A Comprehensive Method for Anomaly Detection in Complex Dynamic IoT Systems

Andrii Liashenko and Larysa Globa
Institute of Telecommunication Systems, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Beresteiska Avenue 37, 03056 Kyiv, Ukraine
andrey.lyashenko44@gmail.com, lgloba@its.kpi.ua

Keywords: Anomaly Detection, Temporal Graphs, Temporal Graph Neural Networks, Autoencoder, Graph Neural Networks, Reconstruction Error, Dynamic Systems, Transportation Networks.

Abstract: Modern dynamic systems, such as transportation networks and IoT infrastructures, generate massive volumes of interrelated temporal data represented as temporal graphs. Conventional methods – like clustering, statistical thresholds, and classical time series analysis – often fail to account for the spatial-temporal dependencies inherent in these systems, leading to high false positive rates or missed complex anomalies. In this paper, we propose a novel anomaly detection approach that combines Temporal Graph Neural Networks (TGNN) with Autoencoders. The method utilizes TGNN to extract robust node representations by capturing both local connectivity and temporal evolution, while an autoencoder is trained to reconstruct normal node behavior. Anomalies are subsequently identified through significant reconstruction errors, which serve as indicators of deviations from typical patterns. Experimental evaluations on the real-world PeMSD7 dataset demonstrate that the proposed TGNN + Autoencoder method improves detection accuracy by 17.33% compared to traditional methods, reduces false positives by 4.71%, and achieves a 6.02% higher F1-score relative to using TGNN or autoencoder individually. These results underline the practical relevance of our approach for real-time monitoring of transportation networks, while also contributing theoretically to the integration of spatial and temporal features in anomaly detection.

1 INTRODUCTION

Dynamic network systems such as transportation networks, the Internet of Things (IoT), financial markets, and cybersecurity generate vast amounts of interconnected temporal data. For example, according to Statista (Fig. 1), the number of IoT devices in the world is projected to reach 39.6 billion by 2033 [1].

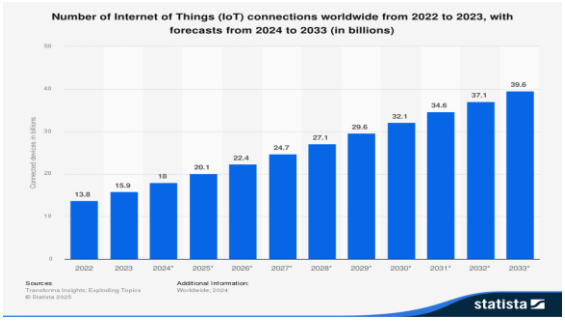

Figure 1: Number of IoT devices by 2033 [1].

Data generated by such devices form dynamic graph structures with nodes and edges that change over time. Anomalies in these networks may indicate transport congestion, IoT failures, financial fraud, or cyberattacks, making early detection critical to prevent accidents, inefficiencies, or financial losses. However, traditional methods like clustering, statistical thresholding, and machine learning have significant limitations – they often ignore the inherent graph structure needed to capture complex network dynamics [2], lack adaptability as approaches such as One-Class SVM or Isolation Forest require pre-training and fail to accommodate new patterns [3], and rely on fixed thresholds that lead to high false positive rates in dynamic environments [4]. Thus, new adaptive approaches that consider both temporal and structural dynamics of graphs are essential for more accurate anomaly detection.

The remainder of the paper is organized as follows. Section 2 describes the methodology, detailing Temporal Graph Neural Networks and Autoencoders for anomaly detection Section 3 reviews traditional methods for anomaly detection. Section 4 defines the problem addressed in this study. Section 5 details the proposed method of anomaly detection, which combines Temporal Graph Neural Networks with Autoencoders. Finally, Section 6 presents the experimental results.

