Partially-Supervised Graph Derivation Network With Meta-Learning for Time-Series Anomaly Detection

Sanli Zhu, Yuan Li, Kang Xu, and Junjun Xu, *Member, IEEE*

***Abstract*—Time-series anomaly detection is essential in various fields, such as industrial monitoring, cybersecurity, and finance. Traditional supervised methods often face challenges due to the limited availability of labeled anomaly instances for training. Moreover, these methods struggle to deal with intricate systems that incorporate information about topological structures. In this article, we propose a novel approach called the partially-supervised graph derivation network with meta learning (PS-GDNML) for time-series anomaly detection. PS-GDNML combines the power of graph-based representations, partially-supervised learning, and meta-learning to enhance the effectiveness and robustness of anomaly detection. The method represents time-series data as a graph, where each data point is a node, and temporal relationships are captured through edges. By leveraging a graph attention neural network (GAT), the model effectively captures complex dependencies and relationships within the data. To address the scarcity of labeled anomaly instances, PS-GDNML adopts a partially-supervised learning framework. It utilizes both labeled and unlabeled data, enabling the model to learn from the available information and generalize to detect anomalies in unseen data. Additionally, to explore the underlying commonalities of data across different time periods and enhance the model's adaptability, we adopted a new meta-learning method called task relation meta-learner (TRMLearner). The purpose of this project is to utilize task relationships to guide the meta-learning optimization process. We evaluated the performance of PS-GDNML on benchmark datasets and compared it with state-of-the-art anomaly detection methods. The experimental results demonstrate that, even with a restricted set of labeled instances, our method excels at accurately detecting anomalies. Furthermore, the meta-learning component enhances the model's capacity to generalize to novel and evolving anomaly patterns.**

***Index Terms*—Anomaly detection, derivation network, graph neural network (GNN), meta learning, time series.**

I. Introduction

THE RAPID expansion of connected sensors and devices in cyber–physical systems (CPSs), such as data centers, industrial systems, and automobiles, has resulted in the continuous generation of large-scale multivariate time-series data. These time series reflect the operational conditions of CPSs and are crucial for detecting anomalies in real time, as illustrated in Fig. 1. Detecting anomalies within these time series [1] is crucial for ensuring system reliability and preventing failures. Since historical data often lacks labeled anomalies and anomalies exhibit highly variable characteristics, unsupervised learning is commonly used for anomaly detection. Unsupervised methods, such as principal component analysis (PCA) [2], clustering, and SVMs [3], detect anomalies by setting thresholds based on historical data, while forecasting-based approaches like AFMF [4] and MTAD [5] predict normal patterns to identify deviations. However, these methods struggle in dynamic environments, are sensitive to data shifts, and fail to leverage limited labeled anomalies, highlighting the need for more adaptable solutions.

As deep learning has advanced in the area of time-series anomaly detection, neural networks exhibit excellent learning capabilities and adaptability to complex network systems. For instance, sequence models that can capture long-term dependencies in time-series data include recurrent neural networks (RNNs) and long short-term memory networks (LSTM) [6]. Furthermore, deep learning models can enhance robustness to anomalous samples through adversarial training, such as disentangled variational autoencoder (DA-VAE) [7] and generative adversarial networks (GANs) [8], which utilize reconstruction errors as anomaly scores. However, these approaches overlook the internal structure relationships among data, and in many practical events, there often exist complex dependencies and relationships between time-series data. This makes these models less effective when dealing with complex systems that involve topological structure information.

Graph neural networks (GNNs) have emerged as a powerful approach for multivariate time-series anomaly detection due to their ability to model complex dependencies and relationships within structured data [9]. By representing multivariate time series as a graph, GNNs enable a more comprehensive understanding of the interactions between different data sources [10]. Graph convolutional networks (GCNs) [11] aggregate information from neighboring nodes to improve feature representations, while graph attention networks

Received 17 February 2025; revised 1 April 2025; accepted 2 April 2025. Date of publication 7 April 2025; date of current version 27 June 2025. This work was supported in part by the Defense Industrial Technology Development Program under Grant JCKY2023130C020. *(Corresponding author: Kang Xu.)*

Sanli Zhu and Junjun Xu are with the College of Automation and Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: zsl@njupt.edu.cn; jjxu@njupt.edu.cn).

Yuan Li and Kang Xu are with the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China, and also with the Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: liyuan1@njust.edu.cn; kxu@njupt.edu.cn).

Digital Object Identifier 10.1109/JIOT.2025.3558273

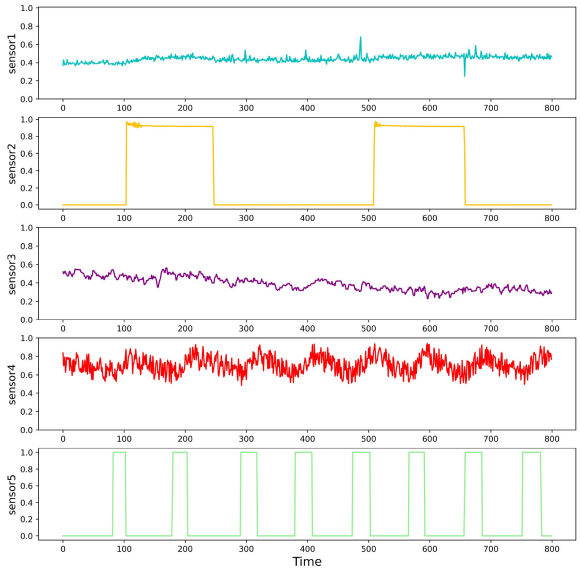


Fig. 1. In the industrial sector, the time series enerated by IoT sensors represent their individual characteristics. These multivariate time-series datasets can be utilized for anomaly detection.

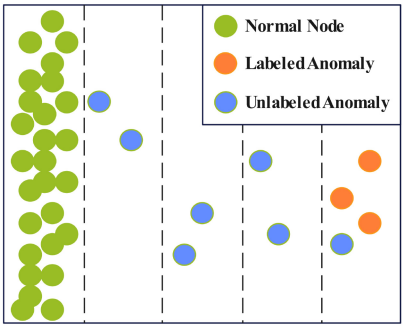


Fig. 2. Anomalies can be well distinguished in an anomaly score space by leveraging the significant differences between abnormal and normal data.

(GATs) [12] enhance this process by learning adaptive attention weights based on node similarities. Furthermore, boosting GNNs (BGNNs) [13] incorporate ensemble learning techniques to improve the robustness of GNN-based anomaly detection models. Despite these advancements, conventional GNNs remain highly dependent on hyperparameter tuning, including the number of graph convolution layers and neighbor selection strategies. More critically, they require a large amount of labeled data to perform effectively. In cases where labeled anomalies are scarce, GNNs struggle to generalize well and may fail to capture rare or subtle anomaly patterns.

To address these challenges, we propose the partially-supervised graph derivation network with meta-learning (PS-GDNML), which integrates graph-based learning, partially-supervised learning, and meta-learning into a unified framework for anomaly detection in multivariate time-series data. At its core, PS-GDNML builds upon the partially-supervised graph deviation network (PS-GDN) to reduce reliance on extensive labeled data. Unlike conventional unsupervised anomaly detection methods, PS-GDN leverages few annotated anomalies as prior knowledge to guide learning. By incorporating deviation loss [14] alongside forecasting-based loss, PS-GDN transforms the traditionally unsupervised anomaly detection task into a partially-supervised learning problem. Deviation loss assumes that the anomaly scores of normal samples follow a prior Gaussian distribution and uses this distribution to define a reference score. During training, normal samples are encouraged to maintain scores close to this reference, while anomalies are expected to deviate significantly. This process enables the model to generalize from few labeled anomalies while effectively distinguishing between normal and anomalous patterns, as illustrated in Fig. 2.

While PS-GDN mitigates the dependency on labeled data, it does not fully address the adaptability and generalization challenges inherent in anomaly detection across different systems. To further enhance PS-GDNML's performance, we introduce task relation meta-learner (TRMLearner) [15], which leverages meta-learning to improve the model's ability to adapt to diverse anomaly detection tasks. Traditional meta-learning approaches often assume that tasks are independent, failing to consider the underlying relationships between different anomaly detection scenarios. However, in real-world settings, anomaly detection tasks often exhibit similarities, such as shared temporal patterns or common structural dependencies. TRMLearner captures these relationships by constructing a task relation matrix based on extracted meta-data and uses this matrix to enforce relation-aware consistency regularization during meta-validation optimization. By leveraging intertask similarities, TRMLearner enables more effective knowledge transfer between tasks, reducing the risk of overfitting to specific datasets while improving overall model generalization.

By integrating graph-based learning, partially-supervised learning, and meta-learning, PS-GDNML provides a robust and scalable anomaly detection framework. GNNs allow the model to capture intricate structural dependencies within time-series data, while the partially-supervised learning approach enables effective detection with minimal labeled anomalies. Furthermore, TRMLearner enhances the adaptability of the model across different anomaly detection tasks, ensuring better generalization in dynamic and complex environments. This holistic approach allows PS-GDNML to outperform conventional methods, making it particularly well-suited for real-world applications where labeled anomalies are scarce, system behaviors are dynamic, and multivariate dependencies are critical for anomaly detection.

In summary, our main contributions can be outlined in three aspects.

1) We propose a PS-GDN anomaly detection method, aiming to reduce reliance on labeled data while leveraging the available labels.
2) We introduce TRMLearner into PS-GDN, utilizing task relationships to enhance generalization and adaptability across diverse different time periods.
3) Comprehensive tests on five publicly accessible datasets show that our model performs better than all existing state-of-the-art techniques, substantiating the effectiveness of our proposed method.

II. Related Work

This section reviews the related work in terms of 1) anomaly detection; 2) GNNs; and 3) meta-learning.

A. Anomaly Detection

The purpose of anomaly detection is to identify data points that deviate from the normal values. Anomaly detection can be implemented using various methods, and it has primarily evolved through three stages: 1) statistical-based methods; 2) machine learning methods; and 3) deep learning methods.

Statistical-Based Methods: Statistical-based anomaly detection methods model the fundamental statistical characteristics of data, such as mean, standard deviation, and distribution, to identify data points deviating from normal behavior. These include simple approaches like mean and standard deviation [16], Z-scores [17], median absolute deviation (MAD) [18], and many others. While these methods are straightforward to understand and implement, their strict assumptions about data distribution may not perform well with non-normal distributions or high-dimensional data. Therefore, in practical applications, they are often combined with more sophisticated techniques to enhance robustness.

