People's Democratic Republic of Algeria

University of Abou Bekr Belkaid Tlemcen

Faculty of Sciences

Department of Computer Science

MASTER'S THESIS

to obtain the diploma of Master in
Intelligent Models and Decision (MID), Computer Science

A New Pattern-Based Geometric Framework for Anomaly Detection in Static Graphs

---

Completed by
Ahmed Youcef BENHALIMA

Presented on 26/06/2024 before the jury composed of:

President:
Fethallah HADJILA

Examiner:
Sidahmed BERRABAH

Invited:
Mohammed Salih BENDELLA

Supervisor:
Houcine MATALLAH

Co-Supervisor 1:
Amin MESMOUDI

Co-Supervisor 2:
Seif Eddine BENKABOU

Acknowledgments

Praise be to **Allah**, we seek His help. Whomsoever **Allah** guides will never be led astray, and whomsoever **Allah** leaves astray, no one can guide. I bear witness that there is none worthy of worship but **Allah** and I bear witness that Muhammad is His slave and Messenger.

I would like to express my sincere gratitude to everyone who has supported me throughout the completion of this master's thesis.

First and foremost, I am deeply thankful to **my Family**, **my Mother** and **my Father** for their unwavering support, understanding, and encouragement during this academic pursuit. May **Allah** bless them.

I am grateful to my thesis advisor, Dr. Amin Mesmoudi and Seif-Eddine Benkabou, for their invaluable guidance, encouragement, and insightful feedback throughout this research journey. Their expertise and dedication have been instrumental in shaping this thesis.

I would also like to thank the members of my thesis committee, for their constructive comments and suggestions.

I am deeply thankful to Abou Bekr Belkaid University of Tlemcen and all its teachers for their continuous support, for imparting knowledge, and for fostering an environment conducive to academic growth and research excellence.

I extend my appreciation to LIAS laboratory for providing the necessary resources and facilities for conducting this research.

Lastly, I am grateful to all my friends and colleagues who have provided encouragement and assistance along the way.

May **Allah** bring peace, security, and relief to the **people of Gaza**, and may He alleviate their suffering and grant them strength during these challenging times.

BENHALIMA Ahmed Youcef

Contents

Chapter 1

Introduction

Anomaly detection is a critical aspect of data analysis, focusing on identifying observations that deviate significantly from the established norm within a dataset. These anomalies, often indicative of errors, fraud, or novel phenomena, can provide valuable insights and prompt timely interventions across various fields. [Ruff et al., 2021]

The significance of anomaly detection spans numerous domains, including finance, healthcare, cybersecurity, and industrial systems. In finance, detecting anomalies can help identify fraudulent transactions or market manipulation, protecting assets and maintaining trust. In healthcare, anomaly detection is vital for early disease diagnosis and monitoring patient health [Karadayi et al., 2020] [Liu et al., 2022a], potentially saving lives by catching issues before they become critical. Cybersecurity benefits from anomaly detection [Evangelou and Adams, 2020] by identifying potential breaches or unusual activity that could signify an attack, thereby safeguarding sensitive information and infrastructure.

The evolution of machine learning and deep learning techniques has significantly enhanced the capabilities of anomaly detection systems. [Kwon et al., 2019] Advanced models such as autoencoders [Chen et al., 2018], generative adversarial networks (GANs) [Sabuhi et al., 2021], and recurrent neural networks (RNNs) [Salehinejad et al., 2017] have been developed to improve the accuracy and efficiency of detecting anomalies in complex and high-dimensional data.[Talagala et al., 2021]

For example, deep learning approaches have been successfully applied to various applications such as corrosion detection [Tan et al., 2016] in structural health monitoring and identifying faults in aerospace systems. These advanced techniques allow for more precise and timely detection of anomalies, which is essential for maintaining the integrity and performance of critical systems.

Overall, the importance of anomaly detection lies in its ability to uncover hidden patterns and irregularities that could indicate significant issues or opportunities. By leveraging sophisticated algorithms and models, anomaly detection helps organizations make informed decisions, enhance security, and improve operational efficiency.

1.1 Complexity of Anomaly Detection

Anomaly detection is inherently complex due to the rarity and diversity of anomalies. Anomalies are typically rare events, making them difficult to identify against a backdrop of normal behavior. This rarity presents a significant challenge in both training models and ensuring they have sufficient sensitivity to detect unusual patterns without generating excessive false positives.

1.1.1 Complexity of data

The complexity of the data itself adds another layer of difficulty. In many real-world applications, data is not only high-dimensional but also structured in complex forms such as graphs and time series.

Graph data

Graphs represent relationships between entities and are used in various domains like social networks, biological networks, and communication systems. Anomalies in graph data can manifest as unexpected structural patterns, such as unusual connectivity or subgraph structures, which are not easily detected using traditional techniques. The interconnected nature of graph data requires sophisticated methods to accurately capture and analyze these anomalies.

Time series data

Time series data, prevalent in fields like finance, healthcare, and IoT, involves data points indexed in time order. Anomalies in time series can be abrupt changes, unexpected spikes, or deviations from established patterns. The temporal dependency and potential non-stationarity of time series data necessitate specialized techniques to effectively identify anomalies.

1.1.2 Existing techniques

Numerous techniques have been developed to tackle anomaly detection, each with its strengths and limitations:

- **Statistical methods**: These include methods like Z-score, Grubbs' test, and the Mahalanobis distance, which assume certain statistical properties of the data. While effective for simpler datasets, they often fall short in handling high-dimensional and complex data structures.
- **Machine learning methods**: Unsupervised techniques like clustering (e.g., k-means [Ahmed et al., 2020], DBSCAN [Ester et al., 1996]) and dimensionality reduction (e.g., PCA [Maćkiewicz and Ratajczak, 1993], t-SNE [Van der Maaten and Hinton, 2008]) can help identify outliers. Supervised methods, including classification algorithms (e.g., SVM, [Hearst et al., 1998] Random Forest), require labeled data, which is often scarce for anomalies.
- **Deep-learning methods**: Advanced models such as autoencoders, recurrent neural networks (RNNs) [Grossberg, 2013], and generative adversarial networks (GANs) [Goodfellow et al., 2020] have shown promise in capturing complex patterns and dependencies in data. These models can learn intricate representations of normal behavior and detect subtle deviations.

1.1.3 New challenges

Despite advancements, several challenges remain in the field of anomaly detection:

- **Anomaly definition and labeling**: The ambiguous nature of what constitutes an anomaly can vary across applications, making it difficult to define and label anomalies consistently. This ambiguity complicates the training and evaluation of anomaly detection models.

- **Interpretability**: Complex models, especially deep learning-based approaches, often act as black boxes, making it difficult to interpret why a particular data point is classified as an anomaly. Developing methods that provide clear and actionable insights is essential.

- **Scalability**: As data volumes grow, the scalability of anomaly detection algorithms becomes critical. Ensuring that methods can handle large datasets efficiently without compromising accuracy is an ongoing challenge.

1.2 Contribution

In this thesis, we first focus on benchmarking unsupervised deep-learning methods for detecting anomalous nodes in static attributed graphs. Our second contribution is the development of the Pattern-based Geometric Framework for Anomaly Detection (PGF-AD), a topological theory that introduces a comprehensive approach to anomaly detection.

Our methodology addresses the ambiguity in anomaly definitions by considering both the individual properties of patterns and their context within the dataset. Unlike approaches assuming predefined anomaly distributions, PGF-AD offers an intuitive and interpretable framework for distinguishing between normal and anomalous patterns.

Furthermore, we introduce Minomaly, an efficient deep learning method based on PGF-AD, designed to detect unexpected structural connections in graphs. Minomaly demonstrates superior performance in empirical evaluations, validating the efficacy of our proposed anomaly detection framework.

1.3 Work Context

This master's thesis is pursued at Abou Bekr Belkaid University of Tlemcen[1] under the specialization of Intelligent Models and Decision (MID). It is complemented by an internship at the LIAS - Laboratory of Computer Science and Automatic Control for Systems[2], located at ISAE-ENSMA on the Futuroscope site in France.

The internship started on February 14, 2024, and will conclude on July 19, 2024. During this period, I have integrated the IDD (Data and Model Engineering) team at LIAS. I am working under the supervision of Dr. Amin Mesmoudi and Seif-Eddine Benkabou, both associate professors in the IDD team.

The thesis endeavors to develop an evaluation and benchmarking framework specifically tailored for deep learning-based anomaly detection methods applied to graph data. This framework aims to systematically compare the performance of diverse approaches, offering comprehensive insights into their effectiveness and practical applicability in anomaly detection contexts.

[1] https://univ-tlemcen.dz/
[2] https://www.lias-lab.fr/

1.4 Organization of the Manuscript

The rest of the manuscript is organized as follows: Chapter 2 provides an overview of the fundamental concepts and surveys the existing literature relevant to anomaly detection, with a focus on techniques applied to graph data. We present our benchmarking process for various unsupervised deep-learning methods aimed at detecting anomalous nodes in static attributed graphs in Chapter 3. In Chapter 4, we introduce Minomaly, our proposed deep learning method designed to detect unexpected structural connections in graphs. This chapter discusses the theoretical foundation, implementation details, and empirical evaluations demonstrating the method's efficacy. Finally, in the concluding chapter, we summarize the main findings of the thesis and provide some directions for future work.

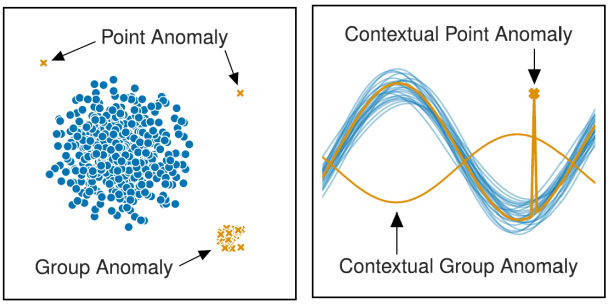
Chapter 2

Background and Related Works

Introduction Anomaly detection in graph data is a rapidly evolving field, driven by the increasing complexity and scale of data generated in various domains such as social networks, biological systems, and cybersecurity. This chapter provides an in-depth review of the fundamental concepts, types of anomalies, and the existing literature relevant to anomaly detection, with a particular focus on techniques applied to graph data. By understanding the background and related works, we can better appreciate the challenges and advancements in this field.

2.1 Anomaly

An anomaly is an observation that significantly deviates from the expected norm within a dataset [Ruff et al., 2021]. Detecting these anomalies is crucial because they often indicate rare but important events.



(a) Point & group anomaly (b) Contextual anomaly

Figure 2.1: Primary anomaly types [1]

Anomalies can be classified into three primary types:

- **Point anomalies**: Single data points that stand out as unusual compared to the rest of the data.
- **Contextual anomalies**: Data points that are unusual within a specific context.

- **Collective anomalies**: Groups of related data points that collectively are unusual, even if individual points may not be.

Furthermore, anomalies can be categorized as either low-level or high-level (semantic) anomalies. For example:

- A low-level anomaly might be a noisy image in the MNIST dataset, where pixel values are corrupted.
- A high-level (semantic) anomaly might be a sentence where the context or meaning is disrupted by irrelevant words.

Anomaly vs. Outlier vs. Novelty

Anomaly Typically an instance from a different distribution, often generated by a distinct process.

Outlier A rare instance within the normal distribution.

Novelty An instance from a new region or mode within the normal distribution.
For example, in the context of bird distribution:

- A cat represents an anomaly.
- A rare breed of bird is an outlier.
- A new breed of cat is a novelty.

Despite these distinctions, all these types are generally treated as anomalies in practice.

2.2 Anomaly detection

Anomaly detection is the study of identifying anomalous observations using various methods, models, and algorithms based on data [Ruff et al., 2021]. This field is crucial in numerous applications, from detecting fraud in finance to identifying faults in industrial systems.

Unsupervised learning is the predominant approach in anomaly detection for several reasons. Firstly, labeled anomalous data is often unavailable, making it difficult to apply supervised methods. Even when labeling is available, it is usually insufficient to fully characterize all notions of anomalies, thus reducing the effectiveness of supervised techniques. Unsupervised methods, on the other hand, help in learning a comprehensive model of normal behavior, which is essential for detecting deviations that signify anomalies. This approach is particularly important in complex, high-dimensional data types such as time series, biological sequences, images, videos, audio, text, and graphs, where effective representation learning is critical.

[1] [Ruff et al., 2021]

Key aspects of anomaly detection Representation learning is essential for high-dimensional data, as it helps in extracting meaningful patterns from raw inputs. In this context, deep-learning methods are particularly useful as they exploit hierarchical and latent structures within data, enhancing the capability to identify anomalies. Furthermore, feature learning is an automated process to derive meaningful features from raw data, which is critical in improving the accuracy of anomaly detection. Given the unsupervised nature of most anomaly detection tasks, it is often unclear which representation learning techniques are most useful, requiring sophisticated methods to determine the best approach.

Major approaches in deep-learning for anomaly detection Several major approaches have been developed for deep-learning-based anomaly detection. One prominent method is deep one-class classification, which involves learning a representation of normal data to identify deviations. Generative adversarial networks (GANs) [Goodfellow et al., 2020] are also widely used to model data distributions and detect anomalies by identifying data points that do not conform to these distributions. Another popular approach is the use of autoencoders, where the model learns to reconstruct normal data and identifies anomalies based on reconstruction errors. Additionally, self-supervised learning techniques, where the data itself acts as its own label, have been effectively applied to anomaly detection tasks.

Challenges in anomaly detection Anomaly detection faces several challenges. One major challenge is the variability within normal data, which can cause false positives due to large variations, such as individual differences in biological data. This variability requires careful modeling to avoid misclassifications. Another challenge is the dispersion of normal data, where normal data might be more spread out than anomalous samples. Techniques like preprocessing, normalization, and feature selection can help mitigate this issue. Furthermore, dominant features with a wide range, noise, or irrelevant data can obscure anomalies by dominating distance calculations. The rarity of anomalous events also poses a significant challenge, as datasets are often unlabeled, necessitating unsupervised learning to model the majority and treat deviations as anomalies. Additionally, the distribution of normal data can change over time due to environmental factors, complicating the detection process. Semantic anomalies, which can be very close to normal instances in the raw feature space, are particularly difficult to identify.

Approaches to anomaly detection Several approaches can be employed in anomaly detection, each assuming a particular view of normality. Classification methods involve learning a model of normal data and identifying deviations. Probabilistic methods model the data distribution and flag data points that have a low probability of occurrence. Reconstruction techniques, such as autoencoders, identify anomalies based on the inability to reconstruct certain data points. Distance-based methods rely on measuring the distance between data points and identifying those that are far from the majority.

For example, Deep SVDD [Ruff et al., 2018] learns a neural feature map that extracts multiple scales of data to fit a hypersphere model in feature space, effectively identifying anomalies based on their distance from the center of the sphere. Reconstruction methods, which do not typically follow a probabilistic approach, flag a point as anomalous if it does not fit its learned 'idealized' representation. Additionally, various regularizations in autoencoders have been introduced to achieve more robust latent representations, reflecting specific assumptions of a given anomaly detection task.

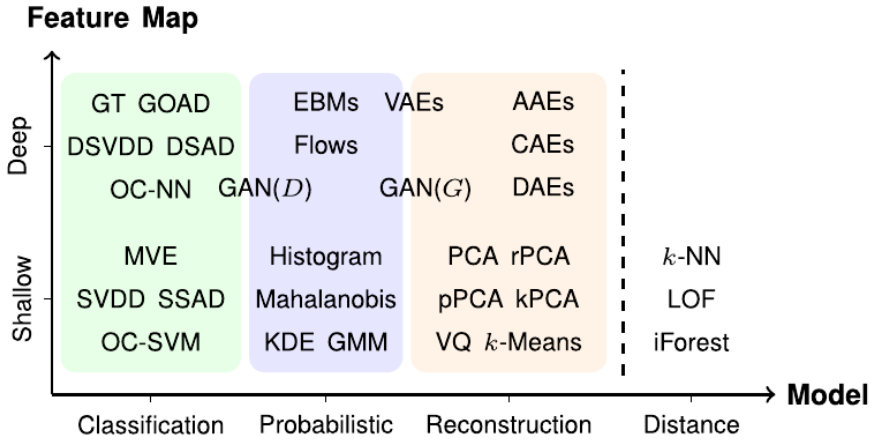


Figure 2.2: Anomaly detection approaches from the proposed unified view in [Ruff et al., 2021]

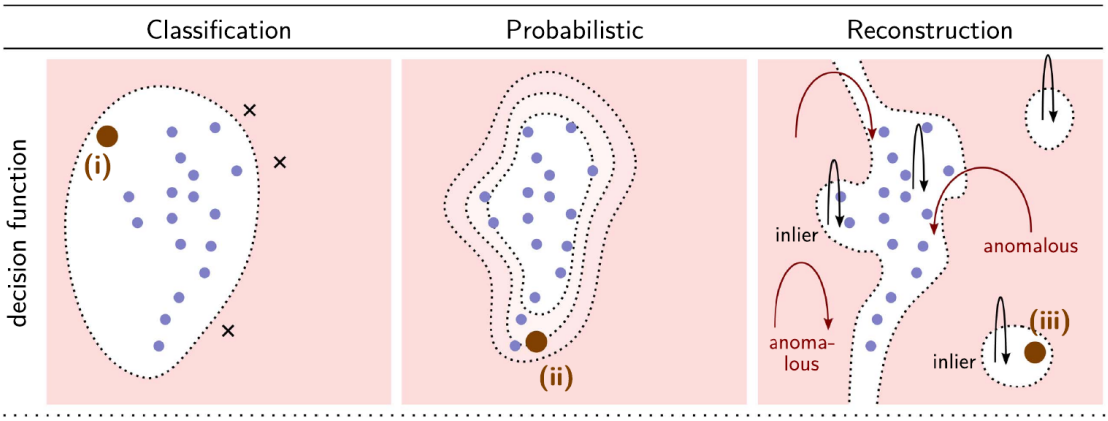


Figure 2.3: Challenges of anomaly detection approaches [2]
(i) - A boundary that is too lenient may result in some anomalies not being detected due to its biased nature.
(ii) - Probabilistic models might fail to accurately capture the tails of a distribution, leading to underfitting or overfitting.
(iii) - Artifacts in manifold or prototype structures can produce misleadingly favorable results.

2.3 Graphs

Plain Graph A plain graph $G = \{V, E\}$ is a static graph composed of a set of nodes $V = \{v_i\}_{i=1}^{n}$ and a set of edges $E = \{e_{i,j}\}$. Here, n represents the number of nodes, and an edge $e_{i,j} = (v_i, v_j)$ indicates a connection between nodes v_i and v_j. The adjacency matrix $A = [a_{i,j}]_{n \times n}$ represents the structure of the graph, where $a_{i,j} = 1$ if there is an edge between nodes v_i and v_j, and $a_{i,j} = 0$ otherwise.

[2] [Ruff et al., 2021]

Attributed Graph An attributed graph $G = \{V, E, X\}$ extends the plain graph by including an attribute set X. This graph comprises a set of nodes V, a set of edges E, and an attribute matrix X. The structure follows the same definition as the plain graph. The attribute matrix $X = [\mathbf{x}_i]_{n \times k}$ contains attribute vectors for each node, where $\mathbf{x}_i$ is the attribute vector for node v_i and k is the dimension of these vectors. The terms attribute and feature are used interchangeably in this context.

