Anomaly Detection in Multiplex Dynamic Networks: from Blockchain Security to Brain Disease Prediction

Ali Behrouz  
Department of Computer Science  
University of British Columbia  
Vancouver, BC, Canada  
alibez@cs.ubc.ca

Margo Seltzer  
Department of Computer Science  
University of British Columbia  
Vancouver, BC, Canada  
mseltzer@cs.ubc.ca

Abstract

The problem of identifying anomalies in dynamic networks is a fundamental task with a wide range of applications. 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 networks with a single type of connection between vertices, while in many applications interactions between objects vary, yielding multiplex networks. We propose ANOMULY, a general, unsupervised edge anomaly detection framework for multiplex, dynamic networks. In each relation type, ANOMULY sees node embeddings at different GNN layers as hierarchical node states and employs a GRU cell to capture temporal properties of the network and update node embeddings over time. We then add an attention mechanism that incorporates information across different types of relations. Our case study on brain networks shows how this approach could be employed as a new tool to understand abnormal brain activity that might reveal a brain disease or disorder. Extensive experiments on nine real-world datasets demonstrate that ANOMULY achieves state-of-the-art performance.

1 Introduction

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 [1, 2, 3, 4, 5]. While several anomaly detection approaches focus on the topological properties of networks [6, 7, 8, 9, 10, 11], detecting anomalies in real-world networks also requires attention to their dynamic nature [5]. Anomalies might appear as malware in computer systems [12], social bots and social spammers in social networks [13], or financial fraud in financial systems [1, 14]. 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 [5, 15, 16, 17, 18, 19, 20, 21]; these networks are called single-layer, dynamic networks. However, in many complex dynamic systems, 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 [22]. 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...
Anomaly detection in (dynamic) Applications: Fraud Detection in Multiple Blockchain Networks.

which is responsible for a specific function, and edges represent high functional correlation between
Applications: Brain Networks.

Monitoring functional systems in the human brain is a fundamental (3)
We demonstrate a new application of edge-anomaly detection in dynamic multiplex networks and
(fMRI), which lets us measure the statistical association between the functionality of ROIs over time.

Same importance for all type of connections: these methods treat each relation type (i.e., layer)
identically, assigning the same importance to each layer. However, real-world multiplex networks
contain noisy/insignificant layers. Moreover, all vertices might not participate equally in
all layers, so which layers are noisy/insignificant can be different for each vertex. (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. 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.

Existing methods for anomaly detection in single-layer dynamic graphs also exhibit limitations. (1)
Structure inflexibility: even in single layer networks, most existing anomaly detection methods for
dynamic networks rely on pre-defined patterns or heuristic rules (see [[23],[26]]). These heuristic rules
are usually content features or long-term temporal factors. However, due the complex nature of
real-world anomalies, these factors are not flexible and are restricted to specific patterns. (2) Memory
usage: deep learning based methods [[19],[20]], which are commonly proposed, require storing entire
snapshots of the network at each time window, consuming large and increasing amounts of memory.

To mitigate the above limitations, we introduce ANOMULY (Anomaly Detection in Multiplex Dynamic Networks). To take advantage of both temporal properties and complementary information
present in multiple relation (edge) types, ANOMULY extends the idea of hierarchical node states [27]
to multiplex dynamic networks by using an attention mechanism that incorporates information about
different 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 outperform existing simple network approaches, because the multiplex network
provides richer and more complex information than does a simple network [3],[4],[5].

Consider the following two applications for anomaly detection in dynamic multiplex networks:

Applications: Brain Networks. Monitoring functional systems in the human brain is a fundamental
task in neuroscience [28],[29]. Each node in a brain network represents a region of interest (ROI),
which is responsible for a specific function, and edges represent high functional correlation between
two ROIs. A temporal brain network is usually derived from functional magnetic resonance imaging
(fMRI), which 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 incomplete [30],[31],
prior work used the average of brain networks from many individuals [32],[33]. 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 [34],[31], where each layer
represents an individual’s brain network. We show that ANOMULY detects abnormal connections in
the brains of people with attention deficit hyperactivity disorder (ADHD). (see section 4).

Applications: Fraud Detection in Multiple Blockchain Networks. Anomaly detection in (dynamic)
blockchain transaction networks has recently attracted enormous attention [35],[36],[37],[38],[39],[40], due
to the emergence of a huge assortment of financial systems’ applications [41],[42],[43],. While most existing
work focuses on detecting illicit activity in a single blockchain network, recent research shows
that cryptocurrency criminals increasingly employ cross-cryptocurrency trades to hide their
identity [44],[45]. Accordingly, Yousaf et al. [46] have recently shown 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 employed to detect suspicious transactions
and identify criminal activities across several blockchain transaction networks more accurately.

