Anomaly Detection in Multiplex Networks: From Human Brain Activity to Financial Networks

by

Ali Behrouz

B.Sc. Computer Engineering, Sharif University of Technology, 2020

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

August 2023

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Anomaly Detection in Multiplex Networks: From Human Brain Activity to Financial Networks

submitted by **Ali Behrouz** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

Margo Seltzer, Professor, Computer Science, UBC
Supervisor

Laks V.S. Lakshmanan, Professor, Computer Science, UBC
Supervisory Committee

Abstract

The problem of identifying anomalies in dynamic networks is a fundamental task with a wide range of applications from understanding brain diseases/disorders to fraud detection in financial networks. However, it raises critical challenges due to the complex nature of anomalies, lack of ground truth knowledge, and complex and dynamic interactions in the network. Most existing approaches usually study simple networks with a single type of connection. However many complex systems exhibit natural relationships with different types of connections, yielding multiplex networks. We first propose ANOMULY, a graph neural network-based unsupervised edge anomaly detection framework for multiplex dynamic networks. ANOMULY learns node encodings in different relation types, separately, and then uses an attention mechanism that incorporates information across different types of relations. To improve generalizability and scalability, we further propose ADMIRE, an *inductive* and unsupervised anomaly detection method that extracts the causality of the existence of connections by temporal network motifs. To extract the temporal network motifs, ADMIRE uses two different casual multiplex walks, *inter-view* and *intra-view* that automatically extract and learn temporal multiplex network motifs. Despite the outstanding performance of ADMIRE, using it in sensitive decision-making tasks requires explanations for the model's predictions. Accordingly, we introduce an interpretable, weighted optimal sparse decision tree model, ADMIRE++, that mimics ADMIRE, to provide explanations for ADMIRE's predictions. With extensive experiments, we show the efficiency and effectiveness of our approaches in detecting anomalous connections in various domains, including social and financial networks. We further focus on understanding abnormal human brain activity of people living with Parkinson's Disease, Attention Deficit

Hyperactivity Disorder, and Autism Spectrum Disorder to show how these methods can assist in understanding the biomarkers for these diseases.

Lay Summary

This thesis focuses on the detection of abnormal connections or interactions in complex systems. Identifying unexpected connections is a fundamental problem with broad applications, ranging from understanding brain diseases/disorders to identifying fraudulent transactions in financial networks. Existing approaches apply only to simple systems in which participants in a system interact in only one way. However, there are many different types of interactions in the real world. We introduce two machine learning-based methods that identify abnormal patterns without having access to examples of such abnormal patterns. We then present a technique that explains why our method made the predictions it did. This explainability procedure is crucial for fostering transparency and trust in decision-making processes, enabling humans to understand the basis of AI-driven decisions. Finally, we demonstrate how the proposed methods contribute to understanding abnormal brain activity that might underlie brain diseases/disorders.

Preface

The work presented in this thesis is an original work done by Ali Behrouz under the supervision of Prof. Margo Seltzer. A version of chapters 4, 5, and 6 of this work have been published as:

1. Ali Behrouz and Margo Seltzer. "Anomaly Detection in Multiplex Dynamic Networks: from Blockchain Security to Brain Disease Prediction." Temporal Graph Learning Workshop in NeurIPS 2022 [17].

2. Ali Behrouz and Margo Seltzer. "Anomaly Detection in Human Brain via Inductive Learning on Temporal Multiplex Networks" Machine Learning for Healthcare Conference 2023 [18].

3. Ali Behrouz and Margo Seltzer. "ADMIRE++: Explainable Anomaly Detection in the Human Brain via Inductive Learning on Temporal Multiplex Networks" Interpretable Machine Learning in Healthcare Workshop in ICML 2023 [19].

Table of Contents

Abstract . . . . . **iii**

Lay Summary . . . . . **v**

Preface . . . . . **vi**

Table of Contents . . . . . **vii**

List of Tables . . . . . **x**

List of Figures . . . . . **xii**

List of Abbreviations . . . . . **xiv**

Acknowledgments . . . . . **xvi**

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Contributions 4

2 Preliminaries and background **6**
- 2.1 Terminology 6
- 2.2 Graph Neural Networks 7
- 2.3 GRU Cell 8
- 2.4 Skip-connections 10
- 2.5 MLP-Mixer 10

2.6 (Weighted) Optimal Sparse Decision Trees 11
2.7 Brain Networks 12

3 Related Work 14
3.1 Anomaly Detection in Dynamic Networks 14
3.2 Dynamic Graph Neural Networks 15
3.3 Multiplex Graph Learning 16
3.4 Anomaly Detection in Multiplex Networks 16

4 ANOMULY: Edge Anomaly Detection Framework in Multiplex Dynamic Networks 18
4.1 Introduction 18
4.2 ANOMULY: **Ano**maly Detection in **Mul**tiplex D**y**namic Networks 19
4.2.1 GNN Architecture 21
4.2.2 Update Modules 21
4.2.3 Attention Mechanism 22
4.2.4 view-dependent Embedding 22
4.2.5 Anomaly Score Computation 23
4.2.6 Training and Loss Function 23
4.3 Experiments 25
4.3.1 Experimental Setup and Metrics 25
4.3.2 Datasets 25
4.3.3 Inject Anomalous Edges in Multiplex Networks 27
4.3.4 Baselines 27
4.3.5 Results 28
4.4 Conclusion 33

5 ADMIRE: Inductive and Scalable Anomaly Detection in Multiplex Dynamic Networks 34
5.1 Introduction 34
5.2 ADMIRE 35
5.2.1 Anonymous Multiplex Temporal Walk 35
5.2.2 Neural Encoding 40
5.2.3 ADMIRE Framework 43

5.3 ADMIRE++: Explainable Anomaly Detection in Multiplex Dynamic Networks . . . 45
5.4 Experiments . . . 46
5.4.1 Experimental Setup . . . 46
5.4.2 Results . . . 47
5.4.3 Ablation Study . . . 48
5.4.4 How Well Does ADMIRE++ Mimic ADMIRE? . . . 50
5.4.5 What Does the Generated Tree Look Like? . . . 50
5.5 Conclusion . . . 51

6 Anomaly Detection in the Human Brain . . . 54
6.1 Introduction . . . 54
6.2 Related Work on Machine Learning for Brain Networks . . . 56
6.3 Why/How to Model Human Brain as a Multiplex Network . . . 57
6.4 Modified Attention Mechanism For Multiplex Brain Networks . . . 58
6.5 Experiments . . . 59
6.5.1 Experimental Setting Details . . . 59
6.5.2 Datasets . . . 59
6.5.3 Baselines . . . 61
6.5.4 Experimental Setup . . . 61
6.5.5 Results . . . 62
6.5.6 Results on Real-world Datasets . . . 66
6.5.7 ADMIRE++ Explanations . . . 69

7 Conclusions . . . 71

Bibliography . . . 73

A Supporting Materials . . . 96
A.1 Performance of the Modified Attention on General Datasets . . . 96

List of Tables

Table 4.1 The value of hyper-parameters of ANOMULY. Here, η and μ control how close the encodings of two nodes should be so their connection is normal. Also, λ controls the tradeoff between the margin-based pairwise loss function and the regularization term. 25

Table 4.2 Network Statistics. Here $|\mathcal{V}|$ is the number of nodes, $|\mathcal{E}|$ is the number of edges in all views, and $|\mathcal{L}|$ is the number of views. . 26

Table 4.3 Performance comparison of ANOMULY and baselines in multiplex networks (AUC). 28

Table 4.4 Performance comparison of ANOMULY and baselines in single-layer networks (AUC). 29

Table 4.5 Ablation study on ANOMULY (AUC). The first row replaces the GRU with an MLP. The second row removes the attention mechanism, and the third row replaces the attention mechanism with SUM(.). 29

Table 5.1 Performance comparison of ADMIRE and baselines in multiplex networks (AUC). 47

Table 5.2 Performance comparison of ADMIRE and baselines in single-layer networks (AUC). 47

Table 5.3 Ablation study on ADMIRE (AUC). In each row, we remove/replace one of the ADMIRE's components and keep the others unchanged. 49

Table 5.4 Accuracy (%) of generated tree explanation on Amazon, DBLP and Ethereum. This table shows how well ADMIRE++ mimics ADMIRE predictions. We embolden the best results for each depth and use an underline to show the best performance for each dataset. 49

Table 6.1 Performance comparison of ADMIRE and baselines in multiplex brain networks (AUC). 63

Table 6.2 Ablation study of ADMIRE on brain network datasets. (AUC). 63

Table 6.3 Accuracy (%) of generated tree explanation. This table shows how well ADMIRE++ mimics ADMIRE predictions. 70

Table A.1 Performance of the modified attention on general datasets (AUC). 97

List of Figures

Figure 2.1 GNN Message passing from the perspective of node x_0. First, each node passes a message to each of its neighbors. After sending messages, each node receives the messages from its neighbors. Then each node aggregates its received messages and updates its node encoding. 9

Figure 4.1 Framework and design of ANOMULY model. ANOMULY first encodes nodes in each snapshot of the network and updates them with a GRU cell (embedding update module). Finally, to train the model in an unsupervised manner, it uses a negative sampling approach that corrupts normal connections to generate negative samples. 20

Figure 4.2 Stability over different training ratios on Ripple. 30

Figure 4.3 Event detection in Ethereum network. The top-4 local maximums all coincide with major events annotated in the figure . 30

Figure 4.4 Anomalous edges in the brain network of a sampled individual. 31

Figure 4.5 Distribution of anomalous edges in ADHD group. 32

Figure 5.1 **Schematic of the ADMIRE model**. ADMIRE consists of four main stages called (1) Walk Sampling, (2) Anonymization, (3) Walk Encoding, and (4) Training via generating negative samples. 36

Figure 5.2 (**Top**) Motif clusters and example of an extracted motif in each cluster. (**Bottom**) The ADMIRE++ tree explanation on the Amazon dataset (depth=4). Different colors in the motif examples show the changes in views. Here, ${P_j}^i$ are the features constructed in Equation 5.20. 51
Figure 5.3 3D visualization of clusters using t-SNE ($k = 4$). While clusters of walks in the Amazon dataset are simply distinguishable, clusters in DBLP and Ethereum datasets are hard to distinguish, which is the reason for the worse performance of ADMIRE++ (see Table 5.4) on these two datasets. 52

Figure 6.1 The effect of hyperparameters on the performance (a-c), and λ evolution (d). 65
Figure 6.2 The advantage of multiplex brain networks over monoplex brain networks. 67
Figure 6.3 The distribution of anomalous edges in PD group. 67
Figure 6.4 The distribution of anomalous edges in ADHD group. 67
Figure 6.5 The distribution of anomalous edges in ASD group. 68
Figure 6.6 (**Left**) Motif clusters and example of an extracted motif in each cluster. (**Right**) The ADMIRE++ tree explanation on PD dataset (depth=3). Different colors in the motif examples show the changes in views. Here, ${P_j}^i$ are the features constructed in Equation 5.20. 69

List of Abbreviations

ABIDE Autism Brain Imaging Data Exchange

ADHD Attention Deficit Hyperactivity Disorder

ANOMULY Anomaly Detection in Multiplex Dynamic Networks

ANT Attention Network Test

ASD Autism Spectrum Disorder

AUC Area Under the ROC Curve

CNN Convolutional Neural Network

DMRI Diffusion Magnetic Resonance Imaging

FMRI Functional Magnetic Resonance Imaging

GCN Graph Convolutional Networks

GELU Gaussian Error Linear Units

GNN Graph Neural Networks

GRU Gated Recurrent Unit

LDA Latent Dirichlet Allocation

MLP Multilayer Perceptron

NS Negative Sampling

PD Parkinson's Disease

RNN Recurrent Neural Network

ROI Region of Interest

SMRI Structural Magnetic Resonance Imaging

TD Typically Developed

Acknowledgments

First of all, I would like to deeply thank my advisor Professor Margo Seltzer for guiding me throughout my M.Sc. study. I learned so much about research from her over the past three years, and I would like to deeply thank her for giving me the opportunity to be a member of the Systopia Lab and learn from her.

I am also grateful to Professor Laks V.S. Lakshmanan, whose research inspired me in many of my research projects. I am fortunate to have had the chance to collaborate with and learn from him during my M.Sc. study.

I would like to thank Professor Cynthia Rudin and Professor Mathias Lécuyer, who I have the chance to work with and learn from. I learned a lot from them which also inspired me for a part of this thesis.

I would like to thank my parents and my sister for supporting me throughout this way. Despite being on the opposite side of the earth, I could still feel the love and care of them.

Finally, I would like to thank my collaborator, my projects partner, my best friend, and my love, Farnoosh, whose support, love, and care have been unwavering throughout this journey. Having her as a collaborator, friend, and partner, I feel fortunate to have the chance to share every moment of this journey with her.

Chapter 1

Introduction

1.1 Motivation

Identifying anomalous activities in networks is a long-standing and vital problem with a wide variety of applications in different domains, e.g., finance, social networks, security, and public health [2, 8, 9, 115, 141]. While several anomaly detection approaches focus on the topological properties of networks [63, 64, 110, 133, 134, 157], detecting anomalies in real-world networks also requires attention to their dynamic nature [141]. Anomalies might appear as malware in computer systems [77], social bots and social spammers in social networks [51], or financial fraud in financial systems [7, 8]. Accordingly, anomaly detection in dynamic (evolving) complex systems has recently attracted much attention.

Most prior work focuses on detecting anomalies in dynamic networks whose edges are all of the same type [24, 32, 40, 65, 66, 141, 180, 188]; these networks are called single-layer, dynamic networks. However, in many complex dynamic systems, such as social, transportation, and financial networks, there are many different kinds of interactions between objects. For example, interactions between people can be social or professional, and professional interactions can differ according to topics. We model graphs with different kinds of edges as Multilayer or Multiplex networks [98], where nodes can interact in multiple views, each of which is associated with a specific type of connection. In these networks, the different types of connections are complementary to each other, providing more complex and

richer information than simple graphs. Surprisingly, anomaly detection in multiplex networks is relatively less explored and has only recently attracted attention.

Existing approaches to anomaly detection in multiplex networks suffer from three main limitations: ① Structure and feature inflexibility: existing methods assume pre-defined anomaly patterns or man-made features. Such approaches are application dependent and do not easily generalize to different domains. Moreover, in real-world networks, anomalies might be more complex in nature, and it is nearly impossible to detect anomalies with high accuracy using pre-defined patterns/roles. ② Same importance for all types of connections: these methods treat each view identically, assigning the same importance to each view. However, real-world multiplex networks can contain noisy/insignificant views [71, 79]. Moreover, all vertices might not participate equally in all layers, so which layers are noisy/insignificant can be different for each vertex [71, 79]. ③ Lack of edge anomaly detection: previous methods for anomaly detection in multiplex networks focus on identifying anomalous nodes, subgraphs, or events. However, in many real-world applications, a connection between two vertices might be an anomaly [24, 40, 180, 188]. This anomalous connection might be a suspicious transaction in a financial network, a fake follower in a social network, or an abnormal functional correlation between two regions of the brain.

In addition to the limitations of existing anomaly detection methods in multiplex networks, existing methods for anomaly detection in single-layer dynamic graphs also exhibit limitations. ⓘ Structure inflexibility: even in single-layer networks, most existing anomaly detection methods for dynamic networks rely on pre-defined patterns or heuristic rules (see [142, 160]). These heuristic rules are usually content features or long-term temporal factors. However, these factors are not flexible and are restricted to specific patterns, while real-world anomalies have complex nature and a fixed description of an anomaly does not apply in all situations. ⓘⓘ Memory usage: deep learning-based methods [180, 188], which are commonly proposed, require storing entire snapshots of the network at each time window, consuming large and increasing amounts of memory. ⓘⓘⓘ Discrete timestamps: existing anomaly detection methods consider time as a discrete variable, while in many complex dynamic systems, interactions are continuous. Accordingly, modeling their connections' timestamps as discrete variables misses temporal properties and patterns.

That is, these methods consider discrete approximations of dynamic systems, but this discretization fails if we have irregularly observed data [96].

To overcome the above limitations, we study the problem of anomaly detection in multiplex, dynamic networks. Since multiplex networks provide richer and more complex information than simple networks, they also improve anomaly detection in simple dynamic graphs [9, 115, 141], delivering better solutions. Next, we discuss two of anomaly detection that apply only to multiplex networks.

Applications: Brain Networks. Monitoring functional systems in the human brain is a fundamental task in neuroscience [26, 139]. Each node in a brain network represents a region of interest (ROI), which is responsible for a specific function, and edges represent an association between two ROIs. For example, one common mode of analysis is to construct a brain network from functional magnetic resonance imaging (FMRI), where edges represent high functional correlation between two ROIs. A temporal brain network lets us measure the statistical association between the functionality of ROIs *over time*. Since a (dynamic) brain network generated from an individual can be noisy and/or incomplete [6, 20, 101, 185], prior work used the average of brain networks from many individuals [1, 42]. However, these methods ignore the complex relationships in each individual's brain. We can capture these missing relationships by modeling the network as a multiplex (dynamic) network [59], where each view represents an individual's brain network. An edge anomaly detection approach in multiplex networks can be used to reveal abnormal brain activity patterns that might cause or be indicative of a brain disease or disorder. In Chapter 6, we discuss this application in more detail.

Applications: Fraud Detection in Multiple Blockchain Networks. Anomaly detection in (dynamic) blockchain transaction networks has recently attracted enormous attention [39, 62, 81, 116, 135, 137], due to the emergence of a huge assortment of financial systems' applications [36, 76, 90]. While most existing work focuses on detecting illicit activity in a single blockchain network, recent research shows that cryptocurrency criminals increasingly use cross-cryptocurrency trades to hide their identity [125, 127]. Accordingly, Yousaf et al. [175] recently showed that analyzing links across several blockchain networks is critical for identifying emerging criminal activity on the blockchain. An edge anomaly detection approach

in multiplex networks can be used to detect suspicious transactions and identify criminal activity across several blockchain transaction networks more accurately.

1.2 Contributions

The contributions of this work are:

1. We address the lack of edge anomaly detection in multiplex networks, by introducing ANOMULY, an end-to-end unsupervised framework for detecting edge anomalies in dynamic multiplex networks. ANOMULY uses a novel layer-aware node embedding approach in multiplex dynamic networks, *Snapshot Encoder*, which uses an attention mechanism to incorporate both temporal and structural information on different relation types. Next, ANOMULY uses a GRU cell to incorporate the outputs of *Snapshot Encoder* for different snapshots, keeping node encodings updated over time. Finally, it uses a negative sampling approach in the training phase to overcome the lack of ground truth data (Chapter 4).

2. While ANOMULY shows outstanding performance in experiments on nine real-world multiplex and simple networks, it exhibits poor scalability, lack of generalizability to abnormal connections whose endpoints have not been seen in the training, and imprecise temporal modeling using a discrete approximation of network dynamics. To address these limitations, we present ADMIRE, an end-to-end inductive unsupervised learning method. ADMIRE uses inter-view and intra-view temporal walks to implicitly extract network motifs and causal relationships across different views and within a view, respectively. Next, ADMIRE adopts a novel anonymization process based on the correlation between multiplex network motifs to hide the identity of nodes and views. Next, it uses an MLP-Mixer to encode the sequence of nodes and views in a walk. By aggregating all the walks started from a node, ADMIRE encodes nodes' neighborhoods. Encoded neighborhoods are fed to a classifier to detect anomalous connections (Chapter 5).

3. Decision-making in sensitive domains (e.g., healthcare), requires models that explain the reason for each prediction to experts. To this end, we design

a post-hoc decision-tree-based explanation method to explain ADMIRE's predictions based on its extracted motifs. ADMIRE++ first uses inter-view and intra-view walks to extract network motifs around each node, and then constructs a feature vector that describes the neighborhood of each node based on the counting of extracted motifs. Finally, it trains an interpretable weighted optimal sparse decision tree model that gets these features as inputs and mimics ADMIRE predictions (Section 5.3).

4. We demonstrate a new application of edge-anomaly detection in dynamic multiplex networks and present several case studies on the brain network of people living with Attention Deficit Hyperactivity Disorder, Parkinson's Disease, and Autism Spectrum Disorder. The results show the potential of ADMIRE and ANOMULY in detecting abnormal brain activity that might reveal a disease or disorder (Chapter 6).

5. We further conduct extensive experiments on nine real-world multiplex and simple networks to evaluate the performance of ANOMULY and ADMIRE in different domains, varying from social, co-purchasing, co-authorship, and human mobility networks to financial networks. These experimental results are reported at the end of each chapter.

Chapter 2

Preliminaries and background

We first describe the terminology used in this thesis for describing multiplex dynamic networks and the problem of anomaly detection. Next, we discuss the preliminary background on graph neural networks, GRU cells, skip-connections, MLP-Mixer, optimal sparse decision trees, and brain networks.

2.1 Terminology

There are two types of representation for dynamic networks, so we begin by formally describing them.

Definition 1 (Multiplex Temporal Networks)**.** *Let $\mathcal{G} = \{G_r\}_{r=1}^{\mathcal{L}} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ denotes a multiplex temporal network, where $G_r = (\mathcal{V}, \mathcal{E}_r, \mathcal{X})$ is a graph of the relation type $r \in \mathcal{L}$ (also known as view), $\mathcal{V}$ is the set of nodes, $\mathcal{E} = \bigcup_{r=1}^{\mathcal{L}} \mathcal{E}_r$ is the set of edges such that each edge is associated with a timestamp τ_e and $\mathcal{X} \in \mathbb{R}^{|\mathcal{V}| \times f}$ is a matrix that encodes node attribute information for nodes in $\mathcal{V}$. Here, f is the number of available features for each node.*

In temporal multiplex networks, we assume that each node and each edge is associated with a timestamp, which shows the time that a node/edge first appears in the network. In the literature, there are two ways to process temporal multiplex networks based on whether time is a discrete or continuous variable. In Chapter 4, we consider time as a discrete variable and take a snapshot-based anomaly detection approach for multiplex dynamic networks: a multiplex dynamic graph $\mathcal{G} = \{\mathcal{G}^{(t)}\}_{t=1}^{T}$

can be represented as a sequence of multiplex network snapshots, where each snapshot is a static multiplex graph $\mathcal{G}^{(t)} = \{G_r^{(t)}\}_{r=1}^{\mathcal{L}} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)}, \mathcal{X}^{(t)})$ with $\mathcal{V}^{(t)} = \{v \in \mathcal{V} | \tau_v = t\}$ and $\mathcal{E}^{(t)} = \{e \in \mathcal{E} | \tau_e = t\}$. Our goal is to detect anomalous edges in $\mathcal{E}^{(t)}$. Note that, here, a snapshot of the network, $\mathcal{G}^{(t)}$, represents the state of a graph at time t.

In Chapter 5, we consider time as a continuous variable and take a streaming anomaly detection approach for multiplex dynamic networks:

Definition 2 (Stream Temporal Multiplex Networks)**.** *A temporal stream multiplex network $\mathcal{G} = \{G_r\}_{r=1}^{\mathcal{L}} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, can be represented as a sequence of connections with different types that arrive over time, i.e., $\mathcal{E} = \{(e_1, t_1), (e_2, t_2), \dots\}$, where $\mathcal{V}$ is the set of nodes, $\mathcal{L}$ is the set of relation types (views), $\{e_1, e_2, \dots\} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{L}$, and $\mathcal{X} \in \mathbb{R}^{|\mathcal{V}| \times f}$ is a matrix that encodes node attribute information for nodes in $\mathcal{V}$.*

Given a relation type r, we use $G_r = (\mathcal{V}, \mathcal{E}_r, \mathcal{X})$ to denote the corresponding graph of the relation type r (also known as the r-th view of the graph), and we denote the set of vertices in the neighborhood of $u \in \mathcal{V}$ in relation r as $\mathcal{N}_r(u)$. Given time t, we use $\mathcal{E}_r^t(u) = \{(e, t') \in \mathcal{E}_r | u \in e \text{ and } t' < t\}$ to represent the set of connections attached to a node u in relation type r before a given time t.

