Noise Matters: Cross Contrastive Learning for Flink Anomaly Detection

Zhihao Zhuang
East China Normal University, China
zhuangzhihao@stu.ecnu.edu.cn

Yingying Zhang
Alibaba Cloud Computing, China
congrong.zyy@alibaba-inc.com

Kai Zhao
Aalborg University, Denmark
kaiz@cs.aau.dk

Chenjuan Guo*
Bin Yang
East China Normal University, China
{cjguo,byang}@dase.ecnu.edu.cn

Qingsong Wen
Squirrel Ai Learning, USA
qingsongedu@gmail.com

Lunting Fan
Alibaba Cloud Computing, China
lunting.fan@taobao.com

ABSTRACT

Flink clusters often suffer from hotspot issues where the monitored job delay and CPU usage keep rising and remain high. This necessitates the detection of anomalous time series to pinpoint the hotspot machines. However, the state-of-the-art unsupervised time series anomaly detection (UTAD) methods are ineffective in this scenario. We identify two main reasons for this. First, the hotspot scenario requires us to pay particular attention to Flink-specific anomalies, e.g., slow-rising and high-level anomalies, which the existing methods struggle to address. Second, the state-of-the-art anomaly detection methods often assume that training datasets do not contain anomalies, but the data collected from the running Flink clusters contains noise, which causes these methods to learn anomalous patterns as normal patterns. In this paper, we first conduct experiments to analyze why existing methods fail in the Flink scenario. To tackle these challenges, we propose a cross-contrastive approach to learn the context information for each timestamp to enable Flink-specific anomaly detection. Then, to address noisy anomalies, we incorporate prior knowledge to set an anomaly boundary to prevent the model from learning anomalous patterns. Extensive experiments show that our method not only outperforms existing methods in the Flink scenario but also achieves state-of-the-art results on public benchmark datasets.

PVLDB Reference Format:
Zhihao Zhuang, Yingying Zhang, Kai Zhao, Chenjuan Guo, Bin Yang, Qingsong Wen, and Lunting Fan. Noise Matters: Cross Contrastive Learning for Flink Anomaly Detection. PVLDB, 18(4): 1159 - 1168, 2024.
doi:10.14778/3717755.3717773

PVLDB Artifact Availability:
The source code, data, and other artifacts have been made available at https://github.com/decisionintelligence/ContraAD.

*Corresponding author

Proceedings of the VLDB Endowment, Vol. 18, No. 4 ISSN 2150-8097.
doi:10.14778/3717755.3717773

1 INTRODUCTION

Apache Flink is a stream and batch processing framework for big data processing and analytics. It is designed to efficiently process large volumes of data in real-time and batch processing modes, and it is suitable for a wide range of use cases in the field of big data analytics. Flink is deployed in a distributed manner across multiple nodes, where each node runs user-defined operators. Data transfer from one operator to another among the nodes and sink to a final result. Due to the streaming nature of the data, a certain node may encounter performance bottlenecks where a significant amount of data is waiting in the buffer and cannot be processed promptly due to factors, such as the complexity of operators, machine efficiency, or data volume. This brings in a problem of data delay on these nodes. As a result, the affected node may either reject or slowly allow data in from its upstream nodes, thus consequently affecting the normal operation of the upstream nodes, which is referred to as a hotspot anomaly node.

Alibaba Cloud offers computing services to companies and individual users, which requires efficient processing of various computing jobs. To ensure efficient data transfer among distributed nodes, there is a need to detect hotspot nodes and then employ scheduling algorithms to migrate jobs from these hotspot nodes to other nodes where no data is delayed. Therefore, it is urgent to detect if there is a hotspot anomaly node. Based on our on-site reliability engineering experience, we monitor job delay and CPU usage as time series and report a hotspot anomaly node if its CPU usage rises sharply in a short period, concurrently accompanied by increasing delays in multiple jobs that run on the node. Thus, in addition to the point-level anomalies exhibited in benchmark datasets [2, 13, 17, 21, 31], our Flink scenario also shows its specific anomalies, such as slow-rising patterns or a piece of time series that, after sharp rising, maintains high without descending, as shown in Figure 1. We need to identify both point-level and Flink-specific time series anomalies.

In the Alibaba Cloud scenario, we collect the CPU usage and job latency information for thousands and thousands of nodes every 20 seconds. In this problem, due to the vast amount of time series data, supervised methods are not feasible because manual labeling is impractical and not cost-effective. Therefore, unsupervised anomaly detection methods become crucial in this setting. Existing unsupervised anomaly detection methods can be categorized into different types, including density estimation-based methods [3, 15, 18, 33], clustering-based methods [14, 25, 34], reconstruction-based [22,

31, 47] methods, auto-regressive methods [13], association-based method [39], forecasting-based method [4, 5, 8, 9, 24, 37, 38, 46] and so on. Recently, the method [42] based on contrastive learning is proposed that aims to learn two view representations for each timestamp and calculate the difference between the two views of the same timestamp. The underlying idea is that the two views of a normal timestamp should be similar, while conversely, the two views of an anomaly timestamp differ significantly. Although these methods have shown good detection accuracy on benchmark datasets, they fall short in our scenarios. We summarize the challenges as follows.

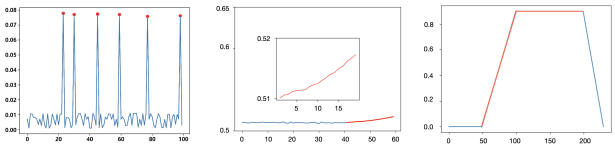
Challenge 1: In addition to the existing point-level anomalies, our Flink scenario sets a requirement to detect its specific anomalies. For example, when the latency of multiple jobs on a node continues to rise or rises and remains at a high level, it often indicates the occurrence of anomalies on that node. However, state-of-the-art anomaly detection methods are not sensitive to such kinds of anomalies. Reconstruction-based approaches [47] often exhibit relatively small reconstruction errors between the Flink-specific anomalies and the normal data and thus fail to identify such anomalies. Association-based [39, 42] methods, which compare different views of the same timestamp, overlook the contextual information of each timestamp, making it challenging to discover the Flink-specific anomalies as well.

Challenge 2: We are facing a very large time series data that originates from running Flink clusters. This leads to the inclusion of numerous anomalies and noises in the collected training dataset. Most of the traditional unsupervised anomaly detection methods [13, 22, 32, 39, 42, 47] are built on the assumption that the training dataset is rather clean and noise-free, where they overlook the presence of actual anomalies and noises. Thus, the effectiveness of these methods is susceptible due to the influence of anomalies and noises and is unsatisfactory.

In Section 3, we first benchmark the state-of-the-art methods on our Flink dataset collected from Alibaba Cloud platforms, to show the flaws of these methods. Different from these methods, we propose a novel representation learning method for unsupervised anomaly detection that effectively addresses the above challenges.

