

# Deep Learning Frameworks for Anomaly Detection in Time Series and Graphs with Limited Labels

by

Jiazhen Chen

A thesis  
presented to the University of Waterloo  
in fulfillment of the  
thesis requirement for the degree of  
Doctor of Philosophy  
in  
Statistics

Waterloo, Ontario, Canada, 2025

© Jiazhen Chen 2025

### **Examining Committee Membership**

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:       Zeyu Zheng  
                                  Associate Professor  
                                  Dept. of Industrial Engineering and Operations Research  
                                  University of California, Berkeley

Internal Member:         Ali Ghodsi  
                                  Professor, Department of Statistics and Actuarial Science  
                                  University of Waterloo

                                  Shoja’eddin Chenouri  
                                  Professor, Department of Statistics and Actuarial Science  
                                  University of Waterloo

Internal-External Member: Justin Wan  
                                  Professor, David R. Cheriton School of Computer Science  
                                  University of Waterloo

Supervisor(s):           Mingbin Feng  
                                  Associate Professor, Dept. of Statistics and Actuarial Science  
                                  University of Waterloo

                                  Tony Wirjanto  
                                  Professor, Dept. of Statistics and Actuarial Science  
                                  University of Waterloo

### **Author's Declaration**

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Statement of Contributions

This thesis comprises five seemingly unrelated studies, presented in Chapters 2–6. I served as the lead author and primary contributor for each of these works. Specifically, I conceived the central research ideas, designed the methodologies, implemented the models, conducted the experiments, and drafted the manuscripts. My supervisors and co-authors provided critical guidance, technical feedback, and intellectual input throughout the research process. My contributions to these studies are summarized as follows:

- **Harnessing Contrastive Learning and Neural Transformation in Time Series Anomaly Detection (Chapter 2):** I developed the core idea, designed the framework, and led the experimental evaluations and the drafting of this chapter. This work has been accepted in the Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2025.
- **Multivariate Time Series Anomaly Detection via Dynamic Graph Forecasting (Chapter 3):** I proposed the dynamic graph forecasting approach, implemented the model, and carried out the experiments, and made the manuscript preparation for this chapter.
- **Prospective Multi-Graph Cohesion for Multivariate Time Series Anomaly Detection (Chapter 4):** I introduced the multi-graph cohesion strategy, conducted the experimental studies, and drafted this chapter. This research has been accepted in the Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM), 2025.
- **Towards Cross-Domain Few-Shot Graph Anomaly Detection (Chapter 5):** In collaboration with my co-authors—who helped identify the specific problem—I designed and implemented the CDFS-GAD framework, performed the evaluations, and wrote this chapter. This work has been accepted in the Proceedings of the IEEE International Conference on Data Mining (ICDM), 2024.
- **Semi-Supervised Anomaly Detection with Extremely Limited Labels in Dynamic Graphs (Chapter 6):** Together with my co-authors, who assisted in framing the research problem, I developed the EL2-DGAD framework, executed the experiments, and authored the manuscript. This work has been accepted in the Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA), 2025.

## Abstract

Anomaly detection involves identifying patterns or behaviors that substantially differ from normal instances in a dataset. It has a wide range of applications in diverse fields such as cybersecurity, manufacturing, finance, and e-commerce. However, real-world anomaly detection often grapples with two main challenges: label scarcity, as anomalies are rare and hard to label, and the complexity of data structures, which can involve intricate dependencies that require careful analysis.

In this thesis, we develop deep learning frameworks designed to work effectively for label-free or extremely limited labeling scenarios, with a focus on time series anomaly detection (TSAD) and graph anomaly detection (GAD). To overcome the issue of label scarcity, our initial work investigates unsupervised TSAD methods that extract meaningful patterns from abundant unlabeled data. Building on recent advances in contrastive learning from NLP and computer vision, we introduce the Contrastive Neural Transformation (CNT) framework. This approach integrates temporal contrastive learning with neural transformations to capture context-aware and discriminative patterns effectively. Moreover, the dual-loss formulation prevents representation collapse by avoiding reliance on negative samples—a common challenge in anomaly detection, where the majority of instances represent normal behavior. While capturing temporal context is essential, understanding inter-series relationships is equally important in multivariate TSAD. Anomalies may seem normal in isolation but reveal abnormal patterns when compared to other series. To address this, we introduce DyGraphAD, a dynamic graph-driven dual forecasting framework that models both intra- and inter-series dependencies through a combination of graph and time series forecasting tasks. This allows anomalies to be detected via significant forecasting errors in both the channel-wise and time-wise dimensions. To further enhance computational efficiency, we propose an alternative framework, termed Prospective Multi-Graph Cohesion (PMGC). PMGC leverages graph structure learning to model inter-series relationships in a task-specific manner, reducing computational load compared to manual sequential graph construction in DyGraphAD. Furthermore, it introduces a multi-graph cohesion mechanism to adaptively learn both long-term dependencies and diverse short-term relationships. A prospective graphing strategy is also introduced to encourage the model to capture concurrent inter-series relationships, reducing reliance solely on historical data.

Beyond TSAD, GAD is also critical due to its prevalence in numerous applications. Graphs provide structural information alongside node and edge attributes, and understanding the interplay between graph structure and attributes is essential for uncovering subtle anomalies not apparent when examining nodes or edges alone. Given that obtaining

labeled data is relatively more feasible in graphs than in time series data for experimental purposes, we focus on GAD settings with limited labeling, more reflecting practical real-world scenarios. Specifically, we make the first attempt to address GAD in cross-domain few-shot settings, aiming to detect anomalies in a sparsely labeled target graph by leveraging a related but distinct source graph. To handle domain shifts, our CDFS-GAD framework incorporates a domain-adaptive graph contrastive learning and a domain-specific prompt tuning, aiming to align the features across two domains while preserving unique characteristics tailored to each domain. A domain-adaptive hypersphere classification loss and a self-training phase are introduced to further refine predictions in the target domain exploiting the limited labeling information. In addition to static graphs, many real-world applications involve dynamic graph data, where both the structure and attributes evolve over time. This adds complexity to anomaly detection, as both temporal and structural variations must be accounted for. Moreover, obtaining sufficient labeled data remains challenging, and related-domain labeled data may not be available in certain scenarios. To tackle the two more practical issues, we propose the EL<sup>2</sup>-DGAD framework, specifically designed for detecting anomalies in dynamic graphs in extremely labeled conditions. This framework enhances model robustness through a transformer-based temporal graph encoder that captures evolving patterns from local and global perspectives. An ego-context hypersphere classification loss is further introduced to adjust the anomaly detection boundary contextually under limited supervision, supplemented by an ego-context contrasting module to improve generalization with unlabeled data.

Overall, this thesis tackles anomaly detection for two commonly used data types, addressing unsupervised, semi-supervised, and cross-domain few-shot scenarios to meet the demands of real-world applications. Our extensive experiments show that the proposed frameworks perform well against various benchmark datasets and competitive anomaly detection baselines.

## Acknowledgements

I would like to express my deepest gratitude to my supervisors, Prof. Mingbin Feng and Prof. Tony Wirjanto, for their continuous support and encouragement throughout my doctoral journey. Their guidance, both technical and practical, as well as their generous financial support, enabled me to pursue innovative ideas and overcome challenging problems. I would not have achieved the progress documented in this thesis without their expertise, insightful feedback, and constant motivation.

I would also like to thank my other coauthors for introducing me to the field of data mining—particularly in the graph domain and anomaly detection. Their expert guidance not only helped me identify promising research directions but also ignited my passion for exploring innovative solutions in these areas.

Finally, I would like to thank my husband for his steadfast support. Without him, I wouldn't have had the courage to pursue a PhD in the first place. His emotional support throughout this challenging journey has been invaluable. Additionally, his expertise in deep learning and our regular academic discussions have enriched my research.

## **Dedication**

This thesis is dedicated to my parents, whose endless love and support have made this journey possible.

# Table of Contents

|                                                    |          |
|----------------------------------------------------|----------|
| Examining Committee                                | ii       |
| Author’s Declaration                               | iii      |
| Statement of Contributions                         | iv       |
| Abstract                                           | v        |
| Acknowledgements                                   | vii      |
| Dedication                                         | viii     |
| List of Figures                                    | xiv      |
| List of Tables                                     | xvii     |
| <b>1 Introduction</b>                              | <b>1</b> |
| 1.1 Deep Learning Architectures . . . . .          | 7        |
| 1.1.1 Perceptron . . . . .                         | 7        |
| 1.1.2 Multi-layer Perceptron (MLP) . . . . .       | 8        |
| 1.1.3 Recurrent Neural Networks (RNNs) . . . . .   | 9        |
| 1.1.4 Convolution Neural Networks (CNNs) . . . . . | 11       |
| 1.1.5 Transformers . . . . .                       | 13       |

|          |                                                                                                    |           |
|----------|----------------------------------------------------------------------------------------------------|-----------|
| 1.1.6    | Graph Neural Networks (GNNs)                                                                       | 16        |
| 1.1.7    | Message Passing Framework                                                                          | 16        |
| 1.1.8    | Graph Convolution Networks (GCNs)                                                                  | 17        |
| 1.1.9    | Graph Attention Networks (GATs)                                                                    | 18        |
| 1.2      | Literature Review For Time Series and Graph Anomaly Detection                                      | 19        |
| 1.2.1    | Time Series Anomaly Detection                                                                      | 19        |
| 1.2.2    | Graph Anomaly Detection                                                                            | 22        |
| <b>2</b> | <b>Harnessing Contrastive Learning and Neural Transformation for Time Series Anomaly Detection</b> | <b>28</b> |
| 2.1      | Introduction                                                                                       | 28        |
| 2.2      | Related Works                                                                                      | 30        |
| 2.3      | Methodology                                                                                        | 31        |
| 2.3.1    | Problem Formulation                                                                                | 31        |
| 2.3.2    | Overview                                                                                           | 32        |
| 2.3.3    | Framework                                                                                          | 33        |
| 2.3.4    | Theoretical Analysis                                                                               | 35        |
| 2.4      | Experiments                                                                                        | 36        |
| 2.4.1    | Experimental Setting                                                                               | 36        |
| 2.4.2    | Comparison Study                                                                                   | 40        |
| 2.4.3    | Ablation Study                                                                                     | 40        |
| 2.4.4    | Analysis Of Training Losses                                                                        | 42        |
| 2.4.5    | Data Visualization                                                                                 | 42        |
| 2.5      | Conclusion                                                                                         | 43        |
| <b>3</b> | <b>Multivariate Time Series Anomaly Detection via Dynamic Graph Forecasting</b>                    | <b>44</b> |
| 3.1      | Introduction                                                                                       | 44        |
| 3.2      | Related Works                                                                                      | 47        |

|       |                                                                                        |           |
|-------|----------------------------------------------------------------------------------------|-----------|
| 3.3   | Methodology . . . . .                                                                  | 48        |
| 3.3.1 | Problem Formulation . . . . .                                                          | 48        |
| 3.3.2 | Overview . . . . .                                                                     | 48        |
| 3.3.3 | Construction of Dynamic Graphss . . . . .                                              | 49        |
| 3.3.4 | Graph Encoder Module . . . . .                                                         | 50        |
| 3.3.5 | Graph Forecasting Module . . . . .                                                     | 51        |
| 3.3.6 | Time Series Forecasting Module . . . . .                                               | 52        |
| 3.3.7 | Training Objective and Anomaly Score . . . . .                                         | 53        |
| 3.4   | Experiments . . . . .                                                                  | 53        |
| 3.4.1 | Experimental Setting . . . . .                                                         | 53        |
| 3.4.2 | Comparison Study . . . . .                                                             | 56        |
| 3.4.3 | Ablation Study . . . . .                                                               | 57        |
| 3.4.4 | Data Visualization . . . . .                                                           | 57        |
| 3.4.5 | Hyper-parameter Analysis . . . . .                                                     | 60        |
| 3.5   | Conclusion . . . . .                                                                   | 61        |
| 4     | <b>Prospective Multi-Graph Cohesion for Multivariate Time Series Anomaly Detection</b> | <b>63</b> |
| 4.1   | Introduction . . . . .                                                                 | 63        |
| 4.2   | Related Works . . . . .                                                                | 65        |
| 4.2.1 | Deep Learning-based Time Series Anomaly Detection . . . . .                            | 65        |
| 4.2.2 | Modeling Time Series via Graphs . . . . .                                              | 66        |
| 4.3   | Methodology . . . . .                                                                  | 67        |
| 4.3.1 | Problem Formulation . . . . .                                                          | 67        |
| 4.3.2 | Overview . . . . .                                                                     | 68        |
| 4.3.3 | Graph Structure Learning . . . . .                                                     | 69        |
| 4.3.4 | Prospective GNN-based Forecasting . . . . .                                            | 73        |
| 4.3.5 | Anomaly Score Calculation . . . . .                                                    | 74        |

|          |                                                                                          |            |
|----------|------------------------------------------------------------------------------------------|------------|
| 4.4      | Experiments . . . . .                                                                    | 75         |
| 4.4.1    | Experimental Setting . . . . .                                                           | 75         |
| 4.4.2    | Comparison Study . . . . .                                                               | 78         |
| 4.4.3    | Ablation Study . . . . .                                                                 | 79         |
| 4.4.4    | Sensitivity Analysis . . . . .                                                           | 80         |
| 4.4.5    | Visualization of Learned Graphs . . . . .                                                | 81         |
| 4.5      | Conclusion . . . . .                                                                     | 82         |
| <b>5</b> | <b>Towards Cross-domain Few-shot Graph Anomaly Detection</b>                             | <b>84</b>  |
| 5.1      | Introduction . . . . .                                                                   | 84         |
| 5.2      | Methodology . . . . .                                                                    | 87         |
| 5.2.1    | Problem Formulation . . . . .                                                            | 87         |
| 5.2.2    | Overview . . . . .                                                                       | 88         |
| 5.2.3    | Feature Learning and Adaptation . . . . .                                                | 89         |
| 5.2.4    | Model Training and Testing . . . . .                                                     | 91         |
| 5.3      | Experiments . . . . .                                                                    | 94         |
| 5.3.1    | Experimental Setting . . . . .                                                           | 94         |
| 5.3.2    | Comparison Study . . . . .                                                               | 96         |
| 5.3.3    | Evaluation under Different Shots . . . . .                                               | 97         |
| 5.3.4    | Ablation Study . . . . .                                                                 | 98         |
| 5.3.5    | Parameters Sensitivity . . . . .                                                         | 99         |
| 5.3.6    | t-SNE Visualization . . . . .                                                            | 101        |
| 5.4      | Conclusion . . . . .                                                                     | 101        |
| <b>6</b> | <b>Semi-supervised Anomaly Detection with Extremely Limited Labels in Dynamic Graphs</b> | <b>103</b> |
| 6.1      | Introduction . . . . .                                                                   | 103        |
| 6.2      | Methodology . . . . .                                                                    | 106        |

|          |                                  |            |
|----------|----------------------------------|------------|
| 6.2.1    | Problem Formulation . . . . .    | 106        |
| 6.2.2    | Overview . . . . .               | 106        |
| 6.2.3    | Graph Encoder . . . . .          | 108        |
| 6.2.4    | Model Training . . . . .         | 109        |
| 6.3      | Experiments . . . . .            | 111        |
| 6.3.1    | Experimental Setting . . . . .   | 111        |
| 6.3.2    | Comparison Study . . . . .       | 114        |
| 6.3.3    | Ablation Study . . . . .         | 115        |
| 6.3.4    | Parameters Sensitivity . . . . . | 116        |
| 6.3.5    | t-SNE Visualization . . . . .    | 117        |
| 6.4      | Conclusion . . . . .             | 118        |
| <b>7</b> | <b>Conclusion</b>                | <b>119</b> |
|          | <b>References</b>                | <b>123</b> |

# List of Figures

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |    |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | An example for a perception, which converts an input $x$ into an output $y$ through a non-linear transformation. . . . .                                                                                                                                                                                                                                                                                                                                                          | 7  |
| 1.2 | An example for a 3-layer MLP. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                             | 8  |
| 1.3 | RNN architecture: (a) shows an example of an RNN neural network with self-loop; (b) shows the expanded version of (a). . . . .                                                                                                                                                                                                                                                                                                                                                    | 10 |
| 1.4 | Operations within an LSTM cell. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                           | 11 |
| 1.5 | An example for operations in a standard CNN block: left side of the figure shows an example of a convolution operation applied on an input with a dimension of $4 \times 4 \times 1$ with a kernel of size $3 \times 3 \times 1 \times 1$ . Zero-padding is used to ensure the output maintains the same height and width as the input; tOn the right side, a max pooling layer is applied to the convolution output, utilizing a $2 \times 2$ kernel with a stride of 2. . . . . | 13 |
| 1.6 | An illustration of a two-layer convolution with a kernel size of 2 applied on top of a given input $X$ . In (a), the convolution has the dilation factor remaining constant at 1. whereas in (b), the convolution has the dilation factor growing at a rate of 2. . . . .                                                                                                                                                                                                         | 14 |
| 1.7 | An illustration of the multi-head attention block. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                        | 14 |
| 2.1 | An illustration of the CNT framework. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                     | 32 |
| 2.2 | An illustration of the feature encoder. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                   | 34 |
| 2.3 | Performance comparison on different model variants in terms of $F1^c$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                   | 41 |
| 2.4 | Loss value per epoch on Wadi and SWaT. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                    | 41 |
| 2.5 | The top row shows the original time series, while subsequent rows display the anomaly scores from CNT, DCL, and CCL. Anomalous regions are highlighted in yellow. . . . .                                                                                                                                                                                                                                                                                                         | 42 |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |     |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 3.1 | a) An event showing abnormal inter-series relations in the highlighted region;<br>b) Node deviation between two consecutive graphs from normal states, or<br>during a transition from a normal (abnormal) to abnormal (normal) state.                                                                                                                                                                                                                                                                                                                                           | 45  |
| 3.2 | An illustration of the DyGraphAD framework. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 49  |
| 3.3 | Performance comparison on different model variants in terms of $F1^c$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 58  |
| 3.4 | (a)&(b) are time series segments from SWaT and SMD. Anomaly scores and<br>thresholds from the graph forecasting, the time series forecasting, and the<br>combination of them are shown in the bottom three rows, respectively; (c) is<br>the anomaly scores from the graph-based models corresponding to the time<br>series segment in (a); (d) is the dynamic graphs constructed by DyGraphAD<br>or MSCRED around the highlighted anomaly in (a). . . . .                                                                                                                      | 59  |
| 3.5 | $F1^*$ against different parameters, while the other parameters fixed at their<br>default values. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 60  |
| 4.1 | An illustrative diagram for the Framework of PMGC. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 67  |
| 4.2 | Performance comparison on different model variants in terms of $F1^c$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 80  |
| 4.3 | Performance variation in terms of $F1^c$ under different hyper-parameters. . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 81  |
| 4.4 | The results of Figures (a), (b), and (c) are based on the SWaT dataset. . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 83  |
| 5.1 | A diagram illustrating the proposed CDFS-GAD framework. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 88  |
| 5.2 | Various $K$ -shot performance comparison with the state-of-the-art GAD meth-<br>ods in terms of AUC-ROC. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 98  |
| 5.3 | Ablation study under the one-shot task in terms of AUC-ROC and AUC-PR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 100 |
| 5.4 | Sensitivity analysis of the hyper-parameters $\alpha$ , $m$ , $\beta_1$ , and $\beta_2$ of CDFS-GAD<br>under the one-shot task in terms of AUC-ROC. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                     | 101 |
| 5.5 | A two-dimensional t-SNE visualization of node embeddings learned by three<br>high-performing graph anomaly detection (GAD) methods under the one-<br>shot setting. The top row (a-c) corresponds to the cross-domain transfer<br>YelpHTL→YelpRes, while the bottom row (d-f) corresponds to YelpHTL→Amazon.<br>Each point represents a node’s embedding, where blue denotes normal nodes<br>and orange denotes anomalous nodes. A clear separation between blue and<br>orange clusters indicates a stronger capability to distinguish anomalies from<br>normal samples. . . . . | 102 |

|     |                                                                                                                                        |     |
|-----|----------------------------------------------------------------------------------------------------------------------------------------|-----|
| 6.1 | An illustration of the architecture of EL <sup>2</sup> -DGAD. . . . .                                                                  | 107 |
| 6.2 | Sensitivity analysis of $\lambda$ and neighbors number when there is one labeled<br>anomaly available in the training dataset. . . . . | 117 |
| 6.3 | t-SNE Visualization on the Wikipedia dataset. . . . .                                                                                  | 118 |

# List of Tables

|     |                                                                                                                                                                                                                          |     |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 2.1 | Summary statistics for datasets . . . . .                                                                                                                                                                                | 37  |
| 2.2 | Performance comparison based on point-wise F1 ( $F1^p$ ), point-adjusted F1 ( $F1^*$ ) and composite F1 ( $F1^c$ ), with the best performance highlight in bold and second best underlined. . . . .                      | 40  |
| 3.1 | Summary statistics for datasets . . . . .                                                                                                                                                                                | 54  |
| 3.2 | Performance comparison based on point-wise F1 ( $F1^p$ ), point-adjusted F1 ( $F1^*$ ), and composite F1 ( $F1^c$ ), with the best performance highlight in bold and second best underlined. . . . .                     | 56  |
| 4.1 | Summary statistics for datasets . . . . .                                                                                                                                                                                | 76  |
| 4.2 | Performance comparison based on point-wise F1 ( $F1^p$ ), point-adjusted F1 ( $F1^*$ ), and composite F1 ( $F1^c$ ), with the best performance highlight in bold and second best underlined. . . . .                     | 78  |
| 5.1 | Summary statistics for datasets used in experiments. . . . .                                                                                                                                                             | 95  |
| 5.2 | one-shot performance comparison with the existing GAD methods in terms of AUC-ROC and AUC-PR. The best results are highlighted in bold. . . .                                                                            | 96  |
| 6.1 | Summary statistics for datasets. . . . .                                                                                                                                                                                 | 113 |
| 6.2 | Experiment comparison with the state-of-the-art GAD methods in terms of AUC. The best results are marked in blue. 1, 2, and 3 in the “labels” row represent the number of labeled anomalies in the training set. . . . . | 115 |
| 6.3 | Performance evaluation on ablation experiments in terms of AUC. <i>w/o</i> denotes without a certain module. . . . .                                                                                                     | 116 |

# Chapter 1

## Introduction

Anomaly detection is a fundamental task in data mining with wide-ranging applications across industries such as cybersecurity, manufacturing, finance, and e-commerce [4, 67, 21, 156, 202, 67]. In this context, anomalies typically refer to abnormal patterns in data, characterized by rare or infrequent deviations from expected or normal behavior [124]. Importantly, what constitutes “abnormal” is highly dependent on the specific context and application scenario. For instance, in time series data generated by sensors within industrial processes, an unusually high sensor reading that is not synchronized with related sensor measurements might indicate a hidden equipment fault. Similarly, in network data, such as social networks or financial transaction networks, an isolated node suddenly establishing numerous unexpected connections could signify potential security threats or fraudulent activities. While anomalies are relatively rare, their occurrences often signal critical issues, including impending system failures, security breaches, or fraudulent behavior. Therefore, detecting anomalies early and accurately is crucial for enabling timely interventions, preventing substantial losses, and maintaining system reliability and safety.

Despite its critical importance, anomaly detection faces significant challenges in real-world applications. One major obstacle is label insufficiency. Anomalies, by their nature, are rare and diverse, making it difficult to obtain sufficient labeled instances for effective model training [124]. Moreover, acquiring labeled anomaly data can be a difficult and resource-intensive process, often necessitating specialized expertise. The scarcity of such labeled data creates challenges for supervised learning techniques, which depend on abundant labeled examples to accurately identify patterns and achieve high performance [126].

Another challenge lies in the complexity of data structures encountered in real-world datasets, which often necessitates analyzing intricate relationships and dependencies among

data points. For instance, in industrial systems that rely on IoT sensors to monitor equipment and processes, the multivariate time series data generated are often high-dimensional and contain intricate dependencies. Anomalous events, such as equipment failures or cyberattacks, are often subtle and require contextual information from multiple sensors to be accurately detected [199]. For example, in a water treatment facility, an increase in water flow might seem normal on its own, but when combined with an abnormal tank level and a valve malfunction, it could indicate a serious issue like a leak. These anomalies only become apparent when analyzing the relationships between multiple sensor readings.

On the other hand, data in fields like e-commerce and finance often form natural graph structures, where entities—such as users, products, or financial accounts—are interconnected through complex relationships [21, 156]. Anomalies in these networks might not appear abnormal when viewed in isolation but become evident when considered within the broader context of their connections. For example, on an e-commerce platform, a user’s behavior might seem typical based on their individual actions. However, when viewed in relation to other users, they might be part of a coordinated group engaging in fraudulent activities like generating fake reviews. In financial networks, a single transaction may not seem suspicious on its own, but as part of a chain of interactions between accounts, it might indicate money laundering.

Given these challenges, this thesis aims to develop robust anomaly detection frameworks that address the aforementioned two challenges of label scarcity and complex data structures. Due to time and resource constraints, we primarily focus on multivariate time series and graph-structured data, which are prevalent across a wide range of applications. Moreover, both data types require detailed analysis of contextual information and intricate relationships. By leveraging recent advancements in deep learning, the proposed frameworks will adapt to the unique characteristics of these data structures, providing more practical and effective solutions for anomaly detection in complex environments.

To achieve the thesis objectives, we first focus on time series anomaly detection (TSAD), which refers to the process of identifying unusual patterns or deviations from expected behavior in time-dependent data [11]. Given the scarcity of labeled anomalies in real-world time series data, traditional supervised approaches struggle to generalize effectively. Unsupervised methods are thus often preferred, as they can capture the underlying patterns of normal behavior without relying on labeled anomalies [27]. In this context, representation learning plays a crucial role in distinguishing normal patterns from deviations.

**CNT: A Contrastive Neural Transformation Framework** In our first contribution, detailed in Chapter 2, we propose the Contrastive Neural Transformation (CNT) frame-

work, which leverages contrastive learning techniques to learn robust representations of normal behavior without relying on labeled anomalies. Contrastive learning trains models to distinguish between similar and dissimilar instances by pulling representations of similar pairs closer and pushing those of dissimilar pairs apart [121]. It has been shown to be highly effective for representation learning across domains such as computer vision and natural language processing, where it helps models learn discriminative features from large amounts of unlabeled data [62, 22, 143].

However, applying contrastive learning directly to anomaly detection presents several challenges. First, contrastive learning typically relies on negative sampling to avoid representation collapse. In anomaly detection, this is problematic, as most samples in the training dataset are “normal”, making it difficult to draw truly dissimilar negative samples. Second, traditional contrastive learning focuses on instance discrimination, which may overlook anomalies that are more easily detected when compared to their temporal context.

To address these challenges, CNT incorporates a window-based contrastive learning strategy fortified with learnable transformations. This dual approach captures temporal anomalies within local regions while simultaneously mitigating the representation collapse issue. Through extensive experiments on diverse real-world industrial datasets, CNT achieves superior performance in comparison to various baselines and model variants.

**DyGraphAD: Focusing on Modeling Inter-Series Dependencies** While CNT primarily focuses on learning the robust temporal patterns within individual time series, we recognized that anomalies can also arise from the relationships between different time series variables. In multivariate time series, anomalies are detected not only by deviations in individual temporal patterns but also by irregular changes in the relationships between series. Recent studies tend to model these inter-series dynamics through graphs, where each time series is represented as a node, and relationships between them are represented as edges [32, 58, 26]. Recent advancements in graph neural networks (GNNs) make it possible to encode these structures, which allows the model to learn each node’s representation by aggregating information from its connected neighbors [208]. However, existing graph-based methods predominantly focus on detecting anomalies through time series forecasting or reconstruction errors, emphasizing temporal deviations over inter-series anomalies.

In Chapter 3, we present DyGraphAD, a dynamic graph-driven dual forecasting framework aimed at addressing both intra-series and inter-series anomalies explicitly. DyGraphAD constructs sequential graphs from multivariate time series data to capture evolving inter-series relationships and employs a dynamic graph encoder to encode both short-

and long-term dependencies. Additionally, it leverages a dual-task forecasting approach, combining graph forecasting with time series forecasting to detect anomalies exhibiting abnormal patterns across both dimensions. Extensive empirical evaluations on real-world industrial datasets demonstrate the superiority of DyGraphAD compared to a variety of recent TSAD methods.

**PMGC: Enhancing Graph Learning with Improved Efficiency** Although DyGraphAD enhances anomaly detection by modeling inter-series relationships, the computational complexity of pre-constructing and processing dynamic graphs at each time step poses practical challenges. Recent studies have increasingly leveraged graph structure learning [210] to learn inter-variable relationships. By leveraging advances in GPU parallelization, they achieve enhanced computational efficiency while generating task-specific, data-driven graph representations. However, existing methods often rely on a single graph representation, which is insufficient for capturing the complex, diverse relationships inherent in multivariate time series. Specifically, a single instance-driven dynamic graph might not capture the diverse and multi-scale inter-series dependencies, and it may adapt too aggressively to recent changes, making it sensitive to instance-wise noise. On the other hand, a single static graph enforces long-term consistency but fails to adapt to transient fluctuations, potentially leading to false alarms in dynamic environments.

To tackle this issue, we propose the Prospective Multi-Graph Cohesion (PMGC) framework for multivariate TSAD in Chapter 4. PMGC exploits spatial correlations by integrating a long-term static graph with a series of short-term instance-wise dynamic graphs, regulated through a graph cohesion loss function. Our theoretical analysis shows that this loss function promotes diversity among dynamic graphs while aligning them with the stable long-term relationships encapsulated by the static graph. Additionally, we introduce a “prospective graphing” strategy to mitigate the limitations of traditional forecasting-based TSAD methods, which often struggle with unpredictable future variations. This strategy allows the model to accurately reflect concurrent inter-series relationships under normal conditions, thereby enhancing anomaly detection efficacy. Empirical evaluations on real-world datasets demonstrate the superior performance of our method compared to existing TSAD techniques.

So far, our focus has been on datasets where graph structures need to be inferred or learned. However, in many real-world cases, predefined graph structures exist, such as social networks [98], financial transaction networks [21], or communication systems [113]. This shifts our focus to graph anomaly detection (GAD), which involves identifying unusual nodes, edges, or subgraphs that deviate from typical network patterns. As with time series

anomaly detection, contextual information is crucial for GAD, as anomalies often arise within the broader network context.

**CDFS-GAD: Addressing the Cross-Domain Few-Shot GAD scenario** Most existing GAD methods are unsupervised, which makes them useful for detecting anomalies without requiring labeled data. However, these methods are often sensitive to noise, leading to false positives. For instance, in financial transaction networks, minor fluctuations in transaction amounts due to regular market activity could be misclassified as anomalies, even though they fall within normal behavioral patterns. In practical applications, collecting a few labeled examples of anomalies is often feasible and can greatly improve detection accuracy. Few-shot GAD has recently gained attention, which aims to identify anomalous patterns with the guidance of a handful of labeled anomalies. Existing few-shot GAD approaches typically adopt meta-training methods trained on richly labeled auxiliary networks to facilitate rapid adaptation to target networks that possess sparse labels. However, these proposed methods often assume that the auxiliary and target networks exist in the same data distributions—an assumption rarely holds in practical settings.

In chapter 5, we explore a more prevalent and complex scenario of cross-domain few-shot GAD, where the goal is to identify anomalies within sparsely labeled target graphs using auxiliary graphs from a related, yet distinct domain. The challenge here is nontrivial owing to inherent data distribution discrepancies between the source and target domains, compounded by the uncertainties of sparse labeling in the target domain. To address these challenges, we propose the Cross-Domain Few-Shot GAD framework (CDFS-GAD), specifically designed to tackle the aforementioned challenges. CDFS-GAD first introduces a domain-adaptive graph contrastive learning module to enhance cross-domain feature alignment. Then, a prompt tuning module is designed to extract domain-specific features tailored to each domain. Moreover, a domain-adaptive hypersphere classification loss is proposed to enhance the discrimination between normal and anomalous instances under minimal supervision, utilizing domain-sensitive norms. Lastly, a self-training strategy is introduced to further refine the predicted scores, enhancing reliability in few-shot settings. Extensive experiments on twelve real-world cross-domain data pairs demonstrate the effectiveness of the proposed CDFS-GAD framework in comparison to various existing GAD methods, including unsupervised, semi-supervised, few-shot, and cross-domain GAD methods.

**EL<sup>2</sup>-DGAD: Tackling the Dynamic GAD Problem with Limited Labels** While CDFS-GAD primarily focuses on anomaly detection in static graphs, many real-world sce-

narios involve dynamic graphs with continuously evolving structures. Additionally, there are practical situations where no source domain data is available, necessitating methods that can operate under extremely limited labeling conditions.

Semi-supervised GAD has recently emerged as a promising approach, aiming to detect anomalies using a moderate amount of labeled data alongside a larger volume of unlabeled data. However, existing methods may suffer significant performance degradation when the labeled data is extremely limited. Moreover, the majority of these approaches primarily address anomaly detection in static graphs, with limited consideration given to the continuous evolution of graph structures over time.

To address these challenges, we propose  $\text{EL}^2\text{-DGAD}$ , a novel framework designed for anomaly detection in dynamic graphs with minimal labeled data.  $\text{EL}^2\text{-DGAD}$  employs a transformer-based graph encoder to capture evolving graph structures beyond local neighborhoods. Additionally, an ego-context hypersphere classification loss enables the model to classify temporal interactions while maintaining a compact separation between normal samples and anomalies. To further enhance model generalization with unlabeled data, the framework incorporates an ego-context contrasting module. Extensive experiments across four datasets and multiple label rates demonstrate that  $\text{EL}^2\text{-DGAD}$  consistently outperforms existing GAD methods, highlighting its effectiveness under dynamic and low-label conditions.

As a summary, the thesis presents a series of frameworks that advance the field of anomaly detection by focusing specifically on the domain of time series and graph-structured data. Beginning with CNT framework (Chapter 2), we tackled unsupervised TSAD by employing contrastive learning to capture robust representations of normal behavior without reliance on labeled anomalies. Next, we proposed DyGraphAD (Chapter 3), which models inter-series dependencies in multivariate time series through dynamic graph-based methods, thereby addressing anomalies arising from interactions among multiple variables. Recognizing the computational limitations inherent in processing dynamic graphs, the PMGC framework (Chapter 4) was introduced to provide an efficient multi-graph learning strategy that leverages GPU parallelization for task-specific graph construction and more effective anomaly detection. Shifting the focus to GAD, we explored methods applicable to scenarios with predefined graph structures. The CDFS-GAD framework in Chapter 5 addresses the challenge of adapting to a target graph with limited labeled data by incorporating domain-adaptive learning techniques, facilitating effective anomaly detection across domains with different data distributions. Lastly, the  $\text{EL}^2\text{-DGAD}$  framework (Chapter 6) was proposed to tackle anomaly detection in dynamic graph settings, further advancing the GAD methods in evolving network structures under extremely limited conditions.

The following sections provide further context and background: Section 1.1 reviews the deep learning architectures employed throughout the thesis, while section 1.2 offers a literature review covering existing TSAD and GAD methods.

## 1.1 Deep Learning Architectures

This section provides an overview of the key deep learning architectures utilized throughout the thesis, including foundational models such as Perceptrons and Multi-Layer Perceptrons (MLPs), as well as more advanced architectures like Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Transformers, and Graph Neural Networks (GNNs). These architectures form the basis for the proposed frameworks, enabling the modeling of complex patterns in time series and graph-structured data for effective anomaly detection.

### 1.1.1 Perceptron

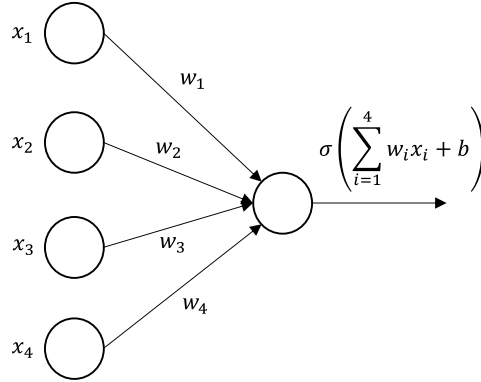


Figure 1.1: An example for a perception, which converts an input  $x$  into an output  $y$  through a non-linear transformation.

A deep neural network is composed of interconnected computational units known as perceptrons, which form the foundation of more complex architectures [52]. As shown in Figure 1.1, a perceptron takes an input vector  $x \in \mathbb{R}^m$  and outputs a scalar value  $y \in \mathbb{R}$ .

through the following operation:

$$\begin{aligned} y &= \sigma \left( \sum_{i=1}^m w_i x_i + b \right) \\ &= \sigma(W^T x + b) \end{aligned}$$

where  $w \in \mathbb{R}^m$  is a learnable weight vector, and  $b \in \mathbb{R}$  is a learnable bias coefficient respectively, and  $\sigma(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$  is an activation function. One simple implementation of  $\sigma(\cdot)$  is a logistic function, which is specified as

$$\sigma(x) = \frac{1}{1 + \exp(x)}$$

Another popular activation function is the ReLU function, which is denoted as

$$\sigma(x) = \max(0, x)$$

Without an activation function, the perceptron's output would be a linear transformation of the input. By incorporating non-linearity, neural networks can model complex relationships, enabling the stacking of multiple perceptrons to form deep networks. This foundational concept extends to more advanced architectures, which are discussed in the following sections.

### 1.1.2 Multi-layer Perceptron (MLP)

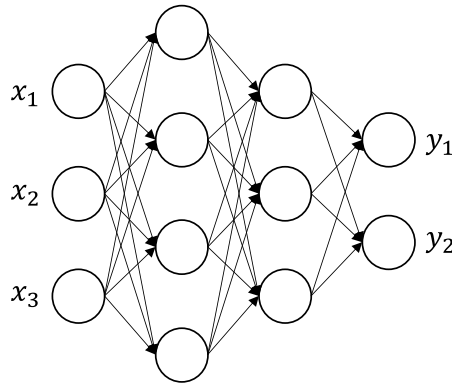


Figure 1.2: An example for a 3-layer MLP.

Building on the concept of the perceptron, a Multi-Layer Perceptron (MLP) extends this basic unit by stacking multiple layers of perceptrons, creating a deeper network [52]. In an MLP, the connections between layers are non-cyclic and fully connected, meaning that every perceptron in one layer is connected to all perceptrons in the preceding layer. This architecture is commonly referred to as a fully connected feedforward neural network, where “feedforward” indicates that the data flows in one direction—from input to output—without any feedback loops [93, 52].

As illustrated in Figure 1.2, a 3-layer MLP takes an input vector  $x \in \mathbb{R}^3$  and processes it through three hidden layers to produce an output vector  $y \in \mathbb{R}^2$ . The transformation at each layer can be expressed as

$$\begin{aligned} h_1 &= \sigma(W_1 x + b_1) \\ h_2 &= \sigma(W_2 h_1 + b_2) \\ y &= \sigma(W_3 h_2 + b_3) \end{aligned}$$

where  $(W_1 \in \mathbb{R}^{4 \times 3}, b_1 \in \mathbb{R}^4)$ ,  $(W_2 \in \mathbb{R}^{3 \times 4}, b_2 \in \mathbb{R}^3)$  and  $(W_3 \in \mathbb{R}^{2 \times 3}, b_3 \in \mathbb{R}^2)$  are learnable weight matrices and bias vectors for the three hidden layers, respectively.

This layered structure lays the foundation for more advanced architectures, which build on the principles of MLPs while incorporating additional mechanisms to better model sequential or structured data.

### 1.1.3 Recurrent Neural Networks (RNNs)

While Multi-Layer Perceptrons (MLPs) can effectively handle fixed-size inputs, they are not well-suited for sequential data, such as natural language or time series, which may have variable lengths [52]. Additionally, MLPs can lead to a large number of parameters when processing long sequences due to their fully connected architecture. Recurrent Neural Networks (RNNs) [112] address these limitations by introducing a network structure that is explicitly designed to model sequential patterns. RNNs maintain a memory cell that allows the network to incorporate information from previous time steps, thereby capturing temporal dependencies in the data. The key advantage of RNNs is parameter sharing across time steps, which significantly reduces the number of learnable parameters compared to MLPs [93].

As shown in Figure 1.3, an RNN processes an input sequence  $X \in \mathbb{R}^{T \times d}$ , where  $T$  is the number of time steps and  $d$  is the dimensionality of each input vector. At each time step  $t$ , the RNN takes the input vector  $x_t$  and the previous hidden state  $h^{t-1}$  to produce an

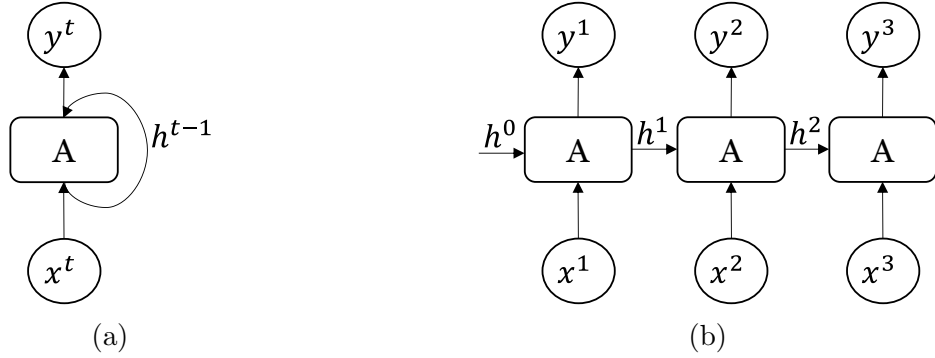


Figure 1.3: RNN architecture: (a) shows an example of an RNN neural network with self-loop; (b) shows the expanded version of (a).

updated hidden state  $h^t$  and an output vector  $y^t$ . The hidden state  $h^t$  encodes information about the past sequence, allowing the model to accumulate knowledge over time. The operations for RNNs are defined as

$$\begin{aligned} h^t &= f_u(x^t, h^{t-1}) \\ y^t &= f_o(h^t) \end{aligned}$$

where  $f_u$  and  $f_o$  are typically implemented as MLPs, and the weights of these functions are shared across all time steps. Figure 1.3b illustrates an expanded view of the RNN architecture.