Machine Learning Methods: With the advancement of machine learning techniques, research in anomaly detection shifted toward supervised learning, unsupervised learning, and semi-supervised learning methods. Unsupervised learning techniques like K-nearest neighbor [19], K-means [20], local outlier factor (LOF) [21], and Isolation Forest [22] have been extensively utilized to identify patterns in the data without labeled examples in order to detect anomalies. However, these methods may struggle to adapt to novel, previously unseen anomaly patterns during training. In such cases, the model may fail to effectively capture and identify these new types of anomalies.

Deep Learning Methods: Deep learning anomaly detection methods encompass various techniques, including reconstruction-based methods sunch as autoencoders (AEs) [23], DA-VAEs [7], and GANs [24]. While GANs can learn the distribution of the real data and perform better, variational autoencoder (VAE) can only learn the mean and variance from the real data. Predicted-based methods are equally vital in deep learning anomaly detection. The intricate correlations observed in time series can be better fitted by traditional deep learning techniques like RNN and LSTM [6], but serial computation may be constrained by long-term dependencies, while recursive operations exhibit less efficient learning patterns. Transformer-based approaches [25] are effective for capturing contextual dependencies in time-series data, thanks to their multihead attention mechanism that allows for parallel processing and addresses the limitations of recursive neural networks. Nevertheless, Transformers struggle to completely take use of interdependence across time series.

B. Graph Neural Networks

GNNs effectively transform graph information into low-dimensional representations, addressing challenges faced by traditional neural networks in handling unstructured data. GNNs model complex relationships between data points using the inherent graph structure. GCNs [26], a representative GNN, perform convolutional computations on related neighbors. Building upon GCN, GATs [12] use attention mechanisms to provide nodes with different weights, learning hidden representations. Additionally, graph isomorphism network (GIN) [27] iteratively aggregates the neighborhood information of nodes to generate node representations. Its flexibility and expressive power contribute to its outstanding performance on graph-structured data. However, GNNs are limited to the input of graph-structured data, often obtained through training data and initially unknown, exhibiting suboptimal performance on datasets with atypical graph topologies. STG [28] uses GCNs to encode the geographic information and manually constructs the graph structure data. In datasets where the network topology is unclear, this approach may not be feasible. GDN [29] integrates structural learning methods with GNNs, utilizing attention weights to capture intricate relationships among sensors and detect and interpret deviations from these relationships. Moreover, GATAMAF [30] leverages graph attention networks to extract temporal features and utilizes masked autoregressive flow for density estimation, treating the resulting values as anomaly scores to detect anomalies. However, these methods face challenges in effectively handling the temporal dynamics of graphs, particularly in situations where the evolution and changes of nodes over time in a time series need to be considered.

C. Meta-Learning

Meta-learning is an effective approach to extract and transfer knowledge, enabling models to be trained more accurately and rapidly with fewer examples. Ravi and Larochelle [31] introduced an LSTM-based meta-learning model that learns suitable parameter updates for scenarios involving a certain number of updates, while also capturing a generic initialization for the learner network to achieve fast convergence during training. Classical meta-learning model MAML [32] achieves strong generalization by explicitly training parameters, making it compatible with any model trained via gradient descent. In addition to learning learner initialization in a single meta-learning process, Meta-stochastic gradient descent (SGD) [33] can learn learner update directions and learning rates. Furthermore, Meta-EGN [34] utilizes meta-learning methods to find optimal initializations for future problems, addressing the combinatorial optimization problem in an unsupervised learning framework. Recently, Pu et al. [35] integrates Bayesian meta-learning with a model-agnostic framework, allowing pretrained models to swiftly adjust to new tasks by learning optimal parameters and hyperparameters. But these methods assume task independence, neglecting the potential to exploit the underlying similarities between tasks.

Recently, researchers have explored integrating GNNs with meta-learning techniques. PA-GNN [36] utilizes meta-optimization algorithms, training the model with adversarial counterparts and clean graphs to enhance its robustness on

the target network. GPN [37] aims to learn a transferable metric space, predicting node labels by finding the nearest class prototypes. Additionally, MD-Gram [38] introduces a strategy for transforming multidomain graph data, which aligns learning tasks from various source graphs with distinct feature spaces into a unified domain to learn generalized knowledge. Effective meta-learning methods require learning a universal representation of tasks for rapid generalization on new tasks. However, how to share and capture common features across different tasks remains a challenging problem.

III. Proposed Approach

A. Problem Statement

Our research focuses on the task of anomaly detection in multivariate time series. During the training phase, we split the time series into N segments, forming N source domains denoted as $\mathbf{D} = \{D_1, \ldots, D_N\}$. Each $D_i \in D$ includes the corresponding time-series data $\mathbf{X}_i = [x_1, \ldots, x_l]$ and corresponding labels $\mathbf{Y}_i = [y_1, \ldots, y_l]$, where l represents the length of the time series. For each time point t, $\mathbf{x}_t = [x_t^1, \ldots, x_t^n]$ is an n-dimensional vector, representing real-time data from n sensors. During the evaluation phase, we consider the entire test dataset as D_{new} and provide few labels to assess the model's performance. Our approach ultimately yields a collection of binary labels Y_{new} that indicate whether each test time point is anomalous, i.e., $y_t \in \{0, 1\}$, where 1 denotes that time t is flagged as anomalous.

In summary, our goal is to enhance anomaly detection by rapidly adapting the model to target time series based on the knowledge gained from the source domains' training. The meta-learning component of the GNN ought to compile the information gained from various time segments in D, producing a more robust representation. This, in turn, aims to achieve optimal performance on D_{new} with minimal real-time data. Meanwhile, the deviation network should exploit available labels to distinguish anomalous data as much as possible.

B. Overview

Fig. 3 illustrates the comprehensive architecture of PS-GDNML, demarcating the training and evaluation phases. Multiple sensor time series are employed as inputs to a GNN. During the training phase, D_i is randomly partitioned into the meta-training set D_i^s and the meta-validation set D_i^q. D_i^s is used to learn the temporary model $f_{\theta_i'}$ and D_i^q is used to update the base model f_θ. During meta-training phase, the graph attention mechanism is utilized to amalgamate node neighbor information, including time-series features and node embedding for predicting time series in the subsequent period. Meanwhile, we calculate the relation matrix based on the tasks in set D_i^s. After obtaining the model's prediction, apart from figuring out the mean squared error between the actual and anticipated values, we introduce a deviation network. This network calculates the loss of the deviation between the anomaly scores of the normal distribution using labeled data and the anomaly scores of abnormal moments. The best graph network parameters θ are optimized by this combined loss from N domains, transforming the unsupervised task into a partially-supervised one. During meta-validation phase, the relation matrix acquired from meta-training phase cooperates with partially-supervised loss to update the original parameters θ on the base of θ'. Leveraging TRMLearner empowers the model with improved cross-task adaptability in anomaly detection, leading to superior generalization capabilities in highly dynamic and complex environments. During the testing phase, the model is fine-tuned with few-shot target task data from D_{new}^s to adapt the model parameters to the target network. At last, the model's performance is then assessed on D_{new}^q.

C. Graph Structure Learning

In this section, we will establish a graph based on the features of a multivariate time series to capture the relationships among different sensors. Let $G = (V, A)$ be defined as follows: V denotes the node set, and A is a $n \times n$ adjacency matrix, where n is the number of sensors. As the relationships between nodes may not be mutual, we use a directed graph. $A_{ij} = 1$ represents a directed edge from v_i to v_j, indicating that the data of v_j depends on the changes in v_i.

We need to figure out the intricate relationships between the sensors in order to construct a graph structure. The correlation between two sensors can be computed using the similarity between nodes. Different sensors have distinct features that not only express their individual characteristics but also establish strong connections with related sensors. To flexibly represent the latent features of sensors, we introduce node embeddings (high-dimensional vectors) for each sensor. The randomly initialized node embedding vectors are updated using a weight matrix as the model evolve. We denote the embeddings as $\mathbf{V}_i$, and the stronger the similarity between them, the stronger the correlation between nodes. This can be utilized for the establishment and updating of the graph structure.

The correlation of node characteristics represented by vectors $\mathbf{V}_1$ and $\mathbf{V}_2$, where $\mathbf{V}_1$ and $\mathbf{V}_2$ correspond to the embeddings of nodes v_1 and v_2, is obtained by computing the cosine similarity between node embeddings. To control the number of neighboring nodes, we predefine the maximum number of neighbors as TopK to limit the number of edges. Users can set the value of TopK based on the number of sensors. The model selects the TopK nodes based on the values of R_{ij} in descending order and assigns $A_{ij} = 1$, establishing edges between the current node and its chosen neighbors

$$R_{ij}(v_i, v_j) = \frac{\mathbf{V}_i \cdot \mathbf{V}_j}{\|\mathbf{V}_i\| \|\mathbf{V}_j\|}. \tag{1}$$

D. Partially-Supervised Graph Deviation Networks

The PS-GDN we propose is a partially supervised model that simultaneously considers time-series prediction and anomaly detection, and fully utilizes few-shot labeled data. PS-GDN updates the network with mean square error loss L_{MSE} and deviation loss L_{DEV}. L_{MSE} mainly focuses on the difference between the predicted values of the model and the true values, while L_{DEV} focuses on the data changes of the predicted values themselves. PS-GDN consists of three key components: 1) prediction based on graph attention and