For **Challenge 1**, instead of calculating a reconstruction error at each timestamp [22, 32, 47] or calculating the discrepancy between the two views of each timestamp [39, 42], we propose a novel cross contrastive learning to pay additional attention to the Flink-specific anomalies. We first utilize an attention mechanism to learn representations from global and local perspectives, based on which we then conduct cross-contrastive learning. The intuition is that observations of adjacent timestamps in normal time series are not far from each other, and therefore, their latent representations should be similar. In contrast, for the observations of adjacent timestamps, where Flink-specific anomalies, *e.g.*, slow-rising trend, happens, we should learn to enlarge the difference among their representations. This gives rise to a different anomaly detection mechanism, where the representation distances among adjacent timestamps are measured, and an anomaly is detected if the distances are much distinguishable from each other.

For **Challenge 2**, we propose a novel loss function that incorporates prior knowledge to co-guide the optimization process, such that the model could treat anomalous and normal timestamps differently even if their labels are unknown. Specifically, we set an anomaly boundary for each timestamp that is the normalization score of its observation, and use it as the prior knowledge to reflect the deviation of the anomalous from the normal. The intuition is that the training loss of normal timestamps that have smaller anomaly boundaries, indicating that these timestamps do not deviate significantly from their normalization, could be optimized as small as possible. In contrast, the training loss of anomalies that have higher anomaly boundaries should not be well optimized as the normal timestamps, such that we could assign larger anomaly scores to anomalous timestamps. Therefore, we only optimize their loss close to their anomaly boundaries. In this way, we effectively take noisy training data into account and mitigate the influence of anomalies and noises during the training process, thereby improving the accuracy of anomaly detection.



(a) NIPS-TS-SWAN (b) Slow-rising pattern (c) High-level pattern

Figure 1: Different types of anomalies. (a) shows point-level anomalies from NIPS-TS-SWAN dataset; (b) and (c) are anomalies from our Flink dataset.

In this paper, we propose ContraAD a cross-contrastive approach to detect anomalies in the Flink scenario. We first utilize a representation network under global and local views to learn a representation for each timestamp. After obtaining these representations, we conduct cross contrastive learning to compare each pair of timestamps using their representations for optimization. During the optimization, we set an anomaly boundary to prevent the model from learning anomalies induced by noise. Our contributions can be summarized as follows:

- We propose a novel representation network and use the cross-contrastive distances among the representations of different timestamps to detect anomalies.
- For each timestamp, we calculate its deviation from the original time series and set it as the anomaly boundary for the optimization to prevent learning anomalous patterns.
- Extensive experiments show that our method not only effectively addresses the challenges in the Flink scenarios but also achieves state-of-the-art results on anomaly detection benchmark datasets.

2 RELATED WORK

In this section, we will introduce the relevant works on unsupervised time series anomaly detection, contrastive learning, and learning with noise data.

Unsupervised Time Series Anomaly Detection. There has been a lot of research on UTAD. For example, the classical density-based anomaly detection method *e.g.*, LOF [3], classical machine learning methods includes Deep SVDD [25] and THOC [28]. There are lots of deep learning based methods. LSTM-VAE [22] utilizes

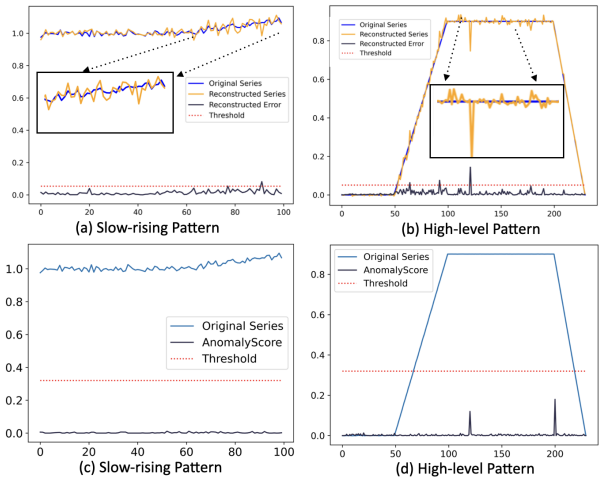


Figure 2: The effectiveness of BeatGAN and DCdetector on Flink-specific anomalies.

LSTM to extract temporal features and then uses VAE for reconstruction in the context of time series data. OmiAnomaly [32] utilizes the reconstruction probability that comes from VAE for anomaly detection. BeatGAN [47] introduces the discriminator's loss as a regularization mechanism for the reconstruction network. However, these methods fail to address the Flink-specific anomalies and assume that the training data does not contain anomalies, as shown in Section 3.

Contrastive Learning. Both DACAD [10] and NCAD [6] inject the anomalies into training set and then construct the constrastive pair for training. DCdetector [42] involves contrasting the representations of the same timestamp from different perspectives. However, these method not effective to our Flink-specific anomaly, and they also highly rely on a noise-free training dataset. Thus, these methods fail in our Flink scenario.

Learning with Noise Data. Learning with noise data [7, 12, 19, 29, 36, 43, 44], especially in computer vision, has been extensively researched in various domains. Noise in the training set can be addressed by designing a loss function [7, 36, 43] in such a way that the minimization of this loss function is tolerable to noise. Some methods [12, 19, 44] also utilize the joint training of two networks to effectively combat noisy labels. However, there is a lack of research addressing this aspect in time series anomaly detection. Although RAE [16] decomposes the original sequence into a noisy and a clean sequence through an autoencoder model, it only can be considered as a smoothing operation and is effective only for handling small perturbations from the noise not for the anomalies in training dataset. Thus, dealing with a noisy training dataset is still a challenge.

3 MOTIVATION

In this section, we use experiments to show that the state-of-the-art methods fall short in our cases.

Experiment 1: Impact of Flink-specific anomalies. This experiment aims to investigate whether state-of-the-art methods can detect the Flink-specific anomalies correctly.

We highlight some key difference of the Flink scenario and other anomaly detection scenes as follows. First, our Flink dataset includes common anomaly types as existing anomaly datasets, such as point-level anomalies and seasonal anomalies. Besides, our Flink dataset contains more segment-level anomalies specific to Flink semantics, such as high-level patterns and slow-rising patterns. A high-level anomaly pattern refers to a situation where a metric transitions from a normal range to an abnormal range and remains in that elevated range for a long time. For example, CPU usage might increase from 50% to 95% and then fluctuate within the 95% range, indicating an abnormal state. A slow-rising pattern anomaly refers to an abnormal pattern where a machine metric shows a gradual upward trend with fluctuations over a period of time, indicating a slow but steady increase in the metric's value. Figure 1(a) represents the time series where point-level anomalies happened as selected from the NIPS-TS-SWAN dataset [2, 17]. Figure 1(b) shows a piece of slow-rising anomaly. Figure 1(c) represents the scenario where job latency rises and remains at a high level for a period. In the Flink dataset, these anomalies indicate a hotspot node. We select the reconstruction-based method BeatGAN [47] and contrastive-based method DCdetector [42] to evaluate their effectiveness on our Flink dataset. All settings are taken from their original papers, and we carefully tune all hyperparameters based on our scenario to achieve the best test results.