Dynamic Graph A dynamic graph $G(t) = \{V(t), E(t), X_v(t), X_e(t)\}$ includes nodes and edges that evolve over time. At any given time step t, $V(t)$ denotes the set of nodes, $E(t)$ represents the set of edges, and $X_v(t)$ and $X_e(t)$ are the attribute matrices for nodes and edges, respectively, if they exist.

In practice, nodes or edges might also be assigned numerical or categorical labels to indicate their classes, such as normal or abnormal. When label information is available, whether fully or partially, it enables the effective training of supervised or semi-supervised detection models. [Ma et al., 2023]

Graph data is increasingly essential in various domains due to its ability to represent complex relationships and interactions between entities. Examples of graph data include social networks, where nodes represent individuals and edges represent friendships or interactions; biological networks, where nodes represent genes or proteins and edges represent their interactions; transportation networks, where nodes represent locations and edges represent routes; and knowledge graphs, where nodes represent concepts and edges represent relationships between them.

The complexity and high dimensionality of graph data arise from several factors:

- **Structural Complexity:** Graphs can represent intricate structures with varying degrees of connectivity and hierarchical organization. This structural complexity is often difficult to capture using traditional data representations.
- **High Dimensionality:** Graphs may include not only a large number of nodes and edges but also rich attribute information for both nodes and edges. This results in a high-dimensional feature space that poses significant challenges for data analysis and machine learning.
- **Dynamic Nature:** Many graphs, such as social networks and communication networks, are dynamic and evolve over time. Capturing the temporal dynamics and changes in the structure and attributes of these graphs adds an additional layer of complexity.
- **Interdependence:** The entities in a graph are often interdependent, meaning that the presence or characteristics of one entity can influence others. This interdependence must be considered in any analysis or learning task.

Due to these challenges, traditional machine learning methods that work well on Euclidean data struggle to perform effectively on graph data. This has led to the development of specialized techniques for graph analysis, among which Graph Neural Networks (GNNs) have emerged as a powerful tool.

2.3.1 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) [Wu et al., 2020] are designed to operate directly on the graph structure, leveraging the connectivity and attribute information to learn meaningful representations of nodes, edges, or entire graphs. GNNs have gained popularity due to their ability to generalize the concept of convolution from grid-like data (such as images) to graph data.

Mechanism of GNNs GNNs work by iteratively aggregating information from a node's neighbors to update its representation. This process, often referred to as message passing, allows the network to capture the local and global structure of the graph. The key steps in a typical GNN include:

- **Aggregation:** Nodes aggregate information from their neighbors. This can involve summing, averaging, or applying a more complex function to the neighbors' attributes.
- **Update:** The aggregated information is combined with the node's current representation to produce an updated representation. This step typically involves applying a neural network layer.
- **Propagation:** The updated representations are propagated through the network, allowing the model to learn higher-level features.

Applications of GNNs GNNs have been successfully applied to a variety of tasks, including:

- **Node Classification:** Predicting the label of a node based on its attributes and the structure of the graph.
- **Link Prediction:** Predicting the existence of an edge between two nodes, which is useful in recommendation systems and social network analysis.
- **Graph Classification:** Classifying entire graphs, which is applicable in chemistry for molecule classification and in bioinformatics for predicting protein functions.
- **Anomaly Detection:** Identifying unusual patterns or outliers in graph data, which is crucial for fraud detection, network security, and fault detection in industrial systems.

Challenges in GNNs Despite their success, GNNs face several challenges:

- **Scalability:** Large-scale graphs with millions or billions of nodes and edges can be computationally expensive to process.
- **Dynamic Graphs:** Capturing the temporal dynamics of evolving graphs requires extensions to standard GNN architectures.
- **Interpretability:** Understanding the decisions made by GNNs can be difficult due to their complex structure.
- **Over-smoothing:** When too many layers are used, node representations can become too similar, leading to a loss of useful information.

2.3.2 Graph Convolutional Networks (GCNs)

Graph Convolutional Networks (GCNs) [Zhang et al., 2019] are a type of Graph Neural Network designed to generalize the concept of convolution from traditional grid-like data (such as images) to graph-structured data. GCNs leverage the connectivity patterns and attributes of nodes to perform various learning tasks on graphs.

Mechanism of GCNs GCNs operate by aggregating information from a node's local neighborhood in the graph. This process, inspired by the convolution operation in image processing, enables GCNs to capture the local structural information and node features. The key components of a GCN layer are:

- **Aggregation:** Each node aggregates feature information from its immediate neighbors. This is typically done using a weighted sum or average of the neighbors' features.

- **Transformation:** The aggregated features are then transformed using a learnable weight matrix. This step can be seen as applying a linear transformation to the aggregated features.

- **Activation:** A non-linear activation function (such as ReLU) is applied to introduce non-linearity into the model.

2.3.3 Graph Attention Networks (GATs)

Graph Attention Networks (GATs) [Veličković et al., 2017] extend the Graph Neural Network (GNN) framework by incorporating attention mechanisms to improve the representation of graph-structured data. GATs allow for the assignment of different importance scores to different nodes in a neighborhood, enhancing the ability to focus on relevant parts of the graph.

Mechanism of GATs GATs leverage attention mechanisms to weigh the influence of neighboring nodes when aggregating information. This mechanism allows the model to learn which neighbors are more important for a particular task. The key components of a GAT layer are:

- **Attention Coefficients:** For each node v_i, attention coefficients α_{ij} are computed for all its neighboring nodes v_j. These coefficients indicate the importance of the neighboring node v_j to v_i.

- **Aggregation:** The node features are aggregated using the computed attention coefficients. Each node's feature vector is updated as a weighted sum of its neighbors' feature vectors, where the weights are the attention coefficients.

- **Multi-Head Attention:** To stabilize the learning process, GATs often employ multi-head attention, where multiple attention mechanisms are applied in parallel, and their results are concatenated or averaged.

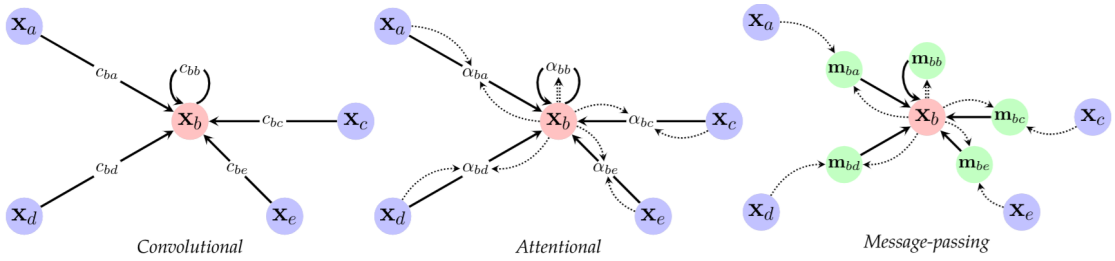


Figure 2.4: Visualization of data flow in three types of GNN layers using the neighborhood of node b. From left to right: convolutional, where sender node features are multiplied by a constant c_{uv}; attentional, where the multiplier is implicitly computed via an attention mechanism $\alpha_{uv} = a(\mathbf{x}_u, \mathbf{x}_v)$, which considers the receiver's attention over the sender; and message-passing, where vector-based messages $\mathbf{m}_{uv} = \psi(\mathbf{x}_u, \mathbf{x}_v)$ are computed based on features of both the sender and receiver nodes.[3]

2.4 Graph anomaly detection

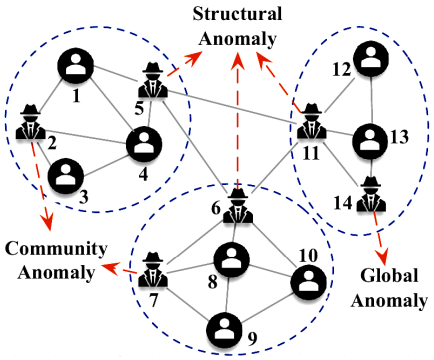
The importance of graph anomaly detection lies in its ability to uncover hidden insights and potential risks that may otherwise go unnoticed. In cybersecurity, for example, detecting anomalous network traffic patterns can help identify potential intrusions or malicious activities. In social network analysis, identifying anomalous behaviors can aid in detecting fake accounts, spam, or unusual communication patterns. In biological networks, anomaly detection can highlight proteins or genes with unusual interactions that may indicate disease mechanisms or novel biological processes.

According to the survey [Ma et al., 2023], the state-of-the-art graph anomaly detection can be categorized based on the type of anomalies being detected. The primary types include anomalous node detection, anomalous edge detection, anomalous sub-graph detection, and anomalous graph detection. Each type can be further subdivided based on the methodologies employed and the nature of the graphs (e.g., static vs. dynamic graphs).

Anomalous nodes There are three types of anomalous nodes:

- **Structural node anomalies:** nodes that exhibit abnormal patterns in their connections.
- **Community node anomalies:** nodes that show significant deviations in their attribute values compared to other nodes within the same community
- **Global node anomalies:** nodes whose attributes are significantly different from all other nodes in the graph.

[3] [Bronstein et al., 2021]



Community	Node	Node Features			
Community 1	1	1	0	0	0
	2	1	1	0	0
	3	1	0	0	0
	4	1	0	0	0
	5	1	0	0	0
Community 2	6	0	1	0	0
	7	0	1	1	0
	8	0	1	0	0
	9	0	1	0	0
	10	0	1	0	0
Community 3	11	0	0	1	0
	12	0	0	1	0
	13	0	0	1	0
	14	0	0	1	1

Figure 2.5: Node anomaly types[4]

Anomalous Node Detection (ANOS ND)

On Plain Graphs

- Traditional Non-Deep Learning Techniques: [Akoglu et al., 2010, Ding et al., 2012, Hooi et al., 2016]
- Network Representation-Based Techniques: [Hu et al., 2016]
- Reinforcement Learning-Based Techniques: [Morales et al., 2021]

On Attributed Graphs

- Deep Neural Network (DNN)-Based Techniques: [Bandyopadhyay et al., 2020]
- Graph Convolutional Network (GCN)-Based Techniques: [Kipf and Welling, 2017], [Ding et al., 2019a], [Wang et al., 2019], [Zhang et al., 2020]
- Reinforcement Learning-Based Techniques: [Ding et al., 2019b]

On Dynamic Graphs

- Network Representation-Based Techniques: [Yu et al., 2018]
- Generative Adversarial Network (GAN)-Based Techniques: [Zheng et al., 2019b]

Anomalous Edge Detection (ANOS ED)

- Deep Neural Network (DNN)-Based Techniques: [Ouyang et al., 2020]
- Graph Convolutional Network (GCN)-Based Techniques: [Duan et al., 2020]

On Dynamic Graphs

- Network Representation-Based Techniques: [Yu et al., 2018]
- GCN-Based Techniques: [Zheng et al., 2019a]

[4] [Ma et al., 2023]

Anomalous Sub-Graph Detection (ANOS SGD) Techniques for detecting anomalous sub-graphs often focus on residual analysis between expected and observed structures. Traditional metrics and deep learning approaches are employed to identify communities or dense sub-graphs that deviate significantly from the norm.

Anomalous Graph Detection (ANOS GD)

- Graph Neural Network (GNN)-Based Techniques: [Dou et al., 2021]

On Dynamic Graphs Techniques in this category aim to detect entire graphs that are anomalous compared to a set of graphs. These techniques often involve learning representations of graphs and comparing them to identify outliers.

2.5 Anomalous node detection deep-learning methods on static graphs

2.5.1 GCNAE: Variational graph auto-encoders

GCNAE [Kipf and Welling, 2016] employs an autoencoder framework where Graph Convolutional Networks (GCNs) serve as both the encoder and decoder. The model processes the graph structure and node attributes as input. The encoder aggregates neighbor information to learn node embeddings, while the decoder reconstructs node attributes by applying another GCN to these embeddings and graph structures. The outlier score for a node is derived from the reconstruction error of the decoder.

2.5.2 DOMINANT: Deep anomaly detection on attributed networks

DOMINANT [Ding et al., 2019a] is an early work integrating GCN and autoencoders for outlier node detection. It utilizes a two-layer GCN for the encoder and another two-layer GCN for the decoder to reconstruct node attributes. Additionally, it employs a one-layer GCN with a dot product as the structural decoder to reconstruct the graph adjacency matrix. The reconstruction errors from both decoders are combined to compute the outlier scores of the nodes.

$$\text{score}(i) = (1-\alpha)\left\|\mathbf{a}_i - \hat{\mathbf{a}}_i\right\|_2 + \alpha\left\|\mathbf{x}_i - \hat{\mathbf{x}}_i\right\|_2$$

2.5.3 DONE & ADONE: Outlier resistant unsupervised deep architectures for attributed network embedding

DONE [Bandyopadhyay et al., 2020] employs separate autoencoders for structural and attribute data to reconstruct the adjacency matrix and node attributes, respectively. Both encoders and decoders are composed of Multi-Layer Perceptrons (MLPs). This method simultaneously optimizes node embeddings and outlier scores using a unified loss function. It measures three anomaly scores for each node

- o_i^a i has similar attributes with nodes in different communities
- o_i^s i connects with other communities
- o_i^{com} i belongs to one community structurally, but the attributes follow the pattern of another community

These scores of each node i are extracted from more reconstruction errors because their structure or attribute patterns do not conform to the standard behavior.

AdONE is a variant of DONE, introducing an additional discriminator to differentiate between the learned structure and attribute embeddings of a node. This adversarial training approach aims to better align the two different embeddings in the latent space.

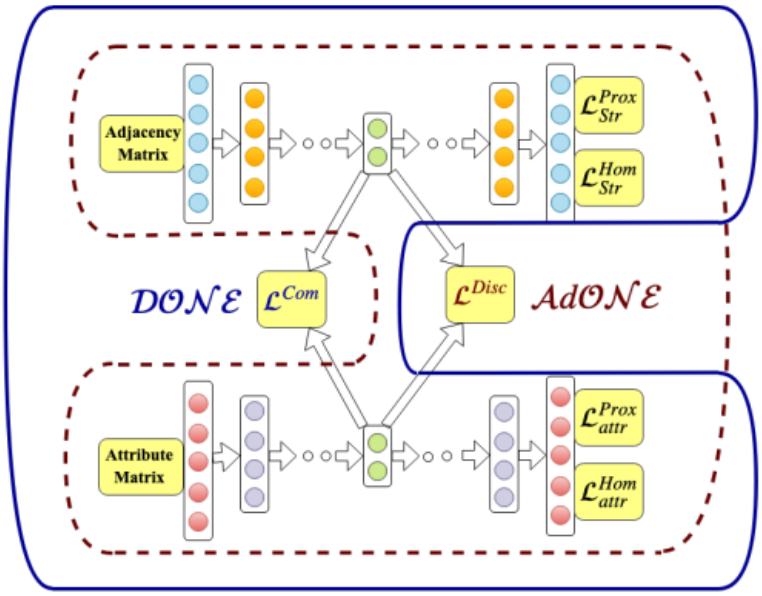


Figure 2.6: DONE & AdONE architecture [5]

2.5.4 ANOMALYDAE: Dual autoencoder for anomaly detection on attributed networks

AnomalyDAE [Fan et al., 2020] leverages both structural and attribute autoencoders for outlier node detection. Its structure encoder processes the adjacency matrix and node attributes, while the attribute decoder reconstructs node attributes using both structure and attribute embeddings.

2.5.5 CONAD: Contrastive attributed network anomaly detection with data augmentation

CONAD [Xu et al., 2022], incorporates graph augmentation and contrastive learning techniques. It generates augmented graphs to impose prior knowledge of outlier nodes. After encoding these graphs using Siamese GNN encoders, a contrastive loss is applied to optimize the encoder. Similar to DOMINANT, the outlier score for a node is obtained using two different decoders.

[5] [Bandyopadhyay et al., 2020]

Conclusion Anomaly detection in graph data is a rapidly advancing field crucial for detecting rare but significant events in domains such as social networks, biology, and cybersecurity. This chapter has provided a comprehensive overview of fundamental concepts, anomaly types, and state-of-the-art techniques in graph anomaly detection. By exploring this background and current literature, we gain insights into the challenges and innovations driving this dynamic field forward.

Chapter 3

Benchmarking Unsupervised Deep-Learning Anomalous Node Methods on Static Attributed Graphs

Introduction This chapter aims to systematically evaluate and benchmark various unsupervised deep-learning methods for anomaly detection in static attributed graphs. By comparing different approaches, we seek to provide comprehensive insights into their performance, strengths, and limitations. The benchmarking process involves rigorous experimental setups, including diverse datasets and a range of performance metrics, to ensure a thorough and fair evaluation.

The evaluation is structured as follows: we begin with a discussion of the statistical classification and confusion matrix, which form the basis for our performance metrics. We then present the datasets used for benchmarking, detailing their characteristics and the process of injecting structural anomalies. Following this, we report the results of our experiments, comparing the performance of various models. Finally, we discuss the implications of our findings and suggest directions for future research in the field of anomaly detection in graph data.

3.1 Evaluation methodology

3.1.1 Statistical classification and confusion matrix

Statistical classification involves identifying the categories of observations based on statistical principles. The confusion matrix visualizes the performance of statistical classification **??**, representing a contingency table of variables that combine dimensions and classes. Here, dimensions are in {'actual', 'predicted'} and include the same sets of classes. Statistically, these dimensions correspond to statistical variables, and the classes denote their respective learning categories.

For two classes c_1 and c_2, the confusion value of a learning algorithm in test is defined as:

$$\text{confusion}(c_1, c_2) = \text{Card}(\{\text{ind} \in \text{test} \mid c_1 = \text{actual}(\text{ind}) \text{ and } c_2 = \text{predict}_{\text{algo}}(\text{ind})\})$$

This denotes the number of test instances of class c_1 predicted as c_2 by the algorithm.

- A test instance ind is classified as a **true positive** if actual(ind) = predict(ind) = positive.
- A test instance ind is classified as a **false positive** if actual(ind) ≠ predict(ind) = negative.

In the context of anomalies, we typically consider two classes:

- **Normal** → negative
- **Anomaly** → positive

Anomaly is considered positive because it is the class of interest. Therefore, we define:

$$\begin{cases} TP = \text{confusion}(\text{positive}, \text{positive}) \\ FP = \text{confusion}(\text{negative}, \text{positive}) \\ TN = \text{confusion}(\text{negative}, \text{negative}) \\ FN = \text{confusion}(\text{positive}, \text{negative}) \end{cases}$$

From these, we derive:

$$\begin{cases} P = TP + FN & \text{(Population of positive cases)} \\ N = FP + TN & \text{(Population of negative cases)} \\ PP = TP + FP & \text{(Predicted positives)} \\ PN = TN + FN & \text{(Predicted negatives)} \end{cases}$$

Population = P + N	**Predicted Positive (PP)**	**Predicted Negative (PN)**
Actual Positive (P)	**True Positives (TP)**	**False Negatives (FN)**
Actual Negative (N)	**False Positives (FP)**	**True Negatives (TN)**

3.1.2 Performance metrics

- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{P + N}$$

  It measures the overall correctness of the model.

