Frequency-Domain Spectrum Discrepancy-Based Fast Anomaly Detection for IIoT Sensor Time-Series Signals

Lei Chen, *Member, IEEE*, Xuxin Liu, Ying Zou, Jiajun Tang, Canwei Liu, Bowen Hu, and Mingyang Lv

***Abstract*— Driven by Industry 4.0, anomaly detection for the Industrial Internet of Things (IIoT) has new demands: high accuracy, high speed, and low resource consumption. However, most existing models prioritize accuracy by constructing deep structures with huge parameters, often neglecting timeliness and resource efficiency in resource-limited IIoT environments. To solve these limitations, a nonneural-network-based sensor signal fast anomaly detection model, named FADSD, is proposed for resource-limited IIoT scenarios. Unlike traditional bulky neural-network solutions, FADSD only uses spectrum discrepancy in the frequency domain, a nonneural-network approach, to meet these three demands. First, the spectrum discrepancy is introduced to break the dilemma of not being able to perform timestamp-level feature extraction and anomaly detection in the frequency domain. Second, both point-level and sequence-level computation components are developed as complementary branches to generate the spectrum discrepancies for each timestamp. Finally, a novel anomaly scoring strategy combines point-level and sequence-level spectrum discrepancies for more accurate anomaly detection. To the best of authors' knowledge, this is the first work to deploy anomaly detection exclusively in the frequency domain. Extensive experiments on eight IIoT sensor signal datasets demonstrate that FADSD outperforms multiple state-of-the-art (SOTA) deep models and can be easily deployed in resource-limited IIoT environments. The source code of FADSD is available at https://github.com/infogroup502/FADSD.**

***Index Terms*— Anomaly detection, fast anomaly detection, frequency domain, Industrial Internet of Things (IIoT), sensor signal, spectrum discrepancy.**

I. Introduction

ANOMALY detection plays a crucial role in ensuring the security and stability of the Industrial Internet of Things (IIoT) [1]. Based on the granularity of anomaly detection, existing methods can be primarily categorized into three types, collective-level, subsequence-level, and timestamp-level detection [2], [3]. Collective-level detection determines whether an entire time series is abnormal. Subsequence-level detection determines whether a time interval consisting of multiple consecutive timestamps is abnormal. Timestamp-level detection, in contrast, assesses whether an individual timestamp is abnormal. With the growing demand for fine-grained intelligent control, timestamp-level anomaly detection has become a hot topic in the research area and also the focus of this article [4].

A. Challenge

Existing timestamp-level IIoT anomaly detection methods mainly focus on developing high-accuracy neural-network models without the aid of supervised labels [2]. To achieve high accuracy, these models make an effort to build bulky neural networks with numerous parameters and deep structures. Inevitably, these models require long training/testing times and consume many resources. When these models are deployed on a cloud with powerful resources, their detection time and resource consumption will not be a concern. Coming to Industry 4.0, large-volume sensor signals, intensive resource scheduling, and frequent computing tasks bring high latency issues for anomaly detection in the cloud. Consequently, anomaly detection models are increasingly being deployed on edge devices with limited computational resources, a paradigm known as edge computing. In this case, it introduces two key challenges for timestamp-level anomaly detection [5].

1) *Neural Network-Based Models May Not Be Deployable and Run Stably:* Unlike servers or clouds, IIoT edge devices have very limited resources. For example, an MCU-based STM32 microcontroller only has <1-MB SRAM and <168-MHz clock speed. Even the high-performance Raspberry Pi 4b has only 2-GB RAM and 1.5-GHz clock speed. These limited resources render neural-network-based models inoperable or unable to run stably on such devices.
2) *Tradeoff Between High Accuracy and High Speed:* Even if the resources of an edge device are sufficient to support deep neural-network-based models, these models face the tradeoff between high speed and high accuracy. Specifically, bulky neural-network models with

Received 15 September 2024; revised 17 January 2025; accepted 5 February 2025. Date of publication 24 March 2025; date of current version 3 April 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62103143 and Grant 62203164, in part by Hunan Provincial Natural Science Foundation of China under Grant 2024JJ5162, in part by the Young Backbone Teacher of Hunan Province under Grant 2022101, and in part by the Scientific Research Fund of Hunan Provincial Education Department under Grant 22B0471. The Associate Editor coordinating the review process was Dr. Dong Wang. *(Corresponding author: Lei Chen.)*

Lei Chen, Xuxin Liu, Ying Zou, Jiajun Tang, Bowen Hu, and Mingyang Lv are with the School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan 411201, China (e-mail: chenlei@hnust.edu.cn; xuxinliu@mail.hnust.edu.cn; 1040134@hnust.edu.cn; jiajuntang@mail.hnust.edu.cn; hubowen@mail.hnust.edu.cn; lvapple@hnust.edu.cn).

Canwei Liu is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: canweiliu@csu.edu.cn).

Digital Object Identifier 10.1109/TIM.2025.3554286

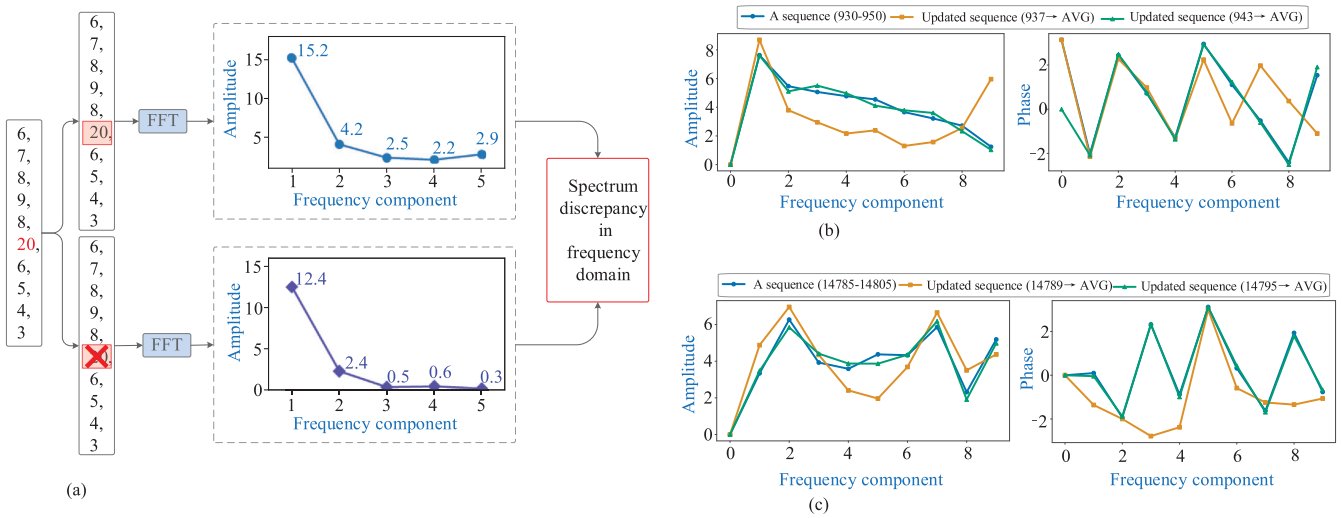


Fig. 1. Spectrum discrepancy in the frequency domain. (a) Illustration of amplitude discrepancy in the frequency domain for a timestamp. (b) Comparison of the amplitude and phase discrepancies between a normal (943 timestamps) and an outlier (937 timestamps) of the Yahoo dataset. (c) Comparison of the amplitude and phase discrepancies between a normal (14 795 timestamps) and an outlier (14 789 timestamps) of the second variable in the SMTP dataset.

deep structures and extensive parameters achieve high accuracy. However, it requires substantial computation time, particularly on resource-constrained edge devices. Conversely, lightweight models with shallow architectures and fewer parameters offer faster detection speeds with a reduction of reduced accuracy.

B. Existing Solution

To address the abovementioned challenges, some studies [6], [7] propose some lightweight neural networks that adapt to limited resources while maintaining accuracy. However, these models require frequent updates and redeployment in the cloud to sustain their accuracy. Other research [8], [9] constructs high-accuracy anomaly detection models by integrating multiple nonneural-network techniques. These models offer advantages such as online updates or operations without updates. Nevertheless, they often exhibit unstable detection accuracy due to limited representation capabilities. In summary, while nonneural-network solutions are more suitable for real-world industrial edge devices with constrained resources, further optimization is necessary to enhance their representation capabilities and detection stability.

Recent studies [10], [11] reveal that the complex feature patterns of sensor signals in the time domain can always be simplified and effectively identified in the frequency domain. Consequently, frequency-domain analysis, such as fast Fourier transform (FFT) and discrete wavelet transform (DWT), emerges as a promising nonneural-network technology for timestamp-level anomaly detection, which provides both strong representation ability and fast processing speed. However, existing frequency-domain analysis-based nonneural-network models mainly focus on collective-level [12] or subsequence-level anomaly detection [13]. Technically, they use FFT or DWT to transform an entire time series or a subsequence into the frequency domain to extract spectral features. These spectral features are then used to determine whether the time series or subsequence is abnormal. However, FFT or DWT cannot directly perform timestamp-level anomaly detection. It is because FFT or DWT can only work when the input is an entire time series or a subsequence, whereas they cannot work when the input is a timestamp. It means that a timestamp in the time domain cannot be directly mapped to the frequency domain to get its frequency-domain features. To take advantage of the frequency domain, existing timestamp-level anomaly detection models [11], [14] incorporate the frequency-domain features of a time series or its subsequences as auxiliary information alongside time-domain data. Neural networks are then employed to extract the features of each timestamp from this combined time–frequency-domain information and determine whether a timestamp is abnormal. Such models are, namely, time–frequency anomaly detection models.

To the best of authors' knowledge, there is no nonneural-network approach that has been developed to perform timestamp-level anomaly detection solely within the frequency domain.

C. New Insight

To fill such a gap, this article aims to design a purely frequency-domain-based and nonneural-network timestamp-level anomaly detection model for resource-limited IIoT environments.

To achieve this objective, two critical tasks should be taken into consideration. First, the feature of a timestamp in the frequency domain must be measured. In this article, the frequency-spectrum (amplitude or phase) discrepancy with and without a timestamp is considered as the feature of this timestamp in the frequency domain, as shown in Fig. 1(a). It can be seen that the two sequences, with and without the red-marked timestamp, form two comparison branches in the figure. The spectrum discrepancy between the two sequences is regarded as the feature of the red-marked timestamp. Second, it is crucial to verify whether this measurement method can effectively distinguish outliers from normal points. To verify

the effectiveness, a sequence of length 20 containing at least one outlier is selected from two real-world datasets, Yahoo and SMTP. Next, the amplitude and phase discrepancies of a normal point and an outlier from two datasets are plotted in Fig. 1(b) and (c), respectively. By observation, it is not difficult to find that the spectrum discrepancy can distinguish outliers from normal points.

D. Contribution

Based on the above discussions, a fast and nonneural-network timestamp-level anomaly detection model is proposed for IIoT sensor signals via spectrum discrepancy in the frequency domain. The main contributions of this work are summarized as follows.

1) Spectrum discrepancy is introduced to measure timestamp-level features in the frequency domain, which overcomes the difficulty of timestamp-level feature extraction and anomaly detection only in the frequency domain.
2) A nonneural-network fast anomaly detection model, namely, FADSD, is proposed for IIoT sensor signals. Technically, FADSD designs customized point-level and sequence-level learning components, and a spectrum discrepancy-based scoring strategy. They work together to achieve anomaly detection with high speed, high accuracy, and low resource consumption. To the best of authors' knowledge, this is the first work to deploy timestamp-level anomaly detection exclusively in the frequency domain.
3) Extensive experiments on eight real-world IIoT datasets demonstrate that FADSD outperforms several state-of-the-art (SOTA) deep models. Moreover, experiments on STM32 and Raspberry Pi 4b verify that FADSD has high speed and is easily deployable on IIoT edge devices with limited resources.