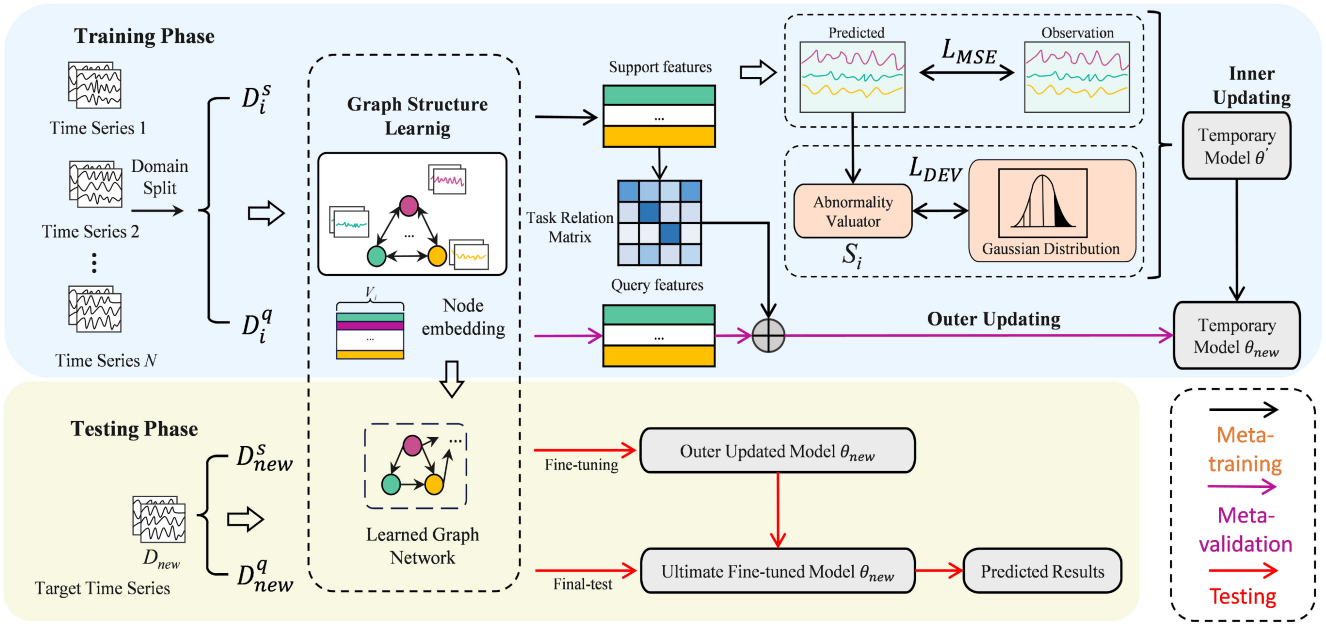


Fig. 3. Overview of the PS-GDNML proposal. During meta-training phase, PS-GDNML uses deviation loss and forcasting-based loss to get a temporary model θ' and calculate the task relation matrix. During the meta-validation phase, the task relation matrix is utilized to guide the learning of original parameters θ on the base of θ'. During testing phase, the pretrained network is fine-tuned on D^s_{new} and assessed on D^q_{new} during the last evaluation stage.

corresponding mean square error; 2) an anomaly evaluator that assigns anomaly scores to each sensor; and 3) the deviation loss obtained from part of labeled data.

Graph Attention-Based Forecasting: To enable the model to recognize deviations in sensors' behaviors, we employ a prediction-based approach by forecasting the expected performance of each sensor using historical data. By comparing actual behavior to expected behavior, the model not only accurately identifies anomalies but also provides an intuitive understanding of why a sensor is deemed anomalous. Therefore, we take $\mathbf{x}_{(t)} = [\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}]$ as the input to the model, where $\mathbf{x}_i = [x_i^1, \ldots, x_i^n]$, which is based on a historical time-series data with a sliding window. Our objective is to obtain the predicted result $\hat{\mathbf{x}}_t = [x_t^1, \ldots, x_t^n]$.

To compile data regarding a node and how it is correlated with its neighbors, we introduce a graph attention mechanism into feature extraction, enabling prediction based on the learned graph structure. The graph attention technique can be extended to accommodate heterogeneous effects from different sensors by using node embeddings. We use $\alpha_{i,j}$ represents attention coefficients, where the value of $\alpha_{i,j}$ indicates the importance of the features of node j to node i. We simultaneously take into account the influence of two nodes and compute the attention value $\epsilon_{i,j}$ between them in order to determine $\alpha_{i,j}$

$$\epsilon_{i,j} = \text{LeakyReLU}\Big(\mathbf{a}((\mathbf{W}\mathbf{x}^i_{(t)} \oplus \mathbf{V}_i) \oplus (\mathbf{W}\mathbf{x}^j_{(t)} \oplus \mathbf{V}_j))\Big) \quad (2)$$

where LeakyReLU is the nonlinear activation function, $\mathbf{a}$ is the learned coefficient vector for the attention mechanism, $\oplus$ represents concatenation, and $\mathbf{W}$ is the updatable linear transformation matrix where each node's weights are shared. By merging the features at the current time series $\mathbf{x}^i_{(t)}$ with the node embedding $\mathbf{V}_i$, a more comprehensive calculation of attention factors can be performed. When aggregating information from neighbors, it is necessary to normalize the attention for all neighbors of each node. The normalized weights after normalization are the attention coefficients $\alpha_{i,j}$

$$\alpha_{i,j} = \text{softmax}_j(\epsilon_{ij}) = \frac{\exp(\epsilon_{ij})}{\sum_{k\in E(i)} \exp(\epsilon_{ik})} \quad (3)$$

where $E(i)$ is the set of neighbors for node i. Subsequently, through the attention coefficients $\alpha_{i,j}$, we get the aggregated representation z_t^i for each node and use the fully-connected layers to predict sensors' value $\hat{x}_t$

$$z_t^i = \text{LeakyReLU}\left(\alpha_{i,i}\mathbf{W}\mathbf{x}^i_{(t)} + \sum_{j\in E(i)} \alpha_{i,j}\mathbf{W}\mathbf{x}^j_{(t)}\right) \quad (4)$$

$$\hat{x}_t = \text{Layer}(z_t^i, \ldots, z_t^n). \quad (5)$$

After obtaining the predicted vector values of sensors, we use the Mean-Squared Error as the minimum loss function between the predicted output $\hat{x}_t$ and the observed value x_t

$$L_{\text{MSE}} = \frac{1}{l-w}\sum_{t=w+1}^{l} \|\hat{x}_t - x_t\|^2. \quad (6)$$

Abnormality Valuator: In order to better obtain anomaly scores from time series suitable for deviation networks, we construct an anomaly evaluator of which the architecture consists of a ReLU layer and a linear layer. Our purpose is to map our time series to a scalar anomaly score s_t^i

$$\mathbf{o}_t^i = \text{ReLU}(\mathbf{W}_o\hat{x}_t^i + \mathbf{b}_o) \quad (7)$$

$$s_t^i = \mathbf{u}_s^T\mathbf{o}_i + b_s \quad (8)$$

where the weight vector and learnable weight matrix are denoted by $\mathbf{u}_s^T$ and $\mathbf{W}_o$. $\mathbf{b}_o$ and b_s represent respective bias parameters. s_t^i is the ultimate anomaly score of node v_i for moment t.

Deviation Loss: The deviation loss is a flexible component that can be determined based on the presence or absence of labeled data. When there are no labeled data, we can use L_{MSE} to update the model. If there are few labeled samples, both L_{MSE} and L_{DEV} can be used to update the model, making the model partially supervised. The model is forced by the deviation loss to assign higher anomaly scores to nodes whose attributes differ greatly from those of normal nodes. We will define a reference score as a measure of the overall anomaly score for the data. Based on previous research, a Gaussian distribution is often a robust approach for fitting anomaly scores across diverse datasets.Therefore, we randomly sample a batch of anomaly scores $[s_1^i, \ldots, s_k^i]$ from a Gaussian prior distribution, calculate the mean to obtain the reference score μ_s, and compute the corresponding standard deviation σ_s. The final anomaly scores can be defined as the standard score

$$\text{dev}\left(s_t^i\right) = \frac{s_t^i - \mu_s}{\sigma_s} \tag{9}$$

where μ_s and σ_s can be replaced by 0 and 1 to ensure the stability of the model's performance because during the experiment we have found that PS-GDN is not sensitive to the values of μ_s and σ_s. We aggregate the sensors using the maximum function in order to calculate the entire anomaly at time ticket t

$$S_t = \max \ \left\{\text{dev}\left(s_t^i\right)\right\}_{i \in n} \tag{10}$$

subsequently, we obtain the final deviation loss based on the contrastive loss

$$L_{\text{DEV}} = (1 - y_i) \cdot |S_t| + y_i \cdot \max \quad (0, r - S_t) \tag{11}$$

where y_i is the true value of the current time ticket data label, and r is a confidence parameter that defines the radius of the bias. Under the influence of deviation loss, PS-GDN endeavors to force the anomaly scores of abnormal nodes to fall between r and μ_s, while simultaneously pushing the anomaly scores of normal nodes as near to μ_s as possible. Therefore, PS-GDN employs part of anomaly-labeled samples to learn high-level abstractions of normal samples. When the features of the data significantly deviate from the normal patterns, PS-GDN assigns a higher anomaly score to such instances.

We add up the two losses to obtain the partially-supervised loss, and allocate two adjustable weights to them for jointly training the model

$$L_{PS} = \lambda_1 L_{\text{MSE}} + \lambda_2 L_{\text{DEV}}. \tag{12}$$

As for the anomaly results of PS-GDN, we adopt a threshold-based discrimination method that takes into account the deviation between the predicted value and the observed value. We define the reference score deviation for v_i at time t as

$$h_t^i = \left|\text{dev}\left(\hat{s}_t^i\right) - \text{dev}\left(s_t^i\right)\right| \tag{13}$$

where $\text{dev}(\hat{s}_t^i)$ is the predicted value's reference score and $\text{dev}(s_t^i)$ is the observed value's reference score.

Similarly, we aggregate to obtain H_t using the maximum function. If the reference score deviation for a node exceeds a fixed threshold, the data at time t is considered anomalous, and we can also identify these nodes for anomaly localization. To avoid introducing additional hyperparameters, we employ a simple approach in our experiments, setting the threshold to the maximum H_t value observed in the validation data.

E. Task Relation Meta-Learner

TRMLearner consists of two phases: 1) inner loop and 2) outer loop. In the inner loop, gradient descent is employed to fine-tune the parameters of a model and the relation matrix is calculated. The outer loop, then, goes a step further to optimize these inner loop parameters guided by related tasks, resulting in a model with optimal initial hyperparameters. In essence, the goal is to find hyperparameters that enable quick adaptation to new target tasks. By making slight adjustments to the parameters in response to changes in the gradient direction, the model achieves significant improvements in the loss on target tasks.

Meta-Training (Inner Loop): During the meta-training phase, let D_i^s and D_j^s represents a constructed anomaly detection task sampled from the meta-training domain D^s. In each epoch, there are N learning tasks. The model's original parameters are denoted as θ, and f_θ represents the model. We first input D_i^s and D_j^s into the feature extractor g of the model f_θ. Then we use a multihead layer to compute the cosine similarity between them

$$m_{i,j} = \frac{1}{K} \sum_{i=1}^{K} \cos\left(w_k g(D_i^s), w_k g(D_j^s)\right) \tag{14}$$

where K is the number of heads, w_k is the learnable vector. The ultimate relation matrix denotes as M.