Although standard RNNs are valuable, they are prone to issues such as vanishing and exploding gradients [119]. This issue arises when gradients from early time steps either decay or grow exponentially during backpropagation, particularly in long sequences. The root cause is the repeated multiplication of gradients over many time steps, which can lead to unstable training.

To mitigate this problem, several variants of RNNs have been developed, including Long Short-Term Memory (LSTM)[53] and Gated Recurrent Unit (GRU)[33]. These architectures introduce gating mechanisms to regulate the flow of information across time steps, enabling better handling of long-term dependencies. Given the similarities between GRUs and LSTMs, we will focus on LSTM in the following section.

## Long Short-Term Memory (LSTM)

Unlike the standard RNN, the LSTM architecture incorporates a memory cell along with three gates—input, forget, and output gates—to control the flow of information [53]. These

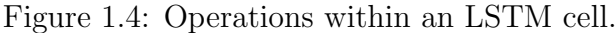


Figure 1.4 illustrates the operations within an LSTM cell, which are defined as follows:

where  $(W_o, b_o)$ ,  $(W_i, b_i)$  and  $(W_c, b_c)$  are the weights and biases for the output, input, and forget gates, respectively.

While Recurrent Neural Networks (RNNs) are effective for modeling sequential data, they may not be the most efficient choice for capturing local spatial patterns, such as those found in image data. Convolutional Neural Networks (CNNs) [100] are specifically designed to handle data with spatial structures, where features such as edges and textures play a significant role. Images typically have dimensions  $(h \times w \times d)$ , where  $h$ ,  $w$ , and  $d$

represent the height, width, and number of channels, respectively. When using MLPs to process images, the fully connected nature of the architecture leads to a large number of parameters, growing at a rate of  $O(hwd)$ , which becomes computationally impractical for high-resolution images [52].

CNNs overcome this challenge by introducing local connectivity and parameter sharing [93]. In a CNN, each neuron is connected only to a small region of the input, known as the receptive field, reducing the number of learnable parameters. Additionally, parameter sharing allows a set of filters to be applied across the entire image, enabling the detection of features such as edges and textures regardless of their position.

A typical CNN is composed of the following layers: convolutional layers, pooling layers, and fully connected layers [63]. Convolutional layers apply learnable kernels to the input data, generating feature maps that capture important spatial information. Given a 3D input  $X \in \mathbb{R}^{h \times w \times d}$ , the convolution operation uses a 4D kernel tensor  $K \in \mathbb{R}^{n \times m \times d \times d'}$  to produce a new output  $H \times \mathbb{R}^{h \times w \times d'}$ . This operation is defined as

$$H_{a,e,c} = \sigma \left( \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^d K_{i,j,k,c} \cdot X_{a+i-1,e+j-1,k} + b_c \right)$$

where  $\sigma(\cdot)$  is an activation function and  $b_c$  is a bias term. Zero-padding is often applied to maintain the input dimensions.

Pooling layers perform downsampling to reduce the dimensions of the feature maps, thereby decreasing computational complexity and making the model more robust to small shifts in the input [193]. For example, Figure 1.5 shows a max pooling operation with a  $2 \times 2$  kernel and stride 2 applied to the output from the convolutional layer. CNNs typically consist of repeated blocks of convolutional and pooling layers, followed by fully connected layers to extract relevant features for tasks such as classification or regression.

## Temporal Convolution Neural Networks (TCNs)

While CNNs are effective for spatial data, they can also be adapted to handle time series data by performing convolutions along the temporal dimension [120]. However, standard CNNs may struggle with long sequences due to their limited receptive field and the assumption that dependencies are symmetric in both directions [93]. Temporal Convolutional Networks (TCNs) [8] address these limitations by employing causal and dilated convolutions to model long-term dependencies more efficiently.

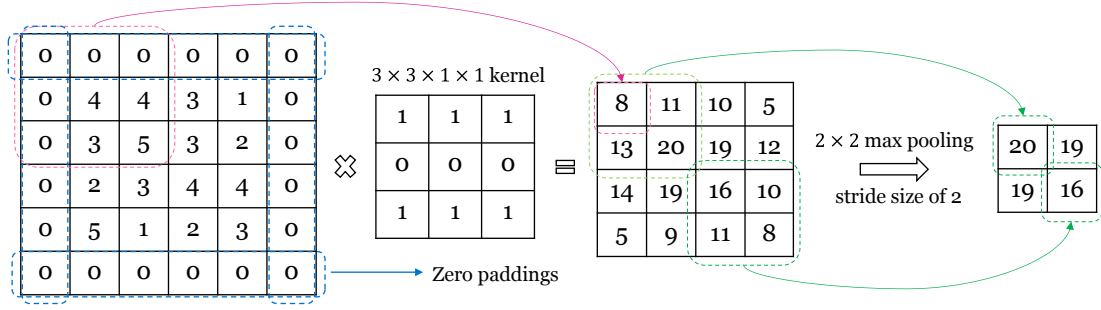


Figure 1.5: An example for operations in a standard CNN block: left side of the figure shows an example of a convolution operation applied on an input with a dimension of  $4 \times 4 \times 1$  with a kernel of size  $3 \times 3 \times 1 \times 1$ . Zero-padding is used to ensure the output maintains the same height and width as the input; On the right side, a max pooling layer is applied to the convolution output, utilizing a  $2 \times 2$  kernel with a stride of 2.

A causal convolution ensures that the output at time  $t$  depends only on historical observations, while a dilated convolution introduces gaps between filter elements, allowing the receptive field to expand without adding more layers [8]. Given an input  $X \in \mathbb{R}^{N \times T}$ , where  $T$  is the sequence length and  $N$  is the number of features, the output  $H \in \mathbb{R}^{C \times T'}$  from a dilated convolution is computed as

$$H_{c,t} = \sigma \left( \sum_{i=1}^N \sum_{j=1}^m K_{j,c} \cdot X_{i,t-d(j-1)} + b_c \right)$$

where  $K \in \mathbb{R}^{m \times C}$  is the kernel,  $d$  is the dilation factor, and  $T'$  is the length of the output sequence. As shown in Figure 1.6b, increasing the dilation factor exponentially allows TCNs to capture long-term dependencies more effectively than standard CNNs, which would require deeper networks.

### 1.1.5 Transformers

Transformers [167] have become a foundational architecture in deep learning, initially introduced for natural language processing (NLP) tasks but now widely adopted across various domains, including time series analysis and computer vision [39, 184]. The key innovation of Transformers lies in their self-attention mechanism, which enables the model to capture dependencies across different positions in the input sequence simultaneously [167]. Unlike

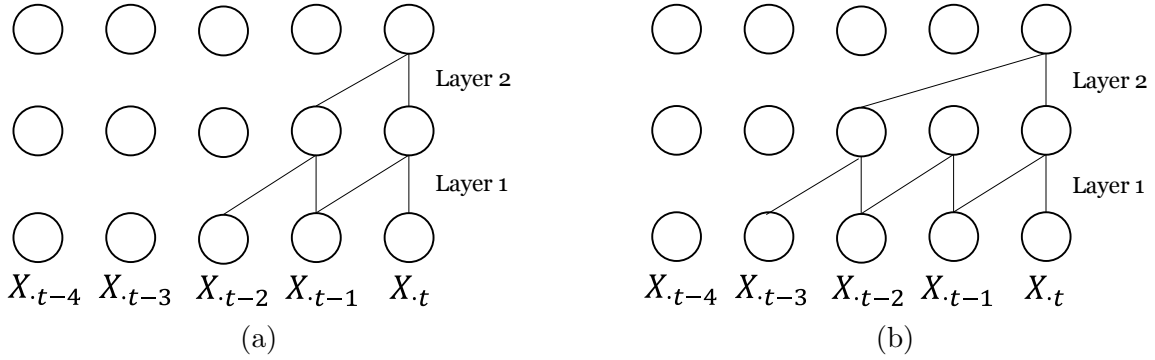


Figure 1.6: An illustration of a two-layer convolution with a kernel size of 2 applied on top of a given input  $X$ . In (a), the convolution has the dilation factor remaining constant at 1, whereas in (b), the convolution has the dilation factor growing at a rate of 2.

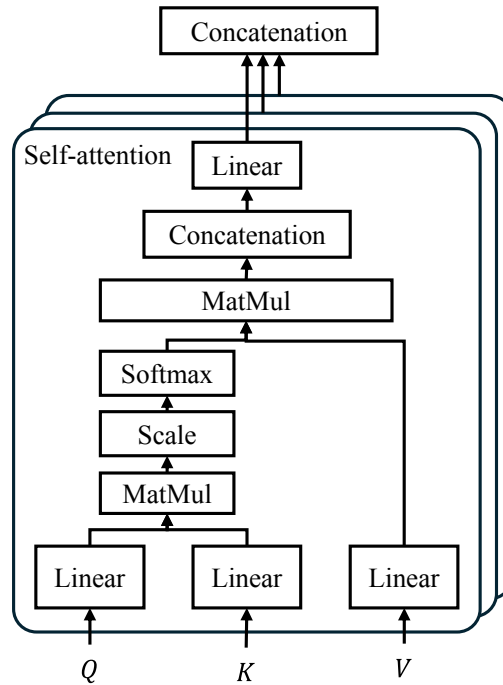


Figure 1.7: An illustration of the multi-head attention block.

traditional architectures such as RNNs, which process sequences step by step, Transformers can model long-range dependencies in parallel, making them well-suited for tasks that involve large datasets with complex dependencies.

At the core of the Transformer architecture is the self-attention mechanism, which allows the model to dynamically focus on different parts of the input sequence. Self-attention operates by computing a weighted sum of values, where the weights are determined based on the similarity between query and key vectors. Given an input sequence represented as  $X$ , self-attention computes the output as:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

Here,  $Q$ ,  $K$ , and  $V$  are the query, key, and value matrices derived from the input sequence, and  $d_k$  is the dimensionality of the key vectors. The softmax function normalizes the similarity scores, allowing the model to assign varying levels of importance to different elements in the sequence. Furthermore, to enhance the model’s ability to capture multiple aspects of the input data, Transformers employ multi-head self-attention. This technique runs several self-attention operations in parallel, each with different learned parameters, and then concatenates their outputs (See Figure 1.7 for an example of the multi-head attention block).

Since the Transformer architecture does not inherently capture the order of elements in a sequence, positional encoding is used to inject information about the position of each element into the input embeddings. The positional encoding proposed in [167] is implemented using sinusoidal functions:

$$\begin{aligned} \text{PE}_{(pos, 2i)} &= \sin \left( \frac{pos}{10000^{2i/d_{\text{model}}}} \right) \\ \text{PE}_{(pos, 2i+1)} &= \cos \left( \frac{pos}{10000^{2i/d_{\text{model}}}} \right) \end{aligned}$$

Here,  $pos$  represents the position, and  $i$  denotes the dimension index. These encodings are added to the input embeddings, allowing the model to distinguish between different positions.

The vanilla Transformer architecture presented in [167] consists of two main components: the encoder and the decoder. The encoder is composed of multiple identical layers, each containing a self-attention mechanism followed by a feedforward neural network. The decoder has a similar structure but includes an additional cross-attention layer that integrates information from the encoder, enabling the model to generate outputs based on the encoded input. Each layer in both the encoder and decoder uses residual connections [63] and layer normalization [95] to stabilize training. The feedforward network typically consists of two fully connected layers with a ReLU activation in between.

### 1.1.6 Graph Neural Networks (GNNs)

Graph neural networks (GNNs) are neural networks that are designed to model complex patterns in graph-structured data [14]. In contrast to data that has a grid-like or sequential format, such as image data or time series, a graph is represented as a collection of nodes interconnected through edges, where each node has an arbitrary number of neighbors without any temporal order. As a result, graph-structured data does not fit into the RNNs or CNNs framework. On the other hand, GNNs operate on graphs by representing each node as a vector and then use a series of neural network layers to recursively update the node representations based on information from neighboring nodes. This allows GNNs to learn intricate relationships and interactions between nodes and edges in the graphs, thereby enabling them to extract meaningful features from graph-structured data.

Over time, a variety of GNNs has emerged in the literature, including graph convolution network (GCN) [86], graph attention network (GAT) [168], and GraphSage [57]. The majority of them follow a message-passing framework, where individual differences among various GNNs lie in the formulation of each step inside the framework. In the following section, we provide an explanation of the message-passing framework used by GNNs.

### 1.1.7 Message Passing Framework

The message-passing scheme is at the heart of GNNs. The scheme involves nodes sending messages to their neighbors based on their current representation, with the messages being aggregated and combined to update the node representations [57].

More formally, let  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  be an input graph with a node feature matrix  $X \in \mathbb{R}^{|\mathcal{V}| \times d}$ , where  $\mathcal{V}$  is a set of nodes in the graph (indexed from 1 to  $|\mathcal{V}|$ ), and  $\mathcal{E}$  is a set of edges between the nodes. The objective of a GNN is to generate a node embedding  $z_u$  for each node  $u \in \mathcal{V}$ . Typically, a GNN network comprises multiple layers, with each layer incorporating information from multiple hops away into each node. The  $k^{th}$  layer of the GNN network can be defined as follows:

$$h_u^{(k)} = f_{\text{update}}^{(k)}(h_u^{(k-1)}, a_u^{(k)}) \quad (1.1)$$

$$a_u^{(k)} = f_{\text{aggregate}}^{(k)}(m_v^{(k-1)}, v \in \mathcal{N}_u) \quad (1.2)$$

$$m_v^{(k-1)} = f_{\text{message}}^{(k)}(h_v^{(k-1)}, h_u^{(k-1)}) \quad (1.3)$$

Here,  $\mathcal{N}_u$  denotes the 1-hop neighbor nodes of  $u$ ,  $f_{\text{update}}^{(k)}$  and  $f_{\text{message}}^{(k)}$  are differentiable functions (e.g., an MLP), and  $f_{\text{aggregate}}^{(k)}$  is a permutation invariant function (e.g., sum,

mean, or max operation). In the  $k^{th}$  layer, the node embedding  $h_v^{(k-1)}$  of neighbor node  $v$  is transformed into a “message” using  $f_{\text{message}}^{(k)}$ . The neighboring messages of  $u$  are then combined using  $f_{\text{aggregate}}^{(k)}$  to obtain  $a_u^{(k)}$ , which contains the full information from 1-hop neighbors of  $u$ . Finally, the  $f_{\text{update}}^{(k)}$  function combines the aggregated message  $a_u^{(k)}$  with the node embedding of  $u$  in the  $(k-1)^{th}$  layer to form a new feature representation of  $u$ . By stacking  $K$  layers of the operation defined from Eq. (1.1) to Eq. (1.3), the feature representation of each node can contain information from up to  $K$ -hop neighbors. Here, The  $K$ -hop neighbors refer to the nodes that are reachable within  $K$  steps or edges from a given node. As GNNs come in many variants, we mainly give an overview of the two most popular GNNs, graph convolution neural network (GCN) and graph attention network (GAT).

### 1.1.8 Graph Convolution Networks (GCNs)

Graph Convolutional Networks (GCNs) [86] are among the pioneering GNN architectures that generalize neural network designs for graph-structured data. The intuition behind GCN is to generalize the idea of convolutions from regular grids, such as images, to irregular and non-Euclidean structures, such as graphs. GCN achieves this by introducing the concept of message passing, where information is passed between neighboring nodes and aggregated to update the node’s features. This message-passing operation is inspired by the convolution operation in regular CNNs, but with some modifications to account for the irregularity and complexity of graph-structured data. Concretely, based on the formulation of the message passing framework, the message, aggregate and update operation at the  $k^{th}$  layer of a GCN network can be defined as

$$\begin{aligned} f_{\text{message}}^{(k)}(h_v^{(k-1)}, h_u^{(k-1)}) &= h_v^{(k-1)} W^{(k)} \\ f_{\text{aggregate}}^{(k)}(\{m_v^{(k-1)}, v \in \mathcal{N}_u\}) &= \sum_{v \in \mathcal{N}_u} \frac{1}{|\mathcal{N}_u|^{1/2} |\mathcal{N}_v|^{1/2}} (m_v^{(k-1)}) \\ f_{\text{update}}^{(k)}(h_u^{(k-1)}, a_u^{(k)}) &= \sigma\left(\frac{1}{|\mathcal{N}_u|^{1/2} |\mathcal{N}_u|^{1/2}} h_u^{(k-1)} W^{(k)} + a_u^{(k)}\right) \end{aligned} \quad (1.4)$$

Here,  $|\mathcal{N}_u|$  denotes the number of neighbors of node  $u$ , and  $W^{(k)}$  is a learnable weight matrix. The superscripts with parenthesis represent the corresponding matrix or vector that belongs to the  $k$ -th layer. It’s worth noting that in GCN,  $\mathcal{N}_u$  usually includes the node  $u$  itself. The entire GCN layer operation can also be expressed in full matrix form as:

$$H^{(k)} = \sigma(D^{-1/2} A D^{-1/2} H^{(k-1)} W^{(k)})$$

Here,  $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$  denotes the adjacency matrix, where  $A_{uv} = 1$  if  $(u, v) \in \mathcal{E}$  or  $u = v$ , and  $A_{uv} = 0$  otherwise. The degree matrix for  $A$  is denoted by  $D \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ , where  $D_{uu} = \sum_{v \in \mathcal{N}_u} A_{uv}$  and  $D_{uv} = 0$  for  $u \neq v$ . Here,  $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$  denotes the adjacency matrix,

### 1.1.9 Graph Attention Networks (GATs)

Graph Attention Networks (GATs) employ attention mechanisms to update node representations while considering the importance of each neighbor [168]. Specifically, for each node in a graph, GAT computes attention weights that determine the contribution of each neighbor to the node's representation. This enables GAT to learn representations of the graph that capture both the local structure and global connectivity patterns.

In contrast to GCN, which uses a simple sum operation to aggregate normalized neighborhood messages of a given node (as shown in Eq. (1.4)), GAT improves the representational capacity of the aggregation function by leveraging an attention mechanism. The key idea behind this mechanism is to assign different weights to the neighbors of a given node based on their node features.

Formally, the aggregation function of a GAT layer is defined as

$$f_{\text{aggregate}}(\{m_v^{(k-1)}, v \in \mathcal{N}_u\}) := \sum_{v \in \mathcal{N}_u} \alpha_{u,v} m_v^{(k-1)}$$

where  $\alpha_{u,v}$  is an attention weight indicating the importance of node  $v$  to node  $u$ . There are different ways to compute the attention weights. One commonly used implementation, presented in [168], is defined as

$$\alpha_{u,v} = \frac{\exp(e_{u,v})}{\sum_{k \in \mathcal{N}_u} \exp(e_{u,k})}$$

$$e_{u,v} = \sigma([(h_u W^{(k)}) \parallel (h_v W^{(k)})] a^T)$$

where  $W^{(k)}$  is a learnable weight matrix, and  $a$  is a learnable weight vector.

GATs can be particularly useful when nodes in a graph contain informative features. For example, in a recommendation system where interactions between users and items form a network, the profile of the users, as well as the categories of items, are both useful information that can be leveraged to refine the buying preferences of the users.

## 1.2 Literature Review For Time Series and Graph Anomaly Detection

### 1.2.1 Time Series Anomaly Detection

#### A Brief Introduction to Time Series Data

Time series data are sequences of data points collected or recorded at successive time intervals, often at uniform frequencies. They are widely utilized across various domains such as finance, healthcare, manufacturing, and environmental monitoring, where understanding temporal dynamics is crucial for forecasting and anomaly detection [12, 73]. Formally, a time series can be defined as a sequential collection of data points or  $N$ -dimensional vectors that are arranged in a temporal order, denoted as  $X = \{X_{:,t}\}_{t \in \mathcal{T}}$ , where  $t \in \mathcal{T} \subseteq \mathbb{Z}^+$  and  $X_{:,t} \in \mathbb{R}^N$ . If  $N = 1$ , it is known as a univariate time series, while for  $N > 1$ , it is referred to as a multivariate time series.

Time series data possess unique characteristics distinguishing them from other data types, with temporal dependency being a defining feature. In time series, observations are not independent but correlated over time, and these dependencies manifest in several ways such as autocorrelation, trends, and seasonality [12]. Autocorrelation refers to the influence of past values on future values, indicating that observations at adjacent time points are related. Trends represent long-term upward or downward movements in the data, reflecting sustained increases or decreases over time. Seasonality denotes regular, repeating patterns at fixed intervals, often driven by cyclical factors.

In addition to temporal dependencies, inter-variable dependency is an important characteristic of multivariate time series, where multiple variables are recorded simultaneously. These variables often have relationships with each other, and analyzing their interactions can provide deeper insights into the system's behavior. For example, in industrial processes, sensors track variables such as temperature, pressure, and vibration to monitor machinery performance [172]. An anomaly may not be evident when examining each variable individually but becomes apparent when considering the combined behavior of these variables. In financial markets, analysts monitor multiple economic indicators or stock prices concurrently to assess market conditions [148]. A stock's price movement might seem normal on its own, but when paired with unexpected changes in related stocks or sectors, it could indicate an anomaly in the broader market.

## A Review of Recent TSAD Methods

In recent years, deep learning-based TSAD methods have shown great promise due to their ability to learn intricate patterns from data without extensive feature engineering [56, 29]. In the following section, we present a brief review of existing deep learning-based time TSAD frameworks. Specifically, we focus primarily on unsupervised learning methods for TSAD, driven by two key factors: the scarcity of labeled anomalies and the difficulty of obtaining sufficient labeled data. However, we recognize the potential of semi-supervised approaches, particularly when small amounts of labeled data are available, and consider this a promising area for future research.

Deep learning-based unsupervised TSAD methods can be broadly categorized into three types: forecasting-based methods, reconstruction-based methods, and self-supervised learning-based methods.

**Forecasting-based Methods** Forecasting-based models make predictions on a data instance at or around a given time step based on past information within a specific time window. An anomaly is identified if the predicted value of the instance deviates substantially from its ground truth value. The rationale behind this branch of methods is that anomalies disrupt the natural patterns of time series data under normal conditions, thus leading to large prediction errors.

Existing works generally leverage different neural network architectures such as RNNs [112], CNNs [100], GNNs [208], and Transformers [167] to encode time series before feeding to the forecasting task [116, 72, 65, 32, 58, 26, 206]. For instance, The work in [65] leverages TCNs as the backbone encoder to model sequential data with large receptive fields, optimized under a time series forecasting objective. Anomalies are detected based on prediction errors calibrated with a multivariate gaussian distribution. Performance is further enhanced by incorporating multi-scale feature mixtures to capture patterns across different temporal scales. LSTM-NDT [72] uses LSTMs to generate time series representations and perform one-step forecasting, with anomalies detected when prediction errors exceed a dynamically calculated threshold based on a short-term window of past errors. GDN [32] employs GNNs to capture inter-variable dependencies in multivariate time series, detecting anomalies as deviations from learned sensor relationships, with anomaly scores computed as the maximum of normalized prediction errors across all sensors. Extending this, GTA [26] uses a Gumbel-Softmax strategy to automatically learn inter-series graphs, combining GNNs and hierarchical dilated CNNs to capture both spatial and temporal dependencies. It also introduces a Transformer-inspired attention mechanism to further enhance the relation

learning among time series. Subsequently, the anomalies are identified based on deviations from forecasted next-ahead time series values.

**Reconstruction-based Methods** Reconstruction-based TSAD methods focus on learning the normal patterns of multivariate time series data and detecting anomalies based on the reconstruction error, i.e., the difference between the original and reconstructed data. Representative TSAD methods in this line of research are techniques based on deep autoencoders (AEs) [128, 195, 58, 199, 184] and Generative Adversarial Networks (GANs) [96, 166, 25].

Autoencoder-based approaches aim to compress and reconstruct the time series, with anomalies flagged when the reconstruction error is high. For example, LSTM-VAE [128] leverages a variational autoencoder with LSTMs to model temporal dependencies, where anomalies are detected when the reconstruction error surpasses a threshold. MSCRED [195] builds multi-scale signature matrices from the time series and uses a CNNs and RNNs-based encoder-decoder architecture to capture both inter-variable and temporal dependencies, identifying anomalies based on errors in reconstructing these matrices. Anomaly Transformer [184] introduces a novel Anomaly-Attention mechanism within an autoencoder framework to model and reconstruct the associations between time series, detecting anomalies by focusing on association discrepancies between predicted and actual patterns.

On the other hand, GAN-based methods leverage adversarial training to generate synthetic time series, flagging anomalies based on poor reconstruction or discrimination by the model. MAD-GAN [96] employs LSTM-based GANs, where a generator is employed to produce synthetic time series data to mimic normal conditions, and a discriminator is used to distinguish it from the true time series instances. Anomalies are thus detected using a combination of reconstruction error and discrimination loss. DAEMON [24] uses an adversarial autoencoder with two discriminators to regularize both the latent space and reconstructed time series. Specifically, adversarial training is applied to the latent space by ensuring that the encoded representations follow a prior distribution, making the learned representations more robust and generalizable. Additionally, the reconstructed time series is regularized through another adversarial process, where the decoder is trained to generate reconstructions that closely match the original data. Anomalies are detected based on the reconstruction error of the time series.

**Self-supervised Learning-based Methods** Self-supervised learning-based methods can be generally categorized into augmentation prediction-based and contrastive learning-based approaches. Augmentation prediction involves generating a set of samples by ap-

plying various data augmentation operations to each input instance and then training a model to classify the specific augmentation that a given augmented sample corresponds to [50, 66, 175]. However, prior works rely on hand-crafted transformations for image data such as cropping or rescaling, making it challenging to design such transformations for data beyond images. To overcome this challenge, NeuTral AD [136] employs learnable transformations to embed transformed data into a semantic space that maintains the similarity of the transformed data to their original form while enabling easy differentiation between different transformations via separation in the representation space.

Contrastive learning is a representation learning technique that learns discriminative features by contrasting positive and negative pairs in the data [121]. It works by creating positive pairs (similar data, often different views or augmentations of the same instance) and negative pairs (dissimilar data, often unrelated instances). Then, it learns to pull the representations of positive pairs closer together in the latent space, while pushing apart the representations of negative pairs. Recently, quite a few works have adapted contrastive learning into TSAD [133, 207, 209, 174, 188]. TRL-CPC [133] applies contrastive learning by using the predicted latent representations of future time steps as positive pairs and sampling unrelated time steps as negative pairs. Anomalies are detected based on the alignment between predicted and actual representations, with large deviations indicating anomalies. CTAD [209] also uses contrastive learning but focuses on generating augmented versions of normal time series to serve as negative examples, aiming to separate normal from synthetic anomalies. COCA [174] treats original and reconstructed time series representations as positive pairs for contrastive learning, identifying anomalies by maximizing the similarity between these representations and capturing only normal behavior to detect deviations. Lastly, DCdetector [188] introduces a dual-attention mechanism to capture multi-scale representations of time series from two different views, using contrastive learning to measure the discrepancy between these views, with large discrepancies highlighting anomalies.

## 1.2.2 Graph Anomaly Detection

### A Brief Introduction to Graph

Graph represents entities and the relationships between them using nodes and edges. Formally, a graph is defined as  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$  is the set of nodes (also known as vertices), and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  is the set of edges connecting pairs of nodes.

In many real-world applications, nodes possess additional attribute information, which is often referred to as an attribute graph. The attribute graph is defined as  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ ,

where  $X = [x_1; x_2; \dots; x_n] \in \mathbb{R}^d$ , and each  $x_i \in \mathbb{R}^d$  represents a  $d$ -dimensional attribute vector associated with node  $v_i$ . Examples of attributed graphs include social networks, where nodes represent users with attributes such as age, location, and interests, and edges represent friendships or interactions [131]; and financial transaction networks, where nodes represent accounts with attributes like account type and balance, and edges represent transactions between accounts [3].

Graphs may also evolve over time, which captures the dynamic nature of relationships in complex systems. The edge  $e \in \mathcal{E}$  in a dynamic graph is typically represented as an event  $\delta^t = (v_i, v_j, t, x_{ij})$ . Here,  $v_i, v_j \in \mathcal{V}$  are the nodes involved in the interaction,  $t \in \mathcal{T}$  is the timestamp indicating when the interaction occurred, and  $x_{ij} \in \mathbb{R}^{d'}$  denotes the attribute vector associated with the edge at time  $t$  [139]. Dynamic graphs are prevalent in various domains. For instance, in e-commerce platforms, nodes can represent users and products, while edges represent user interactions with products (e.g., views, clicks, purchases) at specific times, with attributes including interaction type or rating [37].

Similar to time series data, graphs encapsulate contextual information that is critical for understanding complex system behavior. In contrast to traditional datasets where individual instances are treated independently (e.g., images or tabular data), graphs inherently encode dependencies among entities through edges. These structural dependencies can reveal abnormal patterns that may not be apparent when nodes or entities are analyzed in isolation [3]. For instance, in a financial network, a transaction that appears normal on its own may raise suspicion if it occurs between accounts with no prior transaction history. Additionally, the inclusion of node and edge attributes can provide essential context for detecting anomalies that may not be evident from structure alone. For example, in a social network, users within a tightly connected group often share similar attributes. A user linked to a group of teenagers but exhibiting attributes typical of an older demographic could raise concerns about potential malicious activity. In such cases, the structural position suggests the user belongs to a particular group, but the attribute inconsistency highlights the need for further investigation. This interplay between structure and attributes is crucial for uncovering subtle abnormalities that might otherwise go undetected [34].

## A Review of Recent GAD Methods

Recent advancements in GNNs have revolutionized graph anomaly detection by enabling more effective modeling of complex graph data [82]. GNNs leverage message-passing mechanisms to aggregate information from neighboring nodes and edges recursively, facilitating the learning of graph representation that captures both structural dependencies and attribute patterns [86, 168]. Building on these advances, deep learning-based methods have

become central to recent research in graph anomaly detection. In the following section, we will review the key categories of these methods, focusing on unsupervised, semi-supervised, and few-shot GAD approaches.

**Unsupervised GAD Methods** Unsupervised GAD methods aim to uncover anomalies by analyzing the intrinsic properties of graph structures and attributes without relying on labeled data. These methods operate under the assumption that anomalies deviate substantially from the typical patterns observed in the graph. One line of this branch is reconstruction-based methods, which attempt to reconstruct the original graph data such as graph adjacency matrices or node attributes, and identify anomalies through significant reconstruction errors [34, 41, 108, 200]. For example, DOMINANT [34] employs autoencoders to reconstruct both graph structures and node attributes, detecting anomalies when the reconstruction error for certain nodes is considerably higher than that of the majority. ComGA [108] detects anomalies in attributed graphs by integrating community-specific representations into GCN layers through weighted combinations. It uses an autoencoder to extract community structure and decoders to jointly reconstruct graph structure and attributes, assigning anomaly scores based on reconstruction errors.

Another widely explored direction is contrastive learning-based methods, which enhance the learned representations by contrasting similar and dissimilar pairs within the graph [103, 204, 197, 40, 123]. For instance, GRADATE [40] performs multi-scale contrastive learning by contrasting node-node, node-subgraph, and subgraph-subgraph pairs to capture anomalies at different levels. Positive pairs are generated through graph augmentations (edge dropping/adding), while negative pairs are formed by sampling unrelated nodes or subgraphs. Given that subgraphs are sampled using random walks, the anomaly scores are calculated by aggregating the contrastive loss across multiple rounds to achieve stable scores for each node. PREM [123] extracts ego features and neighbor features using a propagation matrix that aggregates information from neighboring nodes with or without including the node’s own features. Both ego and neighbor features are encoded through MLPs and optimized under contrastive learning, where positive pairs are formed by matching the same node’s ego and neighbor embeddings. The final anomaly score is calculated as the negative cosine similarity between ego and neighbor embeddings, with lower similarity indicating a higher likelihood of the node being anomalous.

Another branch of unsupervised GAD leverages labeled data from a related source domain to enhance anomaly detection in an unlabeled target domain, often referred to as cross-domain GAD methods. These approaches transfer knowledge across domains, enabling effective detection when direct supervision in the target domain is not available. For

instance, COMMANDER [35] aligns embeddings from source and target graphs by training a domain discriminator and encoder adversarially. Subsequently, an anomaly classifier is trained on the labeled source graph, aiming to predict shared anomalies in the target graph. Additionally, an attribute decoder detects unshared anomalies via reconstruction errors, and the final anomaly score combines classifier predictions with reconstruction errors. ACT [173] employs contrastive learning to develop normal node representations in the target graph and aligns them with representations from the source graph, which is trained with sufficient supervision. The alignment is achieved by minimizing the Wasserstein distance between the normal node representations of the two domains. The framework is further enhanced through a self-training mechanism that refines detection accuracy by retraining the model with pseudo-labeled anomalies and normal nodes.

The aforementioned methods primarily focus on static graphs, where the graph structure remains fixed. However, many real-world graphs are dynamic, with nodes, edges, and interactions evolving over time. Detecting anomalies in dynamic graphs requires models capable of capturing both structural dependencies and temporal patterns [203, 15, 104]. For example, AddGraph [203] processes a dynamic graph as a sequence of discrete snapshots, employing GCNs followed by GRUs to model both structural and temporal dependencies. As labeled data is unavailable during training, it applies selective negative sampling by generating negative edges based on node degrees and uses a margin-based loss to ensure normal edges receive lower anomaly scores. Similarly, TADDY [104] operates on graph snapshots using a sliding window to capture substructures around target edges over multiple time steps. It employs a transformer model to learn both static graph patterns and their temporal evolution. TADDY distinguishes anomalies by comparing real edges with pseudo-anomalous edges, which are generated during training, using a classification layer that assigns anomaly scores.

**Semi-supervised GAD Methods** While unsupervised methods are effective in the absence of labeled data, they are often prone to false positives and sensitive to noise due to their reliance on learned patterns from unlabeled data alone. In real-world applications, however, it is often feasible to obtain a small amount of labeled data. Semi-supervised GAD strikes a balance between unsupervised and supervised learning by using this limited labeled data to guide the learning process while still extracting robust patterns from the larger pool of unlabeled data [170, 113, 87, 164].

For example, SemiGNN [170] generates node embeddings by employing a node-level attention mechanism to weigh the importance of neighbors in view-specific graphs. These embeddings are further integrated through view-level attention, which learns the importance of each view to produce a unified node representation. It is then optimized using

both a classification loss on labeled data and an unsupervised graph-based loss on unlabeled data. Similarly, DHE [87] addresses anomaly detection by minimizing the volume of a hypersphere enclosing the embeddings of normal nodes, with the assumption that anomalous nodes will fall outside the hypersphere. The anomaly score is calculated based on the distance from the hypersphere center, and a differentiable AUC loss ensures effective separation between normal and anomalous nodes.

Recent research has also expanded to dynamic graphs, which introduce additional challenges by requiring models to capture temporal dependencies. For instance, Semi-ADC [113] uses a bidirectional GANs to estimate the distribution of normal features, with labeled anomalies providing regularization to enhance robustness. The model integrates self-training by iteratively refining node labels based on both structural and temporal correlations. Anomalies are detected by combining reconstruction errors with the similarity to labeled anomalies. Similarly, SAD [164] enhances dynamic graph anomaly detection through a temporal graph encoder that aggregates structural and temporal features across multiple GNN layers. Anomaly scores are computed using a deviation network [125], with a memory bank tracking historical scores to maintain a statistical distribution of normal and unlabeled nodes. To further leverage unlabeled data, SAD employs a pseudo-label contrastive learning module, grouping nodes with similar scores as positive pairs to improve detection performance.

**Few-shot GAD Methods** While semi-supervised GAD can effectively utilize a small amount of labeled data, its performance may suffer when the number of labeled anomalies is extremely limited. Few-shot GAD methods aim to address this challenge by focusing on scenarios where only a handful of labeled anomalies (typically fewer than 10) are available for training [36, 205, 55]. Existing methods often adapt techniques from meta-learning [70], which involve training the model across a series of related tasks (meta-train) to develop generalized knowledge that can be transferred to new tasks (meta-test). In the context of few-shot GAD, the model is trained on multiple auxiliary networks (additional networks within the same domain that share similar structural and attribute characteristics) during the meta-train phase, learning to recognize patterns that can generalize to the target network. These methods operate under the assumption that a model trained on a set of auxiliary tasks can generalize effectively to new tasks, allowing for rapid adaptation to new graph structures despite the limited availability of labeled data. For example, Meta-GDN [36] employs GNNs to generate node representations and uses a cross-network meta-learning framework to transfer knowledge from multiple auxiliary networks to the target network. It computes anomaly scores using a deviation loss, which enforces the normal instances following the standard normal distribution while pushing away anoma-

lous instances. Similarly, LHML [55] leverages GNNs within a cross-network meta-learning approach but introduces a different loss function based on a learnable hypersphere mechanism. This hypersphere adaptively adjusts its radius to form a decision boundary in the embedding space, separating normal and anomalous nodes. Anomaly scores are then determined by the distance of each node from the hypersphere center, allowing the model to effectively identify anomalies even under few-shot conditions.

## Chapter 2

# Harnessing Contrastive Learning and Neural Transformation for Time Series Anomaly Detection

### 2.1 Introduction

Modern industrial systems rely on IoT technology for communication and information exchange among sub-systems. Due to the vast scale of these distributed systems, unanticipated events such as cyber-attacks can cause system downtime and may even lead to catastrophic failures [114, 32, 96]. For instance, in smart power grids, malicious activities can disrupt electricity distribution, leading to widespread blackouts and significant economic losses. Therefore, an efficient and accurate time series anomaly detection (TSAD) system is crucial for monitoring critical infrastructure and providing timely anomaly alerts, helping prevent failures and safeguard data. Despite the importance of TSAD, its efficacy is often constrained by the rarity of anomalies and the resource-intensive nature of obtaining accurate labels. These issues limit the practicality of supervised methods and drive the need for unsupervised TSAD approaches. Deep learning models excel in this regard, as they can learn complex feature representations without reliance on manual labeling.

Deep learning-based unsupervised TSAD primarily focuses on identifying patterns of normality within unlabeled data, with the expectation that anomalies will manifest as deviations from these identified patterns. Existing methods primarily include forecasting-based [72, 65, 32, 58, 26] and reconstruction-based approaches [96, 7, 195, 166, 184, 25]. The former assumes normal instances are predictable, but their performances can falter

when temporal patterns are unpredictable or rapidly changing [110]. The latter, despite not depending on sample extrapolation, can run into overfitting issues due to an undue emphasis on pixel-wise reconstructions. In recent years, contrastive learning has emerged as a significant area of interest in various domains [62, 22, 143, 150], lauded for its capability to distill valuable feature representations from data. Unlike traditional approaches, contrastive learning operates on the latent feature space, advocating for the alignment of similar instances and the segregation of dissimilar ones. In recent years, a growing number of studies have applied contrastive learning to anomaly detection and shown promising results in various domains such as computer vision, natural language processing, and speech processing [50, 66, 175, 69, 134].

Yet applying contrastive learning to TSAD presents unique challenges. First, anomalies in time series data are context-dependent, necessitating an analysis that not only considers each instance but also understands its relationship to historical patterns. Secondly, contrastive learning typically relies on selecting negative samples to avoid representation collapse, where models converge to trivial solutions mapping different inputs to indistinguishable representations. However, deriving negative samples from a pool of predominantly normal instances is ill-suited for anomaly detection. moreover, the generation of “similar” positive samples through data augmentation, such as geometric transformations for images (e.g., cropping and rotating) [50], is not directly applicable to time series data due to their intricate temporal dependencies.

In this work, we propose a simple but effective TSAD method, CNT, to address the aforementioned challenges. Motivated by Contrastive Predictive Coding (CPC) [121], we implement a relatively simple but effective temporal-wise contrasting where each instance is defined by a prediction time window and is pulled closer to a corresponding context window through a contextual contrastive loss. This strategy naturally discerns anomalies by highlighting significant divergences from historical patterns. However, this alignment alone can result in the representation collapse issue: the feature encoders of both windows produce constant representations irrespective of their input. In traditional contrastive learning, this issue is often circumvented by using negative samples, ensuring diversity in the learned representations. To avoid incorrectly selecting negative samples within the same normal class, we enhance the temporal-wise contrasting process by applying learnable neural transformations to the latent space of the prediction window, ensuring that the resulting representations remain diverse yet consistent with the original inputs. Subsequently, temporal-wise contrasting is executed between these transformed windows and their past observations, with anomalies anticipated to disrupt this contextual consistency more than normal instances. Lastly, our theoretical analysis indicates that this duel-loss circumvents the issue of representation collapse without the need for negative samples.

Extensive experiments are conducted on five real-world industrial datasets to demonstrate the superior performance of our proposed approach over existing baseline methods.

To summarize, our key contribution includes:

1. We propose a simple and computationally efficient TSAD framework, CNT, which harnesses the idea of temporal-wise contrastive learning with neural transformation learning. The approach is specifically tailored for the detection of temporal anomalies through a combination of contextual contrastive loss and discriminative contrastive loss.
2. We provide theoretical analysis demonstrating that a combination of the aforementioned two losses effectively mitigates the representation collapse problem of contrastive learning without introducing negative samples.
3. Extensive experiments are conducted on real-world industrial datasets to demonstrate the superior performance of our proposed approach over existing baseline methods and a variety of model variants.

## 2.2 Related Works

Self-supervised learning-based anomaly detection methods can be generally categorized into augmentation prediction-based and contrastive learning-based approaches. Augmentation prediction involves generating a set of samples by applying various data augmentation operations to each input instance and then training a model to classify the specific augmentation that a given augmented sample corresponds to [50, 66, 175]. However, most of them rely on hand-crafted transformations specialized for image data, making it challenging to design such transformations for data beyond images. To overcome this challenge, NeuTral AD [136] employs learnable transformations to embed transformed data into a semantic space that maintains the similarity of the transformed data to their original form while enabling easy differentiation between different transformations via separation in the representation space.

Contrastive learning brings together representations of data instances and their augmented versions or similar instances while pushing away negative samples. A seminal work in this field is CPC, which is tailored specifically to sequential data. Recently, quite a few works have adapted contrastive learning into TSAD [133, 207, 209, 174, 188]. For example, CAE-AD [207] explores the concept of contextual contrasting, akin to CPC, to

discern temporal patterns within time series data. It applies data augmentation in both the time and frequency domains, promoting similarity in the representations of augmented instances through instance-level contrast. Similarly, DCAD [209] leverages an attention mechanism and a position-aware convolution module to extract both global and local features from time series and their subsequences. It employs a contrastive loss to foster alignment between these global and local representations.

