論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Predictive Analytics in Human Behaviors and Social Trends with Graph Neural Networks
著者(和文)	Jin Ruidong
Author(English)	Ruidong Jin
出典(和文)	学位:博士(学術), 学位授与機関:東京工業大学, 報告番号:甲第12797号, 授与年月日:2024年3月26日, 学位の種別:課程博士, 審査員:村田 剛志,岡崎 直観,徳永 健伸,石田 貴士,金崎 朝子
Citation(English)	Degree:Doctor (Academic), Conferring organization: Tokyo Institute of Technology, Report number:甲第12797号, Conferred date:2024/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Doctoral Dissertation

Predictive Analytics in Human Behaviors and Social Trends with Graph Neural Networks

Ruidong Jin

A thesis submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy

in the

Department of Computer Science
School of Computing
Tokyo Institute of Technology

Supervisor: Tsuyoshi Murata

January, 2024

Abstract

Predictive analytics in human behaviors and social trends is crucial for gaining invaluable insights into the future trajectories of public concerns and interests. Discovering and modeling the representations and relations of real-world entities pose fundamental challenges. Nowadays, the advent of artificial intelligence (AI) has ushered in a transformative era for research on human social activities. AI tools and methodologies have become indispensable for analyzing vast datasets and extracting profound insights.

AI and deep learning techniques have led to significant improvements in fields such as natural language processing, computer vision, and other domains based on regular structured data. However, in many real-world situations, data exhibit irregularities, requiring graph or network models to be explicitly modeled. Such applications involving irregular data structures include social networks, sensor feeds, neuroscience, and various domains. This thesis mainly focuses on a branch within graph models, graph neural networks (GNNs), and their applications in understanding human behaviors and social trends within irregular data structures.

GNN is an extension and evolution of deep learning methods for analyzing graph data. GNNs are designed to perform optimized transformations on all graph attributes while preserving graph topology, which is a powerful tool for capturing intricate relationships and dependencies in various real-world domains. The versatility of GNNs makes them well-suited for addressing the challenges posed by irregular and interconnected data, contributing to the advancement of predictive analytics in research on human social activities.

In this thesis, we start with the formalization of GNNs and consider two dimensions within the categories of graphs: homogeneous/heterogeneous graphs and static/dynamic graphs. Our research objective aims to propose novel GNN models that comprehensively capture complex relationships within entities and effectively learn entity representations in various social activities. Furthermore, these enhanced GNN models demonstrate remarkable versatility, seamlessly transitioning from simple static homogeneous scenarios to challenging dynamic heterogeneous scenarios. We evaluate the performance of the proposed models across three representative real-world human behavior and social trends tasks, covering public healthcare, online commercial activities, and social media networks. Through extensive experimen-

tal results and analysis for each task, we demonstrate that the specially designed GNN models consistently outperform other state-of-the-art and other graph-related methods.

Acknowledgment

Firstly, I would like to express my gratitude to my academic supervisor, Prof. Tsuyoshi Murata, for his unwavering support and the opportunities provided throughout my entire Master's and Doctoral studies. I am also thankful for his generosity in allowing me to choose my research interests freely.

I would like to send my appreciation to my significant mentor, Dr. Xin Liu, for his kind support and the opportunity to join the National Institute of Advanced Industrial Science and Technology (AIST) as a Research Assistant. Thanks to him and AIST, I learned how to be a good researcher and had access to a fast supercomputer that greatly assisted me in exploring various ideas.

I would like to acknowledge the former and current members of the Murata laboratory. I have learned much from them through participating in seminars and engaging in research discussions. I want to send special thanks to Dr. Yin Jun Phua, the assistant professor at Murata Laboratory, for the help with the research and encouragement during the period of my thesis defense examination.

I am grateful to the "Cross the border! Tokyo Tech Pioneering Doctoral Research Project" for the carefully organized research activities and financial support provided by Tokyo Tech.

Finally, I want to express my deepest thanks to my friends and family for their constant support throughout my long overseas journey.

Contents

Abstract ii

Acknowledgment iv

1 Introduction 1

1.1 Predictive Analytics with Artificial Intelligence in Human Behaviors and Social Trends . . . 1

1.1.1 Overview of Human Behaviors and Social Trends . . . 1

1.1.2 Artificial Intelligence Techniques in Analyzing Real-world Data 4

1.1.3 Graph Models in Predictive Analytics on Real-world Data . . 6

1.2 Graphs . . . 7

1.3 Graph Neural Networks . . . 9

1.4 Research Objectives . . . 12

1.4.1 Motivations . . . 12

1.4.2 Challenges . . . 12

1.4.3 Research Goal . . . 13

1.4.4 Research Maps . . . 15

1.5 Thesis Outline . . . 16

1.6 List of Publications . . . 18

2 Background 20

2.1 A Taxonomy of Graphs . . . 20

2.1.1 Undirected, Directed and Weighted Graph . . . 20

2.1.2 Homogeneous & Heterogeneous graph . . . 21

2.1.3 Dynamic Graph . . . 24

2.2 Graph Neural Networks . . . 26

2.2.1 GNNs on Dynamic Graphs . . . 30

2.2.2 GNNs on Heterogeneous Graphs . . . 31

2.3 Graph Downstream Tasks and Applications in Predictive Analytics . 31

2.4 Advanced Modules in GNNs . . . 33

2.4.1 Graph Structure Learning . . . 33

2.4.2 Attention Mechanism . . . 34

2.4.3 Decoders and Loss Functions . . . 37

3 Predictive Analytics on Static Bipartite Graph 41
3.1 Introduction . . . 41
3.2 Related Work . . . 45
3.3 Problem Setup . . . 47
3.3.1 Dataset Description . . . 48
3.3.2 Edge Label Classification in the Hospital-Region Graph . . . 49
3.4 Limitations of Vanilla GCN in Bipartite Graph . . . 50
3.5 Proposed Approach . . . 51
3.5.1 BiGCN . . . 52
3.5.2 Loss Function . . . 54
3.6 Experiments . . . 55
3.6.1 Performance Evaluation . . . 57
3.6.2 Main Factors . . . 58
3.6.3 Parameter Study . . . 61
3.6.4 Case Study . . . 62
3.7 Conclusion and Discussion . . . 62

4 Predictive Analytics on Discrete-time Dynamic Graph 64
4.1 Introduction . . . 64
4.2 Related Work . . . 68
4.2.1 Graph Embedding Learning . . . 68
4.2.2 Trends Prediction Tasks in Real-world Data . . . 69
4.3 Problem Setup . . . 70
4.4 Methodology . . . 71
4.4.1 Overview . . . 71
4.4.2 Graph Structure Learning . . . 72
4.4.3 Node Aggregation . . . 73
4.4.4 Time-sequence Unit . . . 74
4.4.5 Multi-head Attention Layer . . . 74
4.4.6 Loss Function . . . 75
4.4.7 Computational Complexity . . . 76
4.5 Experiments . . . 76
4.5.1 Dataset Description . . . 77
4.5.2 Setup of the Experiment . . . 79
4.5.3 Baselines . . . 80
4.5.4 Evaluation . . . 81
4.5.5 Parameter Sensitivity . . . 84
4.5.6 Ablation Study . . . 85
4.5.7 Statistical Test . . . 86

4.6 Discussion and Conclusion . . . 89
4.6.1 Discussion . . . 89
4.6.2 Conclusion . . . 89

5 Predictive Analytics on Continuous-time Dynamic Graphs 91
5.1 Introduction . . . 91
5.2 Related Work . . . 95
5.3 Problem Setup . . . 97
5.3.1 Dataset Description . . . 97
5.3.2 Research Problem . . . 99
5.4 Constructing Dynamic Graphs . . . 100
5.5 Limitations of Traditional Temporal GCN . . . 106
5.6 Temporal Difference Graph Neural Network (TDGNN) . . . 107
5.6.1 Temporal Difference Aggregation . . . 108
5.6.2 Attention encoder . . . 109
5.6.3 Ensemble MLP decoder . . . 110
5.6.4 Computational Complexity . . . 111
5.6.5 TDGNN vs. TGNNs . . . 111
5.6.6 Strategies for Data Imbalance . . . 112
5.7 Experiments . . . 114
5.7.1 Setup of the Experiment . . . 114
5.7.2 Baselines . . . 115
5.7.3 Evaluation . . . 118
5.7.4 Training Time . . . 119
5.7.5 Time Delay When Updating Node Status . . . 120
5.7.6 Effect of the Strategies for Data Imbalance . . . 123
5.7.7 Parameter Sensitivity . . . 124
5.7.8 Node Embedding Visualization . . . 126
5.8 Conclusion and Discussion . . . 126

6 Discussion 128
6.1 Unified Design Considerations within Proposed GNN Frameworks . . . 128
6.1.1 Process of Addressing Prediction Tasks by GNN frameworks . . . 129
6.1.2 Necessity of Domain-specific GNNs . . . 132
6.1.3 Methodologies for Incorporating Domain-specific Knowledge into GNNs . . . 133
6.2 Relationships Among Our Proposed GNNs . . . 135
6.2.1 Coherency of Subworks . . . 135
6.2.2 Commonalities and Distinctions . . . 136
6.2.3 Challenges and Contributions of Subworks . . . 137
6.3 Evaluations of BiGCN and TDGNN Model on Other Datasets . . . 140

6.3.1 BiGCN on other Bipartite Graph Datasets 141
6.3.2 TDGNN on other CTDG Datasets 142
6.4 What Kind of Real-world Prediction Tasks Can Be Addressed by Our Proposed GNNs 144

7 Conclusion **147**
7.1 Answers to Research Questions 147
7.2 Key Contributions 148
7.3 Future Work 150

List of Figures

1.1 The regular data structures. Images can be represented as grid structures. Text and speech contents can be represented as sequence structures. 5
1.2 The irregular data structures. For example: sensor networks, traffic networks, social networks, neuronal connections, and more. 5
1.3 A visualization of social networks. Users are depicted as nodes, and the edges between them symbolize friendships. The user holds various attributes, which are referred to node features. 8
1.4 Graph embedding learning is to find a proper way to encode the graph structures, including nodes and edges, into low-dimensional embedding vectors while maximally preserving graph structure and information. 8
1.5 A simple formulation of GNN. Input is graph structure and attributes (node/edge features). Output is learned node embeddings, which can be used in node-level, edge-level, and graph-level downstream tasks. 9

2.1 An undirected graph, a directed graph, and a weighted graph with their respective adjacency matrices. 21
2.2 A homogeneous graph. All nodes and all edges are of the same type (e.g., all the nodes are users). 22
2.3 A heterogeneous graph. Nodes and/or edges have multiple types (e.g., nodes include users, reviews, and products. Edges include user-write-review, review-on-product, and user-buy-product). 22
2.4 A bipartite graph. Nodes are divided into two disjoint user group and item group, and edges only connect nodes from two different groups. . 23
2.5 A spatio-temporal graph. Different node colors denote the change of node features. 24
2.6 A discrete-time dynamic graph. 25
2.7 A continuous-time dynamic graph. 26

2.8 Illustration of the message passing update within a GNN with four nodes. In the left figure, node embeddings $\mathbf{h}_1 \sim \mathbf{h}_4$ are assigned to each node. Focusing the center node $\mathbf{h}_2$, intermediate embeddings $\mathbf{h}_{(1,2)}$, $\mathbf{h}_{(3,2)}$ and $\mathbf{h}_{(4,2)}$ are produced using nearby node emebddings $\mathbf{h}_1$, $\mathbf{h}_3$ and $\mathbf{h}_4$, node features $\mathbf{f}_1$, $\mathbf{f}_3$ and $\mathbf{f}_4$, and edge weights (if present) in the message passing step. The right figure shows that the updated node embedding $\mathbf{h'}_u$ is an aggregation of nearby intermediate embeddings. 27
2.9 The overview of TGN model. This figure is cited from [Rossi et al., 2020]. 30
2.10 Common downstream tasks of GNN. 32
2.11 An overview of the multi-head attention module. Left is the Scaled Dot-Product Attention. Right is the Multi-head Attention consists of several attention layers running in parallel. This figure is cited from [Vaswani et al., 2017]. 35
2.12 An overview of a simple MLP decoder. 38

3.1 The hospital-region EMS bipartite graph. 42
3.2 The number of emergency cases in different regions of Tokyo. 47
3.3 Predicting the high/low demand label of hospital-region pairs. 49
3.4 Overview of the structure of our approach. 52
3.5 Parameter study. (a)(b) Results with different α. (c)(d) Results with different numbers of layers. (e)(f) Results with different dimensions of hidden units. 60
3.6 The identified hospitals with high EMS demand. 62

4.1 An example of a social media network consists of users and items. Solid lines denote the interactions between users and items, and dash lines denote the interactions between user-item pairs. Interactions and entity attributes evolve as time goes by. The popularity trend of an item is represented by the visit frequency, which indicates the number of accesses from users (the degree of dashed lines). 66
4.2 Problem setup of the popularity trend prediction in social networks. The social networks are presented as a sequence of graph snapshots $\mathbf{G} = \{\mathbf{G}^{(1)} \sim \mathbf{G}^{(t)}\}$. The proposed method first learns the node embeddings $\mathbf{Z_m}^{(1)} \sim \mathbf{Z_m}^{(t)}$ of the target node type m by an encoder $\mathcal{G}$, then predicts the node labels $\mathbf{l_m}^{(2)} \sim \mathbf{l_m}^{(t+1)}$ in the upcoming graph snapshots by a decoder $\mathcal{F}$. 70

4.3 Overview of the proposed multi-layer temporal GNN framework. Inputs are depicted as white blocks, main components are highlighted in blue, intermediate results are presented in yellow, and final outputs are shown in red. Intra graphs comprise the multiple relations within the target entities of the same type. The bipartite graph indicates the relationship between entities of different types. 71
4.4 The box plot of the target values (true labels) in the YouTube live streaming dataset. The target values are distributed from 1 to more than 10 million. "Month" and "Week" denotes the interval of graph snapshots (one-month interval and one-week interval). 78
4.5 The box plot of the improved percentage of change on target values (true labels) in YouTube live streaming dataset. The percentages are distributed from -1 to 2 after eliminating outliers. "Month" and "Week" denotes the interval of graph snapshots (one-month interval and one-week interval). 79
4.6 The box plot of the results. Four rows denote four datasets and two columns denote two metrics. 82
4.7 Study of parameter sensitivity. 84

5.1 An example of a superchat on the YouTube Live platform is when a viewer named "Milktea" donates \$5.00 to the streamer and posts a superchat message that reads, "I love your live-streams! Thanks for streaming", in a live streaming channel. 93
5.2 Problem description: Our objective is to predict the occurrence of superchat messages and their appearance timing in a given chat message stream. Given the timed sequence of chat messages during a certain period $0 \leq t \leq T$, our method aims to predict the viewers i who send superchat message $\mathbf{msg}_i(t)$ after T and the corresponding time t. 100
5.3 We conducted a test on the pre-trained Sentence Transformers model to measure sentence similarity. We used the source sentence "Thanks for streaming" and compared it with different target sentences. Sentences with positive and similar meanings, such as "Nice!" and "Congratulations!" received higher similarity scores of 0.366 and 0.327, respectively, whereas the sentence with a negative and opposite meaning, "I don't like it", had the lowest score of 0.184. Additionally, the model could identify the meanings of sentences written in different languages or emojis. For instance, the Japanese sentence "Congratulations!" and the emoji "Clap hands" received almost identical similarity scores (0.345 and 0.387, respectively) as their English equivalents. 102

5.4 The diagram illustrates the process of constructing dynamic graphs from live streaming chat messages. In this example, four viewers post chat messages at timestamps t_1 to t_4 in the same batch, with the first message being a superchat. We create nodes 1 to 4 for each viewer and initialize their feature vectors using the sentence embedding vector of the chat message they posted. Next, we compute the cosine similarity between each pair of nodes and generate edges for node pairs with high similarity. The blue node represents a viewer who sent a superchat, while the green nodes represent viewers who only sent regular chat messages. 104
5.5 Overview of the proposed Temporal Difference Graph Neural Network (TDGNN). 107
5.6 Compute the temporal difference in a directed graph. Temporal difference $\partial\mathbf{u}$ represents the information difference between node $\mathbf{u}$ and the surrounding neighbor nodes, indicating the amount of information propagation in dynamic graphs. 108
5.7 A multi-head attention module that calculates the new temporal node embedding $\mathbf{u}_i(t+1)$ according to the relativity between the last updated embedding $\mathbf{u}_i(t)$, temporal difference of neighbor nodes, and newly-updated node features. 109
5.8 Training time (seconds) per epoch in three dynamic graph datasets. . 120
5.9 The update time delay in previous TGNNs. Node status updated during batch process are reflected until the batches are finished, thus resulting in an update time delay. 121
5.10 The zero update time delay in TDGNN. TDGNN has a real-time node status update mechanism that can update node information during batch process. 121
5.11 The AUC score with different numbers of sampled neighbors and MLP decoders. 125
5.12 The training time (seconds) per epoch with different numbers of sampled neighbors and MLP decoders. 125
5.13 The t-SNE visualization of the node embeddings learned by TDGNN. 126

6.1 The unified process of addressing prediction tasks using GNN frameworks. First, we collect real-world data and perform necessary preprocessing steps to enhance its suitability for modeling. Then,we construct graph structures and incorporate feature information, tailoring them to the unique characteristics of the dataset under consideration. These structures and features are input into the GNN frameworks, where the models learn node representation embedding vectors. Finally, the trained models generate predictions for specific tasks. The model performance is assessed against state-of-the-art methods through extensive experimental results on targeted tasks. . 129
6.2 A flowchart of how to construct the appropriate graph from the dataset. 130
6.3 Datasets from different domains may exhibit the same graph structure. The railway network of the Yamanote Line in Tokyo, the molecular structure of Benzene, and interconnected computers forming a circular network all share the same ring-structured graph topology, while the domain-specific node and edge features vary across these disparate datasets. 133
6.4 The connections among three subworks in this thesis. The three works are linearly evolving and improving, progressing from less general and simpler cases to more challenging, general, and complex cases. Each subsequent work addresses the limitations of the previous one and extends the previous model to adapt to more practical situations. . . 136

List of Tables

1.1 Categories of human behaviors and social trends from perspectives of objectives and frequencies. Categories marked in **bold** are involved in the research scope of this thesis. 2
1.2 Representative examples of human behaviors and social trends and practical cases of predictive analytics within them. Cases marked in **bold** are involved in the research scope of this thesis. 3
1.3 Prior studies on GNNs. The GNNs research is evolving from foundational models to sophisticated and widely utilized models. 10
1.4 GNN categories involved in this thesis. 15
1.5 Real-world case studies involved in this thesis. 16

3.1 Features of hospitals and regions. 48
3.2 Statistics of the bipartite graph for the EMS dataset. 50
3.3 Fine-tuned hyperparameters in the heterogeneous GNN baseline methods. 57
3.4 The results for predicting the hospital-region labels by different approaches. 57
3.5 Z-score and p-value of variable coefficients in ZIP regression model (test ratio=0.1) 59

4.1 Statistics of the social media network datasets used in our experiments. 77
4.2 Input information required by the baseline models. 80
4.3 The results of node attribute value prediction tasks on four datasets. Metric is RMSE scores and MAPE scores (%). Results are mean values of 30 runnings. 83
4.4 The ablation study results without specific key components in the proposed method. 85
4.5 The ranks of nine methods assessed on eight tests. $T_1 \sim T_4$ represent RMSE scores on the former half of the YouTube, MOOC, Reddit, and Wikipedia datasets. $T_5 \sim T_8$ represent RMSE scores on the latter half of the YouTube, MOOC, Reddit, and Wikipedia datasets. Any tied ranks are assigned an average rank. 87

4.6 The results of the Wilcoxon sign-rank test between pairs of the proposed method and each baseline method. Results are Benjamini–Hochberg FDR adjusted p-values. 88

5.1 Superchat purchase details 98
5.2 Detailed dataset statistics 99
5.3 Structure distinction among TGNNs 111
5.4 Cost matrix 113
5.5 Statistics of the live streaming dynamic graphs. 114
5.6 Input information required by the baselines 116
5.7 Fine-tuned hyperparameters in the dynamic GNN baseline methods. . 117
5.8 AUC scores for predicting the real-time node labels 118
5.9 Update time delay (seconds) and AUC scores in TGNNs with different batch sizes 122
5.10 AUC scores under the effect of strategies for data imbalance 124

6.1 Architecture distinctions among our proposed GNNs. 137
6.2 Statistics of the bipartite graph datasets. 141
6.3 Fine-tuned common hyperparameters in the baseline methods. 141
6.4 Results in % for link prediction on Cora and Citeseer. Standard deviations over 10 random seeds are shown in parentheses. The best results are marked in bold, and the second bests are marked with underline. 142
6.5 Statistics of the CTDG datasets. 143
6.6 Fine-tuned common hyperparameters in the baseline methods. 143
6.7 AUC score in % for node label prediction tasks on Wikipedia and Reddit datasets. A Standard deviations over 10 random seeds are shown in parentheses. The best results are marked in bold, and the second bests are marked with underline. 144
6.8 Requirements on real-world data that proposed GNNs can be applied to. 145

Chapter 1

Introduction

1.1 Predictive Analytics with Artificial Intelligence in Human Behaviors and Social Trends

Human behavior refers to the actions, reactions, and conduct of individuals or groups in response to internal and external stimuli. It encompasses a wide range of activities, including cognitive processes, economic actions, communication, and social interactions. Social trends refer to the patterns, shifts, or changes in behavior, attitudes, and preferences within a society over time. These trends can manifest in various areas, such as demographic trends, cultural preferences, political engagement, and more. The study of human behavior and social trends involves understanding individual and collective actions, predicting future patterns, and contributing insights that can inform decision-making at various levels, from individual choices to societal policies. Social science studies a broad field that explores various aspects of human behaviors and social trends. Predictive analytics in social science involves applying big-data driven methods, such as statistical models and machine learning techniques, to analyze historical data and make predictions about future events or trends within society [Dinov, 2018; Kumar and Garg, 2018; Mishra and Silakari, 2012; Poornima and Pushpalatha, 2018; Selvaraj and Marudappa, 2018]. They encompass disciplines such as sociology, psychology, anthropology, economics, political science, and more. Social scientists have always aimed to figure out the complex aspects of human behavior, society, and culture.

1.1.1 Overview of Human Behaviors and Social Trends

Categorizing human behaviors and social trends involves considering various lenses through which behaviors and trends can be included in the scope of our research. Table 1.1 categorizes human behaviors and social trends from perspectives of objectives and frequencies, providing an overview understanding of their complexities

Table 1.1: Categories of human behaviors and social trends from perspectives of objectives and frequencies. Categories marked in **bold** are involved in the research scope of this thesis.

Perspectives	Categories	Practical cases
Objectives	Actions by individuals	Decision-making processes, sentimental regulation
	Collective actions within communities and groups	Demographic trends, economic trends, instant messaging
	Relationships between people and other entities	Consumer product preferences, citation networks, public healthcare
	Relationships between entities	IoT, environmental trends, supply chain
Frequencies	**Infrequently change**	Transportation networks, food chain, molecular structures
	Change with regular observation intervals	Population census, sensor network
	Change anytime	Social networks, traffic network congestion
	Change by a certain probability	Weather change, sports match outcomes

and nuances:

- *Objectives*

  The objectives outline the various research targets of human social activities. Individual actions encapsulate cognitive and emotional behaviors that significantly influence decision-making processes [Koot et al., 2021] and sentiment regulation [Chauhan et al., 2021; Kavitha et al., 2018]. Collective actions within communities shed light on broader group characteristics, including demographic [Li et al., 2020a], economic trends [Chu and Qureshi, 2023; Yoon, 2021] at regional or national levels, and instant messaging apps like LINE and Slack that connecting groups of people. Exploring relationships across individuals and other real-world entities reveals intricate human interactions and their impact on the world, encompassing commercial behaviors reflecting consumer product preferences [Chen et al., 2019; Wu et al., 2022], citation networks between authors and papers, and public healthcare connecting patients and hospitals [Krishnamoorthi et al., 2022; Rong et al., 2020]. Discovering relationships between different entities is also significance, including the Internet-of-thing (IoT), environmental trends shaped by human activities [Kim et al., 2020; Lam et al., 2023], and global supply chains [Kosasih and Brintrup, 2022].

- *Frequencies*

  The frequency of events serves as a key determinant of the dynamic nature inherent in human behaviors and social activities, bearing significant implications for both historical analysis and predictive analytics for future occurrences. Activities characterized by infrequent changes can be classified as static systems (e.g., designing transportation networks, food chains among species, and molecular structures), offering a stable foundation for comprehending target objectives over the long term. Conversely, activities undergoing frequent changes are deemed dynamic systems, unveiling the temporal dynamics inherent in target objectives. Dynamic systems can be further classified based

Table 1.2: Representative examples of human behaviors and social trends and practical cases of predictive analytics within them. Cases marked in **bold** are involved in the research scope of this thesis.

Domains	**predictive analytics cases**
Cognitive behaviors	Decision-making processes [Koot et al., 2021]
Emotional behaviors	Sentimental analysis [Chauhan et al., 2021; Kavitha et al., 2018]
Social behaviors	Interpersonal communication and interaction [Tang et al., 2020]
Consumer behaviors	Recommender systems [Chen et al., 2019; Wu et al., 2022]
Deviant behaviors	Fraud detection [Severino and Peng, 2021; Zakaryazad and Duman, 2016]
Demographic trends	Migration patterns and urbanization [Li et al., 2020a]
Economic trends	GDP growth [Chu and Qureshi, 2023; Yoon, 2021]
Public health trends	Healthcare [Krishnamoorthi et al., 2022; Rong et al., 2020]
Environmental trends	Climate change [Kim et al., 2020; Lam et al., 2023]
Social media trends	Influence of contents in social media [Szabo and Huberman, 2010]

on the consistency of their changes: changing with regular observation intervals (e.g., population census, sensor network) and changing at any given time (e.g., social networks, traffic network congestion). These distinctions are crucial as they dictate variations in representation and modeling methodologies. Besides, there is another dynamic patterns that entities change by a certain probability, such as weather change and the outcomes of sports matches.

In real-world situations, the predictive analytics plays a crucial role in providing a disciplinary analysis of human behaviors and social trends across various categories. Predicting human behaviors and social trends can offer numerous benefits across diverse domains. Table 1.2 demonstrates a part of representative domains of human behaviors and social trends and practical cases of predictive analytics within them. For example, in business and marketing domain, predictive analytics aids in comprehending consumer behavior, enabling effective product recommendations tailored to meet customer preferences [Chen et al., 2019; Wu et al., 2022]. Moreover, predictive analytics facilitates customers' efficient decision-making based on their cognitive behaviors [Bolton, 2016], and understands their emotional behaviors by sentimental analysis [Chauhan et al., 2021; Kavitha et al., 2018]. Additionally, it assists in identifying deviant behaviors, contributing to the detection of fraudulent transactions within online commercial activities [Severino and Peng, 2021; Zakaryazad and Duman, 2016].

Based on the previous overviews, the categorizations and practical activities of human behaviors and social trends are too broad to cover in one thesis. In this thesis, the research scope is located in the particular objectives and frequencies. Specifically, we focus on relationships between people and other real-world entities under three frequency patterns, namely infrequent (static), regularly observed (discrete-time dynamic), and continuous changes (continuous-time dynamic). Moreover, this thesis primarily focuses on the practical cases of social behaviors, public health trends, and social media trends, based on considerations of dataset accessibility and real-

world significance. In public healthcare, predictive analytics plays a crucial role in forecasting health-related behaviors and trends, optimizing preventive healthcare measures, and enhancing resource allocation to improve patient care and reduce costs [Jin et al., 2021; Krishnamoorthi et al., 2022; Rong et al., 2020]. Predictive analytics in social behaviors and social media trends assists in understanding interpersonal communication and interactions on the Internet [Tang et al., 2020], anticipating popular content [Jin et al., 2022, 2023], enabling creators and businesses to tailor their offerings [Szabo and Huberman, 2010], while also detecting potential crises or negative trends at an early stage [Sevim et al., 2014].

1.1.2 Artificial Intelligence Techniques in Analyzing Real-world Data

Nowadays, with the rise of artificial intelligence (AI), social sciences research has entered a new phase. AI tools and methods are now crucial in analyzing big data and gaining insights [Kibria et al., 2018]. Social sciences provide the insight into real-world research issues such as economics, politics, and environment, while the computer science contributes expertise in developing AI methods. The combination between social sciences and AI helps us explore existing theories more profoundly and uncover new patterns. This blend enriches our understanding of the human experience and allows us to discover new things about how people live in today's world, and even predicting people's lives in the future. The integration of AI techniques empowers social science researchers to efficiently tackle initial data processing tasks, such as analyzing intricate social networks and identifying key influencers, interaction patterns, and information diffusion [Jin et al., 2022, 2023]. Deep learning methods contribute significantly to various aspects of social sciences research, enhancing the capabilities of researchers in several key areas such as Computer Vision (CV) [Gu et al., 2018], Natural Language Processing (NLP) [Vaswani et al., 2017], AI generated content (AIGC) [Cao et al., 2023], Large-language Models (LLMs) [Radford et al., 2019] and more. Moreover, AI models facilitate predictive analytics in diverse real-world scenarios, including social trends [Chen et al., 2021], economic trends [Ahmadi et al., 2019], public health outcomes [Feng et al., 2021], traffic flows [Akhtar and Moridpour, 2021], and more.

The evolution of AI techniques has elevated predictive analytics to new heights. Predictive analytics via machine learning and deep learning relies on the massive volume of real-world datasets. As shown in Fig. 1.1, some of this data exhibits a regular structure with a symmetrical and compact shape, as seen in images and time-series data. These regular datasets can be represented in an n-dimensional Euclidean space, making them amenable to modeling using traditional machine learning and deep learning techniques. Convolutional Neural Networks (CNNs) are renowned

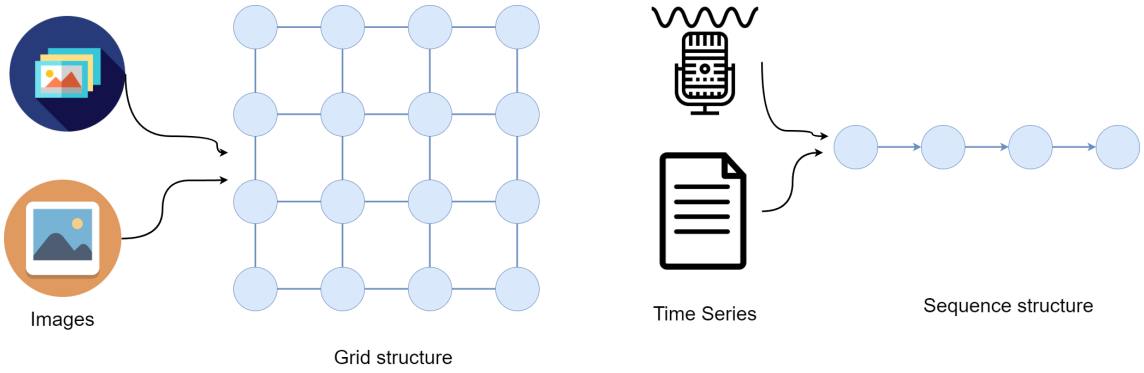


Figure 1.1: The regular data structures. Images can be represented as grid structures. Text and speech contents can be represented as sequence structures.

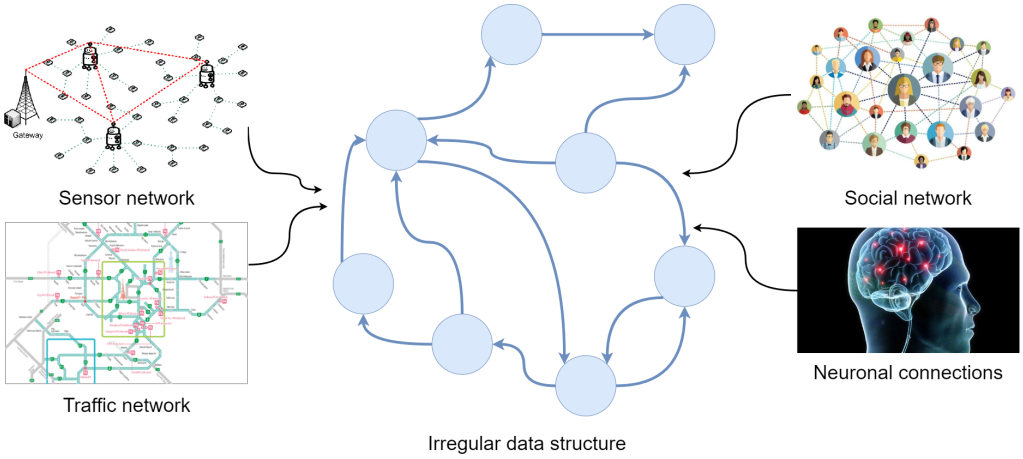


Figure 1.2: The irregular data structures. For example: sensor networks, traffic networks, social networks, neuronal connections, and more.

deep learning frameworks designed for handling image data [LeCun and Bengio, 1998], while the Transformer framework has gained prominence for processing time-series data [Vaswani et al., 2017]. LLMs like Chat-GPT can achieve general-purpose language generation by learning statistical relationships from extensive text document data [Radford et al., 2019]. These well-established methods have achieved numerous successes in their respective domains, providing robust support for predictive analytics within regular data structures.

However, not all real-world data adheres to regular structures; a significant portion exhibits irregular structures where entities cannot be represented in Euclidean space, as depicted in Fig. 1.2. These irregular data structures arise from sensor networks, traffic patterns, social networks, neuronal connections, and more [Bronstein et al., 2017]. With the rapid development of the Internet, the volume of irregular data structures has surged. The Internet, by shortening physical distances between entities, has given rise to numerous network data structures.

Irregular data structures pose a challenge for prior deep learning techniques such as CNNs and transformers, designed primarily for regular data. In a CNN, for in-

stance, the convolution kernel slides through the image from left to right and top to bottom, relying on a grid structure. However, this directional concept is undefined for irregular data lacking a grid structure. Traditional methods for handling with irregular data structures involve data-driven approaches, such as treating irregular data as manifolds through methods like multidimensional scaling (MDS) [Tenenbaum et al., 2000], locally linear embedding (LLE) [Roweis and Saul, 2000], stochastic neighbor embedding (t-SNE) [Van der Maaten and Hinton, 2008], and adapting regular data structure methods like CNNs in spectral domains [Boscaini et al., 2015; Bruna et al., 2013; Henaff et al., 2015]. However, these prior works did not comprehensively model entities and their relationships within irregular data, resulting in intricate operations and inefficient performance.

1.1.3 Graph Models in Predictive Analytics on Real-world Data

Graphs represent a crucial tool developed in the research on irregular data. With a foundation in graph theory, graph models can seamlessly integrate both topology structures and entity features within irregular data structures. In recent years, machine learning and deep learning techniques have experienced rapid development in the graph domain, establishing graph models as powerful tools for analyzing data with irregular structures. Notably, recent advancements in applying graph models to real-world predictive analytics have showcased remarkable breakthroughs in addressing critical real-world problems. Google DeepMind proposed a GraphCast model for faster and more accurate global weather forecasting [Lam et al., 2023]. GraphCast has demonstrated unprecedented accuracy in medium-range weather forecasts, outperforming traditional industry gold-standard weather simulation systems. It provides earlier warnings of extreme weather events, accurately predicting cyclone tracks further into the future, identifying atmospheric rivers associated with flood risk, and forecasting the onset of extreme temperatures. GraphCast contributes to risk reduction and enhanced preparedness, potentially saving lives. Furthermore, another model named GNoME, also from Google DeepMind, showcased the transformative potential of graph methods in material science [Merchant et al., 2023]. GNoME predicts the stability of new materials and has facilitated the discovery of 2.2 million new crystals, including 380,000 stable materials with potential applications in powering future technologies. These two groundbreaking achievement emphasizes the potential power of graph models in accelerating the predictive analytics of human social studies.

Compared to the traditional methods of manifolds and spectral CNNs, graph models provide the following significant advancements to predictive analytics for human behaviors and social trends:

- *Capturing Complex Relationships*

  Traditional methods often struggle to capture the complex, non-linear relationships inherent in social and behavioral data. Graph methods, on the other hand, excel at modeling these intricate network structures, such as social connections, influences, and interactions.

- *Dynamic Analysis*

  Social trends and human behaviors are dynamic, constantly evolving over time. Graph methods are well-suited for analyzing such time-evolving networks, capturing temporal changes more effectively than many traditional methods.

- *Handling Heterogeneous Data*

  Graph methods are adept at handling heterogeneous data, which is common in social and behavioral studies. They can integrate various types of nodes and edges (e.g., different types of relationships, interactions) within a single framework.

- *Interpretable Insights*

  Graph visualizations and analyses can offer more interpretable insights into how individuals or groups are interconnected, how influence propagates, and how communities form and evolve.

- *Enhanced Predictive Performance*

  By capturing the complex structures and relationships within the data more effectively, graph methods often lead to enhanced predictive performance in tasks related to human behavior and social trend analysis.

In light of the aforementioned considerations, this thesis delves into the advancements and applications of predictive analytics in the evolving landscape of human behaviors and social trends, with a particular focus on exploring the transformative potential of graph models in forecasting and understanding complex societal phenomena.

1.2 Graphs

Graphs consist of a set of nodes and a set of edges that define relations between the nodes, offering tremendous flexibility for representing real-world entities and relationships. For example, in a social network illustrated in Fig. 1.3, users are depicted as nodes, and the edges between them symbolize friendships. Users may

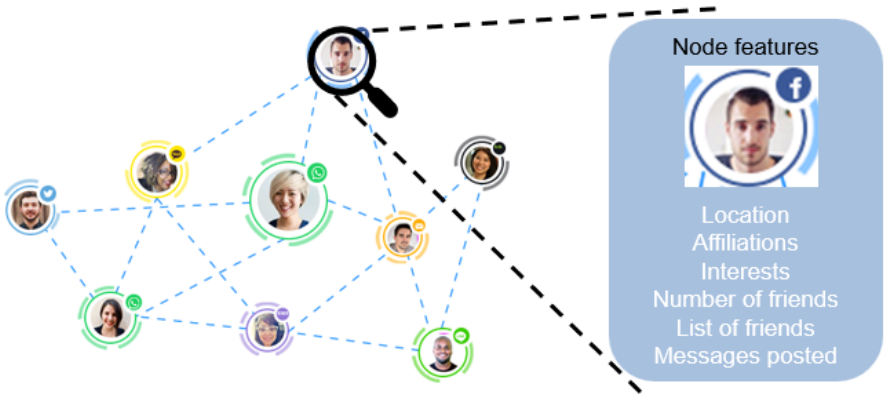


Figure 1.3: A visualization of social networks. Users are depicted as nodes, and the edges between them symbolize friendships. The user holds various attributes, which are referred to node features.

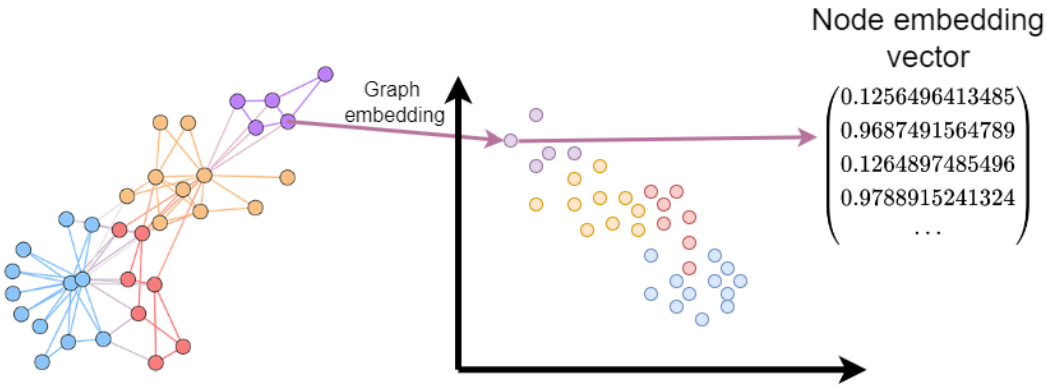


Figure 1.4: Graph embedding learning is to find a proper way to encode the graph structures, including nodes and edges, into low-dimensional embedding vectors while maximally preserving graph structure and information.

share attributes such as geographical location, interests, political affiliations, workplaces, and more, fostering a nuanced understanding of their connections and interactions. Graphs find applicability across diverse datasets, showcasing their versatility in representing various phenomena. They can encapsulate social interactions [Mislove et al., 2007], communication systems [Monge and Contractor, 2003], transport systems [Von Ferber et al., 2009], computer networks [Erciyes, 2013], biological processes [Brohee and Van Helden, 2006], and numerous other domains. This adaptability underscores the efficiency of graph-based models in capturing complex relationships and patterns inherent in different data types.

Graphs serve as fundamental tools in numerous research fields, illustrating patterns of connections within complex systems. Graph embedding learning, a widely recognized technique in graph-related work, has witnessed significant success in recent years [Cai et al., 2018; Cui et al., 2018; Wu et al., 2019b; Zhang et al., 2019b,

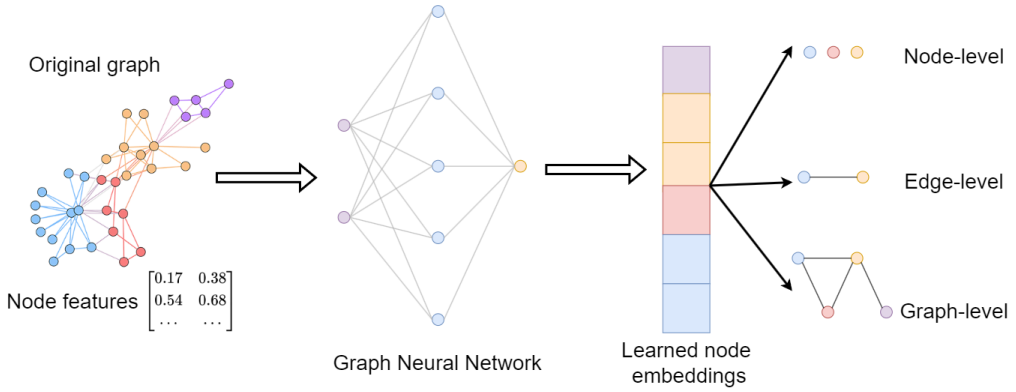


Figure 1.5: A simple formulation of GNN. Input is graph structure and attributes (node/edge features). Output is learned node embeddings, which can be used in node-level, edge-level, and graph-level downstream tasks.

2018b; Zhou et al., 2018]. The graph structure is complicated and hard to do computation, thus we need to transform the graph structure into a format amenable to computation.

Graph embedding learning addresses this challenge by seeking an effective method to encode the structures of the graph, including nodes and edges, into low-dimensional embedding vectors while preserving the graph's structure and information to the maximum extent, as shown in Fig. 1.4. This approach proves powerful for learning representations of real-world graph-structured entities. The acquired embeddings can be utilized as feature inputs for downstream tasks. Various graph embedding methods have been proposed by researchers, including matrix factorization [Ou et al., 2016], edge reconstruction [Tang et al., 2015], random walks plus the skip-gram model [Grover and Leskovec, 2016; Perozzi et al., 2014a], and more.

1.3 Graph Neural Networks

In recent years, deep learning has created remarkable advancements. Large-scale neural network models have consistently yielded state-of-the-art performance across various real-world data and tasks. Capitalizing on this success, researchers have extended the application of neural network models to graph structured data, achieving substantial improvements over traditional graph models. This class of neural network models is called Graph Neural Networks (GNNs). GNNs are deep neural network architectures that encode graph structures and attributes. GNN achieve this by aggregating features of neighbor nodes together. The birth of GNNs can be traced back to the famous CNN [LeCun and Bengio, 1998], a well-known model initially designed for grid-structured data like images. As the generalization version of CNN, GNN is designed for irregular data structures, which could handle the topological structure information within the graph and learn high-level representations

Table 1.3: Prior studies on GNNs. The GNNs research is evolving from foundational models to sophisticated and widely utilized models.

Categories	**Representative prior work**
Recurrent GNNs	GNN [Scarselli et al., 2008a], GraphESN [Gallicchio and Micheli, 2010]
Convolutional GNNs	GCN [Kipf and Welling, 2016a], AGCN [Li et al., 2018], DGCN [Zhuang and Ma, 2018], GAT [Veličković et al., 2018], GraphSage [Hamilton et al., 2017b]
Graph Auto-Encoders	VGAE [Kipf and Welling, 2016b], DNGR [Cao et al., 2016], GraphVae [Simonovsky and Komodakis, 2018], GraphRNN [You et al., 2018]
Graph adversarial networks	GraphGAN [Wang et al., 2018]
Heterogeneous GNNs	HAN [Wang et al., 2019a], HetGNN [Zhang et al., 2019a], PGRA [Chairatanakul et al., 2021]
Dynamic GNNs	GCRN-M1 & GCRN-M2 [Seo et al., 2018], DyRep [Trivedi et al., 2019], JODIE [Kumar et al., 2019], DySAT [Sankar et al., 2020], TGAT [Xu et al., 2020], TGN [Rossi et al., 2020],
Heterogeneous dynamic GNNs	HGT [Hu et al., 2020]

of nodes [Wu et al., 2019b; Zhang et al., 2019b].

For a specific node in a graph, GNNs utilize the graph topology to iteratively aggregate information from its neighboring nodes, deriving its representation through message propagation. This node aggregation process bears similarities to the convolution of pixels in CNN. Through the aggregation of features from neighboring nodes, GNNs can adeptly learn to encode both local and global structures within the graph. The evolution of GNNs within graph embedding learning has introduced novel approaches for comprehending and analyzing intricate relationships and patterns across diverse datasets, further enhancing the efficiency and versatility of deep learning methodologies [Wu et al., 2020; Zhang et al., 2020; Zhou et al., 2020].

GNNs are widely recognized as highly effective approaches for modeling the influence of information diffusion in real-world scenarios. As illustrated in Fig. 1.5, GNNs belong to the category of deep learning structures designed to perform optimized transformations on all graph structures and attributes while preserving graph topology [Wu et al., 2019b; Zhang et al., 2019b]. The versatility of GNNs lies in their direct applicability to graphs, providing a straightforward approach to learning node representation embeddings. These learned node embeddings are valuable results for addressing downstream tasks, including node-level, edge-level, and graph-level applications.

Table. 1.3 categories some of the representative prior GNN studies. The inception of the first GNN was introduced in 2008 [Scarselli et al., 2008b]. Early GNNs were usually associated with a recurrent architecture [Gallicchio and Micheli, 2010]. Then, several influential Convolutional GNNs and Graph Auto-Encoders were released in 2016 and 2017, including GraphSAGE [Hamilton et al., 2017b], Graph At-

tention Networks (GAT) [Veličković et al., 2018], Variational Graph Auto-Encoder (VGAE) [Kipf and Welling, 2016b], and the widely adopted Graph Convolutional Neural Network (GCN) [Kipf and Welling, 2016a]. Besides, with the development of Generative Adversarial Networks (GAN) for estimating generative models via an adversarial process [Goodfellow et al., 2014], its application in graphs named Graph Adversarial Networks are also developed [Wang et al., 2018]. GNNs including, but not limited to the above mentioned ones, have become cornerstone models acknowledged for their effectiveness in various graph-related tasks.

In recent years, research on dynamic GNN came to the public vision. The first dynamic GNN (GCRN-M1 & GCRN-M2) combines CNN for spatial structure identification and Recurrent Neural Networks (RNNs) for dynamic pattern recognition [Seo et al., 2018]. Subsequent models like DyREP and JODIE further refined the concept by enhancing RNN-based dynamic GNNs [Kumar et al., 2019; Trivedi et al., 2019]. DySAT introduced the first dynamic GNN based solely on attention, not relying on RNN [Sankar et al., 2020]. TGAT innovatively encoded time intervals as time embedding vectors [Xu et al., 2020], and TGN improved upon this idea by incorporating a memory module [Rossi et al., 2020]. HAN proposed a heterogeneous graph attention network based on the hierarchical node-level and semantic-level attention [Wang et al., 2019a]. HGT contributed to a heterogeneous graph transformer architecture designed to handle large-scale heterogeneous and dynamic graphs [Hu et al., 2020].

Recent research trends on GNN have showcased a shift toward more intricate scenarios, particularly in handling heterogeneous and dynamic graph structures. The GNNs research is evolving from foundational models to sophisticated and widely utilized models. Moreover, there is a growing focus on expert GNNs designed for specific domains, demonstrating the maturation and diversity of the applications of GNN models. Application domains of GNNs include NLP [Guo et al., 2019; Yin et al., 2020], computer vision [Woo et al., 2018; Yang et al., 2018], social networks [Guo and Wang, 2020; Li et al., 2023], e-commerce [Li et al., 2020b; Liu et al., 2021], recommender system [Fan et al., 2019; Wu et al., 2022; Ying et al., 2018], traffic [Akhtar and Moridpour, 2021; Yang et al., 2023], circuit design [Wang et al., 2020; Yang et al., 2022b], and more. Structural data, such as a social network, inherently have a clear network structure. In contrast, non-structured data, such as text and photos, often necessitate transformation into a structured format before employing GNN.

The robustness and diverse applications of GNNs have spurred continuous efforts to enhance their learning capabilities. Researchers are actively involved in designing new GNN architectures that can adapt to data with diverse underlying patterns and properties, reflecting a commitment to advancing the capabilities of GNN models in addressing challenges across various domains.

1.4 Research Objectives

1.4.1 Motivations

Upon initial inspection, a graph may appear simplistic, featuring nodes interconnected by edges. However, the versatility of graphs is far-reaching. Edges can exist in their basic form (undirected graphs) or be assigned directions (directed graphs), underscoring the uni-directional nature of relationships. Besides, node status and edge connections may change with time, which can model the continuously evolving social media networks. Moreover, nodes can be connected by multiple edges or weighted edges, providing additional layers of complexity. Creating graphs from diverse data sources reveals distinct connectivity patterns between nodes, leading to categorizing graphs into various types, such as homogeneous graphs, heterogeneous graphs, dynamic graphs, and multi-layered graphs.

Within the various alternatives, the graph is an ideal way to capture complex relationships within real-world situations. Graph methods are adept at handling heterogeneous data and time-evolving systems, which are common in social and behavioral studies. Graphs can help capturing temporal changes and integrate various types of nodes and edges (e.g., different types of relationships, interactions) more effectively than many other traditional methods. Moreover, graph visualizations and anlyses can offer more interpretable insights into how entities are connected, influenced, and how communities form and evolve. Such explainability is essential to discover the reasons of human behaviors and social trends.

The connectivity patterns encoded by edges often unveil crucial information about real-world phenomena associated with the dataset. For instance, in scale-free graphs, a few nodes are highly connected to the majority, reflecting a power-law distribution in the degree of nodes. Graphs mirror the phenomenon observed in real-world social networks, where influencers have a vast number of followers compared to the sparser connections of the average person. Understanding such connectivity dynamics is pivotal in applications like curbing the spread of misinformation or strategically disseminating information across a network. Such similar characteristics also improve the compatibility of leveraging graphs to represent real-world datasets.

1.4.2 Challenges

According to the existing work, modeling real-world entity relationships using GNNs presents a multifaceted challenge:

- *Scalability*

  The massive volume and types of data often reaches magnitudes in the millions and beyond, necessitating substantial computational resources and thought-

ful considerations of computation efficiency in model design. It highlights the need for scalable solutions to handle massive datasets while maintaining computational cost.

- *Heterogeneity*

  The existing graph methods primarily focus on plain graphs, which are often insufficient for the complex nature of human social activities. When analyzing such activities, we encounter a rich mix of information types involving various people and objects. Therefore, there is an urgent need for graph model solutions that can adapt to the diversity of real-world entities involved (people, objects, locations, and more) and the multiple interactions among these entities.

- *Dynamics*

  The dynamic nature of most real-world situations introduces an additional layer of complexity: temporal information. Models must exhibit the capability to process temporal information, reflecting the continuously changing of entity status and relationships over time. The challenge lies in achieving accuracy in predictive analytics and ensuring timely responses to changes in the underlying data.

- *Domain-specific expert GNNs*

  The diversity and distinct characteristics of relationships within human activities and social trends underline the importance of specialized methods tailored to specific situations. A one-size-fits-all approach may fall short because of a lack of specific-domain information, emphasizing the necessity for specialist methods capable of addressing the unique features of diverse relationship patterns.