We evaluate the effectiveness of detecting slow-rising and high-level anomalies that are shown in Figure 1(b) and (c). To illustrate their bad performance, we visualize the original time series, the reconstructed time series, the reconstruction errors, and the anomaly threshold as used in the paper for BeatGAN in Figure 2(a) and (b). We also visualize the original time series, the anomaly scores, and the anomaly threshold which is determined by the anomaly score of each timestamp with a predefined anomaly ratio for DCdetector in Figure 2(c) and (d). Notes that if the reconstruction error or the anomaly score of a certain timestamp is higher than the threshold then this timestamp is considered as an anomaly.

It can be observed from Figure 2(a) that, in the slow-rising time series, the reconstructed series is almost the same as the normal time series, as BeatGAN treats the slow-rising time series as the normal time series, and therefore it fails to detect the slow-rising anomalies. Similarly, Figure 2(c) shows that DCdetector fails either, as in the slow-rising trend the contrastive difference between two views for each timestamp does not vary significantly. Figure 2(b) and (d) show that BeatGAN and DCdetector do not demonstrate the ability to detect the high-level pattern either. BeatGAN exhibits a significant drop in its reconstructed series, and the DCdetector can detect only one anomaly in a piece of high-level anomalies. Our interpretation of this phenomenon is that the training dataset may include some high-level anomalies, thus leading to the production of such incorrect outputs.

This indicates that the existing methods have limited capability in identifying Flink-specific anomalies and cannot meet our requirements.

Experiment 2: Impact of noisy training datasets. This experiment aims to show whether the state-of-the-art models are robust to noisy training data occurring in our Flink dataset.

Our Flink dataset is collected from several running flink clusters, and inevitably, it contains many abnormal anomalies as shown

Table 1: Comparison on Flink data. *indicates a relatively clean training dataset.

	Flink dataset			Flink dataset*		
methods	P	R	F1	P	R	F1
Statistical	10.00	38.12	15.83	-	-	-
DCdetector	35.73	11.21	17.07	42.85	13.12	20.09
BeatGAN	38.89	6.19	10.68	32.39	20.35	24.99

in Figure 1. We continue to use BeatGAN and DCdetector in this experiment. We also use a rule-based statistical anomaly detection method which is based on a season decomposition algorithm and uses many prior rules. Further, we remove a proportion of anomalies from the training dataset manually, such that the two baselines could be trained on a relatively clean dataset, indicated by * in Table 1. We report the precision, recall, and F1-score to show their anomaly detection performance.

Based on the experimental results mentioned above, we conclude that the state-of-the-art unsupervised anomaly detection methods fall short in discovering the Flink-specific anomalies and noisy training datasets, which significantly influences their final effectiveness.

4 METHODOLOGY

4.1 Problem Definition

A multivariate time series $\mathcal{X} \in \mathbb{R}^{\mathcal{N}\times C}$ is composed of a sequence of observation $\langle x_1, x_2, ..., x_N\rangle$ of length $\mathcal{N}$, where $x_i \in \mathbb{R}^C$ denotes the data collected or observed at timestamp i with C channels. For time series anomaly detection, we aim to obtain a model $\mathcal{D}$ where $\mathcal{X}_{test}$ is the particular time series from which we aim to detect anomalies and $\mathcal{Y}_{pred} = \{\hat{y_1}, \hat{y_2}, ..., \hat{y_n}\}$ where $\hat{y_i} \in \{0(Normal), 1(Abnormal)\}$ is the output of the detection model $\mathcal{D}$.

$$\mathcal{Y}_{pred} = \mathcal{D}(\mathcal{X}_{test})$$

Unsupervised anomaly detection refers to the scenario where we only have unlabeled data $\mathcal{X}_{train}$ to train the detection model $\mathcal{D}$. The essence of unsupervised time series anomaly detection is to assign an anomaly score for each timestamp x_i. Based on the anomaly score and in combination with certain threshold selection methods [40], we can determine whether a timestamp is anomalous or not.

4.2 Overall Architecture

Figure 4 shows our overall architecture, which consists of four main modules: the preprocessing, the representation network, the cross discrepancy, and the optimization modules.

In the pre-processing module, we apply the normalization to the input time series data $\mathcal{X}$, as shown in Equation 1.

$$\begin{aligned} \mathcal{X}_{norm} = Norm(\mathcal{X}) = \gamma^i\left(\frac{\mathcal{X}^i - \mathbb{E}(\mathcal{X}^i)}{\sqrt{Var(\mathcal{X}^i)+\epsilon}}\right) + \beta^i, \text{where} \\ \mathbb{E}(\mathcal{X}^i) = \frac{1}{\mathcal{N}}\sum_{j=1}^{\mathcal{N}} x_j^i, \quad Var(\mathcal{X}^i) = \frac{1}{\mathcal{N}}\sum_{j=1}^{\mathcal{N}}(x_j^i - \mathbb{E}(\mathcal{X}^i))^2. \end{aligned} \tag{1}$$

Specifically, $\mathcal{X}_{norm} \in \mathbb{R}^{\mathcal{N}\times C}$ denotes the normalized time series, and $\mathcal{X}^i \in \mathbb{R}^{\mathcal{N}}$ denotes the i-th channel of $\mathcal{X}$, where each x_j^i denotes the observation in the i-th channel at the j-th timestamp. β^i and γ^i are learnable affine parameters.

In the representation network, we learn the representation for each timestamp by summing the representations obtained from two perspectives: the attention in a global view and the attention in a local view with its neighboring timestamps. This enables the representation network to learn representations under different receptive fields. After we have the representation for each timestamp, we pass it to the cross discrepancy module where we calculate the cross distance among the representations of different timestamps, which yields a distance matrix. In the optimization module, to account for noises in the training data, given the distance matrix from the cross discrepancy module, we calculate the deviation of each timestamp from the input time series as the anomaly boundary, in order to guide the optimization process.

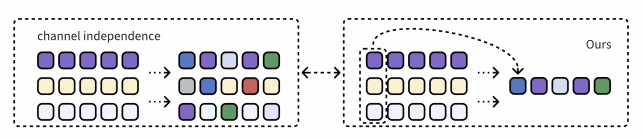


Figure 3: Difference between our work and previous study on the channel aspect

4.3 Representation Network

4.3.1 Channel Fusion. Previous work on representation learning [] often assumed independence between multivariate time series channels, implying no correlation. However, this assumption may not align with the real-world relationships between data. In our approach, we default to the assumption that there is a connection between data in different dimensions.