II. Related Work

The related works in the three categories relevant to our model are summarized in Table I.

TABLE I
RELATED WORK

	Model	Backbone	C1	C2	C3	C4
Neural-network	**TMANomaly**	GAN	✓	×	×	×
	AnomalyTrans	Transformer	✓	×	×	×
	DCdetector	Attention	✓	×	×	×
	DTAAD	TCN+Attention	✓	×	×	×
	PAFormer	Transformer	✓	✓	×	×
	DAD	CNN+LSTM	✓	✓	×	×
	TinyAD	CNN	✓	✓	×	×
	ANNet	MLP+LSTM	✓	✓	×	×
	FCVAE	CAVE	✓	×	×	✓
	KreqGAN	GAN	✓	×	×	✓
	TFAD	TCN	✓	×	×	✓
	ATF-UAD	CNN+GCN	✓	×	×	✓
No-neural-network	**Pascoal et al.**	PCA	×	✓	✓	×
	Optiforest	Isolation forest	×	✓	✓	×
	E-FCM	FCM+PSO	×	✓	✓	×
	MDS_AD	LSH+PCA	×	✓	✓	×
	LSTM-Markov	LSTM+Markov	×	✓	✓	×
	DIF	Isolation forest	×	✓	✓	×

C1: Whether it has high accuracy;
C2: Whether it has high speed;
C3: Whether it has low resource consumption;
C4: Whether the frequency domain features are used.

A. Neural-Network-Based Anomaly Detection

Unsupervised neural-network-based deep models have become the mainstream approaches in recent years because they do not rely on labeled data and have strong representation capabilities. These models are typically designed for deployment on cloud servers with powerful resources, so they only focus on high accuracy by constructing deep structures with huge parameters. For example, the TMANomaly model [15] designs a mutual adversarial network to effectively capture the temporal features in IIoT sensor signals. The AnomalyTrans model [16] employs multilayer Transformer network to learn association discrepancies as anomaly scores. Based on this, the DCdetector model [17] develops a two-branch contrasting network with deep structures and numerous parameters. Similarly, the DTAAD model [18] integrates the deep TCN network with a computation-sensitive attention mechanism. However, these bulky models often require a lot of GPU/CPU and storage resources, and they require long training time and testing time. Therefore, their timeliness is compromised, with processing times potentially extending to several hours or even days for large-volume sensor signals.

To enhance timeliness, acceleration techniques such as parallelization, federated learning, and model compression are often employed to optimize existing deep models. For example, the PAFormer model [19] parallelizes the attention mechanism and Transformer network to improve timeliness. The DAD model [20] employs a lightweight CNN and a parallelized two-stage LTSM autoencoder to quickly detect IoT anomalies. The TinyAD model [6] designs multiple strategies to compress the parameters of the convolutional neural network (CNN). The ANNet model [7] uses a lightweight multilayer perceptron (MLP) network with fewer parameters to replace computationally intensive LSTM cells with huge parameters. Although these models achieve high accuracy and high speed, they still carry numerous parameters, requiring a lot of CPU/GPU and storage resources. As a result, they cannot be directly deployed on edge devices with limited resources.

B. Nonneural-Network-Based Anomaly Detection

Nonneural-network models employ statistical, clustering, and machine learning techniques to identify anomalies in IIoT sensor signals. These models do not have deep structures and large parameters, so they are very fast and take up only a few resources. For instance, Pascoal et al. [21] use the principal component analysis (PCA) to select distinguishable features for anomaly detection. The LODA model [22] aggregates multiple weak anomaly detectors to build a strong anomaly

detector. The optiforest model [23] optimizes the traditional isolation forest method to identify anomalies in IIoT. However, these models often exhibit poor representation capabilities, which result in unsatisfactory accuracy when encountering sensor signals with complex features.

To improve model accuracy, multiple nonneural-network technologies are integrated to build a hybrid anomaly detection model. For example, the extend FCM model [8] combines fuzzy C-means and PSO to enhance anomaly detection accuracy. The MDS_AD model [9] incorporates locality-sensitive hashing (LSH), isolation forest, and PCA techniques. Similarly, the LSTM-MARKOV model [24] integrates Markov processes into the LSTM network. The DIF model [25] uses a shallow neural network to map raw data into a feature space and then employs the isolation forest method for anomaly detection in this new space. Although these models show improved accuracy, they are still not as accurate as deep models. Therefore, these models cannot perfectly meet the accuracy requirements of IIoT anomaly detection in the context of Industry 4.0.

C. Time–Frequency-Based Anomaly Detection

Compared to the time domain, it is always easy to identify the complex feature of signals from sensors in the frequency domain. However, traditional frequency-domain analysis techniques, such as FFT and DWT, are unable to directly extract features from a single timestamp. Thus, these techniques cannot be directly applied to timestamp-level anomaly detection. To overcome this limitation, time–frequency-based models for timestamp-level anomaly detection have been developed. These models utilize frequency-domain features, such as amplitude or phase, derived from an entire time series or its subsequences to complement time-domain data. Subsequently, neural networks are employed to extract features of a single timestamp from the mixed time–frequency information, thereby achieving timestamp-level anomaly detection. Therefore, most existing time–frequency-based models are neural-network-based anomaly detection models. For example, the FCVAE model [11] extracts high-frequency and low-frequency features from the frequency domain and concatenates them with the raw data to enhance time-domain features for anomaly detection. Similarly, the KreqGAN model employs adversarial learning to identify anomalies from concatenated time–frequency information. The TFAD model [10] employs TCN networks to learn features from the combined time–frequency information. Furthermore, the Dual-TF model [26] and the TF-C model [27] design different mechanisms to align the time domain and frequency domain. Additionally, the ATF-UAD model [14] integrates CNN, GCN, and adversarial learning to construct time and frequency reconstructors for learning features of the concatenated time–frequency information. To the best of authors' knowledge, there has been no work that performs timestamp-level anomaly detection only in the frequency domain.

D. Summary

In Industry 4.0, anomaly detection models for IIoT must meet new requirements: high accuracy, high speed, and low resource consumption. Driven by these new requirements, traditional popular cloud deployment solutions are beginning to be replaced by edge deployment solutions. In this case, it is valuable to explore the construction of a nonneural-network timestamp-level anomaly detection model based only on frequency-domain analysis.

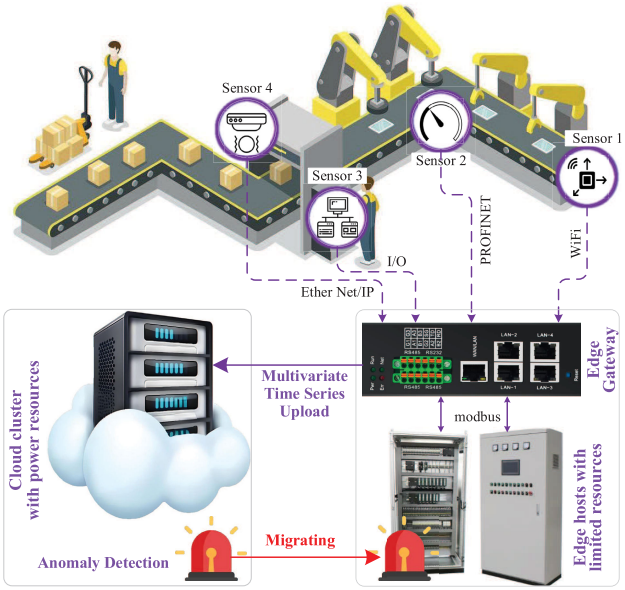


Fig. 2. IIoT environment.

III. Preliminary

A. IIoT Environment

The IIoT facilitates real industrial production processes by enabling remote monitoring and control, as illustrated in Fig. 2. Specifically, various heterogeneous smart sensors, including vibration, pressure, and oil sensors, are deployed at key locations within the production process to monitor the operational status of equipment. The data collected by these sensors are transmitted to edge gateways and resource-constrained edge hosts using various network technologies (Wi-Fi/5G/Ethernet), and protocols (Modbus/UDP/TCP). Subsequently, the edge gateway forwards this data to the cloud with power resources for further monitoring and control.

In this article, we consider an industrial production process employing M sensors for data collection. After T collection intervals, the resulting sensor signal data form a multivariate time series. This time-series data is formally expressed as $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\} \in \mathbb{R}^{M\times T}$, where T denotes the number of timestamps and M represents the number of variables (sensors). The data comprise two dimensions: the time dimension and the variable (sensor) dimension. As for the time dimension, $\mathbf{X} = \{x_t \in \mathbb{R}^{M\times 1}\}_{t=1}^{T}$, where x_t represents the values of M sensors at the tth timestamp. As for the variable dimension, $\mathbf{X} = \{x^m \in \mathbb{R}^{1\times T}\}_{m=1}^{M}$, where x^m represents values from T timestamps at the mth sensor.

B. Problem Statement

Traditionally, anomaly detection models are deployed on the cloud with powerful resources. However, driven by the

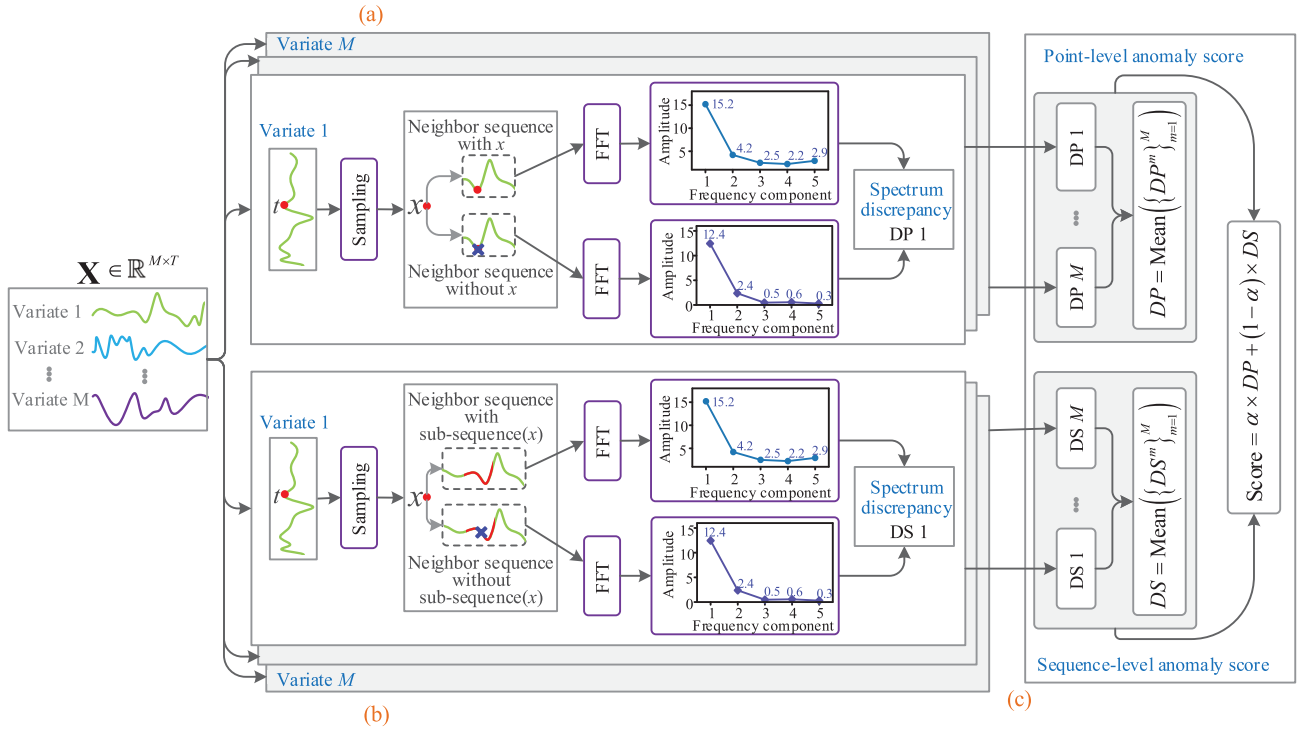


Fig. 3. Overview of the FADSD model. (a) Point-level spectrum discrepancy calculation. (b) Sequence-level spectrum discrepancy calculation. (c) Spectrum discrepancy-based anomaly scoring.

need for fine-grained monitoring, anomaly detection models have begun to migrate from the cloud to resource-limited edge hosts for the goal of real-time detection. Therefore, this article aims to construct a nonneural-network fast anomaly detection model for IIoT sensor signals. First, the anomaly detection model generates an anomaly score for each timestamp in $\mathbf{X}$, which is formulated as follows:

$$\text{Score}_t = \text{FADSD}(\boldsymbol{x}_t) \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^M$ denotes the values of M variables at tth timestamp. Second, the anomaly score is used to classify a timestamp as an anomaly or not, which is defined as

$$\bar{y}_t = \begin{cases} 1, & \text{if Score}_t \geq \lambda \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where λ is a preset threshold ranging from 0 to 1. If $\bar{y}_t = 1$, the tth timestamp is classified as an anomaly; otherwise, it is considered normal. Finally, the predicted labels for all timestamps are expected to align closely with the true labels.