Next, for task D_i^s, the optimization methodology adjusts the original model parameters θ to θ_i' independently. We use $L_{PS}(f_\theta, D_i^s)$ comes from a batch of training data sampled from D_i^s to compute the updated parameters θ_i', representing the temporary model. The specific update process is as follows:

$$\theta_i' = \theta - \alpha \nabla_\theta L_{PS}\left(f_\theta, D_i^s\right) \tag{15}$$

where α is the innner loop's learning rate. Equation (18) pertains only to D_i^s, as the model aims to learn optimal parameters across N domains for all domains. The definition of the meta-objective function is as follows:

$$\min_\theta \ \sum_{i=1}^{N} L_{PS}\left(f_{\theta_i'}, D_i^s\right). \tag{16}$$

Meta-Validation (Outer Loop): Through the meta-training domain D^s, PS-GDNML has acquired relation matrix M and temporary model $f_{\theta'}$ after meta-training. Due to the influence of the task distribution on $m_{i,j}$, directly using $m_{i,j}$ as the relation weight may lead to cumulative errors. TRMLearner is based on the assumption that similar tasks often share similar prediction functions and incorporates a relation-aware consistency regularization term as a more refined alternative to rough weighting. The task relationships on D^s help capture the underlying commonalities between different tasks. This shared knowledge can be transferred to better generalize to the target task, thereby achieving higher performance in D^q

data learning. Therefore, we use the task relationships from D^s to guide the learning of D^q. Let D_i^q consists of (x_i, y_i) represents a task sampled from the meta-validation domain D^q. The regularization term integrates task relations to assign weights to the predictions produced by all other task-specific models. Thus we can get the task relation loss L_{TR}

$$L_{TR} = \frac{1}{N}\sum_{i=1}^{N} L\left(\frac{\sum_{p=1,p\neq i}^{N} m_{ip} f_{\theta_p'}(x_i)}{\sum_{q=1,q\neq i}^{N} m_{iq}}, y_i\right) \tag{17}$$

where L is the loss encourages alignment between the ground truth and the weighted average prediction derived from all other task-specific models, utilizing task relationships to distribute their respective contributions. L_{TR} guides the model to assign more weight to predictions from similar tasks while minimizing reliance on those from less related tasks. Then the total loss can be listed as follows:

$$L_{\text{Total}} = L_{PS} + \lambda_3 L_{TR}. \tag{18}$$

The original model parameters θ are then updated using losses above from each task based on $f_{\theta'}$ through multiple iterations, aiming to enhance the model's generalization to new domains. We employ SGD to update the original parameters θ across all tasks

$$\theta = \theta - \beta\nabla_\theta \sum_{i=1}^{N} L_{\text{Total}}\left(f_{\theta_i'}, D_i^q\right) \tag{19}$$

where β is the outer loop's learning rate. Algorithm 1 illustrates the complete flow of the algorithm.

Algorithm 1 Training and Testing PS-GDNML

Training Procedure
Input: (1) Source domains:$D = \{D_1, ..., D_N\}$;(2) training epochs E,and meta-learning rate α, β.
Output: Pretrained model.
Initialize parameters θ.
while $e < E$ **do**
    Randomly split D into D^s and D^q, $D^s \cap D^q = \varnothing$.
    Meta-training:
    for each domain D_i^s in D^s **do**
        Calculate task relation matrix M;
        Calculate meta-training loss L_{PS};
        Calculate updated parameters: $\theta_i' = \theta - \alpha\nabla_\theta L_{PS}(f_\theta, D_i^s)$;
    end for
    Meta-validation:
    Calculate meta-validation loss loss L_{Total} on D_i^q;
    Update θ: $\theta = \theta - \beta\nabla_\theta \sum_{j=1}^{N} L_{Total}(f_{\theta_i'}, D_i^q)$;
end while
Testing Procedure
Input: Pretrained model,training domain D_{new}^s and testing domain D_{new}^q from D_{new}.
Output: Model's performance on D_{new}^q.
while available training data **do**
    Fine-tune θ on D_{new}^s to get optimal θ';
end while
$Precision, Recall, F1 = f(\theta', D_{new}^q)$;
Return anomaly detection performances on D_{new}^q.

IV. Experiments and Evaluations

A. Datasets

In this experiment, we have used five publicly available datasets to assess our model: SWaT [39], WADI [40], KDDCUP99 [41], MSL [6], and SMD [42]. These datasets, characterized by multivariate time series, provide ample instances for both training and testing phases. The key statistics, including their dimensions and anomalies, are summarized in Table I.

SWaT originates from the Network Security Center at the Singapore University of Technology and Design. The collection, which consists of 264 hours of statistical and network-related data, was gathered from 51 processing units and sensors over the course of 11 days. It encompasses seven days of regular operational data collected when the system was operating as it ought to, along with four days of aberrant data collected during an attack.

As an expansion of the SWaT, WADI represents an extended network that encompasses a higher volume of water distribution pipelines. The WADI dataset's KPIs are based on information gathered from 123 water distribution system sensors and devices. The dataset covers a continuous period of 16 days, capturing information from the network, sensors, and devices. This includes 14 days of normal operational data and two days of anomalous data, featuring 15 instances of attacks from the same attack model.

The KDDCUP99 dataset is a network intrusion benchmark containing 34 traffic features, labeling connections as normal or attack. It includes 39 attack types across four categories, with 22 attacks unique to training and 17 exclusive to testing. This distribution challenges detection systems to handle novel attack patterns. The dataset remains widely used despite its age due to its comprehensive attack coverage.

The MSL dataset is a publicly available real-world expert-labeled dataset collected by NASA. It contains multivariate time-series data from the Mars Science Laboratory rover, with 27 entities monitored by 55 metrics. The data has a time granularity of 1 minute and has been scaled to the range of 0–1.

The SMD dataset is a publicly available multivariate time-series dataset designed for anomaly detection in server performance monitoring. It collects 38 metrics from 28 physical servers in a large Internet company, sampled at 1-min intervals over five weeks.

TABLE I
GENERAL INFORMATION ABOUT DATASET

Dataset	SWaT	WADI	KDDCUP99	MSL	SMD
Dimension	51	123	34	55	38
Training Size	496800	1048571	562387	58317	708405
Testing Size	449919	172801	494021	73729	708420
Anomalies	11.98%	5.99%	80.31%	10.50%	4.16%

B. Baselines

This experiment is conducted with eleven baselines. It includes classical statistical methods, machine learning approaches, and deep learning methods. The remaining baseline models are complicated composite models that have demonstrated exceptional performance in relevant tasks over the past few years.

1) *PCA:* PCA [2] is a linear dimensionality reduction technique, which casts time-series data in several dimensions in order to preserve the variation and inherent qualities of the original data. Potential outliers can be found using PCA by identifying notable deviations of data samples from others in particular directions. This process aids in revealing the key structures and trends in the data.
2) *LSTM:* Traditional deep learning models like LSTM [6] exploit time-series context connections by recursive operations. In the realm of anomaly detection, LSTM-based algorithms primarily leverage a predictive approach. This entails training the LSTM to comprehend intricate patterns within the sequence, enabling it to forecast the next values in the sequence. Instances exhibiting substantial deviations from the predicted values are identified as anomalies.
3) *LSTM-VAE:* LSTM-VAE [43] anomaly detection integrates LSTM with VAE, capturing temporal patterns with LSTM and learning a compact latent representation of data through VAE. During training, the model minimizes reconstruction errors to learn normal time-series patterns, and during testing, instances with high reconstruction errors are flagged as anomalies, providing an effective approach for time-series anomaly detection.
4) *DAGMM:* DAGMM [44] integrates an autoencoder with a Gaussian mixture model, allowing for the simultaneous optimization of parameters in an end-to-end manner. The deep autoencoder is applied in DAGMM to capture the reconstruction error while compressing input data points into a low-dimensional representation. The Gaussian mixture model receives the low-dimensional representation of the data and uses the reconstruction loss to determine if the data is anomalous.
5) *MAD-GAN:* MAD-GAN [45] is a reconstruction-based anomaly detection model using GAN architecture. During training, the generator synthesizes data from inputs and latent variables. For testing, it first infers optimal latent variables matching the test data distribution, then reconstructs sequences and computes reconstruction errors. The anomaly score combines these errors with discriminator outputs for detection.
6) *USAD:* Leveraging a self-encoder architecture driven by GANs, USAD [46] is an unsupervised anomaly detection method for multivariate time series. USAD's encoder–decoder design can learn how to provide more stability than traditional GANs architecture while increasing the reconstruction error of inputs, including anomalies through the use of adversarial training.
7) *GDN:* GDN [29] is a multivariate detection method of temporal anomalies, which utilizes a prediction-based strategy. It learns the graph structure based on node similarity, treats each feature dimension as a node in a GNN, and uses graph attention mechanisms to compute anomaly scores.
8) *CAE:* The convolutional ensemble CAE [47] incorporates multiple basic outlier detection models based on convolutional sequence-to-sequence autoencoders. In order to preserve variation among the base models and increase accuracy, CAE uses a unique diversity-driven training approach. Increased efficiency results from this method's advantages in parallelism.
9) *TranAD:* TranAD [48] is a transformer-based anomaly detection model that combines adversarial training with attention mechanisms. It uses reconstruction error and adversarial scores to detect anomalies in time-series data, achieving high accuracy without labeled data. The architecture efficiently captures temporal patterns while maintaining robustness to noise.
10) *TopoGDN:* TopoGDN [49] is an advanced model designed for anomaly detection in multivariate time series. It effectively captures intricate temporal and feature relationships through multiscale temporal convolutions for extracting fine-grained time features and an upgraded GAT that integrates graph topology into node representations across scales. This design effectively captures complex patterns and boosts detection accuracy.
11) *RANSynCoders:* RANSynCoders [50] aligns signals using spectral analysis on the latent space of a pretrained autoencoder and learns phase shifts to create a synchronized representation. Random subsets of the synchronized data are fed into multiple autoencoders, which optimize quantile reconstruction losses. Anomalies are identified and localized through a majority-vote mechanism across these autoencoders.
12) *DTAAD:* DTAAD [51] integrates a Transformer encoder with Dual temporal convolutional networks (TCNs) in an autoencoder-AR hybrid architecture. By employing scaling methods and feedback mechanisms, it enhances prediction accuracy while maintaining an ultra-lightweight structure with only a single transformer layer, enabling fast and accurate anomaly detection even with limited labels and hardware constraints.
13) *STTD:* STTD [52] is a transformer-based anomaly detection model that jointly models spatial-temporal interactions in time-series data. It uses parallel encoder–decoder architecture with cross-attention to fuse multiscale temporal patterns and variable relationships, enhanced by channel attention for effective feature aggregation.

