuracy in the Cora and CiteSeer data sets, when there are 20 and 5 labels available, respectively. A paired t-test is conducted to compare the mean performance of our method and BGCN on the same datasets (Cora and CiteSeer) over 4 runs. The results of the paired t-test confirm that the improvements introduced by our model for classification tasks under data-scarce situations are statistically significant (p-value $\leq$ 0.05). Nevertheless, the proposed model does not reach the highest accuracy in the Pubmed dataset, for the cases of 10 and 20 labels per community. GCN presents

the most expressive power for the Pubmed dataset. One possible explanation is that there are more low-pass subgraphs in Pubmed and repeated graph propagation is the primary source of the expressive power of GCN [93]. Another determinant of this performance variation is the TF-IDF feature in Pubmed, which contrasts with the binary features in the other datasets. The inherent nature of these TF-IDF features might align more harmoniously with the GCN's low pass filtering mechanism. Additionally, while GAT model offers a fine-grained approach by considering influences of different nodes, its performance may be compromised when paired with the random input graphs generated by the MMSBM method, underscoring a potential trade-off between granularity and stability.

3.7 Potential Future Work

There are several potential extensions to our work that could be addressed as a future study. One is to investigate how to extend the generative models, by accounting for more graph structure information to other graph-based learning tasks. For instance, the direction of the citation could be considered and modelled in the graph generation process. Moreover, extending the method's expressive power with sub-structure counting could also be insightful, from the application perspective. Finally, extending the model with more scalable techniques would allow us to perform practical inference over large-scale graphs under data-scarce situations.

3.8 Summary

This work introduces a novel method for graph-based semi-supervised learning, which allows for considering *uncertainty in the graph generation process*. We present how to incorporate MMSBM with a graph attention mechanism and examine our model on three graph-based deep learning benchmark datasets. The results demonstrate that the proposed model outperforms other graph-based semi-supervised learning methods when there are only a few labels of the nodes known for classification tasks in most settings. Given the robustness and performance of BGAT, it could be used as a new baseline in future generative graph learning. However, a limitation of this study is that it solely focuses on homogeneous graphs. In reality, graphs often comprise various types of nodes and edges. Consequently, the subsequent chapter will delve into exploring the challenges posed by heterogeneity and robustness in heterogeneous graph representation learning.

CHAPTER 4

Heterogeneity problem with graph representation learning

In this chapter, we develop a novel contrastive graph learning model based on existing state-of-the-art, named Neighbour Contrastive Heterogeneous Graph Attention Network (NC-HGAT) to solve the second research question we mentioned in Chapter 1.2:

How to account for the heterogeneity of graphs, which contain diverse types of edges, features and attributes, in graph neural networks?

We apply the proposed NC-HGAT model for short text classification tasks with limited available labels. Entities, sentences and topics are represented in different types of nodes, connected by different types of relations. Our model performs against state-of-the-art and provides a new perspective for the message passing mechanism in graph representation learning. This chapter is not only based on the following

publication as listed in Chapter 1.5:

***Sun, Z.**, Harit, A., Cristea, A. I., Yu, J., Shi, L., Al Moubayed, N. (2022). Contrastive Learning with Heterogeneous Graph Attention Networks on Short Text Classification. In 2022 International Joint Conference on Neural Networks (IJCNN, Core A Ranked Conference). pp. 1-6. IEEE.*

but also extends the work with edge-level attention consideration and demonstrates the improved performance of taking into account edge relations in text classification tasks in section 4.3.3 and improved robustness by contrastive learning, in section 4.7.

4.1 Introduction

Text classification is a fundamental task in natural language processing (NLP), which can be applied to various downstream tasks, such as question answering, machine translation and sentiment analysis [94]. The representation learning ability of textual features is a leading cause of the high performance of text classification models. Consequently, it is a pressing need to study how to extract textual features more effectively. Recently, graph neural networks (GNNs) have been increasingly applied to text classification tasks, due to their advantages of dealing with complex semantics and topological information, by modelling texts as graph structure [8]. Graphs in such studies [95, 96] usually consist of different types of nodes, which represent words or documents, and edges, to indicate relations. These graphs are known as heterogeneous information networks (HIN) or heterogeneous graphs. Different from most existing studies that focus on *long* text classification, we mainly focus on *short*

text classification, as our daily communication is increasingly completed via short texts, such as tweets, messenger and online comments. Thus it has become more important to study this field.

Most existing GNN studies focused on text classification tasks are trained in a semi-supervised manner, similar to the vanilla Graph Convolution Network (GCN) [35]. Such methods require a large set of labelled data, which is often unattainable in many real-life scenarios. Therefore, the shortage of labelled data may undermine the performances of graph neural network models on classification tasks, particularly with large-scale data [22, 56].

On the other hand, although a GCN can encode local topological properties, it may fail to capture the global structural information fully [97]. To be more specific, existing methods for text classification mainly learn direct neighbourhoods and the associated textual features by supervised information aggregation. They may not be able to incorporate the high-order, rich relations among texts [98], and thus undermine the ability to capture inherent heterogeneity in texts, particularly when the connections among nodes are noisy or missing [99], as is the case with the data considering only 40 labels known per class for training in this chapter.

To address the above problems, we propose integrating neighbouring contrastive learning (NC) with the heterogeneous graph attention network (HGAT), forming NC-HGAT. HGAT is the state-of-the-art work for text classification tasks, proposed by [56] to embed HIN with a dual-level attention mechanism for both nodes and relations. However, HGAT mostly focuses on the node influence and neglects the impacts of edges, where the relationships between words (nodes) can provide

crucial context that aids in understanding the meaning of the text. After the base paper was published, this chapter addressed the problem by adding an edge-level attention module in section 4.3.3 in this thesis. Contrastive learning can learn intrinsic and transferable topological information, enhance the performance of graph neural networks [100], and is widely applied in NLP tasks for pre-training [101]. NC learning enables our proposed model to transform k^{th} structural-aware features without using direct message-passing modules, and hence improve the ability to capture the nuanced differences between different types of entities, despite missing connections between words during inference [99], when labelled data is limited.

The contributions of this Chapter are summarised as follows:

- To the best of our knowledge, this study is *the first attempt* to apply contrastive learning with a heterogeneous graph neural network to text classification tasks.

- We propose to use a simple MLP to learn the neighbouring information without direct message-passing, which can be easily applied to existing graph neural network models [99] to text classification.

- Experimental results on three of the four datasets analysed show NC-HGAT outperforming the state-of-the-art on short text classification with limited labelled data, and it also delivers a competitive result on the fourth dataset.

4.2 Related Work

4.2.1 Text Classification

Extensive studies have been conducted on text classification, such as traditional machine learning using manually designed features [102], convolutional neural networks [103], and recurrent neural networks [104]. Recently, graph neural networks (GNNs) have shown promising performance in text classification, as text can be modelled as edges and nodes in a graph structure. For example, TextGCN [105] applied the vanilla GCN to heterogeneous graphs on graphs built from a text corpus and gained improved results.

Hu et al. [56] proposed a novel heterogeneous graph attention networks model (HGAT) with a dual attention mechanism to consider more relations between different nodes. Recently, [57] introduced an orphan category to HGAT to remove unrelated stop-words, which improves classification accuracy. Liu et al. [98] also incorporated the attention mechanism with deep diffusion layers to enrich the context information of texts.

Ding et al. [106] constructed hypergraphs for text classification to capture high-order interaction between words. However, these methods all relied heavily on the direct message-passing function to learn node-wise feature transformation, and the performance decreased when labelled training data was limited. We thus propose, *for the first time*, to the best of our knowledge, to solve the problem by applying contrastive learning of graph structure in text classification tasks.

4.2.2 Contrastive Learning

Contrastive learning is a discriminative approach which aims to learn embeddings of objects, shorten the distance between similar entities, and lengthen the gap among dissimilar entities [107]. It aligns with the classification objective [108] and has increasingly been applied to computer vision and NLP tasks. Contrastive learning can be used in both self-supervised representation learning [109–111] and supervised learning [101, 112, 113]. In NLP tasks, [114] performed contrastive learning on adversarial samples to improve text classification, and [115] applied it to obtain more effective embeddings of words to mitigate the problem of data scarcity.

For graph learning, contrastive learning between global and local objects can better capture structural information [116, 117]. For instance, [118, 119] proposed to implement contrastive learning by maximising the mutual information between representations of substructures (e.g. triangles) and graph-level representation. Peng et al. [120] explored natural supervision signals from a global context to learn useful individual node representations. However, these methods mostly focused on graph-level rather than node-level tasks. [121–125] built contrastive multi-structural views between either node or subgraphs and transformed graphs for node classification tasks. However, [126] pointed out that increasing the number of views does not improve representation learning performance for graph-structured data. The number of contrastive views needs to be confirmed before training and hand-picking of the suitable views will reduce the generalisation capability [127]. Although few works explored how to build the optimal views [128, 129], it is task-dependent and impractical when we have no prior knowledge. Moreover, these studies did not con-

sider the heterogeneity of different types of information (e.g., entities, topics in this study). Therefore, we do not construct different views of graphs but mainly consider the effectiveness of contrastive learning in capturing the heterogeneity on the text classification task.

4.3 Methods

In this section, we will introduce our NC-HGAT model, a merger of the HGAT model [56] and neighboring contrastive (NC) learning adapted from the Graph-MLP model [99].

4.3.1 Construct Heterogeneous Graph from Data

Here we apply the same approach as in [56] to construct heterogeneous graphs from texts. This is briefly illustrated below, for clarity. Specifically, a heterogeneous graph can be denoted as $G = (V, \mathcal{E})$, where V is the union of entities E, topics T, and short texts S; and $\mathcal{E} = \mathcal{E}_1, \mathcal{E}_2, ...$ is the set of edges representing the relations between them, as shown in Figure 4.1.

Each short text (document) is assigned to a number of top M possible topics $(t_1, t_2...t_M)$ using LDA [102] and the entities E are mapped to Wikipedia via TagME[1], an entity linking tool. Edges will be created if an entity e is contained in a document s or the document is assigned to a topic with a threshold of 0.5. If the cosine similarity between entity embeddings calculated by word2vec based on

[1]https://sobigdata.d4science.org/web/tagme/tagme-help

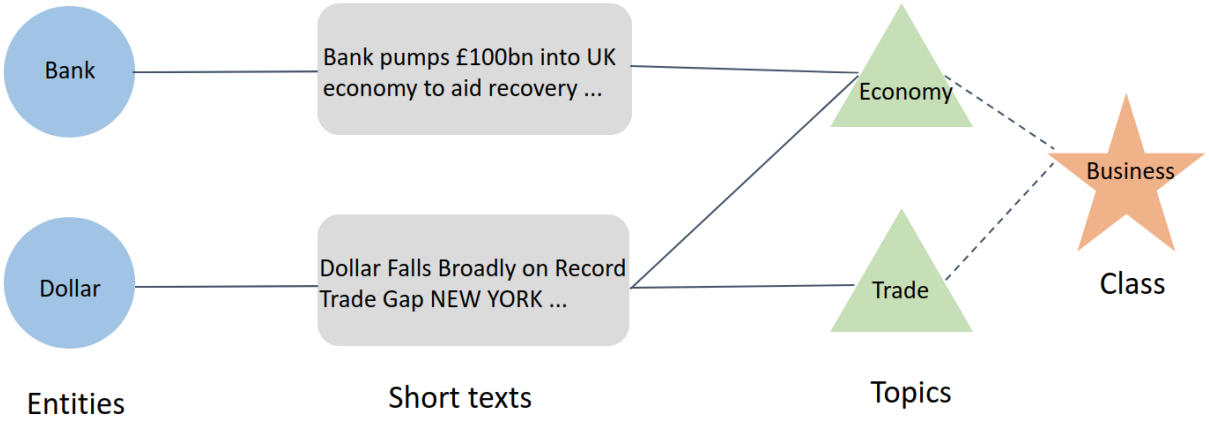


Figure 4.1: Example Snippet from a Heterogeneous Graph Structure

the Wikipedia corpus [2] is higher than a threshold 0.5, then an edge will connect the two entities. This threshold choice is consistent with the setup employed by [56]. We chose to adhere to this established threshold, to maintain consistency with prior work, and to ensure a fair comparison. The rationale behind using a positive value for the threshold is to ensure a reasonably strong semantic similarity between entities. Considering entities, short texts and topics in Figure 4.1, it is highly likely that the documents in the figure would be classified with their correct label as "Business". Therefore, the task can be described by Figure 4.2, which is to predict the label for the new document.

[2] https://code.google.com/archive/p/word2vec/

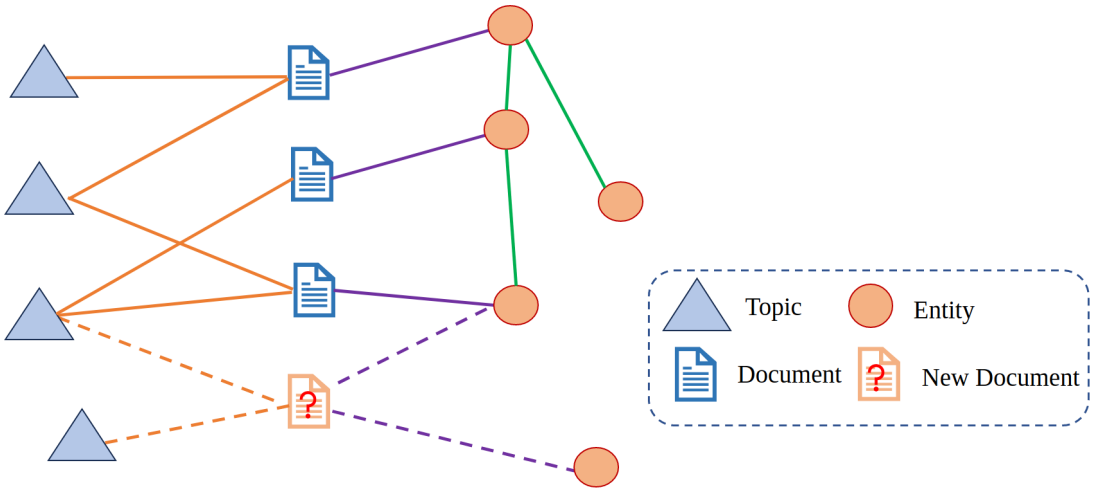


Figure 4.2: Illustration of text classification task

The overall structure of our model is shown in Figure 4.3, where we apply the HGAT model (circled by the orange colour dash line) to construct text graphs and utilise an MLP-based model to update features, then calculate the similarity among nodes within the same input batch, based on an adjacency matrix. The details are explained in the later subsections 4.3.2, 4.3.4 and 4.3.5.

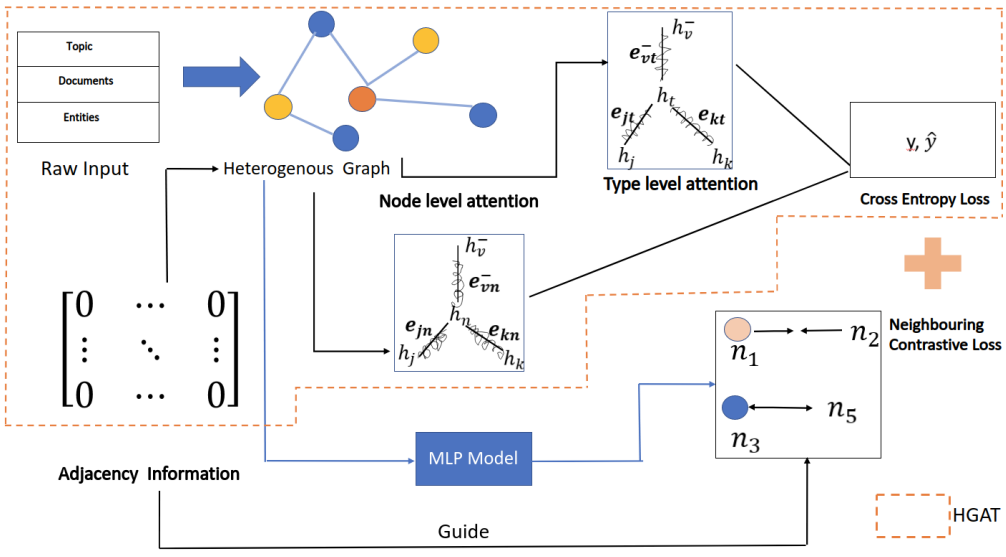


Figure 4.3: Illustration of our NC-HGAT model

It should be noted that there is no edge/relation consideration in the base HGAT model while the syntactic relationships between nodes (such as "subject of ", and "object of ") may provide additional semantic information for understanding the roles of different nodes in text graphs. In this text classification application, three semantic relations for edges connecting with different types of nodes are defined:

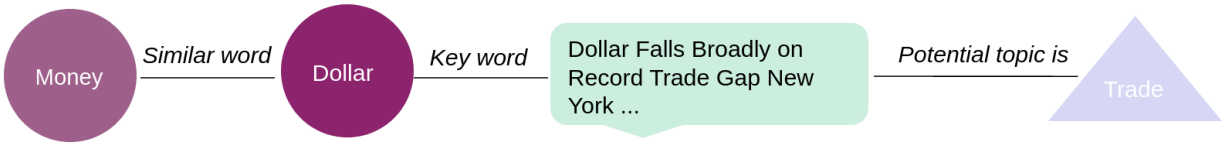


Figure 4.4: Relations between different types of nodes

Relations are '*similar word*' between entities, '*key word*' between documents and entities and '*potential topic is*' between documents and topics.

4.3.2 HGAT

Compared with TextGCN [105], which directly applies GCN to different subgraphs, HGAT introduces a dual attention mechanism: type-level attention and node-level attention, to learn the relative influence of the different types and neighbouring nodes on the target node during information aggregation [56]. The type-level attention a_t is calculated as:

$$a_t = \sigma(\mu_t \cdot [h_i || h_t]) \tag{4.1}$$

Where σ is a LeakyReLU activation, μ_t denotes the attention of the type t of

the node, and operation $||$ is a concatenation. h_i and h_t are specific node and type embedding, respectively. Then, a softmax function is applied, to normalise all types of neighbours of node i as

$$a_t = \frac{exp(a_t)}{\sum_{t' \in T} exp(a_{t'})} \tag{4.2}$$

Where T denotes the set of all node types. The node level attention a_n is formulated based on the type level attention a_t from Equation (4.1):

$$a_n = \sigma(v^T \cdot a_t[h_i||h_{j'}]) \tag{4.3}$$

where v denotes the attention vector, and $h_{j'}$ is the neighbour embedding of node i with type t, is further concatenated with the central node h_i. The type attention weight a_t is obtained from 4.1. The two attention mechanisms are then integrated into the heterogeneous graph convolution, to update the embedding of nodes in the next layer:

$$H^{l+1} = \sigma\left(\sum_{t \in T} a_t \sum_{j \in N_t} a_n^j \hat{A}_t \cdot H_t^l \cdot W_t^l\right) \tag{4.4}$$

Where a_t derived from equation 4.1, signifies the relevance of node type t to the target node. a_n^j caputres the importance of a neighboring node j of tyep t to the target node, with N representing its set of neighbors. For nodes of type t, $\hat{A}$ is the adjacency matrix, H_t^l represent their feature representations in the l-th layer, and W_t^l is the corresponding weight matrix.

4.3.3 Edge-Level Attention

The above section explains the dual attention (node and type level) architecture of baseline HGAT [56]. As previously discussed, the edges among nodes can be categorized into three types of relations: '*potential topic is*' between documents and topics, '*key word*' between documents and entities, and '*similar word*' between entities. These relations can provide a nuanced semantic understanding of the interconnections within the texts, which is investigated in addition to the based published paper here. The initial edge attribute e_{ij}, between node i and j, is assigned as values 1, 2 or 3, corresponding to the relations '*similar word*', '*key word*' and '*potential topic is*', respectively. These values are then processed by a Multi-Layer Perceptron (MLP) to create a richer representation of edge embeddings e_{ij}:

$$\begin{aligned} e'_{ij} &= Fe_{ij} + f, \\ a_e &= \text{softmax}(W \cdot ([h_i || e'_{ij}]) + b), \\ h^{l+1} &= \sum (h_i^l \cdot a_e). \end{aligned} \tag{4.5}$$

Where e'_{ij} is the updated edge features projected from the raw edge features e_{ij}. F, f, W and b are learnable parameters and h_i is the updated node embeddings in equation 4.4. The edge attention addition to my NC-HGAT model is named as neighboring contrastive with heterogeneous graph edge attention network (NC-HGEAT) and the results are presented in Table 4.2 and 4.3

4.3.4 Neighbouring Contrastive Learning

The neighbouring contrastive learning is implemented by calculating the *contrastive loss* for node i. The idea behind it is that neighbouring documents are more likely to have the same class label. The node features X will pass two linear layers with activation σ and layer normalisation LN, and a dropout in between to avoid over-fitting, as in [99]:

$$Z = W^1[Dropout(LN(\sigma(XW^0)))] \tag{4.6}$$

where W^1 and W^0 are the weight matrices of two layers. The number of linear layers could be set differently (from 1-7) as analysed in 4.6. Next, the embedding Z will be used to calculate the neighbouring *contrastive loss*:

$$loss_{NC} = -log\frac{\sum_i \lambda_{i,j} exp(sim(z_i, z_j)/\eta)}{\sum_k exp(sim(z_i, z_k)/\eta)} \tag{4.7}$$

where λ is a connection measure of node j and i and is non-zero only when the node j is within the k-hop neighbourhood of node i; sim is the cosine similarity, and η is the learning 'temperature' parameter. The purpose of this contrastive loss function is to pull the embeddings of related nodes closer (nodes within k-hop) and push non-related nodes apart.

4.3.5 Model Training

Considering the limited labelled data provided, we only use 40 labelled documents per class as training data which is split evenly, with one half used for training and

the remaining half allocated for validation, in line with previous work [56, 57]. All the remaining documents are considered unlabeled for test during model training. We first use the HGAT model to build graphs from the text corpus and learn the representation of nodes with the dual-level attention mechanism. At the same time, we use the MLP-based model to learn more graph structure information, without an explicit message-passing function. To be more specific, the k-hop neighbours are considered more similar to the target node, where this k^{th} power of the neighbouring information is in the range of [1,2,3,4,5,6,7]. If the neighbouring node is not a k-hop of the target node, the neighbours' information is considered zero. Then, we calculate the neighbouring *contrastive loss*, $loss_{NC}$.

$$loss_{total} = loss_{NLL} + \beta * loss_{NC} \tag{4.8}$$

The overall loss function in our model, as shown in equation 4.8, comprises two distinct components: the HGAT negative log-likelihood loss, denoted as $loss_{NLL}$, and the contrastive loss, represented as $loss_{NC}$. To control the trade-off between these components, we introduce a coefficient parameter β. It should be noted that the calculation of contrastive loss $loss_{NC}$ and the original loss$loss_{NLL}$ is separate. Specifically, the updated node features Z obtained from equation 4.6, are exclusively utilised in the computation of the contrastive loss and do not impact the original HGAT loss. The gradient descent algorithm is applied to optimise this total loss.