Figure 3 provides an intuitive illustration of the specific differences between the two assumptions. The main difference lies in their assumption of channel independence, where they learn a representation for each channel's timestamps based on this assumption and concatenate these representations. In contrast, we first fuse the channels, enabling us to derive the representation of a point at a specific moment in one step. In previous research, it finds that Instance Normalization performs better for time series data. Therefore, we initially apply Instance Normalization to preprocess the data where $\mathcal{X} \in \mathbb{R}^{\mathcal{B}\times\mathcal{S}\times C}$ is the raw input time series with batch size $\mathcal{B}$ window size $\mathcal{S}$ and channel dimension C , $\mathcal{X}^i$ denotes the $i-th$ channel of $\mathcal{X}$, $\mathcal{W}_f \in \mathbb{R}^{C\times d_{model}}$ is weight matrix to fuse all channel into embedding dimension and $\mathcal{X}_{emb} \in \mathbb{R}^{\mathcal{B}\times\mathcal{S}\times d_{model}}$ is the embedding of fused channel. Besides $\beta^i \in \mathbb{R}^{C\times 1}$and $\gamma^i \in \mathbb{R}^{C\times 1}$ are learnable affine parameters.

$$\begin{aligned} \mathbb{E}(\mathcal{X}^i) = \frac{1}{\mathcal{S}^i}\sum_{j=1}^{\mathcal{S}^i}\mathcal{X}_j^i, \quad Var(\mathcal{X}^i) = \frac{1}{\mathcal{S}^i}\sum_{j=1}^{\mathcal{S}^i}(x_j^i - \mathbb{E}(\mathcal{X}^i))^2 \\ \mathcal{X}_{emb} = [\gamma^i\left(\frac{\mathcal{X}^i - \mathbb{E}(\mathcal{X}^i)}{\sqrt{Var(\mathcal{X}^i)+\epsilon}}\right) + \beta^i] \times \mathcal{W}_f \end{aligned}$$

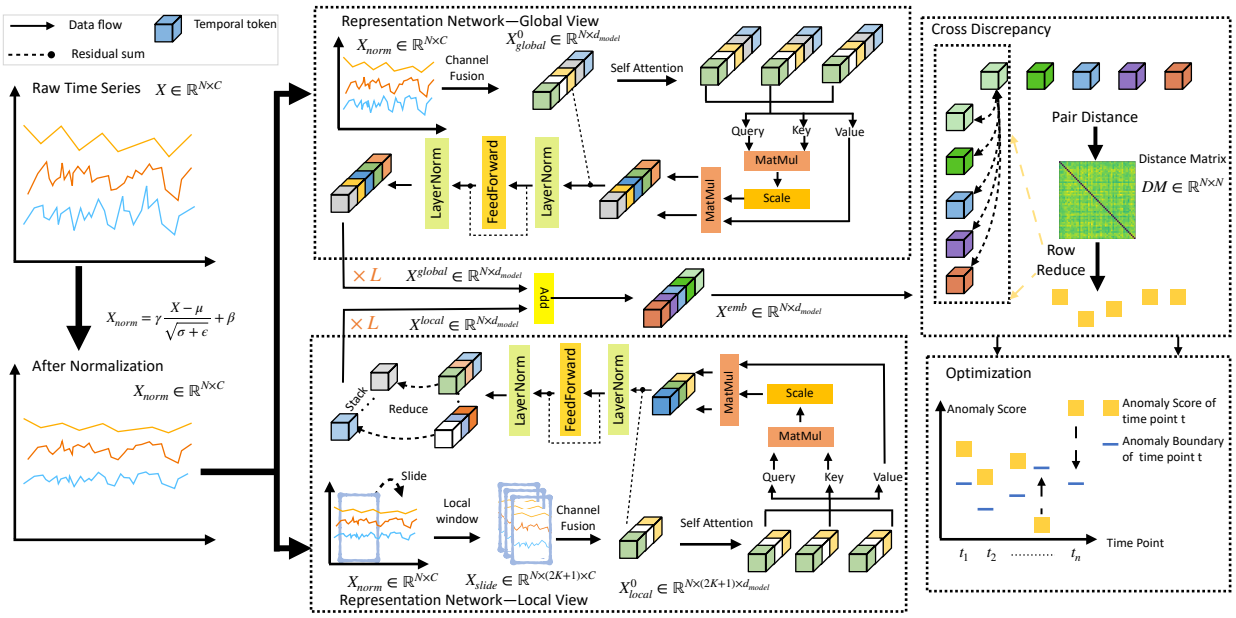


Figure 4: The overall architecture of the proposed ContraAD method.

4.3.2 Two views of attention. We propose an effective attention [35] mechanism as the representation network [41] to capture relationships among different timestamps. Our method calculates an attention score for each timestamp from the global and local perspectives to make the representation network have sights of different receptive fields. Besides, instead of using channel independence which fails to capture correlations among multivariate time series, we use a linear mapping to fuse the channels to capture the correlations.

4.3.3 Global view. The use of global attention is intended to enable the model to learn the connections among timestamps from the time series. Given a normalized time series $\mathcal{X}_{norm} \in \mathbb{R}^{N\times C}$, we first fuse all channels with a linear mapping and yield $\mathcal{X}^0_{global} \in \mathbb{R}^{N\times d_{model}}$. Next, we calculate the association between each of N timestamps and all other timestamps. We utilize Rotary Position Embedding [30] to enhance the model's ability to capture relationships in long sequences. A basic transformer block typically consists of an attention layer, a feedforward layer, and two Normalization layers, and multiple blocks are stacked to form L blocks. We take the l-th transformer block for example.

$$\mathcal{Q}^l, \mathcal{K}^l, \mathcal{V}^l = \mathcal{W}^l_{\mathcal{Q}}\mathcal{X}^{l-1}_{global}, \mathcal{W}^l_{\mathcal{K}}\mathcal{X}^{l-1}_{global}, \mathcal{W}^l_{\mathcal{V}}\mathcal{X}^{l-1}_{global}, \tag{2}$$

$$\mathcal{Q}^l, \mathcal{K}^l = Rotary(\mathcal{Q}^l, \mathcal{K}^l), \tag{3}$$

$$\mathcal{X}^l_{attn} = Softmax(\frac{\mathcal{Q}^l(\mathcal{K}^l)^T}{\sqrt{d_{model}}})\mathcal{V}^l, \tag{4}$$

$$\mathcal{X}^l_{global} = Feed \oplus Norm(\mathcal{X}^l_{attn}). \tag{5}$$

Specifically, $\mathcal{X}^{l-1}_{global}$ denotes the output of $l-1$ layers and $\mathcal{W}^l_{\mathcal{Q}}$, $\mathcal{W}^l_{\mathcal{K}}$, $\mathcal{W}^l_{\mathcal{V}}$ are learnable projection matrices. $Feed \oplus Norm$ denotes the layers composed by the FeedForward layer and the LayerNorm with a residual sum. The final representation is equal to the output of the final layer in the L layers which is $\mathcal{X}^{global} = \mathcal{X}^L_{global}$.