2 METHODOLOGY

The use of Graph Neural Networks (GNN) has significantly improved the analysis of graph data, allowing the creation of deep representations of nodes and edges. However, standard GNNs do not take into account temporal dependencies. Therefore, Temporal Graph Neural Networks (TGNNs) were proposed for dynamic systems (Fig. 2), which take into account the evolution of graphs over time [5]. Autoencoders (AEs) have proven themselves well in anomaly problems, especially in unsupervised learning (Fig. 3). AEs learn to reconstruct data, and nodes that have a high reconstruction error are considered anomalous [6].

Node: In TGNN, a node is a dynamic entity (e.g., a traffic sensor) in a temporal graph, representing time-varying data. *Characteristic (Feature):* These are the measurable, time-dependent attributes of a node, such as sensor readings or traffic speed. *Latent Representation:* A compact, low-dimensional encoding of a node's features, capturing its essential spatial and temporal patterns.

In this paper, we propose a method that combines TGNN with an AE for anomaly detection in temporal graphs. The approach creates latent representations of nodes using a TGNN trained to predict their states over time, then trains an autoencoder to reconstruct normal node states, with anomalies identified by high reconstruction error.

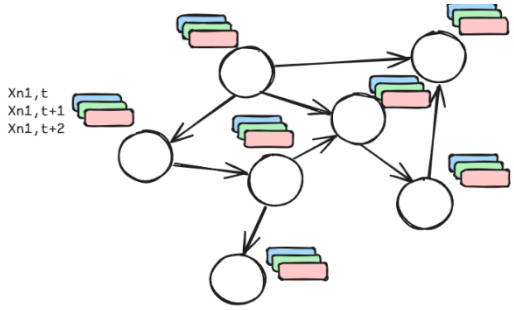

Figure 2: Timestamp graph.

TGNNs extend classical GNNs to process dynamic graphs by incorporating a temporal component. Unlike traditional GNNs, which model static node relationships, TGNNs update node states at each time step based on both neighboring nodes and historical information, often using recurrent mechanisms such as Gated Recurrent Units (GRU) or Long Short-Term Memory (LSTM) [7, 8].

The classical formula for updating the representation of a node in GNN is as follows:

$$h_i^{(t)} = \sigma\left(W \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(t)} h_j^{(t-1)}\right)$$

where:

- $h_i^{(t)}$ – d-dimensional state vector of node i at time t, where each element represents a specific feature (e.g., speed, traffic, etc.).,
- *N(i)* – the set of neighbors of node i,
- $\alpha_{ij}^{(t)}$ – the attention weight between nodes i and j
- W – a trainable weight matrix that performs a linear transformation (projection) of the input node features,
- σ – the activation function (for example, ReLU or Sigmoid).

At each time step t, the state vector $h_i^{(t)}$ of node i is computed by aggregating the state vectors of its neighbors *N(i)*, weighted by the attention coefficients $\alpha_{ij}^{(t)}$ – determine the influence of each neighboring node j on node i. They are typically computed using a small neural network to evaluate the "importance" of each neighbor, and then normalized (e.g., via softmax) so that the contributions sum to one. This mechanism enables the model to focus on the most relevant neighbors when aggregating information, effectively capturing complex spatial-temporal relationships.

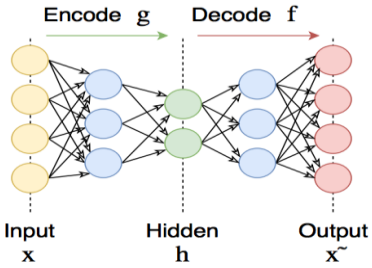

Figure 3: Autoencoder architecture.

The encoder transforms the input data x_t into a compact representation [9] h_t

$$h_t = g(x_t)$$

where:

- x_t – the input data at time t,
- h_t – the hidden feature vector or compressed representation of the data in the hidden space,
- g – the encoder function, which typically includes several layers of neural networks (for example, linear transformations or activation functions).

The decoder reconstructs the input data $\hat{x}_t$ of the hidden representation h_t, trying to recreate the original input data:

$$\hat{x}_t = f(h_t)$$

where:

- $\hat{x}_t$ – the reconstructed output data for time t,
- f – the decoder function, which typically includes several layers of neural networks to reconstruct the output data from the compact representation.