However, a common challenge with these methods is the selection of negative samples from the pool of training instances, which might conflict with the notion that normal samples tend to cluster together in the feature space. To address this, alternative contrastive learning strategies, such as BYOL [54], SimSiam [23], and VICReg [9], eliminate the need for negative samples. These approaches focus on preventing the potential collapse of the feature manifold, which could occur when only positive samples are contrasted. VICReg, for instance, has been adapted for TSAD by COCA [174], which employs a contrastive learning paradigm optimized with a variance term to avoid the collapse of representations. Similarly, DCdetector [188] introduces a dual attention mechanism to learn distinct time series representations from both a patch-wise and within-patch perspective, utilizing a siamese network structure akin to SimSiam but without the need for negative samples, thereby sidestepping the manifold collapse issue.

## 2.3 Methodology

### 2.3.1 Problem Formulation

We are given a multivariate time series  $X \in \mathbb{R}^{N \times T}$ , collected over  $T$  time ticks with  $N$  features, as the training set. The goal is to detect anomalies in a test set, which contains a time series with  $N$  features over a different time span. The training set  $\mathcal{D}$  consists of sub-sequences extracted from  $X$ , i.e.,  $X$  is converted into sliding windows with stride 1. Here, a sample  $X^t$  represents a collection of time ticks within a sliding window of length  $w$ :  $X^t = \{X_{\cdot i}\}_{i=t-w+1}^t$ , where  $X_{\cdot i} \in \mathbb{R}^N$  is the multivariate time series value at timestamp  $i$ . Our objective is to assign an anomaly score to each time tick in the test set, which is later thresholded to classify each tick as normal (0) or abnormal (1). Furthermore, in our framework, each sample  $X^t$  is segmented into two overlapping sub-sequences of length  $c$ : a context sequence  $C^t$  and a prediction sequence  $S^t$ , where  $S^t$  is shifted forward from  $C^t$  by a small time window of length  $p = w - c$ , i.e.,  $C^t = \{X_{\cdot i}\}_{i=t-w+1}^{t-p}$  and  $S^t = \{X_{\cdot i}\}_{i=t-c}^t$ . The time window from  $t - p + 1$  to  $t$  is referred to as the prediction window, while  $t - w + 1$

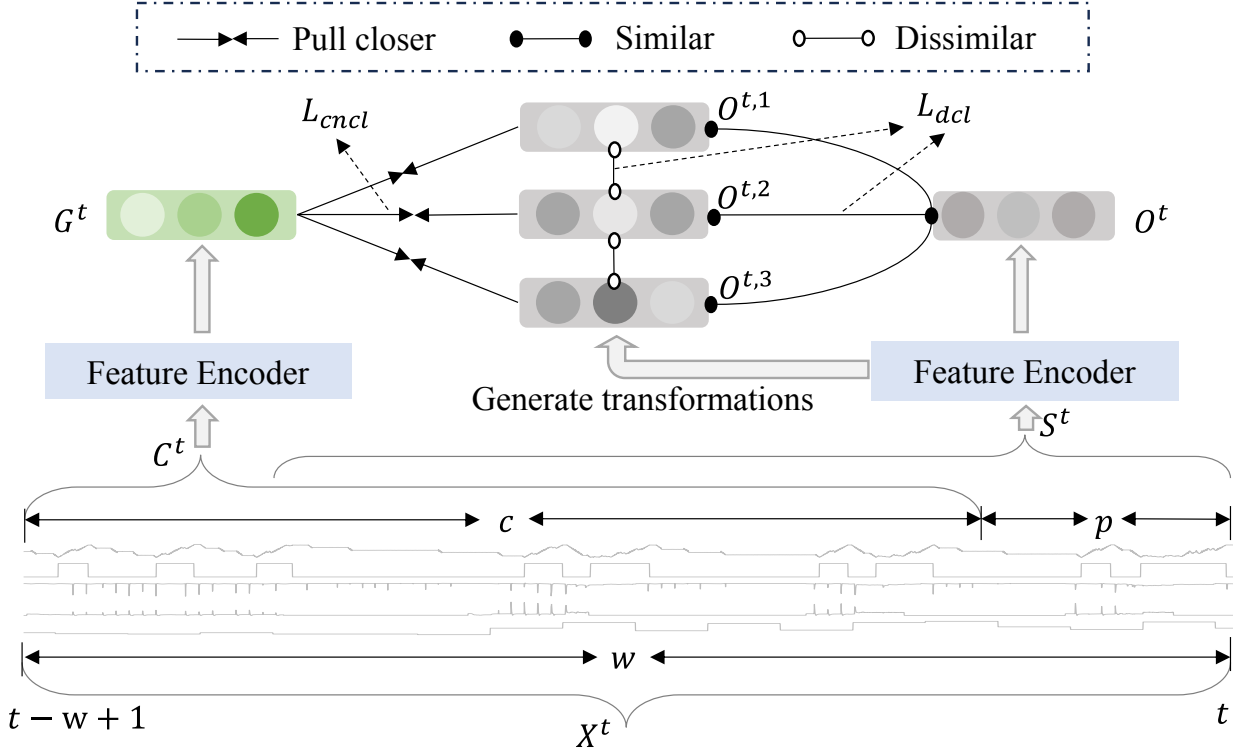


Figure 2.1: An illustration of the CNT framework.

to  $t - p$  is the context window. A visualization of notations is shown at the bottom of Figure 2.1.

### 2.3.2 Overview

This section presents an overview of the proposed framework, CNT, which includes a feature encoder module and a neural transformation module with both jointly optimized using two contrastive losses (See Figure 2.1 for an illustration). Specifically, the framework begins with the feature encoder, which converts the prediction and context sequence of a given sample into hidden representations. Next, the neural transformation network transforms the latent representation of the prediction sequence into multiple latent representations. Finally, model parameters are optimized under two loss functions: a contextual neural contrastive loss and a discriminative contrastive loss. The former aims at minimizing the distance between the context sequence and the transformations of the prediction sequence.

The latter ensures that the transformed sequences remain diverse while maintaining the original semantics of the prediction sequence.

### 2.3.3 Framework

#### Feature Encoder

The feature encoder aims at capturing temporal dependencies within a predict or context sequence, thereby facilitating the detection of anomalies in the predict window by identifying anomalous temporal patterns with respect to the context window. We have defined two overlapping sequences -  $C^t$  and  $S^t$ , with equal lengths, as discussed in Section 2.3.1. Following the methodology employed by [181], we have utilized a temporal convolutional network (TCN) as the base feature encoder, wherein the temporal convolution layer has been replaced by a dilated inception layer (DIL) [181] that enables the use of different kernel sizes. In the following paragraph, we demonstrate the pipeline using  $S^t$  as input.

More specifically, we start our procedure by passing  $S^t$  through a 1D convolution with a channel size of  $d$  and a kernel size of 1 to obtain its initial representation,  $Z^t \in \mathbb{R}^{d \times c}$ . Subsequently, we pass it through  $L$  TCN blocks, each of which is defined as:

$$Z^{t,l} = \text{BN}(\sigma(f_{\text{DIL}}(Z^{t,l-1}))) + Z^{t,l-1}$$

where BN refers to a batch normalization layer [74],  $\sigma(\cdot)$  represents a ReLU function, and  $Z^{t,l}$  is the output from the  $l^{\text{th}}$  block. The dilated inception layer  $f_{\text{DIL}}(\cdot)$  is defined as:

$$f_{\text{DIL}}(X) = (X \star f_2) \parallel (X \star f_3) \parallel (X \star f_5) \parallel (X \star f_7)$$

where  $X \star f_k$  denotes a 1D temporal convolution layer applied to  $X$  with a kernel size of  $k$ , a dilation factor of  $2^{(l-1)}$ , and an output channel size of  $\frac{d}{4}$ , and the concatenation operation  $\parallel$  is performed on the channel dimension. We do not pad the sequences, and thus, each concatenation is performed on the truncated outputs from the four different convolutions, with the truncated size matching the length of the shortest output. After applying a sequence of TCN blocks, we obtain a final output with a dimension of  $(d, 1)$ , which is subsequently fed through another 1D convolution layer with a kernel size of  $(1, 1)$  to derive the ultimate latent representation of  $S^t$ . We denote the latent representations of  $S^t$  and  $C^t$  as  $O^t$  and  $G^t$ , respectively. See the visualization of the feature encoder in Figure 2.2.

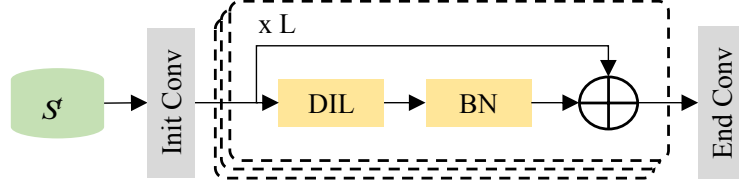


Figure 2.2: An illustration of the feature encoder.

### Contextual Contrastive Loss

Our framework assumes that normal sequences change mildly compared to abnormal sequences. To implement this idea, we minimize the distance between the hidden representations of the prediction and context sequences for normal samples. Specifically, let  $g(\cdot; W) : \mathbb{R}^{N \times c} \rightarrow \mathbb{R}^d$  be the feature encoder with learnable weights  $W$ . We define a contextual contrastive loss as:

$$\mathcal{L}_{\text{ccl}} := \mathbb{E}_{X^t \sim \mathcal{D}} \|g(S^t; W) - g(C^t; W)\|_2^2 \quad (2.1)$$

However, minimizing this loss alone can lead to representation collapse, where the encoder outputs constant values. To address this, inspired by NeuTralAD [136], we combine the contextual contrastive loss with neural transformation learning, ensuring distinct outputs across different contexts.

### Combining With Neural Transformations

Neural transformation learning learns diverse transformations that maintain the original sample’s semantic information [136]. We leverage this approach to prevent representation collapse by contrasting the multifaceted transformed representations of the prediction sequence  $O^t$  against the context sequence  $G^t$ .

Given  $O^t$ , we project it into  $K$  latent representations  $\{O^{t,i}\}_{i=1}^K$  using distinct transformation networks  $T = \{T_1, \dots, T_K | T_k : \mathbb{R}^d \rightarrow \mathbb{R}^d\}$ . Each  $T_k$  is a three-layer MLP with hidden dimension  $d$ . Unlike the original approach in [136], we apply transformations directly in the latent space instead of raw inputs. Thereby, it reduces computational complexity by avoiding repeated passes through the feature encoder.

To maintain diversity and semantic consistency, a discriminative contrastive loss is applied on  $O^t$  and  $\{O^{t,i}\}_{i=1}^K$ :

$$\mathcal{L}_{\text{dcl}} := -\mathbb{E}_{X^t \sim \mathcal{D}} \left[ \sum_{k=1}^K \log \frac{h(O^t, O^{t,k})}{h(O^t, O^{t,k}) + \sum_{l \neq k} h(O^{t,k}, O^{t,l})} \right] \quad (2.2)$$

where  $h(x, y) = \exp(\frac{x^T y}{\tau \|x\|_2 \|y\|_2})$ .  $\tau$  is a hyper-parameter called temperature. We then extend  $\mathcal{L}_{\text{ccl}}$  to a contextual neural contrastive loss  $\mathcal{L}_{\text{cncl}}$ , which is defined as:

$$\mathcal{L}_{\text{cncl}} := \mathbb{E}_{X^t \sim \mathcal{D}} \left[ \sum_{k=1}^K \|O^{t,k} - G^t\|_2^2 \right] \quad (2.3)$$

Here, the transformed representations of the prediction sequence are pulled closer to those of the context sequence.

Finally, the parameters of the feature encoder and neural transformations are optimized under a contextual discriminative contrastive loss  $\mathcal{L}_{\text{cdcl}}$ , defined as:

$$\mathcal{L}_{\text{cdcl}} := \mathcal{L}_{\text{cncl}} + \mathcal{L}_{\text{dcl}} \quad (2.4)$$

During inference, the loss in Eq. (2.4) for an input at time  $t+p-1$ , i.e.,  $X^{t+p-1}$ , is used as the anomaly score for time  $t$ . Thus, a shorter prediction window size  $p$  can introduce a smaller detection delay.

### 2.3.4 Theoretical Analysis

In this section, we discuss the representation collapse issue that arises when applying solely the contextual contrastive loss and demonstrate how integrating a discriminative contrastive loss mitigates this problem. The contextual contrastive loss in Eq. (2.1) can lead to a trivial solution where the feature encoder outputs a constant value, e.g., setting all weights to zero. However, constant encoders are not optimal for minimizing the contextual discriminative contrastive loss in Eq. (2.4), as shown in the following proposition.

**Proposition 2.1.** *If there exists a choice of parameters for the feature encoder  $g(\cdot)$  and the  $K$  transformation networks  $\{T_k(g(\cdot))\}_{k=1}^K$  such that  $\mathcal{L}_{\text{dcl}} < K \log K$ , then the constant parameter setting  $T_k(g(\cdot)) = g(\cdot) = c, \forall c \in \mathbb{R}^d$  is not an optimal solution for minimizing  $\mathcal{L}_{\text{cdcl}}$ .*

**Proof.** For simplicity, let  $g_k(\cdot)$  denote  $T_k(g(\cdot))$ . First, note that when  $g_k(\cdot) = g(\cdot) = c$ , the contextual discriminative contrastive loss  $\mathcal{L}_{\text{cdcl}}$  simplifies to  $K \log K$ . Specifically:

$$\mathcal{L}_{\text{cdcl}} = K \|c - c\|_2^2 - K \log \frac{h(c, c)}{h(c, c) + (K-1)h(c, c)} \quad (2.5)$$

$$= K \log K \quad (2.6)$$

Assuming that there exists a parameter setting such that  $\mathcal{L}_{\text{dcl}} < K \log K$ , let  $\epsilon = K \log K - \mathcal{L}_{\text{dcl}}$ . Since the scoring function  $h(\cdot, \cdot)$  in  $L_{\text{dcl}}$  is defined as  $h(g_k(\cdot), g(\cdot)) = \exp(\frac{g_k(\cdot)^T g(\cdot)}{\tau \|g_k(\cdot)\|_2 \|g(\cdot)\|_2})$ , we can rescale the functions  $\{g_k(\cdot)\}_{k=1}^K$  and  $g(\cdot)$  during training such that  $\|g_k(x)\|_2 \leq \frac{\epsilon}{2K}$  and  $\|g(x)\|_2 \leq \frac{\epsilon}{2K}$  for all  $x$ . This rescaling is possible while ensuring that  $h(g_k(x), g(x))$  remains the same, since cosine similarity between two vectors is invariant to their norms. Hence, we have:

$$\mathcal{L}_{\text{cncl}} = \mathbb{E}_{X^t \sim \mathcal{D}} \left[ \sum_{k=1}^K \|g_k(O^t) - g(G^t)\|_2^2 \right] \quad (2.7)$$

$$\leq \mathbb{E}_{X^t \sim \mathcal{D}} \left[ \sum_{k=1}^K (\|g_k(O^t)\|_2^2 + \|g(G^t)\|_2^2) \right] \quad (2.8)$$

$$< \sum_{k=1}^K \left( \frac{\epsilon}{2K} + \frac{\epsilon}{2K} \right) = \epsilon \quad (2.9)$$

Therefore, we have  $\mathcal{L}_{\text{cncl}} + \mathcal{L}_{\text{dcl}} < \epsilon + \mathcal{L}_{\text{dcl}} = \epsilon + K \log K - \epsilon = K \log K$ .  $\square$

The intuition that  $K \log K$  does not lead to the minimum solution of  $\mathcal{L}_{\text{dcl}}$  is that the objective of minimizing this loss function is to make sure that the generated transformations are diverse among themselves while preserving certain semantic similarities from a given input, which contradicts the constant encoder solution. Additionally, we empirically validate this assumption with  $\mathcal{L}_{\text{dcl}} \ll K \log K$  after training (See in Section 2.4.4).

## 2.4 Experiments

### 2.4.1 Experimental Setting

#### Datasets

We evaluate our framework using five real-world industrial TSAD datasets: SWaT [111], WADI [2], HAI [153], SMAP and MSL [72]. The SWaT and WADI datasets are sourced from scaled-down water treatment and distribution testbeds designed to emulate large-scale urban water systems. SWaT records 51 sensor dimensions over 11 days, with the final 4 days containing simulated cyber attacks, while WADI spans 16 days across 123 channels, with attacks introduced during the last 2 days. The SMAP and MSL datasets, provided by NASA, contain telemetry data from the SMAP satellite and the Mars Science

Laboratory (MSL) rover. Anomalies in these datasets reflect unexpected events in spacecraft operations, represented through sensor readings and command signals. The HAI dataset was collected from an Industrial Control System (ICS) testbed, enhanced with a Hardware-in-the-Loop (HIL) simulator to emulate realistic industrial processes. This testbed includes components such as a boiler, a turbine, and a water treatment process, along with HIL-simulated systems for steam-turbine power generation and pumped-storage hydropower. Anomalies in this dataset reflect operational irregularities, enabling detailed testing of anomaly detection within complex, simulated industrial environments. Each dataset provides a training set under normal operating conditions and a separate test set with labeled anomalies, enabling a comprehensive evaluation of our anomaly detection framework across varied industrial scenarios. Summary statistics, including channel count, training and test set sizes, and anomaly rates, are presented in Table 2.1.

Table 2.1: Summary statistics for datasets

| <b>Dataset</b> | <b># channels</b> | <b># training</b> | <b># test</b> | <b>% anomalies</b> |
|----------------|-------------------|-------------------|---------------|--------------------|
| SWaT           | 51                | 47440             | 44993         | 12.13              |
| WADI           | 123               | 78459             | 17281         | 5.77               |
| MSL            | 55                | 58317             | 73729         | 10.53              |
| SMAP           | 25                | 135183            | 427617        | 12.79              |
| HAI            | 79                | 92163             | 27003         | 1.93               |

## Baselines

We compare our approach to several recent anomaly detection methods, including the reconstruction-based methods (MAD-GAN [96], USAD [7], TranAD [166], AnomalyTransformer [184]), forecasting-based method (CST-GL [206]), and contrastive learning-based methods NeuTralAD [136], COCA [174], and DCdetector [188]. 1) MAD-GAN [96] trains a Generative Adversarial Network (GAN) [30] with LSTM as the base architecture. The generator produces synthetic time series data, while the discriminator identifies real versus synthetic data. Anomalies are detected using a combined score of discrimination loss and reconstruction loss. 2) USAD [7] uses an encoder-decoder architecture with two autoencoders trained in dual phases: reconstruction and adversarial training. The encoder compresses input windows, while decoders focus on reconstructing normal patterns and distinguishing anomalies by amplifying reconstruction errors. 3) TranAD [166] trains a

transformer network to reconstruct the time series data. The reconstruction error is then used to guide the attention network in the Transformer in a subsequent training phase. It also uses two decoders with opposite objectives to amplify the reconstruction errors. 4) AnomalyTransformer [184] employs the Transformer network as the backbone encoder. It proposes a self-attention mechanism that computes the association discrepancy between each time point and the whole series, which is used to compute the anomaly score. During the optimization phase, it alternately minimizes and maximizes the association discrepancy under the reconstruction loss. 5) NeuTralAD [136] uses learnable transformations and an encoder trained jointly with a deterministic contrastive loss. The model optimizes the loss to produce diverse transformations that retain semantic similarity with original samples, enabling the model to distinguish normal patterns from anomalies without additional regularization. 6) DCdetector [188] constructs representations of multivariate time series data by leveraging two branches: patch-wise and in-patch attention mechanisms, both employing self-attention networks. It calculates an anomaly score based on the Kullback-Leibler divergence between the representations from the two branches, with higher scores indicating anomalies. 7) COCA [174] treats the latent representations of input time series and their reconstructions as positive pairs in a contrastive learning framework. It then proposed a loss function that integrates contrastive learning and one-class classification. Specifically, it combines an invariance term to maximize the cosine similarity between representations and a one-class center, and a variance term to prevent hypersphere collapse. Anomalies are detected based on the deviation of representations from the one-class center.

## Evaluation metrics

We use the point-wise F1 (denoted as  $F1^p$ ), point-adjusted F1 (denoted as  $F1^*$ ) [157] and composite F1 (denoted as  $F1^c$ ) [49] score as the performance metrics.

The point-wise F1 is calculated using standard precision and recall at the point level:

$$F1^p = 2 \times \frac{Pr \times Re}{Pr + Re} \quad (2.10)$$

where precision (Pr) and recall (Re) are based on true positive, false positive, and false negative counts for individual time points.

For the point-adjusted F1, we follow the point-adjusting strategy suggested in [157], which treats an entire anomaly segment as correctly detected if at least one point in the segment is identified. This is computed as:

$$F1^* = 2 \times \frac{Pr^* \times Re^*}{Pr^* + Re^*}, \quad (2.11)$$

where precision ( $\text{Pr}^*$ ) and recall ( $\text{Re}^*$ ) are modified to count anomaly segments rather than individual points. In our implementation, the approach works as follows: if the model predicts a 1 (anomaly) for any time point within a contiguous ground-truth anomaly segment, we set the predicted labels for all points in that segment to 1. Standard precision and recall are then computed on this modified prediction vector, ensuring that the evaluation focuses on whether each anomaly segment is detected rather than on the exact number of anomalous time points.

Composite F1 is the harmonic mean of time-wise precision and event-wise recall. As proposed in [49], it is calculated as:

$$\begin{aligned} Pr_t &= \frac{TP_t}{TP_t + FP_t} \\ Re_e &= \frac{TP_e}{TP_e + FN_e} \\ F1^c &= 2 \times \frac{Pr_t \times Re_e}{Pr_t + Re_e} \end{aligned}$$

where  $TP_t$  and  $FP_t$  count individual time-point true positives and false positives, while  $TP_e$  and  $FN_e$  count event-level true positives and false negatives.

Given that  $F1^*$  may overestimate recall [83], and  $F1^p$  does not account for event lengths [83],  $F1^c$  offers a balanced measure that integrates precision and recall across different granularities of anomaly evaluation. For all F1 metrics, we identify the best F1 score by optimizing the threshold for each method based on its anomaly score. This practice ensures unbiased and consistent comparisons across methods.

## Implementations

Our model is trained with an Adam optimizer [84] at a learning rate of 0.001 for 30 epochs. The 20% of the training set is selected as a validation set where the best epoch is stored based on the validation loss. Note that each input sample is a consecutive time window. Each sample is extracted by sliding the multivariate time series along the time dimension with stride 1. We employ early stopping with a patience of 10 epochs. The hidden dimension  $d$  is set to 64, and the maximum number of TCN blocks is set to 8. The temperature  $\tau$  and the number of transformations  $K$  for DCL loss are set to 0.1 and 6, respectively. The sequence length is set to 30 and the prediction window size  $w$  is set to 5. In addition, all performance results are averaged over five independent runs.

Table 2.2: Performance comparison based on point-wise F1 ( $F1^p$ ), point-adjusted F1 ( $F1^*$ ) and composite F1 ( $F1^c$ ), with the best performance highlight in bold and second best underlined.

|              | SWaT        |             |             | WADI        |             |             | MSL         |             |             | SMAP        |             |             | HAI         |             |             |
|--------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Methods      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      |
| USAD         | 76.8        | 45.4        | 83.0        | 28.2        | 42.0        | 35.6        | 23.6        | 42.0        | 60.0        | 28.1        | 31.7        | 25.4        | 39.7        | <u>70.8</u> | 73.6        |
| MAD-GAN      | 73.6        | 44.4        | 80.5        | 15.9        | 36.6        | 63.4        | 20.2        | 28.4        | 77.1        | 24.3        | 28.6        | 67.4        | 5.7         | 7.2         | 17.5        |
| NeuTralAD    | 76.1        | <b>75.0</b> | 90.6        | 30.4        | <u>48.6</u> | 54.2        | 21.0        | 22.3        | 60.6        | 23.2        | 23.3        | 53.3        | 32.0        | 37.4        | 48.4        |
| TranAD       | <u>76.9</u> | 46.4        | 83.2        | 27.3        | 43.1        | 39.0        | 24.6        | 42.3        | 79.2        | 23.5        | 26.3        | 67.4        | 43.5        | <b>69.9</b> | 73.0        |
| AnomalyTrans | 12.9        | 32.2        | 89.6        | 5.0         | 10.4        | 66.6        | 8.9         | 27.1        | <u>93.8</u> | 8.3         | 24.5        | <u>96.4</u> | 5.3         | 17.2        | 56.5        |
| COCA         | 67.9        | 58.8        | 85.3        | 20.6        | 34.3        | 49.0        | 22.9        | 36.8        | 85.6        | 24.9        | 26.5        | 54.9        | 32.3        | 57.1        | 71.5        |
| DCdetector   | 21.9        | 22.7        | <b>91.8</b> | 11.0        | 15.7        | <b>83.2</b> | 19.3        | 25.6        | <b>96.1</b> | 24.2        | 26.2        | <b>96.6</b> | 3.9         | 4.5         | 19.2        |
| CNT          | <b>77.1</b> | <u>72.3</u> | <u>91.7</u> | <b>36.7</b> | <b>64.6</b> | 77.7        | <u>28.9</u> | <b>60.9</b> | 92.2        | <b>34.5</b> | <b>57.4</b> | 91.5        | <b>50.0</b> | 65.1        | <b>79.7</b> |

## 2.4.2 Comparison Study

Table 2.2 shows that CNT outperforms the baseline methods on average across all datasets in terms of the three metrics. Unlike USAD, TranAD, MAD-GAN, and AnomalyTransformer which focus on reconstructing pixel-level details, CNT emphasizes contrasting latent representations, making it more robust and less prone to overfitting. Compared to the other contrastive learning-based method (COCA, DCdetector), CNT employs more diverse augmentations at the latent level, leading to a deeper understanding of the data without relying on manual time series augmentations. Additionally, CNT surpasses NeutralAD by a notable margin. NeuTralAD focuses on the entire time window, which can dilute anomaly detection when normal trends dominate. In contrast, CNT emphasizes contextual semantic similarity, allowing it to detect subtle anomalies by identifying deviations in shorter prediction window relative to its historical context.

## 2.4.3 Ablation Study

We further compare the performance of CNT with several model variants. The performance is shown in Figure 2.3. The CCL setting or DCL setting trains the feature encoder using only  $L_{ccl}$  or  $L_{dcl}$ , respectively. CCL struggles with the representation collapse issue, which may prevent it from learning meaningful time series representations. whereas DCL focuses on matching input windows with their transformations but tends to miss short-term anomalies due to its focus on full windows. Thus, both show inferior performance compared to CNT.

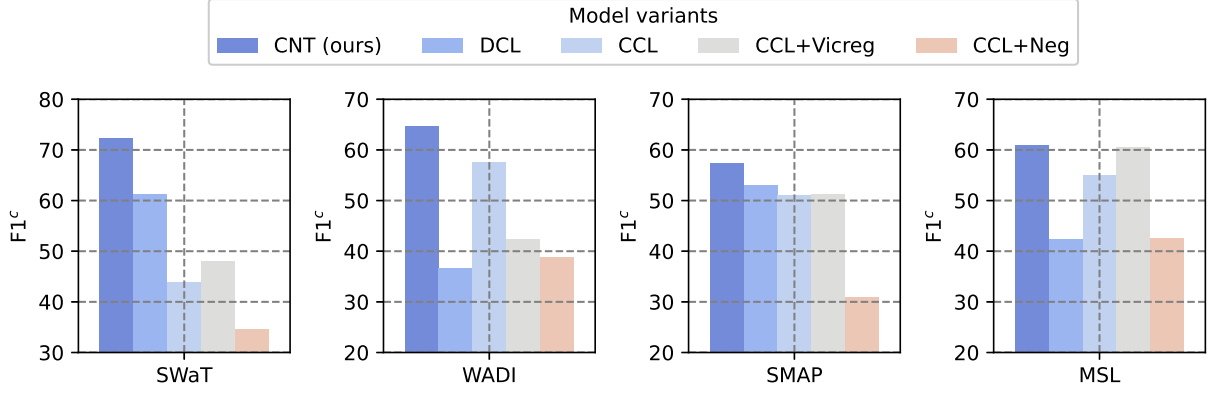


Figure 2.3: Performance comparison on different model variants in terms of  $F1^c$ .

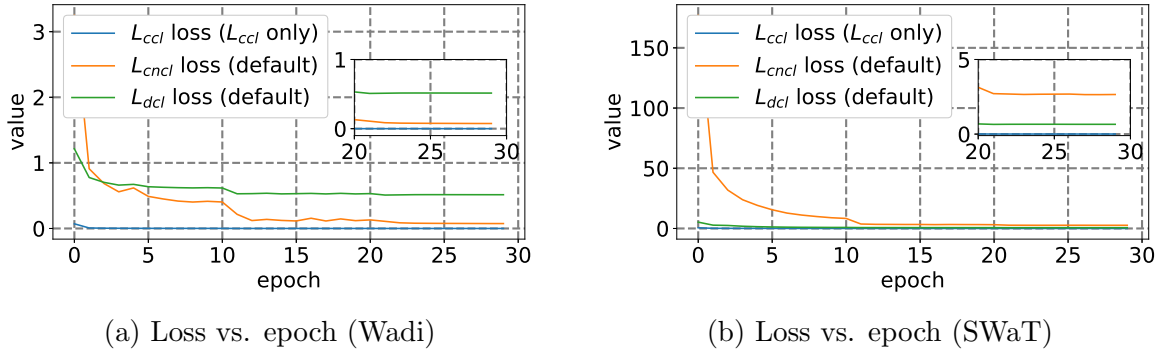


Figure 2.4: Loss value per epoch on Wadi and SWaT.

To explore alternative ways of preventing representation collapse without neural transformation learning, we test two additional variants: CCL + Neg, inspired by [121], introduces negative samples into the CCL framework by randomly selecting context windows from the batch as negatives and training with an infoNCE loss using  $l_2$  distance. CCL + Vicreg, based on [9], adds a regularization term to the CCL loss to ensure that the latent representations have non-zero variance, thus preventing constant encoder outputs. Overall, CNT outperforms all these variants on average across multiple datasets, which further validates the design choices in our model.

### 2.4.4 Analysis Of Training Losses

Figure 2.4 shows the loss values per epoch for our model training under purely  $L_{\text{ccl}}$  loss or training under both  $L_{\text{cncl}}$  and  $L_{\text{dcl}}$  (the default setting). It is clear that  $L_{\text{ccl}}$  decays quickly to 0 without the regularization of  $L_{\text{dcl}}$ . In contrast, with the additional constraints of  $L_{\text{dcl}}$ ,  $L_{\text{cncl}}$  decays much slowly compared to the former. This implies that  $L_{\text{dcl}}$  does add additional complexity to the CCL task, preventing from learning trivial feature representations. Furthermore, the loss value of  $L_{\text{dcl}}$  quickly falls below  $K \log K$  (4.67 when  $K = 6$ , the default parameter setting used in our experiments). This verifies that the assumption in Proposition 2.1 is empirically applicable.

### 2.4.5 Data Visualization

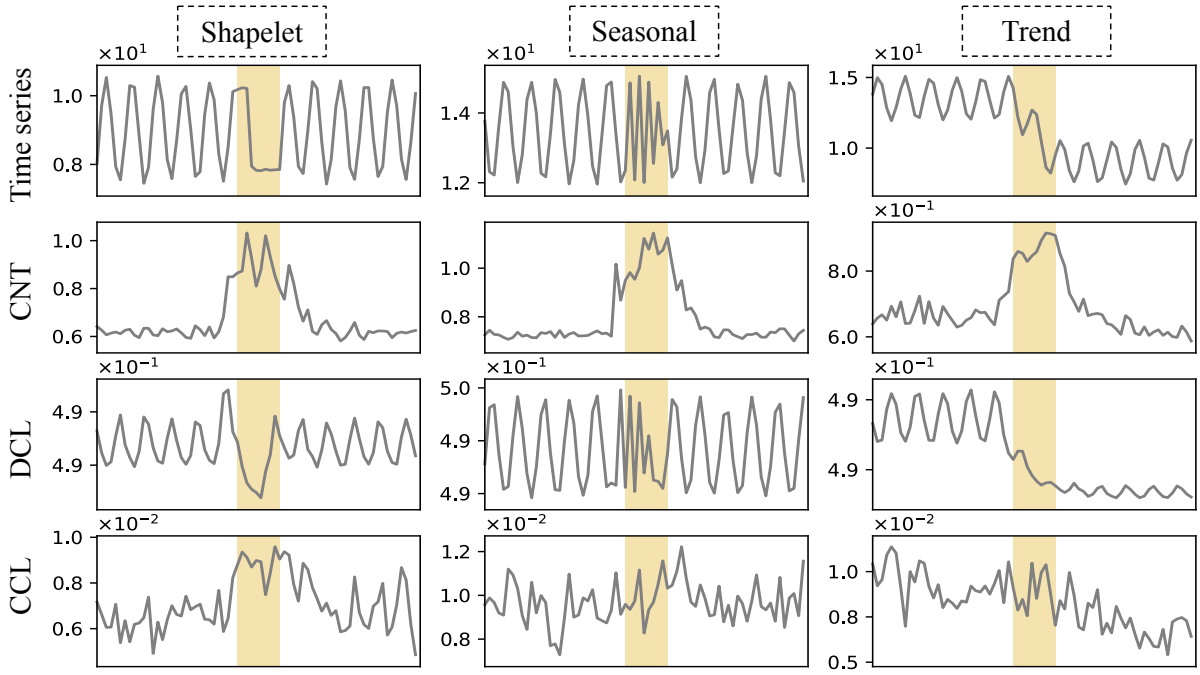


Figure 2.5: The top row shows the original time series, while subsequent rows display the anomaly scores from CNT, DCL, and CCL. Anomalous regions are highlighted in yellow.

We further conducted experiments on a univariate dataset (TODS) generated from the rules outlined in [91]. The TODS dataset adopts a sinusoidal wave as the base shapelet

to generate a univariate time series, and synthetic anomalies of various types are injected using predefined rules. Examples of different types of anomalies are shown in Figure 2.5. TODS is specifically chosen for data visualization as it offers clearly classified anomalies of different types, enabling a better understanding of the performance of our method. Furthermore, the univariate nature of the dataset makes it easy to visualize the results directly from the plot.

Figure 2.5 provides a visual comparative analysis of anomaly detection performances among the CNT, DCL, and CCL models. The first row reveals the original time series data, featuring various types of anomalies such as shapelet, seasonal, and trend-based disturbances, excluding point anomalies for their consistent detection across models. CNT distinguishes itself by generating distinctly elevated anomaly scores in areas of anomalies, demonstrating exceptional sensitivity to aberrant patterns. DCL, however, shows a reduction in performance, particularly with trend anomalies where its scores fail to capture the irregularities effectively, hinting at a potential under-detection of nuanced anomalies in predominantly normal data streams. CCL, while producing sharper scores than DCL, does not achieve the same level of discrimination as CNT, occasionally misidentifying normal regions as anomalous. This underperformance is likely a consequence of the representation collapse issue, wherein the feature encoder defaults to oversimplified representations that overlook temporal dynamics within the data. Overall, CNT demonstrates a superior capacity to detect a wide range of anomalies with its discriminative scoring, while the DCL and CCL models appear constrained by their focus on broader patterns and a vulnerability to representation collapse, respectively.

## 2.5 Conclusion

In our study, we developed a contextually-aware contrastive learning approach for time series anomaly detection. By utilizing a temporal-wise contrasting strategy, we effectively compared latent representations of the predict and context windows, while addressing the representation collapse problem by augmenting the contextual contrastive loss with neutral transformation learning. This refinement enhanced the complexity and effectiveness of our approach, resulting in superior performance over a variety of baseline time series anomaly detection methods on real-world industrial datasets.

## Chapter 3

# Multivariate Time Series Anomaly Detection via Dynamic Graph Forecasting

### 3.1 Introduction

Modern critical infrastructures have been increasingly relying on cyber-physical systems, which integrate computational elements with physical processes, rendering them to be highly interconnected and complex. These intricacies make them vulnerable to unforeseen events such as hardware malfunctions or cyber-attacks, potentially resulting in a system downtime or even a catastrophic failure. To address these risks, real-time monitoring powered by IoT technologies has become indispensable. The vast amounts of time series data generated by IoT sensors offer a unique opportunity for advanced time series anomaly detection (TSAD) techniques to automatically detect any abnormal system behaviors and ensure timely intervention.

Conventionally, univariate TSAD focuses on disruptions in temporal dynamics, such as irregularities in periodicity or trend. However, sensor-generated time series data often involve high-dimensional variables, requiring the integration of both temporal and cross-feature information for more robust anomaly detection. For instance, consider a water treatment system (Figure 3.1a), where under normal conditions, a high water outflow reading from the sensor ('FIT101') would typically follow a drop in a tank level ('LIT101'). During a cyber-attack (highlighted in the yellow region), a forced-open valve ('MV101') leads to a tank overflow, causing both 'LIT101' and 'FIT101' to simultaneously register

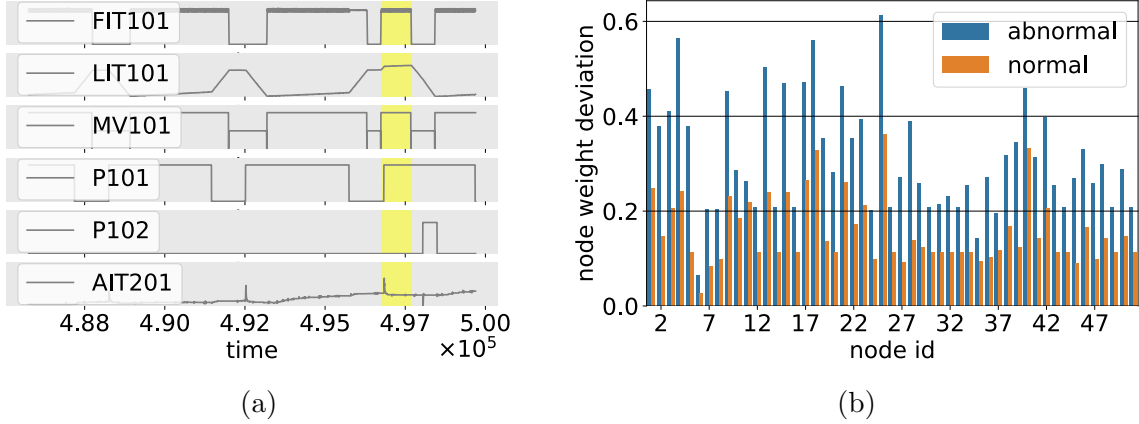


Figure 3.1: a) An event showing abnormal inter-series relations in the highlighted region; b) Node deviation between two consecutive graphs from normal states, or during a transition from a normal (abnormal) to abnormal (normal) state.

high values—an abnormal pattern. While this anomaly may be difficult to detect solely based on temporal irregularities, it becomes evident when considering the inter-series relationship, where the high values of both ‘LIT101’ and ‘FIT101’ persist together. This point underscores the importance of incorporating inter-variable relationships alongside temporal dynamics to improve anomaly detection performance.

Traditional statistical TSAD methods generally have solid mathematical foundations and good interpretability but have limited capability when working with large-scale high-dimensional data. Recent advances in neural networks have led to promising results in anomaly detection, with most of the deep learning methods for TSAD employing Recurrent Neural Networks (RNNs) [112], Convolutional Neural Networks (CNNs) [89], or Transformers [167] to model temporal dependency. However, these architectures do not explicitly learn inter-series relationships, limiting their ability to identify anomalies and provide interpretable results based on the deviation of inter-series relationships from normal to abnormal states.

Recent studies explore Graph Neural Networks (GNNs) [208] to model the relationships among time series. Since the graph structure is often not predefined in multivariate time series, the majority of these approaches rely on graph structure learning techniques to infer inter-series correlations [32, 26, 135, 58, 206]. For example, GDN [32] learns unique embeddings for each time series and constructs a top-K inter-series graph based on the pairwise similarity of these embeddings. The representations are then updated by aggregating information from relevant neighbors using Graph Attention Networks (GATs) [168].

FuSAGNet [58] takes this further by refining the inter-series relationship through recurrent sensor embeddings, which group sensors within the same process. To enhance computational efficiency, GTA [26] applies Gumbel sampling techniques [76] to learn the graph and prune the structure with sparsity constraints. These methods primarily rely on static graphs to assist in learning hidden time series representations and use time series forecasting or reconstruction errors to detect anomalies. However, a static graph may struggle to capture the evolving nature of inter-series relationships over time. Moreover, the time series forecasting or reconstruction objectives tend to emphasize comparisons of individual time points to their historical values, focusing on granular temporal patterns rather than detecting shifts in relationships between time series.

In this chapter, we present DyGraphAD, a dynamic graph-driven dual-task forecasting framework for multivariate TSAD, designed to explicitly detect both the abnormal inter-series and intra-series patterns. DyGraphAD is underpinned by the insight that inter-series relationships evolve differently in normal and abnormal states. To illustrate this point, we conduct a preliminary analysis by constructing a dense similarity graph at each time step, where nodes represent individual time series and edges capture the pairwise similarity between them. We then calculated node deviations—defined as the sum of edge weights for each node—between consecutive graphs. As shown in Figure 3.1b, transitions from normal to abnormal states exhibit sharper variations in the node deviation compared to transitions within the normal states.

Building on these observations, our framework directly incorporates dynamically changing inter-series relationships as an inductive bias, under the assumption that anomalies are likely to violate the evolving inter-series patterns observed in normal states. Specifically, DyGraphAD employs a sequence of similarity graphs, which are encoded through a dynamic graph encoder to capture both the evolving inter-series relationships and the long-term dependencies. These obtained graph latent representations are then leveraged to aid two distinct forecasting tasks: one is based on historical graphs and the other is based on past sequences. Large deviations in graph predictions indicate abnormal inter-series relationships, while significant errors in time series forecasting highlight abnormal temporal patterns. Anomalies are detected by examining the magnitude of these deviations, enabling the framework to effectively capture both abnormal relationships and temporal anomalies.

To summarize, our contributions in this chapter are as follows:

- **Graph Forecasting Framework for Anomaly Detection:** Our approach uniquely frames multivariate time series anomaly detection as a graph forecasting problem,

enhancing the detection of anomalies that disrupt the inter-series patterns during normal conditions.