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

  It measures the proportion of true positives among the predicted positives. High precision ensures that the detected anomalies are true anomalies by minimizing the noise FP. A low precision would mean that many of the flagged instances are actually normal, leading to unnecessary investigation efforts.

- **Recall** (Sensitivity or True Positive Rate):

$$\text{Recall} = \text{TPR} = \frac{TP}{TP + FN}$$

  It measures the proportion of true positives among the actual positives. High recall ensures that most anomalies are detected by minimizing FN. Missing an anomaly (false negative) can be much more costly than incorrectly flagging a normal instance (false positive).

- **F-measure** (F1 Score):
$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
It is the harmonic mean of precision and recall. Balances recall and precision, providing a single measure of overall performance.

- **Specificity** (True Negative Rate):
$$\text{Specificity} = \frac{TN}{TN + FP}$$
It measures the proportion of true negatives among the actual negatives.

- **False Positive Rate** (FPR):
$$\text{FPR} = \frac{FP}{FP + TN}$$
It measures the proportion of false positives among the actual negatives.

- **Average Precision AUC**:
$$\text{Average Precision AUC} = \int_0^1 \text{Precision}(r)\, dr$$
Here, r represents the Recall (TPR). AP AUC represents the area under the precision-recall curve. It summarizes the precision-recall trade-off.

- **ROC AUC**:
$$\text{ROC AUC} = \int_0^1 \text{TPR}(f)\, df$$
Here, f represents the False Positive Rate (FPR). ROC AUC measures the area under the receiver operating characteristic curve, summarizing the model's ability to distinguish between classes.

ROC AUC and Average Precision AUC provide a summary of the performance over different thresholds, useful for evaluating the robustness of the anomaly detector.

Precision, Recall, F-measure, AP-AUC, ROC-AUC metrics help in understanding the effectiveness of the anomaly detection algorithm. Accuracy and specificity, are less interesting in this context due to the typical imbalance between normal and anomaly classes.

3.2 Datasets

To evaluate deep unsupervised outlier node detection methods on static attributed graphs, we use datasets provided by the tool [Liu et al., 2024] within the benchmark [Liu et al., 2022b], where structural anomalies are injected into Cora, Amazon, and Flickr following the methodology of [Ding et al., 2019a]. We reproduce the results and add recall and precision evaluations, comparing our results to those in [Liu et al., 2022b].

For generating structural outliers, the specific procedure is followed: First, m nodes are selected from the network, and these nodes are then made fully connected to form a clique. All m nodes in this clique are considered outliers. This process is repeated iteratively until a total of n cliques are generated. Consequently, the total number of structural outliers in the network becomes $m \times n$.

Dataset	Type	Nodes	Edges	Features	Avg. Degree	Outliers	Outlier Ratio
Weibo	organic	8,405	407,963	400	48.5	868	10.3%
Reddit	organic	10,984	168,016	64	15.3	366	3.3%
Disney	organic	124	335	28	2.7	6	4.8%
Books	organic	1,418	3,695	21	2.6	28	2.0%
Enron	organic	13,533	176,987	18	13.1	5	0.04%
Cora	injected	2,708	11,060	1,433	4.1	138	5.1%
Amazon	injected	13,752	515,042	767	37.2	694	5.0%

Table 3.1: Graph dataset statistics

3.3 Results

The following results are benchmarks of GCNAE, DOMINANT, DONE, ADONE, ANOMALYDAE, and CONAD, which were trained over 10, 100, 300, and 400 epochs using randomized hyperparameter estimation. The results represent the mean of 20 trials for each set of hyperparameters. The results for ANOMALYDAE on the Amazon dataset were not obtained due to GPU memory limitations.

The training environment consisted of a Linux server running Python 3.12.2, CUDA 12, and a TU104 (GeForce RTX 2080 SUPER) GPU with 8GB of RAM. The models were implemented using Torch 2.3.1, Torch Geometry 2.5.3, and PyGraphOD 1.1.0.

The hyperparameter search space was defined as follows:

$$\begin{cases} \text{learning\_rate} \in \{0.01, 0.05, 0.1\} \\ \text{dropout} \in \{0.0, 0.1, 0.3\} \\ \text{weight\_decay} = 0.01 \\ \text{number\_neighbors} \in \{\text{all}\} \\ \text{hidden\_dimension} \in \{16, 32, 64, 128, 256\} \\ \text{- and other specific parameters for each model} \end{cases}$$

3.3.1 Synthetic and organic outliers

	Cora	Amazon	Weibo	Reddit	Disney	Books	Enron	AVG R
GCNAE	70.9 ± 0.0 Max: 70.9 Rank: 6.0	74.2 ± 0.0 Max: 74.2 Rank: 3.0	88.4 ± 1.4 Max: 89.9 Rank: 1.0	50.6 ± 0.0 Max: 50.6 Rank: 6.0	44.3 ± 4.2 Max: 49.9 Rank: 4.0	50.3 ± 3.9 Max: 56.4 Rank: 4.0	42.3 ± 3.8 Max: 44.3 Rank: 6.0	4.3
DOMINANT	75.3 ± 5.7 Max: 83.4 Rank: 5.0	71.6 ± 0.9 Max: 73.7 Rank: 5.0	81.8 ± 14.3 Max: 89.9 Rank: 5.0	56.1 ± 0.0 Max: 56.1 Rank: 1.0	46.8 ± 4.9 Max: 58.6 Rank: 3.0	50.8 ± 7.4 Max: 60.2 Rank: 3.0	53.5 ± 4.7 Max: 58.7 Rank: 2.0	3.4
DONE	82.4 ± 4.9 Max: 87.9 Rank: 1.0	84.6 ± 7.7 Max: 91.2 Rank: 1.0	83.6 ± 2.2 Max: 86.1 Rank: 4.0	55.5 ± 0.4 Max: 56.6 Rank: 3.0	41.6 ± 4.4 Max: 49.4 Rank: 6.0	42.5 ± 4.8 Max: 55.6 Rank: 6.0	47.7 ± 6.2 Max: 62.9 Rank: 5.0	3.7
ADONE	80.1 ± 3.0 Max: 84.6 Rank: 3.0	81.9 ± 5.0 Max: 90.7 Rank: 2.0	80.9 ± 5.8 Max: 86.7 Rank: 6.0	53.1 ± 3.9 Max: 57.0 Rank: 5.0	49.2 ± 4.9 Max: 62.7 Rank: 1.0	53.3 ± 2.4 Max: 56.5 Rank: 2.0	48.3 ± 5.2 Max: 56.4 Rank: 4.0	3.3
ANOMALYDAE	81.2 ± 2.9 Max: 85.4 Rank: 2.0	-	88.3 ± 2.6 Max: 93.7 Rank: 2.0	55.5 ± 0.8 Max: 56.7 Rank: 4.0	48.0 ± 0.8 Max: 49.3 Rank: 2.0	62.1 ± 9.4 Max: 73.0 Rank: 1.0	58.8 ± 9.5 Max: 76.0 Rank: 1.0	2.0
CONAD	77.2 ± 5.0 Max: 83.4 Rank: 4.0	71.7 ± 1.0 Max: 73.7 Rank: 4.0	84.6 ± 9.0 Max: 89.8 Rank: 3.0	56.0 ± 0.2 Max: 56.2 Rank: 2.0	44.2 ± 7.9 Max: 59.7 Rank: 5.0	49.6 ± 5.8 Max: 57.8 Rank: 5.0	51.9 ± 4.4 Max: 57.5 Rank: 3.0	3.7

Table 3.2: AUROC

	Cora	Amazon	Weibo	Reddit	Disney	Books	Enron	AVG Rank
GCNAE	13.2 ± 0.0 Max: 13.2 Rank: 6.0	34.8 ± 0.0 Max: 34.8 Rank: 1.0	29.5 ± 2.6 Max: 32.4 Rank: 1.0	3.4 ± 0.0 Max: 3.4 Rank: 6.0	5.1 ± 0.7 Max: 7.5 Rank: 5.0	2.1 ± 0.2 Max: 2.7 Rank: 5.0	0.0 ± 0.0 Max: 0.0 Rank: 6.0	4.3
DOMINANT	16.7 ± 4.3 Max: 21.1 Rank: 5.0	11.8 ± 0.3 Max: 12.6 Rank: 5.0	24.9 ± 10.9 Max: 32.2 Rank: 5.0	3.7 ± 0.0 Max: 3.7 Rank: 2.0	6.4 ± 3.3 Max: 20.5 Rank: 1.0	2.3 ± 0.6 Max: 3.8 Rank: 4.0	0.1 ± 0.0 Max: 0.1 Rank: 2.0	3.4
DONE	23.4 ± 10.3 Max: 45.5 Rank: 1.0	19.9 ± 6.7 Max: 33.7 Rank: 2.0	26.8 ± 3.9 Max: 32.3 Rank: 3.0	3.8 ± 0.1 Max: 4.0 Rank: 1.0	5.0 ± 0.3 Max: 5.7 Rank: 6.0	1.8 ± 0.3 Max: 2.5 Rank: 6.0	0.1 ± 0.0 Max: 0.2 Rank: 4.0	3.3
ADONE	18.9 ± 4.0 Max: 28.9 Rank: 2.0	19.8 ± 5.8 Max: 30.4 Rank: 3.0	23.4 ± 7.6 Max: 31.5 Rank: 6.0	3.5 ± 0.4 Max: 3.9 Rank: 5.0	5.9 ± 1.0 Max: 8.9 Rank: 2.0	2.4 ± 0.3 Max: 3.3 Rank: 3.0	0.1 ± 0.0 Max: 0.2 Rank: 5.0	3.7
ANOMALYDAE	17.6 ± 2.3 Max: 21.6 Rank: 4.0	-	26.9 ± 6.2 Max: 33.8 Rank: 2.0	3.7 ± 0.0 Max: 3.8 Rank: 4.0	5.6 ± 0.1 Max: 5.7 Rank: 4.0	3.5 ± 1.2 Max: 5.2 Rank: 1.0	0.1 ± 0.0 Max: 0.1 Rank: 1.0	2.7
CONAD	18.0 ± 3.4 Max: 21.1 Rank: 3.0	11.9 ± 0.3 Max: 12.6 Rank: 4.0	25.1 ± 10.2 Max: 32.2 Rank: 4.0	3.7 ± 0.0 Max: 3.7 Rank: 3.0	5.7 ± 1.5 Max: 10.7 Rank: 3.0	2.4 ± 1.0 Max: 5.9 Rank: 2.0	0.1 ± 0.0 Max: 0.1 Rank: 3.0	3.1

Table 3.3: Average Precision

	Cora	Amazon	Weibo	Reddit	Disney	Books	Enron	AVG Rank
GCNAE	22.0 ± 0.0 Max: 22.0 Rank: 6.0	36.3 ± 0.0 Max: 36.3 Rank: 1.0	41.1 ± 4.3 Max: 46.0 Rank: 1.0	4.7 ± 0.0 Max: 4.7 Rank: 5.0	1.1 ± 3.2 Max: 10.5 Rank: 4.5	2.4 ± 1.5 Max: 5.9 Rank: 3.0	0.0 ± 0.0 Max: 0.0 Rank: 6.0	3.8
DOMINANT	29.5 ± 9.4 Max: 36.2 Rank: 3.0	21.3 ± 0.5 Max: 22.5 Rank: 5.0	36.1 ± 16.4 Max: 46.0 Rank: 5.0	6.0 ± 0.1 Max: 6.0 Rank: 4.0	1.6 ± 3.9 Max: 10.5 Rank: 3.0	2.3 ± 2.4 Max: 8.2 Rank: 4.0	0.1 ± 0.1 Max: 0.1 Rank: 3.5	3.9
DONE	27.0 ± 9.2 Max: 40.1 Rank: 4.0	28.3 ± 9.5 Max: 40.9 Rank: 2.0	39.0 ± 4.0 Max: 44.3 Rank: 3.0	6.0 ± 0.3 Max: 7.0 Rank: 2.5	1.1 ± 3.2 Max: 10.5 Rank: 4.5	1.7 ± 1.3 Max: 4.7 Rank: 6.0	0.1 ± 0.1 Max: 0.1 Rank: 5.0	3.9
ADONE	29.7 ± 8.1 Max: 39.1 Rank: 2.0	24.5 ± 8.3 Max: 40.4 Rank: 3.0	35.2 ± 8.5 Max: 45.3 Rank: 6.0	4.6 ± 1.9 Max: 6.1 Rank: 6.0	2.6 ± 4.7 Max: 10.5 Rank: 1.5	3.4 ± 1.4 Max: 5.9 Rank: 2.0	0.1 ± 0.1 Max: 0.1 Rank: 1.0	3.1
ANOMALYDAE	26.2 ± 5.7 Max: 35.7 Rank: 5.0	-	41.1 ± 6.2 Max: 47.3 Rank: 2.0	6.0 ± 0.1 Max: 6.1 Rank: 1.0	0.0 ± 0.0 Max: 0.0 Rank: 6.0	4.5 ± 4.0 Max: 10.7 Rank: 1.0	0.1 ± 0.1 Max: 0.1 Rank: 2.0	2.8
CONAD	32.2 ± 6.5 Max: 36.2 Rank: 1.0	21.4 ± 0.5 Max: 22.5 Rank: 4.0	37.3 ± 13.4 Max: 46.0 Rank: 4.0	6.0 ± 0.1 Max: 6.1 Rank: 2.5	2.6 ± 4.7 Max: 10.5 Rank: 1.5	2.3 ± 2.4 Max: 8.2 Rank: 5.0	0.1 ± 0.1 Max: 0.1 Rank: 3.5	3.1

Table 3.4: F1

Precision and recall results are provided in Appendix A.1.

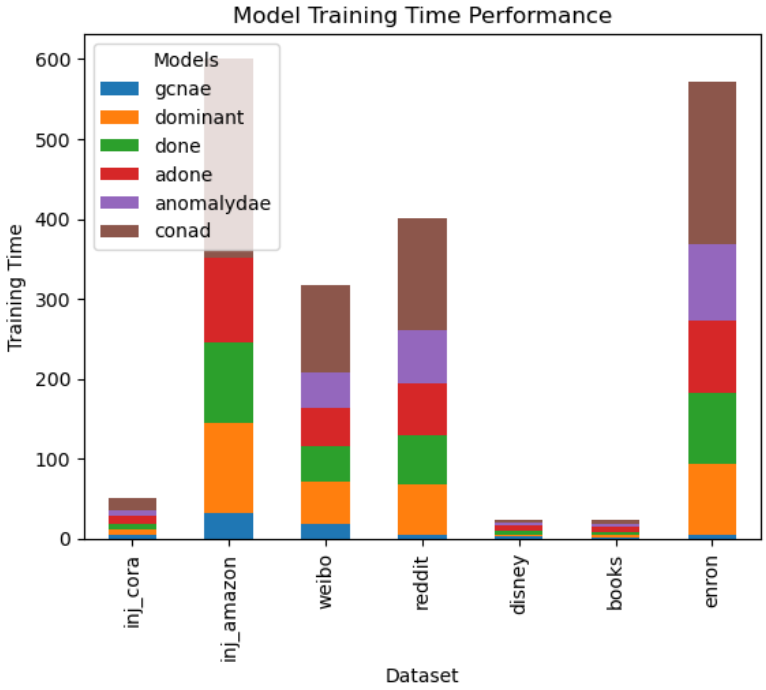


Figure 3.1: Training time (seconds) comparison

3.3.2 Structural and contextual outliers

These are synthetic outliers that are combined with real datasets, such as Cora [Sen et al., 2008] and Amazon [Shchur et al., 2018], which lack inherent outliers. Additionally, we use these injections due to the scarcity of datasets that clearly differentiate between structural and contextual outliers.

	Inj. Cora			Inj. Amazon			AVG Rank		
	All	Contextual	Structural	All	Contextual	Structural	All	Ctx.	Str.
GCNAE	70.9 ± 0.0 Max: 70.9 Rank: 6.0	88.9 ± 0.0 Max: 88.9 Rank: 1.0	52.5 ± 0.0 Max: 52.5 Rank: 6.0	74.2 ± 0.0 Max: 74.2 Rank: 3.0	98.6 ± 0.0 Max: 98.6 Rank: 1.0	49.0 ± 0.0 Max: 49.0 Rank: 5.0	4.5	1.0	5.5
DOMINANT	75.3 ± 5.7 Max: 83.4 Rank: 5.0	56.8 ± 7.1 Max: 71.1 Rank: 6.0	93.1 ± 5.7 Max: 96.1 Rank: 2.0	71.6 ± 0.9 Max: 73.7 Rank: 5.0	51.2 ± 1.8 Max: 55.0 Rank: 5.0	91.3 ± 0.1 Max: 91.6 Rank: 2.0	5.0	5.5	2.0
DONE	82.4 ± 4.9 Max: 87.9 Rank: 1.0	73.6 ± 6.4 Max: 78.0 Rank: 3.0	89.9 ± 6.8 Max: 97.2 Rank: 4.0	84.6 ± 7.7 Max: 91.2 Rank: 1.0	82.5 ± 15.5 Max: 96.8 Rank: 2.0	85.1 ± 5.5 Max: 91.4 Rank: 4.0	1.0	2.5	4.0
ADONE	80.1 ± 3.0 Max: 84.6 Rank: 3.0	67.5 ± 6.1 Max: 77.9 Rank: 4.0	91.6 ± 4.5 Max: 96.8 Rank: 3.0	81.9 ± 5.0 Max: 90.7 Rank: 2.0	76.5 ± 10.6 Max: 97.0 Rank: 3.0	85.9 ± 3.2 Max: 90.9 Rank: 3.0	2.5	3.5	3.0
ANOMALYDAE	81.2 ± 2.9 Max: 85.4 Rank: 2.0	79.5 ± 10.6 Max: 88.1 Rank: 2.0	81.7 ± 11.8 Max: 96.1 Rank: 5.0	-	-	-	2.0	2.0	5.0
CONAD	77.2 ± 5.0 Max: 83.4 Rank: 4.0	59.5 ± 7.9 Max: 71.1 Rank: 5.0	94.1 ± 5.0 Max: 96.1 Rank: 1.0	71.7 ± 1.0 Max: 73.7 Rank: 4.0	51.3 ± 1.8 Max: 55.0 Rank: 4.0	91.4 ± 0.1 Max: 91.5 Rank: 1.0	4.0	4.5	1.0