This transformation allows the autoencoder to reduce the dimensionality of the input data, preserving only the most important information about its structure and temporal dynamics.

To train the autoencoder, we use the loss function (Mean Square Error (MSE)) $L(x_t, \hat{x}_t) = \|x_t - \hat{x}_t\|$.

When working on new data, if the model is unable to accurately reconstruct the input data, this may indicate anomalous changes in the graph. Therefore, an anomaly is defined as a large reconstruction error:

$$\Delta_t = \|x_t - \hat{x}_t\|$$

The proposedmethod was tested on PeMSD7 - a real dataset of transport networks [10].

3 RELATED WORK

Traditional time series techniques–such as ARIMA, Holt-Winters Exponential Smoothing, and Hidden Markov Models–model node changes over time and predict future graph states. Yet, they require transforming graph data into independent time series, resulting in lost structural information, high data requirements, and sensitivity to model parameters [12].

Clustering and machine learning methods (e.g., K-Means, DBSCAN, Isolation Forest, One-Class SVM) also have been used to detect anomalies by grouping nodes or identifying outliers. Their main drawbacks are the need to manually set parameters and the lack of temporal and relational context, limiting their effectiveness in complex graphs [13].

In contrast, modern approaches leverage deep neural networks, particularly GNNs and TGNNs, which capture both spatial and temporal dependencies [14]. This work proposes a combination of TGNN with AEs: the TGNN models local and temporal connections, while the AE learns to reconstruct normal node patterns–nodes with high reconstruction error are marked as anomalous [15, 16].

This integrated approach significantly improves anomaly detection accuracy by reducing false positives and enhancing adaptability to new data.

Table 1: Comparison of anomaly detection methods.

Method	Description	Advantages	Disadvantages
Statistical Methods (Z-score, Grubbs' Test)	An anomaly is defined as a deviation from the mean. Nodes or edges that significantly differ from the norm are labeled as anomalous.	Simplicity, computational efficiency	Does not consider graph structure, sensitive to noise
Time Series Methods (ARIMA)	Utilize time-series models to predict node behavior. Anomalies are flagged when the actual value substantially deviates from the predicted value.	Perform well with periodic trends	Do not account for structural changes, sensitive to parameters
Clustering Methods (K-Means, DBSCAN)	Nodes are clustered based on similarity, and those that lie far from the main groups are considered anomalous.	Effective for group anomalies	Require manual selection of cluster count, do not incorporate temporal data
Machine Learning Methods (Isolation Forest, One-Class SVM)	Models are trained on normal data and classify any deviation as anomalous (one-class learning).	Can operate without extensive manual parameter tuning.	Unsuitable for dynamic graphs without adaptation
GNN (Graph Neural Networks)	Graph Neural Networks analyze structural relationships among nodes but do not account for temporal dynamics.	Capture inter-node connections	Do not model the time-evolving behavior of nodes

According to the Table 1, we can conclude that traditional methods are insufficient for analyzing dynamic graphs because they do not take into account the graph structure and the temporal dynamics of changes. This limits their ability to effectively detect anomalies in complex networks where both spatial and temporal relationships are important.

4 PROBLEM DEFINITION

The paper addresses the challenge of detecting anomalies in temporal graphs representing dynamic systems (e.g., transportation networks) where traditional methods fail to capture the intricate spatial-temporal dependencies, leading to high false positives. The goal is to develop a TGNN-based approach combined with Autoencoders to accurately model node behavior over time and identify deviations via reconstruction errors.

5 PROPOSED COMPLEX METHOD OF ANOMALY DETECTION

Before detailing our method, we define the research problem: given a temporal graph $G = (V, E, X, T)$ – where V are nodes (e.g., traffic sensors), E are time-varying spatial connections, X are time-dependent features (e.g., speed measurements every 5 minutes), and T comprises historical data reflecting normal node behavior–the task is to identify nodes that deviate significantly from these expected patterns, by leveraging both local spatial relationships and long-term temporal dependencies.