4.3.4 Local view. To focus on the adjacent information, we introduce the local attention. Given a normalized time series $\mathcal{X}_{norm} \in \mathbb{R}^{N\times C}$, we first take a slide operation as defined in Equation 6. We pad k zeros to the left and right sides of $\mathcal{X}_{norm}$ to align the shape to $\mathcal{X}_{pad} \in \mathbb{R}^{(N+2k)\times C}$. And then, we use a window of size $2k+1$ to slide over the padded sequence $\mathcal{X}_{pad}$ with step size 1 and pile the slides together, thus producing $\mathcal{X}_{slide} \in \mathbb{R}^{N\times(2k+1)\times C}$. After sliding, we fuse all C channels together using a linear mapping which yields $\mathcal{X}^0_{local} \in \mathbb{R}^{N\times(2k+1)\times d_{model}}$. Next, the attention scores for each timestamp in the window of size $2k+1$ are calculated. The embeddings for all timestamps are aggregated to obtain the representation for the $(k+1)$-th timestamp.

$$\mathcal{X}_{pad} = Pad(\mathcal{X}, 0, k), \quad \mathcal{X}_{slide} = slide(\mathcal{X}_{pad}). \tag{6}$$

We also take the l-th transformer block for example where $\mathcal{W}^l_{\mathcal{Q}}$, $\mathcal{W}^l_{\mathcal{K}}$, $\mathcal{W}^l_{\mathcal{V}}$ are learnable projection matrices and $\mathcal{X}^l_{local} \in \mathbb{R}^{N\times(2k+1)\times d_{model}}$ is the attention output of l-th layer.

$$\mathcal{Q}^l, \mathcal{K}^l, \mathcal{V}^l = \mathcal{W}^l_{\mathcal{Q}}\mathcal{X}^{l-1}_{local}, \mathcal{W}^l_{\mathcal{K}}\mathcal{X}^{l-1}_{local}, \mathcal{W}^l_{\mathcal{V}}\mathcal{X}^{l-1}_{local}, \tag{7}$$

$$\mathcal{X}^l_{attn} = Softmax(\frac{\mathcal{Q}^l(\mathcal{K}^l)^T}{\sqrt{d_{model}}})\mathcal{V}^l, \tag{8}$$

$$\mathcal{X}^l_{local} = Feed \oplus Norm(\mathcal{X}^l_{attn}). \tag{9}$$

Finally, we conduct a reduced operation on the last layer's output $\mathcal{X}^L_{local}$ to get the final representation of local attention for each timestamp. The process is shown in Equation 10.

$$\mathcal{X}^{local} = \mathcal{X}^L_{local}\mathcal{W}_\alpha. \tag{10}$$

$\mathcal{X}^{local}$ denotes the output of local attention and $\mathcal{W}_\alpha$ is a learned vector introducing to reduce the embedding representation of local window to a single temporal token. We initialize $\mathcal{W}_\alpha$ with a softmax Gaussian kernel as shown in Equation 11,

$$\mathcal{W}_\alpha^i = \frac{exp(\frac{1}{\sqrt{2\pi}} exp(-\frac{|i-(k+1)|^2}{2}))}{\sum_{j=0}^{2k+1} exp(\frac{1}{\sqrt{2\pi}} exp(-\frac{|j-(k+1)|^2}{2}))}, \tag{11}$$

where i denotes the index of a timestamp, to incorporate positional information during the reduction process. The intuition is that neighbors should be given more weight during the reduction process.

After computing the embeddings for both global and local views, we add them together to obtain the final representation $\mathcal{X}_{emb}$ for each timestamp.

$$\mathcal{X}_{emb} = \mathcal{X}^{global} + \mathcal{X}^{local}. \tag{12}$$

4.4 Optimization

The state-of-the-art unsupervised anomaly detection method DCdetector [42] is based on contrastive learning, which minimizes the distance between different representations of the same timestamp from patch-wise and in-patch views. However, minimizing the discrepancy of a single timestamp at different views overlooks the information among all timestamps. Besides, this approach still cannot address the issue we raised, namely overfitting on datasets with noise, as anomaly points would also receive equal optimization. In contrast, we first calculate the representation of each timestamp, which is the summation of global and local attention, and it can enhance the intra-channel information. Then we compare discrepancies among all timestamps using the representations, which can help to detect the slow-rising and high-level anomalies by utilizing the context information. Meanwhile, we use prior knowledge of raw time series to alleviate the impact of noise in the training set.

4.4.1 Cross discrepancy. Here, we optimize from the views of all timestamps. Figure 5 intuitively illustrates the differences between our approach and the existing methods [39, 42]. The intuition is that if a piece of time series does not contain anomaly timestamps, then the piece is harmonious and consistent, meaning that the difference between any two of its timestamps should be small. However, if there are anomalies exist, the overall consistency of the time series is affected by these anomalies. Therefore, naturally, we measure the consistency between observations x_i and x_j at any two timestamps using the p-norm of their representation vectors which are the i-th and j-th indices of $\mathcal{X}_{emb}$ respectively. Thus, we have a symmetric distance matrix $\mathcal{DM} \in \mathbb{R}^{N \times N}$.

$$\mathcal{DM}_{i,j} = ||\mathcal{X}_{emb}[i] - \mathcal{X}_{emb}[j]||_p \quad \forall i, j \in \mathcal{N}. \tag{13}$$

We sum the upper triangle of the distance matrix as the loss function of the representation network. Based on this loss, we can update the representation network.

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{DM}_{i,j}. \tag{14}$$

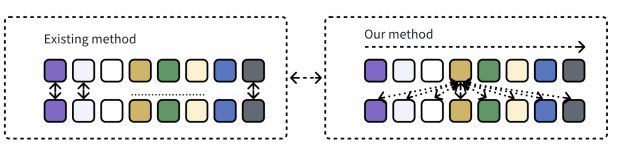


Figure 5: Difference between ours and existing work on contrastive strategy.

4.4.2 Incorporate prior knowledge. We notice that the reason for the model overfitting on datasets with noise is that the state-of-the-art models treat all timestamps as normal timestamps during optimization, and they ignore anomalies and noises in the training set. In such a scenario, their optimization process treats normal and anomalous timestamps equally, and targets to minimize the loss for all timestamps as small as possible. Thus, they lead to the overfitting issue of the noisy data. To address this problem and relieve the impact of noisy data, we set an anomaly boundary for each timestamp that reflects the deviation of each timestamp from the normal timestamps in the time series. Given the normalized input time series $\mathcal{X}_{norm} \in \mathbb{R}^{N \times C}$, we calculate its anomaly boundary ($\mathcal{AnoB}$) as follows.

$$\mathcal{AnoB} = \sum_{i=1}^{C} \left| \frac{\mathcal{X}_{norm}^i - \mathbb{E}(\mathcal{X}_{norm}^i)}{\sqrt{Var(\mathcal{X}_{norm}^i) + \epsilon}} \right|. \tag{15}$$

The reason for using this as the anomaly boundary is that we aim to employ a prior metric that measures the consistency of a certain timestamp in the given window, indicating whether a timestamp deviates from the given window. The standard deviation of the mean is a general way to achieve this. Therefore, we include the absolute value in the standard deviation to reflect the prior anomaly boundary. Finally, we define the final loss of representation network as:

$$\mathcal{L} = \begin{cases} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{DM}_{i,j}, & \mathcal{AnoB} \leq \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{DM}_{i,j} \\ \mathcal{AnoB} - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{DM}_{i,j}, & \mathcal{AnoB} > \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{DM}_{i,j} \end{cases} \tag{16}$$

By considering $\mathcal{AnoB}$ as a boundary in the final loss, the model does not assume that all timestamps have a normal pattern during optimization. There is more optimization space for normal timestamps in the training data, and less optimization space for abnormal timestamps. When optimizing the model, the optimizer will take all timestamps into account to take a step to minimize the loss. As a result, the loss function more likely focuses on the timestamps with a larger optimization space rather than optimizing all timestamps towards $\mathcal{AnoB}$.

4.5 Anomaly Criterion

Given a test time series $\mathcal{X}_{test} \in \mathbb{R}^{N \times C}$, to obtain the final results, we assign an anomaly score to each timestamp. Based on this anomaly score, we then filter anomalies from normal instances.