2.2 Graph Neural Networks

Message-passing Graph Neural Networks (GNNs) are a class of neural network models designed to operate on graph-structured data. The core idea behind message-passing GNNs is to iteratively update the node representations by aggregating and exchanging information with neighboring nodes. This process allows nodes to gather and propagate information across the graph, capturing both local and global structural patterns. We can mathematically formulate the process of message-passing as follows:

$$\mathbf{h}_v^{(0)} = \mathbf{x}_v \quad \forall v \in V \tag{2.1}$$

$$\mathbf{m}_v^{(k)} = \texttt{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) \tag{2.2}$$

$$\mathbf{h}_v^{(k)} = \texttt{COMBINE}^{(k)}\left(\mathbf{h}_v^{(k-1)}, \mathbf{m}_v^{(k)}\right) \tag{2.3}$$

$$\mathbf{h}_v^{(k)} = \texttt{UPDATE}^{(k)}\left(\mathbf{h}_v^{(k-1)}\right) \tag{2.4}$$

In the above equations, $\mathbf{x}_v$ represents the input feature vector for node v in the graph. $\mathbf{h}_v^{(k)}$ denotes the representation of node v at iteration k, and $\mathbf{m}_v^{(k)}$ represents the aggregated message from neighboring nodes of v at iteration k. The functions $\texttt{AGGREGATE}^{(k)}(.)$, $\texttt{COMBINE}^{(k)}(.)$, and $\texttt{UPDATE}^{(k)}(.)$ represent the aggregation, combination, and update functions used in the message-passing process, respectively. The specific choices of these functions can vary depending on the design of the GNN model. Common aggregation functions include SUM(.), MEAN(.), or MAXPOOLING(.), while combination functions typically involve concatenation or element-wise operations. The update function can be a simple feed-forward neural network or a more complex recurrent or attention mechanism.

By iteratively applying the message-passing equations, GNNs capture increasingly refined representations of the graph structure and learn node embeddings that encode both local and global information. These learned node representations can then be used for various downstream tasks, including anomaly detection.

Figure 2.1 shows an example of message passing in a simple graph. Each node passes messages to its neighbors and when a node receives all messages from its neighbors, aggregate them to update its own encoding.

2.3 GRU Cell

The Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that has been widely used for modeling sequential data [48]. It is known for its ability to capture long-term dependencies in the data while mitigating the vanishing gradient problem that can occur in traditional RNNs [48]. The GRU cell consists of various gates that control the flow of information within the network. These gates include

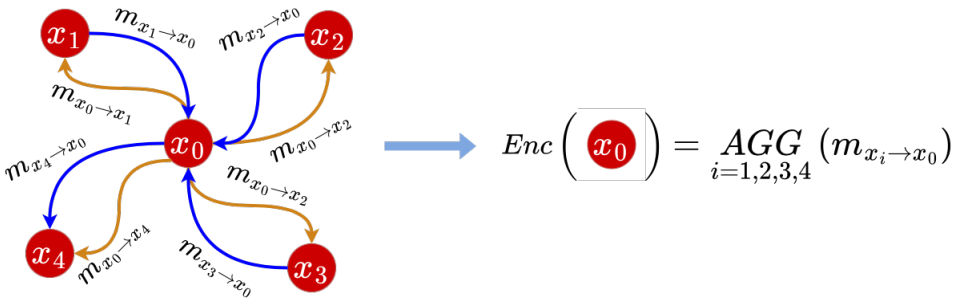


Figure 2.1: GNN Message passing from the perspective of node x_0. First, each node passes a message to each of its neighbors. After sending messages, each node receives the messages from its neighbors. Then each node aggregates its received messages and updates its node encoding.

an update gate, a reset gate, and an output gate. The update gate determines how much of the past information should be retained, the reset gate decides how much of the past information should be forgotten, and the output gate regulates the amount of information to be outputted. The mathematical formulation of the GRU cell can be represented as follows:

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \tag{2.5}$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \tag{2.6}$$

$$\tilde{h}_t = \texttt{tanh}\left(W_h \cdot [r_t \odot h_{t-1}, x_t]\right) \tag{2.7}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{2.8}$$

In the above equations, x_t represents the input at time step t, h_{t-1} is the hidden state of the previous time step, and h_t is the updated hidden state at time step t. W_z, W_r and W_h are weight matrices specific to each gate. $\sigma(.)$ denotes the sigmoid activation function, $\odot$ represents element-wise multiplication, and $[a, b]$ denotes concatenation of vectors a and b. The update gate z_t controls how much of the previous hidden state to retain and how much of the new information to incorporate. The reset gate r_t determines how much of the previous hidden state should be forgotten. The input and previous hidden state are combined with the reset gate

to compute an intermediate hidden state $\tilde{h}_t$. Finally, the new hidden state h_t is obtained by combining the previous hidden state with the updated information from $\tilde{h}_t$ based on the update gate z_t. This formulation allows the GRU cell to selectively retain or forget information, making it well-suited for tasks involving sequential data, such as language modeling, machine translation, and speech recognition.

2.4 Skip-connections

A skip connection [104], also known as residual connection, is a technique commonly used in deep machine learning methods to improve the flow of information and alleviate the vanishing gradient problem [104]. They allow the network to retain and propagate important information from earlier layers to later layers.

Mathematically, in GNNs, skip connections can be represented as follows [104]:

$$\mathbf{h}_v^{(k)} = \text{AGGREGATE}\left(\left\{\mathbf{h}_u^{(k)} : u \in \mathcal{N}(v)\right\}\right) + \mathbf{h}_v^{(k-1)} \tag{2.9}$$

$$\mathbf{h}_v^{(k+1)} = \text{COMBINE}\left(\mathbf{h}_v^{(k)}, \mathbf{h}_v^{(k+1)}\right) \tag{2.10}$$

In the above equations, $\mathbf{h}_v^{(k)}$ represents the hidden state of node v at layer k. The first equation combines the aggregated information from neighboring nodes with the hidden state of the previous layer, allowing the current layer to have direct access to both local and global information. The second equation combines the skip connection output with the output of the current layer using a combining function.

The skip connections enable GNNs to effectively handle deep architectures by facilitating the flow of gradients during training. By retaining information from earlier layers, the network can mitigate the vanishing gradient problem and improve the learning of long-range dependencies. Furthermore, skip connections also help preserve and propagate important features throughout the network.

2.5 MLP-Mixer

MLP-Mixer, an all-Multilayer Perceptron (MLP) architecture, is a type of neural network architecture that has gained attention in the field of computer vision. It is

designed as an alternative approach to traditional convolutional neural networks (CNNs) and Transformers [153]. Each layer in the architecture of MLP-Mixer consists of two distinct sub-layers: a token-mixing layer and a channel-mixing layer. Channels refer to pixel values in an image. The token-mixing layer processes the spatial information within each channel independently, while the channel-mixing layer integrates the information across different channels. This mixing of information helps capture both local and global dependencies within the image.

The mathematical formulation of MLP-Mixer can be represented as follows:
① Token Mixer:

$$\mathbf{H}_{\text{token}} = \mathbf{E} + \mathbf{W}^{(2)}_{\text{token}} \sigma \left(\mathbf{W}^{(1)}_{\text{token}} \texttt{LayerNorm}\left(\mathbf{E}\right)^{T} \right)^{T}, \tag{2.11}$$

① Channel Mixer:

$$\mathbf{H}_{\text{channel}} = \mathbf{H}_{\text{token}} + \mathbf{W}^{(2)}_{\text{channel}} \sigma \left(\mathbf{W}^{(1)}_{\text{channel}} \texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right) \right), \tag{2.12}$$

In the above equations, E is the input matrix and $\sigma(.)$ is a nonlinear activation function (usually Gaussian Error Linear Units (GELU) [84]). The main advantage of MLP-Mixer is its simplicity as well as its ability to capture cross-feature and cross-sample dependencies. That is, the Token-Mixer phase captures the dependencies between all samples across each feature, while the Channel-Mixer phase learns the dependencies of features in each sample. This combination produces promising advantages in various domains such as graph learning [22, 49], computer vision [151, 177], and natural language processing [70].

2.6 (Weighted) Optimal Sparse Decision Trees

Decision trees are one of the most popular forms of interpretable models [144]. While full decision tree optimization is NP-hard [102], it is possible to make assumptions, e.g., feature independence, that simplifies the hard optimization to cases where greedy methods suffice [99]. However, these assumptions are unrealistic in practice, causing suboptimal performance in real-world scenarios. More recently there have been several approaches that fully optimize sparse trees to yield the best combination of performance and interpretability [23, 68, 75, 126]. This work

leverages the fact that the loss takes on a discrete number of values to enable efficient computation [4, 5, 107, 120]. Although in real-world applications, samples might differ in their importance, none of these methods handle weighted samples. To address this limitation, Behrouz et al. [21] extend the decision tree optimization problem to include weighted samples.

Let $\mathcal{T}$ be a decision tree that gives predictions $\{\hat{y}_i^{\mathcal{T}}\}_{i=1}^N$. The weighted loss of $\mathcal{T}$ is:

$$\mathcal{L}_{\mathbf{w}}(\mathcal{T}, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{\sum_{i=1}^N w_i} \sum_{i=1}^N \mathbb{1}[y_i \neq \hat{y}_i^{\mathcal{T}}] \times w_i \,. \tag{2.13}$$

Building on the work of McTavish et al. [120], Behrouz et al. [21] provides the option to use either soft sparsity regularization on the number of leaves, hard regularization on the tree depth, or both:

$$\underset{\mathcal{T}}{\text{minimize}} \;\; \mathcal{L}_{\mathbf{w}}(\mathcal{T}, \tilde{\mathbf{x}}, \mathbf{y}) + \lambda H_{\mathcal{T}} \quad s.t. \;\; \text{depth}(\mathcal{T}) \leq d \,, \tag{2.14}$$

where $H_{\mathcal{T}}$ is the number of leaves in $\mathcal{T}$ and λ is a per-leaf regularization parameter. Behrouz et al. [21] develop three algorithms to find the optimal weighted tree. The first approach directly optimizes the weighted loss function by the dynamic programming algorithm of Lin et al. [107], which requires using floating point vector operations, rather than bit vector operations, leading to poor scalability. The second approach scales better by transforming weights to integer values and using data duplication to transform the weighted decision tree optimization problem into an unweighted problem with more data samples. The third algorithm, which scales to much larger datasets, uses a randomized procedure that samples each data point with a probability proportional to its weight.

2.7 Brain Networks

Brain networks refer to interconnected patterns of neural activity or structural connections within the human brain. These networks provide valuable insights into brain organization, function, and information processing. They are often studied using neuroimaging techniques such as functional magnetic resonance imaging (FMRI) and structural magnetic resonance imaging (SMRI) [191].

Mathematically, brain networks can be represented as graphs, where the brain regions are nodes, and the connections or any other association between them are edges. To obtain brain networks from fMRI data, a common approach is to extract the time series of activity from different brain regions and then calculate the correlation or coherence between the time series of different brain regions. The resulting correlation matrix represents the functional connectivity network of the brain. In the case of sMRI data, the goal is to map the structural connections between different brain regions, which can be done using diffusion MRI (DMRI) and then estimating the white matter pathways connecting different brain regions [57]. The study of brain networks has contributed to our understanding of brain function [14, 20, 80, 152], cognitive processes [15, 145], and neurological disorders [42, 109].

In Chapter 6, we provide more detail on how to obtain the brain network from neuroimaging data.

Chapter 3

Related Work

We begin with a review of anomaly detection algorithms in dynamic simple networks, then discuss methods for dynamic and multiplex graph learning, and finally present the state of the art in anomaly detection in multiplex networks. For additional related work, we refer the reader to the extensive survey by Ma et al. [115].

3.1 Anomaly Detection in Dynamic Networks

There has been much work in anomaly detection for single-layer dynamic networks. That work falls in four categories: **(1)** Probabilistic methods [3, 24, 43, 83, 131, 156, 178] that identify anomalies based on deviations from regular communication patterns. **(2)** Distance-based methods [65, 66, 142, 160, 172] that use certain time-evolving measures of dynamic network structures and use their changes to detect anomalies. **(3)** Density-based methods [31, 85, 168] that view anomalies as subgraphs with high density or as subgraphs with sudden changes in their density. **(4)** Matrix factorization methods [117, 147, 148, 150, 179] that use the low-rank property of network structures and define anomalies as breakers of this property.

All approaches in these four groups are based on pre-defined patterns/rules and cannot learn unforseen anomalous patterns, which limits their application. Learning-based methods that combine the graph embedding method into the anomaly detection approach [33, 87, 111, 180, 188, 189] address this shortcoming. These learning-based models assume the dynamic nature of these graphs is accurately

modeled as a sequence of graph snapshots. Here, each snapshot represents the state of the graph at a specific timestamp. These methods must store the entire snapshot, which requires large memory, limiting their ability to scale to large graphs. Also, all these methods apply to single-layer networks only and do not naturally extend to multiplex networks. That is, these methods are not able to consider different types of connections, missing the semantics of the interactions.

3.2 Dynamic Graph Neural Networks

The problem of learning from dynamic networks has been extensively studied in the literature [22, 45, 72, 80, 105, 106, 128, 132, 149, 171]. The first group of existing methods uses Recurrent Neural Networks (RNN) and then replaces the linear layer with a graph convolution layer [106, 146, 187]. The second group uses a Graph Neural Network (GNN) as a feature encoder and then deploys a sequence model on top of the GNN to encode temporal properties [132, 162, 176]. To address the scalability and model design issue of these methods, You et al. [174] presented ROLAND, a graph learning framework for dynamic graphs that can re-purpose any static GNN to dynamic graphs. Recently, more sophisticated learning methods for temporal graphs have been designed based on temporal random walks [22, 88, 163], line graphs [41], GraphMixer [49], neighborhood representation [113], and subgraph sketching [37]. However, all these methods differ from ADMIRE and ANOMULY as they are designed for temporal graphs with only a single view and cannot easily be extended to multiplex networks.

The *Snapshot Encoder* in ANOMULY can be seen as the extension of ROLAND to multiplex networks. However, natural attempts to use multiplex graph neural networks [16, 35, 47, 136, 169, 182] in the ROLAND framework (e.g., replacing the GNN block with a multiplex GNN) ignore historical data in other views. For example, assume that we want to use ANOMULY's attention mechanism (see Section 4.2.3) in the ROLAND framework. Then, we need to use the attention mechanism in the GNN block, which means we incorporate information about different relation types before the *Embedding update* module. Accordingly, for each timestamp, we do not incorporate historical data on other relation types, which could produce undesirable performance. By introducing an attention mechanism that

incorporates relation-specific hierarchical node states in each snapshot after each *Embedding update*, ANOMULY takes advantage of both multiplex and temporal information.

3.3 Multiplex Graph Learning

In a multiplex network, also known as multilayer, multi-view, or multi-dimensional networks, all nodes have the same type, but edges (relations) have multiple types [98]. Several methods have been proposed to learn network embeddings on multiplex networks by integrating information from individual relation types [35, 38, 136, 161, 166, 169]. Other work proposed Graph Convolutional Networks (GCN) methods for multiplex networks [16, 47, 182]. Inspired by Deep Graph Infomax [155], Park et al. [130] and Jing et al. [89] proposed unsupervised approaches to learn node embeddings by maximizing the mutual information between local patches and the global representation of the entire graph. Zhang et al. [183] proposed a method that uses a latent space to integrate the information across multiple views. Recently, Wang et al. [159] proposed DPMNE to learn from incomplete multiplex networks.

Several methods have been proposed for learning on heterogeneous networks (networks with multiple vertex and edge types) [38, 166, 170]. Although multiplex networks can be seen as a special case of heterogeneous networks (with a single type of nodes), these learning methods on heterogeneous graphs emphasize heterogeneous types of entities connected by different relationships, which is different from the concept of multiplex networks [130]. That is, these methods do not take advantage of the complementary information provided by different types of connections among *the same type of nodes*, missing their complex interactions.

Existing methods for multiplex networks differ from ADMIRE and ANOMULY as they ignore the temporal properties of the graph.

3.4 Anomaly Detection in Multiplex Networks

The problem of anomaly detection in multiplex networks has recently attracted attention. Mittal and Bhatia [122] use eigenvector centrality, page rank centrality, and degree centrality as handcrafted features for nodes to detect anomalies in static multiplex networks. Bindu et al. [25] proposed a node anomaly detection

algorithm in static multiplex networks that uses handcrafted features based on clique/near-clique and star/near-star structures. Bansal and Sharma [12] defined a quality measure, Multi-Normality, which employs the structure and attributes together of each layer to detect attribute coherence in neighborhoods between layers. Maulana and Atzmueller [119] use centrality of all nodes in each layer and apply many-objective optimization with full enumeration based on minimization to obtain Pareto Front. Then, they use Pareto Front as a basis for finding suspected anomaly nodes. Chen et al. [46] proposed ANOMMAN that uses an auto-encoder module and a GCN-based decoder to detect node anomalies in static multiplex networks. Although this model can learn from the data, it is limited to static networks, and it treats each layer equally in the *Structure Reconstruction* step. Finally, Ofori-Boateng et al. [127] developed a new persistence summary and used it to detect events in dynamic multiplex blockchain networks.

All of these approaches are designed to detect topological anomalous subgraphs, nodes, or events, and cannot identify anomalous edges. Moreover, as we discussed in Section 4.1, these methods, except ANOMAN [46], are based on pre-defined patterns/roles or handcrafted features, while real-world network anomalies have complex nature and have varied and unknown patterns that preclude pre-identification. Therefore, these models cannot be generalized to different domains, limiting their application.

Chapter 4

ANOMULY: Edge Anomaly Detection Framework in Multiplex Dynamic Networks

We present ANOMULY, an unsupervised generic graph neural network-based framework for anomaly detection in multiplex dynamic networks. ANOMULYlearns the low-dimensional representation of vertices in graphs by taking advantage of historical data and complementary information provided by different views of the network. Then, we use the vertex encodings to classify future connections as normal or abnormal.

4.1 Introduction

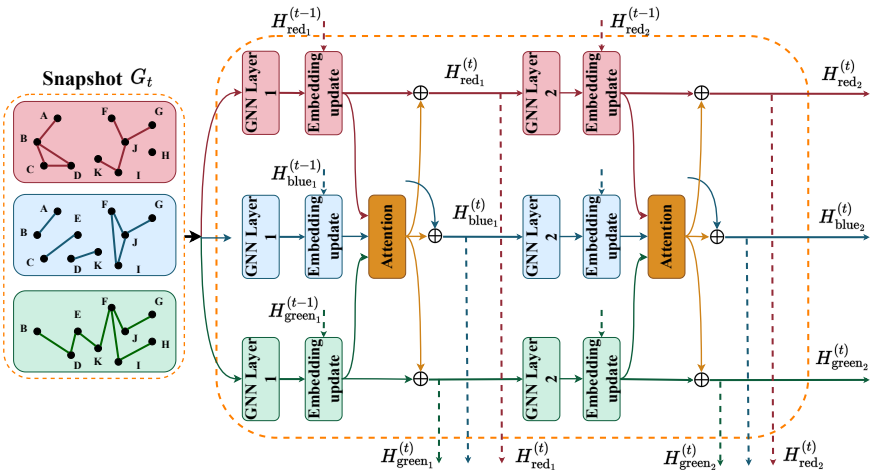
As discussed in Chapter 1, existing approaches to anomaly detection in multiplex networks suffer from three main limitations: ① Structure and feature inflexibility ② Same importance for all types of connections. ③ Lack of edge anomaly detection. We introduce ANOMULY (**Ano**maly Detection in **Mul**tiplex D**y**namic Networks), which mitigates all of these limitations. ANOMULY extends the idea of *hierarchical node states* [174] to multiplex dynamic networks by using an attention mechanism that incorporates information about different relation types to take advantage of both temporal properties and complementary information present in multiple relation

types. Next, it uses selective negative sampling to learn anomalous edges in an unsupervised manner. To the best of our knowledge, ANOMULY is the first edge-anomaly detection method for multiplex networks. Further, when it is possible to model a simple network as a multiplex network, ANOMULY outperforms existing simple network approaches, because the multiplex network provides richer and more complex information than does a simple network [9, 115, 141].

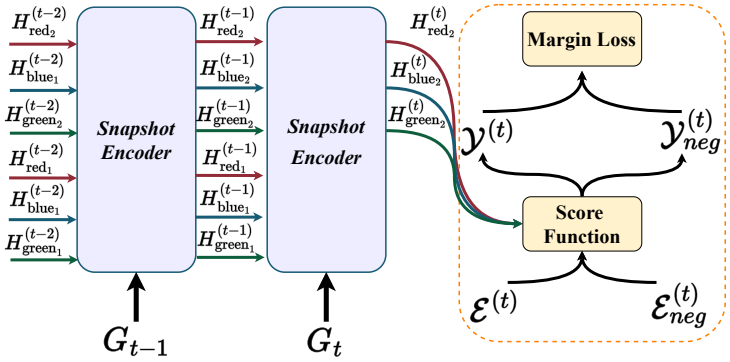
4.2 ANOMULY: Anomaly Detection in Multiplex Dynamic Networks

ANOMULY is a snapshot-based anomaly detection approach for multiplex dynamic networks. Accordingly, we model the multiplex dynamic network as a sequence of multiplex network snapshots as we described in Chapter 2: a multiplex dynamic graph $\mathcal{G} = \{\mathcal{G}^{(t)}\}_{t=1}^{T}$ can be represented as a sequence of multiplex network snapshots, where each snapshot is a static multiplex graph $\mathcal{G}^{(t)} = \{G_r^{(t)}\}_{r=1}^{\mathcal{L}} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)}, \mathcal{X}^{(t)})$ with $\mathcal{V}^{(t)} = \{v \in \mathcal{V} | \tau_v = t\}$ and $\mathcal{E}^{(t)} = \{e \in \mathcal{E} | \tau_e = t\}$. Our goal is to detect anomalous edges in $\mathcal{E}^{(t)}$. Specifically, for each edge $e = (u, v, r) \in \mathcal{E}^{(t)}$, we produce a view-dependent anomalous probability $\varphi_r(e)$ in view $r \in \mathcal{L}$.

Figure 4.1 provides an overview of our framework for edge anomaly detection in dynamic networks with multiple types of interactions. To learn the pattern of normal edges, ANOMULY uses the *Snapshot Encoder* architecture to encode each snapshot of the network. The *Snapshot Encoder* is a GNN that takes as input a snapshot of the graph and produces, for each view of the graph, an embedding for each vertex. It incorporates both structural and temporal properties of each snapshot, as well as node features in each view. Next, it uses an attention mechanism to take advantage of complementary information from different views. The attention mechanism is a module that learns the importance of each view for each node and then uses these weights in a weighted sum to obtain the node encodings. Since the graph is dynamically changing, the node embeddings can change over time, so *Snapshot Encoder* uses a GRU cell [48] (see Chapter 2 for more explanation about GRU cells) to update the node states over time. Finally, we use the hierarchical node states at each timestamp to calculate the view-dependent anomalous probabilities of an existing edge and a negative sampled edge and use them as inputs to a margin



(a) Architecture of the *Snapshot Encoder*



(b) ANOMULY Framework

Figure 4.1: Framework and design of ANOMULY model. ANOMULY first encodes nodes in each snapshot of the network and updates them with a GRU cell (embedding update module). Finally, to train the model in an unsupervised manner, it uses a negative sampling approach that corrupts normal connections to generate negative samples.

loss computation. Next, we explain each part in more detail.

4.2.1 GNN Architecture

A GNN iteratively aggregates messages from the local neighborhood of nodes to learn node embeddings. For a given type of relation r, we use the embedding matrix $\tilde{H}_r^{(\ell)} = \{\tilde{\mathbf{h}}_{r_u}^{(\ell)}\}_{u \in \mathcal{V}}$ to denote the embedding of all vertices in relation type r after applying the ℓ-th GNN layer. Given a relation type r, the ℓ-th layer of the GNN, $\tilde{H}_r^{(\ell)} = \textsc{GNN}_r(\tilde{H}_r^{(\ell-1)})$, is defined as:

$$\begin{aligned} \mathbf{m}_{r_{(v \to u)}}^{\ell} &= W_r^{(\ell)} \textsc{Concat}\left(\tilde{\mathbf{h}}_{r_v}^{(\ell-1)}, \tilde{\mathbf{h}}_{r_u}^{(\ell-1)}, \tau_{(v,u,r)}\right), \\ \tilde{\mathbf{h}}_{r_u}^{(l)} &= \textsc{Agg}^{(\ell)}\left(\left\{\mathbf{m}_{r_{(v \to u)}}^{\ell} | v \in \mathcal{N}_r(u)\right\}\right) + \tilde{\mathbf{h}}_{r_u}^{(\ell-1)}. \end{aligned} \tag{4.1}$$

In our experiments, we follow You et al. [174], and use summation as the aggregation function, i.e., $\textsc{Agg}(.) = \textsc{Sum}(.)$. We also use skip-connections [104] (see Chapter 2) after aggregation.