Table 3.5: AUROC

	Inj. Cora			Inj. Amazon			AVG Rank		
	All	Contextual	Structural	All	Contextual	Structural	All	Ctx.	Str.
GCNAE	13.2 ± 0.0 Max: 13.2 Rank: 6.0	13.7 ± 0.0 Max: 13.7 Rank: 1.0	3.1 ± 0.0 Max: 3.1 Rank: 6.0	34.8 ± 0.0 Max: 34.8 Rank: 1.0	56.7 ± 0.0 Max: 56.7 Rank: 1.0	2.5 ± 0.0 Max: 2.5 Rank: 5.0	3.5	1.0	5.5
DOMINANT	16.7 ± 4.3 Max: 21.1 Rank: 5.0	3.3 ± 1.1 Max: 5.7 Rank: 6.0	19.9 ± 6.5 Max: 24.2 Rank: 3.0	11.8 ± 0.3 Max: 12.6 Rank: 5.0	2.6 ± 0.1 Max: 2.9 Rank: 5.0	12.2 ± 0.1 Max: 12.5 Rank: 2.0	5.0	5.5	2.5
DONE	23.4 ± 10.3 Max: 45.5 Rank: 1.0	7.7 ± 1.4 Max: 10.2 Rank: 3.0	21.6 ± 16.2 Max: 59.6 Rank: 1.0	19.9 ± 6.7 Max: 33.7 Rank: 2.0	13.0 ± 7.7 Max: 30.8 Rank: 3.0	9.6 ± 4.3 Max: 24.7 Rank: 3.0	1.5	3.0	2.0
ADONE	18.9 ± 4.0 Max: 28.9 Rank: 2.0	6.8 ± 1.6 Max: 10.7 Rank: 4.0	16.2 ± 6.6 Max: 30.9 Rank: 4.0	19.8 ± 5.8 Max: 30.4 Rank: 3.0	15.5 ± 8.9 Max: 32.6 Rank: 2.0	8.6 ± 2.0 Max: 12.4 Rank: 4.0	2.5	3.0	4.0
ANOMALYDAE	17.6 ± 2.3 Max: 21.6 Rank: 4.0	9.9 ± 4.2 Max: 13.7 Rank: 2.0	11.8 ± 8.3 Max: 24.5 Rank: 5.0	-	-	-	4.0	2.0	5.0
CONAD	18.0 ± 3.4 Max: 21.1 Rank: 3.0	3.7 ± 1.3 Max: 5.7 Rank: 5.0	21.0 ± 4.7 Max: 24.1 Rank: 2.0	11.9 ± 0.3 Max: 12.6 Rank: 4.0	2.7 ± 0.1 Max: 2.9 Rank: 4.0	12.2 ± 0.1 Max: 12.5 Rank: 1.0	3.5	4.5	1.5

Table 3.6: Average Precision

	Inj. Cora			Inj. Amazon			AVG Rank		
	All	Contextual	Structural	All	Contextual	Structural	All	Ctx.	Str.
GCNAE	22.0 ± 0.0 Max: 22.0 Rank: 6.0	21.7 ± 0.0 Max: 21.7 Rank: 1.0	5.3 ± 0.0 Max: 5.3 Rank: 6.0	36.3 ± 0.0 Max: 36.3 Rank: 1.0	40.6 ± 0.0 Max: 40.6 Rank: 1.0	3.7 ± 0.0 Max: 3.7 Rank: 5.0	3.5	1.0	5.5
DOMINANT	29.5 ± 9.4 Max: 36.2 Rank: 3.0	4.4 ± 2.8 Max: 10.6 Rank: 6.0	31.9 ± 11.1 Max: 38.1 Rank: 2.0	21.3 ± 0.5 Max: 22.5 Rank: 5.0	4.6 ± 0.5 Max: 5.7 Rank: 5.0	21.4 ± 0.3 Max: 21.8 Rank: 2.0	4.0	5.5	2.0
DONE	27.0 ± 9.2 Max: 40.1 Rank: 4.0	12.3 ± 3.3 Max: 17.6 Rank: 3.0	21.2 ± 11.2 Max: 37.0 Rank: 4.0	28.3 ± 9.5 Max: 40.9 Rank: 2.0	20.1 ± 11.8 Max: 37.5 Rank: 2.0	14.5 ± 5.2 Max: 22.0 Rank: 3.0	3.0	2.5	3.5
ADONE	29.7 ± 8.1 Max: 39.1 Rank: 2.0	10.1 ± 3.3 Max: 17.0 Rank: 4.0	26.6 ± 10.9 Max: 39.3 Rank: 3.0	24.5 ± 8.3 Max: 40.4 Rank: 3.0	16.9 ± 9.7 Max: 37.8 Rank: 3.0	13.0 ± 4.0 Max: 21.6 Rank: 4.0	2.5	3.5	3.5
ANOMALYDAE	26.2 ± 5.7 Max: 35.7 Rank: 5.0	15.8 ± 7.3 Max: 21.7 Rank: 2.0	16.7 ± 14.0 Max: 38.1 Rank: 5.0	-	-	-	5.0	2.0	5.0
CONAD	32.2 ± 6.5 Max: 36.2 Rank: 1.0	5.4 ± 3.4 Max: 10.6 Rank: 5.0	34.4 ± 7.5 Max: 38.1 Rank: 1.0	21.4 ± 0.5 Max: 22.5 Rank: 4.0	4.6 ± 0.6 Max: 5.7 Rank: 4.0	21.5 ± 0.1 Max: 21.8 Rank: 1.0	2.5	4.5	1.0

Table 3.7: F1

Precision and recall results are provided in Appendix A.2.

3.4 Discussion and statistical analysis

3.4.1 Synthetic and organic outliers

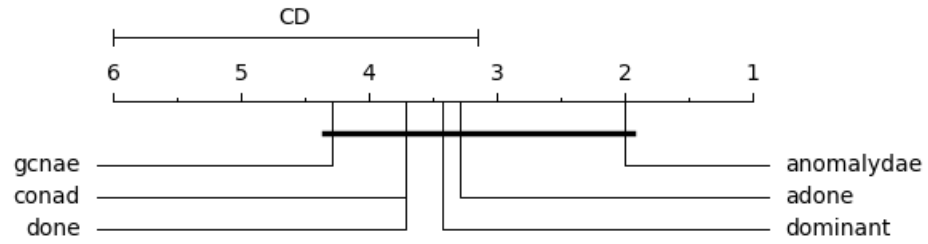


Figure 3.2: Average ranks diagram comparing, in terms of AUROC, the unsupervised deep learning methods for node anomalies in static attributed graphs

Statistical analysis was conducted across six models with seven datasets each. The family-wise significance level was set to alpha=0.050. Initial testing indicated that none of the models deviated significantly from normality (minimal observed p-value=0.014), allowing us to assume normality across all models.

We employed the non-parametric Friedman test as an omnibus test to detect significant differences in mean values across models. Post-hoc analysis using the Nemenyi test was conducted to identify which specific differences were statistically significant. Mean values (M), standard deviations (SD), and mean ranks (MR) were calculated across all models and datasets.

The Friedman test rejected the null hypothesis (p=nan), confirming statistically significant differences in central tendency among the models, as shown in the table 3.8.

According to the post-hoc Nemenyi test, no significant differences were found within the groups of models, as shown in the figure 3.2.

	MR	M	SD	CI	d	Magnitude
gcnae	4.286	0.602	0.177	[0.344, 0.859]	0.000	negligible
done	3.714	0.625	0.201	[0.331, 0.920]	-0.126	negligible
conad	3.714	0.622	0.155	[0.395, 0.848]	-0.122	negligible
dominant	3.429	0.623	0.137	[0.423, 0.823]	-0.134	negligible
adone	3.286	0.638	0.161	[0.402, 0.874]	-0.217	small
anomalydae	2.000	0.656	0.157	[0.409, 0.904]	-0.328	small

Table 3.8: Summary of populations, in terms of AUROC

Most deep learning methods evaluated in our study show limited effectiveness on Disney, Books, and Enron compared to traditional baselines. This is largely attributed to the small-scale nature of the graphs in Disney and Books. The reduced dataset size poses challenges for deep learning models in accurately capturing the distribution of inliers, potentially leading to issues with overfitting.

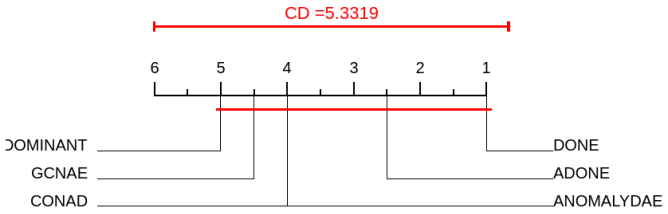
Given that the majority of evaluated methods are optimized for handling structural and contextual outliers as defined in the dataset statistics table 3.1, we sought to understand why

certain methods succeed or fail in detecting organic outliers. To explore this, we analyzed the patterns of organic outliers using metrics aligned with the definitions of structural and contextual outliers. Specifically, we found that the success of many methods on datasets like Weibo stems from the fact that its outliers exhibit characteristics typical of structural outliers [Liu et al., 2022b].

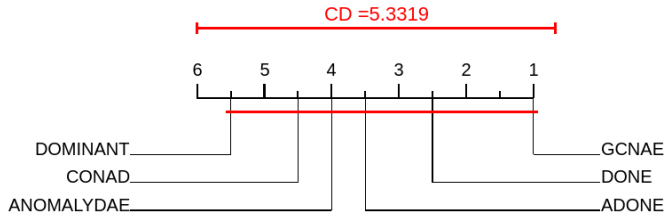
Moreover, ANOMALYDAE consistently performs well across multiple datasets, demonstrating robustness and reliability in various scenarios.

CONAD requires the most time because it utilizes contrastive learning, which involves conducting pairwise comparisons within mini-batches. In contrastive learning, each mini-batch typically involves comparing pairs of instances to learn representations that distinguish between similar and dissimilar instances effectively. This process is computationally intensive compared to methods that do not rely on pairwise comparisons within mini-batches, thereby increasing the overall computational time required for training and inference.

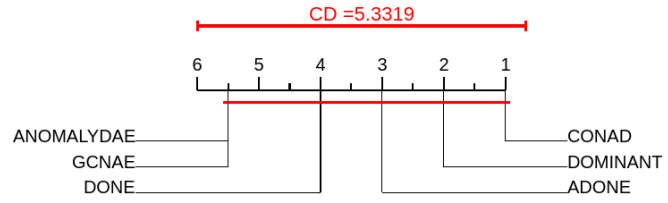
3.4.2 Structural and contextual outliers



(a) All



(b) Contextual



(c) Structural

Figure 3.3: Average ranks diagram comparing, in terms of AUROC, the unsupervised deep learning methods for node anomalies in static attributed graphs

GCNAE consistently excels in detecting contextual anomalies across diverse datasets and metrics, whereas CONAD identifies well structural anomalies.

DOMINANT and DONE achieve similar mean ROC-AUC scores for detecting structural outliers in the Cora and Amazon datasets. Despite DOMINANT's inclusion of 4-hop neighbor information compared to DONE's 1-hop approach, both models perform comparably. This

observation highlights a potential strategy for designing outlier detection models: while higher-order information integration can enhance performance, it must be balanced against computational efficiency concerns, particularly to mitigate issues such as over-smoothing in GNNs. [Liu et al., 2022b]

Conclusion This chapter rigorously evaluates deep-learning methods for anomaly detection in static attributed graphs, revealing varied performance across models and datasets. Findings emphasize the need for tailored approaches to detect structural versus contextual anomalies, with implications for future research in optimizing detection accuracy and computational efficiency in graph data analysis.

Chapter 4

A New Framework for Structural Anomaly Detection

Introduction Detecting structural anomalies in graphs is challenging due to the ambiguity of anomaly definitions and the complex interdependencies within graph structures. Traditional anomaly detection methods, such as distance-based metrics, reconstruction techniques, and probabilistic models, often struggle to provide a clear and interpretable distinction between normal and anomalous patterns, especially when anomaly distributions are not predefined. To address these limitations, we introduce the Pattern-based Geometric Framework for Anomaly Detection (PGF-AD). PGF-AD leverages topological theories to offer a comprehensive approach that considers both local node properties and their global context within the graph. This framework provides an intuitive and interpretable description of anomalies, without requiring assumptions about predefined anomaly distributions.

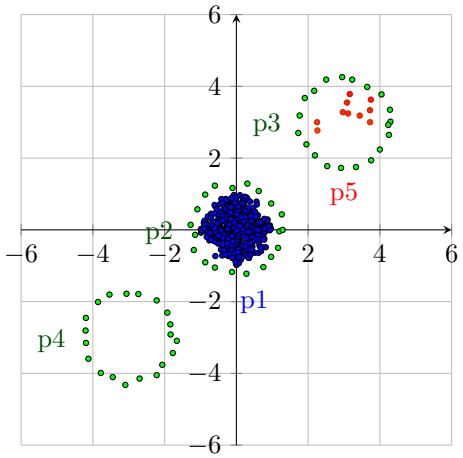


Figure 4.1: This figure illustrates the concept of strong pattern detection in a 2D space. The dense disk of points p_1, shown in **blue**, represents a strong pattern due to its **high weight**. The three clouds of non-dense points (p_2, p_3, p_4), shown in **green**, represent **frequent patterns** despite their low density. The **red** points p_5 scattered randomly across the plot represent **anomalies**, as they do not follow any clear strong pattern.

Our second contribution, Minomaly, extends PGF-AD by introducing an efficient deep learning method specifically designed to detect unexpected structural connections in graphs. By integrating topological insights with deep learning techniques, Minomaly enhances the accuracy and interpretability of anomaly detection models. The motivation behind PGF-AD and Minomaly lies in their ability to address the inherent ambiguities in anomaly definitions and to capture the nuanced relationships within graph structures. In this chapter, we explore the theoretical foundations, implementation details, and empirical validations of PGF-AD and Minomaly, demonstrating their efficacy through rigorous benchmarking and performance evaluations.

4.1 PGF-AD: Pattern-based Geometric Framework for Anomaly Detection

We introduce in this section a topological theory that defines a comprehensive framework for anomaly detection based on the concepts of pattern weight, pattern frequency, and pattern strength.

This approach to anomaly detection takes into account both the internal consistency of a pattern through the weight by quantifying its significance. A high-weight pattern comprises many elements that are closely related. Pattern frequency measures the commonality of a pattern, taking into account its context within the larger space. It is determined by the count of similar patterns.

Pattern strength is a fusion of pattern weight and pattern frequency, signifying the robustness of a pattern in a dataset. We define an anomaly as a weak pattern, or an element of this pattern.

This dual consideration enables the effective identification of anomalies that might be overlooked by methods that only consider one of these factors. See figure 4.1.

Let X be a non-empty set and let $T_F, T_S \in \mathbb{R}_*^+$ and $\alpha \in [0, 1]$.

4.1.1 Preliminary definitions

Pattern

We call p a pattern iff $p \subseteq X$. The set of all patterns of X is denoted P_X

$$P_X = \{p : p \subseteq X\}$$

Pattern cover

We call S_X a pattern cover of X any set of subsets of X i.e. $S_X \subseteq P_X$ such that

$$\bigcup_{p \in S_X} p = X$$

The set of all pattern covers of X is denoted by PS_X.

Pattern similarity

Let d_X be a symmetric and reflexive dissimilarity measure on P_X, i.e., a function $d_X : P_X \times P_X \to \mathbb{R}^+$ such that

$$\forall p_1, p_2 \in P_X : d_X(p_1, p_2) = d_X(p_2, p_1) \text{ and } d_X(p_1, p_1) = 0$$

d_X is called pattern similarity on X. The goal is to represent the dissimilarity between two patterns.

Pattern similarity relation

Let $\sim_{d_X}$ be the relation over P_X, of a pattern similarity d_X defined by

$$\forall p_1, p_2 \in P_X : p_1 \sim_{d_X} p_2 \iff \begin{cases} d_X(p_1, p_2) < T_F \\ \text{or } \exists p \in P_X : d_X(p_1, p) < T_F \text{ and } d_X(p, p_2) < T_F \end{cases}$$

Property $\sim_{d_X}$ is equivalence relation over P_X

Pattern similarity class

We call a pattern class of a pattern p over P_X, of a pattern similarity d_X, denoted $[p]$ the equivalence class of p modulo $\sim_{d_X}$ i.e.

$$[p]_{d_X} = \{p' \in P_X : p \sim_{d_X} p'\}$$

Notes

- p is called a representative pattern of the class $[p]_{d_X}$
- $P_X/\sim_{d_X} = \{[p]_{d_X} | p \in P_X\}$ denotes the quotient set of P_X by $\sim_{d_X}$, the set of all pattern classes over P_X, of a pattern similarity d_X
- $P_X/\sim_{d_X}$ is a partition of P_X

Pattern homogeneity

Let h_X be a function $h_X : P_X \to [0, 1]$. h_X is called pattern homogeneity on X. The goal is to represent how close the elements within a pattern are.

4.1.2 Definition

Let $p \in P_X$, S_X a pattern cover of X and d_X a pattern similarity.

Pattern weight

The weight of p in X is a function $Weight_X : P_X \to [0, 1]$ such that

$$Weight_X(p) = \frac{Card(p)}{Card(X)} \times h_X(p)$$

and h_X is a pattern homogeneity on X. The goal is to represent how weighted the pattern is in the set.

Pattern counter

The counter of $P \subseteq P_X$ is a function $Count_X : P \subseteq P_X \to \mathbb{N}$. Its goal is to define a way to count a set of patterns. Usually $Count_X = Card$.

Pattern frequency

The frequency of p in X is a function $Freq_X : P_X \to [0,1]$ such that

$$Freq_X(p) = \frac{Count_X([p]_{d_X})}{Card(X)}$$

$Count_X$ is a pattern counter and d_X is a pattern similarity on X such that it preserves approximately the cardinality and homogeneity of similar patterns $\forall p_1, p_2 \in S_X$ i.e.

$$d_X(p_1,p_2) < T_F \implies \begin{cases} \exists \varepsilon' > 0 : |Card(p_1) - Card(p_2)| < \varepsilon' \cdot T_F \cdot Card(X) \\ \exists \varepsilon'' > 0 : |h_X(p_1) - h_X(p_2)| < \varepsilon'' \cdot T_F \end{cases}$$

The goal is to represent how frequent the pattern is in the set.

Property

$$d_X(p_1,p_2) < T_F \implies \exists \varepsilon > 0 : |Weight_X(p_1) - Weight_X(p_2)| < \varepsilon \cdot T_F$$

$$p_1 \sim_{d_X} p_2 \implies \exists \varepsilon > 0 : |Weight_X(p_1) - Weight_X(p_2)| < \varepsilon \cdot T_F$$

Pattern strength

The strength of p in X is a function $Strength_X : P_X \to [0,1]$ such that

$$Strength_X(p) = \alpha \cdot Freq_X(p) + (1-\alpha) \cdot Weight_X(p)$$

The goal is to represent how strong the pattern is in the set.

Strong patterns set

The strong patterns set of S_X is a set denoted $SP(S_X)$ such that

$$SP(S_X) = \{p \in S_X : Strength_X(p) > T_S\}$$

.

Weak pattern

A weak pattern p within a cover S_X is a pattern which is not strong, i.e.

$$p \notin SP(S_X)$$

The set of all weak patterns of S_X is

$$WP_X(S_X) = S_X \setminus SP(S_X) = \{p \in S_X : Strength_X(p) \leq T_S\}$$

.

Anomaly

Anomaly is a weak pattern p of a cover S_X or an element of X that belongs to a weak pattern of this cover. The set of all anomalies of a cover S_X is denoted $A(S_X)$ such that

$$A(S_X) = W_X(S_X) \cup \{x \in X | \exists p \in W_X(S_X) : x \in p\}$$

4.1.3 Problem setup

The problem is to find a cover S of a given non-empty set X that maximizes its overall strength of its patterns. We model this by this objective function

$$\max_{S \in PS_X} \sum_{p \in S} Strength(p)$$

4.1.4 Framework projection onto relational spaces

Let X be a non-empty set, $\mathcal{R}$ a binary relation over X and $p, p_1, p_2 \in P_X$.