- **Enhanced Detection through Joint Forecasting Tasks:** We couple the graph and time series forecasting tasks through a unified graph encoder, enabling explicit anomaly detection across both temporal and inter-series dimensions.
- **Integration of static and dynamic graphs:** We introduce a graph encoder that merges static and dynamic graphs to capture both long-term and short-term relationships.
- **Empirical Validation:** Extensive experiments on real-world industrial datasets demonstrate that our model is superior in terms of detection accuracy compared to a variety of baseline methods.

## 3.2 Related Works

The recent advancements in GNNs have increased their adoption in time series anomaly detection to model inter-series relationships. MTAD-GAT [201] leverages two parallel graph attention layers to capture inter-series and temporal dependencies, explicitly modeling correlations between different time-series features and timestamps. Anomaly scores are calculated as a combination of forecasting errors and reconstruction errors. Duo-GAT [94] extends this by incorporating a directed, weighted, time-oriented graph with a differencing-based GAT layer to capture temporal dependencies. GDN [32] constructs a top-K similarity graph where each series is treated as a node. It uses graph attention mechanisms for forecasting future time series values and identifies anomalies as deviations from these learned patterns. GTA [26] further enhances computational efficiency by employing Gumbel sampling to create a sparse and differentiable graph. FuSAGNet [58] generates sparsity-constrained latent representations of time series data using a sparse autoencoder and learns interdependencies among features with a GNN encoder. These components are jointly optimized, with anomaly scores calculated by aggregating reconstruction errors and forecasting errors. CST-GL [206] starts with a static learnable global embedding graph, using a coupled Graph Convolutional Network (GCN) [86] and Temporal Convolutional Network (TCN) [92] module to model spatial and temporal dependencies. anomaly score is calculated as the distance between PCA-reconstructed forecasting errors and the original normalized errors. STADN [165] employs GATs to model the relationships between sensors based on a dynamic graph constructed using correlation coefficients. Anomaly scores are

calculated based on prediction errors, which are refined through an error reconstruction module to enhance the discrimination between normal and anomalous instances.

The aforementioned methods primarily use time series forecasting or reconstruction errors to identify anomalies. In comparison, our framework directly operates on graphs, enabling more explicit detection of abnormal relationships. One method similar to ours is MSCRED [196], which reconstructs a series of inter-series matrices at different scales via an attention-based convolutional Long Short-Term Memory (convLSTM) [151] network. Our framework departs from it by using a dual forecasting objective, improving its ability to detect both abnormal relationships and finer-level temporal anomalies.

## 3.3 Methodology

### 3.3.1 Problem Formulation

In this chapter, we consider a multivariate time series  $X \in \mathbb{R}^{N \times T}$ , collected over  $T$  time ticks with  $N$  features, as the training data. The training sample set  $\mathcal{D}$  consists of sub-sequences extracted from  $X$ . Specifically,  $X$  is divided into a series of sub-sequences with stride 1. Each  $X^t \in \mathcal{D}$  represents a collection of time ticks within a sliding window at timestamp  $t$  of length  $c$ :  $X^t = \{X_{\cdot i}\}_{i=t-c+1}^t$ , where  $X_{\cdot i} \in \mathbb{R}^N$  is the multivariate time series at timestamp  $i$ . The goal is to assign an anomaly score to each time tick in the test set, which is a time series with the same  $N$  features collected over a different span of time ticks from the training set. Formally, for a given test time series  $X_{\text{test}} \in \mathbb{R}^{N \times T_{\text{test}}}$ , the model outputs an anomaly score vector  $S \in \mathbb{R}^{T_{\text{test}}}$ , where each element  $S_t$  represents the anomaly score at time tick  $t$ . These scores can later be used to identify anomalies through thresholding.

### 3.3.2 Overview

In this section, we give a brief overview of the proposed framework: At the beginning, a sequential list of inter-series graphs is constructed for each input sample in the pre-processing stage (Section 3.3.3). Next, these graphs are encoded into hidden representations through a graph encoder module. This module refines the graph structure to capture both short-term and long-term inter-series relations. It then applies graph convolutions to generate latent representations for graphs (Section 3.3.4). The feature representations of graphs then serve as the input to a graph forecasting module (Section 3.3.5) and a time series forecasting

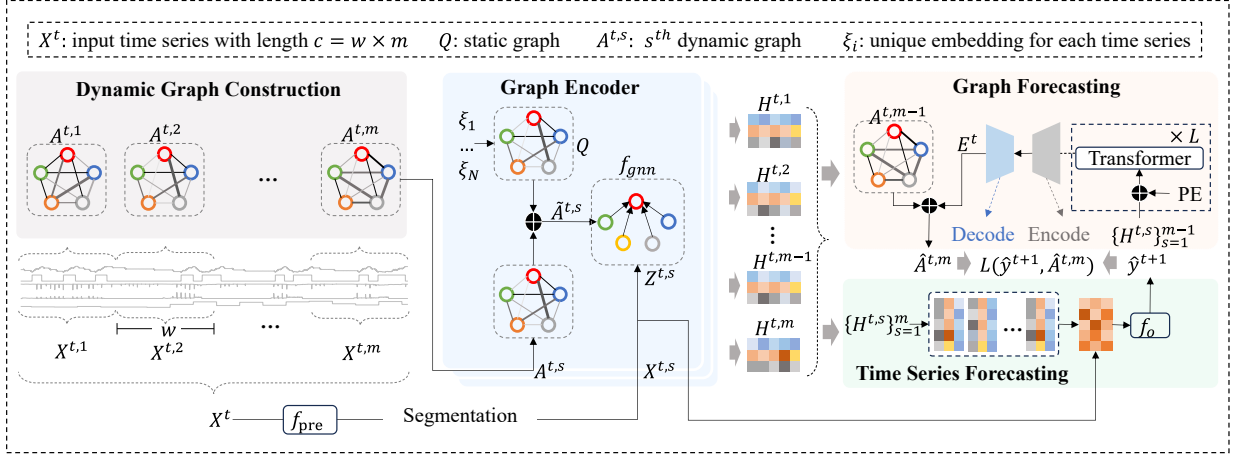


Figure 3.2: An illustration of the DyGraphAD framework.

module (Section 3.3.6). The former uses a Transformer module as the backbone to predict the latest graph based on historical graphs, while the latter forecasts the next-ahead time series value based on the concatenated graph representations along a time axis. The two tasks are optimized jointly during the training stage, and generate a hybrid score during the inference stage to detect anomalies (Section 3.3.7). An illustration of the DyGraphAD framework is shown in Figure 3.2.

### 3.3.3 Construction of Dynamic Graphss

Before training, each input sequence is broken into a list of disjoint segments in order to construct a series of similarity matrices. The purpose of building these matrices is to facilitate the graph forecasting task, i.e., given a set of historical similarity matrices, the goal is to predict the one-step ahead future matrix, which should be readily available during training.

In this chapter, we employ the DTW metric [144] to calculate the similarity between pairwise time series. However, our proposed framework is flexible with respect to other time series distance or similarity measures. We adopt the fast C implementation of DTW from [192] which solves the optimal alignment match problem by using a dynamic programming technique.

Specifically, given each sample  $X^t$  with length  $c$ , where  $c = m \cdot w$ ,  $m$  sequential adjacency matrices  $\{A^{t,s}\}_{s=1}^m$  are constructed, where  $A^{t,s} \in \mathbb{R}^{N \times N}$  is calculated based on the time

series segment  $X^{t,s} = \{X_{t-c+w(s-1)+1}, \dots, X_{t-c+ws}\}$  with a window size of  $w$ .  $w$  and  $m$  are two hyper-parameters tuned for each specific dataset. An entry in  $A^{t,s}$  represents the normalized similarity (i.e., between 0 and 1) between the pairwise time series. Since DTW calculates an unbounded distance between two temporal sequences, we convert the entry of adjacency matrix  $A^{t,s}$  within the range of 0 and 1 by

$$A_{i,j}^{t,s} = \exp \left( -\frac{DTW^2(X_{i\cdot}^{t,s}, X_{j\cdot}^{t,s})}{\tau} \right) \quad (3.1)$$

where  $\tau$  is a hyper-parameter.  $\{A^{t,s}\}_{s=1}^m$  are referred to as dynamic graphs throughout this chapter.

### 3.3.4 Graph Encoder Module

The graph encoder module aims at encoding the dynamic graphs into low-dimensional feature representations. Given an input sample  $X^t$ , we first project it into latent representations  $Z^t \in \mathbb{R}^{d \times N \times c}$ , where  $d$  is a hidden dimension.  $Z^t$  is then broken into a list of disjoint segments  $\{Z^{t,s}\}_{s=1}^m$  along the time dimension ( $3^{rd}$  dimension). Each dynamic graph  $A^{t,s}$  is subsequently encoded through GNNs with  $Z^{t,s}$  as the corresponding node feature. The operations are represented by the following equations:

$$Z^t = f_{pre}(X^t) \quad (3.2)$$

$$Z^{t,s} = \text{Segment}^{(s)}(Z^t) \quad (3.3)$$

$$H^{t,s} = f_{gnn}(Z^{t,s}, A^{t,s}) \quad (3.4)$$

where  $f_{pre}(\cdot)$  is a 2D convolutional layer utilizing a combination of kernel sizes  $\{2, 3, 5, 7\}$ , which are applied and concatenated. The intuition behind  $f_{pre}(\cdot)$  is that aggregating nearby points through different kernel sizes helps the model learn the similarity between pairwise time series up to a certain time shift, thereby mimicking the operation of the DTW distance.  $\text{Segment}^{(s)}(Z^t)$  corresponds to the  $s$ -th segment of length  $w$  from  $Z^t$ , coinciding with the same time window for  $A^{t,s}$ . The function  $f_{gnn}(\cdot, \cdot)$  is implemented by using the MixHop graph convolution [181].

Due to the use of relatively short time windows for constructing each dynamic graph, they are more susceptible to noise. Additionally, long-term inter-series correlation often persist in practical scenarios. For example, in water distribution systems, the correlation between water flow and pressure at different points in the network can remain stable over long periods despite short-term fluctuations. Building on this insight, we replace  $A^{t,s}$  in

Eq. 3.4 with  $\tilde{A}^{t,s}$ , which is calculated as a learnable weighted combination of the dynamic graph  $A^{t,s}$  and a static global graph  $Q$ :

$$\tilde{A}^{t,s} = \sigma(W_1) \odot Q + (1 - \sigma(W_1)) \odot A^{t,s} \quad (3.5)$$

with the entry of  $Q$  being defined as:

$$Q_{ij} = f_a(\xi_i \parallel \xi_j) \odot \sigma(f_m(\xi_i \parallel \xi_j)) \quad (3.6)$$

Here,  $\xi_i \in \mathbb{R}^d$  is a learnable embedding representing time series  $i$ ,  $W_1 \in \mathbb{R}^{N \times N}$  is a weight matrix, and  $\sigma$  is a sigmoid function. The functions  $f_a(\cdot) : \mathbb{R}^{2d} \rightarrow \mathbb{R}$  and  $f_m(\cdot) : \mathbb{R}^{2d} \rightarrow \mathbb{R}$  are two-layer fully connected neural networks, which are used for generating graph entries and pruning the graph structure, respectively.

Intuitively, the static graph represents long-term dependencies by capturing stable correlations across the series, while dynamic graphs focus on short-term variations, adapting to changes in inter-series relationships over time. Combining both enables the graph encoder to learn a more comprehensive representation of temporal dependencies.

### 3.3.5 Graph Forecasting Module

The graph forecasting module aims at predicting the next-ahead dynamic graph based on the historical graphs under normal conditions. Thereby, abnormal inter-series relationships can be detected if the predicted graph deviates substantially from the true graph.

Specifically, the module follows an encoder-decoder structure. The encoder is implemented by using a Transformer network with a learnable positional encoding layer, designed to preserve positional information within the sequence of graphs and learn the temporal dynamics among them. Accordingly, given  $\{H^{t,s}\}_{s=1}^{m-1}$ , the graph encoder outputs, we obtain their encoded feature representations given by:

$$\tilde{H}^{t,s} = \text{Pooling}(H^{t,s}) \quad (3.7)$$

$$\tilde{H}^t = \text{Concat}(\{\tilde{H}^{t,s}\}_{s=1}^{m-1}) \quad (3.8)$$

$$P^t = \text{TransformerEnc}(\tilde{H}^t + \text{PE}(\tilde{H}^t)) \quad (3.9)$$

In this process, Pooling involves taking the mean along the time axis of  $H^{t,s} \in \mathbb{R}^{d \times N \times w}$  to obtain  $\tilde{H}^{t,s} \in \mathbb{R}^{d \times N}$ . These pooled representations are then concatenated to form  $\tilde{H}^t \in \mathbb{R}^{d \times (m-1) \times N}$ , which serves as the input to the Transformer encoder with a sequence length of  $(m-1)$ . Following [167], the Transformer encoder consists of  $L$  layers, with each incorporating multi-head attention (MHA) and layer normalization (LN) components.

Next, we predict the next-ahead dynamic graph  $\hat{A}^{t,m}$  through the following decoding process:

$$O^t = \text{Pooling}(P^t) \quad (3.10)$$

$$J^t = f_d(O^t) \quad (3.11)$$

$$E^t = J^t \cdot J^{t\top} \quad (3.12)$$

$$\hat{A}^{t,m} = \sigma(W_2) \odot E^t + (1 - \sigma(W_2)) \odot A^{t,m-1} \quad (3.13)$$

where the Pooling operation is applied over the segment dimension.  $f_d(\cdot)$  is a two-layer fully connected neural network followed by a  $l_2$  normalization, and  $W_2 \in \mathbb{R}^{N \times N}$  is a learnable weight matrix. Here,  $\hat{A}^{t,m}$  is calculated as a weighted combination of the most recent graph  $A^{t,m-1}$  and the decoded graph  $E^t$ . Given that graph structures typically vary smoothly over time rather than undergoing drastic changes within short periods, we incorporate  $A^{t,m-1}$  to assist the graph decoding process, as the most recent graph at  $(m-1)$ -th step can serve as a good approximation for the  $m$ -th dynamic graph.

### 3.3.6 Time Series Forecasting Module

One limitation of the graph forecasting module is its reliance on discrete graphs constructed from short-time windows, which essentially represent a coarser approximation of the original time series data. Consequently, this approach may fail to capture critical finer-level temporal dynamics within individual time series. To mitigate this problem, we propose to couple the graph forecasting module with a time series forecasting module. This module operates explicitly on the time series rather than on the graphs, with the objective of predicting the next-step ahead time series values. In doing so, it enables a more precise representation of temporal information and emphasizes how time series behave with respect to their past.

Accordingly, given the outputs  $\{H^{t,s}\}_{s=1}^m$  from the graph encoder, we concatenate them along the time axis to obtain the time series latent representation  $M^t \in \mathbb{R}^{d \times N \times c}$ . Unlike the raw sample  $X^t$ , each time series in  $M^t$  contains information propagated from other time series based on the inter-series relationship through a graph convolution.

Next, the predicted next-ahead time series value,  $\hat{y}^{t+1}$ , is calculated as:

$$\hat{y}^{t+1} = f_o(f_{c_1}(Z^t) + f_{c_2}(M^t)) \quad (3.14)$$

where  $f_{c_1}, f_{c_2} : \mathbb{R}^{d \times N \times c} \rightarrow \mathbb{R}^{d \times N}$  are 2D convolution layers, and  $f_o : \mathbb{R}^{d \times N} \rightarrow \mathbb{R}^N$  is a two-layer fully connected neural network.

### 3.3.7 Training Objective and Anomaly Score

During the training stage, the model is trained with an objective to minimize the forecasting loss of both the next-step ahead time series values and the latest dynamic correlation graph at each time tick. Specifically, the overall loss function is defined as:

$$\mathcal{L} = \mathbb{E}_{X^t \sim \mathcal{D}} \left[ \frac{1}{N} \|y^{t+1} - \hat{y}^{t+1}\|_2^2 + \frac{1}{N^2} \|A^{t,m} - \hat{A}^{t,m}\|_F^2 \right] \quad (3.15)$$

where  $y^{t+1}$  and  $A^{t,m}$  are the multivariate time series value at time step  $(t+1)$  and the  $m^{th}$  dynamic correlation graph.

During the inference stage, the anomaly score of each sample is calculated based on both the graph forecasting error and time series forecasting error. Since the two scores have different scales, we define the anomaly score at a given time step  $t$  for each time series  $i$  as

$$\text{Err}_i^t = \frac{1}{\frac{\lambda}{\text{Err}_{i,\text{ts}}^t} + \frac{1-\lambda}{\text{Err}_{i,\text{graph}}^t}} \quad (3.16)$$

where  $\text{Err}_{i,\text{ts}}^t$  is a squared error of the  $i^{th}$  predicted time series value at  $t$  and  $\text{Err}_{i,\text{graph}}^t$  is a mean squared error at the  $i^{th}$  row of the predicted dynamic correlation graph at  $t$ . In practice, we find that  $\lambda = 0.5$  on average achieves the best performance. Finally, the anomaly score at  $t$  is obtained as the average score of  $N$  time series.

## 3.4 Experiments

### 3.4.1 Experimental Setting

#### Datasets

We evaluate our framework using five widely-adopted benchmark datasets for Time Series Anomaly Detection (TSAD): SWaT [111], WADI [2], SMD [157], MSL [72], and SMAP [72]. Each dataset provides an unlabeled training set capturing normal operational data, along with a separate test set that includes labeled anomalies, allowing for robust evaluation across varied industrial and operational settings. The SWaT dataset was collected from a secure water treatment testbed, a controlled environment that emulates large-scale urban water systems. It contains 51 sensor dimensions recorded over 11 days, with cyber

attacks introduced during the last 4 days. Similarly, the WADI dataset is derived from a water distribution testbed and includes 123 channels of sensor data recorded over 16 days, with attacks intermittently introduced in the final 2 days. The SMD dataset, a multi-entity dataset collected from server machines at a large internet company, provides data from 28 different server entities. The performance of SMD is calculated by averaging over these entities. NASA’s MSL and SMAP datasets contain telemetry data from the Mars Science Laboratory rover and the Soil Moisture Active Passive satellite, respectively. These datasets capture anomalies that occur during spacecraft operations, marked by unexpected events or deviations from standard operating behaviors. This collection of datasets enables evaluation across a variety of anomaly types and domains, with summary statistics presented in Table 3.1.

Table 3.1: Summary statistics for datasets

| Dataset | avg. # channels | # training | # test | % anomalies |
|---------|-----------------|------------|--------|-------------|
| SWaT    | 51              | 47440      | 44993  | 12.13       |
| WADI    | 123             | 78459      | 17281  | 5.77        |
| MSL     | 55              | 58317      | 73729  | 10.53       |
| SMAP    | 25              | 135183     | 427617 | 12.79       |
| SMD     | 38              | 25300      | 25300  | 4.21        |

## Baselines

We select the following recent anomaly detection approaches as our baseline methods: LSTM-VAE [128], MAD-GAN [96], USAD [7], MSCRED [196], MTAD-GAT [201], GDN [32], FuSAGNet [58], TranAD [166], and CST-GL [206]. Among them, LSTM-VAE, MAD-GAN, USAD, and TranAD are methods that adopt LSTMs, Transformers, or MLPs as the backbone time series encoder. Whereas GDN, FuSAGNet, MTAD-GAT, and CST-GL utilize GNNs as the backbone encoder, and employ either a time series forecasting or reconstruction-based objective. MSCRED converts time series into sequential graphs and relies on a graph reconstruction task for anomaly detection.

## Evaluation metrics

We use the point-wise F1 (denoted as  $F1^p$ ), point-adjusted F1 (denoted as  $F1^*$ ) [157] and composite F1 (denoted as  $F1^c$ ) [49] score as the performance metrics.

The point-wise F1 is calculated using standard precision and recall at the point level:

$$F1^p = 2 \times \frac{Pr \times Re}{Pr + Re} \quad (3.17)$$

where precision (Pr) and recall (Re) are based on true positive, false positive, and false negative counts for individual time points.

For the point-adjusted F1, we follow the point-adjusting strategy suggested in [157], which treats an entire anomaly segment as correctly detected if at least one point in the segment is identified. This is computed as:

$$F1^* = 2 \times \frac{Pr^* \times Re^*}{Pr^* + Re^*} \quad (3.18)$$

where precision ( $Pr^*$ ) and recall ( $Re^*$ ) are modified to count anomaly segments rather than individual points.

Composite F1 is the harmonic mean of time-wise precision and event-wise recall. As proposed in [49], it is calculated as:

$$\begin{aligned} Pr_t &= \frac{TP_t}{TP_t + FP_t} \\ Re_e &= \frac{TP_e}{TP_e + FN_e} \\ F1^c &= 2 \times \frac{Pr_t \times Re_e}{Pr_t + Re_e} \end{aligned}$$

where  $TP_t$  and  $FP_t$  count individual time-point true positives and false positives, while  $TP_e$  and  $FN_e$  count event-level true positives and false negatives.

Given that  $F1^*$  may overestimate recall [83], and  $F1^p$  does not account for event lengths [83],  $F1^c$  offers a balanced measure that integrates precision and recall across different granularities of anomaly evaluation. For all F1 metrics, we identify the best F1 score by optimizing the threshold for each method based on its anomaly score. This practice ensures unbiased and consistent comparisons across methods.

## Implementations

Our model is trained with an Adam optimizer [84] at a learning rate of 0.001 for 50 epochs. 20% of the training set is selected as a validation set where the best epoch is stored based on the validation loss. Early stopping is employed with a patience of 10. The hidden dimension is set to  $d = 64$  for all modules. The number of MixHop graph convolution layers is set to 2, and the number of transformer blocks is set to  $L = 2$ . The temperature  $\tau$  used to construct the dynamic graphs for SWaT, SMAP, MSL, WADI, and SMD are 1, 1, 5, 5, and 0.5, respectively. The step size and the window size are set to  $m = 6$  and  $w = 5$  for all of the datasets. All results are averaged over 5 independent runs.

Table 3.2: Performance comparison based on point-wise F1 ( $F1^p$ ), point-adjusted F1 ( $F1^*$ ), and composite F1 ( $F1^c$ ), with the best performance highlight in bold and second best underlined.

|           | SWaT        |             |             | WADI        |             |             | MSL         |             |             | SMAP        |             |             | SMD         |             |             |
|-----------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Methods   | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      |
| LSTM-VAE  | 77.2        | 52.4        | 81.9        | 36.6        | 51.7        | 64.6        | 20.1        | 37.9        | 87.5        | 22.7        | 22.7        | 69.5        | 40.3        | 53.7        | 81.8        |
| USAD      | 76.8        | 45.4        | 83.0        | 28.2        | 42.0        | 35.6        | 23.6        | 42.0        | 60.0        | <u>28.1</u> | 31.7        | 25.4        | 42.3        | 56.0        | 82.1        |
| MAD-GAN   | 73.6        | 44.4        | 80.5        | 15.9        | 36.6        | 63.4        | 20.2        | 28.4        | 77.1        | 24.3        | 28.6        | 67.4        | 11.8        | 13.9        | 41.8        |
| TranAD    | 76.9        | 46.4        | 83.2        | 27.3        | 43.1        | 39.0        | 24.6        | 42.3        | 79.2        | 23.5        | 26.3        | 67.4        | 46.1        | 68.3        | 86.8        |
| MSCRED    | 75.9        | 37.1        | 82.9        | 29.6        | 36.4        | 50.6        | 27.9        | 41.7        | 81.1        | 23.0        | 23.6        | 63.2        | <u>48.1</u> | 53.9        | 77.9        |
| MTAD-GAT  | 76.4        | 50.9        | 83.2        | 30.3        | 44.9        | 46.3        | 29.7        | <b>67.7</b> | 94.2        | <b>29.0</b> | 35.0        | 91.3        | 45.5        | 66.7        | 86.1        |
| GDN       | 77.5        | 50.7        | 84.3        | <b>49.5</b> | <b>64.9</b> | 72.1        | 23.9        | 41.7        | 84.0        | 23.6        | 31.5        | 82.2        | <b>53.5</b> | 71.6        | 89.2        |
| FuSAGNet  | <b>81.9</b> | 55.7        | 88.5        | 48.0        | 59.2        | 67.0        | 27.3        | <u>64.4</u> | 89.0        | 22.7        | 43.6        | 91.1        | 43.5        | 52.0        | 79.9        |
| CST-GL    | 76.8        | 59.1        | 86.3        | 34.1        | 58.2        | 65.7        | 22.8        | 24.4        | 87.4        | 24.7        | 31.3        | 71.2        | 42.5        | 69.0        | 90.4        |
| DyGraphAD | <u>79.0</u> | <b>78.5</b> | <b>92.3</b> | <u>44.2</u> | <u>63.9</u> | <b>91.0</b> | <b>30.1</b> | 53.6        | <b>95.8</b> | 25.4        | <b>48.0</b> | <b>94.3</b> | 32.6        | <b>71.9</b> | <b>96.6</b> |

### 3.4.2 Comparison Study

The performance comparison in Table 3.2 demonstrates the effectiveness of DyGraphAD against various anomaly detection methods. Our method achieved the highest performance across all datasets in terms of the point-adjusted F1. While being slightly behind FuSAGNet in terms of the point-wise F1 in certain datasets, DyGraphAD maintains on average the best  $F1^c$  scores, indicating its superiority in balancing between the time-wise precision and event-wise recall.

For other methods, Graph-based frameworks such as FuSAGNet, GDN, and MTAD-GAT have relatively better performance compared to other methods. These methods infer

inter-series relationships through graph structure learning and employ GNNs to refine time series representations. However, our model still outperform them by a notable margin. Specifically, DyGraphAD incorporates an additional graph forecasting task, enabling more explicit detection of abnormal relationships. Moreover, it models inter-series relationships from both short-term and long-term perspectives, thus providing a more nuanced understanding of relations among time series.

Additionally, our model outperforms MSCRED, a similar framework that utilizes sequential graphs to detect inter-series anomalies explicitly. We attribute the performance degradation of MSCRED to its weakness in modeling temporal patterns, as it operates only on graphs, which are a coarser representation of time series data. This limitation makes MSCRED less effective in capturing the nuanced temporal patterns that DyGraphAD can identify through its dual loss functions.

### 3.4.3 Ablation Study

We conducted an ablation study on four datasets to evaluate the impact of each sub-module by removing individual components from the default model. The results are presented in Figure 3.3. Both the graph and time-series forecasting tasks perform well individually, but their combination improves detection accuracy across all datasets, demonstrating the benefit of joint forecasting. Additionally, the removal of dynamic or static graphs results in a moderate performance decrease, this highlights the importance of leveraging both short-term inter-series dynamics and long-term relationships in the model. Lastly, removing the recent graph predictor leads to a notable drop in performance across most datasets and performance metrics, which aligns with our assumption that the inter-series relationships do not vary drastically over recent time steps under normal operations.

### 3.4.4 Data Visualization

#### Combined Effects of Dual Forecasting

We examine the combined effects of the time series and graph forecasting tasks through visual inspections in Figure 3.4a & 3.4b. First, the graph forecasting task is more sensitive to feature-wise information, as evidenced by successfully identifying the highlighted anomaly which has an abnormal inter-series relationship in Figure 3.4a (Recall that in Figure 1a, sensor ‘FIT101’ and ‘LIT101’ are incorrectly positively correlated in the abnormal state). Notably, this anomaly is missed by the time series forecasting task as it has less

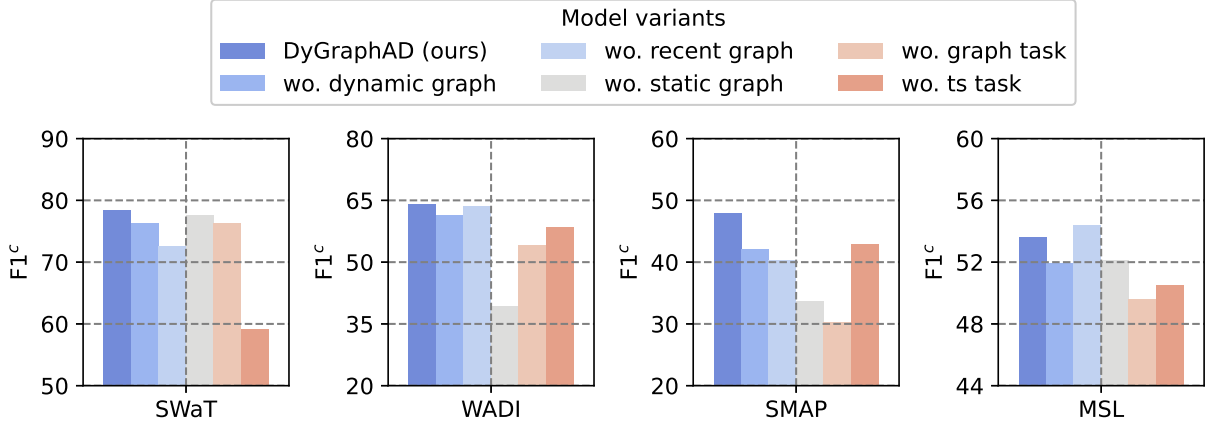
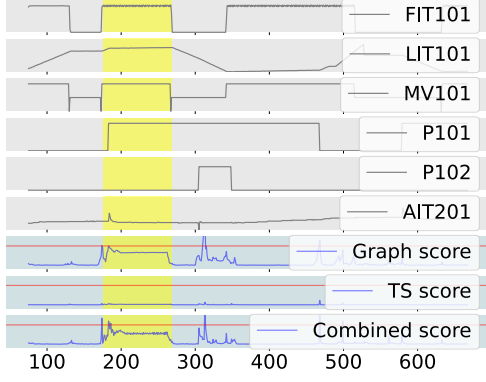


Figure 3.3: Performance comparison on different model variants in terms of F1<sup>c</sup>.

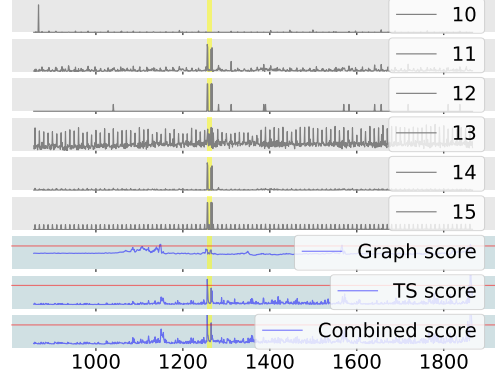
noticeable deviation error in temporal patterns. On the contrary, the time series forecasting task demonstrates proficiency in pinpointing short-span anomalies in the SMD dataset that evade detection by the graph forecasting task due to its coarser granularity (shown in Figure 3.4b). In essence, although both tasks are able to learn temporal patterns and feature-wise relationships to some extent, each of them has a better focus on a different data dimension. Thus, combining them can eventually improve the anomaly detection performance.

### Comparison among Graph-based Methods

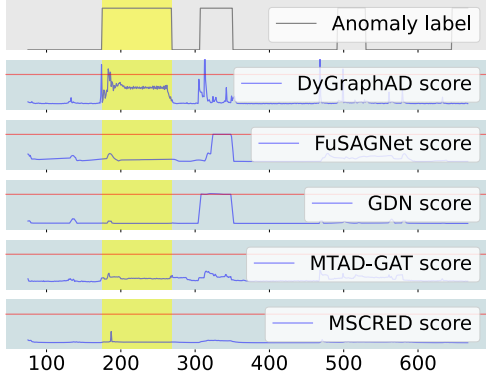
To delve deeper into the variance among graph-based methods, we illustrate their anomaly scores tied to the time series segment in Figure 3.4a. All other models fail to detect the abnormal relationship as indicated by their scores under the red threshold line in Figure 3.4c. DyGraphAD, however, identifies this anomaly due to a notable disparity between the predicted and actual graph evolution patterns, as shown in Figure 3.4d. On the contrary, MSCRED, another graph deviation-based method, presents a considerably lower score than its threshold, mainly attributed to the accurate reconstruction of its similarity matrices in both normal and abnormal stages (shown in the lower two rows of Figure 3.4d). This is a repercussion of an issue inherent to the autoencoder, which often learns an identity mapping function if its architecture lacks proper compression to extract normal samples' salient features. Additionally, MSCRED relies on convLSTM encoder layers which are inadequate for modeling graph-structured data where adjacent points



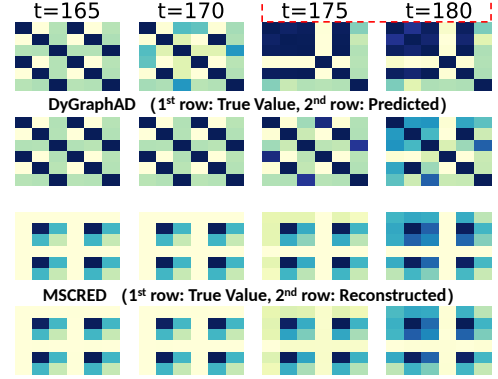
(a) An anomaly missed by the time series forecasting task.



(b) An anomaly missed by the graph forecasting task.



(c) Anomaly scores generated by graph-based models for the anomaly region in (a).



(d) Comparing predicted with true dynamic graphs (DyGraphAD vs. MSCRED).

Figure 3.4: (a)&(b) are time series segments from SWaT and SMD. Anomaly scores and thresholds from the graph forecasting, the time series forecasting, and the combination of them are shown in the bottom three rows, respectively; (c) is the anomaly scores from the graph-based models corresponding to the time series segment in (a); (d) is the dynamic graphs constructed by DyGraphAD or MSCRED around the highlighted anomaly in (a).

might not be related. To address these issues, our model employs a forecasting framework with a graph encoder more fitting for such data.

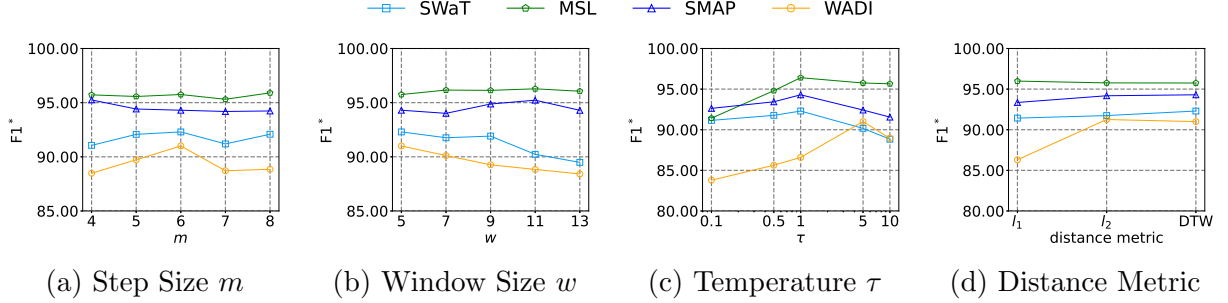


Figure 3.5: F1\* against different parameters, while the other parameters fixed at their default values.

### 3.4.5 Hyper-parameter Analysis

#### Window size

Figure 3.5b shows the variation of F1\* scores across different window sizes. The performance trend varies slightly by dataset: SMAP and MSL generally show stable performance with smaller window sizes, while WADI and SWaT demonstrate a more noticeable decline in F1\* as the window size increases. This is likely because smaller window sizes are more effective for capturing fine-grained temporal dynamics, which are critical for detecting short-duration anomalies. In contrast, larger windows may dilute these granular patterns, hindering the graph forecasting task and reducing anomaly detection accuracy. The results emphasize the importance of tuning the window size for dataset-specific requirements.

#### Step size

Figure 3.5a illustrates the variation of F1\* scores with different step sizes. Performance differs across datasets without explicit trends, likely due to the variability in the inter-series relationships of these datasets from normal to abnormal states. Learning evolving patterns across smaller step sizes can still greatly improve anomaly detection performance. However, the optimal step size appears to be dataset-dependent, highlighting the need for careful tuning to maximize the model's effectiveness.

### Graph temperature $\tau$

Figure 3.5c illustrates the variation of  $F1^*$  scores with respect to the temperature parameter  $\tau$  in Eq. (3.1), showing differing sensitivities across datasets. For example, datasets like SMAP and SWaT exhibit relatively stable performance across a wide range of  $\tau$  values, whereas WADI shows a more pronounced sensitivity. The parameter  $\tau$  plays a critical role in controlling the sparsity of the graph by emphasizing or attenuating pairwise similarity values, which helps filter out noisy relationships. This is particularly useful in real-world scenarios, where only a subset of time series has a significant influence on any given series, with the rest predominantly contributing noise. Overall,  $\tau$  is a key hyper-parameter that requires careful tuning to optimize performance for specific datasets. Notably, we observe that anomaly detection performance is positively correlated with the forecasting error on the validation set, suggesting that forecasting error can serve as a practical criterion for identifying the optimal  $\tau$  value.

### Graph similarity measure

Figure 3.5d shows the variation in  $F1^*$  scores in relation to different distance measures used to construct the dynamic graphs. Specifically, we evaluated two additional common distance metrics besides DTW (Manhattan ( $l_1$ ) and Euclidean ( $l_2$ )). DTW consistently achieves slightly higher scores across most datasets, particularly for SMAP and SWaT, while Manhattan and Euclidean distances also show competitive performance. This demonstrates the flexibility of our framework in accommodating various graph similarity measures without a significant impact on overall anomaly detection performance. There is also a competitive advantage, as using simpler metrics like Manhattan and Euclidean distances can reduce computational costs while maintaining strong detection performance.

## 3.5 Conclusion

In this work, we introduced DyGraphAD, a novel multivariate time series anomaly detection framework that combines dynamic graph forecasting with time series forecasting to capture both intra-series and inter-series anomalies. Our framework is designed around two complementary forecasting tasks. The graph forecasting module captures abnormal inter-series relationships by predicting future graphs based on historical ones, while the time series forecasting module highlights deviations within individual series. By jointly optimizing these tasks, DyGraphAD bridges the gap between coarse-grained inter-series

and fine-grained temporal anomaly detection. Moreover, the integration of static and dynamic graphs enhances the model’s ability to handle both stable long-term correlations and rapidly changing short-term patterns. This dual forecasting strategy not only improves the detection of subtle and complex anomalies but also provides greater interpretability by disentangling anomalies along temporal and relational dimensions. We validated the efficacy of DyGraphAD through extensive experiments on multiple real-world benchmark datasets. Our results demonstrate that DyGraphAD consistently outperforms a variety of baselines, showcasing its robustness and versatility across diverse industrial and operational settings.

# Chapter 4

## Prospective Multi-Graph Cohesion for Multivariate Time Series Anomaly Detection

### 4.1 Introduction

Time series anomaly detection (TSAD) is a process of pinpointing unusual or unexpected observations within time series data that deviate significantly from an expected pattern. This research area has extensive applications across industries, including device monitoring, cyber-security, and fraud detection [5, 138, 29, 97]. In recent years, the proliferation of IoT sensors has resulted in vast amounts of high-dimensional data, elevating the importance of multivariate TSAD. Analyzing relationships among multiple time series allows for detecting anomalies that may be invisible in isolated univariate analysis. By leveraging these inter-series dependencies, multivariate TSAD can uncover complex patterns and thus enhance the effectiveness of anomaly detection.

Existing TSAD methods can be categorized into supervised and unsupervised learning techniques. Due to the inherent scarcity of labeled anomalies in real-world datasets, unsupervised learning is often preferred, assuming all training examples represent normal behavior. Traditional unsupervised anomaly detection methods, such as OCSVM [146], PCA [154], and ARIMA [12], have solid mathematical foundations and offer good interpretability. However, their performances may be hampered when confronted with large-scale, high-dimensional data. Recent advancements have seen a shift toward deep learning-based methods, which excel in handling various complex data types. These methods pri-

marily utilize neural network architectures such as Convolutional Neural Networks (CNNs) (e.g., [195, 65]), Recurrent Neural Networks (RNNs) (e.g., [110, 157, 72, 24]), or Transformers (e.g., [184, 99, 188, 44]) to process time series data and detect anomalies through forecasting errors or reconstruction errors. Despite their efficacy in capturing temporal dependencies, these neural networks are not inherently designed to explicitly model the relationships among variables, which is crucial for uncovering anomalies in high-dimensional datasets.

Recently, there has been a growing trend in employing Graph Neural Networks (GNNs) to better characterize inter-series relationships within multivariate time series data. These methods typically construct learnable graphs to represent inter-series relationships and generate rich representations through GNNs based on the learned graphs. For instance, GDN [32], GTA [26], and FuSAGNet [58] construct an inter-series graph based on global learnable embeddings to describe the long-term relationships among time series. Similarly, MTAD-GAT [201], GReLeN [199] and DuoGAT [94] utilize short-term time series windows to infer these inter-series relationships. Subsequently, they refine the multivariate time series representations through GNNs and optimize under objectives such as forecasting or reconstruction.

However, current GNN-based methods typically rely on a single type of graph, either static or dynamic, to represent relationships among time series. Static graphs capture long-term relationships but lack the adaptability to respond to dynamic changes, while dynamic graphs are often overly sensitive to short-term noise, potentially misinterpreting regular fluctuations as anomalies. For example, in an industrial context, a static graph might mistakenly flag normal operational changes as anomalies due to its inability to adapt to routine fluctuations. Conversely, dynamic graphs, while responsive to short-term changes, may misinterpret long-term sensor variations as significant anomalies, thus triggering false alarms.

To address the aforementioned issues, we introduce a Prospective Multi-Graph Cohesion (PMGC) framework for multivariate TSAD. Specifically, PMGC integrates a list of learnable dynamic graphs with a static graph to capture both the long-standing relationships and the evolving short-term interactions among time series. To seamlessly integrate these two types of graphs, we introduce a graph cohesion loss. Our theoretical analysis shows that this loss function encourages the dynamic graphs to remain diverse while converging towards the static graph, thus ensuring a balance between the cohesiveness and diversity of graph representations. Leveraging these integrated graphs, we employ GNNs to refine the representations of the multivariate time series data, which are optimized under a forecasting objective. Anomalies are subsequently identified based on significant deviations in the forecasting errors.