1.4.3 Research Goal

In addressing these challenges, this thesis endeavors to develop innovative GNN models that are specifically tailored to the complexity of human social activities. The research proposal is developing advanced GNNs frameworks to learn entity representations within real-world graph-structured dataset and tackle domain-specific prediction tasks. In consideration of the challenges above, this thesis aims to solve the following research questions:

1. *How to address prediction tasks on heterogeneous data by GNNs?*

   As mentioned, real-world human social activities often entail a rich mix of information types involving various people, objects, and more. The heterogeneity of real-world data manifests not only in the diversity of entity types

but also in the various types of relationships presented. Therefore, an ideal GNN framework should possess the capability to learn and adapt to both the diversity of real-world entities (such as people, objects, locations) and the multitude of interactions among these entities (for example, followers and followees in social networks, users buying items in an e-commerce platform). In Chapter 3 and Chapter 4, we introduce GNN frameworks that can effectively learn entity representation embeddings in both simplistic and intricate heterogeneous real-world data.

2. *How to address prediction tasks on dynamic data by GNNs?*

   Dyanmic GNNs can capture the dynamic nature of most real-world data. However, several challenges make them more complex than the GNNs for static graphs. Firstly, nodes and edges are added or removed over time, leading to temporal changes in graph topology. Additionally, nodes and edges in dynamic graphs may have evolving features over time, necessitating continuous monitoring of node and edge statuses. Furthermore, maintaining computational efficiency during training and inference becomes crucial, since dynamic graphs require continuous updates to the model. It demands strategies that can update the model with minimal costs. In Chapter 4 and Chapter 5, we introduce GNN frameworks tailored for two dynamic patterns: discrete-time dynamic graphs and continuous-time dynamic graphs. We address the aforementioned challenges through specifically designed GNN architectures.

3. *How to tailor GNN architectures to better fit domain-specific prediction tasks?*

   In certain real-world scenarios, data from different domains can be effectively represented by the same graph structure. For example, the ring-structured computer networks, the molecular structure of benzene, and the railway network of the Yamanote line in Tokyo can all be encapsulated within the same ring-structured graph. However, GNNs are often employed to learn from abstract graphs, neglecting domain-specific knowledge that could enhance the model's learning capabilities. Consequently, the conventional one-size-fits-all approach of GNNs may not always be optimal. There is an urgent need for expert GNNs tailored to specific domains. In this thesis, the proposed advanced GNN frameworks are customized for specific real-world domains. Chapter 3 introduces a GNN method designed for prediction tasks in the public health domain. Chapter 4 presents a GNN framework focused on predicting popularity trends in social media networks. Chapter 5 develops a sophisticated GNN model for forecasting commercial activities on live streaming platforms. Through case studies spanning various domains, we aim to gain insights into the critical considerations on how to designing expert GNNs for prediction tasks within specific domains.

Table 1.4: GNN categories involved in this thesis.

GNN categories	Static graph	Discrete-time dynamic graph snapshots	Continuous-time dynamic graph actions
Homogeneous graph	Plain GNNs (e.g., GCN)	Stacks of plain GNNs	Chapter 5
Heterogeneous graph	Chapter 3	Chapter 4	Future direction

1.4.4 Research Maps

This thesis presents an interdisciplinary study that integrates GNNs from the graph models into the exploration of human behaviors and social trends within social sciences. The main goal of this thesis is to devise innovative GNN models capable of capturing complex relationships among entities in real-world data. The proposed models aim to efficiently learn nuanced representations of entities and subsequently employ predictive analytics to address real-world challenges related to human behaviors and social trends.

Table 1.4 lists the categories of GNNs discussed in this thesis. The discussion revolves around the combination of two fundamental graph characteristics: time factors and node/edge types. Graphs with no notion of time are called static graphs, while those incorporating time factors are called dynamic graphs. Dynamic graphs can be further classified into discrete-time dynamic graph snapshots, which consist of graph snapshots observed at regular intervals, and continuous-time dynamic graph actions, which record each graph action (change) individually along with its timestamp. In another hand, graphs comprising only one node type and one edge type are referred to as homogeneous graphs, whereas those with multiple node types or edge types are termed heterogeneous graphs.

The basic static homogeneous GNN has been extensively discussed in prior models like GCN. In the case of homogeneous discrete-time dynamic graph snapshots, they are typically represented as a sequence of static graphs. Therefore, a stack of plain GNNs can be directly applied to handle this situation. However, this thesis primarily concentrates on more challenging scenarios, namely heterogeneous static graphs (Chapter 3), heterogeneous discrete-time dynamic graphs (Chapter 4), and homogeneous continuous-time dynamic graphs (Chapter 5). Our novel GNN models, introduced in this thesis, demonstrate remarkable versatility, seamlessly adapting to scenarios ranging from straightforward static homogeneous networks to intricate dynamic heterogeneous structures. The heterogeneous continuous-time dyanmic graphs, the most complex situation, remains room for future exploration. Our proposed GNN models are specifically designed to offer a comprehensive understanding of complex social phenomena by capturing the dynamic nature of social interactions and the rich mix of information types.

Table 1.5: Real-world case studies involved in this thesis.

Case studies	Chapter 3	Chapter 4	Chapter 5
Prediction target	EMS demand	Popularity trends	Income of live streamers
Domain	Public health	Public interest	Commercial activity
Background	Tokyo metropolitan	Social media networks	Live streaming services
Categories	Human behaviors	Social trends	Human behaviors & Social trends

In another hand, we evaluate the aforementioned proposed GNNs through distinct real-world case studies, as listed in Table 1.5. The involved case studies include Emergency Medical Services (EMS) demand prediction (Chapter 3), popularity trends prediction in social media networks (Chapter 4), and forecasting the income of live streamers (Chapter 5).

The prediction of EMS demand in Tokyo metropolitan serves as a meaningful metric for public health outcomes, representing a vital aspect of human behaviors in social infrastructure services. The anticipation of popularity trends in social media networks (e.g., X, Instagram, and Reddit) provides insights into the future trajectory of public interests and concerns, indicating the social trends in the future. Forecasting the income in online live streaming services (e.g., YouTube Live and Twitch) reflects the dynamics of a significant commercial activity on the Internet. Live streaming services have experienced a surge, particularly in the post-COVID-19 era, becoming a prevalent mode for individuals to study, work, and earn a livelihood through online streaming. This scenario reflects a new combination of human behaviors and social trends.

In summary, this thesis comprises multiple subworks that introduce advanced GNNs tailored for challenging scenarios beyond the capabilities of traditional GNNs. Each subwork is associated with a real-world case study, providing a comprehensive assessment on the performance of proposed GNNs. These case studies, though limited in scope, serve as quintessential representatives of human behaviors and social trends. They effectively demonstrate the versatility and efficiency of our proposed GNN models across diverse domains. The evaluations conducted validate the robustness of our models and offer valuable insights into their practical utility in addressing complex challenges within the social sciences.

1.5 Thesis Outline

The thesis is organized as the following structure:

1. In Chapter 1, we offer a concise overview of the evolution of predictive analytics with AI techniques in human behaviors and social trends, including the

limitation of traditional methods and the advantages of graph models. Following this, we delve into the preliminaries of graphs and the GNN research, specifically focusing on the utilize of GNN in real-world situations. Building upon existing GNN approaches, the primary objective of this thesis is to propose novel GNN methods for efficiently learning entity representation embeddings within graph-structured real-world data. The ultimate goal is to enhance the ability of proposed GNN to interpret predictive analytics and address the challenging tasks within human behaviors and social trends.

2. In Chapter 2, we provide a comprehensive overview of the preliminary knowledge and background literature, delving into the fundamental knowledge of graphs and the conception of GNNs. Our exploration encompasses a discussion on the fundamental structure of GNNs and their alternative architectures tailored for various types of graphs. In addition, we introduce some advanced modules that enhance the performance of GNNs. Finally, we extend our inquiry to the practical downstream tasks of GNNs in predictive analytics. By synthesizing insights from existing research, we aim to establish a robust foundation for understanding the evolving landscape of GNNs and the subworks involved in this thesis.

3. Chapter 3 presents a model designed to learn node embeddings within a simplistic static bipartite graph. To illustrate the practical application of this model, we introduce a case study centered around predicting the EMS demand in the Tokyo metropolitan area. Our novel proposed model yields outstanding results in prediction accuracy, surpassing the performance of established baselines. This work is meaningful in helping public health management of local government.

4. In Chapter 4, we introduce an extended framework designed to efficiently learn temporal node embeddings within a chronologically ordered sequence of heterogeneous graph snapshots. This framework is tailored to capture evolving relationships and dynamics within the graphs. Furthermore, we showcase the practical application of the proposed method in predicting popularity trends within social media networks. This application is demonstrated through experiments conducted on several real-world social media networks, providing insights into the dynamic evolution of popularity trends and the efficiency of the proposed framework in capturing these dynamic patterns.

5. In Chapter 5, we present the extension of an innovative dynamic GNN method designed to effectively learn temporal node embeddings within intricate large-scale continuously-time dynamic graphs. Notably, this method is equipped to monitor the timing of specific graph actions, providing a nuanced understand-

ing of temporal dynamics compared to the work in Chapter 4. To illustrate the practical applicability of our proposed model, we introduce a income prediction task within live streaming services. This real-world application demonstrates the model's efficiency in modelling complex, time-varying graph structures and extracting valuable insights in online commercial activities.

6. In Chapter 6, we provide a comprehensive examination of the unified design considerations of proposed GNN frameworks. We delineate the relations among three subwork involved in this thesis. Besides, we test the extensibility of our proposed GNN frameworks on other datasets. Finally, we clarify the requirements of real-world human behaviors and social trends data that can be handled by our proposed GNN methods. By discussing these aspects, we aim to contribute to the responsible and thoughtful advancement of GNN research and application.

7. Finally, in Chapter 7, we answer the research questions proposed in Section 1.4.3, conclude the key contributions of this thesis, and provide an outlook into potential future directions.

1.6 List of Publications

Here, we provide a list of publications that this thesis focuses on.

Journals

- **Ruidong Jin**, Tianqi Xia, Xin Liu, Tsuyoshi Murata and Kyoung-Sook Kim. Predicting Emergency Medical Service Demand With Bipartite Graph Convolutional Networks. *IEEE Access*, vol. 9, pp. 9903-9915, 2021 [Jin et al., 2021].

- **Ruidong Jin**, Xin Liu, Tsuyoshi Murata. Predicting Potential Real-time Donations in YouTube Live Streaming Services via Continuous-time Dynamic Graph. *Machine Learning*, pp. 1-35, 2023 [Jin et al., 2023].

- **Ruidong Jin**, Xin Liu, Tsuyoshi Murata. Predicting Popularity Trend in Social Media Networks with Multi-layer Temporal Graph Neural Networks. *Complex & Intelligent Systems* (Accepted).

Conferences

- **Ruidong Jin**, Xin Liu, Tsuyoshi Murata. Predicting Potential Real-time Donations in YouTube Live Streaming Services via Continuous-time Dynamic Graph. In *25th International Conference on Discovery Science (DS' 2022)*,

Montpellier, France, October 10–12, 2022, Proceedings. Springer-Verlag, Berlin, Heidelberg, 59–73 (Oral presentation) [Jin et al., 2022].

Chapter 2

Background

This chapter provides foundational background knowledge crucial for the subsequent chapters. We commence by defining the fundamental graph structure and elucidate the alternatives for complex graphs frequently employed in modeling real-world datasets. Subsequently, we provide a brief introduction to GNNs, which are essential methods for comprehending the content of this thesis. Lastly, we introduce graph downstream tasks commonly utilized in predictive analytics. This preliminary background sets the stage for a deeper exploration of advanced methodologies and applications in the ensuing chapters.

2.1 A Taxonomy of Graphs

First, we give the definition of a graph:

Definition 2.1.1 (Graph)**.** *A graph is denoted as* $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{F})$*, where* $\mathbf{V}$ *represents the set of nodes (vertices) and* $\mathbf{E}$ *represents the set of edges (links). Graph is represented as an adjacency matrix* $\mathbf{A} \in \{0,1\}^{\|\mathbf{V}\|\times\|\mathbf{V}\|}$ *with each element* $A_{ij} = 1$ *if there exists an edge between node* u_i *and* u_j*, otherwise* $A_{ij} = 0$*. In some cases, each node is associated with a d-dimensional node feature (attribute) vector* $\mathbf{f}_i \in \mathcal{R}^d$*, and the feature matrix for all nodes is represented as* $\mathbf{F} \in \mathcal{R}^{\|\mathbf{V}\|\times d}$*.*

In what follows, we describe some common categories of graphs.

2.1.1 Undirected, Directed and Weighted Graph

For undirected graphs, their adjacency matrices are symmetric, i.e., $\mathbf{A}_{ij} = \mathbf{A}_{ji}$. Fig. 2.1 illustrates the respective adjacency matrices of a directed and an undirected graph.

Definition 2.1.2 (Directed graph)**.** *A directed graph (a.k.a digraph) is a graph in which edges have orientations, indicating a specific direction from one vertex to another. In a directed graph, the edges* e_ij *and* e_ji *are treated as distinct edges*

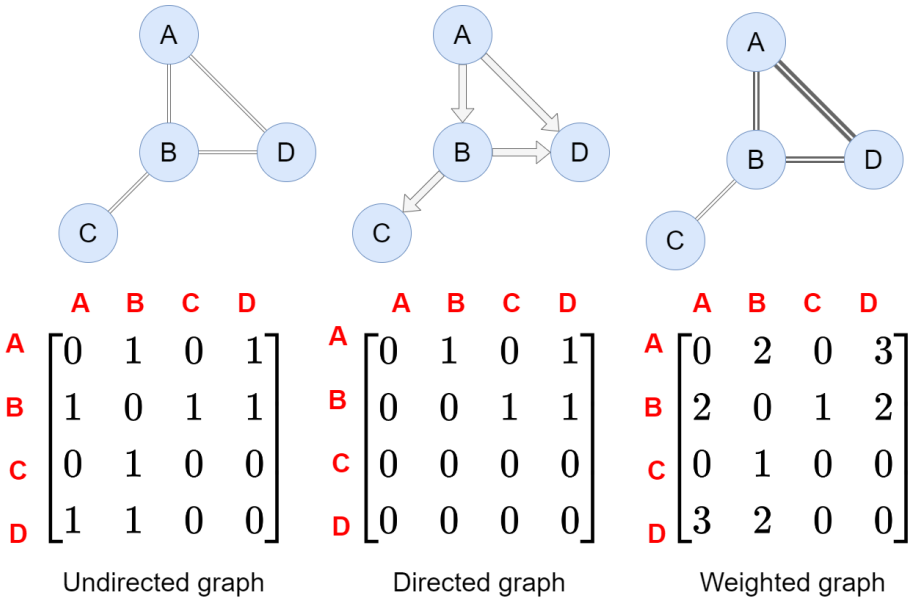


Figure 2.1: An undirected graph, a directed graph, and a weighted graph with their respective adjacency matrices.

with potentially different meanings. The adjacency matrix of the directed graph is asymmetric, i.e., $\mathbf{A}_{ij} \neq \mathbf{A}_{ji}$.

Besides, edges can be weighted. An unweighted graph uses a binary notation $\mathbf{A} \in \{0,1\}^{\|\mathbf{V}\|\times\|\mathbf{V}\|}$ within its adjacency matrix, indicating the existence (1) or absence (0) of edges. A weighted graph is one in which a number (the weight) $\mathbf{A} \in \mathcal{R}^{\|\mathbf{V}\|\times\|\mathbf{V}\|}$ is assigned to each edge. As shown in Fig. 2.1, the adjacency matrix of a weighted graph is populated with a value of edge weight instead of binary representation. The weights in weighted graphs encapsulate diverse metrics such as costs, lengths, or capacities, depending on the specific problem under consideration.

2.1.2 Homogeneous & Heterogeneous graph

Graphs can further be categorized as either homogeneous or heterogeneous based on the nature of their nodes and edges.

Definition 2.1.3 (Homogeneous graph)**.** *A homogeneous graph is a graph with a single type of node and a single type of edge.*

As shown in Fig. 2.2, in a homogeneous graph, uniformity prevails, with all nodes representing instances of the same type and all edges denoting relations of the same type. This uniformity ensures that the graph encapsulates a singular entity type, fostering a cohesive representation. For example, a social network where nodes correspond to individuals and edges signify connections between them. The graph is homogeneous in this context as it revolves around a single entity type—people and their relationships.

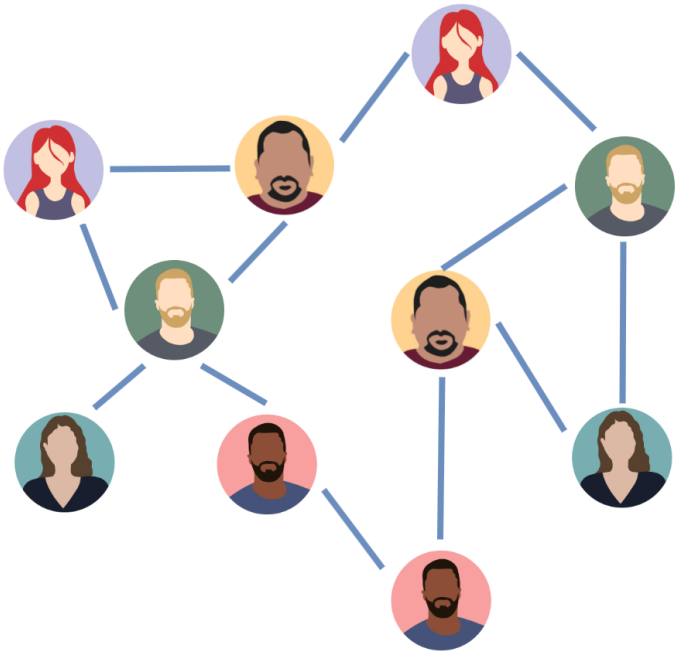


Figure 2.2: A homogeneous graph. All nodes and all edges are of the same type (e.g., all the nodes are users).

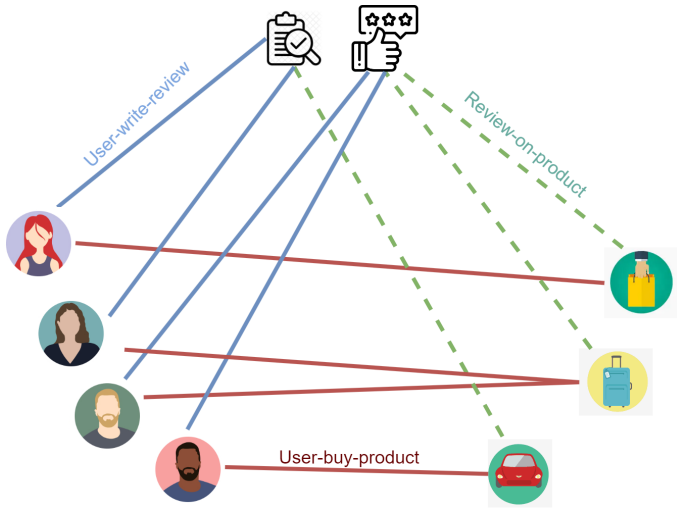


Figure 2.3: A heterogeneous graph. Nodes and/or edges have multiple types (e.g., nodes include users, reviews, and products. Edges include user-write-review, review-on-product, and user-buy-product).

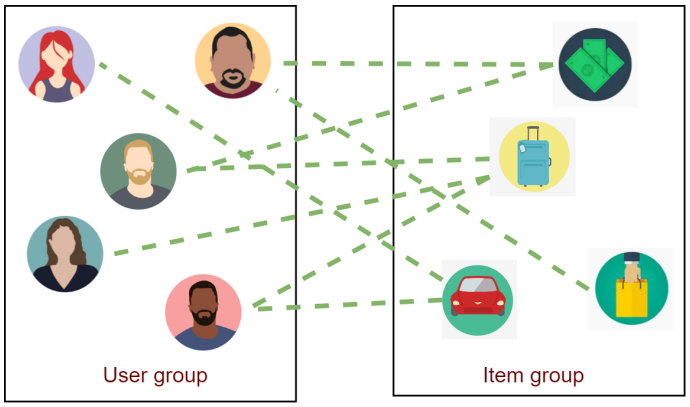


Figure 2.4: A bipartite graph. Nodes are divided into two disjoint user group and item group, and edges only connect nodes from two different groups.

Definition 2.1.4 (Heterogeneous graph)**.** *A heterogeneous graph is a graph with two or more types of nodes and/or two or more types of edges.*

In contrast, heterogeneous graphs present a more diverse structure where nodes and edges can belong to different types. As shown in Fig. 2.3, consider a graph modeling the relationships within online shopping, where the nodes represent distinct entities of users, products, and reviews. In this heterogeneous graph, the edges have various types such as *review-on-product* (green dash lines), *user-buy-product* (red solid lines), *user-write-review* (blue solid lines), and so on, reflecting the multifaceted relationships between different entities. The versatility of heterogeneous graphs allows for a richer representation of complex systems where diverse entities interact through many relationships, making them particularly well-suited for capturing the intricate dynamics of interconnected ecosystems.

Definition 2.1.5 (Bipartite graph)**.** *A bipartite graph* $\mathbf{G} = (\mathbf{U}, \mathbf{V}, \mathbf{E})$ *is a graph with nodes divided into two disjoint sets* $\mathbf{U}$ *and* $\mathbf{V}$ *such that the edge* $e_{(u,v)} \in \mathbf{E}, u \in \mathbf{U}, v \in \mathbf{V}$ *only connects nodes from one set to the other.*

The bipartite graph is a distinctive and frequently employed subtype of heterogeneous graphs characterized by edges exclusively between nodes of two distinct types. This specialized structure is precious in scenarios where relationships exist solely between entities of different classes. As shown in Fig 2.4, a recommender system, where user interactions with items, can be aptly represented using a bipartite graph. In this context, one set of nodes corresponds to users, the other set represents items, and edges establish connections based on user-item interactions. The bipartite graph model proves instrumental in capturing and analyzing such relationships, providing a precise and efficient representation that aligns with the inherent structure of diverse real-world systems, especially those involving interactions between entities of distinct types.

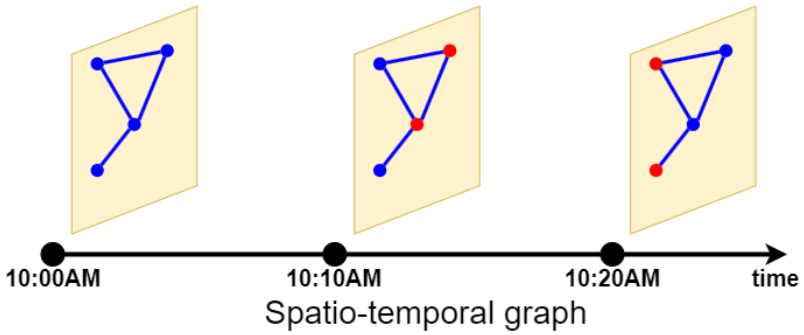


Figure 2.5: A spatio-temporal graph. Different node colors denote the change of node features.

2.1.3 Dynamic Graph

Definition 2.1.6 (Dynamic (temporal) graph)**.** *A dynamic (temporal) graph is one in which nodes and/or edges change with time.*

Real-world graphs often exhibit dynamic behaviors, evolving over time to capture the changing relationships and interactions within complex systems. Such dynamics are particularly evident in domains such as social networks, financial transactions, and recommender systems. Unlike static graphs, which provide a snapshot of relationships at a single point in time, dynamic graphs encapsulate the temporal evolution of connections, offering valuable insights into the evolving nature of these real-world situations.

Learning on dynamic graphs presents considerably greater complexity compared to static graphs. Different applications give rise to different types of dynamic graphs and prediction problems. Thus, it is crucial to identify the type of dynamic graph and its static and evolving parts and clearly understand the prediction problem. As pointed out in [Kazemi et al., 2020], dynamic graphs can be divided into spatio-temporal, discrete-time, and continuous-time categories.

Definition 2.1.7 (Spatio-temporal graph)**.** *A spatio-temporal graph is a dynamic graph* $\{\mathbf{G}^{(1)}, \mathbf{G}^{(2)}, \cdots, \mathbf{G}^{(T)} | \mathbf{G}^{(t)} = \{\mathbf{V}, \mathbf{E}, \mathbf{F}^{(t)}\}\}$ *made of static graph structures* $(\mathbf{V}, \mathbf{E})$ *and time-varying features* $(\mathbf{F}^{(t)})$*, where* t *denotes the timestamp.*

Spatio-temporal is a combination of two words, where "spatio" refers to space and "temporal" refers to time. As illustrated in Fig. 2.5, spatio-temporal graphs are a fundamental type of dynamic graphs. The topology in spatio-temporal graphs remains fixed, but the node and/or edge features change over time. Any system that comprises static structural relationships and dynamic time information can be considered a spatio-temporal graph, such as traffic patterns and network loads. However, it is important to note the limitation of spatio-temporal graphs, as they cannot handle changes in graph topology.

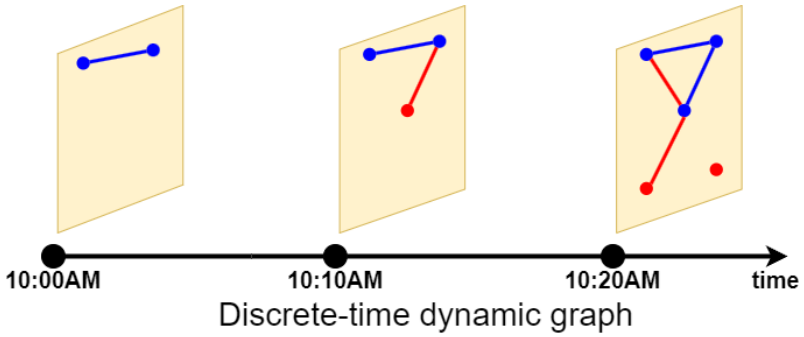


Figure 2.6: A discrete-time dynamic graph.

Definition 2.1.8 (Discrete-time dynamic graph (DTDG)). *A discrete-time dynamic graph is consist of a timed sequence of static graph snapshots* $\{\mathbf{G}^{(1)}, \mathbf{G}^{(2)}, \cdots, \mathbf{G}^{(T)}\}$, *where each graph snapshot* $\mathbf{G}^{(t)} = \{\mathbf{V}^{(t)}, \mathbf{E}^{(t)}, \mathbf{F}^{(t)}\}$ *has node set* $\mathbf{V}^{(t)}$, *edge set* $\mathbf{E}^{(t)}$ *and feature matrix* $\mathbf{F}^{(t)}$ *at a specific timestamp* t. *The graph snapshots are captured at regularly-spaced intervals.*

As illustrated in Fig. 2.6, a DTDG embodies changes in both topology and features over time. The temporal dynamics are captured through a sequence of static graph snapshots observed at regular intervals. DTDG fits any system that is observed at regular intervals. However, it is essential to note that these methods do not provide insights into the events occurring between two consecutive observations.

The challenge lies in understanding the continuous evolution of the graph, discerning the transitions, and predicting the intermediate states. This temporal information gap necessitates the development of dynamic graph models that go beyond static snapshot analysis, delving into the nuanced changes occurring between observed time points. Addressing this temporal granularity is crucial for capturing the full spectrum of dynamic interactions and evolutions within the graph structure.

Definition 2.1.9 (Continuous-time dynamic graph (CTDG)). *A continuous-time dynamic graph is a pair* $(\mathbf{G}^{(\mathbf{t_0})}, \mathbf{S})$, *where* $(\mathbf{G}^{(\mathbf{t_0})} = \{\mathbf{V}^{(t_0)}, \mathbf{E}^{(t_0)}, \mathbf{F}^{(t_0)}\}$ *signifies a static graph representing the initial state at timestamp* t_0, *and* $\mathbf{S}$ *constitutes a timed sequence of graph actions. These graph actions encompass a spectrum of transformations, including edge creation or deletion, node creation or deletion, and the evolution of node or edge statuses. Each individual graph action is temporally tagged, providing a precise timestamp that indicates when the action occurred in the continuous timeline.*

CTDG is a more general style of dynamic graph. As illustrated in Fig. 2.7, this formulation encapsulates the dynamic nature of the graph, illustrating not only its instantaneous state at the initial timestamp but also the sequence of each change in the graph that unfolds over time. The graph actions, with their associated timestamps, serve as a comprehensive representation of the continuous evolution of the

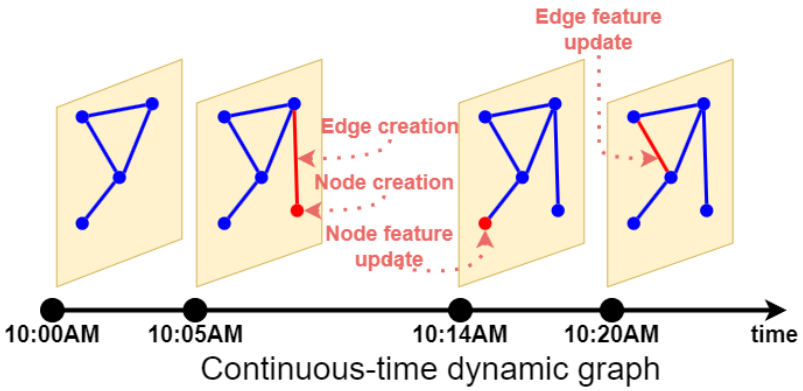


Figure 2.7: A continuous-time dynamic graph.

dynamic graph, enabling a nuanced understanding of the temporal dynamics within the complex system, such as social networks, interaction networks, financial transaction networks, and more.

2.2 Graph Neural Networks

The rapid advancement of technologies has ushered in an era where vast amounts of data are readily available. Most of this data exhibits a non-Euclidean or irregular structure, defying conventional methods for regular data structures. In response to this complexity, graphs have emerged as a powerful tool for processing irregular data structures. The analysis of graph-structured data proves invaluable, revealing latent attributes and uncovering missing information that might otherwise remain obscured.

Graphs provide a flexible framework for representing relationships and dependencies in various domains, including social networks, biological systems, transportation networks, and recommendation systems. By embracing the inherent irregularity and connectivity of real-world data, graph-based approaches enable a more nuanced understanding of complex systems.

Graph embedding methods transform complex graph attributes, including nodes and edges, into low-dimensional embedding vectors while maximally preserving the essential graph structural information. These methods have consistently demonstrated superior performance compared to traditional techniques when it comes to modeling graph-structured data [Cai et al., 2018; Cui et al., 2018; Zhang et al., 2018b; Zhou et al., 2018]. Graph Neural Network (GNN) is an extension and evolution of deep learning-based methods for analyzing graph data. GNNs are designed to perform optimized transformations on all graph attributes while preserving graph topology [Wu et al., 2019b; Zhang et al., 2019b]. As stated in Definition 2.1.1, a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ consists of a set of nodes $\mathbf{V}$ and a set of edges $\mathbf{E}$. $u \in \mathbf{V}$ and $v \in \mathbf{V}$ represent nodes in $\mathbf{V}$, and $e_{(u,v)}$ represents the edge connecting node u and

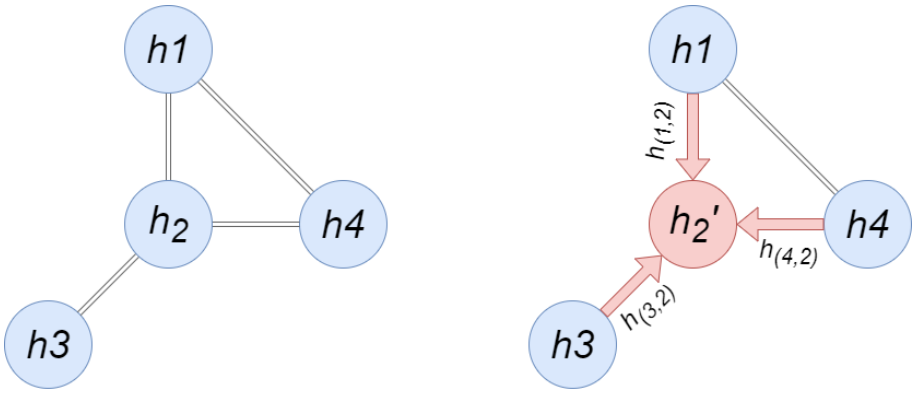


Figure 2.8: Illustration of the message passing update within a GNN with four nodes. In the left figure, node embeddings $\mathbf{h}_1 \sim \mathbf{h}_4$ are assigned to each node. Focusing the center node $\mathbf{h}_2$, intermediate embeddings $\mathbf{h}_{(1,2)}$, $\mathbf{h}_{(3,2)}$ and $\mathbf{h}_{(4,2)}$ are produced using nearby node emebddings $\mathbf{h}_1$, $\mathbf{h}_3$ and $\mathbf{h}_4$, node features $\mathbf{f}_1$, $\mathbf{f}_3$ and $\mathbf{f}_4$, and edge weights (if present) in the message passing step. The right figure shows that the updated node embedding $\mathbf{h'}_u$ is an aggregation of nearby intermediate embeddings.

v. Nodes are usually associated with feature vectors $\mathbf{f}_u \in \mathbf{F}$. Initially, an instance of a graph $\mathbf{G}$ and the corresponding node features are fed to be the input of GNN. $\mathbf{h}_u$ denotes the hidden representation embeddings for node u. We use $\mathbf{h}_u = \mathbf{f}_u$ as the initialization. The graph's structure dictates the message passing and node updates to get updated node embedding $\mathbf{h'}_u$.

Fig. 2.8 provides an illustration of the message passing and update processes within a GNN. Typically, the message propagation in a GNN involves two crucial steps: message passing and node update. These steps collectively contribute to the iterative refinement of the hidden embeddings for each node within the graph. The message passing operation is responsible for updating the hidden embedding $\mathbf{h}_u^{(n+1)}$ of each node u in every iteration. Message passing revolves around aggregating messages based on some pooling strategy (e.g., max pooling, mean pooling, and more) from the neighbor nodes $\mathcal{N}_u$ of node u. The aggregated messages are then employed to update the previous node embedding $\mathbf{h}_u^{(n)}$ through a non-linear transformation. The message passing aggregation and update operations can be mathematically expressed as follows:

$$\mathbf{h_u^{(n+1)}} = Update(\mathbf{h_u^{(n)}}, M_{\mathcal{N}_u}^{(n)}), \tag{2.1}$$

$$M_{\mathcal{N}_u}^{(n)} = Aggregate(\{\mathbf{h_v^{(n)}}, v \in \mathcal{N}_u\}), \tag{2.2}$$

where both *Update* and *Aggregate* functions are arbitrarily differentiable and usually implemented as neural networks. $M_{\mathcal{N}_u}^{(n)}$ denotes the aggregated messages from the neighbor nodes $M_{\mathcal{N}_u}^{(n)}$ of node u at the n-th step in the iterations. The *Update* function then combines the message $M_{\mathcal{N}_u}^{(n)}$ with the previous node embedding $\mathbf{h}_u^{(n)}$

to produce the updated embeddings $\mathbf{h}_u^{(n+1)}$. The output of the last layer is usually used to define the learned node embeddings after performing n-th iterations of the GNN layers and given as:

$$\mathbf{z}_u = \mathbf{h}_u^{(n)}, u \in \mathbf{V}. \tag{2.3}$$

In a general perspective, GNNs encode the graph structure $\mathbf{A}$ and node features $\mathbf{F}$ into node embeddings $\mathbf{Z}$, which can be represented by the following equation:

$$\mathbf{Z} = \mathcal{F}\left(\mathbf{A}, \mathbf{F} | \mathbf{\Theta}\right), \tag{2.4}$$

where $\mathcal{F}$ denote the GNN encoder and $\mathbf{\Theta}$ denote the trainable parameter.

Prior research has commonly employed two types of strategies in the aggregation operation: mean-pooling and attention mechanism. Mean-pooling involves averaging the information from a central node's neighbors, while attention mechanisms assign different weights to the neighbor nodes based on their relevance to the central node. In the update operation, various strategies have been explored to refine the central node's representation, including concatenation, summation, and Gated Recurrent Unit (GRU). Besides, there are two basic types of Graph Neural Network (GNN) models: spectral models and spatial models. Spectral models leverage graph convolution to address graph-related tasks, whereas spatial models extract features through iterative aggregation of a central node's neighbors. The critical steps of both models involve iteratively collecting information from a central node's neighbors and aggregating this information to derive a high-level representation of the node. In the following, we briefly introduce several noteworthy GNN models.

The Graph Convolutional Network (GCN) is a prominent spectral model and has gained widespread recognition as one of the most widely adopted models within GNN frameworks [Kipf and Welling, 2016a]. Comprising several concatenated graph convolutional layers, GCN has demonstrated remarkable efficiency in aggregating information from neighboring nodes. The graph convolutional layer in GCN is typically formulated using the following equation:

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{L}^{\mathrm{GCN}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{2.5}$$

$$= \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)} \tag{2.6}$$

where $\mathbf{L}$ is the aggregation matrix, $\mathbf{H}^{(l)}$ is the node embedding matrix in l-th layer, $\mathbf{H}^{(0)}$ is initialized with the node feature matrix, $\mathbf{W}^{(l)}$ is the trainable weight matrix in l-th layer, and $\sigma(\cdot)$ is an activation function. GCN is originally designed for homogeneous graphs, and it adopts the symmetric normalized Laplacian matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ as the aggregator $\mathbf{L}^{\mathrm{GCN}}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree diagonal

matrix of $\tilde{\mathbf{A}}$. $\tilde{\mathbf{A}}$ represents the graph with a self-connection on each node.

GraphSage extends GCN to accommodate inductive learning, enabling effective handling of unknown graph nodes [Hamilton et al., 2017b]. In GraphSage, the approach involves sampling neighbor nodes of the central node and aggregating information from these sampled neighbors to update the embedding of the central node through several aggregators. This inductive learning strategy enhances the model's ability to generalize to new, unseen nodes within the graph. The aggregation process in GraphSage is formulated by the following equations:

$$\mathbf{h}_{\mathcal{N}_u}^{(n)} = Aggregate(\{\mathbf{h}_{\mathbf{v}}^{(\mathbf{n})}, v \in \mathcal{N}_u\}), \tag{2.7}$$

$$\mathbf{h}_{\mathbf{u}}^{(\mathbf{n+1})} = \sigma(\mathbf{W}^{(n)} \left[\mathbf{h}_{\mathbf{u}}^{(\mathbf{n})} \| \mathbf{h}_{\mathcal{N}_u}^{(n)}\right]), \tag{2.8}$$

where u is the central node, $\mathcal{N}_u$ is the sampled neighbors of u, $\mathbf{W}^{(n)}$ is the weight matrix of layer n, and *Aggregate* denotes aggregation options, such as mean, sum and max.

Graph Attention Networks (GAT) represent a significant advancement by incorporating attention mechanisms, assigning importance weights to different neighbor nodes [Veličković et al., 2018]. The attention mechanism enhances the model's ability to discern the varying importance of neighbors, making it particularly effective in scenarios where specific nodes play a more critical role in influencing the central node's characteristics. The aggregation process in GAT is formulated by the following equations:

$$\mathbf{h}_{\mathbf{u}}^{(\mathbf{n+1})} = \sigma\left(\sum_{v \in \mathcal{N}_u} \alpha_{u,v} \mathbf{W}^{(n)} \mathbf{h}_{\mathbf{v}}^{(\mathbf{n})}\right), \tag{2.9}$$

$$\alpha_{u,v} = \frac{\exp(LeakyReLU(\alpha^T[\mathbf{W}^{(n)}\mathbf{h}_{\mathbf{u}}^{(\mathbf{n})} \| \mathbf{W}^{(n)}\mathbf{h}_{\mathbf{v}}^{(\mathbf{n})}]))}{\sum_{k \in \mathcal{N}_u} \exp(LeakyReLU(\alpha^T[\mathbf{W}^{(n)}\mathbf{h}_{\mathbf{u}}^{(\mathbf{n})} \| \mathbf{W}^{(n)}\mathbf{h}_{\mathbf{k}}^{(\mathbf{n})}]))}, \tag{2.10}$$

where $\alpha_{u,v}$ is the weight from node j to node u, $\mathcal{N}_u$ is the neighbors of node u and the attention mechanism is a fully connected neural network by a learnable vector α through softmax function.

To further bolster the performance of GNNs, researchers have introduced advanced models that build upon the foundational architecture of GCN. Based on the encoder-decoder paradigms, Graph Autoencoder and Variational Graph Autoencoder empower GNNs to undertake unsupervised learning tasks [Kipf and Welling, 2016b]. Moreover, the landscape of GNNs is enriched by extensions such as ML-GCN and ML-GAT, which elevate the original GCN and GAT to multi-layer networks [Zangari et al., 2021]. This extension enables these models to capture more intricate

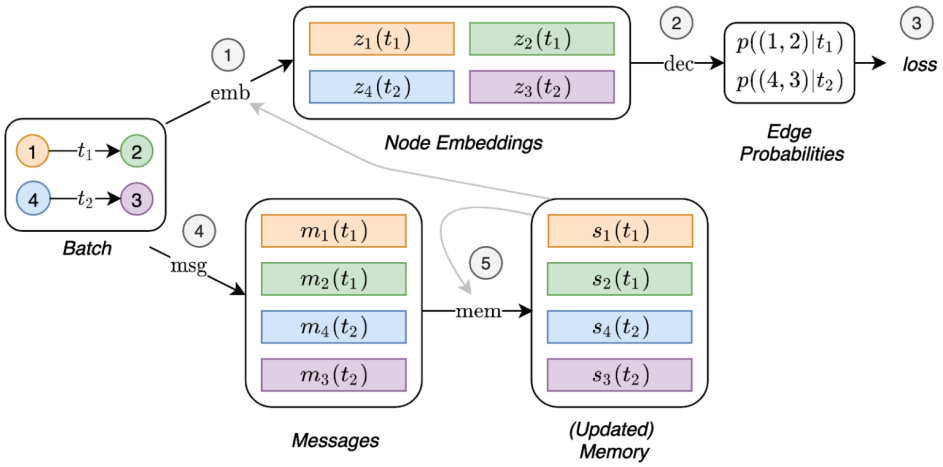


Figure 2.9: The overview of TGN model. This figure is cited from [Rossi et al., 2020].

relations among nodes within expansive graphs, contributing to their effectiveness in handling large-scale and complex graph structures. The continuous evolution and diversity of GNN architectures underscore their versatility and adaptability across various graph-related tasks.

2.2.1 GNNs on Dynamic Graphs

Learning embedding with GNNs on dynamic graphs is much more complex than on static graphs. Initially, research on dynamic graphs focused on spatio-temporal graphs and DTDG, which consists of a timed sequence of graph snapshots [Gao et al., 2022a; Liben-Nowell and Kleinberg, 2007; Sankar et al., 2020]. Existing static graph methods can be directly applied to each graph snapshot, and then concatenate the respective results in each graph snapshots together. However, most real-life graph-structured data is in a state of constant evolution. A more general style of dynamic graph is the CTDG, which consists of a timed list of graph events, including edge creation or deletion, node creation or deletion, and node or edge status evolution. Recently, several studies on CTDG have been proposed, such as JODIE [Kumar et al., 2019], Continuous-time Dynamic Network Embedding [Nguyen et al., 2018], DyRep [Trivedi et al., 2019], Temporal Graph Networks (TGN) [Rossi et al., 2020], Temporal Graph Attention (TGAT) [Xu et al., 2020], Asynchronous Propagation Attention Network (APAN) [Wang et al., 2021], Meta-learning framework MetaDyGNN [Yang et al., 2022a], and more.

Typically, TGN is a generic, efficient framework for deep learning operating on CTDGs by incorporating an originally designed memory modules and graph-based operators. Fig. 2.9 demonstrates the flow of operations that TGN used to train the memory-related modules. *Raw Message Store* stores the necessary raw information

to compute messages for interactions the model has previously processed. It allows the model to delay the memory update brought by an interaction with later batches. At first, the memory is updated using messages computed from raw messages stored in previous batches (1, 2, 3). The embeddings can then be computed using the just updated memory (grey link) (4). By doing this, the computation of the memory-related modules directly influences the loss (5, 6), and they receive a gradient. Finally, the raw messages for this batch interactions are stored in the raw message store (6) to be used in future batches.

2.2.2 GNNs on Heterogeneous Graphs

There is extensive research on learning embeddings with GNNs for heterogeneous graphs, which contain multi-typed nodes and/or multi-typed edges . Metapath2vec uses meta-path based random walks to aggregate the heterogeneous neighborhood nodes and then exploit a skip-gram model to learn the node embeddings [Dong et al., 2017]. HAN is a heterogeneous graph neural network based on the hierarchical attention mechanism, including node-level and semantic-level attention [Wang et al., 2019a]. It fully considers the importance of node neighbors and different meta-paths. HetGNN combines heterogeneous structural information and node attributes [Zhang et al., 2019a]. It performs excellently in graphs with multiple types of nodes and edges.

Note that bipartite graphs that consist of two disjoint sets of nodes can be regarded as a particular class of heterogeneous graphs. The bipartite graph is a ubiquitous data structure to model the relationship between two types of entities. Some bipartite graphs include the author-paper graph, the customer-product graph, the player-event graph, the actor-movie graph, and the keyword-document graph. BiNE is an approach by performing biased random walks and preserving the long-tail distribution of nodes in bipartite graphs [Gao et al., 2018]. BGNN is another approach that utilizes inter-domain message passing and intra-domain alignment towards information fusion [He et al., 2019]. However, these two methods only employ the graph structure information, not the node feature information. Besides, both are unsupervised learning methods that only partially utilize the training set to optimize the models, resulting in suboptimal performance.

2.3 Graph Downstream Tasks and Applications in Predictive Analytics

Using graph-based methods to tackle predictive analytics is a relatively recent approach. As listed in Fig. 2.10, most graph-related tasks are classified into one of those tasks:

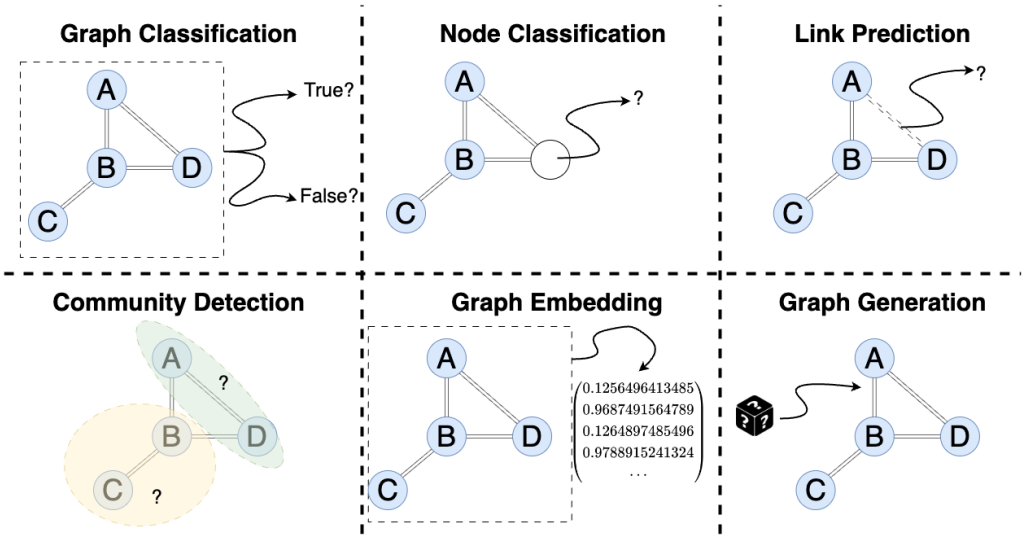


Figure 2.10: Common downstream tasks of GNN.

- **Graph Classification**: This task involves categorizing graphs into different classes and finding applications in areas such as social network analysis and text classification.
- **Node Classification**: In node classification, the objective is to predict missing node labels by leveraging information from neighboring nodes in the graph.
- **Link Prediction**: Link prediction aims to predict the presence or absence of edges between pairs of nodes in a graph with an incomplete adjacency matrix. This task is commonly applied in social network analysis.
- **Community Detection**: Community detection involves partitioning nodes into distinct clusters based on the graph's edge structure. It learns patterns from edge weights, distances, and other graph characteristics.
- **Graph Embedding**: Graph embedding tasks focus on mapping graphs into vectors while preserving essential information about nodes, edges, and overall structure. It facilitates downstream machine-learning applications.
- **Graph Generation**: Graph generation tasks involve learning from a distribution of sample graphs to generate new graph structures that are similar but not identical to the training data. It is particularly useful in scenarios where new, diverse graphs need to be created.

Time-aware predictive analytics have recently attracted much attention in both academia and industry [Altshuler et al., 2012; Rousidis et al., 2020; Zeng et al., 2013]. Existing works about predictive analytics can be categorized into two primary patterns: The first is to predict the growth and decline of entities based on past characteristics and early-stage patterns. Yang et al. found that the temporal

patterns reveal how the content popularity fluctuates during the post-propagation [Yang and Leskovec, 2011]. The second is to predict the value of specific target attributes based on temporal attributes or dynamic signals. Zhao et al. incorporated human reaction time as temporal variables in self-exciting point processes [Zhao et al., 2015]. He et al. designed a time-aware bipartite graph for estimating [He et al., 2014]. Cao et al. developed a coupled GNN model to solve the popularity trend prediction task [Cao et al., 2020]. Li et al. adopted a graph kernel approach to predict the node popularity within a cascade graph sequence [Li et al., 2021a]. Hou et al. proposed a spatial–temporal multi-graph convolutional network for casualty prediction of terrorist attacks [Hou et al., 2023]. Yang et al. predicted traffic propagation flow in urban road networks with a multi-graph convolutional network model [Yang et al., 2023].

2.4 Advanced Modules in GNNs

In this section, we introduce some advanced modules to improve the performance of GNNs, which are commonly utilized in prior work and involved in our proposed GNN frameworks. Graph Structure Learning (GSL) enhances the quality of input graphs. The attention mechanism introduced in graph models improves the efficiency of message aggregation and node updating. Task-specific decoders and loss functions contribute to model training.

2.4.1 Graph Structure Learning

Noisy or incomplete graphs can often lead to suboptimal representations, hindering the efficient learning of node embeddings, especially when dealing with real-world graph data. A primary concern of improving robustness of GNN models is to produce a denoised graph structure for learning representations [Jin et al., 2020]. Recently, much literature has emerged around the central theme of GSL, which aims at jointly learning an optimized graph structure and corresponding representations [Yu et al., 2021].

Generally, a graph is denoted as $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{F})$, where $\mathbf{V}$ represents the set of nodes (vertices) and $\mathbf{E}$ represents the set of edges (links). Node feature matrix is denoted as $\mathbf{F} \in \mathcal{R}^{\|\mathbf{V}\| \times d}$. The original graph structure is represented as an adjacency matrix $\mathbf{A} \in \{0,1\}^{\|\mathbf{V}\| \times \|\mathbf{V}\|}$ for binary graphs or $\mathbf{A} \in \mathcal{R}^{\|\mathbf{V}\| \times \|\mathbf{V}\|}$ for weighted graphs. A GNN encoder $\mathcal{F}(\mathbf{A}, \mathbf{F})$ receives the graph structure and node features as input and produces node embeddings $\mathbf{Z}$ in low dimensions for downstream tasks. The goal of GSL is to learn a refined adjacency matrix $\mathbf{A}^*$ and corresponding node embeddings:

$$\mathbf{Z}^* = \mathcal{F}(\mathbf{A}^*, \mathbf{F}). \tag{2.11}$$

Most existing GSL work could be regarded as a plug-in module on top of other GNN models. The main methods of obtaining the refined graph structure fall into modifying the edge sets by adding new edge, removing existing edges, and changing edge weights [Zhu et al., 2021b]. The previous studies mainly consist of three components: structure modeling, message propagation, and learning objectives.

Structure modeling

The core conception of GSL is an encoding function that generates the optimal refined graph structure $\mathbf{A}^*$, defined by the edge weights via pairwise distance or learnable parameters [Li et al., 2018; Wang et al., 2020; Wu et al., 2018; Yu et al., 2021].

Message propagation

After obtaining an optimized graph structure $\mathbf{A}^*$, node features are then utilized to refine neighborhoods via GNN encoders. Notably, given the complexity of optimizing the graph structure, many approaches iteratively repeat structure modeling and message propagation until several stopping conditions are satisfied.

Learning objective

To train the model with the refined graph structures, most of existing GSL methods defines a learning objective containing two parts:

$$\mathcal{L} = \mathcal{L}_{task}(\mathbf{Z}^*, \mathbf{Y}) + \lambda \mathcal{L}_{reg}(\mathbf{A}^*, \mathbf{A}), \tag{2.12}$$

where $\mathcal{L}_{task}$ refers to a task-specific objective with respect to the ground truth label $\mathbf{Y}$ (e.g., cross entropy for classification tasks, MSE loss for regression tasks), and $\mathcal{L}_{reg}$ regularizes the learned graph structure $\mathbf{A}^*$ to meet some prior topology constraints. λ is a compromising hyperparameter that balances the two terms.