4.4 Experiments

4.4.1 Datasets

We use the same four benchmark short text datasets as in [56], for a fair comparison to prior work. All datasets have been preprocessed by eliminating non-English characters, stop words and words that appear fewer than five times. The movie review dataset (MR) [69] has 5,331 positive and 5,331 negative reviews, each of which is one sentence. Twitter, a sentiment classification dataset provided by the NLTK library of Python, contains 5,000 positive and negative tweets, respectively. Ohsumed is a bibliographic database provided by [55] where a graph convolution network model is applied for text classification. Each of 7,000 documents is labelled with 23 types of diseases. We use 3,357 documents in the training set and the remaining documents in the test set. AGNews randomly selected 6,000 news items from [68], which are classified into four classes: world, sports, business and sci/tech.

Table 4.1: Dataset summary

Datasets	Docs	Token	Entities	Classes
MR	10662	7.6	1.8(76%)	2
Twitter	10000	3.5	1.1(63%)	2
AGNews	6000	18.4	0.9(72%)	4
Ohsumed	7400	6.8	3.1(96%)	23

4.4.2 Experimental Setup

We implement our model with Pytorch 1.10.2 and CUDA 10.2. The hyper-parameters of NC-HGAT are mainly borrowed from the experiments of HGAT [56] and Graph-MLP [99]. 40 labelled documents per class are randomly selected and split equally into training and validation sets. We use two layers and the number of hidden units is 512, learning rate 0.005, with an 80% dropout rate at each layer. The dimension of pre-trained word embeddings is set to 100. The k^{th} power of the adjacency matrix, the temperature parameter η and the coefficient balance parameter β are set using grid search. The range of η and β are [0,1,2] and [0.5, 1.0, 2.0, 3.0], respectively.

4.4.3 Baselines

We consider three widely applied NLP models and three graph neural network models, applied as baselines for text classification. The parameter settings of all baseline models are the same as in [56, 57].

- SVM + TFIDF and SVM + LDA are conventional machine learning classifiers, using classic features, including TF-IDF and LDA features [102, 130].
- BERT, deploying a bidirectional Transformer encoder [131], is a widely-applied model in NLP. Due to the design of BERT's tokenizer for raw corpus, datasets are not pre-processed like other baselines and the model (Bert-base) has been fine-tuned as in [57].
- TextGCN is the first study that applies GCN to text by building heterogeneous graphs from a text corpus [55].

- HAN considers the importance of both node and meta-path, by introducing an attention mechanism into the heterogeneous graph neural network [105].
- HGAT integrates a dual attention mechanism into a heterogeneous information network [56, 57], representing the current state-of-the-art on short text classification tasks.

4.5 Results and Discussion

Table 4.2 and Table 4.3 show the average classification performance of different models on the four benchmark datasets. The proposed model NC-HGAT outperforms all baselines on three datasets, demonstrating the effectiveness of the neighbouring contrastive learning with the heterogeneous graph attention network on short text classification. The improvement made by our proposed model with respect to the Ohsumed dataset in terms of the F1 score is statistically significant, based on the t-test ($t = -9.738$ and p-value ≤ 0.001). With edge attention consideration, the NC-HGEAT model demonstrates superior performance, compared to all baseline models, as well as our prior model, the NC-HGAT, except for a minor decrement in the F1 score on the Ohusmed dataset. This highest accuracy and F1 score performance indicate that, with edge consideration, the model can capture essential patterns and relationships within the data, more effectively.

Table 4.2: Model Accuracy on Datasets

Dataset	**AGNews**	**MR**	**Ohusmed**	**Twitter**
SVM+TFIDF	59.45%	54.29%	39.02%	53.69%
SVM+LDA	65.16%	54.40%	38.61%	54.34%
Bert	69.45%	53.48%	21.76%	52.00%
Text-GCN	67.61%	59.12%	41.56%	60.15%
HAN	62.64%	57.11%	36.97%	53.75%
HGAT	72.10%	62.75%	42.68%	63.21%
NC-HGAT	73.15%	62.46%	43.27%	63.76%
NC-HGEAT	**73.50%**	**63.32%**	**43.33%**	**64.24%**

Table 4.3: Model F1-score on Datasets

Dataset	**AGNews**	**MR**	**Ohusmed**	**Twitter**
SVM+TFIDF	59.79%	48.13%	24.78%	52.45%
SVM+LDA	64.79%	48.39%	25.03%	53.97%
Bert	69.31%	46.99%	4.81%	43.34%
Text-GCN	67.12%	58.98%	27.43%	59.82%
HAN	61.23%	56.46%	26.88%	53.09%
HGAT	71.61%	62.36%	24.82%	62.48%
NC-HGAT	72.06%	62.14%	27.98%	62.94%
NC-HGEAT	**72.40%**	**63.32%**	27.67%	**63.99%**

However, when focusing on the Ohsumed dataset, where the accuracy is 43.3%, it is essential to consider the dataset's inherent challenges. A closer examination

reveals that Ohsumed has the most classes from our four considered datasets (see Fig. 4.5), where Y-axis denotes the label of each document and X-axis represents the number of documents classified with that label. It is clear that the Ohsumed dataset is not uniformly distributed. It is also not a uniform distribution, while the classes are equally distributed in the MR, Twitter and AGNews datasets. Given the inherent complexities of the Ohsumed dataset, achieving a 43.3% accuracy demonstrates the resilience of the NC-HGEAT model, particularly with only 20 labels per class available. Moreover, the introduction of edge consideration in NC-HGEAT, compared to NC-HGAT, underscores its enhanced capabilities. This also shows that contrastive learning can enhance the ability to learn the structural node distribution and thus improve the classification accuracy, particularly when there are multi-classes.

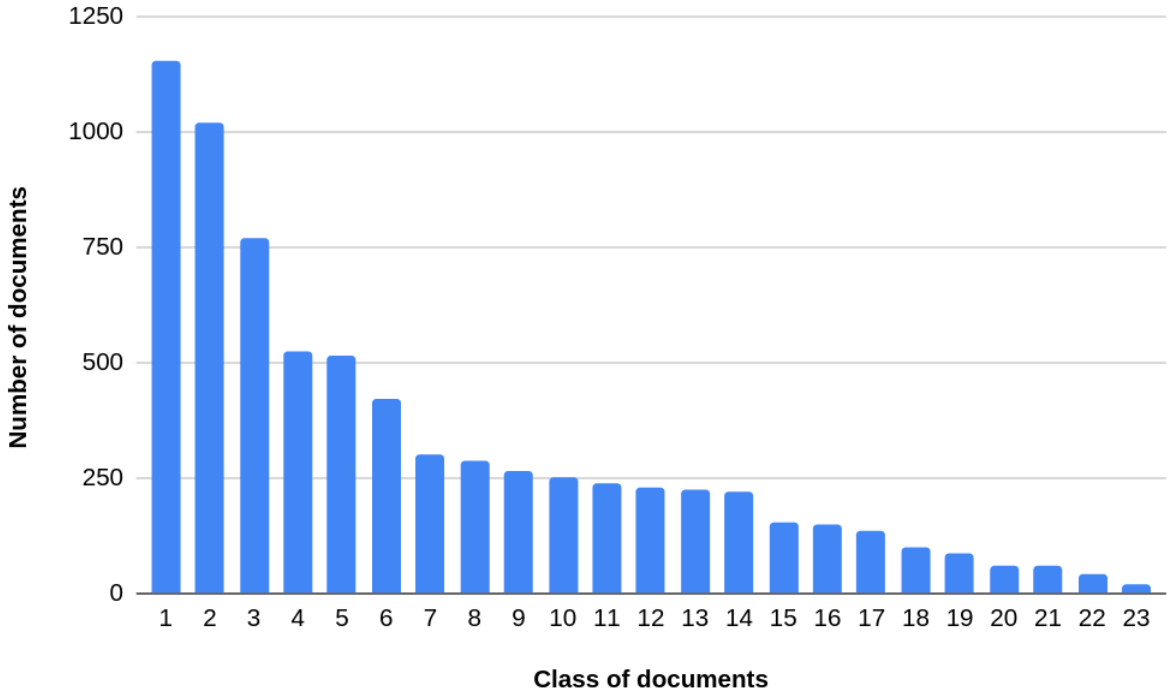


Figure 4.5: Label distribution of the Ohsumed Dataset

The minor under-performance of NC-HGAT on the MR dataset may be due to the fact that it captures more background information or stop-words, which are unrelated to a specific class, thus diminishing the result. Another reason suggested by [55], who also found the under-performance of TextGCN model on the MR dataset, is that edges in the constructed text graphs of MR are fewer than other datasets, as the documents are very short and thus limit the embedding learning ability.

4.6 Impact of Layer Numbers of MLP

To investigate the impact of the MLP layer number deployed in section 4.3.4, we evaluate our NC-HGAT model with 1-7 layers, inspired by [132], on the Twitter (very short sentence with fewer tokens seen in the Table 4.1) and AGNews (relatively long sentence with more tokens) datasets as examples. As shown in Table 4.4, the model with two layers performs better on the AGNews dataset; for the Twitter dataset, six layers perform the best. As for the AGNews dataset, the vanishing gradient and over-processed information will lead to an unstable model if the number of layers is excessive. The node representations may also become indistinguishable, known as the oversmoothing problem [133]. For the Twitter dataset, however, distant words may still be able to classify the document, and six layers can capture sufficient structural information.

Table 4.4: Model Performance with Different Layers on Twitter and AGNews Dataset

	Twitter		AGNews	
Number of Layers	Accuracy	F1	Accuracy	F1
1	63.04%	62.99%	73.00%	71.72%
2	61.86%	61.22%	**73.15%**	**72.06%**
3	61.05%	60.93%	72.50%	71.81%
4	63.66%	62.50%	72.85%	71.61%
5	63.28%	62.63%	72.50%	71.03%
6	**63.76%**	**62.92%**	72.60%	71.16%
7	62.79%	62.28%	72.30%	71.45%

4.7 Robustness of Neighbouring Contrastive Learning

The outperformed experimental results above also show that contrastive learning can enhance the robustness of graph representation learning when there are limited labelled data. Consequently, the effectiveness of the neighbouring contrastive learning method is investigated and compared with the proposed BGAT model, which incorporates a parametric random graph model to account for uncertainty in the graph structure. This comparison is presented in Chapter 3. Because nodes in these datasets are not heterogeneous (all nodes refer to academic papers), the GAT model is implemented as a replacement for the HGAT model, with the neighbouring

contrastive learning techniques. The results compared with BGAT in chapter 3 are summarised in Table 4.5

Table 4.5: Average Prediction Accuracy of Models in Datasets

	Cora			CiteSeer			Pubmed		
Model	5 labels	10 labels	20 labels	5 labels	10 labels	20 labels	5 labels	10 labels	20 labels
ChebyNet	61.7±6.8	72.5±3.4	78.8±1.6	58.5±4.8	65.8±2.8	67.5±1.9	62.7±6.9	68.6±5.0	74.3±3.0
GCN	70.0±3.7	76.0±2.2	79.8±1.8	58.5±4.7	65.4±2.6	67.8±2.3	69.7±4.5	73.9 ± 3.4	**77.5 ± 2.5**
GAT	70.4±3.7	76.6±2.8	79.9±1.8	56.7±5.1	64.1±3.3	67.6±2.3	68.0±4.8	72.6±3.6	76.4±3.0
BGCN	74.6±2.8	77.5±2.6	80.2±1.5	63.0±4.8	69.9±2.3	71.1±1.8	70.2±4.5	73.3±3.1	76.0±2.6
BGAT	74.8 ± 4.5	78.8 ± 2.8	**84.3 ± 1.8**	**68.6 ± 4.6**	**71.4 ± 2.6**	**74.2 ± 1.6**	71.4 ± 4.7	72.3±3.4	74.5±2.4
GAT-NC	67.5±0.1	74.7±0.6	76.5±0.3	63.2±0.2	64.7±0.6	71.1±0.2	**72.2 ± 0.002**	**74.8 ± 0.01**	76.6±0.003

For text classification, especially with extremely imbalanced datasets, like Ohsumed, relying solely on accuracy can be misleading. Thus we incorporate the F1 score, as an additional metric. For citation networks, on Cora, CiteSeer and Pubmed, where class distributions are more balanced, accuracy alone provides a clear measure of model effectiveness.

Compared with the vanilla GAT, GAT with neighbour contrastive learning performs better in Pubmed datasets, but with lower accuracy in the Cora and CiteSeer dataset. A plausible reason for this is the nature of the features: Pubmed employs TF-IDF features, which capture richer contextual information, allowing nodes to have more distinctive representations. In contrast, Cora and CiteSeer use binary features, indicating the absence or presence of a word from a dictionary, as discussed in the previous chapter. This binary representation makes it challenging for nodes to have unique information, which diminishes the effectiveness of neighbouring contrastive learning, which works by distinguishing nodes from their neighbours.

In all three datasets, GAT-NC shows a much smaller standard deviation compared with all other models. It achieves the highest accuracy in Pubmed when only 5 and 10 labels are available per class, which proves the robustness of this neighbouring contrastive learning method.

4.8 Potential Future Work

Several potential extensions to our work could be addressed in future studies:

1. First is to investigate how to augment both feature and topology levels to improve the performance of graph learning models. For example, the links among documents could be directed; thus, hypergraphs which allow one edge to link to more than two vertices can be applied to capture more complex group-wise relationships.

2. Another research direction is to investigate how to extend graph learning methods with pretrained large language models (LLMs) such as Bidirectional Encoder Representations from Transformers (BERT) [131], Generative Pre-trained Transformer (GPT) [134], Text-to-Text Transfer Transformer (T5) [135], Pathways Language Model (PaLM) [136], Large Langeuage Model Meta AI (LLaMA) [137] and many other advanced models. Existing studies [138, 139] demonstrated that pretrained models capture certain types of semantic and syntactic relationships in language. Building on this, [140] present an efficient method to learn on text-attributed graphs (TAGs), by alternating updates between LLMs and GNNs, which indicates that LLMs have the po-

tential to serve as a valuable adjunct to our graph learning strategies within NLP.

4.9 Summary

In this chapter, we propose, for the first time, to use contrastive learning to capture the topological information with HGAT on short text classification tasks. Extensive experiments on different datasets with different numbers of target classes illustrate that neighbour contrastive learning studies the contrast between neighbouring data points. It can effectively distinguish between different types of entities, thereby improving the accuracy of classification results, particularly when there are limited labelled data. This chapter investigates the heterogeneity problem by leveraging heterogeneous graph neural networks and contrastive learning to capture rich semantic relationships and contextual information with textual data. Additionally, the study demonstrates that contrastive learning enhances the robustness of models, as evidenced by a comparison with the proposed BGAT method discussed in Chapter 3. In the next chapter, we will illustrate how to capture higher-order relationships by answering the third research question for graphs with more complex relations.

CHAPTER 5

Hypergraph learning for complex relations

In this chapter, we propose a novel ensemble learning framework, shortened as 'MONEY', to capture the complex group-level and pairwise relations, answering the third research question regarding relation extraction in Chapter 1.2:

How to improve the structural learning ability, e.g. relation extraction and positional encoding for existing graph representation learning?

We apply the proposed model for stock price prediction tasks and demonstrate the effectiveness of the model as it outperforms the state-of-the-art in extensive experiments. We also prove that for such a multivariate time series prediction task, a sequence model like LSTM should be applied after considering the influence of other relevant factors (in this case, industry and shareholder). Otherwise, the performance will be impaired by first assuming historical information is linear or stationary when

using RNN models.

This Chapter is based on the following manuscript, which has been accepted as listed in Chapter 1.5:

***Sun, Z.**, Harit, A., Cristea, A. I., Yu, J., Lei, Shi., Al Moubayed, N. MONEY: A Novel enseMble Learning: cOnvolutional Network with advErsarial hYpergraph Model for Stock Price Movement Prediction. AI Open Journal*

5.1 Introduction

Stock prediction has been a crucial research topic for a long time, and investors are always interested in having a higher predictive accuracy model to gain profit. Whilst it is notoriously difficult to predict stock price movements when extensive uncertainty factors, such as policies or social conditions, like pandemics, can influence the financial market [141], evidence shows that certain factors, like returns of industry portfolios, can forecast the entire stock market by up to two months [142]. Moreover, stocks in a similar industry will behave differently from the ones outside that industry [143]. These are thus relevant relational factors necessary to integrate in a comprehensive manner. Graph neural networks (GNNs) are a powerful technology leveraging the advantage of network structure [144], thus promisingly suitable for such a task. Indeed, some initial solutions have been proposed [59], but are limited. Existing GNNs usually model pairwise relations of stocks and other information as simple graphs to predict if the price will rise or fall on the next trading day. However, stock prices are more likely to co-move for similar companies and for such complex behaviour [145], more advanced models are required to capture it.

Therefore, we propose to build hypergraphs[1], which allow modelling group-level relations among stocks from industry and fund-holding (mutual fund as shareholders) aspects [59].

Despite the success of GNNs on simple graphs, the study of deep learning on hypergraphs is still at an early stage, and most existing GNNs cannot exploit the high-order structure encoded by hyperedges [147]. Few studies explored the area: [148] proposed a gated temporal convolution over hypergraphs to capture stock trends and [59] applied hypergraph attention networks to predict stock price movement and validated the effectiveness of using hypergraphs to capture similar patterns of stocks in the same group. However, they mainly focused on the group-level analysis and neglected the pairwise correlations between two similar companies, which exist in financial markets [149].

For instance, stocks in the same industry[2] display similar volatility patterns [145], which is illustrated in Figure 5.1, with stocks from Ningbo Bank (NB), China Merchants Bank (CMB) and Bank of China (BOC). The correlation coefficient between two stocks can be calculated as:

$$R_{xy} = \frac{\sum_{k=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_i - \bar{x})^2 \sum_{k=1}^{n}(y_i - \bar{y})^2}} \tag{5.1}$$

The correlation coefficient R_{xy} between stock prices of NB (x) and CMB (y) is 0.72, while the correlation coefficient between NB and BOC is 0.64. The strong correlation between these two pairs can be partially attributed to them being held

[1]The hypergraph is a special graph, where hyperedges can connect multiple vertices at the same time [146], and the effectiveness of convolution on hypergraphs has been demonstrated [147].

[2]https://uqer.datayes.com/

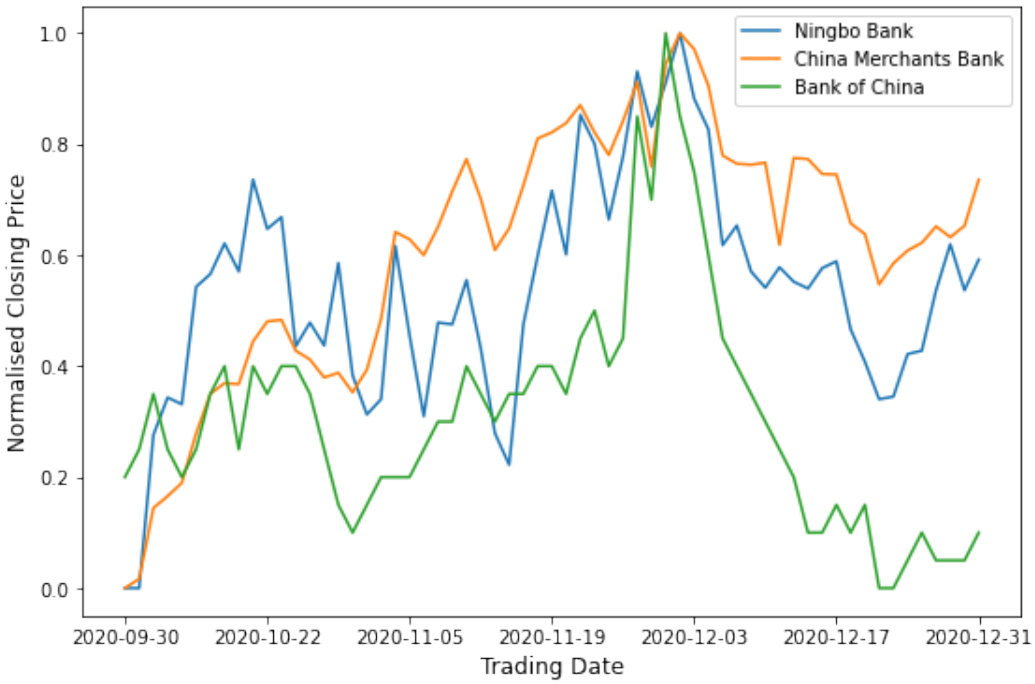


Figure 5.1: Price Movement of Three Stocks

by the same mutual fund between 30/09/2020 and 31/12/2020. Thus, we expect other stocks in the same industry and held by the same fund to also display pairwise relations.

Unlike prior works [36, 59, 148, 150], we consider not only price information but also industry information via a graph convolution network (GCN) to augment the pairwise behaviour of similar companies, before using recurrent neural networks (RNNs). Different from other studies using RNNs and GCN with historical price features [151], we apply GCN first so that the long-term dependencies of pairwise companies can be captured by RNNs later and avoid assuming price movement to be linear or stationary, as [145] suggested.

We do not process fund-holding information via the GCN at this stage, as it is included in the later hypergraph convolution. Moreover, compared to the industry information, fund-holding information is less influential for single stock due to

the diversification investment strategy, which requires mutual funds to hold widely spread portfolios across different types of securities [152], but can be a good signal of market trend. To sum up, our method, MONEY, can consider both pairwise and group-level relations between companies in the same industry and held by the same fund, capturing dynamic historical price information.

Moreover, as the financial market is volatile and the price movement is stochastic, standard training may cause overfitting. Therefore, the resulting (sub-) research questions are:

1. Can stock price movement prediction be improved by capturing complex relational industry information?

2. How to enhance the robustness of model while not compromising its predictive performance?

As explained in the introduction and addressing the research questions above, we utilise hypergraph convolution, to adeptly capture group-level interactions among stocks. Complementing this, adversarial training is employed to enhance the model's robustness, by considering perturbed features. This results in an adversarial prediction whose associated adversarial loss is incorporated into the total loss calculation, fortifying the model against market volatility. Additionally, we use the voting ensemble learning method to classify the predicted classes yielded from standard training and adversarial training, together with the highest number of votes. The contributions of the chapter are summarised as follows:

- This is a novel ensemble learning framework that firstly applies both hypergraph convolution and simple graph learning to capture complex relations

among similar stocks and address the stochasticity of stocks by adversarial training, which can be used as a solid baseline for future research.

- To the best of our knowledge, *this is the first attempt* to demonstrate the effectiveness of integrating auxiliary information (e.g. industry) via GCN before using RNNs so that the long-term dependencies of pairwise relations of similar companies can be learnt by RNNs later, unlike prior approaches, which deploy RNNs first.

- Experimental results on a real-world dataset prove that our proposed model, MONEY, significantly outperforms the state-of-the-art on stock price movement prediction for most indicators (accuracy, precision, recall and F1 score) and has more stable performance.

5.2 Related Work

5.2.1 GNN for Price Prediction

Researchers have attempted to apply GNNs to learn stock representations by modelling stock relations as graphs. For instance, [153] combined a Long Short-Term Memory (LSTM) with GCN to learn shareholding information. Later, [36] applied a multi-graph convolution network to predict stock movement, treating the embedding of shareholding, industry and news relation graphs equally in the prediction.