C. Experiment Setup

Specifically, we initialized node embedding dimensions and sliding window sizes for each dataset as follows: SWaT (64, 5), WADI (128, 5), KDDCUP99 (64, 10), MSL(64, 15), and SMD(64, 15). The confidence in the deviation network is set to 5, and reference scores are computed based on 10 000 scores sampled from a Gaussian prior distribution. Toward training data, we divide the time series into five segments, corresponding to five domains, including recent, medium-term, and long-term time series, constituting the source domain D. We use 50% of D to constitute D^s and the rest as D^q. For model training, we configure 100 epochs for PS-GDNML, and set early stopping at 15. In the meta-learning process, we iterate five times to compute updates for θ'. Both outer learning rate and inner learning rate are set to 0.001. Fine-tuning allows the model to adjust its parameters to better adapt to new tasks, fully leveraging the high generalizability advantage of meta-learning tasks. So we perform fine-tuning using 20% of the data for the target domain D_{new}, while the remaining 80% is reserved for testing. To implement our model, we use the PyTorch framework, and related experiments are conducted on an NVIDIA TITAN Xp graphics processor.

D. Experiment Results

Evaluation Metrics: We assessed the model's performance on three metrics: 1) Precision; 2) Recall; and 3) $F1$_score. Precision measures the accuracy of positive predictions, indicating the percentage of positives that are accurately predicted. Recall assesses the model's capacity to take into account all relevant positive instances. $F1$_score combines Precision and Recall, providing a balanced metric that considers both false positives and false negatives.The formulas of three metrics are defined as follows:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ F_1 &= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{20}$$

where TP, TN, FN, and FP denote true positive, true negative, false negative, and false positive. Precision emphasizes prediction accuracy, Recall focuses on capturing all relevant instances, and $F1$_score strikes a balance between them.

Baseline Compasrison: In the experiment, we use the SWaT, WADI, KDDCUP99 MSL, and SMD datasets to compare PS-GDNML's performance with the baselines, as shown in Table II and Fig. 4. The results indicate that our model achieved the highest $F1$_score values across all datasets except MSL with Precision and Recall also outperforming most models. Specifically, the SWaT dataset performs the best in terms of Recall and $F1$_score. With reference to the second-best baseline, there is an improvement of 4.6% in Recall and 3.3% in $F1$_score. Given the high dimensionality and imbalanced data distribution in WADI, previous models typically show mediocre performance in terms of $F1$_score. GNN-based methods, such as GDN and TopoGDN are limited by their fully unsupervised nature. By leveraging few labeled anomalies through deviation loss, our model demonstrates superior effectiveness with $F1$_score increasing 9.5% compared to the second baseline DTAAD, indicating its suitability for handling high-dimensional data. Though PS-GDNML fails to get the best $F1$_score in MSL, its Recall is high as 99.41% and the gaps between the other two metrics compared with the optimal baseline are extremely narrow. For dataset KDDCUP99 and SMD, which have low dimensions, our model achieves the best $F1$_score. The consistent excellent performance across low to high dimensions suggests that our model can effectively adapt to features of varying dimensions. Besides, in KDDCUP99, where anomalies dominate, many models struggle to differentiate between normal and anomalous patterns, but PS-GDNML effectively captures structural dependencies using its graph-based representation, maintaining high recall while preventing overfitting to frequent anomalies. Conversely, in SMD, where anomalies are rare, traditional models often suffer from low recall due to insufficient supervision, yet PS-GDNML's partially-supervised learning strategy leverages limited labeled anomalies to enhance detection without excessive false positives. These results highlight the robustness and adaptability of PS-GDNML across datasets with vastly different anomaly distributions. Moreover, with TRMLearner, our model achieves stable results scores across all datasets by leveraging task relationships for better generalization. This prevents overfitting and ensures consistent, high-performance anomaly detection in diverse scenarios.

Few-Shot Evaluation: In fine-tuning, introducing few-shot data can achieve better results. To assess the impact of few-shot data on model fine-tuning, we conducted experiments with 1-shot, 5-shot, 10-shot, and 20-shot scenarios, evaluating the model's performance on the target domain. It's worth noting that due to the partially supervised nature of the model, labeled data is not mandatory. During the evaluation phase, we provided labeled data equal to the number of shots. Table III illustrates the performance of PS-GDNML under various few-shot settings based on three metrics. Despite the limited number of samples in the 1-shot scenario, the model still achieved results superior to most baselines. This indicates the effectiveness of fine-tuning with few-shot data. As the number of shots increased, there were fluctuations in Precision and Recall, but the $F1$_score gradually improved. The overall model performance showed enhancement with the increase in shots, demonstrating the comprehensive utilization of few-shot data.

Label Proportions: To validate whether our model can effectively utilize labels, i.e., the feasibility of partially-supervised learning, we introduce label proportions in the training set and observe the model's performance on five datasets under different label quantities. According to Fig. 5, both SWaT and WADI datasets exhibit some performance degradation at a 20% label proportion. As shown in Table IV, the SWaT dataset has 11.98% anomalous data, while WADI has 5.99%. Hence, in situations where anomalous data distribution is uneven and the proportion is small, a limited number of labels are not advantageous for the model's learning; instead, they may interfere with performance. Contrastingly, for the

TABLE II
PERFORMANCE OF PS-GDNML AND 14 BASELINES ON FIVE DATASETS

	SWaT			WADI			KDDCUP99			MSL			SMD		
Methods	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
PCA	0.4712	0.4423	0.4563	0.3826	0.1822	0.2468	0.8544	0.3458	0.4923	0.1262	0.5612	0.2061	0.2718	0.3457	0.3050
LSTM	0.5945	0.5276	0.5591	0.7242	0.2793	0.4043	0.7015	0.9038	0.7900	0.8073	0.7501	0.7776	0.8224	0.7461	0.7824
DAGMM	0.7031	0.4713	0.5643	0.5491	0.2869	0.3768	0.8531	0.9746	0.9098	0.8259	0.7693	0.7966	0.8953	0.9670	0.9296
LSTM-VAE	0.9540	0.5949	0.7328	0.4587	0.3212	0.3778	0.9285	0.8274	0.8751	0.8449	0.7964	0.8199	0.7929	0.9120	0.8483
MAD-GAN	0.9333	0.6245	0.7483	0.4056	0.3873	0.3962	**0.9628**	0.7106	0.8178	0.8116	0.9830	0.8891	**0.9779**	0.8144	0.8887
USAD	0.9635	0.6446	0.7724	0.8632	0.2787	0.4213	0.9569	0.9181	0.9370	0.9005	0.8034	0.8492	0.8452	0.8903	0.8661
GDN	0.9499	0.6415	0.7658	0.8947	0.3908	0.5439	0.8494	0.9636	0.9029	0.8671	0.9065	0.8864	0.8015	0.9127	0.8535
CAE	0.9645	0.5856	0.7287	0.4736	0.1652	0.2249	0.9241	0.5334	0.7288	0.8512	0.7454	0.7947	0.8639	0.9216	0.8917
TranAD	0.9623	0.7026	0.8122	0.3529	**0.8296**	0.4951	0.8102	0.9448	0.8709	**0.9226**	0.8668	0.8938	0.8942	0.9679	0.9293
TopoGDN	0.9201	0.6504	0.7621	0.5808	0.4485	0.5059	0.9528	0.9061	0.9289	0.8164	0.9867	0.8935	0.9757	0.8339	0.8992
RANSynCoders	0.9364	0.7024	0.8026	0.3459	0.4008	0.3713	0.9283	0.9657	0.9466	0.8586	0.8891	0.8736	0.8621	0.8042	0.8321
DTAAD	0.9697	0.6957	0.8101	**0.9017**	0.3910	0.5455	0.8391	0.9912	0.9094	0.8808	0.9893	**0.9292**	0.8463	**0.9974**	0.9147
STTD	**0.9762**	0.6962	0.8118	0.4833	0.6138	0.5408	0.8478	**0.9984**	0.9169	0.8708	0.9906	0.9268	0.8526	0.9835	0.9134
PS-GDNML	0.9708	**0.7483**	**0.8452**	0.8433	0.5168	**0.6409**	0.9537	0.9895	**0.9721**	0.8635	**0.9941**	0.9242	0.9081	0.9892	**0.9432**

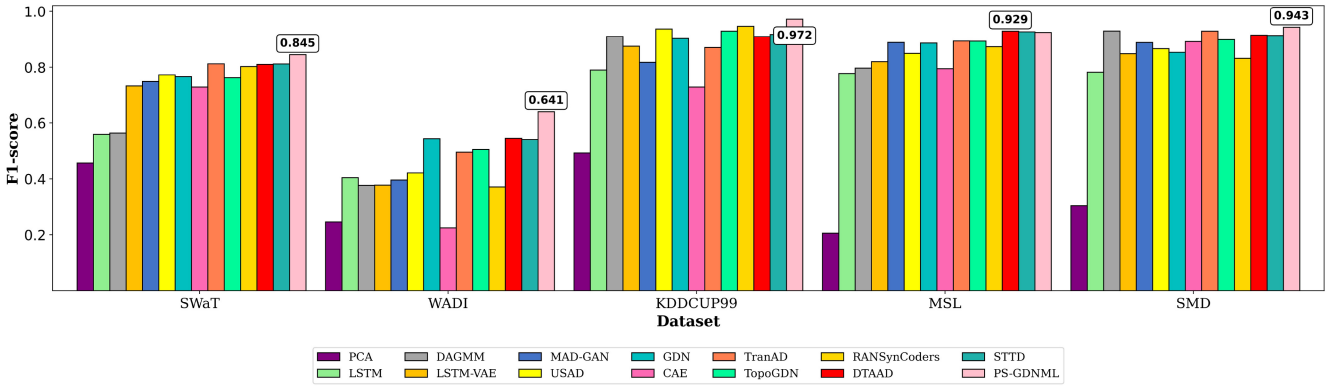


Fig. 4. F1-score of PS-GDNML with baselines.