In summary, GSL enhances the quality of input graphs and contributes to learning robust representations with GNN models. In Chapter 4, the proposed GNN framework is implemented with a GSL module for refining the noises in input graphs.

2.4.2 Attention Mechanism

The attention mechanism in machine learning intuitively imitates cognitive attention, enabling models to focus on specific parts of the input sequence when making predictions. It assigns different "weights" to different parts of the input according to their dependencies and relationships, emphasizing more on the relevant information for a given task.The attention mechanism has found widespread use in various tasks, especially in natural language processing (NLP) and computer vision (CV).

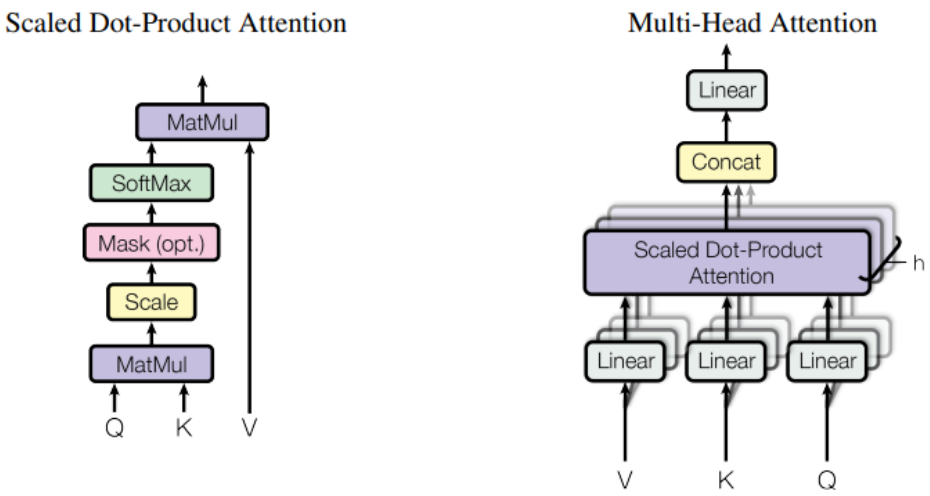


Figure 2.11: An overview of the multi-head attention module. Left is the Scaled Dot-Product Attention. Right is the Multi-head Attention consists of several attention layers running in parallel. This figure is cited from [Vaswani et al., 2017].

Additionally, many attention-based GNN models leverage this mechanism to infer neighbor importance, manipulating the weight of their propagation in the message aggregation and node updating processes [Chairatanakul et al., 2021; Guo et al., 2019; Sankar et al., 2020; Veličković et al., 2017].

Transformer and multi-head attention

The Transformer architecture has gained prominence for its effectiveness in handling sequential data, particularly in natural language processing tasks. Unlike traditional recurrent models that process input sequences sequentially, Transformers leverage attention mechanisms to capture global dependencies between input and output elements in parallel [Vaswani et al., 2017]. This approach has demonstrated superior performance compared to RNN methods. The success of Transformers in applications such as machine translation, speech recognition, and large-language models (LLMs) underscores the effectiveness of the attention mechanism, leading to increased focus and exploration by deep learning research.

The multi-head attention mechanism is a central component of the Transformer architecture, as illustrated in Fig. 2.11. In the Transformer model, the attention module performs its computations multiple times in parallel, with each instance referred to as an attention head. Within this module, the Query, Key, and Value parameters are split N ways, and each split is independently processed through a distinct attention head. The results of these parallel attention calculations are then combined to produce a final attention score. The computation formula for multi-head attention can be expressed as:

$$head_i = Attn\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) \tag{2.13}$$

$$= softmax\left(\frac{\mathbf{Q}\mathbf{W}_Q(\mathbf{K}_i\mathbf{W}_K)^T}{\sqrt{d_k}}\right)\mathbf{V}\mathbf{W}_V \tag{2.14}$$

$$MultiHead\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = Concat(head_1, head_2 \cdots, head_h)\mathbf{W}, \tag{2.15}$$

where $\mathbf{Q}$ denotes the query vectors, $\mathbf{K}$ and $\mathbf{V}$ denote the key-value pairs, and d_k denotes the dimension of keys. The attention function serves as a mapping from a query and a set of key-value pairs to an output. The output is determined by a weighted sum of the values, with the weights assigned based on a compatibility function that considers the relationship between the query and the corresponding key. The incorporation of the Multi-head attention module in the Transformer enhances its capacity to capture diverse relationships and intricacies for each objects in the input sequence. The versatility of the multi-head attention module allows for seamless extension to various domains.

Attentions in graph models

Introducing self-attention in GNNs allows each node to dynamically weigh its own features and those of its neighbors differently, enabling the model to capture both local and global dependencies. GAT, as a representative work in this domain, employs masked self-attentional layers to address the limitations of earlier methods relying on graph convolutions or their approximations [Veličković et al., 2017]. Through stacked layers, where nodes can attend to their neighborhoods' features, GAT implicitly assigns varying weights to different nodes in a neighborhood without the need for computationally expensive operations like matrix inversion or a priori knowledge of the graph structure.

Multi-head attention, initially popularized in Transformer models designed for sequence-based tasks, has been extended to GNNs to augment their capacity for capturing intricate relationships and dependencies in graph-structured data. The concept involves employing multiple attention heads to independently grasp different facets of the relationships between nodes in a graph. This enables the model to simultaneously attend to various patterns and interactions, thereby enhancing its capabilities for learning representations. The implementation of multi-head attention is prevalent in dynamic GNN models [Jin et al., 2022; Rossi et al., 2020; Wang et al., 2021; Xu et al., 2020], as the parallel processing is efficient in handling large batches of graph changes. In Chapter 4 and Chapter 5, the proposed GNN framework is implemented with a multi-head attention layer for enhancing the efficiency of learning temporal node embeddings.

2.4.3 Decoders and Loss Functions

The primary objective of GNN frameworks is to learn node embeddings. These learned node embeddings can then be employed in various downstream tasks through the use of different decoders and loss functions. The selection of the decoder and loss function is task-dependent. In this section, we introduce some commonly used decoders and loss functions in GNN frameworks.

Support vector machine decoder

Support Vector Machine (SVM) is a traditional classifier for supervised binary classification tasks [Hearst et al., 1998]. Binary classification tasks are common in graphs, such as binary node classification and link prediction [Jalili et al., 2017; Jin et al., 2021; Yuan et al., 2019]. SVM is one of the most studied machine learning algorithms. SVM can efficiently perform linear classification with a hard margin and non-linear classification with a soft margin and kernels. In the case of graph embedding learning, most of the learned embeddings are non-linearly separable. Thus, SVM with a soft margin using the hinge loss function is usually employed:

$$\frac{1}{N}\max\left(0, 1 - \mathbf{Y}(\mathbf{W}^T\mathbf{X} - \mathbf{b})\right) + \lambda\|\mathbf{W}\|_2^2, \tag{2.16}$$

where $\mathbf{Y}$ denotes the true labels, $\mathbf{X}$ denotes the predicted labels, and $\mathbf{b}$ denotes the biases. N is the total number of prediction samples. The parameter λ determines the trade-off between increasing the margin size and ensuring the l2-regularization of trainable parameter $\mathbf{W}$. By deconstructing the hinge loss, this optimization problem can be massaged into minimizing the objective functions above. In Chapter 3, the proposed model is employed with an SVM decoder for predicting the binary edge labels.

Softmax function decoder

Softmax function is usually used as the decoder for classification tasks such as node classification, graph classification and link prediction tasks. The softmax function takes a k-dimensional vector as input, and normalizes it into a probability distribution consisting of k probabilities proportional to the exponentials of the input numbers. Each element in the output of softmax function lies in the range $[0, 1]$ and all the elements will sum up to 1, thus they can be interpreted as probabilities. The standard softmax function is defined as:

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_j^k \exp(x_j)}, i \leq k, \tag{2.17}$$

where x_i is the i-th element in the input k-dimensional vectors.

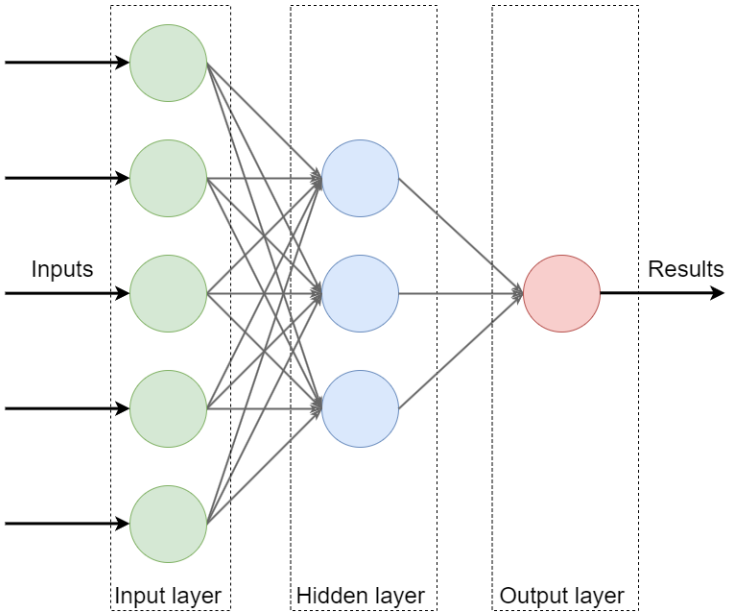


Figure 2.12: An overview of a simple MLP decoder.

The output of softmax function can be interpreted as the probabilities of the predicted classes, with the index of the largest element serving as the predicted label. The softmax function is applicable to both binary classification tasks (such as link prediction) and multi-class classification tasks (including node classification and graph classification). It is widely implemented in various models, including GCN [Kipf and Welling, 2016a], GAT [Veličković et al., 2017], GraphSage [Hamilton et al., 2017b], and more. In Chapter 5, the proposed model is employed with an softmax decoder for the node classification task.

Multilayer perceptron decoder

A multilayer perceptron (MLP) is a fully connected feed-forward artificial neural network. Fig. 2.12 illustrates a simple MLP decoder. It consists of at least three layers: an input layer, an output layer, and at least one hidden layer. An activation function is usually associated with each layer in MLP, helping transform the inputs into the desired format based on specific tasks. Commonly used activation functions include the ReLU function, the Sigmoid function, the Tanh function, and more:

$$ReLU(x) = \max(0, x) \tag{2.18}$$

$$Sigmoid(x) = \frac{1}{1 + \exp(-x)} \tag{2.19}$$

$$Tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}. \tag{2.20}$$

In GNN, the acquired node embeddings are fed into an MLP decoder with a

one-dimensional output layer. The output of the MLP decoder is considered as the predicted labels or values. The MLP decoder is flexible for different kinds of tasks just by changing the activation function. It works in binary classification tasks with the Sigmoid activation function or regression tasks with the ReLU activation function. The MLP decoder is also widely implemented in prior works such as TGAT [Xu et al., 2020], TGN [Rossi et al., 2020], APAN [Wang et al., 2021], and more. In Chapter 4 and Chapter 5, the proposed GNN frameworks employ MLP decoders for node label classification tasks and regression tasks.

Cross-entropy loss function

Cross-entropy is commonly employed to define a loss function in machine learning and optimization, serving to quantify the dissimilarity between two probability distributions. This metric assesses the performance of a classification model generating probability values within the range of 0 to 1. The cross-entropy loss escalates as the predicted probability deviates from the actual label. It is typically utilized in conjunction with the softmax function decoder, considering that the output of the softmax function is inherently interpreted as probabilities. In binary classification, cross-entropy is referred to as Binary Cross Entropy (BCE), and it can be calculated as follows:

$$BCELoss(p, y) = -(y \log(p) + (1 - y) log(1 - p)), \tag{2.21}$$

where y denotes the true label and p denotes the predicted probability.

In multi-class classification tasks with M classes, the cross-entropy is calculated as follows:

$$CELoss(p, y) = -\sum_{i=1}^{M} y_i \log(p_i), \tag{2.22}$$

where $y_i = 1$ if the true label is i and $y_i = 0$ otherwise, p_i denotes the predicted probability of the label i.

Cross-entropy loss is a potent loss function commonly used in training models for classification tasks. In Chapter 5, the proposed model is trained using Binary Cross Entropy (BCE) loss.

Mean square error loss function

The Mean Squared Error (MSE), also known as L2 loss, assesses the proximity of a learned regression line to a set of actual samples. MSE loss is commonly applied in regression tasks. A larger MSE indicates that the predicted values (predicted labels) are widely dispersed around their actual values (true labels), while a smaller MSE suggests the opposite. The MSE loss is calculated as:

$$MSELoss(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^2, \tag{2.23}$$

where $\hat{\mathbf{Y}}$ denotes the set of predicted values, and $\mathbf{Y}$ denotes the set of actual values. In Chapter 4, the proposed model is trained with MSE losses for the regression tasks.

Chapter 3

Predictive Analytics on Static Bipartite Graph

3.1 Introduction

Our research explores learning node representation embeddings, commencing with a fundamental yet impactful case—a static bipartite graph. In this endeavor, we introduce the Bipartite Graph Convolutional Network (BiGCN) model, strategically designed to leverage the multi-modal features inherent in the data to learn comprehensive node embeddings. A noteworthy observation emerges from our investigations, revealing the inadequacies of traditional GCNs when applied to bipartite graphs. Specifically, traditional GCN confounds information from the two disjoint node sets, impeding its effectiveness. In response, our proposed BiGCN model innovatively addresses this challenge by segregating the convolution operation for the two distinct node sets, thereby surmounting the limitations of conventional GCN.

The empirical validation of our approach showcases compelling results, with accuracy ranging from $77.3\% - 87.7\%$ in the label prediction task. This performance surpasses baseline traditional machine learning algorithms and statistical models and outshines the latest graph-based methods. To further attest to the real-world applicability of our model, we conduct tests in the context of a pertinent case study: predicting emergency medical service demand in the Tokyo metropolitan area. The success of our BiGCN model in this practical scenario underscores its efficacy in addressing complex, real-world challenges. It signifies a promising step forward in the realm of graph-based learning.

This work has been published in *IEEE Access* Journal [Jin et al., 2021].

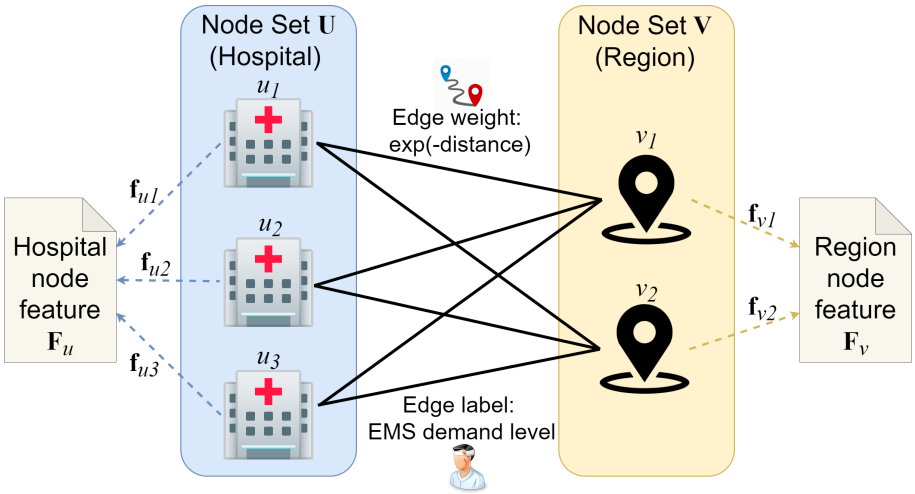


Figure 3.1: The hospital-region EMS bipartite graph.

Application: Predicting Emergency Medical Service Demand in Tokyo Metropolitan

Emergency Medical Services (EMS) are responsible for providing out-of-hospital medical care to illness and injury patients and transporting them to a medical facility. However, the EMS service is not evenly distributed. For example, the Tokyo Metropolis consists of 23 wards, each with different structures, functions, and population composition. Some regions have high emergency demand but relatively few EMS resources. The unbalanced EMS resource may raise the risk of a shortage of available EMS resources and result in the delay of first aid treatment, especially when some high-priority medical emergencies such as sudden cardiac arrest or a big event as the Tokyo Olympic Games takes place. Besides, EMS is time-sensitive, i.e., the later the service arrives at the incident sites, the more severe the damage to the patient's health will be. Thus, accurate prediction of EMS demand can help provide quick and efficient medical treatment and increase the survival rate for elderly patients [Aringhieri et al., 2016].

Moreover, Japan is experiencing a "super-aging" society. People aged 65 and older who currently make up a quarter of the total population are estimated to reach a third by 2050. The seniors tend to have a high EMS demand for their declining immune function and physical function. Thus, Japan urgently needs to discover the hidden relationship of EMS supply-demand, predict the incoming EMS demand, and take precautions against unexpected emergencies. Therefore, we are motivated to focus on Tokyo, the largest metropolis in "super-aging" society Japan, to study the relationship between EMS supply and demand. Predicting the incoming EMS demand in advance would induce a better allocation of resources and be of considerable significance to public health emergency management.

Previous research on EMS demand prediction can be classified into model-based and data-driven approaches. The model-based approaches predict specific demand

by applying the total capacity with multiple rules such as supply-demand ratio and distance decay [Anderson, 2011]. Model-based approaches work well in an actual situation and easy to understand. However, these approaches rely on predefined rules and cannot clearly explain the anomalies [Hou and Wang, 2013]. On the other hand, the data-driven approaches take the demand prediction problem as a classification or regression problem based on the observational data and employ statistical methods or naive machine learning algorithms for the solution. Data-driven approaches can fit complicated situations where the mathematical model is difficult to establish. Specifically, some statistical methods utilize social demographic data, socioeconomic factors, and land use factors to improve EMS performance in a particular city [Amorim et al., 2017; Aringhieri et al., 2016; Grekousis and Liu, 2019]. Moreover, predict EMS demand over time by a spatio-temporal statistical model raises recent attention. The spatio-temporal statistical model utilizes location information and time-series information to predict time and location accurately [Channouf et al., 2007; Setzler et al., 2009; Zhou and Matteson, 2015]. However, in complex real-world situations, the statistical model may be complicated with too high order or too much non-linearity, or even unavailable, where huge factors need to be considered. It should be noted that previous researches overwhelmingly focus on EMS demand prediction in specific regions, while few works study the demand between regions and hospitals. Hospitals play a vital role in the medical emergency. Accurate EMS demand prediction between regions and hospitals is meaningful to urban public health emergency management and better EMS resource allocation.

With the development of artificial intelligence, graph embedding methods have been proved a better performance than the traditional methods with flat inputs [Fadaee and Haeri, 2019; Taskar et al., 2004], which indicates a new way of predicting EMS demand via the demand-supply relation graph. And this topic in bipartite graphs is seldom studied in substantial research on learning graph data [Cai et al., 2018; Cui et al., 2018; Wu et al., 2019b; Zhang et al., 2019b, 2018b; Zhou et al., 2018]. Those graph embedding methods learn graph nodes as low-dimensional vector representations. With the help of node representation vectors, subsequent graph problems like node classification, link prediction, and node clustering can be easily solved by combining them with some existing machine learning methods. Start with DeepWalk [Perozzi et al., 2014a] and Node2vec [Grover and Leskovec, 2016], many graph embedding architectures are developed. Specifically, graph convolutional networks (GCN) has been proved an efficient neural network architecture regarding graph embedding researches. Many methods re-define the convolution operation for graph structure data and are developed as members of GCN based models [Kipf and Welling, 2017; Monti et al., 2017; van den Berg et al., 2017]. The core concept of GCN is iteratively aggregating feature information from a node's neighborhood nodes in a graph. Compared with other graph embedding methods that focus on

graph contents, GCN utilizes both node features and holds the potential of exploiting the graph topology structure. Intuitively, a hypothesis is suggested that the thinking of GCN may also perform well in modeling the EMS demand-supply relation graph and give an accurate prediction regarding the EMS demand between hospitals and regions.

With the concerns mentioned above, in this work, we study the EMS demand-supply relation between regions and hospitals, then propose an approach for predicting the EMS demand at the hospital-region level. Our motivation is based on the idea that the EMS demand between a region and a hospital is mainly affected by population demographics, regional socioeconomic factors, and hospital conditions. Thus, we collect data from various sources, including regional demographic data (daytime population number / census population number / crime number), regional land-use data (industrial / residential / commercial area ratio), regional historical emergency data (illness / injury case number), hospital information data (number of beds and doctors / past number of patients), and transportation data (distance between region and hospital).

We develop a Bipartite Graph Convolutional Network (BiGCN) model that In exploits the multi-modal features of the data for EMS demand prediction. Specifically, we transform the demand prediction problem to an edge label classification problem in a hospital-region bipartite graph, as shown in Fig. 3.1. The bipartite graph is a particular type of graph whose nodes divide into two disjoint sets such that the edge connects nodes from one set to the other. Hospitals and regions serve as two individual node sets. Hospitals and regions serve as two individual node sets. The edge connects a hospital node and a region node, indicating that an emergency happened in this region, and the injured people are sent to this hospital. Feature attributes are attached to each node and each edge.

Notably, we find that traditional GCN does not work correctly in bipartite graphs because it confuses the information from the two disjoint node sets. Our BiGCN model separates the convolution operation of the two node sets and overcomes the shortcomings of traditional GCN. The experimental results demonstrate that our approach achieves $77.3\% - 87.7\%$ accuracy in the label prediction task, which is significantly superior to baseline traditional machine learning algorithms, statistical models, and the latest graph-based methods. Finally, We use a case study to illustrate that our approach can provide valuable suggestions for allocating injured people in emergencies. It proves that our work is meaningful to urban public health emergency management, make the public aware of the significance of EMS demand prediction, and help local governments better allocate EMS resource and decrease the emergency risk.

The main contributions of the paper are summarized as follows:

- We are the first to analyze EMS demand at the hospital-region level in a

metropolis like Tokyo. Our work is of considerable significance to public health emergency management.

- We represent the ambulance record data as a hospital-region bipartite graph and transform the EMS demand prediction problem to an edge label classification problem in the bipartite graph.
- We analyze the limitations of GCN in bipartite graphs and propose BiGCN by fully considering the structure characteristics of bipartite graphs. BiGCN is not limited to the hospital-region bipartite graph in this paper but has the potential to become a general model for learning node embeddings and accomplishing supervised learning tasks in non-specific bipartite graphs.
- We conduct experiments and demonstrate that our approach achieves excellent performance in the demand label prediction task and significantly outperforms baseline methods, including traditional machine learning algorithms, statistical models, and state-of-the-art graph-based methods. We discover the main factors that affect the EMS demand prediction most. We use a case study to show how our approach can contribute to public health emergency management.

3.2 Related Work

EMS Demand Analysis

Past research on EMS supply and demand analysis can be classified into model-based and data-driven approaches. The model-based approaches predict specific demand by applying the total capacity with multiple rules such as supply-demand ratio and distance decay [Anderson, 2011]. The main topic is accessibility as an important indicator in evaluating the justice of medical service [Luo, 2014]. Moreover, the large variance of accessibility indicates an unequal spatial distribution of EMS facilities. Therefore, EMS facility location optimization (FLO) is another popular topic. There are several types of EMS facilities for FLO models, including emergency devices [Bonnet et al., 2015], emergency centers [Silva and Serra, 2008], and the ambulance stations [Zhang and Jiang, 2014].

The data-driven approaches take the demand prediction problem as a classification or regression problem based on the observational data and employ a statistical or naive machine learning model for the solution. Specifically, the statistics-based approaches perform well in spatial EMS analysis [Aringhieri et al., 2017]. uEMS has the object to maximize the EMS vehicle coverage with limited ambulance stations in an urban area and provides a generalized linear model to locate the urban EMS ambulance station [Amorim et al., 2017]. Steins et al. use a Zero-Inflated Poisson (ZIP)

regression approach to develop a statistical EMS demand forecasting model [Steins et al., 2019]. Grekousis et al. propose a spatial-based Artificial Neural Networks (ANN) approach that identifies the geographical location of expected emergency events [Grekousis and Liu, 2019]. Besides, predicting EMS demand over time has raised recent attentions [Chen et al., 2015; Zaric, 2013; Zhou and Matteson, 2015]. Some spatio-temporal approaches are proposed to utilize both location information and time-series information since EMS prediction needs to be accurate for both time and location. Setzler et al. design an ANN to forecast EMS demand volume of specific areas during different times of the day [Setzler et al., 2009]. Channouf et al. develop and compare several regression models to analyze the time-series information of a major Canadian city's daily and hourly EMS call volume [Channouf et al., 2007]. Zhou et al. introduce a Gaussian Mixture Model (GMM) to estimate the ambulance demand distribution in Toronto, Canada [Zhou et al., 2015]. However, previous research overwhelmingly focuses on EMS demand prediction in specific regions, while few studies study the demand between regions and hospitals. Hospitals play a vital role in the medical emergency. Accurate EMS demand prediction between regions and hospitals is meaningful to urban public health emergency management and better EMS resource allocation.

Graph Embedding

Graphs are used in many science branches to represent the patterns of connections between the components of complex systems. There is a surge of interest in graph embedding [Cai et al., 2018; Cui et al., 2018; Wu et al., 2019b; Zhang et al., 2019b, 2018b; Zhou et al., 2018]. The goal is to learn a mapping that embeds nodes as points in a low-dimensional vector space. The learned embeddings can be taken as feature inputs for downstream machine learning tasks, and this technology has achieved great success in many applications. Researchers have proposed various graph embedding methods such as matrix factorization [Ou et al., 2016], edge reconstruction [Tang et al., 2015], random walks plus skip-gram model [Grover and Leskovec, 2016; Perozzi et al., 2014a], and graph neural networks [Hamilton et al., 2017a; Kipf and Welling, 2017; Veličković et al., 2018].

However, graph embedding in bipartite graphs is relatively less studied. The bipartite graph is a ubiquitous data structure to model the relationship between two types of entities. Examples of bipartite graphs include the author-paper graph, the customer-product graph, the player-event graph, the actor-movie graph, and the keyword-document graph. As far as we know, there are few works on bipartite graph embedding. BiNE is an approach by performing biased random walks and preserving the long-tail distribution of nodes in bipartite graphs [Gao et al., 2018]. BGNN is another approach that utilizes inter-domain message passing and intra-domain alignment towards information fusion [He et al., 2019]. However, these two methods

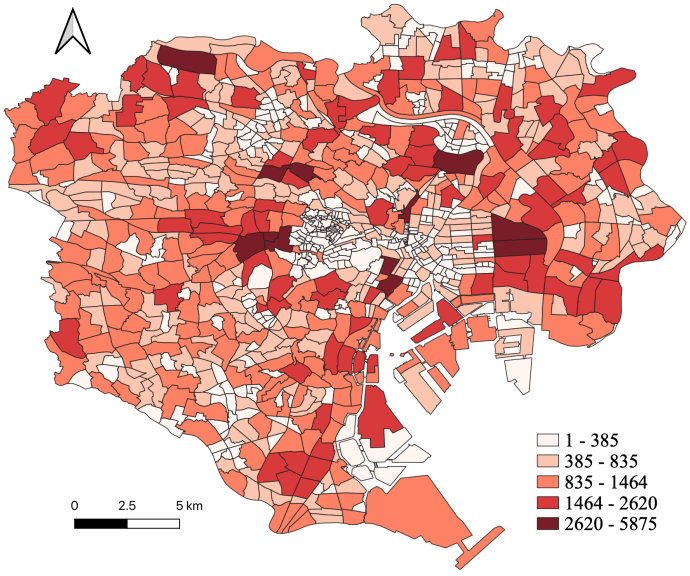


Figure 3.2: The number of emergency cases in different regions of Tokyo.

do not precisely solve our problem for two reasons. First, BiNE only employs the graph structure information but not the node feature information. Second, both are unsupervised learning methods that do not fully utilize the training set to optimize the models, resulting in suboptimal performance.

On the other hand, there are extensive researches on graph embedding for heterogeneous graphs, which contain multi-typed nodes or multi-typed edges. Note that bipartite graphs that contain two typed nodes can be regarded as a particular class of heterogeneous graphs. Metapath2vec uses meta-path based random walks to aggregate the heterogeneous neighborhood nodes and then exploit a skip-gram model to learn the node embeddings [Dong et al., 2017]. HAN is a heterogeneous graph neural network based on the hierarchical attention mechanism, including node-level and semantic-level attentions [Wang et al., 2019a]. It fully considers the importance of node neighbors and different meta-paths. HetGNN combines heterogeneous structural information and node attributes. It has an excellent performance in graphs with multiple types of nodes and edges [Zhang et al., 2019a].

3.3 Problem Setup

This section first describes the dataset we used in our research. It then represents the EMS data as a hospital-region bipartite graph and transforms the problem to an edge label classification problem in the graph.

Table 3.1: Features of hospitals and regions.

Node Type	Feature Category	Feature Name
Region	Population	Day time population Census population Crime number
	Land-Use Zoning	Industrial area ratio Residential area ratio Commercial area ratio
	Historical Emergency Number (patients sent to hospitals)	Total case number Injury case number Disease case number
	Historical Emergency Number (patients not sent to hospitals)	Total case number Injury case number Disease case number
Hospital	Capacity	Number of beds Number of doctors
	Historical Emergency Number	Total case number Injury case number Disease case number

3.3.1 Dataset Description

This study collects a dataset of ambulance records from 1st January 2017 to 31st December 2017 for the Central Tokyo area. The dataset is provided by the Tokyo Fire Department. It contains 624,062 emergency cases. Each record includes the patient's age, gender, ambulance types (disease or injury), ambulance scene, and hospital address.

There are 931 administrative regions and 291 hospitals that can provide emergency medical care in Tokyo. Administrative regions are divided by following the official district division principle of government. The map of regions with the number of cases is visualized in Fig. 3.2. We are interested in the emergency demand from one region to one hospital. The raw data aggregates into 270,921 hospital-region pairs, with only 23,308 pairs having demand. We attach a binary label denoting the low/high demand level for each pair. The label is determined by the percentage of the demand in the capacity of the respective hospital. As a result, 13,603 pairs are labeled "high", and 9,705 pairs are labeled "low". Moreover, we collect a bunch of feature information for the regions and hospitals, as listed in Table 3.1. Also, we follow [Anderson, 2011] to measure the Euclidean distance between regions and hospitals.

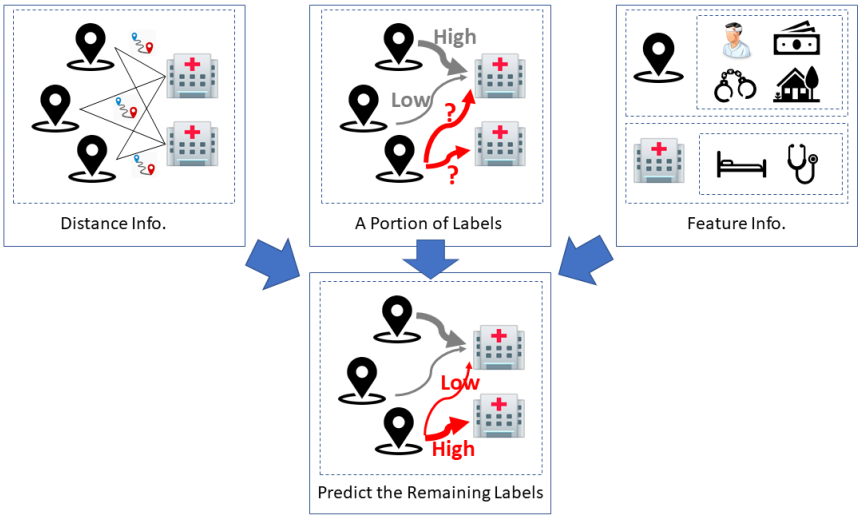


Figure 3.3: Predicting the high/low demand label of hospital-region pairs.

3.3.2 Edge Label Classification in the Hospital-Region Graph

We represent the EMS data as a bipartite graph $\mathbf{G} = (\mathbf{U}, \mathbf{V}, \mathbf{E})$, as shown in Fig. 3.1. $\mathbf{U}$ and $\mathbf{V}$ are the node sets for hospitals and regions, respectively. $M = |\mathbf{U}|$ and $N = |\mathbf{V}|$ are the number of nodes in the two sets. $\mathbf{E} \subseteq \mathbf{U} \times \mathbf{V}$ denotes the edge set. $K = |\mathbf{E}|$ is the number of edges. $e_{ij} \in \mathbf{E}$ represents the EMS demand relationship between the hospital node $u_i \in \mathbf{U}$ and region node $v_j \in \mathbf{V}$. The edge weight is

$$\omega_{ij} = \exp(-\text{distance}(u_i, v_j)), \tag{3.1}$$

which indicates the "closeness" of u_i and v_j. $distance\,(\cdot)$ denotes the Euclidean distance between two node, and $\exp(\cdot)$ denotes the exponential function.

The hospital node u_i and region node v_j are associated with feature vectors $\mathbf{f}_{u_i} \in \mathcal{R}^P$ and $\mathbf{f}_{v_j} \in \mathcal{R}^Q$, which are based on preprocessed raw features in Table 3.1, and P and Q are the respective number of features. $\mathbf{F}_u = [\mathbf{f}_{u_1}, \ldots, \mathbf{f}_{u_M}]^\top \in \mathcal{R}^{M \times P}$ and $\mathbf{F}_v = [\mathbf{f}_{v_1}, \ldots, \mathbf{f}_{v_N}]^\top \in \mathcal{R}^{N \times Q}$ denote the feature matrix of hospitals and regions. Moreover, $y_{ij} \in \{-1, 1\}$ is the label for edge e_{ij} (which corresponds to low/high demand between hospital and region) and $\mathbf{Y} = \{y_{ij} | e_{ij} \in \mathbf{E}\}$ is the set of edge labels.

Since the edge connects nodes between $\mathbf{U}$ and $\mathbf{V}$, the adjacency matrix $\mathbf{A}$ takes a block off-diagonal form

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{M \times M} & \mathbf{B}_u \\ \mathbf{B}_v & \mathbf{0}_{N \times N} \end{bmatrix}, \tag{3.2}$$

where $\mathbf{B}_u \in \mathcal{R}^{M \times N}$ and $\mathbf{B}_v \in \mathcal{R}^{N \times M}$ are the incidence matrix for $\mathbf{U}$ and $\mathbf{V}$, respec-

Table 3.2: Statistics of the bipartite graph for the EMS dataset.

Item	Statistics
Number of Hospital Nodes M	291
Number of Region Nodes N	931
Number of Edges K	23,308
Number of Hospital Features P	5
Number of Region Features Q	12
Average/Maximal Degree of Hospital Nodes	80.1/616
Average/Maximal Degree of Region Nodes	25.0/92

tively. $\mathbf{B}_u = \mathbf{B}_v^\top$, and $\mathbf{B}_{u(i,j)} = \mathbf{B}_{v(j,i)} = \omega_{ij}$. $\mathbf{D}$ is the degree diagonal matrix:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_u & \mathbf{0}_{M\times N} \\ \mathbf{0}_{N\times M} & \mathbf{D}_v, \end{bmatrix}, \tag{3.3}$$

where $\mathbf{D}_u = \mathrm{Diag}(\sum_i \mathbf{B}_{u(1,i)}, \ldots, \sum_i \mathbf{B}_{u(M,i)})$ and $\mathbf{D}_v = \mathrm{Diag}(\sum_i \mathbf{B}_{v(1,i)}, \ldots, \sum_i \mathbf{B}_{v(N,i)})$ are the diagonal degree matrices of $\mathbf{U}$ and $\mathbf{V}$.

Table 3.2 reports the statistics of the hospital-region bipartite graph. The problem in Section 4.3 can be represented as an edge label classification problem in the graph. Suppose $\mathbf{Y}$ is split into a training set $\mathbf{Y}^{\text{train}}$, a validation set $\mathbf{Y}^{\text{val}}$, and a test set $\mathbf{Y}^{\text{test}}$. Then, our problem (as illustrated in Fig. 3.3) is set up as follows: given the features of regions and hospitals ($\mathbf{F}_u$, $\mathbf{F}_v$), the distance of hospital-region pairs ($\mathbf{A}$), and a portion ($10\% - 80\%$) of known labels ($\mathbf{Y}^{\text{train}}$, and $\mathbf{Y}^{\text{val}}$), how can we predict the remaining edge labels ($\mathbf{Y}^{\text{test}}$).

3.4 Limitations of Vanilla GCN in Bipartite Graph

In this section, we discuss the limitations of traditional GCN and explain why it cannot correctly work in the bipartite graph.

GCN [Kipf and Welling, 2017] is a type of neural network architectures that can leverage the graph structure and node features for graph analysis, and this model has achieved great success in many tasks. GCN consists of aggregators and updaters. The aggregator gathers information guided by the graph structure, and the updater updates nodes' hidden states according to the gathered information. The core of the method is learning node representations, or *embeddings*, in a low-dimensional vector space that encode information about the graph. Specifically, the convolutional layer is based on the following equation:

$$\mathbf{H}^{(t+1)} = \sigma(\mathbf{L}\mathbf{H}^{(t)}\mathbf{W}^{(t+1)}), \tag{3.4}$$

where $\mathbf{L}$ is the aggregation matrix, $\mathbf{H}^{(t)}$ is the node embedding matrix in t-th layer, $\mathbf{H}^{(0)}$ is initialized with the node feature matrix, $\mathbf{W}^{(l)}$ is the trainable weight matrix in l-th layer, and $\sigma(\cdot)$ is the activation function.

GCN is originally designed for plain graphs, and it adopts the symmetric normalized Laplacian matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ as the aggregator $\mathbf{L}^{\mathrm{GCN}}$, where $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree diagonal matrix of $\tilde{\mathbf{A}}$. Here $\mathbf{I}$ denotes the identity matrix. In the following, we analyze the limitations of applying GCN in bipartite graphs. To simplify the analysis, we consider an approximate form $\mathbf{L}^{\mathrm{GCN}} \approx \mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$. Moreover, we have to concatenate the two feature matrices $\mathbf{F}_u$ and $\mathbf{F}_v$ as the initialization of the node embedding matrix $\mathbf{H}^{(0)}$. However, since the two matrices have different column dimensions and $P < Q$, we have to expand $\mathbf{F}_u$ with zero values to make an alignment:

$$\mathbf{H}^{(0)} = \left[\begin{array}{c|c} \mathbf{F}_u & \mathbf{0}_{M\times(Q-P)} \\ & \mathbf{F}_v \end{array}\right], \tag{3.5}$$

where $|$ is an auxiliary symbol for matrix partitions, and $\mathbf{0}$ is a zero matrix with the given shape. Then, the first convolutional layer in Eq. (3.4) becomes

$$\mathbf{H}^{(1)} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^{(0)}\mathbf{W}^{(1)} \tag{3.6}$$

$$= \left[\begin{array}{c} \left(\mathbf{D}_u^{-\frac{1}{2}}\mathbf{B}_u\mathbf{D}_v^{-\frac{1}{2}}\mathbf{F}_v\right) + \left(\mathbf{D}_u^{-1}\mathbf{F}_u|\mathbf{0}_{M\times(Q-P)}\right) \\ \left(\mathbf{D}_v^{-\frac{1}{2}}\mathbf{B}_v\mathbf{D}_u^{-\frac{1}{2}}\mathbf{F}_u|\mathbf{0}_{N\times(Q-P)}\right) + \left(\mathbf{D}_v^{-1}\mathbf{F}_v\right) \end{array}\right]\mathbf{W}^{(1)}. \tag{3.7}$$

From Eq. (3.7), we can find that the aggregator sums the hospital feature matrix $\mathbf{F}_u$ and the region feature matrix $\mathbf{F}_v$. Note that $\mathbf{F}_u$ and $\mathbf{F}_v$ refer to information from different sources, which implies that they have different dimensions and they are based on different scaling systems. Therefore, it is not reasonable to sum them together. For the same reason, it is insufficient to use the same matrix $\mathbf{W}^{(1)}$ for the filter operation. As a result, GCN does not work correctly in bipartite graphs. A new approach is in demand to wisely apply the thinking of GCN to bipartite graphs.

3.5 Proposed Approach

In this section, we propose an approach to address the problem in the last section. We hypothesize that the EMS demand between a region and a hospital can be predicted by modeling the demographic information and the distance of regions and hospitals. To overcome the disadvantage of traditional GCN and apply the thinking of GCN in bipartite graphs, we introduce an improved GCN based model, bipartite graph convolutional network structure BiGCN. Fig. 5.5 illustrates the overview structure of BiGCN model. The core method of BiGCN is learning the embedding representation of nodes by graph convolution operation, then learning the edge em-

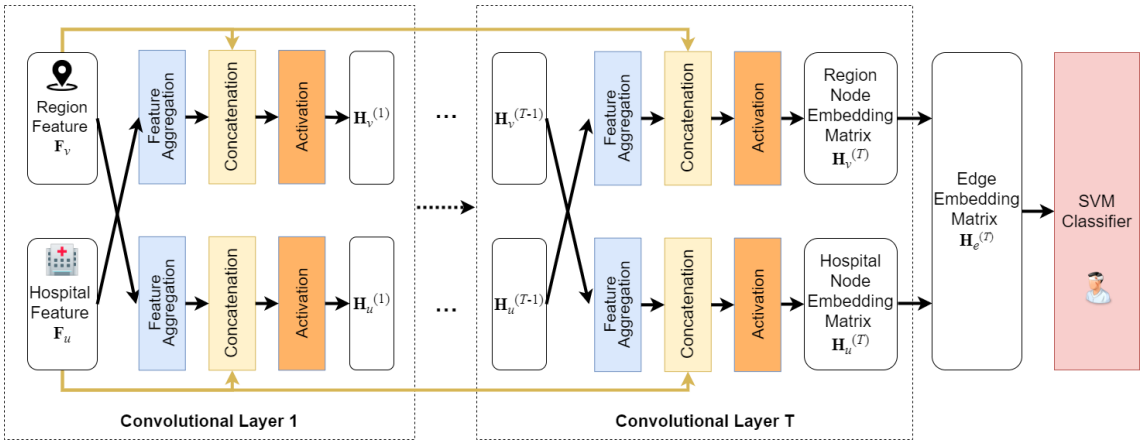


Figure 3.4: Overview of the structure of our approach.

bedding representations based on its two side node embeddings, finally predicting the edge labels based on the edge embeddings. The spotlight of BiGCN is the utilization of both the graph structure information and node features, and the individual operation for two disjoint node sets. The model evaluation depends on the accuracy and f1 score metrics regarding the EMS demand prediction task.

3.5.1 BiGCN

Based on the structural characteristics of bipartite graphs that $\mathbf{U}$ and $\mathbf{V}$ are disjoint node sets with distinct properties (degree distributions, associated features), our idea is to separate the convolution operation for $\mathbf{U}$ and $\mathbf{V}$. Specifically, the mathematical expression of our convolutional layer is

$$\mathbf{H}_u^{(0)} = \mathbf{F}_u \tag{3.8}$$

$$\mathbf{H}_v^{(0)} = \mathbf{F}_v \tag{3.9}$$

$$\mathbf{H}_u^{(t+1)} = \sigma\big(\big[\mathbf{D}_u^{-1}\mathbf{B}_u\mathbf{H}_v^{(t)}\mathbf{W}_u^{(t+1)} \parallel \mathbf{F}_u\boldsymbol{\omega}_u^{(t+1)}\big]\big) \tag{3.10}$$

$$\mathbf{H}_v^{(t+1)} = \sigma\big(\big[\mathbf{D}_v^{-1}\mathbf{B}_v\mathbf{H}_u^{(t)}\mathbf{W}_v^{(t+1)} \parallel \mathbf{F}_v\boldsymbol{\omega}_v^{(t+1)}\big]\big). \tag{3.11}$$

$\mathbf{H}_u^{(t)}$ and $\mathbf{H}_v^{(t)}$ are the learned node embedding matrix in the t-th layer for $\mathbf{U}$ and $\mathbf{V}$, respectively. $\mathbf{W}_u^{(t)}$, $\mathbf{W}_v^{(t)}$, $\boldsymbol{\omega}_u^{(t)}$, and $\boldsymbol{\omega}_v^{(t)}$ are trainable weight matrices (filters) in t-th layer. $\parallel$ is the concatenation operation.

The process of the convolutional layer contains three steps, as shown in Fig. 5.5. Take node $u_i \in \mathbf{U}$ as an example. We first aggregate features from its neighbors in $\mathbf{V}$ and update the aggregated features by a filter (this step corresponds to $\mathbf{D}_u^{-1}\mathbf{B}_u\mathbf{H}_v^{(t)}\mathbf{W}_u^{(t+1)}$ in Eq.(3.10)). In parallel, we update the original features of u_i by another filter (this step corresponds to $\mathbf{F}_u\boldsymbol{\omega}_u^{(t+1)}$ in Eq.(3.10)). Finally, we concatenate the updated features from two sources and pass them to an activation function.

Similar process applies for node $v_i \in \mathbf{V}$.

There are three differences between GCN and BiGCN. First, GCN only use one aggregator and one filter for $\mathbf{U}$ and $\mathbf{V}$, while BiGCN enforces independent aggregators ($\mathbf{D}_u^{-1}\mathbf{B}_u$, $\mathbf{D}_v^{-1}\mathbf{B}_v$) and filters ($\mathbf{W}_u^{(t)}$, $\mathbf{W}_v^{(t)}$, $\boldsymbol{\omega}_u^{(t)}$ for the two node sets. Secondly, GCN intentionally adding self-connections ($\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$) to preserve the feature of the node itself. On the other hand, BiGCN uses the concatenation operation for this purpose and overcomes the shortcomings of simply summing the features from different sources. Thirdly, GCN uses the symmetric normalized Laplacian matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ as the aggregator. However, the node degree distributions of $\mathbf{U}$ and $\mathbf{V}$ are compositionally distinct from each other, and thus the symmetric normalization is not meaningful. Instead, BiGCN uses the random walk Laplacian matrix $\mathbf{D}_u^{-1}\mathbf{B}_u$ and $\mathbf{D}_v^{-1}\mathbf{B}_v$ to take the average of neighboring node features in the aggregation process. All of the above imply that BiGCN fully considers the structure characteristics of bipartite graphs.

We explain about the dimensionality. Note that in the hidden layer, we concatenate the features from $\mathbf{U}$ and $\mathbf{V}$. We hope to keep the dimension in proportion to the original dimension of the two sources. Therefore, we assume $\lceil c^{(t)}P \rceil + \lceil c^{(t)}Q \rceil$ is the dimension of t-th hidden units. $c^{(t)} \in \mathcal{R}^+$ is a dimension scaling parameter. $\lceil \cdot \rceil$ is the ceiling function. Following this assumption, we have:

$$\mathbf{H}_u^{(t)} \in \mathcal{R}^{M \times (\lceil c^{(t)}P \rceil + \lceil c^{(t)}Q \rceil)} \tag{3.12}$$

$$\mathbf{H}_v^{(t)} \in \mathcal{R}^{N \times (\lceil c^{(t)}P \rceil + \lceil c^{(t)}Q \rceil)} \tag{3.13}$$

$$\mathbf{W}_u^{(1)} \in \mathcal{R}^{Q \times \lceil c^{(1)}Q \rceil} \tag{3.14}$$

$$\mathbf{W}_v^{(1)} \in \mathcal{R}^{P \times \lceil c^{(1)}P \rceil} \tag{3.15}$$

$$\mathbf{W}_u^{(t)} \in \mathcal{R}^{(\lceil c^{(t-1)}P \rceil + \lceil c^{(t-1)}Q \rceil) \times \lceil c^{(t)}Q \rceil}, \text{for } t \geq 2 \tag{3.16}$$

$$\mathbf{W}_v^{(t)} \in \mathcal{R}^{(\lceil c^{(t-1)}P \rceil + \lceil c^{(t-1)}Q \rceil) \times \lceil c^{(t)}P \rceil}, \text{for } t \geq 2 \tag{3.17}$$

$$\boldsymbol{\omega}_u^{(t)} \in \mathcal{R}^{P \times \lceil c^{(t)}P \rceil} \tag{3.18}$$

$$\boldsymbol{\omega}_v^{(t)} \in \mathcal{R}^{Q \times \lceil c^{(t)}Q \rceil}. \tag{3.19}$$

Next, we analyze the computational complexity of the convolutional layer for GCN and BiGCN. With regard to Eqs.(3.7), (3.10), (3.11), the complexity is dominated by matrix multiplication. We equally assume that both models have the same dimension D for the hidden units. Moreover, we suppose $P < Q$. In the case of a sparse bipartite graph, the complexity of GCN is $\mathcal{O}(KQ + KP + (M+N)QD)$ $= \mathcal{O}(KQ + (M+N)QD)$, while the complexity of BiGCN is $\mathcal{O}(KQ + MQD +$ $MPD + KP + NPD + NQD) = \mathcal{O}(KQ + (M+N)QD)$. In the case of a dense bipartite graph, the complexity of GCN is $\mathcal{O}(MNQ + NMP + (M+N)QD) =$ $\mathcal{O}(MNQ + (M+N)QD)$, while the complexity of BiGCN is $\mathcal{O}(MNQ + MQD +$

$MPD + NMP + NPD + NQD) = \mathcal{O}(MNQ + (M+N)QD)$. Therefore, GCN and BiGCN have the same computational complexity.

3.5.2 Loss Function

We now turn to edge label classification. We have obtained node embedding matrix $\mathbf{H}_u^{(T)} = [\mathbf{h}_{u_1}^{(T)}, \ldots, \mathbf{h}_{u_M}^{(T)}]^\top$ and $\mathbf{H}_v^{(T)} = [\mathbf{h}_{v_1}^{(T)}, \ldots, \mathbf{h}_{v_N}^{(T)}]^\top$, where T is the number of layers in BiGCN, and $\mathbf{h}_{u_i}^{(T)}$ and $\mathbf{h}_{v_j}^{(T)}$ are the embedding for u_i and v_j. For an edge e_{ij}, we can generate its embedding $\mathbf{h}_{e(ij)}^{(T)} \in \mathcal{R}^{(\lceil c^{(T)}P \rceil + \lceil c^{(T)}Q \rceil)}$ using the Hadamard product [Grover and Leskovec, 2016]

$$[\mathbf{h}_{e(ij)}^{(T)}]_l = [\mathbf{h}_{u_i}^{(T)}]_l \cdot [\mathbf{h}_{v_j}^{(T)}]_l, \tag{3.20}$$

where $l \in 1, \cdots, (\lceil c^{(T)}P \rceil + \lceil c^{(T)}Q \rceil)$ denotes the subscript of the l-th element. In this way, we can generate an edge embedding matrix $\mathbf{H}_e^{(T)} = [\mathbf{h}_{e_1}^{(T)}, \ldots, \mathbf{h}_{e_K}^{(T)}]^\top \in \mathcal{R}^{K \times (\lceil c^{(T)}P \rceil + \lceil c^{(T)}Q \rceil)}$.

Then, we use Support Vector Machine (SVM) classifier to predict the binary edge labels. Specifically, the predicted labels can be formulated as

$$\hat{\mathbf{y}} = \operatorname{sgn}\left(\mathbf{H}_e^{(T)}\boldsymbol{\omega}\right), \tag{3.21}$$

where $\boldsymbol{\omega} \in \mathcal{R}^{(\lceil c^{(T)}P \rceil + \lceil c^{(T)}Q \rceil)}$ is a trainable vector and sgn is the sign function.

For model training, we evaluate the classification error over all examples in the training set based on the hinge loss function:

$$\mathcal{L}_c = \mathcal{H}(\mathbf{H}_e^{(T)}) \tag{3.22}$$

$$= \frac{1}{|\mathbf{Y}^{\text{train}}|} \sum_{k \in \mathcal{I}(\mathbf{Y}^{\text{train}})} \max\left(0, 1 - \mathbf{y}_k \left(\mathbf{H}_e^{(T)}\boldsymbol{\omega}\right)_k\right) + \frac{1}{2}\|\boldsymbol{\omega}\|_2^2, \tag{3.23}$$

where $\mathbf{y}$ is the vector of edge labels and $\mathcal{I}(\mathbf{Y}^{\text{train}})$ is the set of edge indices in $\mathbf{Y}^{\text{train}}$. Moreover, to alleviate the information loss in the hidden layer of BiGCN, we formulate a recurrent loss by applying the hinge loss to the Hadamard product of hidden units

$$\mathcal{L}_r = \sum_{t=1}^{T-1} \mathcal{H}(\mathbf{H}_e^{(t)}). \tag{3.24}$$

Finally, the total loss is

$$\mathcal{L} = \mathcal{L}_c + \alpha \mathcal{L}_r, \tag{3.25}$$

where α is a hyperparameter for balancing $\mathcal{L}_c$ and $\mathcal{L}_r$.

3.6 Experiments

We conducted experiments on the edge label classification task to answer the following questions regarding BiGCN model:

- Does BiGCN agree with our hypothesis and perform well in bipartite graphs? How does it compare with other models?
- Which input feature influences the prediction result most?
- How many layers should be used in BiGCN?
- Is recurrent loss necessary? How does it impact model performance?
- How can our approach be used for public health emergency management?