IV. Proposed Model

A. Overview

In this article, we replace traditional deep neural-network-based solutions by developing a nonneural-network anomaly detection model only in the frequency domain. Additionally, we introduce spectrum discrepancy to measure the importance of each timestamp in the frequency domain, which can circumvent the inherent barrier of the inability to extract timestamp-level features in the frequency domain. In summary, a timestamp-level anomaly detection model, namely, FADSD, is designed to achieve high-speed and high-accuracy anomaly detection on resource-limited IIoT edge devices. To the best of authors' knowledge, this is the first work to deploy timestamp-level anomaly detection exclusively in the frequency domain. The overview of the FADSD model is shown in Fig. 3, which consists of three components.

1) *The first component* calculates the frequency-spectrum discrepancy for each timestamp from a point-level perspective, as shown in Fig. 3(a). Specifically, a customized point-level sampling strategy works with the FFT to generate point-level spectrum (amplitude or phase) discrepancies in the frequency domain. This component aims to accurately identify explicit anomalies in IIoT sensor signals.
2) *The second component* complements the first by calculating another frequency-spectrum discrepancy from a sequence-level perspective, as shown in Fig. 3(b). Specifically, a sequence-level sampling strategy is designed and combined with *FFT* to generate sequence-level amplitude or phase discrepancies. This component aims to identify implicit anomalies.
3) *The third component* combines the point-level and sequence-level frequency-spectrum discrepancies to generate a final anomaly score for each timestamp, as shown in Fig. 3(c). Based on this score, (2) is employed to decide whether the timestamp is an anomaly.

Note that the IIoT sensor signals $\mathbf{X}$ need to be instance normalized [17] before being processed by our model. Additionally, the point-level and sequence-level spectrum discrepancy calculations are independent and can be executed in parallel. In both components, the calculation process of M variables is also performed independently and in parallel.

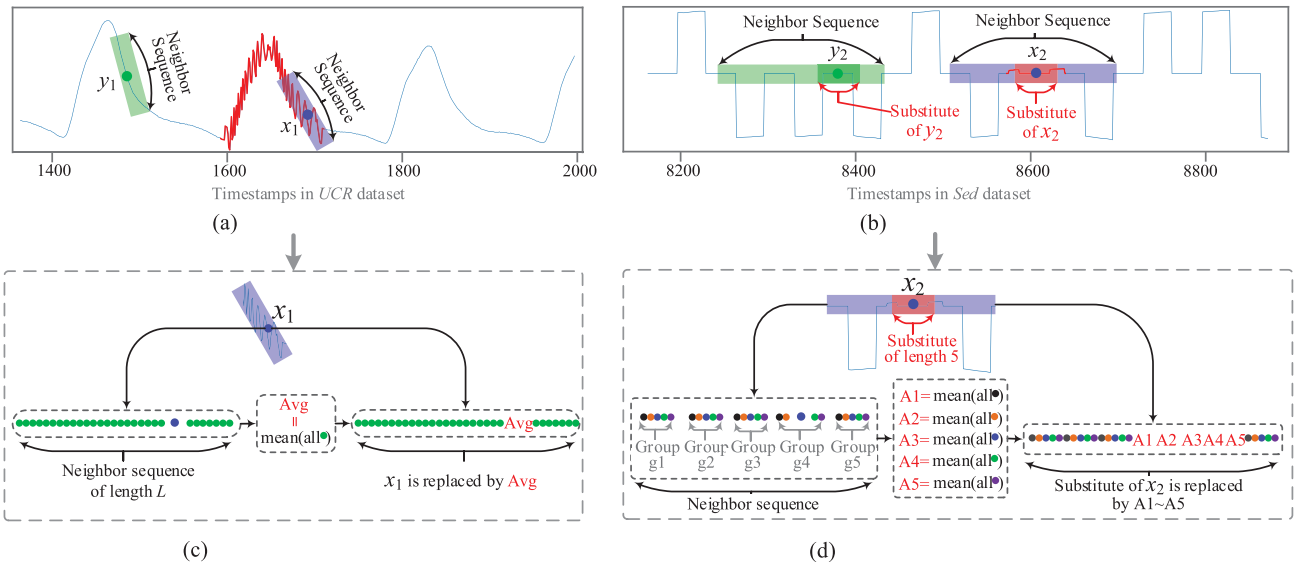


Fig. 4. Point-level and sequence-level sampling. (a) Explicit outlier. (b) Implicit outlier. (c) Point-level sampling. (d) Sequence-level sampling.

B. Point-Level Spectrum Discrepancy Calculation

Anomalies in IIoT sensor signals generally fall into two categories: explicit anomalies and implicit anomalies. These anomalies often persist over multiple timestamps. If fine-grained timestamp-level anomaly detection is performed, timely warnings can be issued at the early stage of an anomaly, thereby ensuring the security and stability of IIoT systems. In this section, we focus on designing a component to perform timestamp-level anomaly detection in the frequency domain for explicit anomalies.

In an explicit anomaly, the values of each timestamp are significantly different from each other and from normal timestamps. As illustrated in Fig. 4(a), the timestamps between 1600 and 1700 in the 136th variable of the real-world UCR dataset present an explicit anomaly. In the figure, x_1 represents an abnormal timestamp and y_1 represents a normal timestamp. When comparing these two timestamps within neighbor sequences of the same length (purple region for x_1 and green region for y_1), x_1 is significantly different from its neighbors, while y_1 is relatively smooth with its neighbors. This implies that x_1 is not similar to its neighbors, whereas y_1 is more similar to its neighbors. Therefore, the similarity of a timestamp with its neighbors is a good indicator to accurately identify explicit anomalies in the frequency domain. For this goal, we need to get the features of each timestamp in the frequency domain.

However, an inherent obstacle arises: the features of a timestamp cannot be directly measured in the frequency domain. This obstacle stems from the lack of a one-to-one correspondence between the time domain and the frequency domain. For example, a sensor signal of length L in the time domain is transformed by FFT into a superposition of $L/2$ frequency components in the frequency domain. As a result, the features of a specific timestamp are dispersed across these $L/2$ frequency components, making direct measurement impossible. To solve this problem, we convert from measuring the features of each timestamp to evaluating its replaceability in the frequency domain. If a timestamp is not replaceable in its neighbor sequence, it is not similar to its neighbors. On the contrary, it is similar to its neighbors. Therefore, spectrum discrepancy is introduced to quantify the replaceability of each timestamp within its neighbor sequence. The core idea is to observe the amplitude or phase discrepancy of the same neighbor sequence with or without a timestamp, as shown in Fig. 1(a).

Based on the analysis, a novel point-level spectrum discrepancy calculation component is designed to identify explicit anomalies in the frequency domain.

1) Point-Level Sampling: To calculate the spectrum discrepancy, a variable-independent point-level sampling strategy is designed to generate a neighbor sequence for each timestamp and a replacement sequence without it, as shown in Fig. 4(c). Taking the tth timestamp of the mth variable as an example, the generation process for these two sequences is outlined as follows.

1) *Neighbor Sequence Generation:* Timestamps from the range $(t - L/2)$ to $(t + L/2)$ are extracted from the mth variable $x^m \in \mathbb{R}^{1\times T}$ to form the neighbor sequence $P_t^m \in \mathbb{R}^{1\times L}$.
2) *Replacement Sequence Generation:* The average Avg of all timestamps in P_t^m except the tth timestamp is calculated. The value at the tth timestamp in P_t^m is then replaced by the Avg. The generated sequence is the replacement sequence $P_{\text{no}-t}^m \in \mathbb{R}^{1\times L}$.

Finally, these two sequences of length L are used to assess the replaceability of timestamp t in the frequency domain. It is important to note that the sampling processes for the M variables and the timestamps within each variable can be executed in parallel.

2) Spectrum Discrepancy Calculation: Once the two neighbor sequences are generated, the spectrum discrepancy calculation is very simple for each timestamp. We use the tth timestamp of the mth variable as an example to outline the detailed calculation process in the frequency domain.

First, the FFT is employed to transform two neighbor sequences P_t^m and $P_{\text{no}-t}^m$ of length L into $L/2$ frequency components, respectively,

$$\begin{cases} \text{FP}_t^m = \text{FFT}\left(P_t^m\right) \\ \text{FP}_{\text{no}-t}^m = \text{FFT}\left(P_{\text{no}-t}^m\right) \end{cases} \tag{3}$$

where $[\text{FP}_t^m, \text{FP}_{\text{no}-t}^m] \in \mathbb{C}^{1\times(L/2)}$ and $[P_t^m, P_{\text{no}-t}^m] \in \mathbb{R}^{1\times L}$

Second, the amplitudes or phases of FP_t^m and $\text{FP}_{\text{no}-t}^m$ are calculated as the frequency-domain features of P_t^m and $P_{\text{no}-t}^m$

$$\begin{cases} \text{AP}_t^m = \text{abs}\left(\text{FP}_t^m\right), & \text{AP}_{\text{no}-t}^m = \text{abs}\left(\text{FP}_{\text{no}-t}^m\right) \\ \text{PP}_t^m = \text{phase}\left(\text{FP}_t^m\right), & \text{PP}_{\text{no}-t}^m = \text{phase}\left(\text{FP}_{\text{no}-t}^m\right) \end{cases} \tag{4}$$

where $[\text{AP}_t^m, \text{AP}_{\text{no}-t}^m] \in \mathbb{R}^{1\times(L/2)}$ and $[\text{PP}_t^m, \text{PP}_{\text{no}-t}^m] \in \mathbb{R}^{1\times(L/2)}$. Here, abs$(\cdot)$ denotes the amplitude calculation function, and phase$(\cdot)$ denotes the phase calculation function.

Finally, the mean-square-error (mse) function is used to measure the amplitude or phase discrepancy in the frequency domain for the tth timestamp

$$\text{DP}_t^m = \begin{cases} \text{MSE}\left(\text{AP}_t^m, \text{AP}_{\text{no}-t}^m\right) \\ \text{MSE}\left(\text{PP}_t^m, \text{PP}_{\text{no}-t}^m\right) \end{cases} \tag{5}$$

where $\text{DP}_t^m \in \mathbb{R}$.

It is worth noting that the amplitude or phase discrepancy calculations for each timestamp in the M variables can be performed in parallel for faster speed.