Pattern isomorphism

$$p_1 \text{ and } p_2 \text{ are isomorphic} \iff p_1 \cong_{\mathcal{R}} p_2 \iff \begin{cases} \exists f : p_1 \to p_2 \\ f \text{ is bijective function} \\ \forall x_1, x_2 \in p_1 : x_1 \mathcal{R} x_2 \iff f(x_1) \mathcal{R} f(x_2) \end{cases}$$

Sub-pattern

$$p_1 \text{ is sub-pattern of } p_2 \iff p_1 \preceq_{\mathcal{R}} p_2 \iff \exists p \subseteq p_2 : p_1 \cong_{\mathcal{R}} p$$

Pattern intersection

The intersection of p_1 and p_2 is the maximum common pattern between them with respect to $\cong_{\mathcal{R}}$

$$p_1 \curlywedge p_2 = \arg\max_{p_1' \subseteq p_1} \{Card(p_1') | \forall p_2' \subseteq p_2 : p_1' \cong_{\mathcal{R}} p_2'\}$$

Isomorphic-based pattern similarity

Let's define $d_X : P_X \times P_X \to [0, 1]$ the pattern similarity by

$$d_X(p_1, p_2) = \begin{cases} 1 - \frac{Card(p_1)}{Card(p_2)} & \text{if } p_1 \preceq_{\mathcal{R}} p_2 \\ 1 - \frac{Card(p_2)}{Card(p_1)} & \text{if } p_2 \preceq_{\mathcal{R}} p_1 \\ 1 & \text{otherwise} \end{cases}$$

- $\cong_{\mathcal{R}}$ is equivalence relation over P_X
- $\preceq_{\mathcal{R}}$ is partial order relation over P_X with respect to $\cong_{\mathcal{R}}$
- d_X is reflexive and symmetric

Pattern homogeneity

Let's define $h_X : P_X \to [0, 1]$ the pattern homogeneity by

$$h_X(p) = \begin{cases} 1 & \text{if } \exists (x_1, x_2, \cdots, x_{Card(p)}) \in p^{Card(p)} : \forall i \in [[1, Card(p) - 1]] : x_i \neq x_{i+1} \text{ and } x_i \mathcal{R} x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

That's mean, a homogenous pattern is a connected pattern by the relation $\mathcal{R}$

Pattern order-embedding function

Pattern order-embedding is an injective function $\phi_X : P_X \to (\mathbb{R}^+)^n$ with respect to $\cong_{\mathcal{R}}$, such that $n \in \mathbb{N}^*$ and

$$p_1 \preceq_{\mathcal{R}} p_2 \iff \phi_X(p_1) \preceq \phi_X(p_2)$$

[McFee and Lanckriet, 2009]

Notes

- Element-wise vectors comparison: $\forall z = (z_1, z_2, \cdots, z_n) \in \mathbb{R}^n$, $\forall z' = (z'_1, z'_2, \cdots, z'_n) \in \mathbb{R}^n$:

$$z \preceq z' \iff \forall i \in [[1, n]] : z_i \leq z'_i$$

- ϕ_X **injective** with respect to $\cong_{\mathcal{R}}$:

$$\phi_X(p_1) = \phi_X(p_2) \implies p_1 \cong_{\mathcal{R}} p_2$$

- As it is shown in figure 4.2, **frequency of pattern p can be quantified by counting the upper-right embeddings** z' that are above the pattern embedding z of p ($z \preceq z'$).

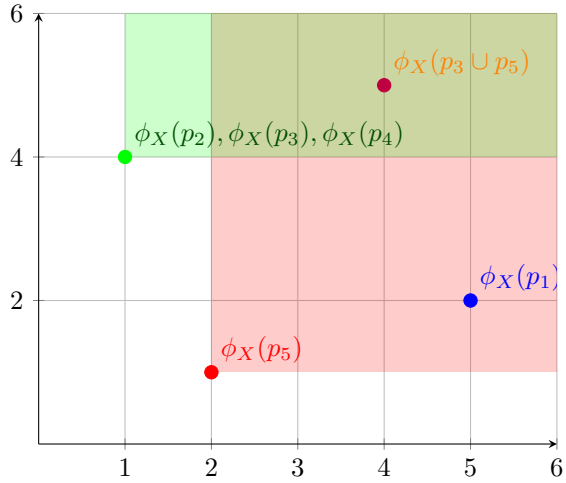


Figure 4.2: This figure illustrates the concept of pattern embedding in a 2D space ($n = 2$) of the given example in the figure 4.1. p_2, p_3, p_4 have the same coordinates because the injectivity of the pattern embedding. The pattern $p_3 \cup p_5$ is in the upper-right of p_2, p_3, p_4 and p_5 because p_2, p_3, p_4 and p_5 are sub-patterns of $p_3 \cup p_5$. Also, p_5 is in lower-left of p_1 because $p_5 \preceq_{\mathcal{R}} p_1$.

Feature space data

In the realm of our suggested theoretical framework for anomaly detection (PGF-AD), translating the problem definition into the domains of feature space data brings about distinct difficulties. The main obstacle in the context of feature space data is the establishment of the relation $\mathcal{R}$.

In spaces with high dimensionality, formulating a significant relation $\mathcal{R}$ that can encapsulate the intrinsic organization of the data is not a simple task. This complexity arises from the sparsity of data points in high-dimensional spaces, which diminishes the relevance of concepts

such as closeness or similarity between points, commonly used to define relations in spaces of lower dimensions. [Pestov, 2013]

High-dimensional spaces render the notion of distance almost meaningless. This phenomenon demonstrates that in high-dimensional spaces, points are increasingly sparse, yet their relative distances become smaller, undermining the relevance of distance measures and posing challenges for distance-based Machine Learning algorithms such as kNN (see figure 4.3).

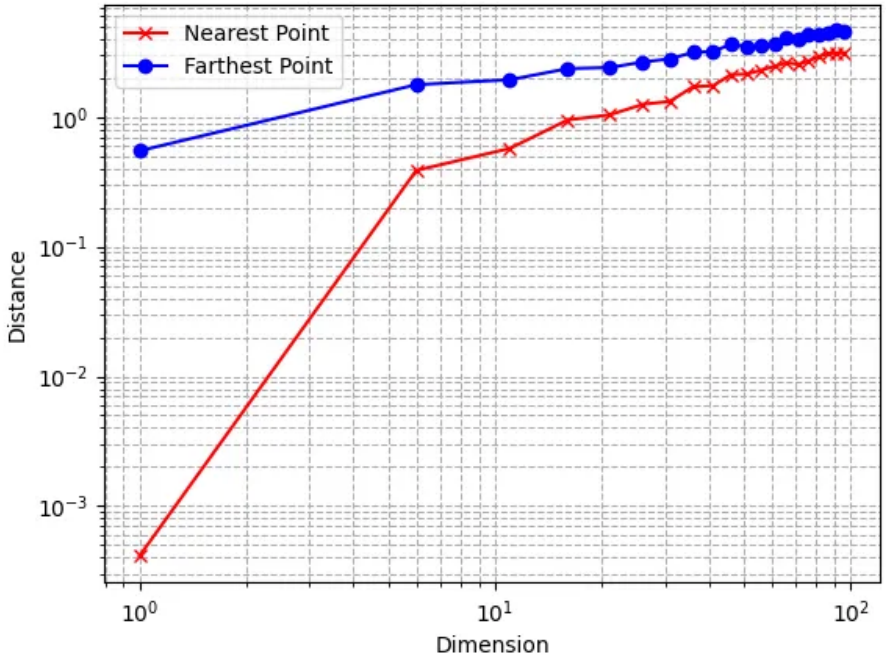


Figure 4.3: An experiment is conducted within an n-dimensional unit hypercube, generating a dataset by randomly sampling points to simulate a multivariate uniform distribution. A query point is then sampled, and the distances to its nearest and farthest neighbors in the dataset are measured. Results, shown on a log scale, indicate that as the dimensionality increases, the relative difference between the nearest and farthest neighbor distances decreases[1]

Graph data

On the other hand, when dealing with graph data, the problem definition fits quite well due to the well-defined relation. In graph data, the relation $\mathcal{R}$ is explicitly defined by the edges of the graph. This explicit relation provides a clear structure that can be leveraged to define patterns and detect anomalies under our proposed framework PGF-AD, making anomaly detection in graph data a more tractable problem.

We will be validating this assumption on graphs during the experimental phase 4.3.3 of our approach 4.2, providing empirical evidence to support our theoretical findings.

[1] https://towardsdatascience.com/

4.2 Mining for structural anomalies in graph nodes

Given a graph $G = (V, E)$ with the associated binary relation $\mathcal{R}$ over the non-empty set of nodes $X = V$ that is defined by the relation-graph E of edges with the same definitions in 4.1.4. Consequently, $P_X = P_V = \{V' : V' \subseteq V\}$

4.2.1 Definitions on graphs

Induced graph

$G' = (V', E')$ is induced graph of G by V' iff

$$\begin{cases} V' \subseteq V \\ E' = \{(x_1, x_2) \in E | x_1, x_2 \in V\} \end{cases}$$

V' is the pattern of G'

Node-anchored graph

(G', x) is x anchored induced graph of G iff

$$\begin{cases} G' = (V', E') \text{ is graph of } G \text{ induced by } V' \\ x \in V' \end{cases}$$

x is called the anchor associated to the graph G' of pattern V', and we define the function $anchor : P_V \times V \to V$ that maps each pattern to its corresponding anchor ($anchor(V', x) = x$).

Graph isomorphism

$G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ graphs of G are isomorphic iff $V_1 \cong_{\mathcal{R}} V_2$.

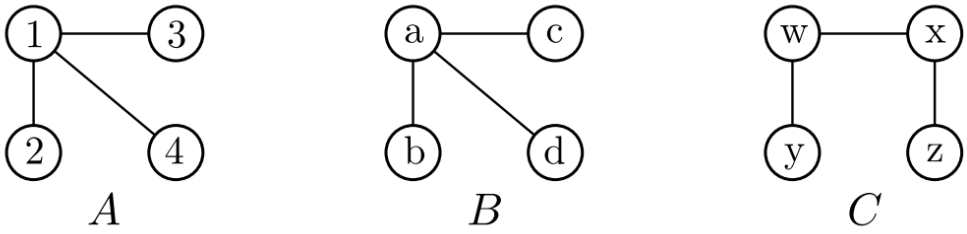


Figure 4.4: We have two graphs, A and B, which are isomorphic to each other, and a third graph, C, which is not isomorphic to either A or B. In these graphs, the mapping is such that the node '1' corresponds to 'a', the node '2' corresponds to 'b', and so on, up to the node '4'.[2]

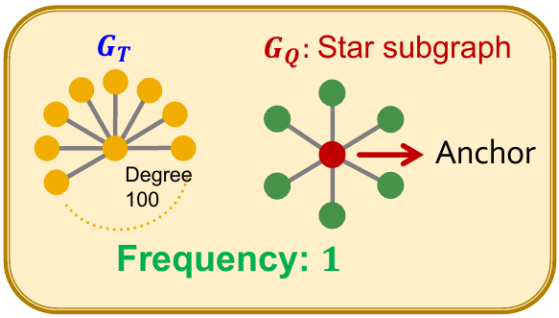
Problem complexity The graph isomorphism problem's complexity is unclear, leading to the creation of a new class, **GI**. This class includes problems reducible to the graph isomorphism problem in polynomial time. If the problem is solvable in polynomial time, GI=P. If it's NP-complete, GI=NP, making all NP problems solvable in quasi-polynomial time [Booth and Colbourn, 1979].

[2] https://www2.math.upenn.edu/

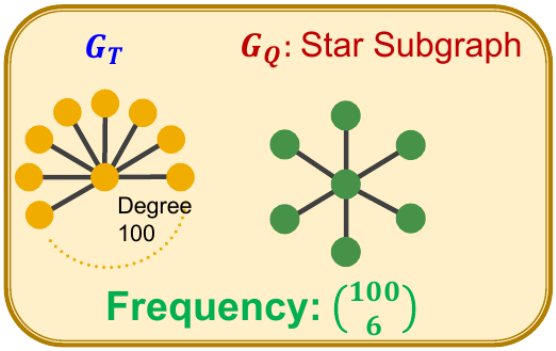
Node-anchored graph counting

We define the pattern counter $Count_V$ (4.1.2) by the number of the group of similar anchors of a set of patterns [Rex et al., 2020].

Consequently, $Freq_V$ will count the number of nodes y in G for which some x anchored graph G' of G is similar by d_X (isomorphic if $T_F \to 0$) to another y anchored graph G'' of G where x maps y by the isomorph.



(a) Node-level graph counting



(b) Graph-level graph counting

Figure 4.5: This figure demonstrates the efficiency of the anchored node graph counting definition in comparison to the graph counting definition. The anchored node definition reduces combinatorial complexity and also yields a more interpretable frequency. G_Q represents the query graph, which is confirmed to be a sub-pattern of G_T. $\binom{100}{6}$ is the number of possible subgraphs with 6 nodes in the larger motif of 100 nodes. [3]

Anti-monotonicity counting property A pattern frequency is considered to be anti-monotonic, if $\forall G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ graphs of G induced by V_1, V_2 resp.

$$V_1 \preceq V_2 \implies Freq_V(V_2) \leq Freq_V(V_1)$$

This implies that if a pattern is deemed frequent, then all its super-patterns are also frequent. Consequently, the frequency of a pattern is never greater than the frequency of any of its sub-patterns. The correctness of the counter measure computation is typically assessed through the anti-monotonicity property [Borgelt, 2009].

[3] Jure Leskovec, Stanford CS224W

Node structural anomaly

A node of the graph G is considered anomalous structurally if it has different connection patterns (unexpected neighborhood) according to the majority of normal node connection patterns [Ma et al., 2023].

Framework definition Projecting this concept on our framework PGF-AD, it can be defined by a node x that belongs to a **weak pattern** 4.1.2 i.e., $\exists G' = (V', E')$ graph of G such that

$$Strength_V(V') < T_S$$

Notes

- $Card(V')$ should be too small to represent the neighborhood connections of the node x.
- If V' is a weak pattern, then $\forall y \in V'$ is an anomaly and $\forall V'' \in [V']_{d_X}$ is also a weak pattern.
- In this case, the goal is to solve the optimization problem in 4.1.3 in order to discover the maximal normal connection patterns, then select the weak patterns that cannot be expanded to be more normal.

4.3 Minomaly: Node Structural Anomaly Miner

We present **Minomaly**, a Graph Neural Network (GNN) approach designed for the detection of Node Structural Anomalies in graphs. Minomaly employs the NeuroMatch method [Rex et al., 2020] inspired by [Vendrov et al., 2016] to encode pattern neighborhoods into an order embedding space. This encoding facilitates the frequency count of small structural anchored graph neighborhoods of nodes, providing a robust mechanism for anomaly detection.

Our approach bears similarities to the SPMiner decoder, the first neural approach to mining frequent subgraphs, which also relies on NeuroMatch for embedding encoding. However, while SPMiner [Ying et al., 2024] focuses on finding frequent motifs, Minomaly is designed to identify weak patterns of small infrequent motifs. Although SPMiner was successful in identifying most small frequent motifs, it lacked accuracy in predicting graph-level frequency. We exploit this discrepancy by focusing on the accurate identification of infrequent small graph substructures.

Minomaly introduces an effective greedy search method to expand weak neighboring patterns. These patterns are evaluated using a strength scoring system that we have proposed. Patterns that meet a certain threshold are identified as structural anomalies. The resulting graph represents the anomalous substructure, demonstrating the interpretability of our method.

A key feature of Minomaly is its ability to exploit the properties of the order embedding space. This allows it to select potential node anchors from embedded graphs that produce weak patterns, rather than verifying the normality of all nodes in a graph dataset. This strategic selection process enhances the computational complexity.

In comparison to existing non-interpretable deep learning methods, Minomaly demonstrates superior performance across all measures of the binary classifier, particularly in terms of the Area Under the Receiver Operating Characteristic (AUROC), Recall, and Precision. This underscores the robustness and effectiveness of Minomaly in the field of anomaly detection.

Our approach works in two stages, an embedding stage and a search stage:

- **Encoder** Embedding candidate subgraphs
- **Decoder** Anomalous neighborhoods search procedure

4.3.1 Minomaly Encoder

The primary function of the encoder is to break down the graph G into small subgraphs. These subgraphs are then processed through a Graph Neural Network (GNN) to generate their respective embeddings. This procedure enables to identify whether a given graph is a subgraph of another.

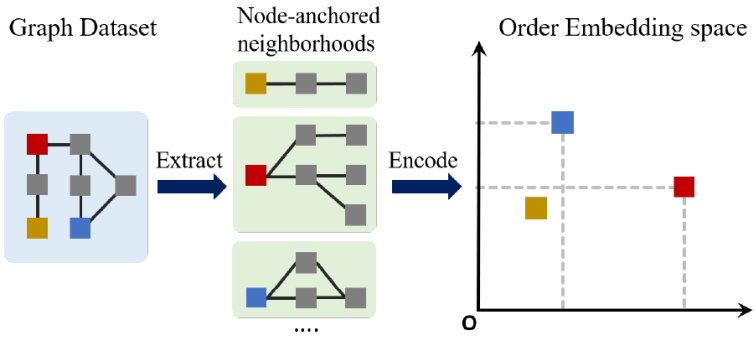


Figure 4.6: Minomaly encoding process[4]

Graph order space construction

Reminder Formally, given $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ graphs of G induced by V_1, V_2 resp. and ϕ_V a pattern order-embedding function defined in 4.1.4

$$V_1 \preceq_{\mathcal{R}} V_2 \iff \phi_V(V_1) \preceq \phi_V(V_2) \tag{4.1}$$

Goal The goal is to train GNN-based model to learn an order embedding function ϕ_V with respect to the definition 4.1.

Network architecture The network consists of a graph neural network (GNN) architecture, which combines a `SkipLastGNN` module for graph embedding with a multi-layer perceptron (MLP) for binary classification ($V_1 \preceq_{\mathcal{R}} V_2$). See figure 4.7.

1. **Input Layer**: The network takes two graphs, G_1 and G_2, as inputs.
2. **SkipLastGNN Module**:
   - **Preprocessing Layer**: Applies an initial linear transformation to the node features.
   - **Convolutional Layers**: The input passes through 8 layers of graph convolutions using the `SAGEConv` [Hamilton et al., 2018], with learnable skip connections.
   - **Post-processing Layer**: A sequence of linear transformations followed by non-linear activations (LeakyReLU) and dropout to produce the final embeddings.
3. **Embedding Outputs**: Produces embeddings $\phi_V(V_1)$ and $\phi_V(V_2)$.
4. **Concatenation**: The embeddings are concatenated into a single vector.
5. **MLP**: The concatenated embeddings pass through two linear layers with ReLU activations. The first layer has 256 units, and the second layer has 2 units, providing the final output.

