Online Data Drift Detection for Anomaly Detection Services based on Deep Learning towards Multivariate Time Series

Gou Tan[1], Pengfei Chen[2,*], and Min Li[1]
[1]School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
tang29@mail2.sysu.edu.cn, chenpf7@mail.sysu.edu.cn, limin258@mail2.sysu.edu.cn
*corresponding author

Abstract—Deep learning models have been successfully adopted in anomaly detection for multivariate time series data in various fields. These models are good at capturing complex time dependencies and extracting meaningful patterns from time series data. However, the trained models may become outdated due to unforeseen changes in real-world data, which can lead to a decrease in the quality of model service. Therefore, it is crucial to continuously monitor the performance of the model and analyze its behavior to ensure its reliability and availability. We propose an online data drift detection method that uses an unsupervised deep learning network, Variational Autoencoder (VAE), to monitor deep learning models in the field of multivariate time series anomaly detection. This method consists of three main steps namely data collection and statistical analysis, real-time drift detection, and drift interpretation. We collect raw time series data and model prediction data non-invasively from the model server. Then they are separated into windows for drift detection. Furthermore, the method can provide analysis and interpretation when drift is detected. Our evaluation experiments involve three real-world datasets from various industrial domains and four different structured anomaly detection models. We validate the effectiveness of drift detection in multivariate time series, and then test how the anomaly detection models perform during data drift detection. The highest improvement in F1 score is approximately 0.16. In addition, we provide an analysis of the interpretability of the model performance.

Keywords–drift detection; interpretability; anomaly detection services; deep learning; multivariate time series; monitor

1. Introduction

In recent years, deep learning models have made significant advancements in anomaly detection (AD) for multivariate time series (MTS). These models are widely used in various fields such as finance, healthcare, and industrial monitoring to detect anomalies [1], [2]. Deep learning networks such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have demonstrated their ability to capture complex temporal dependencies and extract meaningful representations from MTS. Their automatic learning capabilities make them well suited for anomaly detection tasks, allowing them to learn patterns and detect deviations from normal behavior [3]. However, deploying deep learning-based anomaly detection services in real-world scenarios requires more than just training and evaluation. It is critical to continuously monitor the model's performance, analyze its behavior, and identify potential problems.

In the field of Machine Learning Operations (MLOps), training data and models can become outdated due to unforeseen changes in data and real-world conditions. This can lead to a degradation of the model's inference performance [4]. MLOps monitoring focuses on the performance of models in production environments and aims to identify model performance degradation, concept drift, and other anomalies that may affect the reliability and effectiveness of models [5].

For anomaly detection services based on deep learning, monitoring MTS involves several aspects. Firstly, dynamic data needs to be continuously monitored to detect potential patterns or changes in data distribution. Secondly, the performance of deep learning models can degrade due to data drift and other factors [6]. This paper focuses on monitoring raw data (input and output), statistical values (such as mean and standard deviation), and data drift (including covariate drift and concept drift) [4]. (1) Covariate drift refers to a difference in the distribution between production data and training data [7]. (2) Concept drift refers to the evolution of the relationship between input and output over time [8].

When monitoring anomaly detection services for MTS in real-world scenarios, several challenges need to be addressed.

- **The Quality of Model Services Lacks Standards.** While latency and overhead are commonly used to evaluate system quality, there is a lack of uniform standards for evaluating the quality of model services. We propose the use of raw data (input and output) and statistical values as metrics for evaluating model service quality.
- **Lack of Data Drift Labels.** When the model service is online, it is difficult to obtain ground truth in time [9]. Therefore, it is challenging to determine whether data drift has occurred. Traditional supervised and semi-supervised methods are not suitable for data drift detection in this context. To solve this problem, we need to apply unsupervised data drift detection methods and improve detection accuracy by reducing noise alerts.
- **The Detection Method Needs to be Updated in Time.** Traditional time series drift detection methods typically involve dynamic fitting based on historical data. These methods tend to be relatively simple, and their performance

2693-9177/23/$31.00 ©2023 IEEE
DOI 10.1109/QRS60937.2023.00011

relies on the quality of the historical data. However, when conducting online data drift detection for anomaly detection services, it is crucial to update the detection method in real time. This approach helps minimize the need for manual adjustments and reduces model maintenance costs.