In the following, we first explain experimental settings and baselines. After that, we discuss the results.

Experiment Setup

We randomly split $\mathbf{Y}$ into $\mathbf{Y}^{\text{train}}$, $\mathbf{Y}^{\text{val}}$, and $\mathbf{Y}^{\text{test}}$. $\mathbf{Y}^{\text{val}}$ accounts for 10%, $\mathbf{Y}^{\text{test}}$ ratio ranges from $10\% \sim 80\%$, and the remaining is for $\mathbf{Y}^{\text{train}}$. As for the implementation of BiGCN, we chose Parametric Rectified Linear Unit (PReLU) as the activation function. We trained the model by Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.0001. We set the maximum training iteration as 500 and applied an early stopping strategy if the validation loss does not decrease for 10 iterations. The implementation is in Python and PyTorch with Intel Xeon CPU @ 2.20GHz, NVIDIA Tesla P100 16GB GPU, and 25GB memory.

Baselines

We consider traditional machine learning classifiers, statistical models, and the latest graph-based methods such as graph embedding and graph neural networks as baselines. The details are listed below.

1. **SVM** (Support Vector Machine), **GBDT** (Gradient Boosting Decision Tree), and **LR** (Logistic Regression): These are traditional machine learning classifiers.
2. **GMM** [Zhou et al., 2015]: This is a Gaussian Mixture Model approach.
3. **ZIP** [Steins et al., 2019]: This is a Zero-Inflated Poisson regression approach.

4. **Node2Vec** [Grover and Leskovec, 2016]: This is an unsupervised algorithm for learning graph node embeddings based on random walks and skip-gram model.

5. **GCN** [Kipf and Welling, 2017]: This is one of the most widely used graph convolutional networks. This method, as stated in Section 3.4, can be interpreted as smoothing the node features in the neighborhoods guided by the graph structure.

6. **VGAE** [Kipf and Welling, 2016b]: This is a variational graph autoencoder, in which GCN is used as an encoder to learn node embeddings.

7. **GraphSAGE** [Hamilton et al., 2017a]: This is another popular method for generating node embeddings based on graph structure and node features.

8. **BiNE** [Gao et al., 2018]: This is an extension of unsupervised graph embedding algorithm to bipartite graphs.

9. **BGNN** [He et al., 2019]: This is a bipartite graph neural network for learning node embeddings in an unsupervised fashion.

10. **Metapath2vec** [Dong et al., 2017]: This is an extension of conventional graph embedding technique to heterogeneous graphs, which contain multi-typed nodes or multi-typed edges. This approach formalizes meta-path-based random walks to construct the heterogeneous neighborhood of a node to learn embeddings. Note that

11. **HetGNN** [Zhang et al., 2019a]: This is a graph neural network model for learning node vector representations in heterogeneous graphs.

12. **HAN** [Wang et al., 2019a]: This is a heterogeneous graph neural network based on the hierarchical node-level and semantic-level attentions.

We employ traditional machine learning classifiers (SVM, GDBT, LR) to predict the hospital-region bipartite graph edge labels with flat inputs consist of distance information and node features. The two baseline statistical models (GMM, ZIP) are designed for EMS demand forecasting in the prior works. The statistical model is much more understandable and simpler to implement than machine learning models. In our experiment, two models receive flat inputs consist of distance information and node features. Graph structure information is not involved. The graph-based approaches (Node2Vec, GCN, VGAE, GraphSAGE, BiNE, BGNN, Metapath2vec, HetGNN, HAN) receive both graph topology structure and node features as input. They are initially designed for only learning node embeddings. Thus, after obtaining the node embeddings by these baseline methods, we generate edge embeddings by

Table 3.3: Fine-tuned hyperparameters in the heterogeneous GNN baseline methods.

	BGNN	**Metapath2vec**	**HetGNN**	**HAN**	**BiGCN**
Learning rate	0.001	0.0005	0.005	0.001	0.001
Weight decay	0.001	0.005	0.0	0.001	0.1
Dropout	0.2	0.1	0.1	0.2	0.2
Hidden dim.	128	128	128	128	128
# Layers	2	3	3	2	3

Table 3.4: The results for predicting the hospital-region labels by different approaches.

Model	Test Ratio									
	10%		20%		40%		60%		80%	
	Acc.	F1.	Acc.	F1.	Acc.	F1.	Acc.	F1.	Acc.	F1.
SVM	0.734	0.769	0.716	0.759	0.715	0.750	0.707	0.747	0.696	0.737
GBDT	0.812	0.843	0.796	0.840	0.815	0.844	0.782	0.832	**0.778**	**0.830**
LR	0.624	0.529	0.628	0.539	0.602	0.489	0.637	0.564	0.582	0.443
GMM	0.667	0.198	0.555	0.407	0.645	0.470	0.632	0.454	0.623	0.385
ZIP	0.851	0.592	0.785	0.706	0.596	0.047	0.592	0.037	0.591	0.051
Node2Vec	0.756	0.791	0.762	0.800	0.758	0.800	0.751	0.794	0.745	0.788
GCN	0.730	0.767	0.718	0.753	0.728	0.764	0.729	0.760	0.700	0.736
VGAE	0.736	0.785	0.723	0.775	0.725	0.776	0.728	0.776	0.694	0.738
GraphSAGE	0.422	0.584	0.411	0.583	0.411	0.582	0.576	0.729	0.585	0.738
BiNE	0.687	0.743	0.686	0.738	0.683	0.737	0.679	0.744	0.678	0.738
BGNN	0.698	0.756	0.701	0.736	0.696	0.775	0.790	0.754	0.704	0.762
Metapath2vec	0.747	0.788	0.762	0.801	0.750	0.790	0.748	0.789	0.740	0.776
HetGNN	0.691	0.773	0.681	0.780	0.679	0.753	0.704	0.768	0.681	0.768
HAN	0.689	0.769	0.725	0.764	0.709	0.760	0.695	0.758	0.706	0.758
BiGCN	**0.877**	**0.895**	**0.860**	**0.881**	**0.848**	**0.871**	**0.827**	**0.852**	0.773	0.804
Performance Gain (%)	3.0 | 107.8	6.1 | 352.0	8.0 | 109.2	4.8 | 116.4	4.0 | 106.3	3.1 | 1853.2	4.6 | 43.5	2.4 | 2302.7	-0.6 | 32.8	-3.1 | 1576.5

applying Hadamard product to these node embeddings and finally feed them to an SVM classifier to predict the hospital-region labels. Notably, the hyperparameters of heterogeneous GNN baseline methods (BGNN, Metapath2vec, HetGNN, and HAN) are fine-tuned to exhibits their best performance. The values of hyperparameters are listed in Table 3.3.

3.6.1 Performance Evaluation

Table 5.8 displays the results by BiGCN (with loss balancing hyperparameter $\alpha = 0.1$, dimension of hidden unit $\lceil c^{(t)}P \rceil + \lceil c^{(t)}Q \rceil = 50$, and number of convolution layer $T = 3$) and the baselines. We evaluated the performance in terms of accuracy and F1 score. BiGCN demonstrates the best overall performance. In particular, it achieves

the highest accuracy and F1 score when the test ratio ranges from 10% $\sim$ 60%, with 2.4% - 8.0% performance gain over GBDT that is consistently better than the other baselines. The only exception arises when the test ratio reaches 80%, or the training set only accounts for 10%, where our model suffers from an overfitting problem. GBDT, as an additive model, has a good strategy to prevent overfitting and thus obtains superior performance.

It is interesting to note that BiGCN outperforms the other graph-based approaches by a large margin. We attribute this to two reasons. The first reason is the deliberate network structure of BiGCN that considers the characteristics of bipartite graphs. In contrast, GCN, VGAE, and GraphSAGE are designed for plain graphs and do not adapt to bipartite graphs. The second reason is the end-to-end model optimization based on our proposed cost function. Conversely, Node2Vec, VGAE, BiNE, and BGNN are unsupervised learning methods to preserve the original graph's topology structure. They thus do not fully utilize the training set to optimize the models. Note that Metapath2vec, HetGNN, and HAN are originally proposed for heterogeneous graphs, and we find that they also do not fit well with bipartite graphs. For example, although HAN utilizes the node-level and semantic-level attention mechanism, the trivial structure and single meta-path in bipartite graph restrict its performance. As for the statistical models, GMM gives similar performances in all test ratios, while ZIP's performance is severely impacted by the training data volume. It gives an average performance when there is enough training data (test ratio = 10% $\sim$ 20%), although still lower than BiGCN. However, with the training data decreasing, they tend to give an extraordinarily unbalanced prediction result, i.e., predicting many "high" labels and a few "low" labels. It covers almost all the "high" cases but seldom predicts "low" cases correctly. This result leads to a high precision and a low recall, resulting in a low f1 score in total. This result may be caused by the low adaptability of statistical models in a complicated situation.

3.6.2 Main Factors

The neural network based approach BiGCN is a "black box" and lacks interpretability. On the other hand, although having a lower accuracy, statistical models allow for an understanding of the main factors that influence the prediction. The BiGCN model input involves regional demographic information, land-use information, historical emergency information, hospital information, and distance information. Some information may have a significant impact on EMS demand prediction, and some may have less effect. There is a need for an understanding of the main factors that affect the model output most. The discovery of the main factors in EMS demand prediction contributes to better EMS demand estimation.

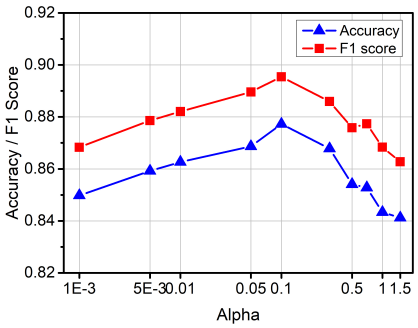
Using the statistical method ZIP, we can obtain the coefficient distribution for each variable. The coefficients are interpreted as the ones in a standard Poisson

Table 3.5: Z-score and p-value of variable coefficients in ZIP regression model (test ratio=0.1)

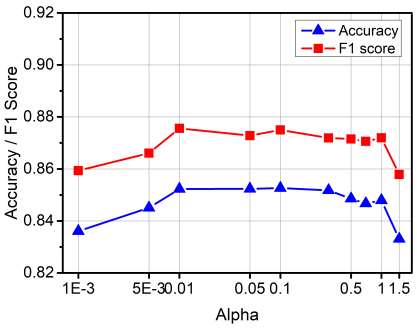
Variable name	z-score	p-value
Day time population	-24.406	0.000
Census population	62.036	0.000
EMS total case number	25.681	0.000
EMS injury case number	-898.339	0.000
EMS disease case number	898.346	0.000
Crime Number	22.873	0.000
Residential area ratio	-5.358	0.000
Business area ratio	-5.358	0.000
Industry area ration	-5.358	0.000
Ems total case number sent to hospital	26.367	0.000
EMS injury case number sent to hospital	898.338	0.000
EMS disease case number sent to hospital	-898.347	0.000
EMS total case number not sent to hospital	25.652	0.000
EMS injury case number not sent to hospital	898.339	0.000
EMS disease case number not sent to hospital	-898.346	0.000
Number of doctors	4.145	0.000
Number of beds	-20.783	0.000
EMS total case number	9.806	0.000
EMS injury case number	0.250	0.803
EMS disease case number	-12.376	0.000
EMS total case number in main wards	16.803	0.000
EMS injury case number in main wards	-20.059	0.000
EMS disease case number in main wards	-15.373	0.000
Euclidean distance	941.555	0.000

model: the excepted value of dependent variable changes by $\exp(coef.)$ for each unit increase in the corresponding variable. We do a z-test on variable coefficients. Then mainly use two statistical terms, z-score and p-value, to evaluate the variable coefficients. The z-test is a statistical test that indicates whether a variable exhibits statistical significance or exhibits a random pattern. Specifically, in this experiment, z-scores are the standard deviations of variable coefficients, and p-values are probabilities that the variable coefficients are created by some random process. Suppose a variable coefficient has a very high ($z > +2.5$) or very low ($z < -2.5$) z-score, associated with very small p-values ($p < 0.01$). In that case, this variable is likely to be statistically significant clustering or dispersion, which means it is positively/negatively correlated with the output. The larger the $|z|$ is, the stronger the relationship is.

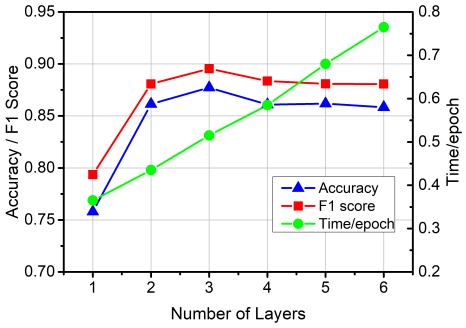
Table 3.5 demonstrates the z-scores and p-values of the variable coefficient for each feature. The upper part is the regional feature, and the lower part is the hospital feature. As a result, almost all the features significantly impact the prediction



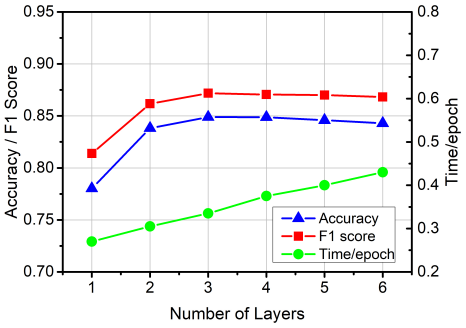
(a) Test ratio = 10%



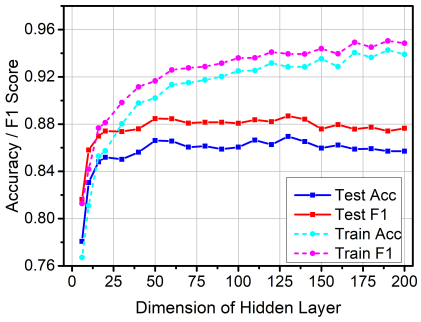
(b) Test ratio = 40%



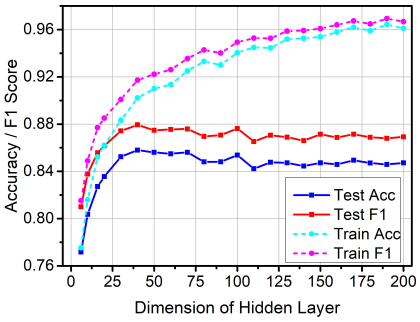
(c) Test ratio = 10%



(d) Test ratio = 40%



(e) Test ratio = 10%



(f) Test ratio = 40%

Figure 3.5: Parameter study. (a)(b) Results with different α. (c)(d) Results with different numbers of layers. (e)(f) Results with different dimensions of hidden units.

result ($|z| > 2.5$ and $p < 0.01$). Specifically, "Euclidean distance" ($z = 941.555$), regional "EMS injury case number" ($z = -898.339$) and regional "EMS disease case number" ($z = 898.346$) are three main factors that affect the EMS demand prediction result most. Features "EMS injury/disease case number sent to hospital" and "EMS injury/disease case number not sent to hospital" are linearly correlated with "EMS injury/disease case number", thus having similar z-scores. Besides, it is noticeable that the hospital "EMS injury case number" feature has $z = 0.250$ and $p = 0.803$. It means that this feature is likely to be randomly distributed and has less influence on the prediction result.

3.6.3 Parameter Study

Balancing Parameter α

As introduced in Section 3.5.2, the loss function is a combination of classification loss $\mathcal{L}_c$ and recurrent loss $\mathcal{L}_r$, with parameter α as a balancing weight. We conducted experiments to investigate how α affects performance.

Fig. 3.5(a) and 3.5(b) present the results with α ranging from 0.001 to 1.5. A small α such as 0.001 indicates placing a dominant emphasis on $\mathcal{L}_c$ and ignoring $\mathcal{L}_r$ in the cost, while a high α implies including more weight on $\mathcal{L}_r$ in the cost. We can find that the performance improves as α increases from 0.001 to 0.1, then it gradually declines. Our model reaches a high accuracy at around $\alpha = 0.1$, where recurrent loss $\mathcal{L}_r$ positively contributes to the performance.

Number of Convolutional Layers

We studied the influence of the number of convolutional layers on performance. Fig. 3.5(c) and 3.5(d) report the results. The best results are obtained with three layers. There is a modest decrease in performance when more layers are used because the learned embeddings are over-smoothed [Li et al., 2018]. Over-smoothing means if a neural network has many convolutional layers, the output node features may become indistinguishable and give a poor performance. Also, the running time grows as the number of layers increases.

Dimension of Hidden Unit

We analyzed the impact of the dimension of hidden units introduced as $\lceil c^{(T)}P \rceil + \lceil c^{(T)}Q \rceil$ in Section 3.5.1. Fig. 3.5(e) and 3.5(f) demonstrate the training and testing performance for hidden dimension ranging from 6 to 200. With the increasing hidden dimension, the test performance keeps increasing until the dimension reaches around 50, then it becomes relatively stable. However, the training performance still improves even after 50, and gradually goes far beyond the test performance, finally becomes stable after 170. The figure shows that when the hidden dimension exceeds 50, the model begins to be overfitting. The increasing hidden dimension can only fit the training set better instead of improving the test accuracy. When the increasing hidden dimension exceeds around 170, it can no longer improve the training performance. It indicates that the model has achieved its best with the current architecture and cannot easily be improved by merely adding the hidden dimension.

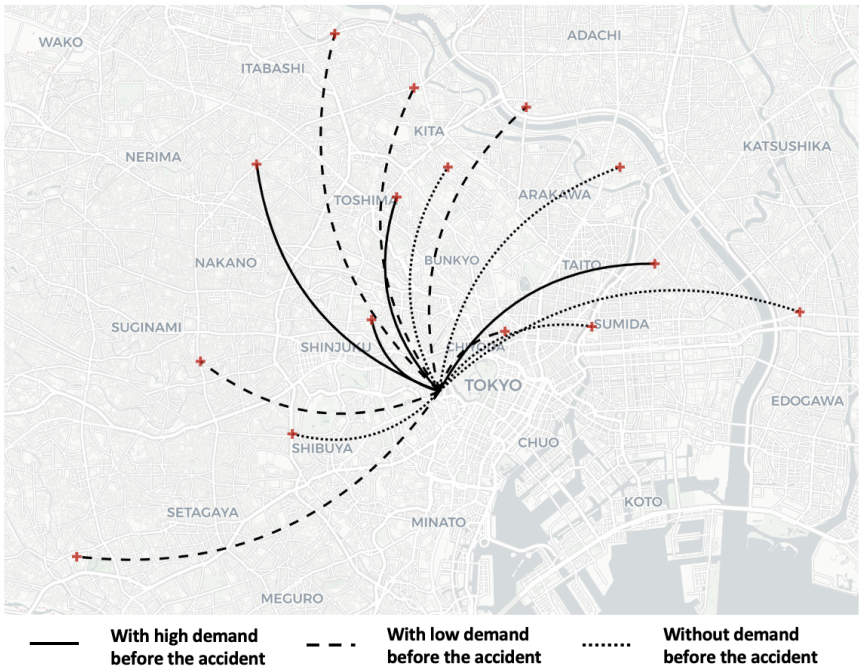


Figure 3.6: The identified hospitals with high EMS demand.

3.6.4 Case Study

Finally, we demonstrate an application of our approach in the case of a sudden emergency. We imagine a severe accident occurring in a region named "Hirakawacho", which is close to the home of many government agencies such as the National Diet Building and the Prime Minister's Official Residence. The region initially connects to four hospitals with high demand. Then, we modified the region feature *injury number* by adding 1,000 cases to simulate an injury accident where 1,000 people need first aid treatment. We used our model to predict the hospitals with "high" demand after the accident according to the new feature input.

The result is shown in Fig. 3.6. The red cross mark denotes the hospitals, and the center point indicates the region. Our model identifies eleven additional hospitals with high demand, including six hospitals with low demand and five hospitals without demand before the accident. This result is reasonable because we can see that the newly identified hospitals are all close to the region and have high capacity. Such a result is of great value for suggesting the allocation of injured people. It also helps public health emergency management in preparation for similar emergencies in the future.

3.7 Conclusion and Discussion

We analyzed the ambulance record data for Tokyo and presented the first approach to predict the EMS demand at the hospital-region level. We represented the data as a hospital-region bipartite graph and proposed a novel BiGCN model to predict

the EMS demand between hospital-region pairs. Our approach achieves excellent performance on the prediction accuracy, outperforming the baselines, including traditional machine learning algorithms, statistical models, and the latest graph-based methods by a large margin. We applied a case study to prove the feasibility of the BiGCN model in a real-world situation. Our work is meaningful to urban public health emergency management, make the public aware of the significance of EMS demand prediction, and help local governments better allocate EMS resources and decrease the emergency risk.

Limitations

In real-world scenarios, the efficiency of learning models is often contingent upon their adaptability to the dynamic and evolving nature of the underlying data. While the BiGCN model demonstrates prowess in effectively learning from static bipartite graphs, the imperative to extend its capabilities to accommodate time-series data becomes increasingly apparent. The inherent dynamism of most real-world situations necessitates a modeling framework that can seamlessly integrate temporal dynamics, capturing evolving patterns and trends. Incorporating time-series data into the BiGCN model is a compelling avenue for enhancing its predictive capacities. This extension holds significant promise in domains. For example, EMS demand predictions based on temporal dependencies can yield valuable insights into service demands, response times, and other critical metrics. By addressing the temporal dimension, BiGCN has the potential to evolve into a versatile tool capable of navigating the complexities of dynamic scenarios, contributing to advancements in predictive analytics and decision-making processes. As an integral aspect of future work, extending BiGCN to process time-series data signifies a proactive approach to advancing the model's applicability in dynamic, real-world settings.

Chapter 4

Predictive Analytics on Discrete-time Dynamic Graph

4.1 Introduction

In Chapter 3, we introduce the BiGCN model, specifically designed to learn node representation embeddings within static bipartite graph scenarios adeptly. While highlighting its efficacy in such situations, we also acknowledge its limitation in processing time-series data. As part of future work, we emphasize the imperative of extending BiGCN to enhance its versatility in dynamic, real-world settings. Based on this foundation, this chapter presents a novel multi-layer temporal GNN framework. This framework is designed to proficiently learn temporal node representations within intricate heterogeneous graphs, thereby addressing the limitations associated with static bipartite graphs.

Noteworthy is the adaptability of the proposed method, which transcends the confines of static bipartite graphs. It is tailored to handle temporal heterogeneous graphs characterized by multi-layer relations and temporal information, particularly within a sequence of graph snapshots. To assess its practicability, the proposed method undergoes rigorous evaluation across four popularity trend prediction tasks, employing real-world social media datasets. The experimental results underscore the superiority of our method, surpassing various baselines, including traditional regression approaches, prior trend prediction methods, and alternative heterogeneous GNN models. This chapter marks a significant stride in advancing the capabilities of GNNs, particularly in the context of dynamic and complex graph structures, and sets the stage for further exploration in temporal graph representation learning.

This work is now under review for *Complex & Intelligent Systems* Journal.

Application: Predicting Popularity Trends in Social Media Networks

The proliferation of the internet and the widespread adoption of mobile devices have contributed to the exponential growth in the usage of social media networks, such as X (Twitter), Facebook, YouTube, and Reddit. These platforms collectively boast hundreds of millions of users worldwide. Owing to this vast user base and the copious amounts of data continuously posted and shared, social networks have emerged as invaluable data sources for diverse areas of research, including social recommendation [Wu et al., 2019a], community detection [Chunaev, 2020], anomaly detection [Wang et al., 2022], and more. Analyzing and predicting trends within social media networks is an increasingly attractive topic [Rousidis et al., 2020]. Trend prediction refers to the process that uses historical trend data and current status data to help interpret the future estimated behavioral pattern. This research provides invaluable insights into the future trajectories of public concerns and interests grounded in the wealth of available social data. From a platform perspective, precise trend prediction holds great significance in enhancing user experiences, elevating service quality, and benefiting various applications, including advertising [Arasu et al., 2020], cryptocurrency [Poongodi et al., 2021], spatiotemporal fusion [Xiao et al., 2023], navigation [Aggarwal; Maini and Aggarwal, 2018], genomics analysis [Kaur et al., 2019], and beyond. On an individual level, anticipatory trend knowledge is empowering individuals to stay ahead of the competition, align their studies and work in the right direction, and seize opportunities for informed decision-making [Mohamed, 2023b].

Trend prediction in social media networks poses a formidable challenge because of its heterogeneity and temporal dynamics. As illustrated in Fig. 4.1, many social media networks exhibit heterogeneity, featuring various types of entities and multiple types of relations. These relations encompass connections within entities of the same type and connections between entities of different types. Assuming that the trend prediction task is to predict the visit frequency of items in the future, the representations of item entities are acquired only based on the entity attributes. However, it is necessary to consider the impact of information diffusion within entities, such as the interactions and messages exchanging within entities. Messages of social media information diffuse through the relations from one entity to another, exerting an impact on their future evolution. Thus, it is necessary to consider the impact of information diffusions within entities. Moreover, it is imperative to capture the temporal dynamics of entities at various timestamps. Social media networks are in a constant state of flux over time, which implies that entity attributes and relations undergo continuous transformations. Temporal dynamics are valuable for interpreting the change pattern of the predicted target.

Based on the analysis above, this work needs to solve the following research questions to address the challenging points mentioned above: RQ1: How to learn the relationships within entities in social media networks? RQ2: How to incorporate

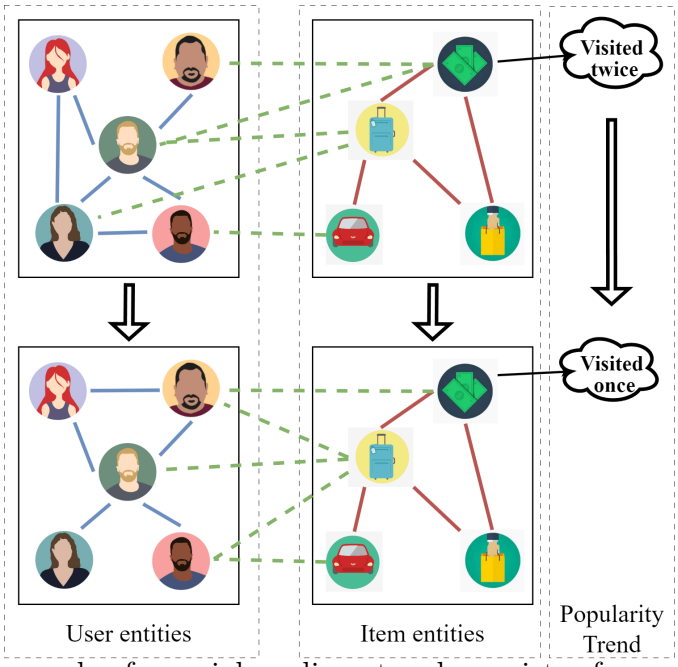


Figure 4.1: An example of a social media network consists of users and items. Solid lines denote the interactions between users and items, and dash lines denote the interactions between user-item pairs. Interactions and entity attributes evolve as time goes by. The popularity trend of an item is represented by the visit frequency, which indicates the number of accesses from users (the degree of dashed lines).

the temporal dynamics of entities and relationships into analyzing? RQ3: How to learn the dynamic representations of entities in social media networks? RQ4: How to quantify the degree of future popularity trends and predict it? RQ5: How to test the efficiency of our proposed method?

Several prior works have delved into the task of popularity trend prediction in social networks [Altshuler et al., 2012; Cao et al., 2020; Li et al., 2021a; Zeng et al., 2013]. However, these studies predominantly focus on selecting characteristics and historical statistics within social media networks, often overlooking the substantial impact of information diffusion among entities. Furthermore, most of these prior works employ simplistic machine learning or statistical models (e.g., Support Vector Machine (SVM), Logistic Regression (LR), and more) as encoders and decoders, resulting in suboptimal learning efficiency. Consequently, there exists a pressing need for an advanced framework capable of considering the influence of information diffusion in social media networks while efficiently mastering the representations of target entities.

Graph embedding methods and Graph Neural Networks (GNN) are widely regarded as ideal approaches for modeling the influence of information diffusion in social media networks. Graph embedding methods transform complex graph attributes into low-dimensional embedding vectors while maximally preserving the essential graph structural information. These methods have consistently demonstrated superior performance compared to traditional techniques for modeling graph-structured

data [Cai et al., 2018; Cui et al., 2018]. GNNs, on the other hand, belong to the category of deep learning structures designed to perform optimized transformations on all graph attributes while preserving graph topology [Wu et al., 2019b]. GNNs acquire node representations by iteratively aggregating information from neighbor nodes, mirroring the information flow dynamics seen in social media networks. Prior research has also highlighted the adaptability of GNNs in handling temporal graphs [Gao et al., 2022a; Jin et al., 2023; Liben-Nowell and Kleinberg, 2007; Sankar et al., 2020] and heterogeneous graphs [Chairatanakul et al., 2021; Dong et al., 2017; Wang et al., 2019a; Zhang et al., 2019a], laying the foundation for learning in the context of social media networks.

In light of the aforementioned considerations, this paper introduces a novel approach employing a multi-layer temporal GNN to address the task of popularity trend prediction in social media networks. The method input is a timed sequence of graph snapshots demonstrating the dynamics of social media networks, and the output is an estimated popularity attribute (e.g., visit frequency, expected revenue, and so on) in the future, indicating the popularity trends of the target entities. The novelty of the proposed method is its unified structure that integrates a multi-layer GNN encoder and a time-sequence unit to effectively learn the temporal node representations by modeling the latent influence of heterogeneous relations and temporal dynamics within social media networks (Answer to RQ1, RQ2, and RQ3). Besides, a Graph Structure Learning (GSL) module is implemented to enhance the quality of input graphs, which is particularly essential when dealing with real-world graphs. Specifically, the evolution of social media networks is characterized as a discrete-time temporal graph comprising a timed sequence of graph snapshots separated by specific time intervals. Within each graph snapshot, node attributes and the intricate multi-layer graph structures are scrutinized to forecast how a specific target node status will evolve in subsequent graph snapshots (Answer to RQ4).

The proposed method is evaluated on four real-world popularity trend prediction tasks using benchmark datasets (Answer to RQ5). The experimental results consistently demonstrate the high efficiency of the proposed approach. The approach consistently outperforms various baseline methods, including traditional linear regression algorithms, time-sequence models, prior popularity trend prediction methods, and recently introduced heterogeneous GNN methods.

In this paper, we make several significant contributions to the field of trend prediction tasks in social media networks:

- We introduce an advanced multi-layer temporal GNN framework designed to learn entity representations and predict trends within complex real-world social networks. It addresses the crucial task of forecasting trajectories of public concerns and interests based on extensive real-world data.
- We address the limitations of previous research regarding popularity trend

predictions, which often neglect the latent influence of complex relationships among entities. The proposed approach presents its novelty in considering the latent influence of information diffusion in social networks while efficiently mastering the representations of target entities, resulting in a substantial performance enhancement.

- We demonstrate the effectiveness of the proposed approach through experiments conducted on real-world social network datasets, demonstrating the feasibility of the proposed method in real-world situations. The proposed method consistently outperforms various baseline methods, including linear regression approaches, time-series models, previous popularity trend prediction methods, and recently proposed heterogeneous GNN methods.

4.2 Related Work

4.2.1 Graph Embedding Learning

Graph embedding learning has achieved notable success in various domains [Cai et al., 2018; Cui et al., 2018]. The techniques for generating embedding vectors on graphs have gained widespread recognition, particularly for downstream graph-related tasks such as node classification [Kipf and Welling, 2016a], link prediction [Zhang and Chen, 2018], and graph classification [Zhang et al., 2018a]. The primary challenge in graph learning lies in discovering effective methods to encode graph structures, encompassing nodes and edges, into low-dimensional hidden embedding vectors while maximizing the preservation of essential graph structural information.

GNNs have emerged as a robust deep learning framework for graph embedding learning [Wu et al., 2019b]. Among these, the Graph Convolutional Network (GCN) [Kipf and Welling, 2016a] stands out as one of the most widely adopted models. GCN efficiently aggregates neighbor node information through graph convolutional layers. To further enhance GNN performance, numerous researchers have introduced advanced GNN models building upon the GCN architecture. For example, GraphSAGE extends GCN into inductive learning that can handle unknown graph nodes [Hamilton et al., 2017b]. Graph Attention Networks employs attention mechanisms that assign importance weights to different neighbor nodes [Veličković et al., 2018]. Graph Autoencoder and Variational Graph Autoencoder are GCN-based encoder-decoder models that can handle unsupervised learning tasks [Kipf and Welling, 2016b]. Moreover, ML-GCN and ML-GAT extend the original GCN and GAT to multi-layer networks, which can capture more complex relations among nodes within large-scale graphs [Zangari et al., 2021].

Learning on temporal graphs presents considerably greater complexity compared to static graphs. Numerous studies have focused on discrete-time temporal graphs,

which consist of a timed sequence of graph snapshots [Gao et al., 2022a; Leblay et al., 2020; Liben-Nowell and Kleinberg, 2007; Sankar et al., 2020]. Existing static graph methods can be directly applied to each individual graph snapshot. In parallel, substantial research efforts have been dedicated to graph embedding learning for heterogeneous graphs [Chairatanakul et al., 2021; Dong et al., 2017]. Heterogeneous graphs encompass nodes and edges of multiple types. HetGNN combines heterogeneous structural information and node attributes. It performs excellently in graphs with multiple types of nodes and edges [Zhang et al., 2019a]. HAN is a heterogeneous GNN based on the hierarchical attention mechanism, including node-level and semantic-level attention [Wang et al., 2019a]. It fully considers the importance of node neighbors and different meta-paths. This research endeavors to develop a unified framework that integrates the characteristics of discrete-time temporal graphs and multi-layer heterogeneous graphs.

4.2.2 Trends Prediction Tasks in Real-world Data

Time-aware trend prediction tasks have recently attracted much attention in both academia and industry [Altshuler et al., 2012; Mohamed, 2023a; Rousidis et al., 2020; Zeng et al., 2013]. Existing works about trend prediction can be categorized into two primary patterns: The first is to predict the growth and decline of entities based on past characteristics and early-stage patterns. Yang et al. found that the temporal patterns reveal how the content popularity fluctuates during the post-propagation [Yang and Leskovec, 2011]. The second is to predict the value of specific target attributes based on temporal attributes or dynamic signals. Zhao et al. incorporated human reaction time as temporal variables in self-exciting point processes [Zhao et al., 2015]. Also, some hybrid methods between metaheuristics and machine learning lead to a novel research field, which successfully combines machine learning and swarm intelligence approaches and proved capable of obtaining outstanding results in different trend prediction areas [Bacanin et al., 2021, 2022; Malakar et al., 2020].

Employing graph-based methods to tackle trend prediction tasks is a relatively recent research direction. He et al. designed a time-aware bipartite graph for estimating the future popularity of items [He et al., 2014]. Cao et al. developed a coupled GNN model to solve the popularity trend prediction task [Cao et al., 2020]. Li et al. adopted a graph kernel approach to predict the node popularity within a cascade graph sequence [Li et al., 2021a]. Hou et al. proposed a spatial–temporal multi-graph convolutional network for casualty prediction of terrorist attacks [Hou et al., 2023]. Yang et al. predicted traffic propagation flow in urban road networks with a multi-graph convolutional network model [Yang et al., 2023]. Mohamed et al. proposed an Internet of Things (IoT) for voice communication [Mohamed et al., 2023]. However, these methods did not account for multiple relations among various

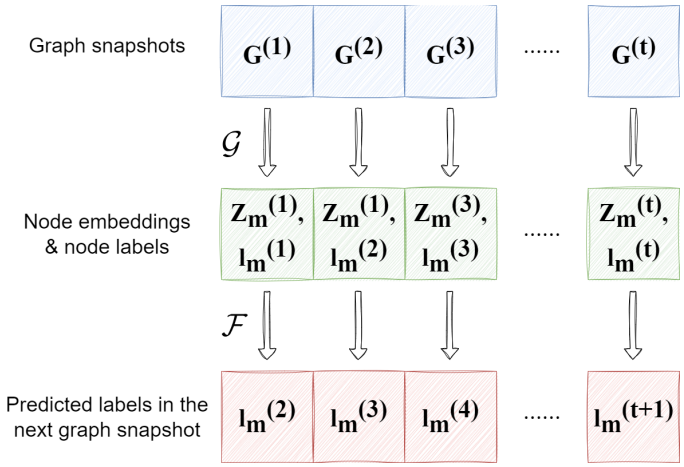


Figure 4.2: Problem setup of the popularity trend prediction in social networks. The social networks are presented as a sequence of graph snapshots $\mathbf{G} = \{\mathbf{G}^{(1)} \sim \mathbf{G}^{(t)}\}$. The proposed method first learns the node embeddings $\mathbf{Z_m}^{(1)} \sim \mathbf{Z_m}^{(t)}$ of the target node type m by an encoder $\mathcal{G}$, then predicts the node labels $\mathbf{l_m}^{(2)} \sim \mathbf{l_m}^{(t+1)}$ in the upcoming graph snapshots by a decoder $\mathcal{F}$.

types of entities, limiting their applicability in complex real-world scenarios.

4.3 Problem Setup

The temporal social network data is structured as a sequence of graph snapshots, each separated by a specific time interval. Many social networks exhibit heterogeneity, featuring various types of entities and multiple types of relations. These relations encompass connections within entities of the same type and connections between entities of different types. To represent such complicated networks, a sequence of graph snapshots $\mathbf{G} = \{\mathbf{G}^{(t)} \mid t \in \{1, 2, \cdots, T\}\}$ is proposed, where each graph snapshot $\mathbf{G}^{(t)} = \{\mathbf{V}, \mathbf{E}, \mathbf{F}\}$ contains node sets of different types $\mathbf{V} = \{\mathbf{V_n} \mid n \in \{1, 2, \cdots, N\}\}$, edge sets $\mathbf{E} = \{\mathbf{E_{jk}} \mid j, k \in \{1, 2, \cdots, N\}\}$ between two nodes in $\mathbf{V_j}$ and $\mathbf{V_k}$, and node attribute matrices $\mathbf{F} = \{\mathbf{F_n} \mid n \in \{1, 2, \cdots, N\}\}$ of each node set. Specifically, for edge set $\mathbf{E_{jk}}$, if $j \neq k$, $\mathbf{E_{jk}}$ present the bipartite relations of node pairs between $\mathbf{V_j}$ and $\mathbf{V_k}$. If $j = k$, $\mathbf{E_{jk}}$ present the intra-relations within node set $\mathbf{V_j}/\mathbf{V_k}$.

The research problem is set up as demonstrated in Fig. 5.2. The task is to predict the popularity trend on entities of a certain target type m, quantified by a label $\mathbf{L}_m$. The proposed method first learns the embeddings $\mathbf{Z}_m^{(t)}$ of nodes in $\mathbf{V}_m$ of each graph snapshot $\mathbf{G^{(t)}}$ by an encoder $\mathcal{G}$. Specifically, node embedding $\mathbf{Z}_m^{(t)}$ is obtained based on the node attribute matrices $\mathbf{F}$ and edges $\{\mathbf{E_{mj}} \mid j \in \{1, 2, \cdots, N\}\}$ connecting the target nodes, as formulated in (4.1):

$$\mathcal{G} : (\mathbf{V}, \{\mathbf{E_{mj}} \mid j \in \{1, 2, \cdots, N\}\}, \mathbf{F}) \rightarrow \mathbf{Z}_m^{(t)}. \tag{4.1}$$

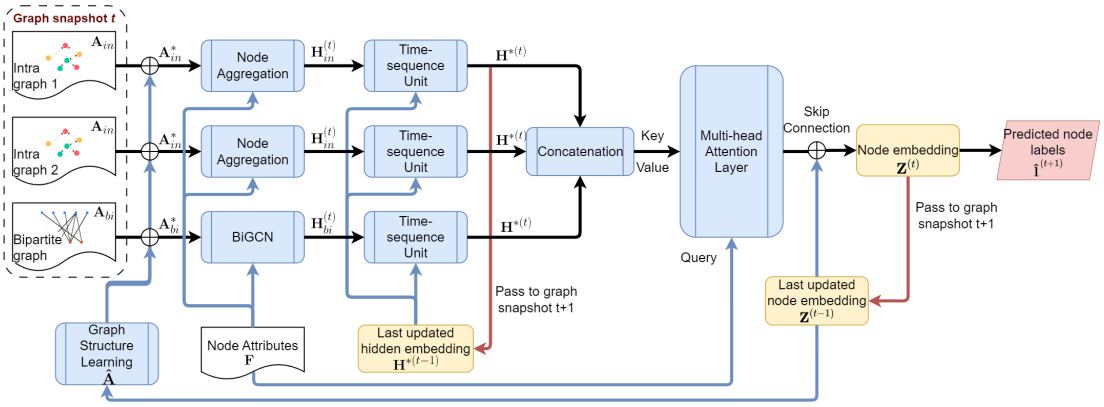


Figure 4.3: Overview of the proposed multi-layer temporal GNN framework. Inputs are depicted as white blocks, main components are highlighted in blue, intermediate results are presented in yellow, and final outputs are shown in red. Intra graphs comprise the multiple relations within the target entities of the same type. The bipartite graph indicates the relationship between entities of different types.

Labels $\mathbf{l}_m^{(t)} \in \mathbf{L}_m$ are assigned to nodes in $\mathbf{V}_m$ of each graph snapshot $\mathbf{G^{(t)}}$. The label indicates the target value to be predicted. We aim to using learned node embeddings $\mathbf{Z}_m^{(t)}$ and known node labels $\mathbf{l}_m^{(t)}$ to predict the node labels $\hat{\mathbf{l}}_m^{(t+1)}$ in the upcoming graph snapshot $\mathbf{G^{(t+1)}}$ by a decoder $\mathcal{F}$, as formulated in (4.2):

$$\mathcal{F} : (\mathbf{Z}_m^{(t)}, \mathbf{l}_m^{(t)}) \rightarrow \hat{\mathbf{l}}_m^{(t+1)}. \tag{4.2}$$

4.4 Methodology

Previous studies regarding popularity trend prediction tasks in social networks suffer from neglecting the latent influence of complex relationships among entities. To overcome the limitations in prior work, this work introduces an advanced multi-layer temporal GNN framework designed to effectively capture the intricate relations and learn the representation embeddings of target entities in social networks.

In the following, we first demonstrate the overall framework of the proposed method. Then, we explain the details of the key components. Finally, we analyze the computational complexity of the proposed methods.

4.4.1 Overview

Fig. 5.5 illustrates an overview of the proposed method. The input is a timed sequence of graph snapshots $\mathbf{G} = \{\mathbf{G^{(t)}} \mid t \in \{1, 2, \cdots, T\}\}$. Each graph snapshot contains adjacency matrices $\mathbf{A}$, which is a tensor for expressing multiple adjacency matrices, along with a node attribute matrix $\mathbf{F}$. The input adjacency matrices encompass 'intra graphs' $\mathbf{A_{in}}$ and 'bipartite graphs' $\mathbf{A_{bi}}$. Intra graphs are homogeneous, signifying the multiple relations within the target entities of the same type,

while bipartite graphs represent the relationship between entities of different types. Node attributes $\mathbf{F}$ are shared across all the graphs, with only edge weights varying between individual graphs.

This framework mainly has four key components: Graph Structure Learning (GSL) module, Node aggregation module, Time-sequence unit, and Multi-head attention layer. Firstly, input graphs $\mathbf{A}$ are enhanced by the GSL module, which is employed to improve the quality of input adjacency matrices by a refined graph structure $\hat{\mathbf{A}}$ learned from the node embeddings $\mathbf{Z}^{(t-1)}$ of the last iteration. Next, the enhanced adjacency matrices $\mathbf{A}^*_{\mathbf{in}}$ and $\mathbf{A}^*_{\mathbf{bi}}$ are fed to Node Aggregation module to learn the hidden node embeddings $\mathbf{H}^{(t)}_{in}$ and $\mathbf{H}^{(t)}_{bi}$ by the information on neighbor nodes. Then, $\mathbf{H}^{(t)}_{in}$ and $\mathbf{H}^{(t)}_{bi}$ are sent to the Time-sequence Unit to incorporate the temporal information $\mathbf{H}^{*(t-1)}_{in}$ between the adjacent graph snapshots. Finally, a Multi-head Attention layer receives the merged hidden node embeddings $\mathbf{H}^{*(t)}_{in}$ and node attributes $\mathbf{F}$ for the learning of node embeddings. The learned node embeddings $\mathbf{Z}^{(t)}$ are responsible for predicting the node labels $\hat{\mathbf{l}}^{(t+1)}$, representing the target attribute for popularity trend prediction.

The details of key components are introduced in the following:

4.4.2 Graph Structure Learning

Noisy or incomplete graphs can often lead to suboptimal representations, hindering the efficient learning of node embeddings, especially when dealing with real-world graph data. GSL is designed to jointly learn an optimized graph structure and corresponding adjacency matrix [Zhu et al., 2021a]. Based on the assumption that edges tend to connect nodes with similar representation, the proposed method refines the input graphs by computing the similarity between the last updated node embedding pairs, which are then used as the edge weights.

Euclidean distance is no longer an effective metric for graph node similarity. Instead, a generalized adaptive Mahalanobis distance is employed to quantify node similarity, as formulated in (4.3):

$$\phi(\mathbf{z}_i, \mathbf{z}_j) = \sqrt{(\mathbf{z}_i - \mathbf{z}_j)^T \mathbf{W}_d \mathbf{W}_d^T (\mathbf{z}_i - \mathbf{z}_j)}, \tag{4.3}$$

where $\mathbf{z}_i, \mathbf{z}_j$ denote the embedding of two node i and j learned in the last graph snapshot, and $\mathbf{W}_d$ is a trainable weight. $\mathbf{W}_d \mathbf{W}_d^T$ constructs a symmetric positive semi-definite trainable parameter. Mahalanobis distance can adaptively fit the task and the node embeddings during training. Then, this distance is exploited to calculate the refined adjacency matrix by exploiting the Gaussian kernel:

$$\hat{\mathbf{A}}_{\mathbf{ij}} = \exp\left(\frac{-\phi(\mathbf{z}_i, \mathbf{z}_j)}{2\sigma^2}\right), \tag{4.4}$$

where $\hat{\mathbf{A}}_{\mathbf{ij}}$ denote the elements in the refined adjacency matrix $\hat{\mathbf{A}}$, and σ denote the size of Gaussian kernel.

After obtaining the refined graph adjacency matrix $\hat{\mathbf{A}}$, the residual connections [He et al., 2016] is used to incorporate the original input adjacency matrices $\mathbf{A}$ by a hyperparameter α, as formulated in (4.5):

$$\mathbf{A}^* = \alpha\hat{\mathbf{A}} + (1-\alpha)\mathbf{A}, \tag{4.5}$$

where α is used to mediate the influence of each part, and $\mathbf{A}^*$ is the finally improved adjacency matrix.

4.4.3 Node Aggregation

The core of GNN is recurrently aggregating information from neighbor nodes. The proposed method uses the graph convolutional layer [Kipf and Welling, 2017] to aggregate the neighbor node information in intra graphs $\mathbf{A}^*_{\mathbf{in}}$, as formulated in (4.6):

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}^*_{\mathbf{in}}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l+1)}_{in}), \tag{4.6}$$

where $\tilde{\mathbf{A}^*_{\mathbf{in}}} = \mathbf{A}^*_{\mathbf{in}} + \mathbf{I}$, $\tilde{\mathbf{D}}$ is the degree diagonal matrix of $\tilde{\mathbf{A}_{\mathbf{in}}}*$, $\mathbf{H}^{(l)}$ is the node embedding matrix in l-th layer, $\mathbf{H}^{(0)}$ is initialized with the node feature matrix $\mathbf{F}$, $\mathbf{W}^{(l)}_{in}$ is the trainable weight matrix in l-th layer, and $\sigma(\cdot)$ is the activation function. The result of the last graph convolutional layer is stored as the output of node aggregation on intra graphs, which is denoted as $\mathbf{H}^{(t)}_{in}$. t denotes the t-th graph snapshot $\mathbf{G}^{(t)}$.

Specifically, for bipartite graph $\mathbf{A}^*_{\mathbf{bi}}$ that models the relation between the target node set $\mathbf{V}_m$ and other types of node set $\mathbf{V}_k$, a dedicated bipartite GCN layer [Jin et al., 2021] is exploited to separately execute graph convolutional operations on $\mathbf{V}_m$ and $\mathbf{V}_k$, as formulated in (4.7) and (4.8):

$$\mathbf{H}^{(l+1)}_m = \sigma(\Big[\mathbf{D}^{-1}_m\mathbf{B}_m\mathbf{H}^{(l)}_k\mathbf{W}^{(l+1)}_m \parallel \mathbf{F}_m\boldsymbol{\omega}^{(l+1)}_j\Big]) \tag{4.7}$$

$$\mathbf{H}^{(l+1)}_k = \sigma(\Big[\mathbf{D}^{-1}_k\mathbf{B}_k\mathbf{H}^{(l)}_m\mathbf{W}^{(l+1)}_k \parallel \mathbf{F}_k\boldsymbol{\omega}^{(l+1)}_k\Big]), \tag{4.8}$$

where $\mathbf{B}_m \in \mathcal{R}^{\|\mathbf{V}_m\|\times\|\mathbf{V}_k\|}$ and $\mathbf{B}_k \in \mathcal{R}^{\|\mathbf{V}_k\|\times\|\mathbf{V}_m\|}$ are the incidence matrix for two node sets $\mathbf{V}_m$ and $\mathbf{V}_k$ in the bipartite graph $\mathbf{A}^*_{\mathbf{bi}}$, respectively, $\mathbf{D}_m = \mathrm{Diag}(\sum_i \mathbf{B}_{m(1,i)}, \ldots, \sum_i \mathbf{B}_{m(\|\mathbf{V}_m\|}$ and $\mathbf{D}_k = \mathrm{Diag}(\sum_i \mathbf{B}_{k(1,i)}, \ldots, \sum_i \mathbf{B}_{k(\|\mathbf{V}_k\|,i)})$ are the diagonal degree matrices of $\mathbf{V}_m$ and $\mathbf{V}_k$, $\mathbf{H}^{(l)}_m$ and $\mathbf{H}^{(l)}_k$ are the learned node embedding matrix in the l-th layer. $\mathbf{W}^{(l)}_m$, $\mathbf{W}^{(l)}_k$, $\boldsymbol{\omega}^{(l)}_m$, and $\boldsymbol{\omega}^{(l)}_k$ are trainable parameters in the l-th layer. $\parallel$ is the concatenation operation.

The BiGCN layer discards the result $\mathbf{H}_k^{(l)}$ of other node set, and only outputs the result $\mathbf{H}_m^{(l)}$ on the target node set $\mathbf{V}_m$. The result is denoted as $\mathbf{H}_{bi}^{(t)}$, where t denotes the t-th graph snapshot $\mathbf{G}^{(t)}$.

4.4.4 Time-sequence Unit

Time-sequence Unit receives the most recently updated node embeddings from the previous graph snapshot and transmits these learned node embeddings to the subsequent graph snapshot. An LSTM cell [Hochreiter and Schmidhuber, 1997] is implemented to achieve this function.

$$\mathbf{h}_t, C_t = LSTMCell(\mathbf{X}_t, \mathbf{h}_{t-1}, C_{t-1}) \tag{4.9}$$

$$\mathbf{H}^{*(t)}, C_t = LSTMCell(\mathbf{H}^{(t)}, \mathbf{H}^{*(t-1)}, C_{t-1}). \tag{4.10}$$

LSTM cell has three inputs and two outputs: input embedding $\mathbf{X}_t$, input hidden state $\mathbf{h}_{t-1}$, input cell state C_{t-1}, output hidden state $\mathbf{h}_t$, and output cell state C_t, as formulated in (4.9). Outputs from Node Aggregation module $\mathbf{H}_{in}^{(t)}$ and $\mathbf{H}_{bi}^{(t)}$ (denoted as $\mathbf{H}^{(t)}$ for convenience in the following) are fed into the LSTM cell as $\mathbf{X}_t$, and the updated hidden embeddings $\mathbf{H}^{*(t-1)}$ from last graph snapshot $\mathbf{G}^{(t-1)}$ are fed into LSTM cell as the input hidden state $\mathbf{h}_{t-1}$. The output hidden state $\mathbf{h}_t$ will be stored as the updated hidden embeddings $\mathbf{H}^{*(t)}$. $\mathbf{H}^{*(t)}$ and cell state C_t are passed to the following graph snapshot $\mathbf{G}^{(t+1)}$. The LSTM cells are implemented for each individual graph. The inputs and outputs are formulated in (4.10).