4.2.2 Update Modules

Given the snapshot $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)}, \mathcal{X}^{(t)})$ at time t and a relation type r, we denote the embedding matrix in relation r after the ℓ-th GNN layer at time t by $\tilde{H}_r^{(t)^{(\ell)}} = \{\tilde{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}\}_{u \in \mathcal{V}}$. To take advantage of historical data and update the node embeddings at each timestamp, in the *Embedding update* block (see Figure 4.1), we use a GRU cell [48]. Given a relation type r, the output of the ℓ-th *Embedding update*, $\hat{H}_r^{(t)^{(\ell)}} = \{\hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}\}_{u \in \mathcal{V}}$, is:

$$\hat{H}_r^{(t)^{(\ell)}} = \textsc{GRU}_r\left(\tilde{H}_r^{(t)^{(\ell)}}, H_r^{(t-1)^{(\ell)}}\right), \tag{4.2}$$

where $\tilde{H}_r^{(t)^{(\ell)}}$ is the output of the ℓ-th GNN layer, and $H_r^{(t-1)^{(\ell)}}$ is the view-dependent embedding matrix at time $t-1$. The view-dependent embedding matrix, $H_r^{(t)^{(\ell)}}$, is defined later in this section (see Equation 4.5).

4.2.3 Attention Mechanism

The role of our attention mechanism is to incorporate information from different relation types in a weighted manner. As discussed in Chapter 1, the importance of a view can differ for different nodes, so we cannot calculate a single weight for each view. Accordingly, we introduce an attention mechanism that learns the importance of view r for an arbitrary node $u \in \mathcal{V}$. Let $\zeta_u^{(t)^{(\ell)}}$ be the aggregated hidden feature of node $u \in \mathcal{V}$ after the ℓ-th attention layer at time t, we call it a network-level embedding, and $\alpha_{r_u}^{(\ell)}$ indicates the importance of relation type r for vertex u, then:

$$\zeta_u^{(t)^{(\ell)}} = \sum_{r=1}^{L} \alpha_{r_u}^{(\ell)} \hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}, \tag{4.3}$$

where $\hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}$ is the output of ℓ-th *Embedding update* for node u at time t. Following the recent attention-based models [11, 130], we use the $\texttt{Softmax}(.)$ function to define the importance weight of relation type r for node u:

$$\alpha_{r_u}^{(\ell)} = \frac{\exp\left(\sigma\left(\mathbf{s}_r^{(t)^{(\ell)}T} \mathbf{W}_r^{\text{att}} \ \hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}\right)\right)}{\sum_{k=1}^{L} \exp\left(\sigma\left(\mathbf{s}_k^{(t)^{(\ell)}T} \mathbf{W}_k^{\text{att}} \ \hat{\mathbf{h}}_k^{(t)^{(\ell)}}\right)\right)}, \tag{4.4}$$

where $\mathbf{s}_r^{(t)^{(\ell)}}$ is a summary of the network in relation type r at time t, i.e., $\mathbf{s}_r^{(t)^{(\ell)}} = \sum_{u \in \mathcal{V}} \hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}$, and $\mathbf{W}_r^{\text{att}}$ is a trainable weight matrix. In our experiments, we use $\tanh(.)$ as the activation function $\sigma(.)$. The main intuition behind this attention mechanism is that the numerator of Equation 4.4 captures how much a specific node encoding (e.g., $\hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}$) contributes to the encoding of a view (e.g., $\mathbf{s}_r^{(t)^{(\ell)}}$). That is, it uses the inner product of these two vectors followed by a multiplication by a learnable matrix $\mathbf{W}_r^{\text{att}}$ which captures the weighted similarity of these two vectors.

4.2.4 view-dependent Embedding

The output of the attention mechanism is a network-level node embedding matrix, which summarizes the properties of nodes over all relation types. Given a relation type r, to obtain the view-dependent node embedding of a vertex $u \in \mathcal{V}$, we

aggregate the output of the *Embedding update* block, i.e. $\hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}$, and this network-level node embedding, i.e. $\zeta_u^{(t)^{(\ell)}}$. That is:

$$\mathbf{h}_{r_u}^{(t)^{(\ell)}} = \textsc{Agg}^{(\ell)}\left(\hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}}, \zeta_u^{(t)^{(\ell)}}\right). \tag{4.5}$$

Based on Equation 4.5, we obtain the view-dependent node embedding matrix, $H_r^{(t)^{(\ell)}} = \{\mathbf{h}_{r_u}^{(t)^{(\ell)}}\}_{u\in\mathcal{V}}$, for any relation type r. Note that we use the view-dependent node embedding matrix at time $t-1$, $H_r^{(t-1)^{(\ell)}}$, in Equation 4.2 to update node embeddings after the ℓ-layer GNN.

4.2.5 Anomaly Score Computation

Now, we get the view-dependent node embedding matrix $H_r^{(t)} = H_r^{(t)^{(L)}}$ at time t, for each relation type r. Here L is the number of GNN layers. Inspired by Zheng et al. [188], for an edge $(u, v) \in \mathcal{E}_r$, we define its anomaly score as follows:

$$\varphi_r^{(t)}(u, v) = \sigma\left(\eta.\left(||\mathbf{a} \odot \mathbf{h}_{r_u}^{(t)} + \mathbf{b} \odot \mathbf{h}_{r_v}^{(t)}||_2^2 - \mu\right)\right), \tag{4.6}$$

where $\sigma(.)$ is an activation function, $\mathbf{a}$ and $\mathbf{b}$ are trainable vectors, and η and μ are hyperparameters.

4.2.6 Training and Loss Function

In the training phase, we use a negative sampling approach to corrupt edges and generate anomalous connections. Inspired by the negative sampling methods proposed by Wang et al. [164] and Zheng et al. [188], given a relation type r, and a normal edge $(u, v) \in \mathcal{E}_r$, we use a Bernoulli distribution such that we replace u (resp. v) with probability $\frac{deg_r(u)}{deg_r(u)+deg_r(v)}$ (resp. $\frac{deg_r(v)}{deg_r(u)+deg_r(v)}$) in relation type r to generate random negative samples. The main intuition is: changing one endpoint of an edge might result in another valid connection, and to avoid this, we prefer to replace an endpoint with a larger degree and keep the endpoint with a smaller degree. Since the corrupted edges might be normal, a strict loss function (e.g., cross-entropy) can affect the performance. Accordingly, we use the margin-based pairwise loss [28] in each relation type r. Given a relation type r, we also use a $L2$-regularization loss, $\mathscr{L}_r^{reg}$, which is the summation of the $L2$ norm of all trainable parameters, to

Algorithm 1 ANOMULY Training Algorithm

Input: A multiplex dynamic network $\mathcal{G} = \{\mathcal{G}^{(t)}\}_{t=1}^{T}$
Output: Node embeddings for all relation types $\{H_r^{(T)}\}_{r=1}^{\mathcal{L}}$

Initialize $\left\{ \left\{ H_r^{(0)^{(\ell)}} \right\}_{\ell=1}^{L} \right\}_{r=1}^{\mathcal{L}}$ with features or one-hot encoding;
while not converged **do**
  for $t = 1, \ldots, T$ **do**
    for $r = 1, \ldots, \mathcal{L}$ **do**
      Let $\mathscr{L}_r^{(t)} = 0$;
      for $\ell = 1, \ldots, L$ **do**
        $\tilde{H}_r^{(t)^{(\ell)}} = \text{GNN}_r \left(H_r^{(\ell-1)} \right)$;
        $\hat{H}_r^{(t)^{(\ell)}} = \text{GRU}_r \left(\tilde{H}_r^{(t)^{(\ell)}}, H_r^{(t-1)^{(\ell)}} \right)$;
        $Z_r^{(t)^{(\ell)}} = \{\zeta_u^{(t)^{(\ell)}}\}_{u \in \mathcal{V}} = \left\{ \sum_{r=1}^{L} \alpha_{r_u}^{(\ell)} \hat{\mathbf{h}}_{r_u}^{(t)^{(\ell)}} \right\}_{u \in \mathcal{V}}$;
        $H_r^{(t)^{(\ell)}} = \text{AGG}^{(\ell)} \left(\hat{H}_r^{(t)^{(\ell)}}, Z_r^{(t)^{(\ell)}} \right)$
      for $(u, v) \in \mathcal{E}_r^{(t)}$ **do**
        Sample (u', v') for $\varphi_r^{(t)}(u, v)$;
        $\mathscr{L}_r^{(t)} = \mathscr{L}_r^{(t)} + \max \left\{ 0, \gamma + \varphi_r^{(t)}(u, v) - \varphi_r^{(t)}(u', v') \right\}$;
      $\mathscr{L}_r^{(t)} = \mathscr{L}_r^{(t)} + \lambda \mathscr{L}_r^{reg}$;
    $\mathscr{L}^{(t)} = \frac{1}{\mathcal{L}} \left(\sum_{r=1}^{\mathcal{L}} \mathscr{L}_r^{(t)} \right)$;
    Minimize $\mathscr{L}^{(t)}$;
return $\{H_r^{(T)}\}_{r=1}^{\mathcal{L}}$

avoid overfitting. Finally, to aggregate the loss function over all relation types in the multiplex networks, we use the average of loss functions, i.e.:

$$\mathscr{L} = \frac{1}{|\mathcal{L}|} \left(\sum_{r=1}^{\mathcal{L}} \sum_{(u,v) \in \mathcal{E}_r} \sum_{(u',v') \notin \mathcal{E}_r} \max \left\{ 0, \gamma + \varphi_r(u, v) - \varphi_r(u', v') \right\} + \lambda \mathscr{L}_r^{reg} \right). \tag{4.7}$$

Here, $0 \leq \gamma \leq 1$ is the margin between normal and corrupted edges.

ANOMULY's training algorithm appears in detail in Algorithm 1.

4.3 Experiments

4.3.1 Experimental Setup and Metrics

The *Snapshot Encoder* uses a GNN with 200 hidden dimensions for node states, layers with skip-connection, sum aggregation, and batch-normalization. We tune hyper-parameters by cross-validation on a rolling basis and search the hyper-parameters over **(i)** the numbers of GNN layers (1 to 5); **(ii)** the learning rate (0.001 to 0.01); and **(iii)** the margin γ (in increments of 0.05 in the range 0.3 to 0.7). The values of other hyper-parameters are reported in Table 4.1. The hidden dimension of GNN is fine-tuned by searching in $\{32, 64, 100, 200, 320\}$.

Table 4.1: The value of hyper-parameters of ANOMULY. Here, η and μ control how close the encodings of two nodes should be so their connection is normal. Also, λ controls the tradeoff between the margin-based pairwise loss function and the regularization term.

Dataset	Multiplex Networks						Single-layer Networks		
	DKPol	RM	Amazon	DBLP	Ethereum	Ripple	Bitcoin	Amazon-S	DBLP-S
η	1	1	1	3	1	1	1	1	3
μ	0.3	0.25	0.5	0.5	0.3	0.3	0.3	0.5	0.5
λ	5×10^{-7}	5×10^{-7}	5×10^{-7}	5×10^{-7}	5×10^{-7}	5×10^{-7}	5×10^{-7}	5×10^{-7}	5×10^{-7}

We implement ANOMULY with the GraphGym library [173] and use an NVIDIA V100 GPU in the experiments. We use the AUC (the area under the ROC curve) as the metric of comparison. The higher the AUC value, the higher the quality of the method.

4.3.2 Datasets

We use nine real-world public datasets [20, 78, 82, 97, 100, 127] whose domains cover social, brain, co-authorship, blockchain, and co-purchasing networks. We summarize their statistics in Table 4.2.

Social Networks. RM [97] has 10 networks, each with 91 nodes. Nodes represent phones and an edge exists if two phones detect each other in a mobile network. Each network describes connections between phones in a month. DKPol [78] is

Table 4.2: Network Statistics. Here $|\mathcal{V}|$ is the number of nodes, $|\mathcal{E}|$ is the number of edges in all views, and $|\mathcal{L}|$ is the number of views.

Dataset	Multiplex Networks						Single-layer Networks		
	RM	DKPol	Amazon	Ethereum	Ripple	DBLP	Bitcoin	Amazon-S	DBLP-S
$\|\mathcal{V}\|$	91	490	17.5K	221K	54K	513K	3.7K	8.6K	23K
$\|\mathcal{E}\|$	14K	20K	282K	473K	837K	1M	24.1K	90K	95.2K
$\|\mathcal{L}\|$	10	3	2	6	5	10	1	1	1

Twitter dataset collected during the month leading up to the 2015 Danish parliamentary election. Nodes are Twitter accounts of Danish politicians, and relations are "Retweet", "Reply", and "Topical Interaction" [78].

Co-purchasing Network. Amazon [82] is a co-purchasing network, where each node is an item and the type of connections are "Also-view" and "Also-bought". We focused on items with four categories, i.e., Beauty, Automotive, Patio Lawn and Garden, and Baby.

Co-authorship Network. DBLP is a co-authorship network until 2014 from dblp team [58] and pre-processed by Behrouz et al. [20]. In this dataset, each node is a researcher, an edge shows co-authorship, and each type of connection is a topic of research. For each collaboration, we consider the bag of words drawn from the titles of the paper and apply Latent Dirichlet Allocation (LDA) topic modeling [27] to automatically identify 240 topics. We then cluster their non-zero elements into ten known research topics. Each view in the network represents connections in one of the ten topics.

Blockchain Networks. Ethereum [127] is a blockchain transaction network over 576 days, where views are different tokens, nodes are addresses of investors, and edges denote the transferred token values. Since the same address may trade multiple tokens, the address appears in networks of all the tokens it has traded. Ripple [127] is derived from the Ripple Credit Network and covers a timeline of Oct-2016 to Mar-2020. Similar to the Ethereum dataset, nodes are investors and edges represent transactions. Here views (relation types) correspond to the five most issued fiat currencies on the Ripple network: JPY, USD, EUR, CCK, and CNY.

Single-layer Networks. DBLP-S is a subgraph of the multiplex DBLP network, where we collect a subset of researchers who work on data mining and related areas. Amozon-S, is a subgraph of the multiplex Amazon network, where we focus on the "Also-view" relation type. The Bitcoin dataset contains who-trusts-whom network of people who trade on the Alpha platforms [100].

Note that all of the datasets are anonymized and do not contain any personally identifiable information or offensive content.

4.3.3 Inject Anomalous Edges in Multiplex Networks

Since the ground truth for anomaly detection is difficult to obtain [9], we follow the methodology used in existing studies[9, 180, 188] and inject anomalous edges into our datasets. However, existing methods inject anomalous edges only in single-layer networks and cannot add complex anomalies in multiplex networks. Accordingly, here we divide injected edges into two groups (50% each), **(1)** view-independent anomalies, and **(2)** view-dependent anomalies. For the first group, we use existing methods [9], which randomly change one of the endpoints of some existing connections. In multiplex networks, the rich information about node connections leads to repetitions, meaning edges between the same pair of nodes repeatedly appear in multiple views. Repeated connections are more likely to represent a strong tie and may even suggest that the nodes in comprising these connections belong to the same community [20, 181]. Accordingly, in the second type of injected anomalies, we inject random connections that do not appear in any relation type. That is, we first choose a random edge (u, v) such that $(u, v) \notin \mathcal{E}_k$ for all $k \in \mathcal{L}$, and then we inject this edge to a random relation type $r \in \mathcal{L}$. Since this connection does not appear in any relation type, it is more likely to be an anomaly. This type of anomaly helps us to understand whether ANOMULY can take advantage of complementary information of different relation types.

4.3.4 Baselines

Since there is no prior work on edge anomaly detection in multiplex networks, we first compare ANOMULY with single-layer edge anomaly detection methods: GOutlier [3] builds a generative model for edges in a node cluster. CM-Sketch [142]

Table 4.3: Performance comparison of ANOMULY and baselines in multiplex networks (AUC).

Methods	RM		DKPol		Amazon		DBLP		Ethereum		Ripple	
Anomaly %	1%	5 %	1%	5 %	1%	5 %	1%	5 %	1%	5 %	1%	5 %
Single-layer Methods												
GOUTLIER	0.7138	0.6982	0.6844	0.6597	0.6973	0.6672	0.7059	0.6901	0.7017	0.6799	0.7036	0.6851
CM-SKETCH	0.7127	0.7012	0.7058	0.6930	0.6881	0.6719	0.7186	0.6915	0.7408	0.7277	0.7360	0.7194
NETWALK	0.7739	0.7641	0.7706	0.7581	0.7228	0.7122	0.7742	0.7523	0.7956	0.7885	0.7904	0.7823
ADDGRAPH	0.8005	0.8093	0.8149	0.8087	0.7796	0.7735	0.8024	0.7995	0.8133	0.8090	0.8205	0.8217
Multiplex Methods												
MNE	0.7994	0.7955	0.8050	0.7913	0.7108	0.7017	0.7532	0.7499	0.7541	0.7495	0.7813	0.7754
ML-GCN	0.7921	0.7886	0.7915	0.7907	0.7344	0.7263	0.7519	0.7439	0.7940	0.7918	0.8115	0.8072
ANOMULY	**0.8783**	**0.8729**	**0.8694**	**0.8610**	**0.8289**	**0.8195**	**0.8825**	**0.8754**	**0.8906**	**0.8852**	**0.8938**	**0.8871**
Improvement	9.71%	7.85%	6.68%	6.46%	6.32%	5.92%	9.98%	9.49%	9.50%	9.41%	8.93%	7.96%

uses a Count-Min sketch for approximating global and local structural properties. NetWalk [180] uses a random walk to learn a unified embedding for each node and then dynamically clusters the nodes' embeddings. AddGraph [188] is an end-to-end approach that uses an extended GCN in temporal networks. Finally, we compare with two multiplex network embedding baselines, ML-GCN [16] and MNE [183]. We apply K-means clustering on their obtained node embeddings for anomaly detection [180].

4.3.5 Results

Results on Multiplex Networks. We compare ANOMULY with the baseline methods on both dynamic and static multiplex networks with different percentages of anomalous edges (i.e., 1%, 5%). Table 4.3 reports the AUC for both the baselines and ANOMULY. Our method outperforms *all* baselines in all datasets and improves the best baseline results by 8.18% on average. There are three reasons for ANOMULY's superior performance: **(1)** it outperforms competitors for static datasets, because it can learn structural anomaly patterns in the network, rather than depending on pre-defined patterns/roles. **(2)** ANOMULY outperforms single-layer methods due to its attention mechanism that incorporates complementary information from different relation types. **(3)** ANOMULY outperforms multiplex methods as it is an end-to-end method and is optimized for anomaly detection. It is also a

Table 4.4: Performance comparison of ANOMULY and baselines in single-layer networks (AUC).

Methods	Bitcoin		Amazon-S		DBLP-S	
Anomaly %	1%	5 %	1%	5 %	1%	5 %
GOUTLIER	0.7143	0.7091	0.6923	0.6614	0.7108	0.6995
CM-SKETCH	0.7146	0.7015	0.7049	0.6621	0.7084	0.6877
NETWALK	0.8375	0.8367	0.7483	0.7302	0.7779	0.7590
ADDGRAPH	0.8534	0.8416	0.7872	0.7828	0.7911	0.7932
ANOMULY	**0.8707**	**0.8661**	**0.8014**	**0.7943**	**0.8129**	**0.8236**
Improvement	2.03%	2.91%	1.80%	1.47%	2.76%	3.83%

Table 4.5: Ablation study on ANOMULY (AUC). The first row replaces the GRU with an MLP. The second row removes the attention mechanism, and the third row replaces the attention mechanism with SUM(.).

Methods	Amazon	DBLP	Ethereum
ANOMULY	**0.8289**	**0.8825**	**0.8906**
w/ MLP	0.8023	0.8606	0.8718
w/o Attention	0.7831	0.8219	0.8275
w/ Summation	0.8007	0.8571	0.8688

dynamic method and can take advantage of the temporal properties of the network. The ablation study below illustrates these effects in more detail.

Results on Single-layer Dynamic Networks. We also compare ANOMULY with single-layer method baselines on single-layer dynamic datasets. Table 4.4 summarizes the results. Once again, we see that ANOMULY outperforms all the baselines, even in single-layer graphs, by 2.46% on average. This is mainly due to *Snapshot Encoder*'s architecture, which enables our method to incorporate the outputs of GNN layers after each view and recurrently update them over time using a GRU cell.

Parameter Sensitivity. We evaluate the effect of the training ratio of the dataset:

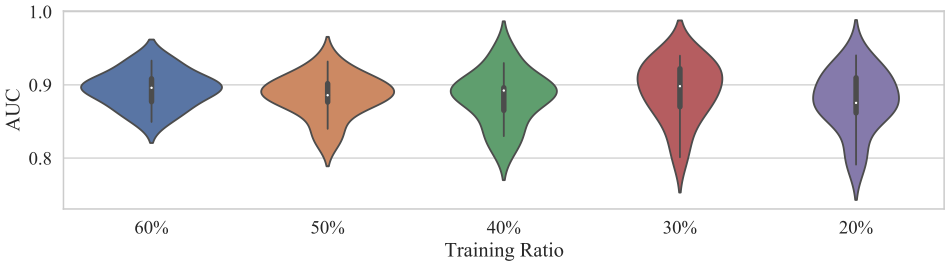


Figure 4.2: Stability over different training ratios on Ripple.

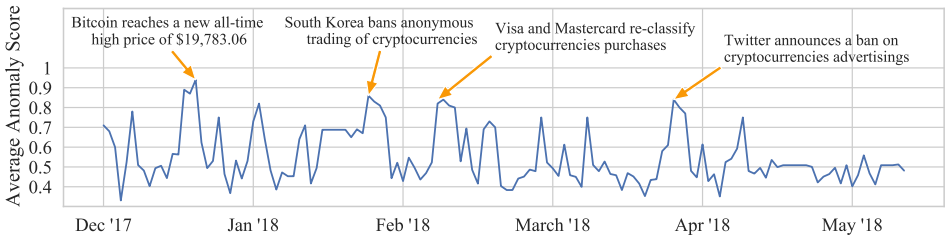


Figure 4.3: Event detection in Ethereum network. The top-4 local maximums all coincide with major events annotated in the figure

we change the training ratio from 60% to 20% and report the results on the Ripple dataset in Figure 4.2. Decreasing the training ratio tends to increase both the average and maximum AUC, except for the 20% case, and decreases the minimum AUC for all training ratios. These results show that performance stays relatively stable, which demonstrates that our framework is robust in the presence of a small amount of training data.

Ablation Study. Next, we conduct experiments to show that the ANOMULY architecture design is effective in boosting performance. We examine the effect of GRU cells by replacing them with a 2-layer MLP. We also investigate the effect of the attention mechanism by **(1)** removing the attention, learning node embeddings in each relation-type separately, and **(2)** aggregating the information of different relation types by summation (without weights). Table 4.5 summarizes the results. We found that both our attention mechanism and the GRU cells are important for

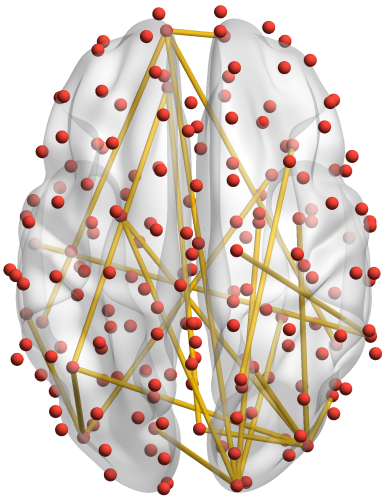


Figure 4.4: Anomalous edges in the brain network of a sampled individual.

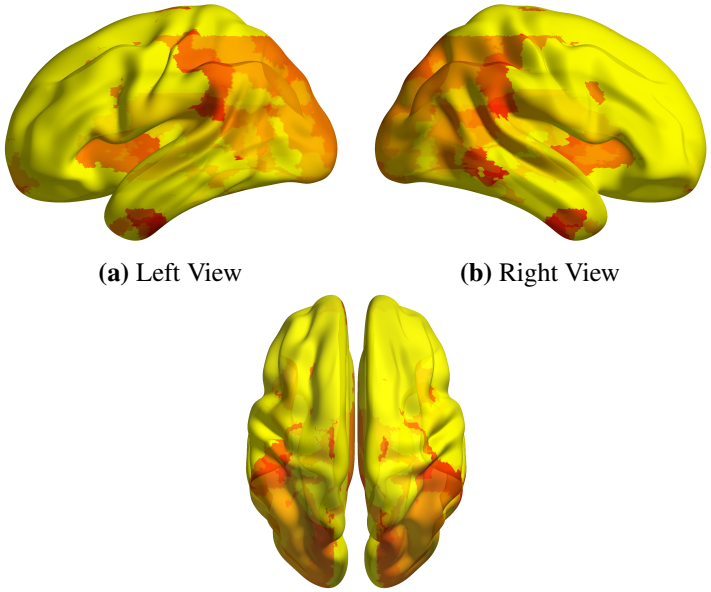
ANOMULY, producing significant performance boosts.

Effectiveness in Detecting Events. Next, we evaluate how well ANOMULY detects events in the Ethereum transaction network. In each timestamp, we calculate the anomaly score for all the edges in the snapshot. We then compute the average of the top-15 edge anomaly scores and report them in Figure 4.3. We find that the top-4 local maximums all coincide with major events annotated in the figure.