- **Interpretability.** When data drift is detected, it is crucial to provide accurate interpretations of the model's service quality [10]. Simply applying drift detection algorithms to a single type of data is insufficient for interpreting drifts. Therefore, it is necessary to develop a method that can effectively detect abnormal situations in model services and offer reasonable interpretations.

To address the above challenges, our goal is to develop an accurate, efficient, and scalable drift detection method for drift detection in MTS anomaly detection services. We propose an online data drift detection method for anomaly detection services based on deep learning, as shown in Figure 3. (1) The method collects raw time series data and model prediction data from the anomaly detection service. (2) It applies an unsupervised MTS drift detection method based on VAE to detect real-time changes in the data. (3) When a drift is detected, we provide a reasonable interpretation by incorporating statistical data and drift metrics. The main contributions of this paper include:

- We have developed a novel data-driven monitoring scheme for MTS anomaly detection services. By analyzing data in real time, including windowed data, differences between reconstructed data and original data, the anomaly ratio of predictions, and their statistical values, we evaluate the service quality of the model and monitor anomaly detection models in production environments.
- Our solution is one of the earliest attempts to apply deep learning-based drift detection algorithms to MLOps monitoring. It achieves high-performance analysis through unsupervised deep learning without requiring manual labeling.
- During the drift interpretation stage, we interpret the changes in service quality with data information. We have successfully designed a method that accurately detects data drift in services based on deep learning and provides reasonable interpretations.

The rest of this paper is organized as follows. Section 2 describes the background of relevant researches including MLOps, MTS anomaly detection, and drift detection. Section 3 introduces the technical architecture of our online data drift detection approach for anomaly detection services. In Section 4, we evaluate our online data drift detection method quantitatively and qualitatively by using different datasets and deep learning models for anomaly detection. Section 5 provides an overview of related works. Finally, Section 6 concludes.

2. Background

2.1 MLOps

In recent years, a number of standardized MLOps platforms have been developed to accelerate the development process of machine learning algorithms and to support the large-scale deployment and maintenance. Popular platforms include AWS's SageMaker, Google's TensorFlow Extended (TFX), Microsoft's Azure ML, Kubeflow, Metaflow [11]. The lifecycle of the ML model is shown in the Figure 1.

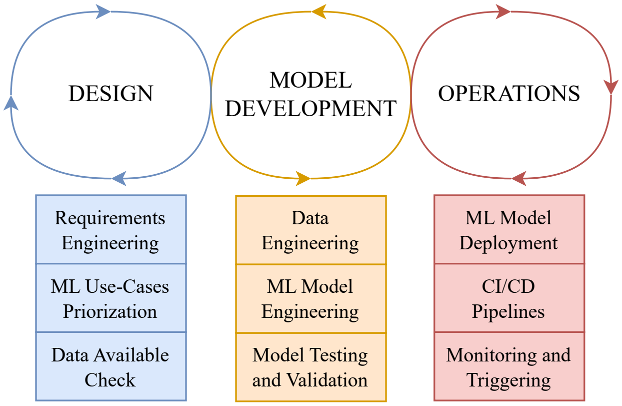


Figure 1: Lifecycle of the ML model

Model Training and Deployment. A common task in MLOps is training and deploying models for accurate inference. These platforms are designed to assist data scientists and developers in quickly training and deploying high-quality ML models. They provide fully managed tools for each step of ML development, including data preparation, feature engineering, training, validation, etc. Once the model is trained, it can be deployed in a production environment for prediction serving [12]. Model servers, such as TorchServe (PyTorch community), TensorFlow Serving (Google), ONNX Runtime (Microsoft), and KFServing (Kubeflow community), are used for deploying and managing machine learning models [13]. They offer a convenient and efficient way to apply trained models in production environments, supporting various model formats and deployment methods (e.g., HTTP, REST API, gRPC API).