TABLE III
FEW-SHOT PERFORMANCE EVALUATION OF PS-GDNML

	SWaT			WADI			KDDCUP99			MSL			MSD		
Setting	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1-shot	0.9241	0.6824	0.7851	0.7671	0.4632	0.5776	0.9055	0.9443	0.9245	0.8428	0.9275	0.8831	0.8566	0.9379	0.8951
5-shot	0.9396	0.7132	0.8190	0.8203	0.4847	0.6093	0.9258	0.9511	0.9383	0.8528	0.9570	0.9019	0.8708	0.9533	0.9099
10-shot	0.9618	0.7396	0.8362	0.8287	0.5054	0.6279	0.9485	0.9742	0.9612	0.8596	0.9888	0.9197	0.8852	0.9744	0.9276
20-shot	**0.9708**	**0.7483**	**0.8452**	**0.8433**	**0.5168**	**0.6409**	**0.9537**	**0.9895**	**0.9721**	**0.8635**	**0.9941**	**0.9242**	**0.9081**	**0.9892**	**0.9432**

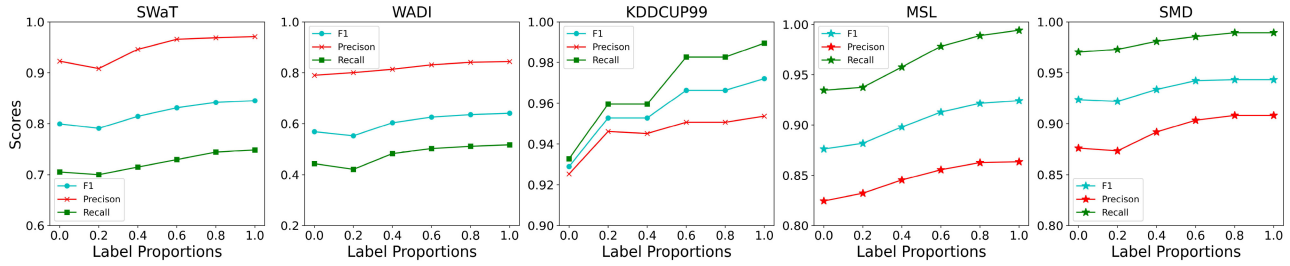
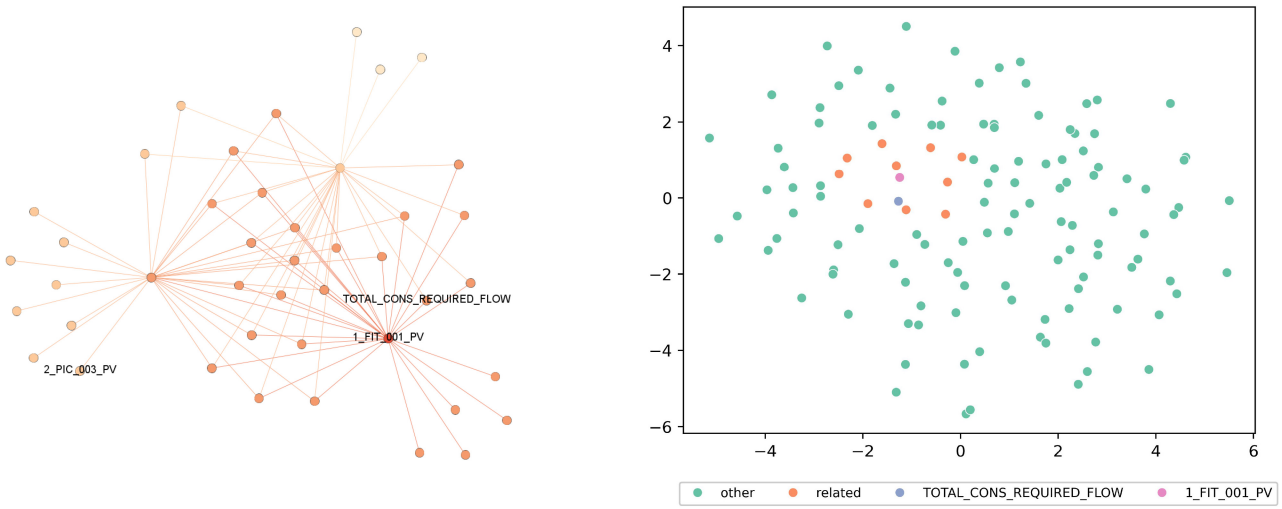


Fig. 5. Performance of different label proportions on five datasets.

KDDCUP99 and SMD datasets, in one of them, the anomaly proportion is as high as 80.31%, and in the other, the anomaly proportion is as low as 4.16%. The model performance shows minimal changes when the label proportions increase to 60%, demonstrating its robustness to varying anomaly distributions through few labels. The MSL dataset contains 10.50% anomaly data, yet its performance is still improved under a 20% label proportion. This is because the dataset has a relatively low dimension, which allows the model to learn appropriate graph structures and extract information effectively even with limited labeled data.

For SWaT and WADI datasets, when the label proportion increases to 40%, there is a noticeable improvement in model performance compared to the unlabeled scenario. SWaT's $F1$_score increases by 1.5%, and WADI's $F1$_score increases by 3.8%. Furthermore, as the label proportion continues to rise,



(a) (b)

Fig. 6. (a) Illustrates a partial graph structure of the model around sensor 1_FIT_001_PV in the WADI dataset. In (b), WADI's node embeddings are reduced to a 2-D plane through PCA.

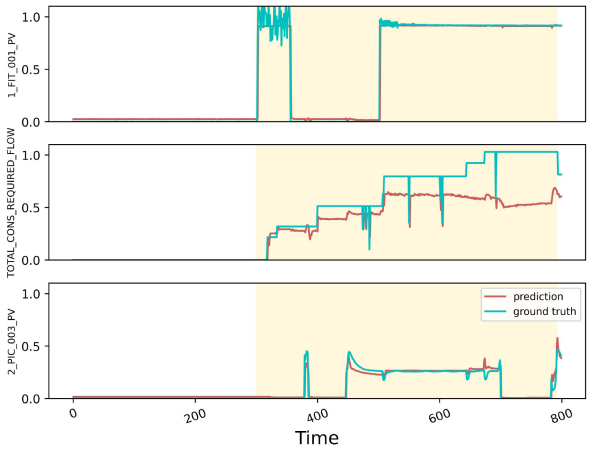


Fig. 7. Sensor observation data and prediction data during abnormal time.

$F1$_score consistently improve. At a 100% label proportion, SWaT and WADI datasets see $F1$_score increases of 4.6% and 7.3%, respectively. In the case of a 60% label proportion, SWaT's $F1$_score is only 1.4% lower than with 100% labels, and WADI's $F1$_score is only 1.5% lower. This indicates that our model can achieve excellent performance with partial labels, reducing dependency on labels while fully leveraging the available ones, demonstrating the effectiveness of partially-supervised learning.

E. Qualitative Analysis

In this section, we will investigate the interpretability of the model. In our experiments, we focus on the analysis of the 1_FIT_001_PV sensor in the WADI dataset. Our exploration centers around two main aspects: 1) graph node embeddings and 2) the effectiveness of time-series anomaly detection.

Graph Node Embeddings: As the graph node embedding vectors represent the features of individual nodes in the graph, playing a crucial role in constructing the graph structure, we utilized PCA to project the embedding vectors of 127 sensors in the WADI dataset onto a 2-D coordinate system. We selectively chose a subset of sensors with higher relevance to sensor 1_FIT_001_PV based on the projection distance, as depicted in Fig. 6. In Fig. 6(a), darker node colors indicate a higher degree of information aggregation from neighboring nodes. From the information in Fig. 6(a), we observe that sensor 1_FIT_001_PV is associated with TOTAL_CONS_REQUIRED_FLOW. Marking these two points in Fig. 6(b), we find that their embedding vectors are close in the 2-D plane. This confirms that the fine-tuned graph structure accurately reflects the relationships between nodes in the current task. It also suggests that node embeddings can serve as node features to represent the dependencies between nodes.

Time-Series Anomaly Detection: In accordance with the documentation of WADI, our investigation is conducted on a case with a known cause of anomaly. We selected three sensors: 1) 1_FIT_001_PV; 2) 2_PIC_003_PV; and 3) TOTAL_CONS_REQUIRED _FLOW. The time-series predictions and true values for this period are depicted in Fig. 7, where the yellow region indicates anomalous moments, with TOTAL_CONS_REQUIRED_FLOW having the highest anomaly score during this time. The anomaly in this case is caused by a misreading of sensor 1_FIT_001_PV. Despite the sensor being under attack, its data changes still fall within the normal range. This type of anomaly is challenging to detect with conventional single-instance detection methods. Utilizing the graph structure, we observed that the sensor TOTAL_CONS_REQUIRED_FLOW, closely related to sensor 1_FIT_001_PV, exhibits a significant difference between prediction data and detection data during the anomalous period. As the valve of sensor 1_FIT_001_PV is actually closed, the true total flow should be decreasing. However, the predicted values for TOTAL_CONS_REQUIRED_FLOW follow the changing pattern of sensor 1_FIT_001_PV, leading to a rise during this period. This results in a discrepancy

TABLE IV
LABEL PROPORTION PERFORMANCE EVALUATION OF PS-GDNML

	SWaT			WADI			KDDCUP99			MSL			MSD		
Setting	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
0%	0.9224	0.7055	0.7995	0.7898	0.4434	0.5679	0.9254	0.9328	0.9290	0.8245	0.9346	0.8761	0.8760	0.9702	0.9236
20%	0.9080	0.7002	0.7907	0.8003	0.4212	0.5519	0.9461	0.9596	0.9528	0.8322	0.9375	0.8817	0.8733	0.9728	0.9219
40%	0.9458	0.7150	0.8144	0.8113	0.4827	0.6053	0.9461	0.9596	0.9528	0.8453	0.9575	0.8979	0.8917	0.9808	0.9336
60%	0.9657	0.7296	0.8312	0.8306	0.5022	0.6259	0.9507	0.9826	0.9664	0.8556	0.9781	0.9128	0.9034	0.9854	0.9422
80%	0.9686	0.7443	0.8418	0.8405	0.5110	0.6356	0.9507	0.9826	0.9664	0.8627	0.9888	0.9215	0.9081	0.9892	0.9432
100%	**0.9708**	**0.7483**	**0.8452**	**0.8433**	**0.5168**	**0.6409**	**0.9537**	**0.9895**	**0.9721**	**0.8635**	**0.9941**	**0.9242**	**0.9081**	**0.9892**	**0.9432**

between true and predicted values, and the model identifies it as an anomaly based on the anomaly score.