This task involves analyzing the dynamic evolution of each node's state, which is influenced not only by its own historical data but also by the states of its immediate neighbors, as encoded by the adjacency matrix and attention coefficients. The challenge lies in integrating these local interactions with the long-term trends present in the node features over extended periods. To address this, the proposed method (Fig. 4) employs a TGNN to jointly capture spatial and temporal information, along with an AE to learn compact latent representations. Anomalies are then detected by identifying significant deviations in the reconstruction error, which signal abnormal behavior.

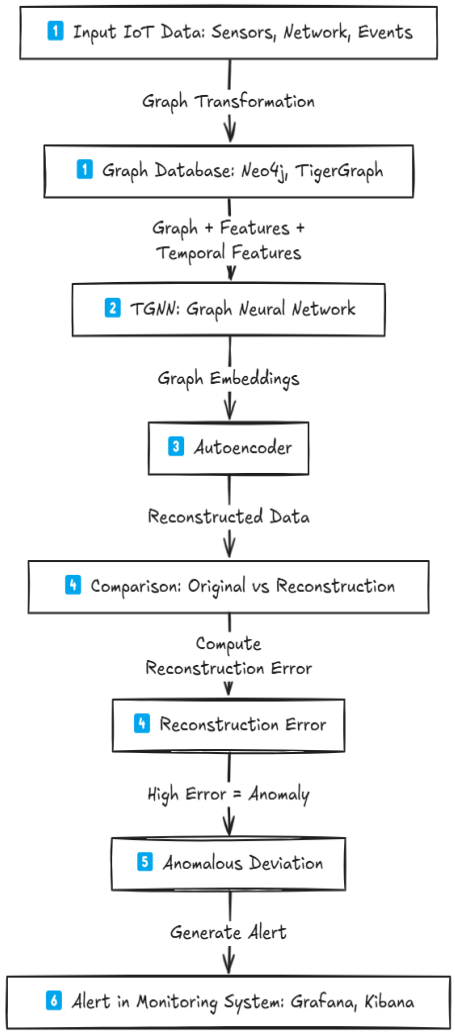

Figure 4: Proposed complex method.

This proposed IoT anomaly detection system operates through six key stages:

1) Incoming IoT data (sensors, network events) are converted into graph structures and transferred to the Graph Database.
2) TGNN takes as input a graph, node characteristics and timestamp history. It creates graph embeddings to represent relationships in the data.
3) Autoencoder receives these embeddings, tries to reconstruct the original data.
4) Comparison of the original and reconstructed data → reconstruction error is calculated.
5) If the error is high → anomaly is detected.
6) Anomaly deviation triggers an alert in the Monitoring System (Grafana, Kibana).

We designed our TGNN architecture using a three-layer A3TGCN model combined with a single-layer GRU to efficiently capture both spatial and temporal dependencies in dynamic IoT networks.

The model consists of three graph convolutional layers with 64 hidden units each, using ReLU activation and an attention-based aggregation mechanism to refine node representations by weighting the influence of neighboring nodes. The three-layer A3TGCN model was selected to ensure that nodes incorporate information from both direct neighbors and second-order connections, effectively modeling localized interactions in dynamic graphs. Increasing the depth beyond three layers resulted in diminishing improvements while increasing computational costs, making deeper architectures inefficient for real-time IoT applications. A shallower model, on the other hand, did not provide sufficient contextual information for anomaly detection. The use of attention-based aggregation further refines node representations, ensuring that structurally important nodes have a greater impact on anomaly detection.

To model temporal dependencies, we integrate a single-layer GRU instead of more complex recurrent architectures like LSTM. GRUs provide a faster and more memory-efficient alternative to LSTMs while maintaining comparable performance in capturing long-term dependencies. Unlike simple RNNs, GRUs effectively retain relevant historical information while dynamically controlling memory updates, making them well-suited for large-scale, continuously evolving IoT networks. The combination of TGNN for spatial learning and GRU for sequential modeling allows the system to distinguish between normal fluctuations and true anomalies over time.