Model Monitoring. Model monitoring aims to achieve a closed-loop ML lifecycle in MLOps by continuously monitoring deployed models to ensure their performance and reliability [5]. However, the importance of managing the service quality of models and accurately monitoring their key metrics is often ignored. Model monitoring is essential to prevent model degradation [4].

2.2 Drift Detection

2.2.1 Definition of Data Drift

Assuming that an ML model is trained on a source distribution S and predicts on a target distribution T, it can be expressed as the joint distribution $P(x, y)$ of the input data x and the label y, which can be further decomposed as

$$P(x, y) = P(x)P(y|x) \tag{1}$$

The drift of ML can be defined as the change in the joint distribution between training and prediction stages, where $P(x_S, y_S)$ is not equal to $P(x_T, y_T)$ [14]. According to the formula 1, researchers divided drift into two types: covariate drift and concept drift.

- **Covariate Drift.** The distribution of input features changes, and the relationship between input and output remains unchanged.

$$P(x_S) \neq P(x_T) \text{ and } P(y_S|x_S) = P(y_T|x_T) \quad (2)$$

- **Concept Drift.** The distribution of input features remains unchanged, and the relationship between the input and output of the model changes.

$$P(y_S|x_S) \neq P(y_T|x_T) \text{ and } P(x_S) = P(x_T) \quad (3)$$

In reality, drift changes with time in different representations, as shown in Figure 2. For example, it can manifest as a gradual or an incremental drift over a long period of time (months or even years), a sudden drift driven by some external events, or a reoccurring drift due to seasonal and other reasons [15].

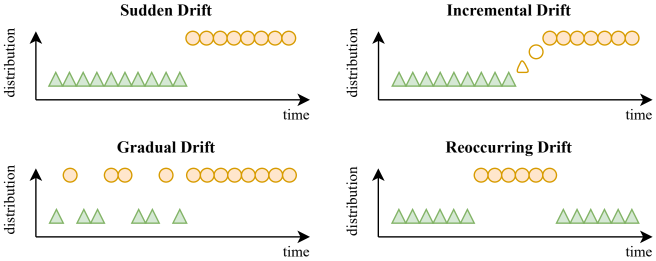


Figure 2: Representation of data drift

2.2.2 Drift Detection Methods

Drift detection methods can be roughly divided into supervised and unsupervised technology. Supervised technology aims to detect changes in error rates (e.g., DDM, EDDM, HDDM) [14]. Although supervised methods are efficient, they are not applicable in real-world situations due to the need for ground truth.

On the other hand, unsupervised technology focuses on monitoring the data distribution and detecting changes in the distribution of data over time. Typically, streaming data is divided into time windows, and the distribution of the current time window is compared against a reference one from the training data [16]. The advantage of unsupervised methods is that they do not need to label data during testing, and the computation speed is faster. How to compare the distribution of MTS is the primary issue of unsupervised technology, and there are several popular methods available:

- **Statistical Methods.** These methods compare the distribution of feature values between two sets of data and evaluates whether the difference is statistically significant. A commonly used statistical test is the Kolmogorov-Smirnov test [17].
- **Distance-Based Methods.** These methods measure the distance between the feature distributions of two data sets and trigger a drift alert when the predefined threshold is exceeded. Commonly used distance metrics include the Kullback-Leibler divergence [18] and the Hellinger distance [19].
- **Ensemble Methods.** These methods use different subsets of data or different algorithms to train multiple models [20], [21]. By comparing the model prediction results of two data sets on the same instances, we can test for significant drift.

Anomaly detection services may be affected by data drift. MTS anomaly detection services based on deep learning learn normal time series and identify deviating time series as outliers. However, such models may incorrectly identify unknown normals as anomalies. Anomaly detection is also a classification problem that involves distinguishing between normals and anomalies [22]. And the training data may not cover all normal situations. Because its training uses only one type of data, it weakens the learning effectiveness. We focus on real scenarios and use unsupervised drift detection, as shown in Figure 3.