The contributions of this work are: (1) We present 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. (2) We present ANOMULY, a
general end-to-end unsupervised learning method for anomalous edge detection in multiplex dynamic
networks, using a GRU cell to incorporate the outputs of Snapshot Encoder for different snapshots.
(3) We demonstrate a new application of edge-anomaly detection in dynamic multiplex networks and
present a case study on brain networks of people living with attention deficit hyperactivity disorder
(ADHD). Our results show the effectiveness and usefulness of ANOMULY in identifying abnormal connections of different ROIs of the human brain. This approach could be employed as a new tool to understand abnormal brain activity that might reveal a disease or disorder. We conduct extensive experiments on nine real-world multiplex and simple networks. Results show the superior performance of ANOMULY in both single-layer and multiplex networks.

2 Related Work

We begin with a brief review of anomaly detection algorithms in dynamic simple networks, then methods for dynamic and multiplex graph learning, and finally anomaly detection in multiplex networks. To situate our research in a broader context, we discuss work on anomaly detection in brain and blockchain networks (Appendix A). For additional related work, we refer the reader to the extensive survey by Ma et al. [4].

Anomaly Detection in Dynamic Networks. There has been much work in anomaly detection for single-layer dynamic networks. That work falls in five categories: (1) Probabilistic methods [15, 47, 48, 49, 50, 51, 52] that identify anomalies based on deviations from regular communication patterns. (2) Distance-based methods [16, 18, 25, 26, 53] that use certain time-evolving measures of dynamic network structures and use their changes to detect anomalies. (3) Density-based methods [54, 55, 56] that view anomalies as subgraphs with high density or as subgraphs with sudden changes in their density. (4) Matrix factorization methods [57, 58, 59] that use the low-rank property of network structures and define anomalies as breakers of this property. (5) Learning-based methods [19, 20, 62, 63, 64, 65], that combine the graph embedding method into the anomaly detection approach. These learning-based models must store the entire snapshot, which requires large memory, limiting their scalability. To show the effectiveness of ANOMULY in even single-layer networks, we compare it with NETWALK [20], and ADDGRAPH [19] in section 4. All these methods apply to single-layer networks only and do not naturally extend to multiplex networks. We further discuss the novelty of the architecture of our approach in Appendix A.

Dynamic Graph Neural Networks. The problem of learning from dynamic networks has been extensively studied in the literature [66, 67, 68, 69, 70, 71, 72, 73]. The first group of existing methods use Recurrent Neural Networks (RNN) and then replace the linear layer with a graph convolution layer [72, 74, 75]. The second group uses a GNN as a feature encoder and then deploys a sequence model on top of the GNN to encode temporal properties [67, 76, 77]. However, all these models have limitations in both model design and training strategy [27]. To address these limitations, You et al. [27] proposed ROLAND, a graph learning framework for dynamic graphs that can re-purpose any static GNN to dynamic graphs. However, this framework cannot be used for graphs with different types of edges (multiplex networks). Our work extends ROLAND to multiplex networks and introduces an attention mechanism that incorporates the relation-specific hierarchical node states in each snapshot, taking advantage of additional information present in multiplex networks.

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 [22]. Several methods have been proposed to learn network embeddings on multiplex networks by integrating information from individual relation type [78, 79, 80, 81, 82, 83]. Other work proposed Graph Convolutional Networks (GCN) methods for multiplex networks [84, 85, 86]. Inspired by Deep Graph Infomax [87], Park et al. [88] 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. [90] proposed a method that uses a latent space to integrate the information across multiple views. Recently, Wang et al. [91] proposed DPMNE to learn from incomplete multiplex networks.

Anomaly Detection in Multiplex Networks. The problem of anomaly detection in multiplex networks has recently attracted attention. Mittal and Bhatia [22] use eigenvector centrality, page rank centrality, and degree centrality as handcrafted features for nodes to detect anomalies in static multiplex networks. Bindu et al. [92] 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 [94] defined a quality measure, Multi-Normality, which employs the structure and attributes together of each layer to detect attribute coherence in neighborhoods between layers.
We now introduce our framework for edge anomaly detection in dynamic networks with multiple types of interactions. Figure 1 provides an overview of the framework. To learn the pattern of normal edges, ANOMULY uses the Snapshot Encoder architecture to encode each snapshot of the network. Snapshot Encoder captures the structural information of each snapshot and generates a representation that captures the temporal dynamics of the network.

Figure 1: Framework and design of ANOMULY model.

Maulana and Atzmueller [95] 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. [96] proposed ANOMAN 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. [45] 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 1, these methods, except ANOMAN [96], are based on pre-defined patterns/roles or handcrafted features, while real-world network anomalies have complex nature. Therefore, these models cannot be generalized to different domains, limiting their application.

3 Proposed Method: ANOMULY

3.1 Preliminaries

We first precisely define multiplex dynamic networks, and then we formalize the problem of edge anomaly detection in multiplex dynamic networks.