Finally, a fully connected output layer maps the learned node embeddings to a single scalar value per node, indicating its predicted state. The model is trained using the Mean Squared Error (MSE) loss function, which evaluates the reconstruction error between predicted and actual states.

The encoder network consists of three fully connected layers with 128, 64, and 32 neurons, each followed by a LeakyReLU activation function to introduce non-linearity while preserving small gradient updates for low-activation values. The final layer of the encoder maps the data into a latent space of 16 dimensions, providing a compressed representation of node embeddings while maintaining key structural and temporal information.

The decoder network mirrors the encoder, consisting of three fully connected layers (32, 64, 128 neurons), using LeakyReLU activation in the hidden layers and a linear activation in the final layer to reconstruct the original input. This symmetrical structure ensures effective reconstruction while preserving node-specific features.

To improve generalization and prevent overfitting, we apply dropout (0.2 probability) and batch normalization after each hidden layer. The AE is trained using MSE loss, which quantifies the difference between reconstructed and actual node embeddings, helping the model learn normal patterns in graph data.

Correctly setting the threshold τ is crucial: too low increases false positives, while too high leads to false negatives. In this work, we adopt a quantile method that automatically adapts τ to the reconstruction error distribution by setting it at the 95th percentile.

$$\tau = Q_{95},$$

where Q_{95} is the value above which 5% of nodes with the largest reconstruction error are located. This approach allows for dynamic detection of anomalies, reducing the risk of false positive detections, since the model adapts to changes in the graph structure and the dynamics of its nodes.

The 95th percentile is chosen empirically because it reduces false positives by avoiding the misclassification of normal nodes, provides noise immunity by adapting to the specific error distribution rather than relying on a manually set threshold, and effectively captures rare events by identifying the top 5% of the most deviant nodes. To validate this choice, we performed a threshold sensitivity analysis, summarized in the table above. Selecting the 90th percentile increases recall to 0.88 but significantly raises false positives due to lower precision. Conversely, the 98th percentile improves precision to 0.91 and reduces false positives, but recall drops sharply to 0.74, causing many anomalies to be missed. The 95th percentile offers the best balance between precision and recall with an F1-score of 0.88, ensuring both effective anomaly detection and a manageable false positive rate.

Unlike fixed-threshold methods, the quantile approach adapts automatically to the graph's state: in stable conditions, the error distribution is narrow and τ is low, while significant changes adjust τ to new data. Our method captures the entire process in one formula: the state vector of node i at time t is computed by aggregating neighbor states (weighted by attention coefficients), transforming them via a trainable weight matrix and activation function, and then passing the result through an encoder-decoder to

reconstruct the original state. This entire process is encapsulated in the following expression:

$$\hat{h}_t(t) = f_{DEC}\left(f_{ENC}\left(\sigma\left(W\sum_{j\in N(i)}\alpha_{ij}(t)h_j(t-1)+b\right)\right)\right)$$

In this formulation, $h_i(t)$ denotes the state vector of node i at time t, $\alpha_{ij}(t)$ represents the attention coefficient reflecting the influence of neighbor j on node i, and $W$is a trainable weight matrix that projects the aggregated features into a new space before applying the non-linear activation σ. The functions f_{ENC} та f_{DEC} correspond to the encoder and decoder of the autoencoder, respectively, which learn a compact latent representation and reconstruct the original signal.

$$Anomaly(i,t) = (\left|\left|h_i(t) - \hat{h}_i(t)\right|\right|^2 > \tau).$$

The anomaly indicator is then determined by comparing the reconstruction error $\left|\left|h_i(t) - \hat{h}_i(t)\right|\right|^2$ a threshold τ; if the error exceeds τ, the node is flagged as anomalous. This compact representation captures the entire method's essence, seamlessly integrating spatial and temporal dynamics for effective anomaly detection.

Variables and notation:

- $h_i(t)$ – is the node representation from the TGNN at time t;
- f_{ENC} and f_{DEC} – are the encoder and decoder functions of the autoencoder;
- σ – is an activation function (e.g., ReLU);
- W and b trainable parameters;
- $\alpha_{ij}(t)$ – represents the attention weight between node i and its neighbor j;
- τ – is the threshold for anomaly detection.