Case Study of Brain Networks. Behavioral disturbances in Attention Deficit Hyperactivity Disorder (ADHD) are thought to be caused by the dysfunction of spatially distributed, interconnected neural systems [74]. We used ANOMULY to detect anomalous connections in the brain network of people with ADHD. We use the ADHD-Brain dataset [30] derived from the functional magnetic resonance imaging (fMRI) of 40 individuals–20 individuals in the condition group, labeled ADHD, and 20 individuals in the control group, labeled Typically Developed (TD)–using the same methodology used by Lanciano et al. [101]. Here, each view (relation type) is the brain network of an individual person, where nodes are brain regions, and each edge measures the statistical association between the functionality of its endpoints.

Anomalous functional correlations between the brain network of people with

ADHD compared to those without can help us understand which brain regions are involved in ADHD. We present two results: **(1)** 74% of all detected anomalies are connections in the brain networks of people in the ADHD group. **(2)** 69% of all found anomalies in the ADHD group correspond to edges in the frontal and occipital cortex of the brain. Figure 4.4 illustrates the anomalous edges in the brain network of an individual in the ADHD group. These findings show an unexpected functional correlation of occipital and frontal lobes regions with other parts, which are consistent with previous studies on ADHD [42, 158].



(a) Left View **(b)** Right View

(c) Up View

Figure 4.5: Distribution of anomalous edges in ADHD group.

Next, we investigate how anomalous edges found by ANOMULY are distributed in the brain. Figure 4.5 reports the average distribution of anomalous edges in the brain networks of people living with ADHD. Most anomalous edges found by ANOMULY have a vertex in the Occipital lobes. Moreover, the Temporal lobes are the brain regions with the most anomalous connections with the Occipital lobes. These findings can help to reveal new insights into understanding ADHD and the regions of the brain that are connected to its symptoms. These findings also show the potential of ANOMULY to extract features that can help predict brain diseases

or disorders.

In Chapter 6, we further discuss the application of anomaly detection in understanding the abnormal brain activities that might cause a brain disease or disorder.

4.4 Conclusion

We present ANOMULY, an end-to-end unsupervised framework for detecting edge anomalies in dynamic multiplex networks. ANOMULY is based on a new architecture that uses GNN and GRU cells to take advantage of both temporal and structural properties and adds an attention mechanism that effectively incorporates information across different types of connections. Finally, it uses a negative sampling approach in training to overcome the lack of ground truth data. Extensive experiments show the power of ANOMULY to effectively identify temporal and structural anomalies in both single-layer and multiplex networks. Our case studies on brain networks and blockchain transaction networks show the usefulness of ANOMULY in widely varying domains.

Chapter 5

ADMIRE: Inductive and Scalable Anomaly Detection in Multiplex Dynamic Networks

5.1 Introduction

While ANOMULY achieves state-of-the-art performance in all datasets, it suffers from several limitations: ① Transductive learning: The ANOMULY framework is designed in a transductive setting and cannot be applied to unseen nodes/patterns. That is, ANOMULY in the message-passing procedure uses the identity of nodes in training, limiting its ability to predict anomalous connections between nodes that are unseen in the training phase. ② Memory and scalability: The ANOMULY framework is snapshot-based. That is, it requires storing the entire snapshot of the temporal network at each timestamp, which consumes a great deal of memory. Moreover, since it uses different GNN modules for each type of connection, it cannot be used for multiplex networks with a large number of views. ③ Lack of generalizability: The ANOMULY framework uses a simple negative sampling method by randomly changing one endpoint of a connection to learn anomalous interactions. While this negative sampling method is fast and lets the model be trained in an unsupervised manner, these negative sample generator methods are

too simple and can cause poor performance in more complicated datasets [138].

We now present ADMIRE, an inductive, scalable, unsupervised method that learns the underlying dynamic laws of the multiplex dynamic networks to detect anomalous patterns, thus addressing the limitation of ANOMULY. ADMIRE uses two different casual multiplex walks, *inter-view* and *intra-view*, to automatically extract and learn temporal network motifs. It then uses an anonymization strategy to hide node and relation type identities, making the model inductive. ADMIRE is a streaming method, requiring only constant memory (see Section 5.2.1). Moreover, its random walk encoder scales to multiplex networks with more than 100 views. ADMIRE introduces a novel negative sampling method for multiplex networks that addresses the generalizability issue in ANOMULY.

5.2 ADMIRE

The main intuition behind ADMIRE is to use multiplex temporal walks as a proxy for temporal motifs in multiplex networks and extract the causality of the existence of an edge in a specific type of connection. After extracting temporal multiplex motifs, we use an anonymization process to hide nodes' identities in walks, keeping the model inductive in the training phase. Next, we design a simple yet effective walk encoding method to encode each walk. Using encoded walks, we define the node encoding of a vertex u as the aggregation of all walks started from u. Using the obtained node encoding, we classify connections based on the encoding of their endpoints.

5.2.1 Anonymous Multiplex Temporal Walk

We define the multiplex temporal walk as a walk that starts from a connection of interest and backtracks over time by traversing adjacent edges. By traversing adjacent edges in reverse time sequence, this allow us to discover and encode the underlying causality of network dynamics. The main challenge in extracting the causality of an edge in multiplex networks is determining whether there is a causal relationship between different views. In multiplex networks, different views provide complementary information about the network and each vertex. However, this does not always mean that there is a causal relationship between different views. For

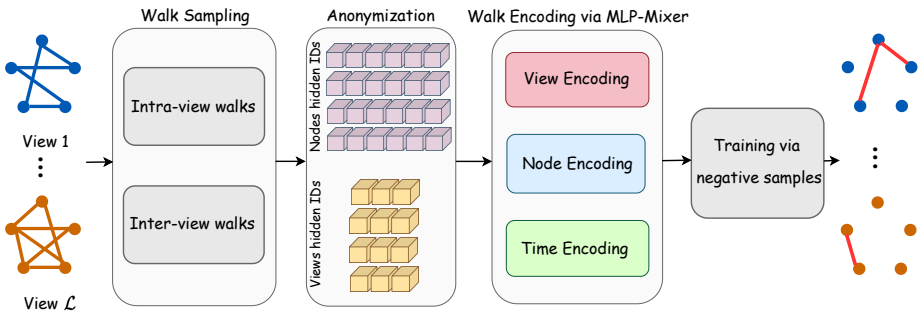


Figure 5.1: Schematic of the ADMIRE model. ADMIRE consists of four main stages called (1) Walk Sampling, (2) Anonymization, (3) Walk Encoding, and (4) Training via generating negative samples.

example, consider the flight transportation network [34], where each node shows a city, and each view shows flights operated by a specific airline. Two cities in view r are connected if there is a flight between them operated by airline r. In this example, there is no causal relationship between different views, as flights in different views are operated by different airlines. To address this challenge, we design two multiplex temporal walks, intra-view, and inter-view walks, to extract the causality within a specific view and across different views, respectively. When there is no causal relationship between different views, we expect the model to ignore inter-view walks.

Intra-view Temporal Walk. An intra-view walk is a time-ordered random walk among edges of a specific type (i.e., within a single view). Given a type of relation r, an intra-view temporal walk W^r_{intra} on a view of a temporal multiplex network can be represented as:

$$W^r_{\text{intra}} = ((u_0, r, t_0), (u_1, r, t_1), \ldots, (u_m, r, t_m))\,; \;\; t_0 \geq t_1 \geq \cdots \geq t_m, \tag{5.1}$$

where $(u_i, u_{i+1}, r, t+1) \in \mathcal{E}$. Note that we only allow walks that progress backward in time, to limit ourselves to possible sources of causality. Since different views still provide complementary information about the global property of the network as well as the local structure around each node, in Section 5.2.2, we use an attention mechanism that incorporates the information provided by intra-view

walks in different types of connections.

Inter-view Temporal Walk. To capture the correlation between different views and extract the causality of an edge from different views of the network, in the inter-view temporal walk, we let the walker walk across views. Accordingly, an inter-view temporal walk W_{inter} on temporal multiplex networks can be represented as:

$$W_{\text{inter}} = ((u_0, r_0, t_0), (u_1, r_1, t_1), \ldots, (u_m, r_m, t_m))\,;\ t_0 \geq t_1 \geq \cdots \geq t_m, \quad (5.2)$$

where $(u_i, u_{i+1}, r_{i+1}, t+1) \in \mathcal{E}$. That is, not only does the walker walk over time and capture the temporal causality of an edge, but also can walk over different views to capture the dependencies of connections in different views, taking advantage of complementary information provided by different types of relations. Note that r_is are not necessarily distinct. We let $W_{\text{inter}}(i)$ denote the i-th element of the temporal multiplex walk, (u_i, r_i, t_i). Also, we use $W_{\text{inter}}(i, 0), W_{\text{inter}}(i, 1)$, and $W_{\text{inter}}(i, 2)$ to refer to u_i, r_i, and t_i, respectively.

How to sample temporal multiplex walk? As discussed in previous studies, newer connections in temporal networks are often more informative [88, 163]. To this end, we use a biased sampling method with hyperparameter μ to control the importance of recent connections. Given the time of a previously sampled edge, t_0, we sample an adjacent edge at time $t < t_0$ with probability proportional to $\exp(\mu(t - t_0))$. In multiplex networks, the correlation of different pairs of views can be different [17, 130] and for connections in a given view r, a subset of views might play more important roles in causality extraction. Accordingly, in inter-view temporal walks, we use a biased sampling method and sample link (u'_1, u'_2, r', t') after previously sampled link (u_1, u_2, r, t) with probability proportional to $\varphi(r, r')$. In fact, $\varphi(r, r')$ shows the importance of view r for view r'. In §5.2.2, we discuss how to calculate $\varphi(r, r')$.

The first step in our sampling is to compute the sampling probability of an incoming connection in relation type r. For an incoming edge $e = (u, v, r, t)$ we

Algorithm 2 Temporal multiplex walk sampling procedure

Input: The edge set $\mathcal{E}$, previously sampled node w_p in view r_p at time t, and hyperparameter μ

Output: Next sampled connection (w_n, w_p, r_n, t_n)

1: **for** $e = (w_n, w_p, r_n, t_n) \in \mathcal{E}^{t_p}(w_p)$ **do**
2: Sample $b \sim$ UNIFORM$(0, 1)$;
3: **if** $b < q_{r_p}^{t_p}(w_p, e) \times \varphi(r_p, r_n)$ **then**
4: **return** $e = (w_n, w_p, r_n, t_n)$;
5: **return** (e_X, t_X); $\triangleright$ (e_X, r_X, t_X) is a dummy empty edge signaling the end of the algorithm.

compute the probabilities $q_r^t(w)$ for $w \in \{u, v\}$ as follows:

$$q_r^t(w, e) = \frac{\exp(\mu t)}{\sum_{(w_0, t') \in \mathcal{N}_r^t(w)} \exp(\mu t')}, \tag{5.3}$$

where $\mathcal{N}_r^t(w)$ represents the set of w's neighbor in view r and before time t. This probability needs to be computed one time when the connection arrives and does not need to be updated anymore. Also, for calculating the probability of sampling this connection after a connection from another relation type r', we simply multiply this probability by $\varphi(r, r')$.

Algorithm 2 shows the sampling procedure. Given a previously sampled connection in view r at time t, we sample the next connection in view r' at time $t' < t$ with a probability proportional to $\exp(\mu(t' - t)) \times \varphi(r, r')$. It is not hard to show that Algorithm 2 samples the next connection with a probability proportional to $\exp(\mu(t_n - t_p)) \times \varphi(r_p, r_n)$. Inspired by Wang et al. [163], in our experiments, we store the k most recent connections with $k \propto \mathcal{O}\left(\frac{1}{\mu}\right)$. The intuition is that if we sort connections in $\mathcal{E}^t(w_p)$ by their timestamp $\{t_i\}_{i=1}^h$, and assume that $\exp(\mu(t_i - t))$ are i.i.d., the probability of sampling the j-th connection is:

$$\mathbb{P}[\text{sampling } j\text{-th connection}] = \frac{\prod_{i=1}^{j} \exp(\mu(t_i - t)) \times \varphi(r_p, r_i)}{\sum_{i=1}^{h} \prod_{s=1}^{i} \exp(\mu(t_s - t)) \times \varphi(r_p, r_s)}.$$

This probability decreases when we increase the value of j. Accordingly, in practice, we need to store only a constant number of the most recent connections at each

time.

Given a (potential) link (u, v, r, t), we use the above procedure to generate M inter-view and M' intra-view walks with m steps starting from each of nodes u and v. We use $\mathcal{S}_{\text{inter}}(u)$, $\mathcal{S}_{\text{intra}}(u)$, $\mathcal{S}_{\text{inter}}(v)$, and $\mathcal{S}_{\text{intra}}(v)$ to store started walks from u and v, respectively.

Anonymization Process. Recent studies argue that traditional anonymization methods (e.g., [121]), where we assign node IDs based on only one walk, suffer from several limitations (e.g., missing the correlation of different walks) and suggest using an anonymization process that assigns hidden node IDs based on the correlation of sampled walks [88, 163]. On the other hand, in multiplex networks, we need to hide the identity of both nodes and views (e.g., relation types) to keep the model inductive. Given a (potential) link (u, v, r, t), let $w_0 \in \{u, v\}$. To capture the correlation across different walks, which could be a key to reflecting the network dynamics [163], for a given node w that appears on at least one walk in $\mathcal{S}_{\text{inter}}(u) \cup \mathcal{S}_{\text{inter}}(v)$, we use a relative vector $\mathcal{C}(\mathcal{S}_{\text{inter}}(w_0), w) \in \mathbb{Z}^{m+1}$ that represents the number of times in $\mathcal{S}_{\text{inter}}(w_0)$ that node w appears at certain positions. That is,

$$\mathcal{C}_i\left(\mathcal{S}_{\text{inter}}(w_0), w\right) = \left|\{W_{\text{inter}} | W_{\text{inter}} \in \mathcal{S}_{\text{inter}}(w_0), w = W_{\text{inter}}(i, 0)\}\right|, \forall 0 \leq i \leq m. \tag{5.4}$$

Similarly, we define $\mathcal{C}(\mathcal{S}_{\text{intra}}(w_0), w)$ over intra-view temporal walks. Until now, node identities were accessible, but we now remove node identities and use only these four vectors in the training phase to represent each node, thereby hiding their identity:

$$\textsc{Id}(w) = \{\mathcal{C}(\mathcal{S}_{\text{inter}}(u), w), \mathcal{C}(\mathcal{S}_{\text{inter}}(v), w), \mathcal{C}(\mathcal{S}_{\text{intra}}(u), w), \mathcal{C}(\mathcal{S}_{\text{intra}}(v), w)\}. \tag{5.5}$$

We hide the identity of relations, r, in a similar manner. Given a set of walks (e.g., $\mathcal{S}_{\text{inter}}(w_0)$), we count the number of times we see a relation type at each position when we start from a specific relation type; this captures the correlation among different views. For a given relation type r, we use a relative vector $\mathcal{C}^{\text{view}}(\mathcal{S}_{\text{inter}}(w_0), r) \in \mathbb{Z}^{m+1}$ that counts times in $\mathcal{S}_{\text{inter}}(w_0)$ that a relation with type

r appears at each position:

$$\mathcal{C}_i^{\text{view}}\left(\mathcal{S}_{\text{inter}}(w_0), r\right) = \left|\{W_{\text{inter}} | W_{\text{inter}} \in \mathcal{S}_{\text{inter}}(w_0), r = W_{\text{inter}}(i,1)\}\right|, \forall 0 \leq i \leq m. \tag{5.6}$$

In this way, $\text{ID}^{\text{view}}(r) = \{\mathcal{C}^{\text{view}}\left(\mathcal{S}_{\text{inter}}(u), r\right), \mathcal{C}^{\text{view}}\left(\mathcal{S}_{\text{inter}}(v), r\right)\}$ hides the identity of view r. Note that, although intra-view walks are within a single view, we still need to hide the identity of that view, and we use the same $\text{ID}^{\text{view}}(r)$ as above.

5.2.2 Neural Encoding

We next present our fast and simple, yet effective and generalizable, neural network to encode temporal multiplex walks so that we can extract structural and temporal information from the network. The intuition of this neural encoding is to use anonymous temporal multiplex walks to learn the structural and temporal properties as well as casual rules of the network in an inductive manner.

Most existing methods on walk encoding treat a walk as a sequence of vertices and use sequence encoders such as RNNs or TRANSFORMERS to encode each walk. The main drawback of these methods is that they fail to directly process temporal walks with irregular gaps between timestamps. That is, sequential encoders can be seen as discrete approximations of dynamic systems; however, this discretization often fails if we have irregularly observed data [96]. We present a neural network to encode temporal multiplex walks so that we can extract structural and temporal information from the network with continuous time dynamics. The process consists of ① a time encoding module to encode the time, ② a node encoding module to encode the position of vertices, ③ a view encoding module to encode relation type, and ④ a walk encoding module to encode each extracted motif.

Time Encoding. Existing methods in temporal graph learning [49, 163] use random Fourier features [95] to encode time. However, this approach captures only periodicity in the data, while in many domains, e.g., brain activity patterns, we also need to learn non-periodic patterns dependent on the progression of time (e.g., in task-based fMRI). We address this need by adding a learnable linear term to the feature representation of time encoding. That is, we encode a given time t as:

$$\mathscr{T}(t) = \left(\boldsymbol{\omega}_l t + \mathbf{b}_l\right) || \cos(t\boldsymbol{\omega}), \tag{5.7}$$

where $\boldsymbol{\omega}_l, \mathbf{b}_l \in \mathbb{R}$ and $\boldsymbol{\omega} \in \mathbb{R}^d$ are learnable parameters, and $||$ denotes concatenation.

Node Encoding. We now define a node encoding function $\zeta(.)$ that encodes each node w based on $\textsc{ID}(w)$. Since the concept and task of intra-view and inter-view walks are different, we first break the $\zeta(.)$ function into $\zeta_{\text{intra}}(.)$ and $\zeta_{\text{inter}}(.)$, respectively, and then interpolate between them by a learnable parameter λ to obtain $\zeta(.)$.

For each node w that appears on at least one walk in $\mathcal{S}_{\text{inter}}(u) \cup \mathcal{S}_{\text{inter}}(v)$, we use *one* simple MLP to encode the w's hidden identities:

$$\zeta_{\text{inter}}(w) = \texttt{MLP}\left(\mathcal{C}(\mathcal{S}_{\text{inter}}(u), w)\right) + \texttt{MLP}\left(\mathcal{C}(\mathcal{S}_{\text{inter}}(v), w)\right). \tag{5.8}$$

While inter-view walks naturally capture the causal relationship and correlation between different types of connections, intra-view walks have a different role and capture causality within a single view. However, we need to aggregate the information provided by inter-view walks in different views to take advantage of complementary information in multiplex networks. As discussed in Section 4.2.3, the importance of each view for each vertex can be different; we use the attention mechanism introduced in Section 4.2.3 to learn the importance of views r for vertex u, $\psi(w, r)$.

$$\psi(w, r) = \frac{\exp\left(\sigma\left({\mathbf{s}_r}^T \mathbf{W}^{\text{att}}\ \zeta^r_{\text{intra}}(w)\right)\right)}{\sum_{k=1}^{\mathcal{L}} \exp\left(\sigma\left({\mathbf{s}_k}^T \mathbf{W}^{\text{att}}\ \zeta^k_{\text{intra}}(w)\right)\right)}, \tag{5.9}$$

where $\mathbf{s}_r$ is a summary of the network in relation type r, i.e., $\mathbf{s}_r = \sum_{u \in \mathcal{V}} \zeta^r_{\text{intra}}(u)$, and $\mathbf{W}^{\text{att}}$ is a trainable weight matrix. In our experiments, we use $\tanh(.)$ as the activation function $\sigma(.)$. Similarly, we define the importance of node w for view r as follows:

$$\psi(r, w) = \frac{\exp\left(\sigma\left({\mathbf{s}_r}^T \mathbf{W}^{\text{att}}\ \zeta^r_{\text{intra}}(w)\right)\right)}{\sum_{w_0 \in \mathcal{V}} \exp\left(\sigma\left({\mathbf{s}_r}^T \mathbf{W}^{\text{att}}\ \zeta^r_{\text{intra}}(w_0)\right)\right)}, \tag{5.10}$$

In these equations, $\zeta^r_{\text{intra}}(w)$ is the view-based node encoding of w in view r, which we define as follows: Given a relation type $r' \in \mathcal{L}$, we define view-based node

encoding $\zeta^{r'}_{\text{intra}}(w)$ as:

$$\zeta^{r'}_{\text{intra}}(w) = \texttt{MLP}\left(\mathcal{C}(\mathcal{S}^{r'}_{\text{intra}}(u), w)\right) + \texttt{MLP}\left(\mathcal{C}(\mathcal{S}^{r'}_{\text{intra}}(v), w)\right). \tag{5.11}$$

Next, we aggregate these node embeddings to incorporate information from different views and obtain $\zeta_{\text{intra}}(w)$:

$$\zeta_{\text{intra}}(w) = \sum_{r' \in \mathcal{L}} \psi(w, r')\zeta^{r'}_{\text{intra}}(w). \tag{5.12}$$

Now, we use a learnable parameter λ to automatically learn the importance of each $\zeta_{\text{intra}}(w)$ and $\zeta_{\text{inter}}(w)$ based on the data. This formulation lets our model learn to interpolate between Equation 5.8 and Equation 5.12, which enables it to be flexible in whether causal relationship between views exists or not. Therefore, $\zeta(w)$ is defined as:

$$\zeta(w) = \zeta_{\text{intra}}(w) + \lambda \times \zeta_{\text{inter}}(w). \tag{5.13}$$

When there is no causal relation between different views, our model is expected to set $\lambda \approx 0$ (see §5.4).

View Encoding. For each view $r \in \mathcal{L}$, we use *one* simple MLP to encode the r's hidden identities:

$$\eta(r) = \texttt{MLP}\left(\mathcal{C}^{\text{view}}(\mathcal{S}_{\text{inter}}(u), r)\right) + \texttt{MLP}\left(\mathcal{C}^{\text{view}}(\mathcal{S}_{\text{inter}}(v), r)\right). \tag{5.14}$$

As discussed in Section 5.2, we use the importance of view r for view r', $\varphi(r, r')$, to sample the next connection in inter-view walks. In fact, $\varphi(r, r')$ measures how similar these two views are. Given the vector representation of views, we use the inner product to measure the similarity of different views. Now based on the obtained view encoding, $\eta(r)$, we define:

$$\varphi(r, r') = \frac{\eta(r).\eta(r')}{\sum_{\hat{r} \in \mathcal{L}} \eta(r).\eta(\hat{r})}. \tag{5.15}$$

Walk Encoding. Given a walk $\hat{W} \in \{W_{\text{inter}}, W_{\text{intra}}\}$, we use node encoding function $\zeta(.) : \mathbb{Z}^{(m+1)\times 4} \rightarrow \mathbb{R}^{k_1}$ to encode hidden node identities and $\eta(.)$:

$\mathbb{Z}^{(m+1)\times 2} \rightarrow \mathbb{R}^{k_2}$ to encode hidden view identities. We then concatenate their outputs with the embedding of the node's corresponding timestamp. Finally, we use an MLP-Mixer [151] to mix these encodings to obtain the walk encoding:

$$\textsc{Enc}(\hat{W}) = \textsc{Mean}\left(\mathbf{H}_{\text{token}} + \mathbf{W}^{(2)}\sigma\left(\texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right)\mathbf{W}^{(1)}\right)\right), \quad (5.16)$$

where the i-th row of $\mathbf{H}_{\text{token}}$ is defined as:

$$\mathbf{H}^{i}_{\text{token}} = \left[\zeta\left(\textsc{Id}\left(\hat{W}(i,0)\right)\right) \,||\, \eta\left(\textsc{Id}^{\text{view}}\left(\hat{W}(i,1)\right)\right) \,||\, \mathscr{T}(t_i)\right]. \quad (5.17)$$

In the above equations, $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(1)}_{\text{token}}$, and $\mathbf{W}^{(2)}_{\text{token}}$ are learnable parameters, `LayerNorm` is layer normalization [10] and $\sigma(.)$ is a nonlinear function (e.g., Gaussian error linear units, GeLU [84]).