**Definition 1 (Multiplex Dynamic Networks).** Let \( \mathcal{G} = \{G_r\}_{r=1}^C = (\mathcal{V}, \mathcal{E}, \mathcal{X}) \) denote a multiplex dynamic network, where \( G_r = (\mathcal{V}, \mathcal{E}_r, \mathcal{X}) \) is a graph of the relation type \( r \in \mathcal{L} \) (aka layer), \( \mathcal{V} \) is the set of nodes, \( \mathcal{E} = \bigcup_{r=1}^C \mathcal{E}_r \) is the set of edges, and \( \mathcal{X} \in \mathbb{R}^{\mathcal{V} \times \mathcal{E}} \) is a matrix that encodes node attribute information for nodes in \( \mathcal{V} \). Given a relation type \( r \), we denote the set of vertices in the neighborhood of \( u \in \mathcal{V} \) in relation \( r \) as \( \mathcal{N}_r(u) \). Each edge \( e = (u, v, r_e) \in \mathcal{E} \) is associated with an edge type (layer) \( r_e \in \mathcal{L} \) and a timestamp \( \tau_e \), and each node \( v \in \mathcal{V} \) is associated with a timestamp \( \tau_v \).

We take a snapshot-based anomaly detection approach for multiplex dynamic networks: a multiplex dynamic graph \( \mathcal{G} = \{G^{(t)}\}_{t=1}^T \) can be represented as a sequence of multiplex network snapshots, where each snapshot is a static multiplex graph \( G^{(t)} = \{G^{(t)}_r\}_{r=1}^C = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)}, \mathcal{X}^{(t)}) \) 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 layer-dependent anomalous probability \( \phi_r(e) \) in layer \( r \in \mathcal{L} \).

3.2 ANOMULY Framework

We now introduce our framework for edge anomaly detection in dynamic networks with multiple types of interactions. Figure 1 provides an overview of the framework. To learn the pattern of normal edges, ANOMULY uses the Snapshot Encoder architecture to encode each snapshot of the network. Snapshot
Encoder uses GNNs and incorporates structural and temporal properties of the graph as well as node features in each layer. Next, it uses an attention mechanism to take advantage of complementary information from different layers. Since the graph is dynamically changing, the node embeddings need to change. To this end, Snapshot Encoder uses a GRU cell \([77]\) to update the hierarchical node states over time. Finally, we use the hierarchical node states at each timestamp to calculate the layer-dependent anomalous probabilities of an existing edge and a negative sampled edge and use them as inputs to a margin loss computation. ANOMULY’s algorithm appears in detail in Appendix B.

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 \(\hat{H}_r^{(\ell)} = \{\hat{h}_u^{(\ell)}\}_{u \in V}\) to denote the embedding of all vertices in relation type \(r\) after applying \(\ell\)-th GNN layer. Given a relation type \(r\), the \(\ell\)-th layer of the GNN, \(\hat{H}_r^{(\ell)} = \text{GNN}_r(\hat{H}_r^{(\ell-1)})\), is defined as:

\[
\begin{align*}
\mathbf{m}_{(v \rightarrow u)}^{\ell} &= \mathbf{W}_r^{(\ell)} \text{CONCAT} \left( \hat{h}_u^{(\ell-1)}, \hat{h}_v^{(\ell-1)}, \tau(v,u,r) \right), \\
\hat{h}_u^{(\ell)} &= \text{AGG}^{(\ell)} \left( \{ \mathbf{m}_{(v \rightarrow u)}^{\ell} | v \in N_r(u) \} \right) + \tilde{h}_u^{(\ell-1)}.
\end{align*}
\]

(1)

In our experiments, we follow You et al. \([27]\), and use summation as the aggregation function, i.e., \(\text{AGG}(\cdot) = \text{SUM}(\cdot)\). We also use skip-connections \([98]\) after aggregation.

Update Modules. Given the snapshot \(G^{(t)} = (V^{(t)}, E^{(t)}, X^{(t)})\) at time \(t\) and a relation type \(r\), we denote the embedding matrix in relation \(r\) after the \(\ell\)-th GNN layer at time \(t\) by \(\hat{H}_r^{(t)(\ell)} = \{\hat{h}_u^{(t)(\ell)}\}_{u \in V}\). To take advantage of historical data and update the node embeddings at each timestamp, in the Embedding update block (see Figure 1), we use a GRU cell \([77]\). Given a relation type \(r\), the output of the \(\ell\)-th Embedding update, \(\hat{H}_r^{(t)(\ell)} = \{\hat{h}_u^{(t)(\ell)}\}_{u \in V}\), is:

\[
\hat{H}_r^{(t)(\ell)} = \text{GRU}_r \left( \hat{H}_r^{(t)(\ell)}, H_r^{(t-1)(\ell)} \right),
\]

(2)

where \(\hat{H}_r^{(t)(\ell)}\) is the output of the \(\ell\)-th GNN layer, and \(H_r^{(t-1)(\ell)}\) is the layer-aware embedding matrix at time \(t - 1\). The layer-aware embedding matrix will be defined later in this section (see Equation 5).