However, one drawback of the traditional forecasting TSAD approaches is their reliance on historical data, which may not adequately capture unpredictable concurrent contextual changes. For example, in manufacturing, production lines might reconfigure workflows during high-demand periods, causing short-term changes in process relationships that are normal but unpredictable from historical data alone. To overcome this limitation, we propose a simple but effective strategy called “Prospective Graphing”. This technique incorporates current time series values into a graph construction process. By incorporating the most recent observations, the graphs are able to more accurately reflect the current inter-series relationships.

To summarize the results in this chapter, our contributions are listed as follows:

- We present a novel framework that integrates instance-specific dynamic graphs with a universal static graph, effectively capturing both the long-standing relationships and the evolving short-term interactions among time series.
- We introduce a graph cohesion loss function that ensures diversity among graphs while aligning with the overarching patterns captured by the static graph. Our theoretical analysis demonstrates the effectiveness of this loss function in balancing the cohesiveness and diversity in the graph representations.
- We propose a simple yet effective technique called “Prospective Graphing”, which enhances the traditional forecasting process by taking concurrent inter-series relationships into account.
- Empirical experiments carried out in this chapter show that PMGC achieves a superior performance on five real-world industrial datasets over a variety of existing TSAD methods.

## 4.2 Related Works

### 4.2.1 Deep Learning-based Time Series Anomaly Detection

The mainstream deep learning-based TSAD techniques consist of the forecasting-based and reconstruction-based methods [49]. Forecasting-based methods predict future values based on historical data, and anomalies can be identified when substantial deviations between predicted and actual values occur. Existing works for forecasting-based methods generally

leverage various neural architectures like RNNs, CNNs, Transformers, or hybrid combinations to learn feature representation of multivariate time series [72, 65, 116, 201, 26, 165]. For example, LSTM-NDT [72] utilizes Long Short-Term Memory networks (LSTMs) to predict future values and dynamically detects anomalies by analyzing prediction errors with a nonparametric threshold. Similarly, DeepANT [116] uses CNNs to forecast subsequent time points based on historical data and identifies anomalies from the deviation between predicted and observed values.

Reconstruction-based methods focus on the accurate reconstruction of time series, under the premise that anomalies will result in high reconstruction errors. Those methods include deep autoencoders (AEs) [157, 184, 99, 24, 78] and Generative Adversarial Networks (GANs) [96, 10, 25, 44] with varying aforementioned sequence encoder architectures to learn the characteristics within time series. For instance, MAD-GAN [96] integrates a GAN with an LSTM-based reconstruction model, using outputs from both the discriminator and the reconstruction model to determine anomaly scores. AnomalyTransformer [184] introduces a dual-branch attention mechanism to enforce long-term dependencies between time points and computes anomaly scores from a combination of association discrepancies and reconstruction loss.

Despite the powerful capabilities of those base neural architectures to capture temporal dependencies, these models are not designed to explicitly model the inter-series relationship. For instance, LSTMs and Transformers typically project input features into dense vectors, which may not explicitly capture the pairwise relationships between different series. Temporal Neural Networks (TCNs) [8], known for handling temporal sequences, primarily aggregate information through pooling operations, which might not sufficiently address the spatial relationships between multiple series.

### 4.2.2 Modeling Time Series via Graphs

Thanks to the rapid development of GNNs, deep learning-based methods are capable of modeling graph-structured data. This progress has been leveraged in time series anomaly detection, with an increasing number of studies integrating GNNs into their model design. MTAD-GAT [201] utilizes graph attention networks (GATs) with learnable intra- and inter-series graphs to capture spatial-temporal relationships. DuoGAT [94] extends this by incorporating a directed, weighted, time-oriented graph with a differencing-based GAT layer for temporal dependencies. GReLeN [199] uses a VAE structure to learn a probabilistic graph that captures the dependence relationship among different sensors in a system. STADN [165] leverages GATs to model the relationship between sensors based on a dynamic graph constructed using correlation coefficients.

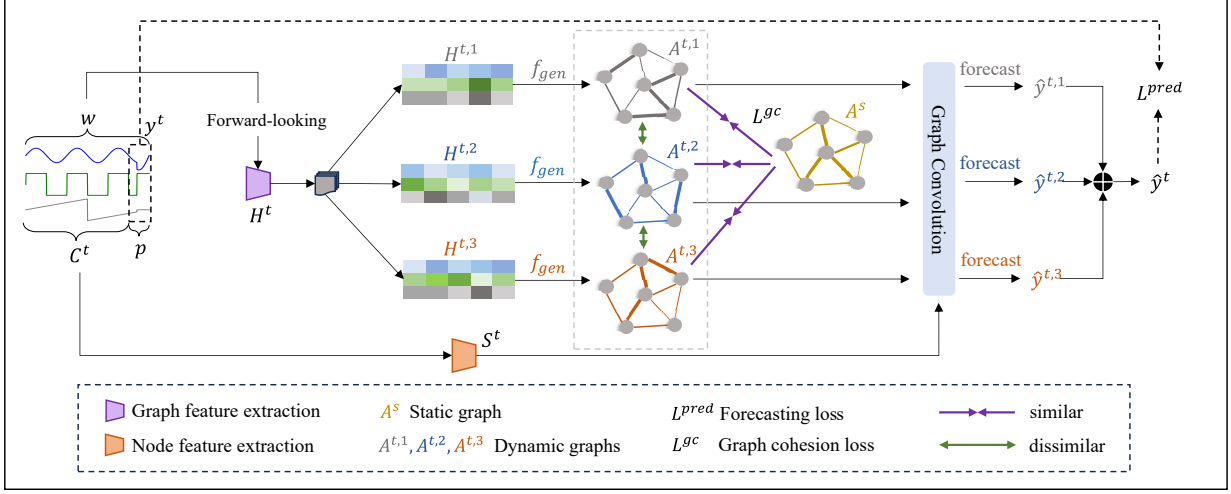


Figure 4.1: An illustrative diagram for the Framework of PMGC.

Apart from those approaches that use a short-term time window for graph learning, GDN [32] creates a global static graph using one-hot embeddings and imposes sparsity constraints on it by retaining only top-K entries per node. Similarly, FuSAGNet [58] constructs a top-K graph which additionally takes account of external information like categories of sensors, and refines its approach by using sparse auto-encoders to generate node features. GTA [26] further enhances computational efficiency by using Gumbel sampling to create a sparse, differentiable graph, bypassing the need for dot-product multiplication. CST-GL [206], starting with a static learnable global embedding graph, uses a coupled GCN and TCN module to model spatial and temporal dependencies. Other methods include MAD-SGCN [135], which adopts self-supervised learning for its graph structure, and MEGA [171], which learns a multi-scale graph based on Discrete Wavelet Transform. Our method differs from the majority of the existing approaches by decoupling inter-series relationships into short-term and long-term dependencies through a combination of dynamic and static graphs.

## 4.3 Methodology

### 4.3.1 Problem Formulation

Following commonly used notations, the  $i^{th}$  row and  $j^{th}$  column of an arbitrary 2D matrix  $A$  are denoted by  $A_i$  and  $A_j$  respectively. The  $(i, j)^{th}$  entry of  $A$  is denoted by  $A_{ij}$ . The

$l_2$ -norm of a vector is represented by  $\|\cdot\|_2$ , and the Frobenius norm for a matrix is referred as  $\|\cdot\|_F$ .

The training data is a multivariate time series  $X \in \mathbb{R}^{N \times T_{\text{train}}}$ , which consists of  $N$  univariate time series collected over  $T_{\text{train}}$  time ticks. We denote  $X_i$  as the  $i$ -th time series and  $X_t$  as the time series value at  $t$ . We adopt a window-based strategy to extract sub-sequences as training or test samples from the multivariate time series by sliding over the multivariate time series along the time dimension with a stride of 1. In other words, the training set  $\mathcal{D}$  will be a collection of sub-sequences, where a sample  $X^t = \{X_{\cdot i}\}_{i=(t-w+1)}^t \in \mathcal{D}$  is a collection of time ticks over  $w$  time steps.

Our goal is to detect anomalies in a test set, a multivariate time series  $X \in \mathbb{R}^{N \times T_{\text{test}}}$  but collected over a separate time span of time ticks. Specifically, we aim to assign an anomaly score to each time tick in the test set, with higher values indicating a greater likelihood of an anomaly. The output of our method is a list of anomaly scores over the  $T_{\text{test}}$  time ticks. The anomaly score can be thresholded to a binary value with 0 being normal and 1 being abnormal.

### 4.3.2 Overview

An overview of the proposed framework, PMGC, is presented in this section. Our framework unfolds in the following steps:

- Graph Structure Learning (Section 4.3.3): A list of adjacency matrices are derived from the hidden representations of the input time window (Section 4.3.3). Subsequently, a graph cohesion loss is imposed on the learned graphs to ensure diversity among them while adhering to certain global properties through a learnable static graph (Section 4.3.3). We provide a theoretical analysis to support the effectiveness of the loss function (Section 4.3.3).
- Prospective GNN-based Forecasting (Section 4.3.4): Future information is incorporated into the graph construction process of dynamic graphs. Subsequently, each dynamic graph is fed into a graph convolution with the encoded context window serving as node features. The outputs from the graph convolutions are averaged for the time series forecasting.
- Anomaly Score Calculation (Section 4.3.5): The anomaly score is calculated according to the forecasting error.

### 4.3.3 Graph Structure Learning

#### Multi-Graph Generation

In this section, we delve into the graph structure learning module, which aims to capture the intricate inter-series relationships in a task-specific manner. Given the evolving nature of real-world time series data, we start by constructing a dynamic graph based on each input time window. Additionally, inspired by the Multi-Head Attention (MHA) mechanism [167], we embrace a multi-graph approach instead of just generating one graph, under the premise that multiple graphs can be attuned to distinctive aspects of time series data like shapes, frequencies, and granularities. An additional advantage of employing multiple graphs stems from an enhancement in model robustness. Specifically, each graph is learned to capture specific patterns and relationships within the data. This diversification ensures that for an anomaly to go undetected, it would have to simultaneously fit the distinct pattern of all graphs.

Given a time window  $X^t$ , we initiate the process by extracting its feature representation  $H^t$  via a two-layer fully connected neural network. Similar to the MHA mechanism,  $H^t$  is then partitioned into  $k$  distinct hidden representations  $\{H^{t,i}\}_{i=1:k}$ , where  $H^{t,i} \in \mathbb{R}^{N \times d}$ . These representations are then processed through a graph generator function  $f_{gen}(\cdot) : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times N}$  to yield  $k$  dynamic graphs  $\{A^{t,i}\}_{i=1:k}$ . There are various ways to perform graph structure learning, such as those based on attention mechanisms [201] or gumbel-softmax techniques [26]. In this work, we use the pairwise cosine similarity to generate edge weights, owing to its computational efficiency. Specifically, the operation of  $f_{gen}(H)$  is defined as:

$$A_{ij}^* = \frac{H_i \cdot H_j^T}{\|H_i\|_2 \|H_j\|_2} \quad (4.1)$$

$$A = \text{ReLU}(A^*) \quad (4.2)$$

Here, each of the  $N$  series is treated as a node, with the edge weights being calculated based on the representations of pairwise time series.

#### Graph Cohesion Loss

One drawback of relying solely on those dynamic graphs is their susceptibility to noise and transient fluctuations. To mitigate this issue, we propose a solution by encouraging dynamic graphs to emulate the long-term dependencies captured by a learnable static graph

$A^s$ . Here,  $A^s$  is a static learnable global adjacency matrix, obtained by  $A^s = f_{gen}(\xi)$ , where  $\xi \in \mathbb{R}^{N \times d}$  contains  $N$  learnable vectors uniquely defined for each channel of the multivariate time series.

A simple loss formulation to combine the two types of graphs involves directly minimizing the distance between each dynamic graph and the static graph. We refer to this loss function as a simplified version of a graph cohesion loss, which is defined as:

$$\mathcal{L}^{gc, simple} = \mathbb{E}_{X^t \sim \mathcal{D}} \left[ \sum_{i=1}^k \text{dist}(A^s, A^{t,i}) \right] \quad (4.3)$$

Here,  $\text{dist}(\cdot, \cdot)$  measures the distance between two graphs, which could be quantified using metrics such as the Frobenius norm. However, directly minimizing the distance between dynamic and static graphs could result in them becoming identical, inadvertently merging into a single graph.

To prevent this, we introduce a new graph cohesion loss which is designed to maintain long-term dependencies produced by the static graph while preserving the unique patterns of each dynamic graph. The graph cohesion loss is defined as:

$$\mathcal{L}^{gc} = \mathbb{E}_{X^t \sim \mathcal{D}} \left[ \sum_{i=1}^k -\log \frac{h(A^s, A^{t,i})}{h(A^s, A^{t,i}) + \sum_{j \neq i} h(A^{t,i}, A^{t,j})} \right] \quad (4.4)$$

where  $h(\cdot, \cdot) : (\mathbb{R}^{N \times N}, \mathbb{R}^{N \times N}) \rightarrow \mathbb{R}$  is formulated as:

$$h(A^i, A^j) = \exp(-\text{dist}(A^i, A^j)/\tau) \quad (4.5)$$

$$\text{dist}(A^i, A^j) = \|A^i - A^j\|_F^2 \quad (4.6)$$

Here, Eq. 4.5 measures the similarity between two adjacency matrices with  $\tau$  as a temperature hyper-parameter. Intuitively,  $\mathcal{L}^{gc}$  propels each dynamic graph towards congruence with the global static adjacency matrix  $A^s$ , ensuring that a universal structure underpins each graph. Concurrently, it encourages diversity among the dynamic graphs, preventing them from converging into identical matrices and, thus ensuring that each retains its unique representational capacity.

## Theoretical Analysis

In this section, we present a theoretical analysis to show that the simplified graph cohesion loss,  $\mathcal{L}^{gc, simple}$ , fails to ensure diversity among dynamic graphs and may cause them to converge into a static graph. In contrast, the improved graph cohesion loss,  $\mathcal{L}^{gc}$ , does not have the aforementioned issues.

**Proposition 4.1.** *The optimal solution for minimizing  $\mathcal{L}^{gc, simple}$  is achieved only when  $A^{t,1} = A^{t,2} = \dots = A^{t,k} = A^s$  for all  $X^t \in \mathcal{D}$ .*

**Proof.** When the condition  $A^{t,1} = A^{t,2} = \dots = A^{t,k} = A^s$  is satisfied, it is trivial to show that  $\mathcal{L}^{gc, simple} = 0$ . Now assume that there exists at least one  $A^{t,j} \neq A^s$ . In this case, we have  $\|A^{t,j} - A^s\|_F^2 > 0$ , resulting in  $\mathcal{L}^{gc, simple} > 0$ . Thus, the optimal solution for  $\mathcal{L}^{gc, simple}$  is achieved only when all dynamic graphs collapse into the static graph  $A^s$ .  $\square$

Next, proposition 4.2 demonstrates that the situation in which all dynamic graphs collapse into a static graph is not an optimal solution for minimizing  $\mathcal{L}^{gc}$ .

**Proposition 4.2.** *The uniform graph case where  $A^{t,1} = A^{t,2} = \dots = A^{t,k} = A^s$  is not an optimal solution for minimizing  $\mathcal{L}^{gc}$ .*

**Proof.** Given an  $X^t \in \mathcal{D}$  and arbitrary  $A^s$ , define  $l_1(X^t)$  as a loss when all dynamic graphs are equal to  $A^s$ , i.e.,  $A^{t,1} = A^{t,2} = \dots = A^{t,k} = A^s$ . Without loss of generality, define  $l_2(X^t)$  as a loss when the first dynamic graph deviates from  $A^s$  by a nonzero amount, i.e.,  $\|A^{t,1} - A^s\|_F = d_t > 0$ , while the remaining graphs are still equal to  $A^s$ . Additionally, we assume that  $\tau = 1$  for simplicity. Then it follows:

$$l_1(X^t) = \sum_{j=1}^k \left[ -\log \left( \frac{1}{k} \right) \right] = k \log(k) \quad (4.7)$$

$$l_2(X^t) = \log(k) - (k-1) \log \left( \frac{1}{k-1 + e^{-d_t^2}} \right) \quad (4.8)$$

Subtracting  $l_2(X^t)$  from  $l_1(X^t)$  yields:

$$l_1(X^t) - l_2(X^t) = k \log(k) - \log(k) + (k-1) \log \left( \frac{1}{k-1 + e^{-d_t^2}} \right) \quad (4.9)$$

$$= (k-1) \log \left( \frac{k}{k-1 + e^{-d_t^2}} \right) = \Delta_{d_t} \quad (4.10)$$

Given that  $e^{-d_t^2} < 1$ ,  $k > 1$  and  $\log(\cdot)$  is monotonically increasing, it follows that  $l_1(X^t) - l_2(X^t) = \Delta_{d_t} > 0$ .

Let  $\mathcal{P}$  be a probability measure over the distribution  $\mathcal{D}$ . let  $\mathcal{L}_1$  be an overall loss when  $\|A^{t,j} - A^s\|_F = 0$ , for all  $j$  and  $X^t$ , i.e.,  $\mathcal{L}_1 = \int l_1(X^t) d\mathcal{P}$ . Now, assuming that the model is trained such that the condition  $\|A^{t,1} - A^s\|_F > d$  ( $d > 0$ ) holds for a subset of  $\mathcal{D}$ , by

which we denote the set as  $\mathcal{B}$ , i.e.,  $\mathcal{B} = \{X^t \in \mathcal{D} : \|A^{t,1} - A^s\|_F > d\}$ . Defining an overall loss for this scenario as  $\mathcal{L}_2$ , i.e.,  $\mathcal{L}_2 = \int_{\mathcal{B}} l_2(X^t) d\mathcal{P} + \int_{\mathcal{B}^c} l_1(X^t) d\mathcal{P}$ . It then follows:

$$\mathcal{L}_1 - \mathcal{L}_2 = \int l_1(X^t) d\mathcal{P} - \left( \int_{\mathcal{B}} l_2(X^t) d\mathcal{P} + \int_{\mathcal{B}^c} l_1(X^t) d\mathcal{P} \right) \quad (4.11)$$

$$= \int_{\mathcal{B}} l_1(X^t) d\mathcal{P} - \int_{\mathcal{B}} l_2(X^t) d\mathcal{P} \quad (4.12)$$

$$> \mathcal{P}(\mathcal{B}) \Delta_d \quad (4.13)$$

Since  $\Delta_d > 0$  and  $\mathcal{P}(\mathcal{B}) > 0$ , it follows that the uniform graph case is not the optimal solution due to  $\mathcal{L}_1 > \mathcal{L}_2$ . Furthermore, by increasing the number of samples that satisfy  $\|A^{t,1} - A^s\|_F \geq d$ ,  $\mathcal{P}(\mathcal{B})$  increases, thereby further minimizing the bound in Eq. 4.13. This ultimately avoids a uniform graph scenario for all  $X^t$ .  $\square$

Next, Proposition 4.3 shows that even if the dynamic graphs are identical (but not necessarily equal to  $A^s$ ), they still do not minimize  $\mathcal{L}^{gc}$ . This guarantees the diversity among dynamic graphs.

**Proposition 4.3.** *The homogeneous dynamic graph case where  $A^{t,1} = A^{t,2} = \dots = A^{t,k}$  does not minimize  $\mathcal{L}^{gc}$ .*

**Proof.** Building on top of Proposition 4.2, it remains to show that the homogeneous dynamic graph case does not generate a better solution in comparison to the uniform graph scenario in Proposition 4.2.

Given an  $X^t \in \mathcal{D}$  and an arbitrary  $A^s$ , assume that  $A^{t,1} = A^{t,2} = \dots = A^{t,k} = A^{t,*}$ , where  $A^s \neq A^{t,*}$ . Define  $\|A^s - A^{t,*}\|_F = c_t$ . A loss under this setting,  $l_3(X^t)$ , is calculated as:

$$l_3(X^t) = \sum_{j=1}^k -\log \left( \frac{e^{-c_t^2}}{e^{-c_t^2} + (k-1)} \right) \quad (4.14)$$

$$= \sum_{j=1}^k -\log \left( \frac{1}{1 + \frac{k-1}{e^{-c_t^2}}} \right) = \Delta_{c_t} \quad (4.15)$$

Since  $\frac{k-1}{e^{-c_t^2}} > k-1$  as  $e^{-c_t^2} < 1$ , we have:

$$l_3(X^t) > \sum_{j=1}^k -\log \left( \frac{1}{k} \right) = l_1(X^t) \quad (4.16)$$

Now, define  $\mathcal{L}_1$  as the overall loss where  $\|A^s - A^{t,*}\| = 0$  for all  $X^t \in \mathcal{D}$ , and  $\mathcal{L}_3$  as the loss where  $\|A^s - A^{t,*}\| > c$  with  $c > 0$  for some subset of  $\mathcal{D}$  (denote this set as  $\mathcal{C}$ ). Similarly to Eq. 4.11, it can be shown that:

$$\mathcal{L}_3 - \mathcal{L}_1 > \mathcal{P}(\mathcal{C})\Delta_c \quad (4.17)$$

Since  $\Delta_c > 0$  and  $\mathcal{P}(\mathcal{C}) > 0$ , it follows that  $\mathcal{L}_3 > \mathcal{L}_1$ . This shows that the homogeneous dynamic graph case does not yield a better solution compared to the uniform graph case. Since in Proposition 4.2, it was shown that the uniform graph case is not optimal, it follows that the configuration where  $A^{t,1} = A^{t,2} = \dots = A^{t,k}$  is also not an optimal solution for minimizing  $\mathcal{L}^{gc}$ .  $\square$

#### 4.3.4 Prospective GNN-based Forecasting

In this study, we follow a convention adopted in the forecasting-based TSAD approach, which identifies anomalies by measuring deviations between predicted and actual values. Traditional forecasting-based TSAD methods often rely on historical patterns, which may fail to account for concurrent contextual changes. For instance, in manufacturing, different batches of products might require unique processing settings, and during high-demand periods, production lines may be temporarily reconfigured. These changes introduce dynamic inter-process relationships that are normal but unpredictable using historical data alone.

To address this issue, we introduce a technique termed “prospective graphing”, which incorporates current time steps into the graph modeling instead of solely using them as labels. By using up-to-date information to construct dynamic graphs, the model adjusts in real time, accurately reflecting current inter-series relationships. Unlike the reconstruction-based TSAD methods, which suffer from issues such as identity mapping, our approach solely incorporates current time into the graph construction process. The predictions are then based on how these graphs aggregate historical observations, by considering current inter-series relationships. Additionally, these dynamic graphs are further regularized by  $\mathcal{L}^{gc}$ , to mitigate the identity mapping issue and enhance model robustness.

Next, GNNs are used as the feature encoder to aggregate the contextual information for each time series based on these prospective dynamic graphs. The extraction of node features is done by applying linear layers to raw time series data. As highlighted in [194], this simplified approach yields a competitive performance on time series forecasting tasks in comparison to other advanced neural architectures (e.g., Transformers and RNNs), offering an optimized balance between performance and computational efficiency.

Concretely, the input sample  $X^t$  is split into a context window  $C^t = \{X_i\}_{i=(t-w+1)}^{(t-p)}$  and a prediction window  $y^t = \{X_i\}_{i=(t-p+1)}^t$ . The context window which is encoded by the linear layers, together with the dynamic graphs  $\{A^{t,i}\}_{i=1}^k$ , is fed into GNNs in turn to generate a list of hidden features which are decoded into the  $p$ -step future sequence,  $\hat{y}^t$ . The operations unfold as follows:

$$S^t = f_e(C^t) \quad (4.18)$$

$$E^{t,i} = f_{gnn}(S^t, A^{t,i}) \quad (4.19)$$

$$\hat{y}^{t,i} = f_d(E^{t,i}) \quad (4.20)$$

$$\hat{y}^t = \frac{1}{k} \sum_{i=1}^k \hat{y}^{t,i} \quad (4.21)$$

where  $f_e(\cdot) : \mathbb{R}^{(w-p)} \rightarrow \mathbb{R}^d$  is a linear layer that encodes the context vector into its hidden representations, and  $f_d(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^p$  is a linear layer that decodes the output of GNNs into the final predictions. We adopt MIXHOP [181] as the graph convolution layer for  $f_{gnn}(Z, A)$ . The operation of each layer is defined as:

$$Z = \beta Z + (1 - \beta) \tilde{A} Z W \quad (4.22)$$

where  $\beta$  controls the ratio that retains the original states,  $\tilde{A}$  is the symmetric normalized Laplacian matrix of  $A$ , and  $W$  is a learnable weight matrix. The predicted value  $\hat{y}^t$  is obtained from an averaging of the predictions  $\hat{y}^{t,i}$  derived from different dynamic graphs.

The prediction loss is defined as:

$$\mathcal{L}^{pred} = \mathbb{E}_{X^t \sim \mathcal{D}} \|y^t - \hat{y}^t\|_2^2 \quad (4.23)$$

where  $y^t$  is the true value within the prediction window for  $X^t$ . Thus, the overall training loss is a combination of the prediction loss and the graph cohesion loss:

$$\mathcal{L} = \mathcal{L}^{pred} + \lambda \mathcal{L}^{gc} \quad (4.24)$$

where  $\lambda$  is a hyper-parameter that controls the influence of the graph cohesion loss on the learned graphs.

### 4.3.5 Anomaly Score Calculation

The anomaly score is calculated based on the forecasted error. However, since different time series have different behaviors which can result in very different scales of forecasted

errors, we adopt the same normalization method as proposed by GDN [32], which leverages the median and interquartile range (IQR) to standardize the forecasting error of each time series. Specifically, the anomaly score for the  $i$ -th time series at time  $t$ , denoted as  $a_i^t$ , is calculated as:

$$a_i^t = \frac{Err_i^t - \tilde{\mu}_i}{\tilde{\sigma}_i} \quad (4.25)$$

Here,  $Err_i^t$  represents the forecasted error for time series  $i$  at time  $t$ , while  $\tilde{\mu}_i$  and  $\tilde{\sigma}_i$  denotes the median and IQR of the error set for that specific time series. The final anomaly score at  $t$  is then obtained by taking the maximum among all sensors. Anomalies are then identified based on a predefined threshold: if the anomaly score at time  $t$  exceeds the threshold, the instance is flagged as anomalous.

## 4.4 Experiments

### 4.4.1 Experimental Setting

#### Datasets

We adopt five widely-used real-world benchmark datasets for the purpose of performance evaluation: SWaT [111], WADI [2], SMD [157], MSL [72], and SMAP [72]. Each dataset provides an unlabeled training set capturing normal operational data, along with a separate test set that includes labeled anomalies, allowing for a robust evaluation across varied industrial and operational settings. The SWaT dataset was collected from a secure water treatment testbed, a controlled environment that emulates large-scale urban water systems. It contains 51 sensor dimensions recorded over 11 days, with cyber attacks introduced during the last 4 days. Similarly, the WADI dataset is derived from a water distribution testbed and includes 123 channels of sensor data recorded over 16 days, with attacks intermittently introduced in the final 2 days. The SMD dataset, a multi-entity dataset collected from server machines at a large internet company, provides data from 28 different server entities. The performance of SMD is calculated by averaging over these entities. NASA’s MSL and SMAP datasets contain telemetry data from the Mars Science Laboratory rover and the Soil Moisture Active Passive satellite, respectively. These datasets capture anomalies that occur during spacecraft operations, marked by unexpected events or deviations from standard operating behaviors. This collection of datasets enables evaluation across a variety of anomaly types and domains, with summary statistics being presented in Table 4.1.

Table 4.1: Summary statistics for datasets

| Dataset | avg. # channels | # training | # test | % anomalies |
|---------|-----------------|------------|--------|-------------|
| SWaT    | 51              | 47440      | 44993  | 12.13       |
| WADI    | 123             | 78459      | 17281  | 5.77        |
| MSL     | 55              | 58317      | 73729  | 10.53       |
| SMAP    | 25              | 135183     | 427617 | 12.79       |
| SMD     | 38              | 25300      | 25300  | 4.21        |

## Baselines

We select the following recent anomaly detection approaches as our baseline methods: LSTM-VAE [128], MAD-GAN [96], USAD [7], MSCRED [196], MTAD-GAT [201], GDN [32], FuSAGNet [58], TranAD [166], and CST-GL [206]. Among them, LSTM-VAE, MAD-GAN, USAD, and TranAD are methods that adopt LSTMs, Transformers, or MLPs as the backbone time series encoder. GDN, FuSAGNet, MTAD-GAT, CST-GL, and MSCRED are all graph-based approaches, which adopt either GNNs or convolutional LSTMs to encode the graphs constructed based on pairwise time series relationships. Similar to ours, GDN, FuSAGNet, MTAD-GAT, and CST-GL leverage graph structural learning to infer the inter-series dependencies during training, and employ either a time series forecasting or reconstruction as the objective. The graph structures in GDN, FuSAGNet, and CST-GL are inferred by using uniquely defined global one-hot embeddings for each time series, whereas MTAD-GAT learns a dynamic inter-series graph using a short-term time window.

## Evaluation metrics

We use the point-wise F1 (denoted as  $F1^p$ ), point-adjusted F1 (denoted as  $F1^*$ ) [157] and composite F1 (denoted as  $F1^c$ ) [49] score as the performance metrics.

The point-wise F1 is calculated using standard precision and recall at the point level:

$$F1^p = 2 \times \frac{Pr \times Re}{Pr + Re} \quad (4.26)$$

where precision (Pr) and recall (Re) are based on true positive, false positive, and false negative counts for individual time points.

For the point-adjusted F1, we follow the point-adjusting strategy suggested in [157], which treats an entire anomaly segment as correctly detected if at least one point in the segment is identified. This is computed as:

$$F1^* = 2 \times \frac{Pr^* \times Re^*}{Pr^* + Re^*} \quad (4.27)$$

where precision ( $Pr^*$ ) and recall ( $Re^*$ ) are modified to count anomaly segments rather than individual points.

Composite F1 is the harmonic mean of time-wise precision and event-wise recall. As proposed in [49], it is calculated as:

$$\begin{aligned} Pr_t &= \frac{TP_t}{TP_t + FP_t} \\ Re_e &= \frac{TP_e}{TP_e + FN_e} \\ F1^c &= 2 \times \frac{Pr_t \times Re_e}{Pr_t + Re_e} \end{aligned}$$

where  $TP_t$  and  $FP_t$  count individual time-point true positives and false positives, while  $TP_e$  and  $FN_e$  count event-level true positives and false negatives.

Given that  $F1^*$  may overestimate recall [83], and  $F1^p$  does not account for event lengths [83],  $F1^c$  offers a balanced measure that integrates precision and recall across different granularities of anomaly evaluation. For all F1 metrics, we identify the best F1 score by optimizing the threshold for each method based on its anomaly score. This practice ensures unbiased and consistent comparisons across methods.

## Implementation

PMGC is trained with an Adam optimizer [84] at a learning rate of  $10^{-3}$  for 10 epochs. 20% of the training set is selected as a validation set where the best epoch is stored based on the validation loss. The hidden dimension is set to  $d = 64$ . The number of MixHop graph convolution layers is set to 2 with  $\beta = 0.05$ . The prediction window size is set to 5. The weight of the graph cohesion loss  $\lambda$  is  $10^{-5}$ . The number of dynamic graphs  $k$  is set to 5. The sequence length of the input sample  $w$  is set to 100 and 20 for SMAP and WADI, and 40 for MSL, SWaT, and SMD. All of the results are averaged over 5 independent runs.

Table 4.2: Performance comparison based on point-wise F1 ( $F1^p$ ), point-adjusted F1 ( $F1^*$ ), and composite F1 ( $F1^c$ ), with the best performance highlight in bold and second best underlined.

|             | SWaT        |             |             | WADI        |             |             | MSL         |             |             | SMAP        |             |             | SMD         |             |             |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Methods     | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      | $F1^p$      | $F1^c$      | $F1^*$      |
| LSTM-VAE    | 77.2        | 52.4        | 81.9        | 36.6        | 51.7        | 64.6        | 20.1        | 37.9        | 87.5        | 22.7        | 22.7        | 69.5        | 40.3        | 53.7        | 81.8        |
| USAD        | 76.8        | 45.4        | 83.0        | 28.2        | 42.0        | 35.6        | 23.6        | 42.0        | 60.0        | 28.1        | 31.7        | 25.4        | 42.3        | 56.0        | 82.1        |
| MAD-GAN     | 73.6        | 44.4        | 80.5        | 15.9        | 36.6        | 63.4        | 20.2        | 28.4        | 77.1        | 24.3        | 28.6        | 67.4        | 11.8        | 13.9        | 41.8        |
| TranAD      | 76.9        | 46.4        | 83.2        | 27.3        | 43.1        | 39.0        | 24.6        | 42.3        | 79.2        | 23.5        | 26.3        | 67.4        | 46.1        | 68.3        | 86.8        |
| MSCRED      | 75.9        | 37.1        | 82.9        | 29.6        | 36.4        | 50.6        | 27.9        | 41.7        | 81.1        | 23.0        | 23.6        | 63.2        | <b>48.1</b> | 53.9        | 77.9        |
| MTAD-GAT    | 76.4        | 50.9        | 83.2        | 30.3        | 44.9        | 46.3        | <u>29.7</u> | <b>67.7</b> | <b>94.2</b> | <u>29.0</u> | 35.0        | <u>91.3</u> | 45.5        | 66.7        | 86.1        |
| GDN         | 77.5        | 50.7        | 84.3        | <u>49.5</u> | <u>64.9</u> | <u>72.1</u> | 23.9        | 41.7        | 84.0        | 23.6        | 31.5        | 82.2        | 53.5        | <b>71.6</b> | <u>89.2</u> |
| FuSAGNet    | <b>81.9</b> | 55.7        | <b>88.5</b> | 48.0        | 59.2        | 67.0        | 27.3        | 64.4        | 89.0        | 22.7        | <u>43.6</u> | 91.1        | 43.5        | 52.0        | 79.9        |
| CST-GL      | 76.8        | <u>59.1</u> | 86.3        | 34.1        | 58.2        | 65.7        | 22.8        | 24.4        | 87.4        | 24.7        | 31.3        | 71.2        | 42.5        | <u>69.0</u> | <b>90.4</b> |
| PMGC (ours) | <u>81.2</u> | <b>71.5</b> | <u>87.4</u> | <b>53.4</b> | <b>66.6</b> | <b>72.2</b> | <b>33.6</b> | <u>58.7</u> | <u>90.0</u> | <b>30.0</b> | <b>69.4</b> | <b>94.5</b> | <u>46.2</u> | 66.5        | 72.1        |

#### 4.4.2 Comparison Study

Table 4.2 presents the comparative results of all of the models. Overall, PMGC achieves the best performance on average across all of the datasets (We rank the first and second for the majority of datasets). For non-graph-based methods such as TranAD, LSTM-VAE, USAD, and MAD-GAN, the performance lags are likely attributed to their relatively weaker ability to model inter-series relationships. These methods primarily focus on capturing the temporal pattern within individual time series while the interactions among them are only learned implicitly, limiting their effectiveness in capturing the complex dependencies within multivariate time series. For graph-based methods, MSCRED operates directly on dynamic sequential graphs, which is at a coarser scale compared to the raw time series, thus limiting its ability to capture fine-grained temporal dependencies. Similar to ours, GDN, FuSAGNet, MTAD-GAT, and CST-GL are graph-based methods utilizing learnable graphs to model relationships between variables and leverage forecasting or reconstruction errors to identify anomalies. However, these methods rely primarily on a single static graph or dynamic graph, which may restrict their ability to learn diverse and dynamic evolving relations within multivariate time series. In contrast, PMGC leverages a combination of concurrent dynamic graphs and a static graph, enabling the model to capture both real-time instance-wise short-term relationships and long-term correlations. This comprehensive approach allows anomalies to be detected more effectively if they violate any of these patterns.

### 4.4.3 Ablation Study

We conduct an ablation study based on the following variants:

- **w/o dynamic graph:** Predictions are generated based on the graph convolution outputs of the static graph.
- **w/o static graph ( $\mathcal{L}^{gc}$ ):** The graph cohesion loss is removed, so only dynamic graphs are utilized in the model.
- **w/o prospective graphing:** The construction of dynamic graphs are solely based on historical observations.
- **static  $\oplus$  dynamic:**  $\mathcal{L}^{gc}$  is removed. The outputs of the dynamic graphs and the static graph are averaged to generate the final prediction.
- **static || dynamic:** The output embeddings from each dynamic graph’s convolution layer are concatenated with those from the static graph.
- **with  $\mathcal{L}^{gc, simple}$ :** We directly minimize the distance between the learned dynamic graphs and the static graph by replacing  $\mathcal{L}^{gc}$  with  $\mathcal{L}^{gc, simple}$  (Eq. 4.3).

Figure 4.2 presents the performance for possible model variants of MGPC across four datasets. Firstly, removing dynamic graphs leads to a noticeable performance drop among the majority of the datasets. Dynamic graphs adapt to changing patterns and interactions within the time series data, which provides a more responsive and accurate modeling. Furthermore, excluding the graph cohesion loss results in a large performance decline. The reasons are twofold. Firstly, it forces the model to rely solely on short-term information, making it more susceptible to noise. Additionally, it fails to encourage diversity among the dynamic graphs. Using diverse dynamic graphs ensures that anomalies are detected under various graph structures, making it harder for anomalies to have low prediction errors consistently. Additionally, performance drops significantly without prospective graphing, highlighting the importance of using real-time information in constructing dynamic graphs. This approach ensures that the graphs accurately reflect the current interactions among time series during a normal state.

We also compare three variants of ways to combine static and dynamic graphs. Specifically, merely minimizing the distance (by using  $\mathcal{L}^{gc, simple}$ ) between the two graph types can result in overly similar graphs, compromising their ability to preserve the diverse relationships within the time series. Likewise, simply averaging the prediction outputs from each

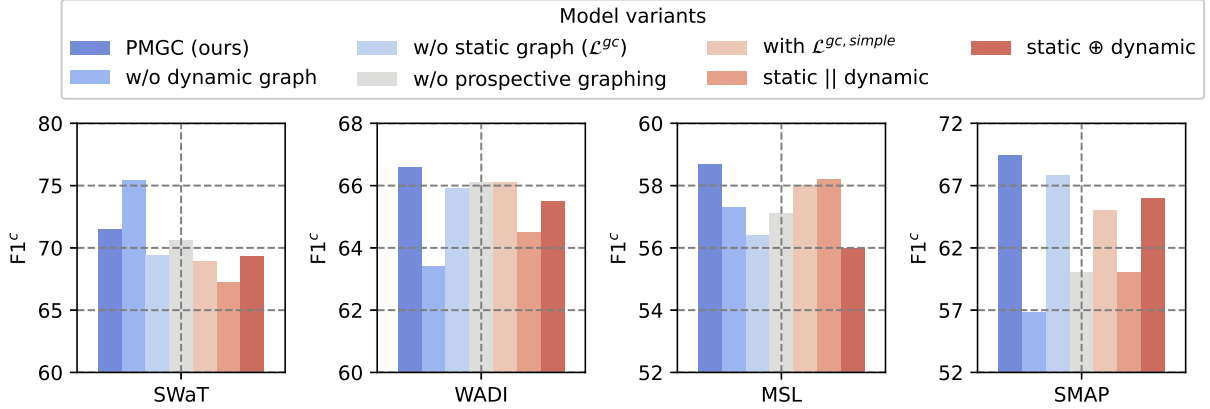


Figure 4.2: Performance comparison on different model variants in terms of  $F1^c$ .

type of graph or concatenating their graph convolution outputs shows a marked performance decline. These methods do not promote diversity among the graphs, thus limiting their ability to capture multifaceted inter-series patterns. Meanwhile, they fail to address the potential drawbacks of the noise in dynamic graphs and the lack of real-time adaptation in static graphs. In comparison, our loss mitigates these issues by ensuring dynamic graphs maintain long-term consistency, thus enhancing the overall quality of the graph representations.

#### 4.4.4 Sensitivity Analysis

Figure 4.3 presents the impact of four parameters on the performance of PMGC. For the loss weights of the graph cohesion loss, performance peaked at  $10^{-5}$  and tends to decline slightly with larger weights. This outcome is expected, as excessively high loss weights may overly prioritize graph diversification, detracting from the primary forecasting task, while too small weights can reduce the effectiveness of dynamic graph regularization. The number of graphs,  $k$ , shows optimal performance around 5, suggesting that a moderate number of graphs can maintain a good anomaly detection performance. Furthermore, this relatively small number of graphs ensures a high computational efficiency. For the context window size  $w$ , the performance varies across datasets. For instance, SMAP demonstrates improved performance with larger window sizes, while WADI exhibits a decline. These discrepancies may be attributed to the trade-off between information richness and sample correlation. Larger windows provide richer contextual information but may lead to overfitting due to increased sample-wise correlation, while smaller windows have the opposite

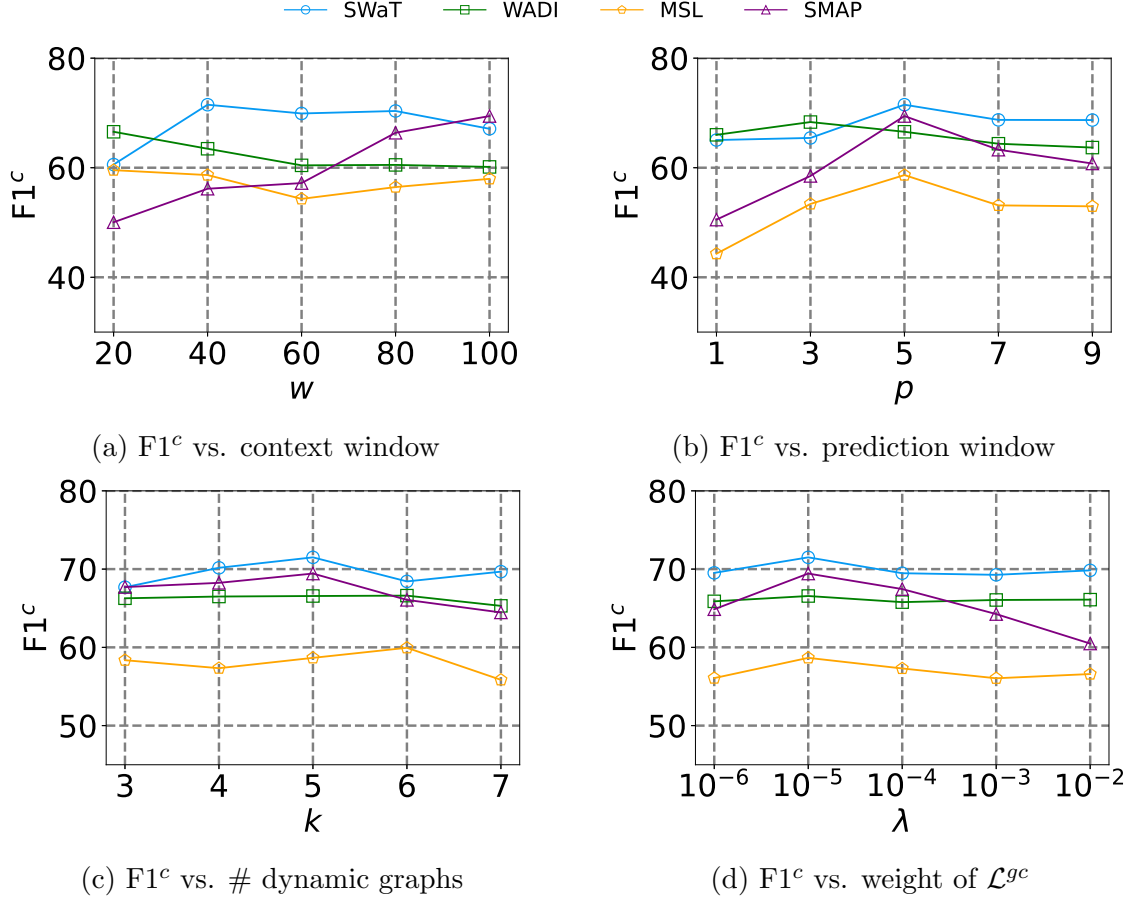


Figure 4.3: Performance variation in terms of  $F1^c$  under different hyper-parameters.

effect Conversely. Overall, the choice of  $w$  is dataset-specific and guided by validation forecasting performance. Lastly, for the prediction window  $p$ , performance peaks around  $p = 5$ , partly because incorporating more future steps better utilizes dynamic graphs, which in turn accurately reflects the current inter-series relationship among time series.