2.3 Time Series Anomaly Detection

The general framework of anomaly detection consists of the following three steps:

- Modeling and learning a representation of the data as shown in the formula 4. In this step, the model M is trained using the preprocessed features $F(\cdot)$.

$$M = Train(F(\cdot)) \quad (4)$$

  Here, $F(\cdot) = \{F(X_1), F(X_2), \ldots, F(X_n)\}$ represents the features extracted from the time series X_i.
- Calculating the anomaly score as shown in the formula 5. The anomaly score is calculated based on the generation result $G(\cdot)$ and the preprocessed features $F(\cdot)$.

$$S = Score(G(\cdot), F(\cdot)) \quad (5)$$

  Here, $G(\cdot) = \{G(X_1), G(X_2), \ldots, G(X_n)\}$ represents the generation result obtained by using the model M.
- Setting the anomaly detection threshold as shown in the formula 6. The anomaly flag is determined based on the anomaly score S and a predefined threshold.

$$A = Detect(S, threshold) \quad (6)$$

The first step is to extract important information from time series, such as seasonality and time dependence. The traditional method is regression prediction such as ARMA and ARIMA [23]. Recently, deep learning has also shown excellent performance with popular networks LSTM and CNN [24].

The second step aims to obtain more uniformly and independently distributed errors by calculating the difference between observation and prediction.

The third step involves setting a threshold based on the anomaly score to detect abnormal points. A widely used method in the industry is to calculate the prediction residual from the first step. Then, the 3σ method or setting a p-value threshold can be applied [25]. Another approach is to map the residuals to the kernel space and use a hyperplane to separate most of the training data points from a few data points that