To demonstrate that sensors unrelated to the attacked sensor are less affected, we selected sensor 2_PIC_003_PV. In Fig. 6(a), there is no edge between sensor 2_PIC_003_PV and sensor 1_FIT_001_PV, indicating that they are unrelated. In Fig. 7, even during the anomalous period, the predicted values of sensor 2_PIC_003_PV closely match the true values, suggesting that it is unaffected by the anomaly in the other sensor. This indicates that the model can obtain the expected performance of the data through prediction. The larger the discrepancy between real and expected data, the higher the anomaly score, making it more likely to be considered anomalous data.

F. Ablation Study

We carried out several types of ablation experiments to more thoroughly investigate the function of the suggested essential elements in the model. In these ablation experiments, we experimented with the original GAT, GAT+PS-GDN, GAT+MAML, GAT+TRMLearner, and PS-GDNML on the test sets of five datasets. According to Table V and Fig. 8, PS-GDNML consistently achieves the highest $F1$_score compared to baseline and component variants. Regarding to PS-GDN module, GAT+PS-GDN substantially improves Recall and $F1$_score over the base GAT and GAT+MAML. In KDDCUP99, GAT+PS-GDN even reaches the highest Recall of 99.17%. The learned adjacency matrix guides message passing by capturing node dependencies, while anomalous nodes are identified through both abnormal connectivity patterns and statistical deviations. The partial labels jointly optimize edge weights and deviation boundaries, with the graph structure dynamically adjusting based on the network's anomaly scores. This alternating optimization of graph structure and deviation loss creates a closed-loop system that simultaneously refines topological relationships and detection accuracy, effectively leveraging both data structure and limited supervision. To highlight the advantages of the TRMLearner, we also conducted a comparative experiment with GAT + MAML. The experimental results demonstrate that TRMLearner outperforms standard MAML across all metrics, with particularly notable gains in precision. Unlike MAML's task-agnostic approach, TRMLearner learns a dynamic task relation matrix from meta-data, enabling selective knowledge transfer between related tasks while avoiding interference from dissimilar ones. This relation-aware optimization better preserves task-specific features while leveraging shared patterns, yielding more balanced anomaly detection. The consistent improvements across datasets confirm the value of explicit task relationship modeling.

TABLE V
PERFORMANCE OF FIVE METHODS

Data	Methods	Precision	Recall	F1
SWAT	GAT	0.9034	0.5915	0.7149
	GAT+PS-GDN	0.9132	0.7165	0.8038
	GAT+MAML	0.9224	0.7055	0.7995
	GAT+TRMLeaner	**0.9768**	0.7137	0.8248
	PS-GDNML	0.9708	**0.7483**	**0.8452**
WADI	GAT	0.7047	0.3105	0.4311
	GAT+PS-GDN	0.7633	0.5079	0.6097
	GAT+MAML	0.7898	0.4434	0.5679
	GAT+TRMLeaner	0.8375	0.4958	0.6228
	PS-GDNML	**0.8433**	**0.5168**	**0.6409**
KDDCUP99	GAT	0.7894	0.8661	0.8261
	GAT+PS-GDN	0.9030	**0.9917**	0.9455
	GAT+MAML	0.9254	0.9328	0.9290
	GAT+TRMLeaner	0.9359	0.9818	0.9582
	PS-GDNML	**0.9537**	0.9895	**0.9721**
MSL	GAT	0.8052	0.8235	0.8142
	GAT+PS-GDN	0.8083	0.9647	0.8812
	GAT+MAML	0.8245	0.9346	0.8761
	GAT+TRMLeaner	0.8544	0.9720	0.9087
	PS-GDNML	**0.8635**	**0.9941**	**0.9242**
MSD	GAT	0.8015	0.9127	0.8535
	GAT+PS-GDN	0.8372	0.9635	0.8980
	GAT+MAML	0.8629	0.9466	0.9027
	GAT+TRMLeaner	0.8760	0.9702	0.9236
	PS-GDNML	**0.9081**	**0.9892**	**0.9432**

PS-GDNML model combines above strengths, achieving both the highest precision and recall in most cases, which demonstrates the synergistic effect of jointly optimizing graph structure learning, partial supervision, and meta-learning. The consistent performance gains across diverse datasets highlight the framework's robustness and adaptability to different anomaly detection scenarios. Notably, the improvements are particularly pronounced on complex industrial datasets like SWAT and WADI, suggesting the method's strong suitability for real-world CPS applications where both label scarcity and dynamic environments are common challenges.

G. Hyperparameter Analysis

In the hyperparameter tuning experiments, we investigated the effects of L_{PS} and L_{TR} on model performance to determine the optimal loss weight configuration. During the experiments, we fixed certain hyperparameters while adjusting the target

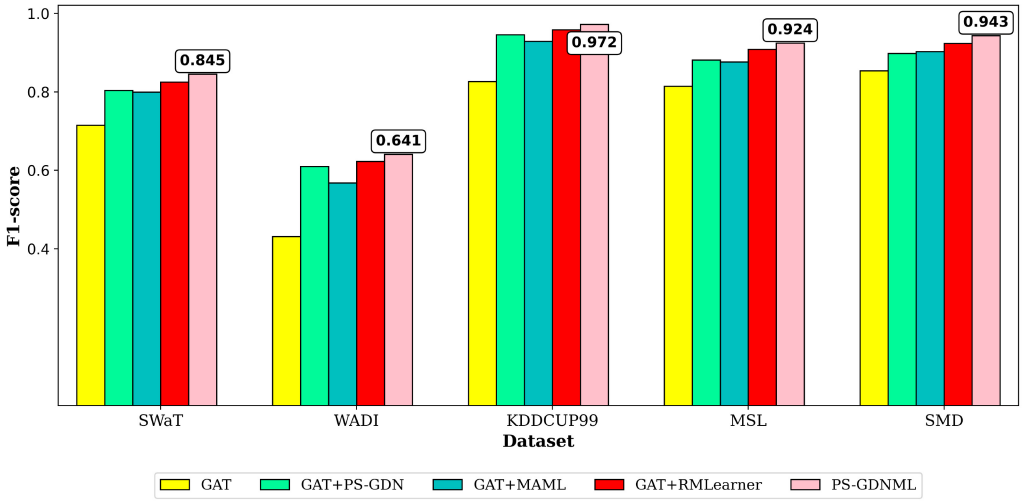


Fig. 8. Comparison of five methods' respective performances on five datasets. The performance on the SWaT, WADI, and KDDCUP99 datasets is displayed, accordingly, from left to right.

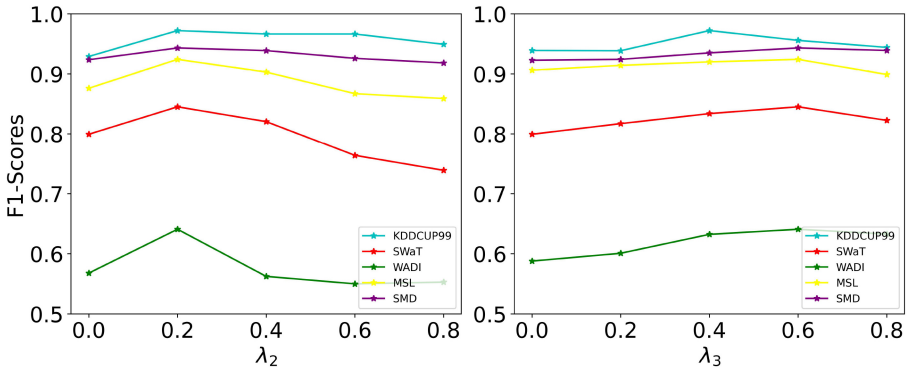


Fig. 9. Performance of PS-GDNML under different λ_2 and λ_3, where λ_2 represents the weight of deviation loss and λ_3 represents the weight of task relation loss.

parameter to observe the changes in the model's $F1$_score across different datasets. Fig. 9 presents the $F1$_score variations for five datasets (KDDCUP99, SWaT, WADI, MSL, and SMD) under different values of λ_2 and λ_3.

To analyze the impact of λ_2, we set $\lambda_3 = 0.6$ and ensured that $\lambda_1 + \lambda_2 = 1$. The results indicate that KDDCUP99 and SMD datasets exhibit relatively low sensitivity to λ_2, with the $F1$_score improving slightly as λ_2 increases to 0.2 but remaining stable beyond that point. For SWaT, WADI, and MSL datasets, the model achieves the highest $F1$_score when $\lambda_2 = 0.2$, suggesting that an appropriate balance between different loss terms is crucial for performance. As λ_2 increases further, the $F1$_score declines, particularly for SWaT and WADI, highlighting that an excessive contribution from the deviation network negatively impacts performance. This indicates that in the learning process, the deviation network should play a supportive role rather than a dominant one.

For the analysis of λ_3, we set $\lambda_1 = 0.8, \lambda_2 = 0.2$, then varied λ_3 to observe its impact on performance. From Fig. 9, we find that as λ_3 increases from 0 to 0.4, the model's $F1$_score improves across all datasets. However, after λ_3 reaches 0.6, the performance starts to decline. The KDDCUP99 dataset achieves the best $F1$_score at $\lambda_3 = 0.4$, whereas all other datasets reach their peak performance at $\lambda_3 = 0.6$. This suggests that a moderate contribution from L_{TR} enhances the model's ability to detect anomalies, but an excessively high λ_3 may introduce undesired effects, reducing overall performance.

Based on these observations, we conclude that setting $\lambda_1 = 0.8, \lambda_2 = 0.2$, and $\lambda_3 = 0.6$ provides a balanced contribution of different loss components, leading to optimal performance across most datasets.

V. Limitations

While our proposed PS-GDNML demonstrates significant advancements in dynamic multitime series anomaly detection, several limitations warrant consideration.

Dependency on Partial Labels: The model's performance remains sensitive to the quality and availability of partially labeled anomalies, which may limit its applicability in fully unsupervised scenarios.

Computational Overhead: Integrating meta-learning with graph-based modeling introduces scalability challenges, particularly for large-scale or high-frequency time series data.