Anomaly Score. To assign an anomaly score to a given link $e = (u, v, r, t) \in \mathcal{E}$, we first sample temporal multiplex walks and then encode each walk $W \in \mathcal{S}_{\text{inter}}(u) \cup \mathcal{S}_{\text{inter}}(v) \cup \mathcal{S}_{\text{intra}}(u) \cup \mathcal{S}_{\text{intra}}(v)$ as described above. Next, we use an $\textsc{Agg}(.)$ function (e.g., mean-pooling) to aggregate walks' encodings and encode link e. Finally, we use a 2-layer perceptron to make the anomaly score:

$$\Upsilon(e) = \texttt{MLP}\left(\frac{1}{M + M'}\sum_{\hat{W}} \textsc{Enc}(\hat{W})\right), \quad (5.18)$$

where M and M' are the numbers of inter-view and intra-view walks.

5.2.3 ADMIRE Framework

We next explain how we use the view-aware edge encoding method to detect anomalous interactions. Figure 5.1 illustrates the ADMIRE framework.

Negative Sample Generator. We generate negative samples (NS) to train ADMIRE in an unsupervised manner. Previous anomaly detection methods mostly use (simple or biased) random negative samples [17, 188], which limit their generalizability to real anomalous patterns [138]. Moreover, these methods are designed for simple networks and cannot generalize to anomalous patterns in multiplex networks. Inspired by Poursafaei et al. [138], we design a novel negative sampling method for

temporal *multiplex* networks.

Let $\mathcal{E}_{\text{train}}$ and $\mathcal{E}_{\text{t}} \subseteq \mathcal{E}_{\text{train}}$ be the set of edges in the training set and in timestamp t, respectively. For each edge in the training set $e = (u, v, r, t) \in \mathcal{E}$, we generate three types of negative samples: ① Inter-view negative samples: We use these negative samples so our model learns to detect connections that are anomalous across different views. We randomly generate a negative connection with relation type r with probability inversely proportional to the number of views (or the summation of views' weight, in the case that the importance of views are different) in which this connection appears. The intuition is that if two nodes are already connected with several types of connections, a connection of yet another type is unlikely to be an anomalous connection. ② Intra-view negative samples: Here, we follow previous negative sampling generation methods [17, 188] and randomly change one endpoint of a connection to another node and keep the type of connection unchanged. ③ Historical negative samples: we generate negative edges from the set of edges that have been observed during previous timestamps but are absent in the current timestamp. That is, we randomly sample an edge $e \in \mathcal{E}_{\text{train}} \cap \bar{\mathcal{E}}_{\text{t}}$.

Training and Loss Function. Let $\mathcal{E}_{\text{train}}$ be the set of edges in the training set and $\mathcal{E}_{\text{neg}}$ be the set of generated negative samples. For each edge $e \in \mathcal{E}_{\text{train}} \cup \mathcal{E}_{\text{neg}}$ we generate temporal multiplex walks to find a view-aware edge encoding of e. Next, we use the margin-based pairwise loss [28] to train the model. To avoid overfitting, we also use an $L2$-regularization loss, $\mathscr{L}_r^{reg}$, which is the summation of the $L2$ norm of all trainable parameters. This produces the loss function:

$$\mathscr{L} = \sum_{(u,v,r,t)\in\mathcal{E}_{\text{train}}} \sum_{(u',v',r,t)\in\mathcal{E}_{\text{neg}}} \max\left\{0, \gamma + \Upsilon(u, v, r, t) - \Upsilon(u', v', r, t)\right\} + \lambda\mathscr{L}^{reg}, \tag{5.19}$$

where $0 \leq \gamma \leq 1$ is the margin between normal and negative sampled edges.

5.3 ADMIRE++: Explainable Anomaly Detection in Multiplex Dynamic Networks

To apply ADMIRE in real-world sensitive applications (e.g., health-related domains or financial networks), it is not sufficient to simply make accurate predictions; domain experts often want to understand *why* the given model made the prediction it did. We design a post-hoc explainability method to mimic the predictions of ADMIRE. Recall that the main intuition behind ADMIRE is to extract the causality of network dynamics by backtracking over time, extracting network multiplex motifs. We use these extracted motifs to describe the neighborhood of each node and then use this data to train a decision tree model. Our goal is that the decision tree can explain why a connection is abnormal based on the neighborhoods of its endpoints.

The main challenge is that we have diverse network motifs in large and dense networks, so we need many features to accurately describe the neighborhood of each node. We use k-mean clustering [112] on the walk embeddings obtained in Section 5.2.2, grouping network motifs based on their similarity. Clearly, bigger values for k produce better explanations.

Let M_{inter} and M_{intra} be the set of sampled walks started from the endpoints of all connections in the training set. We encode all walks $w \in M_{\text{inter}} \cup M_{\text{intra}}$ by using the procedure introduced in Section 5.2.2. Next, we apply k-mean clustering [112] to obtain $\mathscr{C}_1, \ldots,$ and $\mathscr{C}_k$. Then, for each connection $e = (u_1, u_2, r) \in \mathcal{E}_{\text{train}}$, we construct feature vector $\mathbf{v}_e = (\mathbf{v}_{u_1} \quad \mathbf{v}_{u_2})$ as follows:

$$\mathbf{v}_e = \left(\underbrace{p_1^1 \;\; p_2^1 \;\; \ldots \;\; p_k^1}_{\mathbf{v}_{u_1}} \;\; \underbrace{p_1^2 \;\; p_2^2 \;\; \ldots \;\; p_k^2}_{\mathbf{v}_{u_2}} \right), \tag{5.20}$$

where p_i^1 and p_i^2 are normalized counting number of sampled walks (motifs) starting from u_1 and u_2 in cluster i. That is, $p_i^1 = \frac{C_i^1}{C}$, where C_i^1 is the number of sampled walks starting from u_1 that are in cluster $\mathscr{C}_i$, and C is the total number of sampled walks. Similarly, $p_i^2 = \frac{C_i^2}{C}$, where C_i^2 is the number of sampled walks starting from u_2 that are in cluster i. Therefore, in this design, p_i^1s and p_i^2s describe the distribution of motifs in the neighborhoods of nodes u_1 and u_2, respectively.

Given the set of connections in the training set $\mathcal{E}_{\text{train}}$, let $\mathbf{V} = \{\mathbf{v}_e | e \in \mathcal{E}_{\text{train}}\}$, and $\mathcal{M}$ be the ADMIRE model trained on $\mathcal{E}_{\text{train}}$. We use $\mathcal{M}(e)$ to represent the prediction of ADMIRE for e. We use $\psi(r, u)$ to denote the importance of node u for view r (see Equation 5.10), and based on that, we define the importance of connection $e = (u, v, r)$ as $\psi(e) = \frac{\psi(r,u)+\psi(r,v)}{2}$. Finally, we use Ψ to denote the set of $\psi(e)$ for $e\ in \mathcal{E}_{\text{train}}$. Let $\mathbf{y} = \{y_e \in \{\text{“Normal”, “Anomaly”}\} | \mathcal{M}(e) = y_e\}$, we construct a dataset $D_{\text{tree}} = (\mathbf{V}, \mathbf{y}, \Psi)$ to train a weighted optimal decision tree. Here, the constructed feature for each connection is a data sample, the prediction of ADMIRE for this data sample is the label, and $\psi(e)$ is the weight of each sample. Finally, we train our decision tree on this dataset via the algorithm from Behrouz et al. [21]; the sparsity in optimal sparse decision trees guarantees interoperability and the algorithm guarantees the performance quality of the model [21].

To choose the value of k, we need to consider the number of walks, the number of samples, and the computing resources. That is, larger k leads to more accurate clustering and better performance, accordingly; however, it can significantly affect the running time of the weighted sparse decision tree algorithm. To this end, we tune $2 \times k$ as the maximum number of features that the weighted sparse decision tree algorithm can find the optimal solution in less than a time threshold.

Inference. In the inference phase, for each connection $e = (u, v, r)$ we first sample inter-view and intra-view walks starting from u and v. Based on the sampled walks, we construct the feature $\mathbf{v}_e$ using Equation 5.20. Finally, we use the constructed feature vector to make a prediction for e by the trained optimal tree. Since the model is an interpretable tree the path of prediction explains ADMIRE's prediction.

5.4 Experiments

5.4.1 Experimental Setup

We tune hyper-parameters by cross-validation and search the hyper-parameters over ① $\mu \in \{0.5, 1, 2, 4\} \times 10^{-5}$, ② Inter-view sampling number $M \in \{32, 64, 128, 256\}$, ③ Intra-view sampling number per view $M' \in \{8, 16, 32, 64\}$, ④ Walk length $m \in \{2, 4, 8, 12\}$. In training, we use a learning rate of 0.0001, hidden dimension 100 in MLP-Mixer, and batch size of 600.

Table 5.1: Performance comparison of ADMIRE and baselines in multiplex networks (AUC).

Methods	RM		DKPol		Amazon		DBLP		Ethereum		Ripple	
Anomaly %	1%	5 %	1%	5 %	1%	5 %	1%	5 %	1%	5 %	1%	5 %
Single-layer Methods												
GOUTLIER	0.7138	0.6982	0.6844	0.6597	0.6973	0.6672	0.7059	0.6901	0.7017	0.6799	0.7036	0.6851
CM-SKETCH	0.7127	0.7012	0.7058	0.6930	0.6881	0.6719	0.7186	0.6915	0.7408	0.7277	0.7360	0.7194
NETWALK	0.7739	0.7641	0.7706	0.7581	0.7228	0.7122	0.7742	0.7523	0.7956	0.7885	0.7904	0.7823
ADDGRAPH	0.8005	0.8093	0.8149	0.8087	0.7796	0.7735	0.8024	0.7995	0.8133	0.8090	0.8205	0.8217
Multiplex Methods												
MNE	0.7994	0.7955	0.8050	0.7913	0.7108	0.7017	0.7532	0.7499	0.7541	0.7495	0.7813	0.7754
ML-GCN	0.7921	0.7886	0.7915	0.7907	0.7344	0.7263	0.7519	0.7439	0.7940	0.7918	0.8115	0.8072
ANOMULY	0.8783	0.8729	0.8694	0.8610	**0.8289**	**0.8195**	0.8825	0.8754	**0.8906**	**0.8852**	0.8938	0.8871
ADMIRE	**0.9016**	**0.9125**	**0.9093**	**0.8952**	0.8097	0.8041	**0.8873**	**0.8824**	0.8416	0.8392	**0.9160**	**0.8899**
Improvement	2.65%	4.53%	4.59%	3.97%	-2.31%	-1.88%	0.54%	0.79%	-5.50%	-5.19%	2.48%	0.32%

Table 5.2: Performance comparison of ADMIRE and baselines in single-layer networks (AUC).

Methods	Bitcoin		Amazon-S		DBLP-S	
Anomaly %	1%	5 %	1%	5 %	1%	5 %
GOUTLIER	0.7143	0.7091	0.6923	0.6614	0.7108	0.6995
CM-SKETCH	0.7146	0.7015	0.7049	0.6621	0.7084	0.6877
NETWALK	0.8375	0.8367	0.7483	0.7302	0.7779	0.7590
ADDGRAPH	0.8534	0.8416	0.7872	0.7828	0.7911	0.7932
ANOMULY	0.8707	0.8661	**0.8014**	0.7943	0.8129	0.8236
ADMIRE	**0.8996**	**0.8902**	0.7935	**0.8007**	**0.8521**	**0.8509**
Improvement	3.32%	2.78%	0.99%	0.81%	4.82%	3.31%

The experimental setup, datasets, baselines, and anomaly injection approach are all the same as what we used for ANOMULY as described in Section 4.3.

5.4.2 Results

We compare ADMIRE with the baseline methods as well as ANOMULY on both dynamic and static multiplex networks with different percentages of anomalous

edges (i.e., 1%, 5%). Table 5.1 reports the AUC for ADMIRE and both the baselines and ANOMULY. ADMIRE outperforms *all* baselines in all datasets and outperforms ANOMULY in four out of six multiplex network datasets. It improves the ANOMULY results by 2.48%, on average, in the four datasets. There are three reasons for ADMIRE's superior performance: **(1)** ADMIRE is an inductive method and does not rely on node identities, which makes it a powerful tool for the detection of complex patterns. **(2)** ADMIRE uses a time encoder module that can capture the temporal patterns in continuous timestamps, instead of approximating time by discrete snapshots. **(3)** ADMIRE uses a novel negative sample generator that allows it to learn more complex anomalous patterns.

ANOMULY outperforms ADMIRE in two datasets: Amazon and Ethereum. These datasets have discrete timestamps and are naturally snapshot-based, so ANOMULY correctly captures their temporal properties; ADMIRE's continuous time modeling is not useful. Also, these two datasets are particularly sparse, and random-walk-based methods might not be able to capture neighborhoods properly. This sparsity might cause the inter-view and intra-view walks in ADMIRE to treat far nodes as neighbors.

Table 5.2 reports the AUC for both the baselines and ANOMULY on single-layer networks. Results show the same pattern as in Table 5.1. The only exception is Amazon-S data when we have 5% anomalous connections. ADMIRE's ability to detect more complex anomalous patterns coupled with the relatively large percentage of anomalous connections makes this task more difficult for ANOMULY than for ADMIRE, leading to ADMIRE outperforming ANOMULY.

5.4.3 Ablation Study

We next conduct ablation studies on the Amazon, DBLP, and Ethereum datasets to validate the effectiveness of ADMIRE's critical components. Table 5.3 shows AUC results for the anomaly detection task in multiplex networks with an anomaly rate of 1%. The first row reports the performance of the complete ADMIRE implementation. Each subsequent row shows results for ADMIRE with one module modification: the second row removes inter-view walks, the third row removes intra-view walks, and the fourth row removes learnable parameter λ and uses the summation of inter-view

Table 5.3: Ablation study on ADMIRE (AUC). In each row, we remove/replace one of the ADMIRE's components and keep the others unchanged.

	Methods	Amazon	DBLP	Ethereum
1	ADMIRE	**0.8097**	**0.8873**	**0.8416**
2	w/o inter-view	0.7901	0.8699	0.8254
3	w/o intra-view	0.7619	0.8307	0.8352
4	w/o λ ($\lambda = 1$)	0.7946	0.8651	0.8390
5	w/o attention	0.7953	0.8612	0.8208
6	w/o time encoding	0.7615	0.8644	0.8025
7	w/o inter-view NS	0.7998	0.8819	0.8211
8	w/o intra-view NS	0.8032	0.8794	0.8387
9	w/o historical NS	0.8055	0.8537	0.8376
10	w/ RNN	0.7812	0.7929	0.8341

Table 5.4: Accuracy (%) of generated tree explanation on Amazon, DBLP and Ethereum. This table shows how well ADMIRE++ mimics ADMIRE predictions. We embolden the best results for each depth and use an underline to show the best performance for each dataset.

Depth	Amazon				DBLP				Ethereum			
	$k=3$	$k=4$	$k=5$	$k=6$	$k=3$	$k=4$	$k=5$	$k=6$	$k=3$	$k=4$	$k=5$	$k=6$
4	67.04	74.19	**75.73**	75.54	65.33	67.15	69.78	**70.12**	65.23	67.91	68.52	**70.36**
5	71.95	79.47	81.28	**82.56**	70.49	71.62	73.91	**75.35**	66.07	70.24	74.50	**79.48**
6	72.62	82.80	84.36	**<u>86.71</u>**	71.15	75.86	76.02	**<u>78.11</u>**	69.19	75.09	82.39	**<u>84.05</u>**

and interview encodings. The fifth row removes the attention module, the sixth row moves the time encoding module, the seventh to ninth rows remove different types of negative samples in the training, and the last row replaces MLP-Mixer with an RNN. The results show that each component contributes to ADMIRE's performance. The greatest contribution comes from intra-view, MLP-Mixer, and time encoding module.

5.4.4 How Well Does ADMIRE++ Mimic ADMIRE?

We evaluate how well ADMIRE++ mimics ADMIRE by reporting the accuracy of ADMIRE++ on the Amazon, DBLP, and Ethereum datasets in Table 5.4. As expected, considering more features (larger k) results in better performance, because increasing the k allows ADMIRE to more accurately distinguish different network motifs. However, even with a small number of features, $k = 4$, we can achieve good performance (Acc $\geq 75\%$) on mimicking ADMIRE's prediction.

5.4.5 What Does the Generated Tree Look Like?

We now examine how to use ADMIRE++ to interpret predictions. Figure 5.2 shows the generated tree by ADMIRE++ and the motif clusters with an extracted motif in each cluster when $k = 3$ and depth$= 4$. One can interpret the decision tree prediction as: a link is abnormal if one of its endpoints has some connections in only one view. However, if both of its endpoints have the same status (both have neighbors in either only one view or different views), the connection is normal. That is, given the fact that the Amazon dataset has only two views, clusters 1 and 3 represent the neighborhood of a node where most of its connections are in one view. Cluster 2 represents the neighborhood of a node where its connections with neighbors have diverse types. Therefore, for a normal connection, we expect to see the same distribution of clusters 1 and 3 in sampled walks started from the connection endpoints.

What Do the Clusters Look Like? As discussed in Table 5.4, the number of clusters (i.e., k) can directly affect the performance of ADMIRE++. However, it is not the only factor as the distribution of the walk encodings in the embedding space is also important. That is, if walk encodings in the embedding space are hard to distinguish, the decision tree might not be able to distinguish them either. To this end, we use t-SNE and visualize the walk encodings in a 3D space. The results are shown in Figure 5.3. While clusters of walks in the Amazon dataset are easily distinguishable, clusters in DBLP and Ethereum datasets are harder to distinguish. Accordingly, it is harder for the decision tree to mimic ADMIRE on the DBLP and Ethereum than on the Amazon dataset, which explains ADMIRE++'s performance in Table 5.4.

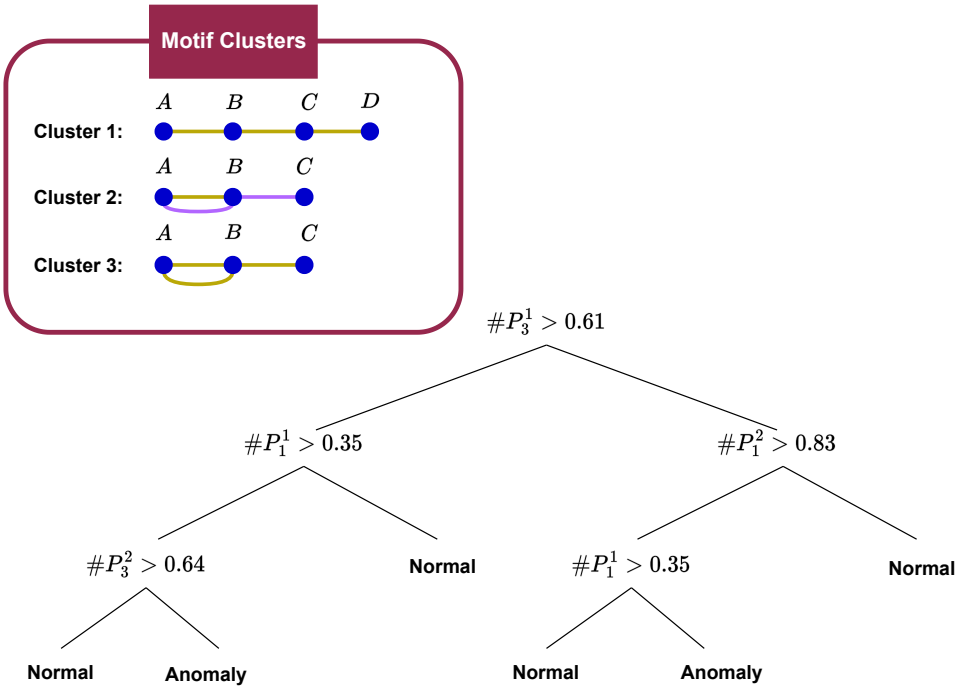
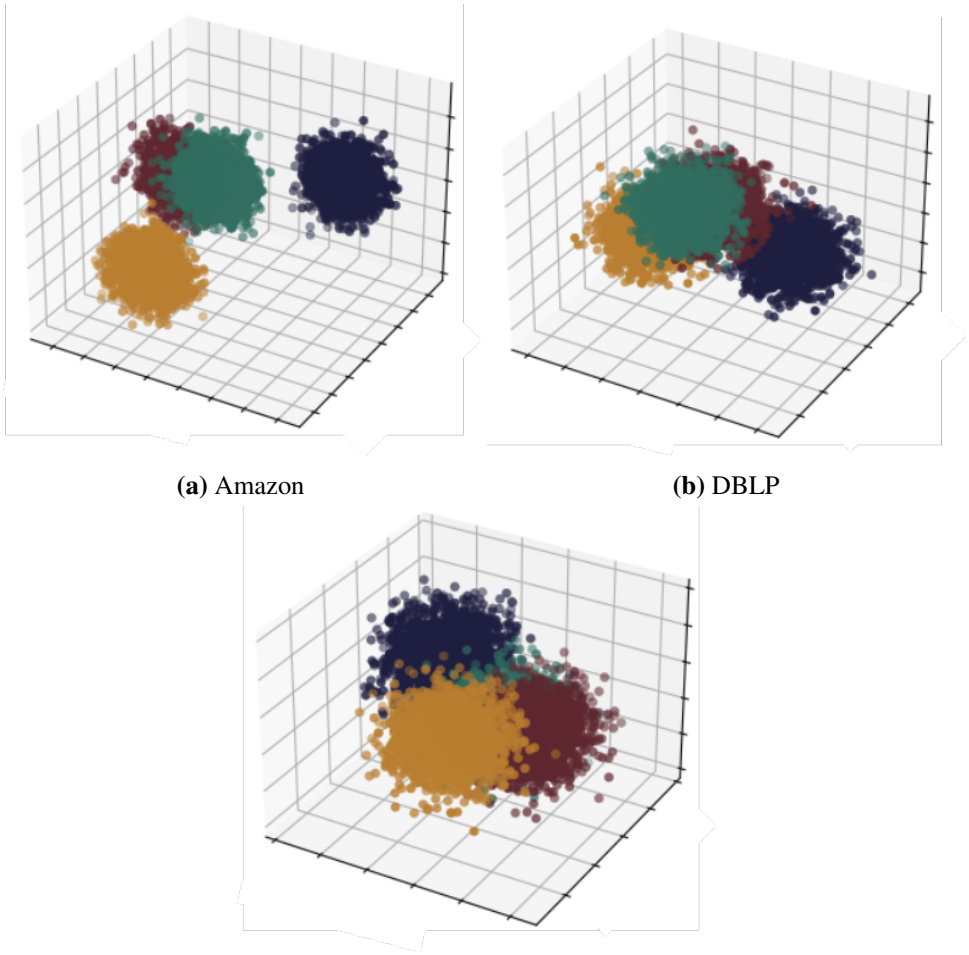


Figure 5.2: (**Top**) Motif clusters and example of an extracted motif in each cluster. (**Bottom**) The ADMIRE++ tree explanation on the Amazon dataset (depth=4). Different colors in the motif examples show the changes in views. Here, ${P_j}^i$ are the features constructed in Equation 5.20.

We further discuss ADMIRE++ for analyzing the human brain in Chapter 6.

5.5 Conclusion

We present ADMIRE, an end-to-end inductive unsupervised learning method on multiplex networks to detect anomalous connections. ADMIRE uses inter-view (resp. intra-view) temporal walks to implicitly extract network motifs and causal relationships across different views (resp. within a view) and adopts novel anonymization based on the correlation between multiplex network motifs to hide the identity of nodes and views. Next, it uses an MLP-Mixer to encode the sequence of nodes and views in a walk. By aggregating all the walks starting from a node,



(a) Amazon

(b) DBLP

(c) Ethereum

Figure 5.3: 3D visualization of clusters using t-SNE ($k = 4$). While clusters of walks in the Amazon dataset are simply distinguishable, clusters in DBLP and Ethereum datasets are hard to distinguish, which is the reason for the worse performance of ADMIRE++ (see Table 5.4) on these two datasets.

ADMIRE encodes its neighborhood. Finally, it uses a classifier that labels connections as normal/abnormal based on their endpoints' neighborhood. To explain ADMIRE's prediction in sensitive domains, we present ADMIRE++ that leverages the extracted inter-view and intra-view walks in ADMIRE to explain why a connection is predicted as abnormal. It uses a weighted optimal sparse decision tree model, which is a provably powerful interpretable model, to mimic ADMIRE's predictions. Our experimental results show the superior performance of ADMIRE against ANOMULY and baselines.

Chapter 6

Anomaly Detection in the Human Brain

We now turn to a particularly interesting application of ANOMULY and ADMIRE at the intersection of health and neuroscience. We discuss three different approaches to modeling multimodal neuroimaging data as multiplex networks and show how ANOMULY and ADMIRE can address the limitations of existing methods in anomaly detection in human brain networks. Our experiments on Parkinson's Disease (PD), Attention Deficit Hyperactivity Disorder (ADHD), and Autism Spectrum Disorder (ASD) show the efficiency and effectiveness of our approach in detecting anomalous brain activity in people living with these diseases or disorders.