Recently, researchers also proposed to construct different types of graphs to model the relationships between stock prices and other relevant information. Xiong et al. [154] built a heterogeneous graph for stock prediction by aggregating event

level and contextual information. Cui et al. [59] deployed a hypergraph convolutional network to represent the impact of industry and fund-holding on stock movements, showcasing the ability of hypergraphs to capture complex group-level relationships among stocks. It is noteworthy to mention that the prowess of hypergraph neural networks is not limited to the finance sector. Recent literature, like [155–158], indicated their success in recommendation and learning analytics domains. For readers interested in a broader view of hypergraph modeling application across diverse fields, these studies provide valuable insights.

Despite the success of the aforementioned financial prediction methods, most of them first fed only historical price features of stocks into RNN models to obtain new price embeddings. Then, the updated embedding could be processed with the other factors, such as similar corporations [153], news [154], industry [159] or shareholding information [36, 150], by using GCN or other graph neural networks. However, this post-processing came too late. This chapter proves that incorporating auxiliary information through GNNs before using RNNs in multivariate time series analysis enhances the understanding of long-term dependencies between similar companies. Additionally, existing hypergraph models overly emphasize group-level relations, neglecting the equally important pairwise interactions among stocks. Moreover, none of these studies dealt appropriately with overfitting (here, due to continuous market fluctuations).

5.2.2 Adversarial Training

Arguably the most effective method to enhance the generalisation of models, by 'defending' against perturbed examples, it received considerable recent attention [160]. The main purpose of adversarial training is to augment clean data with adversarial examples, so that models can still deliver consistent results when facing adversarial attacks. For stock movement prediction, their stochastic and dynamic characteristics require dealing with overfitting. Addressing the problem, [161] applied adversarial training to financial time-series analytics and validated the effectiveness of simulating the stochasticity of stock features. Recently, [162] proposed a sentiment-guided generative adversarial network, to explore the stock prediction problem. Li et al. [163] then combined adversarial training with transfer learning and obtained competitive results.

5.2.3 Ensemble Learning for Price Prediction

Ensemble learning is a machine-learning technique for improving classification or regression performance by considering multiple algorithms [164]. It has achieved great success in many applications and has been increasingly integrated with deep neural networks (ensemble deep learning) [165]. A few studies have applied ensemble deep learning to price forecasting and showed that it outperformed the single models on financial time series [166]. For instance, [167] proposed a denoising autoencoders (SDAE) method with bootstrap aggregation (bagging) to model complex relationships of oil prices with its factors. Li and Pan [168] utilised a blending ensemble learning method, consisting of two RNNs, to predict the S&P 500 Index. Jiang et

al. [166] incorporated four types of tree-based ensemble algorithms: random forest, extremely randomised trees, XGBoost and LightGBM, with four types of RNNs: vanilla RNN, Bidirectional RNN, LSTM and gated recurrent unit (GRU), into a stacking framework for stock index prediction. Different from previous studies, we incorporate GNN to model the relation of the stock price with industry and fund-holding factors and consider the stochasticity by introducing adversarial training into a max-voting ensemble learning framework.

5.3 Methods

5.3.1 Problem Formulation

The goal of this research is to predict the movement direction of the stock for the following trading day. Given different lengths (5, 10, 20 trading days [59]) of past daily transaction X_s and industry features I_s, as well as fund-holding information F_s, we consider three movement directions of stock prices compared with the price P on the previous trading day $t-1$: *rise, steady, fall*. In line with previous work [59], rise (1) means the closing price of a stock on the next trading day P_t is over 0.55% higher than the closing price before P_{t-1}. If P_t is more than 0.50% lower than P_{t-1}, then the price movement is considered as fall (-1); otherwise, steady (0). These 0.55% and 0.50% settings consider the transaction costs such as tax and also balance different types of samples. The cross-entropy loss function of stock price movement prediction can be defined as:

$$loss_{hinge} = -\sum_{c=0}^{C} y_{o,c} log(p_{o,c}) \tag{5.2}$$

where C is the number of classes, which, in our work, is 3 (rise, steady, fall); y is the binary indicator, showing if the prediction of observation o is correctly classified, and p is the predicted probability of observation o as c class. If we consider adversarial training, then the total loss becomes:

$$loss_{total} = -\sum_{c=0}^{C} y_{o,c} log(p_{o,c}) - \beta \sum_{c=0}^{C} y_{o,c} log(p_{a,c}) \tag{5.3}$$

Where the addition term is the adversarial loss and $p_{a,c}$ is the predicted probability of a perturbed observation o classified correctly as c class; β is a hyperparameter balancing the two types of loss, so that the model is encouraged to classify both original objects and perturbed samples, correctly. In essence, this adversarial training aims to make the model resilient to small perturbations, ensuring that such minor changes do not alter the predicted class label, thus enhancing its robustness against adversarial attacks and potential noise in the input data.

The overall structure of our proposed framework is shown in Figure 5.2. It consists of two models;

model A: industry information first passes stock price (a) through a GCN (b); then a GRU and temporal attention layer (c) are applied to capture the time-series characteristics of stocks, followed by a hypergraph construct (d); then separate convolution networks (e) for industry and fund-holding information; and, finally, a linear classification neural network (f);

model B: As we have considered the pairwise industry information via GCN in model A, model B starts with a GRU, to capture price-time dependency, following existing research, and without the GCN (b), to process industry information

at the beginning; it includes instead an extra input of adversarial samples to the classification layer (g).

The predictions yielded by Models A and B, which comprise modules *a,b,c,d,e,f* and *a,c,d,e,f,g*, respectively, will be integrated into a new embedding, denoted as $\hat{y}_s$. Model B is also designed to handle adversarial examples, producing an adversarial prediction, represented by $\hat{y}_{adv}$. $\hat{y}_f$ is the final prediction, derived by voting between the concatenated prediction $\hat{y}_s$ and adversarial prediction $\hat{y}_{adv}$. During loss computation (equation 5.3), $\hat{y}_{adv}$ is weighted by β, to enhance model robustness against adversarial attacks.

The differences between our MONEY model and the state of the art, HGTAN [59] are threefold: 1) we use GCN first to augment the pairwise industry relation before using RNN to capture the long-term dependency of price and demonstrating the effectiveness of using GCN first; 2) we acknowledge the inherent random movements of stock market and employ adversarial training to enhances our models' generalisation, by simulating potential market fluctuations through input perturbations; 3) we do not use triple attention mechanisms among hyperedges and hypergraphs, but still outperform the HGTAN as shown in Table 5.2. Details are explained in the following subsections, and we will explain how to construct hypergraphs first.

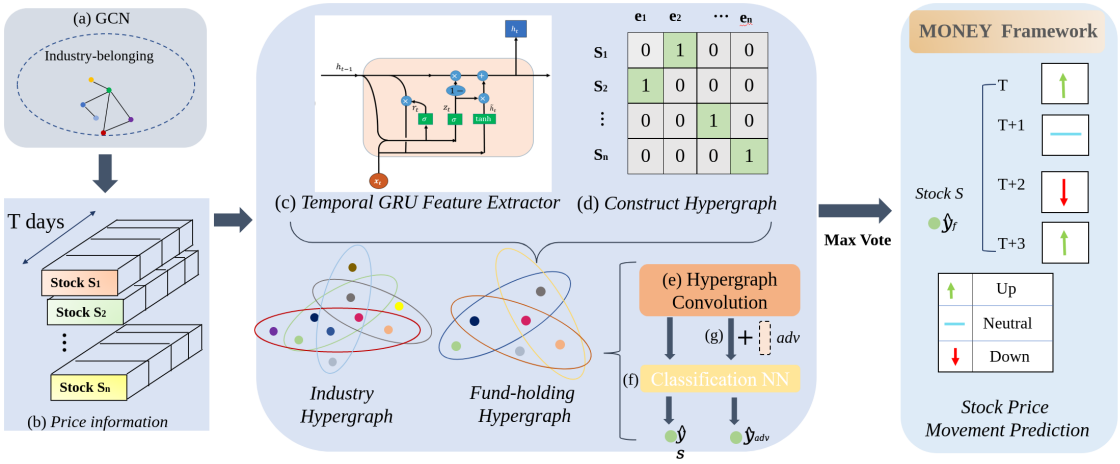


Figure 5.2: Illustration of our MONEY model

5.3.2 Hypergraphs Construction

Definition 1. Hypergraph An undirected hypergraph can be denoted as $G = (V, E)$ where V is the set of N nodes and E is the set of M hyperedges. In a hypergraph, any of the edges e can join any number of vertices v to describe more complex relationships between entities, shown as (b) in Figure 5.3; while in simple graphs, one edge can only link two vertices, representing a pairwise relation between nodes as (a) in Figure 5.3. A hypergraph is often denoted as an incidence matrix $H \in R^{N \times M}$:

$$H(i,j) = \begin{cases} 1 \text{ if node } i \text{ is included in hyperedge } j \\ 0 \text{ otherwise} \end{cases}$$

Same as [59], we construct two hypergraphs to model the industry and fund-holding relations among stocks separately. In the industry hypergraph, all the com-

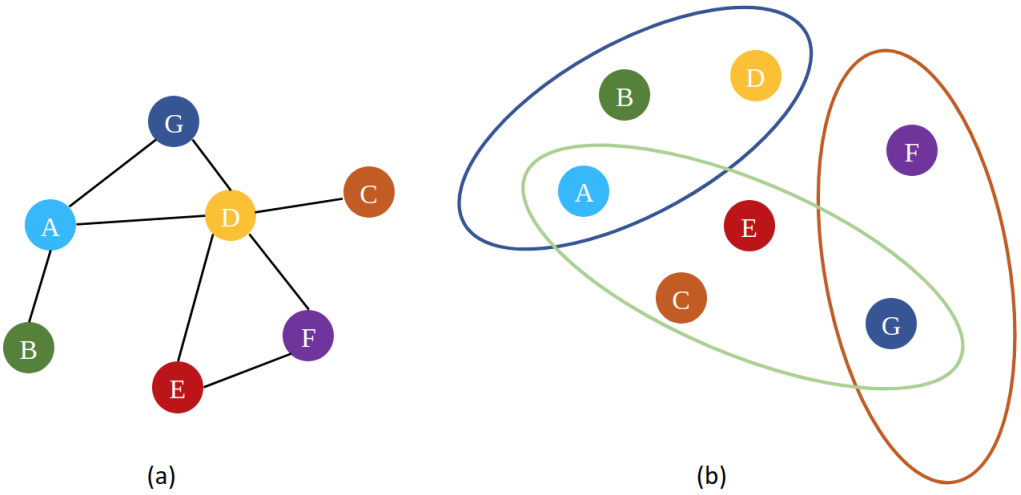


Figure 5.3: Simple Graph and Hypergraph

panies in the same industry are connected by one hyperedge and each node $v \in V$ has features with ten dimensions; this is similar to the fund-holding hypergraph. We apply graph convolutions to the two hypergraphs to update embeddings of each stock (node) to be later used in the price movement classification.

5.3.3 GCN for Industry Information

For each trading day t, we input the price attributes $x_{s,t}$ of stock s and its associated industry information I_s into a GCN model $f(\theta)$. This helps us capture the pairwise patterns among similar companies. Consequently, this process yields an updated industry-specific embedding for the stock, denoted as $x^t_{s,i}$:

$$x^t_{s,i} = f(\theta)(x^t_s, I_s) \tag{5.4}$$

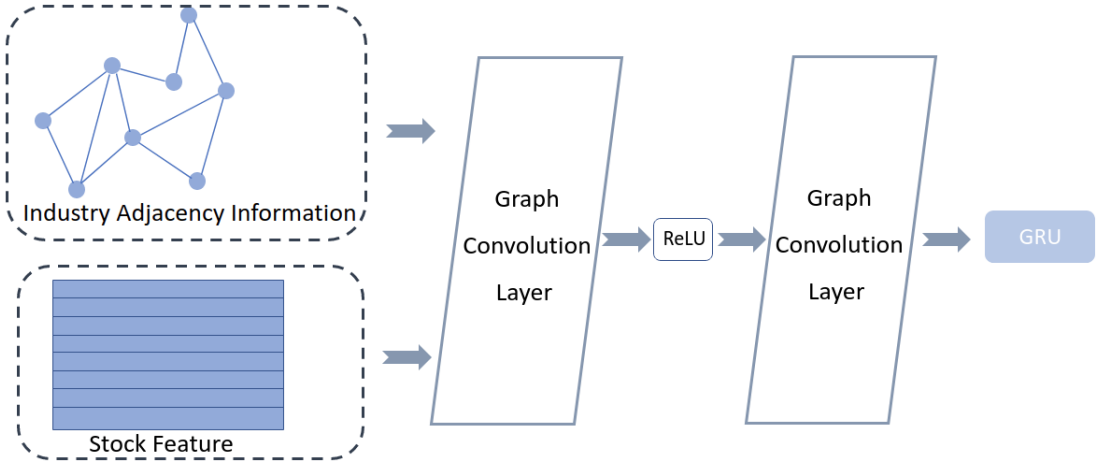


Figure 5.4: Overview of GCN

We build undirected edges between stocks that are in the same industry and use two layers of convolution, as shown in Figure 5.4. We did not use GCN to process fund-holding information, which is considered at a later stage. Moreover, compared to the industry information, fund-holding information is less influential for single stock due to the diversification investment strategy, which requires mutual funds to hold widely spread portfolios across different types of securities [152], but can be a good signal of market trend. After the GCN layer, the new embedding x_i^t will be passed to GRU to capture the time-series information.

5.3.4 Gated Recurrent Unit for Long Term Dependency

Following [59], we use a gated recurrent unit neural network (GRU) model [169] to learn the embedding of stock features due to its ability to capture long-term dependency for sequential data. Note that the input stock feature has been augmented by considering the influence of industry. The main purpose of GRU is to deploy a

gated process, to manage and update the flow of information between cells of neural network units, which can be formulated as:

$$
\begin{aligned}
z_{s,i}^{t} &= \sigma(W_z \cdot x_{s,i}^{t} + U_z \cdot h_{s,i}^{t-1} + b_z), \\
r_{s,i}^{t} &= \sigma(W_r \cdot x_{s,i}^{t} + U_r \cdot h_{s,i}^{t-1} + b_r), \\
\widehat{h}_{s,i}^{t} &= tanh(W_h \cdot x_{s,i}^{t} + r_{s,i}^{t} * U_h \cdot h_{s,i}^{t-1} + b_z), \\
h_{s,i}^{t} &= z_{s,i}^{t} * h_{s,i}^{t-1} + (1 - z_{s,i}^{t}) * \widehat{h}_{s,i}^{t}.
\end{aligned}
\tag{5.5}
$$

where $x_{s,i}^{t}$ is obtained from equation (5.4) and W_z, W_r, W_h are trainable weight matrices. $h_{s,i}^{t-1}$ is the hidden state, which includes historical information from the previous trading day for stock s in industry i. The matrices U_z,U_r, and U_h are also trainable parameters. On trading day t for stock s in i industry:

- $z_{s,i}^{t}$ serves as the reset gate to decide how much historical information should be disregarded.
- $r_{s,i}^{t}$ is the update gate to decide how much of the past information should be forwarded for future considerations. The update and reset gates could alleviate the vanishing gradient problem during the backpropagation of time-series data.
- $\widehat{h}_{s,i}^{t}$ represents the current information intended for the hidden state $h_{s,i}^{t}$. This updated embedding is subsequently directed to the temporal attention layer.

5.3.5 Temporal Attention

The attention mechanism has been increasingly applied to predict stock price trends [59, 170–173]. Attention can capture the different influences of the hidden represen-

tations on the overall learned embedding at different time steps [161]. Due to the recency bias hypothesis [174], stating that the most recent price has a stronger correlation with the future movement, it is reasonable to use a temporal attention layer to yield the aggregated temporal dynamics representations of stocks s_d in industry i:

$$
\begin{aligned}
s_{d,i} &= \sum_{t=1}^{T} \alpha_{s,i}^{t} h_{s,i}^{t} \\
\alpha_{s,i}^{t} &= \frac{exp^{\widehat{\alpha_{s,i}^{t}}}}{\sum_{t=1}^{T} exp^{\widehat{\alpha_{s,i}^{t}}}} \\
\widehat{\alpha_{s,i}^{t}} &= tanh(W_{\alpha} h_{s,i}^{t} + b_{\alpha}) U_{\alpha}^{T}
\end{aligned}
\tag{5.6}
$$

where $\alpha_{s,i}^{t}$ is the weight of the hidden state at time t of stock s. $W_{\alpha} h_{s}^{t}$, b_{α} and U_{α}^{T} are parameters need to be learned. The aggregated temporal representation $s_{d,i}$ will then feed into the hypergraph convolution layer, explained in section 5.3.6.

5.3.6 Hypergraph Convolution with Adversarial Training

To model the group-level relationships among stocks, we apply hypergraph convolution to both the industry graph G_i and the fund-holding hypergraph G_f. For instance, in the industry graph G_i, each stock connects via the same edge e_i and will aggregate messages passed from its neighbours. One step of convolution on the hypergraph can be formulated as:

$$
h_{s}^{l+1} = \sigma(\sum_{j=1}^{M} \sum_{i=1}^{N} h^{l} F) \tag{5.7}
$$

where h^{l+1} is the feature matrix of stock s at the next layer and σ is a nonlinear activation LeakyReLU. The terms $\sum_{j=1}^{M} and \sum_{i=1}^{N}$ iterate over all hyperedges and nodes, respectively. Within each hyperedge j, each node updates its features using local neighbors, and this process is repeated for every node starting from i. F is a weight matrix to be learnt between the l and $l+1$ layers. After the convolution, we have updated the industry embedding h_i and the fund-holding embedding h_f, which will be concatenated, to obtain a new embedding h_s^m for the prediction layer and further adversarial training in equation 5.8.

Adversarial Training is to augment each minibatch of clean data with adversarial examples (AEs), which are generated by adding adversarial perturbation [175]. Feng et al. [161], validated the effectiveness of adversarial training to address the stochastic property of stock price. Therefore, we add adversarial perturbation to stock embedding at a higher level after the temporal attention layer before the prediction layer.

$$h_s^{adv} = h_s + p_s^{adv},\ p_s^{adv} = argmax_{p_s, \|p_s\| \leq \varepsilon}\ loss_{adv}(y, y^{adv})) \tag{5.8}$$

$$p_s^{adv} = \epsilon \frac{G^s}{||G^s||_2}, G^s = \frac{\partial l(y, y^{adv})}{\partial h_s} \tag{5.9}$$

where h_{adv}^s denotes the latent representation of an adversarial example and h^s is the concatenated embedding obtained from the hypergraph convolution above. p_s^{adv} is the adversarial perturbation that maximises the loss, but with a constraint on its magnitude to be less than or equal to a small value ε. This perturbation is

updated by the fast gradient approximation method [176] in equation 5.9, where G is the gradient of the adversarial loss with L2 norm constraint of stock s and ϵ is to change the scale of perturbation. y represents the true labels and y^{adv} denotes the adversarial predictions. Feng et al. [161] empirically demonstrated that adversarial training enforced the decision boundary to be close to original objects so that the model is able to capture stochasticity and classify the perturbed samples correctly.

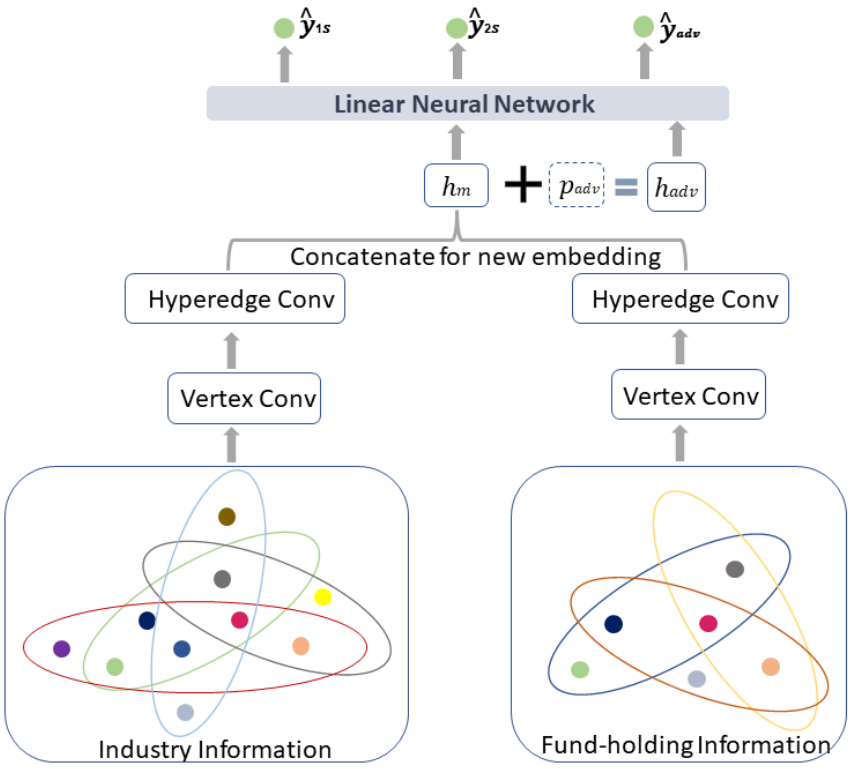


Figure 5.5: Hypergraph Convolution with Adversarial Training

Figure 5.5 describes the process of using a hypergraph convolution network for the industry and fund-holding hypergraphs and the h_s and its adversarial example h^s_{adv} will be fed into a linear prediction layer for classification. $\hat{y_{1s}}$ and $\hat{y_{2s}}$ is the prediction generated by the two base models, also illustrated in Figure 5.2 which will be explained in section 5.3.7.

5.3.7 Ensemble Learning

Various ensemble strategies can be utilised, including bagging, boosting, stacking, blending, averaging, weighted average, AdaBoost and so on [177]. In this Chapter, we propose two base models, as shown in Figure 5.2: A) The stock price information will be processed with industry information via GCN and then the updated embeddings of the stock will be fed into a GRU with a temporal attention layer, a hypergraph convolution network and a linear neural network, to obtain a prediction. B) Stock features will directly pass through a GRU, temporal attention layer and hypergraph convolution network in order to obtain another prediction, which will be concatenated with the one generated by model A as $\hat{y}_s$. Additionally, adversarial examples after the hypergraph convolution will be fed into the linear prediction layer, yielding a $\hat{y}_{adv}$. We use the max voting here to classify the sample as the class $\hat{y}_s$ and $\hat{y}_{adv}$ with the highest votes. The rationale behind it is multi-fold:

- Consensus decision making: max voting leverages the collective decision power of both $\hat{y}_{adv}$ and $\hat{y}_s$. When both models agree on the classification, it increases our confidence in the decision.

- Robustness: in situations where one model may be misled by certain features or noise, the other model can potentially correct this, leading to a more reliable classification.

5.4 Experiments

5.4.1 Datasets

In this Chapter, we utilise the dataset from [59], containing 758 frequently traded stocks collected from the A-share market in China between 01/04/2013 and 12/31/2019, to be fairly compared with [59]. While this dataset provides valuable insights, we acknowledge that our findings might be influenced by the specific characteristics of this particular dataset. To ensure broader applicability and to address potential dataset-specific biases, future work will involve collecting and analysing data from other markets such as US or Europe. This will allow for a more comprehensive validation of our framework and its generalisability across diverse market conditions.