[4] [Ying et al., 2024]

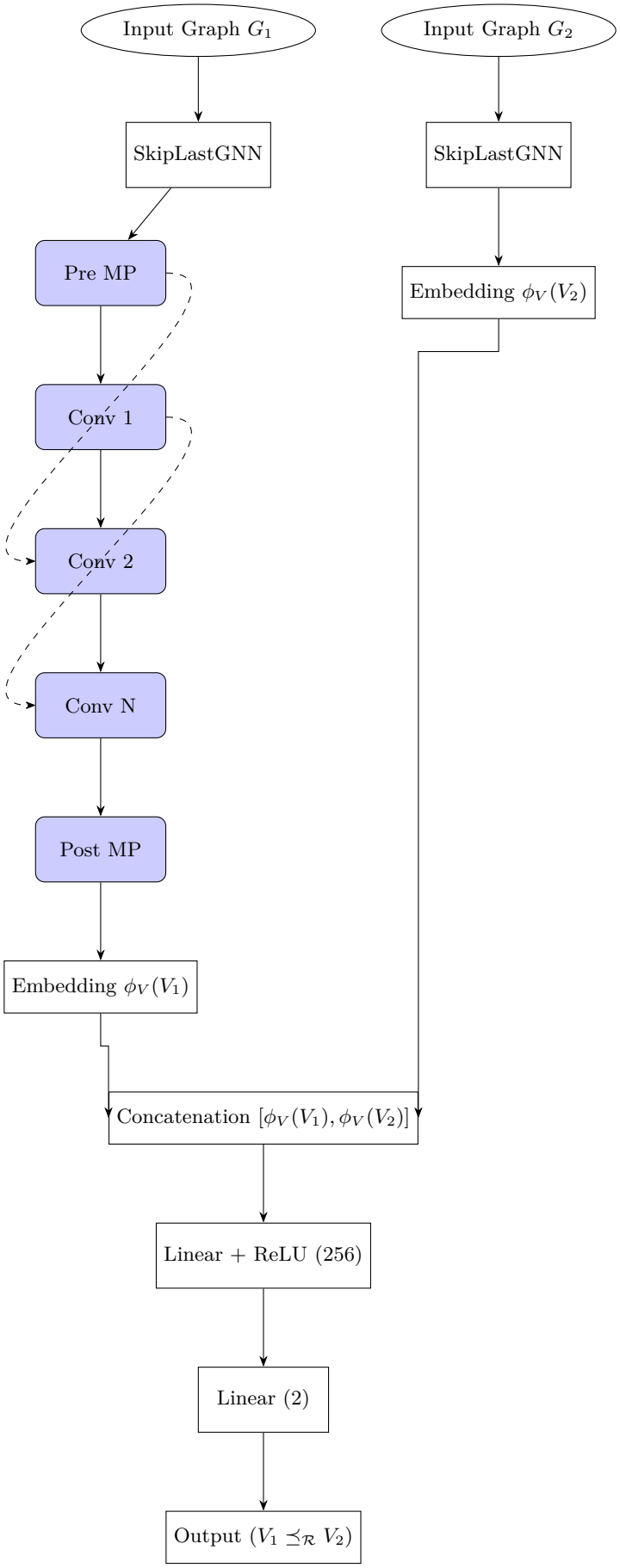


Figure 4.7: Minomaly encoding model architecture

Train data To train the Minomaly Network, we generate a set of positive and negative training instances. For positive examples, we sample random subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ from the graph G such that $V_1 \subseteq V_2$. For negative examples, we either add random nodes or edges to G_1 or sample random graphs G_1 and G_2.

Loss function We adopt the learning penalty proposed by [Ying et al., 2024], defined as

$$E(V_1, V_2) = \| \max(0, \phi_V(V_1) - \phi_V(V_2)) \|^2 \tag{4.2}$$

The total hinge loss is then given by:

$$\sum_{(V_1,V_2)\in \text{Positives}} E(V_1, V_2) + \sum_{(V_1',V_2')\in \text{Negatives}} \max\left(0, \alpha - E\left(V_1', V_2'\right)\right) \tag{4.3}$$

where α is the margin hyperparameter which controls the separation between penalties of positive and negative examples.

Consequently, for positives $V_1 \preceq V_2$, embeddings will be learned to be $\phi_V(V_1) \preceq \phi_V(V_2)$ because the penality $E(V_1, V_2)$ will be 0 in case of good learning, and large to be minimized during training. For negatives V_1' and V_2', the violation $E(V_1', V_2')$ is learned to be at least α for seperation from positives and in order to have minimized total loss.

Count prediction To determine the frequency of a pattern V_1, we count the occurrences of random embeddings V_2 in the space where $V_1 \preceq_{\mathcal{R}} V_2$, as predicted by the network's output layer.

Anti-monotonicity counting property preservation

As depicted in figure 4.8, we've visualized the learned order embeddings in a 2-D space. These embeddings represent random neighborhoods of varying sizes (from 1 to 30 nodes) from the Inj. Cora dataset.

We've used PCA visualization to maintain straight lines. This characteristic suggests that all points, which are subgraphs of a given point, are contained within a quadrilateral region.

The graphs are color-coded based on the number of nodes they contain: lighter shades for smaller graphs and darker shades for larger ones.

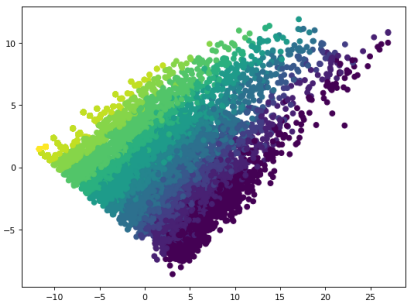


Figure 4.8: PCA visualization of graph size-based order embeddings

Interestingly, the model learns to structure the space according to graph size. This is logical since larger graphs cannot be subgraphs of smaller ones 4.1.4. This observation also indicates that the model preserves the anti-monotonicity of pattern counting 4.2.1.

Furthermore, the model appears to discern some intriguing global structure, as evidenced by some points being significantly distant from the main sequence.

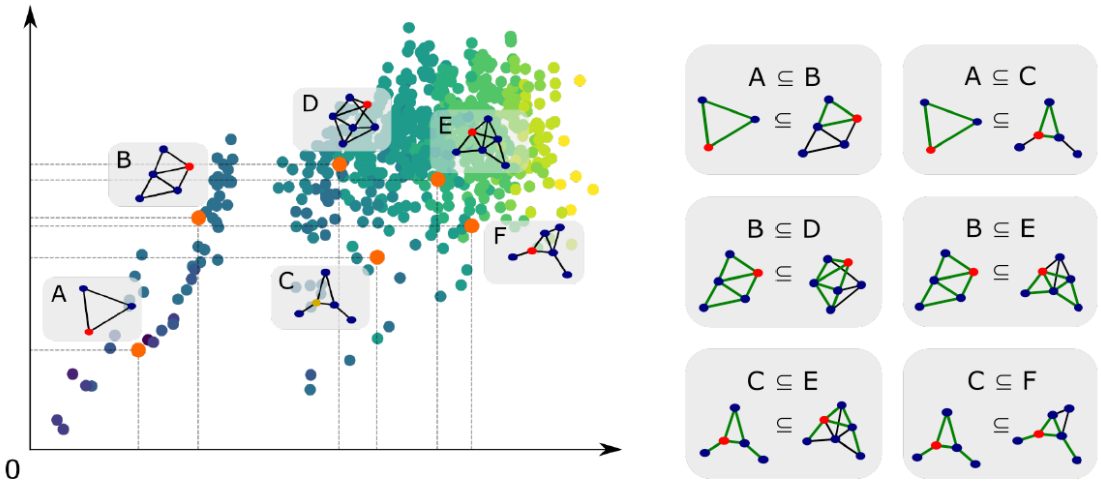


Figure 4.9: 2D visualization of the learned order-embedding space. Darker shades for smaller graphs and lighter shades for larger ones[5]

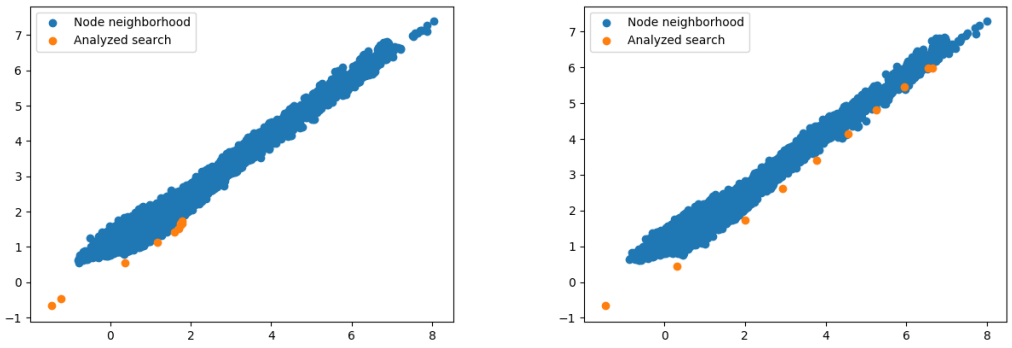
4.3.2 Minomaly Decoder

Given the encoder GNN ϕ_V that maps node-anchored graphs to order embeddings, the goal of the search procedure is to identify anomalous node-anchored neighborhoods that appear weak in the given graph dataset according to our theoretical definition 4.1.2 by minimizing their strength. Minomaly uses an approach of generating anomalous small substructures by iteratively adding nodes from potentially anomalous starting nodes.

Figure 4.9 shows that the learned order embedding by the decoder provides a well-behaved space, making the search process efficient and effective. This is because of the preserved anti-monotonicity property $V_1 \preceq V_2 \implies Freq_V(V_2) \leq Freq_V(V_1)$. Thus, each time a node is added to the pattern, the pattern frequency either decreases from a high value or remains stable due to the fact that very small patterns start from high frequency in normal situations because they are included in big patterns like single nodes patterns (see figure 4.10). In the other side, the strength of the pattern (refer to definition 4.1.2) decreases randomly from high value in general but decreases smoothly for weak patterns (see figure 4.11).

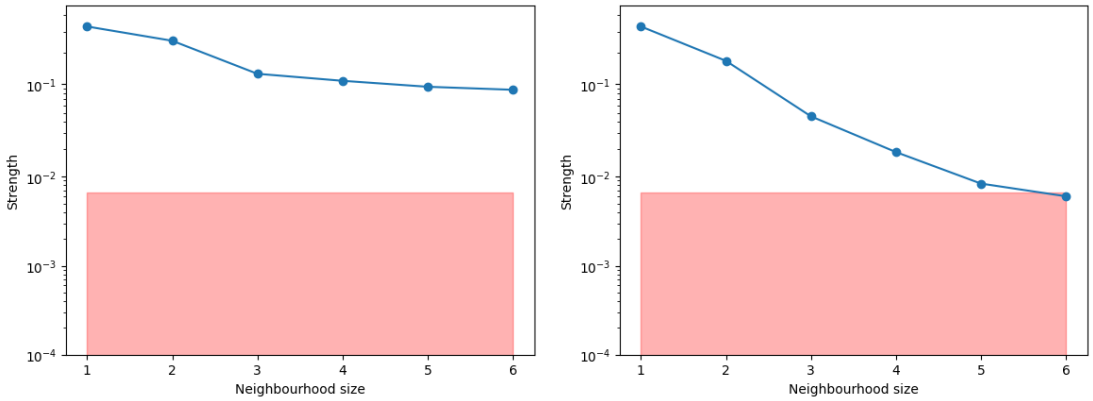
Minomaly adopts a **greedy search** for anomalous structures starting from a node, growing by adding the weakest candidate node from the neighborhood frontier of the current structure until a certain threshold of abnormality, T_S or the maximum weight, are reached (see the definition of node structural anomalies 4.2.1), or there is no improvement. As a result, the final small neighborhood will be identified as structurally anomalous according to its final reached strength.

[5] [Ying et al., 2024]



(a) Short monotonic walk of normal node (b) Long monotonic walk of anormal node

Figure 4.10: This figure illustrates the monotonic walk behavior of normal and anomalous neighborhoods in the ordered embedding space of Inj. Cora dataset. The long monotonic walk of an anormal node neighborhood is observed because it has few upper right neighborhoods. The short one of the normal node is due to the many upper right neighborhoods of the grown pattern.



(a) Normal node (b) Anomalous node

Figure 4.11: This figure illustrates the analysed strength of monotonic walk behavior of normal and anomalous neighborhoods in the ordered embedding space of Inj. Cora dataset. It shows strength changes with respect to their weights. 4.11b is anomalous because its grown neighborhood strength reached the threshold of abnormality T_S shaded by red. 4.11a is normal because its grown neighborhood size reached the maximum weight without reaching T_S.

Algorithm 1 Anomalous Node-Anchored Neighborhood Search

Require: Encoder GNN ϕ_V, graph dataset G, embeddings *embs*, node v, parameters α, T_S, *max_no_improve*, *max_steps*
Ensure: Anomalous node-anchored neighborhoods
Initialize counts and verified structures
Initialize search with node v and create initial BeamSet
Initialize step counter $steps = 1$
while search not done **do**
    Generate candidate nodes to expand current structure
    Compute strength scores for each candidate using ϕ_V
    Prune candidates based on T_S and *max_no_improve*
    Sort candidates by strength scores and select top *n_beams*
    Extract verified anomalous structures
    if add_verified_neighs is True **then**
        Add verified neighbors to the set of verified structures
    end if
    Update current BeamSet with new candidates
    Increment *steps*
    if $steps \geq max\_steps$ **then**
        break
    end if
end while
return Verified anomalous structures

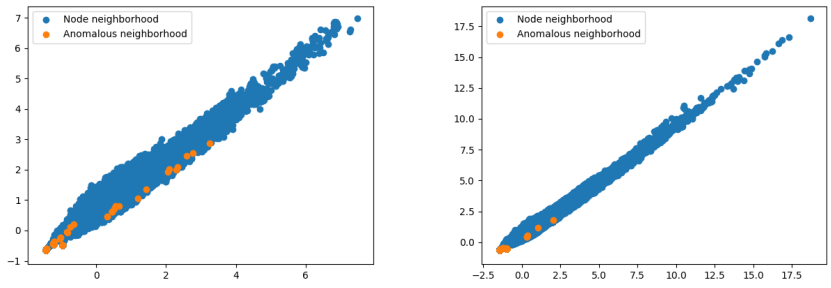
4.3.3 Experimental evaluation

Datasets

To evaluate our Minomaly approach, we use synthetic datasets provided by the tool [Liu et al., 2024] within the benchmark [Liu et al., 2022b], where structural anomalies are injected following the methodology of [Ding et al., 2019a]. These synthetic outliers are combined with real datasets, such as Cora [Sen et al., 2008] and Amazon [Shchur et al., 2018], which lack inherent outliers. Additionally, we use these injections due to the scarcity of datasets that clearly differentiate between structural and contextual outliers.

Dataset	Nodes	Edges	Avg. Degree	Structural Outliers	Injected %
Inj. Cora	2,708	11,060	4.1	70	5.1%
Inj. Amazon	13,752	515,042	37.2	350	5.0%

Table 4.1: Details of the injected datasets



(a) Inj. Cora (b) Inj. Amazon

Figure 4.12: This figure illustrates the true injected anomalous neighborhoods of Cora and Amazon.

Results

We conducted our experiments using the same hardware and hyperparameter configuration as detailed in Section 3.3, utilizing Python 3.6, Torch 1.4.0, Torch Geometric 2.0.3, and DeepSNAP 0.1.2.

We performed hyperparameter search for Minomaly decoder with $\alpha \in \{0.1, 0.15, 0.2, 0.25, 0.3, 0.33, 0.5, 0.6\}$, achieving optimal results with $\alpha = 0.15$ both for the two datasets. 10.000 neighborhoods were embedded by the encoder of Inj. Cora, and 30.000 of Inj. Amazon. To reduce time complexity, we configured the decoder to start from anchor nodes of infrequent encoded-embedded neighborhoods and from outlier neighborhoods identified by Isolation Forest, rather than from all nodes. Notably, we achieved a recall of 98% when decoding from all nodes, but this approach was time and memory consuming.

The statistics of the anomalous substructures in Inj. Cora and Inj. Amazon from [Liu et al., 2022b], are:

- $m = 7$ and $n = 70$ for Inj. Cora,
- $m = 10$ and $n = 350$ for Inj. Amazon.

Therefore, the parameters of the algorithm 1 are $m = max\_steps$, $n = max\_freq$.

$$T_S = \alpha \times max\_freq + (1 - \alpha) \times max\_steps \tag{4.4}$$

$$T_S = \alpha \times n + (1 - \alpha) \times m \tag{4.5}$$

Thus, **the choice of decoder hyperparameters is interpretable**, unlike in other benchmarked deep learning algorithms. Statistically, we found $\alpha \approx 0.15$ or 0.1 to be optimal for all synthetic datasets we tested, setting the maximum potential neighborhood at $max\_steps = 10$. The parameter max_freq was the only unknown and should be known a priori for the target graph dataset. It corresponds to the ratio of outliers in the target dataset, which is typically close to 10% [Emmott et al., 2013].

The Minomaly encoder, pretrained on synthetic graphs, achieves 95% accuracy in determining whether one graph is a subgraph of another with linear time complexity. It was also used for decoding all the benchmarked datasets.

GUIDE [Yuan et al., 2024] employs a graph motif counting algorithm to extract structural features, which is time-consuming on the CPU. Although it shares the core approach of counting with Minomaly, its results are not included in the benchmark due to memory limitations.