#### 4.4.5 Visualization of Learned Graphs

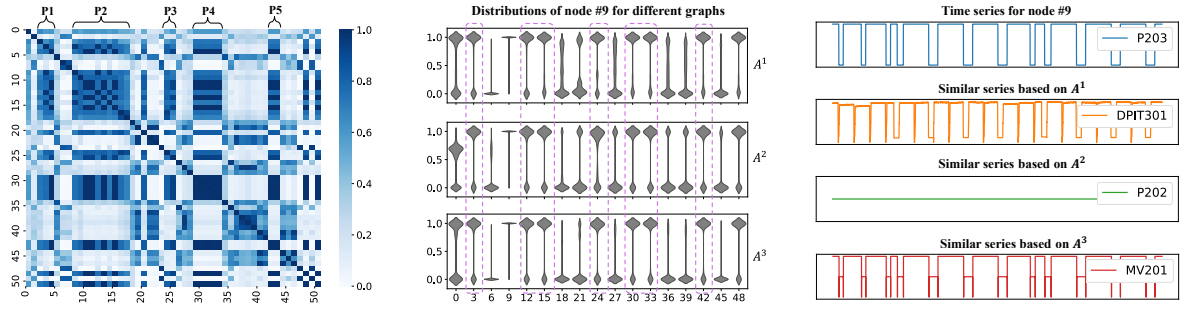
In this section, we examine the visual patterns of the learned static and dynamic graphs. First, as shown in Figure 4.4a, the heatmap of the learned static graph visualizes the weight matrix that represents long-term inter-series correlations among nodes. In this heatmap,

each entry corresponds to the learned correlation (or affinity) between a pair of nodes over the entire observation period. High-intensity colors indicate a strong relationship, while lower intensities suggest weaker dependencies. For instance, nodes 8-18 are highly correlated, attributable to their collective function in P2 (Chemical Dosing), where chemical concentrations and dosing pumps operate in tandem. Furthermore, cross-process correlations are also discernible. Nodes 2-4 are central in P1 (Raw water inflow), which governs the initiation of the entire treatment sequence by controlling raw water inflow. Nodes 8-18 in P2 are important in crafting the water’s chemical profile, a prerequisite for subsequent filtration efficacy. Whereas nodes 24-26, anchored in P3 (Ultra-filtration), manage the pressure dynamics, ensuring an optimal filtration performance. Each node’s function is intricately connected. Capturing these inter-dependencies enhances the capacity of our model to pinpoint anomalies that could disrupt or violate the inherent process dynamics.

We next explore the dynamic graphs. As demonstrated in the violin plot of node #9 in Figure 4.4b, the three dynamic graphs exhibit different distributions. These variations underscore the dynamic nature of the graph, adapting in response to real-time data. However, a closer inspection of the figure reveals a consistency: the highlighted nodes on those graphs maintain strong correlations with node #9 across all dynamic graphs, echoing the patterns observed in the static graph. To see what distinct perspectives are encapsulated within each dynamic graph, we extract the top one similar to the original series with respect to node 9 (excluding those that are identical across all of the three graphs) based on the weights of different dynamic graphs on Figure 4.4c. The fourth sub-figure mirrors the shape of node #9, while the second figure, though morphologically distinct, highlights a frequency alignment. The third, a flat series, suggests a focus on long-term correlations, capturing the stationary behavior of node #9 over an extended duration. These diverse representations show the ability of those dynamic graphs to capture a different view of inter-series relationships among time series.

## 4.5 Conclusion

In this chapter, we have presented a multi-graph-based TSAD framework that integrates dynamic and static graphs to model both short-term and long-term inter-series relationships. A graph cohesion loss function is proposed, supported by the theoretical analysis, which balances diversity and cohesiveness in graph representations. This ensures that dynamic graphs adhere to the fundamental inter-series relationships while adapting to the multifaceted short-term variations. Additionally, the “prospective graphing” technique notably enhances predictive accuracy by leveraging concurrent information during the graph



(a) Heatmap of the Learned static graph. (b) Violin map for different dynamic graphs. (c) Time series selected based on the top indexes.

Figure 4.4: The results of Figures (a), (b), and (c) are based on the SWaT dataset.

construction process. Empirical evaluations on five real-world industrial datasets have demonstrated that PMGC significantly outperforms various existing baseline methods, showcasing its effectiveness in accurately detecting anomalies in multivariate time series data.

# Chapter 5

## Towards Cross-domain Few-shot Graph Anomaly Detection

### 5.1 Introduction

Graph anomaly detection (GAD) is the process of identifying unusual nodes, edges, or subgraphs within graph-structured data that significantly deviate from the majority. As a critical area in data mining, GAD finds extensive applications across diverse domains such as social networks, financial networks, and e-commerce [21, 98, 156, 46, 45]. These network structures often harbor various anomalies, which might manifest as spammers in social networks, money laundering in financial markets, or fraudulent transactions in e-commerce. The complexity of these datasets and the severe implications of overlooking anomalies have intensified the focus on GAD research in recent years. Among the various applications of GAD, detecting anomalous nodes is particularly crucial as it directly relates to pinpointing individual nodes exhibiting potentially harmful or irregular behaviors within networks. For example, in social networks, node-level GAD might involve detecting profiles that spread misinformation or engage in harmful activities. In financial networks, it could focus on identifying accounts involved in unusual transaction patterns that may indicate money laundering or financial fraud. In this paper, we mainly focus on addressing the intricate and practical challenges of node-level GAD.

Existing GAD techniques can be primarily categorized into unsupervised GAD, semi-supervised GAD, few-shot GAD and cross-domain GAD. Unsupervised GAD focuses on mining the intrinsic properties of graph structures and attributes solely based on unlabeled data. It operates under the assumption that anomalies will substantially deviate

from learned patterns. Studies in this area predominantly employ reconstruction-based methods [34, 41, 108, 47] or contrastive learning-based approaches [103, 204, 197, 40, 123] to identify these deviations. For instance, DOMINANT [34] uses autoencoders to reconstruct graph structures and attributes, spotting anomalies through significant reconstruction errors in comparison to the majority of nodes in a network. PREM [123] evaluates the cohesiveness of node embeddings with their neighborhoods using contrastive learning scores, based on the principle that anomalous nodes are substantially different from their surroundings. While unsupervised GAD is valuable in label-free scenarios, its effectiveness can be compromised by noise, leading to potential false positives or undetected anomalies.

Semi-supervised GAD aims to bridge the gap between unsupervised and fully supervised GAD by leveraging both labeled and unlabeled data to enhance anomaly detection performance. These methods typically utilize labeled examples to direct the identification of anomalies, while employing unlabeled data to improve model generalization [170, 87, 113, 164]. For instance, SemiGNN [170] trains a binary classifier on labeled samples and enhances the learning of normal representations through a hierarchical attention mechanism. SemiADC [113] uses generative adversarial networks to model the distribution of normal node features from unlabeled data, subsequently employing a binary classifier to differentiate these normal samples from labeled anomalies. SAD [164] introduces a memory-enhanced deviation network to accentuate the differences between abnormal and normal nodes in dynamic graphs, supplemented by a contrastive learning module to leverage unlabeled data. Despite its potential advantages, semi-supervised GAD often hinges on the availability of an adequate number of labeled samples, a condition that is rarely met in real-world scenarios, thus hampering its effectiveness.

In recent years, few-shot GAD has garnered increasing attention for its ability to effectively operate in label-scarce environments. This line of research aims to detect anomalies using only a limited number of labeled anomalies [36, 55, 205, 183]. For instance, ANEMONE-FS [79] employs a multi-scale contrastive learning framework that utilizes few-shot labeled anomalies to enhance model discrimination between normal and anomalous behaviors. Meta-GDN [36] introduces a graph deviation network, which is trained under a cross-network meta-learning framework. The framework involves meta-training across multiple auxiliary networks to establish a generalized model, subsequently fine-tuned on the target network with a few labeled examples. LHML [55] introduces a dynamic learnable hypersphere that adjusts its decision boundary to include normal samples. It also applies cross-subgraph meta-learning for rapid adaptation from auxiliary to target networks. Despite the promising results of these methodologies, most of them rely on sufficiently labeled auxiliary networks that are presumed to share the same distributions with the target network. Nevertheless, acquiring a sufficient volume of labeled data in the same domain can

be prohibitively costly and challenging in practical scenarios.

While acquiring sufficient labels within a single domain can be challenging in real-world scenarios, obtaining adequate labeled data from pre-existing, related domains is often feasible. For example, hotel and dining services from Yelp may share similar fraudulent review patterns due to overlapping user bases and behaviors. A few recent studies focus on tackling the cross-domain GAD problem, which leverages labeled data from a source domain to improve anomaly detection in an unlabeled target domain. For instance, COMMANDER [35] adopts a graph-attentive encoder and domain discriminator to learn domain-invariant features. It further includes an anomaly classifier and an attribute decoder to detect anomalies that are both common and unique across domains. ACT [173] employs contrastive learning to develop normal node representations in the target graph, and aligns these with source graph representations trained under sufficient source domain supervision. The framework is further enhanced by a self-training mechanism to refine detection accuracy. However, the efficacy of cross-domain GAD can be compromised without direct supervision from the target domain, especially when there are substantial cross-domain gaps.

In practice, it is often feasible to acquire a small number of labeled anomalies at a relatively low cost. Therefore, a more practical approach for the above-mentioned challenges is to leverage these scarce but valuable labeled anomalies from the target domain to guide the domain adaptation process. This aids in the better extraction and integration of both domain-invariant and domain-specific knowledge. In this study, we mainly focus on the cross-domain few-shot GAD problem. Specifically, we seek to improve GAD performance under label-scarce conditions with the help of adequate labeled data from a different but related domain. This task is complex owing to large domain shifts, which manifest as variances in graph structures, attributes, and anomaly distributions between the source and target domains [211, 35]. These differences necessitate a robust model capable of assimilating common knowledge across diverse domains while preserving the unique characteristics inherent to each domain. Furthermore, the scarcity of labels in the target domain exacerbates the training complexity, introducing additional noise and uncertainty. Therefore, it is crucial to explore effective methodologies that can successfully transfer knowledge from the source domain with minimal supervision.

In this chapter, a simple and effective framework is proposed to address the challenges of domain shift and label scarcity encountered in cross-domain few-shot GAD settings, termed CDFS-GAD. Specifically, CDFS-GAD introduces a domain-adaptive graph contrastive learning module, aiming to enhance intra-domain node representations while aligning cross-domain distributions via inter-domain graph contrastion. To further retain the unique characteristics of anomaly distribution in the target domain, a domain-specific prompt-tuning module is proposed that adapts the backbone encoder to the nuances of

each domain without compromising domain-invariant feature extraction. To more effectively leverage the scarce labels of target domains, a domain-adaptive hypersphere classification loss is introduced. It encloses normal instances within a hypersphere, with the center adjusted contextually based on domain-invariant and domain-specific information. Post-training, a simple but effective self-training phase is introduced to further enhance the confidence of the model’s prediction in few-shot settings.

To summarize, our contributions are listed as follows:

- **Novel Problem:** To the best of our knowledge, we are the first to investigate and address the graph anomaly detection problem under the cross-domain few-shot setting.
- **Enhanced Cross-domain Feature Extraction:** We propose a domain-adaptive graph contrastive learning module alongside a domain-specific prompt-tuning module. These components are designed to extract both common and unique features across two domains, optimizing for domain generalization.
- **Domain-aware Few-shot Graph Anomaly Detection:** A domain-adaptive hypersphere classification loss is introduced, which shapes the detection boundary using domain insights under minimal supervision. A self-training process is implemented to further reinforce robust learning under few-shot conditions.
- **Extensive Experimental Validation:** We conduct comprehensive experiments on twelve cross-domain pairs under varying few-shot conditions. The experiments demonstrate the superior performance of our method compared to existing GAD methods.

## 5.2 Methodology

### 5.2.1 Problem Formulation

In this chapter, we study the cross-domain few-shot GAD problem on attributed graphs. An attributed graph is represented as  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ , where  $\mathcal{V}$  represents the set of nodes  $\{v_1, v_2, \dots, v_n\}$ ,  $\mathcal{E}$  represents the set of edges  $\{e_1, e_2, \dots, e_m\}$ , and  $X = [x_1; \dots; x_n] \in \mathbb{R}^{n \times d}$  denotes the node attributes. Additionally,  $A \in \mathbb{R}^{n \times n}$  denotes the adjacency matrix of  $\mathcal{G}$ , which encodes the edge connectivity structure of the graph. In the cross-domain few-shot GAD setting, we are given an attributed graph  $\mathcal{G}^s = (\mathcal{V}^s, \mathcal{E}^s, X^s)$  from the source domain,

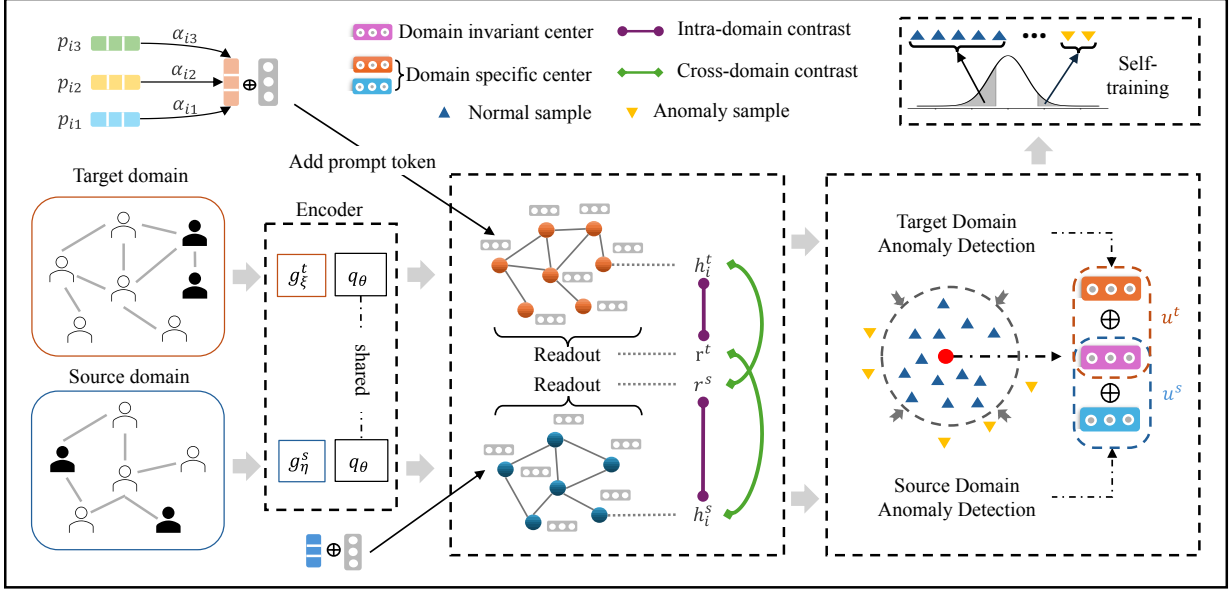


Figure 5.1: A diagram illustrating the proposed CDFS-GAD framework.

and an attributed graph  $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, X^t)$  from the target domain. For the source domain, we have access to the true label of each of the nodes in  $\mathcal{G}^s$ . Conversely, in the target domain, we only have access to a limited subset of labels ( $|\mathcal{V}_l^t| \ll |\mathcal{V}_u^t|$ ), where  $\mathcal{V}_l^t$  and  $\mathcal{V}_u^t$  respectively denote the labeled anomalies and the unlabeled nodes in the target domain.

The primary objective is to utilize the labeled data across both domains to identify anomalous nodes in  $\mathcal{G}^t$ . We aim to construct an anomaly detection model that calculates an anomaly score for each node in  $\mathcal{G}^t$ . The model will prioritize higher scores for anomalous nodes and lower scores for normal nodes, in alignment with common practices that frame GAD as a ranking problem [34].

### 5.2.2 Overview

The overall pipeline of CDFS-GAD is illustrated in Figure 5.1. CDFS-GAD initiates by jointly training a model using labeled data from both source and target domains (Section 5.2.3). To bridge the domain gap while enhancing the feature representation of normality patterns within each domain, we propose a domain-adaptive contrastive learning module as elaborated in Section 5.2.3. This module includes both inter-graph and intra-graph contrasting components to align cross-domain features and refine within-domain

normality representations. To further enhance the capability of the proposed CDFS-GAD to distinguish between domains, we incorporate unique, learnable prompt tokens into the backbone model. This integration improves the flexibility of the model to extract domain-specific features (Section 5.2.3). For model optimization, we introduce a domain-adaptive hypersphere classification loss, which effectively segregates normal instances from anomalies into a domain-sensitive hypersphere, as described in Section 5.2.4. Lastly, we introduce a self-training strategy that refines target domain predictions using pseudo-labels generated by the trained model, as discussed in Section 5.2.4.

### 5.2.3 Feature Learning and Adaptation

#### Encoder

To address the challenge of limited label availability in the target domain, we commence by training a joint model based on labeled data from both the source domain and the target domain. We utilize GraphSAGE [57] as the backbone GNN encoder, which is shared across both domains. Owing to the inherent differences in node feature dimensionality and distribution between the two domains, we employ distinct MLP networks to preprocess the node features for each domain separately. These initial representations, tailored to their respective domains, are then inputted into the GNN encoder. In the following section, we denote the GNN encoder function as  $q_\theta(\cdot)$ , and the MLP mapping function of source and target domain to be  $g_\eta^s(\cdot)$  and  $g_\xi^t(\cdot)$ , respectively.

#### Domain-Adaptive Contrastive Learning

Reliance solely on labeled data, particularly the scarce labels available in the target domain, may compromise the model’s ability to generalize effectively. This may pose challenges in learning domain-invariant features from the source domain that are essential for the target domain. To address this, we introduce a domain-adaptive contrastive learning module that leverages the abundant unlabeled datasets available across both domains. This module is devised with two objectives: firstly, to improve the representation quality of normal nodes that constitute the majority of the dataset, and secondly, to align node representations between domains, thereby bridging the domain discrepancies.

Drawing inspiration from Deep Graph Infomax (DGI) [169], our proposed module consists of two distinct components: intra-domain and inter-domain contrastive learning. The intra-domain contrastive learning enhances within-domain representation by contrasting

individual node representations against their corresponding graph-level representations. For node  $i$  in a target domain, we define the positive pair as the node representation  $h_i^t = q_\theta(g_\xi^t(x_i^t), \tilde{A}^t)$  and the graph representation  $r^t = \mathcal{R}(\{h_i^t | v_i \in \mathcal{V}^t\})$ . Here,  $\tilde{A}^t = A^t \odot M$  is the adjacency matrix after an edge-dropping operation, with  $M$  being a binary mask created by sampling from a Bernoulli distribution, i.e.,  $m_{ij} \sim \text{Bernoulli}(p)$ . The readout function  $\mathcal{R}(\cdot)$  calculates the graph embedding by aggregating the node embeddings. Negative pairs are constructed using corrupted node representations  $\tilde{h}_i^t = q_\theta(g_\xi^t(\tilde{x}_i^t), \tilde{A}^t)$ , where  $\tilde{x}_i^t$  is generated by shuffling node features across the graph. The intra-domain contrastive loss for the target domain is then defined as:

$$\mathcal{L}_{intra}^t = \mathbb{E}_{v_i \sim \mathcal{V}^t} \left[ -\log D(h_i^t, r^t) - \log(1 - D(\tilde{h}_i^t, r^t)) \right] \quad (5.1)$$

where  $D(\cdot, \cdot)$ , defined as  $D(x, y) = \sigma(x^t W y)$ , computes the similarity between two representations.

On the other hand, the inter-domain contrastive learning module aims at aligning the feature distribution across domains. Recognizing the inherent challenge of establishing direct node mappings between source and target domains, we opt to contrast the overall graph representations of each domain. Specifically, positive pairs are formed between the source and target graph representations, while negative pairs involve contrasting the corrupted source domain graph representation with the target domain representation. The resulting inter-domain contrastive loss is given by:

$$\mathcal{L}_{inter}^t = -\log D(r^t, r^s) - \log(1 - D(r^t, \tilde{r}^s)) \quad (5.2)$$

where  $\tilde{r}^s$  represents the corrupted source domain graph representation, calculated by  $\tilde{r}^s = \mathcal{R}(\{\tilde{h}_i^s | v_i \in \mathcal{V}^s\})$ .

Similarly, the inter and intra-domain contrastive learning is applied to the source domain, to ensure that the learning process is balanced across both domains. The final domain-adaptive contrastive loss is defined as:

$$\mathcal{L}_{contra} = \mathcal{L}_{intra}^s + \mathcal{L}_{inter}^s + \mathcal{L}_{intra}^t + \mathcal{L}_{inter}^t \quad (5.3)$$

## Domain-Specific Prompt Learning

While the contrastive learning module and the shared encoder can facilitate the learning of domain-invariant features, it may inadvertently overlook the unique information intrinsic

to each domain. To address this limitation, we augment the backbone encoder with a domain-specific prompt learning module. This enhancement is specifically designed to capture the distinct characteristics of each domain, while ensuring that it does not impede the ability of the encoder to learn domain-invariant features.

Prompt learning is a key technique in NLP that adapts pre-trained large language models to diverse tasks by transforming inputs into specific prompts [102]. The idea has been adapted to graph-structured data, empowering pre-trained models to address a multitude of graph-related tasks via context-relevant prompts [42, 158, 160, 105]. Motivated by recent studies, we incorporate prompt learning into our backbone model. This ensures that it complements, rather than compromises the extraction of domain-invariant features by the encoder.

We primarily follow the methodology proposed in [42], where each node is assigned a unique learnable prompt token. The prompt token is constructed as a weighted combination of a set of independent basis vectors  $\{p_j\}_{j=1:m}$  with the weights calculated as:

$$\alpha_{ij}^{(l)} = \frac{\exp(h_i^{(l)} \cdot p_j)}{\sum_{k=1}^m \exp(h_i^{(l)} \cdot p_k)} \quad (5.4)$$

where  $h_i^{(l)}$  represents the  $l^{th}$ -layer outputs of the GNN encoder for node  $i$ . Note that each domain has a set of different basis vectors, i.e.,  $\{p_j^s\}_{j=1:m}$  and  $\{p_j^t\}_{j=1:m}$  for the source and target domains, respectively. Therefore, the prompt tokens can focus on learning the unique characteristics of each domain.

Finally, the prompt-enhanced feature for each node of the  $l^{th}$  layer is formulated by:

$$z_i^{(l)} = h_i^{(l)} + \sum_{j=1}^m \alpha_{ij}^{(l)} p_j \quad (5.5)$$

which allows  $z_i^{(l)}$  to maintain its original information while being augmented with domain-specific insights encoded in the domain-specific prompt tokens. Then,  $z_i^{(l)}$  is used as the input to the  $(l+1)^{th}$  GNN layer.

## 5.2.4 Model Training and Testing

### Domain Adaptive Graph Anomaly Detection

We adopt the Hypersphere Classification (HSC) Loss [142], which is tailored for anomaly detection in scenarios with scarce anomaly labels. The core idea of this loss function is to

cluster normal samples around a central point while ensuring that anomalous samples are kept at a distance. However, applying a single center for training across source and target domains may lack flexibility due to their distinct anomaly distributions. To circumvent this, we append the center with a domain-specific prompt for each domain. In this way, the loss function can dynamically adjust the center to accommodate the unique characteristics of each domain while leveraging commonalities.

Specifically, we define the center for the target domain as  $c^t = u^t + c$ , where  $u^t$  denotes the target domain-specific prompt, and  $c$  embodies the domain-invariant center common to both domains. Both  $c$  and  $u^t$  are learnable vectors. Then, we define the domain-adaptive hypersphere classification loss for the target domain as:

$$\mathcal{L}_{dahsc}^t = \mathbb{E}_{v_i \sim \mathcal{G}^t} [-y_i^t \log l(z_i^t, c^t) - (1 - y_i^t) \log (1 - l(z_i^t, c^t))] \quad (5.6)$$

where  $z_i^t$  and  $y_i^t$  are the hidden representations and label for the node  $i$  respectively, and  $l(x, y) = \exp(-\|x - y\|^2)$ . Similarly, the loss  $\mathcal{L}_{dahsc}^s$  is defined for the source domain, with the learnable center as  $c^s = u^s + c$ .

By integrating with the contrastive loss in the previous section, the overall training loss is defined as:

$$\mathcal{L} = \mathcal{L}_{dahsc}^t + \mathcal{L}_{dahsc}^s + \alpha \mathcal{L}_{contra} \quad (5.7)$$

where  $\alpha$  is a hyper-parameter controlling the contribution of the contrastive loss. During inference, the anomalous score of node  $i$  for the target domain is obtained by:

$$s_i^t = 1 - l(z_i^t, c^t) \quad (5.8)$$

## Self-Training

To mitigate the uncertainty inherent in the limited availability of labeled data within the target domain, we implement a simple but effective self-training strategy. This strategy creates a more reliable dataset for subsequent model refinement phases by designating pseudo-anomalous and pseudo-normal samples. We introduce percentile-based criteria for pseudo-label selection. Specifically, the pseudo-anomalous set  $\mathcal{D}^+$  and the pseudo-normal set  $\mathcal{D}^-$  are obtained as:

$$\mathcal{D}^+ = \{v_i | s_i \text{ is in the top } \beta_1 \text{ percentile of scores}\} \quad (5.9)$$

$$\mathcal{D}^- = \{v_i | s_i \text{ is in the bottom } \beta_2 \text{ percentile of scores}\} \quad (5.10)$$

---

**Algorithm 1:** Training Process of CDFS-GAD

---

**Input:**  $\mathcal{G}^s$  and  $\mathcal{G}^t$ ,  $N_a$  and  $N_b$  as the epochs for joint training and self-training

**Output:** Anomaly scores for all nodes in  $\mathcal{G}^t$

```
1 for  $i \leq N_a$  do
2   Obtain the augmented adjacency matrix  $\tilde{A}^t$  and  $\tilde{A}^s$ 
3   Obtain the node representations  $H^t, H^s \leftarrow q_\theta(g_\xi^t(X^t, \tilde{A}^t)), q_\theta(g_\eta^s(X^s, \tilde{A}^s))$ 
4   Obtain the graph representations  $r^t, r^s \leftarrow \mathcal{R}(H^t), \mathcal{R}(H^s)$ 
5   Compute the domain-adaptive contrastive loss  $\mathcal{L}_{\text{contra}}$  based on  $H^t, H^s, r^t, r^s$ 
      by Eq. 5.3
6   Obtain the node representations with prompt tokens  $Z^t, Z^s$  by Eq. 5.5
7   Compute the domain-adaptive hypersphere classification losses  $\mathcal{L}_{\text{dahsc}}^s$  and
       $\mathcal{L}_{\text{dahsc}}^t$  based on  $Z^s, Z^t$  by Eq. 5.6
8   Take the gradient steps and update the parameters based on the loss
       $\mathcal{L}_{\text{dahsc}}^s + \mathcal{L}_{\text{dahsc}}^t + \alpha \mathcal{L}_{\text{contra}}$ 
9 end
10 Obtain pseudo-labels of target graph by Eq. 5.9 and Eq. 5.10
11 for  $i \leq N_b$  do
12   obtain the node representations  $H^t \leftarrow q_\theta(g_\xi^t(A^t, X^t))$ 
13   Compute the loss  $\mathcal{L}_{\text{dahsc}}^t$  for target domain
14   Take gradients and update parameters
15 end
16 Compute anomaly score for the nodes in  $\mathcal{G}^t$ 
```

---

where  $\beta_1$  and  $\beta_2$  are hyperparameters representing the percentage of nodes to be included in the pseudo-anomalous and pseudo-normal sample sets, respectively. These pseudo-labels are then used to retrain the model using the hypersphere classification loss. The final anomaly score for each node is computed according to Eq. 5.8. The overall training procedure of CDFS-GAD is summarized in Algorithm 1.

## 5.3 Experiments

### 5.3.1 Experimental Setting

#### Datasets

In this study, the datasets utilized in the experiments contain YelpHotel, YelpRes, YelpNYC, and Amazon. The summary statistics of these four datasets are presented in Table 5.1. Specifically, YelpHotel comprises user reviews centered on the accommodation sector in Chicago, with nodes representing users and edges representing mutual hotel reviews [137]. Yelp employs a sophisticated filtering algorithm to categorize users as either regular or anomalous, according to their review activities. YelpRes follows a similar structure, which focuses on reviews related to dining experiences within the same urban area [137]. Expanding the geographical coverage, YelpNYC encompasses restaurant reviews from New York City, preserving the intrinsic data architecture but diversifying in terms of geographic and urban dynamics [137]. Diverging from the service industry focus, the Amazon dataset contains user reviews from the e-commerce sector [80]. Users are identified as fraudulent if they have reviewed multiple products involved in crowd-sourcing campaigns.

Following similar settings in [35, 173], we have constructed twelve cross-domain pairs by pairing each dataset with every other. These pairs display varying degrees of domain similarities and dissimilarities. For instance, while YelpNYC and YelpRes both contain user reviews related to restaurants, they differ in their geographic location. YelpHotel and YelpRes, while both reside in Chicago, cater to distinct service sectors—accommodation and dining. The shift from YelpHotel to YelpNYC is relatively larger as they span not just the service type but also the urban environment. Nevertheless, their anomaly detection methodologies are similar as unified by Yelp’s filtering algorithm. The progression from YelpNYC to Amazon presents a relatively larger domain transition, moving from restaurant reviews to e-commerce. This transition engages diverse user interaction dynamics and review behaviors.

#### Implementation Details

The proposed CDFS-GAD framework is implemented via PyTorch, and Adam optimizer [84] with a learning rate of 0.0005 is used for model training. The architecture of the backbone encoder is composed of a two-layer GraphSAGE with dimensionalities of 256 and 64, respectively. Within the contrastive learning module, the edge-dropping probability  $p$  is set

Table 5.1: Summary statistics for datasets used in experiments.

| Datasets  | # nodes | # edges   | # attributes | # anomalies |
|-----------|---------|-----------|--------------|-------------|
| YelpHotel | 4322    | 101,800   | 8000         | 250         |
| YelpRes   | 5102    | 355,144   | 8000         | 250         |
| YelpNYC   | 21,040  | 1,658,137 | 10000        | 1000        |
| Amazon    | 17,496  | 495,213   | 10000        | 705         |

as 0.1, and the balance weight  $\alpha$  is set to 0.5. The readout function  $\mathcal{R}(\cdot)$  is chosen to be the simple average over the input node representations. For the self-training phase, the parameters  $\beta_1$  and  $\beta_2$  are chosen to be 0.02 and 0.25, respectively. In the prompt learning module, we set the number of bases  $m$  to 5 for both source and target graphs. The training epochs for the joint training stage and the self-training stage are set to 50 and 100 epochs, respectively.

For the empirical evaluation, we partition the nodes of the target graph into three segments: 40% for training, 20% for validation, and the remaining 40% for testing purposes. Additionally, the reported results are averaged over five independent trials with each initialized with different random seeds.

## Baselines and Evaluation Metrics

We compare eight recent state-of-the-art methods spanning several categories: unsupervised GAD methods (COLA [103], SL-GAD [204], ANEMONE [79], and PREM [123]), semi-supervised GAD (SemiGNN), few-shot GAD methods (meta-GDN [36] and ANEMONE-FS [205]) and cross-domain GAD (ACT [173]). The unsupervised GAD methods leverage solely the unlabeled data from the target domain, while cross-domain GAD utilizes both the labeled data from the source domain and the unlabeled data from the target domain. SemiGNN operates exclusively on the labeled data of the target domain. Similarly, ANEMONE-FS focuses on the labeled data in the target domain. For meta-GDN, we follow a similar setting in its original implementation by splitting the source domain graph into disjoint graphs, and then we treat those sub-graphs as the auxiliary networks and the target domain graph as the target network.

We adopt two widely used metrics for performance evaluation: the area under the receiver operating characteristic curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR). These metrics quantify the anomaly detection performance over a diverse range of decision thresholds.

Table 5.2: one-shot performance comparison with the existing GAD methods in terms of AUC-ROC and AUC-PR. The best results are highlighted in bold.

| Datasets           | YelpHotel→YelpNYC |              | YelpNYC→YelpHotel |              | YelpHotel→YelpRes |              | YelpRes→YelpHotel |              |
|--------------------|-------------------|--------------|-------------------|--------------|-------------------|--------------|-------------------|--------------|
| Evaluation Metrics | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       |
| SemiGNN            | 0.624             | 0.133        | 0.737             | 0.200        | 0.684             | 0.131        | 0.737             | 0.200        |
| ANEMONE            | 0.437             | 0.039        | 0.457             | 0.051        | 0.463             | 0.043        | 0.457             | 0.051        |
| meta-GDN           | 0.840             | 0.360        | 0.688             | 0.219        | 0.949             | 0.564        | 0.687             | 0.235        |
| ANEMONE-FS         | 0.474             | 0.078        | 0.752             | 0.239        | 0.446             | 0.042        | 0.752             | 0.239        |
| COLA               | 0.670             | 0.083        | 0.495             | 0.064        | 0.626             | 0.064        | 0.495             | 0.064        |
| ACT                | 0.833             | 0.269        | 0.790             | 0.224        | 0.901             | 0.301        | 0.785             | 0.218        |
| PREM               | 0.557             | 0.054        | 0.662             | 0.111        | 0.498             | 0.048        | 0.662             | 0.111        |
| SL-GAD             | 0.699             | 0.152        | 0.601             | 0.173        | 0.753             | 0.204        | 0.601             | 0.173        |
| CDFS-GAD (ours)    | <b>0.864</b>      | <b>0.361</b> | <b>0.796</b>      | <b>0.264</b> | <b>0.992</b>      | <b>0.903</b> | <b>0.798</b>      | <b>0.268</b> |
|                    | YelpHotel→Amazon  |              | Amazon→YelpHotel  |              | YelpNYC→YelpRes   |              | YelpRes→YelpNYC   |              |
|                    | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       |
| SemiGNN            | 0.374             | 0.040        | 0.737             | 0.200        | 0.684             | 0.131        | 0.624             | 0.133        |
| ANEMONE            | 0.585             | 0.044        | 0.457             | 0.051        | 0.463             | 0.043        | 0.437             | 0.039        |
| meta-GDN           | 0.772             | 0.221        | 0.677             | 0.244        | 0.950             | 0.603        | 0.853             | 0.386        |
| ANEMONE-FS         | 0.486             | 0.036        | 0.752             | 0.239        | 0.446             | 0.042        | 0.474             | 0.078        |
| COLA               | 0.593             | 0.045        | 0.495             | 0.064        | 0.626             | 0.064        | 0.670             | 0.083        |
| ACT                | 0.638             | 0.052        | 0.788             | 0.219        | 0.932             | 0.361        | 0.832             | 0.270        |
| PREM               | 0.511             | 0.040        | 0.662             | 0.111        | 0.498             | 0.048        | 0.557             | 0.054        |
| SL-GAD             | 0.832             | <b>0.319</b> | 0.601             | 0.173        | 0.753             | 0.204        | 0.699             | 0.152        |
| CDFS-GAD (ours)    | <b>0.835</b>      | 0.271        | <b>0.789</b>      | <b>0.246</b> | <b>0.991</b>      | <b>0.885</b> | <b>0.872</b>      | <b>0.387</b> |
|                    | YelpNYC→Amazon    |              | Amazon→YelpNYC    |              | YelpRes→Amazon    |              | Amazon→YelpRes    |              |
|                    | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       | AUC-ROC           | AUC-PR       |
| SemiGNN            | 0.374             | 0.040        | 0.624             | 0.133        | 0.374             | 0.040        | 0.684             | 0.131        |
| ANEMONE            | 0.585             | 0.044        | 0.437             | 0.039        | 0.585             | 0.044        | 0.463             | 0.043        |
| meta-GDN           | <b>0.863</b>      | <b>0.351</b> | 0.865             | <b>0.397</b> | 0.753             | 0.224        | 0.953             | 0.604        |
| ANEMONE-FS         | 0.486             | 0.036        | 0.474             | 0.078        | 0.486             | 0.036        | 0.446             | 0.042        |
| COLA               | 0.593             | 0.045        | 0.670             | 0.083        | 0.593             | 0.045        | 0.626             | 0.064        |
| ACT                | 0.641             | 0.052        | 0.834             | 0.278        | 0.629             | 0.050        | 0.912             | 0.315        |
| PREM               | 0.511             | 0.040        | 0.557             | 0.054        | 0.511             | 0.040        | 0.498             | 0.048        |
| SL-GAD             | 0.832             | 0.319        | 0.699             | 0.152        | 0.832             | <b>0.319</b> | 0.753             | 0.204        |
| CDFS-GAD (ours)    | 0.784             | 0.200        | <b>0.865</b>      | 0.366        | <b>0.855</b>      | 0.308        | <b>0.991</b>      | <b>0.903</b> |

### 5.3.2 Comparison Study

Table 5.2 presents the one-shot performance of all models across 12 data pairs in terms of AUC-ROC and AUC-PR. Predominantly, our model outperforms other state-of-the-art methods across most data pairs. In particular, the improvements are evident in pairs involving YelpNYC, YelpRes, and YelpHotel, which share relatively larger domain similarities. For example, YelpRes and YelpHotel contain data collected under the same urban context, while YelpNYC and YelpRes belong to the same service industry. Conversely, our model experiences a marginal performance decline with pairs that include the Amazon dataset, likely due to the discrepancy in the anomaly generation mechanism and service industry between the Amazon and Yelp datasets.

In comparison to meta-GDN, a competing method that also utilizes the label from two domains, our model surpasses it by a noticeable margin. We attribute this enhanced performance to our inclusion of domain-adaptive components that are adept at capturing domain-invariant and domain-specific information. In contrast, the performance of meta-GDN suffers due to its assumption that auxiliary and target networks belong to the same domain—an assumption that does not hold in our cross-domain setup and leads to its underperformance.

Regarding other methods, the unsupervised GAD methods generally underperform compared to the cross-domain unsupervised GAD method. ACT benefits from incorporating labeled information from the source domain, which enhances its learning capabilities for the target domain. On the other hand, the few-shot GAD method meta-GDN on average performs better than the unsupervised GAD methods and ACT, which is likely due to its access to additional labeled data from the target domain. However, methods like SemiGNN and ANEMONE-FS, which also rely on few-shot labels in the target domain, do not consistently perform well. Their underperformance relative to meta-GDN and even some unsupervised methods can be primarily attributed to the constraints of their loss functions (e.g., cross-entropy), which are less robust in extremely label-scarce environments than the few-shot enhanced deviation network employed by meta-GDN.

In summary, the superiority of our model to other methods can be attributed to the integration of the domain-adaptive modules that accommodate the similarity and difference between two domains, augmented by specialized anomaly detection loss and self-training for enhanced learning under few-shot scenarios. These elements collectively contribute to the final performance by effectively leveraging cross-domain information and sparsely labeled data.