Each stock in the dataset has six attributes: the opening price, high price, low price, close price, trading amount, and value. All the input features of stocks have been min-max normalised. If some stocks lack trading data during a temporary suspension period, the price attributes of the most recent day before the suspension will be used. Following the approach in [59], we partition the dataset chronologically: the initial 60% spanning the early years serves as training set, the subsequent 20% for validation, and final 20% for testing. The validation set is utilised to optimise the hyperparameters of our model. The industry information groups stocks into 104 industry categories, defined by the Shenwan Industry Classification Standard[1], and 'fund-holding' information is learnt from quarterly portfolio reports of the 61 mutual funds established before 2013 in the A-share market, similar as [59]. Details

[1] http://www.swsindex.com/idx0530.aspx

of the dataset are shown in Table 5.1 below.

Table 5.1: Dataset summary

Data	Number
Node (Stock)	758
Attributes	6
Industry-belonging relationships	104
Fund-holding relationships	61
Percentage of steady	24.7%
Percentage of rising	38%
Percentage of dropping	37.3%

5.4.2 Experimental Setup

We implement the proposed ensemble learning framework with PyTorch 1.10.2 and CUDA 10.2. According to [36], the length of historical information impacts model performance; here, we thus test the model with different lengths of trading information: the past 5, 10 and 20 trading days. Hyperparameters are borrowed from [59], optimised with the same validation set, for a fair comparison: the feature dimension of a stock is set as 16 and batch size as 32; the hidden units' size of GRU is 32 and the dimensions of d_k and d_v in the temporal attention mechanism are both 8. The maximum number of epochs is 600 and the dropout rate is 0.5. β in the loss function 5.3 is set as $1e-2$, as in [161]. We use the same settings for the baseline models as their public implementations.

5.4.3 Baselines

We compare against prior works, from one trading method, mean reversion (MR), one conventional LSTM model, one dual attention LSTM and five recently proposed graph neural network algorithms as baselines for stock movement prediction.

- MR: the model applied a mean reversion indicator to predict the local trend reversion by assuming extreme changes in the price will revert back to its previous state [178].
- LSTM: the model applied LSTM to predict future movements of stock prices [179].
- DARNN: the model proposed a dual stage attention with recurrent neural network, to selectively harness relevant features for efficient prediction [180].
- GCN+LSTM: the model applied GCN to learn relationships among stocks via feeding embeddings of stocks to an LSTM network [153].
- HATS: the model used a hierarchical attention network to adaptively learn the importance of different relation types for stock prediction [150].
- TGC: applied a novel temporal graph convolution to model the temporal evolution jointly and relational embeddings of stocks for prediction [143].
- STHGCN: proposed a novel Spatio-Temporal Hypergraph Convolution Network to learn stock price evolution over stock industry relations [148].
- HGTAN: proposed a novel hypergraph tri-attention network to predict the stock price movement and considered both industry-belonging and fund-holding

information [59]. This represents the current state of the art.

5.4.4 Evaluation

In financial investment, the decision-making process requires more than just overall accuracy of predictions. It is critical to minimise false positives (erroneous buy signals), while ensuring genuine profit opportunities are not missed. Therefore, in addition to accuracy, we incorporate precision, recall and the F1 score, to evaluate the classification performance of the proposed model, comprehensively. These metrics are calculated as follows:

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F1 - score &= 2 * \frac{Precision * Recall}{Precision + Recall} \end{aligned} \tag{5.10}$$

Where TP and TN are the correctly predicted positive classes and negative classes, respectively. FP and FN denote the falsely predicted positive classes and negative classes, respectively. As we implement multi-label classification (3 movement directions), the metric is calculated in macro-setting using the scikit-learn library. For instance, precision is calculated as:

$$Precision = (\frac{TP_r}{TP_r + FP_r} + \frac{TP_s}{TP_s + FP_s} + \frac{TP_f}{TP_f + FP_f}) * \frac{1}{3} \tag{5.11}$$

where r, s and f refers to rise, steady and fall, respectively.

5.5 Results and Discussion

5.5.1 Effectiveness Results on the Real World Dataset

Table 5.2 show the future stock price movement prediction performance of different models on the real-world benchmark datasets, with the past 5, 10 and 20 trading days as lengths of the look-back window. Compared with the traditional RNN method (LSTM), the significant improvement of the prediction performance obtained by all other models demonstrates the effectiveness of using graph neural networks for stock price trend prediction.

The state-of-the-art method, HGTAN, boasts a lightly superior accuracy compared to MONEY, when using the past 5 or 20 days of price records for prediction. However, accuracy is not the sole determinant in assessing the efficacy of stock movement classification. MONEY mostly significantly outperforms all baselines in precision, recall and F1 score metrics, where it exceeds the second best by an average of 1.14%, 3.63% and 2.45% for three trading lengths.

In the context of stock investment, precision corresponds to the model's capability to correctly predict a stock's upward movement, thereby minimising 'false buy' signals or false alarms. Concurrently, recall measures the model's sensitivity, ensuring no profit opportunities are overlooked. F1 score, a harmonic mean of precision and recall, can enhance a model's sensitivity and generalisation ability for performance evaluation [181]. It encapsulates the model's prowess in mitigating false buys, while maximising genuine profit opportunities. While accuracy is an important metric, it can be disproportionately influenced by the most common categories. In a high-stakes domain, like finance, where balancing risk with opportunity

is paramount, the F1 score offers a more comprehensive and nuanced evaluation. Therefore, we advocate that F1 score stands out as the most critical metric among the ones considered.

In our evaluation across 5, 10 and 20 trading days as record, the proposed MONEY model cumulatively enhanced the F1 score by 7.34% compared to the state-of-the-art HGTAN. We use the Wilcoxon signed-rank test for this comparison, due to the fact that it is suitable for non-parametric data and for comparing two related samples. Over six runs, the test yielded a W value of 0, which matches the critical value for $N = 6$ at $p \leq 0.05$, thus indicating a statistically significant improvement by the MONEY model at the 0.05 significance level. Follow-up ablation studies demonstrate the effectiveness of capturing group-level and pairwise information via hypergraph convolution and GCN on stock price movement. Our model obtains the highest value for the ten trading days window for the accuracy metric and performs relatively competitively with knowledge of the past 20 and 5 trading days.

Table 5.2: Performance of baselines versus MONEY over 5, 10, 20 trading days

Method	Trading Days	Accuracy	Precision	Recall	F1
MR	5	35.59%	39.37%	33.77%	36.36%
	10	34.73%	29.34%	31.79%	30.52%
	20	35.32%	38.03%	33.60%	35.68%
LSTM	5	34.92%	35.34%	33.91%	34.27%
	10	35.09%	38.09%	34.37%	35.90%
	20	35.03%	36.43%	34.23%	35.20%
DARNN	5	37.68%	37.81%	35.17%	36.43%
	10	38.89%	38.59%	35.22%	36.82%
	20	38.41%	37.99%	39.24%	38.60%
GCN+LSTM	5	37.24%	37.23%	33.54%	35.22%
	10	37.44%	39.07%	34.49%	36.62%
	20	37.30%	39.28%	34.16%	36.54%
HATS	5	38.74%	36.92%	34.29%	35.52%
	10	38.05%	39.23%	34.52%	36.67%
	20	38.85%	38.70%	35.06%	36.78%
TGC	5	37.43%	38.28%	34.05%	36.01%
	10	38.42%	39.35%	35.72%	37.44%
	20	37.81%	36.96%	34.49%	35.67%
STHGCN	5	38.53%	37.35%	34.65%	35.89%
	10	38.81%	36.57%	35.11%	35.75%
	20	38.45%	37.22%	32.82%	34.87%
HGTAN	5	**39.51%**	38.90%	36.96%	37.89%
	10	39.83%	**41.72%**	37.32%	39.37%
	20	**40.02%**	41.77%	39.03%	40.32%
MONEY	5	38.67%	**42.06%**	**41.80%**	**41.93%**
	10	**41.04%**	39.83%	**41.79%**	**40.79%**
	20	39.90%	**43.92%**	**40.61%**	**42.20%**

5.5.2 Investment Simulation and Profitability

We assess our MONEY model from 04/23/2019 to 05/09/2019, a 10 trading day span, in line with the benchmark set by [59]. This time frame was chosen based on the baseline study's emphasis on HGTAN's performance during this period when most stock prices fell. As depicted in Table 5.3, MONEY outperforms by achieving the highest accuracy over 5 days, emphasizing its reliability even in challenging market condition.

Table 5.3: Accuracy of stock movement prediction between 04/23/2019 and 05/09/2019

Date	MR	GCN+LSTM	HATS	TGC	STHGCN	HGTAN	MONEY
04/23	16.23%	22.43%	37.86%	69.39%	21.24%	57.78%	**71.37%**
04/24	**52.77%**	**48.02%**	38.26%	21.24%	27.44%	22.03%	28.76%
04/25	9.76%	15.44%	31.40%	77.97%	15.04%	**88.26%**	70.84%
04/26	17.68%	21.50%	38.13%	43.40%	23.88%	60.16%	**72.03%**
04/29	19.31%	23.22%	43.67%	65.57%	14.78%	**72.43%**	27.18%
04/30	64.78%	63.72%	50.53%	65.44%	66.62%	16.62%	**71.90%**
05/06	7.12%	11.35%	48.55%	15.17%	12.67%	**76.52%**	71.77%
05/07	73.48%	71.11%	50.26%	15.30%	**75.20%**	31.27%	27.84%
05/08	25.33%	28.36%	44.59%	48.15%	26.65%	44.99%	**70.84%**
05/09	17.81%	18.73%	33.25%	22.69%	19.26%	56.07%	**71.77%**

MONEY achieves over 70% accuracy (all accuracies higher than 70% underlined) in seven days, while state of the art, HGTAN only has such performance in three days, and the remaining models mostly have accuracy higher than 70% in one of the ten trading days. We normalised four important market indexes: Hushen 300,

Shenzheng Zongzhi, Shangzheng 50 and Zhongxiao 300 to visualise in Figure 5.6, and they all plummeted into a so-called 'bear market' during the ten trading days. While it is conceivable that the relative performance of different methods may vary under different market conditions, we can evaluate their robustness during challenging market dynamics. Moreover, the bear market presents critical scenarios, where consistent accurate prediction can be particularly valuable to investors.

More specifically, the most representative index Hushen 300, consisting of the 300 largest market capitalization and most liquid stocks, fell from 4019 on 04/23/2019 to 3599.7 on 05/09/2019 rapidly, and the investors all panicked. Figure 5.7 shows falling stocks are the majority for seven days (04/23, 04/25, 04/26, 04/29, 05/06, 05/08, 05/09), and our method constantly obtains over 70% accuracy except for one day, which distinct the effectiveness of our method with other models to avoid loss. However, it is important to note that this evaluation is based on a specific time window, and that broader and varied time-frames might yield different insights into the model's performance. Therefore, we further compare the profitability of different models from 08/31/2018 to 10/31/2019, similarly to [59], as shown in Table 5.4.

Table 5.4: Profitability of different models during back test

Method	CR	SR
MR	4.89%	0.123
LSTM	4.73%	0.147
DARNN	3.23%	0.083
GCN+LSTM	6.51%	0.217
HATS	11.55%	0.697
TGC	8.23%	0.513
STHGCN	6.23%	0.248
HGTAN	19.78%	0.699
MONEY	22.33%	0.775

The cumulative return rate, CR, aggregates the return rates of all stocks and is

frequently used in profitability analysis [182]. SR denotes the Sharpe ratio, which measures investment return with the associated risk [183, 184]:

$$SharpeRatio = \frac{E(R_s) - R_f}{\sigma_s} \tag{5.12}$$

where E is the expected value, R_s is the return rate of stock s and R_f is the risk-free rate (1.5% as one-year deposite interest rate in 2019), and σ_s denotes the standard deviation of the stock excess return. Our method reaches the highest cumulative return rate, 22.33%, with competitive performance at risk management with the highest Sharpe ratio of 0.775, which indicates it returns more profit compared with other models under the same risk situation.

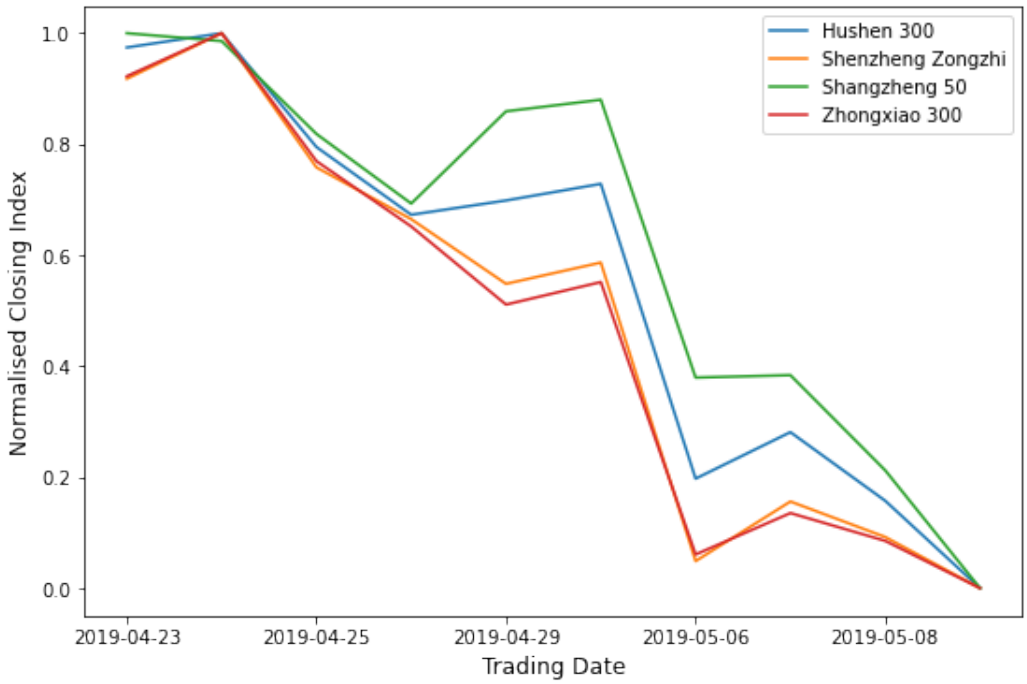


Figure 5.6: China A Share Market Index Trend

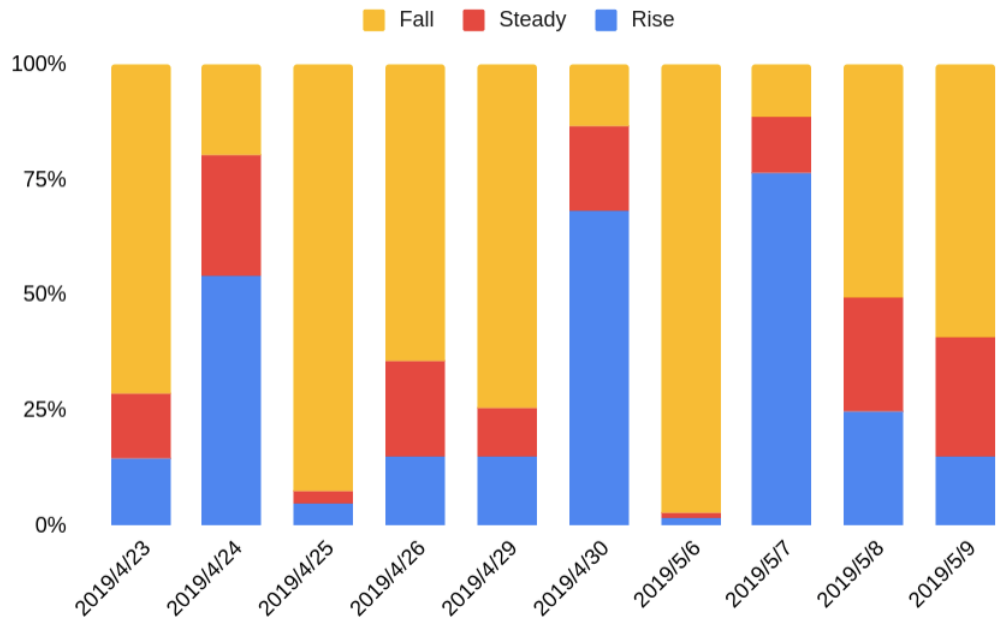


Figure 5.7: Different Movement of Stocks

5.5.3 Ablation Study

We compare the classification performance of our MONEY framework, its separate components and the second best model HGTAN, in Table 5.5, on the dataset with knowledge of the 10 past trading days. We thus demonstrate the improvement brought by each module, first separately, and then as a whole, via our MONEY framework:

- GCN+GRU+TA+HGCN (Module A): GCN is firstly applied to process historical price features with industry information augmenting the *pairwise relations*, followed by a GRU network with a temporal attention layer (TA) and the generated embedding is then fed into a hypergraph convolution network (HGCN) to learn the industry and fund-holding *group-level* relations of stocks.

- GRU+TA+HGCN (Module B): a variant without pairwise relations using GCN; otherwise it is the same as variant A.

- GRU+TA+HGCN+Adv (Module C): this variant does not consider pairwise relations using GCN but extends HGCN with adversarial training to enhance the generalisation ability to deal with the *stochasticity* of stock features, as [161] suggested, compared with variant A.

- GRU+TA (Module D): this is the base component, which solely considers *historical* price information.

- GRU+TA+GCN+HGCN (Module E): this is to compare with Module A and demonstrates the effectiveness of applying GCN to capture pairwise relations before RNN models.

- GCN + GRU + TA + Adv (Module F): this variant contrasts with the MONEY method to validate the value of HGCN.

Table 5.5: An ablation study of MONEY on dataset with 10 trading days as record

Method	Accuracy	Precision	Recall	F1
Module A	**41.38**%	42.07%	39.35%	40.66%
Module B	41.28%	41.29%	38.11%	39.64%
Module C	40.01%	**42.87%**	38.56%	40.60%
Module D	38.86%	39.02%	38.20%	38.61%
Module E	40.37%	42.39%	36.42%	39.18%
Module F	39.38%	40.04%	37.34%	38.66%
HGTAN	39.83%	41.72%	37.32%	39.37%
MONEY	41.04%	39.83%	**41.79%**	**40.79%**

Overall, each module improves the performance compared to the baseline (Module D) and our model outperforms all variants in terms of recall and F1 score by using the synergistic power of all the modules together. MONEY also significantly outperforms the state-of-the-art competitor, HGTAN, in terms of accuracy, recall and F1 score and delivers significantly stable performance in a bear market, as addressed in section 5.5.2. To be specific, when comparing modules A and B, GCN improves all four metrics in total 3.14%, particularly 1.03% in the most critical metric, F1 value, as discussed before. This validates our point that pairwise relations should not be neglected, even if we have already considered group-level information. Compared with Module E, Module A significantly improves three measurements in total 5.11% (1.49% in F1 value) and performs competitively in terms of precision.

The result illustrates that price movement can be learnt more effectively by using GCN before RNNs. GCN will first capture similar volatility patterns of stocks in the same industry, enhancing the prediction ability of the following RNN models. Moreover, adversarial training can significantly enhance the model's robustness by considering the stock price's random movements, as the precision, recall and F1 score of module C are all improved, compared with module B, at a reasonable cost of downgrading the accuracy. Adding a hypergraph convolution to module D, the fundamental function would improve by 2.41 % in terms of accuracy, shown in module B. The precision and F1 score can also increase by 2.27% and 1.03%. In addition, the comparison between MONEY and module F also demonstrates the need to use hypergraphs to deal with the group-level information for stock price movement prediction.

5.6 Potential Future Work

In this study, we only test our model in one market, and the time period does not include Covid-19 to be comparable with [59]. It is a limitation and in future, we will explore the method in different markets. More advanced deep learning methods, including graph contrastive learning [23] and graph dynamic attention [185], can also be applied for stock prediction tasks. There are several promising future research directions to explore:

- One is the *relational structure discovery* [186], aiming to learn the optimal computational graph structure. Such a learning method does not require explicit supervision and can enhance the interpretability of models.

- The current results can be improved, by considering more demographic information, such as the services or products provided by the company, as [150] suggested, or other information, such as the global market context [187] and the investor sentiment [188, 189].

- Our proposed MONEY framework can be applied to other markets to explore the price movement prediction of more assets, such as bonds, options and commodities.

- Develop a fast ensemble deep learning solution to avoid massive computing time and save space overheads when facing large-scale datasets.

5.7 Summary

In this Chapter, we point out that existing work for stock movement prediction suffers from insufficiently capturing both group-wise and pairwise relations of relevant information rather than solely historical price features, leading to a weak generalisation ability due to the stochastic characteristics of stocks. We also demonstrate that pairwise relation learning should be applied before RNN models rather than later, as stocks display similar volatility patterns and using the GCN model first to capture the patterns can enhance the prediction ability of the following RNN models.

Addressing these problems, we propose a novel ensemble learning framework, MONEY, to better assist investors in predicting future trends of stocks. To effectively capture *pairwise information* of industry, a graph convolution network is applied before RNN models. To capture the *group-level information* of both industry and fund-holding, a hypergraph convolution network is implemented after the GRU model with a temporal attention layer. Adversarial training is introduced before the final prediction layer in the model (B) to simulate the *stochastic movement* during training. Ensemble learning allows the two models to complement each other, keeping their benefits and enhancing learning about these relations and robustness. All components are jointly trained on real-world stock market datasets. Our model significantly outperforms the state-of-the-art for most of the indicators without using complex triple attention mechanisms among hyperedges and hypergraphs and provides much more stable performance, particularly when facing a bear market.

It is prudent to note that the evaluation is majorly based on a single dataset

from the A-share market in China. While this dataset offers a wealth of insights, the findings might carry inherent biases, specific to its unique characteristics. For a more wholistic understanding of MONEY's transferability and broader applicability, future research endeavours should consider engaging with diverse datasets, possibly from varied markets, like the US or Europe, as discussed in section 5.6.

This Chapter tackles the challenge of graph structure learning by taking a comprehensive approach to relation extraction. The next Chapter will delve deeper into the structure learning problem from position and geometry information learning.

CHAPTER 6

A novel framework for positional information learning

In this Chapter, we propose an effective framework to enhance the expressive power of the graph representation learning model to address the third research question in Chapter 1.2 from a positional information perspective:

How to improve the structural learning ability, e.g. relation extraction and positional encoding for existing graph representation learning?

We convert graphs into patches with fixed-length vectorial representation and consider the Ricci curvature and contrastive patch embedding to learn a robust representation of nodes without changing the graph structure. Then a standard GNN model and Performer are implemented to obtain the final graph-level representation. The effectiveness of the framework has been validated on real-world datasets.