Dynamic Adaptation Constraints: Although the task relation matrix enables adaptability, rapid shifts in temporal patterns or abrupt concept drift may still require additional fine-tuning. *Real-World Deployment:* Resource-intensive components, such as the meta-learner and graph synchronization, could hinder real-time implementation on edge devices. Addressing these limitations will be critical for enhancing PS-GDNML's practicality and extending its utility to broader industrial applications.

VI. Conclusion

In this article, we propose PS-GDNML, a meta-learning-based graph deviation network designed to adapt quickly to anomaly detection tasks across diverse time periods using limited labeled data. The task meta-learner enables rapid adjustment to new graph structures by leveraging generalized knowledge from meta-training tasks, while the deviation network enforces statistical differences between normal and abnormal nodes using partial anomaly labels. Comprehensive ablation experiments validate the critical roles of these components, and integrated evaluations on five public datasets demonstrate superior performance over state-of-the-art methods. Future work will focus on optimizing meta-learning strategies, graph construction, and anomaly scoring, particularly for dynamic GNNs.

Acknowledgment

We would like to thank the reviewers for their comments, which helped improve this article considerably.

References

[1] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 1–33, 2022.

[2] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, and G. Maciá-Fernández, "PCA-based multivariate statistical network monitoring for anomaly detection," *Comput. Secur.*, vol. 59, pp. 118–137, Jun. 2016.

[3] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.

[4] L. Shen, Y. Wei, Y. Wang, and H. Li, "AFMF: Time series anomaly detection framework with modified forecasting," *Knowl. Based Syst.*, vol. 296, Jul. 2024, Art. no. 111912.

[5] M. A. Belay, A. Rasheed, and P. S. Rossi, "MTAD: Multiobjective transformer network for unsupervised multisensor anomaly detection," *IEEE Sensors J.*, vol. 24, no. 12, pp. 20254–20265, Jun. 2024.

[6] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2018, pp. 387–395.

[7] X. Jie et al., "Disentangled anomaly detection for multivariate time series," in *Proc. Companion ACM Web Conf.*, 2024, pp. 931–934.

[8] J. Miao, H. Tao, H. Xie, J. Sun, and J. Cao, "Reconstruction-based anomaly detection for multivariate time series using contrastive generative adversarial networks," *Inf. Process. Manag.*, vol. 61, no. 1, 2024, Art. no. 103569.

[9] B. Chen, H. Lu, Y. Chen, H. Yuan, and M. Wang, "DGNN: Dynamic graph neural networks for anomaly detection in multivariate time series," in *Proc. 35th Int. Conf. Softw. Eng. Knowl. Eng.*, 2023, pp. 415–420.

[10] P. Qi, D. Li, and S.-K. Ng, "MAD-SGCN: Multivariate anomaly detection with self-learning graph convolutional networks," in *Proc. 38th IEEE Int. Conf. Data Eng.*, 2022, pp. 1232–1244.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–14.

[12] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[13] Z. Guo, C. Zhang, Y. Fan, Y. Tian, C. Zhang, and N. V. Chawla, "Boosting graph neural networks via adaptive knowledge distillation," in *Proc. 37th AAAI Conf. Artif. Intell.*, 2023, pp. 7793–7801.

[14] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2019, pp. 353–362.

[15] J. Wang, W. Qiang, C. Sun, C. Zheng, and J. Li, "Rethinking meta-learning from a learning lens," 2025, *arXiv:2409.08474*.

[16] B. Chen, M. Sinn, J. Ploennigs, and A. Schumann, "Statistical anomaly detection in mean and variation of energy consumption," in *Proc. 22nd Int. Conf. Pattern Recognit.*, 2014, pp. 3570–3575.

[17] Z.-G. Zhou and P. Tang, "Continuous anomaly detection in satellite image time series based on Z-scores of season-trend model residuals," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2016, pp. 3410–3413.

[18] A. Marjuni, T. B. Adji, and R. Ferdiana, "Unsupervised software defect prediction using median absolute deviation threshold based spectral classifier on signed Laplacian matrix," *J. Big Data*, vol. 6, p. 87, Sep. 2019.

[19] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *Proc. 6th Eur. Conf. Princ. Data Min. Knowl. Discov.*, 2002, pp. 15–27.

[20] M. Jain, G. Kaur, and V. Saxena, "A K-means clustering and SVM based hybrid concept drift detection technique for network anomaly detection," *Expert Syst. Appl.*, vol. 193, May 2022, Art. no. 116510.

[21] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2000, pp. 93–104.

[22] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Min.*, 2008, pp. 413–422.

[23] A. Goodge, B. Hooi, S. Ng, and W. S. Ng, "Robustness of autoencoders for anomaly detection under adversarial impact," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 1244–1250.

[24] R. Liu et al., "Anomaly-GAN: A data augmentation method for train surface anomaly detection," *Expert Syst. Appl.*, vol. 228, Oct. 2023, Art. no. 120284.

[25] S. Ma, J. Nie, S. Guan, Z. He, and M. Gao, "MPFormer: Multipatch transformer for multivariate time-series anomaly detection with contrastive learning," *IEEE Internet Things J.*, vol. 11, no. 23, pp. 38221–38237, Dec. 2024.

[26] W. Chen, L. Tian, B. Chen, L. Dai, Z. Duan, and M. Zhou, "Deep variational graph convolutional recurrent network for multivariate time series anomaly detection," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 3621–3633.

[27] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[28] D. Wang, P. Wang, J. Zhou, L. Sun, B. Du, and Y. Fu, "Defending water treatment networks: Exploiting Spatio-temporal effects for cyber attack detection," in *Proc. 20th IEEE Int. Conf. Data Min.*, 2020, pp. 32–41.

[29] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 4027–4035.

[30] H. Liu, W. Luo, L. Han, P. Gao, W. Yang, and G. Han, "Anomaly detection via graph attention networks-augmented mask autoregressive flow for multivariate time series," *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19368–19379, Jun. 2024.

[31] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–11.

[32] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic Meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[33] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few-shot learning," 2017, *arXiv:1707.09835*.

[34] H. P. Wang and P. Li, "Unsupervised learning for combinatorial optimization needs meta-learning," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023, pp. 1–19.

[35] Q. Pu, Y. Chen, M. Zhou, J. K.-Y. Ng, and R. Cai, "Bayesian meta-learning: Toward fast adaptation in neural network positioning techniques," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 14924–14937, Apr. 2024.

[36] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang, "Transferring robustness for graph neural network against poisoning attacks," in *Proc. 13th ACM Int. Conf. Web Search Data Mini.*, 2020, pp. 600–608.
[37] K. Ding, J. Wang, J. Li, K. Shu, C. Liu, and H. Liu, "Graph prototypical networks for few-shot learning on attributed networks," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, 2020, pp. 295–304.
[38] M. Lin, W. Li, D. Li, Y. Chen, G. Li, and S. Lu, "Multi-domain generalized graph Meta learning," in *Proc. 37th AAAI Conf. Artif. Intell.*, 2023, pp. 4479–4487.
[39] A. P. Mathur and N. O. Tippenhauer, "SWaT: A water treatment testbed for research and training on ICS security," in *Proc. Int. Workshop Cyber-Phys. Syst. Smart Water Netw.*, 2016, pp. 31–36.
[40] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "WADI: A water distribution testbed for research in the design of secure cyber physical systems," in *Proc. 3rd Int. Workshop Cyber-Phys. Syst. Smart Water Netw.*, 2017, pp. 25–28.
[41] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Def. Appl.*, 2009, pp. 1–6.
[42] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2019, pp. 2828–2837.
[43] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1544–1551, Jul. 2018.
[44] B. Zong et al., "Deep Autoencoding Gaussian mixture model for unsupervised anomaly detection," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–19.
[45] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *Proc. 28th Int. Conf. Artif. Neural Netw.*, 2019, pp. 703–716.
[46] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: UnSupervised anomaly detection on multivariate time series," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2020, pp. 3395–3404.
[47] D. Campos et al., "Unsupervised time series outlier detection with diversity-driven convolutional ensembles," *Proc. VLDB Endow.*, vol. 15, no. 3, pp. 611–623, 2021.
[48] S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep transformer networks for anomaly detection in multivariate time series data," *Proc. VLDB Endow.*, vol. 15, no. 6, pp. 1201–1214, 2022.
[49] Z. Liu, X. Huang, J. Zhang, Z. Hao, L. Sun, and H. Peng, "Multivariate time-series anomaly detection based on enhancing graph attention networks with topological analysis," in *Proc. 33rd ACM Int. Conf. Inf. Knowl. Manag.*, 2024, pp. 1555–1564.
[50] A. Abdulaal, Z. Liu, and T. Lancewicki, "Practical approach to asynchronous multivariate time series anomaly detection and Localization," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2021, pp. 2485–2494.
[51] L.-r. Yu, Q.-h. Lu, and Y. Xue, "DTAAD: Dual Tcn-attention networks for anomaly detection in multivariate time series data," *Knowl. Based Syst.*, vol. 295, Jul. 2024, Art. no. 111849.
[52] S. Yang et al., "Spatial-temporal interaction decoding transformer for unsupervised multivariate time series anomaly detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2024, pp. 5440–5444.

Sanli Zhu received the Ph.D. degree in electric engineering from the Chongqing University, Chongqing, China, in 2019.

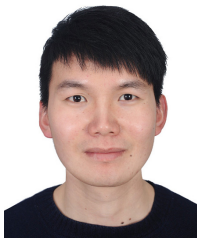
He is a Lecturer with the College of Automation and Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing, China. His research topics are industrial control network security technology.

Yuan Li is currently pursuing the master's degree with School of Automation, Nanjing University of Science and Technology, Nanjing, China.

His research interests include Knowledge graph and AIops.

Kang Xu received the Ph.D. degree in software engineering from the Southeast University, Nanjing, China, in 2017.

He is currently a Lecturer with Nanjing University of Posts and Telecommunications, Nanjing, China. His research interests include Knowledge graph, AIops, natural language processing, and LLMs.



Junjun Xu (Member, IEEE) received the B.S. degree in power system and its automation from Nanjing Institute of Technology, Nanjing, China, in 2012, the M.S. degree in agricultural electrification and automation from the School of Electrical and Information Engineering, Jiangsu University, Zhenjiang, China, in 2015, and the Ph.D. degree in electrical engineering from Southeast University, Nanjing, China, in 2019.

He is an Associated Professor with the College of Automation and Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing, China. His research interests include distribution network state estimation, self-healing control, and uncertainty modeling approaches.