### 5.3.3 Evaluation under Different Shots

Figure 5.2 presents the  $K$ -shot performance of semi-supervised and few-shot GAD methods in terms of AUC-ROC. We can observe that there is a large discrepancy in performance between SemiGNN and ANEMONE-FS, which rely solely on the target domain, and those like CDFS-GAD and meta-GDN which utilize the label information from both domains. The performance of SemiGNN and ANEMONE-FS lags significantly behind, particularly in scenarios with extremely limited labeled data (e.g., one-shot). In contrast, CDFS-GAD and meta-GDN show relative stability across different shots. This stability can be attributed to two factors: firstly, the inclusion of cross-domain labeled data provides a richer set of examples for model training. Secondly, the anomaly detection frameworks in CDFS-GAD and meta-GDN are tailored for few-shot scenarios, i.e., hypersphere classification

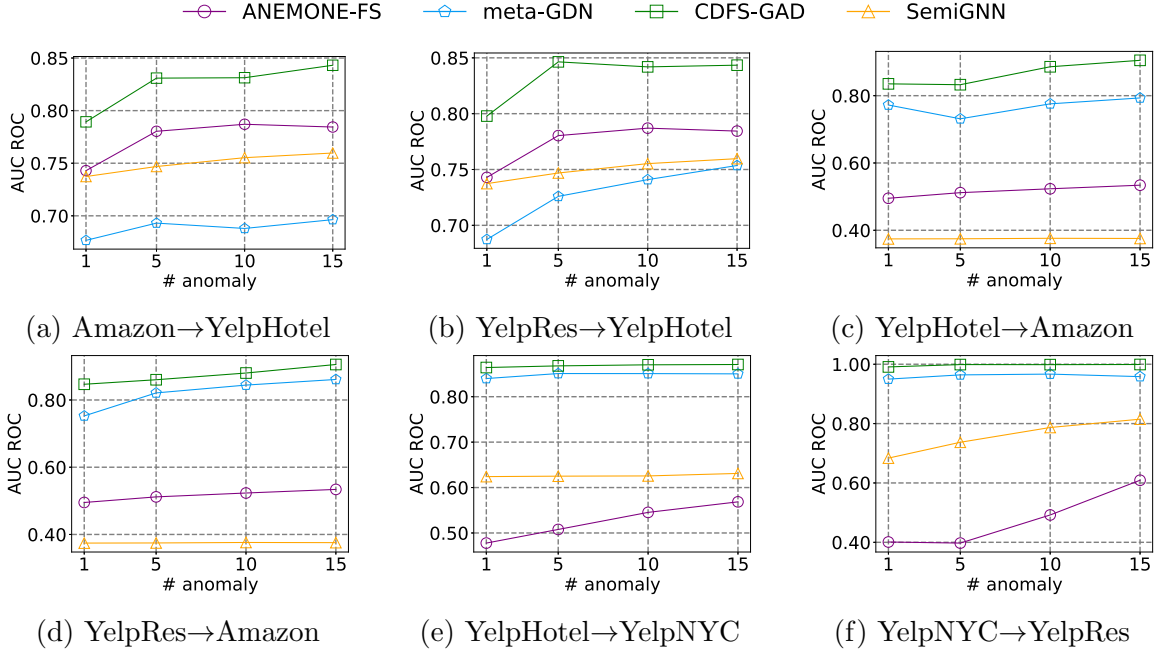


Figure 5.2: Various  $K$ -shot performance comparison with the state-of-the-art GAD methods in terms of AUC-ROC.

or deviation network, thereby improving their performance when labeled data is scarce. Despite the strengths of meta-GDN, our model consistently outperforms it across all  $K$ -shot settings. Specifically, CDFS-GAD is able to extract the relevant features from the source domain while retaining the unique characteristics of the target graph anomaly distribution, thereby improving the model generalization in the target domain.

### 5.3.4 Ablation Study

We conduct an ablation study for one-shot settings according to the following variants:

- $w/o$  domain-specific prompt: Removing the domain-specific prompt module, which is added to the hidden representations generated from the GNN encoder.
- $w/o \mathcal{L}_{intra}$ : Removing the intra-domain contrastive loss.
- $w/o \mathcal{L}_{intra} + \mathcal{L}_{inter}$ : Removing both the intra-domain and inter-domain contrastive loss.

- $\mathcal{L}_{hsc}$  only: Replacing the  $\mathcal{L}_{dahsc}$  with  $\mathcal{L}_{hsc}$ , meaning that we adopted separate anomaly detectors that rely on different learnable centers for the source and target domain, respectively.
- *w/o* source domain supervision: We train the model based on purely the labeled data from the target domain.
- *w/o* self-training: We use the prediction results generated purely from joint learning based on the labeled data from both the source and target domain.

We summarize our findings according to the results presented in Figure 5.3: First, the omission of the domain-specific prompt learning module results in a notable performance deduction. This suggests that while capturing domain-invariant information is critical, the domain-specific attributes are also important, as part of the normal/abnormal node behaviors can still be tied to their respective domains uniquely. Secondly, the drop in performance upon removal of the intra-contrastive learning component suggests that aligning source and target feature representations is vital for transferring knowledge between two domains. The absence of this module would impair the ability of our model to extract domain-invariant attributes between the two domains. In contrast, inter-contrastive learning helps to improve the representation quality of normal nodes within each domain, removing it along with intra-contrastive learning would impede the ability of our model to effectively represent features that are common across and unique to each domain. Moreover, replacing  $\mathcal{L}_{dahsc}$  with distinct  $\mathcal{L}_{hsc}$  for each domain also leads to a decline in performance. The inclusion of a domain-invariant center in addition to the domain-specific center improves the flexibility of our model to identify anomalies based on both within-domain and cross-domain information. Additionally, there is a notable drop in performance when removing source domain supervision. This agrees with our assumptions that labeled data from the source domain contributes to learning a more accurate model for the target domain. Lastly, the performance downturn in the absence of self-training indicates its critical role in addressing the high uncertainty associated with few-shot scenarios. This process helps mitigate overfitting by retraining the model on its most confident predictions.

### 5.3.5 Parameters Sensitivity

Figure 5.4 presents the impact of hyper-parameters  $\alpha$ ,  $m$ ,  $\beta_1$  and  $\beta_2$  under one-shot task, as measured by AUC-ROC. The balance weight  $\alpha$  controls the influence of the contrastive learning module, which includes the intra and inter-contrastive components. Figure 5.4a

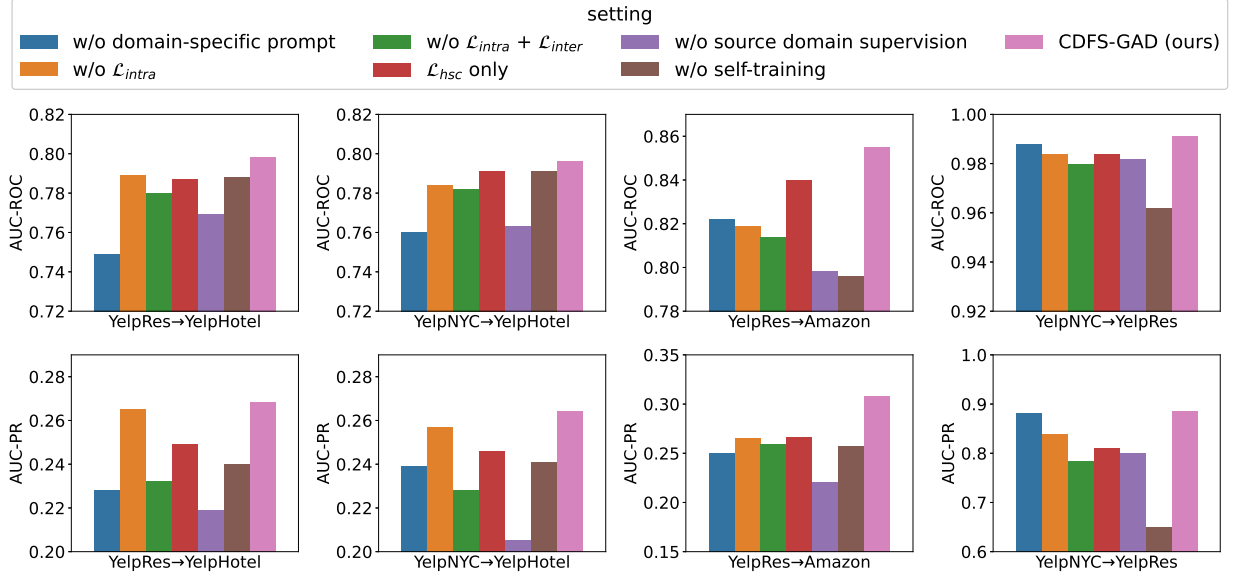


Figure 5.3: Ablation study under the one-shot task in terms of AUC-ROC and AUC-PR.

shows that performance drops moderately at the two boundaries of  $\alpha$ . This is reasonable as a lower  $\alpha$  may weaken the ability of our model to learn robust and domain-invariant representations, whereas a higher  $\alpha$  risks overshadowing the primary loss function. For the prompt tuning module, the basis number  $m$  exhibits a larger impact on the data pair YelpRes and Amazon compared to others. In particular, a larger basis is more beneficial for adapting to the substantial domain gap between the Yelp and Amazon datasets, as it grants the model increased flexibility, enabling more precise extraction of domain-specific knowledge. In terms of the self-training, we observe an optimal performance band when  $\beta_1$  and  $\beta_2$  are set around 0.03 and 0.25, respectively. Too few anomalies may not provide a strong enough learning signal, while too many may introduce noises that adversely affect the performance. Additionally, a higher  $\beta_2$  reflects the real-world distribution of normal behavior, which typically outnumbers anomalies, thereby guiding the model to a more accurate representation of normality. Overall, despite small variations across different values of the hyper-parameters, the performance of CDFS-GAD still exhibits relative stability in terms of AUC-ROC. This suggests that our model is relatively less sensitive to various parameter settings.

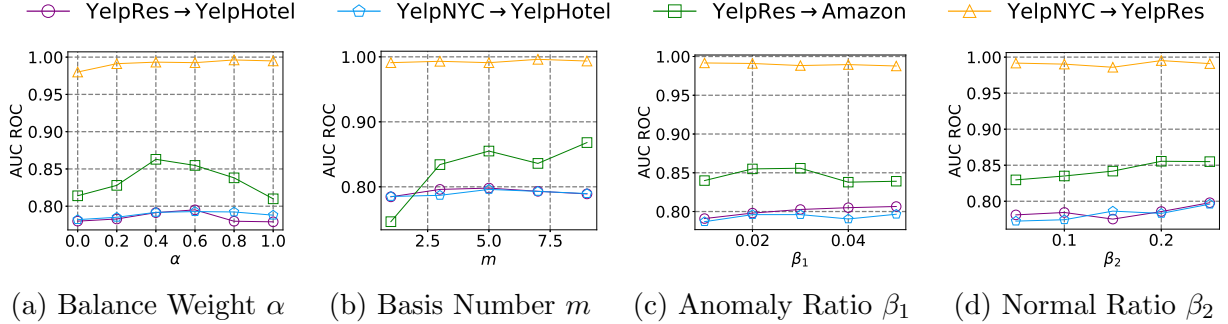


Figure 5.4: Sensitivity analysis of the hyper-parameters  $\alpha$ ,  $m$ ,  $\beta_1$ , and  $\beta_2$  of CDFS-GAD under the one-shot task in terms of AUC-ROC.

### 5.3.6 t-SNE Visualization

Figure 5.5 presents t-SNE visualizations of the node embeddings generated by three high-performing models in the one-shot setting. We employ t-SNE to reduce the high-dimensional embeddings to two dimensions, making it easier to visually assess how well each method separates normal and anomalous nodes. From the plots, it is evident that CDFS-GAD exhibits a clear separation between normal (blue) and anomalous (orange) nodes, forming two well-defined clusters. In contrast, meta-GDN and ACT yield more interspersed distributions, with anomalies scattered among normal samples. This difference suggests that the representations learned by CDFS-GAD are more discriminative, enabling it to isolate anomalous nodes even under severe label scarcity. The superior clustering performance of CDFS-GAD can be attributed to its ability to leverage the richly labeled source domain alongside the single labeled anomaly in the target domain. By aligning the representation spaces of the source and target domains, while still preserving domain-specific characteristics, CDFS-GAD learns to better distinguish anomalies from normal nodes in the target graph. Consequently, it achieves a clearer decision boundary in the latent space, as illustrated by the more notable orange-blue cluster separation in the t-SNE plots.

## 5.4 Conclusion

In this chapter, we present the first investigation into cross-domain few-shot graph anomaly detection and also propose the CDFS-GAD framework to address this challenge. CDFS-GAD effectively bridges the gap between source and target domains through the integration

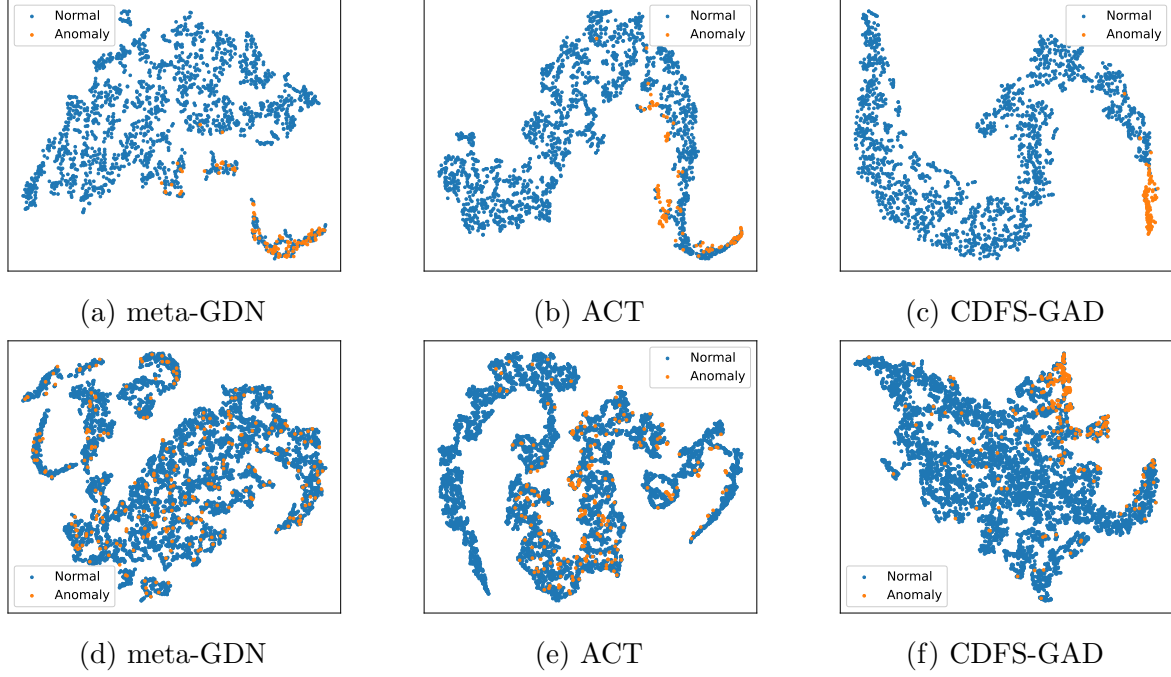


Figure 5.5: A two-dimensional t-SNE visualization of node embeddings learned by three high-performing graph anomaly detection (GAD) methods under the one-shot setting. The top row (a-c) corresponds to the cross-domain transfer YelpHTL→YelpRes, while the bottom row (d-f) corresponds to YelpHTL→Amazon. Each point represents a node’s embedding, where blue denotes normal nodes and orange denotes anomalous nodes. A clear separation between blue and orange clusters indicates a stronger capability to distinguish anomalies from normal samples.

of a domain-adaptive graph contrastive learning module and a prompt tuning module. Together, these components facilitate the extraction of common features across domains while preserving the unique, domain-specific features of each domain. To effectively utilize the few-shot labels from the target domain, we proposed a domain-adaptive hypersphere classification loss complemented by a self-training stage, both of which are crucial for optimizing performance in environments with scarce labeled data. These integrated components collectively enhance the detection accuracy in the target domain under few-shot constraints. Our experimental results confirm the effectiveness of each component and demonstrate superior performance across various domain pairs and few-shot conditions in comparison to existing GAD methods.

# Chapter 6

## Semi-supervised Anomaly Detection with Extremely Limited Labels in Dynamic Graphs

### 6.1 Introduction

Anomaly detection (AD) is a process of identifying irregular patterns in the data that deviate from expected behavior [18]. In today’s interconnected world, many datasets from various domains inherently form complex network structures [176, 186, 187, 21, 161]. These network structures are often contaminated with anomalies. For instance, financial networks may exhibit irregular transactional patterns indicative of fraud, while social networks could contain spurious accounts engaged in malicious activities. In light of the inherent complexity of these graph-structured datasets and the severe consequences of ignoring anomalies, there has been a growing research emphasis on graph anomaly detection (GAD) in recent years.

Due to the rarity of anomaly events in real-world applications, existing GAD methods primarily rely on unsupervised learning-based approaches. These methods typically utilize reconstruction-based [34, 108, 200] and contrastive learning-based techniques [103, 204, 197, 123] to learn normal patterns within unlabeled data. Anomalies are then detected as deviations from these patterns. For example, reconstruction-based methods like DOMINANT [34] and ComGA [108] employ autoencoders to reconstruct graph structures and attributes, whereas anomalies are identified through substantial reconstruction errors. On the other hand, contrastive learning-based methods, such as COLA [103] and PREM [123],

detect anomalies by comparing nodes with their surrounding structures and identifying anomalies that have large inconsistencies highlighted via contrastive losses. However, these methods primarily focus on static graphs, failing to consider the dynamic nature of real-world graphs that are characterized by constantly evolving nodes, edges, and attributes. Such dynamic information can be crucial for identifying anomalies. For example, a sudden spike in an account creation during off-peak hours could indicate an inauthentic behavior, while large financial transfers during low-transaction periods may signal irregular activities.

Recently, detecting anomalies in dynamic graphs has received increasing research interest, which requires models capable of capturing both structural dependencies and temporal patterns [203, 15, 104]. For example, AddGraph [203] processes dynamic graphs as sequences of snapshots, by using GCNs [86] and GRUs [28] to model these dependencies within the dynamic graph data; it also applies a selective negative sampling by generating negative edges based on node degrees and uses a margin-based loss to ensure normal edges receive lower anomaly scores. Similarly, TADDY [104] operates on discrete graph snapshots, by utilizing a transformer model to learn both the static graph patterns and their temporal evolution. A binary classifier is trained to distinguish anomalies by comparing real edges with pseudo-anomalous edges. However, these aforementioned methods are primarily unsupervised learning-based and also rely heavily on the quality of the pseudo labels, which may risk identifying noisy samples as anomalies due to the lack of prior anomaly knowledge.

In practice, it is often possible to acquire a limited set of labeled examples. This has spurred increased attention towards semi-supervised GAD, which combines labeled data with unlabeled data to improve detection performance [170, 87, 113, 36, 164]. For instance, SemiADC [113] uses generative adversarial networks to learn the feature distribution of normal nodes and trains a classifier to distinguish these normal features from the labeled anomalies. The anomaly scores are computed by combining the classifier output with a temporal consistency score, which is derived from evaluating node behavior across consecutive graph snapshots. Similarly, SAD [164] enhances the node representation learning with a temporal graph encoder that aggregates features across continuous time steps. The model is trained under a supervised deviation network which enforces significant differences between normal and abnormal nodes. It also introduces a pseudo-label-based contrastive learning component to improve performance. Despite the promising results of these semi-supervised dynamic GAD methods, they often hinge on the availability of an adequate number of labeled anomalies. In reality, the rarity and difficulty of accurately identifying anomalies make this assumption unrealistic in real-world applications, thereby hampering the effectiveness of these methods.

To address these challenges, we propose EL<sup>2</sup>-DGAD, a novel GAD approach designed

to enhance the robustness of the existing methods for dynamic graph anomaly detection, especially under the constraints of limited labeled data. Our framework focuses on two critical aspects: improving feature learning in dynamic graph structures and strengthening the robustness of the loss function.

First, we introduce a new transformer-based dynamic graph encoder designed to effectively capture both the temporal and structural properties of dynamic graphs. Unlike existing methods that primarily focus on local dependencies by using GNN-based approaches [182, 164], our encoder leverages both local and global attention mechanisms. The former extracts fine-grained patterns from a node’s neighborhood, while the latter aggregates these patterns across the graph, thereby enhancing the detection of subtle and extensive anomalies across the graph. Furthermore, we integrate continuous-time embeddings into the transformer’s attention mechanisms, allowing our model to seamlessly incorporate the temporal dynamics and learn the precise, time-sensitive representations. This contrasts with the discrete snapshot-based methods [203, 103, 15], which lose precision by grouping events into fixed intervals, making it harder to detect time-sensitive anomalies.

Furthermore, to effectively utilize the limited labels, we propose a novel ego-context hypersphere classification loss. It learns a context-aware hypersphere boundary by considering “ego” (target instance) and its “context” (historically evolving graph dynamics). By learning a robust boundary for abundant normal data, rather than modeling the irregular and diverse patterns of anomalies, we reduce the reliance on a large number of labeled anomalies. Instances that deviate from this learned dynamic boundary can be effectively identified as anomalies, even in scenarios with minimal supervision. Lastly, to further improve model generalization, we integrate an ego-context contrastive loss, which ensures the consistency of neighboring information with target instances by utilizing the vast amount of unlabeled data.

To summarize, our main contributions in this chapter include:

- **Novel Problem Setting:** To the best of our knowledge, we are the first to make an attempt to investigate and address anomaly detection problem in dynamic graphs with extremely scarce labeling.
- **Improving Dynamic Graph Feature Learning:** We propose a transformer-based encoder tailored for dynamic graphs. The encoder aims at capturing local and global structural dependencies while integrating continuous-time dynamics, thus enhancing the model’s ability to detect subtle and time-sensitive anomalies within evolving graph structures.

- **Enhancing Anomaly Detection with Limited Labels:** We propose an ego-context hypersphere classification loss to construct a robust and context-aware boundary for normal patterns, alongside an ego-context contrastive loss to leverage unlabeled data for improved generalization. These two losses jointly enable effective anomaly detection in dynamic graphs under extremely limited labeling conditions.
- **Empirical Validation:** Extensive experiments on four datasets and three extremely low label rates demonstrate the effectiveness of the proposed method in comparison to the existing GAD methods.

## 6.2 Methodology

### 6.2.1 Problem Formulation

In this chapter, we define a dynamic graph as  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  represents the set of nodes and  $\mathcal{E}$  consists of temporal edges. Each edge, denoted by  $\delta^t = (v_i, v_j, t, x_{ij})$ , captures an interaction from node  $v_i$  to node  $v_j$  at time  $t$ , with  $x_{ij}$  serving as the edge feature. To model the temporal dynamics effectively, we introduce two key subgraphs: the ego-graph  $\mathcal{G}^t$ , which includes all events up to and including time  $t$ , and the context graph  $\mathcal{G}^{t-}$ , which contains all events up to time  $t$  but excludes the event occurring at  $t$ . Specifically, our method learns to capture the consistency between the ego-graph and its historical context graph. If an event at time  $t$  significantly deviates from this learned consistency, it is flagged as potentially anomalous.

Our proposed  $\text{EL}^2\text{-DGAD}$  framework aims at detecting anomalous edges in this dynamic graph setting, where each edge  $\delta^t$  is associated with a latent label  $y^t$ , indicating whether the edge is normal ( $y^t = 0$ ) or abnormal ( $y^t = 1$ ). In this study, we have access to only a limited number of labeled edges, denoted as  $Y^L$ , while the majority of edges remain unlabeled ( $Y^U$ ), with  $|Y^L| \ll |Y^U|$ . The goal of our study is to develop an algorithm that assigns an anomaly score  $s^t$  to each edge, represented as  $f(\delta^t) = s^t$ , to effectively distinguish between normal and anomalous edges.

### 6.2.2 Overview

The overview of  $\text{EL}^2\text{-DGAD}$  is shown in Figure 6.1. Specifically, for each input sample, an ego and a context graph are simultaneously defined. Each of the graphs is encoded

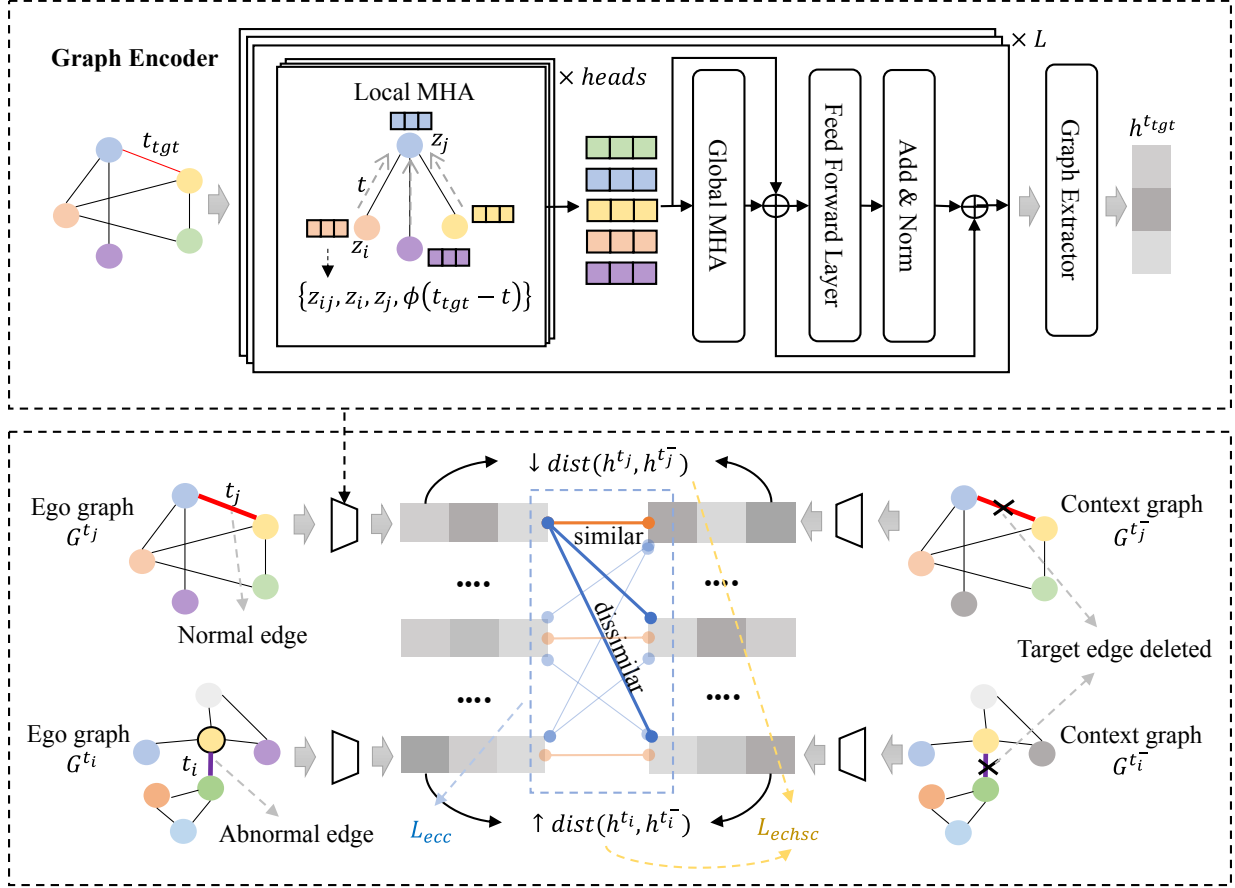


Figure 6.1: An illustration of the architecture of EL<sup>2</sup>-DGAD.

through a transformer-based dynamic graph encoder, which builds on top of a series of local and global attentions (Section 6.2.3). During model training, we introduce an ego-context hypersphere classification loss in Section 6.2.4 that classifies edges by quantifying the similarity between their ego representation and the corresponding dynamic neighboring context. Finally, an ego-context contrastive loss in Section 6.2.4 is incorporated that is designed to mine and learn from the patterns of normality prevalent within the graph based on unlabeled data.

### 6.2.3 Graph Encoder

In scenarios with extremely limited labeled data, it is crucial to design a robust model that effectively captures the temporal and structural dependencies within dynamic graphs. These graphs are constantly evolving, and anomalies may manifest through unexpected temporal patterns or deviations from typical dynamic behaviors. For example, a node that is usually inactive suddenly exhibiting a surge in activity could indicate an anomalous behavior. In addition to temporal patterns, both the influence of a node’s neighbors and the overall graph structure can reveal anomalies. Neighbor relationships capture local patterns within a node’s immediate vicinity, where anomalies might appear as atypical connections or attribute deviations. Conversely, graph-wide structures represent the global organization of the graph, where anomalies may disrupt community structures or introduce unexpected cross-community interactions. Therefore, a robust backbone model must integrate both the local and global information while simultaneously capturing the temporal evolution of the graph.

To address these challenges, we propose a transformer-based dynamic graph encoder designed to effectively capture the temporal and structural properties of dynamic graphs. Drawing inspiration from recent advancements in graph transformers [190, 20], our approach leverages local attention mechanisms to extract fine-grained patterns from a node’s neighborhood and global attention mechanisms to aggregate these patterns across the entire graph, thereby modeling both local and global dependencies. Furthermore, continuous-time embeddings are integrated into the attention mechanism, allowing the model to seamlessly incorporate temporal dynamics.

Concretely, our graph encoder takes a subgraph at time  $t$  as the input (e.g.,  $\mathcal{G}^t$  or  $\mathcal{G}^{t^-}$ ) and generates node representations that reflect the dynamic status of the nodes. We start by aggregating 1-hop neighborhood information through a masked multi-head self-attention, which we refer to as the local MHA. To capture the evolving nature of dynamic graphs, we define the entries of  $Q$ ,  $K$ , and  $V$  at layer  $l$  as follows:

$$Q_{ij}^l = \text{MLP}(\text{Concat}(x_{ij}, z_i^{l-1}, z_j^{l-1}, \phi(\Delta_t)))W^{Q,l} \quad (6.1)$$

$$K_{ij}^l = \text{MLP}(\text{Concat}(x_{ij}, z_i^{l-1}, z_j^{l-1}, \phi(\Delta_t)))W^{K,l} \quad (6.2)$$

$$V_{ij}^l = \text{MLP}(\text{Concat}(x_{ij}, z_i^{l-1}, z_j^{l-1}, \phi(\Delta_t)))W^{V,l} \quad (6.3)$$

where  $z_i^{l-1}$  is the output for node  $i$  from layer  $l-1$ , and  $z_i^0$  is a learnable embedding vectors specified by the degree of  $v_i$ , as detailed in [190]. MLP denotes multi-layer perceptrons.  $\Delta_t = t - t_{ij}$ , where  $t_{ij}$  is the occurring time of edge  $(v_i, v_j)$ .  $\phi(\cdot)$  is the sinusoidal embedding function [167], which generates a time representation by applying sine and cosine functions to the input time difference  $\Delta_t$ , scaled by a range of frequencies.

Then, the attention output of node  $i$  for a single head can be formulated as:

$$a_i^l = \sum_j \text{softmax}_j \left( \frac{Q_{ij}^l K_{ij}^{lT}}{\sqrt{d}} M_{ij} \right) V_{ij}^l \quad (6.4)$$

where  $M$  is the adjacency matrix of the input graph. Next, we fed the node representation obtained from the local MHA, as the input to the regular multi-head self-attention in [167], which we referred to as the global MHA. The local MHA focuses on learning neighborhood information which is similar to GNNs but with their inherent limitations, whereas the global attention broadens the perspective without any explicit structural reliance. Thus, this combination enhances the expressiveness of our model in representing graph structures, effectively balancing the strengths of local and global information processing.

Finally, the output of layer  $l$  is obtained as

$$z^l = \text{LN}(\text{FFN}(z^{l-1,loc} + z^{l-1,glo})) + z^{l-1,loc} + z^{l-1,glo} \quad (6.5)$$

where  $z^{l-1,loc}$  and  $z^{l-1,glo}$  are the output from the local MHA and global MHA, LN and FFN denote the layer normalization [95] and the feed-forward blocks respectively.

## 6.2.4 Model Training

### Ego-context HSC loss

To train the model, we incorporate the hypersphere classification (HSC) loss [142], which is well-suited for anomaly detection tasks where labeled anomaly data is limited. Traditional classification approaches often assume that similar data points cluster naturally, but this assumption does not hold for anomalies, which typically do not form distinct clusters [18]. The HSC loss addresses this limitation by constructing a decision boundary in the form of a hypersphere, tailored to distinguish between normal and anomalous data.

Under our context, the HSC loss of a given sample at  $t$  is defined as:

$$\mathcal{L}^t = -y^t \log l(x^t) - (1 - y^t) \log (1 - l(x^t)) \quad (6.6)$$

where  $l(z) = \exp(-\|z\|^2)$ ,  $x^t$  and  $y^t$  are the latent representation and label of an interaction event at time  $t$ . This formulation is a variant of the cross-entropy loss but with a unique focus: it constructs a spherical decision boundary that compacts normal samples towards a fixed center (the zero vector), while pushing anomalies away by maximizing their distance

from this center. In essence, the goal is to pull normal data points closer to the origin, creating a compact representation, and to push out anomalous points to better distinguish them. The final classification output is given by  $1 - l(x^t)$ , which represents the probability of an interaction being classified as normal.

To better align with the dynamic nature of graphs, where anomalous interactions are defined relative to their evolving environment, we modify the original HSC loss to explicitly incorporate temporal and structural context. Specifically, we represent the interaction at time  $t$  using two graph perspectives: the ego graph  $\mathcal{G}^t$ , which captures the evolving interactions including the event at  $t$ , and the context graph  $\mathcal{G}^{t-}$ , which reflects the recent neighborhood structure up to but not including  $t$ .

The latent representation for the interaction at time  $t$  is then defined as the difference between these two graph perspectives:

$$x^t = h^t - h^{t-} \quad (6.7)$$

where  $h^t$  and  $h^{t-}$  are the graph representations for ego and context graphs, respectively. Here,  $h^t$  is computed as

$$h^t = f_o(\text{concat}(z_i^t, z_j^t)) \quad (6.8)$$

where  $f_o$  is a two-layer MLP and  $z_i^t$  is the outputs of node  $i$  for  $\mathcal{G}^t$  from the graph encoder. A separate graph encoder with different weights is used to compute  $h^{t-}$  for the context graph.

By replacing the original latent representation in the HSC loss with  $h^t - h^{t-}$ , we formulate the ego-context HSC loss as

$$\mathcal{L}_t^{echsc} = y^t \|h^t - h^{t-}\|^2 - (1 - y^t) \log(1 - \exp(-\|h^t - h^{t-}\|^2)) \quad (6.9)$$

This modified loss directly measures how the addition of an interaction affects the deviation of the graph representation from the scenario without the interaction. Essentially,  $\mathcal{L}_t^{echsc}$  classifies interactions based on how well an edge aligns with the established patterns in the evolving graph structure from the past.

## Ego-context Contrastive Loss

To mitigate the risk of overfitting to a limited number of labeled samples, we enhance the proposed framework with a contrastive learning component by utilizing the abundant unlabeled data. Here, we aim to harness the rich information in the unlabeled samples, thereby

enhancing the ability of the model to discern between normal and anomalous patterns. A straightforward adaptation of the current framework is to maximize the similarity within the ego-context graph pairs from the same edge while minimizing the similarity for pairs from different edges. The underlying assumption is that the majority of edges are aligned with their context, given that the anomalies are rare in real-world situations. Specifically, the loss for an edge occurring at time  $t$  is defined as:

$$\mathcal{L}_t^{ecc} = -\log(S(f_p(h^t), f_p(h^{t^-}))) - \log(1 - S(f_p(h^t), f_p(h^{t_{neg}^-}))) \quad (6.10)$$

where  $h^{t_{neg}^-}$  represents the context graph representation of a different edge at  $t_{neg}$  that is selected randomly within a training batch. The function  $S(\cdot, \cdot)$  computes the cosine similarity between two inputs and applies a linear mapping to convert the value within 0 and 1. We use the cosine similarity instead of the one based on Euclidean distance as in Eq. 6.9 to let the model focus on learning the normal patterns from a different angle without compromising its ability to identify anomalies through the semi-supervised loss. The use of  $f_p(\cdot)$ , an MLP projector function, aligns with standard practices in contrastive learning which ensures that the unsupervised learning framework does not overshadow the primary task.

Finally, the overall training loss is defined as:

$$\mathcal{L} = \mathcal{L}^{echsc} + \lambda \mathcal{L}^{ecc} \quad (6.11)$$

where  $\lambda$  is a hyper-parameter controlling the contribution of the contrastive loss.

## 6.3 Experiments

### 6.3.1 Experimental Setting

#### Datasets

We evaluate our approach on the following dynamic network datasets: UCI [122], Digg [31], Wikipedia [88], and Reddit [88]. The Wikipedia dataset tracks user interactions, where each interaction corresponds to a user editing a wiki page, with the network evolving as users contribute over time. Dynamic labels indicate whether a user is banned from posting. The UCI Messages dataset represents a social network from an online community of students at the University of California, Irvine, where each node denotes a user, and edges signify exchanged messages. The Digg dataset is collected from the social news website Digg.com,

which captures user interactions with nodes representing users and edges indicating replies between them. The Reddit dataset is a dynamic network that tracks active users posting across various subreddits, with each interaction representing a user posting in a specific subreddit. Dynamic labels indicate whether a user is banned from posting in a particular subreddit.

Due to the scarcity of real-world dynamic network datasets with true anomaly labels—and following a common practice in the literature—we introduce synthetic anomalies into the UCI and Digg datasets [104, 203]. Specifically, since these datasets do not contain labeled anomalous edges, we adopt an anomaly injection procedure similar to that in [104]. The process involves clustering nodes into multiple clusters using spectral clustering based on the graph’s structure and then generating a small portion of anomalous edges by randomly connecting nodes from different clusters, ensuring these connections are absent in the original graph. 3% of anomalies are injected to reflect the realistic scenario where anomalies are typically rare. To seamlessly integrate these synthetic anomalies into the dynamic setting, the generated edges are uniformly inserted at random positions among the original edges. Each anomalous edge is assigned a timestamp by randomly selecting a value between the nearest preceding and succeeding normal edges’ timestamps, thus maintaining temporal consistency with the original data.

The datasets are chronologically divided into training, validation, and testing sets based on interaction timings, with a split ratio of 50/20/30. This means that 50% of the interactions (or edges) from the entire dataset—representing earlier periods—are used for training, 20% for validation, and 30% for testing. This temporal division ensures that the model is trained on earlier interactions, validated on more recent data, and tested on the latest interactions, which mirrors real-world scenarios where future data is unknown at training time. In our experiments, we focus on evaluating performance under extremely limited label conditions by providing only 1, 2, or 3 labeled anomaly examples. Specifically, within the 50% training set, only 1, 2, or 3 labeled anomaly examples are provided. For the labeled normal samples, we retain a proportion that matches the ratio of labeled anomalies to the total number of anomalies. The rest of the samples remained unlabeled. The summary statistics of all datasets are shown in Table 6.1.

## Baselines and Performance Evaluation

We benchmark our proposed method against several recent GAD models, namely GDN [36], SL-GAD [204], PREM [123], AddGraph [203], TADDY [104], TGAT [182], and SAD [164]. Among these, GDN, SL-GAD, and PREM are designed for static graph anomaly detection, while AddGraph, TADDY, TGAT, and SAD are tailored for dynamic graph settings.

Table 6.1: Summary statistics for datasets.

| Dataset   | # Nodes | # Edges | # Anomalies |
|-----------|---------|---------|-------------|
| UCI       | 1899    | 14253   | 415         |
| Digg      | 29955   | 87709   | 2554        |
| Reddit    | 10984   | 672447  | 366         |
| Wikipedia | 9227    | 157474  | 217         |

GDN employs simplified GCNs [177] to generate node representations. The model is optimized under a deviation loss that enforces anomaly scores to follow a Gaussian distribution. SL-GAD constructs contextual subgraphs around each node and uses attribute reconstruction with multi-view contrastive learning to capture both attribute and structural anomalies. PREM extracts each node’s neighbor features through a propagation matrix and aligns the node’s features with its neighborhood information via contrastive learning.

For dynamic GAD models, AddGraph processes dynamic graphs as discrete snapshots, applying GCNs followed by GRUs to capture both structural and temporal dependencies, and optimizes a margin-based loss to maximize the distance between real and randomly sampled edges. Similarly, TADDY slides through dynamic graphs to create discrete snapshots and employs a transformer model to capture multi-time-step substructures. Anomalies are then detected by comparing them to pseudo-anomalous edges. TGAT, built on Graph Attention Networks (GATs) [168], incorporates self-attention and time encoding to capture temporal and topological patterns. TGAT also serves as the backbone encoder in SAD, which further enhances dynamic GAD by integrating a deviation network to leverage labeled data and a pseudo-label contrastive learning module to utilize unlabeled data.

Note that only GDN, TADDY, AddGraph, and SAD can utilize label information by design. For a fair, label-inclusive comparison for other models, we perform a similar setting as in [164]: applying a cross-entropy loss to the edge representations, which are derived from the concatenated node embeddings of the edge endpoints. Specifically, these edge representations are computed using an MLP network, as outlined in Eq. 6.8. Thus, the performance of all models can be evaluated based on the classification results in terms of AUC-ROC.

## Implementation

To construct the ego-graph  $\mathcal{G}^t$ , we include all edges with timestamps up to and including time  $t$ , capturing the interactions leading up to that specific time point. For the context

graph  $\mathcal{G}^{t-}$ , we similarly include all edges up to time  $t$ , but explicitly exclude any edge that occurs exactly at time  $t$ . This differentiation ensures that the context graph only represents historical information, without the current event. To manage the potentially large sizes of  $\mathcal{G}^t$  and  $\mathcal{G}^{t-}$ , we implement a sampling strategy that retains a subset of neighboring nodes for each edge endpoint. Specifically, we randomly sample 25, 10, and 5 neighboring nodes at the first, second, and third hops, respectively, with the condition that each sampled neighbor event must have a timestamp no later than the target edge’s timestamp, ensuring that only past interactions are considered.

In addition, our model uses a two-layer transformer with 4 attention heads, and the hidden size is set to 128 across all modules. The contrastive loss weight  $\lambda$  is set to 0.01 for Digg, UCI, Reddit and 10 for Wikipedia. The models are trained for 20 epochs using the Adam optimizer [84], with a learning rate of 0.0001 for the Digg and UCI datasets, and 0.001 for the Wikipedia and Reddit dataset.

### 6.3.2 Comparison Study

Extensive results in Table 6.2 show that our  $\text{EL}^2\text{-DGAD}$  consistently surpasses all comparison methods across four datasets, particularly in scenarios where labeled anomalies are extremely limited. The results highlight the effectiveness of our approach in incorporating both temporal and structural context, enabling the model to maintain robustness even under severe label scarcity.

For models like GDN, PREM, and SL-GAD, which do not incorporate temporal information into their design, the decline in their performances underscores the critical importance of capturing temporal dependencies in dynamic graph anomaly detection. Although AddGraph and TADDY are designed for dynamic graphs, their performances still lag behind ours by a significant margin. These methods rely on discrete graph snapshots, where timestamps of events within a single snapshot are assumed to be identical. Consequently, this can miss important details about the continuous evolution of the graph’s structure. Moreover, they lack a robust semi-supervised learning mechanism, making it challenging to effectively utilize the limited labeled data. In contrast, our approach operates directly on continuous dynamic graphs, with the transformer-based encoder attending to the most relevant neighbors based on fine-grained temporal information. Additionally, we introduce an ego-context hypersphere classification loss that enforces the alignment of each edge with its immediate surrounding temporal and structural context. Together, these features enable  $\text{EL}^2\text{-DGAD}$  to more efficiently utilize the continuously evolving graph patterns, making it better suited to detect anomalies that are difficult to align with the normal behavior.