This Chapter is based on the following manuscript, which is accepted as listed

in Chapter 1.5:

***Sun, Z.**, Harit, A., Cristea, A. I, Lio, P and Wang, J. A Ricci Curvature Contrastive PerformerMixer Framework for Graph Representation Learning. In 2023 IEEE International Conference on Big Data (Big Data)*

6.1 Introduction

As presented in chapter 2, (GNNs) have become a rapidly growing field, widely studied in various applications, including natural language processing [23, 56], healthcare [190, 191] and recommendation [33], where data is non-Euclidean [192]. Concomitantly, transformers [43] are prevalent in computer vision (CV) [193], natural language processing (NLP) [131], speech recognition [194], and reinforcement learning [195], with the advantage of encoding the relative influence of any object to the target entity. Consequently, researchers became interested in the performance of transformers for graph representation learning. Ying et al. [63] proposed Graphormer, a model directly capturing the structural information (*centrality, spatial and edge*) of graphs via a standard transformer, and demonstrated the effectiveness of positional encoding and enhanced the expressive power in the Open Graph Benchmark Large-Scale Challenge [10]. Others used Laplacian eigenvectors [50, 52] or relative distance [196], to add positional information to node features.

However, there is no guarantee that such positional encoding captures the intricate structural relations between nodes. Given that transformers generate same embeddings for nodes, irrespective of their surrounding structures and considering the inherent permutation invariance of self-attention, there is a potential risk of

overlooking distinct structural differences between nodes [2], as depicted in Figure 6.1.

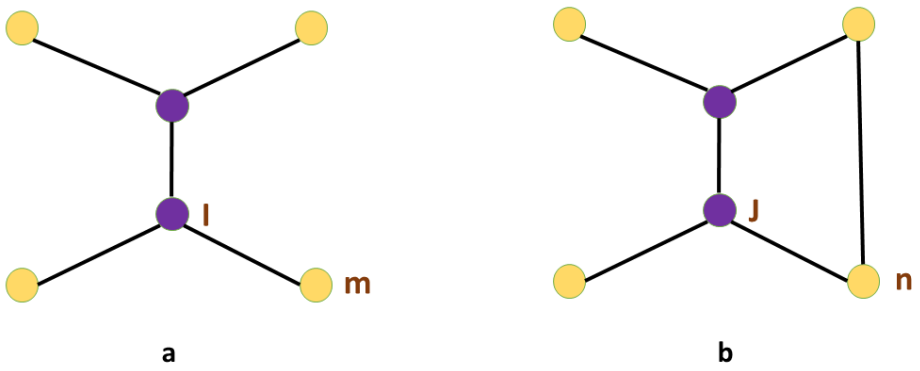


Figure 6.1: *Position-encoding* vs *structural encoding*. The idea for the figure is taken from [2, Figure 1]. Shortest path positional encoding will generate the same representation for nodes I and J in a) and b) because they have identical shortest paths to other nodes, even though local structures are different.

Moreover, as [61] proposed that for graph representation learning, a competent position encoding should be: 1) *local*: nodes should be aware of their role and position in a local community, 2) *global*: nodes should know their global position in a graph, 3) *relative*: nodes should be clear about the distances between them and other nodes. However, considering these three types of positional information simultaneously in graph representation learning and how to improve it without significantly increasing the computational complexity is relatively complex.

This Chapter addresses the complex challenge of considering these three types of positional information together and how to refine this representation further and proposes a more general and effective framework, the *Ricci Curvature Contrastive PerformerMixer (RCPP)* for graph patch learning.

It is known that position encoding in a transformer would be simple for sequence learning with inherent ordering [197]. Thus, we aim to convert graphs into sequences better to capture the structural relations for graph representation learning. However, graphs can be irregular, and it is a challenge to process various sizes of nodes and edges into a fixed length of tokens. Google proposed ViT [198], a transformer with decomposed input images, as different fixed-size patches (see (a) in Figure 6.2), which performed well, with fewer computational costs than its competitors.

Therefore, we explore a similar method to decompose graphs into patches (alternatively tokens) and apply a transformer for graph representation learning, illustrated as (b) in Figure 6.2.

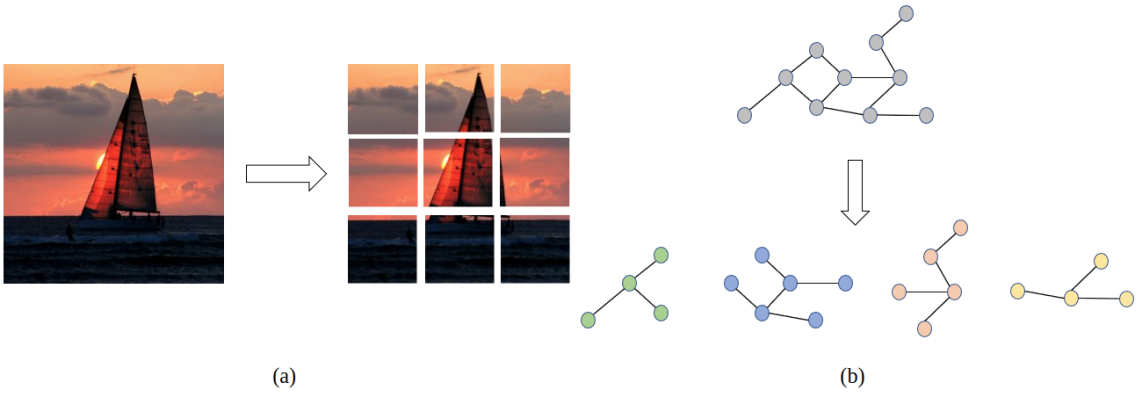


Figure 6.2: Partition images and graphs

In accordance with [61], our framework comprehensively incorporates *local*, *global* and *relative* structural information. After patch transformation, a linear transformation aguments node features with local positional information, utilising random-walk structural encoding [16] as detailed in section 6.3.1. We then incorporate more topological information (via Ricci curvature), an effective geometrical measure that enriches both edge and global manifold characteristics in graphs [199, 200]. Rooted

in the intrinsic curvature properties of manifolds - continuous, smooth spaces [201], the Ricci curvature serves as a powerful attribute for our framework to mitigate the *over-squash* problem that hinders GNNs' distant information processing capabilities [202]. The consideration of the Ricci curvature amplifies the expressive power of our framework in capturing nuanced global structures, and more details of the mathematical definitions and formulas are presented in section 6.3.2. Complementarily, our PerformerMixer, elaborated in section 6.3.4 also consider global graph structures. Furthermore, relative positions among patches are incorporated to preserve inter-patch relationships post-graph partition.

Next, we apply message passing based GNNs to the partitioned graph patches to obtain fixed-length embeddings for arbitrary graphs. However, message passing may lead to the problem of over-squashing [203]. This pertains to the distortion of information passed from distant nodes, especially along edges with negative Ricci curvature values (negatively-curved edges) [199]. One solution is curvature-based graph rewiring by adding and removing edges [199]. However, this method will change the graph structure and the graph may become disconnected soon by removing the negatively curved edges [204]. Such edge perturbation will hurt downstream performance on biochemical molecules [116, 205] when discriminating between molecules of different classes (graph classification tasks). Therefore, instead of explicitly changing the graph structure, we implicitly consider the impacts of topological information via Ricci curvature in contrastive learning.

The *contrastive learning method* is deployed to encode underlying structures, address the over-squashing problem implicitly and learn more robustly about variation

representations, such as rotations and scale changes. It should be noted that conventional contrastive learning assumes all negative samples are equally negative against positive samples. This leads to the *faulty negative problem*, where instances similar to the anchor are treated the same as more negative instances during contrastive training [206, 207]. Addressing the issue, the negative samples here are updated according to the pairwise distance between them and the positive ones together. Our contrastive learning method is the first model to consider the edge Ricci curvature information in node embedding update.

Next, a standard GNN is implemented for each graph patch independently to update nodes and edges information, similar to [3]. Thereafter, we implement *Performer*, an efficient computing transformer for long-length sequences, together with an efficient image patch learning model, MLP-Mixer [208] to value the importance of patches and obtain a weighted average graph level representation, inspired by the idea that a comprehensive architecture of transformers is essential for vision task.

Given the above, our main contributions can be summarised as follows:

1. We propose a novel *effective general and expressive framework, the Ricci Curvature Contrastive PerformerMixer (RCPP)*, for graph representation learning with linear complexity by converting graphs into patches and capturing structural and positional relations comprehensively.

2. Unlike the base model [3], we are *the first to apply contrastive learning to graph patches, thus removing the faulty negatives problem*, to the best of our knowledge.

3. Additionally, we propose a novel Ricci curvature guided contrastive learning

method, which alleviates the issue of over-squashing without changing graph structure.

4. Finally, our empirical evaluation on real-world datasets demonstrates that RCPP framework surpasses existing methods and attains 3-WL expressive power. Furthermore, the modular design of RCPP serves as a robust baseline for future academic research in graph representation learning.

6.2 Related Work

6.2.1 Contrastive Learning for Molecule

Contrastive learning is becoming prominent for pre-trained methods. As explained in sections 2.3 and 4.2, it aims to learn embeddings of objects by comparing different samples instead of learning the mapping function between input and labels [41, 209, 210]. Notable models such as Momentum Contrast [211], SimCLR [42] and SimSiam [212] have been introduced for visual representation learning (images and videos). This Chapter mainly uses contrastive learning for molecules which has been proven to be effective at capturing unique cheminformatics and graphical structures [207].

Conventional contrastive learning methods focused on generating diverse views for the same sample [205], including masking nodes or transforming graph structure [100, 116, 213], which will hurt the semantics inside molecules. Wang et al. [214] introduced MoICLR and proposed augmentation strategies for building contrastive pairs. Zhang et al. [215] proposed a new sampling strategy by leveraging extracted motifs to select informative subgraphs for contrastive learning. Li et al. [216] con-

ducted contrastive learning between 2D and 3D geometric structures to incorporate 3D information. However, as [207] pointed out, all these studies believed that other molecules are equal negative pairs and thus brought faulty negative problems. Unlike those studies, we do not assume that all the negative samples are equally dissimilar from the positive sample, and we have further adjusted them by calculating the pairwise distance between them. Others [205, 217, 218] incorporated external domain knowledge for graph augmentation and then applied contrastive learning to maximise the agreement between the augmented and the original molecular graphs. This domain knowledge enriched with knowledge graphs can be the future extension of our work.

6.2.2 Transformers with GNN and Variants

Transformer models have been increasingly applied to graph-structured data, as they can encode the structural information via positional relationships between nodes and thus avoid structural inductive bias [2]. Zhang e al. [7] introduced Graph-BERT, which employs self-attention on sampled linkless subgraphs to capture structural information. Their approach highlights the importance of attention mechanisms for learning graph structures. Min et al. [219] summarised three typical ways to combine graph representation learning with a vanilla transformer: 1) Using GNN as auxiliary modules in a transformer to leverage the advantages of learning local representation and global reasoning via pairwise interaction, such as GraphTrans [203]; 2) Improved position embedding from graphs that compresses graph structure without adjustment of transformer architecture including Laplacian eigenvectors

in Graph Transformer [50] or random walk [16, 61]; 3) Revised graph attention matrix based on graph information, e.g. GraphiT [196], which encoded various graph kernels, by extending the adjacent matrix to a kernel matrix and Structure-Aware Transformer (SAT) [2] which incorporates subgraph representation as structural information into self-attention. However, as [3] suggested, those graph transformer methods significantly increase the computing complexity when addressing the long-range dependencies and over-squashing problems.

On the other hand, as transformer architectures are indispensable for deep learning, a number of improvements compared to the original transformers have been proposed [220], including Linear transformer [221], Longformer [222], Performers [223], Switch Transformer [224] and so on. Most focus on improving the attention mechanism's memory complexity and computing efficiency. Here we use Performer, which will be most efficient when L, the sequence length is relatively large (i.e. 4096) by using random kernels. More comprehensive information regarding the recent variants of transformers can be found in a recent survey [220].

6.3 Method

In this section, we introduce our Ricci Curvature Contrastive PerformerMixer (RCPP) framework, which consists of the Ricci curvature consideration of graphs, patch contrastive learning and Performer [225] with MLP-Mixer [208], based on Graph MLP-Mixer model [3]. The overall structure of our framework is shown in Figure 6.3.

The general pipeline structure of the RCPP framework means that it and each

component can be easily integrated with other graph representation learning models. The patch partition method can be applied to graphs in different domains. The position information of nodes and patches is also considered and will be explained later. The message passing mechanism aggregates information from neighbouring nodes and is particularly suited to capturing local structure (details explained in Chapter 2.2). In addition, the Ricci curvature, considered later, establishes a connection between geometry and topology by utilizing local information [200]. For global structural information, we apply a comprehensive transformer, Performer-Mixer and use an adjacency matrix to encode the relative positional information of nodes in different patches. We consider Ricci curvature to address the over-squashing problem faced in most GNN methods and develop a novel Ricci curvature guided contrastive patch learning to capture unique substructural information and be more robust simultaneously. PerformerMixer successfully decreases the time complexity to linear, mainly when input length L is substantial, which can replace a standard transformer in existing GNN with transformer study.

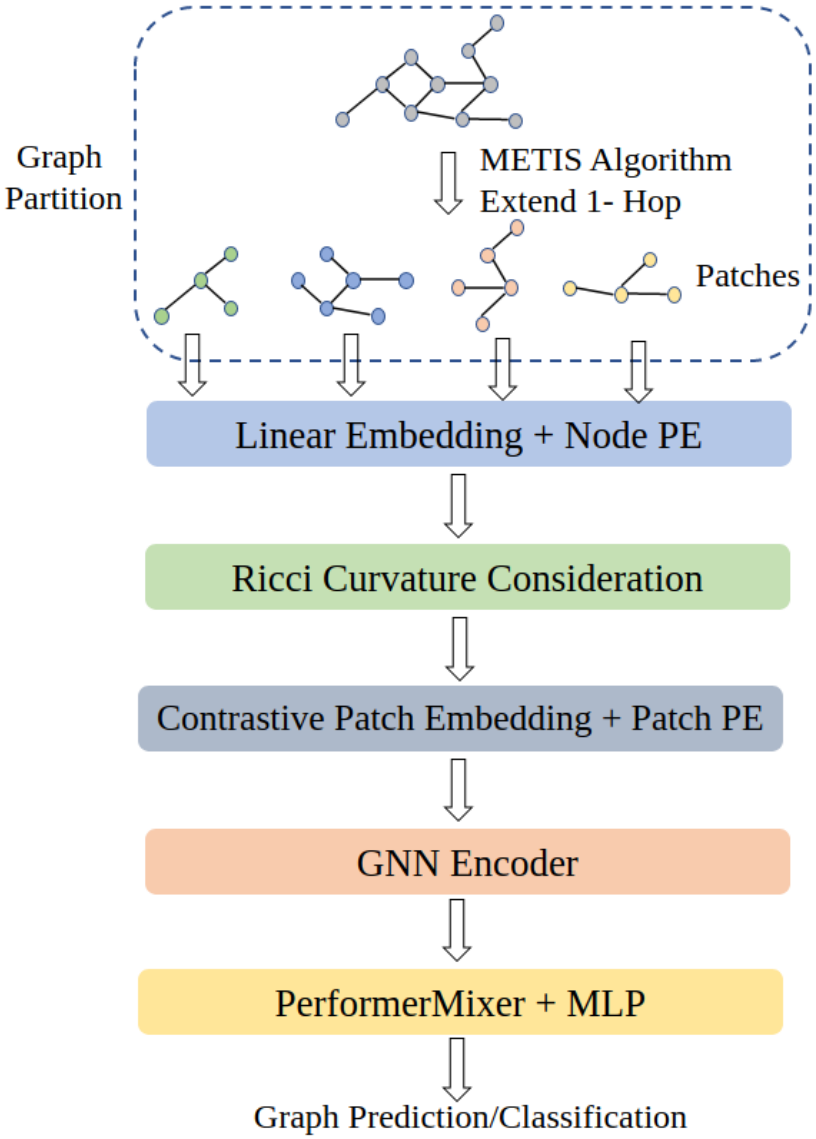


Figure 6.3: Proposed RCPP framework

For graph partition, we apply the METIS algorithm [3] to cluster raw graphs $G(V, E)$ into a number of P patches $V_1, V_2, ...V_P$. METIS is a multi-level partitioning technique, which divides a graph into smaller, balanced subgraphs by iteratively optimizing the cut size (minimize inter-subgraph connections) and minimizing communication costs (load balance) [226]. This partition is extended to the overlapping patches, in which case, if the two nodes of an edge belong to different patches, they

will also be included in both patches. Next, we deploy a linear embedding to the given raw input nodes and edges in the obtained graph patches separately (see equation 6.1). Our RCPP framework converts graphs into patches and is followed by: Ricci curvature consideration, contrastive patch embedding (learn more robust representation and address the over-squashing problem simultaneously), GNN Encoder (updating nodes and edges within patches and transforming them in fixed-length vectors), PerformerMixer (a comprehensive architecture of linear time complexity transformer and MLP-Mixer for patch vector) and a MLP (multi-layer perceptron) for the final prediction/classification task (see Figure 6.3).

6.3.1 Positional Encoding

Node Position Encoding. In section 2.4, we introduce the random walk structural encoding (RWSE) p_i for node i [16]. This encoding captures both local and global positional information, as detailed in equation 2.13 of Chapter 2.By ensuring most nodes in real-world graphs receive unique representations, RWSE bolsters the expressiveness of GNNs. This enhancement is depicted in Figure 6.3 and further elaborated in equation 6.1:

$$h_i = \sigma(w_n)Tp_i + U\alpha_i + u; \quad e_{ij} = F\beta_{ij} + f \tag{6.1}$$

h_i and e_{ij} are the updated node and edge features in d dimensions, projected from the original input node feature and edge feature α_i and β_{ij}, respectively. T, U, F, u and f are parameters to train. Differing from the baseline, we introduce a parameter w_n initialised with a scalar value of 10.0. We then apply a sigmoid

function σ to the tensor. The purpose of this trainable parameter is to determine the significance of node position information when it is incorporated into the node embedding. Essentially, it allows the model to learn the optimal degree of reliance on position information.

Patch Position Encoding. Additionally, we incorporate the adjacency matrix A to encode the structural relationships between patches, where the entry at row i and column j is the number of shared nodes between patch i and j. To capture the relative positions among patches, we perform eigenvalue decomposition on A', a modified version of A to enhance numerical stability and obtain its eigenvectors P:

$$A' = A + eps * I, A' = \sigma(w_p)P\Lambda P^{-1} \tag{6.2}$$

Where eps is a small constant and I is an identity matrix. These eigenvectors P serve as a form of relative positional information and are integrated into the patch embeddings later. This approach is inspired by the proven ability of eigenvectors to preserve intrinsic graph structures, as evidenced in [227]. To modulate the significance of this relative positional information, we introduce a learnable parameter tensor w_p, initialised to a scalar value of 10.0 for patch position information, mirroring the approach used for node position information.

6.3.2 Ricci Curvature Consideration

As noted, we incorporate the Ricci curvature into our pipeline, to bolster topological insights, address the 'over-squashing' problem in GNNs and enhance our framework's capability to discern structural information. This curvature provides a

more localised perspective and an accurate view of the bottleneck in message passing from distant nodes (long-range dependencies) [199].

First, we consider the Ricci curvature because we will implement contrastive patch learning in the next step. Prior works have shown that motif-level contrastive learning methods [214, 215, 228] all sample subgraphs within each molecule, leading to the neglection of unique chemical substructure pattern [207]. Therefore, we consider edges with negative Ricci curvature for each molecule to avoid such information loss and address the over-squashing problem. The primary purpose is to identify negatively curved edges, as depicted by the green edge in Figure 6.4 (a), which contribute to the over-squashing problem [18]. Unlike the conventional rewiring method [199] to introduces additional edges to alleviate the bottleneck, illustrated by red dashed lines in Figure 6.4 (b) which alters the inherent structure, we update the Ricci curvature of edges and subsequently perform contrastive learning in the next step based on the updated values.

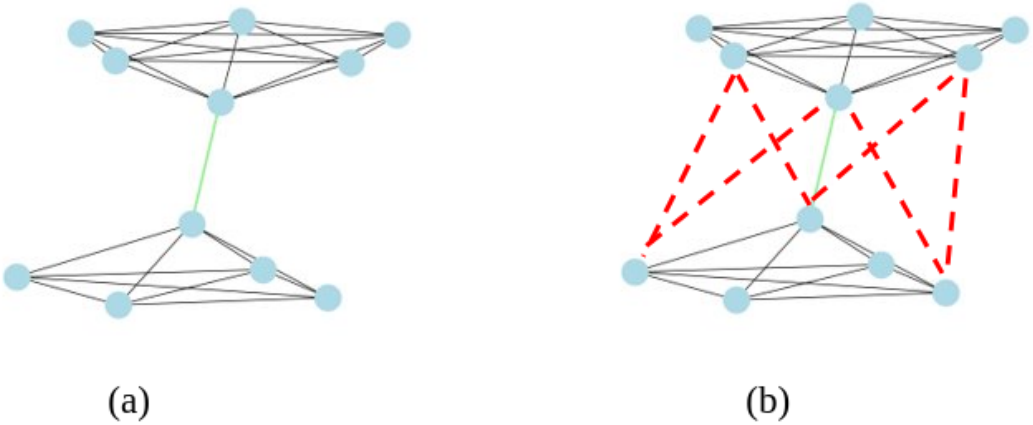


Figure 6.4: Curvatures on graphs

According to [204], the Ricci curvature κ_{ij} of edge e_{ij} is determined by comparing the distance traveled from node i's neighbors to node j's neighbors (measured by Wasserstein distance $W(m_i, m_j)$), relative to the length of edge $d(i, j)$, formulated as:

$$\kappa_{ij} = 1 - \frac{W(m_i, m_j)}{d(i, j)} \tag{6.3}$$

and the Wasserstein distance $W(m_i, m_j)$ can be computed by a linear programming:

$$\min_{M} \sum_{i,j} d(ij) M(ij) \text{s.t.} \quad \sum_{j} M(ij) = m\alpha(i), \sum_{i} M(ij) = m\alpha_j(j) \tag{6.4}$$

Where $m\alpha(i)$ is the probability measure at node i. The universe for this measure consists of node i, its set of neighboring nodes N and all other nodes in the graph. It is defined as:

$$m\alpha_i = \begin{cases} 1 & \text{if at node } i \\ (1-\omega)/k & \text{if node } i \in N \\ 0 & \text{otherwise} \end{cases}$$

We utilise a parameter ω that falls within the range of [0,1] in our model. This parameter is responsible for allocating a specific probability mass at node i while evenly distributing the remaining probability mass among number of neighbouring nodes k. For a more detailed explanation, readers can refer to [204].

These Ricci curvatures of edges are precomputed and stored in datasets. During training, we can load it directly and then use the same GCN [35] implementation as [199] to update edge features:

$$h_i' = ReLU(GCNConv_1(h_i, e_{ij})),$$
$$e_{ij}a' = e_{ij}a + mean(GCNConv_2(h_i', e_{ij})). \tag{6.5}$$

Where h_i and e_{ij} are the node and edge features obtained from equation 6.1, with the Ricci curvature already incorporated into e_{ij}. The first GCN convolution layer $GCNConv_1$ updates the node features to h_i'. The second layer $GCNConv_2$, uses the updated node feature to update the edge features and calculate their mean. $e_{ij}a$ is the attribute information contained in edge ij, and it is updated to be used as edge information to guide contrastive learning in the next step.