Thus, the proposed approach allows not only to train high-quality representations of nodes in temporal graphs, but also to effectively identify anomalous nodes using the reconstruction error. The use of the quantile threshold selection method ensures the adaptability of the model, which allows avoiding problems associated with excessive sensitivity to noise in the data. The proposed method combines the advantages of TGNN in training representations with the advantages of the autoencoder in detecting deviations, which makes it an effective tool for analyzing anomalies in dynamic graphs.

6 EXPERIMENT

This section evaluates the proposed TGNN+AE method for anomaly detection on the PeMSD7 dataset, which contains temporal graphs of a transport network (nodes = road sensors, edges = spatial connections). The goal is to detect traffic anomalies (e.g., accidents, congestion) by analyzing reconstruction errors and comparing results with traditional methods (Isolation Forest, One-Class SVM, K-Means). Traffic speeds were normalized using Z-score, and sequences of 24 previous values were used to predict the current state. The TGNN extracts spatiotemporal features, while the autoencoder compresses these into a latent space, with the 95th percentile of the reconstruction error used as the anomaly threshold. The experimental results show that the TGNN+AE method improves detection accuracy by 17.33%, reduces false positives by 4.71%, and increases the F1-score by about 6% compared to using each method separately.

Table 2: Comparison of results.

Method	Precision	Recall	F1-score
Isolation Forest	0.72	0.68	0.70
One-Class SVM	0.76	0.64	0.69
K-Means	0.78	0.72	0.75
Autoencoder (AE)	0.81	0.76	0.79
TGNN	0.85	0.81	0.83
TGNN + AE	0.89	0.85	0.88

From Table 2, it can be seen that TGNN + AE improves the anomaly detection accuracy by 10.67% compared to the best traditional method (K-Means):

$$Percentage\ Increase = \frac{F1_{TGNN+AE} - max\,(F1_{Traditional})}{max\,(F1_{Traditional})} \times 100 = \frac{0.88 - 0.75}{0.75} = 17.33\%.$$

TGNN + AE provides 6.02% higher F1-measure compared to using Autoencoder or TGNN alone:

$$Percentage\ Increase = \frac{F1_{TGNN+AE} - max\,(F1_{AE}, F1_{TGNN})}{max\,(F1_{AE}, F1_{TGNN})} \times 100 = \frac{0.88 - 0.83}{0.83} = 6.02\%.$$

TGNN + AE reduces the number of false positives by 4.71% compared to TGNN due to improved Precision:

$$False\ Positive\ Reduction = \frac{Precision_{TGNN+AE} - Precision_{TGNN}}{Precision_{TGNN}} \times 100 = \frac{0.89 - 0.85}{0.89} = 4.71\%.$$

To assess the efficiency, the distribution of the reconstruction error and the definition of the 95th percentile as the anomaly threshold were used. The graph (Fig. 5) shows how the threshold and the error distribution were calculated:

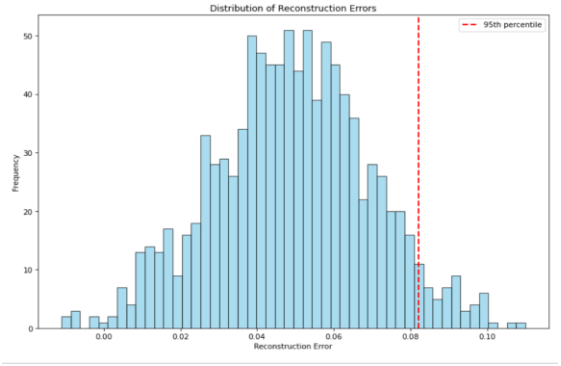

Figure 5: Distribution of reconstruction error.

This graph shows the distribution of the reconstruction error and the anomaly threshold (red dashed line), which is determined by the 95th percentile. Nodes with a reconstruction error greater than this threshold are classified as anomalous. Comparative analysis proved that the proposed complex method outperforms traditional approaches, providing higher accuracy and lower number of false positive detections.