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

\[
\zeta_u^{(t)(\ell)} = \sum_{r=1}^{L} \alpha_{r_u}^{(t)} \hat{h}_u^{(t)(\ell)},
\]

(3)

where \(\hat{h}_u^{(t)(\ell)}\) is the output of \(\ell\)-th Embedding update for node \(u\) at time \(t\). Following the recent attention-based models \([88, 99]\), we use the softmax function to define the importance weight of relation type \(r\) for node \(u\):

\[
\alpha_{r_u}^{(t)} = \frac{\exp \left( \sigma \left( S_r^{(t)(\ell)} W_r^{att} \hat{h}_u^{(t)(\ell)} \right) \right)}{\sum_{k=1}^{L} \exp \left( \sigma \left( S_r^{(t)(\ell)} W_k^{att} \hat{h}_k^{(t)(\ell)} \right) \right)},
\]

(4)

where \(S_r^{(t)(\ell)}\) is a summary of the network in relation type \(r\) at time \(t\), i.e., \(S_r^{(t)(\ell)} = \sum_{u \in V} \hat{h}_u^{(t)(\ell)}\), and \(W_r^{att}\) is a trainable weight matrix. In our experiments, we use \(\text{tanh}(\cdot)\) as the activation function \(\sigma(\cdot)\).

Layer-aware 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 layer-aware node embedding of a vertex \(u \in V\), we aggregate the output of the Embedding
We tune hyper-parameters by cross-validation on a rolling basis and search the hyper-parameters on a V100 GPU in the experiments. We use the AUC (the area under the ROC curve) as the metric of performance.

Now, we get the layer-aware node embedding matrix \( H_{r_u}^{(t)} \), and this network-level node embedding, i.e., \( \zeta_{r_u}^{(t)} \). That is:

\[
H_{r_u}^{(t)} = \text{AGG}^{(t)} \left( \hat{h}_{r_u}^{(t)}, \zeta_{r_u}^{(t)} \right). 
\]  

(5)

Based on Equation 5, we obtain the layer-aware node embedding matrix, \( H_{r_u}^{(t)} = \{ h_{r_u}^{(t)} \}_{u \in V} \), for any relation type \( r \). Note that we use the layer-aware node embedding matrix at time \( t - 1, H_{r_u}^{(t-1)} \), in Equation 2 to update node embeddings after the \( \ell \)-layer GNN layer.

**Anomalous Score Computation.** Now, we get the layer-aware node embedding matrix \( H_{r_u}^{(t)} = H_{r_u}^{(t)} \) at time \( t \), for each relation type \( r \). Here \( L \) is the number of GNN layers. Inspired by Zheng et al. [19], for an edge \( (u, v) \in E_r \), we define its anomalous score as follows:

\[
\varphi_r^{(t)}(u, v) = \sigma \left( \eta \left( ||a \odot \hat{h}_{r_u}^{(t)} + b \odot h_{r_u}^{(t)}||_2^2 - \mu \right) \right),
\]

(6)

where \( \sigma(.) \) is an activation function, \( a \) and \( b \) are trainable vectors, and \( \eta \) and \( \mu \) are hyperparameters.

**Training and Loss Function.** In the training phase, we employ a negative sampling approach in multiplex networks to corrupt edges and generate anomalous connections. Inspired by the negative sampling methods proposed by Wang et al. [100] and Zheng et al. [19], given a relation type \( r \), and a normal edge \( (u, v) \in E_r \), we employ a Bernoulli distribution such that we replace \( u \) (resp. \( v \)) with probability \( \frac{\text{deg}_r(u)}{\text{deg}_r(u) + \text{deg}_r(v)} \) (resp. \( \frac{\text{deg}_r(v)}{\text{deg}_r(u) + \text{deg}_r(v)} \)) in relation type \( r \) to generate random negative samples. Since the corrupted edges might be normal, a strict loss function (e.g., cross-entropy) can affect the performance. Accordingly, we employ the margin-based pairwise loss [101] in each relation type \( r \). Given a relation type \( r \), we also employ a \( L2 \)-regularization loss, \( \mathcal{L}^{reg}_{r} \), which is the summation of the \( L2 \) norm of all trainable parameters, to 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.:

\[
\mathcal{L} = \frac{1}{|E|} \left( \sum_{r=1}^{L} \sum_{(u,v) \in E_r} \sum_{(u',v') \in E_r} \max \left\{ 0, \gamma + \varphi_r(u, v) - \varphi_r(u', v') \right\} + \lambda \mathcal{L}^{reg}_{r} \right).
\]

(7)

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

### 4 Experiments

**Experimental Setup and Metrics.** The **Snapshot Encoder** employs 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 layers (1 to 5); (ii) learning rate (0.001 to 0.01); and (iii) the margin \( \gamma \) (in increments of 0.05 in the range 0.3 to 0.7). The values of the hyper-parameters are reported in **Table 5** in **Appendix D**. We implement ANOMULY with the GraphGym library [102] 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.