Table 2: Performance results of the model. We evaluated Precision (P), Recall (R), and F1-score on five datasets. All results are presented in percentages. The best F1 score is highlighted in bold, and the second is underlined.

Dataset	Flink			SMD			SMAP			SWaT			PSM			MSL		
Metric	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LOF	69.35	15.45	25.27	56.34	39.86	46.68	58.93	56.33	57.60	72.15	65.43	68.62	57.89	90.49	70.61	47.72	85.25	61.18
Matrix Profile	8.30	13.52	10.58	55.36	15.49	24.21	93.18	73.81	82.37	77.21	47.20	58.58	-	-	-	92.70	57.12	70.68
TS-CP2	24.32	31.69	27.52	87.42	66.25	75.38	87.65	83.18	85.36	81.23	74.10	77.50	82.67	78.16	80.35	86.45	68.48	76.42
U-Time	19.83	23.27	21.41	65.95	74.75	70.07	49.71	56.18	52.75	46.20	87.94	60.58	82.85	79.34	81.06	57.20	71.66	63.62
GrammarViz	23.12	18.91	20.08	4.26	1.0	8.18	39.78	22.60	28.83	94.65	72.38	82.03	90.94	81.35	85.88	14.47	44.10	21.79
OCSVM	3.39	31.82	6.13	44.34	76.72	56.19	53.85	59.07	56.34	45.39	49.22	47.23	62.75	80.89	70.67	59.78	86.87	70.82
IForest	39.63	80.30	53.07	42.31	73.29	53.64	52.39	59.07	55.53	49.29	44.95	47.02	76.09	92.45	83.48	53.94	86.54	66.45
Deep-SVDD	13.64	12.93	13.28	78.54	79.67	79.10	89.93	56.02	69.04	80.42	84.45	82.39	95.41	86.49	90.73	91.92	76.63	83.58
LSTM-VAE	23.56	67.64	34.95	75.76	90.08	82.30	92.20	67.75	78.10	76.00	89.50	82.20	73.62	89.92	80.96	85.49	79.94	82.62
BeatGAN	38.39	6.19	10.68	72.90	84.09	78.10	92.38	55.85	69.61	64.01	87.46	73.92	90.30	93.84	92.04	89.75	85.42	87.53
NCAD	11.23	60.05	18.92	-	-	80.16	-	-	94.45	-	-	95.28	-	-	-	-	-	95.60
AnomalyTrans	12.50	12.31	12.40	88.47	92.28	90.33	93.59	99.41	96.41	89.10	99.28	94.22	96.94	97.81	97.37	91.92	96.03	93.93
DCdetector	35.73	11.21	17.07	83.59	91.10	87.18	94.32	98.64	96.43	93.22	99.77	96.38	97.24	98.11	97.67	92.37	97.69	94.96
ContraAD (Ours)	60.98	70.16	**65.25**	86.73	98.39	**92.19**	96.02	98.71	**97.35**	96.27	98.96	**97.60**	98.27	98.70	**98.49**	93.07	98.55	**95.73**

According to our trained model, we calculate the p-norm for each timestamp's representation in $\mathcal{X}_{test}$, and derive $\mathcal{DM}$ to measure the consistency of timestamps in the time series. We calculate the anomaly score for the k-th timestamp x_k as the sum of the k-th column in $\mathcal{DM}$.

$$AnomalyScore(x_k) = \sum_{i=1}^{N} \mathcal{DM}\left[i,k\right], \tag{17}$$

After obtaining an anomaly score for each timestamp, we use a threshold filtering method [39, 42] to determine whether a timestamp is an anomaly. If the anomaly score for a timestamp is greater than the threshold σ, then that timestamp is considered anomalous; conversely, if the anomaly score is less than the threshold σ, then the timestamp is considered normal. The σ is a pre-defined parameter that should be tuned for each dataset.

$$\hat{y_i} = \begin{cases} 1 : Anomaly & AnomalyScore(x_i) \geq \sigma \\ 0 : Normaly & AnomalyScore(x_i) < \sigma \end{cases}. \tag{18}$$

5 EXPERIMENTS

We conducted extensive experiments to validate our approach. For implementation details please refer to Appendix **??** [48].

5.1 Settings

5.1.1 Datasets. To show the scalability of our method, we validated the performance of our approach not only on our Flink dataset but also on seven benchmark datasets including SMD [31], SMAP [13], SWaT [20], PSM [1], NIPS-TS-GECCO [17], NIPS-TS-SWAN [2] and MSL [13]. Due to page limitation, please refer to Appendix [48] for details of all datasets.

5.1.2 Baselines and Metrics. We compare our model with a total of ten baseline methods, please refer to Appendix **??** [48]. including Matrix Profile [45], U-Time [23], TS-CP2 [11], GrammarViz [27], LSTM-VAE [22], BeatGAN [47], Anomaly Transformer [39], LOF [3], Deep-SVDD [25], OCSVM [26], IForest [18], NCAD [6] and DCdetector [42]. In addition, we also validated our model on various metrics as used by previous work does [42].

5.2 Result Analysis

Table 2 shows the overall results on six datasets, including Flink, SMD, SMAP, SWaT, MSL, and PSM, and report Precision (P), Recall (R), and F1-score. To fairly compare with the state-of-the-art methods of DCdetector and Anomaly Transformer, we report results on the NIPS-TS-GECCO and NIPS-TS-SWAN datasets with more metrics. Results are reported in Appendix [48] Table **??**.

Key observations are summarized as follows. First, in our Flink dataset, all existing methods, especially those based on deep learning, do not perform well. These methods fail to predict the anomalies due to the noise in the training dataset and the disadvantage of detecting the Flink-specific anomalies. However, our method demonstrates better performance, indicating that our approach is effective in detecting Flink-specific anomalies and handling noisy training datasets. Second, on public datasets, our method has also achieved the state-of-the-art results which outperform the previous method, *e.g.,* DCdetector. This indicates that our approach is capable of detecting point-level anomalies as well. Last, we notice a significant improvement in our model across traditional metrics (P, R, F1), *e.g.,* a 23.53% increase over DCdetector and a 43.18% increase over AnomalyTransformer in F1-score on NIPS-TS-GECCO dataset. At the same time, our model exhibits excellent performance on new metrics, and it outperforms previous models in almost all of the metrics. These observations collectively indicate that our method is not only effective in addressing the Flink hotspot detection problem but also demonstrates versatility and robustness by adapting well to existing anomaly patterns. Here, we would like to provide an intuitive explanation for the performance and generality of ContraAD. We calculate the distance of a specific timestamp with the rest timestamp and using the summation of the distances as the anomaly score to distinguish the anomaly. This capability holds true for both Flink and other anomaly detection scenarios. Besides, we assign a soft label for each timestamp to conduct a soft optimization which can migrate the impact of noise in training set.

5.3 Further Experiments

5.3.1 Ablation Study. In this section, we evaluated the effectiveness of various modules in our model, including global attention, local attention, cross discrepancy, and prior knowledge. Results

Table 3: Ablation study. $\mathcal{DM}$ denotes the distance matrix, $\mathcal{AnoB}$ denotes the anomaly boundary, *Global* and *Local* denotes the different views of representation network respectively.