7 CONCLUSIONS

This paper proposes a comprehensive method for anomaly detection in temporal graphs based on a combination of TGNN and a graph AE.

The TGNN captures spatiotemporal relationships, while the AE learns hidden node representations and measures deviations through reconstruction error. Experimental results show that the TGNN+AE approach improves detection accuracy by 17.3%, reduces false positives by 4.71%, and increases the F1-measure by 6.02% compared to using each method alone. A key element is setting the anomaly threshold at the 95th percentile of the reconstruction error, which adaptively identifies anomalous nodes without manual tuning, thereby enhancing stability. Overall, this integrated method effectively predicts node dynamics and identifies anomalies, paving the way for further research into adaptive anomaly detection in complex dynamic systems.

REFERENCES

[1] Statista, "Number of IoT-connected devices worldwide 2024–2033," [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/, [Accessed: Apr. 2024].

[2] Y. Wu, H. N. Dai, and H. Tang, "Graph neural networks for anomaly detection in industrial IoT," IEEE Internet of Things Journal, vol. 8, no. 3, pp. 212-223, Mar. 2021.

[3] J. Yang and Z. Yue, "Learning hierarchical spatial-temporal graph representations for anomaly detection," IEEE Transactions on Knowledge and Data Engineering, early access, 2022.

[4] Y. Feng and J. Chen, "Full graph autoencoder for one-class group anomaly detection," IEEE Internet of Things Journal, vol. 9, no. 5, pp. 4021-4030, May 2022.

[5] Y. Lai, Y. Zhu, and L. Li, "STGLR: A spacecraft anomaly detection method using spatio-temporal graph learning," Sensors, vol. 25, no. 4, pp. 1123-1137, 2025.

[6] T. N. Kipf and M. Welling, "Variational Graph Autoencoders," 2016.

[7] J. Wan, L. Cao, J. Bai, and J. Li, "Learning multiple types of features on a hybrid neural network for blockchain transaction behavior detection," SSRN, 2024.

[8] M. W. Asres, C. W. Omlin, L. Wang, P. Parygin, and D. Yu, "Data quality monitoring through transfer learning on anomaly detection for the Hadron calorimeters," 2024.

[9] J. Liu, X. Han, and X. Shang, "Spatial-Temporal Memories Enhanced Graph Autoencoder for Anomaly Detection in Dynamic Graphs," 2024.

[10] PEMS-D7 Dataset, [Online]. Available: https://paperswithcode.com/dataset/pemsd7, [Accessed: Apr. 2024].

[11] J. Qi, C. Zeng, Z. Luan, S. Huang, S. Yang, and Y. Lu, "Beyond window-based detection: A graph-centric framework for discrete log anomaly detection," preprint, 2025.

[12] M. Wen, Z. H. Chen, Y. Xiong, and Y. C. Zhang, "LGAT: A novel model for multivariate time series anomaly detection with improved anomaly transformer and learning graph structures," Neurocomputing, vol. 554, pp. 126-138, 2025.

[13] R. Hosseini, F. Simini, V. Vishwanath et al., "A Deep Probabilistic Framework for Continuous Time Dynamic Graph Generation," 2024.

[14] F. Xiao, S. Chen, Y. Ma et al., "SENTINEL: Insider threat detection based on multi-timescale user behavior interaction graph learning," IEEE Transactions on Information Forensics and Security, vol. 19, pp. 432-445, 2024.

[15] J. Li, X. Deng, and B. Yao, "Enhanced anomaly detection of industrial control systems via graph-driven spatio-temporal adversarial deep support vector data description," Expert Systems with Applications, vol. 224, p. 119899, 2025.

[16] J. Cao, X. Di, J. Li, K. Yu, and L. Zhao, "IoVST: An anomaly detection method for IoV based on spatiotemporal feature fusion," Future Generation Computer Systems, vol. 145, pp. 409-421, 2025.