6.3.3 Contrastive Patch Embedding

Different from [3], who directly input token representation into mixed MLP layers to fuse token and channel information, we utilise contrastive learning for nodes. The idea is that even if they are overlapping patches, graph information in each patch will not be similar. The current sample is assumed to be positive and the remaining are all negative. Unlike most existing contrastive learning methods [228–230] in graph and molecular representation learning, we do not assume all negative samples are equally dissimilar from the positive sample. We calculate the pairwise distance between the current positive sample S_p and other k numbers of negative samples S_n in the same group and update the embedding of the negative sample based on the distance by adding a weighting factor proportional to the pairwise distance, as illustrated in the following equations 6.6:

$$
\begin{aligned}
Dis(h_i, h_j) &= ||h_i - h_j||^2, \\
h_i^{update} &= h_i + \sum_{j \in k} W_{ij}(1 + \theta M_{ij} * e_{ij} a') * Dis(h_i, h_j)/\eta,
\end{aligned}
\tag{6.6}
$$

Where Dis represents distance, $||\ ||^2$ denotes the L2 norm, and η is the temperature factor to control the scale of the negative sample update. h_i is the embedding of node i. Since the pairwise distance computation is done only between positive and k negative samples in the same group, the overall time complexity of this operation is linear concerning the size of input samples, which will be explained more in 6.7. θ is a learned parameter representing the scale of influence of negative edge Ricci curvatures in message passing. When θ is large, the embeddings of nodes connected by negatively curved edges will be pushed further away. M_{ij} denotes a binary mask indicating if the Ricci curvature of edge ($e_{ij}a'$ in equation 6.5) connecting node i and j is negative or not. This enables the model to focus more on updating negative samples, which are closer to the positive sample, learn a more robust representation of the input and address the over-squashing problem simultaneously.

After contrastive learning, a standard GNN, GINE proposed by [121], is implemented to update nodes in each patch and average the overlapping patches for nodes and edges included in more than one patch, similar to [3]:

$$
h_{i,p}^{l+1} = f_{\text{node}}((1+\epsilon)h_{i,p}^{l} + \sum_{j \in N(i)} \text{ReLU}(h_{j,p}^{l} + e_{i,j}^{p})) + g_{p-n}(h_{i,p}^{l}). \tag{6.7}
$$

Where l is the index of the convolution layer, f_{node} is a neural network function to update nodes $h_{i,p}$.Directly adding $h_{i,p}$ serve as an implicit form of residual con-

nection, allowing the model to preserve original information and aiding in gradient flow. ϵ denotes a small positive number, acting as a slight bias to prevent nodes from being overly influenced by their immediate neighbors and the default value is 0 in the GINE model in the Pytorch Geometric library. i, j are the nodes and p is the index of graph patches. Node i and edge e_{ij} are included in patch c. $g_{p-n}(h_p^l)$ is a MLP function that further acts as an explicit residual connection during the GINE updating process. This explicit residual mechanism ensures that during deeper layers of convolution, the original node information is not overshadowed by the aggregated information, providing the model with a balance between local and neighbor information. Afterwards, mean pooling is applied to all nodes in each patch to obtain fixed-length patch embedding, which then passes to the Performer, as explained in equation 6.8:

$$h_{i,p}^{l+1} = Mean(h_{i,c}^{l+1}) \tag{6.8}$$

6.3.4 PerformerMixer

The Performer reduced computation complexity using orthogonal random features (FAVOR) [225] with random kernels. In the transformer, the time complexity was $\mathcal{O}(n^2)$ when the input sentence length was L. However, with the utilization of orthogonal random features (ORF) and modification of the vanilla attention from equation 6.9 to equation 6.10, [225] was able to improve the situation.

$$Att = softmax(QK^T/\sqrt{d}) * V \tag{6.9}$$

$$\hat{Att} = \hat{D}^{-1}(Q'((K')V)), \hat{D} = diag(Q'((K')T)1_L) \tag{6.10}$$

Here, Att is the attention, Q, K V are input matrices and d is the dimension of the hidden unit, and 1_L is a vector of ones with L length. A pivotal aspect of the Performer's efficiency comes from the nonlinear projection of queries and keys into a lower dimensional space of dimension r. This projection achieved through random orthogonal features, allows for the computation of approximated attention without evaluating the full QK^T product [225]. As a result, the time complexity becomes $\mathcal{O}(Lrd)$. To clarify further, the dimension r of the nonlinear projection essentially determines the resolution of the approximation. A smaller r results in a more coarse-grained approximation but is faster. When L is significantly larger than r and d, the advantage is that the time complexity will be approximately $\mathcal{O}(L)$.

After the patch transformation step, token representations are obtained. Then they will be input into the Performer with MLP-Mixer, which we name as PerformerMixer, as illustrated in Figure 6.3.

MLP-Mixer model has the advantage of alternating between channel and token mixing steps without an attention mechanism, which enhances information fusion between tokens and channels with outstanding performance for vision tasks [208]:

$$\begin{aligned} \text{Token mixer } U &= X + (W_2\sigma(W_1 LayerNorm(\hat{A}X))) \\ \text{Channel mixer } Y &= U + (W_4\sigma(W_3 LayerNorm(U)) \end{aligned} \tag{6.11}$$

However, the ability of transformers to capture long-range interactions can cir-

cumvent the over-squashing problem [231], which should not be neglected. According to [232], a general architecture of the transformers, rather than a particular component like a specific token mixer plays a more critical role in the model's effectiveness. Additionally, as explained before, position information is crucial for graph representation learning and is naturally captured by attention mechanisms. Therefore, our work aims to build a comprehensive architecture by combining Performer with MLP-Mixer to leverage linear time complexity of attention in Performer and efficient information mixing of MLP-Mixer on image patch learning.

This comprehensive PerformerMixer can effectively capture complex dependencies between different parts of the patch and considers the relative positions of tokens, leading to a better understanding of the graph's global and local structure. Before the final prediction layer, it should be noted that small graphs may partition empty patches, and we should mask them out and average the non-empty patches to obtain the graph level representation h'_G, similar to [3] using equation 6.12:

$$h'_G = \frac{\sum_p m_p \cdot h_{G_p}}{\sum_p m_p} \tag{6.12}$$

Where m_p is the binary mask for patch p. Lastly, we apply an MLP for the classification or regression task output y is given by:

$$y = MLP(h'_G) \tag{6.13}$$

6.4 Experiments

6.4.1 Datasets

We use the same four benchmark datasets as in [3], ZINC, Peptides-func, MolHIV and CIFAR 10, to fairly evaluate our method. ZINC is a molecular graph dataset that predicts the constrained solubility (regression task), a molecular property. We use the 12K molecules subset and follow the same data split as [48]. The peptides-func dataset is used for long-range interaction graph learning [62]. The MolHIV dataset is used for molecular property prediction, where nodes denote atoms and edges represent chemical bonds [70]. The above three datasets are used for graph classification tasks to differentiate among graphs from various classes. Each sample is a chemical compound depicted through a graph. The labels for graph classification tasks can take various forms: binary (as in MolHIV), multiclass (as in Peptides-func), or continuous values for regression tasks (as in ZINC). For CIFAR 10, an eight nearest-neighbour graph is used to create it from the classical image classification dataset [48]. Given its origin and structure, applying Ricci curvature, which is typically suited for smooth manifolds or their discrete approximations, may not be meaningful or informative for CIFAR10. The result presented in Table 6.2 of CIFAR10 is performed with the entire RCPP framework, except the Ricci curvature consideration and θ is set as zero in equation 6.6. These datasets are chosen for their diversity in representing various graph-related tasks, including regression, multiclass classification, binary classification, while ensuring both chemical compounds and image-derived graphs. While the baseline includes additional dataset such as MNIST, MolTOX21 and Peptides-struct, we believe our dataset choices are

Table 6.1: Dataset summary

Datasets	Graphs	Nodes	Avg Nodes	Class
ZINC	12000	9-37	23.2	Regression (1)
Peptides-func	15535	8-444	150.9	10
MolHIV	41127	2-222	25.5	2
CIFAR 10	60000	85-150	117.6	10
CSL	150	41	41	10
SR25	15	25	25	15
EXP	1200	32-64	44.4	2

sufficiently representative:

- CIFAR10: with its larger graph size, offers a more challenging image-based evaluation than MNIST.
- MolHIV, sourced from MoleculeNet like MolTOX21, makes MolTOX21's inclusion redundant.
- Both Peptides-func and Peptides-struct datasets are for long-range interactions tasks and even use same graph set. Therefore we just evaluate on Peptides-func in this chapter.

We also evaluate the expressive power of RCPP on three simulated datasets: CSL, SR25 and EXP, which will be presented in section 6.6. The details of all datasets are illustrated in Table 6.1.

6.4.2 Experimental Setup

Our model is implemented with Pytorch 1.12.1 and CUDA 11.3. The Ricci curvature is precomputed and stored in datasets using the GraphRicciCurvature library based on equation 6.3 and 6.4. The hyper-parameters for the METIS partition, patch

encoder part and training settings are mostly borrowed from Graph MLP-Mixer [3], including the number of patches 32, Adam optimizer with default setting and learning rate of 0.001, learning patience 100 for Zinc dataset, 20 for Peptides-func dataset and 50 for MolHIV and CIFAR 10 dataset. The dropout rate is 0.3, the number of hidden units d is 128, the number of GNN encoding layers is 4, and L, the max sequence of length and depth for the Performer, are set as 4096 and 6, respectively. Temperature η is set as 0.05 followed by [233]. The whole model is implemented on PyTorch Geometric.

6.4.3 Baselines

Five graph neural network models are evaluated with our method.

- GCN is a seminal and effective graph representation learning model, which propagates node features through its neighbors using an adjacency matrix to update vectorial representations for nodes [35].

- GINE proposes pre-training an expressive GNN to learn local and global representations [121].

- Graphormer incorporates structural information of graphs into a transformer, including centrality, edge and spatial information [63].

- Structure-Aware Transformer (SAT) considers structural information in terms of subgraph and incorporates it into transformer [2].

- Subgraph GNN (SUN) models graphs as a collection of subgraphs and demonstrates it is as effective as the 3-WL graph isomorphism test and achieves

better performance on multiple datasets [234].

- GraphTrans utilises a global permutation-invariant transformer to learn long-range contextual information after standard GNN [203].

- Graph MLP-Mixer, state of the art, proposes to transform graphs into patches and then apply GNN and transformer for downstream learning tasks [3].

6.4.4 Results and Discussion

We compare in the table the results reported in [3] with our own (average result out of 4 runs) and demonstrate that our framework outperforms the best results in baselines in three datasets. Notably, on Peptides-func, a large-scale dataset of more than 2 million nodes, our proposed method outperforms the state-of-the-art with an average 0.005 precision improvement. On the MolHIV dataset, our method also improve 0.01 ROCAUC. Overall, results show that our method either outperforms the previous state-of-the-art or is competitive, demonstrating its effectiveness.

Overall, results show that our method either outperforms the previous state-of-the-art or is competitive, demonstrating its effectiveness.

Table 6.2: Comparison of our Model to Baseline methods

Methods	ZINC	Peptides-func	MolHIV	CIFAR 10
Metric	MAE	Precision	ROCAUC	Accuracy
GCN [35]	0.1952±0.0057	0.6328±0.0086	0.7813±0.0081	0.5423±0.0056
GINE [121]	0.1072±0.0037	0.6405±0.0077	0.7885±0.0034	0.6131±0.0035
SUN [234]	0.084±0.002	0.6730±0.0078	0.8003±0.0055	-
GraphTrans [203]	0.1230±0.0018	0.6313±0.0039	0.7884±0.0104	0.6809±0.0020
Graphormer [63]	0.122±0.006	-	0.805±0.004	-
K-Subtree SAT [2]	0.102±0.005	-	-	-
Graph MLP-Mixer [3]	0.0733±0.0014	0.6970±0.0054	0.7997±0.0102	**0.7396**±0.0033
RCPP	**0.0724±0.0009**	**0.7029±0.0043**	**0.8121±0.0064**	0.7383±0.0027

6.4.5 Ablation Study

We compare the performance of our RCPP framework and its separate components in Table 6.3 on the ZINC, Molhiv, Peptides-func and CIFAR 10 datasets. The presented averaged results over 4 runs demonstrate the improvement of each module. Specifically, we compare the performance of considering Ricci curvature in contrastive learning (RCL) or not (CL). In Table 6.3, -CL, -RC, -PE and -PerformerMixer denote the RCPP framework without contrastive learning, using contrastive learning without Ricci curvature consideration, without positional encoding and PerformerMixer, respectively. The PerformerMixer is the most crucial module for capturing comprehensive structural information in all four datasets. Position encoding is also important for graph representation learning performance, particularly in the ZINC dataset. One possible reason is that most nodes receive a unique node representation and molecules will have different random walk position

encodings [16]. Considering Ricci curvature in contrastive learning also improves an average 0.01 precision, ROCAUC and accuracy on Peptides-func and MolHIV datasets. As explained in section 6.4.1, we do not consider Ricci curvature information for CIFAR 10 dataset, as it is constructed from an image dataset by connecting eight nearest neighbours, which lacks a manifold structure. However, its competitive performance still demonstrates the effectiveness of our RCPP framework.

Table 6.3: An ablation study of RCPP on four datasets

Methods	ZINC	Peptides-func	MolHIV	CIFAR 10
Metric	MAE	Precision	ROCAUC	Accurcay
-CL	0.0771±0.0019	0.6889±0.0073	0.7921±0.0127	0.7015±0.0051
-RC	0.0741±0.0030	0.6978 ±0.0061	0.8084 ±0.0064	-
-PE	0.1129 ±0.0090	0.6885 ±0.0054	0.7916 ±0.0116	0.6919 ±0.0084
-PerformerMixer	0.0971±0.0036	0.6846±0.0111	0.6849±0.0050	0.6672±0.0192
RCPP	**0.0724±0.0009**	**0.7027±0.0043**	**0.8121±0.0064**	**0.7383±0.0027**

6.5 Position Information Study

We present the importance of assigning node and patch positional encodings in Tables 6.4 and 6.5. The ratio is calculated by dividing the node pe weight by the patch pe weight to evaluate the relative importance of node to patch position information. The optimal position encoding weights are approximately 1 for both node and patch position encodings for all four datasets. According to the ratio, in real-world graph datasets, ZINC, Peptides-func and MolHIV, node position encodings are slightly more important than patch position encodings, while in the image dataset, CIFAR

10, the patch position encoding has a slightly higher influence compared to the node pe weight. This observation aligns with the intuition that in graph structured data, nodes represent individual entities and their positions often carry specific semantic meanings. In image datasets, the spatial arrangement of patches plays a crucial role in capturing the global context and preserving the spatial relationships between different image regions, making patch position encoding slightly more critical for learning meaningful representations.

Table 6.4: Relative importance of position information

Datasets	ZINC	Peptides-fun	MolHIV	CIFAR 10
Node PE	0.9995	0.9999	1.0000	0.9998
Patch PE	0.9990	0.9908	1.0000	1.0000
Ratio	1.0005	1.0092	1.0000	0.9998

Table 6.5: Performance with different position information setting

Metric	MAE	Precision	ROCAUC	Accuracy
0.5 PE	0.0738±0.0021	0.6938 ±0.0078	0.7974±0.0084	0.6979±0.0047
1.0 PE	**0.0724±0.0009**	**0.7027±0.0043**	**0.8121±0.0064**	**0.7383±0.0027**
2.0 PE	0.0817±0.0026	0.6882 ±0.0091	0.7911±0.0091	0.7094±0.0026

We also analyse the performances of RCPP with different weights of position encoding and find that reducing the weight of position encoding yields better model performance compared to amplifying its influence in real-world graph datasets. While in image datasets, the opposite conclusion is once again drawn. However, the optimal performance is achieved when the weight of position encoding is set to

1.0, without any reduction or amplification of its influence. This finding suggests that the inherent position information should not be overly diminished or exaggerated in order to attain the best model performance.

6.6 Expressivity Study

The message passing based GNNs have a limitation in their capacity to distinguish non-isomorphic graphs, which has been examined through the Weisfeiler-Leman graph isomorphism test [38] based on colour refinement. In response to this, [39] then propose a general class of k-WL-GNNs which are able to represent any class of k-WL graphs universally. However, these models come with memory and speed complexities of $O(N^k)$, where N represents the number of nodes. This k-WL test is widely employed to evaluate the expressive power of GNNs [40]. However, directly comparing the proposed neural network with the Weisfeiler-Lehman test is challenging due to the nonlocal manner in which information is transferred between layers [3]. Therefore we conduct a comparative analysis with other existing methods that are unable to pass the k-WL test to demonstrate the effectiveness of RCPP. CSL [71] consists of 150 4-regular graphs, categorised into 10 classes to test GNNs expressivity, which cannot be distinguished by a 1-WL isomorphism test [3]. SR25 [72] includes 15 strongly regular graphs, and each is a different class with 25 nodes. A 3-WL test cannot differentiate the 15 graphs. The experimental results demonstrate that our model enhanced expressive power significantly across all three datasets, while conventional MP-GNNs methods couldn't accomplish as evidenced in Table 6.6. EXP [73] has 600 pairs of non-isomorphic graphs that both 1-WL and 2-WL

tests cannot differentiate [3].

Table 6.6: Average accuracy of expressive power on three simulation datasets. Some results are taken from [3]

Models	**CSL**	**SR25**	**EXP**
GCN	10.00±0.00	6.67±0.00	51.90±1.96
GatedGCN	10.00±0.00	6.67±0.00	51.73±1.65
GINE	10.00±0.00	6.67±0.00	50.69±1.39
CraphTrans	10.00±0.00	6.67±0.00	52.35±2.32
RCPP	100.00±0.00	100.00±0.00	100.00±0.00

6.7 Complexity Analysis

For a graph $G(V, E)$ where a number of nodes and edges are V and E, the METIS operation takes $O(E)$ time complexity. In the Ricci curvature consideration step, the time complexity is $(O(|V| + |E|)$ in the Ricci curvature computation step. For contrastive learning, we iterate over the top z groups and calculate the pairwise distance between positive and number of k negative samples in each group, which results in a linear time complexity of O (number of groups $\times$ group size $\times$ number of negative samples). The number of groups and negative samples are all set fixed ($z = 32$, k =3), and when the group size (number of nodes) lies within a constant range, shown in Table 6.7, it exhibits linear time complexity. The time complexity in the base GNN is $O(|V| + |E|)$. In the PerformerMixer, time complexity is $O(P)$(MLP-Mixer) + $O(Lrd)$(PerformerMixer) = $O(P + Lrd)$, where P is the

number of partitioned patches and the time complexity of PerformerMixer has been explained in the section 6.3.4.

Table 6.7: Statistics of datasets in graph patches, cited from [3]

Datasets	Patch	Min	Max	Avg
ZINC	32	2	7	3.15
Peptides-func	32	1	20	7.08
MolHIV	32	1	13	3.27
CIFAR10	32	10	35	17.2

6.8 Potential Future Work

There are several promising future research directions can be investigated:

1. How to further enhance the scalability of graph structural learning for large-scale graphs, which consists of millions or billions of nodes and edges, instead of such partition method, in which case global information may be overlooked.

2. How to partition the graph into a flexible number of patches according to datasets rather than predefined the number of clusters. Additionally, the partitioned patch size is also fixed in this study, and the Google research team recently addressed the limitation by proposing a new flexible ViT model [235], which can be integrated with our study.

3. The standard message passing mechanism in the GNN encoder pass information of different types of edges to nodes at the same distance. However, in molecule datasets, different types of edges can represent *single*, *double* and

aromatic bonds, and the information should be passed to nodes adaptively instead of universally to all nodes at the same distance. This limitation of message passing mechanism may be investigated further [236].

6.9 Summary

This Chapter introduces a more general but effective framework, RCPP, for graph deep learning. Although we convert graphs into patches similar to [3], the difference between existing work [3] and the proposed framework RCPP are the following three components: *Ricci Curvature Consideration, Contrastive Patch Embedding* and *PerformerMixer*, as illustrated in Figure 6.3.

Firstly, we comprehensively consider the position information with random walk structural node position encoding and relative patch position encoding. Then Ricci curvature is calculated and updated for encoding the unique geometric structural knowledge (edge-related information) to avoid the potential information loss in the subsequent contrastive sampling within each molecule.

Contrastive learning is implemented for different patches and emphasises negative samples, which are difficult to discriminate from positive samples, by updating them according to the distance between negative and positive samples. This step also considers the impacts of negative Ricci curvature in message passing and addresses the over-squashing problem implicitly without changing the graph structure. This enables RCPP to learn, we argue, more robust representations than obtained in prior works and render it able to differentiate based on the differences or similarities of the underlying structure of patches.

Our method successfully incorporates comprehensive structural *local, global and relative* and unique information with linear complexity, making it more efficient than previous works that used GNN with standard transformers. We also analyse the relative importance of node and patch position encoding in different datasets and explore if the influence of position information should be reduced or amplified to obtain optimal performance. Experiments on four datasets validate that our method can enhance the performance of existing GNN models significantly with linear computation complexity and attain 3-WL expressive power.

Finally, this Chapter also suggests a new perspective on graph patch learning for broader applications. Graphs in other domains, such as social networks, finance and NLP can also be studied by a similar method with better positional information learning and fewer over-squashing problem.

CHAPTER 7

Conclusion

In this dissertation, we navigate the topic of graph representation learning to challenges in GNNs and tackle them in different applications. This thesis aims to enhance the overall representation learning ability of Graph Neural Networks (GNNs) by developing methods that are more robust, expressive, effective, and efficient. In the following sections, the main contributions, which were presented in detail in the previous chapters, are summarised, and potential directions for future research are discussed.

7.1 Summary of Contributions

In the realm of graph representation learning, GNNs have emerged as the predominant standard due to their exceptional performance and flexibility. However, several

challenges, including robustness, heterogeneity and structural learning (proposed in Section 1.2) remain to be investigated more. The main contribution of this thesis is to address the above problems by investigating the effectiveness of GNN models and extending them with various learning methods in different scenarios (Chapters 3 - 6). This thesis identified the limitations of existing models in graph representation learning and explored the corresponding solutions for the three research questions, respectively. Details are summarised as follows:

For **research question 1** that *when dealing with limited available graph data, how to enhance the resilience of graph neural networks in representation learning and consider uncertainties,* we propose a Bayesian GAT method by using a non-parametric graph inference technique in Chapter 3. The initial results proved the need to consider the uncertainty of input graphic data for representation learning. The performance demonstrates that the proposed model is competitive when there are only a few labels of the nodes known. In addition, the construction of the model is motivated by the Bayesian neural network and can also be viewed under the framework of probabilistic modelling and variational inference, which offers a new potential research direction that could be explored in future semi-supervised learning task with a consideration of uncertainty.

To answer **research question 2** that *How can we accommodate the diverse nature of graphs, characterized by a variety of edges, features, and attributes, within graph neural networks,* we propose to construct heterogeneous graphs to encode

various kinds of information on short text classification tasks in Chapter 4, based on [56]. To enrich the text graph, potential topics are generated by the LDA model and entities (keywords) are mapped to Wikipedia from short texts respectively. Entities, short texts and topics are represented as three types of nodes, and the edge connected with each two of them also has different properties. A heterogeneous graph attention model extended with neighbouring contrastive learning is deployed for the text graphs. In addition to the base method, we consider the impacts of different edge relations among objects and extensive experiments illustrate that the proposed method effectively learns rich contextual information among different types of entities and thus enhances the heterogeneity of the existing model, particularly when there are limited labelled data. We also validate that neighbouring contrastive learning can enhance robustness by comparing it with the proposed BGAT method on citation network datasets, which combines these two research questions as introduced in Chapter 1.2.