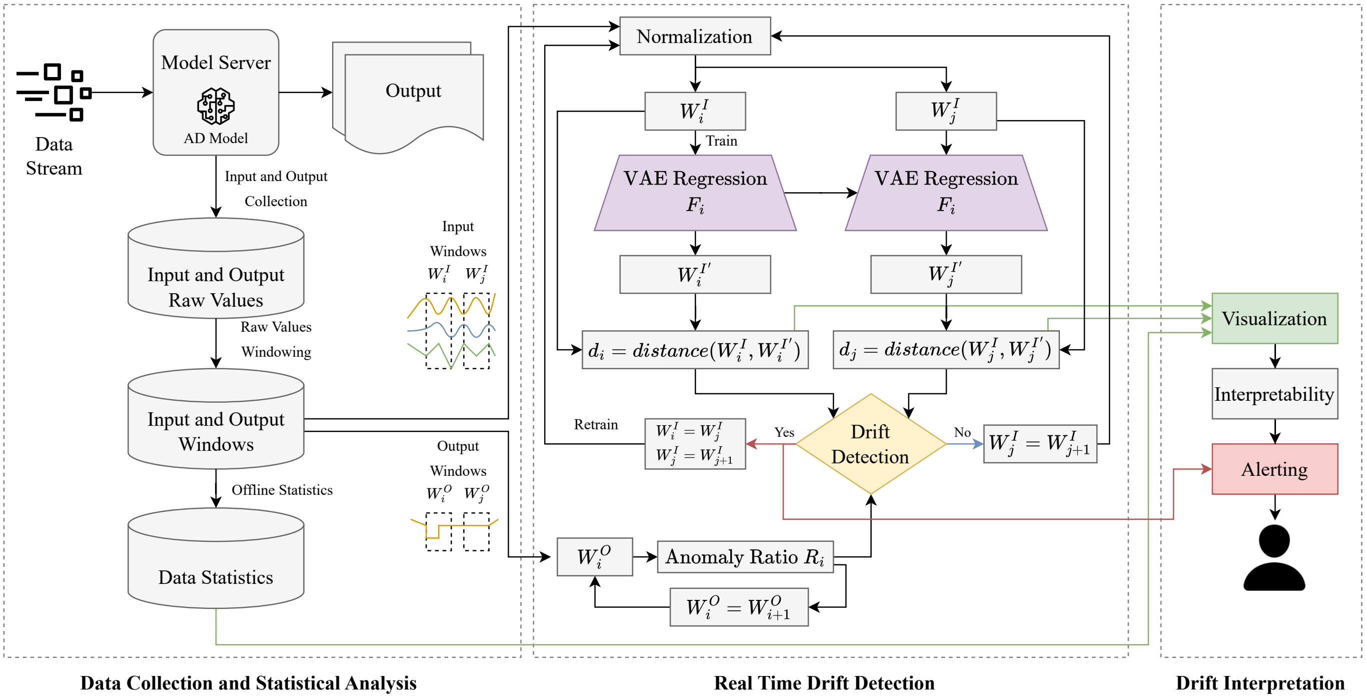


Figure 3: The framework of online data drift detection for anomaly detection services

are considered as exceptions [26]. However, in practice, the unsupervised anomaly detector is easily affected by random noise, leading to many false alarms [27].

As shown in the figure 4, MTS is the continuously sampled multivariate observations $X_N = \{x_N^1, x_N^2, \dots, x_N^M\}$, where M and N are metric and time respectively. MTS data typically exhibits time dependency within a single dimension (e.g., periodicity and seasonality of CPU utilization) and inter dimensional dependency across different dimensions (e.g., positive correlation between TCP traffic and CPU utilization) [28]. The goal of MTS anomaly detection is to determine whether the observed value X_i is abnormal. Furthermore, interpretability involves identifying a set of dimensional metrics that are most likely to cause an anomaly.

MTS anomalies can be roughly divided into three types: temporal anomalies, interdimensional anomalies, and interdimensional temporal anomalies [28]. Temporal anomalies occur when the data of a single dimension deviates significantly from its historical normal pattern. Interdimensional anomalies occur when multiple dimensions exhibit a certain relationship (e.g., positive or negative correlation), but the relationship violates the historical pattern. Interdimensional temporal anomalies involve violations of both interdimensional and time dependencies.

MTS anomaly detection is commonly applied in various domains such as the Internet of Things, spacecraft, and network attacks [24], [29], [30], [31], [32]. RNN and CNN are frequently employed in these applications. Dimension reduction is one of the significant contributions of them [29].

Our method uses an unsupervised nonparametric deep learning, which offers several advantages. The advantage is that we do not need data labels to train the model and avoid manually configuring drift detection rules. This approach is particularly beneficial in practice.

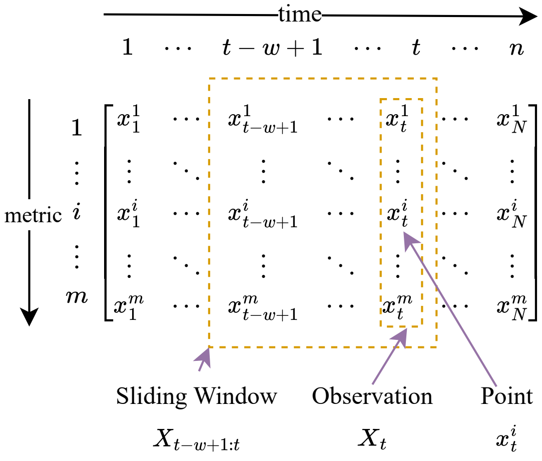

time
1 $\cdots$ $t-w+1$ $\cdots$ t $\cdots$ n
metric
1
$\vdots$
i
$\vdots$
m
x_1^1 $\cdots$ x_{t-w+1}^1 $\cdots$ x_t^1 $\cdots$ x_N^1
x_1^i $\cdots$ x_{t-w+1}^i $\cdots$ x_t^i $\cdots$ x_N^i
x_1^m $\cdots$ x_{t-w+1}^m $\cdots$ x_t^m $\cdots$ x_N^m
Sliding Window
$X_{t-w+1:t}$
Observation
X_t
Point
x_t^i

Figure 4: Data formulation of MTS $X \in R^{m \times n}$ [28]

3. Methodology

In this section, we introduce the drift detection method for anomaly detection services. Figure 3 shows the end-to-end approach for monitoring the quality of the model service, which consists of three main components:

- **Data Collection and Statistical Analysis.** We modify the Python source code of the model server (e.g., TorchServe, TensorFlow Serving). When the model server executes prediction tasks, it automatically collects the input and output data of the model without invasion. We then aggregate and calculate statistical values to reflect the state of the model.