	Inj. Cora	Inj. Amazon	AVG Rank
GCNAE	52.5 ± 0.0 Max: 52.5 Rank: 7.0	49.0 ± 0.0 Max: 49.0 Rank: 6.0	6.5
DOMINANT	93.1 ± 5.7 Max: 96.1 Rank: 3.0	91.3 ± 0.1 Max: 91.6 Rank: 2.0	2.5
DONE	89.9 ± 6.8 Max: 97.2 Rank: 5.0	85.1 ± 5.5 Max: 91.4 Rank: 5.0	5.0
ADONE	91.6 ± 4.5 Max: 96.8 Rank: 4.0	85.9 ± 3.2 Max: 90.9 Rank: 4.0	4.0
ANOMALYDAE	81.7 ± 11.8 Max: 96.1 Rank: 6.0	-	6.0
CONAD	94.1 ± 5.0 Max: 96.1 Rank: 1.0	91.4 ± 0.1 Max: 91.5 Rank: 1.0	1.0
MINOMALY	93.2 ± 0.0 Max: 93.2 Rank: 2.0	90.4 ± 0.0 Max: 90.4 Rank: 3.0	2.5

(a) AUROC

	Inj. Cora	Inj. Amazon	AVG Rank
GCNAE	3.1 ± 0.0 Max: 3.1 Rank: 7.0	2.5 ± 0.0 Max: 2.5 Rank: 6.0	6.5
DOMINANT	19.9 ± 6.5 Max: 24.2 Rank: 4.0	12.2 ± 0.1 Max: 12.5 Rank: 3.0	3.5
DONE	21.6 ± 16.2 Max: 59.6 Rank: 2.0	9.6 ± 4.3 Max: 24.7 Rank: 4.0	3.0
ADONE	16.2 ± 6.6 Max: 30.9 Rank: 5.0	8.6 ± 2.0 Max: 12.4 Rank: 5.0	5.0
ANOMALYDAE	11.8 ± 8.3 Max: 24.5 Rank: 6.0	-	6.0
CONAD	21.0 ± 4.7 Max: 24.1 Rank: 3.0	12.2 ± 0.1 Max: 12.5 Rank: 2.0	2.5
MINOMALY	71.5 ± 0.0 Max: 71.5 Rank: 1.0	58.9 ± 0.0 Max: 58.9 Rank: 1.0	1.0

(b) Average Precision

	Inj. Cora	Inj. Amazon	AVG Rank
GCNAE	3.3 ± 0.0 Max: 3.3 Rank: 7.0	2.3 ± 0.0 Max: 2.3 Rank: 6.0	6.5
DOMINANT	20.1 ± 7.0 Max: 24.0 Rank: 3.0	13.4 ± 0.2 Max: 13.7 Rank: 3.0	3.0
DONE	13.4 ± 7.1 Max: 23.2 Rank: 5.0	9.1 ± 3.2 Max: 13.8 Rank: 4.0	4.5
ADONE	16.7 ± 6.8 Max: 24.7 Rank: 4.0	8.1 ± 2.5 Max: 13.5 Rank: 5.0	4.5
ANOMALYDAE	10.5 ± 8.8 Max: 24.0 Rank: 6.0	-	6.0
CONAD	21.6 ± 4.7 Max: 24.0 Rank: 2.0	13.5 ± 0.1 Max: 13.7 Rank: 2.0	2.0
MINOMALY	83.6 ± 0.0 Max: 83.6 Rank: 1.0	70.2 ± 0.0 Max: 70.2 Rank: 1.0	1.0

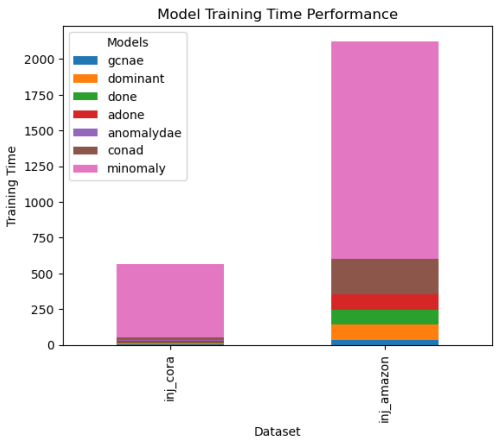
(c) Precision

	Inj. Cora	Inj. Amazon	AVG Rank
GCNAE	12.9 ± 0.0 Max: 12.9 Rank: 7.0	9.1 ± 0.0 Max: 9.1 Rank: 6.0	6.5
DOMINANT	77.8 ± 27.0 Max: 92.9 Rank: 3.0	52.8 ± 0.7 Max: 53.7 Rank: 3.0	3.0
DONE	51.7 ± 27.3 Max: 90.0 Rank: 5.0	35.6 ± 12.7 Max: 54.3 Rank: 4.0	4.5
ADONE	64.7 ± 26.5 Max: 95.7 Rank: 4.0	32.0 ± 9.9 Max: 53.1 Rank: 5.0	4.5
ANOMALYDAE	40.6 ± 34.2 Max: 92.9 Rank: 6.0	-	6.0
CONAD	83.8 ± 18.2 Max: 92.9 Rank: 2.0	53.1 ± 0.3 Max: 53.7 Rank: 2.0	2.0
MINOMALY	87.1 ± 0.0 Max: 87.1 Rank: 1.0	82.0 ± 0.0 Max: 82.0 Rank: 1.0	1.0

(d) Recall

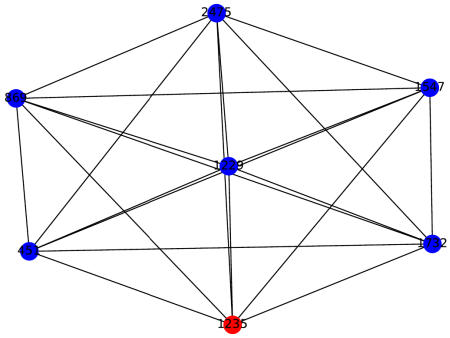
	Inj. Cora	Inj. Amazon	AVG Rank
GCNAE	5.3 ± 0.0 Max: 5.3 Rank: 7.0	3.7 ± 0.0 Max: 3.7 Rank: 6.0	6.5
DOMINANT	31.9 ± 11.1 Max: 38.1 Rank: 3.0	21.4 ± 0.3 Max: 21.8 Rank: 3.0	3.0
DONE	21.2 ± 11.2 Max: 37.0 Rank: 5.0	14.5 ± 5.2 Max: 22.0 Rank: 4.0	4.5
ADONE	26.6 ± 10.9 Max: 39.3 Rank: 4.0	13.0 ± 4.0 Max: 21.6 Rank: 5.0	4.5
ANOMALYDAE	16.7 ± 14.0 Max: 38.1 Rank: 6.0	-	6.0
CONAD	34.4 ± 7.5 Max: 38.1 Rank: 2.0	21.5 ± 0.1 Max: 21.8 Rank: 2.0	2.0
MINOMALY	85.3 ± 0.0 Max: 85.3 Rank: 1.0	75.6 ± 0.0 Max: 75.6 Rank: 1.0	1.0

(e) F1 Score

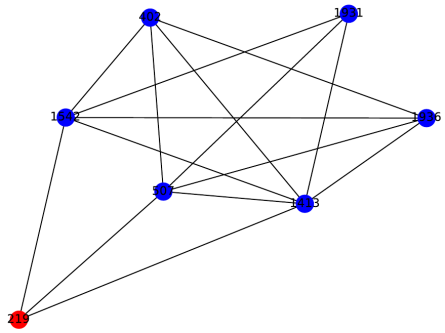


(f) Training time (seconds)

Figure 4.13: Results

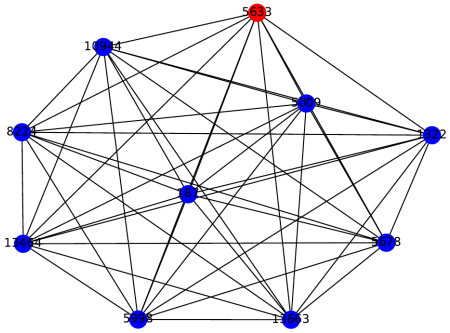


(a) Anomalous predicted clique
1235 is **True Positive**

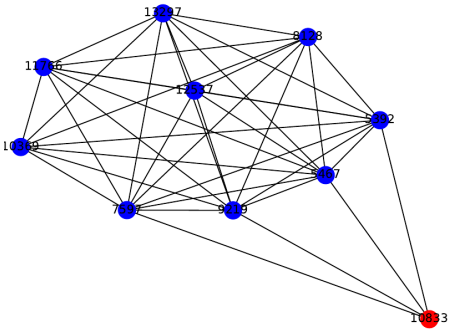


(b) Anomalous predicted node connected
to a clique
219 is **False Positive**

Figure 4.14: Discovered weak structural patterns of Inj. Cora by Minomaly decoder.

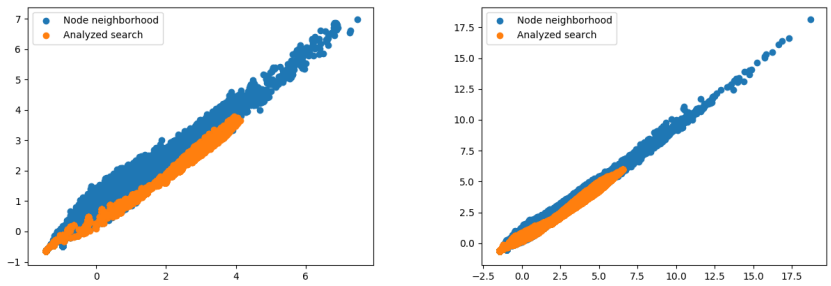


(a) Anomalous predicted clique
5633 is **True Positive**



(b) Anomalous predicted node connected
to a clique
10833 is **False Positive**

Figure 4.15: Discovered weak structural patterns of Inj. Amazon by Minomaly decoder.



(a) Inj. Cora (b) Inj. Amazon

Figure 4.16: This figure illustrates the decoder search of anomalous neighborhoods in the order embedding space of Inj. Cora and Inj. Amazon.

Discussion

Decoder Search Our Minomaly decoder has demonstrated its effectiveness in identifying anomalous regions by step-walking over the anomalous neighborhoods, as illustrated in figure 4.16 and figure 4.12. In contrast to existing deep-learning methods, Minomaly provides interpretability in its anomaly search results, which is lacking in other methods that rely on a probabilistic framework rather than a geometric one, as we have proposed for Minomaly.

Unlike other methods reviewed in [Lamichhane and Eberle, 2024] and [Ma et al., 2023], Minomaly does not rely on prior assumptions about how anomalous structures form in the graph or use any predefined anomaly probabilistic distribution. This approach allows Minomaly to be more flexible and applicable to a wider range of datasets to search new unexpected patterns.

Evaluation The benchmark results for structural anomalies in graphs indicate the following observations:

1. **AUROC** 4.13a: Minomaly achieved an average rank of 2.5, with AUROC scores of 93.2% on Inj. Cora and 90.4% on Inj. Amazon. This performance is comparable to leading methods like DOMINANT and CONAD.

2. **Average Precision** 4.13b: Minomaly ranked first overall, with scores of 71.5% on Inj. Cora and 58.9% on Inj. Amazon. This suggests a strong capability in identifying relevant anomalous substructures within the graphs.

3. **Precision** 4.13c: Minomaly also led in this metric, achieving 83.6% on Inj. Cora and 70.2% on Inj. Amazon, indicating high precision in detecting true positive anomalies.

4. **Recall** 4.13d: With scores of 87.1% on Inj. Cora and 82.0% on Inj. Amazon, Minomaly ranked first, demonstrating its effectiveness in identifying a high proportion of actual anomalies. As mentioned, This recall was increased to 98% when decoding from all nodes, but this approach was time and memory consuming.

5. **F1 Score** 4.13e: Minomaly outperformed other methods with F1 scores of 85.3% on Inj. Cora and 75.6% on Inj. Amazon, reflecting a balanced performance in precision and recall.

Overall, Minomaly performed consistently well across different metrics and datasets. In comparison to other deep-learning methods, these results highlight Minomaly's effectiveness in detecting structural anomalies in graphs, indicating its potential utility in various anomaly detection tasks.

Training time Minomaly's higher training time suggests greater computational cost, potentially limiting its use in real-time anomaly detection. However, evaluating its performance metrics is crucial to determine if the trade-off is justified.

Interpretable results The method's geometric approach provides interpretability, which is a notable difference from the probabilistic frameworks used by other methods. This approach offers weak pattern detection, as discussed in 4.1.2, which allows users to understand why the mined pattern is weak due to its frequency or weight.

Unlike other related deep learning methods, which cannot explain why a node is structurally anomalous, the Minomaly decoder provides the neighborhood graph for each predicted anomalous node, as shown in figure 4.15. This feature enables users to understand and evaluate the reasons behind the anomaly detection.

We believe our approach achieved a **high precision** score because it identified the true injected clique anomalies in Cora 4.14a and Amazon 4.15a, and also correctly predicted that nodes connected to these synthetic outliers are anomalous, as shown in figures 4.14b and 4.15b. Therefore, these are not false positives.

Conclusion PGF-AD and Minomaly represent significant advancements in anomaly detection for graph data, addressing challenges posed by ambiguous anomaly definitions and complex graph structures. PGF-AD introduces a pattern-based geometric framework that considers both local and global graph properties, offering intuitive anomaly descriptions without predefined distributions. Minomaly extends this by leveraging deep learning to detect structural anomalies efficiently, demonstrating superior performance and interpretability compared to traditional methods. Together, these approaches pave the way for more accurate and adaptable anomaly detection in diverse graph applications.

Chapter 5

Conclusion and Future Work

In this thesis, we proposed Minomaly and the Pattern-based Geometric Framework for Anomaly Detection (PGF-AD), focusing on their application to detecting anomalous nodes in static attributed graphs. The results demonstrate the efficacy of our approach in identifying unexpected structural connections, highlighting the potential of our methods in various practical contexts. Our work has shown that integrating topological theories with deep learning can lead to more accurate and interpretable anomaly detection models. By addressing the challenges associated with the ambiguity in anomaly definitions and the complexity of graph data, we have provided a robust framework for distinguishing between normal and anomalous patterns.

Furthermore, we have developed a comprehensive benchmarking framework to systematically evaluate various unsupervised deep-learning methods for anomaly detection. This benchmark is crucial as it allows for a detailed analysis of the limitations and strengths of existing approaches. By providing a standardized evaluation method, our benchmark helps in identifying the areas where current models fall short, thereby guiding future improvements and innovations in the field.

In summary, the contributions of this thesis lay a strong foundation for future research. We have shown that combining deep learning with topological insights can improve the detection of complex anomalies, offering significant benefits in fields such as finance, healthcare, and cybersecurity. The robustness and adaptability of the Minomaly framework make it a promising tool for various anomaly detection tasks, ensuring its relevance in both academic and industrial applications.

Looking ahead, several areas are proposed for further research and development to enhance the capabilities and applicability of Minomaly and PGF-AD. First, theoretical enrichment is crucial. This involves developing new theorems and properties to strengthen the framework, thereby gaining deeper insights into the functionality and applicability of PGF-AD. Additionally, discovering new strength properties beyond frequency, size, and homogeneity will enhance the accuracy of the anomaly detection process. Proving the anti-monotonicity counting property is another important step, providing a solid mathematical basis for efficient counting.

Second, algorithmic improvements are essential. Investigating new methods for selecting starting nodes can significantly reduce time and computational complexity, thereby enhancing the scalability of Minomaly. Furthermore, integrating the Minomaly decoder into existing deep learning methods could improve their interpretability and reduce false positives.

Methodological advancements and framework integration also present further opportunities for development. Implementing a feature space anomaly detection method using k-nearest graph neighborhoods and Minomaly can help detect unexpected local structural properties of featured

samples. Extending the geometric approach to time series anomaly detection could open new avenues for application in various fields such as finance and sensor networks. Additionally, exploring the projection of traditional anomaly detection methods, such as one-class classification, density, quantile, and reconstruction methods, onto the Minomaly framework could provide a generalized theoretical view of all anomalies, enabling rigorous measurement of their foundations.

Adapting the Minomaly encoder for attributed graphs, where nodes and edges have additional attributes, will make the encoder applicable to a wider range of real-world datasets that include rich contextual information. Collecting and curating real datasets containing organically occurring structural node anomalies will allow for more rigorous testing and validation of Minomaly under realistic conditions.

Lastly, comprehensive benchmarking of additional existing deep learning and generative methods is essential. This will provide a clearer comparison of the Minomaly framework's performance against a broader spectrum of techniques, ensuring that the proposed methods are thoroughly evaluated and validated. By addressing these future research directions, we aim to further advance the field of anomaly detection and its practical applications, ultimately leading to more secure and efficient systems.

Appendices

Appendix A

Appendix: Benchmark

A.1 Synthetic and organic outliers

	Cora	Amazon	Weibo	Reddit	Disney	Books	Enron	AVG Rank
GCNAE	16.6 ± 0.0 Max: 16.6 Rank: 6.0	27.3 ± 0.0 Max: 27.3 Rank: 1.0	29.0 ± 3.0 Max: 32.5 Rank: 1.0	3.1 ± 0.0 Max: 3.1 Rank: 5.0	0.8 ± 2.4 Max: 7.7 Rank: 4.5	1.4 ± 0.9 Max: 3.5 Rank: 3.0	0.0 ± 0.0 Max: 0.0 Rank: 6.0	3.8
DOMINANT	22.3 ± 7.1 Max: 27.3 Rank: 3.0	16.0 ± 0.4 Max: 16.9 Rank: 5.0	25.5 ± 11.6 Max: 32.5 Rank: 5.0	4.0 ± 0.0 Max: 4.0 Rank: 4.0	1.2 ± 2.8 Max: 7.7 Rank: 3.0	1.4 ± 1.4 Max: 4.9 Rank: 4.0	0.1 ± 0.0 Max: 0.1 Rank: 3.5	3.9
DONE	20.4 ± 6.9 Max: 30.3 Rank: 4.0	21.3 ± 7.2 Max: 30.7 Rank: 2.0	27.6 ± 2.8 Max: 31.3 Rank: 3.0	4.0 ± 0.2 Max: 4.6 Rank: 2.0	0.8 ± 2.4 Max: 7.7 Rank: 4.5	1.0 ± 0.8 Max: 2.8 Rank: 6.0	0.1 ± 0.0 Max: 0.1 Rank: 5.0	3.8
ADONE	22.4 ± 6.1 Max: 29.5 Rank: 2.0	18.4 ± 6.3 Max: 30.4 Rank: 3.0	24.9 ± 6.0 Max: 32.0 Rank: 6.0	3.1 ± 1.3 Max: 4.1 Rank: 6.0	1.9 ± 3.4 Max: 7.7 Rank: 1.5	2.0 ± 0.9 Max: 3.5 Rank: 2.0	0.1 ± 0.0 Max: 0.1 Rank: 1.0	3.1
ANOMALYDAE	19.8 ± 4.3 Max: 26.9 Rank: 5.0	-	29.0 ± 4.4 Max: 33.4 Rank: 2.0	4.0 ± 0.1 Max: 4.1 Rank: 1.0	0.0 ± 0.0 Max: 0.0 Rank: 6.0	2.8 ± 2.4 Max: 6.4 Rank: 1.0	0.1 ± 0.0 Max: 0.1 Rank: 2.0	2.8
CONAD	24.3 ± 4.9 Max: 27.3 Rank: 1.0	16.1 ± 0.4 Max: 16.9 Rank: 4.0	26.4 ± 9.5 Max: 32.5 Rank: 4.0	4.0 ± 0.0 Max: 4.1 Rank: 3.0	1.9 ± 3.4 Max: 7.7 Rank: 1.5	1.4 ± 1.4 Max: 4.9 Rank: 5.0	0.1 ± 0.0 Max: 0.1 Rank: 3.5	3.1