4.4.5 Multi-head Attention Layer

After the Time-sequence Unit, a concatenation layer is exploited to merge the updated hidden embeddings from each LSTM cell as a single hidden embedding matrix, denoted as $\mathbf{H}^{*(t)}$. Then, a multi-head attention layer is exploited to obtain the node embeddings. Multi-head attention mechanism originated from the famous Transformer [Vaswani et al., 2017] and proved efficient in learning temporal node representation [Rossi et al., 2020; Wang et al., 2021; Xu et al., 2020].

The multi-head attention layer plays an efficient role in aggregating neighbor node embeddings and node attributes. The layer operates by computing the dot product of a query vector with a set of key vectors, ultimately generating a weight vector that assigns importance scores to the corresponding value vectors. The mechanism is defined as follows:

$$\mathbf{Q} = \mathbf{F}\mathbf{W}_Q \tag{4.11}$$

$$\mathbf{K} = \mathbf{H}^{*(t)}\mathbf{W}_K, \mathbf{V} = \mathbf{H}^{*(t)}\mathbf{W}_V \tag{4.12}$$

$$head_i = Attn\left(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i\right) \tag{4.13}$$

$$= softmax\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d}}\right)\mathbf{V}_i \tag{4.14}$$

$$MultiHead\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = Concat(head_1, head_2 \cdots, head_h)\mathbf{W}. \tag{4.15}$$

In the multi-head attention layer, the node hidden embedding $\mathbf{H}^{*(t)}$ is passed to the key and value parameters $\mathbf{K}$ and $\mathbf{V}$, and the node attribute matrix $\mathbf{F}$ is passed to the query parameter $\mathbf{Q}$. $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ and $\mathbf{W}$ are trainable parameters. $Attn(\cdot)$ is a scaled dot-product attention encoder implemented by the softmax function, where d denotes the dimension of the query parameter. $MultiHead(\cdot)$ concatenates the attention score for each head into a single attention score matrix.

Besides, a skip-connection is exploited to add the output attention score matrix $MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ and the last updated node embedding matrix $\mathbf{Z}^{(t-1)}$ together, which is passed from the last graph snapshot $\mathbf{G}^{(t-1)}$. The skip-connection is formulated in (4.16):

$$\mathbf{Z}^{(t)} = \mathbf{Z}^{(t-1)} + MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \tag{4.16}$$

$$= \mathbf{Z}^{(t-1)} + MultiHead(\mathbf{F}, \mathbf{H}^{*(t)}, \mathbf{H}^{*(t)}) \tag{4.17}$$

The result of skip-connection $\mathbf{Z}^{(t)}$ is the final obtained node embedding matrix.

4.4.6 Loss Function

The obtained node embedding $\mathbf{Z}^{(t)}$ is used to predict the node labels $\hat{\mathbf{l}}^{(t+1)}$ by exploiting an two-layer MLP decoder. The predicted node labels $\hat{\mathbf{l}}^{(t+1)}$ need to be as close as the target value of popularity trend prediction $\mathbf{l}^{(t+1)}$ in the next graph snapshot $\mathbf{G}^{(t+1)}$. Thus, it is a regression problem between the predicted value $\hat{\mathbf{l}}^{(t+1)}$ and actual value $\mathbf{l}^{(t+1)}$. The Mean Squared Error(MSE) loss is exploited to train the parameters:

$$\mathcal{L}_{reg} = MSELoss(\hat{\mathbf{l}}^{(t+1)}, \mathbf{l}^{(t+1)}) \tag{4.18}$$

$$= \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{l}}_i^{(t+1)} - \mathbf{l}_i^{(t+1)}), \tag{4.19}$$

where N denotes the total number of nodes in the training set.

Besides, the learnable parameter in the GSL module also needs restraints to accelerate the training and increase the stability of the learned graph topology structure. An MSE loss is exploited to restrain the gap between the refined $\hat{\mathbf{A}}$ and the original $\mathbf{A}$ for each graph:

$$\mathcal{L}_{gsl} = \sum^{\|\mathbf{A}\|} MSELoss(\hat{\mathbf{A}}, \mathbf{A}) \tag{4.20}$$

$$= \sum^{\|\mathbf{A}\|} \frac{1}{N} \sum_{i=1}^{N} (\hat{\mathbf{A}}_i - \mathbf{A}_i), \tag{4.21}$$

where $\|\mathbf{A}\|$ denotes the number of adjacency matrices in the tensor $\mathbf{A}$.

Finally, the total loss is:

$$\mathcal{L} = \mathcal{L}_{reg} + \lambda \mathcal{L}_{gsl}, \tag{4.22}$$

where λ is a hyperparameter for balancing two losses.

4.4.7 Computational Complexity

This section conducts a computational complexity analysis of the proposed method, with a particular focus on key components, including the GSL module, Node Aggregation module, Time-sequence Unit, and the Multi-head Attention layer. Regarding Eqs.(4.3), (4.4), (4.6)∼(5.8), the complexity is primarily dominated by matrix multiplication. For convenience, this analysis assumes that the average node number of each node set is N, and both the node attributes $\mathbf{F}$ and node embeddings ($\mathbf{H}^{(t)}$ and $\mathbf{Z}^{(t)}$) share the same hidden dimension D. The computational complexity is calculated as follows:

The complexity of the GSL module is $\mathcal{O}(N^2D^2)$. The complexity of the graph convolutional layer in Node Aggregation is $\mathcal{O}(N^2D + ND^2)$. The complexity of the LSTM cell is $\mathcal{O}(ND)$. The complexity of the Multi-head Attention layer is $\mathcal{O}(N^2D + ND^2)$. GSL module costs the most time complexity, followed by the Node Aggregation module and Multi-head Attention layer. Time-sequence Unit costs the least time complexity. As a result, the total complexity of the proposed method is $\mathcal{O}(N^2D^2) + \mathcal{O}(N^2D + ND^2) + \mathcal{O}(ND) + \mathcal{O}(ND^2) + \mathcal{O}(N^2D + ND^2) = \mathcal{O}(ND(ND + 2N + 3D + 1)) = \mathcal{O}(N^2D^2)$.

4.5 Experiments

In this section, the effectiveness of the proposed method is validated in real-world popularity trend prediction tasks. Experiments are conducted on four social media network datasets and evaluate the performance of the proposed method against baseline methods. Additionally, the sensitivity of the model hyperparameters is

Table 4.1: Statistics of the social media network datasets used in our experiments.

Dataset	YouTube	MOOC	Reddit	Wikipedia
Durations	16 months	1 month	1 month	1 month
# Items	1,358	98	1,000	1,000
# Users	13.6k	7,047	10,000	8,227
# Edges	49.9k	411,749	672,447	157,474

evaluated. Furthermore, an ablation study is performed to assess the significance of the key modules within the proposed method.

4.5.1 Dataset Description

The real-world social media network datasets used in the experiments are listed in Table 5.5.

- **YouTube live streaming dataset**[1] [Uechi, 2022]: this public dataset comprises information on 1,358 live streaming channels and more than 136,000 viewers, spanning from April 2021 to July 2022. The attributes of both channels and viewers are represented as vectors. Relations within channels include the rate of common viewers, the conflict in streaming time, and the similarity of streaming content. Relations between channels and viewers present the interaction of donations and the amount of income. The dataset is separated into several graph snapshots, each separated by a one-month interval. The objective of the popularity trend prediction task within this dataset is to forecast the donation income for each channel in the upcoming months.

- **MOOC students and courses**[2] [Kumar et al., 2019]: this public dataset encompasses interactions performed by students on MOOC (Massive Open Online Course) platform, involving 7,047 users engaging with 98 courses, resulting in a total of 411,749 interactions. Both student and course attributes are represented as feature vectors. The dataset is separated into several graph snapshots, each separated by a two-day interval. The objective of the popularity trend prediction task within this dataset is to predict the frequency of visits to each course in the forthcoming days.

- **Reddit post dataset**[3] [Kumar et al., 2019]: this public dataset comprises one month of posts from 10,000 active users on the 1,000 most active topics on the Reddit community forum, resulting in 672,447 interactions. The text of each post is converted into feature vectors. The dataset is separated into

[1]https://www.kaggle.com/datasets/uetchy/vtuber-livechat
[2]https://snap.stanford.edu/jodie/
[3]https://snap.stanford.edu/jodie/

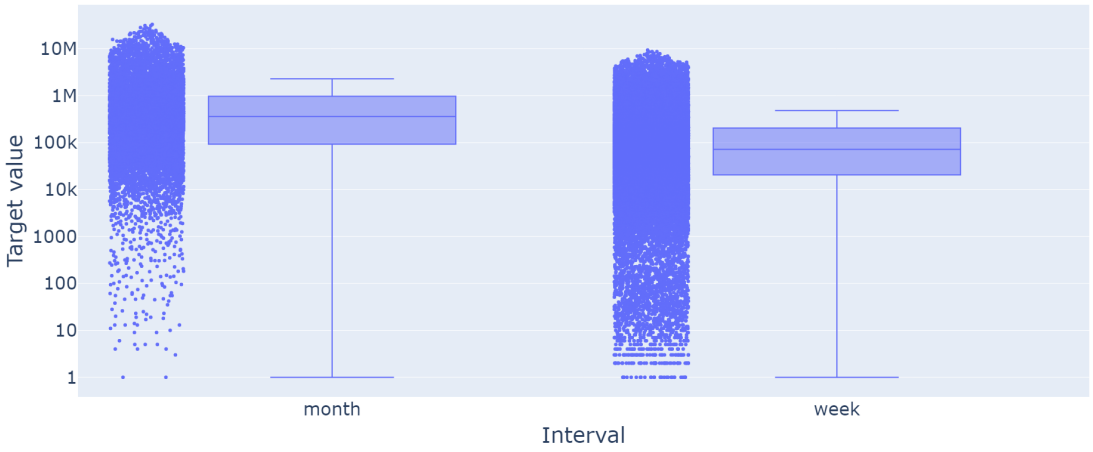


Figure 4.4: The box plot of the target values (true labels) in the YouTube live streaming dataset. The target values are distributed from 1 to more than 10 million. "Month" and "Week" denotes the interval of graph snapshots (one-month interval and one-week interval).

several graph snapshots, each separated by a two-day interval. The objective of the popularity trend prediction task within this dataset is to predict the post numbers on each topic in the upcoming days.

- **Wikipedia edits**[4] [Kumar et al., 2019]: this public dataset encompasses one month of edits made to Wikipedia pages, comprising the 1,000 most frequently edited pages. It involves 8,227 editors and a total of 157,474 edit records. Like the Reddit dataset, the edit text is transformed into feature vectors. The dataset is separated into several graph snapshots, each separated by a two-day interval. The objective of the popularity trend prediction task within this dataset is to predict the frequency of edits on each page in the forthcoming days.

This work aims to predict values that reflect the popularity degree of target entities. These values correspond to the amount of income (YouTube live streaming dataset), the number of visits (MOOC and Reddit datasets), and the number of edits (Wikipedia dataset). However, these values often vary across different magnitudes, ranging from 1 to more than 10 million in some cases. For example, the distribution of target values in the YouTube live streaming dataset is illustrated in Fig. 4.4. Predicting target values across such a large and imbalanced distribution of magnitudes poses a challenge for the proposed model, potentially resulting in significant biases in the prediction results.

To address this challenge, we implement a transformation on the prediction target values, converting them into percentages of changes. Thus, the proposed model predicts the percentage of change in the popularity degree values rather than the

[4]https://snap.stanford.edu/jodie/

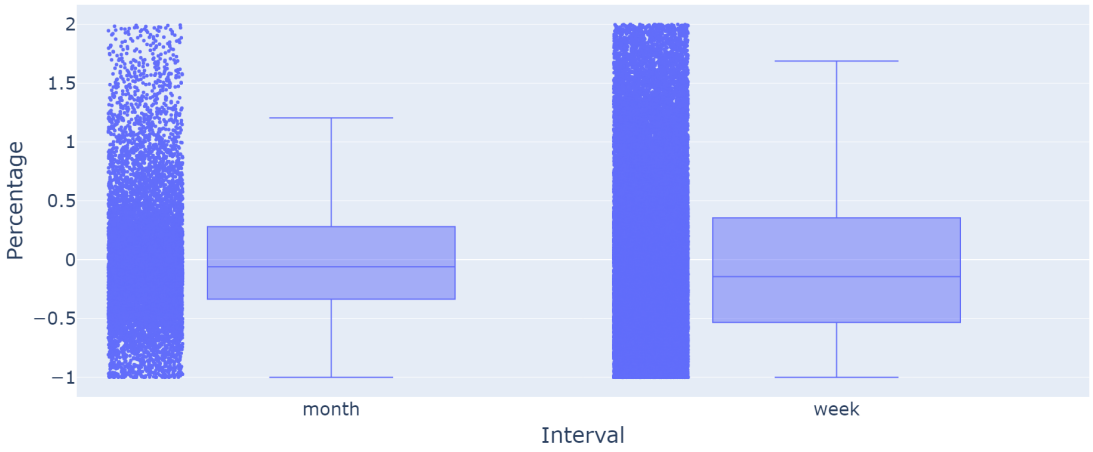


Figure 4.5: The box plot of the improved percentage of change on target values (true labels) in YouTube live streaming dataset. The percentages are distributed from -1 to 2 after eliminating outliers. "Month" and "Week" denotes the interval of graph snapshots (one-month interval and one-week interval).

original values. As depicted in Fig 4.5, the percentages only range from -1 to 2 after eliminating outliers. Consequently, the proposed model can provide more accurate predictions within this smaller range. This approach allows us to mitigate the negative impact stemming from the vast magnitudes of target values, which are influenced by confounding effects.

4.5.2 Setup of the Experiment

The datasets introduced above provide the intra graphs of item entities, bipartite graphs representing user-item relationships, and feature vectors for the input. Specifically, in cases where the datasets solely comprise bipartite graphs of user-item pairs, the intra graph is constructed based on the two-hop neighborhood of item entities. It means that item entities are interconnected in the intra graph if they share connections with the same user entity in the bipartite graph.

Experiments are executed on a platform with Intel Xeon Platinum 8360Y CPU and NVIDIA A100 for NVLink 40GiB HBM2 GPU. The standard model hyperparameters are fine-tuned, including the interval of graph snapshots, learning rate, epochs, and dropout rate. Regarding the dataset configuration, we test different choices of the interval of graph snapshots for each dataset and finally determine the choice with the best performance: a one-month interval for the YouTube live streaming dataset and a two-day interval for the MOOC, Reddit, and Wikipedia datasets. During training, the Adam optimizer is employed with a learning rate set to 0.01. The model is trained for 30 epochs, including training, validation, and testing phases. A dropout rate of 0.2 is applied. The multi-head attention layer configures two attention heads. Furthermore, an early stopping strategy is

Table 4.2: Input information required by the baseline models.

	Node attributes	Graph structure	Temporal information
Linear regression algorithm			
GBR	✓	×	×
XGBoost	✓	×	×
MLP	✓	×	×
Time-sequence model			
LSTM-FCN	✓	×	✓
Static heterogeneous graph method			
HetGNN	✓	✓	×
HAN	✓	✓	×
Temporal heterogeneous graph methods			
HGT	✓	✓	✓
CoupledGNN	✓	✓	✓
Proposed method	✓	✓	✓

implemented, which halts training if the validation loss fails to decrease for five consecutive iterations. Additionally, sensitivity experiments are performed for specific hyperparameters through grid search, as explained later.

All the configurations above are determined by manual grid search. Regarding the sensitivity analyses of the configurations above, the interval of graph snapshots severely impacts the model performances because the information in graph snapshots is totally different with distinct intervals. In comparison, the learning rate and the dropout rate are less sensitive to the experimental results. Finally, as introduced above, the number of attention heads in the multi-head attention layer significantly impacts the computational costs.

4.5.3 Baselines

The following baseline methods are included for comparison in experiments, including basic linear regression approaches, time-sequence models, previous popularity trend prediction methods, and recently proposed heterogeneous GNN methods:

1. **Gradient Boost Regressor (GBR)**: A gradient boost regressor from the scikit-learn toolkit.

2. **XGBoost**: An ensemble gradient boosting decision tree model from XGBoost library.

3. **Multi-layer Perceptron (MLP)**: A basic two-layer fully connection neural network.

4. **LSTM-FCN**[5] [Karim et al., 2018]: A time-sequence deep learning model combining LSTM and a fully convolutional network.

[5]https://github.com/titu1994/LSTM-FCN

5. **HetGNN**[6] [Zhang et al., 2019a]: A GNN model for learning node embedding representations in heterogeneous graphs.

6. **HAN**[7] [Wang et al., 2019a]: A heterogeneous graph attention network based on the hierarchical node-level and semantic-level attentions.

7. **HGT**[8] [Hu et al., 2020]: A heterogeneous graph transformer architecture that can deal with large-scale heterogeneous and temporal graphs.

8. **CoupledGNN**[9] [Cao et al., 2020]: A model solves the network-aware popularity prediction problem, capturing the cascading effect explicitly by two coupled GNNs.

Table 5.6 presents the required input information for each baseline method. The default parameter settings are maintained during training and testing for all baseline methods. The specific experimental configurations are as follows:

- Linear regression algorithms (GBR and XGBoost) and the Neural Network (MLP) receive every feature vector and learn to predict the target value.

- Time-sequence models (LSTM-FCN) handle the time-sequence data, taking in sequences of feature vectors as input. These models generate a sequence of hidden state embeddings, which are subsequently used for predicting the target value.

- Static heterogeneous graph methods (HetGNN and HAN) are typically applied to model graph-structured data containing multiple types of entities and relations. In the case of this work, the user-item relations in the datasets can be viewed as a unique heterogeneous graph. These models receive feature vectors and the graph structure information as input.

- Temporal heterogeneous graph methods (HGT and CoupledGNN) operate on dynamic heterogeneous graphs encompassing node attributes, graph structure, and temporal information. These models are considered the strongest baseline methods compared to the proposed method.

4.5.4 Evaluation

Table 5.8 and Fig. 4.6 present the experimental results for the popularity trend prediction tasks. The performance evaluation is based on the Rooted Mean Squared Error (RMSE) score and the Mean Absolute Percentage Error (MAPE):

[6]https://github.com/chuxuzhang/KDD2019_HetGNN
[7]https://github.com/Jhy1993/HAN
[8]https://github.com/UCLA-DM/pyHGT
[9]https://github.com/CaoQi92/CoupledGNN

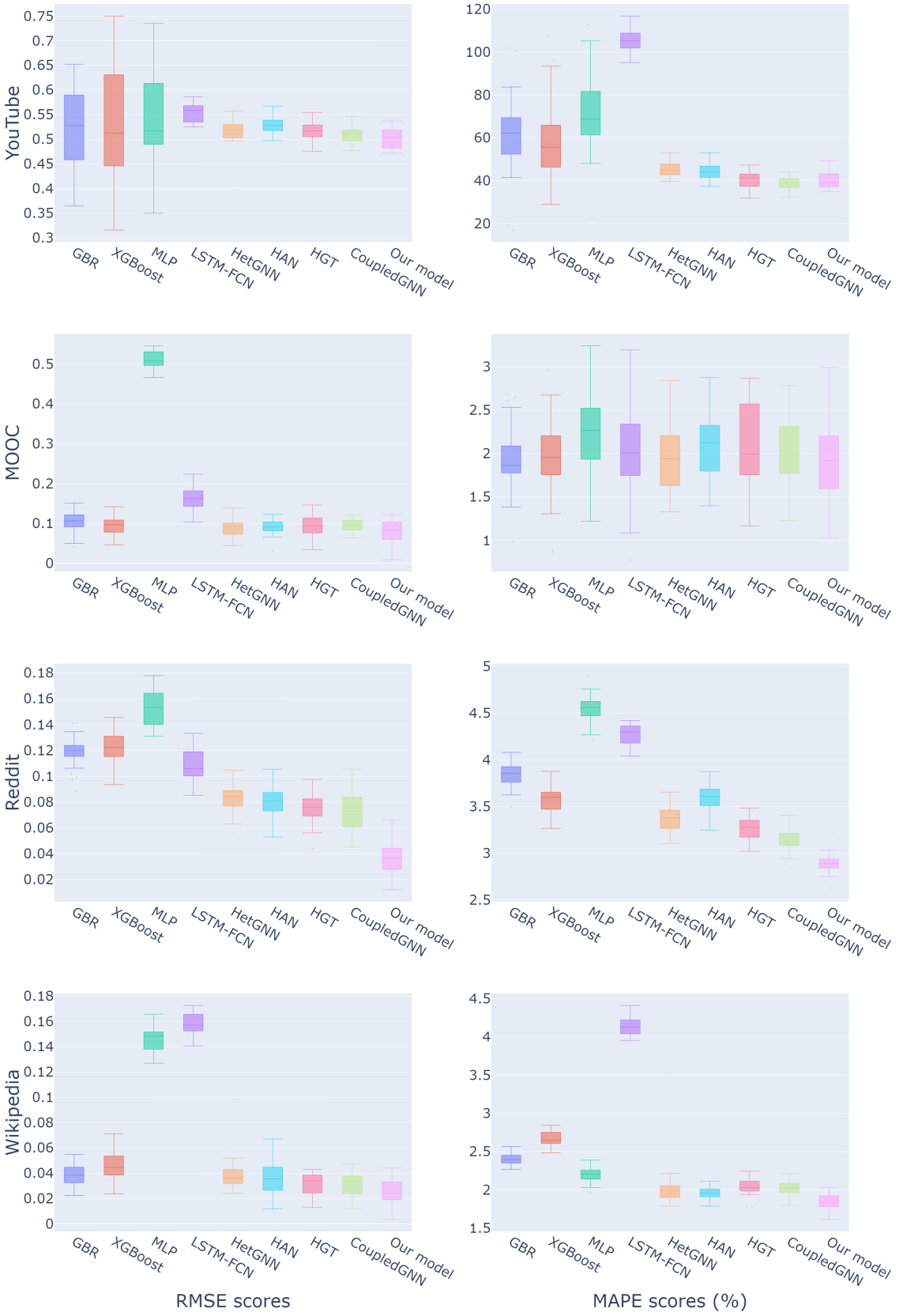


Figure 4.6: The box plot of the results. Four rows denote four datasets and two columns denote two metrics.

Table 4.3: The results of node attribute value prediction tasks on four datasets. Metric is RMSE scores and MAPE scores (%). Results are mean values of 30 runnings.

Model	YouTube		MOOC		Reddit		Wikipedia	
Metrics	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
Linear regression algorithm								
GBR	0.524	59.747	0.095	2.137	0.122	3.839	0.040	2.412
XGBoost	0.535	61.814	0.095	2.012	0.123	3.578	0.042	2.684
MLP	0.536	62.281	0.505	2.182	0.152	4.569	0.143	2.192
Time-sequence model								
LSTM-FCN	0.557	103.931	0.153	2.002	0.106	4.288	0.158	4.128
Static heterogeneous graph method								
HetGNN	0.518	44.325	0.092	1.968	0.082	3.403	0.036	1.992
HAN	0.524	44.933	0.092	1.971	0.082	3.602	0.036	1.933
Temporal heterogeneous graph method								
HGT	0.512	40.526	0.091	1.976	0.079	3.246	0.032	2.029
CoupledGNN	0.510	40.193	0.090	1.971	0.071	3.110	0.030	1.992
Our method	**0.507**	**39.652**	**0.080**	**1.960**	**0.036**	**2.839**	**0.027**	**1.847**

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=i}^{N} (y_i - \hat{y}_i)^2}{N}} \tag{4.23}$$

$$MAPE(y, \hat{y}) = \frac{1}{N} \sum_{i=i}^{N} \frac{\|y_i - \hat{y}_i\|}{\|y_i\|}, \tag{4.24}$$

which quantify the average difference between predicted values $\hat{y}$ and actual values y of total N samples. Smaller RMSE and MAPE indicate superior performance. The proposed method has the best overall performance, achieving the lowest RMSE and MAPE scores on all four datasets, as indicated by bold font in the table. The respective performance of the baselines is evaluated as follows:

- Linear regression algorithms (GBR and XGBoost) and the Neural Network (MLP) rely solely on the attribute vectors of items, which is often impractical for predicting trends.

- Time-sequence models (LSTM-FCN) take in sequences of attribute vectors and generate corresponding sequences of hidden state embeddings for each graph snapshot. However, these models are susceptible to input noise in the early steps, which can significantly impact subsequent outputs. Consequently, relying solely on long sequences of attribute vectors may not perform as well. In some cases, they may even underperform compared to linear regression algorithms that only use attribute vectors.

Figure 4.7: Study of parameter sensitivity.

- Static heterogeneous GNN methods (HetGNN and HAN) are good at analyzing graph structural information and edge interactions among different types of nodes. These methods outperform linear regression methods and time-sequence models. However, they are not designed to handle temporal data, a crucial aspect of popularity trend prediction.

- Temporal heterogeneous GNN methods (HGT and CoupledGNN) receive inputs similar to the proposed method. Their performance depends on their underlying architectures and ability to effectively capture temporal and graph structure information. The proposed method outperforms HGT and CoupledGNN, owing to distinctive architectural choices such as incorporating GSL and Multi-head Attention layer.

4.5.5 Parameter Sensitivity

In this section, an experiment is conducted to determine the optimal values for significant hyperparameters, namely the hyperparameter α in GSL, the balance hyperparameter λ in the loss function, and the dimension of hidden embeddings. These hyperparameters are fine-tuned through grid search within specific ranges.

In Fig. 4.7, the performance on four datasets is demonstrated concerning each hyperparameter, using the RMSE score as the metric. The rows represent the results

Table 4.4: The ablation study results without specific key components in the proposed method.

	YouTube	MOOC	Reddit	Wikipedia
w/o bi-graph	0.537	**0.100**	0.039	0.029
w/o intra-graph	0.537	0.090	0.039	0.029
w/o time-sequence unit	0.536	0.086	0.039	**0.031**
w/o GSL	**0.538**	0.095	0.039	0.029
Completed method	0.507	0.080	0.036	0.027

for four datasets, while the columns correspond to values of α ranging from 0.01 to 0.9, λ ranging from 0.01 to 1.0, and hidden dimensions varying from 16 to 256.

Based on the results, the proposed method is not highly parameter-sensitive. The relatively optimal hyperparameter choices are $\alpha = 0.1$, $\lambda = 0.1$, and a hidden dimension of 128. These settings are employed in the experimental results for the popularity trend prediction tasks in the previous evaluation section.

4.5.6 Ablation Study

An ablation study is conducted to assess the significance of key components in the proposed method, namely GSL, intra graphs, bipartite graphs, and the time-sequence unit. It is interesting to discover what component contributes the most to different application cases in experiments. Each of these components is responsible for specific types of input information:

- GSL enhances the quality of input graphs.
- Intra graphs represent intra-relations within target entities.
- Bipartite graphs depict relations between target entities and other types of entities.
- The time-sequence unit receives and propagates temporal information between adjacent graph snapshots.

In Table 4.4, the performance is demonstrated when each component is omitted, with the metric being the RMSE score. The results without specific components consistently underperform compared to the completed method, highlighting the indispensability of all components to the overall performance of the proposed method. The worst results in each dataset are marked in bold font. These results indicate that the corresponding component is the most significant for various datasets. To provide specific insights:

- In the YouTube dataset, the method without GSL exhibits the worst performance, highlighting the importance of improving input graph quality.

- In the MOOC dataset, the absence of bipartite graph input leads to the poorest result, emphasizing the significance of item-user relations.
- In the Reddit dataset, all results are equally achieved, suggesting that no single component stands out as the most crucial.
- In the Wikipedia dataset, the most crucial component is the time-sequence unit, highlighting the valuable information in the edit history.

It is reasonable that the most significant components vary across different datasets, as the most abundant information differs in each case. In summary, those findings emphasize that all components are essential, and their significance depends on the specific application scenarios.

4.5.7 Statistical Test

This section introduces a statistical test to determine the statistical significance of observed differences among baseline methods in Section 4.5.4. The test involves formulating a null hypothesis H_0 and an alternative hypothesis H_1 as follows:

Hypothesis 0. *All the methods have the same performance.*

Hypothesis 1. *The performance of methods has significant differences.*

Next, we employ two statistical tests, the Friedman test and the Wilcoxon sign-rank test, to assess the differences between the proposed method and baseline methods and validate the formulated hypotheses.

Friedman Test

The Friedman test, a non-parametric statistical method, is employed to identify significant differences in the performance of two or more methods across multiple test attempts [Friedman, 1937, 1940]. The Friedman test is the analog of the repeated measures Analysis of Variance (ANOVA) in non-parametric statistical procedures.

The initial step in calculating the Friedman test statistic involves converting the experimental results in Table 5.8 into ranks. Specifically, we assess the performance of nine methods (eight baseline methods and one proposed method) on four datasets (YouTube, MOOC, Reddit, and Wikipedia) by splitting the datasets into two subsequences: the former half of the graph snapshots and the latter half of the graph snapshots, resulting in a total of eight independent tests. The results are ranked for each test, ranging from 1 (indicating the best result) to 9 (representing the worst result). Subsequently, the average rank for each method is calculated based on all eight tests. Table 4.5 illustrates the ranks of the nine methods across the eight

Table 4.5: The ranks of nine methods assessed on eight tests. $T_1 \sim T_4$ represent RMSE scores on the former half of the YouTube, MOOC, Reddit, and Wikipedia datasets. $T_5 \sim T_8$ represent RMSE scores on the latter half of the YouTube, MOOC, Reddit, and Wikipedia datasets. Any tied ranks are assigned an average rank.

	Proposed method	GBR	XGBoost	MLP	LSTM-FCN	HetGNN	HAN	HGT	Coupled GNN
T_1	1	8	7	5.5	9	3	5.5	4	2
T_2	1	7	6	9	8	3	4	5	2
T_3	1	7	8	9	6	4	2	3	5
T_4	1	7	6	8	9	5	4	3	2
T_5	1.5	7	5	8	9	6	4	1.5	3
T_6	1	4	7	8	9	6	2.5	5	2.5
T_7	1	8	6	9	7	3.5	3.5	5	2
T_8	1	3	6	8	9	4.5	7	2	4.5
Avg. rank	1.0625	6.375	6.375	8.0625	8.25	4.375	4.0625	3.5625	2.875

tests, with tied ranks assigned an average value. By using these ranks, the Friedman statistic can be computed as follows:

$$\chi_f^2 = \frac{12n}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right], \tag{4.25}$$

which follows a χ^2 distribution with $k-1$ degrees of freedom. Here, n represents the number of tasks, k is the count of baseline methods included in the comparison, and R_j denotes the average rank of each method j.

Iman and Davenport propose a derivation from the Friedman statistic [Iman and Davenport, 1980] as follows:

$$F_{ID} = \frac{(n-1)\chi_f^2}{n(k-1) - \chi_f^2}, \tag{4.26}$$

which follows an F distribution with $k-1$ and $(k-1)(n-1)$ degrees of freedom.

Based on the formula above, the test statistic F_{ID} is calculated as 50.4795, and the corresponding p-value is 3.3049×10^{-8}. As the p-value is less than $\alpha = 0.05$, we reject the null hypothesis H_0, which indicates that all baseline methods exhibit the same performance. In other words, there is substantial evidence to support the presence of statistical significance among these methods.

Wilcoxon Sign-rank Test

While the Friedman test excels at detecting overall differences across multiple comparisons, its limitation lies in its inability to pinpoint significant differences within specific pairs of methods. To address this constraint, the Wilcoxon sign-rank test emerges as a non-parametric alternative to the paired t-test, particularly suited for

Table 4.6: The results of the Wilcoxon sign-rank test between pairs of the proposed method and each baseline method. Results are Benjamini–Hochberg FDR adjusted p-values.

Vs.	GBR	XGBoost	MLP	LSTM-FCN	HetGNN	HAN	HGT	Coupled GNN
Adjusted p-values	0.0175	0.0175	0.0175	0.0175	0.0175	0.0175	0.0323	0.0175

non-normally distributed samples in this work.

In the Wilcoxon sign-rank test, the differences between sample averages for all method pairs are calculated in multiple comparisons. The test is defined as follows:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{4.27}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i), \tag{4.28}$$

where d_i represents the difference between the performance scores of two methods on the ith out of n tests. R^+ is the sum of ranks for tests where the first algorithm outperformed the second, and R^- is the sum of ranks for the opposite. Ranks of $d_i = 0$, indicating ties, are evenly distributed between the sums. $T = \min(R^+, R^-)$ is the smaller of the sums. If T is less than the critical value from the Wilcoxon distribution for n degrees of freedom, the null hypothesis H_0 is rejected, signifying that the given method outperforms the other, with associated p-values.

In this study, we employ the Python library *scikit-posthocs*[10] to conduct the Wilcoxon sign-rank test and calculate p-values for each pairwise comparison between the proposed method and baseline methods. Similar to the Friedman test, this assessment involves ranking the performance of nine methods across eight tests.

To address the issue of inflated Type I error (family-wise error rate) in multiple comparisons, the test incorporates adjusted p-values using the Benjamini–Hochberg false discovery rate (FDR) method [Benjamini and Hochberg, 1995]. The results, including the calculated adjusted p-values, are presented in Table 4.6.

All adjusted p-values for the proposed method versus baseline methods are below the significance level of $\alpha = 0.05$, indicating statistical significance compared to these methods. It is noticeable that all the adjusted p-values except for the pair vs. HGT are all the same because the proposed method consistently achieves the first rank across all eight tests when comparing with those models, resulting in T always equaling 0. Only one different adjust p-value occurs when comparing with HGT, since the proposed method and HGT achieve the tied first place on T_5.

[10]https://scikit-posthocs.readthedocs.io/en/latest/index.html

4.6 Discussion and Conclusion

4.6.1 Discussion

In this section, we discuss the improvements achieved by comparing our method to the most recent and influential studies concerning GNNs in real-world prediction tasks. In contrast to other state-of-the-art heterogeneous GNNs, such as HetGNN [Zhang et al., 2019a], HAN [Wang et al., 2019a], HGT [Hu et al., 2020], and CoupledGNN [Cao et al., 2020], our method introduces a specialized multi-layer architecture. This architecture decomposes the intricate multiple relationship types into two categories: intra connections within entities of the same type and inter connections across entities of different types. This simplification process proves advantageous for handling heterogeneous graphs. Moreover, our method incorporates several modules to enhance overall performance, including the GSL module for improving the quality of input graphs and a multi-head attention layer for efficiently combining outputs from each graph layer.

In comparison to recent trend prediction studies unrelated to graphs, such as those in IoT [Mohamed et al., 2023], citation networks [Li et al., 2021a], terrorist attacks [Hou et al., 2023], and traffic flow [Yang et al., 2023], these methods often focus primarily on selecting characteristics and statistical features of target entities. They tend to overlook information diffusions along relationships and commonly rely on simplistic machine learning or statistical methods, resulting in suboptimal learning quality. To address these shortcomings, our method emphasizes the "message passing" along relationships within entities and employs advanced deep-learning-based encoders and decoders, leading to high learning efficiency.

In summary, our method addresses real-world trend prediction tasks and demonstrates remarkable improvements compared to the most recent and influential studies.

4.6.2 Conclusion

This work provides an effective solution for learning the representation of entities within social media networks and predicting trends in real-world scenarios. To achieve this, we develop an advanced multi-layer temporal GNN framework that captures information diffusion and temporal dynamics among different types of entities. The experimental results demonstrate the efficiency of the proposed method compared to various baseline methods, including basic linear regression approaches, time-sequence models, previous popularity trend prediction methods, and recently proposed heterogeneous GNN methods. Besides, massive experiments are conducted to explore the optimal choices of hyperparameters and assess the significance of key components in the proposed method.

In addition, this work makes a substantial contribution to the rapidly growing and promising field of trend prediction in social media. Social media is a primary source for obtaining information about emerging trends worldwide. By predicting trends in social media, the proposed method can assist users in gaining a deeper understanding of the future trajectories of public concerns and interests. Anticipating trends in advance holds significant value, as it enables individuals and organizations to stay ahead of the competition, align their studies and work in the right direction, and seize opportunities for informed decision-making in the future.

Limitations

Acknowledging the limitations inherent in our work is essential for a comprehensive understanding of its scope and applicability. We recognize that the proposed method is specifically tailored to accommodate input in the form of sequences of graph snapshots (DTDG). Consequently, it is essential to acknowledge that the method, as it stands, does not support other types of temporal graphs, such as sequences of graph actions accompanied by timestamps (CTDG). This limitation signifies a potential avenue for future research to expand the method's applicability to a broader range of temporal graph structures.

Moreover, it is crucial to emphasize that real-world situations are inherently dynamic and influenced by many factors. As with any predictive model, our method operates within the bounds of uncertainties and the potential existence of missing information. It is imperative to understand that accurately predicting the future in complex and dynamic scenarios is inherently challenging. Therefore, the results provided by our method should be considered as reference points rather than definitive conclusions.

Chapter 5

Predictive Analytics on Continuous-time Dynamic Graphs

5.1 Introduction

Chapter 4 introduces a multi-layer temporal GNN framework designed to learn node temporal representations within intricate heterogeneous discrete-time dynamic graphs (DTDG). However, a significant challenge arises in prediction tasks due to unknown information between consecutive graph snapshots, leading to potential inefficiencies in prediction results. To fill this gap, the current chapter presents a substantial enhancement: extending the proposed model to learn node representations within continuous-time dynamic graphs (CTDG). Given the inherent difficulties posed by learning on CTDG, this extension represents a considerable advancement where existing static graph methods are rendered ineffective. The research on learning within CTDG is characterized by its challenging nature and is still in its early stages.

This chapter commences by elucidating how real-world data can be represented in a computable CTDG format. Subsequently, we introduce a novel model termed the Temporal Difference Graph Neural Network (TDGNN), expressly designed to learn node temporal embeddings from CTDG graphs. Notably, our proposed model exhibits the capability to predict real-time graph events, answering questions such as when an edge will emerge between specific nodes or how frequently a particular node will change status in the next period. Moreover, the model addresses imbalanced situations commonly encountered in real-world scenarios, incorporating several imbalanced learning strategies to enhance learning in minority classes.

The feasibility in real-world situations lies in evaluating the TDGNN model, focusing on the challenging task of predicting real-time donations on live streaming services platforms. Extensive experiments conducted on three live-streaming video datasets demonstrate the efficacy and robustness of our proposed model. TDGNN

outperforms other baseline methods from various fields, providing more effective and precise predictions of both the donation posters and the exact times when donations will appear. This chapter signifies a substantial contribution to the nascent field of learning on continuous-time dynamic graphs, showcasing the potential of TDGNN in real-world applications.

This work has been published and orally presented at *25th International Conference on Discovery Science* [Jin et al., 2022], and an extension version has been published in the *Machine Learning* Journal [Jin et al., 2023].

Application: Predicting Real-time Donations in Online Live Streaming Service

The growth of the internet and the increasing use of mobile devices have led to a surge in the popularity of live streaming services. These services offer a wide range of content, including news, sports, entertainment, and video games [Yang and Lee, 2018], and provide several advantages over traditional television, such as a large variety of content, low costs, flexibility in viewing, personalized channels, and uninterrupted programming [Lee et al., 2016; Yang and Lee, 2018]. The COVID-19 pandemic has significantly impacted the live streaming industry, as many people have turned to online courses and remote work, resulting in an increase in the global market for online video streaming.

YouTube Live[1] is one of the most popular online live streaming platforms, with millions of user-generated content shared among billions of active users. Viewers on YouTube Live can communicate with others by sending real-time chat messages, which streamers can see and interact with in real time. These chat messages bring viewers and streamers closer together, creating a sense of community for viewers and making popular streamers into "celebrities" for their audiences [Fietkiewicz et al., 2018]. Additionally, some viewers are willing to donate money to their favorite streamers through the "superchat" donation system on YouTube Live. As shown in Fig. 5.1, superchat is a special chat message associated with an amount of money ranging from $1 to $500 that is highlighted in the live streaming chatbox for some time. This system brings profits, popularity, and motivation for streamers to create high-quality content. This situation raises some interesting questions: Who tends to send superchats? When are superchats sent? Can superchats be predicted? Understanding the answers to these questions will help streamers to predict their expected donation income better.

Research on live streaming services is still in its early stages. Scholars are studying various aspects of video content, such as improving video quality and using image recognition in live streaming. Additionally, live-streaming platforms provide

[1] https://www.youtube.com/

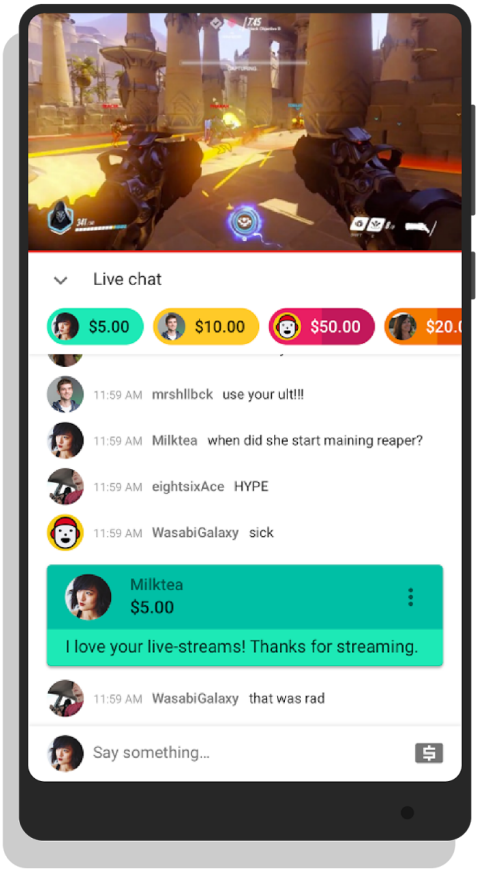


Figure 5.1: An example of a superchat on the YouTube Live platform is when a viewer named "Milktea" donates $5.00 to the streamer and posts a superchat message that reads, "I love your live-streams! Thanks for streaming", in a live streaming channel.

a valuable data source for chat data analysis. Some works focus on highlight detection [Chu and Chou, 2017], sentimental analysis [Kavitha et al., 2018], and fraud detection [Li et al., 2021b]. However, there is limited research on analyzing virtual donations in live streaming services. Current studies focus on static donation prediction without considering temporal information, using simple machine learning algorithms [Jia et al., 2021; Wang et al., 2019b], and are thus not practically applicable.

This work presents a novel approach to discovering when and who sends superchat messages in YouTube live streaming. We study real-time chat messages and interactions among viewers in live streaming. We find that chat messages increase significantly when a superchat is posted, as streamers appreciate the donation and other viewers respond to the superchat. Additionally, superchat messages are usually longer and better organized to attract attention. Therefore, a superchat message is usually followed by many response chat messages and has unique text content.

Drawing on these insights, we propose an approach that models the dynamic interactions among viewers and the text content of chat messages. Our method utilizes a continuous-time dynamic graph to capture the complex relationships among

thousands of viewers and millions of chat messages. By transforming the superchat detection problem into a dynamic node label classification problem in the graph, we are able to predict both the superchat messages and the timing of their posting on YouTube Live. To learn temporal node representations in the graph, we introduce a novel Temporal Difference Graph Neural Network (TDGNN) that exploits the information gap between connected nodes.

The experimental results demonstrate that our proposed approach is highly effective in predicting superchat messages with AUC scores up to 0.916, outperforming various baseline methods. These baseline methods include decision tree algorithms(e.g., GBDT, XGBoost), time sequence models (e.g., LSTM-FCN [Karim et al., 2018]), static graph-structured models (e.g., GCN [Kipf and Welling, 2016a], GAT [Veličković et al., 2018]), text classification models (e.g., BERT [Devlin et al., 2018]), and the latest temporal graph neural network (TGNN) models (e.g., TGN [Rossi et al., 2020], APAN [Wang et al., 2021], TGAT [Xu et al., 2020]), MetaDyGNN [Yang et al., 2022a].

In this paper, we make several important contributions to the field of donation prediction in live streaming platforms, including:

- We present the first analysis and prediction of real-time virtual donations in live streaming platforms. This is a significant and innovative topic as it has the potential to help live streaming services gain more popularity and increase profits.

- We propose an approach that represents live streaming chat messages and interactions among viewers as a continuous-time dynamic graph. We transform the superchat detection problem into a dynamic node label classification problem in the graph and introduce a TDGNN model that learns temporal node representations and predicts real-time superchat donations.

- Our TDGNN model has a real-time node label updating mechanism that identifies the precise timing of updating node labels. This is in contrast to traditional TGNNs, where node information is not updated until batches are finished. By solving the dynamic node label classification task, our approach can predict the exact time when a superchat appears.

- We conduct experiments to demonstrate the effectiveness of our proposed approach, which outperforms various baseline methods, including traditional machine learning algorithms, time sequence models, static graph-structured models, text classification models, and the latest TGNN models.

5.2 Related Work

Online Live Streaming Service

Online live streaming services have become increasingly popular with the advent of high-speed internet and mobile devices. These services have led to the formation of various live streaming communities, which are often centered around different content genres [Hamilton et al., 2014]. For instance, YouTube live streamers often share their daily life and experiences and are commonly referred to as "vloggers" [Ladhari et al., 2020]. In contrast, game streaming channels are more popular on the Twitch[2] platform, where viewers enjoy watching others play video games to relieve stress, pass time, and engage in common topics with friends [Sjöblom and Hamari, 2017].

The live streaming platform enables creators to monetize their live videos and generate revenue from live streaming. Streamers on most live streaming platforms, such as YouTube Live, Twitch, and TikTok, can generate revenue in various ways, such as receiving real-time gifts from fans, paid subscriptions from viewers, and ads. Ads revenue of a live streaming channel is generally stable and positively correlated to the number of viewers and followers, while the amount of real-time gifts and paid subscriptions can vary greatly depending on the popularity of the streamer and the generosity of their viewers. Paid subscriptions and real-time gifts are also referred to as virtual donations. Virtual donation in live streaming is a promising topic and has attracted the attention of many researchers. Current research examines the reasons behind virtual donations, such as how they represent viewers' appreciation and approval of the streamer or recognition and happiness for shared content [Lee et al., 2019]. Paid subscriptions enable viewers to make monthly donations to support their favorite channels on a recurring or one-time basis [Kim et al., 2019; Wohn et al., 2019; Yu and Jia, 2022]. Subscribers can gain access to some channel benefits, such as customized emotes and badges. Real-time gifts are special gift donations or chat messages associated with a particular amount of money. They are often accompanied by special effects and highlighted in the live streaming chatbox for a while [Jin et al., 2022; Zhan and Zhang, 2023].

To investigate the impact of virtual donation revenue in live streaming, we check the revenue data of streamers from two popular live streaming platforms: Twitch and YouTube Live. According to a Twitch channel ranking site[3], one of the most popular streamers "JYNXZI" on Twitch had $59,459$ paid subscriptions and $29,748$ gift donations in April 2023, generating more than $\$294,240$ USD revenue. In another hand, according to a YouTube channel ranking site[4], one of the most popular

[2]https://www.twitch.tv/

[3]https://twitchtracker.com/jynxzi/subscribers

[4]https://playboard.co/en/channel/UCLg4NCAJxhIvD4IRV_-LOFg

YouTube Live streamer "Pastor Jerry Eze" had 55,358 concurrent live streaming viewers in average and 26,757 superchat donations generating $183,108 USD in April 2023. Assuming that the number of paid subscribers is 10% of the concurrent viewers and the monthly fee of paid subscription (membership) is $5.00 USD, the revenue from paid subscription was $27,679 USD, and the total revenue from virtual donation was $210,787 USD in April 2023. The data above shows that both paid subscriptions and real-time gift donations are significant sources of revenue for streamers. Therefore, it is meaningful to conduct research on the revenue from real-time gift donations in live streaming, and YouTube Live is a suitable platform for this purpose. Some patterns of sending real-time superchat donations during live video streaming have been identified, such as only a small number of viewers sending a majority of the gifts, and viewers being motivated to send gifts after observing other viewers' gift-giving behaviors [Zhu et al., 2017]. Additionally, donation information is entirely public on live streaming channels, meaning that when viewers donate to the streamer, others will notice it. Other viewers tend to be affected by such noticeable actions and are likely to follow the group and send more donations [Payne et al., 2017]. Therefore, donations in online streaming services signal a group interaction and an event for viewers to interact with others. Some large-scale analyses of real-time virtual donations in on-demand video sites and online live streaming platforms have been carried out in several works [Jia et al., 2021; Lu Jia et al., 2019, 2020; Wang et al., 2019b]. However, those works only cover a short period of virtual donation activities and focus on revealing the static properties of viewers and streamers using naive machine learning algorithms.

Dynamic Graph Learning

Graph learning has produced many successful applications [Zhou et al., 2018]. The main challenge in graph learning is to find an appropriate method to encode the graph structure, including nodes and edges, into low-dimensional hidden embedding vectors while preserving the topology structure and node information. These embedding vectors can be utilized by machine learning models and deep learning architectures, such as random-walk-based algorithms [Perozzi et al., 2014b] and graph neural networks [Wu et al., 2019b]. Learning embedding vectors on graphs is widely recognized for graph-related downstream tasks, such as node classification [Kipf and Welling, 2016a], link prediction [Zhang and Chen, 2018], community detection [Interdonato et al., 2017], and graph classification [Zhang et al., 2018a].

Graph Neural Networks (GNNs) have emerged as a powerful deep learning framework for graph learning. Among various GNNs, Graph Convolutional Network (GCN) is one of the most widely used models that aggregates neighbor node information efficiently using a convolutional layer [Kipf and Welling, 2016a]. To further improve the GNN performance, many researchers have proposed advanced GNN

models based on the GCN structure. For example, GraphSAGE extends GCN into inductive learning that can handle unknown nodes in graphs [Hamilton et al., 2017b]. Graph Attention Networks employ attention mechanisms that assign importance weights to different neighbor nodes [Veličković et al., 2018]. Graph Autoencoder and Variational Graph Autoencoder are GCN-based encoder-decoder models that can handle unsupervised learning tasks [Kipf and Welling, 2016b]. Moreover, ML-GCN and ML-GAT extend the original GCN and GAT to multilayer networks, which can capture more complex relations among nodes in large-scale graphs [Zangari et al., 2021].

Learning on dynamic graphs is much more complex than on static graphs. Initially, research on dynamic graphs focused on discrete-time dynamic graphs, which consist of a timed sequence of graph snapshots [Gao et al., 2022a; Liben-Nowell and Kleinberg, 2007; Sankar et al., 2020]. Existing static graph methods can be directly applied to each graph snapshot. However, most real-life graph-structured data is in a state of constant evolution. A more general style of dynamic graph is the continuous-time dynamic graph, which consists of a timed list of events, including edge creation or deletion, node creation or deletion, and node or edge status evolution. Recently, several studies on continuous-time dynamic graphs have been proposed, including JODIE [Kumar et al., 2019], Continuous-time Dynamic Network Embedding [Nguyen et al., 2018], DyRep [Trivedi et al., 2019], Temporal Graph Networks (TGN) [Rossi et al., 2020], Temporal Graph Attention [Xu et al., 2020], Asynchronous Propagation Attention Network (APAN) [Wang et al., 2021], and Meta-learning framework MetaDyGNN [Yang et al., 2022a]. However, these continuous-time dynamic graph methods need two-step model training processes that require high data volume and long training time, causing low training efficiency. Additionally, they lack accuracy in predicting the exact timing of superchat messages because they have an update delay in model training and inference, rendering them impractical for our research target. A new approach is needed to address these issues and predict the post time of superchat messages more accurately and efficiently.

5.3 Problem Setup

5.3.1 Dataset Description

We use a publicly available YouTube live streaming dataset in this study: *VTuber 1B: Large-scale Live Chat and Moderation Events Dataset* [5]. VTuber 1B is a massive collection of over a billion live chat messages, superchats, and moderation

[5]The dataset can be downloaded from `https://www.kaggle.com/datasets/uetchy/vtuber-livechat`.

Table 5.1: Superchat purchase details

Purchase amount(USD)	Color	Max. message length	Max. time in the chatbox
$ 1.00 - 1.99	Blue	0 characters	0 seconds
$ 2.00 - 4.99	Light blue	50 characters	0 seconds
$ 5.00 - 9.99	Green	150 characters	2 minutes
$ 10.00 - 19.99	Yellow	200 characters	5 minutes
$ 20.00 - 49.99	Orange	225 characters	10 minutes
$ 50.00 - 99.99	Magenta	250 characters	30 minutes
$ 100.00 - 199.99	Red	270 characters	1 hour
$ 200.00 - 299.99	Red	290 characters	2 hours
$ 300.00 - 399.99	Red	310 characters	3 hours
$ 400.00 - 499.99	Red	330 characters	4 hours
$ 500.0	Red	350 characters	5 hours

events (ban and deletion) across hundreds of YouTubers' live streams, especially for English and Japanese Streamers. Our research uses the chat message data from Mar. 2021 to Apr. 2021, including 377 live streaming channels, 5,684 streaming videos, and over 58 million live chat messages. Over 54,000 viewers posted 230,025 superchat messages. Each channel has 15.07 live-streaming videos that last for 8.72 hours on average. There are 172.76 viewers, 10,245.84 chat messages posted, and 9.66 superchat donations on average in each video. The total purchase amount of superchat exceeds 3.4 million USD.