6.1 Introduction

Recently, the fields of neuroscience and brain imaging research have undergone a significant shift in focus from region-specific analyses to network models [13, 123], largely due to the rapid development of modern neuroimaging technology. Network models of the brain represent regions of interest (ROIs) as nodes and calculate pairwise similarities between regions to form edges [69], usually derived from functional Magnetic Resonance Imaging (fMRI) or structural Magnetic Resonance Imaging (sMRI). These models have been helpful in enhancing our understanding of brain diseases and disorders [42, 140]. As a result, empirical data on brain networks

has substantially increased in size and complexity, leading to a strong demand for appropriate tools and methods to model and analyze it [140].

At the same time, there has been significant interest in machine learning methods for analyzing graph-structured data in various domains, such as drug discovery [167], neuroscience [1], and biology [73]. While several studies demonstrated the efficacy of machine learning on graphs for analyzing human brain networks, most focus on graph or node classification tasks [54, 94]. These tasks involve detecting diseases [190], predicting biological features [94], and identifying functional systems [16]. However, detecting abnormal brain activity in people with neurological disorders is a crucial step in understanding the causal mechanisms of symptoms, facilitating early detection and the development of medical treatments. Most existing studies consider a single brain network (from a single type of neuroimage or a single subject), which can be noisy or incomplete [6, 185]. To address this limitation, De Domenico [59] suggests using static multiplex networks. Multiplex networks are graphs where nodes can be connected by different types of edges [98]. Edge types can be the brain networks of different subjects or different neuroimaging modalities.

Limitation of Previous Methods. Although anomaly detection in graphs is a well-studied problem, brain networks have five unique traits that make directly applying existing graph anomaly detection models impractical: ① Noisy data: a single neuroimaging data sample can be extremely noisy and inaccurate [6], which hinders the identification of biological insights into the structure of brain networks. Existing general anomaly detection methods can use only a single brain image, making them sensitive to noise, or one must aggregate different neuroimages as a pre-processing step, missing complex brain activity in each brain image. ② Multimodal neuroimaging: while several studies discussed the importance of using different neuroimage types (e.g., fMRI, sMRI, etc.), because different modalities provide complementary information [186, 191], existing works are limited to a single type of neuroimage and are unable to incorporate information about different modalities. ③ Complex activity: brain activities are complex and potentially different in different subjects, while existing methods are designed in the transductive setting, which limits their generalizability to unseen nodes or patterns.

④ Time alignment: existing methods assume that the timestamps in different graphs are meaningfully related. However, while modeling neuroimage data as *temporal* brain networks, the timestamps might be shifted and are unlikely to be aligned across brain images of different subjects. ⑤ Explainability: decision making on health-related data, which is sensitive, requires explainable models, but existing methods are uninterpretable black boxes.

There are two other limitations that plague existing studies: ⓘ These studies assume pre-defined anomaly patterns or man-made features. Such approaches do not easily generalize to the brain activity of different individuals. Moreover, in a real-world scenario, brain activity might be more complex in nature, and it is nearly impossible to detect anomalies with high accuracy using pre-defined patterns/roles. ⓘⓘ These methods are designed for static brain networks, missing the dynamics of brain activity over time.

6.2 Related Work on Machine Learning for Brain Networks

Feature Learning in Brain Networks. In recent years, several studies focused on analyzing brain networks to understand and distinguish healthy and diseased human brains [44, 86, 165]. Recently, due to the success of GNNs in analyzing graph-structured data, deep models have been proposed to predict brain diseases by learning the graph structures of brain networks [52, 54, 91, 92, 190]. All these methods are designed for either graph or node classification and cannot easily be extended to edge-anomaly detection.

Anomaly Detection in Brain Networks. Several recent studies focus on analyzing brain networks to distinguish healthy and diseased human brains [44, 86, 165]. Due to the success of GNNs in analyzing graph-structured data, deep models have been proposed to predict brain diseases by learning the brain network structure [52, 54, 91, 92, 190]. Several anomaly detection methods have been proposed to find anomalous regions or subgraphs in the brain, which might indicate the presence of a disease [42, 109, 184]. All these methods are designed for node or subgraph anomaly detection tasks in *single* brain networks and cannot easily be extended to detect anomalous edges in *multiplex* brain networks. Also, these methods do not

learn anomalous patterns; they find only pre-defined patterns/rules for anomalies.

6.3 Why/How to Model Human Brain as a Multiplex Network

Given a neuroimaging dataset, we answer the following three questions:

Q1: How does multiplex modeling improve upon modeling a single network? While modeling the human brain as a network has gained much attention in the neuroscience community in recent years [108, 114], most studies have focused on a single type of simple brain networks [42, 109]. However, recent research on brain network analysis suggests that different modalities of brain networks provide complementary information [186, 191]. The fusion of multiple modalities can lead to consistent improvements in brain analysis. Additionally, several studies suggest that brain networks generated from an individual can be noisy and incomplete [6, 101, 185]. Consequently, researchers are exploring the possibility of studying the human brain without necessarily discarding or aggregating the vast amount of data available [59]. Multiplex brain networks, where each view represents a type of neuroimaging data or the neuroimage of an individual, are accurate models that can capture the complementary information provided by different neuroimaging data and increase the model's robustness to noise and incompleteness in individual neuroimages.

Q2: How can neuroimaging data be modeled as (temporal) multiplex networks? We focus on three ways to model neuroimaging data as multiplex networks.

① **Activity in different frequency bands**: in the context of fMRI images, previous work utilizes filtering procedures to extract signals within a particular frequency range, typically between 0.01 and 0.1 Hz [59, 61]. However, the selection of the frequency band carries significant implications for the functional representation of the brain. In fact, existing methods do not distinguish the contributions coming from different frequency bands, overlooking the contributions of different ranges, and instead, concentrate on a single frequency range. To this end, De Domenico et al. [60] shows that brain signals in a range between 0.01 and 0.25 Hz, in steps of 0.02 Hz, provide unique information and should be neither aggregated nor neglected. We suggest using multiplex brain networks, where each view represents the graph of signals in a specific range.

② **Multimodal brain networks**: Recently, several studies discussed the importance of using different neuroimage types (e.g., fMRI, sMRI, Diffusion Tensor Imaging (DTI), etc.) in brain network analysis, as different modalities of brain networks provide complementary information [186, 191]. A multiplex brain network can represent a multimodal brain network, where each view models the brain network produced from a specific type of neuroimage (e.g., fMRI or sMRI).
③ **Different subjects**: Previous studies discuss the challenges of the existence of noise in a brain network generated from an individual [101, 185]. Most existing methods aggregate (e.g., averaging) the data from different individuals to mitigate the noise in the dataset [1, 42]. However, this aggregation discards complex patterns in each individual's brain activity, causing missing information. Moreover, in brain network disease/disorder analysis, it is known that individuals having the same disease or disorder share similar patterns [55, 93], which means that disorder/disease-specific anomalous activity requires consideration of the brain networks of different subjects. In a multiplex brain network, each view can represent the brain network of an individual.

6.4 Modified Attention Mechanism For Multiplex Brain Networks

As we discussed in Chapter 1 and Section 4.2.3, different views might exhibit different importance. For example, one disease might be more correlated with functional connectivity than structural connectivity, or a brain network of an individual can be noisy, and we need to automatically ignore it in the training process. We addressed this in Section 4.2.3 by using an attention mechanism that learns the importance of each view for each node. While this attention mechanism is designed for *general* multiplex networks, assuming that the importance of each view for each node is different, it is not the best approach to balance the trade-off between scalability and performance. That is, although this mechanism is more general, it requires learning many parameters, limiting its scalability to large networks with a large number of views. In our experimental evaluations on multiplex *brain* networks, we observe that the importance of one view for different nodes is almost the same. Therefore, we design a more efficient and scalable attention mechanism for multiplex brain

networks that learns the importance of each view for other views (independent of nodes), because in a multiplex brain network we might have a large number of views (e.g., a large number of subjects, a large number of image modalities, or a large number of frequency bands). One can interpret this attention mechanism as a model that learns the correlation between each pair of views.

Given two arbitrary views r_1, r_2, let $\eta(r_1)$ and $\eta(r_2)$ be the learned encoding of r_1 and r_2 (see Section 5.2.2). Inspired by Graph Attention Networks [154], we define the importance of r_2 for r_1, $\psi(r_1, r_2)$, as:

$$\psi(r_1, r_2) = \frac{\exp\left(\sigma\left(\vec{a}^T.[\mathbf{W}^{\text{att}}\,\eta(r_1)\ ||\ \mathbf{W}^{\text{att}}\,\eta(r_2)]\right)\right)}{\sum_{r' \in \mathcal{L}} \exp\left(\sigma\left(\vec{a}^T.[\mathbf{W}^{\text{att}}\,\eta(r_1)\ ||\ \mathbf{W}^{\text{att}}\,\eta(r')]\right)\right)},$$

where $\vec{a}$ and $\mathbf{W}^{\text{att}}$ are learnable parameters and $\sigma(.)$ is an activation function (e.g., ReLU). The main intuition of this attention mechanism is to (1) use learnable matrix $\mathbf{W}^{\text{att}}$ to project view encodings to the same embedding space, and (2) use $\vec{a}$ to learn the importance of each view in the concatenation. Note that the softmax function is used to normalize weights.

6.5 Experiments

6.5.1 Experimental Setting Details

We tune hyper-parameters by cross-validation, and search the hyper-parameters over ① $\mu \in \{0.5, 1, 2, 4\} \times 10^{-5}$, ② Inter-view sampling number $M \in \{32, 64, 128, 256\}$, ③ Intra-view sampling number per view $M' \in \{8, 16, 32, 64\}$, ④ Walk length $m \in \{2, 4, 8, 12\}$. Also, in the training, following the previous chapters, we use a learning rate of 0.0001, hidden dimension 100 in MLP-Mixer, and batch size of 600.

To visualize the average distribution of anomalous connections, we use BrainPainter [118] with the Desikan-Killiany atlas.

6.5.2 Datasets

We evaluate ADMIRE and ANOMULY on multiplex brain networks using three real-world datasets, PD [57], ADHD [29], and ASD [50]. Each of the datasets

represents one type of multiplex brain network modeling proposed in Section 6.3.

PD Dataset. Attention dysfunction is a common symptom of Parkinson's disease (PD) and has a significant impact on quality of life. This dataset [57] uses the Attention Network Test (ANT) [67] and is designed to study three aspects of attention: alerting (maintaining an alert state), executive control (resolving conflict), and orienting. It consists of both structural and functional MRI images of participants with and without PD, with six repetitions of the ANT task [67]. It contains data for 25 subjects (7 female, age = 66.1 ± 10.0 yrs, years since disease onset = 8.4 ± 4.8) in the PD group and 21 subjects (12 female, age = 62.1 ± 9.9 yrs) in the healthy control group. We model the data using a temporal multiplex brain network with two views, 114 ROIs, and six timestamps (fMRI during each task). The first view represents the brain network obtained from fMRI, while the second view represents that generated from T1-weighted structural MRI.

ADHD Dataset. This dataset consists of resting fMRI data taken from the USC Multimodal Connectivity Database [29]. The dataset contains data for 50 subjects (27 female, age = 9.84 ± 3.57 yrs) in the ADHD group and 50 subjects (25 female, age = 12.74 ± 4.1 yrs) in the typically developed (TD) control group. This dataset is preprocessed (https://ccraddock.github.io/cluster_roi/atlases.html). We model this data using a temporal multiplex brain network with 50 views, 190 ROIs, and 10 timestamps; each view represents the brain network of an individual.

ASD Dataset. This dataset consists of resting fMRI data taken from the Autism Brain Imaging Data Exchange (ABIDE) [50]; it contains data for 45 subjects (23 female, age $= 23.1 \pm 8.1$ yrs) in the ASD group and 45 subjects (22 female, age $= 25.4 \pm 8.9$ yrs) in the typically developed control group. We have followed the five pre-processing strategies denoted as DPARSF, followed by Band-Pass Filtering with different filters in a range between 0.01 and 0.25 Hz, in steps of 0.02 Hz. This range and steps are previously motivated by De Domenico et al. [60]. We model this data using a temporal multiplex brain network with 12 views, 116 ROIs, and 10 timestamps; the i-th view represents the brain network obtained by filtering the fMRI values in the range $[0.01 + (i-1) \times 0.02, 0.01 + i \times 0.02]$ Hz.

Synthetic Anomalies. In addition to real-world case studies on PD, ADHD, and

ASD, we use these datasets and inject anomalies (as we have done in Section 4.3) to investigate the parameter sensitivity of ADMIRE and also compare the performance of ADMIRE and ANOMULY in inductive and transductive setting on the brain network analysis domain.

Pre-pocessing. Unless stated otherwise, for preprocessing and constructing brain networks from original fMRI and DTI data, we use the FSL toolbox and BrainGB [53].

6.5.3 Baselines

We evalute both ANOMULY and ADMIRE in the transductive setting and then further evaluate ADMIRE in the inductive setting. Accordingly, in addition to the baselines we used in Sections 4.3 and 5.4, we use three new methods that are specifically designed for inductive settings, because none of the existing methods for anomaly detection are designed for inductive settings. These new baselines help us to investigate the performance of ADMIRE in the inductive setting. CAW-N [163] is an inductive method that uses causal anonymous walks to extract network motifs and a novel set-based anonymization process that keeps the model inductive by hiding the identity of nodes during the training phase. EvolveGCN [129] uses a RNN to estimate the GCN parameters for future snapshots. TGAT [56] uses GAT [154] to extract node representations where the nodes' neighbors are sampled from the history and then encode temporal information via random Fourier features. We exclude ANOMULY in the inductive setting as it is designed for the transductive setting and cannot generalize to unseen patterns or nodes.

6.5.4 Experimental Setup

In each real-world experiment, we use the neuroimages of people in the control group (either healthy or TD) to train the models. Once the models are trained, we test them by using the neuroimages of people in the condition group (living with PD, ADHD, or ASD). In the inductive setting, we follow previous work [88, 163] and randomly hide 10% of nodes in the training phase.

6.5.5 Results

Effectiveness Evaluation. We first compare the effectiveness of ADMIRE and ANOMULY with baselines on the task of detecting synthetic anomalous connections. Table 6.1 reports the AUC of all the methods on different datasets. ADMIRE outperforms ANOMULY and all baselines by a significant margin ($\min = 6.42\%$ and $\max = 18.04\%$ performance improvement over the ANOMULY performance) in the transductive setting. There are four reasons for ADMIRE's superior performance: ① ADMIRE outperforms monoplex methods as it is a multiplex method and is able to learn from different data sources, image modalities, or frequency bands by using inter-view walks and attention mechanism over intra-view walks that incorporates complementary information from different views. ② It outperforms multiplex methods as it can capture the complex underlying rules of brain activities through its two causal walks over time. Also, its anonymization process hides the identity of nodes and views and captures the correlation between walks, increasing its ability to generalize better to unseen patterns in the test data. ANOMULY uses GRU cells to update node embeddings in discretized time steps, limiting its ability to model time as a continuous variable, which misses some continuous temporal patterns in brain activity. That is, ADMIRE is a stream-based method and uses a time encoding module to capture time information, while ANOMULY is snapshot-based and aggregates edges into network snapshots, which remove some useful time information [163]. ③ ADMIRE is scalable with respect to the number of views and can be trained on many data sources, image modalities, or frequency bands. ANOMULY, as well as ML-GCN, use different GNN modules for each view, making them infeasible for large networks with a large number of views (e.g., ADHD dataset with 50 subjects).

Table 6.1 reports the performance of ADMIRE and inductive baselines in the inductive setting. We attribute ADMIRE's superior performance (with $\min = 11.08\%$ and $\max = 26.89\%$ improvement over the best baseline) to two main reasons: ① ADMIRE is an end-to-end anomaly detection method with an exclusive

Table 6.1: Performance comparison of ADMIRE and baselines in multiplex brain networks (AUC).

	Methods	PD		ADHD		ASD	
	Anomaly %	1%	5 %	1%	5 %	1%	5 %
Transductive	Monoplex Methods						
	GOUTLIER	61.42	59.98	65.37	64.70	60.85	59.13
	NETWALK	69.71	0.6902	70.29	69.86	69.07	68.52
	ADDGRAPH	71.94	70.33	71.89	70.11	71.30	70.96
	Multiplex Methods						
	MNE	70.39	70.54	73.78	72.31	70.19	69.94
	ML-GCN	68.50	68.33	-*	-*	69.56	69.35
	ANOMULY	78.07	79.85	-*	-*	77.14	77.08
	ADMIRE	**85.09**	**84.98**	**88.67**	**88.53**	**91.06**	**89.95**
Inductive	EvolveGCN	55.18	55.06	57.23	57.41	56.89	56.21
	TGAT	59.34	58.72	60.19	60.10	60.28	59.93
	CAW-N	75.85	75.90	71.64	71.02	71.31	71.96
	ADMIRE	**84.72**	**84.31**	**88.03**	**88.97**	**90.49**	**90.28**

* Training time exceeds the threshold.

Table 6.2: Ablation study of ADMIRE on brain network datasets. (AUC).

	Methods	PD	ADHD	ASD
1	ADMIRE	**85.09**	**88.67***	**91.06**
2	w/o inter-view	78.59	**88.73***	80.65
3	w/o intra-view	77.14	69.59	79.62
4	w/o λ ($\lambda = 1$)	80.42	80.36	89.30
5	w/o attention	84.79	86.14	86.57
6	w/o time encoding	84.16	82.78	85.92
7	w/o inter-view NS	84.77	84.28	83.46
8	w/o intra-view NS	79.91	78.75	81.09
9	w/o historical NS	84.68	84.16	84.31
10	w/ RNN	83.90	85.32	89.13
11	Monoplex-ADMIRE	76.52	72.07	74.15

* There is no causal relation between views.

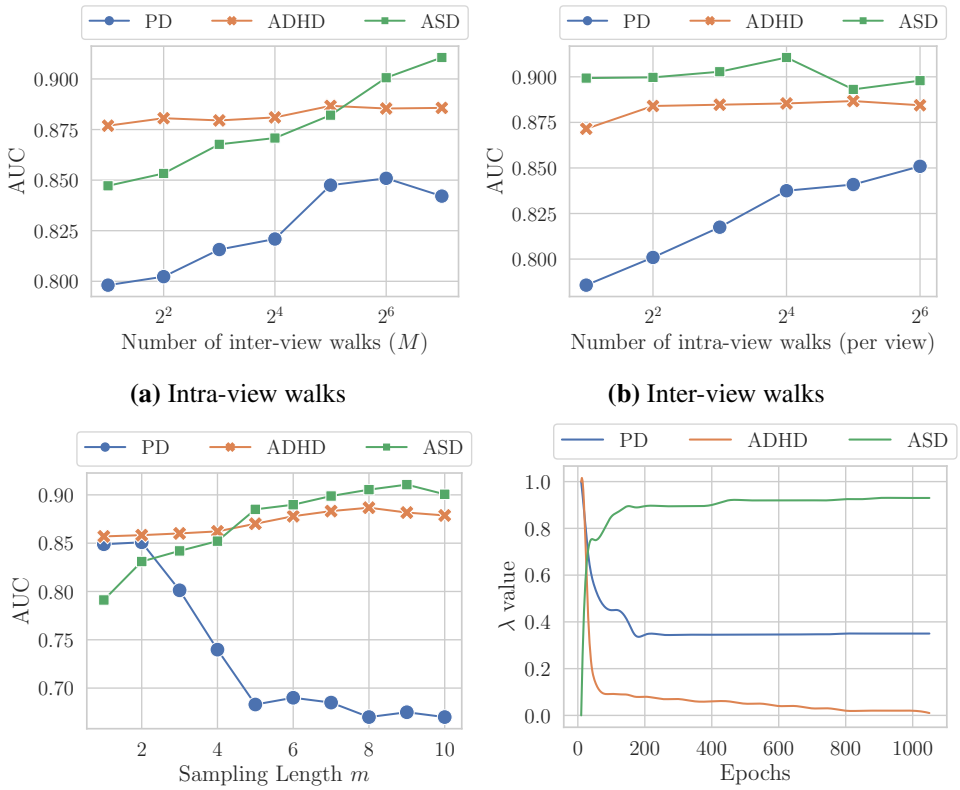
design of generating negative samples and training process, while baselines are designed to learn the temporal and structural properties of the network. ② ADMIRE is a multiplex method, while baselines are monoplex methods.

Ablation Studies. We further conduct ablation studies to validate the effectiveness of each component of ADMIRE. The results are summarized in Table 6.2. Rows 2 and 3 show the effectiveness of inter-view and intra-view walks. The only exception is removing the inter-view walks in the ADHD dataset. As we discussed in Section 5.2.2, when there is no causal relation between views, inter-view walks are not informative and our model is expected to learn to ignore these walks (set $\lambda = 0$). Accordingly, removing these walks from ADMIRE does not hurt the performance on the ADHD dataset, because there is no causal relation between views. Rows 4 and 5 show the importance of the learnable parameter λ and the attention mechanism that incorporates information from different views. Rows 7, 8, and 9 show the importance of our new negative sample generator. When using RNN instead of MLP-Mixer in the walk encoding phase (row 10), we gain better performance due to its ability to learn the dependency of nodes' encoding in a walk. Finally, the last row shows the superior performance of multiplex ADMIRE over monoplex ADMIRE, when using only one brain network generated from an individual, image modality, or frequency band. These results show the importance of multiplex modeling of neuroimages.

Why Modified Attention Mechanism? Given a view r and a node u, we use $\Omega(u, r)$ to show the importance of view r for node u. We use the attention mechanism proposed in Section 4.2.3 and train the model on PD, ADHD, and ASD datasets. We observe that $\Omega(u, r) \approx \Omega(v, r)$ for any given view r and arbitrary nodes u and v. That is, given a view r, the maximum variance of $\Omega(u, r)$ for different nodes u is 0.02, 0.05, and 0.02 in PD, ADHD, and ASD datasets, respectively.

Our experimental evaluation shows that this attention mechanism, on average, requires $\approx 45\%$ less training time on PD, ADHD, and ASD datasets.

Parameter Sensitivity. We systematically analyze the effect of hyperparameters used in ADMIRE on the performance. Figure 6.1(a) shows that only a small number of intra-view walks are enough to achieve competitive performance. While more inter-walks improve the performance until some point, the performance gain is



(a) Intra-view walks

(b) Inter-view walks

(c) Walk Length

(d) Evolution of λ

Figure 6.1: The effect of hyperparameters on the performance (a-c), and λ evolution (d).

saturated after that. A similar pattern can be seen for increasing the number of intra-view walks (Figure 6.1(b)). Note that this figure reports the number of intra-view walks per view. Accordingly, we expect to see more performance gain on datasets with a fewer views (e.g., PD). Figure 6.1(c) shows that ADMIRE might achieve the best performance at a certain walk length, while the exact value depends on the complexity of higher-order motifs that are required to learn the underlying network dynamics as well as the number of views. Networks with a large number of views might need longer walks to learn the causal relation in different views, because it takes more steps to visit all the interrelated views. Finally, Figure 6.1(d) shows the evolution of λ in training. As expected, in datasets with no causal relationship

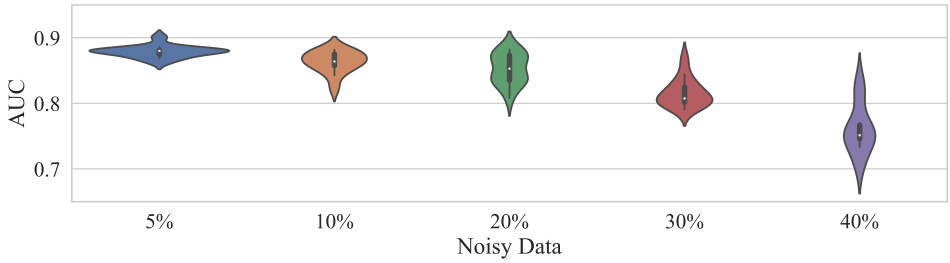
between different views (e.g., ADHD), ADMIRE learns to set $\lambda \leq 0.1$ after only a few epochs. For other datasets, it shows that ADMIRE converges quickly to the best value of λ.

Is Multiplex Modeling Robust to Noise? As discussed in Section 6.3, one of the main motivations for modeling neuroimaging datasets as multiplex networks is to make the model more robust to noise in each brain image. To validate the efficacy of this approach, we add Gaussian noise to a subset of brain images ($5\%, 10\%, 20\%, 30\%$ and 40%) in the ADHD dataset. We model the noisy dataset as a multiplex brain network and use it to train ADMIRE. Next, as a baseline, following previous methods [101, 185], we take the average of all brain images in the noisy dataset and use it to train the monoplex ADMIRE. Figure 6.2 reports the performance of ADMIRE and monoplex ADMIRE with a varying fraction of noisy samples. Not only does ADMIRE achieve superior performance by a significant margin, but it is more robust to noise than monoplex ADMIRE. This experiment shows the importance of multiplex modeling and also the efficacy of the proposed attention mechanism that learns to ignore noisy samples.