Table A.1: Precision

	Cora	Amazon	Weibo	Reddit	Disney	Books	Enron	AVG Rank
GCNAE	32.6 ± 0.0 Max: 32.6 Rank: 6.0	54.2 ± 0.0 Max: 54.2 Rank: 1.0	70.4 ± 7.3 Max: 78.7 Rank: 1.0	9.3 ± 0.0 Max: 9.3 Rank: 6.0	1.7 ± 5.1 Max: 16.7 Rank: 4.5	7.1 ± 4.5 Max: 17.9 Rank: 3.0	0.0 ± 0.0 Max: 0.0 Rank: 6.0	3.9
DOMINANT	43.7 ± 13.9 Max: 53.6 Rank: 3.0	31.7 ± 0.8 Max: 33.6 Rank: 5.0	61.8 ± 28.0 Max: 78.7 Rank: 5.0	11.9 ± 0.1 Max: 12.0 Rank: 4.0	2.5 ± 6.1 Max: 16.7 Rank: 3.0	7.0 ± 7.4 Max: 25.0 Rank: 4.5	14.0 ± 9.4 Max: 20.0 Rank: 4.0	4.1
DONE	40.1 ± 13.6 Max: 59.4 Rank: 4.0	42.2 ± 14.2 Max: 61.0 Rank: 2.0	66.8 ± 6.9 Max: 75.8 Rank: 3.0	12.0 ± 0.6 Max: 13.9 Rank: 2.5	1.7 ± 5.1 Max: 16.7 Rank: 4.5	5.2 ± 4.1 Max: 14.3 Rank: 6.0	14.0 ± 9.4 Max: 20.0 Rank: 4.0	3.7
ADONE	44.0 ± 12.0 Max: 58.0 Rank: 2.0	36.6 ± 12.4 Max: 60.2 Rank: 3.0	60.3 ± 14.6 Max: 77.5 Rank: 6.0	9.3 ± 3.8 Max: 12.3 Rank: 5.0	4.2 ± 7.4 Max: 16.7 Rank: 1.5	10.2 ± 4.4 Max: 17.9 Rank: 2.0	17.0 ± 7.3 Max: 20.0 Rank: 1.0	2.9
ANOMALYDAE	38.8 ± 8.4 Max: 52.9 Rank: 5.0	-	70.3 ± 10.6 Max: 81.0 Rank: 2.0	12.1 ± 0.2 Max: 12.3 Rank: 1.0	0.0 ± 0.0 Max: 0.0 Rank: 6.0	13.0 ± 11.7 Max: 32.1 Rank: 1.0	16.0 ± 8.2 Max: 20.0 Rank: 2.0	2.8
CONAD	47.8 ± 9.6 Max: 53.6 Rank: 1.0	31.9 ± 0.8 Max: 33.6 Rank: 4.0	63.9 ± 23.0 Max: 78.7 Rank: 4.0	12.0 ± 0.1 Max: 12.3 Rank: 2.5	4.2 ± 7.4 Max: 16.7 Rank: 1.5	7.0 ± 7.2 Max: 25.0 Rank: 4.5	14.0 ± 9.4 Max: 20.0 Rank: 4.0	3.1

Table A.2: Recall

A.2 Structural and contextual outliers

	Inj. Cora			Inj. Amazon			AVG Rank		
	All	Contextual	Structural	All	Contextual	Structural	All	Ctx.	Str.
GCNAE	16.6 ± 0.0 Max: 16.6 Rank: 6.0	13.7 ± 0.0 Max: 13.7 Rank: 1.0	3.3 ± 0.0 Max: 3.3 Rank: 6.0	27.3 ± 0.0 Max: 27.3 Rank: 1.0	25.4 ± 0.0 Max: 25.4 Rank: 1.0	2.3 ± 0.0 Max: 2.3 Rank: 5.0	3.5	1.0	5.5
DOMINANT	22.3 ± 7.1 Max: 27.3 Rank: 3.0	2.8 ± 1.8 Max: 6.6 Rank: 6.0	20.1 ± 7.0 Max: 24.0 Rank: 2.0	16.0 ± 0.4 Max: 16.9 Rank: 5.0	2.9 ± 0.3 Max: 3.6 Rank: 5.0	13.4 ± 0.2 Max: 13.7 Rank: 2.0	4.0	5.5	2.0
DONE	20.4 ± 6.9 Max: 30.3 Rank: 4.0	7.7 ± 2.1 Max: 11.1 Rank: 3.0	13.4 ± 7.1 Max: 23.2 Rank: 4.0	21.3 ± 7.2 Max: 30.7 Rank: 2.0	12.6 ± 7.4 Max: 23.5 Rank: 2.0	9.1 ± 3.2 Max: 13.8 Rank: 3.0	3.0	2.5	3.5
ADONE	22.4 ± 6.1 Max: 29.5 Rank: 2.0	6.4 ± 2.1 Max: 10.7 Rank: 4.0	16.7 ± 6.8 Max: 24.7 Rank: 3.0	18.4 ± 6.3 Max: 30.4 Rank: 3.0	10.6 ± 6.1 Max: 23.7 Rank: 3.0	8.1 ± 2.5 Max: 13.5 Rank: 4.0	2.5	3.5	3.5
ANOMALYDAE	19.8 ± 4.3 Max: 26.9 Rank: 5.0	9.9 ± 4.6 Max: 13.7 Rank: 2.0	10.5 ± 8.8 Max: 24.0 Rank: 5.0	-	-	-	5.0	2.0	5.0
CONAD	24.3 ± 4.9 Max: 27.3 Rank: 1.0	3.4 ± 2.2 Max: 6.6 Rank: 5.0	21.6 ± 4.7 Max: 24.0 Rank: 1.0	16.1 ± 0.4 Max: 16.9 Rank: 4.0	2.9 ± 0.4 Max: 3.6 Rank: 4.0	13.5 ± 0.1 Max: 13.7 Rank: 1.0	2.5	4.5	1.0

Table A.3: Precision

	Inj. Cora			Inj. Amazon			AVG Rank		
	All	Contextual	Structural	All	Contextual	Structural	All	Ctx.	Str.
GCNAE	32.6 ± 0.0 Max: 32.6 Rank: 6.0	52.9 ± 0.0 Max: 52.9 Rank: 1.0	12.9 ± 0.0 Max: 12.9 Rank: 6.0	54.2 ± 0.0 Max: 54.2 Rank: 1.0	100.0 ± 0.0 Max: 100.0 Rank: 1.0	9.1 ± 0.0 Max: 9.1 Rank: 5.0	3.5	1.0	5.5
DOMINANT	43.7 ± 13.9 Max: 53.6 Rank: 3.0	10.8 ± 6.8 Max: 25.7 Rank: 6.0	77.8 ± 27.0 Max: 92.9 Rank: 2.0	31.7 ± 0.8 Max: 33.6 Rank: 5.0	11.2 ± 1.3 Max: 14.0 Rank: 5.0	52.8 ± 0.7 Max: 53.7 Rank: 2.0	4.0	5.5	2.0
DONE	40.1 ± 13.6 Max: 59.4 Rank: 4.0	30.0 ± 8.1 Max: 42.9 Rank: 3.0	51.7 ± 27.3 Max: 90.0 Rank: 4.0	42.2 ± 14.2 Max: 61.0 Rank: 2.0	49.4 ± 29.2 Max: 92.6 Rank: 2.0	35.6 ± 12.7 Max: 54.3 Rank: 3.0	3.0	2.5	3.5
ADONE	44.0 ± 12.0 Max: 58.0 Rank: 2.0	24.6 ± 8.1 Max: 41.4 Rank: 4.0	64.7 ± 26.5 Max: 95.7 Rank: 3.0	36.6 ± 12.4 Max: 60.2 Rank: 3.0	41.7 ± 23.9 Max: 93.1 Rank: 3.0	32.0 ± 9.9 Max: 53.1 Rank: 4.0	2.5	3.5	3.5
ANOMALYDAE	38.8 ± 8.4 Max: 52.9 Rank: 5.0	38.4 ± 17.8 Max: 52.9 Rank: 2.0	40.6 ± 34.2 Max: 92.9 Rank: 5.0	-	-	-	5.0	2.0	5.0
CONAD	47.8 ± 9.6 Max: 53.6 Rank: 1.0	13.1 ± 8.4 Max: 25.7 Rank: 5.0	83.8 ± 18.2 Max: 92.9 Rank: 1.0	31.9 ± 0.8 Max: 33.6 Rank: 4.0	11.3 ± 1.4 Max: 14.0 Rank: 4.0	53.1 ± 0.3 Max: 53.7 Rank: 1.0	2.5	4.5	1.0

Table A.4: Recall

Bibliography

[Ahmed et al., 2020] Ahmed, M., Seraj, R., and Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295.

[Akoglu et al., 2010] Akoglu, L., McGlohon, M., and Faloutsos, C. (2010). Oddball: Spotting anomalies in weighted graphs. In *Pacific-Asia Conf. Knowl. Discov. Data Mining*, pages 410–421. Springer.

[Bandyopadhyay et al., 2020] Bandyopadhyay, S., N, L., Vivek, S. V., and Murty, M. N. (2020). Outlier resistant unsupervised deep architectures for attributed network embedding. In *Proceedings of the 13th international conference on web search and data mining*, pages 25–33.

[Booth and Colbourn, 1979] Booth, K. S. and Colbourn, C. J. (1979). *Problems polynomially equivalent to graph isomorphism*. Computer Science Department, Univ.

[Borgelt, 2009] Borgelt, C. (2009). Graph mining: An overview. In *Proc. 19th GMA/GI Workshop Computational Intelligence*, pages 189–203.

[Bronstein et al., 2021] Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges.

[Chen et al., 2018] Chen, Z., Yeo, C. K., Lee, B. S., and Lau, C. T. (2018). Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium (WTS)*, pages 1–5. IEEE.

[Ding et al., 2019a] Ding, K., Li, J., Bhanushali, R., and Liu, H. (2019a). Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 594–602. SIAM.

[Ding et al., 2019b] Ding, K., Li, J., and Liu, H. (2019b). Interactive anomaly detection on attributed networks. In *Proc. ACM 12th Int. Conf. Web Search Data Mining*, pages 357–365.

[Ding et al., 2012] Ding, Q., Katenka, N., Barford, P., Kolaczyk, E. D., and Crovella, M. (2012). Intrusion as (anti)social communication: characterization and detection. In *Proc. ACM SIGKDD 18th Int. Conf. Knowl. Discov. Data Mining*, pages 886–894.

[Dou et al., 2021] Dou, Y., Shu, K., Xia, C., Yu, P. S., and Sun, L. (2021). User preference-aware fake news detection. *arXiv preprint arXiv:2104.12259*.

[Duan et al., 2020] Duan, D., Tong, L., Li, Y., Lu, J., Shi, L., and Zhang, C. (2020). Aane: Anomaly aware network embedding for anomalous link detection. In *Proc. IEEE Int. Conf. Data Mining*, pages 1002–1007.

[Emmott et al., 2013] Emmott, A. F., Das, S., Dietterich, T., Fern, A., and Wong, W.-K. (2013). Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, pages 16–21.

[Ester et al., 1996] Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. ACM Int. Conf. Knowl. Discov. Data Mining*, pages 226–231.

[Evangelou and Adams, 2020] Evangelou, M. and Adams, N. M. (2020). An anomaly detection framework for cyber-security data. *Computers & Security*, 97:101941.

[Fan et al., 2020] Fan, H., Zhang, F., and Li, Z. (2020). Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5685–5689. IEEE.

[Goodfellow et al., 2020] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

[Grossberg, 2013] Grossberg, S. (2013). Recurrent neural networks. *Scholarpedia*, 8(2):1888.

[Hamilton et al., 2018] Hamilton, W. L., Ying, R., and Leskovec, J. (2018). Inductive representation learning on large graphs.

[Hearst et al., 1998] Hearst, M., Dumais, S., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

[Hooi et al., 2016] Hooi, B., Song, H. A., Beutel, A., Shah, N., Shin, K., and Faloutsos, C. (2016). Fraudar: Bounding graph fraud in the face of camouflage. In *Proc. ACM SIGKDD 22nd Int. Conf. Knowl. Discov. Data Mining*, pages 895–904.

[Hu et al., 2016] Hu, R., Aggarwal, C. C., Ma, S., and Huai, J. (2016). An embedding approach to anomaly detection. In *Proc. IEEE 32nd Int. Conf. Data Eng.*, pages 385–396.

[Karadayi et al., 2020] Karadayi, Y., Aydin, M. N., and Öğrencí, A. S. (2020). Unsupervised anomaly detection in multivariate spatio-temporal data using deep learning: early detection of covid-19 outbreak in italy. *Ieee Access*, 8:164155–164177.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

[Kipf and Welling, 2017] Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learn. Represent.*

[Kwon et al., 2019] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., and Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. *Cluster Computing*, 22:949–961.

[Lamichhane and Eberle, 2024] Lamichhane, P. B. and Eberle, W. (2024). Anomaly detection in graph structured data: A survey.

[Liu et al., 2022a] Liu, G., Niu, Y., Zhao, W., Duan, Y., and Shu, J. (2022a). Data anomaly detection for structural health monitoring using a combination network of ganomaly and cnn. *Smart Struct. Syst*, 29(1):53–62.

[Liu et al., 2024] Liu, K., Dou, Y., Ding, X., Hu, X., Zhang, R., Peng, H., Sun, L., and Yu, P. S. (2024). PyGOD: A Python library for graph outlier detection. *Journal of Machine Learning Research*, 25(141):1–9.

[Liu et al., 2022b] Liu, K., Dou, Y., Zhao, Y., Ding, X., Hu, X., Zhang, R., Ding, K., Chen, C., Peng, H., Shu, K., Sun, L., Li, J., Chen, G. H., Jia, Z., and Yu, P. S. (2022b). BOND: Benchmarking unsupervised outlier node detection on static attributed graphs. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27021–27035. Curran Associates, Inc.

[Ma et al., 2023] Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., Xiong, H., and Akoglu, L. (2023). A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12012–12038.

[Maćkiewicz and Ratajczak, 1993] Maćkiewicz, A. and Ratajczak, W. (1993). Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342.

[McFee and Lanckriet, 2009] McFee, B. and Lanckriet, G. (2009). Partial order embedding with multiple kernels. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 721–728.

[Morales et al., 2021] Morales, P., Caceres, R. S., and Eliassi-Rad, T. (2021). Selective network discovery via deep reinforcement learning on embedded spaces. *Appl. Network Sci.*, 6(1):1–20.

[Ouyang et al., 2020] Ouyang, L., Zhang, Y., and Wang, Y. (2020). Unified graph embedding-based anomalous edge detection. In *Proc. Int. Joint Conf. Neural Netw.*, pages 1–8.

[Pestov, 2013] Pestov, V. (2013). Is the k-nn classifier in high dimensions affected by the curse of dimensionality? *Computers and Mathematics with Applications*, 65(10):1427–1437. Grasping Complexity.

[Rex et al., 2020] Rex, Ying, Lou, Z., You, J., Wen, C., Canedo, A., and Leskovec, J. (2020). Neural subgraph matching.

[Ruff et al., 2021] Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., and Muller, K.-R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795.

[Ruff et al., 2018] Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR.

[Sabuhi et al., 2021] Sabuhi, M., Zhou, M., Bezemer, C.-P., and Musilek, P. (2021). Applications of generative adversarial networks in anomaly detection: a systematic literature review. *Ieee Access*, 9:161003–161029.

[Salehinejad et al., 2017] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

[Sen et al., 2008] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.

[Shchur et al., 2018] Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

[Talagala et al., 2021] Talagala, P. D., Hyndman, R. J., and Smith-Miles, K. (2021). Anomaly detection in high-dimensional data. *Journal of Computational and Graphical Statistics*, 30(2):360–374.

[Tan et al., 2016] Tan, C., Shee, Y., Yap, B., and Adikan, F. M. (2016). Fiber bragg grating based sensing system: Early corrosion detection for structural health monitoring. *Sensors and Actuators A: Physical*, 246:123–128.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

[Veličković et al., 2017] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

[Vendrov et al., 2016] Vendrov, I., Kiros, R., Fidler, S., and Urtasun, R. (2016). Order-embeddings of images and language.

[Wang et al., 2019] Wang, J., Wen, R., Wu, C., Huang, Y., and Xion, J. (2019). Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Proc. Int. Conf. World Wide Web*, pages 310–316.

[Wu et al., 2020] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

[Xu et al., 2022] Xu, Z., Huang, X., Zhao, Y., Dong, Y., and Li, J. (2022). Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 444–457. Springer.

[Ying et al., 2024] Ying, R., Fu, T., Wang, A., You, J., Wang, Y., and Leskovec, J. (2024). Representation learning for frequent subgraph mining.

[Yu et al., 2018] Yu, W., Cheng, W., Aggarwal, C. C., Zhang, K., Chen, H., and Wang, W. (2018). Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proc. ACM SIGKDD 24th Int. Conf. Knowl. Discov. Data Mining*, pages 2672–2681.

[Yuan et al., 2024] Yuan, X., Zhou, N., Yu, S., Huang, H., Chen, Z., and Xia, F. (2024). Higher-order structure based anomaly detection on attributed networks.

[Zhang et al., 2019] Zhang, S., Tong, H., Xu, J., and Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23.

[Zhang et al., 2020] Zhang, S., Yin, H., Chen, T., Nguyen, Q. V. H., Huang, Z., and Cui, L. (2020). Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, pages 689–698.

[Zheng et al., 2019a] Zheng, L., Li, Z., Li, J., Li, Z., and Gao, J. (2019a). Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *Proc. Int. Joint Conf. Artif. Intell.*, pages 4419–4425.

[Zheng et al., 2019b] Zheng, P., Yuan, S., Wu, X., Li, J., and Lu, A. (2019b). One-class adversarial nets for fraud detection. In *Proc. AAAI Conf. Artif. Intell.*, volume 33, pages 1286–1293.

ملخّص

يُعَدُّ الكشف عن الشّذوذ أمرًا بالغ الأهمية في مجالات عدّة مثل الماليّة والرّعاية الصّحيّة والأمن السيبراني، حيث يُعزِّز تحديد الاتّصالات الهيكلية غير المتوقّعة من الأمان والكفاءة. تعرض هذه الأطروحة: **Minomaly**، وإطار العمل الهندسيّ القائم على الأنماط للكشف عن الشّذوذ (**PGF-AD**)، اللذان يجمعان بين النّظريّات الطّبولوجيّة والتّعلّم العميق لتحسين الكشف عن الشّذوذ في الرسوم البيانيّة السّاكنة. يُقيّم إطار المقارنة الشّامل لدينا بصرامة الأساليب المختلفة، وتظهر النّتائج فعاليّة الإمكانات المستقبليّة لتحسين أنموذجات الكشف عن الشّذوذ.

الكلمات المفتاحيّة: كشف الشّذوذ، الرّسوم البيانيّة، التّعلّم العميق، إطار قائم على الأنماط، Minomaly، PGF-AD

Abstract

Anomaly detection is vital in fields like finance, healthcare, and cybersecurity, where identifying unexpected structural connections enhances security and efficiency. This thesis presents Minomaly and the Pattern-based Geometric Framework for Anomaly Detection (PGF-AD), which combine topological theories with deep learning to improve anomaly detection in static graphs. Our comprehensive benchmarking framework rigorously evaluates various methods, and results show the efficacy and potential for future improvements in anomaly detection models.

Keywords: Anomaly Detection, Graphs, Deep Learning, Pattern-based Framework, PGF-AD, Minomaly

Résumé

La détection des anomalies est cruciale dans des domaines tels que la finance, la santé et la cybersécurité, où l'identification de connexions structurelles inattendues améliore la sécurité et l'efficacité. Cette thèse présente Minomaly et le Cadre Géométrique basé sur les Modèles pour la Détection des Anomalies (PGF-AD), qui combinent des théories topologiques avec l'apprentissage profond pour améliorer la détection des anomalies dans les graphes statiques. Notre cadre de benchmarking complet évalue rigoureusement diverses méthodes, et les résultats montrent l'efficacité et le potentiel pour des améliorations futures des modèles de détection des anomalies.

Mots-clés: Détection d'anomalies, Graphes, Apprentissage profond, Cadre théorique basé sur les motifs, PGF-AD, Minomaly