C. Sequence-Level Spectrum Discrepancy Calculation

Unlike explicit anomalies, implicit anomalies exhibit more complex patterns. In such anomalies, some timestamps differ significantly from normal timestamps, while others closely resemble them. For instance, the timestamps between 8580 and 8640 in the real-world *Sed* dataset present an implicit anomaly, as shown in Fig. 4(b). In the figure, x_2 represents an abnormal timestamp and y_2 is a normal timestamp. Notably, x_2 has the same value as y_2, and their local neighbor sequences are very similar. In this case, the replaceability of x_2 and y_2 in their respective neighbor sequences is also very similar. This similarity makes the previously proposed point-level spectrum discrepancy approach ineffective for accurately identifying anomalies like x_2.

To solve this problem, the most straightforward approach is to make x_2 and y_2 significantly distinguishable in their neighbor sequences. Thus, we extract multiple local neighbors as the substitute sequence of a timestamp. This approach introduces significant differences between the substitute sequences of initially similar timestamps. It also means that the replaceability of these substitute sequences differs within their similar neighbor sequences. For example, we use the dark red subsequence as the substitute for x_2 and the dark green subsequence as the substitute for y_2. As a result, the replaceability of the subsequences for x_2 and y_2 within their respective sequence-level neighbor sequences (long purple and long green areas) becomes significantly different.

Based on this analysis, a sequence-level spectrum discrepancy calculation component is designed to identify implicit anomalies in the frequency domain from a sequential perspective.

1) Sequence-Level Sampling: Similar to the point-level sampling, a variable-independent sequence-level sampling strategy is designed to generate a neighbor sequence for the substitute of each timestamp and a replacement sequence without the substitute, as shown in Fig. 4(d). Taking the tth timestamp of the mth variable as an example, the sampling process is expressed as follows.

1) *Neighbor Sequence Generation:* Timestamps in the range $(t - \text{LS}/2)$–$(t + \text{LS}/2)$ are extracted from the mth variable $x^m \in \mathbb{R}^{1\times T}$ to form the first neighbor sequence $S_t^m \in \mathbb{R}^{1\times \text{LS}}$.
2) *Grouping:* All data in S_t^m are sequentially divided into G groups, each containing LG timestamps, where $\text{LS} = G \times LG$.
3) *Substitute Generation:* The group where timestamp t is located is considered its substitute, with a length of LG.
4) *Replacement Sequence Generation:* The value of each timestamp in the substitute group is replaced by the average of the corresponding positions in the other groups to generate the replacement sequence $S_{\text{no}-t}^m \in \mathbb{R}^{1\times \text{LS}}$, which excludes the substitute of the tth timestamp.

Finally, these two sequences of length LS are used to measure the sequence-level replaceability of timestamp t in the frequency domain.

2) Spectrum Discrepancy Calculation: Similar to the point-level calculation, the sequence-level spectrum discrepancy calculation for each timestamp is straightforward. Taking the tth timestamp of the mth variable as an example, the calculation process is described as follows.

First, the FFT is used to transform two neighbor sequences S_t^m and $S_{\text{no}-t}^m$, each of length LS, into $LS/2$ frequency components

$$\begin{cases} \text{FS}_t^m = \text{FFT}\left(S_t^m\right) \\ \text{FS}_{\text{no}-t}^m = \text{FFT}\left(S_{\text{no}-t}^m\right) \end{cases} \tag{6}$$

where $[\text{FS}_t^m, \text{FS}_{\text{no}-t}^m] \in \mathbb{C}^{1\times(\text{LS}/2)}$ and $[S_t^m, S_{\text{no}-t}^m] \in \mathbb{R}^{1\times \text{LS}}$.

Second, the amplitudes or phases of FS_t^m and $\text{FS}_{\text{no}-t}^m$ are calculated as

$$\begin{cases} \text{AS}_t^m = \text{abs}\left(\text{FS}_t^m\right), & \text{AS}_{\text{no}-t}^m = \text{abs}\left(\text{FS}_{\text{no}-t}^m\right) \\ \text{PS}_t^m = \text{phase}\left(\text{FS}_t^m\right), & \text{PS}_{\text{no}-t}^m = \text{phase}\left(\text{FS}_{\text{no}-t}^m\right) \end{cases} \tag{7}$$

where $[\text{AS}_t^m, \text{AS}_{\text{no}-t}^m] \in \mathbb{R}^{1\times(\text{LS}/2)}$ and $[\text{PS}_t^m, \text{PS}_{\text{no}-t}^m] \in \mathbb{R}^{1\times(\text{LS}/2)}$.

Finally, the mse function is employed to measure sequence-level spectrum discrepancy for each timestamp

$$\text{DS}_t^m = \begin{cases} \text{MSE}\left(\text{AS}_t^m, \text{AS}_{\text{no}-t}^m\right) \\ \text{MSE}\left(\text{PS}_t^m, \text{PS}_{\text{no}-t}^m\right) \end{cases} \tag{8}$$

where $\text{DS}_t^m \in \mathbb{R}$ is the sequence-level spectrum discrepancy of the tth timestamp.

Similar to the point-level calculation, the sequence-level spectrum discrepancy calculation for each timestamp in the M variables can be performed in parallel to enhance computational efficiency.

D. Spectrum Discrepancy-Based Anomaly Scoring

In this section, a spectrum discrepancy-based scoring strategy is proposed to combine point-level and the sequence-level perspectives for more accurate anomaly detection. The scoring process for the tth timestamp is illustrated in Fig. 3(c) and is summarized as follows.

First, the tth timestamp $\boldsymbol{x}_t \in \mathbb{R}^{M\times 1}, t \in (1,\ldots,T)$ is simultaneously input into the point-level and sequence-level spectrum discrepancy calculation components. Second, each of M variables at tth timestamp independently calculate its spectrum discrepancy in both two components. For each variable $m \in (1,\ldots,M)$, the point-level and sequence-level spectrum discrepancies, $\mathrm{DP}_t^m \in \mathbb{R}$ and $\mathrm{DS}_t^m \in \mathbb{R}$, are generated as the point-level and the sequence-level anomaly scores, respectively. Third, the point-level and sequence-level average anomaly scores for the M variables are calculated as $\mathrm{DP}(x_t) = \mathrm{mean}(\{\mathrm{DP}_t^m\}_{m=1}^M)$ and $\mathrm{DS}(x_t) = \mathrm{mean}(\{\mathrm{DS}_t^m\}_{m=1}^M)$, respectively. Finally, the final score for the tth timestamp is generated as

$$\mathrm{Score}_t = \alpha \times \mathrm{DP}(x_t) + (1-\alpha) \times \mathrm{DS}(x_t) \tag{9}$$

where α is a preset parameter with the range of [0∼1]. When $\alpha = 0$, we only consider the sequence-level spectrum discrepancy. When $\alpha = 1$, only the point-level spectrum discrepancy is considered.

After getting the anomaly score, (2) is employed to determine whether the tth timestamp is an anomaly.

E. Algorithm and Analysis

The pseudocode of FADSD is shown in Algorithm 1.

Complexity: The computational costs of the FADSD model come from three sequential stages: spectrum discrepancy calculation, spectrum discrepancy-based anomaly scoring, and anomaly detention. In the first stage, the spectrum discrepancy calculation for M variables is performed in parallel, and point-level and sequence-level spectrum discrepancy calculation on a variable can also be performed in parallel. Specifically, the complexity of the point-level spectrum discrepancy calculation is $\mathcal{O}(T \times (\underbrace{L}_{\text{sampling}} + \underbrace{L \times \log_L}_{\text{FFT}}))$, where T is the length of the time series and L is the point-level neighbor length. Similarly, the complexity of the sequence-level spectrum discrepancy calculation is $\mathcal{O}(T \times (\underbrace{\mathrm{LS}}_{\text{sampling}} + \underbrace{\mathrm{LS} \times \log_{\mathrm{LS}}}_{\text{FFT}}))$, where T is the length of the time series and LS is the sequence-level neighbor length. Therefore, the complexity of spectrum discrepancy calculation is $\mathcal{O}(T \times (L + L \times \log_L + \mathrm{LS} + \mathrm{LS} \times \log_{\mathrm{LS}}))$, where $L, L \times \log_L, \mathrm{LS}, \mathrm{LS} \times \log_{\mathrm{LS}} \ll T$. In the second stage, the mean spectrum discrepancy of M variables is converted into an anomaly score of a timestamp, and its complexity is $\mathcal{O}(T \times M)$, where $M \ll T$. In the last stage, each timestamp is determined to be an anomaly, and its complexity is $\mathcal{O}(T)$. In summary, ignoring relatively small factors (L, LS), the time complexity of our FADSD model is $\mathcal{O}(T \times (M + L \times \log_L + \mathrm{LS} \times \log_{\mathrm{LS}}))$, which is nearly linear because $(M + L \times \log_L + \mathrm{LS} \times \log_{\mathrm{LS}}) \ll T$.

Algorithm 1 FADSD

input : Time-series $\mathbf{X} \in \mathbb{R}^{M\times T}$; Parameters α and λ.
output: Anomaly labels for all timestamps $\bar{\mathbf{y}}$.

for $t \in \{1,\ldots,T\}$ **do**
  // **Spectrum discrepancy calculation**
  for $m \in \{1,\ldots,M\}$ **parallel do**
    // point-level discrepancy
    $P_t^m; P_{no-t}^m \leftarrow$ point-level sampling for x_t^m;
    $FP_t^m; FP_{no-t}^m \leftarrow \mathrm{FFT}(P_t^m; P_{no-t}^m)$ by Eq. (3);
    $AP_t^m; AP_{no-t}^m \leftarrow \mathrm{abs}(FP_t^m; FP_{no-t}^m)$ by Eq. (4);
    $DP_t^m \leftarrow \mathrm{MSE}(AP_t^m; AP_{no-t}^m)$ by Eq. (5);
    // sequence-level discrepancy
    $S_t^m; S_{no-t}^m \leftarrow$ sequence-level sampling for x_t^m;
    $FS_t^m; FS_{no-t}^m \leftarrow \mathrm{FFT}(S_t^m; S_{no-t}^m)$ by Eq. (6);
    $AS_t^m; AS_{no-t}^m \leftarrow \mathrm{abs}(FS_t^m; FS_{no-t}^m)$ by Eq. (7);
    $DS_t^m \leftarrow \mathrm{MSE}(AS_t^m; AS_{no-t}^m)$ by Eq. (8);
  end
  // **Anomaly scoring**
  $DP(x_t) = \mathrm{mean}(\{DP_t^m\}_{m=1}^M)$;
  $DS(x_t) = \mathrm{mean}(\{DS_t^m\}_{m=1}^M)$;
  $Score_t = \alpha \times DP(x_t) + (1-\alpha) \times DS(x_t)$;
  // **Anomaly detection**
  $\bar{y}_t = \begin{cases} 1, & \text{if } Score_t \geq \lambda \\ 0, & \text{otherwise} \end{cases}$
end
Return $\bar{\mathbf{y}}$;

V. Experiment

A. Experimental Goal

We conduct a series of experiments to address the following research questions.

1) *RQ1 (Accuracy):* Does the FADSD model outperforms other popular deep learning models?
2) *RQ2 (Deployability):* Can our model be easily deployed on resource-limited IIoT edge devices?
3) *RQ3 (Timeliness and Resource Consumption):* Does our model has good timeliness and low resource consumption?
4) *RQ4 (Robustness):* How well does our model performs in noisy environments?
5) *RQ5 (Ablation):* Does each innovative component of our model enhances overall performance?
6) *RQ6 (Sensitivity):* How do different parameters affect model performance?