The dataset contains a considerable amount of superchat donations, which are categorized into multiple levels based on their purchase amount. Each level is represented by a different color, as shown in Table 5.1. Higher purchase amounts allow for longer message length and longer highlighting time in the chatbox. In our research, we classify all the chat messages into significance levels regarding its donation amount, indicating whether they are superchat messages or not. These labels will be utilized as prediction targets in our proposed framework.

The dataset used in this study is based on a cluster system[6] that was specifically designed to collect data from certain YouTube channels' live streams via YouTube Live streaming API. All personal information that could identify individual users, such as usernames and profile images, has been removed from the dataset to protect their privacy. User IDs and channel IDs have also been anonymized using the SHA-1 hashing algorithm and an undisclosed salt, further ensuring user anonymity. The *VTuber 1B* dataset has been used in various research studies, including toxic chat classification, spam detection, demographic visualization, superchat analysis, and training neural language models. With over a billion live chat messages, superchats, and moderation events, this dataset is a valuable resource for researchers interested in studying online behavior and language use. By utilizing this dataset, we can

[6]https://github.com/sigvt/honeybee

Table 5.2: Detailed dataset statistics

Description	Value
Start time	2021-03-15 23:19:38
End time	2021-04-11 15:15:36
# Live streaming channels	377
# Live streaming videos	5,684
# Live streaming viewers	981,996
# Chat messages	58,237,423
# Superchat messages	230,025
# Superchat donors	54,918
# Videos per channel	15.07
Duration (hrs.) per video	8.72
# Viewers per video	172.76
# Chat messages per video	10245.85
# Superchat messages per video	9.66
Total amount of super chats (USD)	$3,466,216.61

gain a better understanding of how users interact with each other in live streaming chatrooms and develop more effective tools for moderating and managing online conversations.

The detailed statistics are listed in Table 5.2.

5.3.2 Research Problem

In YouTube Live, viewers communicate with others by sending real-time chat messages, which streamers can see and interact with in real-time. In this way, these chat messages bring viewers and streamers closer together. Some of the chat messages are superchat with donations. The chat messages dataset presented above raises interesting questions: Who sends superchat messages? When are superchat messages typically sent? Can we predict superchat messages? These questions can give streamers valuable insights into their core fans and expected donation income. Predicting superchat messages can be helpful for content creators, moderators, and platform administrators to identify and engage with their most valuable viewers, leading to a more engaged and loyal audience. Our research task is to predict when and who sends superchat messages, i.e., we hope to identify the donor as soon as he/she sends a superchat.

We mathematically formalize the research problem shown in Fig. 5.2 as follows: Consider we have a history sequence of chat messages for the past period [0, T], i.e., $\mathbf{M} = \{\mathbf{msg}_i(t) | 0 \leq t \leq T\}$, where $\mathbf{msg}_i(t)$ include the textual content, the information about the viewer i who sent the chat message, and the correspond-

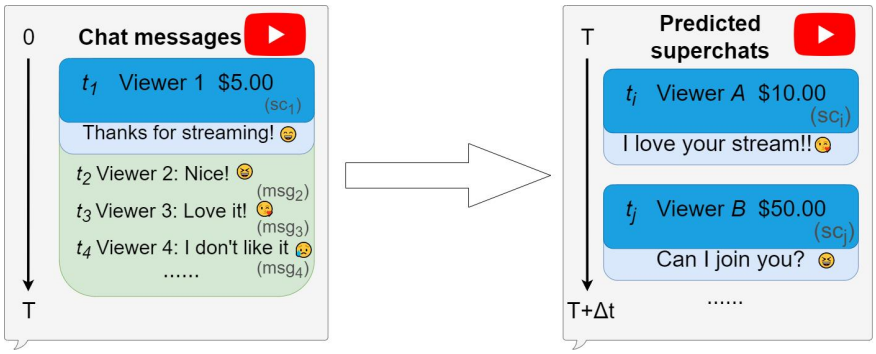


Figure 5.2: Problem description: Our objective is to predict the occurrence of superchat messages and their appearance timing in a given chat message stream. Given the timed sequence of chat messages during a certain period $0 \leq t \leq T$, our method aims to predict the viewers i who send superchat message $\mathbf{msg}_i(t)$ after T and the corresponding time t.

ing timestamp t. We also know which chat messages are superchats in $\mathbf{M}$, i.e., $\mathbf{D} = \{label(\mathbf{msg}_i(t))|0 \leq t \leq T\}$. Here $label(\mathbf{msg}_i(t))$ is a label function where $label(\mathbf{msg}_i(t)) = 1$ means $\mathbf{msg}_i(t)$ is a superchat message and $label(\mathbf{msg}_i(t)) = 0$ otherwise. For the prediction period $[T, T + \Delta T]$, we are given a sequence of chat messages in a stream fashion, i.e., $\mathbf{M}^* = \{\mathbf{msg}_i(t)|T \leq t \leq T + \Delta T\}$. Our aim is to learn the message label $\mathbf{D}^* = \{label(\mathbf{msg}_i(t))|T \leq t \leq T + \Delta T\}$ as soon as the message $\mathbf{msg}_i(t) \in \mathbf{M}^*$ was sent. The message labels indicate the information about viewers who send superchat messages and the timestamp when superchat messages are posted.

5.4 Constructing Dynamic Graphs

In this section, we describe our proposed method for identifying real-time superchat donations in YouTube live streaming services. We have empirically observed that superchat messages have specific characteristics that distinguish them from regular chat messages. For instance, when a superchat message appears, chat messages tend to increase as the streamer expresses their gratitude for the donation, and other viewers respond. Moreover, superchat messages are often longer and crafted in a way to attract attention. Consequently, superchat messages tend to elicit a high number of response chat messages and have distinctive textual content. To leverage these observations, we use text content information and viewer interactions

to identify superchat donations and its donor.

We utilize a continuous-time dynamic graph to capture the intricate relationships among a large number of viewers and chat messages, which number in the millions. The graph comprises batches of graph actions over time, such as node creation/deletion, edge creation/deletion, and node/edge status evolution. Nodes represent viewers, and edges represent interactions between them, i.e., whether two viewers have sent similar chat messages. Each node is assigned A dynamic node label, which changes with time. The label represents whether the viewer sends a superchat in a time window, i.e., if a viewer sends a superchat, its label temporarily changes from 0 to 1 until the time window ends. Thus, we transform the superchat message prediction problem into a dynamic node label classification problem in the continuous-time dynamic graph.

As previously mentioned, the prediction of superchat messages is based on two aspects: the textual content of chat messages, and the interactions between viewers. To achieve this, we convert the textual content of chat messages into sentence embedding vectors. Also, we measure the interaction between viewers by the similarity in meaning between the latest chat messages they post, i.e., two viewers interact with each other by sending similar chat messages. Specifically, we calculate the cosine similarity of two sentence embedding vectors of their most recently posted chat messages.

We use a pre-trained Sentence Transformers[7] [Reimers and Gurevych, 2019, 2020] language model to encode all the chat message texts into sentence embedding vectors. Sentence Transformers is a Python framework for generating state-of-the-art sentence, text, and image embeddings. It is commonly used in research tasks such as semantic textual similarity, semantic search, and paraphrase mining. Sentence Transformers has been extensively evaluated for its quality in embedding sentences (Performance Sentence Embeddings) and embedding search queries & paragraphs (Performance Semantic Search). We use a pre-trained multi-lingual model named *paraphrase-xlm-r-multilingual-v1*[8] to encode the chat messages into sentence embedding vectors. This model generates aligned vector spaces, meaning that similar inputs in different languages are mapped close together in the vector space.

Figure 5.3 illustrates a test of the pre-trained model's performance on sentence similarity. The embedding vectors of sentences with similar meanings have a higher sentence similarity score than those with opposite meanings. Moreover, the pre-trained model can identify the meanings of two sentences written in different languages or even in emojis. This test demonstrated that the model is capable of extracting meaningful information from short chat messages and encoding them into sentence embedding vectors while preserving their meaning.

[7]https://www.sbert.net/

[8]https://huggingface.co/sentence-transformers/paraphrase-xlm-r-multilingual-v1

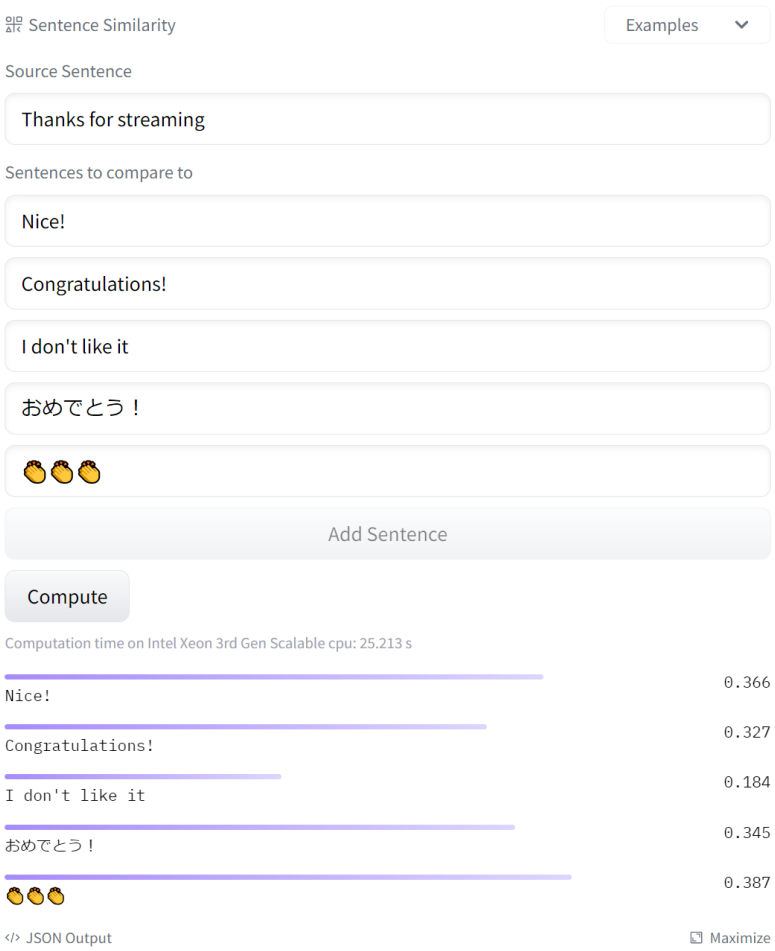


Figure 5.3: We conducted a test on the pre-trained Sentence Transformers model to measure sentence similarity. We used the source sentence "Thanks for streaming" and compared it with different target sentences. Sentences with positive and similar meanings, such as "Nice!" and "Congratulations!" received higher similarity scores of 0.366 and 0.327, respectively, whereas the sentence with a negative and opposite meaning, "I don't like it", had the lowest score of 0.184. Additionally, the model could identify the meanings of sentences written in different languages or emojis. For instance, the Japanese sentence "Congratulations!" and the emoji "Clap hands" received almost identical similarity scores (0.345 and 0.387, respectively) as their English equivalents.

With the sentence embedding vectors, we construct a continuous-time dynamic graph that encodes viewers' chat messages and interactions in live streaming videos. The graph comprises batches of graph actions over time, such as node creation/deletion, edge creation/deletion, and node/edge status evolution. Nodes represent viewers, and edges represent interactions between them, i.e., whether two viewers send similar chat messages. Node creation/deletion occurs when viewers enter or leave the streaming channel, while edge changes occur when viewers send new chat messages and interact with others. Additionally, each node is associated with a node feature vector, which is dynamically updated based on the sentence embedding vector of the latest chat message posted by the corresponding viewer. The edge weight repre-

Algorithm 5.1: Construct dynamic graph from a timed sequence of live streaming chat messages

Data: A timed sequence $\mathbf{M}$ of chat message. The time window Δt for separating batches. The threshold θ_1 for judging duplicated chat messages. The threshold θ_2 for generating edges by cosine similarity.

Result: A continuous-time dynamic graph $\mathbf{G} = (\mathbf{V}(t), \mathbf{E}(t), \mathbf{F}(t) \| t \in \mathbf{T})$, including node set $\mathbf{V}(t)$, edge set $\mathbf{E}(t)$, node feature set $\mathbf{F}(t)$ and the timestamp set of graph actions $\mathbf{T}$

initialization;
Separate $\mathbf{M}$ into several *batches* by time window Δt;
for *batch* $\leftarrow$ *batches* **do**
    for $\mathbf{msg}_i(t_1), \mathbf{msg}_j(t_2) \in \mathbf{M}[batch]$ *and* $t_1 < t_2$ `// t denotes the timestamp`
    do
        if *SequenceMatcher(*$\mathbf{msg}_i(t_1)$, $\mathbf{msg}_j(t_2)$*)* $> \theta_1$ **then**
            delete $\mathbf{msg}_j(t_2)$ `// Drop duplicated chat messages`
        end
    end
    for $\mathbf{msg}_i(t) \leftarrow \mathbf{M}[batch]$ **do**
        if *The viewer* $\mathbf{u}_i(t)$ *who posts* $\mathbf{msg}_i(t)$ *and* $\mathbf{u}_i(t) \notin \mathbf{V}(t)$ **then**
            Create a new node $\mathbf{u}_i(t) \in \mathbf{V}(t)$
        end
        Update the node feature $\mathbf{f}_i(t) \in \mathbf{F}(t)$ by the sentence embedding vector of $\mathbf{msg}_i(t)$;
        for $\mathbf{u}_j(t^-) \leftarrow$ *active_list* **do**
            if *cosine_similarity(*$\mathbf{u}_j(t^-)$, $\mathbf{u}_i(t)$*)*$> \theta_2$ **then**
                Generate an directed edge $\mathbf{e}_{ji}(t) \in \mathbf{E}(t)$ from node $\mathbf{u}_j(t^-)$ to $\mathbf{u}_i(t)$;
                Edge weight $\mathbf{e}_{ji}(t) = cosine\_similarity(\mathbf{u}_j(t^-), \mathbf{u}_i(t))$
            end
        end
        $\mathbf{u}_i(t) \rightarrow$ *active_list* `// Keep` $\mathbf{u}_t$ `active for a time window`
        for $\mathbf{u}_j(t^-)$ *in active_list* **do**
            if $(t - t^-) > \Delta t$ **then**
                delete $\mathbf{u}_j(t^-)$ from *active_list* `// Drop expired nodes`
            end
        end
    end
end

sents the cosine similarity of the node feature vectors of the two nodes at either end, and synchronously changes everytime node feature vectors are updated. The edge direction is always from the formerly updated node to the newly updated nodes. This ensures that new chat messages are influenced by old ones, and information propagates from old chat messages to new ones.

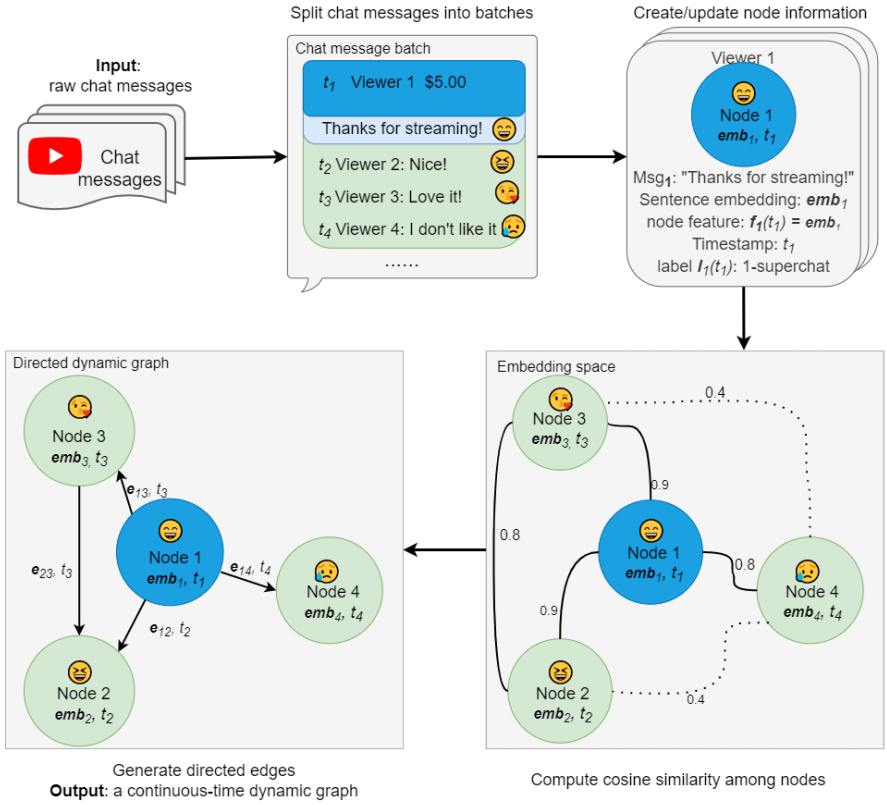


Figure 5.4: The diagram illustrates the process of constructing dynamic graphs from live streaming chat messages. In this example, four viewers post chat messages at timestamps t_1 to t_4 in the same batch, with the first message being a superchat. We create nodes 1 to 4 for each viewer and initialize their feature vectors using the sentence embedding vector of the chat message they posted. Next, we compute the cosine similarity between each pair of nodes and generate edges for node pairs with high similarity. The blue node represents a viewer who sent a superchat, while the green nodes represent viewers who only sent regular chat messages.

Algorithm 5.1 outlines how to construct a dynamic graph from a timed sequence of chat messages by utilizing their textual content, sentence embedding vectors, viewer ID, timestamps, and the dynamic label indicating whether it is a superchat message. The algorithm proceeds as follows:

1. The first step is to preprocess the raw chat messages. All the chat messages $\mathbf{M}$ are separated into several batches by a time window Δt, and any duplicated, nonsensical, or too-short messages are filtered out. The $SequenceMatcher()$ method from the Python library *difflib*[9] is used to check for duplicated messages. This preprocessing step helps to ensure that the subsequent graph construction is based on a clean and manageable sequence of chat messages. (lines 1-9)

2. Traverse all the chat messages $\mathbf{msg}_i(t)$ in the batch. And for each $\mathbf{msg}_i(t)$,

[9]https://docs.python.org/3/library/difflib.html

check if the viewer i who posted it already has a node $\mathbf{u}_i(t)$ in the graph G. If not, create a new node $\mathbf{u}_i(t) \in \mathbf{V}(t)$ for the viewer i. (lines 10-13)

3. Update the node feature vector $\mathbf{f}_i(t) \in \mathbf{F}(t)$ of the node $\mathbf{u}_i(t)$ at time t by the sentence embedding vector of $\mathbf{msg}_i(t)$. (line 14)

4. Compute the cosine similarity between the newly-updated node $\mathbf{u}_i(t)$ and each node $\mathbf{u}_j(t^-)$ in the active list. Node $\mathbf{u}_j(t^-)$ is updated earlier than node $\mathbf{u}_i(t)$ ($t^- < t$). If the cosine similarity of node $\mathbf{u}_i(t)$ and $\mathbf{u}_i(t)$ is greater than the threshold θ_2, a directed temporal edge $\mathbf{e}_{ji}(t) \in \mathbf{E}(t)$ is generated between the two nodes. The edge direction is from the previously updated node $\mathbf{u}_j(t^-)$ to the newly updated node $\mathbf{u}_i(t)$. The edge weight is equal to the value of cosine similarity between the two end nodes. (lines 15-20)

5. Add the newly updated node $\mathbf{u}_i(t)$ into the active list and keep active for a time window Δt. (line 21)

6. Traverse all the nodes $\mathbf{u}_j(t^-)$ in the active list and check their timestamps t^-. The node expiring time is set equal to the time window Δt. If the interval between the current timestamp t and t^- is larger than Δt, it means that the node $\mathbf{u}_j(t^-)$ is expired and will be removed from the active list. (lines 22-26)

7. Repeat step 2 to 6 until all the graph actions are visited. (line 27)

We assign a binary dynamic node label $l_i(t) \in \{0, 1\}$ to each node in the dynamic graph. The dynamic node label, which changes with time, represents whether the viewer $\mathbf{u}_i(t)$ has sent a superchat message $label(\mathbf{msg}_i(t)) = 1$ in the past time window Δt from timestamp t:

$$l_i(t) = max(\{label(\mathbf{msg}_i(t^-))|t - \Delta t < t^- < t\}). \tag{5.1}$$

Label $l_i(t) = 0$ indicates that the node $\mathbf{u}_i(t)$ did not send any superchat at time t, while label $l_i(t) = 1$ indicates that the node $\mathbf{u}_i(t)$ sent a superchat at time t. More precisely, if a viewer sends a superchat, its label temporarily changes from 0 to 1 until the time window ends. We hope to track the node label changes in a relatively short time window. Thus we could identify the viewer as soon as he/she sends a superchat message.

Fig. 5.4 is an intuitive description of constructing a dynamic graph from chat messages. The input is a batch raw chat messages. Four viewers post chat messages $\mathbf{msg}_1$ to $\mathbf{msg}_4$ at timestamp t_1 to t_4, with the first message being a superchat. Four chat messages appear in the same batch, and the chat messages are encoded to sentence embedding vectors $\mathbf{emb}_1$ to $\mathbf{emb}_4$. Node 1 to 4 are created to represent the corresponding viewers. The node features are initialized by $\mathbf{emb}_1$ to $\mathbf{emb}_4$.

A dynamic node label is associated with each node to identify the superchat and normal messages. The blue node represents a viewer who sent a superchat, while the green nodes represent viewers who only sent normal chat messages. Then we calculate the cosine similarity and generate edges for the node pairs in the active node list. Node 2 and node 3 have similar opinions toward node 1, while node 4 has the opposite. Therefore, edge $\mathbf{e}_{12}$ from node 1 to node 2, edge $\mathbf{e}_{23}$ from node 2 to node 3, and edge $\mathbf{e}_{13}$ from node 1 to node 3 are generated. Node 4 responds to node 1 but does not positively correlate to node 2 and node 3. Thus, only an edge $\mathbf{e}_{14}$ from node 1 to node 4 is generated.

5.5 Limitations of Traditional Temporal GCN

In this section, we discuss the limitations of traditional TGNNs and their potential shortcomings when applied to predicting YouTube live superchats.

TGNNs are designed for learning temporal node embeddings in dynamic graphs. Many existing TGNNs update the temporal node embedding by indiscriminate neighbor aggregation and timestamp information [Rossi et al., 2020; Wang et al., 2021; Xu et al., 2020]. Moreover, once the model is successfully trained, it can be flexibly modified for different downstream tasks.

However, there are limitations to existing TGNNs that need to be addressed:

- Existing TGNNs require a two-step training process: First, training the model parameters, and then training the decoders for downstream tasks. This approach requires high data volume and results in low training efficiency.

- The existing TGNNs focus on indiscriminate neighbor aggregation, which updates the central node embedding by collecting information from neighboring nodes. However, predicting node status in the next time window requires knowing in advance the direction and rate of information propagation. This means that more information is needed beyond just collecting information from neighboring nodes.

- In the specific task of predicting superchat messages, it is necessary to predict the exact time when these messages are posted. However, existing TGNNs only update node labels after training batches are finished, resulting in a update time delay in the node label prediction task.

The prediction of superchat messages and their post time is a challenging task, and existing TGNNs need to be more efficient and accurate in this regard. A new approach is needed to tackle the problem of predicting dynamic node labels in continuous-time dynamic graphs more intelligently.

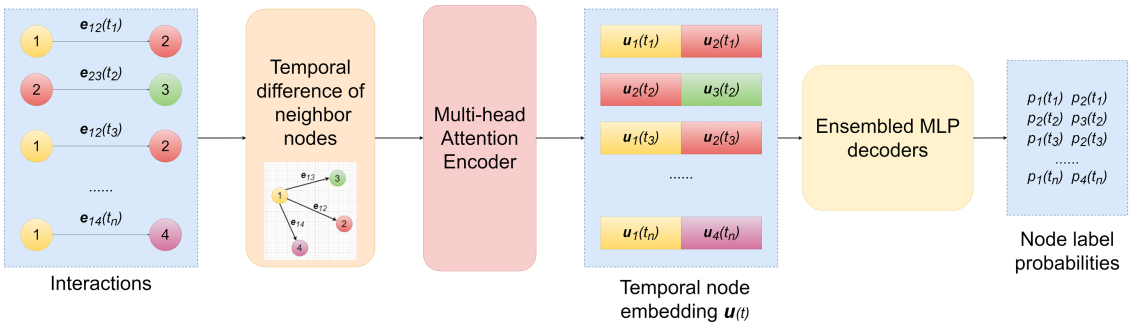


Figure 5.5: Overview of the proposed Temporal Difference Graph Neural Network (TDGNN).

5.6 Temporal Difference Graph Neural Network (TDGNN)

To address the gaps mentioned above, we introduce an end-to-end, meticulously designed TDGNN model that predicts dynamic node labels in continuous-time dynamic graphs. The primary objective of TDGNN is to learn temporal node embedding vectors that encapsulate node features and neighborhood node information. The term *temporal difference* here implies that TDGNN's focus is on the difference between the node and its adjacent node embeddings over time. The temporal difference between two adjacent nodes represents the "gradient", indicating the direction and rate of information propagation. When a graph event occurs, TDGNN calculates the amount of information change on adjacent nodes based on the product of temporal difference and updated node features.

The generated dynamic graph is denoted as $\mathbf{G} = (\mathbf{V}(t), \mathbf{E}(t), \mathbf{F}(t) \mid t \in \mathbf{T})$, where $t \in \mathbf{T}$ represents the timestamp. $\mathbf{u}_i(t) \in \mathbf{V}(t)$ and $\mathbf{u}_j(t) \in \mathbf{V}(t)$ represent the temporal nodes at timestamp t, while $\mathbf{e}_{ij}(t) \in \mathbf{E}(t) \subset \mathbf{V}(t) \times \mathbf{V}(t)$ is the directed temporal edge from node $\mathbf{u}_i(t)$ to node $\mathbf{u}_j(t)$ at timestamp t. The temporal edge weight is computed as the cosine similarity of two node embedding vectors. Each node $\mathbf{u}_i(t) \in \mathbf{V}(t)$ is associated with a node feature $\mathbf{f}_i(t) \in \mathbf{F}(t)$ and a temporal node label $l_i(t) \in \mathbf{L}(t)$. $\mathbf{L}(t)$ is the set of dynamic node labels for all graph actions in $\mathbf{G}$, and graph actions update the node feature $\mathbf{f}_i(t)$ and node labels $l_i(t)$ continuously.

We split timestamps $\mathbf{T}$ into three sets: a training set $\mathbf{T}^{train}$, a validation set $\mathbf{T}^{val}$, and a test set $\mathbf{T}^{test}$. Our research objective can be formalized as follows: Given a period of continuous-time dynamic graph $\mathbf{G} = (\mathbf{V}(t), \mathbf{E}(t), \mathbf{F}(t) \mid t \in \mathbf{T}^{train} \cup \mathbf{T}^{val})$ and a portion of known dynamic node labels $(\mathbf{L}(t) \mid t \in \mathbf{T}^{train} \cup \mathbf{T}^{val})$, we aim to learn a mapping function $\mathcal{F} : \mathbf{G} = (\mathbf{V}(t), \mathbf{E}(t), \mathbf{F}(t) \mid t \in \mathbf{T}^{test}) \rightarrow (\mathbf{L}(t) \mid t \in \mathbf{T}^{test})$ to predict the remaining dynamic node labels $(\mathbf{L}(t) \mid t \in \mathbf{T}^{test})$.

The proposed TDGNN framework is illustrated in Fig. 5.5. It consists of three main components: a temporal difference module, a multi-head attention encoder, and an ensembled Multi-Layer Perceptron (MLP) decoder. The temporal difference

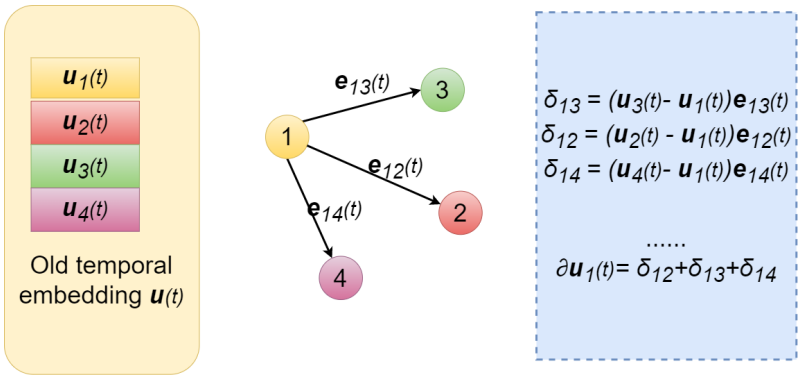


Figure 5.6: Compute the temporal difference in a directed graph. Temporal difference $\partial \mathbf{u}$ represents the information difference between node $\mathbf{u}$ and the surrounding neighbor nodes, indicating the amount of information propagation in dynamic graphs.

module captures the difference between the node and adjacent node embeddings over time. After the graph interactions occur, the temporal difference and the updated node features are passed to the multi-head attention encoder to update the node embeddings. The updated node embeddings are then fed to the ensemble MLP decoder to compute the probabilities of predicted node labels. This work addresses the real-time dynamic node label prediction task, where node labels change over time, in contrast to traditional node label classification, where node labels are static and constant.

The detailed steps are introduced as follows:

5.6.1 Temporal Difference Aggregation

We define the temporal difference of node $\mathbf{u}_i$ at time t as:

$$\partial \mathbf{u}_i(t) = \sum_{n \in \mathcal{N}_i} (\mathbf{u}_n(t) - \mathbf{u}_i(t)) \cdot \mathbf{e}_{ni}(t), \tag{5.2}$$

where $\mathcal{N}_i = \{\mathbf{u}_n(t) \mid \mathbf{e}_{ni}(t) \in \mathbf{E}\}$ denotes the set of neighbor nodes of $\mathbf{u}_i(t)$, and $\mathbf{e}_{ni}(t)$ denotes the edge weight.

The concept of temporal difference is crucial in understanding the direction and rate of information propagation in a directed graph. In the case of our proposed model, the node embedding difference serves as the "gradient" of information propagation from all neighboring nodes at time t, while the edge weight indicates the strength of the connection. In essence, $\partial \mathbf{u}_i(t)$ represents the total amount of information propagated on node $\mathbf{u}_i$ from all its neighbor nodes at time t. To illustrate this point, consider the example shown in Fig. 5.6, where node $\mathbf{u}_1$ passes information to its three connected neighbors, namely nodes $\mathbf{u}_2$, $\mathbf{u}_3$, and $\mathbf{u}_4$. In this scenario, $\partial \mathbf{u}_1$ is the amount of information that is passed on to these nodes.

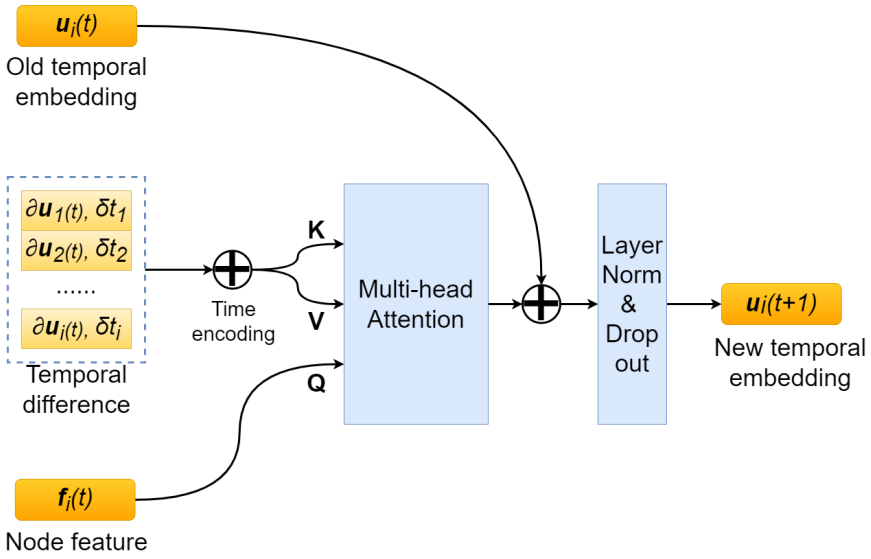


Figure 5.7: A multi-head attention module that calculates the new temporal node embedding $\mathbf{u}_i(t+1)$ according to the relativity between the last updated embedding $\mathbf{u}_i(t)$, temporal difference of neighbor nodes, and newly-updated node features.

5.6.2 Attention encoder

The TDGNN model updates node embeddings by utilizing both temporal difference aggregation and updated node features. Inspired by the Fundamental theorem of calculus, temporal difference acts as a gradient' in node information propagation, while the updated node feature plays the role of step size'. The TDGNN calculates the product of temporal difference and the updated node features, which represents the total amount of information change. It then adds the result to the old node embeddings to compute the new node embeddings. The complete procedure is shown in the following equation:

$$\mathbf{u}_i(t+1) = \mathbf{u}_i(t) + \phi\left(\partial\mathbf{u}_i(t), \mathbf{f}_i(t)\right), \tag{5.3}$$

where ϕ represents a function that computes the change in the value of updated node feature $\mathbf{f}_i(t)$ regarding the node $\mathbf{u}_i$. The multi-head attention module is an efficient mechanism to combine the temporal difference and node features. The attention layer works by computing the dot product of a query vector with a set of key vectors, resulting in a weight vector that assigns importance scores to the values. The mechanism is defined as follows:

$$Attn\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right)=softmax\left(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{d}}\right)\mathbf{V} \tag{5.4}$$

$$\mathbf{Q}=\left[\mathbf{f}_i(t)\|\delta t_s\right]\mathbf{W}_Q \tag{5.5}$$

$$\mathbf{K}=\left[\partial\mathbf{u}_i(t)\|\delta t_s\right]\mathbf{W}_K,\mathbf{V}=\left[\partial\mathbf{u}_i(t)\|\delta t_s\right]\mathbf{W}_V \tag{5.6}$$

$$head_i=Attn\left(\mathbf{Q}_i,\mathbf{K}_i,\mathbf{V}_i\right) \tag{5.7}$$

$$MultiHead\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right)=[head_1\|head_2\|\cdots]\mathbf{W}, \tag{5.8}$$

where δt_s denotes the time interval since last update, $[\cdot\|\cdot]$ represents the concatenation of matrices, and $\mathbf{W}_Q,\mathbf{W}_K,\mathbf{W}_V,\mathbf{W}_K$ are training parameters. As shown in Fig. 5.7, the node feature vector is fed into the matrix $\mathbf{Q}$, while the temporal difference vector is fed into matrices $\mathbf{K}$ and $\mathbf{V}$. To incorporate the time order of node updates, a time encoder is used to encode the time information into vectors that are also fed into the multi-head attention layer. This allows the attention mechanism to take into account both the node features and the temporal difference information at each time step, as well as the time order of the updates.

Finally, the output of the multi-head attention encoder is concatenated with the old node embeddings:

$$\mathbf{u}_i(t+1)=\mathbf{u}_i(t)+\phi\left(\partial\mathbf{u}_i(t),\mathbf{f}_i(t)\right) \tag{5.9}$$

$$=\mathbf{u}_i(t)+MultiHead\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right). \tag{5.10}$$

A normalization layer is set here to limit the mean and variance of the obtained node embeddings.

5.6.3 Ensemble MLP decoder

Node embeddings can be used for various downstream tasks. In this work, an ensemble MLP decoder is used to solve the real-time node label prediction task. Ensemble learning techniques are effective for imbalanced data [Galar et al., 2011]. They combine the results from several classifiers to improve the performance of a single classifier by reducing the variance of results. On the other hand, single classifiers are easily affected by the imbalanced dataset and tend to skew towards the majority classes, resulting in poorer performance on the minority classes. The ensemble MLP decoder is composed of multiple individual MLPs. Each MLP maps the learned node embedding $\mathbf{u}_i(t)$ to the probability of node labels. The sum of all MLP outputs is then fed to a softmax function to assign a predicted node label probability $\hat{y}_i(t)$:

Table 5.3: Structure distinction among TGNNs

	Aggregator	Encoder	Classifier	Training
TGAT	Self-attention	Multi-head attention	MLP	Two-step
TGN	Memory & message aggregator	Multi-head attention	MLP	Two-step
APAN	Asynchronous mail propagator	Multi-head attention	MLP	Two-step
MetaDyGNN	Hierarchically Adaptive Meta-Learner	Attention mechanism	MLP	End-to-end
TDGNN	Temporal difference aggregation	Multi-head attention	Ensembled MLPs	End-to-end

$$\hat{y}_i(t) = Softmax\left(\frac{1}{N}\sum_{N} MLP\left(\mathbf{u}_i(t)\right)\right). \tag{5.11}$$

Finally, we use a Binary Cross Entropy Loss function to train the TDGNN model.

$$BCELoss(\hat{y}_i, l_i) = -\left(l_i \cdot \log(\hat{y}_i) + (1 - l_i) \cdot \log(1 - \hat{y}_i)\right), \tag{5.12}$$

where $\hat{y}_i$ is the predicted node label probability and $l_i \in \mathbf{L}$ is the true node label.

5.6.4 Computational Complexity

We analyze the computational complexity of the Temporal Difference Aggregation, Multi-head Attention Encoder, and Ensemble MLP decoders. With regard to Eqs.(5.2), (5.4)∼(5.8), (5.11), the complexity is dominated by matrix multiplication. We assume that the number of edges(graph actions) is N, and the node embedding $\mathbf{V}(t)$ and node features $\mathbf{F}(t)$ have the same dimension D for the hidden units. The complexity of Temporal Difference Aggregation is $\mathcal{O}(ND)$. The complexity of the Multi-head Attention Encoder is $\mathcal{O}(N^2D + ND^2)$. The complexity of Ensemble MLP decoders is $\mathcal{O}(ND)$. Therefore, the total complexity of TDGNN is $\mathcal{O}(ND) + \mathcal{O}(N^2D + ND^2) + \mathcal{O}(ND) = \mathcal{O}(ND(N + D + 2)) = \mathcal{O}(ND(N + D))$.

Considering real-time computation, we must process the input graph actions in batches within a certain time window. We assume that all the batches have the same size K, and then the total number of batches is N/K. In this case, the total complexity of TDGNN is $\mathcal{O}((N/K) \times (KD(K + D))) = \mathcal{O}(ND(K + D))$. Since K is much smaller than N, the complexity in the real-time computation case is smaller than in the normal computation case. The complexity analysis above demonstrates that our framework works more efficiently in the case of real-time computation.

5.6.5 TDGNN vs. TGNNs

Table 5.3 compares the structural distinctions between our TDGNN model and three state-of-the-art TGNNs: TGAT [Xu et al., 2020], TGN [Rossi et al., 2020], APAN [Wang et al., 2021], and MetaDyGNN [Yang et al., 2022a]. TGNNs have

similar indiscriminate neighbor aggregators associated with their respective modules: Self-attention, Memory & message aggregator, Asynchronous mail propagator, and hierarchically adaptive Meta-learner. TGAT, TGN, and APAN all use a Multi-head attention layer as the encoder and an MLP as the classifier. Additionally, they all require two-step training: training the model parameters first, and then training the decoder parameters for specific downstream tasks. MetaDyGNN is comparably a more lightweight framework since it develops a novel hierarchically adaptive meta-learner, and does not use the multi-head attention encoder and two-step training. On the other hand, the proposed TDGNN model has a temporal difference aggregator and an ensembled MLP classifier, and it is trained end-to-end. The proposed TDGNN model addresses the limitations of existing TGNNs mentioned in Section 5.5 from the following aspects:

1. TDGNN is an end-to-end model that only needs to be trained once. It is more efficient and convenient than those TGNNs that require two-step training.

2. The temporal difference module computes both the direction and rate of information propagation, which is significant for predicting temporal node embeddings in the future.

3. TDGNN updates temporal node labels in real-time during the training batches, enabling the prediction of dynamic node labels and the time at which they were changed. In contrast, TGNNs update temporal nodes after the training batch is finished, causing a delay in predicting temporal information.

5.6.6 Strategies for Data Imbalance

The superchat donations only make up a small fraction of all the chat messages in the live stream, resulting in a highly imbalanced dataset that confuses the model and leads to poor performance. To address this issue, we employ the following strategies to mitigate the negative impact of data imbalance. In Section 5.7.6, we present experimental results that demonstrate how these strategies affect the imbalance ratio and model performance.

Filtering on original data

We have refined the dataset by removing duplicated and meaningless chat messages, as well as filtering out messages that are too short. To do this, we first removed punctuation marks, numbers, and exceptional control characters from the chat messages. We then filtered out messages with fewer than 5 characters. Finally, we used the Python library *difflib* to check for duplicate messages that appear within a certain time window. As a result of this filtering process, 1.6% of the non-superchat samples in the dataset were removed.

Table 5.4: Cost matrix

	Actual Negative	Actual Positive
Predicted Negative	$C(0,0)$, TN	$C(0,1)$, FN
Predicted Positive	$C(1,0)$, FP	$C(1,1)$, TP

Tuning on graph generation

We controlled the number of dynamic edges and further the proportion of positive samples by adjusting the cosine similarity threshold θ_2 in Algorithm 5.1. A higher θ_2 generates fewer edges, and a lower θ_2 generates more edges. Specifically, we set θ_2 as $\cos(\pi/12)$. Additionally, we tested two other values of θ_2, namely $\cos(\pi/6)$ and $\cos(\pi/3)$, and evaluated their impact on the model performance.

Undersampling on training samples

We employed an under-sampling strategy to address the imbalance between positive and negative samples in our dataset. We kept all the data in the minority class and reduced the size of the majority class during model training. This approach corrected the imbalanced data and reduced the risk of skewing towards the majority class. We tested various undersampling ratios of the two classes in our experiments and ultimately set the ratio as $1:1$.

Cost-sensitive loss function

We apply a cost-sensitive learning method to self-adjust the penalty factor in the loss function during model training. Most machine learning algorithms assume that all misclassification errors made by a model are equal. This is often not the case for imbalanced classification problems where missing a minority class case is worse than incorrectly classifying an example from the majority class. Cost-sensitive learning is a subfield of machine learning that takes the costs of prediction errors into account when training a machine learning model [Elkan, 2001]. In cost-sensitive learning, instead of each sample being either correctly or incorrectly classified, each class is given a misclassification cost. Thus, instead of trying to optimize the accuracy, the problem is then to minimize the total misclassification cost.

Specifically, we set a cost matrix that assigns a cost to each cell in the confusion matrix. Table 5.4 shows the cost matrix we used, where $C(,)$ denotes the cost of predicting one class when the actual class is another. The acronyms of each cell from the confusion matrix are also listed (e.g., False Positive is FP).

This work defines costs based on the inverse class distribution, assuming a minority-to-majority class ratio of $1:N$ in the dataset. We invert this ratio to obtain the cost of misclassification errors, where the cost of a False Negative $C(0,1)$

Table 5.5: Statistics of the live streaming dynamic graphs.

Dataset length	Short	Mid	Long
Durations (hrs.)	8.61	47.22	78.49
# Nodes	6,225	28,582	41,156
# Edges	1,660,813	9,498,600	15,097,110
# Positive labels	105,207	258,079	525,964
% Positive labels	6.3%	2.7%	3.4%

is N, and the cost of a False Positive $C(1,0)$ is 1. The cost of True Negative $C(0,0)$ and True Positive $C(1,1)$ are set as 0 since they are correctly predicted. Therefore, the total cost of a classifier is defined as the cost-weighted sum of False Negatives and False Positives using this framework:

$$Total\_cost = C(0,1) * FN + C(1,0) * FP + C(0,0) * TN + C(1,1) * TP \quad (5.13)$$

$$= N * FN + 1 * FP. \quad (5.14)$$

5.7 Experiments

We conducted experiments on the task of predicting dynamic node labels. First, we explain the setup of experiments and baselines used in our study. Then, we evaluate the experimental results and model performance. Finally, we discuss additional factors such as training time, time delay in real-time prediction, effect of imbalance strategies, parameter sensitivity, and node embedding visualization.

5.7.1 Setup of the Experiment

We prepared three continuous-time dynamic graphs from the dataset mentioned in Sec. 5.3.1. The detailed statistics are listed in Table 5.5. The three graphs represent videos of different lengths: the *Short* dataset contains chat messages in an 8-hour live streaming video, the *Mid* dataset contains chat messages in a 47-hour video compilation of a week, and the *Long* dataset contains chat messages in a 78-hour video compilation of two weeks. The model training is customized for each streamer to optimize prediction performance. However, it is worth noting that the ratios of positive labels are minimal in all three dynamic graphs as superchat messages only represent a minor portion of all chat messages in real-world scenarios. Therefore, the experiments will be conducted on an imbalanced dataset. We split the chat messages in the dataset into training, validation, and test sets based on the time order. The first 50%/70%/90% of chat messages are used as the training set, and the remaining messages are equally divided into validation and test sets.

Our model processes the chat messages into batches based on the time window. Within each time window, the chat messages (represented as updated node features) and node interactions (represented as edges) are included as a sequence of graph actions. All these sequences are then fed into our model to predict the dynamic node labels (when and who sends superchat). The model updates the involved node status and edges based on the graph action sequence and provides predictions on the changes in dynamic node labels.

We fine-tune the common model hyperparameters, such as learning rate, batch size, and drop-out rate, by manual search. Also we conduct parameter sensitivity experiments for some special hyperparameters in TDGNN by grid search, which will be introduced in Section 5.7.6 and Section 5.7.7. The model is trained using the Adam optimizer with a learning rate of 0.0001, a batch size of $5,000$ for training, validation, and testing, and a dropout rate of 0.2. We set the number of attention heads to 2. For the ensembled MLP decoders, we use a three-layer linear neural network with hidden sizes of 64 and 10. The training process is limited to a maximum of 20 iterations. Furthermore, if the validation loss does not improve for five consecutive iterations, training will be stopped early.

The node embedding dimension and node feature dimension are set to 128. The number of maximum neighbor sampling is 10, and the number of ensembled MLP decoders is 15 for all three datasets in default. We will test the parameter sensitivity in the following experiments and prove that our proposed model results are not sensitive to hyperparameters.

5.7.2 Baselines

We consider traditional decision tree methods, sequence-based models, static graph representation learning methods, NLP text classification methods, and TGNNs as baselines, as listed below.

1. **GBDT**: A Gradient Boost Decision Tree (GBDT) classifier from the scikit-learn toolkit.

2. **XGBoost**: An ensemble gradient boosting decision tree model from XGBosst library.

3. **LSTM-FCN** [Karim et al., 2018]: A time sequence model combining long short-term memory (LSTM) networks and a fully convolutional network (FCN).

4. **ALSTM-FCN** [Karim et al., 2018]: An alternative LSTM-FCN with attention layers following the LSTM cells.

5. **GCN** [Kipf and Welling, 2016a]: Graph Convolutional Networks (GCN) on static graphs.

Table 5.6: Input information required by the baselines

		Chat messages texts	Graph structure	Temporal information
Gradient boosting algorithms	**GBDT**	✓	×	×
	XGBoost	✓	×	×
Time sequence model	**LSTM-FCN**	✓	×	✓
	ALSTM-FCN	✓	×	✓
Static graph methods	**GCN**	✓	✓	×
	GAT	✓	✓	×
NLP	**BERT**	✓	×	×
Dynamic graph methods	**TGAT**	✓	✓	✓
	APAN	✓	✓	✓
	TGN	✓	✓	✓
	MetaDyGNN	✓	✓	✓
Proposed methods	**TDGNN**	✓	✓	✓

6. **GAT** [Veličković et al., 2018]: Graph Atention Networks (GAT) on static graphs.

7. **BERT** [Devlin et al., 2018]: Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model for NLP pre-training.

8. **TGAT** [Xu et al., 2020]: A temporal graph attention structure to aggregate temporal-topological neighborhood features and to learn the time-feature interactions

9. **APAN** [Wang et al., 2021]: An asynchronous continuous time dynamic graph algorithm for real-time temporal graph embedding.

10. **TGN** [Rossi et al., 2020]: A generic, efficient neural network framework for deep learning specialized in continuous-time dynamic graphs.

11. **MetaDyGNN** [Yang et al., 2022a]: A meta-learning framework for few-shot scenarios in dynamic networks.

Table 5.6 demonstrates the input information required by each baseline method. We keep the default parameter settings in respective methods during training and testing. The detailed experimental settings of baselines are as follows:

1. The gradient boosting algorithms (GBDT and XGBoost) receive every individual sentence embedding vector of chat messages and generate a predicted label indicating whether the corresponding chat message is a superchat. The implementation of these algorithms is based on scikit-learn[10] and XGBoost[11].

2. The time sequence models (LSTM-FCN and ALSTM-FCN) achieved state-of-the-art performance on the task of time sequence classification. We first

[10]https://scikit-learn.org/
[11]https://xgboost.readthedocs.io/en/stable/

Table 5.7: Fine-tuned hyperparameters in the dynamic GNN baseline methods.

	TGN	**TGAT**	**APAN**	**TDGNN**
Learning rate	0.0001	0.0005	0.0001	0.0001
Dropout	0.1	0.2	0.1	0.2
Hidden dim.	128	128	128	128
# Attention heads	2	2	2	2
# GNN encoder layers	2	2	2	2
# MLP decoder layers	3	2	2	2

separate the sequences of sentence embeddings over time for each viewer and then feed these sequences to the models. The models generate a sequence of hidden state embeddings for each viewer, which are then used to predict the labels of chat messages. The implementation of these models refers to the GitHub repository LSTM-FCN[12].

3. The static graph methods (GCN and GAT) are commonly used to model graph-structured real-world entities. We built static graphs to represent viewer relations based on the continuously-time dynamic graphs generated in Section 5.4. The nodes represent viewers, and the edge weights represent the frequency of edge changes in the dynamic graphs. The implementation is based on the GitHub repositories PyGCN[13] and GAT[14].

4. BERT is a transformer-based machine learning technique for NLP pre-training. We exploit a Japanese BERT pretrained model to encode the chat messages and integrate them as long paragraphs for each viewer. The BERT for sequence classification model provided by Huggingface[15] is used to process the long paragraphs and predict the appearance of superchats.

5. The dynamic graph neural networks (TGAT, APAN, TGN, and MetaDyGNN) use the same inputs and hyper-parameters as the proposed TDGNN. The implementation refers to the GitHub repositories TGAT[16], APAN[17], TGN[18], and MetaDyGNN[19]. Notably, the hyperparameters of TGAT, APAN, and TGN are fine-tuned to exhibits their best performance. The values of hyperparameters are listed in Table 5.7.

[12]https://github.com/titu1994/LSTM-FCN
[13]https://github.com/tkipf/pygcn
[14]https://github.com/psh150204/GAT
[15]https://huggingface.co/cl-tohoku/bert-base-japanese
[16]https://github.com/StatsDLMathsRecomSys/Inductive-representation-learning-on-temporal-graphs
[17]https://github.com/WangXuhongCN/APAN
[18]https://github.com/twitter-research/tgn
[19]https://github.com/BUPT-GAMMA/MetaDyGNN

Table 5.8: AUC scores for predicting the real-time node labels

Model	Training set Ratio								
	50%			70%			90%		
	Short	Mid.	Long	Short	Mid.	Long	Short	Mid.	Long
GBDT	0.500	0.472	0.500	0.552	0.469	0.515	0.489	0.458	0.472
XGBoost	0.500	0.509	0.500	0.553	0.495	0.518	0.513	0.462	0.499
LSTM-FCN	0.468	0.505	0.507	0.497	0.499	0.506	0.431	0.499	0.500
ALSTM-FCN	0.485	0.505	0.508	0.499	0.499	0.499	0.500	0.501	0.472
GCN	0.500	0.499	0.499	0.500	0.499	0.499	0.500	0.499	0.499
GAT	0.510	0.510	0.510	0.531	0.531	0.531	0.548	0.548	0.548
BERT	0.630	0.560	0.580	0.570	0.570	0.590	0.620	0.540	0.670
TGAT	0.587	0.672	0.668	0.596	0.754	0.687	0.630	0.794	0.703
APAN	0.521	0.647	0.679	0.506	0.616	0.680	0.589	0.653	0.667
TGN	0.654	0.610	0.784	0.620	0.610	0.795	0.520	0.854	0.902
MetaDyGNN	0.616	0.671	0.673	0.624	0.689	0.684	0.771	0.746	0.695
TDGNN	**0.765**	**0.738**	**0.799**	**0.679**	**0.765**	**0.805**	**0.805**	**0.884**	**0.916**
TDGNN w/o Diff	0.708	0.572	0.688	0.599	0.621	0.653	0.628	0.834	0.830

5.7.3 Evaluation

Table 5.8 presents the experimental results of both the TDGNN model and the baselines. We evaluated the performance in terms of the Area under the ROC Curve (AUC) score, which measures the model's prediction quality, regardless of the classification threshold and how the datasets are imbalanced. Our proposed model shows the best overall performance, achieving the highest AUC scores on all three datasets and all training set ratios, which are marked in bold in the table. We attribute the excellent performance of TDGNN to the deliberately designed continuous-time dynamic graph that considers continuously changing edges and frequently updated node embeddings. We analyzed the performance of the baselines and inferred the reasons for their respective performances as follows:

1. The gradient boosting algorithms (GBDT and XGBoost) only consider the sentence embeddings of chat messages. However, as the textual content of superchat messages is not distinct from normal chat messages, their performance suffers a loss of $0.126 \sim 0.444$ compared to TDGNN.