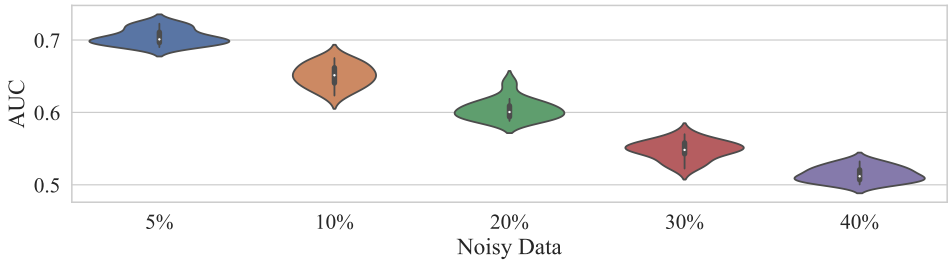
6.5.6 Results on Real-world Datasets

We next report findings from applying ADMIRE to real-world datasets. We train our model on the healthy control group and then test it on the condition group (living with PD, ADHD, ASD) to find anomalous brain activity of people in the condition group.

Parkinson's Disease. We focus on abnormal *brain structure* and *functional* activity of PD patients. Since the brain of each individual in each task might also have unique complex activity, we need to focus on common or more frequently appeared anomalous connections between ROIs over different subjects to capture abnormal activity that might be correlated to PD. To this end, we show how anomalous connections found by ADMIRE are distributed in the brain of people living with PD. Figure 6.3 reports the average distribution of anomalous edges in the brain networks of people living with PD. Most anomalous edges found by ADMIRE have a vertex in either *Posterior Cingulate*, *Superior Parietal*, *Medial Orbitofrontal*,



(a) Performance of ADMIRE



(b) Performance of monoplex ADMIRE

Figure 6.2: The advantage of multiplex brain networks over monoplex brain networks.

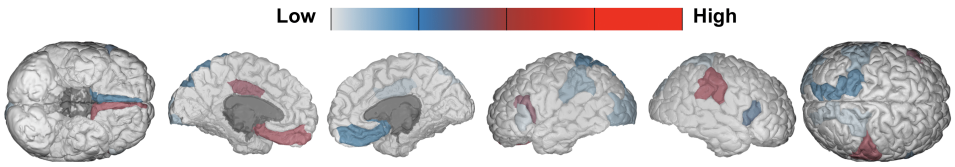


Figure 6.3: The distribution of anomalous edges in PD group.

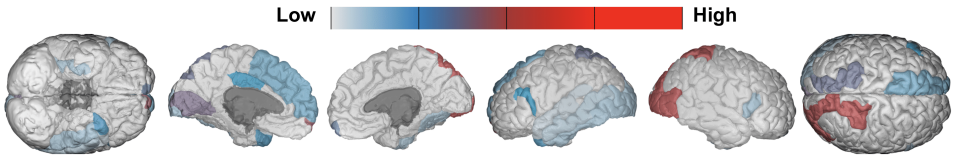


Figure 6.4: The distribution of anomalous edges in ADHD group.

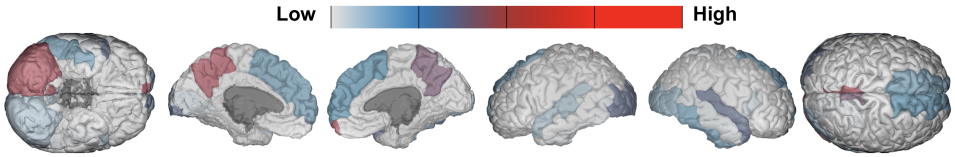


Figure 6.5: The distribution of anomalous edges in ASD group.

Pars Opercularis, or *Supramarginal Gyrus* ($\geq 95\%$ of all found anomalies). Most anomalous edges found by ANOMULY have a vertex in the same brain regions.

Next, we apply ADMIRE and ANOMULY on the healthy control group to see whether these findings are exclusive to the PD group and to identify possible noise in the dataset. We observe that ADMIRE finds 94.2% fewer anomalous connections in the healthy control group, most of which have a node in either *Temporal Pole* or *Anterior Insula*. ANOMULY finds 85.91% fewer anomalous connections in the healthy control group, most of which have a node in *Temporal Pole*.

Attention Deficit Hyperactivity Disorder. Following the previous experiment, we first focus on abnormal brain activity in subjects in the ADHD group. Figure 6.4 shows the average distribution of anomalous edges in the brain networks of subjects in the condition ADHD group. Most abnormal connections found by ADMIRE have an endpoint in either *Frontal Pole*, *Right Lateral Occipital Cortex*, *Lingual Gyrus*, *Left Temporal Pole*, or *Right Superior Parietal Lobule* ($\geq 95\%$ of all found anomalies). Interestingly, these findings are consistent with previous studies on ADHD, using voxel-wise estimation of regional tissue volume changes [158], abnormality in DTI images [103], and Forman–Ricci curvature changes [42], which shows the potential of ADMIRE in revealing abnormal connections that might be correlated to a brain disease or disorder. ANOMULY misses abnormal activities in Lingual Gyrus, Left Temporal Pole, and Right Superior Parietal Lobule.

Applying ADMIRE on the healthy control group, we observe that ADMIRE finds 89.6% fewer anomalous connections in the healthy control group, most of which have an endpoint in either *Planum Polare* or *Angular Gyrus*. ANOMULY finds 74.36% fewer anomalous connections in the healthy control group, most of which have a node in either *Frontal Pole* or *Angular Gyrus*.

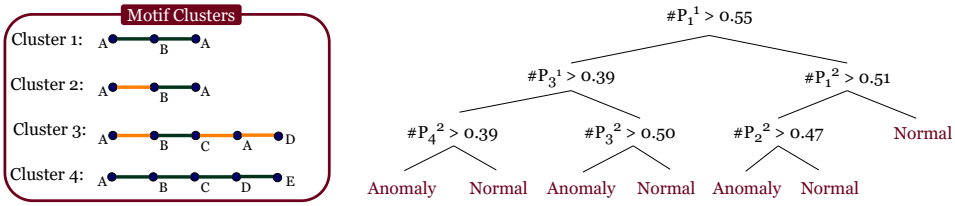


Figure 6.6: (**Left**) Motif clusters and example of an extracted motif in each cluster. (**Right**) The ADMIRE++ tree explanation on PD dataset (depth=3). Different colors in the motif examples show the changes in views. Here, ${P_j}^i$ are the features constructed in Equation 5.20.

Autism Spectrum Disorder. Figure 6.5 shows the average distribution of anomalous edges in the brain networks of subjects in the condition ASD group. Most abnormal connections found by ADMIRE have an endpoint in either *Right Superior Temporal Gyrus*, *Right Cerebellum Cortex*, *Right Precuneus*, *Frontal Pole*, *Left Lateral Occipital* ($\geq 95\%$ of all found anomalies). Although several studies about ASD found different abnormality patterns, there is still no known ASD biomarker [124]. However, part of our findings about abnormal activity in the cerebellum cortex is consistent with previous studies [143]. ANOMULY misses abnormal activities in Left Lateral Occipital and Right Cerebellum Cortex.

Applying ADMIRE on the healthy control group, ADMIRE finds 93.7% fewer anomalous connections in the healthy control group, most of which have an endpoint in either *Temporal Pole* or *Posterior Cingulate Cortex*. ANOMULY finds 79.14% fewer anomalous connections in the healthy control group, most of which have a node in *Temporal Pole*.

6.5.7 ADMIRE++ Explanations

To evaluate the quality of ADMIRE++, we use the PD dataset and set $k = 4$, so we have 4 clusters and 8 features for each edge (4 features for each of its endpoints). Figure 6.6 (left) and (middle) show an example of motifs in each cluster and a decision tree with depth 3 that mimics the ADMIRE's predictions. While motifs in cluster 1 and 2 shows that the neighborhood of a node is sparse, motifs in cluster 3 and 4 show that the neighborhood of the node is dense. One can interpret the decision tree prediction as: a link is normal if the neighborhoods of its endpoints

Table 6.3: Accuracy (%) of generated tree explanation. This table shows how well ADMIRE++ mimics ADMIRE predictions.

Depth	PD	ADHD	ASD
3	75.43	74.98	83.39
4	81.76	77.15	90.28
5	87.52	83.17	91.06

are both sparse or dense and is abnormal otherwise.

The Table 6.3 reports the accuracy of how well ADMIRE++ mimics ADMIRE's predictions. Even with small depths, ADMIRE++ produces explanations with high accuracy.

Chapter 7

Conclusions

We introduced two novel frameworks, ANOMULY and ADMIRE, for detecting edge anomalies in dynamic multiplex networks. ANOMULY utilizes a combination of GNN layers and GRU cells, incorporating temporal and structural properties, along with an attention mechanism that integrates information across different types of connections. It further uses negative sampling during training to learn the anomalous patterns in an unsupervised manner. While ANOMULY achieves good performance and outperforms baselines, it suffers from poor scalability and memory usage and an inability to generalize to unseen nodes.

To address these limitatons, we introduced ADMIRE, which leverages inter-view and intra-view temporal walks to capture network motifs and causal relationships within and across different views, respectively. ADMIRE uses an anonymization technique to hide node and view identities, keeping the model inductive. Next, it utilizes an MLP-Mixer to encode the sequence of nodes and views in a walk. By aggregating walks from a node, ADMIRE encodes its neighborhood and use a classifier to label connections as normal or abnormal based on their endpoints' neighborhoods. We then introduced ADMIRE++ to explain ADMIRE's predictions using weighted optimal sparse decision trees.

We demonstrate the efficacy of ANOMULY and ADMIRE in analyzing the abnormal brain activity that might cause a brain disease/disorder. We conduct case studies on brain networks of individuals with Attention Deficit Hyperactivity Disorder, Parkinson's Disease, and Autism Spectrum Disorder. The results highlight

the potential of these frameworks to identify abnormal brain activity associated with various diseases and disorders. Overall, these frameworks provide valuable tools for analyzing dynamic multiplex networks and detecting anomalies with potential applications in diverse domains.

Bibliography

[1] C. Abrate and F. Bonchi. Counterfactual graphs for explainable classification of brain networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery; Data Mining*, KDD '21, page 2495–2504, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi:10.1145/3447548.3467154. URL https://doi.org/10.1145/3447548.3467154. → pages 3, 55, 58

[2] C. C. Aggarwal. Outlier analysis. In *EDBT*. Springer Cham, 2019. ISBN 978-3-319-83772-7. doi:https://doi.org/10.1007/978-3-319-47578-3. → page 1

[3] C. C. Aggarwal, Y. Zhao, and P. S. Yu. Outlier detection in graph streams. In *2011 IEEE 27th International Conference on Data Engineering*, pages 399–409, 2011. doi:10.1109/ICDE.2011.5767885. → pages 14, 27

[4] S. Aghaei, M. J. Azizi, and P. Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1418–1426, Jul. 2019. doi:10.1609/aaai.v33i01.33011418. URL https://ojs.aaai.org/index.php/AAAI/article/view/3943. → page 12

[5] S. Aghaei, A. Gómez, and P. Vayanos. Strong optimal classification trees. *arXiv preprint arXiv:2103.15965*, 2021. → page 12

[6] U. Agrawal, E. N. Brown, and L. D. Lewis. Model-based physiological noise removal in fast fmri. *NeuroImage*, 205:116231, 2020. ISSN 1053-8119. doi:https://doi.org/10.1016/j.neuroimage.2019.116231. URL https://www.sciencedirect.com/science/article/pii/S1053811919308225. → pages 3, 55, 57

[7] M. Ahmed, A. N. Mahmood, and M. R. Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, 2016. → page 1

[8] L. Akoglu and C. Faloutsos. Anomaly, event, and fraud detection in large network datasets. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, page 773–774, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450318693. doi:10.1145/2433396.2433496. URL https://doi.org/10.1145/2433396.2433496. → page 1

[9] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3): 626–688, May 2015. ISSN 1573-756X. doi:10.1007/s10618-014-0365-y. URL https://doi.org/10.1007/s10618-014-0365-y. → pages 1, 3, 19, 27

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. → page 43

[11] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473. → page 22

[12] M. Bansal and D. Sharma. Ranking and discovering anomalous neighborhoods in attributed multiplex networks. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, CoDS COMAD 2020, page 46–54, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450377386. doi:10.1145/3371158.3371164. URL https://doi.org/10.1145/3371158.3371164. → page 17

[13] D. S. Bassett and O. Sporns. Network neuroscience. *Nature Neuroscience*, 20(3):353–364, Mar 2017. ISSN 1546-1726. doi:10.1038/nn.4502. URL https://doi.org/10.1038/nn.4502. → page 54

[14] R. E. Beaty, M. Benedek, P. J. Silvia, and D. L. Schacter. Creative cognition and brain network dynamics. *Trends in cognitive sciences*, 20(2):87–95, 2016. → page 13

[15] R. E. Beaty, M. Benedek, P. J. Silvia, and D. L. Schacter. Creative cognition and brain network dynamics. *Trends in cognitive sciences*, 20(2):87–95, 2016. → page 13

[16] A. Behrouz and F. Hashemi. Cs-mlgcn: Multiplex graph convolutional networks for community search in multiplex networks. In *Proceedings of the 31st ACM International Conference on Information and Knowledge*

Management, CIKM '22, page 3828–3832, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi:10.1145/3511808.3557572. URL https://doi.org/10.1145/3511808.3557572. → pages 15, 16, 28, 55

[17] A. Behrouz and M. Seltzer. Anomaly detection in multiplex dynamic networks: from blockchain security to brain disease prediction. In *NeurIPS 2022 Temporal Graph Learning Workshop*, 2022. URL https://openreview.net/forum?id=UDGZDfwmay. → pages vi, 37, 43, 44

[18] A. Behrouz and M. Seltzer. Anomaly detection in human brain via inductive learning on temporal multiplex networks. In *Machine Learning for Healthcare Conference*. PMLR, 2023. → page vi

[19] A. Behrouz and M. Seltzer. ADMIRE++: Explainable anomaly detection in the human brain via inductive learning on temporal multiplex networks. In *ICML 3rd Workshop on Interpretable Machine Learning in Healthcare (IMLH)*, 2023. URL https://openreview.net/forum?id=t4H8acYudJ. → page vi

[20] A. Behrouz, F. Hashemi, and L. V. S. Lakshmanan. Firmtruss community search in multilayer networks. *Proc. VLDB Endow.*, 16(3):505–518, nov 2022. ISSN 2150-8097. doi:10.14778/3570690.3570700. URL https://doi.org/10.14778/3570690.3570700. → pages 3, 13, 25, 26, 27

[21] A. Behrouz, M. Lécuyer, C. Rudin, and M. Seltzer. Fast optimization of weighted sparse decision trees for use in optimal treatment regimes and optimal policy design. *CEUR Workshop Proc*, 3318, Oct. 2022. → pages 12, 46

[22] A. Behrouz, F. Hashemi, S. Sadeghian, and M. Seltzer. Cat-walk: Inductive hypergraph learning via set walks, 2023. → pages 11, 15

[23] D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017. → page 11

[24] S. Bhatia, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos. Midas: Microcluster-based detector of anomalies in edge streams. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3242–3249, Apr. 2020. doi:10.1609/aaai.v34i04.5724. URL https://ojs.aaai.org/index.php/AAAI/article/view/5724. → pages 1, 2, 14

[25] P. Bindu, P. S. Thilagam, and D. Ahuja. Discovering suspicious behavior in multilayer social networks. *Computers in Human Behavior*, 73:568–582, 2017. ISSN 0747-5632. doi:https://doi.org/10.1016/j.chb.2017.04.001. URL https://www.sciencedirect.com/science/article/pii/S0747563217302303. → page 16

[26] B. B. Biswal, M. Mennes, X.-N. Zuo, S. Gohel, C. Kelly, S. M. Smith, C. F. Beckmann, J. S. Adelstein, R. L. Buckner, S. Colcombe, et al. Toward discovery science of human brain function. *Proceedings of the National Academy of Sciences*, 107(10):4734–4739, 2010. → page 3

[27] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, mar 2003. ISSN 1532-4435. → page 26

[28] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013. → pages 23, 44

[29] J. A. Brown, J. D. Rudie, A. Bandrowski, J. D. Van Horn, and S. Y. Bookheimer. The UCLA multimodal connectivity database: a web-based platform for brain connectivity matrix sharing and analysis. *Front Neuroinform*, 6:28, Nov. 2012. → pages 59, 60

[30] J. A. Brown, J. D. Rudie, A. Bandrowski, J. D. Van Horn, and S. Y. Bookheimer. The ucla multimodal connectivity database: a web-based platform for brain connectivity matrix sharing and analysis. *Frontiers in neuroinformatics*, 6:28, 2012. → page 31

[31] J. Cadena, F. Chen, and A. Vullikanti. Graph anomaly detection based on steiner connectivity and density. *Proceedings of the IEEE*, 106(5):829–845, 2018. → page 14

[32] J. Cadena, F. Chen, and A. Vullikanti. Graph anomaly detection based on steiner connectivity and density. *Proceedings of the IEEE*, 106(5):829–845, 2018. doi:10.1109/JPROC.2018.2813311. → page 1

[33] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*, pages 3747–3756, 2021. → page 14

[34] A. Cardillo, J. Gómez-Gardenes, M. Zanin, M. Romance, D. Papo, F. d. Pozo, and S. Boccaletti. Emergence of network features from multiplexity. *Scientific reports*, 3(1):1344, 2013. → page 36

[35] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1358–1368, 2019. → pages 15, 16

[36] S. Chakraborty, S. Aich, S. J. Seong, and H.-C. Kim. A blockchain based credit analysis framework for efficient financial systems. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pages 56–60. IEEE, 2019. → page 3

[37] B. P. Chamberlain, S. Shirobokov, E. Rossi, F. Frasca, T. Markovich, N. Y. Hammerla, M. M. Bronstein, and M. Hansmire. Graph neural networks for link prediction with subgraph sketching. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=m1oqEOAozQU. → page 15

[38] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 119–128, 2015. → page 16

[39] T.-H. Chang and D. Svetinovic. Improving bitcoin ownership identification using transaction patterns analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):9–20, 2018. → page 3

[40] Y.-Y. Chang, P. Li, R. Sosic, M. H. Afifi, M. Schweighauser, and J. Leskovec. F-fade: Frequency factorization for anomaly detection in edge streams. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 589–597, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. doi:10.1145/3437963.3441806. URL https://doi.org/10.1145/3437963.3441806. → pages 1, 2

[41] S. Chanpuriya, R. A. Rossi, S. Kim, T. Yu, J. Hoffswell, N. Lipka, S. Guo, and C. N. Musco. Direct embedding of temporal network edges via time-decayed line graphs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Qamz7Q_Ta1k. → page 15

[42] T. Chatterjee, R. Albert, S. Thapliyal, N. Azarhooshang, and B. DasGupta. Detecting network anomalies using forman–ricci curvature and a case study for human brain networks. *Scientific Reports*, 11(1):8121, Apr 2021. ISSN

2045-2322. doi:10.1038/s41598-021-87587-z. URL https://doi.org/10.1038/s41598-021-87587-z. → pages 3, 13, 32, 54, 56, 57, 58, 68

[43] F. Chen and D. B. Neill. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1166–1175, 2014. → page 14

[44] G. Chen, B. D. Ward, C. Xie, W. Li, Z. Wu, J. L. Jones, M. Franczak, P. Antuono, and S.-J. Li. Classification of alzheimer disease, mild cognitive impairment, and normal cognitive status with large-scale network analysis based on resting-state functional mr imaging. *Radiology*, 259(1):213, 2011. → page 56

[45] J. Chen, X. Wang, and X. Xu. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv preprint arXiv:1812.04206*, 2018. → page 15

[46] L.-H. Chen, H. Li, and W. Yang. Anomman: Detect anomaly on multi-view attributed networks, 2022. URL https://arxiv.org/abs/2201.02822. → page 17

[47] J. Cheng, Q. Wang, Z. Tao, D. Xie, and Q. Gao. Multi-view attribute graph convolution networks for clustering. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 2973–2979, 2021. → pages 15, 16

[48] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. → pages 8, 19, 21

[49] W. Cong, S. Zhang, J. Kang, B. Yuan, H. Wu, X. Zhou, H. Tong, and M. Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ayPPc0SyLv1. → pages 11, 15, 40

[50] C. Craddock, Y. Benhajali, C. Chu, F. Chouinard, A. Evans, A. Jakab, B. S. Khundrakpam, J. D. Lewis, Q. Li, M. Milham, et al. The neuro bureau preprocessing initiative: open sharing of preprocessed neuroimaging data and derivatives. *Frontiers in Neuroinformatics*, 7:27, 2013. → pages 59, 60

[51] S. Cresci. A decade of social bot detection. *Communications of the ACM*, 63 (10):72–83, 2020. → page 1

[52] H. Cui, W. Dai, Y. Zhu, X. Li, L. He, and C. Yang. Brainnnexplainer: An interpretable graph neural network framework for brain network based disease analysis. *arXiv preprint arXiv:2107.05097*, 2021. → page 56

[53] H. Cui, W. Dai, Y. Zhu, X. Kan, A. A. Chen Gu, J. Lukemire, L. Zhan, L. He, Y. Guo, and C. Yang. BrainGB: A Benchmark for Brain Network Analysis with Graph Neural Networks. *IEEE Transactions on Medical Imaging (TMI)*, 2022. → page 61

[54] H. Cui, W. Dai, Y. Zhu, X. Li, L. He, and C. Yang. Interpretable graph neural networks for connectome-based brain disorder analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 375–385. Springer, 2022. → pages 55, 56

[55] H. Cui, W. Dai, Y. Zhu, X. Li, L. He, and C. Yang. Interpretable graph neural networks for connectome-based brain disorder analysis. In L. Wang, Q. Dou, P. T. Fletcher, S. Speidel, and S. Li, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022*, pages 375–385, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-16452-1. → page 58

[56] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJeW1yHYwH. → page 61

[57] T. K. M. Day, T. M. Madhyastha, M. K. Askren, P. Boord, T. J. Montine, and T. J. Grabowski. Attention network test fMRI data for participants with parkinson's disease and healthy elderly. *F1000Res*, 8:780, June 2019. → pages 13, 59, 60

[58] T. dblp team. dblp computer science bibliography. https://dblp.uni-trier.de, 2014. → page 26

[59] M. De Domenico. Multilayer modeling and analysis of human brain networks. *GigaScience*, 6(5), 02 2017. ISSN 2047-217X. doi:10.1093/gigascience/gix004. URL https://doi.org/10.1093/gigascience/gix004. gix004. → pages 3, 55, 57

[60] M. De Domenico, S. Sasai, and A. Arenas. Mapping multiplex hubs in human functional brain networks. *Front Neurosci*, 10:326, July 2016. → pages 57, 60

[61] F. De Vico Fallani, J. Richiardi, M. Chavez, and S. Achard. Graph analysis of functional brain networks: practical issues in translational neuroscience. *Philos Trans R Soc Lond B Biol Sci*, 369(1653), Oct. 2014. → page 57

[62] S. Dey. Securing majority-attack in blockchain using machine learning and algorithmic game theory: A proof of work. In *2018 10th computer science and electronic engineering (CEEC)*, pages 7–10. IEEE, 2018. → page 3

[63] K. Ding, J. Li, R. Bhanushali, and H. Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 594–602. SIAM, 2019. → page 1

[64] K. Ding, Q. Zhou, H. Tong, and H. Liu. Few-shot network anomaly detection via cross-network meta-learning. In *Proceedings of the Web Conference 2021*, WWW '21, page 2448–2456, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi:10.1145/3442381.3449922. URL https://doi.org/10.1145/3442381.3449922. → page 1

[65] D. Eswaran and C. Faloutsos. Sedanspot: Detecting anomalies in edge streams. In *2018 IEEE International conference on data mining (ICDM)*, pages 953–958. IEEE, 2018. → pages 1, 14

[66] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra. Spotlight: Detecting anomalies in streaming graphs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1378–1386, 2018. → pages 1, 14

[67] J. Fan, B. D. McCandliss, J. Fossella, J. I. Flombaum, and M. I. Posner. The activation of attentional networks. *Neuroimage*, 26(2):471–479, 2005. → page 60

[68] A. Farhangfar, R. Greiner, and M. Zinkevich. A fast way to produce near-optimal fixed-depth decision trees. In *Proceedings of the 10th International Symposium on Artificial Intelligence and Mathematics (ISAIM-2008)*, 2008. → page 11