B. Dataset

We select eight publicly available real-world time-series datasets from the IIoT environment, including two univariate and six multivariate datasets, as listed in Table II. Among these, MSL, GECCO, Genesis, HAI, SKAB, and SWAT are multivariate IIoT sensor signals, whereas Dodgers and UCR are univariate IIoT sensor signals. These datasets cover various industrial domains, including aviation, water treatment,

TABLE II
EIGHT PUBLIC DATASETS

Dataset	Features	Train Set	Test Set	URL
MSL	55	58317	73729	MSL-URL
GECCO	9	69260	69261	GECCO-URL
Genesis	18	16220	16220	Genesis-URL
HAI	86	132480	87840	HAI-URL
SKAB	8	135183	427617	SKAB-URL
SWAT	51	494999	449918	SWAT-URL
Dodgers	1	182319	45580	Dodgers-URL
UCR	1	2300	5200	UCR-URL

mechanical manufacturing, and transportation, to ensure a fair and unbiased evaluation.

Specifically, the MSL dataset captures the operating status of multiple sensors or controllers on the Mars rover. The GECCO dataset stores drinking water quality data from multiple IoT sensors. The Genesis dataset records five continuous signals and 13 discrete signals form IIoT multisensors. The SWAT dataset consists of 51-D data collected by sensors in public water treatment infrastructure. The HAI dataset contains multiple industrial signals from pumps, valves, pressure sensors, and other common industrial equipments. The Dodgers dataset consists of traffic flow data collected by induction loop sensors (loop sensors) on roads near Dodger Stadium in Los Angeles. The UCR dataset is a publicly available IIoT dataset containing only a single signal and one recorded anomaly.

C. Baseline, Metric, and Environment

1) Baseline: To validate the effectiveness of our FADSD model, we select nine representative SOTA models as competitors, as listed in Table III. These baselines are chosen based on their model structure (neural-network usage) and their focus on the time and frequency domains.

1) *Considering network structure*, DCdetector, ATF-UAD, FCVAE, DTAAD, and DIF are deep neural-network-based models with multilayer structures and numerous parameters. In contrast, IForest, PCA, Optiforest, LODA, and FADSD are nonneural-network-based shallow models.
2) *Regarding time and frequency domains*, DIF, Optiforest, DCdetector, LODA, PCA, IFoest, and DTAAD focus solely on the time domain for anomaly detection. And FCVAE and ATF-UAD combine time and frequency domains to perform anomaly detection, while FADSD exclusively uses the frequency domain.

2) Metric: To comprehensively evaluate our model, we employ four widely recognized metrics: Accuracy (ACC), Precision, Recall, and $F1$-score ($F1$). ACC measures the ratio of correctly predicted samples to the total sample count. *Precision* indicates the proportion of accurately predicted normal timestamps among all predicted normal timestamps. *Recall* reflects the proportion of actual normal timestamps that are correctly identified. The $F1$ provides a balanced evaluation by combining *Recall* and *Precision*. In general, these metrics range from 0 to 1, with higher values indicating better anomaly detection performance.

3) Experimental Environment: All experiments are conducted in a Windows 11 environment equipped with an Intel[1] Core[2] i7-10700KF CPU, an NVIDIA GeForce RTX 3090 GPU (24-GB VGA-RAM), and 64 GB of system memory. The FADSD model is developed in Python using PyCharm as the integrated development environment (IDE). The source code is publicly available at https://github.com/infogroup502/FADSD. For the other models, official Python implementations are obtained from the websites of respective authors, as detailed in Table III.

D. RQ1 (Accuracy)

To answer *RQ1*, we compare the FADSD against nine baselines across eight IIoT datasets.

Table IV presents the accuracy of ten models on eight IIoT datasets. In the table, the best performance is highlighted in bold, while the second-best performance is underlined. From the table, several findings can be observed.

1) *Overall Performance:* FADSD achieves the best performance, followed by DCdetector, FCVAE, ATF-UAD, DIF, and DTAAD, while Optiforest, LODA, IForest, and PCA show the weakest performance. For example, on the SWAT dataset, FADSD gets an ACC of 0.9924 and an $F1$ of 0.9688, whereas PCA only scores an ACC of 0.5761 and an $F1$ of 0.5852.
2) *Stability:* FADSD demonstrates the most stable performance across all four metrics, with DCdetector, DIF, and FCVAE following closely. For instance, on the HAI and SKAB datasets, FADSD achieves three first-place and one second-place stable results.
3) *Time and Frequency Domains:* The FADSD model outperforms other time–frequency-based models, such as FCVAE and ATF-UAD. On the MSL dataset, the ACC is 0.9926 and the $F1$ is 0.9378 for FADSD, while 0.9632 and 0.8440 for FCVAE, and 0.9223 and 0.7789 for ATF-UAD.
4) *Deep or Shallow Structures:* Deep neural-network-based models (DIF, DCdetector, FCVAE, ATF-UAD, DTAAD) generally outperform shallow models (IForest, PCA, Optiforest, LODA). However, the shallow FADSD exceeds multiple deep models. For example, on the SWAT dataset, FADSD achieves an $F1$-score of 0.9688, surpassing DCdetector's 0.9649, FCVAE's 0.9459, IForest's 0.4182, and LODA's 0.7469.
5) *In Summary:* The nonneural-network FADSD shows performance that demonstrates performance that rivals, and in some cases surpasses, deep models. This is because, for sensor signals, the features of abnormal patterns are easier to identify in the frequency domain than in the time domain. In this case, FADSD using a nonneural network directly in the frequency domain can achieve better accuracy than other SOTA models using deep neural network in the time domain.

[1]Registered trademark.
[2]Trademarked.

TABLE III
BASELINES

Models	Full Name	Year	Source Code
FCVAE [11]	Revisiting VAE for Unsupervised Time Series Anomaly Detection: A Frequency Perspective	2024	github.com/CSTCloudOps/FCVAE
DTAAD [18]	DTAAD: Dual Tcn-Attention Networks for Anomaly Detection in Multivariate Time Series Data	2024	github.com/Yu-Lingrui/DTAAD
DCdetector [17]	Dual Attention Contrastive Representation Learning for Time Series Anomaly Detection	2023	github.com/DAMO-DI-ML/KDD2023-DCdetector
DIF [25]	Deep isolation forest for anomaly detection	2023	github.com/xuhongzuo/deep-iforest
ATF-UAD [14]	An Adversarial Time-Frequency Reconstruction Network for Unsupervised Anomaly Detection	2023	github.com/wzhSteve/ATF-UAD
Optiforest [23]	OptIForest: Optimal Isolation Forest for Anomaly Detection	2023	github.com/xiagll/OptIForest
LODA [22]	LODA: Lightweight on-line detector of anomalies	2016	github.com/OpsPAI/MTAD
IForest [28]	Isolation Forest	2008	github.com/OpsPAI/MTAD
PCA [29]	A novel anomaly detection scheme based on principal component classifier	2003	github.com/OpsPAI/MTAD

TABLE IV
PERFORMANCE ANALYSIS ON EIGHT IIOT DATASETS

		FADSD	DIF	DCdetector	FCVAE	ATF-UAD	DTAAD	IForest	Optiforest	LODA	PCA
SWAT	ACC	**0.9924**	0.9466	0.9902	0.9906	0.9598	0.9598	0.6491	0.9024	0.9508	0.5761
	Precision	0.9447	0.9477	0.9301	0.9919	**1**	**1**	0.5818	0.7456	0.5975	0.4218
	Recall	**0.9980**	0.8036	0.9954	0.9041	0.6879	0.6879	0.2979	0.3012	0.8958	0.6559
	F1	**0.9688**	0.8697	0.9649	0.9459	0.8151	0.8151	0.4182	0.7350	0.7469	0.5852
GECCO	ACC	0.9952	0.9808	0.9844	**0.9970**	0.9928	0.9930	0.4862	0.7826	0.9850	0.6852
	Precision	**0.6945**	0.4862	0.3817	0.6160	0.6049	0.6159	0.3260	0.4118	0.2814	0.2853
	Recall	**0.9685**	0.7466	0.5955	0.9003	0.3521	0.3521	0.3400	0.9105	0.2814	0.2914
	F1	**0.8089**	0.5889	0.4661	0.7624	0.5069	0.5150	0.3329	0.4232	0.2755	0.2853
HAI	ACC	**0.9920**	0.9919	0.9919	0.9526	0.9446	0.9910	0.9030	0.8924	0.9489	0.5211
	Precision	0.8732	**0.8848**	0.8835	0.7016	0.4993	0.8544	0.3536	0.5785	0.3120	0.4048
	Recall	**1**	0.9664	0.9564	0.6492	**1**	**1**	0.2415	0.8290	0.5681	0.6536
	F1	**0.9323**	0.9238	0.9239	0.6744	0.6661	0.9285	0.2872	0.6727	0.4028	0.6869
SKAB	ACC	**0.9959**	0.9928	0.9928	0.8400	0.9168	0.9830	0.8046	0.8347	0.9534	0.8851
	Precision	**0.9886**	0.9787	0.9787	0.9816	0.8081	0.9537	0.6680	0.3475	0.7975	0.6630
	Recall	**1**	**1**	**1**	0.9998	**1**	**1**	0.7472	**1**	0.8640	0.8511
	F1	0.9943	0.9892	0.9892	**0.9957**	0.8938	0.9763	0.7054	0.5157	0.8294	0.7454
Genesis	ACC	**0.9950**	0.9276	0.9899	0.9828	0.9942	0.9944	0.6957	0.8738	0.8941	0.5929
	Precision	0.8154	0.4404	0.6154	0.6252	**0.8421**	0.8148	0.2874	0.5361	0.4200	0.3278
	Recall	0.9600	0.9514	0.9585	**1**	0.9478	0.9504	0.8913	0.9067	0.8039	0.9091
	F1	0.8547	0.6038	0.7500	0.3676	0.6972	**0.9796**	0.8542	0.7210	0.8119	0.7229
MSL	ACC	**0.9926**	0.8383	0.9905	0.9632	0.9223	0.9814	0.3764	0.8053	0.7298	0.4927
	Precision	**0.9841**	0.7992	0.9455	0.9393	0.6379	0.8807	0.8829	0.8053	0.4187	0.5858
	Recall	0.9761	0.9060	0.9681	0.7663	**1**	**1**	0.2854	0.9478	0.1743	0.3492
	F1	0.9378	0.8492	**0.9567**	0.8440	0.7789	0.9366	0.4314	0.1906	0.2461	0.4799
Dodgers	ACC	0.9414	**0.9549**	**0.9549**	0.9044	0.8686	0.7908	0.8742	0.9402	0.1853	0.6418
	Precision	**0.9325**	0.8854	0.8854	0.8663	0.6043	0.6509	0.5758	0.8311	0.7861	0.5639
	Recall	**0.9232**	0.6547	0.6580	0.8157	0.9071	0.6672	0.5265	0.5173	0.4408	0.5846
	F1	0.7181	0.7050	0.7099	**0.8402**	0.6483	0.4599	0.5500	0.6819	0.2469	0.4211
UCR	ACC	**0.9965**	0.9692	0.9962	0.9190	0.9961	0.9990	0.9595	0.6490	0.5673	0.5196
	Precision	0.8500	**0.9808**	0.8430	0.6963	0.8284	0.9487	0.6630	0.7201	0.7157	0.7025
	Recall	**1**	**1**	**1**	0.8379	**1**	**1**	0.6697	0.8856	0.8318	0.6196
	F1	0.9389	**0.9903**	0.9148	0.7606	0.9061	0.9373	0.3760	0.5392	0.4609	0.4385

E. RQ2 (Deployability)

To address *RQ2*, we conduct deployment experiments for ten models on two IIoT edge devices.