Research question 3 is to study *how to learn better structural components, such as extracting relations and understanding position information.* Regarding the relation extraction problem, we propose a novel ensemble learning consisting of hypergraph convolution and adversarial training, which is applied to predict stock price movement in Chapter 5. We point out that existing work for stock movement prediction suffers from insufficiently capturing both group-wise and pairwise relations of relevant information rather than solely historical price features, leading to a weak generalisation ability due to the stochastic characteristics of stocks. Address-

ing these problems, we propose a novel ensemble learning framework, MONEY, to better assist investors in predicting future trends of stocks. To effectively capture *pairwise information* of industry, a graph convolution network is applied before RNN models.

To capture the *group level information* of both industry and fund-holding, a hypergraph convolution network is implemented after the GRU model with a temporal attention layer. Adversarial training is introduced before the final prediction layer in model (B), shown in Figure 5.2, to simulate the *stochastic movement* during training. The ensemble learning allows model A and model B model to complement each other, keeping the benefits of learning better about these relations and being more robust. All components are jointly trained on real-world stock market datasets. The model significantly outperforms the state-of-the-art for most indicators and provides a much more stable performance, particularly when facing a bear market.

Addressing the position information learning problem, we propose a novel and effective framework, *Ricci curvature Contrastive Patch Performer* (RCPP) in Chapter 6, to learn more graph structure information. Given the challenge of long-range dependencies in GNNs with a transformer, we convert the input graph into a predefined n number of patches using the metis algorithm, where nodes in each patch are closely linked. The position information of nodes and patches is considered comprehensively and we analyse the influences of different scales of position encodings. To avoid information loss during clustering, edges connected with two nodes in different patches will be kept in both patches. Addressing the over-squashing problem, we consider the influence of Ricci curvature that edges with negative curvature in

different patches without changing graph structure directly using contrastive learning. Since overlapping edges from the partition process are preserved in patches, contrastive patch learning implements learn more underlying unique information about each patch by updating the representation of the negative samples in the same batch. A standard GNN model is deployed later to generate fixed-length vectors for patches, followed by a Performer module with linear complexity. The RCPP framework leverages the advantages of GNN, Transformer and contrastive learning to capture more underlying structural information and alleviate over-squash problems. Therefore, RCPP enhances the expressive power of the graph representation learning model and experiments on real-world datasets demonstrate the effectiveness of the proposed method in graph structure learning.

7.2 Discussion

This thesis aims to enhance the expressive power of Graph Neural Networks (GNNs) in graph representation learning by addressing three key perspectives: *robustness* (Chapter 3), *heterogeneity* (Chapter 4), and *structural learning* (Chapters 5 and 6).

The choice of distinct datasets and domains for each research question is deliberate. Each selected dataset is considered a benchmark for its respective application, be it citation networks, text classification, financial investment or graph structure learning. More crucially, these datasets exhibit the challenges that the respective research questions aim to tackle. For instance, citation networks datasets like Cora, CiteSeer and Pubmed face robustness issues; and text classification datasets, such as

AGNews, MR, Twitter and Ohsumed, encounter challenges related to heterogeneity. The China A share market dataset is chosen for financial investment, due to its suitability for exploring complex relation capturing; while the molecule datasets, like ZINC, MolHIV and Peptides-func, are appropriate for graph structure and position information learning.

While the diverse dataset selection allowed for a thorough examination of each solution's efficacy in its most relevant context, we acknowledge the limitation of not evaluating all research problems within a single domain, for instance, that of social recommendations. Although each aspect is individually addressed in this thesis, it is important to note that combining the proposed solutions when necessary is a possibility, albeit not explored in this work.

In practical scenarios, multiple challenges may arise simultaneously. For instance, in tasks like social recommendation where graph structures are susceptible to noise and unnoticed malicious perturbations [237], particularly where only limited labeled data is available alongside auxiliary data of diverse types. In such cases, the proposed BGAT method from Chapter 3 and the neighboring contrastive learning with an extended heterogeneous graph neural network introduced in Chapter 4 can be leveraged to generate more robust and expressive embeddings. Additionally, the MONEY model proposed in Chapter 5 can be adapted to capture complex user-item interactions and evolving preferences within social networks. By combining these approaches, a more accurate and comprehensive understanding of the social recommendation problem can be achieved.

Another application domain is drug discovery, such as pharmacological property

estimation (e.g. toxicity), where the RCPP method developed in Chapter 6 can be directly applied to explore the subtle patterns about molecular structures. Additionally, the models proposed in Chapters 4 and 5 allow GNNs to model diverse molecule types and their interactions, enhancing prediction accuracy by capturing complex relations and interactions among atoms. The generative BGAT model from Chapter 3 can also be adapted to improve prediction robustness by accounting for uncertainties.

7.3 Limitations of Study

This thesis acknowledges certain research limitations influenced by factors such as time constraints, budgetary limitations, availability of research equipment, and unanticipated obstacles. These factors will be discussed from both experimental and application perspectives.

At the experimental level, various methodologies have been proposed to address the research questions. One restriction within this work is the nature of the datasets utilized. As discussed in Chapter 2.5, domain-specific datasets with specific settings have been employed in Chapters 3 to 6 to align with the unique characteristics of each research question. For example, the investigation of capturing complex relationships relies on the China A-share market dataset without including data from other national markets in Chapter 5. Furthermore, while this thesis aims to enhance the learning ability of Graph Neural Networks (GNNs) in graph learning tasks, with a focus on aspects including robustness, heterogeneity and structural learning, other challenges such as dynamicity and scalability have received relatively

less attention in this study.

From the application level, this thesis addresses three distinct research challenges and demonstrates the effectiveness of the proposed methodologies through tasks including citation networks, natural language processing, finance, and molecule learning. However, it is important to note that the proposed methods can be applied to a wide range of other tasks, such as traffic control, recommendation systems, social network analysis, drug discovery [238], and various computer vision tasks [17]. These areas provide avenues for future investigation and exploration.

By acknowledging these limitations and focusing on specific research perspectives, this PhD thesis makes valuable contributions to graph representation learning and sets the stage for further study in addressing the broader challenges associated with GNNs.

7.4 Future Research Directions

There is a keen interest in exploring various exciting facets of graph representation learning in future work. In addition to the challenges of robustness, heterogeneity and structure learning addressed in this thesis, there are other significant future research avenues in graph representation learning. These include the increasingly crucial issues of fairness and interpretability

Fairness Fairness aims to ensure that protected features like race or gender do not influence outcomes of models [239]. In graph representation learning, fairness is a growing concern, as models are prone to inheriting biases from real-world datasets, model architectures or performance-driven design choices [240]. Achieving fairness

requires meticulous design of both model and data, and it remains an open problem in the field.

Interpretability Despite the wide application of GNNs, the lack of interpretability hampers trust and limits the potential for further improvement [241]. As ethical concerns and legal requirements for interpretability rise, researchers are exploring various approaches from ad-hoc explanations to model architecture design, while a consensus on best practices is yet to be reached.

Furthermore, graph representation learning serves as a critical stepping stone towards achieving Artificial General Intelligence (AGI). By efficiently capturing complex relationships and knowledge in a machine-understandable format, it paves the way for advanced reasoning and inference capabilities. This level of intelligence entails understanding or learning tasks akin to human beings, potentially incorporating a broader range of logic and real-world knowledge. By delving into these areas, researchers will be able to push the boundaries of GNNs capabilities and unlock their potential for achieving AGI-like capabilities. The aspiration can also extends to developing scalable novel GNNs across various industries to address practical challenges. Other key areas for future exploration in improving GNNs encompass the following aspects.

Neural Logic Neural logic is a framework which combines the strengths of symbolic logic and neural network to develop more powerful models for artificial intelligence [242]. Symbolic logic is an application of the formal approach of mathematics to logic and is suitable for making inferences and arguments concerning abstract notions [243]. Additionally, neural logic models can offer more explicit

and understandable justifications for their judgements and predictions by introducing symbolic thinking into GNNs, which is essential for vital industries like drug development and diagnosis.

Knowledge Graph Reasoning Knowledge graphs store complex graph structured knowledge in an organised manner and thus enable machines to reason and learn from knowledge [156, 244–246]. Furthermore, knowledge graphs offer a clear representation of knowledge that is simple for humans to understand, greatly enhancing the explainability of predicted outcomes [247–249]. A knowledge network may also have billions of nodes and edges. Thus, a compelling ambition is to create more scalable and effective methods, ones that are capable of not only capturing intricate, high-order graph structures but also unlocking the potential to apply complex reasoning.

Reinforcement Learning Reinforcement learning allows machines to learn from experience by receiving rewards or penalties based on actions of agents in complex and dynamic environments with a giving policy [250, 251]. It can solve different tasks without prior knowledge and is regarded as a feasible path to AGI [252]. Owing to its superior ability to generalise without human interference, the fusion of reinforcement learning with GNNs is seen as a potential game-changer in the realm of graph representation learning.

Bibliography

[1] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, 2022. (document), 2.6, 2.4

[2] D. Chen, L. O'Bray, and K. Borgwardt, "Structure-aware transformer for graph representation learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 3469–3489. (document), 1.1, 6.1, 6.1, 6.2.2, 6.4.3, 6.2

[3] X. He, B. Hooi, T. Laurent, A. Perold, Y. LeCun, and X. Bresson, "A generalization of vit/mlp-mixer to graphs," *arXiv preprint arXiv:2212.13350*, 2022. (document), 2.2.2, 2.1, 6.1, 2, 6.2.2, 6.3, 6.3, 6.3.3, 6.3.3, 6.3.4, 6.4.1, 6.4.2, 6.4.3, 6.4.4, 6.2, 6.6, 6.6, 6.7, 6.9

[4] C. Manning and H. Schutze, *Foundations of statistical natural language processing*. MIT press, 1999. 1.1

[5] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020. 1.1, 3.1

[6] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. 1.1, 1.1, 2.2.2

[7] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-bert: Only attention is needed for learning graph representations," *arXiv preprint arXiv:2001.05140*, 2020. 1.1, 6.2.2

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020. 1.1, 2.2, 2.2, 3.2, 4.1

[9] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, and L. Wang, "Deep graph structure learning for robust representations: A survey," *arXiv preprint arXiv:2103.03036*, vol. 14, 2021. 1.1

[10] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "Ogb-lsc: A large-scale challenge for machine learning on graphs," *arXiv preprint arXiv:2103.09430*, 2021. 1.1, 6.1

[11] A. Bojchevski and S. Günnemann, "Certifiable robustness to graph perturbations," *arXiv preprint arXiv:1910.14356*, 2019. 1.1

[12] X. Li, L. Sun, M. Ling, and Y. Peng, "A survey of graph neural network based recommendation in social networks," *Neurocomputing*, p. 126441, 2023. 1.1

[13] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 66–74. 1.1

[14] Y. Qin, X. Wang, Z. Zhang, and W. Zhu, "Graph differentiable architecture search with structure learning," *Advances in neural information processing systems*, vol. 34, pp. 16 860–16 872, 2021. 1.1

[15] Z. Zhang, Y. Feng, S. Ying, and Y. Gao, "Deep hypergraph structure learning," *arXiv preprint arXiv:2208.12547*, 2022. 1.1

[16] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," *arXiv preprint arXiv:2110.07875*, 2021. 1.1, 2.4, 2.4, 6.1, 6.2.2, 6.3.1, 6.4.5

[17] C. Chen, Y. Wu, Q. Dai, H.-Y. Zhou, M. Xu, S. Yang, X. Han, and Y. Yu, "A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective," *arXiv preprint arXiv:2209.13232*, 2022. 1.1, 7.3

[18] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," *arXiv preprint arXiv:2006.05205*, 2020. 1.1, 6.3.2

[19] Y. Li, J. Yin, and L. Chen, "Informative pseudo-labeling for graph neural networks with few labels," *Data Mining and Knowledge Discovery*, pp. 1–27, 2022. 1.2

[20] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of the web conference 2020*, 2020, pp. 2704–2710. 1.2, 2.1

[21] W. Li, L. Ni, J. Wang, and C. Wang, "Collaborative representation learning for nodes and relations via heterogeneous graph neural network," *Knowledge-Based Systems*, vol. 255, p. 109673, 2022. 1.2

[22] Z. Sun, A. Harit, J. Yu, A. I. Cristea, and N. Al Moubayed, "A generative bayesian graph attention network for semi-supervised classification on scarce data," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–7. 1.5, 4.1

[23] Z. Sun, A. Harit, A. I. Cristea, J. Yu, L. Shi, and N. Al Moubayed, "Contrastive learning with heterogeneous graph attention networks on short text classification," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–6. 1.5, 5.6, 6.1

[24] Z. Sun, A. Harit, A. I. Cristea, J. Wang, and P. Lio, "Money: Ensemble learning for stock price movement prediction via a convolutional network with adversarial hypergraph model," *AI Open*, vol. 4, pp. 165–174, 2023. 1.5

[25] S. Khoshraftar and A. An, "A survey on graph representation learning methods," *arXiv preprint arXiv:2204.01855*, 2022. 2

[26] P. Zhang and G. Chartrand, *Introduction to graph theory*. Tata McGraw-Hill, 2006. 2.1, 2.1

[27] H. Singh and R. Sharma, "Role of adjacency matrix & adjacency list in graph theory," *International Journal of Computers & Technology*, vol. 3, no. 1, pp. 179–183, 2012. 2.1

[28] R. Merris, "Degree maximal graphs are laplacian integral," *Linear algebra and its applications*, vol. 199, pp. 381–389, 1994. 2.1

[29] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE transactions on knowledge and data engineering*, vol. 30, no. 9, pp. 1616–1637, 2018. 2.1

[30] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proceedings of The Web Conference 2020*, 2020, pp. 2331–2341. 2.1

[31] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE, 2005, pp. 729–734. 2.2, 3.2

[32] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008. 2.2, 3.2

[33] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022. 2.2, 6.1

[34] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 5363–5370. 2.2

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016. 2.2.1, 2.2.1, 2.1, 3.1, 3.5.3, 4.1, 6.3.2, 6.4.3, 6.2

[36] J. Ye, J. Zhao, K. Ye, and C. Xu, "Multi-graph convolutional network for relationship-driven stock movement prediction," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 6702–6709. 2.2.1, 2.5, 5.1, 5.2.1, 5.4.2

[37] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017. 2.2.2, 3.2

[38] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018. 2.2.2, 6.6

[39] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," *arXiv preprint arXiv:1812.09902*, 2018. 2.2.2, 6.6

[40] N. T. Huang and S. Villar, "A short tutorial on the weisfeiler-lehman test and its variants," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8533–8537. 2.2.2, 6.6

[41] A. Khan, S. AlBarri, and M. A. Manzoor, "Contrastive self-supervised learning: a survey on different architectures," in *2022 2nd International Conference on Artificial Intelligence (ICAI)*. IEEE, 2022, pp. 1–6. 2.3, 6.2.1

[42] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1725–1735. 2.3, 2.3, 6.2.1

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008. 2.4, 2.4, 6.1

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 2.4

[45] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016. 2.4

[46] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 3159–3166. 2.4

[47] X. Liu, H.-F. Yu, I. Dhillon, and C.-J. Hsieh, "Learning to encode position for transformer with continuous dynamical model," in *International conference on machine learning*. PMLR, 2020, pp. 6327–6335. 2.4

[48] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020. 2.4, 2.4, 2.1, 2.5, 6.4.1

[49] P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance encoding: Design provably more powerful neural networks for graph representation learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4465–4478, 2020. 2.4, 2.4

[50] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," *arXiv preprint arXiv:2012.09699*, 2020. 2.4, 6.1, 6.2.2

[51] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007. 2.4

[52] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 618–21 629, 2021. 2.4, 6.1

[53] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017. 2.1, 3.2, 3.3.1, 3.3.1, 3.4.2, 3.5.2, 3.5.3

[54] Y. Zhang, S. Pal, M. Coates, and D. Ustebay, "Bayesian graph convolutional neural networks for semi-supervised classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5829–5836. 2.1, 3.1, 3.2, 3.4.2, 3.4.2, 3.5.2, 3.5.3

[55] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 7370–7377. 2.1, 2.5, 4.4.1, 4.4.3, 4.5

[56] L. Hu, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 4821–4830. 2.1, 4.1, 4.2.1, 4.3, 4.3.1, 4.3.1, 4.3.2, 4.3.3, 4.3.5, 4.4.1, 4.4.2, 4.4.3, 6.1, 7.1

[57] T. Yang, L. Hu, C. Shi, H. Ji, X. Li, and L. Nie, "Hgat: Heterogeneous graph attention networks for semi-supervised short text classification," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 3, pp. 1–29, 2021. 2.1, 4.2.1, 4.3.5, 4.4.3

[58] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond," *arXiv e-prints*, pp. arXiv–2004, 2020. 2.1

[59] C. Cui, X. Li, J. Du, C. Zhang, X. Nie, M. Wang, and Y. Yin, "Temporal-relational hypergraph tri-attention networks for stock trend prediction," *arXiv preprint arXiv:2107.14033*, 2021. 2.1, 5.1, 5.1, 5.2.1, 5.3.1, 5.3.1, 5.3.2, 5.3.4, 5.3.5, 5.4.1, 5.4.2, 5.4.3, 5.5.2, 5.5.2, 5.6

[60] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 657–668, 2022. 2.1

[61] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *arXiv preprint arXiv:2205.12454*, 2022. 2.1, 6.1, 6.1, 6.2.2

[62] V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini, "Long range graph benchmark," *arXiv preprint arXiv:2206.08164*, 2022. 2.1, 2.5, 6.4.1

[63] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform badly for graph representation?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 877–28 888, 2021. 2.1, 6.1, 6.4.3, 6.2

[64] C. Bodnar, F. Frasca, N. Otter, Y. Wang, P. Lio, G. F. Montufar, and M. Bronstein, "Weisfeiler and lehman go cellular: Cw networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2625–2640, 2021. 2.1

[65] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang, "How powerful are k-hop message passing graph neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 4776–4790, 2022. 2.1

[66] M. H. Dupty, Y. Dong, and W. S. Lee, "Pf-gnn: Differentiable particle filtering based approximation of universal graph representations," in *International Conference on Learning Representations*, 2021. 2.1

[67] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48. 2.5

[68] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015. 2.5, 4.4.1

[69] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," *arXiv preprint cs/0506075*, 2005. 2.5, 4.4.1

[70] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020. 2.5, 6.4.1

[71] R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, "Relational pooling for graph representations," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4663–4673. 2.5, 6.6

[72] M. Balcilar, P. Héroux, B. Gauzere, P. Vasseur, S. Adam, and P. Honeine, "Breaking the limits of message passing graph neural networks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 599–608. 2.5, 6.6

[73] R. Abboud, I. I. Ceylan, M. Grohe, and T. Lukasiewicz, "The surprising power of graph neural networks with random node initialization," *arXiv preprint arXiv:2010.01179*, 2020. 2.5, 6.6

[74] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017. 3.1

[75] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1416–1424. 3.1

[76] J. Ma, W. Tang, J. Zhu, and Q. Mei, "A flexible generative framework for graph-based semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 3281–3290. 3.1, 3.2

[77] Y. C. Ng, N. Colombo, and R. Silva, "Bayesian semi-supervised learning with graph gaussian processes," in *Advances in Neural Information Processing Systems*, 2018, pp. 1683–1694. 3.2

[78] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059. 3.2, 3.3.2, 3.4.2

[79] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," *arXiv preprint arXiv:1803.07294*, 2018. 3.3.1

[80] N. Seedat and C. Kanan, "Towards calibrated and scalable uncertainty representations for neural networks," *arXiv preprint arXiv:1911.00104*, 2019. 3.3.2

[81] R. van de Schoot, S. Depaoli, R. King, B. Kramer, K. Märtens, M. G. Tadesse, M. Vannucci, A. Gelman, D. Veen, J. Willemsen *et al.*, "Bayesian statistics and modelling," *Nature Reviews Methods Primers*, vol. 1, no. 1, pp. 1–26, 2021. 3.3.2

[82] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *Journal of machine learning research*, vol. 9, no. Sep, pp. 1981–2014, 2008. 3.3.3, 3.4.2, 3.4.2

[83] Y. Zhang and A. Ramesh, "Struct-mmsb: Mixed membership stochastic blockmodels with interpretable structured priors," *arXiv preprint arXiv:2002.09523*, 2020. 3.3.3

[84] W. Huang, Y. Liu, Y. Chen *et al.*, "Mixed membership stochastic blockmodels for heterogeneous networks," *Bayesian Analysis*, vol. 15, no. 3, pp. 711–736, 2020. 3.3.3, 3.3.3

[85] S. Sun, C. Chen, and L. Carin, "Learning structured weight uncertainty in bayesian neural networks," in *Artificial Intelligence and Statistics*, 2017, pp. 1283–1292. 3.4.2

[86] C. Louizos and M. Welling, "Multiplicative normalizing flows for variational bayesian neural networks," *arXiv preprint arXiv:1703.01961*, 2017. 3.4.2

[87] R. M. Neal, *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993. 3.4.2

[88] A. Korattikara Balan, V. Rathod, K. P. Murphy, and M. Welling, "Bayesian dark knowledge," *Advances in Neural Information Processing Systems*, vol. 28, pp. 3438–3446, 2015. 3.4.2

[89] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. 3.4.2

[90] L. Peng, L. Carvalho *et al.*, "Bayesian degree-corrected stochastic blockmodels for community detection," *Electronic Journal of Statistics*, vol. 10, no. 2, pp. 2746–2779, 2016. 3.4.2

[91] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008. 3.5.1

[92] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844–3852, 2016. 3.5.3

[93] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019. 3.6

[94] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, "A survey on text classification: From shallow to deep learning," *arXiv preprint arXiv:2008.00364*, 2020. 4.1

[95] Z. Wang, X. Liu, P. Yang, S. Liu, and Z. Wang, "Cross-lingual text classification with heterogeneous graph neural network," *arXiv preprint arXiv:2105.11246*, 2021. 4.1

[96] R. Ragesh, S. Sellamanickam, A. Iyer, R. Bairi, and V. Lingam, "Hetegcn: heterogeneous graph convolutional networks for text classification," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 860–868. 4.1

[97] Y. Yang, Z. Guan, J. Li, W. Zhao, J. Cui, and Q. Wang, "Interpretable and efficient heterogeneous graph convolutional network," *arXiv preprint arXiv:2005.13183*, 2020. 4.1

[98] Y. Liu, R. Guan, F. Giunchiglia, Y. Liang, and X. Feng, "Deep attention diffusion graph neural networks for text classification," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 8142–8152. 4.1, 4.2.1

[99] Y. Hu, H. You, Z. Wang, Z. Wang, E. Zhou, and Y. Gao, "Graph-mlp: Node classification without message passing in graph," *arXiv preprint arXiv:2106.04051*, 2021. 4.1, 4.3, 4.3.4, 4.4.2

[100] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1150–1160. 4.1, 6.2.1

[101] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," *arXiv preprint arXiv:2011.01403*, 2020. 4.1, 4.2.2