**Datasets.** We use nine real-world public datasets [31, 45, 103, 104, 105, 106] whose domains cover social, co-authorship, blockchain, and co-purchasing networks. We summarize their statistics in **Table 1** and provide detailed descriptions in **Appendix C**. Since the ground truth for anomaly detection is difficult to obtain [3], we extend the methodology in existing studies [3, 19, 20] to multiplex networks and propose a new approach to inject anomalous edges into our datasets (more detailed in **Appendix C**).
Ablation Study. Next, we conduct experiments to show that the \textsc{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 summarizes the results. We found that both our attention mechanism and the GRU cells are important for \textsc{AnoMuly}, producing significant performance boosts.

### Table 2: Performance comparison in multiplex networks (AUC).

<table>
<thead>
<tr>
<th>Methods</th>
<th>RM</th>
<th>DKPol</th>
<th>Amazon</th>
<th>DBLP</th>
<th>Ethereum</th>
<th>Ripple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly %</td>
<td>1%</td>
<td>5%</td>
<td>1%</td>
<td>5%</td>
<td>1%</td>
<td>5%</td>
</tr>
<tr>
<td>Single-layer Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOutlier</td>
<td>0.7138</td>
<td>0.6982</td>
<td>0.6844</td>
<td>0.6597</td>
<td>0.6973</td>
<td>0.6672</td>
</tr>
<tr>
<td>CM-SKETCH</td>
<td>0.7127</td>
<td>0.7012</td>
<td>0.7058</td>
<td>0.6930</td>
<td>0.6881</td>
<td>0.6719</td>
</tr>
<tr>
<td>NetWalk</td>
<td>0.7739</td>
<td>0.7641</td>
<td>0.7706</td>
<td>0.7581</td>
<td>0.7228</td>
<td>0.7122</td>
</tr>
<tr>
<td>AddGraph</td>
<td>0.8005</td>
<td>0.8093</td>
<td>0.8149</td>
<td>0.8087</td>
<td>0.7796</td>
<td>0.7735</td>
</tr>
<tr>
<td>ANOMUy</td>
<td>0.8783</td>
<td>0.8729</td>
<td>0.8694</td>
<td>0.8610</td>
<td>0.8209</td>
<td>0.8195</td>
</tr>
<tr>
<td>Multiplex Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNE</td>
<td>0.7994</td>
<td>0.7955</td>
<td>0.8050</td>
<td>0.7913</td>
<td>0.7108</td>
<td>0.7017</td>
</tr>
<tr>
<td>ML-GCN</td>
<td>0.7921</td>
<td>0.7886</td>
<td>0.7915</td>
<td>0.7907</td>
<td>0.7344</td>
<td>0.7263</td>
</tr>
<tr>
<td>ANOMUy</td>
<td>0.8783</td>
<td>0.8729</td>
<td>0.8694</td>
<td>0.8610</td>
<td>0.8209</td>
<td>0.8195</td>
</tr>
<tr>
<td>Improvement</td>
<td>9.71%</td>
<td>7.85%</td>
<td>6.68%</td>
<td>6.46%</td>
<td>6.32%</td>
<td>5.92%</td>
</tr>
</tbody>
</table>

### Table 3: Performance comparison in simple networks (AUC).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Bitcoin</th>
<th>Amazon-S</th>
<th>DBLP-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly %</td>
<td>1%</td>
<td>5%</td>
<td>1%</td>
</tr>
<tr>
<td>Single-layer Methods</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOutlier</td>
<td>0.7143</td>
<td>0.7091</td>
<td>0.6923</td>
</tr>
<tr>
<td>CM-SKETCH</td>
<td>0.7146</td>
<td>0.7015</td>
<td>0.7049</td>
</tr>
<tr>
<td>NetWalk</td>
<td>0.8375</td>
<td>0.8367</td>
<td>0.7483</td>
</tr>
<tr>
<td>AddGraph</td>
<td>0.8534</td>
<td>0.8416</td>
<td>0.7872</td>
</tr>
<tr>
<td>ANOMUy</td>
<td>0.8707</td>
<td>0.8661</td>
<td>0.8014</td>
</tr>
<tr>
<td>Multiplex Methods</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNE</td>
<td>0.7994</td>
<td>0.7955</td>
<td>0.8050</td>
</tr>
<tr>
<td>ML-GCN</td>
<td>0.7921</td>
<td>0.7886</td>
<td>0.7915</td>
</tr>
<tr>
<td>ANOMUy</td>
<td>0.8783</td>
<td>0.8729</td>
<td>0.8694</td>
</tr>
<tr>
<td>Improvement</td>
<td>9.71%</td>
<td>7.85%</td>
<td>6.68%</td>
</tr>
</tbody>
</table>

### Table 4: Ablation study (AUC).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Amazon</th>
<th>DBLP</th>
<th>Ethereum</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOMUy</td>
<td>0.8289</td>
<td>0.8825</td>
<td>0.8906</td>
</tr>
<tr>
<td>w/ MLP</td>
<td>0.8023</td>
<td>0.8606</td>
<td>0.8718</td>
</tr>
<tr>
<td>w/o Attention</td>
<td>0.7831</td>
<td>0.8219</td>
<td>0.8275</td>
</tr>
<tr>
<td>w/ Summation</td>
<td>0.8007</td>
<td>0.8571</td>
<td>0.8688</td>
</tr>
</tbody>
</table>

Details are in Appendix C. Note that the DKPol and RM datasets are static multiplex networks, and we use them to show the effectiveness of \textsc{AnoMuly} in capturing content and structural features.