2. The time sequence models (LSTM-FCN and ALSTM-FCN) receive the sequence of sentence vectors and generate a sequence of hidden state embeddings for each viewer. However, as a majority of viewers never send superchat messages, the sequences of those viewers cannot provide efficient training, resulting in a performance loss of $0.182 \sim 0.416$ compared to TDGNN.

3. The static graph methods (GCN and GAT) are good at exploiting graph structure information and edge interactions among nodes. However, static graph

models are not designed to fit temporal data, and each node only stores one sentence embedding, leading to the loss of much temporal and chat message information. The experimental results show a performance loss of $0.179 \sim 0.368$ compared to TDGNN.

4. The BERT model is not designed to capture temporal information, and it only stores the all-time sentence embeddings of chat messages, without considering graph structure information. As a result, it shows a performance loss of $0.109 \sim 0.246$ compared to TDGNN.

5. Although dynamic graph neural networks (TGAT, APAN, TGN, and MetaDyGNN) are designed for continuous-time dynamic graphs, the performance of these models depends on their underlying architectures and their ability to capture both temporal and graph structure information effectively. TDGNN outperforms TGAT, APAN, TGN, and MetaDyGNN with a performance gain of $0.026 \sim 0.285$, which can be attributed to the distinctive architecture of TDGNN as explained in Section 5.6.5.

In addition, we conducted an experiment to verify the contribution of the temporal difference module in the TDGNN model. We tested the TDGNN model by replacing the temporal difference module with the raw aggregated neighbor node embeddings, and the results are presented in the table as TGDNN w/o Diff. The TGDNN w/o Diff model showed a performance loss of $0.040 \sim 0.177$ compared to the TDGNN model. Furthermore, when compared with the strongest baseline TGN, TGDNN w/o Diff outperformed TGN by 0.108 on the Short dataset but had a performance loss of $0.038 \sim 0.142$ on the mid and long datasets. The results indicate that the temporal difference module plays a crucial role in the TDGNN model, and its absence leads to a significant decrease in performance.

5.7.4 Training Time

In this section, we compare the training time of our proposed model with that of other baselines, particularly the three TGNN baselines: TGAT, APAN, TGN, and MetaDyGNN. Given that our proposed model needs to process real-time live streaming chat messages and predict potential superchat donors as soon as possible, it is crucial to be highly time-efficient. To evaluate the training time, we ran the models on three live streaming dynamic graphs using an Intel Xeon Platinum 8360Y CPU (9 cores, 2.40GHz) and an NVIDIA A100 for NVLink 40GiB HBM2 GPU.

Fig. 5.8 demonstrate the training time (seconds) per each epoch on the three live streaming dynamic graph datasets. We compute the total training time of TGN, TGAT, APAN, and MetaDyGNN, including the model parameter and decoder training. TDGNN has close time efficiency to TGN. Training the TGN model takes

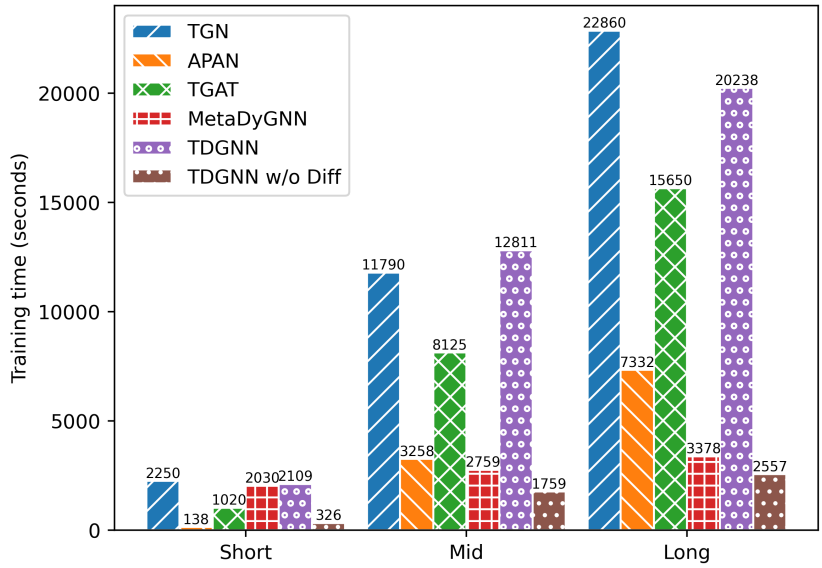


Figure 5.8: Training time (seconds) per epoch in three dynamic graph datasets.

longer than the TDGNN model for both short and long datasets, while the TDGNN model takes longer to train for the mid dataset. The TGAT model takes less time to train than both the TDGNN and TGN models, and the APAN and MetaDyGNN models take the least training time out of all the models.

We believe that the time efficiency of these models is largely dependent on their respective structures. TGAT is a classic TGNN structure that utilizes a neighbor node aggregation module and a multi-head attention encoder. TGN, on the other hand, improves upon TGAT by adding a memory module to update temporal node embeddings. However, this extra component requires more training time. APAN implements an asynchronous propagation mechanism that greatly reduces training time. MetaDyGNN also performs less training time for its lightweight framework. As a result, our proposed TDGNN model achieves the best AUC score and has training time that is comparable to TGN. Additionally, since TDGNN is an end-to-end structure that only needs to be trained once, it is more convenient than other TGNN baselines. Additionally, it is worth noting that TDGNN without the temporal difference module (TDGNN w/o diff) takes considerably less time than TDGNN. This indicates that the temporal difference module is the primary factor that significantly increases the training time.

5.7.5 Time Delay When Updating Node Status

In the temporal node prediction task, the node status (e.g., node embeddings, node labels) is constantly evolving in continuous-time dynamic graphs. As shown in

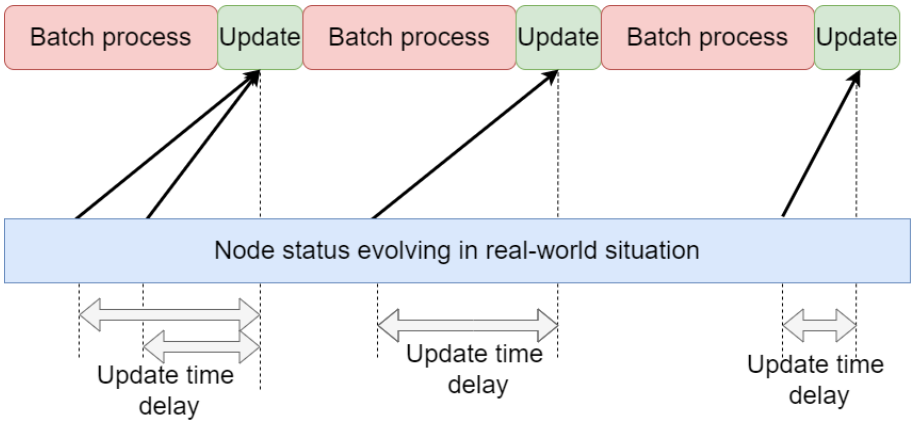


Figure 5.9: The update time delay in previous TGNNs. Node status updated during batch process are reflected until the batches are finished, thus resulting in an update time delay.

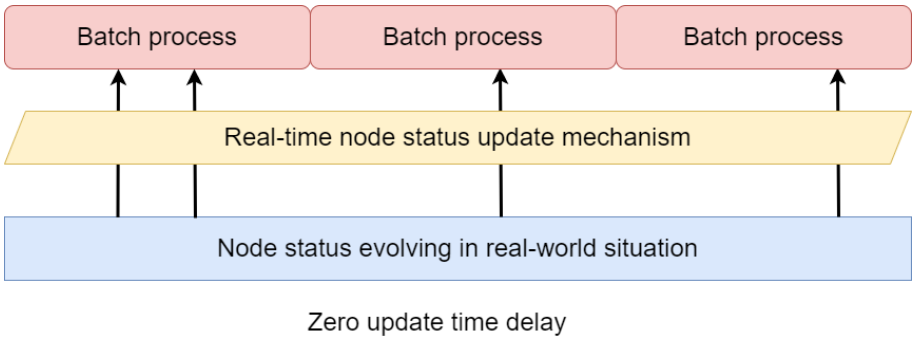


Figure 5.10: The zero update time delay in TDGNN. TDGNN has a real-time node status update mechanism that can update node information during batch process.

Fig 5.9, TGN, TGAT, and APAN process the graph actions in batches, which may result in some delay in training and inference. This delay can lead to a situation where a node's status has changed in the real world, but the change was not reflected in the models until the batch processing was completed. We refer to this delay as the *update time delay*. The update time delay can significantly affect the timeliness and freshness of the predicted results, especially in constantly evolving situations. If the delay is too long, the predicted results may become irrelevant as the target node status would have changed. Moreover, the outdated node status can negatively impact the model training. Our research target is to identify the exact timing when node labels change and predict the exact timing when superchats are posted. Therefore, it is critical to minimize the update time delay during model training and inference to achieve accurate predictions that are sensitive to time.

To mitigate the impact of update time delay, the proposed TDGNN model is equipped with a real-time node status updating mechanism that allows for the simultaneous updating of node status during training batches, as illustrated in Fig 5.10. Unlike TGN, TGAT, and APAN, which are constrained by the batch size, TDGNN can update node status as soon as it changes, regardless of batch size limitations. As

Table 5.9: Update time delay (seconds) and AUC scores in TGNNs with different batch sizes

Batch size	Update time delay					AUC score				
	TGN	**TGAT**	**APAN**	MetaDyGNN	**TDGNN**	**TGN**	**TGAT**	**APAN**	MetaDyGNN	**TDGNN**
100	1.97	2.27	1.21	15.75	0.00	0.392	0.528	0.603	0.655	0.465
200	3.96	4.55	2.42	34.75	0.00	0.425	0.524	0.644	**0.771**	0.596
500	9.94	9.59	4.35	OOM	0.00	0.410	0.520	0.601	OOM	0.602
1,000	20.42	22.79	5.40	OOM	0.00	0.499	0.520	0.656	OOM	0.638
2,000	39.76	45.58	12.01	OOM	0.00	0.520	**0.591**	**0.667**	OOM	0.741
5,000	99.22	117.93	36.72	OOM	0.00	**0.609**	0.553	0.520	OOM	**0.805**
10,000	197.76	227.93	48.77	OOM	0.00	0.507	0.528	0.559	OOM	0.608
20,000	392.86	455.87	60.60	OOM	OOM	0.460	0.453	0.519	OOM	OOM
50,000	988.93	941.80	137.42	OOM	OOM	0.513	0.443	0.572	OOM	OOM
100,000	1911.95	1824.74	364.13	OOM	OOM	0.488	0.476	0.502	OOM	OOM

a result, TDGNN can provide more timely predictions for the timing of superchat messages.

Table 5.9 shows the average update time delay (in seconds) and AUC performance of TGNNs with different batch sizes. Unlike TGN, TGAT, APAN. and MetaDyGNN, TDGNN has a real-time updating mechanism, and thus has zero update time delay, regardless of batch size. However, zero update time delay only means that node status in batches is updated as soon as the node changes in the real world, and does not necessarily imply that the predicted results given by TDGNN have zero error. In fact, out-of-memory errors can occur when the batch size is too large, presumably due to the high computational cost associated with the real-time node status updating mechanism for large-scale batches.

The update time delays in TGN and TGAT are almost directly proportional to the batch sizes, while the AUC scores do not fluctuate significantly. In contrast, APAN trains the model asynchronously, and the update time delay is less affected by batch sizes. MetaDyGNN has a hierarchical structure and thus sensitive to the batch size. Small increase in batch size will cause large extra demand in GPU memory and tend to suffer from OOM (Out-of-Memory) problem. However, the AUC scores decrease significantly as batch sizes increase. Notably, TGN and TDGNN perform poorly in terms of AUC scores when batch sizes are too small, likely due to the imbalanced dataset. When the batch size is too small, there may be only a few positive samples or even no positive samples in the batches, causing the model to produce identical prediction results for all samples, which results in low training efficiency. TDGNN performs better than TGN in this scenario, thanks to several strategies employed to alleviate the influence of imbalanced datasets.

In summary, our study highlights the trade-off between model performance and update time delay in the temporal node prediction task. While large batch sizes require significant computational resources, small batch sizes result in poor performance, particularly in imbalanced datasets. Our approach strikes a balance between the two factors by using a batch size of 5,000, enabling us to achieve high performance while maintaining low update time delay. Our findings have implications for

other similar tasks, where researchers need to carefully weigh the costs and benefits of different model architectures and batch sizes to optimize both performance and efficiency.

5.7.6 Effect of the Strategies for Data Imbalance

In this section, we discuss the effects of the strategies for addressing data imbalance that were mentioned in Section 5.6.6. Specifically, we test how the cosine similarity threshold, undersampling ratios, and cost-sensitive loss function impact both the data imbalance and the model performance. The experiment is conducted on the mid dataset, with a training set ratio of 90%.

The cosine similarity thresholds θ_2 have an impact on the number of dynamic edges and further the proportion of positive samples in the dataset. We tested three values of θ_2: $\cos(\pi/3)$, $\cos(\pi/6)$, and $\cos(\pi/12)$. With $\theta_2 = \cos(\pi/3)$, we obtained 840,732 positive samples out of a total of 8,129,340 samples, resulting in an imbalance ratio of 10.3%. With $\theta_2 = \cos(\pi/6)$, we obtained 271,925 positive samples out of 11,176,545 samples, resulting in an imbalance ratio of 2.4%. Finally, with $\theta_2 = \cos(\pi/12)$, we obtained 258,079 positive samples out of 9,498,600 samples, resulting in an imbalance ratio of 2.7%.

The undersampling ratio refers to the ratio between the majority and minority classes in training batches, and it reduces the risk of the model predicting results that skew towards the majority class. We test three different undersampling ratios: $1:1$, $3:1$, and $5:1$. Additionally, we also test the model's performance without using any undersampling strategy, which means that the default imbalance ratio of the original dataset is maintained.

The cost-sensitive loss function adjusts the penalty factor in the loss function during model training. We use the *BCEWithLogitsLoss* as the loss function, and the cost of positive samples is dynamically adjusted based on the ratio of negative samples to positive samples in each training batch. Specifically, when we use undersampling and set the undersampling ratio to $1:1$, the cost-sensitive loss function is equivalent to the *BCELoss* function.

Table 5.10 shows the test results of the strategies for addressing data imbalance. Among the three datasets with different cosine similarity thresholds θ_2, the dataset with $\theta_2 = \cos(\pi/12)$ achieved the best overall performance. This may be due to the fact that a high threshold for generating dynamic edges ensures the quality of positive samples in the dataset. The dataset with $\theta_2 = \cos(\pi/6)$ had the lowest imbalance ratio, while the dataset with $\theta_2 = \cos(\pi/3)$ had the highest imbalance ratio but the lowest quality of positive samples, resulting in poorer performance than the dataset with $\theta_2 = \cos(\pi/12)$.

Regarding the undersampling strategy, the best results were obtained with an undersampling ratio of $1:1$, which is highlighted in bold in the table. As the

Table 5.10: AUC scores under the effect of strategies for data imbalance

Cosine similarity threshold θ_2		$\cos(\pi/3)$	$\cos(\pi/6)$	$\cos(\pi/12)$
Imbalance ratio		10.3%	2.4%	2.7%
With cost-sensitive	**No undersampling**	0.717	0.736	0.716
	Neg:Pos $= 1:1$	0.761	0.780	**0.884**
	Neg:Pos $= 3:1$	0.742	0.712	0.820
	Neg:Pos $= 5:1$	0.737	0.676	0.700
W/o cost-sensitive	**No undersampling**	0.498	0.619	0.714
	Neg:Pos $= 1:1$	0.761	0.780	**0.884**
	Neg:Pos $= 3:1$	0.618	0.683	0.727
	Neg:Pos $= 5:1$	0.653	0.652	0.698

undersampling ratio increased, the performance gradually decreased, with the worst result being obtained without the undersampling strategy.

In terms of the use of a cost-sensitive loss function, the results were the same as with an undersampling ratio of $1:1$. This is because, as mentioned earlier, the cost-sensitive loss function is equivalent to the *BCELoss* function when the undersampling ratio is $1:1$. However, the performance with a cost-sensitive loss function was relatively better than that without.

In summary, we evaluated the model's performance with different strategies for addressing data imbalance, and found that these strategies had a significant effect on performance. Based on the experimental results, the best choices of strategies were a cosine similarity threshold $\theta_2 = \cos(\pi/12)$, an imbalance ratio of $1:1$, and using a cost-sensitive loss function. We used these settings in the test described in Section 5.7.3.

5.7.7 Parameter Sensitivity

In Fig.5.11 and Fig.5.12, we present the performance of the TDGNN model with respect to two significant hyperparameters, namely the number of sampled neighbor nodes in temporal difference and the number of ensembled MLP decoders. The results are based on the short dataset with a training set ratio of 90%. The highest AUC score is obtained with 15 MLP decoders and 10 sampled neighbor nodes. It can be concluded that increasing the number of ensembled MLP decoders or sampled neighbor nodes beyond a certain value does not improve the performance. The number of sampled neighbors is a common parameter in GNN models. If too many neighbors are aggregated, the model may fail to identify the most critical information. On the other hand, if too few neighbors are sampled, some important neighbors may be ignored.

Also, the number of sampled neighbor nodes affects training time much more than the number of MLP decoders. The shortest and longest training times fluctu-

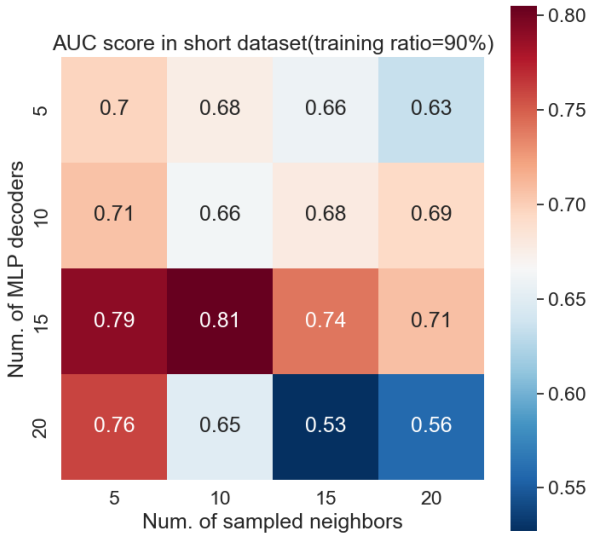


Figure 5.11: The AUC score with different numbers of sampled neighbors and MLP decoders.

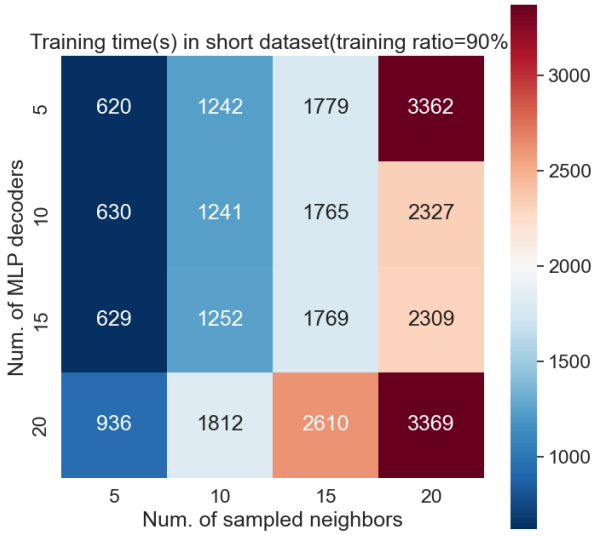


Figure 5.12: The training time (seconds) per epoch with different numbers of sampled neighbors and MLP decoders.

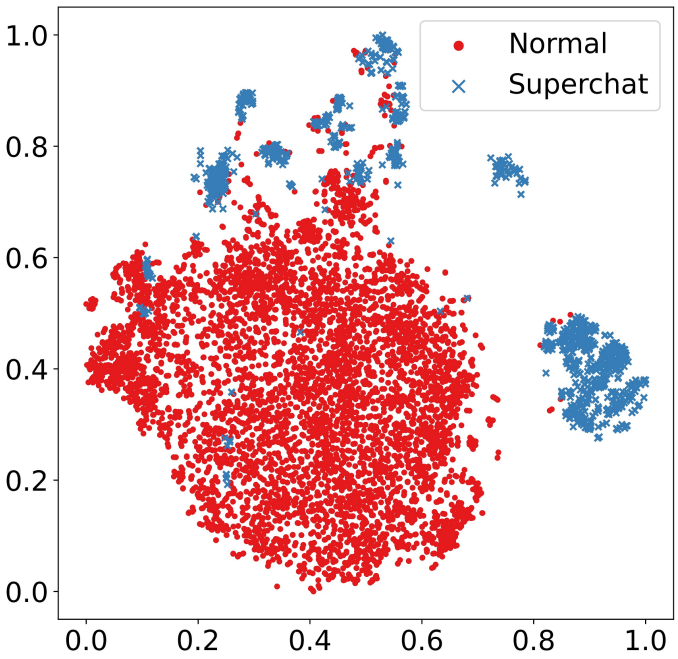


Figure 5.13: The t-SNE visualization of the node embeddings learned by TDGNN.

ate by 2,742 seconds with different numbers of sampled neighbors, while the same fluctuation is only 1,060 seconds with different MLP decoders. This result shows that the number of sampled neighbors is the main factor affecting computational complexity.

5.7.8 Node Embedding Visualization

We provide an illustrative visualization of the node embeddings by t-SNE tools [Van der Maaten and Hinton, 2008]. Fig. 5.13 shows the distribution of node embeddings learned by our model on the long dataset with the training set ratio equaling 70%. Blue cross markers denote the node embedding of viewers who send superchats, and red circle markers denote the node embedding of those who did not. We observe that node embeddings with the same label are clustering, and a large margin separates node embeddings with different labels. The results prove that our method can learn discriminative node embeddings to distinguish different labels.

5.8 Conclusion and Discussion

In summary, our work provides a comprehensive solution for predicting real-time donations in online live streaming services by leveraging dynamic graph neural network models. By constructing a continuous-time dynamic graph representation of viewer interactions and chat messages, our model can capture the temporal dynamics of the data and effectively predict potential donations. The experimental results demonstrate the superiority of our proposed TDGNN model over various

baseline methods. Additionally, our study confirms the feasibility of the model's predictions in real-world situations by examining the training time, update delay, and visualization.

Our work also contributes to the rapidly growing and promising area of study on real-time donations in live streaming services. By predicting donations, our model can help content creators better understand their audience and improve engagement. Furthermore, our approach of using dynamic graph neural networks can be applied to other real-time prediction tasks in various domains. Overall, our work provides valuable insights and contributions to both the live streaming and dynamic graph neural network research communities.

Limitations

Firstly, the training time and inference time, as demonstrated in Fig. 5.8, do not exhibit overwhelming advantages when compared to other TGNN baselines, only outperform due to our end-to-end training style. Following the comparison between TDGNN and TDGNN w/o Diff, it is evident that the most time-consuming component within the TDGNN model is the temporal difference module. Hence, there is a compelling need to optimize the temporal difference module with the goal of reducing the training time and inference time of TDGNN.

Secondly, as listed in Table 5.9, the TDGNN model encounters out-of-memory issues when dealing with large batch sizes, in contrast to TGN, TGAT, and APAN, which do not exhibit such limitations. This highlights a space where our TDGNN model can be further enhanced in terms of computational complexity to effectively handle more scalable graphs.

Finally, this work exclusively delved into the scenario of homogeneous CTDG. However, the broader and more intricate real-world datasets often involve heterogeneous CTDGs. Consequently, our forthcoming work will be geared towards extending the TDGNN model to cater to the complexities of heterogeneous CTDGs.

Chapter 6

Discussion

In this chapter, we delve into additional considerations related to the proposed GNN models in real-world prediction tasks. We present a unified process for addressing prediction tasks using GNNs, emphasizing the need for domain-specific GNNs and exploring general methodologies for developing expert GNN frameworks. Additionally, we delineate the relationships among our proposed GNN frameworks, highlighting commonalities, distinctions, challenges addressed, and contributions achieved by each. Furthermore, we conduct additional experiments to evaluate the performance of our proposed GNN models on various benchmark real-world datasets, demonstrating the extensibility of our methods across different domains. Finally, we provide examples of real-world human social activities categorized under the most appropriate GNN frameworks, illustrating the diverse benefits of predictive analytics in understanding and predicting human behaviors and social trends.

6.1 Unified Design Considerations within Proposed GNN Frameworks

In this section, we conduct a comprehensive review and analysis of our proposed models, examining them through a unified perspective. First, we encapsulate the typical process of constructing GNN frameworks to model real-world data and tackle prediction tasks. Then, we discuss the imperative need for expert GNNs tailored to domain-specific real-world tasks. Finally, we introduce several critical methodologies for designing and developing such specialized GNNs. This holistic examination provides insights into the underlying principles and strategies that contribute to the effectiveness of our proposed GNN models across diverse applications and domains.

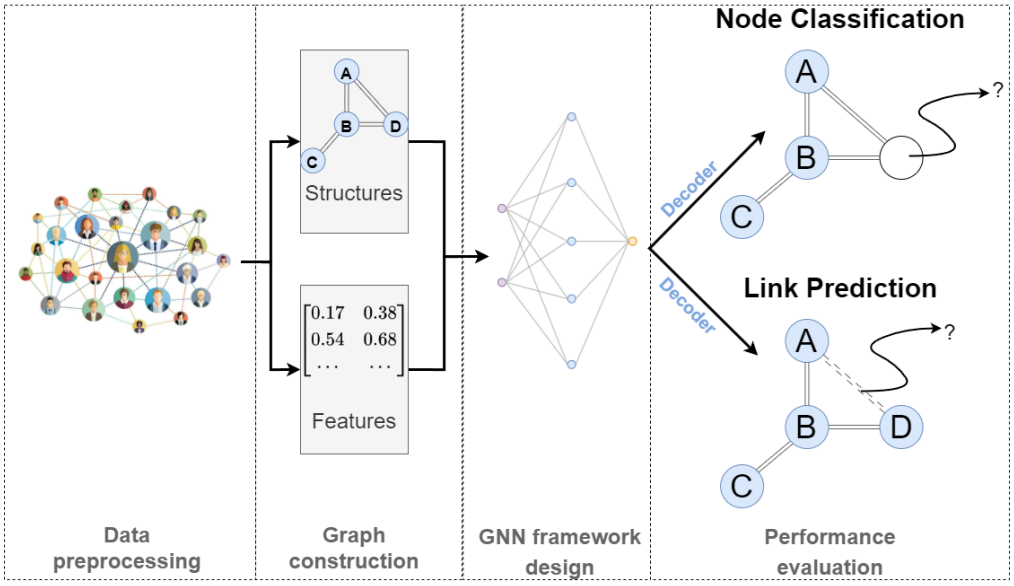


Figure 6.1: The unified process of addressing prediction tasks using GNN frameworks. First, we collect real-world data and perform necessary preprocessing steps to enhance its suitability for modeling. Then,we construct graph structures and incorporate feature information, tailoring them to the unique characteristics of the dataset under consideration. These structures and features are input into the GNN frameworks, where the models learn node representation embedding vectors. Finally, the trained models generate predictions for specific tasks. The model performance is assessed against state-of-the-art methods through extensive experimental results on targeted tasks.

6.1.1 Process of Addressing Prediction Tasks by GNN frameworks

Fig. 6.1 illustrates a comprehensive overview that encapsulates the systematic process employed by GNN frameworks in addressing prediction tasks. First, we collect real-world graph-structured data and carefully extract essential structures and features. This curated information is fed to the designated GNN frameworks, where the models autonomously learn node representation embedding vectors. Finally, the acquired embedding vectors are inputted into decoders, generating precise prediction results tailored to specific tasks. This streamlined workflow ensures the robustness and adaptability of GNN frameworks across various domains and prediction scenarios.

Data preprocessing

Data preprocessing constitutes a pivotal phase in implementing deep learning graph models, as the quality and format of input data can significantly impact the model's performance. The first step in data preprocessing involves data cleaning, wherein missing data and outliers are identified and addressed to mitigate potential negative

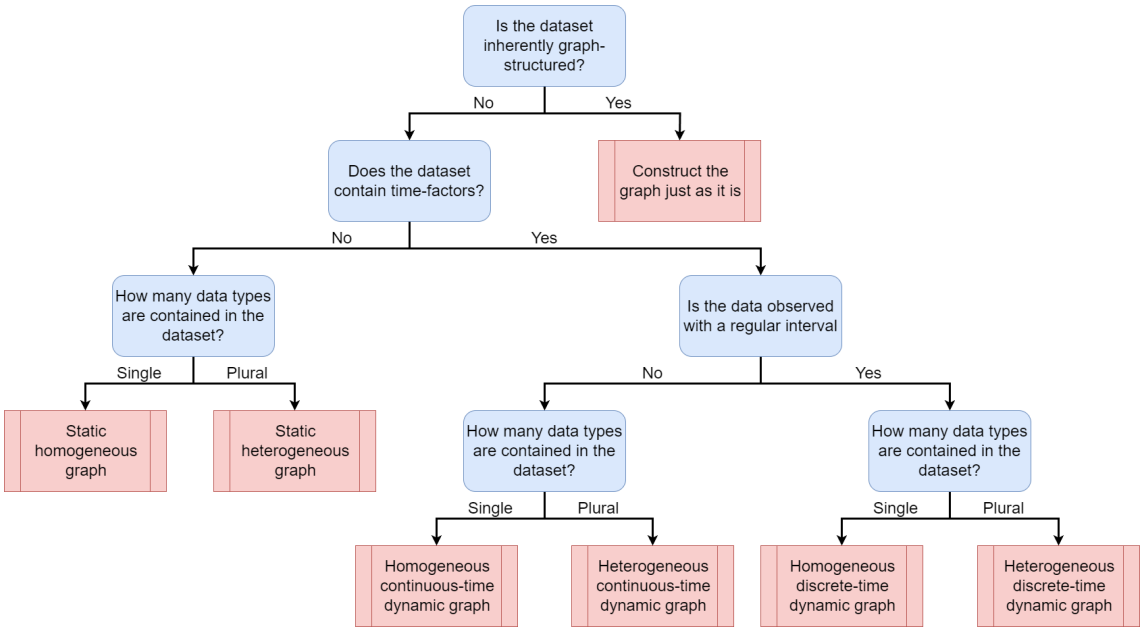


Figure 6.2: A flowchart of how to construct the appropriate graph from the dataset.

impacts on model performance. Notably, graph augmentation techniques, such as graph structure learning (GSL) [Jin et al., 2020] and graph contrastive learning (GCL) [You et al., 2020], are widely employed to enhance the overall quality of input graphs.

Subsequently, careful consideration is given to selecting critical data, which serves as node or edge features fed into GNNs. Numerical features undergo normalization to ensure alignment on a similar scale, with standard techniques encompassing min-max scaling or z-score normalization. In parallel, categorical features necessitate conversion into computable numerical formats, achieved through one-hot or label encoding methods.

Furthermore, the dataset is split into training, validation, and test sets. This division facilitates the training process, validation of the model's performance during training, and evaluation of the model's performance on previously unseen data.

Graph construction

In some cases, real-world data exhibits inherent graph-like structures, making it directly amenable for GNN models. However, there are instances where the data is not explicitly graph-structured, such as the live streaming data discussed in Chapter 5. In such cases, the construction of a graph structure becomes imperative.

Carefully considering the dataset's characteristics is essential to construct the most fitting graph structure. Fig. 6.2 outlines a comprehensive flowchart illustrating selecting the appropriate graph from the dataset. If the dataset inherently possesses a graph structure, it can be utilized directly. For non-graph-structured data, two key aspects come into play: time factors and the number of data types within the

dataset.

Dynamic graphs are adept at modeling datasets accompanied by time factors, while static graphs are more suitable for datasets lacking such temporal considerations. Additionally, a discrete-time dynamic graph (DTDG) is well-suited for datasets observed at regular intervals, whereas continuous-time dynamic graphs (CTDGs) are preferable for datasets with irregular temporal dynamics. Another critical consideration is the number of data types within the dataset. A homogeneous graph is appropriate if the dataset comprises a single data type. Conversely, datasets housing multiple data types are better represented using heterogeneous graphs. This strategic approach ensures the construction of a tailored and contextually relevant graph structure for effective utilization in GNN models.

GNN framework design

The architectural design of GNNs encompasses essential components, prominently featuring encoders and decoders. The configuration of encoders is determined by the specific characteristics of input graph structures and node/edge features, whereas decoders are tailored to address distinct downstream prediction tasks.

For tasks involving the prediction of categorical values, such as node/edge label classification and graph classification, the softmax function is commonly employed as a decoder. Utilizing the softmax function is advantageous, as its output results can be interpreted as probabilities assigned to predicted classes.

Conversely, when tackling tasks that involve predicting the numerical values of a specific attribute for a target entity, commonly referred to as regression tasks, a fully connected multilayer perceptron (MLP) is often employed as the decoder. The flexible and adaptive nature of the MLP allows it to effectively approximate continuous values from non-linearly separable data, making it well-suited for regression-based prediction tasks within the context of GNN architectures. The tailored encoder-decoder synergy aligns with the dataset and prediction task at hand, optimizing the overall performance and efficiency of the GNN model.

Performance evaluation

The evaluation of proposed GNN models is a pivotal stage in determining their performance on specific tasks compared to state-of-the-art models. This assessment occurs on the test set, which is distinct from the original dataset and remains unseen during model training.

The selection of evaluation metrics hinges on the task's nature, be it classification, regression, or other specialized applications. For classification tasks, metrics such as Accuracy, Recall, Precision, F1-score, AUC-ROC score, and mean Average Precision (AP) are representative. Meanwhile, regression tasks commonly employ

metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). Comprehensive performance understanding is often derived by monitoring multiple metrics throughout deep learning experiments.

In addition to output result evaluations, deep learning studies frequently incorporate significant analyses. Ablation studies analyze the impact of removing or modifying specific components or inputs in a model to discern their contribution to overall performance. The aim is to pinpoint crucial elements for a given task and gain insights into the model's behavior.

Parameter sensitivity analysis delves into how changes in model hyperparameters influence performance. Given the numerous hyperparameters in deep learning models, this analysis aids in fine-tuning for optimal performance and provides insights into their relative importance. Techniques such as grid search or random search systematically explore hyperparameter space.

Statistical tests are indispensable for drawing reliable conclusions from experimental results. They help validate whether the proposed model outperforms baseline methods on a statistically significant level, especially when model performance's attribution to the model itself or dataset affinity is ambiguous. Common statistical tests include t-tests, chi-squared tests, Analysis of Variance (ANOVA) tests, and more.

6.1.2 Necessity of Domain-specific GNNs

Deep learning researchers consistently strive to create general models with broad applicability across diverse domains. In computer vision (CV), efforts are directed towards developing models capable of recognizing objects from various sources [He et al., 2016]. These models undergo training on extensive datasets encompassing diverse visual objects. By extracting distinctive features, they are intended for deployment in various vision-related tasks. Similarly, in natural language processing (NLP), pre-trained language models like BERT [Devlin et al., 2018] acquire knowledge of word semantics, grammar rules, and contextual information from vast corpora containing billions of sentences. The aim is to train a versatile model that can be effectively employed in many language-related tasks.

However, in graph models, researchers tend to diverge from pursuing one-size-fits-all solutions and lean towards developing domain-specific models. The performance of GNNs is contingent not only on the analysis of graph structures but also on the input feature vectors. There are scenarios where datasets from disparate domains exhibit similar graph structures while possessing significantly distinct node and edge features. As illustrated in Fig. 6.3, graph-structured data from molecular structures, railway networks, and computer networks may share a common ring-like graph structure but yield entirely different outcomes due to their domain-specific features. It necessitates the creation of specialized GNNs tailored for specific domains.

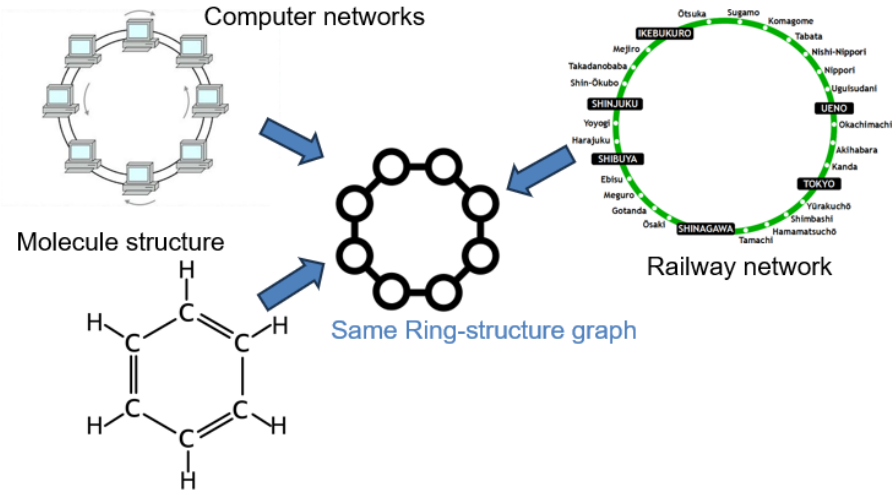


Figure 6.3: Datasets from different domains may exhibit the same graph structure. The railway network of the Yamanote Line in Tokyo, the molecular structure of Benzene, and interconnected computers forming a circular network all share the same ring-structured graph topology, while the domain-specific node and edge features vary across these disparate datasets.

Expert GNNs prove advantageous in addressing tasks within particular contexts more effectively than generalized models designed to fit all scenarios.

Numerous existing studies have dedicated efforts to developing specialized GNNs tailored for distinct scenarios. These prior models often exploit different graph types to encapsulate the distinctive characteristics present in diverse datasets, such as homogeneous, heterogeneous, static, dynamic, and more. In this thesis, we contribute novel GNN frameworks that harness domain-specific knowledge to deliver optimal predictions within their designated contexts. Our approach is aligned with the trend in the field, recognizing the importance of expert GNNs in addressing the nuanced requirements of specific situations and datasets. By introducing these frameworks, we seek to enhance the adaptability and performance of GNNs in diverse domains, showcasing the potential of domain-specific models to outperform generic counterparts in specific scenarios.

6.1.3 Methodologies for Incorporating Domain-specific Knowledge into GNNs

In light of the considerations above, enhancing GNN capabilities is achievable by incorporating domain-specific knowledge into the architectural design of novel GNN models. Unfortunately, many existing expert GNNs have been introduced independently in various domains, lacking a comprehensive review from a general perspective. To address this gap, this thesis provides a consolidated overview of several unified methodologies for crafting domain-specific GNNs. This endeavor addresses one of our research questions: how to design expert GNNs for prediction tasks

within specific domains? By synthesizing and generalizing these methodologies, we contribute to the foundational understanding of effectively designing expert GNNs optimized for specific application areas.

Understanding the domain background knowledge and data features

A crucial aspect of designing domain-specific GNNs is a comprehensive understanding of the background knowledge and a clear definition of the task within specific domains. This foundational knowledge is the basis for selecting the most pertinent data features. For example, in Chapter 3, we construct a bipartite GNN framework for predicting EMS demand. Here, a thorough exploration of the background of emergency medical services informs our approach to demand prediction tasks. This knowledge enables us to identify and collect prominent data features related to regions and hospitals, directly contributing to the enhanced performance of the model.

Incorporating domain knowledge in GNN frameworks

Selecting the most suitable graph representation for the data is a critical step in the design of GNNs. When dealing with non-graph-structured data, a key challenge is to transform the data into a graph format. This transformation involves defining nodes and edges, and associating features with each node and edge. Critical considerations include data characteristics such as the presence of a time factor, node/edge types, directed or undirected relationships, and weighted edges. Section 6.1.1 provides detailed insights into selecting appropriate GNNs based on these considerations.

In cases where existing GNN architectures prove inadequate, developing a custom architecture tailored to domain-specific tasks becomes necessary. For example, in Chapter 4, we introduce a multi-layer temporal GNN framework designed for predicting popularity trends in social media networks. By manually exploring distinct relationships in each dataset, we tailor the input layers of the proposed GNN frameworks to align with the specific characteristics of diverse domains. This tailored approach significantly contributes to achieving optimal performance across various prediction tasks.

Purposeful optimization towards characteristics of prediction targets

Setting appropriate prediction targets is a crucial aspect of optimizing model performance, as it directly influences the model's effectiveness in achieving desired outcomes. To attain optimal results for specific tasks, purposeful optimization strategies must be employed, considering the characteristics of the prediction targets. These strategies encompass optimizing label distributions, addressing imbalances in original datasets, fine-tuning essential hyperparameters, and more.

In Chapter 4, we observe that prediction target values often exhibit significant variations in magnitudes, ranging from 1 to over 10 million in some cases, with a highly biased distribution. Predicting target values across such a broad and imbalanced distribution poses challenges for the proposed model, potentially resulting in significant biases in the prediction results. To mitigate this challenge, we implement a transformation on the prediction target values, converting them into percentages of changes. Consequently, the proposed model predicts the percentage change in popularity degree values rather than the original values, narrowing the range to $[-1, 2]$ after eliminating outliers. This optimization strategy enables more accurate predictions within a smaller range, mitigating the negative impact of vast target value magnitudes.

Additionally, in Chapter 5, we identify a highly imbalanced dataset where positive prediction labels constitute only a tiny fraction of all labels in the original live streaming dataset. This imbalance can confuse the model and lead to poor performance. We employ several effective strategies to address this issue, including filtering the original data, fine-tuning during graph generation, undersampling majority samples during training, and utilizing cost-sensitive loss functions. These purposeful strategies toward data imbalance yield significant improvements in model performance.

6.2 Relationships Among Our Proposed GNNs

In this section, we elucidate the relationships within the three subworks introduced in Chapter 3, Chapter 4, and Chapter 5. First, we explain the coherence among the three subworks. Next, we clarify their commonalities and distinctions. Finally, we conclude by summarizing the challenges and contributions of each work.

6.2.1 Coherency of Subworks

Fig. 6.4 illustrates an intuitive flow highlighting the connections among the three subworks. The three subworks are linearly evolving and improving, progressing from less general and simpler cases to more challenging, general, and complex cases. Each subsequent work addresses the limitations of the previous one and extends the proposed model to adapt to more practical situations.

First, in Chapter 3, we propose a BiGCN model for efficiently learning entity representations within static bipartite graphs, a typical structure frequently observed in real-world datasets. The application of the BiGCN model extends to the task of predicting the EMS demand in the Tokyo metropolitan area, serving as a meaningful metric for public health outcomes and representing a vital aspect of human behaviors in social infrastructure services.

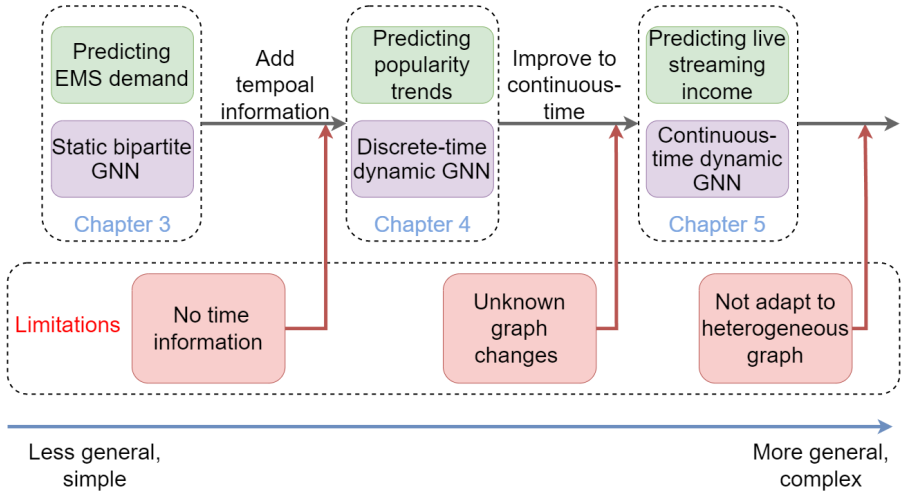


Figure 6.4: The connections among three subworks in this thesis. The three works are linearly evolving and improving, progressing from less general and simpler cases to more challenging, general, and complex cases. Each subsequent work addresses the limitations of the previous one and extends the previous model to adapt to more practical situations.

Subsequently, in Chapter 4, we enhance the proposed BiGCN model by incorporating temporal information to process timed sequences of graph snapshots. The improved multi-layer temporal GNN, an extension of the BiGCN model, is applied to predict popularity trends in social media networks such as X, Instagram, and Reddit. This application provides valuable insights into the future trajectory of public interests and concerns, indicating the social trends in the future.

Finally, in Chapter 5, we propose a novel GNN model to learn representations in homogeneous continuous-time dynamic graphs. This model addresses limitations identified in the previously discussed DTDG framework from Chapter 4. Moreover, it extends the earlier multi-layer temporal GNN framework to accommodate a more practical and general dynamic graph representation style. To illustrate its effectiveness, we conduct a case study predicting income trends in online live streaming services (e.g., YouTube Live and Twitch). Live streaming services have been booming as a style of individuals studying, working, and earning a livelihood through online streaming, particularly in the post-COVID-19 era. It reflects a new combination of evolving human behaviors and emerging social trends.

6.2.2 Commonalities and Distinctions

Table 6.1 demonstrates the commonalities and distinctions among our proposed GNN frameworks. These models employ different strategies tailored to their specific tasks. Despite the distinct nature of the tasks, there are common approaches aimed at enhancing performance by customizing the default unified frameworks of GNNs.

BiGCN and Multi-layer Temporal GNN are both designed to handle heteroge-

Table 6.1: Architecture distinctions among our proposed GNNs.

	BiGCN	**Multi-layer Temporal GNN**	**TDGNN**
Node/Edge classes	Bipartite	Heterogeneous	Homogeneous
Temporal information	Static	DTDG	CTDG
Tasks	Edge label classification	Dynamic node attribute regression	Dynamic node label classification
Aggregator	Bipartitie graph convolution	Graph convolution + BiGCN	Temporal Difference
Encoder	Hadamard product	Multi-head attention	Multi-head attention
Classifier	SVM	MLP	Ensembled MLPs + Softmax
Loss function	Hinge loss	MSELoss	BCELoss

neous graphs. A common finding in both models involves the separate treatment of different types of nodes and edges during node aggregation. BiGCN achieves this by employing two graph convolution layers for distinct node groups, while Multi-layer Temporal GNN extends this concept with a multi-layer architecture to accommodate various relationships. Such separate treatments efficiently analyze the influence of "message passing" within multiple node and edge types. Experimental results demonstrate that the notion of separate processing contributes to the improved performance of these GNN frameworks.

On the other hand, multi-layer temporal GNN and TDGNN are designed for dynamic graphs. The critical components in these models include multi-head attention layers and MLP decoders. This combination draws inspiration from the Transformer architecture [Vaswani et al., 2017] and has proven effective in learning dynamic node embeddings across various dynamic GNN frameworks [Rossi et al., 2020; Wang et al., 2021; Xu et al., 2020]. The integration of attention mechanisms into GNN architectures has expanded the capabilities of GNN researchers for handling graphs in dynamic scenarios. Multi-layer temporal GNN and TDGNN have the similar multi-head attention encoder and skip-connections in learning temporal node embedding based on the aggregated neighbor node information, newly updated node attributes, and the most recent updated temporal node embeddings.

6.2.3 Challenges and Contributions of Subworks

The three subworks effectively tackled specific challenges, each making notable contributions. These achievements are elaborated upon in detail below:

Challenges in work 1

A bipartite graph is a distinctive and frequently employed subtype of heterogeneous graphs characterized by edges exclusively between nodes of two distinct types. This specialized structure is precious in scenarios where relationships exist solely between entities of different classes. However, the plain GNNs, such as GCN and GAT, perform poor efficiency in bipartite graphs. Why are such methods not directly adaptable, and how can we adjust the architecture of GNNs to be adapted to bipartite situations?

Remarkable contribution 1

We proposed the first graph convolutional neural network model adopted to static bipartite graphs. An application for predicting the EMS demand in the Tokyo metropolitan area proves the efficiency and accuracy of the proposed model.

In Chapter 3, we introduce a BiGCN model strategically designed to leverage the multi-modal features inherent in the data to learn comprehensive node embeddings. A noteworthy observation emerges from our investigations, revealing the inadequacies of traditional GNNs when applied to bipartite graphs. Specifically, traditional GNNs confound information from the two disjoint node sets, impeding their effectiveness. To overcome this limitation, our proposed BiGCN model innovatively addresses this challenge by separating the convolution operation for the two distinct node sets, respectively, thereby surmounting the limitations of conventional GCN.

The empirical validation of our approach showcases compelling results, with accuracy ranging from $77.3\% - 87.7\%$ in the label prediction task. This performance surpasses baseline traditional machine learning algorithms and statistical models and outshines the latest graph-based methods. To further evaluate the real-world feasibility of our model, we conduct tests by a pertinent case study: predicting EMS demand in the Tokyo metropolitan area. The success of our BiGCN model in this practical scenario underscores its applicability in addressing complex, real-world challenges. It signifies a promising step forward in the realm of both GNN research and public health management.

Challenges in work 2

Entities and relationships in real-world situations are constantly changing with time. One way to model the dynamics is to record the graph status within regular time intervals and concatenate those graph snapshots in time order, namely the DTDG. Such sequences of graph snapshots have the potential to do predictive analytics by exploiting the present and past information. However, the real-world dataset is not only dynamic but also heterogeneous in most cases. There is usually more than one kind of entity and more than one kind of relationship within entities. Given

that nodes denote entities, and edges denote the relationships within entities, how to represent the multiple node types and multiple kinds of relationships via GNN? Besides, how to incorporate the temporal information into the predictive analytics?

Remarkable contribution 2

We introduce an advanced multi-layer dynamic GNN framework designed to learn entity representations and predict trends within complex social media networks. This framework addresses the challenging task of forecasting trajectories of public concerns and interests based on extensive real-world datasets.

In Chapter 4, we emphasize the imperative of extending BiGCN to enhance its versatility in dynamic, real-world settings. Based on this foundation, this chapter presents a novel multi-layer dynamic GNN framework. We address the limitations of previous research, which focuses too much on selecting characteristics and historical statistics and often neglects the latent influence of relationships among entities. The proposed approach presents its novelty in considering the latent influence of information diffusion in social networks and efficiently mastering the representations of entities, resulting in a substantial performance enhancement. This framework is specialized to proficiently learn temporal node representations within intricate heterogeneous graphs, thereby addressing the limitations associated with static bipartite graphs.

Noteworthy is the adaptability of the proposed method, which transcends the confines of static bipartite graphs. It is tailored to handle dynamic heterogeneous graphs characterized by several multi-layer graphs, particularly forming a sequence of graph snapshots. To assess its practicability, the proposed method undergoes rigorous evaluation across four popularity trend prediction tasks, employing real-world social media datasets. The experimental results underscore the superiority of our method, surpassing various baselines, including traditional regression approaches, prior trend prediction methods, and alternative heterogeneous GNN models. This chapter marks a significant stride in advancing the capabilities of GNNs, particularly in the context of dynamic and heterogeneous graph structures, and sets the stage for further exploration in dynamic graph representation learning.

Challenges in work 3

The DTDG is a simple way to model dynamic graphs. However, a significant shortcoming arises in prediction tasks due to unknown information between consecutive graph snapshots, leading to potential inefficiencies in prediction results. To address this limitation, a more general CTDG style is proposed, which records each change in the graph and its timestamp. The graph actions, with their associated timestamps, serve as a comprehensive representation of the continuous evolution of the

dynamic graph. Prior static GNN methods can not be directly adapted to such a new graph representation style. The research on CTDG is still in an early stage. Therefore, there is an urgent demand to develop a new model that efficiently learns the node representations in CTDG.