Module				Flink			SWaT			SMAP			PSM		
$\mathcal{DM}$	$\mathcal{AnoB}$	*Global*	*Local*	P	R	F1	P	R	F1	P	R	F1	P	R	F1
✓	✗	✓	✓	55.23	47.98	51.35	96.00	94.97	95.48	95.71	98.93	97.29	97.20	97.00	97.10
✗	✓	✓	✓	40.73	20.98	27.69	90.23	93.13	91.66	93.10	95.13	94.10	96.19	96.78	96.48
✓	✓	✓	✗	57.23	62.31	59.66	94.94	95.86	95.39	96.47	98.00	97.23	97.81	85.08	96.43
✓	✓	✗	✓	55.73	63.20	59.23	96.04	92.42	94.20	95.40	87.71	91.40	97.83	95.65	96.73
✓	✗	✗	✗	34.32	43.82	38.49	92.20	90.07	91.12	89.97	85.73	88.3	92.37	91.89	92.12
✗	✓	✗	✗	23.12	18.27	20.41	91.32	87.23	89.23	91.23	84.72	88.39	93.12	92.53	92.82
✓	✓	✗	✗	47.45	49.53	48.47	95.70	90.21	92.87	92.89	85.23	88.90	96.05	90.50	93.19
✓	✓	✓	✓	**60.98**	**70.16**	**65.25**	**96.27**	**98.97**	**97.60**	**95.80**	**99.21**	**97.48**	**98.28**	**98.70**	**98.49**

are reported in Table 3. First, it can be observed that the attention from the two views complements each other, and combining both of them leads to a better representation for each timestamp, thus producing a better result. Second, it can be observed that our cross discrepancy, compared to optimizing discrepancies in different representations for the same timestamp, leads to significant improvement. Third, in the case of training datasets with anomalies, our anomaly boundary can significantly alleviate this effect, giving rise to in a better overall performance. To better show the need of attention mechanism to capture the pattern of the given time series and separating out the improvement of each module. we use a MLP-based model equipped with the $\mathcal{AnoB}$ and $\mathcal{DM}$ module to conduct the ablation study. The result is shown in Table 3. As shown in the table, applying the$\mathcal{AnoB}$ and $\mathcal{DM}$ modules to an MLP-based model also yields certain performance gains. Furthermore, using the attention mechanism to represent each timestamp, compared to an MLP, results in a significant improvement. This highlights the effectiveness of attention in capturing richer representations for anomaly detection.

5.3.2 Other analysis. The efficiency analysis can be found in Appendix **??**. We validate the performance of our method on anomalies in the Flink dataset, as shown in Appendix Figure **??**. Comparing with Figure 2, it can be observed that despite the potential inclusion of anomalies in the training dataset, our method demonstrates the capability to detect the Flink-specific anomaly patterns. We conduct sensitivity experiment on various parameters of the model, which is shown in Appendix [48] Figure **??**. We discuss why our method has broad applicability to other benchmark datasets in Appendix **??** [48].

5.4 Real-world deployment

5.4.1 Offline training. In our monitoring system, we collect metrics data from all nodes in the Flink cluster every 20 seconds and store it into a database. When training is needed, we retrieve the latest three days of data from the database for the cluster and use it for training. Through training, we obtain a trained representation model, a threshold σ, and the optimal detect window size $\mathcal{N}$. We store the model and the threshold for online detection.

5.4.2 Online detection. The Flink system collects machine metrics every 20 seconds and we use the collected metrics for anomaly detection. The anomaly detection process is triggered every minute. Each time, we pass the latest collected $\mathcal{N}$ timestamps into the representation model and calculate the anomaly score for each time step. Then, we compare it with the offline threshold σ to determine whether that timestamp is anomalous. If a specific node is determined to be anomalous, an alert is triggered in our monitoring system.

5.4.3 Post-launch performance. We deploy our model and a statistic model as mentioned in Section 3, integrating two models into the Flink production scenario. Once the two models detect hotspot anomalies, they will trigger alert notification. Our model has been successfully deployed online for several months. During this period, We conduct multiple evaluations, using a three-day period as the evaluation interval. Our Flink experts collect all exact hotspot issues within three days using their domain knowledge to identify real hotspots and compare all the alert notifications from the two models. We report the average result in Appendix Table **??**. It shows that our model can effectively diminish the likelihood of erroneously removing machines due to false identification and detect potential hotspot issues earlier and more comprehensively.

6 CONCLUSION AND LEARNED LESSON

In this paper, we propose an unsupervised time series anomaly detection method based on contrastive learning. Our approach leverages contrastive learning among different timestamps, endowing it with the capability to address the Flink-specific anomalies. Additionally, we incorporate prior knowledge to mitigate the impact of a noisy training dataset. Our approach proves highly effective not only for the Flink-specific anomalies but also achieves state-of-the-art results on public datasets.

Learned Lesson: Data matters. Existing unsupervised anomaly detection methods in academia mostly rely on high-quality datasets *e.g.,* a noise-free dataset, which are often difficult or costly to obtain in real-world applications. In the real-world applications, there is more focus on how to achieve optimal results without incurring significant costs *e.g.,* labor cost. Thus, finding solutions to mitigate dataset issues *e.g.,* a noisy dataset, without additional human resources is more practically significant in real-world scenarios.

ACKNOWLEDGMENTS

This work was supported by Alibaba Innovative Research Program.

REFERENCES

[1] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. 2021. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *SIGKDD*. 2485–2494.

[2] Rafal Angryk, Petrus Martens, Berkay Aydin, Dustin Kempton, Sushant Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Xumin Cai, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Micheal Schuh, and Manolis Georgoulis. 2020. SWAN-SF. https://doi.org/10.7910/DVN/EBCFKM

[3] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *SIGMOD*. 93–104.

[4] David Campos, Bin Yang, Tung Kieu, Miao Zhang, Chenjuan Guo, and Christian S. Jensen. 2024. QCore: Data-Efficient, On-Device Continual Calibration for Quantized Models. *Proc. VLDB Endow.* 17, 11 (2024), 2708–2721.

[5] David Campos, Miao Zhang, Bin Yang, Tung Kieu, Chenjuan Guo, and Christian S. Jensen. 2023. LightTS: Lightweight Time Series Classification with Adaptive Ensemble Distillation. *Proc. ACM Manag. Data* 1, 2 (2023), 171:1–171:27.

[6] Chris U. Carmona, François-Xavier Aubet, Valentin Flunkert, and Jan Gasthaus. 2022. Neural Contextual Anomaly Detection for Time Series. In *IJCAI*, Lud De Raedt (Ed.). IJCAI Organization, 2843–2851. https://doi.org/10.24963/ijcai.2022/394 Main Track.

[7] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems* 30 (2017).

[8] Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. 2024. Pathformer: Multi-scale Transformers with Adaptive Pathways for Time Series Forecasting. In *ICLR*.