**Baselines.** Since there is no prior work on edge anomaly detection in multiplex networks, we first compare \textsc{AnoMuly} with single-layer edge anomaly detection methods: GOutlier [47] builds a generative model for edges in a node cluster. CM-SKetch [26] uses a Count-Min sketch for approximating global and local structural properties. NetWalk [20] uses a random walk to learn a unified embedding for each node and then dynamically clusters the nodes’ embeddings. AddGraph [19] 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 [84] and MNE [90]. We apply K-means clustering on their obtained node embeddings for anomaly detection [20].

**Results on Multiplex Networks.** We compare \textsc{AnoMuly} with the baseline methods on both dynamic and static multiplex networks with different percentages of anomalous edges (i.e., 1%, 5%). Table 2 reports the AUC for both the baselines and \textsc{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 \textsc{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) \textsc{AnoMuly} outperforms single-layer methods due to its attention mechanism that incorporates complementary information from different relation types. (3) \textsc{AnoMuly} outperforms multiplex methods as it is an end-to-end method and is optimized for anomaly detection. It is also a dynamic method and can take advantage of the temporal properties of the network.

**Results on Single-layer Dynamic Networks.** We also compare \textsc{AnoMuly} with single-layer method baselines on single-layer dynamic datasets. Table 3 summarizes the results. Once again, we see that \textsc{AnoMuly} outperforms all the baselines, even in single-layer graphs, by 2.46% on average. This is mainly due to \textit{Snapshot Encoder}’s architecture, which enables our method to incorporate the outputs of GNN layers after each layer and recurrently update them over time using a GRU cell.

**Ablation Study.** Next, we conduct experiments to show that the \textsc{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 summarizes the results. We found that both our attention mechanism and the GRU cells are important for \textsc{AnoMuly}, producing significant performance boosts.
Additional experimental results on the effect of the training ratio can be found in Appendix E.

**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 2. We find that the top-4 local optimums all coincide with major events annotated in the figure.

**Case Study of Brain Networks.** Behavioral disturbances in ADHD are thought to be caused by the dysfunction of spatially distributed, interconnected neural systems [107]. We used ANOMULY to detect anomalous connections in the brain network (BN) of people with ADHD. Anomalous functional correlations between BN of people with ADHD compared to those without can help us understand which brain regions are involved in ADHD. Our dataset [108] is 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 TD) with the same methodology used by Lanciano et al. [30]. Here, each layer (relation type) is the BN of an individual person, where nodes are brain regions, and each edge measures the statistical association between the functionality of its endpoints. We present two results: (1) 74% of all detected anomalies are edges in the BNs of people in 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 3 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 [32, 109]. More results and visualizations are reported in Appendix E.

5 Conclusion, Limitations, and Future Work

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 employs 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 a wide array of applications and domains. The success raises many interesting directions for future studies: (1) ANOMULY shows the potential to detect anomalous connections in human brains, which can help predict brain diseases or disorders. One future direction is to design an end-to-end model based on the ANOMULY architecture and directly optimize it to predict the presence of disease. (2) ANOMULY assigns the same importance to all snapshots, while earlier information might have less impact and importance than more recent information.

Acknowledgments and Disclosure of Funding

This research was enabled in part by support provided by WestGrid (https://www.westgrid.ca) and The Digital Research Alliance (https://alliancecan.ca/en). We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).
References


13


[88] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. Unsupervised attributed multiplex network embedding. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 5371–5378, 2020.


[99] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly
learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, 3rd International
Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015,

[100] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by
translating on hyperplanes. In Proceedings of the AAAI conference on artificial intelligence,

[101] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana
Yakhnenko. Translating embeddings for modeling multi-relational data. Advances in neural
information processing systems, 26, 2013.


https://doi.org/10.1145/2854006.2854013

[104] Obaida Hanteer, Luca Rossi, Davide Vega D’Aurelio, and Matteo 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,

[105] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrah-

[106] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion
trends with one-class collaborative filtering. In proceedings of the 25th international conference

[107] Kristi R. Griffiths, Taylor A. Braund, Michael R. Kohn, Simon Clarke, Leanne M. Williams,
and Mayuresh S. Korgaonkar. Structural brain network topology underpinning adhd and
ISSN 2158-3188. doi: 10.1038/s41398-021-01278-x. URL https://doi.org/10.1038/s41398-
021-01278-x

[108] Jesse A Brown, Jeffrey D Rudie, Anita Bandrowski, John D Van Horn, and Susan Y
Bookheimer. The ucla multimodal connectivity database: a web-based platform for brain

differences in boys with attention-deficit/hyperactivity disorder with the use of deformation-

[110] Biao Jie, Mingxia Liu, Xi Jiang, and Daoqiang 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.