[69] E. S. Finn, X. Shen, D. Scheinost, M. D. Rosenberg, J. Huang, M. M. Chun, X. Papademetris, and R. T. Constable. Functional connectome fingerprinting: identifying individuals using patterns of brain connectivity. *Nature Neuroscience*, 18(11):1664–1671, Nov 2015. ISSN 1546-1726. doi:10.1038/nn.4135. URL https://doi.org/10.1038/nn.4135. → page 54

[70] F. Fusco, D. Pascual, and P. Staar. pnlp-mixer: an efficient all-mlp architecture for language. *arXiv preprint arXiv:2202.04350*, 2022. → page 11

[71] E. Galimberti, F. Bonchi, F. Gullo, and T. Lanciano. Core decomposition in multilayer networks: Theory, algorithms, and applications. *ACM Trans. Knowl. Discov. Data*, 14(1), 2020. ISSN 1556-4681. doi:10.1145/3369872. → page 2

[72] J. Gao and B. Ribeiro. On the equivalence between temporal and static equivariant graph representations. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7052–7076. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/gao22e.html. → page 15

[73] Z. Gao, C. Jiang, J. Zhang, X. Jiang, L. Li, P. Zhao, H. Yang, Y. Huang, and J. Li. Hierarchical graph learning for protein–protein interaction. *Nature Communications*, 14(1):1093, 2023. → page 55

[74] K. R. Griffiths, T. A. Braund, M. R. Kohn, S. Clarke, L. M. Williams, and M. S. Korgaonkar. Structural brain network topology underpinning adhd and response to methylphenidate treatment. *Translational Psychiatry*, 11(1):150, Mar 2021. ISSN 2158-3188. doi:10.1038/s41398-021-01278-x. URL https://doi.org/10.1038/s41398-021-01278-x. → page 31

[75] O. Günlük, J. Kalagnanam, M. Li, M. Menickelly, and K. Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, pages 1–28, 2021. → page 11

[76] Y. Guo and C. Liang. Blockchain application and outlook in the banking industry. *Financial innovation*, 2(1):1–12, 2016. → page 3

[77] X. Han, T. F. J. Pasquier, A. Bates, J. Mickens, and M. I. Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. URL https://www.ndss-symposium.org/ndss-paper/unicorn-runtime-provenance-based-detector-for-advanced-persistent-threats/. → page 1

[78] O. Hanteer, L. Rossi, D. V. D'Aurelio, and M. Magnani. From interaction to participation: The role of the imagined audience in social media community

detection and an application to political communication on twitter. In *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '18, page 531–534. IEEE Press, 2018. ISBN 9781538660515. → pages 25, 26

[79] F. Hashemi, A. Behrouz, and L. V. Lakshmanan. Firmcore decomposition of multilayer networks. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 1589–1600, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi:10.1145/3485447.3512205. URL https://doi.org/10.1145/3485447.3512205. → page 2

[80] F. Hashemi, A. Behrouz, and M. R. Hajidehi. Cs-tgn: Community search via temporal graph neural networks. In *Companion Proceedings of the Web Conference 2023*, WWW '23, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3543873.3587654. URL https://doi.org/10.1145/3543873.3587654. → pages 13, 15

[81] M. U. Hassan, M. H. Rehmani, and J. Chen. Anomaly detection in blockchain networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2022. → page 3

[82] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016. → pages 25, 26

[83] N. A. Heard, D. J. Weston, K. Platanioti, and D. J. Hand. Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4 (2):645–662, 2010. → page 14

[84] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus), 2020. → pages 11, 43

[85] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 895–904, 2016. → page 14

[86] B. Jie, M. Liu, X. Jiang, and D. Zhang. Sub-network based kernels for brain network classification. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 622–629, 2016. → page 56

[87] M. Jin, Y. Liu, Y. Zheng, L. Chi, Y.-F. Li, and S. Pan. Anemone: Graph anomaly detection with multi-scale contrastive learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 3122–3126, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi:10.1145/3459637.3482057. URL https://doi.org/10.1145/3459637.3482057. → page 14

[88] M. Jin, Y.-F. Li, and S. Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=NqbktPUkZf7. → pages 15, 37, 39, 61

[89] B. Jing, C. Park, and H. Tong. Hdmi: High-order deep multiplex infomax. In *Proceedings of the Web Conference 2021*, pages 2414–2424, 2021. → page 16

[90] A. Kafshdar Goharshady, A. Behrouz, and K. Chatteriee. Secure credit reporting on the blockchain. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1343–1348, 2018. doi:10.1109/Cybermatics_2018.2018.00231. → page 3

[91] X. Kan, H. Cui, Y. Guo, and C. Yang. Effective and interpretable fmri analysis via functional brain network generation. *arXiv preprint arXiv:2107.11247*, 2021. → page 56

[92] X. Kan, H. Cui, J. Lukemire, Y. Guo, and C. Yang. Fbnetgen: Task-aware gnn-based fmri analysis via functional brain network generation. *arXiv preprint arXiv:2205.12465*, 2022. → page 56

[93] X. Kan, H. Cui, J. Lukemire, Y. Guo, and C. Yang. FBNETGEN: Task-aware GNN-based fMRI analysis via functional brain network generation. In *Medical Imaging with Deep Learning*, 2022. URL https://openreview.net/forum?id=oWFphg2lKon. → page 58

[94] X. Kan, W. Dai, H. Cui, Z. Zhang, Y. Guo, and C. Yang. Brain network transformer. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=1cJ1cbA6NLN. → page 55

[95] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019. → page 40

[96] P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707, 2020. → pages 3, 40

[97] J. Kim and J.-G. Lee. Community detection in multi-layer graphs: A survey. *SIGMOD Rec.*, 44(3):37–48, dec 2015. ISSN 0163-5808. doi:10.1145/2854006.2854013. URL https://doi.org/10.1145/2854006.2854013. → page 25

[98] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 07 2014. ISSN 2051-1310. doi:10.1093/comnet/cnu016. → pages 1, 16, 55

[99] A. R. Klivans, R. A. Servedio, and D. Ron. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7(4), 2006. → page 11

[100] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341, 2018. → pages 25, 27

[101] T. Lanciano, F. Bonchi, and A. Gionis. Explainable classification of brain networks via contrast subgraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*, KDD '20, page 3308–3318, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi:10.1145/3394486.3403383. URL https://doi.org/10.1145/3394486.3403383. → pages 3, 31, 57, 58, 66

[102] H. Laurent and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976. → page 11

[103] D. Lei, J. Ma, X. Du, G. Shen, X. Jin, and Q. Gong. Microstructural abnormalities in the combined and inattentive subtypes of attention deficit hyperactivity disorder: a diffusion tensor imaging study. *Scientific reports*, 4 (1):6875, 2014. → page 68

[104] G. Li, M. Muller, A. Thabet, and B. Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019. → pages 10, 21

[105] J. Li, Z. Han, H. Cheng, J. Su, P. Wang, J. Zhang, and L. Pan. Predicting path failure in time-evolving graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1279–1289, 2019. → page 15

[106] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ. → page 15

[107] J. Lin, C. Zhong, D. Hu, C. Rudin, and M. Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning (ICML)*, pages 6150–6160, 2020. → page 12

[108] J. Liu, M. Li, Y. Pan, W. Lan, R. Zheng, F.-X. Wu, and J. Wang. Complex brain network analysis and its applications to brain disorders: A survey. *Complexity*, 2017:8362741, Oct 2017. ISSN 1076-2787. doi:10.1155/2017/8362741. URL https://doi.org/10.1155/2017/8362741. → page 57

[109] J. Liu, W. Zhao, Y. Hong, S. Gao, X. Huang, Y. Zhou, A. D. N. Initiative, et al. Learning features of brain network for anomaly detection. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 900–905. IEEE, 2020. → pages 13, 56, 57

[110] N. Liu, X. Huang, and X. Hu. Accelerated local anomaly detection via resolving attributed networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2337–2343, 2017. doi:10.24963/ijcai.2017/325. URL https://doi.org/10.24963/ijcai.2017/325. → page 1

[111] Y. Liu, S. Pan, Y. G. Wang, F. Xiong, L. Wang, Q. Chen, and V. C. Lee. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021. doi:10.1109/TKDE.2021.3124061. → page 14

[112] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. → page 45

[113] Y. Luo and P. Li. Neighborhood-aware scalable temporal network representation learning. In *The First Learning on Graphs Conference*, 2022. URL https://openreview.net/forum?id=EPUtNe7a9ta. → page 15

[114] C. W. Lynn and D. S. Bassett. The physics of brain network structure, function and control. *Nature Reviews Physics*, 1(5):318–332, May 2019. ISSN 2522-5820. doi:10.1038/s42254-019-0040-8. URL https://doi.org/10.1038/s42254-019-0040-8. → page 57

[115] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021. doi:10.1109/TKDE.2021.3118815. → pages 1, 3, 14, 19

[116] D. D. F. Maesa, A. Marino, and L. Ricci. Detecting artificial behaviours in the bitcoin users graph. *Online Social Networks and Media*, 3:63–74, 2017. → page 3

[117] M. Mardani, G. Mateos, and G. B. Giannakis. Dynamic anomalography: Tracking network anomalies via sparsity and low rank. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):50–66, 2012. → page 14

[118] R. Marinescu, A. Eshaghi, D. Alexander, and P. Golland. Brainpainter: A software for the visualisation of brain structures, biomarkers and associated pathological processes. *arXiv preprint arXiv:1905.08627*, 2019. → page 59

[119] A. Maulana and M. Atzmueller. Centrality-based anomaly detection on multi-layer networks using many-objective optimization. In *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*, volume 1, pages 633–638, 2020. doi:10.1109/CoDIT49905.2020.9263819. → page 17

[120] H. McTavish, C. Zhong, R. Achermann, I. Karimalis, J. Chen, C. Rudin, and M. Seltzer. Fast sparse decision tree optimization via reference ensembles. In *AAAI Conference on Artificial Intelligence*, volume 36, 2022. → page 12

[121] S. Micali and Z. A. Zhu. Reconstructing markov processes from independent and anonymous experiments. *Discrete Applied Mathematics*, 200:108–122, 2016. → page 39

[122] R. Mittal and M. Bhatia. Anomaly detection in multiplex networks. *Procedia Computer Science*, 125:609–616, 2018. → page 16

[123] B. Mišić and O. Sporns. From regions to connections and networks: new bridges between brain and behavior. *Current Opinion in Neurobiology*, 40: 1–7, 2016. ISSN 0959-4388. doi:https://doi.org/10.1016/j.conb.2016.05.003. URL

https://www.sciencedirect.com/science/article/pii/S095943881630054X. Systems neuroscience. → page 54

[124] R.-A. Müller and A. Linke. Functional connectivity in autism spectrum disorders: Challenges and perspectives. *Brain Network Dysfunction in Neuropsychiatric Illness: Methods, Applications, and Implications*, pages 239–272, 2021. → page 69

[125] D. Nelson. Crypto criminals have already stolen $1.4b in 2020, says ciphertrace, June 2020. URL https://www.coindesk.com/policy/2020/06/02/crypto-criminals-have-already-stolen-14b-in-2020-says-ciphertrace/. → page 3

[126] S. Nijssen and E. Fromont. Mining optimal decision trees from itemset lattices. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 530–539, 2007. → page 11

[127] D. Ofori-Boateng, I. S. Dominguez, C. Akcora, M. Kantarcioglu, and Y. R. Gel. Topological anomaly detection in dynamic multilayer blockchain networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 788–804. Springer, 2021. → pages 3, 17, 25, 26

[128] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5363–5370, 2020. → page 15

[129] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020. → page 61

[130] C. Park, D. Kim, J. Han, and H. Yu. Unsupervised attributed multiplex network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5371–5378, 2020. → pages 16, 22, 37

[131] L. Peel and A. Clauset. Detecting change points in the large-scale structure of evolving networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. → page 14

[132] H. Peng, H. Wang, B. Du, M. Z. A. Bhuiyan, H. Ma, J. Liu, L. Wang, Z. Yang, L. Du, S. Wang, et al. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences*, 521: 277–290, 2020. → page 15

[133] Z. Peng, M. Luo, J. Li, H. Liu, and Q. Zheng. Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3513–3519. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/488. URL https://doi.org/10.24963/ijcai.2018/488. → page 1

[134] Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng. A deep multi-view framework for anomaly detection on attributed networks. *IEEE Transactions on Knowledge and Data Engineering*, 34(6):2539–2552, 2022. doi:10.1109/TKDE.2020.3015098. → page 1

[135] T. Pham and S. Lee. Anomaly detection in bitcoin network using unsupervised learning methods. *arXiv preprint arXiv:1611.03941*, 2016. → page 3

[136] L. Pio-Lopez, A. Valdeolivas, L. Tichit, É. Remy, and A. Baudot. Multiverse: a multiplex and multiplex-heterogeneous network embedding approach. *Scientific Reports*, 11(1):1–20, 2021. → pages 15, 16

[137] B. Podgorelec, M. Turkanović, and S. Karakatič. A machine learning-based method for automated blockchain transaction signing including personalized anomaly detection. *Sensors*, 20(1):147, 2019. → page 3

[138] F. Poursafaei, A. Huang, K. Pelrine, and R. Rabbany. Towards better evaluation for dynamic link prediction. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=1GVpwr2Tfdg. → pages 35, 43

[139] J. D. Power, A. L. Cohen, S. M. Nelson, G. S. Wig, K. A. Barnes, J. A. Church, A. C. Vogel, T. O. Laumann, F. M. Miezin, B. L. Schlaggar, and S. E. Petersen. Functional network organization of the human brain. *Neuron*, 72(4):665–678, Nov 2011. ISSN 1097-4199. doi:10.1016/j.neuron.2011.09.006. URL https://pubmed.ncbi.nlm.nih.gov/22099467. S0896-6273(11)00792-6[PII]. → page 3

[140] M. G. Preti, T. A. Bolton, and D. Van De Ville. The dynamic functional connectome: State-of-the-art and perspectives. *Neuroimage*, 160:41–54, 2017. → pages 54, 55

[141] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova. Anomaly detection in dynamic networks: A survey. *WIREs Comput. Stat.*, 7(3):223–247, may 2015. ISSN 1939-5108. doi:10.1002/wics.1347. URL https://doi.org/10.1002/wics.1347. → pages 1, 3, 19

[142] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova. A scalable approach for outlier detection in edge streams using sketch-based approximations. In *Proceedings of the 2016 SIAM international conference on data mining*, pages 189–197. SIAM, 2016. → pages 2, 14, 27

[143] T. D. Rogers, E. McKimm, P. E. Dickson, D. Goldowitz, C. D. Blaha, and G. Mittleman. Is autism a disease of the cerebellum? an integration of clinical and pre-clinical research. *Front Syst Neurosci*, 7:15, May 2013. → page 69

[144] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022. → page 11

[145] D. A. Seminowicz and K. D. Davis. Pain enhances functional connectivity of a brain network evoked by performance of a cognitive task. *Journal of neurophysiology*, 97(5):3651–3659, 2007. → page 13

[146] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pages 362–373. Springer, 2018. → page 15

[147] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, 2006. → page 14

[148] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 366–377. SIAM, 2007. → page 14

[149] A. Taheri, K. Gimpel, and T. Berger-Wolf. Learning to represent the evolution of dynamic graphs with recurrent models. In *Companion proceedings of the 2019 world wide web conference*, pages 301–307, 2019. → page 15

[150] X. Teng, Y.-R. Lin, and X. Wen. Anomaly detection in dynamic networks using multi-view time-series hypersphere learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 827–836, 2017. → page 14

[151] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. P. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=EI2KOXKdnP. → pages 11, 43

[152] M. P. Van Den Heuvel and H. E. H. Pol. Exploring the brain network: a review on resting-state fmri functional connectivity. *European neuropsychopharmacology*, 20(8):519–534, 2010. → page 13

[153] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. → page 11

[154] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ. → pages 59, 61

[155] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rklz9iAcKQ. → page 16

[156] H. Wang, M. Tang, Y. Park, and C. E. Priebe. Locality statistics for anomaly detection in time series of graphs. *IEEE Transactions on Signal Processing*, 62(3):703–717, 2013. → page 14

[157] J. Wang, R. Wen, C. Wu, Y. Huang, and J. Xiong. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 310–316, New York, NY, USA, 2019. Association for Computing

Machinery. ISBN 9781450366755. doi:10.1145/3308560.3316586. URL https://doi.org/10.1145/3308560.3316586. → page 1

[158] J.-z. Wang, T.-z. Jiang, Q.-j. Cao, and Y. Wang. Characterizing anatomic differences in boys with attention-deficit/hyperactivity disorder with the use of deformation-based morphometry. *American Journal of Neuroradiology*, 28(3):543–547, 2007. → pages 32, 68

[159] Q. Wang, Y. Fang, A. Ravula, R. He, B. Shen, J. Wang, X. Quan, and D. Liu. Deep partial multiplex network embedding. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 1053–1062, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391306. doi:10.1145/3487553.3524717. URL https://doi.org/10.1145/3487553.3524717. → page 16

[160] T. Wang, C. Fang, D. Lin, and S. F. Wu. Localizing temporal anomalies in large evolving graphs. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 927–935. SIAM, 2015. → pages 2, 14

[161] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, and S. Mou. Dynamic heterogeneous information network embedding with meta-path based proximity. *IEEE Transactions on Knowledge and Data Engineering*, 2020. → page 16

[162] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, pages 1082–1092, 2020. → page 15

[163] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KYPz4YsCPj. → pages 15, 37, 38, 39, 40, 61, 62

[164] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014. → page 23

[165] C.-Y. Wee, P.-T. Yap, W. Li, K. Denny, J. N. Browndyke, G. G. Potter, K. A. Welsh-Bohmer, L. Wang, and D. Shen. Enriched white matter connectivity networks for accurate identification of mci patients. *Neuroimage*, 54(3): 1812–1822, 2011. → page 56

[166] Y. Xie, Z. Ou, L. Chen, Y. Liu, K. Xu, C. Yang, and Z. Zheng. Learning and updating node embedding on dynamic heterogeneous information network. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 184–192, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. doi:10.1145/3437963.3441745. URL https://doi.org/10.1145/3437963.3441745. → page 16

[167] Z. Xiong, D. Wang, X. Liu, F. Zhong, X. Wan, X. Li, Z. Li, X. Luo, K. Chen, H. Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019. → page 55

[168] H. Yan, Q. Zhang, D. Mao, Z. Lu, D. Guo, and S. Chen. Anomaly detection of network streams via dense subgraph discovery. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2021. → page 14

[169] Y. Yan, S. Zhang, and H. Tong. Bright: A bridging algorithm for network alignment. In *Proceedings of the Web Conference 2021*, pages 3907–3917, 2021. → pages 15, 16

[170] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han. Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond. *CoRR*, abs/2004.00216, 2020. URL https://arxiv.org/abs/2004.00216. → page 16

[171] C. Yang, C. Wang, Y. Lu, X. Gong, C. Shi, W. Wang, and X. 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, 2022. → page 15

[172] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos. Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 647–657, 2019. → page 14

[173] J. You, R. Ying, and J. Leskovec. Design space for graph neural networks. In *NeurIPS*, 2020. → page 25

[174] J. You, T. Du, and J. Leskovec. Roland: Graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 2358–2366, New York, NY, USA, 2022. Association for Computing Machinery. ISBN

9781450393850. doi:10.1145/3534678.3539300. URL https://doi.org/10.1145/3534678.3539300. → pages 15, 18, 21

[175] H. Yousaf, G. Kappos, and S. Meiklejohn. Tracing transactions across cryptocurrency ledgers. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 837–850, 2019. → page 3

[176] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, pages 3634–3640, 2018. URL https://doi.org/10.24963/ijcai.2018/505. → page 15

[177] T. Yu, X. Li, Y. Cai, M. Sun, and P. Li. S2-mlp: Spatial-shift mlp architecture for vision. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 297–306, 2022. → page 11

[178] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang. On anomalous hotspot discovery in graph streams. In *2013 IEEE 13th International Conference on Data Mining*, pages 1271–1276. IEEE, 2013. → page 14

[179] W. Yu, C. C. Aggarwal, and W. Wang. Temporally factorized network modeling for evolutionary network analysis. In *Proceedings of the Tenth ACM International conference on web search and data mining*, pages 455–464, 2017. → page 14

[180] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2672–2681, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi:10.1145/3219819.3220024. URL https://doi.org/10.1145/3219819.3220024. → pages 1, 2, 14, 27, 28

[181] X. Zhai, W. Zhou, G. Fei, W. Liu, Z. Xu, C. Jiao, C. Lu, and G. Hu. Null model and community structure in multiplex networks. *Scientific reports*, 8 (1):1–13, 2018. → page 27

[182] C. Zhang, H. Fu, Q. Hu, X. Cao, Y. Xie, D. Tao, and D. Xu. Generalized latent multi-view subspace clustering. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):86–99, 2018. → pages 15, 16

[183] H. Zhang, L. Qiu, L. Yi, and Y. Song. Scalable multiplex network embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3082–3088.

International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/428. URL https://doi.org/10.24963/ijcai.2018/428. → pages 16, 28

[184] J. Zhang, B. Cao, S. Xie, C.-T. Lu, P. S. Yu, and A. B. Ragin. Identifying connectivity patterns for brain diseases via multi-side-view guided deep architectures. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 36–44. SIAM, 2016. → page 56

[185] R.-Y. Zhang, X.-X. Wei, and K. Kay. Understanding multivariate brain activity: Evaluating the effect of voxelwise noise correlations on population codes in functional magnetic resonance imaging. *PLOS Computational Biology*, 16(8):1–29, 08 2020. doi:10.1371/journal.pcbi.1008153. URL https://doi.org/10.1371/journal.pcbi.1008153. → pages 3, 55, 57, 58, 66

[186] X. Zhang, L. He, K. Chen, Y. Luo, J. Zhou, and F. Wang. Multi-View graph convolutional network and its applications on neuroimage analysis for parkinson's disease. *AMIA Annu Symp Proc*, 2018:1147–1156, Dec. 2018. → pages 55, 57, 58

[187] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2019. → page 15

[188] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao. Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4419–4425. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi:10.24963/ijcai.2019/614. URL https://doi.org/10.24963/ijcai.2019/614. → pages 1, 2, 14, 23, 27, 28, 43, 44

[189] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y.-P. P. Chen. Generative and contrastive self-supervised learning for graph anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 2021. → page 14

[190] Y. Zhu, H. Cui, L. He, L. Sun, and C. Yang. Joint embedding of structural and functional brain networks with graph neural networks for mental illness diagnosis. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 272–276. IEEE, 2022. → pages 55, 56

[191] Y. Zhu, H. Cui, L. He, L. Sun, and C. Yang. Joint embedding of structural and functional brain networks with graph neural networks for mental illness diagnosis, 2022. → pages 12, 55, 57, 58

Appendix A

Supporting Materials

A.1 Performance of the Modified Attention on General Datasets

In Section 4.2.3, we define a complex attention mechanism that can learn the importance of each view for each node, while in Section 6.4, we design a more scalable mechanism that considers the importance of a view for other views, independent of the nodes. We discuss its advantages for brain networks in Section 6.4 and 6.5 but its performance on general multiplex networks remains unexplored. Here, we explore its performance for general multiplex networks (different domains). Table A.1 reports the results. ANOMULY and ADMIRE uses the complex attention mechanism we proposed in Section 4.2.3, while ANOMULY-v2 and ADMIRE-v2 uses the scalable attention mechanism that proposed in Section 6.4. The results show the superior performance of the complex attention mechanism. As motivated in Section 6.4, the modified attention mechanism is useful for improving the scalability in the domain of human brain networks.

Table A.1: Performance of the modified attention on general datasets (AUC).

Methods	RM		DKPol		Amazon		DBLP		Ethereum		Ripple	
Anomaly %	1%	5 %	1%	5 %	1%	5 %	1%	5 %	1%	5 %	1%	5 %
AnoMuLY	0.8783	0.8729	0.8694	0.8610	**0.8289**	**0.8195**	0.8825	0.8754	**0.8906**	**0.8852**	0.8938	0.8871
ADMire	**0.9016**	**0.9125**	**0.9093**	**0.8952**	0.8097	0.8041	**0.8873**	**0.8824**	0.8416	0.8392	**0.9160**	**0.8899**
AnoMuLY-v2	0.8416	0.8387	0.8502	0.8439	0.7990	0.8021	0.8235	0.7992	0.8421	0.8399	0.8562	0.8286
ADMire-v2	0.8912	0.8841	0.8498	0.8515	0.7975	0.7926	0.8129	0.7953	0.8478	0.8257	0.0000	0.0000