[9] Yunyao Cheng, Peng Chen, Chenjuan Guo, Kai Zhao, Qingsong Wen, Bin Yang, and Christian S. Jensen. 2023. Weakly Guided Adaptation for Robust Time Series Forecasting. *Proc. VLDB Endow.* 17, 4 (2023), 766–779.

[10] Zahra Zamanzadeh Darban, Geoffrey I Webb, and Mahsa Salehi. 2024. DACAD: Domain Adaptation Contrastive Learning for Anomaly Detection in Multivariate Time Series. *arXiv preprint arXiv:2404.11269* (2024).

[11] Shohreh Deldari, Daniel V. Smith, Hao Xue, and Flora D. Salim. 2021. Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM. https://doi.org/10.1145/3442381.3449903

[12] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems* 31 (2018).

[13] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *SIGKDD*. 387–395.

[14] Neha Kant and Manish Mahajan. 2019. *Time-Series Outlier Detection Using Enhanced K-Means in Combination with PSO Algorithm: ICoEVCI 2018, India*. 363–373. https://doi.org/10.1007/978-981-13-1642-5_33

[15] Paweł Karczmarek, Adam Kiersztyn, Witold Pedrycz, and Ebru Al. 2020. K-Means-based isolation forest. *Knowledge-based systems* 195 (2020), 105659.

[16] Tung Kieu, Bin Yang, Chenjuan Guo, Christian S Jensen, Yan Zhao, Feiteng Huang, and Kai Zheng. 2022. Robust and explainable autoencoders for unsupervised time series outlier detection. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 3038–3050.

[17] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. 2021. Revisiting Time Series Outlier Detection: Definitions and Benchmarks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, J. Vanschoren and S. Yeung (Eds.), Vol. 1. Curran. https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/ec5decca5ed3d6b8079e2e7e7bacc9f2-Paper-round1.pdf

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *ICDM*. 413–422. https://doi.org/10.1109/ICDM.2008.17

[19] Eran Malach and Shai Shalev-Shwartz. 2017. Decoupling" when to update" from" how to update". *Advances in neural information processing systems* 30 (2017).

[20] Aditya P. Mathur and Nils Ole Tippenhauer. 2016. SWaT: a water treatment testbed for research and training on ICS security. In *CySWater*. 31–36.

[21] Steffen Moritz, Frederik Rehbach, and Thomas Bartz-Beielstein. 2020. *GECCO Industrial Challenge 2019 Dataset: A water quality dataset for the 'Internet of Things: Online Event Detection for Drinking Water Quality Control' competition at the Genetic and Evolutionary Computation Conference 2019, Prague, Czech Republic.*

[22] Zijian Niu, Ke Yu, and Xiaofei Wu. 2020. LSTM-based VAE-GAN for time-series anomaly detection. *Sensors* 20, 13 (2020), 3738.

[23] Mathias Perslev, Michael Jensen, Sune Darkner, Poul Jø rgen Jennum, and Christian Igel. 2019. U-Time: A Fully Convolutional Network for Time Series Segmentation Applied to Sleep Staging. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/57bafb2c2dfeefba931bb03a835b1fa9-Paper.pdf

[24] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. *Proc. VLDB Endow.* 17, 9 (2024), 2363–2377.

[25] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Lucas Deecke, Shoaib A. Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep One-Class Classification. In *ICML*, Vol. 80. 4393–4402.

[26] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.

[27] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P Boedihardjo, Crystal Chen, and Susan Frankenstein. 2015. Time series anomaly discovery with grammar-based compression.. In *Edbt*. 481–492.

[28] Lifeng Shen, Zhuocong Li, and James Kwok. 2020. Timeseries anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems* 33 (2020), 13016–13026.

[29] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2022. Learning from Noisy Labels with Deep Neural Networks: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[30] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. RoFormer: Enhanced Transformer with Rotary Position Embedding. *CoRR* abs/2104.09864 (2021). arXiv:2104.09864 https://arxiv.org/abs/2104.09864

[31] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *SIGKDD*. 2828–2837.

[32] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *SIGKDD*. 2828–2837.

[33] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. 2002. Enhancing effectiveness of outlier detections for low density patterns. In *PAKDD*. Springer, 535–548.

[34] David MJ Tax and Robert PW Duin. 2004. Support vector data description. *Machine learning* 54 (2004), 45–66.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[36] Qizhou Wang, Bo Han, Tongliang Liu, Gang Niu, Jian Yang, and Chen Gong. 2021. Tackling instance-dependent label noise via a universal probabilistic model. In *AAAI*, Vol. 35. 10183–10191.

[37] Xinle Wu, Xingjian Wu, Bin Yang, Lekui Zhou, Chenjuan Guo, Xiangfei Qiu, Jilin Hu, Zhenli Sheng, and Christian S. Jensen. 2024. AutoCTS++: zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting. *VLDB J.* 33, 5 (2024), 1743–1770.

[38] Xinle Wu, Dalin Zhang, Miao Zhang, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2023. AutoCTS+: Joint Neural Architecture and Hyperparameter Search for Correlated Time Series Forecasting. *Proc. ACM Manag. Data* 1, 1 (2023), 97:1–97:26.

[39] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. In *International Conference on Learning Representations*. https://openreview.net/forum?id=LzQQ89U1qm_

[40] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. 2019. Outlier detection: how to threshold outlier scores?. In *International Conferences on Artificial Intelligence, Information Processing and Cloud Computing*. https://api.semanticscholar.org/CorpusID:209168456

[41] Sean Bin Yang, Chenjuan Guo, Jilin Hu, Jian Tang, and Bin Yang. 2021. Unsupervised Path Representation Learning with Curriculum Negative Sampling. In *IJCAI*. 3286–3292.

[42] Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. 2023. DCdetector: Dual Attention Contrastive Representation Learning for Time Series Anomaly Detection. In *SIGKDD (KDD '23)*. ACM.

[43] Yu Yao, Tongliang Liu, Bo Han, Mingming Gong, Jiankang Deng, Gang Niu, and Masashi Sugiyama. 2020. Dual t: Reducing estimation error for transition matrix in label-noise learning. *Advances in neural information processing systems* 33 (2020), 7260–7271.

[44] Yazhou Yao, Zeren Sun, Chuanyi Zhang, Fumin Shen, Qi Wu, Jian Zhang, and Zhenmin Tang. 2021. Jo-src: A contrastive approach for combating noisy labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5192–5201.

[45] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Anh Dau, Diego Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. 1317–1322. https://doi.org/10.1109/ICDM.2016.0179

[46] Kai Zhao, Chenjuan Guo, Yunyao Cheng, Peng Han, Miao Zhang, and Bin Yang. 2023. Multiple Time Series Forecasting with Dynamic Graph Modeling. *Proc. VLDB Endow.* 17, 4 (2023), 753–765.

[47] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. 2019. Beatgan: Anomalous rhythm detection using adversarially generated time series.. In *IJCAI*, Vol. 2019. 4433–4439.

[48] Zhihao Zhuang, Yingying Zhang, Kai Zhao, Chenjuan Guo, Bin Yang, Qingsong Wen, and Lunting Fan. 2024. Appendix. https://anonymous.4open.science/r/ContraAD-265F/VLDB_Anomaly_detection_appendix.pdf