[111] Chong-Yaw Wee, Pew-Thian Yap, Wenbin Li, Kevin Denny, Jeffrey N Browndyke, Guy G
Potter, Kathleen A Welsh-Bohmer, LiHong Wang, and Dinggang Shen. Enriched white

[112] Xuan Kan, Hejie Cui, Ying Guo, and Carl Yang. Effective and interpretable fmri analysis via


[120] The dblp team. dblp computer science bibliography. [https://dblp.uni-trier.de](https://dblp.uni-trier.de) 2014.


A Additional Related Work

Novelty of the Snapshot Encoder Architecture. In the literature, several studies combine GNNs with recurrent models such as GRU cells or Transformers. Here, we elaborate on how the Snapshot Encoder’s architecture is quite different from this prior work. As discussed in section 2, we can categorize previous work on combining GRU and GNNs into two groups. The first group replaces RNN’s linear layer with a graph convolution layer [72, 74, 75]. The second group uses a GNN as a feature encoder and then deploys a sequence model on top of the GNN to encode temporal properties [67, 76, 77], which ignores the evolution of lower-level node embeddings. Also, these models are limited in their training strategy [27]. ROLAND [27] has addressed these limitations, but it is limited to single-layer graphs. Moreover, natural attempts to use multiplex graph neural networks [78, 79, 80, 84, 85, 86] in the ROLAND framework (e.g., replacing the GNN block with a multiplex GNN or GCN) lead to ignoring historical data in other relation types (layers). For example, assume that we want to use our attention mechanism (see section 3) in the ROLAND framework. Then, we need to use the attention mechanism in the GNN block, which means we incorporate the information about different relation types before embedding the Embedding update module. Accordingly, for each timestamp, we do not incorporate historical data on other relation types, which could produce undesirable performance.

Feature Learning and Anomaly Detection in Brain Networks. In recent years, several studies focused on analyzing brain networks to understand and distinguish healthy and diseased human brains [110, 111, 112]. 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 [113, 114, 115, 116, 117]. In addition to predicting disease in brain networks, understanding the cause of the disease is important. To this end, several anomaly detection methods have been proposed to find anomalous connections, regions, or subgraphs in the brain, which can cause a disease [32, 118, 119]. However, all these anomaly detection models apply to single-layer networks only and do not naturally extend to multiplex networks, while a brain network generated from an individual can be noisy and incomplete [30, 31]. To the best of our knowledge, ANOMULY is the first method for detecting anomalous connections in multiplex brain networks.

Anomaly Detection in Blockchain Networks. Anomaly detection in blockchain transaction networks has recently attracted enormous attention [35, 36, 37, 38, 39, 40], due to the emergence of a huge assortment of financial systems’ applications [41, 42, 43]. However, most existing work focuses on detecting illicit activity in a single blockchain network, while recent research shows that cryptocurrency criminals increasingly employ cross-cryptocurrency trades to hide their identity [44, 45]. Accordingly, Yousaf et al. [46] has recently shown that analyzing links across several blockchain networks is critical for identifying emerging criminal activity on the blockchain. To this end, Ofori-Boateng et al. [45] developed a new persistence summary and utilized it to detect events in dynamic multiplex blockchain networks. For additional related work on single blockchain transaction networks, we refer the reader to the extensive survey by Hassan et al. [40]. To the best of our knowledge, ANOMULY is the first method for detecting anomalous transactions in multiple blockchain networks.

B ANOMULY Algorithm

ANOMULY’s algorithm appears in detail in Algorithm 1.

C Datasets

We use nine real-world public datasets [31, 45, 103, 104, 105, 106] whose domains cover social, co-authorship, blockchain, and co-purchasing networks. We summarize their statistics in Table 1.

Social Networks. RM [103] 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 [104] is collected during the month leading to the 2015 Danish parliamentary election on Twitter. Nodes are Twitter accounts of Danish politicians, and relations are “Retweet”, “Reply”, and “Topical Interaction” [104].
Algorithm 1 ANOMULY Algorithm

Input: A multiplex dynamic network \( G = \{G^{(t)}\}_{t=1}^{T} \)
Output: Node embeddings for all relation types \( \{H^{(t)}_r\}_{r=1}^C \)