1) Deployment on Raspberry Pi 4b With General Resources: A Raspberry Pi 4b with a 1.5-GHz ARM Cortex-A72 processor and 2-GB RAM is selected as the first IIoT edge device. In this experiment, two shallow models (PCA and FADSD) are deployed using an online solution, where they perform real-time anomaly detection directly without training and update. The other three machine leaning-based shallow models (IForest, LODA, and Optiforest) and five deep models (DIF, DCdetector, FCVAE, ATF-UAD, and DTAAD) are deployed using an offline solution, where they first are trained in the cloud and then are deployed on the edge device. Fig. 5(a) shows the deployment results of ten models on Raspberry Pi 4b. From the figure, two observations can be found.

1) *Deployability:* All ten models are successfully deployed. In comparison, shallow models are easier to deploy.

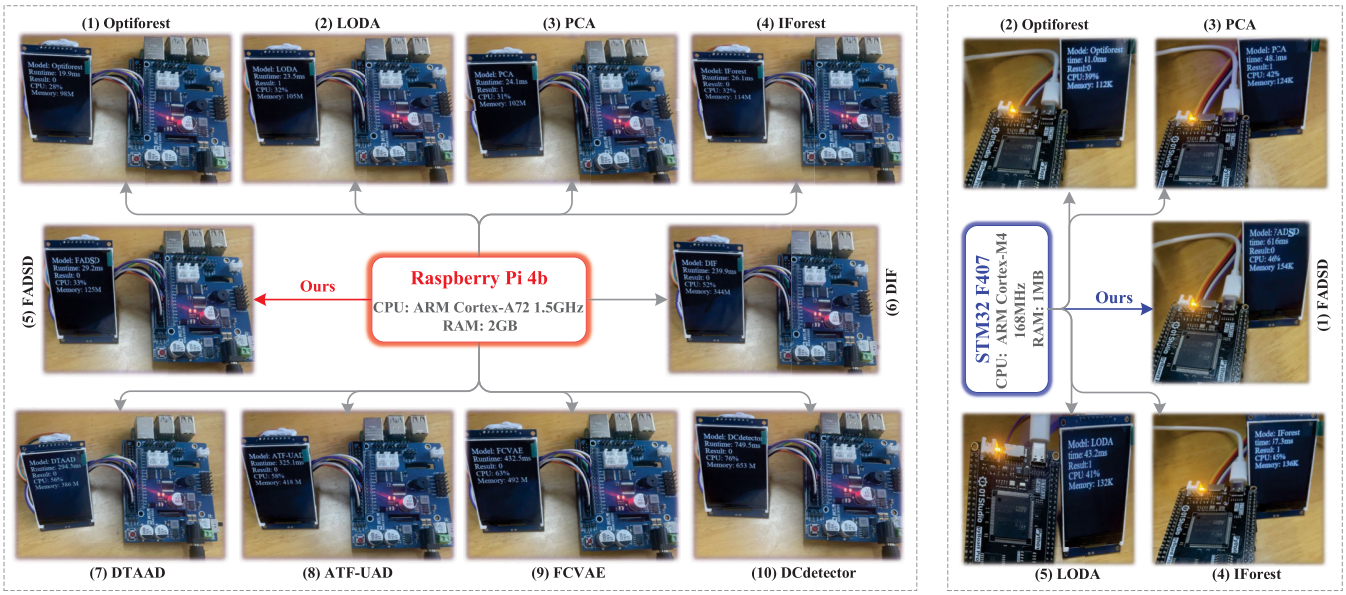


Fig. 5. Deployment results on two IIoT edge devices, where (a) Raspberry Pi 4b on the left and (b) STM32 on the right.

2) *Convenience:* In the IIoT environment, production processes frequently changes, causing time signals to fluctuate dynamically. When faced with dynamically changing IIoT time signals, shallow models deployed online can be used directly. Meanwhile, deep models deployed offline must be fine-tuned in the cloud and redeployed on the edge before being reused. Moreover, shallow models are usually faster and take up less resources.

2) Deployment on STM32 With Little Resources: The STM32 F407 with a 160-MHz ARM Cortex-M4 processor and 1-MB RAM is chosen as the second IIoT edge device. Due to 1-MB RAM limitation, we remove dependence packages exceeding 1 MB from ten models to make them deployable. The deployment results on STM32 are illustrated in Fig. 5(b). Some similar findings can be obtained from the figure.

1) *Deployability:* Only the five nonneural-network-based shallow models are successfully deployed since the deep neural-network-based models require more than 1 MB of RAM.
2) *Convenience:* The FADSD model is easier to deploy compared to other shallow models, and it demonstrates comparable speed and resource consumption.

3) In Summary: The nonneural-network-based models are easier to deploy and operate on resource-limited IIoT edge devices. Therefore, FADSD, which directly designs a nonneural-network solution only in the frequency domain, is highly suitable for the resource-constrained IIoT environment and can achieve reliable and stable accuracy.

F. RQ3 (Timeliness and Resource Consumption)

To answer *RQ3*, we compare the timeliness and resource consumption of ten models across three environments: a PC with powerful resources, a Raspberry Pi 4b with general resources, and an STM32 F407 with limited resources. In the PC environment, five metrics are selected for experimental evaluation. They are model parameters, time for one epoch with the same batch_size (One-epoch-time), CPU and GPU usage (CPU-GPU-Usage), RAM utilization (RAM-Usage), and VGA RAM utilization (VGA-RAM-Usage). In the Raspberry Pi 4b environment, only three metrics are considered: detection time for 100 timestamps (100-timestamp time), CPU usage (CPU-Usage), and RAM-Usage. Similarly, the STM32 environment employs three metrics: detection time for one timestamp (1-timestamp time), CPU-Usage, and RAM-Usage.

Table V lists the comparison results of the ten models.

1) *Timeliness:* The five shallow models (IForest, PCA, Optiforest, LODA, and FADSD) are significantly faster than the deep models (DIF, DCdetector, FCVAE, ATF-UAD, and DTAAD) across all three deployment environments. For example, the average time for the five shallow models in the PC environment is 81.18 s, compared to 326.74 s for the five deep models.
2) *Computing Consumption:* Shallow models exhibit slightly higher computing resource utilization than deep models. This is mainly because deep models are deployed offline on edge devices, and the computing consumption required for their training is all in the cloud. For example, the CPU-Usage on the Raspberry Pi 4b is 52% for the shallow IForest model, while 32% for the deep ATF-UAD model.
3) *Storage Consumption:* Deep models consume significantly more storage resources than shallow models. For example, in the PC environment, the RAM-Usage is 18.1 GB and VGA-RAM-Usage is 15.3 GB for the deep DCdetector model, while 2.9 GB for our FADSD model.
4) *In Summary:* Our FADSD model demonstrates good timeliness and low resource consumption, which means it can work well on resource-limited IIoT edge devices.

G. RQ4 (Robustness)

To address *RQ4*, inspired by the latest assessment benchmarks [30], we introduced different levels of artificial noise

TABLE V
TIMELINESS AND RESOURCE CONSUMPTION

		Shallow Anomaly Detection					Deep Anomaly Detection				
		FADSD	IForest	Optiforest	LODA	PCA	DIF	DCdetector	FCVAE	ATF-UAD	DTAAD
PC with powerful resources (CPU: i7-10700KF GPU: RTX 3090 RAM: 64G VGA-RAM: 24G)	Parameters	-	-	-	-	-	175.1K	505.5K	177.9K	157.3K	126.4K
	One-epoch-time	90.4s	74.5s	71.7s	83.1s	86.2s	321.1s	471.8s	325.8s	288.6s	226.4s
	CPU-GPU-Usage	21%	10%	15%	12%	10%	22%	59%	18%	16%	10%
	RAM-Usage	2.9GB	2.1GB	1.8GB	2.1GB	1.9GB	12.3GB	18.1GB	9.8GB	7.1GB	9.5GB
	VGA-RAM-Usage	-	-	-	-	-	8.6GB	15.3GB	7.5GB	6.4GB	7.8GB
Raspberry Pi 4b (CPU: ARM Cortex-A72 RAM: 2G)	100-timestamps-time	29.2s	26.1s	19.9s	23.5s	24.1s	239.9s	749.5s	432.5s	325.1s	294.3s
	CPU-Usage	33%	52%	76%	63%	58%	56%	32%	28%	32%	31%
	RAM-Usage	125MB	114MB	98MB	105MB	102MB	344MB	653MB	492MB	418MB	386MB
STM32 with little resources (CPU: ARM Cortex-M4 RAM: 1M)	1-timestamp-time	61.6ms	57.3ms	41.0ms	43.2ms	48.1ms	-	-	-	-	-
	CPU-Usage	46%	45%	39%	41%	42%	-	-	-	-	-
	RAM-Usage	154KB	136KB	112KB	132KB	124KB	-	-	-	-	-

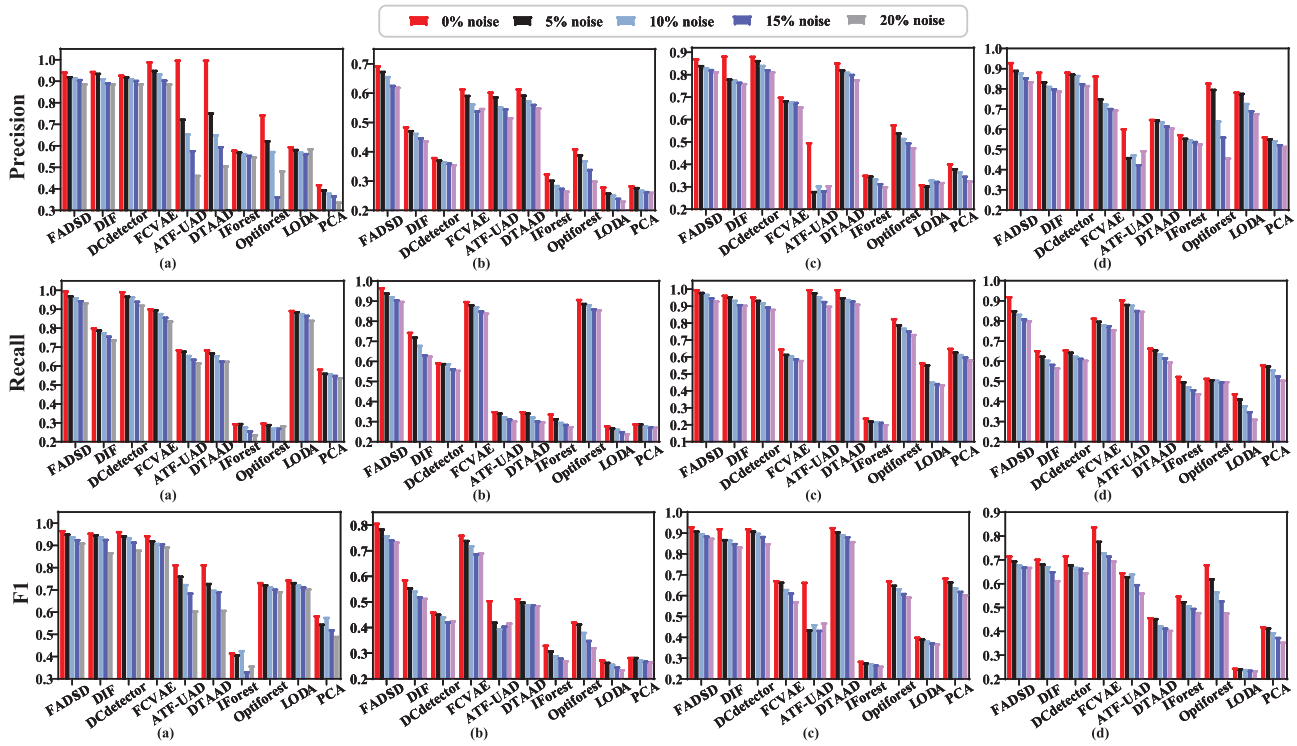


Fig. 6. Robustness analysis on noisy datasets. (a) SWAT. (b) GECCO. (c) HAI. (d) Dodgers.

into the anomaly score calculation process to simulate a noisy environment. Specifically, we select four datasets (SWAT, GECCO, HAI, and Dodgers) to conduct robustness experiments under noise levels of 0%, 5%, 10%, 15%, and 20%. All models are tested across these five noisy environments.