Table 6.2: Experiment comparison with the state-of-the-art GAD methods in terms of AUC. The best results are marked in blue. 1, 2, and 3 in the “labels” row represent the number of labeled anomalies in the training set.

| Dataset               | Wikipedia    |              |              | UCI          |              |              | Digg         |              |              | Reddit       |              |              |
|-----------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Labels                | 1            | 2            | 3            | 1            | 2            | 3            | 1            | 2            | 3            | 1            | 2            | 3            |
| AddGraph              | 0.631        | 0.641        | 0.663        | 0.732        | 0.732        | 0.754        | 0.739        | 0.740        | 0.742        | 0.578        | 0.580        | 0.582        |
| TGAT                  | 0.660        | 0.674        | 0.694        | 0.764        | 0.790        | 0.808        | 0.800        | 0.804        | 0.812        | 0.527        | 0.541        | 0.555        |
| GDN                   | 0.516        | 0.529        | 0.533        | 0.510        | 0.546        | 0.547        | 0.511        | 0.510        | 0.513        | 0.532        | 0.531        | 0.533        |
| SL-GAD                | 0.467        | 0.524        | 0.526        | 0.503        | 0.497        | 0.502        | 0.494        | 0.497        | 0.504        | 0.503        | 0.505        | 0.500        |
| TADDY                 | 0.616        | 0.617        | 0.618        | 0.769        | 0.765        | 0.772        | 0.811        | 0.841        | 0.842        | 0.516        | 0.517        | 0.518        |
| SAD                   | 0.661        | 0.674        | 0.710        | 0.774        | 0.798        | 0.816        | 0.809        | 0.820        | 0.841        | 0.561        | 0.566        | 0.592        |
| PREM                  | 0.549        | 0.555        | 0.555        | 0.531        | 0.532        | 0.536        | 0.512        | 0.513        | 0.515        | 0.516        | 0.535        | 0.536        |
| EL <sup>2</sup> -DGAD | <b>0.722</b> | <b>0.725</b> | <b>0.732</b> | <b>0.839</b> | <b>0.840</b> | <b>0.842</b> | <b>0.839</b> | <b>0.844</b> | <b>0.846</b> | <b>0.611</b> | <b>0.615</b> | <b>0.616</b> |

While SAD is also a semi-supervised anomaly detection framework designed for dynamic graphs, it exhibits a notable drop in performance under extreme label scarcity. Specifically, SAD relies on a deviation network as the main supervised learning component, using it to generate pseudo-labeled data that guides a separate contrastive learning process. However, when only a few labeled examples are available, the quality of these pseudo-labels declines, undermining the effectiveness of the contrastive learning component. In contrast, our approach decouples the unsupervised learning from the supervised training, allowing the model to retain its ability to generalize effectively even under extremely limited conditions. Furthermore, our approach offers additional advantages. First, the ego-context hypersphere classification loss provides a more precise instance-wise contextual comparison, compared to the deviation network used in SAD, which relies on comparisons to only global statistics. Second, our model employs a more robust transformer-based encoder, compared to the GAT encoder used in SAD, allowing it to more effectively capture temporal interactions between nodes from both local and global perspectives.

### 6.3.3 Ablation Study

Table 6.3 presents the results of the ablation study. Overall, EL<sup>2</sup>-DGAD achieves the best performance when all of the components are included. Removing the local MHA leads to a significant performance drop, highlighting its essential role in capturing neighborhood structures. Similarly, omitting the global MHA also results in a noticeable decrease in performance, underscoring the importance of capturing global node interactions. The exclusion of the fine-grained timestamp integration within the Transformer’s attention mechanism also considerably degrades performance, especially in the Wikipedia dataset,

Table 6.3: Performance evaluation on ablation experiments in terms of AUC. *w/o* denotes without a certain module.

| Dataset                                         | Wikipedia    |              |              | UCI          |              |              | Digg         |              |              |
|-------------------------------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Labels                                          | 1            | 2            | 3            | 1            | 2            | 3            | 1            | 2            | 3            |
| <i>w/o</i> Local MHA                            | 0.563        | 0.569        | 0.582        | 0.771        | 0.775        | 0.772        | 0.808        | 0.796        | 0.802        |
| <i>w/o</i> Global MHA                           | 0.694        | 0.709        | 0.716        | 0.835        | 0.836        | 0.838        | 0.802        | 0.807        | 0.800        |
| <i>w/o</i> $\phi(t)$ in MHA                     | 0.465        | 0.473        | 0.474        | 0.789        | 0.791        | 0.800        | 0.719        | 0.730        | 0.756        |
| <i>w/o</i> $\mathcal{L}^{ecc}$                  | 0.653        | 0.683        | 0.683        | 0.838        | 0.837        | 0.840        | 0.815        | 0.821        | 0.821        |
| <i>w/o</i> ego-context in $\mathcal{L}^{echsc}$ | 0.650        | 0.683        | 0.704        | 0.824        | 0.824        | 0.837        | 0.730        | 0.745        | 0.745        |
| EL <sup>2</sup> -DGAD                           | <b>0.722</b> | <b>0.725</b> | <b>0.732</b> | <b>0.839</b> | <b>0.840</b> | <b>0.842</b> | <b>0.839</b> | <b>0.844</b> | <b>0.846</b> |

where a node-to-node interaction can occur at many distinct timestamps. Ignoring these timestamps results in a large information loss, as it prevents the model from capturing the temporal dynamics necessary to detect time-sensitive anomalies effectively. This further emphasizes the importance of modeling temporal changes in dynamic graphs. Additionally, the drop in performance without the contrastive loss component  $\mathcal{L}^{ecc}$  indicates the value of leveraging unlabeled data to improve anomaly detection accuracy. Lastly, removing the ego-context contrast from  $\mathcal{L}^{echsc}$  leads to a notable decline in performance. Without ego-context contrast, the model no longer minimizes the distance between each ego graph and its context graph; instead, all normal instances are pulled toward a single center, imposing a stronger regularization. This limits the model’s ability to capture the diverse characteristics of normal instances, thereby reducing its ability to distinguish anomalies from normal events.

### 6.3.4 Parameters Sensitivity

Figure 6.2 illustrates the AUC performance for three datasets (Digg, UCI, and Wikipedia) as it varies with changes in the balance parameter  $\lambda$  for the ego-context contrastive learning component and the number of neighbors in the first hop. In Figure 2(a), the Wikipedia dataset shows improved AUC scores with higher  $\lambda$  values compared to Digg and UCI. This may be due to the greater complexity of anomalies in Wikipedia, making it a more challenging task under extremely limited conditions. Consequently, increasing the value of  $\lambda$  enhances the model’s ability to capture general patterns of normal samples, which improves the robustness of the model.

In Figure 2(b), AUC performance for all three datasets slightly increases as the number of neighbors rises, indicating that additional neighborhood information helps the model better distinguish between normal and anomalous interactions within the graphs. Overall,

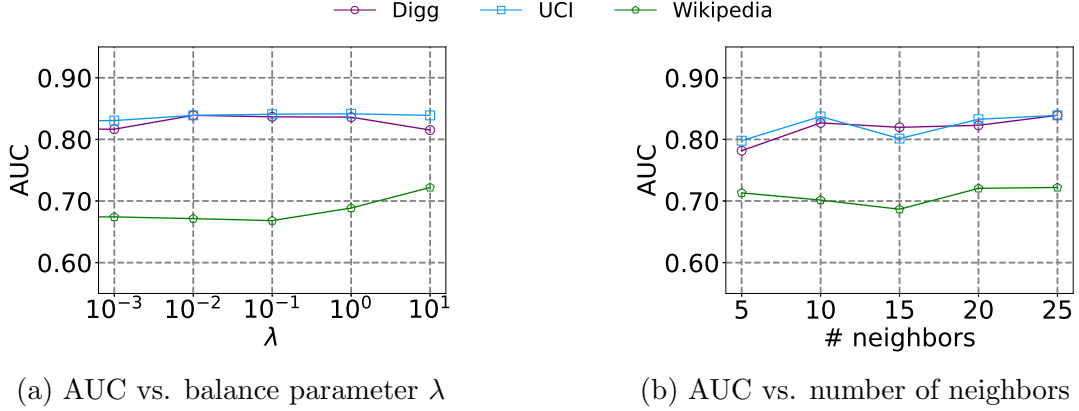


Figure 6.2: Sensitivity analysis of  $\lambda$  and neighbors number when there is one labeled anomaly available in the training dataset.

the model’s sensitivity to changes in the value of  $\lambda$  and the number of neighbors remains relatively low, suggesting that it is robust to variations in these hyperparameters.

### 6.3.5 t-SNE Visualization

Figure 6.3 shows t-SNE visualizations of the output embeddings from the layer before the anomaly scoring in the Wikipedia dataset, where the selected four models were trained with supervision from only one labeled anomaly. In these visualizations, blue points represent the normal instances, while orange points denote the anomalies. In the embeddings produced by TGAT and AddGraph, the anomalies are dispersed throughout the normal samples, indicating a limited capacity for distinguishing between the normal and anomalous instances. Similarly, SAD shows a slight improvement with a somewhat better separation, but the anomalies remain scattered and intermingled within the normal clusters, lacking a clear boundary.

In contrast, our proposed  $EL^2$ -DGAD exhibits a relatively clearer separation, with the anomalies forming distinct clusters that are well-separated from the normal samples. This distinct clustering suggests that  $EL^2$ -DGAD more effectively learns discriminative embeddings, allowing it to capture the subtle distinctions between the normal and anomalous interactions. This improved separation highlights the robustness and effectiveness of our approach for the GAD task, particularly in the challenging scenario of training with extremely limited labeled anomalies.

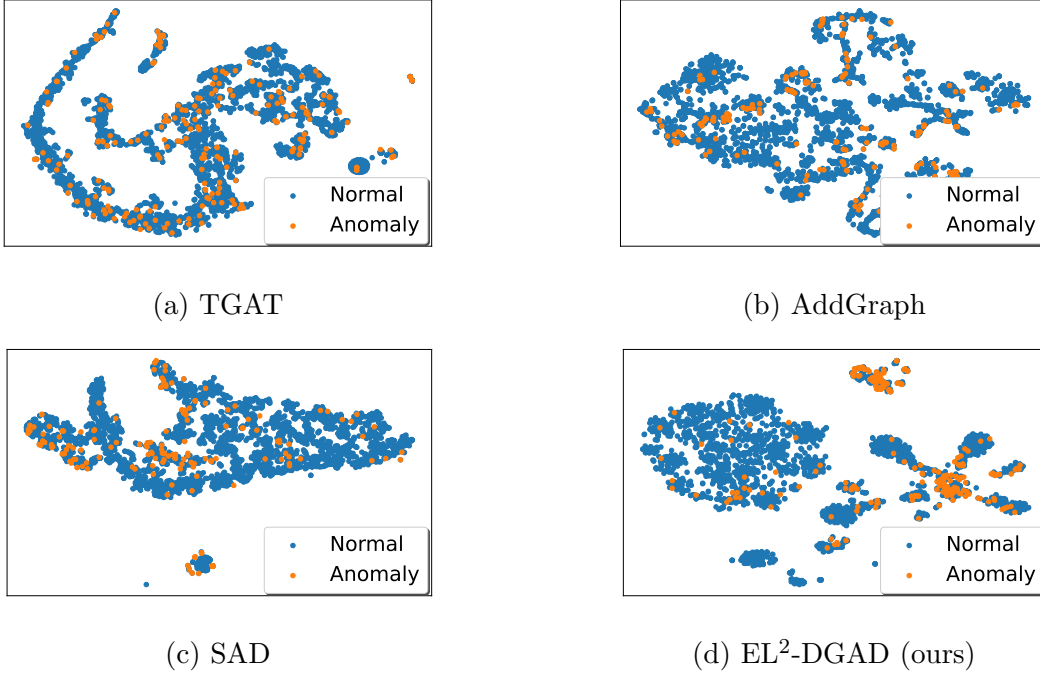


Figure 6.3: t-SNE Visualization on the Wikipedia dataset.

## 6.4 Conclusion

In this chapter, we have presented EL<sup>2</sup>-DGAD, the first framework designed to address the anomaly detection problem in dynamic graphs under the challenging condition of extremely limited labeled data. Our approach integrates a transformer-based dynamic graph encoder that captures evolving graph patterns from both local and global perspectives, leveraging continuous-time embeddings to preserve temporal precision. This design is further complemented by an ego-context hypersphere classification loss and an unsupervised ego-context contrastive loss, which together effectively utilize the limited labeled anomalies while harnessing the rich information from abundant unlabeled data. Through extensive experiments on multiple benchmark datasets across varying label rates, EL<sup>2</sup>-DGAD demonstrates significant performance improvements over existing graph anomaly detection methods. The results validate the effectiveness of our model in accurately detecting anomalies across diverse dynamic graph scenarios, even under severe label constraints.

# Chapter 7

## Conclusion

In this thesis, we introduce a series of frameworks designed to advance anomaly detection under label-scarce scenarios, with a particular focus on time series and graph-structured data. Our first three contributions focus on time series anomaly detection (TSAD). Labeled anomalies are typically rare in industrial applications, making unsupervised approaches essential. The core idea of the unsupervised anomaly detection approach is to learn the general pattern within abundant unlabeled data by assuming anomalies only occupy a small portion of the data. Thus, anomalies can be detected as deviations from these learned patterns.

Our first contribution, the CNT framework, adapts the powerful representation capabilities of contrastive learning to the TSAD domain to enhance feature learning in unlabeled datasets. However, this adaptation presents unique challenges due to the contextual nature of anomalies, as they are often defined in relation to historical observations, and the difficulty of defining negative samples in datasets where the majority of instances are normal. To address these issues, we introduce a dual loss function that combines a temporal-wise contrastive learning with neural transformations, enabling effective anomaly detection without the need for negative sampling and simultaneously preventing representation collapse.

Beyond temporal dependencies, capturing inter-variable dependencies is also critical for multivariate TSAD, as certain patterns may appear to be normal in a single time series but reveal anomalies when compared with other series. To address this issue, we propose two additional frameworks: DyGraphAD and PMGC. Unlike existing methods, which often employ graphs implicitly to support certain downstream tasks, DyGraphAD makes a first attempt to explicitly formulate the TSAD problem as a combination of graph

forecasting and time series forecasting tasks. This design enables an explicit detection of both intra- and inter-series anomalies by examining the forecasting errors across both dimensions. PMGC tackles the problem from a different angle by modeling the short-term and long-term inter-series dependencies through a combination of learnable dynamic and static graphs. Departing from existing studies that typically rely on a single type of graph, PMGC offers a more comprehensive view of inter-variable relationships. In this way, anomalies are difficult to conceal, as they must violate all of the learned inter-series relationships to go undetected.

Our last two works shift focus from time series anomaly detection to graph anomaly detection (GAD), where the goal is to identify unusual nodes, edges, or subgraphs within predefined graph structures. This shift is motivated by the prevalence of structured data in applications such as social networks, financial transaction networks, and communication systems, where the relationships between entities are as crucial as the attributes of the entities themselves. Unlike in multivariate TSAD, where inter-variable dependencies must be inferred, GAD operates on readily available graph structures, allowing us to directly model and detect anomalies within these predefined relational contexts.

For our first contribution in this area, we make a first attempt to address the GAD problem in a cross-domain few-shot setting. This setting is highly practical for real-world applications where target graphs have limited labeled data but can benefit from auxiliary graphs in related domains. However, it poses significant challenges due to differences between the two domains, including variations in graph structure, node attributes, and anomaly distributions. To tackle these challenges, we propose CDFS-GAD, which incorporates a series of domain-adaptive components to effectively facilitate a knowledge transfer from the source domain under limited supervision in the target domain.

For our final contribution, we address an even more challenging scenario where graph structures are dynamically evolving and labeled data is extremely limited. Many real-world networks, such as social or communication systems, evolve continuously, with new nodes, edges, and patterns emerging over time. These changes can alter the structure and behavior of the network, making it difficult to rely on static models for effective anomaly detection. Moreover, we assume only a limited number of labeled anomalies are available, and it is not feasible to obtain sufficient labeled data from the related domains, as in CDFS-GAD. To address these issues, we propose  $EL^2$ -DGAD, which tackles the problem from two perspectives. Specifically, we introduce a transformer-based graph encoder to enhance the model’s robustness in learning from dynamically evolving graph structures. Furthermore, we incorporate an ego-context hypersphere classification loss to establish a context-aware decision boundary that effectively separates anomalies from normal instances, under conditions of extremely limited labeling.

Collectively, these frameworks address a range of practical challenges in anomaly detection, spanning time series data and graph-structured data, and tackling real-world settings, including unsupervised, semi-supervised, and cross-domain few-shot scenarios. These contributions offer practical solutions for industries where reliable anomaly detection is essential, despite the constraints of label scarcity and complex data relationships.

While the frameworks above propose promising solutions for anomaly detection targeted at time series and graph-structured data under label-scarce scenarios, several limitations remain. First, scalability is a challenge, particularly for TSAD frameworks that rely on graph construction that results in  $O(N \times N)$  complexity ( $N$  denotes the number of nodes/series), which quickly becomes unmanageable as the dimension of the time series grows. Second, explainability is limited in the current models, making it difficult to interpret why specific data points are identified as anomalies, which can hinder transparency and trust in practical applications. Third, the models are primarily trained on domain-specific data, meaning their generalization is closely tied to the data available within each domain. In cases where even the unlabeled data is sparse, the models may struggle to generalize effectively.

To address these limitations, future research could explore several directions. To improve scalability, especially for graph-based TSAD frameworks, some directions could involve using approximation techniques to reduce the pairwise computations of graphs to a linear complexity, or adopting sparse graph learning methods that retain only essential edges, thereby decreasing the overall computational load [178]. Enhancing model explainability is another important direction, as more interpretable models would facilitate better understanding and acceptance in real-world settings. Since our proposed anomaly detection frameworks leverage GNNs as the base architecture to learn inter-series relationships or node interactions, recent methods (e.g., [145, 191, 48]) that identify key patterns or graph structures influencing predictions could be particularly useful. These explainability methods can be adapted to highlight influential time series, neighboring nodes, or sub-graph structures that contribute to detected anomalies, offering clearer insights into model decisions. Additionally, to address the challenges of data dependence and limited generalization, a promising direction is to explore the integration of large language models (LLMs) with graph neural networks (GNNs). LLMs excel at capturing complex patterns and contextual information from vast amounts of textual data. However, applying LLMs directly to graph-structured data is challenging, as graph structures are inherently non-textual. To bridge this gap, a growing number of recent studies have focused on aligning textual and structural information through the combination of GNNs and LLMs [106, 43, 64]. For example, GraphPrompter [106] introduces a GNN-generated “soft prompt” to guide LLMs, effectively combining graph and textual information to enhance model generalization across

diverse domains. Drawing inspiration from these recent advances, we could leverage LLMs to improve the generalization of the proposed anomaly detection models in learning the node interactions or inter-series relationships even with limited domain-specific data.

# References

- [1] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2485–2494, 2021.
- [2] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. Wadi: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*, pages 25–28, 2017.
- [3] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29:626–688, 2015.
- [4] Antonio L Alfeo, Mario GCA Cimino, Giuseppe Manco, Ettore Ritacco, and Gigliola Vaglini. Using an autoencoder in the design of an anomaly detector for smart manufacturing. *Pattern Recognition Letters*, 136:272–278, 2020.
- [5] Archana Anandakrishnan, Senthil Kumar, Alexander Statnikov, Tanveer Faruque, and Di Xu. Anomaly detection in finance: editors’ introduction. In *KDD 2017 Workshop on Anomaly Detection in Finance*, pages 1–7. PMLR, 2018.
- [6] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *European conference on principles of data mining and knowledge discovery*, pages 15–27. Springer, 2002.
- [7] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3395–3404, 2020.

- [8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.
- [9] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.
- [10] Md Abul Bashar and Richi Nayak. Tanogan: Time series anomaly detection with generative adversarial networks. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1778–1785. IEEE, 2020.
- [11] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *ACM computing surveys (CSUR)*, 54(3):1–33, 2021.
- [12] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [13] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.
- [14] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
- [15] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 3747–3756, 2021.
- [16] Chris U. Carmona, François-Xavier Aubet, Valentin Flunkert, and Jan Gasthaus. Neural contextual anomaly detection for time series. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2843–2851, 2022.
- [17] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.
- [18] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

- [19] Wanpracha Art Chaovaitwongse, Ya-Ju Fan, and Rajesh C Sachdeo. On the time series  $k$ -nearest neighbor classification of abnormal brain activity. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(6):1005–1016, 2007.
- [20] Dexiong Chen, Leslie O’Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *Proceedings of the International Conference on Machine Learning*, pages 3469–3489, 2022.
- [21] Tianyi Chen and Charalampos Tsourakakis. Antibenford subgraphs: Unsupervised anomaly detection in financial networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2762–2770, 2022.
- [22] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [23] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.
- [24] Xuanhao Chen, Liwei Deng, Feiteng Huang, Chengwei Zhang, Zongquan Zhang, Yan Zhao, and Kai Zheng. Daemon: Unsupervised anomaly detection and interpretation for multivariate time series. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2225–2230. IEEE, 2021.
- [25] Xuanhao Chen, Liwei Deng, Yan Zhao, and Kai Zheng. Adversarial autoencoder for unsupervised time series anomaly detection and interpretation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 267–275, 2023.
- [26] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. Learning graph structures with transformer for multivariate time-series anomaly detection in iot. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2021.
- [27] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. *IEEE Access*, 2021.

- [28] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [29] Andrew A Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2019.
- [30] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.
- [31] Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. Social synchrony: Predicting mimicry of user actions in online social media. In *Proceedings of the International Conference on Computational Science and Engineering*, volume 4, pages 151–158, 2009.
- [32] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4027–4035, 2021.
- [33] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.
- [34] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the SIAM International Conference on Data Mining*, pages 594–602, 2019.
- [35] Kaize Ding, Kai Shu, Xuan Shan, Jundong Li, and Huan Liu. Cross-domain graph anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2406–2415, 2022.
- [36] Kaize Ding, Qinghai Zhou, Hanghang Tong, and Huan Liu. Few-shot network anomaly detection via cross-network meta-learning. In *Proceedings of the Web Conference*, pages 2448–2456, 2021.
- [37] Linlin Ding, Baishuo Han, Shu Wang, Xiaoguang Li, and Baoyan Song. User-centered recommendation using us-elm based on dynamic graph model in e-commerce. *International Journal of Machine Learning and Cybernetics*, 10:693–703, 2019.

- [38] Thang N Dinh, Ying Xuan, and My T Thai. Towards social-aware routing in dynamic communication networks. In *2009 IEEE 28th International Performance Computing and Communications Conference*, pages 161–168. IEEE, 2009.
- [39] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.
- [40] Jingcan Duan, Siwei Wang, Pei Zhang, En Zhu, Jingtao Hu, Hu Jin, Yue Liu, and Zhibin Dong. Graph anomaly detection via multi-scale contrastive learning networks with augmented view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7459–7467, 2023.
- [41] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5685–5689, 2020.
- [42] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. In *Advances in Neural Information Processing Systems*, 2023.
- [43] Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 747–758, 2024.
- [44] Yuchen Fang, Jiandong Xie, Yan Zhao, Lu Chen, Yunjun Gao, and Kai Zheng. Temporal-frequency masked autoencoders for time series anomaly detection. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1228–1241. IEEE, 2024.
- [45] Sichao Fu, Xueqi Ma, Yibing Zhan, Fanyu You, Qinmu Peng, Tongliang Liu, James Bailey, and Danilo Mandic. Finding core labels for maximizing generalization of graph neural networks. *Neural Networks*, page 106635, 2024.
- [46] Sichao Fu, Qinmu Peng, Yang He, Baokun Du, and Xinge You. Towards unsupervised graph completion learning on graphs with features and structure missing. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1019–1024, 2023.

- [47] Sichao Fu, Qinmu Peng, Yang He, Xiaorui Wang, Bin Zou, Duanquan Xu, Xiao-Yuan Jing, and Xinge You. Multilevel contrastive graph masked autoencoders for unsupervised graph-structure learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.
- [48] Thorben Funke, Megha Khosla, Mandeep Rathee, and Avishek Anand. Zorro: Valid, sparse, and stable explanations in graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8687–8698, 2022.
- [49] Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2508–2517, 2021.
- [50] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. *Advances in neural information processing systems*, 31, 2018.
- [51] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.
- [52] Ian Goodfellow. Deep learning, 2016.
- [53] Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.
- [54] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.
- [55] Qiuyu Guo, Xiang Zhao, Yang Fang, Shiyu Yang, Xuemin Lin, and Dian Ouyang. Learning hypersphere for few-shot anomaly detection on attributed networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 635–645, 2022.
- [56] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering*, 26(9):2250–2267, 2013.

- [57] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.
- [58] Siho Han and Simon S Woo. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2977–2986, 2022.
- [59] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 430–433. IEEE, 2004.
- [60] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.
- [61] Akira Hayashi, Yuko Mizuhara, and Nobuo Suematsu. Embedding time series data for classification. In *International workshop on machine learning and data mining in pattern recognition*, pages 356–365. Springer, 2005.
- [62] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.
- [63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [64] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning, 2023.
- [65] Yangdong He and Jiabao Zhao. Temporal convolutional networks for anomaly detection in time series. In *Journal of Physics: Conference Series*, volume 1213, page 042050. IOP Publishing, 2019.
- [66] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, 32, 2019.

- [67] Waleed Hilal, S Andrew Gadsden, and John Yawney. Financial fraud: a review of anomaly detection techniques and recent advances. *Expert systems With applications*, 193:116429, 2022.
- [68] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.
- [69] Hadi Hojjati and Narges Armanfard. Self-supervised acoustic anomaly detection via contrastive learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3253–3257. IEEE, 2022.
- [70] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.
- [71] Wenjie Hu, Yang Yang, Ziqiang Cheng, Carl Yang, and Xiang Ren. Time-series event prediction with evolutionary state graph. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 580–588, 2021.
- [72] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pages 387–395, 2018.
- [73] RJ Hyndman. *Forecasting: principles and practice*. OTexts, 2018.
- [74] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- [75] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.
- [76] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
- [77] Yungi Jeong, Eunseok Yang, Jung Hyun Ryu, Imseong Park, and Myungjoo Kang. Anomalybert: Self-supervised transformer for time series anomaly detection using data degradation scheme. *arXiv preprint arXiv:2305.04468*, 2023.

- [78] Xin Jie, Xixi Zhou, Chanfei Su, Zijun Zhou, Yuqing Yuan, Jiajun Bu, and Haishuai Wang. Disentangled anomaly detection for multivariate time series. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 931–934, 2024.
- [79] Ming Jin, Yixin Liu, Yu Zheng, Lianhua Chi, Yuan-Fang Li, and Shirui Pan. Anemone: Graph anomaly detection with multi-scale contrastive learning. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 3122–3126, 2021.
- [80] Parisa Kaghazgaran, James Caverlee, and Anna Squicciarini. Combating crowd-sourced review manipulators: A neighborhood-based approach. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 306–314, 2018.
- [81] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.
- [82] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. Graph anomaly detection with graph neural networks: Current status and challenges. *IEEE Access*, 10:111820–111829, 2022.
- [83] Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a rigorous evaluation of time-series anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7194–7201, 2022.
- [84] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [85] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [86] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [87] Atsutoshi Kumagai, Tomoharu Iwata, and Yasuhiro Fujiwara. Semi-supervised anomaly detection on attributed graphs. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2021.

- [88] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1269–1278, 2019.
- [89] Donghwoon Kwon, Kathiravan Natarajan, Sang C Suh, Hyunjoo Kim, and Jinoh Kim. An empirical study on network anomaly detection using convolutional neural networks. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1595–1598. IEEE, 2018.
- [90] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.
- [91] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. Revisiting time series outlier detection: Definitions and benchmarks. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)*, 2021.
- [92] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 156–165. IEEE, 2017.
- [93] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [94] Jongsoo Lee, Byeongtae Park, and Dong-Kyu Chae. Duogat: Dual time-oriented graph attention networks for accurate, efficient and explainable anomaly detection on time-series. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1188–1197, 2023.
- [95] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *ArXiv e-prints*, pages arXiv–1607, 2016.
- [96] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Madgan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International conference on artificial neural networks*, pages 703–716. Springer, 2019.

- [97] Jia Li, Shimin Di, Yanyan Shen, and Lei Chen. Fluxev: a fast and effective unsupervised framework for time-series anomaly detection. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 824–832, 2021.
- [98] Yangyang Li, Yipeng Ji, Shaoning Li, Shulong He, Yinhao Cao, Yifeng Liu, Hong Liu, Xiong Li, Jun Shi, and Yangchao Yang. Relevance-aware anomalous users detection in social network via graph neural network. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2021.
- [99] Yuxin Li, Wenchao Chen, Bo Chen, Dongsheng Wang, Long Tian, and Mingyuan Zhou. Prototype-oriented unsupervised anomaly detection for multivariate time series. In *International Conference on Machine Learning*, pages 19407–19424. PMLR, 2023.
- [100] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.
- [101] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth IEEE international conference on data mining*, pages 413–422. IEEE, 2008.
- [102] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.
- [103] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2378–2392, 2022.
- [104] Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12081–12094, 2021.
- [105] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference*, pages 417–428, 2023.
- [106] Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V Chawla. Can we soft prompt llms for graph learning tasks? In *Companion Proceedings of the ACM on Web Conference 2024*, pages 481–484, 2024.

- [107] Qingqing Long, Yuchen Yan, Peiyan Zhang, Chen Fang, Wentao Cui, Zhiyuan Ning, Meng Xiao, Ning Cao, Xiao Luo, Lingjun Xu, et al. Towards graph prompt learning: A survey and beyond. *arXiv preprint arXiv:2408.14520*, 2024.
- [108] Xuexiong Luo, Jia Wu, Amin Beheshti, Jian Yang, Xiankun Zhang, Yuan Wang, and Shan Xue. Comga: Community-aware attributed graph anomaly detection. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 657–665, 2022.
- [109] Junshui Ma and Simon Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745. IEEE, 2003.
- [110] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.
- [111] Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE, 2016.
- [112] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.
- [113] Xuying Meng, Suhang Wang, Zhimin Liang, Di Yao, Jihua Zhou, and Yujun Zhang. Semi-supervised anomaly detection in dynamic communication networks. *Information Sciences*, 571:527–542, 2021.
- [114] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960, 2018.
- [115] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.
- [116] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access*, 7:1991–2005, 2018.

- [117] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.
- [118] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.
- [119] Seol-Hyun Noh. Analysis of gradient vanishing of rnns and performance comparison. *Information*, 12(11):442, 2021.
- [120] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [121] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [122] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2):155–163, 2009.
- [123] Junjun Pan, Yixin Liu, Yizhen Zheng, and Shirui Pan. Prem: A simple yet effective approach for node-level graph anomaly detection. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1253–1258, 2023.
- [124] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2):1–38, 2021.
- [125] Guansong Pang, Chunhua Shen, and Anton Van Den Hengel. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 353–362, 2019.
- [126] Guansong Pang, Anton van den Hengel, Chunhua Shen, and Longbing Cao. Toward deep supervised anomaly detection: Reinforcement learning from partially labeled anomaly data. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1298–1308, 2021.
- [127] John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. *Proceedings of the VLDB Endowment*, 15(11):2774–2787, 2022.

- [128] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.
- [129] R Pascanu, T Mikolov, and Y Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.
- [130] Eduardo HM Pena, Marcos VO de Assis, and Mario Lemes Proença. Anomaly detection using forecasting methods arima and hwds. In *2013 32nd international conference of the chilean computer science society (sccc)*, pages 63–66. IEEE, 2013.
- [131] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
- [132] Lucas Pinheiro Cinelli, Matheus Araújo Marins, Eduardo Ant3nio Barros da Silva, and S3rgio Lima Netto. Variational autoencoder. In *Variational Methods for Machine Learning with Applications to Deep Networks*, pages 111–149. Springer, 2021.
- [133] Theivendiram Pranavan, Terence Sim, Arulmurugan Ambikapathi, and Savitha Ramasamy. Contrastive predictive coding for anomaly detection in multi-variate time series data. *arXiv preprint arXiv:2202.03639*, 2022.
- [134] Jiaxing Qi, Zhongzhi Luan, Shaohan Huang, Carol Fung, Hailong Yang, Hanlu Li, Danfeng Zhu, and Depei Qian. Logencoder: Log-based contrastive representation learning for anomaly detection. *IEEE Transactions on Network and Service Management*, 20(2):1378–1391, 2023.
- [135] Panpan Qi, Dan Li, and See-Kiong Ng. Mad-sgcN: Multivariate anomaly detection with self-learning graph convolutional networks. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1232–1244. IEEE, 2022.
- [136] Chen Qiu, Timo Pfroemer, Marius Kloft, Stephan Mandt, and Maja Rudolph. Neural transformation learning for deep anomaly detection beyond images. In *International Conference on Machine Learning*, pages 8703–8714. PMLR, 2021.
- [137] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 985–994, 2015.

- [138] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3009–3017, 2019.
- [139] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.
- [140] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.
- [141] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.
- [142] Lukas Ruff, Robert A Vandermeulen, Billy Joe Franks, Klaus-Robert Müller, and Marius Kloft. Rethinking assumptions in deep anomaly detection. In *Proceedings of the International Conference on Machine Learning Workshop*, 2021.
- [143] Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive learning of general-purpose audio representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3875–3879. IEEE, 2021.
- [144] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.
- [145] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for {nlp} with differentiable edge masking. In *International Conference on Learning Representations*, 2021.
- [146] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

- [147] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999.
- [148] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.
- [149] Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems*, 33:13016–13026, 2020.
- [150] Jiaxin Shi, Chen Liang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Hanwang Zhang. Deepchannel: Saliency estimation by contrastive learning for extractive document summarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6999–7006, 2019.
- [151] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.
- [152] Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep ehr: a survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604, 2017.
- [153] Hyeok-Ki Shin, Woomyo Lee, Jeong-Han Yun, and Byung-Gi Min. Two ics security datasets and anomaly detection contest on the hil-based augmented ics testbed. In *Cyber Security Experimentation and Test Workshop*, CSET ’21, page 36–40, New York, NY, USA, 2021. Association for Computing Machinery.
- [154] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE foundations and new directions of data mining workshop*, pages 172–179. IEEE Press, 2003.
- [155] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1067–1075, 2017.

- [156] Junshuai Song, Xiaoru Qu, Zehong Hu, Zhao Li, Jun Gao, and Ji Zhang. A subgraph-based knowledge reasoning method for collective fraud detection in e-commerce. *Neurocomputing*, 461:587–597, 2021.
- [157] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2828–2837, 2019.
- [158] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2131, 2023.
- [159] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.
- [160] Zhen Tan, Ruocheng Guo, Kaize Ding, and Huan Liu. Virtual node tuning for few-shot node classification. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2177–2188, 2023.
- [161] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 990–998, 2008.
- [162] Nusrat Tasnim and Joong-Hwan Baek. Dynamic edge convolutional neural network for skeleton-based human action recognition. *Sensors*, 23(2):778, 2023.
- [163] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.
- [164] Sheng Tian, Jihai Dong, Jintang Li, Wenlong Zhao, Xiaolong Xu, Bowen Song, Changhua Meng, Tianyi Zhang, Liang Chen, et al. Sad: Semi-supervised anomaly detection on dynamic graphs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2306–2314, 2023.
- [165] Zhiwen Tian, Ming Zhuo, Leyuan Liu, Junyi Chen, and Shijie Zhou. Anomaly detection using spatial and temporal information in multivariate time series. *Scientific Reports*, 13(1):4400, 2023.

- [166] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proceedings of VLDB*, 15(6):1201–1214, 2022.
- [167] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [168] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.
- [169] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *Proceedings of the International Conference on Learning Representations*, 2019.
- [170] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. A semi-supervised graph attentive network for financial fraud detection. In *Proceedings of the IEEE International Conference on Data Mining*, pages 598–607, 2019.
- [171] Jing Wang, Shikuan Shao, Yunfei Bai, Jiaoxue Deng, and Youfang Lin. Multiscale wavelet graph autoencoder for multivariate time-series anomaly detection. *IEEE Transactions on Instrumentation and Measurement*, 72:1–11, 2022.
- [172] Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X Gao, and Dazhong Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of manufacturing systems*, 48:144–156, 2018.
- [173] Qizhou Wang, Guansong Pang, Mahsa Salehi, Wray Buntine, and Christopher Leckie. Cross-domain graph anomaly detection via anomaly-aware contrastive alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4676–4684, 2023.
- [174] Rui Wang, Chongwei Liu, Xudong Mou, Kai Gao, Xiaohui Guo, Pin Liu, Tianyu Wo, and Xudong Liu. Deep contrastive one-class time series anomaly detection. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 694–702. SIAM, 2023.
- [175] Siqi Wang, Yijie Zeng, Xinwang Liu, En Zhu, Jianping Yin, Chuanfu Xu, and Marius Kloft. Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. *Advances in neural information processing systems*, 32, 2019.

- [176] Shiwei Wei, Yuping Wang, and Yiu-ming Cheung. A branch elimination-based efficient algorithm for large-scale multiple longest common subsequence problem. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2179–2192, 2023.
- [177] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the International Conference on Machine Learning*, pages 6861–6871. PMLR, 2019.
- [178] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.
- [179] Yuhang Wu, Mengting Gu, Lan Wang, Yusan Lin, Fei Wang, and Hao Yang. Event2graph: Event-driven bipartite graph for multivariate time-series anomaly detection. *arXiv preprint arXiv:2108.06783*, 2021.
- [180] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.
- [181] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 753–763, 2020.
- [182] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *Proceedings of the International Conference on Learning Representations*, 2020.
- [183] Fan Xu, Nan Wang, Xuezhi Wen, Meiqi Gao, Chaoqun Guo, and Xibin Zhao. Few-shot message-enhanced contrastive learning for graph anomaly detection. In *Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, pages 17–21, 2023.
- [184] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2022.
- [185] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*, 2019.

- [186] Xinghao Yang, Weifeng Liu, and Wei Liu. Tensor canonical correlation analysis networks for multi-view remote sensing scene recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(6):2948–2961, 2022.
- [187] Xinghao Yang, Weifeng Liu, Wei Liu, and Dacheng Tao. A survey on canonical correlation analysis. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2349–2368, 2021.
- [188] Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. Dcdetector: Dual attention contrastive representation learning for time series anomaly detection. In *Proceedings of 29th ACM SIGKDD International SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 3033–3045, 2023.
- [189] Asad Yaseen. The role of machine learning in network anomaly detection for cybersecurity. *Sage Science Review of Applied Machine Learning*, 6(8):16–34, 2023.
- [190] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, volume 34, pages 28877–28888, 2021.
- [191] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.
- [192] Aras Yurtman, Pieter Robberechts, Dany Vohl, Eric Ma, Gust Verbruggen, Marco Rossi, Mazhar Shaikh, Muhammad Yasirroni, Wojciech Zieliński, Toon Van Craenendonck, et al. wannesm/dtaidistance: v2. 3.5. *Zenodo*, 2021.
- [193] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.
- [194] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.
- [195] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate

- time series data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1409–1416, 2019.
- [196] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1409–1416, 2019.
  - [197] Jiaqiang Zhang, Senzhang Wang, and Songcan Chen. Reconstruction enhanced multi-view contrastive learning for anomaly detection on attributed networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2376–2382, 2022.
  - [198] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.
  - [199] Weiqi Zhang, Chen Zhang, and Fugee Tsung. Grelen: Multivariate time series anomaly detection from the perspective of graph relational learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2390–2397, 2022.
  - [200] Zheng Zhang and Liang Zhao. Unsupervised deep subgraph anomaly detection. In *Proceedings of the IEEE International Conference on Data Mining*, pages 753–762, 2022.
  - [201] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 841–850. IEEE, 2020.
  - [202] Li Zheng, Zhao Li, Jun Gao, Zhenpeng Li, Jia Wu, and Chuan Zhou. Domain adaptation for anomaly detection on heterogeneous graphs in e-commerce. In *European Conference on Information Retrieval*, pages 304–318. Springer, 2023.
  - [203] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. Addgraph: Anomaly detection in dynamic graph using attention-based temporal gc. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 4419–4425, 2019.

- [204] Yu Zheng, Ming Jin, Yixin Liu, Lianhua Chi, Khoa T Phan, and Yi-Ping Phoebe Chen. Generative and contrastive self-supervised learning for graph anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12220–12233, 2023.
- [205] Yu Zheng, Ming Jin, Yixin Liu, Lianhua Chi, Khoa T Phan, Shirui Pan, and Yi-Ping Phoebe Chen. From unsupervised to few-shot graph anomaly detection: A multi-scale contrastive learning approach. *arXiv preprint arXiv:2202.05525*, 2022.
- [206] Yu Zheng, Huan Yee Koh, Ming Jin, Lianhua Chi, Khoa T Phan, Shirui Pan, Yi-Ping Phoebe Chen, and Wei Xiang. Correlation-aware spatial-temporal graph learning for multivariate time-series anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
- [207] Hao Zhou, Ke Yu, Xuan Zhang, Guanlin Wu, and Anis Yazidi. Contrastive autoencoder for anomaly detection in multivariate time series. *Information Sciences*, 610:266–280, 2022.
- [208] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.
- [209] Wei Zhu, Weijian Li, E Ray Dorsey, and Jiebo Luo. Unsupervised anomaly detection by densely contrastive learning for time series data. *Neural Networks*, 168:450–458, 2023.
- [210] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 2021.
- [211] Yun Zhu, Yaoke Wang, Haizhou Shi, Zhenshuo Zhang, Dian Jiao, and Siliang Tang. Graphcontrol: Adding conditional control to universal graph pre-trained models for graph domain transfer learning. In *Proceedings of the ACM Web Conference*, 2024.
- [212] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.