[102] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003. 4.2.1, 4.3.1, 4.4.3

[103] Y. Chen, "Convolutional neural network for sentence classification," Master's thesis, University of Waterloo, 2015. 4.2.1

[104] P. Liu, X. Qiu, X. Chen, S. Wu, and X.-J. Huang, "Multi-timescale long short-term memory neural network for modelling sentences and documents," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 2326–2335. 4.2.1

[105] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032. 4.2.1, 4.3.2, 4.4.3

[106] K. Ding, J. Wang, J. Li, D. Li, and H. Liu, "Be more with less: Hypergraph attention networks for inductive text classification," *arXiv preprint arXiv:2011.00387*, 2020. 4.2.1

[107] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2021. 4.2.2

[108] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021. 4.2.2

[109] W. Falcon and K. Cho, "A framework for contrastive self-supervised learning and designing a new approach," *arXiv preprint arXiv:2009.00104*, 2020. 4.2.2

[110] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie, "Cert: Contrastive self-supervised learning for language understanding," *arXiv preprint arXiv:2005.12766*, 2020. 4.2.2

[111] M. Kim, J. Tack, and S. J. Hwang, "Adversarial self-supervised contrastive learning," *arXiv preprint arXiv:2006.07589*, 2020. 4.2.2

[112] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *arXiv preprint arXiv:2004.11362*, 2020. 4.2.2

[113] M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, and B. Carro, "Supervised contrastive learning over prototype-label embeddings for network intrusion detection," *Information Fusion*, vol. 79, pp. 200–228, 2022. 4.2.2

[114] L. Pan, C.-W. Hang, A. Sil, S. Potdar, and M. Yu, "Improved text classification via contrastive adversarial training," *arXiv preprint arXiv:2107.10137*, 2021. 4.2.2

[115] Z. Guo, Z. Liu, Z. Ling, S. Wang, L. Jin, and Y. Li, "Text classification by contrastive learning and cross-lingual data augmentation for alzheimer's disease detection," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 6161–6171. 4.2.2

[116] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5812–5823, 2020. 4.2.2, 6.1, 6.2.1

[117] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 2, pp. 2412–2429, 2022. 4.2.2

[118] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *arXiv preprint arXiv:1908.01000*, 2019. 4.2.2

[119] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." *ICLR (Poster)*, vol. 2, no. 3, p. 4, 2019. 4.2.2

[120] Z. Peng, Y. Dong, M. Luo, X.-M. Wu, and Q. Zheng, "Self-supervised graph representation learning via global context prediction," *arXiv preprint arXiv:2003.01604*, 2020. 4.2.2

[121] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019. 4.2.2, 6.3.3, 6.4.3, 6.2

[122] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, "Sub-graph contrast for scalable self-supervised graph representation learning," in *2020 IEEE international conference on data mining (ICDM)*. IEEE, 2020, pp. 222–231. 4.2.2

[123] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," *arXiv preprint arXiv:2006.04131*, 2020. 4.2.2

[124] ——, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, 2021, pp. 2069–2080. 4.2.2

[125] S. Thakoor, C. Tallec, M. G. Azar, M. Azabou, E. L. Dyer, R. Munos, P. Veličković, and M. Valko, "Large-scale representation learning on graphs via bootstrapping," *arXiv preprint arXiv:2102.06514*, 2021. 4.2.2

[126] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International conference on machine learning*. PMLR, 2020, pp. 4116–4126. 4.2.2

[127] Z. Tong, Y. Liang, H. Ding, Y. Dai, X. Li, and C. Wang, "Directed graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 580–19 593, 2021. 4.2.2

[128] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *Advances in neural information processing systems*, vol. 33, pp. 6827–6839, 2020. 4.2.2

[129] T. Xiao, X. Wang, A. A. Efros, and T. Darrell, "What should not be contrastive in contrastive learning," *arXiv preprint arXiv:2008.05659*, 2020. 4.2.2

[130] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988. 4.4.3

[131] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 4.4.3, 2, 6.1

[132] L. Melas-Kyriazi, "Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet," *arXiv preprint arXiv:2105.02723*, 2021. 4.6

[133] C. Yang, R. Wang, S. Yao, S. Liu, and T. Abdelzaher, "Revisiting oversmoothing in deep gcns," *arXiv preprint arXiv:2003.13663*, 2020. 4.6

[134] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, pp. 681–694, 2020. 2

[135] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020. 2

[136] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022. 2

[137] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023. 2

[138] Z. Bouraoui, J. Camacho-Collados, and S. Schockaert, "Inducing relational knowledge from bert," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 7456–7463. 2

[139] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 2020. 2

[140] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang, "Learning on large-scale text-attributed graphs via variational inference," *arXiv preprint arXiv:2210.14709*, 2022. 2

[141] C.-Y. Lee and V.-W. Soo, "Predict stock price with financial news based on recurrent convolutional neural networks," in *2017 conference on technologies and applications of artificial intelligence (TAAI)*. IEEE, 2017, pp. 160–165. 5.1

[142] H. Hong, W. Torous, and R. Valkanov, "Do industries lead stock markets?" *Journal of Financial Economics*, vol. 83, no. 2, pp. 367–396, 2007. 5.1

[143] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua, "Temporal relational ranking for stock prediction," *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 2, pp. 1–30, 2019. 5.1, 5.4.3

[144] D. Matsunaga, T. Suzumura, and T. Takahashi, "Exploring graph neural networks for stock market predictions with rolling window analysis," *arXiv preprint arXiv:1909.10660*, 2019. 5.1

[145] S. Aghabozorgi and Y. W. Teh, "Stock market co-movement assessment using a three-phase clustering method," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1301–1314, 2014. 5.1, 5.1, 5.1

[146] C. Zhang, S. Hu, Z. G. Tang, and T. H. Chan, "Re-revisiting learning on hypergraphs: confidence interval and subgradient method," in *International Conference on Machine Learning*. PMLR, 2017, pp. 4026–4034. 1

[147] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, vol. 110, p. 107637, 2021. 5.1, 1

[148] R. Sawhney, S. Agarwal, A. Wadhwa, and R. R. Shah, "Spatiotemporal hypergraph convolution network for stock movement forecasting," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 482–491. 5.1, 5.1, 5.4.3

[149] W. Qie, "Market volatility's relationship with pair-wise correlation of stocks and portfolio man-ager's performance," *Stockholm University*, 2011. 5.1

[150] R. Kim, C. H. So, M. Jeong, S. Lee, J. Kim, and J. Kang, "Hats: A hierarchical graph attention network for stock movement prediction," *arXiv preprint arXiv:1908.07999*, 2019. 5.1, 5.2.1, 5.4.3, 5.6

[151] S. Peng, "Stock forecasting using neural network with graphs," Ph.D. dissertation, University of York, 2020. 5.1

[152] J. B. Berk and J. H. Van Binsbergen, "Measuring skill in the mutual fund industry," *Journal of financial economics*, vol. 118, no. 1, pp. 1–20, 2015. 5.1, 5.3.3

[153] Y. Chen, Z. Wei, and X. Huang, "Incorporating corporation relationship via graph convolutional neural networks for stock price prediction," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1655–1658. 5.2.1, 5.4.3

[154] K. Xiong, X. Ding, L. Du, T. Liu, and B. Qin, "Heterogeneous graph knowledge enhanced stock market prediction," *AI Open*, vol. 2, pp. 168–174, 2021. 5.2.1

[155] L. Guo, H. Yin, T. Chen, X. Zhang, and K. Zheng, "Hierarchical hyperedge embedding-based representation learning for group recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 40, no. 1, pp. 1–27, 2021. 5.2.1

[156] Z. Sun, A. Harit, J. Yu, A. I. Cristea, and L. Shi, "A brief survey of deep learning approaches for learning analytics on moocs," in *Intelligent Tutoring Systems: 17th International Conference, ITS 2021, Virtual Event, June 7–11, 2021, Proceedings 17*. Springer, 2021, pp. 28–37. 5.2.1, 7.4

[157] X. Wang, W. Ma, L. Guo, H. Jiang, F. Liu, and C. Xu, "Hgnn: Hyperedge-based graph neural network for mooc course recommendation," *Information Processing & Management*, vol. 59, no. 3, p. 102938, 2022. 5.2.1

[158] M. Li, Y. Zhang, X. Li, L. Cai, and B. Yin, "Multi-view hypergraph neural networks for student academic performance prediction," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105174, 2022. 5.2.1

[159] X. Hou, K. Wang, C. Zhong, and Z. Wei, "St-trader: A spatial-temporal deep neural network for modeling stock market movement," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 5, pp. 1015–1024, 2021. 5.2.1

[160] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," *arXiv preprint arXiv:2102.01356*, 2021. 5.2.2

[161] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, "Enhancing stock movement prediction with adversarial training," *arXiv preprint arXiv:1810.09936*, 2018. 5.2.2, 5.3.5, 5.3.6, 5.3.6, 5.4.2, 5.5.3

[162] Y. Zhang, J. Li, H. Wang, and S. C. Choi, "Sentiment-guided adversarial learning for stock price prediction," *Frontiers in Applied Mathematics and Statistics*, vol. 7, p. 8, 2021. 5.2.2

[163] Y. Li, H.-N. Dai, and Z. Zheng, "Selective transfer learning with adversarial training for stock movement prediction," *Connection Science*, pp. 1–19, 2022. 5.2.2

[164] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, 2020. 5.2.3

[165] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018. 5.2.3

[166] M. Jiang, J. Liu, L. Zhang, and C. Liu, "An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms," *Physica A: Statistical Mechanics and its Applications*, vol. 541, p. 122272, 2020. 5.2.3

[167] Y. Zhao, J. Li, and L. Yu, "A deep learning ensemble approach for crude oil price forecasting," *Energy Economics*, vol. 66, pp. 9–16, 2017. 5.2.3

[168] Y. Li and Y. Pan, "A novel ensemble deep learning model for stock prediction based on stock prices and news," *International Journal of Data Science and Analytics*, pp. 1–11, 2021. 5.2.3

[169] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014. 5.3.4

[170] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1407–1418, 2018. 5.3.5

[171] M.-H. Yu and J.-L. Wu, "Ceam: A novel approach using cycle embeddings with attention mechanism for stock price prediction," in *2019 IEEE international conference on big data and smart computing (BigComp)*. IEEE, 2019, pp. 1–4. 5.3.5

[172] Y. Yu and Y.-J. Kim, "Two-dimensional attention-based lstm model for stock index prediction," *Journal of Information Processing Systems*, vol. 15, no. 5, pp. 1231–1242, 2019. 5.3.5

[173] Y. Li, L. Li, X. Zhao, T. Ma, Y. Zou, and M. Chen, "An attention-based lstm model for stock price trend prediction using limit order books," in *Journal of Physics: Conference Series*, vol. 1575. IOP Publishing, 2020, p. 012124. 5.3.5

[174] Y. Hao, H.-H. Chu, K.-Y. Ho, and K.-C. Ko, "The 52-week high and momentum in the taiwan stock market: Anchoring or recency biases?" *International Review of Economics & Finance*, vol. 43, pp. 121–138, 2016. 5.3.5

[175] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *Advances in Neural Information Processing Systems*, vol. 32, 2019. 5.3.6

[176] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014. 5.3.6

[177] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15. 5.3.7

[178] A. F. Serban, "Combining mean reversion and momentum trading strategies in foreign exchange markets," *Journal of Banking & Finance*, vol. 34, no. 11, pp. 2720–2727, 2010. 5.4.3

[179] D. M. Nelson, A. C. Pereira, and R. A. De Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International joint conference on neural networks (IJCNN)*. Ieee, 2017, pp. 1419–1426. 5.4.3

[180] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017. 5.4.3

[181] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*. Springer, 2006, pp. 1015–1021. 5.5.1

[182] B. I. Jacobs and K. N. Levy, "Disentangling equity return regularities: New insights and investment opportunities," *Financial Analysts Journal*, vol. 44, no. 3, pp. 18–43, 1988. 5.5.2

[183] W. F. Sharpe, "Mutual fund performance," *The Journal of business*, vol. 39, no. 1, pp. 119–138, 1966. 5.5.2

[184] M. Magdon-Ismail and A. F. Atiya, "Maximum drawdown," *Risk Magazine*, vol. 17, no. 10, pp. 99–102, 2004. 5.5.2

[185] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021. 5.6

[186] T. N. Kipf, "Deep learning with graph-structured representations," *NARCIS*, 2020. 5.6

[187] J. Yoo, Y. Soun, Y.-c. Park, and U. Kang, "Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2037–2045. 5.6

[188] A. Mittal and A. Goel, "Stock prediction using twitter sentiment analysis," *Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf)*, vol. 15, p. 2352, 2012. 5.6

[189] N. Jing, Z. Wu, and H. Wang, "A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction," *Expert Systems with Applications*, vol. 178, p. 115019, 2021. 5.6

[190] E. Choi, M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun, "Gram: graph-based attention model for healthcare representation learning," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 787–795. 6.1

[191] Z. Sun, A. Harit, A. I. Cristea, J. Yu, N. Al Moubayed, and L. Shi, "Is unimodal bias always bad for visual question answering? a medical domain study with dynamic attention," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 5352–5360. 6.1

[192] J. M. Thomas, A. Moallemy-Oureh, S. Beddar-Wiesing, and C. Holzhüter, "Graph neural networks designed for different graph types: A survey," *arXiv preprint arXiv:2204.03080*, 2022. 6.1

[193] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022. 6.1

[194] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7829–7833. 6.1

[195] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021. 6.1

[196] G. Mialon, D. Chen, M. Selosse, and J. Mairal, "Graphit: Encoding graph structure in transformers," *arXiv preprint arXiv:2106.05667*, 2021. 6.1, 6.2.2

[197] P. Dufter, M. Schmitt, and H. Schütze, "Position information in transformers: An overview," *Computational Linguistics*, vol. 48, no. 3, pp. 733–763, 2022. 6.1

[198] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. 6.1

[199] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," *arXiv preprint arXiv:2111.14522*, 2021. 6.1, 6.3.2, 6.3.2

[200] J. Southern, J. Wayland, M. Bronstein, and B. Rieck, "Curvature filtrations for graph generative model evaluation," *arXiv preprint arXiv:2301.12906*, 2023. 6.1, 6.3

[201] J. Cheeger and D. Gromoll, "The splitting theorem for manifolds of nonnegative ricci curvature," *Journal of Differential Geometry*, vol. 6, no. 1, pp. 119–128, 1971. 6.1

[202] K. Nguyen, N. M. Hieu, V. D. Nguyen, N. Ho, S. Osher, and T. M. Nguyen, "Revisiting over-smoothing and over-squashing using ollivier-ricci curvature," in *International Conference on Machine Learning*. PMLR, 2023, pp. 25 956–25 979. 6.1

[203] Z. Wu, P. Jain, M. Wright, A. Mirhoseini, J. E. Gonzalez, and I. Stoica, "Representing long-range context for graph neural networks with global attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 266–13 279, 2021. 6.1, 6.2.2, 6.4.3, 6.2

[204] Z. Ye, K. S. Liu, T. Ma, J. Gao, and C. Chen, "Curvature graph network," in *International conference on learning representations*, 2019. 6.1, 6.3.2, 6.3.2

[205] Y. Fang, Q. Zhang, H. Yang, X. Zhuang, S. Deng, W. Zhang, M. Qin, Z. Chen, X. Fan, and H. Chen, "Molecular contrastive learning with chemical element knowledge graph," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 3968–3976. 6.1, 6.2.1

[206] P. Morgado, I. Misra, and N. Vasconcelos, "Robust audio-visual instance discrimination," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 934–12 945. 6.1

[207] Y. Wang, R. Magar, C. Liang, and A. Barati Farimani, "Improving molecular contrastive learning via faulty negative mitigation and decomposed fragment contrast," *Journal of Chemical Information and Modeling*, vol. 62, no. 11, pp. 2713–2725, 2022. 6.1, 6.2.1, 6.3.2

[208] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24 261–24 272, 2021. 6.1, 6.3, 6.3.4

[209] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, 2020. 6.2.1

[210] P. Kumar, P. Rawat, and S. Chauhan, "Contrastive self-supervised learning: review, progress, challenges and future research directions," *International Journal of Multimedia Information Retrieval*, vol. 11, no. 4, pp. 461–488, 2022. 6.2.1

[211] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738. 6.2.1

[212] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15 750–15 758. 6.2.1

[213] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Graphcrop: Subgraph cropping for graph classification," *arXiv preprint arXiv:2009.10564*, 2020. 6.2.1

[214] Y. Wang, J. Wang, Z. Cao, and A. Barati Farimani, "Molecular contrastive learning of representations via graph neural networks," *Nature Machine Intelligence*, vol. 4, no. 3, pp. 279–287, 2022. 6.2.1, 6.3.2

[215] S. Zhang, Z. Hu, A. Subramonian, and Y. Sun, "Motif-driven contrastive learning of graph representations," *arXiv preprint arXiv:2012.12533*, 2020. 6.2.1, 6.3.2

[216] S. Li, J. Zhou, T. Xu, D. Dou, and H. Xiong, "Geomgcl: Geometric graph contrastive learning for molecular property prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 4541–4549. 6.2.1

[217] M. Sun, J. Xing, H. Wang, B. Chen, and J. Zhou, "Mocl: Contrastive learning on molecular graphs with multi-level domain knowledge," *arXiv preprint arXiv:2106.04509*, vol. 9, 2021. 6.2.1

[218] Y. Fang, H. Yang, X. Zhuang, X. Shao, X. Fan, and H. Chen, "Knowledge-aware contrastive molecular graph learning," *arXiv preprint arXiv:2103.13047*, 2021. 6.2.1

[219] E. Min, R. Chen, Y. Bian, T. Xu, K. Zhao, W. Huang, P. Zhao, J. Huang, S. Ananiadou, and Y. Rong, "Transformer for graphs: An overview from architecture perspective," *arXiv preprint arXiv:2202.08455*, 2022. 6.2.2

[220] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, 2022. 6.2.2

[221] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165. 6.2.2

[222] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020. 6.2.2

[223] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, D. Belanger, L. Colwell *et al.*, "Masked language modeling for proteins via linearly scalable long-context transformers," *arXiv preprint arXiv:2006.03555*, 2020. 6.2.2

[224] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res*, vol. 23, pp. 1–40, 2021. 6.2.2

[225] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser *et al.*, "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794*, 2020. 6.3, 6.3.4, 6.3.4

[226] A. Di Nardo, M. Di Natale, C. Giudicianni, R. Greco, and G. F. Santonastaso, "Water distribution network clustering: Graph partitioning or spectral algorithms?" in *Complex Networks & Their Applications VI: Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications)*. Springer, 2018, pp. 1197–1209. 6.3

[227] Z. Zhang, P. Cui, J. Pei, X. Wang, and W. Zhu, "Eigen-gnn: A graph structure preserving plug-in for gnns," *IEEE Transactions on Knowledge and Data Engineering*, 2021. 6.3.1

[228] S. Liu, H. Wang, W. Liu, J. Lasenby, H. Guo, and J. Tang, "Pre-training molecular graph representation with 3d geometry," *arXiv preprint arXiv:2110.07728*, 2021. 6.3.2, 6.3.3

[229] J. Zhu, Y. Xia, T. Qin, W. Zhou, H. Li, and T.-Y. Liu, "Dual-view molecule pre-training," *arXiv preprint arXiv:2106.10234*, 2021. 6.3.3

[230] Y. Bengio, Y. Lecun, and G. Hinton, "Deep learning for ai," *Communications of the ACM*, vol. 64, no. 7, pp. 58–65, 2021. 6.3.3

[231] L. Müller, M. Galkin, C. Morris, and L. Rampášek, "Attending to graph transformers," *arXiv preprint arXiv:2302.04181*, 2023. 6.3.4

[232] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 819–10 829. 6.3.4

[233] C. Xiao, Y. Long, and N. A. Moubayed, "On isotropy and learning dynamics of contrastive-based sentence representation learning," *arXiv preprint arXiv:2212.09170*, 2022. 6.4.2

[234] F. Frasca, B. Bevilacqua, M. M. Bronstein, and H. Maron, "Understanding and extending subgraph gnns by rethinking their symmetries," *arXiv preprint arXiv:2206.11140*, 2022. 6.4.3, 6.2

[235] L. Beyer, P. Izmailov, A. Kolesnikov, M. Caron, S. Kornblith, X. Zhai, M. Minderer, M. Tschannen, I. Alabdulmohsin, and F. Pavetic, "Flexivit: One model for all patch sizes," *arXiv preprint arXiv:2212.08013*, 2022. 2

[236] Z. Sun, A. I. Cristea, P. Lio, and J. Yu, "Adaptive distance message passing from the multi-relational edge view," in *The First Tiny Paper Tracks at ICLR 2023*, 2023. 3

[237] L. Sang, M. Xu, S. Qian, and X. Wu, "Adversarial heterogeneous graph neural network for robust recommendation," *IEEE Transactions on Computational Social Systems*, 2023. 7.2

[238] P. Veličković, "Everything is connected: Graph neural networks," *Current Opinion in Structural Biology*, vol. 79, p. 102538, 2023. 7.3

[239] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao *et al.*, "A comprehensive survey on deep graph representation learning," *arXiv preprint arXiv:2304.05055*, 2023. 7.4

[240] J. Ma, J. Deng, and Q. Mei, "Subgroup generalization and fairness of graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1048–1061, 2021. 7.4

[241] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021. 7.4

[242] J. W. Shavlik, "A framework for combining symbolic and neural learning," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 1992. 7.4

[243] D. Hilbert and W. Ackermann, *Principles of mathematical logic*. American Mathematical Society, 2022, vol. 69. 7.4

[244] J. Pujara, H. Miao, L. Getoor, and W. Cohen, "Knowledge graph identification," in *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I 12*. Springer, 2013, pp. 542–557. 7.4

[245] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, A. Wahler, D. Fensel *et al.*, "Introduction: what is a knowledge graph?" *Knowledge graphs: Methodology, tools and selected use cases*, pp. 1–10, 2020. 7.4

[246] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021. 7.4

[247] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 5329–5336. 7.4

[248] F. Lecue, "On the role of knowledge graphs in explainable ai," *Semantic Web*, vol. 11, no. 1, pp. 41–51, 2020. 7.4

[249] D. Tena Cucala, B. Cuenca Grau, E. V. Kostylev, and B. Motik, "Explainable gnn-based models over knowledge graphs," in *International conference on learning representations*, 2022. 7.4

[250] C. Shan, Y. Shen, Y. Zhang, X. Li, and D. Li, "Reinforcement learning enhanced explainer for graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 523–22 533, 2021. 7.4

[251] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: exploring a routing optimization use case," *Computer Communications*, vol. 196, pp. 184–194, 2022. 7.4

[252] K. Aggarwal, M. M. Mijwil, A.-H. Al-Mistarehi, S. Alomari, M. Gök, A. M. Z. Alaabdin, S. H. Abdulrhman *et al.*, "Has the future started? the current growth of artificial intelligence, machine learning, and deep learning," *Iraqi Journal for Computer Science and Mathematics*, vol. 3, no. 1, pp. 115–123, 2022. 7.4