1: Initialize \( \{\{H^{(0)(t)}_r\}_{r=1}^C\}_{t=1}^T \); 
2: while not converged do 
3: for \( t = 1, \ldots, T \) do 
4: for \( r = 1, \ldots, L \) do 
5: Let \( \mathcal{L}^{(t)}_r = 0 \); 
6: for \( \ell = 1, \ldots, L \) do 
7: \( H^{(t)(\ell)}_r = \text{GNN}_r \left( H^{(t-1)}_r \right) \); 
8: \( \hat{H}^{(t)(\ell)}_r = \text{GRU}_r \left( \hat{H}^{(t)(\ell)}_r, H^{(t-1)(\ell)}_r \right) \); 
9: \( Z^{(t)(\ell)}_u = \{z^{(t)(\ell)}_u\}_{u \in V} = \{\sum_{r=1}^{L} \phi^{(t)(\ell)}_{u} \hat{h}^{(t)(\ell)}_{u} \}_{u \in V} \); 
10: \( H^{(t)(\ell)}_r = \text{AGG}_r \left( \hat{H}^{(t)(\ell)}_r, Z^{(t)(\ell)}_u \right) \); 
11: for \((u, v) \in \mathcal{E}^{(t)}_r \) do 
12: Sample \((u', v')\) for \( \phi^{(t)}_{v'}(u, v) \); 
13: \( \mathcal{L}^{(t)}_r = \mathcal{L}^{(t)}_r + \max \left\{ 0, \gamma + \phi^{(t)}_{v'}(u, v) - \phi^{(t)}_{v}(u', v') \right\} \); 
14: \( \mathcal{L}^{(t)}_r = \mathcal{L}^{(t)}_r + \lambda \mathcal{L}^{(t)}_r^{reg} \); 
15: \( \mathcal{L}^{(t)}_r = \frac{1}{T} \sum_{r=1}^{\mathcal{C}} \mathcal{L}^{(t)}_r \); 
16: return \( \{H^{(T)}_r\}_{r=1}^C \)

Co-purchasing Network. Amazon [106] 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.

Collaboration Network. DBLP is a co-authorship network until 2014 from [120]. In this dataset, each node is a researcher, an edge shows collaboration, 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 LDA topic modeling [121] to automatically identify 240 topics. We then cluster their non-zero elements into ten known research topics. Each layer in the network represents connections in one of the ten topics.

Blockchain Networks. Ethereum [45] is a blockchain transaction network over 576 days, where layers 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 [45] 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 layers (relation types) correspond to the five most issued fiat currencies on the Ripple network: JPY, USD, EUR, CCK, and CNY.

Brain Network. We use the ADHD-Brain dataset [108] 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 TD) using the same methodology used by Lanciano et al. [30]. Here, each layer (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.

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 [105].
Note that all of the datasets are anonymized and do not contain any personally identifiable information or offensive content.

C.1 Inject Anomalous Edges in Multiplex Networks

Since the ground truth for anomaly detection is difficult to obtain [3], we follow the methodology used in existing studies [3, 19, 20] and inject anomalous edges into our datasets. However, existing methods only inject anomalous edges in a single-layer network and cannot add complex anomalies in multiplex networks. Accordingly, here we divide injected edges into two groups (50% each), (1) layer-independent anomalies, and (2) layer-dependent anomalies. For the first group, we use existing methods [3]. In multiplex networks, the rich information about node connections leads to repetitions, meaning edges between the same pair of nodes repeatedly appear in multiple layers. Repeated connections are more likely to be a strong tie and it may even suggest that its nodes belong to the same community [31, 122]. 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 \((u, v) \notin E_k\) 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.

D Experimental Configuration

In the architecture of Snapshot Encoder we use 200 hidden dimensions for node states, GNN 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) numbers of layers (1 layer to 5 layers); (ii) learning rate (0.001 to 0.01); and (iii) the margin \(\gamma\) (0.3 to 0.7). The values of other hyper-parameters are reported in Table 5.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Multiplex Networks</th>
<th>Single-layer Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKPol</td>
<td>RM</td>
<td>Amazon</td>
</tr>
<tr>
<td>η</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>μ</td>
<td>0.3</td>
<td>0.25</td>
</tr>
<tr>
<td>λ</td>
<td>(5 \times 10^{-7})</td>
<td>(5 \times 10^{-7})</td>
</tr>
<tr>
<td>RM</td>
<td>DBLP</td>
<td>Ethereum</td>
</tr>
<tr>
<td>η</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>μ</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>λ</td>
<td>(5 \times 10^{-7})</td>
<td>(5 \times 10^{-7})</td>
</tr>
<tr>
<td>Amazon</td>
<td>Ripple</td>
<td>Bitcoin</td>
</tr>
<tr>
<td>η</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>μ</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>λ</td>
<td>(5 \times 10^{-7})</td>
<td>(5 \times 10^{-7})</td>
</tr>
<tr>
<td>DBLP</td>
<td>Amazon-S</td>
<td>DBLP-S</td>
</tr>
<tr>
<td>η</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>μ</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>λ</td>
<td>(5 \times 10^{-7})</td>
<td>(5 \times 10^{-7})</td>
</tr>
</tbody>
</table>

E Additional Experimental Results

E.1 Parameter Sensitivity

We evaluate the effect of the training ratio of the dataset: we change the training ratio from 60% to 20% and report the results on the Ripple dataset in Figure 4. 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.
E.2 Additional Results on Brain Networks

In this experiment, we investigate how anomalous edges found by ANOMULY are distributed in the brain. Figure 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.

F Reproducibility

The code, datasets, and supplements are available at https://github.com/ubc-systopia/ANOMULY.