Fig. 6 shows the performance changes of ten models. From the figure, several observations can be found.

1) *Considering the Noise Levels:* As noise levels increase, the performance of all models deteriorates. Among the models, FADSD and DTAAD exhibit the greatest stability. For example, on the Dodgers dataset, the average decrease in $F1$ is 4.67% for FADSD, 5.84% for DTAAD, 8%–10% for DCdetector and DIF, 10%–12% for ATF-UAD and FCVAE, and 21.62% for Optiforest.
2) *Considering the performance*, FADSD, DCdetector, DTAAD, DIF, and FCVAE consistently outperform other models, as indicated by their higher precision scores. For instance, on the HAI dataset at 5% noise, the Precision is 0.8425 for FADSD, 0.8652 for DCdetector, 0.8239 for DTAAD, 0.782 for DIF, and 0.6862 for FCVAE, with other models scoring lower.
3) *In summary*, FADSD demonstrates more stable and superior performance across different noise environments, indicating strong noise tolerance. Such performance mainly lies in the distinct differences between abnormal and normal timestamps in the frequency domain. Even with slight noise, its impact is insufficient to obscure these significant differences.

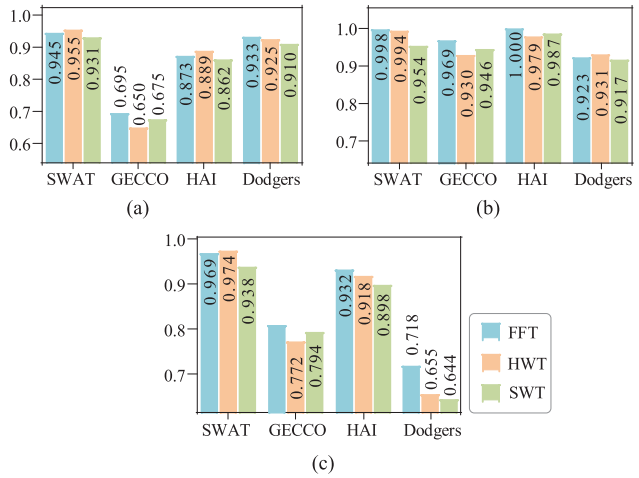


Fig. 7. Different frequency-domain transformation techniques. (a) Precision. (b) Recall. (c) $F1$.

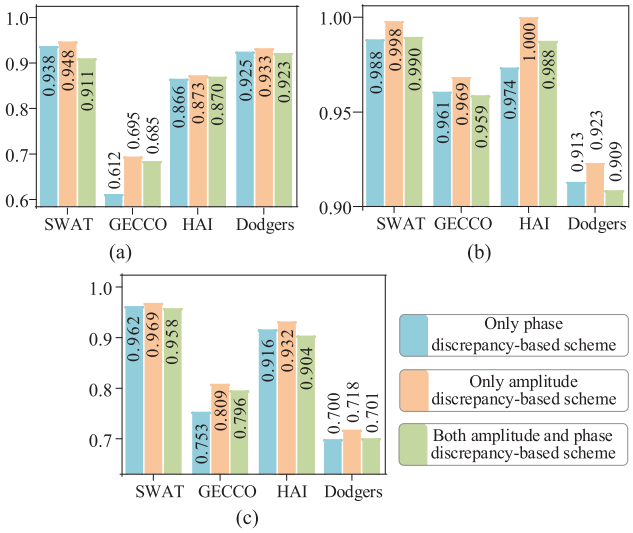


Fig. 8. Amplitude or phase discrepancy-based scheme. (a) Precision. (b) Recall. (c) $F1$.

H. RQ5 (Ablation)

To answer *RQ5*, three ablation experiments are conducted to evaluate the effectiveness of each key component in FADSD. Four IIoT datasets and three metrics (Precision, Recall, and $F1$) are chosen for these experiments.

1) Frequency-Domain Transform: FADSD initially applies FFT to convert time-domain signals into spectral components, and then uses the spectral discrepancy to perform timestamp-level anomaly detection. This experiment aims to verify the impact of different frequency-domain transformation techniques on FADSD performance. To achieve this, transformation methods are compared: FFT, Haar wavelet transform (HWT), and Symlet wavelet transform (SWT).

Fig. 7 presents the accuracy achieved by three frequency-domain transformation techniques. The results in the figure demonstrate that all three configurations exhibit high performance, with their performance bars showing minimal variation in height. For instance, on the SWAT dataset, the Recall score is 0.9980 for FFT, 0.9942 for HWT, and 0.7451 for SWT. The main reason for this is that the frequency-domain transform in FADSD is only responsible for converting the time signal into frequency-domain components. The final frequency-domain features at each timestamp are determined by the customized spectrum discrepancy, rather than the raw frequency-domain components. As a result, FADSD maintains robustness across different frequency-domain transformation techniques.

2) Amplitude or Phase Discrepancy-Based Scheme: FADSD utilizes only spectrum discrepancy in the frequency domain to generate an anomaly score for each timestamp. However, the frequency domain usually contains two types of spectrum: amplitude and phase. Therefore, this experiment is designed to verify which spectrum information is more appropriate for anomaly detection. In this experiment, we construct three competing models: only phase discrepancy-based scheme (first scheme), only amplitude discrepancy-based scheme (second scheme), and amplitude and phase discrepancy-based scheme (third scheme).

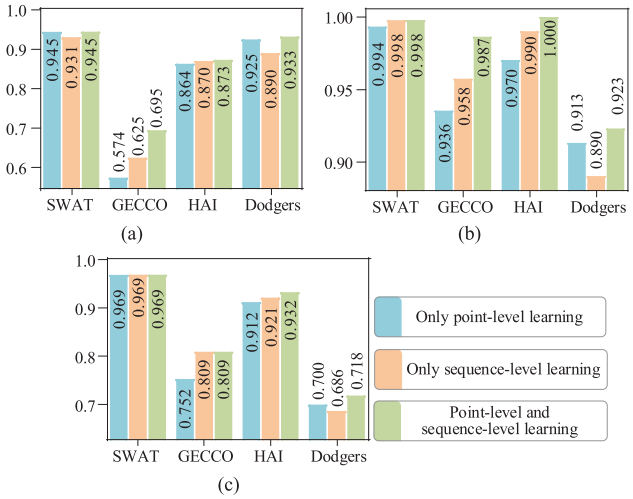


Fig. 9. Point-level and sequence-level learnings. (a) Precision. (b) Recall. (c) $F1$.

Fig. 8 presents the effectiveness of three schemes. From the figure, several observations can be made.

1) *Overall Performance:* Only amplitude discrepancy-based scheme performs significantly better than the other two, achieving the highest performance across all metrics and datasets. For example, the precision score is 0.948 on the SWAT dataset and is 0.933 on the Dodgers dataset for the second scheme, while 0.938 and 0.925 for the first scheme, and 0.911 and 0.923 for the third scheme.
2) *Joint of Amplitude and Phase:* The combination of amplitude and phase discrepancies reduces the overall performance. On the GECCO dataset, the $F1$-score is 0.796 for the third joint scheme, which is smaller than 0.809 of the second scheme but higher than 0.753 of the first scheme.
3) *In Summary:* The only amplitude discrepancy-based scheme is more suitable for anomaly detection in the IIoT environment.

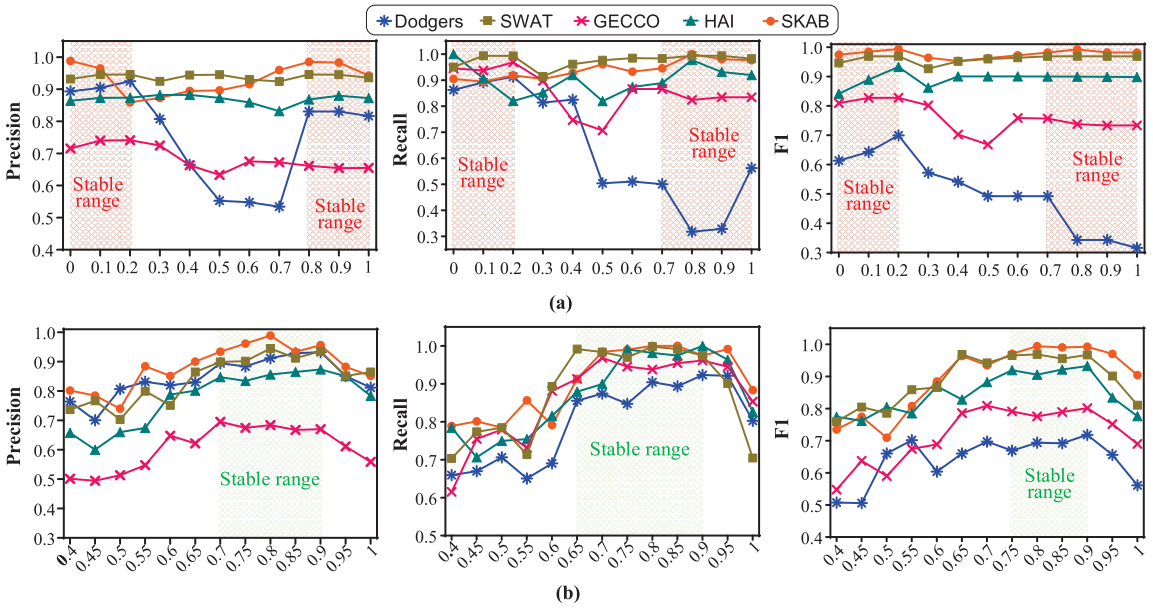


Fig. 10. Sensitivity for two key parameters, where (a) is α and (b) is λ.

3) Point-Level and Sequence-Level Learnings: Point-level and sequence-level spectrum discrepancy calculations are two critical components of FADSD. Among them, the first one is designed to identify explicit anomalies, and the other is designed for implicit anomalies. These two components complement each other and work together to enhance anomaly detection accuracy. In this experiment, three configurations are compared: only point-level learning (first configuration), only sequence-level learning (second configuration), and both point-level and sequence-level learnings (third configuration).

Fig. 9 illustrates the accuracy across the three configurations.

1) *Overall Performance:* The joint point-level and sequence-level learning (third configuration) demonstrates better performance. For example, on the HAI dataset, the $F1$-score is 0.932 for the third configuration, 0.912 for the first configuration, and 0.921 for the second configuration.
2) *Only Comparing the First and Second Configurations:* They have their own advantages in different datasets and metrics. For instance, on the SWAT dataset, the first configuration has better Precision. Conversely, on the HAI dataset, the second configuration has better Precision, Recall, and $F1$. This implies that the first and second configurations complement each other.
3) *In Summary:* Both point-level and sequence-level learnings contribute positively to model performance, and their combination yields even better results.

I. RQ6 (Sensitivity)

To address *RQ6*, we evaluate the impact of two key parameters α and λ, which range from 0 to 1.

The parameter α integrates point-level and sequence-level anomaly scores into the final anomaly score. When $\alpha = 1$, only the point-level score is used, and when $\alpha = 0$, only the sequence-level score is considered. For $0 < \alpha < 1$, the point-level and sequence-level scores are combined using (9). Fig. 10(a) presents the performance changes across five datasets under different parameter α values.