Remarkable contribution 3

First, we propose an algorithm that encodes the non-graph-structured data into a continuous-time dynamic graph. Then, we develop a novel dynamic GNN model that learns continuous-time temporal node representations. Also, we present the first analysis and prediction of real-time donation income in live streaming services. It is a significant and innovative topic as it has the potential to help live streaming services gain more popularity and increase profits.

Chapter 5 commences by elucidating how real-world data can be represented in a computable CTDG format. Subsequently, we introduce a novel model termed TDGNN, expressly designed to learn node temporal embeddings from CTDG graphs. Notably, our TDGNN model can predict real-time graph events, answering questions such as when an edge will emerge between specific nodes or how frequently a particular node will change status in the next period. Moreover, the model addresses imbalanced situations commonly encountered in real-world scenarios, incorporating several imbalanced learning strategies to enhance learning in minority classes.

The feasibility in real-world situations lies in evaluating the TDGNN model, focusing on the challenging task of predicting real-time donation income in live streaming services. Extensive experiments conducted on three live-streaming video datasets demonstrate the efficiency and robustness of our proposed model. TDGNN outperforms other baseline methods from various fields, providing more efficient and precise predictions of both the donation posters and the exact timing when donations will appear. Moreover, our TDGNN model proves its efficiency in timeliness, giving prediction results faster than other state-of-the-art continuous-time dynamic GNNs. This chapter signifies a substantial contribution to the promising field of learning on continuous-time dynamic graphs, showcasing the potential of TDGNN in real-world applications.

6.3 Evaluations of BiGCN and TDGNN Model on Other Datasets

In Chapter 3 and Chapter 5, we introduced the innovative BiGCN and TDGNN models tailored for predicting EMS demand and channel incomes in live streaming services, respectively. BiGCN exhibits proficiency in handling bipartite graphs, while TDGNN is adept at addressing CTDGs. However, the empirical validation

Table 6.2: Statistics of the bipartite graph datasets.

Items	Cora	Citeseer
# Edges	2,314	1,665
# Nodes U	789	742
# Nodes V	1312	1,237
# Features U	1,433	3,703
# Features V	1,433	3,703
Data split	80%-10%-10%	80%-10%-10%

Table 6.3: Fine-tuned common hyperparameters in the baseline methods.

	BiNE	BGNN	Metapath2vec	HetGNN	HAN	BiGCN
Learning rate	0.01	0.001	0.0005	0.001	0.005	0.001
Weight decay	−−	0.0005	0.005	0.0	0.001	0.5
Dropout	−−	0.2	0.4	0.5	0.6	0.2
Hidden dim.	128	48	24	128	128	50
# Layers	−−	2	3	3	1	3
# Neighbor sampling size	4	−−	7	23	−−	−−

of these models was limited to specific datasets, focusing on EMS and live streaming chat scenarios. To establish the broader applicability of BiGCN and TDGNN across diverse real-world cases, this section undertakes a comprehensive evaluation by assessing their performance on additional benchmark datasets featuring bipartite and continuous-time dynamic graph structures. This extensive analysis provides insights into their adaptability to various graph-based prediction tasks beyond the initial contexts of EMS demand and live streaming income prediction.

6.3.1 BiGCN on other Bipartite Graph Datasets

In this section, we evaluate the performance of the BiGCN model proposed in Chapter 3 on additional bipartite graph datasets. Table 6.2 presents the statistics of the datasets used in these experiments.

Cora and **Citeseer** [He et al., 2019] are benchmark synthetic bipartite graph datasets generated from citation networks. In these datasets, documents and citation links between them are considered as nodes and undirected edges, respectively. To create bipartite structures, we follow the process in prior studies that randomly partition the nodes into two sets and remove all edge connections between nodes belonging to the same set. Additionally, we ensure that the node features in the two sets are distinct. This experiment aims to assess the adaptability and generalization of the BiGCN model across diverse bipartite graph scenarios beyond the EMS domain.

We assess the BiGCN model on a link prediction task with strong baselines introduced in Chapter 3, namely BiNE, BGNN, Metapath2vec, HetGNN, and HAN. The

Table 6.4: Results in % for link prediction on Cora and Citeseer. Standard deviations over 10 random seeds are shown in parentheses. The best results are marked in bold, and the second bests are marked with underline.

Model	Dataset			
	Cora		Citeseer	
	Acc.	F1.	Acc.	F1.
BiNE	73.6 (0.9)	73.7 (0.5)	60.7 (0.2)	60.9 (0.1)
BGNN	80.4 (0.9)	83.3 (0.6)	70.1 (0.1)	71.5 (0.1)
Metapath2vec	80.2 (0.8)	78.4 (0.7)	65.6 (0.8)	66.2 (1.4)
HetGNN	81.5 (0.3)	83.7 (0.2)	71.4 (0.1)	72.0 (0.2)
HAN	**86.8 (0.1)**	**86.2 (0.3)**	**73.8 (0.1)**	73.9 (0.1)
BiGCN	82.7 (0.1)	82.7 (0.1)	72.8 (0.3)	**74.4 (0.3)**

links in datasets are split into training (80%), validation (10%), and test (10%) sets. To ensure a fair comparison, we fine-tune the hyperparameters through manual grid search while testing the performance of the baseline models. The shared hyperparameters involved in these experiments are detailed in Table 6.3. The experiments are carried out on a platform equipped with an Intel Xeon Platinum 8360Y CPU and an NVIDIA A100 for NVLink 40GiB HBM2 GPU.

Table 6.4 presents the accuracy and f1-scores in % with Standard Deviations (over 10 random seeds) for the link prediction task on the Cora and Citeseer datasets. Notably, our BiGCN model achieves the best result in one case and secures the second-best performance in two instances, narrowly surpassed by HAN. Neural network models, including BGNN, HetGNN, HAN, and BiGCN, consistently outperform random-walk-based methods such as BiNE and Metapath2vec. It underscores the effectiveness of leveraging neural networks for learning node embeddings in bipartite graphs. Comparing BiGCN to the robust baseline HAN, the noteworthy performance of HAN can be attributed to its utilization of attention mechanisms. Despite lacking crucial input information, namely edge weights, BiGCN still demonstrates strong performance, securing the second-best results. It underscores its adaptability and extensibility to various real-world bipartite graphs beyond the EMS dataset.

6.3.2 TDGNN on other CTDG Datasets

In this section, we evaluate the performance of the TDGNN model proposed in Chapter 5 on additional CTDG datasets. Table 6.5 provides an overview of the key statistics for each dataset.

The **Wikipedia** dataset[1] [Kumar et al., 2019] is a bipartite temporal graph encompassing approximately 9,300 nodes and 160,000 temporal edges over one month.

[1]https://snap.stanford.edu/jodie/

Table 6.5: Statistics of the CTDG datasets.

Items	Wikipedia	Reddit
# Edges	157,474	672,447
# Nodes	9,227	10,984
# Feature dim.	172	172
Timespan	30 days	30 days
Data split	70%-15%-15%	70%-15%-15%
Label type	editing ban	posting ban

Table 6.6: Fine-tuned common hyperparameters in the baseline methods.

	TGAT	TGN	APAN	TDGNN
Learning rate	1×10^{-4}	3×10^{-4}	1×10^{-4}	1×10^{-4}
Dropout	0.1	0.1	0.1	0.2
Hidden dim.	100	100	172	172
# Attention heads	2	2	2	2
# GNN encoder layers	2	2	2	2
# MLP decoder layers	2	3	2	2

In this dataset, nodes represent users and wiki pages, while interaction edges signify a user editing a page. The dynamic labels denote whether a user is banned from posting, associated with a timestamp of the actions.

The **Reddit** dataset[2] [Kumar et al., 2019] also constitutes a bipartite temporal graph capturing user interactions over one month. With around 11,000 nodes and 700,000 temporal edges, this dataset reflects user engagement with subreddits through posts. The dynamic binary labels indicate whether a user is banned from posting under a particular subreddit.

For both the Wikipedia and Reddit datasets, the construction of the graph involves selecting the top popular items and the most active users. The textual features of user edits are converted into 172-dimensional Linguistic Inquiry and Word Count (LIWC) feature vectors. The datasets are split into training (70%), validation (15%), and test (15%) sets based on the interaction timestamps.

In this evaluation, we assess the performance of the TDGNN model proposed in Chapter 5 on a dynamic node label prediction task against robust baselines, namely TGAT, APAN, and TGN. To ensure a fair comparison, we fine-tune the hyperparameters through manual grid search while testing the performance of the baseline models. The shared hyperparameters involved in these experiments are detailed in Table 6.6. The experiments are carried out on a platform equipped with an Intel Xeon Platinum 8360Y CPU and an NVIDIA A100 for NVLink 40GiB HBM2 GPU, ensuring consistency in the experimental environment.

[2]https://snap.stanford.edu/jodie/

Table 6.7: AUC score in % for node label prediction tasks on Wikipedia and Reddit datasets. A Standard deviations over 10 random seeds are shown in parentheses. The best results are marked in bold, and the second bests are marked with underline.

Model	Dataset	
	Wikipedia	**Reddit**
TGAT	83.6 (0.7)	65.5 (0.7)
TGN	88.5 (0.3)	**68.6 (0.7)**
APAN	**89.8 (0.3)**	65.3 (0.4)
TDGNN	87.8 (0.3)	67.0 (0.6)

Due to the skew of label distribution, we employ the AUC score as the metric. Table 6.7 shows the AUC scores in % with Standard Deviations (over 10 random seeds) for node label prediction tasks on the Wikipedia and Reddit datasets. Our TDGNN model obtains the second-best performance on the Reddit dataset but only gets third place on the Wikipedia dataset. TGN achieves the overall best results, followed by APAN and TDGNN, and TGAT in the lowest position. Notably, the gap between the poorest and the best models is only 6.2% on Wikipedia and 3.1% on Reddit, indicating they are close in performance. TDGNN demonstrates similar performance as other strong baselines, showcasing its adaptability and extensibility to various real-world bipartite graphs beyond the live streaming dataset.

In conclusion, according to the two additional experiments, we find that the BiGCN and TDGNN models require edge weights as a significant input. Hence, they perform well on datasets like EMS and YouTube live streaming that include edge weights. However, they perform less effectively on citation network datasets that lack edge weights.

In another hand, other baseline GNN models are designed as general models that apply to various graph-structured datasets. However, not every dataset includes edge weights. As a result, these models do not consider the presence or absence of edge weights in the data because their architectures do not incorporate edge weights in computation. They neither require edge weights as mandatory inputs nor are their performance results significantly affected by the presence or absence of edge weights.

6.4 What Kind of Real-world Prediction Tasks Can Be Addressed by Our Proposed GNNs

The scope of human behaviors and social trends is so broad that it is challenging for our proposed GNN frameworks to cover all diverse real-world prediction tasks. It is essential to clarify the types of real-world prediction tasks that can be addressed

Table 6.8: Requirements on real-world data that proposed GNNs can be applied to.

	BiGCN (Chapter 3)	**Multi-layer temporal GNN** (Chapter 4)	**TDGNN** (Chapter 5)
Nodes	Two node types	Multiple	Single
Edges	Single, weighted, only connect different types of nodes	Multiple	Single, weighted
Time-factors	No	Changes regularly observed (Discrete-time)	Changes at anytime (Continuous-time)
Learned embeddings	Node & Edge embeddings	Temporal node embeddings	Temporal node embeddings
Prediction targets	Edge labels (Classification)	Numerical values of node attributes (Regression)	Dynamic node labels & timing (Classification)

by our proposed GNN frameworks. Table 6.8 outlines the characteristics of data to which our proposed GNNs can be applied.

- The BiGCN developed in Chapter 3 is specifically designed to tackle edge label prediction tasks on bipartite graphs. Therefore, it necessitates a dataset with only two types of node groups and a single weighted edge type that solely connects two nodes of different types. Additionally, BiGCN operates optimally in a static system without any alterations regarding node and edge attributes. Finally, the BiGCN learns node and edge embeddings and predicts edge labels, but it can be adapted to other classification tasks by customizing the encoders and decoders.

- The Multi-layer temporal GNN proposed in Chapter 3 is tailored for predicting dynamic node attribute values in heterogeneous DTDGs. As such, it necessitates datasets with multiple node types and multiple edge types, where edges connect nodes of the same type and between nodes of different types, respectively. Additionally, regular observations at discrete time intervals are essential for this framework, as the model input comprises a timed sequence of graph snapshots with regular intervals. Finally, this model learns temporal node embeddings and tackles the regression task of predicting numerical values of node attributes.

- The TDGNN model introduced in Chapter 5 is tailored for predicting dynamic node labels in homogeneous CTDGs. It operates optimally with datasets featuring a single node type and a single edge type associated with edge weights. TDGNN incorporates continuous time-factors that can be observed at any time. Lastly, TDGNN learns temporal node embeddings and predicts dynamic node labels and the timing of when node labels change.

In summary, our proposed GNN frameworks have demonstrated practical applicability in addressing specific real-world prediction tasks. Each model is designed to accommodate the unique characteristics of the data it handles, allowing it to effectively capture and leverage relevant information for making accurate predictions. By tailoring our models to suit the specific requirements of each task, we have been able to achieve notable success in various domains.

While our models have effectively addressed limited real-world prediction tasks, there is still room for further exploration and refinement. Future research may focus on extending and adapting our frameworks to handle a broader range of tasks and datasets, and explore innovative techniques for improving model performance and scalability. Overall, our work highlights the potential of GNNs in addressing real-world prediction challenges and underscores the importance of tailored-modeling approaches in achieving success in human behaviors and social trends.

Chapter 7

Conclusion

Predictive analytics in human behaviors and social trends is pivotal for gaining invaluable insights into the future trajectories of public concerns and interests. Graphs serve as an ideal tool to model the intricate relationships within real-world entities, especially in social activities. The primary goal of this thesis is to design novel and advanced GNN models that efficiently capture relationships within entities engaged in social activities while simultaneously learning entity representations and then address the challenging topics in human behaviors and social trends. GNN models in the preceding chapters showcase remarkable versatility, seamlessly adapting to scenarios ranging from simple static homogeneous networks to intricate dynamic heterogeneous structures.

One of the primary focuses of our thesis is to reveal the dynamic nature of social interactions and evolving relationships over time. This emphasis allows our models to comprehensively understand complex social phenomena, enabling more accurate predictions and valuable insights. In this concluding chapter, we comprehensively summarize the key contributions made throughout the thesis. Additionally, we acknowledge the limitations inherent in our proposed GNN methods, offering insights into potential future research directions that could further enhance the capability of GNNs. By concluding these aspects, we hope to contribute to the ongoing dialogue surrounding developing and deploying GNNs in understanding human behaviors and social trends.

7.1 Answers to Research Questions

In this section, we formally answer to the research questions proposed in Section 1.4.3.

1. *How to address prediction tasks on heterogeneous data by GNNs?*

   BiGCN (Chapter 3) and Multi-layer Temporal GNN (Chapter 4) are both designed to handle heterogeneous graphs. A shared strategy between these

two models involves the separate treatment of different types of nodes and edges during node aggregation. BiGCN achieves this by employing two graph convolution layers for distinct node groups, while Multi-layer Temporal GNN extends this concept with a multi-layer architecture to accommodate various relationships. Experimental results demonstrate that the notion of separate processing contributes to the improved performance of these GNN frameworks.

2. *How to address prediction tasks on dynamic data by GNNs?*

   The representation of temporal dynamics in graphs is categorized into discrete-time dynamic graphs (DTDGs) and continuous-time dynamic graphs (CTDGs). We employ recurrent time-sequence units in the proposed GNN framework to handle the temporal information within DTDGs (Chapter 4). In another hand, we develop a real-time node status updating mechanism that allows for the simultaneous updating of node status within CTDGs (Chapter 5). Besides, multi-head attention layers and MLP decoders are critical common components in proposed dynamic GNN frameworks. This combination draws inspiration from the Transformer architecture and has proven effective in learning dynamic node embeddings.

3. *How to tailor GNN architectures to better fit domain-specific prediction tasks?*

   Our proposed GNNs are enhanced by incorporating domain-specific knowledge into the architectural design of GNN frameworks. We conclude a unified process of addressing prediction tasks by using GNNs, incorporating data preprocessing, graph construction, GNN framework design, and performance evaluation. Each step is associated with unique methodologies of incorporating the domain-specific knowledge into GNN designs, such as understanding the domain background knowledge and data features, incorporating domain knowledge in GNN frameworks, and employing purposeful optimization strategies toward characteristics of prediction targets. By synthesizing and generalizing these methodologies as a unified process, we contribute to the foundational understanding of effectively designing expert GNNs optimized for specific application areas.

7.2 Key Contributions

In this thesis, we propose a spectrum of GNN models designed to understand a variety of real-world human behaviors and social trends situations. The thesis begins in Chapter 1 with an introduction to the evolution of AI techniques in predictive analytics concerning human behavior and social trends. Subsequently, we delve into GNNs and elucidate why they are recognized as one of the most essential and

fastest-growing fields in deep learning, as well as one of the most powerful tools of analyzing real-world data. Chapter 2 provides a summary of preliminary topics that serves as the foundation for the work described in subsequent chapters.

In Chapter 3, we explore a critical static scenario, namely EMS demand prediction. We are the first to analyze EMS demand at the hospital-region level in a metropolis like Tokyo. We analyze the limitations of convolutional GNNs in bipartite graphs and propose BiGCN, a novel approach adept at learning the node embeddings of binary entity groups by fully considering the structure characteristics of bipartite graphs. Based on the experimental observations, BiGCN significantly enhances the accuracy of EMS demand predictions, underscoring its efficiency in capturing nuanced patterns within static bipartite graph-structured datasets. BiGCN is not limited to the hospital-region bipartite graph in this paper but can potentially become a general model for accomplishing supervised learning tasks in non-specific bipartite graphs.

Expanding our focus to dynamic scenarios, Chapter 4 delves into the challenges of learning in dynamic graphs. Recognizing the heightened difficulty of dynamic settings, we propose a temporal GNN framework tailored for DTDG, which consists of a timed sequence of graph snapshots. This novel approach addresses the limitations of prior research, which predominantly focuses on selecting characteristics and historical statistics within social media networks and employing simplistic machine learning or statistical models as encoders and decoders, resulting in suboptimal learning efficiency. The proposed approach presents its novelty in considering the latent influence of information diffusion in social networks while efficiently learning the representations of target entities, resulting in a substantial performance enhancement. We assess the efficiency of the proposed approach through experiments conducted on real-world social network datasets. Based on the experimental observations, the proposed approach consistently outperforms various baseline methods, showcasing its feasibility in real-world situations. This work tackles the crucial task of forecasting trajectories of public concerns and interests based on extensive real-world data, contributing valuable insights to the social trends.

Chapter 5 extends our dynamic GNN approach to CTDG by addressing the limitations of unknown information between consecutive graph snapshots within DTDG, which is likely to lead to potential inefficiencies in prediction results. Firstly, we introduce an algorithm that encode the non-Euclidean-structured data into a style of CTDG. By developing a novel TDGNN model, we can predict real-time graph events and efficiently learn node temporal embeddings within CTDGs. Notably, TDGNN exhibits robustness in addressing imbalanced situations commonly encountered in real-world scenarios, showcasing its versatility in handling diverse challenges. In addition, we present the first analysis and prediction of real-time donation income in live streaming services. It is a significant and innovative topic as live streaming

has become one of the most popular entertainment way and work style nowadays, and it has the potential to help live streamers gain more popularity and increase profits. To achieve this, firstly, we propose an approach representing live streaming chat messages and viewer interactions as a continuous-time dynamic graph. Then, we transform the income prediction problem into a dynamic node label classification problem in CTDG. We utilize our proposed TDGNN model that learns temporal viewers' and streamers' representations and predicts real-time donation incomes. We conduct experiments to demonstrate the effectiveness of the TDGNN model, which outperforms various baseline methods regarding both accuracy and timeliness.

Chapter 6 provides some discussions regarding the relationships within the three subworks introduced before. First, we conclude the unified conceptions hidden in the design considerations of proposed GNN frameworks. Then, we clarify the coherency within the three subworks, which shows a linear progress from simpler and less general cases to the more challenging, complex and general cases. In addition, we exhibit the extensibility of the proposed GNNs by assessing them on other benchmark bipartite graphs and continuous-time dynamic graphs. The outstanding experimental results prove that our proposed GNN frameworks are not only designed for particular datasets, but feasible in various domains. Finally, we clarify the requirements of real-world human behaviors and social trends data that can be handled by our proposed GNN methods.

In summary, the key contributions of this thesis lie in developing innovative GNN models that evolve from simple static homogeneous cases to intricate dynamic heterogeneous scenarios, providing practical solutions for various real-world situations. Each method is meticulously designed to tackle the unique challenges of different real-world situations. Our proposed GNN models significantly enhance understanding of complex relationships within evolving systems. This contribution reflects our commitment to advancing the field of GNNs, paving the way for more effective and versatile applications in predictive analytics for human behaviors and social trends.

7.3 Future Work

We explore potential future directions for research and development in the GNN research, considering avenues for more intricate GNN models, data quality enhancement, and scalability associated with real-world applications. By identifying areas where improvements can be made, we contribute to the ongoing evolution of GNN methodologies.

Continuous-time Dynamic Heterogeneous Graphs

In this thesis, we have discussed the GNNs across various graph structures, including static homogeneous graphs, discrete-time dynamic heterogeneous graphs, and

continuous-time dynamic homogeneous graphs. Naturally, the following exploration should be the most intricate case: the continuous-time dynamic homogeneous graph. Although some prior research has already started the learning on continuous-time dynamic homogeneous graphs, the existing research in this domain remains at a nascent stage [Gao et al., 2022b; Sajadmanesh et al., 2019; Zhu et al., 2022]. Consequently, there exists a pressing need for further exploration and discovery of GNNs and their applications for continuous-time dynamic homogeneous graphs. This uncharted territory presents a unique opportunity for future research, promising advancements in understanding and efficiently modeling more intricate relationships within continuously evolving homogeneous graphs.

Missing & Noise Data in Real-world Datasets

Addressing missing data and mitigating noise are critical aspects of handling real-world data, especially in complex scenarios that often involve incomplete or noisy information. The challenges posed by missing data and noise necessitate robust data preprocessing and cleaning strategies, as well as purposefully designed GNNs.

Filling in missing data requires careful consideration of the nature of the missing values and the impact on downstream analyses. Imputation techniques, such as mean or median imputation, regression imputation, or sophisticated machine learning-based methods, can be employed to estimate missing values based on available information. The choice of methods should align with the specific characteristics of the data and the objectives of the analysis. GNNs also show their potential of estimating the missing data and mitigating the negative impact [Gordon et al., 2021; Rossi et al., 2022].

Noise in real-world data can arise from various sources, including measurement errors, outliers, or external interference. Robust statistical techniques, outlier detection algorithms, and data filtering methods can be applied to identify and mitigate the effects of noise. Moreover, employing advanced GNN models resistant to noise or incorporating ensemble methods can enhance the overall robustness of data analyses [Dai et al., 2021; NT et al., 2019].

Scalability of Real-world Datasets

Scalability is a pressing concern as the size and complexity of real-world datasets continue to grow exponentially. In the ever-expanding landscape of GNNs, ensuring the scalability of models becomes imperative to harness the full potential of these networks in large-scale applications. Future research efforts should be dedicated to developing scalable GNN solutions that can adeptly manage massive volumes of data while maintaining computational efficiency.

One avenue for addressing scalability involves exploring parallel and distributed

computing approaches. Leveraging the power of distributed systems and parallel processing can significantly enhance the efficiency of GNN computations, enabling them to handle vast datasets seamlessly. Investigating novel algorithms and model architectures designed for distributed computing environments will be crucial in achieving scalable solutions [Wan et al., 2023].

Additionally, the design of scalable GNNs should consider the dynamic nature of real-world data and evolving relationships. Models should be capable of adapting to changes in the data distribution over time, allowing for continuous learning and scalability in dynamic scenarios. For example, a sample-based training can significantly mitigate the computational cost during scalable GNN training [Serafini and Guan, 2021].

Furthermore, advancements in hardware technologies, including Graphics Processing Units (GPUs) and specialized accelerators, offer opportunities to boost the scalability of GNNs. Research should focus on optimizing GNN implementations to leverage the parallel processing capabilities of modern hardware, ensuring that computations are distributed efficiently across multiple devices.

It is worth noting that although we highlight the scalability issue as a challenge in GNN studies in Section 1.4.2, the subworks included in this thesis do not involve datasets of such a large scale that they cannot be processed by the supercomputer we used. The largest dataset used is the YouTube live streaming dataset, comprising approximately 15 million edges and 40,000 nodes. Although we implement several strategies to mitigate the adverse effects of large-scale input data, such as batch processing, data sampling, and reducing model complexity, we have yet to explicitly address the scalability issue as a research question in this thesis. The studies conducted here have not delved into the actual intrinsic challenges of scalability in GNNs. It explains why we highlight the scalability issue in the challenges but do not mention it in the research questions.

Privacy Issue

The widespread integration of GNNs into real-world applications accentuates the need for a conscientious consideration of privacy issues. As these sophisticated GNN models permeate diverse domains such as healthcare, finance, and social sciences, critical consideration of people's privacy issues becomes paramount. This landscape surrounding GNNs encompasses multifaceted dimensions that warrant careful scrutiny.

Within social sciences, where GNNs contribute to understanding human behavior and social trends, privacy considerations cover issues of consent, data privacy, and the responsible dissemination of research findings. Ensuring that GNNs are applied to respect individuals' rights and avoid perpetuating social biases is essential to privacy issues.

In sensitive domains like healthcare, where GNNs may be employed for tasks such as disease prediction or personalized medicine, the responsible use of these models is imperative. Privacy issues include ensuring patient privacy, securing informed consent, and mitigating potential biases in healthcare data that could impact the accuracy and fairness of GNN predictions.

Financial applications of GNNs, such as fraud detection and investment analysis, introduce privacy concerns related to transparency, accountability, and the potential consequences of algorithmic decision-making. Striking a balance between utilizing GNNs for enhanced efficiency and maintaining privacy standards is crucial to fostering trust in financial systems.

Research efforts in the privacy considerations of GNNs can yield frameworks, guidelines, and standards that guide their responsible deployment. These initiatives include developing transparent and interpretable GNN models, instituting privacy review processes for GNN applications in sensitive domains, and establishing norms for fair and unbiased data representation.

References

Ashwani Kumar Aggarwal. A hybrid approach to gps improvement in urban canyons. *International Journal of Engineering Sciences & Research Technology*, 4(10):358–363.

Mohsen Ahmadi, Saeid Jafarzadeh-Ghoushchi, Rahim Taghizadeh, and Abbas Sharifi. Presentation of a new hybrid approach for forecasting economic growth using artificial intelligence approaches. *Neural Computing and Applications*, 31:8661–8680, 2019.

Mahmuda Akhtar and Sara Moridpour. A review of traffic congestion prediction using artificial intelligence. *Journal of Advanced Transportation*, 2021:1–18, 2021.

Yaniv Altshuler, Wei Pan, and Alex Pentland. Trends prediction using social diffusion models. In *Social Computing, Behavioral-Cultural Modeling and Prediction: 5th International Conference, SBP 2012, College Park, MD, USA, April 3-5, 2012. Proceedings 5*, pages 97–104. Springer, 2012.

Marco Amorim, Sara Ferreira, and António Couto. Road safety and the urban emergency medical service (uems): Strategy station location. *Journal of Transport Health*, 6:60 – 72, 2017.

James E Anderson. The gravity model. *Annu. Rev. Econ.*, 3(1):133–160, 2011.

B Senthil Arasu, B Jonath Backia Seelan, and N Thamaraiselvan. A machine learning-based approach to enhancing social media marketing. *Computers & Electrical Engineering*, 86:106723, 2020.

R. Aringhieri, M.E. Bruni, S. Khodaparasti, and J.T. van Essen. Emergency medical services and beyond: Addressing new challenges through a wide literature review. *Computers Operations Research*, 78:349 – 368, 2017.

Roberto Aringhieri, Giulian Carello, and Daniela Morale. Supporting decision making to improve the performance of an italian emergency medical service. *Annals of Operations Research*, 236:131 – 148, 2016.

Nebojsa Bacanin, Ruxandra Stoean, Miodrag Zivkovic, Aleksandar Petrovic, Tarik A. Rashid, and Timea Bezdan. Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: Application for dropout regularization. *Mathematics*, 9(21), 2021.

Nebojsa Bacanin, Miodrag Zivkovic, Fadi Al-Turjman, K Venkatachalam, Pavel Trojovskỳ, Ivana Strumberger, and Timea Bezdan. Hybridized sine cosine algorithm with convolutional neural networks dropout regularization application. *Scientific Reports*, 12(1):6302, 2022.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

John Bolton. Predicting and managing demand in social care. *American Journal of Medical Research*, 3(2):152–187, 2016.

Bryan Bonnet, Dante Gama Dessavre, Keith Kraus, and Jose Emmanuel Ramirez-Marquez. Optimal placement of public-access aeds in urban environments. *Computers & Industrial Engineering*, 90:269–280, 2015.

Davide Boscaini, Jonathan Masci, Simone Melzi, Michael M Bronstein, Umberto Castellani, and Pierre Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer graphics forum*, volume 34, pages 13–23. Wiley Online Library, 2015.

Sylvain Brohee and Jacques Van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC bioinformatics*, 7:1–19, 2006.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

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

Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. Popularity prediction on social platforms with coupled graph neural networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 70–78, 2020.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun. A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt, 2023.

Nuttapong Chairatanakul, Xin Liu, and Tsuyoshi Murata. Pgra: projected graph relation-feature attention network for heterogeneous information network embedding. *Information Sciences*, 570:769–794, 2021.

Nabil Channouf, Pierre L'Ecuyer, Armann Ingolfsson, and Athanassios N. Avramidis. The application of forecasting techniques to modeling emergency medical system calls in calgary, alberta. *Health Care Management Science*, 10 (1):25–45, 2007.

Priyavrat Chauhan, Nonita Sharma, and Geeta Sikka. The emergence of social media data and sentiment analysis in election prediction. *Journal of Ambient Intelligence and Humanized Computing*, 12:2601–2627, 2021.

Albert Y Chen, Tsung-Yu Lu, Matthew Huei-Ming Ma, and Wei-Zen Sun. Demand forecast using data analytics for the preallocation of ambulances. *IEEE journal of biomedical and health informatics*, 20(4):1178–1187, 2015.

Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. Collaborative similarity embedding for recommender systems. In *TheWebConf*, pages 2637–2643, 2019.

Yunsong Chen, Xiaogang Wu, Anning Hu, Guangye He, and Guodong Ju. Social prediction: a new research paradigm based on machine learning. *The Journal of Chinese Sociology*, 8:1–21, 2021.

Ba Chu and Shafiullah Qureshi. Comparing out-of-sample performance of machine learning methods to forecast us gdp growth. *Computational Economics*, 62(4): 1567–1609, 2023.

Wei-Ta Chu and Yung-Chieh Chou. On broadcasted game video analysis: event detection, highlight detection, and highlight forecast. *Multimedia Tools and Applications*, 76(7):9735–9758, 2017.

Petr Chunaev. Community detection in node-attributed social networks: a survey. *Computer Science Review*, 37:100286, 2020.

Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2018.

Enyan Dai, Charu Aggarwal, and Suhang Wang. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 227–236, 2021.

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

Ivo D Dinov. Data science and predictive analytics. *Cham, Switzerland: Springer*, 2018.

Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. Metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, pages 135–144, 2017.

Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

Kayhan Erciyes. Distributed graph algorithms for computer networks. 2013.

Seyed Amin Fadaee and Maryam Amir Haeri. Classification using link prediction. *Neurocomputing*, 359:395–407, 2019.

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

Shuo Feng, Zebang Feng, Chen Ling, Chen Chang, and Zhongke Feng. Prediction of the covid-19 epidemic trends based on seir and ai models. *PLoS One*, 16(1): e0245101, 2021.

Kaja J. Fietkiewicz, Isabelle Dorsch, Katrin Scheibe, Franziska Zimmer, and Wolfgang G. Stock. Dreaming of stardom and money: Micro-celebrities and influencers on live streaming services. In *Social Computing and Social Media. User Experience and Behavior*, pages 240–253, 2018.

Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200): 675–701, 1937.

Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The annals of mathematical statistics*, 11(1):86–92, 1940.

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

Chao Gao, Junyou Zhu, Fan Zhang, Zhen Wang, and Xuelong Li. A novel representation learning for dynamic graphs based on graph convolutional networks. *IEEE Transactions on Cybernetics*, 2022a.

Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. Bine: Bipartite network embedding. In *SIGIR*, pages 715—724, 2018.

Peng Gao, Weiyong Yang, Haotian Zhang, Xingshen Wei, Hao Huang, Wang Luo, Zhimin Guo, Yunhe Hao, et al. Detecting unknown threat based on continuous-time dynamic heterogeneous graph network. *Wireless Communications and Mobile Computing*, 2022, 2022b.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

David Gordon, Panayiotis Petousis, Henry Zheng, Davina Zamanzadeh, and Alex AT Bui. Tsi-gnn: Extending graph neural networks to handle missing data in temporal settings. *Frontiers in big Data*, 4:693869, 2021.

George Grekousis and Ye Liu. Where will the next emergency event occur? predicting ambulance demand in emergency medical services using artificial intelligence. *Computers, Environment and Urban Systems*, 76:110 – 122, 2019.

Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864, 2016.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

Zhijiang Guo, Yan Zhang, and Wei Lu. Attention guided graph convolutional networks for relation extraction. *arXiv preprint arXiv:1906.07510*, 2019.

Zhiwei Guo and Heng Wang. A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics*, 17(4): 2776–2783, 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034. 2017a.

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

William A Hamilton, Oliver Garretson, and Andruid Kerne. Streaming on twitch: fostering participatory communities of play within live mixed media. In *SIGCHI2014*, pages 1315–1324, 2014.

Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Yanfang Li, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. Bipartite graph neural networks for efficient node representation learning. *arXiv preprint arXiv:1906.11994*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Xiangnan He, Ming Gao, Min-Yen Kan, Yiqun Liu, and Kazunari Sugiyama. Predicting the popularity of web 2.0 items based on user comments. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 233–242, 2014.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

Zhiwen Hou, Yuchen Zhou, Xiaowei Wu, and Fanliang Bu. Attention-based spatial–temporal multi-graph convolutional networks for casualty prediction of terrorist attacks. *Complex & Intelligent Systems*, pages 1–22, 2023.

Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

Ronald L Iman and James M Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, 9(6): 571–595, 1980.

Roberto Interdonato, Andrea Tagarelli, Dino Ienco, Arnaud Sallaberry, and Pascal Poncelet. Local community detection in multilayer networks. *Data Mining and Knowledge Discovery*, 31:1444–1479, 2017.

Mahdi Jalili, Yasin Orouskhani, Milad Asgari, Nazanin Alipourfard, and Matjaž Perc. Link prediction in multiplex online social networks. *Royal Society open science*, 4(2):160863, 2017.

Adele Lu Jia, Yuanxing Rao, and Siqi Shen. Analyzing and predicting user donations in social live video streaming. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 1256–1261, 2021.

Ruidong Jin, Tianqi Xia, Xin Liu, Tsuyoshi Murata, and Kyoung-Sook Kim. Predicting emergency medical service demand with bipartite graph convolutional networks. *Ieee Access*, 9:9903–9915, 2021.

Ruidong Jin, Xin Liu, and Tsuyoshi Murata. Predicting potential real-time donations in youtube live streaming services via continuous-time dynamic graph. In *International Conference on Discovery Science*, pages 59–73. Springer, 2022.

Ruidong Jin, Xin Liu, and Tsuyoshi Murata. Predicting potential real-time donations in youtube live streaming services via continuous-time dynamic graphs. *Machine Learning*, pages 1–35, 2023.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE Access*, 6:1662–1669, 2018.

Amandeep Kaur, Ajay Pal Singh Chauhan, and Ashwani Kumar Aggarwal. Machine learning based comparative analysis of methods for enhancer prediction in genomic data. In *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 142–145, 2019.

G Kavitha, B Saveen, and Nomaan Imtiaz. Discovering public opinions by performing sentimental analysis on real time twitter data. In *International Conference on Management of Data2018*, pages 1–4, 2018.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research*, 21(1):2648–2720, 2020.

Mirza Golam Kibria, Kien Nguyen, Gabriel Porto Villardi, Ou Zhao, Kentaro Ishizu, and Fumihide Kojima. Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. *IEEE Access*, 6:32328–32338, 2018. doi: 10.1109/ACCESS.2018.2837692.

Jina Kim, Kunwoo Bae, Eunil Park, and Angel P del Pobil. Who will subscribe to my streaming channel? the case of twitch. In *Conference Companion Publication of the 2019 on Computer Supported Cooperative Work and Social Computing*, pages 247–251, 2019.

Moon Keun Kim, Yang-Seon Kim, and Jelena Srebric. Predictions of electricity consumption in a campus building using occupant rates and weather elements with sensitivity analysis: Artificial neural network vs. linear regression. *Sustainable Cities and Society*, 62:102385, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907v4*, 2017.

Martijn Koot, Martijn RK Mes, and Maria E Iacob. A systematic literature review of supply chain decision making supported by the internet of things and big data analytics. *Computers & Industrial Engineering*, 154:107076, 2021.

Edward Elson Kosasih and Alexandra Brintrup. A machine learning approach for predicting hidden links in supply chain with graph neural networks. *International Journal of Production Research*, 60(17):5380–5393, 2022.

Raja Krishnamoorthi, Shubham Joshi, Hatim Z Almarzouki, Piyush Kumar Shukla, Ali Rizwan, C Kalpana, Basant Tiwari, et al. A novel diabetes healthcare disease prediction framework using machine learning techniques. *Journal of healthcare engineering*, 2022, 2022.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD2019*, pages 1269–1278, 2019.

Vaibhav Kumar and ML Garg. Predictive analytics: a review of trends and techniques. *International Journal of Computer Applications*, 182(1):31–37, 2018.

Riadh Ladhari, Elodie Massa, and Hamida Skandrani. Youtube vloggers' popularity and influence: The roles of homophily, emotional attachment, and expertise. *Journal of Retailing and Consumer Services*, 54:102027, 2020.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336.

Julien Leblay, Melisachew Wudage Chekol, and Xin Liu. Towards temporal knowledge graph embeddings with arbitrary time precision. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 685–694, 2020.

Yann LeCun and Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, pages 255–258. 1998.

Minhyung Lee, HanByeol Choi, Daegon Cho, and Heeseok Lee. Cannibalizing or complementing?. the impact of online streaming services on music record sales. *Procedia Computer Science*, 91:662–671, 2016.

So-Eun Lee, Mideum Choi, and Seongcheol Kim. They pay for a reason! the determinants of fan's instant sponsorship for content creators. *Telematics and Informatics*, 45:101286, 2019.

Ce Li, Fan Zhou, Xucheng Luo, and Goce Trajcevski. Kernel-based structural-temporal cascade learning for popularity prediction. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021a.

Q. Li, Z. Han, and X.-M. Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*, pages 3538 – 3545, 2018.

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xi-Jie Li, Ming-Fei Wu, Jian Ma, Bo-Ya Gao, Qiu-Lin Wu, Ai-Dong Chen, Jie Liu, Yu-Ying Jiang, Bao-Ping Zhai, Regan Early, et al. Prediction of migratory routes of the invasive fall armyworm in eastern china using a trajectory analytical approach. *Pest Management Science*, 76(2):454–463, 2020a.

Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, page 126441, 2023.

Zhao Li, Xin Shen, Yuhang Jiao, Xuming Pan, Pengcheng Zou, Xianling Meng, Chengwei Yao, and Jiajun Bu. Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1677–1688. IEEE, 2020b.

Zhao Li, Haishuai Wang, Peng Zhang, Pengrui Hui, Jiaming Huang, Jian Liao, Ji Zhang, and Jiajun Bu. Live-streaming fraud detection: A heterogeneous graph neural network approach. In *KDD2021*, pages 3670–3678, 2021b.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

Weiwen Liu, Yin Zhang, Jianling Wang, Yun He, James Caverlee, Patrick PK Chan, Daniel S Yeung, and Pheng-Ann Heng. Item relationship graph neural networks for e-commerce. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4785–4799, 2021.

Adele Lu Jia, Xiaoxue Shen, Siqi Shen, Yongquan Fu, and Liwen Peng. User donations in a user generated video system. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 1055–1062, New York, NY, USA, 2019. Association for Computing Machinery.

Adele Lu Jia, Yuanxing Rao, Hongru Li, Ran Tian, and Siqi Shen. Revealing donation dynamics in social live video streaming. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 30–31, New York, NY, USA, 2020. Association for Computing Machinery.

Jun Luo. Integrating the huff model and floating catchment area methods to analyze spatial access to healthcare services. *Transactions in GIS*, 18(3):436–448, 2014.

Surita Maini and Ashwani Kumar Aggarwal. Camera position estimation using 2d image dataset. *International Journal of Innovations in Engineering and Technology (IJIET)*, 10(2):199–203, 2018.

Samir Malakar, Manosij Ghosh, Showmik Bhowmik, Ram Sarkar, and Mita Nasipuri. A ga based hierarchical feature selection approach for handwritten word recognition. *Neural Computing and Applications*, 32:2533–2552, 2020.

Amil Merchant, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023. doi: 10.1038/s41586-023-06735-9.

Nishchol Mishra and Sanjay Silakari. Predictive analytics: A survey, trends, applications, oppurtunities & challenges. *International Journal of Computer Science and Information Technologies*, 3(3):4434–4438, 2012.

Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, 2007.

Abdel-Basset Mohamed, Abouhawwash Mohamed, Askar Sameh, Hawash Hossam, and Nayyar Anand. An internet on neuro-nano-things based device for voice communication for voice and hearing impaired users, Oct. 26 2023. URL `https://patentscope2.wipo.int/search/en/detail.jsf?docId=DE411700577&_cid=P20-LQBPYR-57176-1`.

Mona Mohamed. Agricultural sustainability in the age of deep learning: Current trends, challenges, and future trajectories. *Sustainable Machine Intelligence Journal*, 4:(2):120, Sep. 2023a.

Mona Mohamed. Empowering deep learning based organizational decision making: A survey. *Sustainable Machine Intelligence Journal*, 3, Jun. 2023b.

Peter R Monge and Noshir S Contractor. *Theories of communication networks*. Oxford University Press, USA, 2003.

Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *NeurIPS*, pages 3697–3707. 2017.

Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018.

Hoang NT, Choong Jun Jin, and Tsuyoshi Murata. Learning graph neural networks with noisy labels. *arXiv preprint arXiv:1905.01591*, 2019.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *KDD*, pages 1105–1114, 2016.

Katherine Payne, Mark J. Keith, Ryan M. Schuetzler, and Justin Scott Giboney. Examining the learning effects of live streaming video game instruction over twitch. *Computers in Human Behavior*, 77:95–109, 2017.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710, 2014a.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD2014*, pages 701–710, 2014b.

M Poongodi, Tu N Nguyen, Mounir Hamdi, and Korhan Cengiz. Global cryptocurrency trend prediction using social media. *Information Processing & Management*, 58(6):102708, 2021.

S Poornima and M Pushpalatha. A survey of predictive analytics using big data with data mining. *International journal of bioinformatics research and applications*, 14 (3):269–282, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*, 2020.

Guoguang Rong, Arnaldo Mendez, Elie Bou Assi, Bo Zhao, and Mohamad Sawan. Artificial intelligence in healthcare: review and prediction case studies. *Engineering*, 6(3):291–301, 2020.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Emanuele Rossi, Henry Kenlay, Maria I Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. In *Learning on Graphs Conference*, pages 11–1. PMLR, 2022.

Dimitrios Rousidis, Paraskevas Koukaras, and Christos Tjortjis. Social media prediction: a literature review. *Multimedia Tools and Applications*, 79(9-10):6279–6311, 2020.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Sina Sajadmanesh, Sogol Bazargani, Jiawei Zhang, and Hamid R Rabiee. Continuous-time relationship prediction in dynamic heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13 (4):1–31, 2019.

Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM2020*, pages 519–527, 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008a.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008b.

Poornima Selvaraj and Pushpalatha Marudappa. A survey of predictive analytics using big data with data mining. *International Journal of Bioinformatics Research and Applications*, 14:269, 01 2018. doi: 10.1504/IJBRA.2018.092697.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*, pages 362–373. Springer, 2018.

Marco Serafini and Hui Guan. Scalable graph neural network training: The case for sampling. *ACM SIGOPS Operating Systems Review*, 55(1):68–76, 2021.

Hubert Setzler, Cem Saydam, and Sungjune Park. Ems call volume predictions: A comparative study. *Computers Operations Research*, 36(6):1843 – 1851, 2009.

Matheus Kempa Severino and Yaohao Peng. Machine learning algorithms for fraud prediction in property insurance: Empirical evidence using real-world microdata. *Machine Learning with Applications*, 5:100074, 2021.

Cuneyt Sevim, Asil Oztekin, Ozkan Bali, Serkan Gumus, and Erkam Guresen. Developing an early warning system to predict currency crises. *European Journal of Operational Research*, 237(3):1095–1104, 2014.

Francisco Silva and Daniel Serra. Locating emergency services with different priorities: the priority queuing covering location problem. *Journal of the Operational Research Society*, 59(9):1229–1238, 2008.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pages 412–422. Springer, 2018.

Max Sjöblom and Juho Hamari. Why do people watch others play video games? an empirical study on the motivations of twitch users. *Computers in human behavior*, 75:985–996, 2017.

Krisjanis Steins, Niki Matinrad, and Tobias Granberg. Forecasting the demand for emergency medical services. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, pages 1855–1864, 2019.

Gabor Szabo and Bernardo A Huberman. Predicting the popularity of online content. *Communications of the ACM*, 53(8):80–88, 2010.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *TheWebConf*, pages 1067–1077, 2015.

Xianfeng Tang, Yozen Liu, Neil Shah, Xiaolin Shi, Prasenjit Mitra, and Suhang Wang. Knowing your fate: Friendship, action and temporal explanations for user engagement prediction on social apps. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2269–2279, 2020.

Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *NeurIPS*, pages 659–666, 2004.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *ICLR2019*, 2019.

Yasuaki Uechi. Vtuber 1b: Large-scale live chat and moderation events dataset, 2 2022. URL `https://holodata.org/vtuber-1b`.

Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903v3*, 2018.

Christian Von Ferber, Taras Holovatch, Yu Holovatch, and V Palchykov. Public transport networks: empirical analysis and modeling. *The European Physical Journal B*, 68:261–275, 2009.

Xinchen Wan, Kaiqiang Xu, Xudong Liao, Yilun Jin, Kai Chen, and Xin Jin. Scalable and efficient full-graph gnn training for large graphs. *Proceedings of the ACM on Management of Data*, 1(2):1–23, 2023.

Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Huan Wang, Qing Gao, Hao Li, Hao Wang, Liping Yan, and Guanghua Liu. A structural evolution-based anomaly detection method for generalized evolving social networks. *The Computer Journal*, 65(5):1189–1199, 2022.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *WWW*, pages 2022–2032, 2019a.

Xiaodong Wang, Ye Tian, Rongheng Lan, Wen Yang, and Xinming Zhang. Beyond the watching: Understanding viewer interactions in crowdsourced live video broadcasting services. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(11):3454–3468, 2019b.

Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, et al. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *SIGMOD2021*, pages 2628–2638, 2021.

Donghee Yvette Wohn, Peter Jough, Peter Eskander, John Scott Siri, Masaho Shimobayashi, and Pradnya Desai. Understanding digital patronage: why do people

subscribe to streamers on twitch? In *Proceedings of the annual symposium on computer-human interaction in play*, pages 99–110, 2019.

Sanghyun Woo, Dahun Kim, Donghyeon Cho, and In So Kweon. Linknet: Relational embedding for scene graph. *Advances in neural information processing systems*, 31, 2018.

Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 235–244, 2019a.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

Xuan Wu, Lingxiao Zhao, and Leman Akoglu. A quest for structure: Jointly learning the graph structure and semi-supervised classification. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 87–96, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019b.

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

Juan Xiao, Ashwani Kumar Aggarwal, Uday Kiran Rage, Vaibhav Katiyar, and Ram Avtar. Deep learning-based spatiotemporal fusion of unmanned aerial vehicle and satellite reflectance images for crop monitoring. *IEEE Access*, 11:85600–85614, 2023.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

Cheng Yang, Chunchen Wang, Yuanfu Lu, Xumeng Gong, Chuan Shi, Wei Wang, and Xu Zhang. Few-shot link prediction in dynamic networks. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1245–1255, 2022a.

Haiqiang Yang, Zihan Li, and Yashuai Qi. Predicting traffic propagation flow in urban road network with multi-graph convolutional network. *Complex & Intelligent Systems*, pages 1–13, 2023.

Heetae Yang and Hwansoo Lee. Exploring user acceptance of streaming media devices: an extended perspective of flow theory. *Information Systems and e-Business Management*, 16(1):1–27, 2018.

Jaewon Yang and Jure Leskovec. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186, 2011.

Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685, 2018.

Zhihao Yang, Dong Li, Yingxueff Zhang, Zhanguang Zhang, Guojie Song, Jianye Hao, et al. Versatile multi-stage graph neural network for circuit representation. *Advances in Neural Information Processing Systems*, 35:20313–20324, 2022b.

Yongjing Yin, Fandong Meng, Jinsong Su, Chulun Zhou, Zhengyuan Yang, Jie Zhou, and Jiebo Luo. A novel graph-based multi-modal fusion encoder for neural machine translation. *arXiv preprint arXiv:2007.08742*, 2020.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

Jaehyun Yoon. Forecasting of real gdp growth using machine learning models: Gradient boosting and random forest approach. *Computational Economics*, 57(1): 247–265, 2021.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. Graph-revised convolutional network. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*, pages 378–393. Springer, 2021.

Jiajun Yu and Adele Lu Jia. User donations in online social game streaming: The case of paid subscription in twitch. tv. In *Companion Proceedings of the Web Conference 2022*, pages 215–218, 2022.

Weiwei Yuan, Kangya He, Donghai Guan, Li Zhou, and Chenliang Li. Graph kernel based link prediction for signed social networks. *Information Fusion*, 46:1–10, 2019.

Ashkan Zakaryazad and Ekrem Duman. A profit-driven artificial neural network (ann) with applications to fraud detection and direct marketing. *Neurocomputing*, 175:121–131, 2016.

Lorenzo Zangari, Roberto Interdonato, Antonio Calió, and Andrea Tagarelli. Graph convolutional and attention models for entity classification in multilayer networks. *Applied Network Science*, 6(1):1–36, 2021.

Gregory S Zaric. *Operations research and health care policy*. 2013.

An Zeng, Stanislao Gualdi, Matúš Medo, and Yi-Cheng Zhang. Trend prediction in temporal bipartite networks: the case of movielens, netflix, and digg. *Advances in Complex Systems*, 16(04n05):1350024, 2013.

Jinming Zhan and Nan Zhang. Exploring the impact of virtual anchor features and live content on viewers' willingness to pay for "superchat" in live entertainment scenarios. *Highlights in Business, Economics and Management*, 6:189–205, 2023.

Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous graph neural network. In *KDD*, pages 793–803, 2019a.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NeurIPS2018*, NIPS'18, page 5171–5181, 2018.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. *AAAI2018*, 32(1), 2018a.

Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1): 11, 2019b.

Zhi-Hai Zhang and Hai Jiang. A robust counterpart approach to the bi-objective emergency medical service design problem. *Applied Mathematical Modelling*, 38 (3):1033–1040, 2014.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202*, 2018b.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2020.

Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1513–1522, 2015.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

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

Zhengyi Zhou and David S Matteson. Predicting ambulance demand: A spatio-temporal kernel approach. In *SIGKDD*, pages 2297–2303, 2015.

Zhengyi Zhou, David S. Matteson, Dawn B. Woodard, Shane G. Henderson, and Athanasios C. Micheas. A spatio-temporal point process model for ambulance demand. *Journal of the American Statistical Association*, 110(509):6–15, 2015.

Jiarun Zhu, Xingyu Wu, Muhammad Usman, Xiangyu Wang, and Huanhuan Chen. Link prediction in continuous-time dynamic heterogeneous graphs with causality of event types. *International Journal of Crowd Science*, 6(2):80–91, 2022.

Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. A survey on graph structure learning: Progress and opportunities. *arXiv preprint arXiv:2103.03036*, 2021a.

Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 14, 2021b.

Zhenhui Zhu, Zhi Yang, and Yafei Dai. Understanding the gift-sending interaction on live-streaming video websites. In Gabriele Meiselwitz, editor, *Social Computing and Social Media. Human Behavior*, pages 274–285, Cham, 2017. Springer International Publishing.

Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 world wide web conference*, pages 499–508, 2018.