1) *Overall Performance:* The results display a trend of better performance at both ends and worse performance in the middle. For instance, on the GECCO dataset, the trend lines for all three metrics rise, then fall, and eventually rise again.
2) *Dataset Differences:* All metrics for multivariate datasets (SWAT, GECCO, HAI, and SKAB) exhibit more stable compared to the univariate dataset (Dodgers). When α is close to 1, performance on multivariate datasets is better. Conversely, when α is close to 0, performance on univariate datasets is better. It means that the point-level anomaly score is more suitable for multivariate datasets, whereas the sequence-level anomaly score is more effective for univariate datasets.
3) *In Summary:* The intervals $[0, 0.2]$ and $[0.8, 1]$ are relatively stable where our FADSD performs best.

The parameter λ is used as a threshold to determine whether a timestamp is an anomaly. When the anomaly score is greater than λ, the timestamp is an anomaly. Otherwise, it is normal. Due to different characteristics of each dataset, the optimal value of the parameter λ for different datasets always changes dynamically. As a result, it is difficult to find a stable interval for the parameter λ, which is good for different datasets. To solve this problem, the maximum anomaly score for each dataset is used as a benchmark, and the value of the parameter λ is determined by multiplying the benchmark by a ratio. In this way, the value of the parameter λ is compressed to the range [0, 1], which makes it easier to find stable intervals for different datasets. Fig. 10(b) shows the sensitivity of parameter λ across five datasets.

1) *Overall Performance:* As parameter λ increases, the performance of FADSD first increases, then stabilizes, and finally decreases. For instance, on the SWAT dataset, the $F1$-score increases rapidly in the interval $[0, 0.75]$, remains stable in $[0.75, 0.9]$, and slowly decreases in $[0.9, 1]$.
2) *Dataset Differences:* The changing trends of parameter λ on different datasets are very similar.
3) *In Summary:* The intervals $[0.7, 0.9]$ are relatively stable for our FADSD model.

J. Discussion

This article aims to pursue timestamp-level anomaly detection with high speed, high accuracy, and low resource consumption in resource-limited IIoT environments. The results of extensive experiments demonstrate that our FADSD model achieves the expected three goals.

1) *High Speed:* The timeliness and deployability experiments show that FADSD significantly reduces detection time compared to SOTA deep learning models. This advantage is attributed to the fact that it is a nonneural-network solution. Furthermore, FADSD does not require dynamic updates, effectively mitigating the accuracy distortion problem common in neural-network-based and machine-learning-based models.
2) *High Accuracy:* The accuracy, robustness, ablation, and sensitivity experiments indicate that FADSD, leveraging only frequency-domain analysis, achieves top-two detection accuracy across eight datasets and four metrics. Additionally, it demonstrates a strong tolerance to slight noise.
3) *Low Resource Consumption:* Due to its nonneural-network design, FADSD exhibits significantly lower resource consumption than other SOTA deep learning models. This efficiency makes that it can be deployed on resource-constrained devices such as the STM32 407, which has only 1 MB of RAM.

VI. Conclusion

This study focuses on the problem of real-time anomaly detection in a resource-limited environment and develops FADSD, a frequency-spectrum discrepancy-based fast anomaly detection model for IIoT sensor signals. FADSD fills the gap of not being able to perform timestamp-level anomaly detection only in the frequency domain by nonneural-network solutions. Comparative evaluations on eight IIoT sensor time-series datasets demonstrate the superior performance of FADSD. Specifically, timeliness and resource consumption experiments reveal that FADSD operates 10–30 times faster than deep learning-based SOTA models on a Raspberry Pi 4b. Additionally, its resource consumption is only one-third of these models on the same platform. Accuracy experiments further confirm its effectiveness, with FADSD achieving the highest Recall score on six datasets and the second-highest on the remaining two. Moreover, FADSD's average Recall score across all eight datasets surpasses the second-ranked baseline (DCdetector) by 9.73%. These results underscore the ability of FADSD to deliver high speed and low resource consumption while maintaining high accuracy.

Nonetheless, the performance of FADSD exhibits to parameter α on some datasets. In the future, the main goal is to design a lightweight optimization mechanism for the proposed model to automatically select appropriate parameters for different datasets.

References

[1] S. García-de-Villa, D. Casillas-Pérez, A. Jiménez-Martín, and J. J. García-Domínguez, "Inertial sensors for human motion analysis: A comprehensive review," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–39, 2023.

[2] A. Sgueglia, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "A systematic literature review of IoT time series anomaly detection solutions," *Future Gener. Comput. Syst.*, vol. 134, pp. 170–186, Sep. 2022.

[3] X. Huang, F. Zhang, R. Wang, X. Lin, H. Liu, and H. Fan, "KalmanAE: Deep embedding optimized Kalman filter for time series anomaly detection," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, 2023.

[4] Y. Bai, J. Wang, X. Zhang, X. Miao, and Y. Lin, "CrossFuN: Multiview joint cross-fusion network for time-series anomaly detection," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–9, 2023.

[5] N. Mejri, L. Lopez-Fuentes, K. Roy, P. Chernakov, E. Ghorbel, and D. Aouada, "Unsupervised anomaly detection in time-series: An extensive evaluation and analysis of state-of-the-art methods," *Expert Syst. Appl.*, vol. 256, Dec. 2024, Art. no. 124922.

[6] Y. Sun, T. Chen, Q. V. H. Nguyen, and H. Yin, "TinyAD: Memory-efficient anomaly detection for time series data in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 20, no. 1, pp. 824–834, Jan. 2024.

[7] G. Sivapalan, K. K. Nundy, S. Dev, B. Cardiff, and D. John, "ANNet: A lightweight neural network for ECG anomaly detection in IoT edge sensors," *IEEE Trans. Biomed. Circuits Syst.*, vol. 16, no. 1, pp. 24–35, Feb. 2022.

[8] J. Li, H. Izakian, W. Pedrycz, and I. Jamal, "Clustering-based anomaly detection in multivariate time series data," *Appl. Soft Comput.*, vol. 100, Mar. 2021, Art. no. 106919.

[9] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure Industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6503–6511, Sep. 2022.

[10] C. Zhang, T. Zhou, Q. Wen, and L. Sun, "TFAD: A decomposition time series anomaly detection architecture with time-frequency analysis," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2022, pp. 2497–2507.

[11] Z. Wang et al., "Revisiting VAE for unsupervised time series anomaly detection: A frequency perspective," in *Proc. ACM Web Conf.*, New York, NY, USA, May 2024, pp. 3096–3105.

[12] M. Thill, W. Konen, and T. Bäck, "Time series anomaly detection with discrete wavelet transforms and maximum likelihood estimation," in *Proc. Int. Conf. Time Ser. (ITISE)*, vol. 2, 2017, pp. 11–23.

[13] Y.-C. Park, J.-G. Jang, and U. Kang, "Fast and accurate partial Fourier transform for time series data," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1309–1318.

[14] J. Fan, Z. Wang, H. Wu, D. Sun, J. Wu, and X. Lu, "An adversarial time–frequency reconstruction network for unsupervised anomaly detection," *Neural Netw.*, vol. 168, pp. 44–56, Nov. 2023.

[15] L. Zhang, W. Bai, X. Xie, L. Chen, and P. Dong, "TMANomaly: Time-series mutual adversarial networks for industrial anomaly detection," *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 2263–2271, Feb. 2024.

[16] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2021, pp. 1–20.

[17] Y. Yang, C. Zhang, T. Zhou, Q. Wen, and L. Sun, "DCdetector: Dual attention contrastive representation learning for time series anomaly detection," in *Proc. 29th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2023, pp. 3033–3045.

[18] 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.

[19] N. Bai, X. Wang, R. Han, Q. Wang, and Z. Liu, "PAFormer: Anomaly detection of time series with parallel-attention transformer," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 2, pp. 3315–3328, Feb. 2025.

[20] H. Nizam, S. Zafar, Z. Lv, F. Wang, and X. Hu, "Real-time deep anomaly detection framework for multivariate time-series data in industrial IoT," *IEEE Sensors J.*, vol. 22, no. 23, pp. 22836–22849, Dec. 2022.

[21] C. Pascoal, M. R. de Oliveira, R. Valadas, P. Filzmoser, P. Salvador, and A. Pacheco, "Robust feature selection and robust PCA for Internet traffic anomaly detection," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1755–1763.

[22] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Mach. Learn.*, vol. 102, no. 2, pp. 275–304, Feb. 2016.

[23] H. Xiang et al., "OptIForest: Optimal isolation forest for anomaly detection," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, Aug. 2023, pp. 2379–2387.

[24] V. Shanmuganathan and A. Suresh, "LSTM-Markov based efficient anomaly detection algorithm for IoT environment," *Appl. Soft Comput.*, vol. 136, Mar. 2023, Art. no. 110054.

[25] H. Xu, G. Pang, Y. Wang, and Y. Wang, "Deep isolation forest for anomaly detection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12591–12604, Dec. 2023.

[26] Y. Nam et al., "Breaking the time-frequency granularity discrepancy in time-series anomaly detection," in *Proc. ACM Web Conf.*, New York, NY, USA, May 2024, pp. 4204–4215.

[27] X. Zhang, Z. Zhao, T. Tsiligkaridis, and M. Zitnik, "Self-supervised contrastive pre-training for time series via time-frequency consistency," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022, pp. 3988–4003.

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

[29] M. Shyu, S. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *Proc. Int. Conf. Data Mining*, Jan. 2003, pp. 172–179.

[30] J. Paparrizos, P. Boniol, T. Palpanas, R. S. Tsay, A. Elmore, and M. J. Franklin, "Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection," *Proc. VLDB Endowment*, vol. 15, no. 11, pp. 2774–2787, Jul. 2022.

Lei Chen (Member, IEEE) received the M.Sc. degree in computer science and engineering and the Ph.D. degree in automatic control and electrical engineering from Hunan University, Changsha, China, in 2012 and 2017, respectively.

He is currently an Associate Professor with the School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan, China. His current research interests include anomaly detection, lightweight design, time-series analysis, deep learning, information security of industrial control system, and big data analysis.

Xuxin Liu received the B.Eng. degree in automation from Hunan University of Science and Technology, Xiangtan, China, in 2024, where he is currently pursuing the master's degree in control science and engineering.

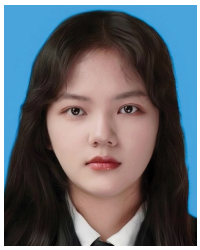
His current research interests include data-driven anomaly detection, deep learning, and fault diagnosis.

Ying Zou received the B.Eng. degree in measurement and control technology and instruments from Central South University, Changsha, China, in 2013, the M.Sc. degree in instrument science and technology from Beihang University (BUAA), Beijing, China, in 2016, and the Ph.D. degree in control science and engineering from Nanyang Technological University, Singapore, in 2020.

Since then, she has been with the School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan, China, where she is currently an Associate Professor. Her research interests include adaptive control, autonomous robotic systems, intelligent systems, and distributed systems.

Jiajun Tang received the B.Eng. degree in automation from Hunan Institute of Science and Technology, Xiangtan, China, in 2023, where he is currently pursuing the master's degree in control science and engineering.

His current research interests include data-driven anomaly detection, deep learning, and fault diagnosis.



Canwei Liu received the B.E. degree in computer science and technology from the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China, in 2024. She is currently pursuing the M.S. degree in computer technology with the School of Computer Science and Engineering, Central South University, Changsha, China.

Her current research interests include data mining, deep learning, time-series analysis, and medical image analysis.

Bowen Hu received the B.Eng. degree in automation from Hunan University of Science and Technology, Xiangtan, China, in 2024, where he is currently pursuing the master's degree in control science and engineering.

His current research interests include data-driven anomaly detection, deep learning, and fault diagnosis.

Mingyang Lv received the Ph.D. degree in control theory and engineering from Hunan University, Changsha, China, in 2020.

He is currently a Lecturer with the School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan, China. His research interests include image processing, modeling, fault diagnosis, and safety control for complex system based on machine learning and deep learning.