- **Real Time Drift Detection.** We apply deep learning drift detection algorithm to the collected data in order to detect any changes in the model service.
- **Drift Interpretation.** We analyze the impact of each dimensional feature to find the reason triggering the drift. And we can interpret the drift by combining raw values, statistical values and drift detection metrics.

3.1 Data Collection and Statistical Analysis

In order to collect data for monitoring the performance of the model, we modify the Python source code of the model server (e.g., TorchServe, TensorFlow Serving). Taking TorchServe as an example, we modify the function handle(self, data, context) located in "site-packages/ts/torch_handler/base_handler.py". This modification allows us to extract and transfer the input and output data of the model service to other customized components.

By modifying the model server, we can capture the input and output data of the anomaly detection model during the prediction task. The collected data includes the input features provided to the model and the corresponding model predictions or outputs. We aggregate and calculate statistical values, including the average and standard deviation of the input, as well as the proportional changes of the output types.

In order to analyze the behavior of the model within a specific time interval, we utilize the concept of the window. By using the sliding window, we can observe changes in the model's behavior within a defined time interval. It enables us to capture temporal patterns, calculate statistical metrics within each window, and detect potential drift over time. The size of the window depends on the specific requirements of the user. We can adjust it to capture short-term or long-term variations in the model's behavior. The Figure 4 shows the windowed MTS.

3.2 Real Time Drift Detection

In Section 3.1, we have implemented real-time windowing of the raw data. In this section, we focused on analyzing the time series of the windowed input features and the windowed output results separately to detect both covariate drift and concept drift. Our method stands out by combining deep learning with dynamic one-class learning techniques [33] that can adapt and update themselves. This unique approach enable us to effectively analyze complex and dynamic MTS.

3.2.1 Input Windows

We first normalize the data. It is common to standardize the entire time series during training. However, this approach has two risks [25]: (1) The impact of future data. During normalization, the mean and standard deviation are calculated from the entire time series, which means that the current time series is influenced by future data. It is unlikely in reality; (2) The complexity of the input space. Time series with similar trends may be mapped to different data points after normalization, resulting in a complex input space and decreased generalization ability of the model.

To address these issues, we employ a real-time dynamic normalization using sliding windows. We apply the normalization to each dimension of the historical input using the standard zero mean and unit standard deviation normalization. It avoids introducing future time series values. Additionally, by mapping all input features to the same scale, it simplifies the input space and enhances the generalization ability of the model for various types of time series.

The regression network used in our method is versatile and can be any network architecture, such as CNN, LSTM, VAE, etc. We choose the VAE regression network to generate the input windows.

The Variational Auto-Encoder (VAE) is a generative model consisting of an encoder and a decoder. The encoder maps the input data x to a latent space z, while the decoder samples from the latent space z to reconstruct the input data x'. During training, the VAE learns the latent representation of the data by maximizing the variational lower bound.

Assuming the input window $X_{t-w+1:t}$ is an $M \times W$ dimensional matrix, which is flattened into an $M \times W$ dimensional vector x. The latent space z is a K dimensional vector. The encoder maps x to the distribution of z, and the decoder samples from z to reconstruct x. We can get a variational lower bound as shown in the formula 7:

$$\log p_\theta(x) \geq E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \quad (7)$$

- Encoder: $q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma^2_\phi(x))$. ($\mu_\phi(x)$ and $\sigma^2_\phi(x)$ are mean and variance of z respectively, ϕ denotes parameters).
- Decoder: $p_\theta(x|z) = \mathcal{N}(x|\mu_\theta(z), \sigma^2_\theta(z))$. ($\mu_\theta(z)$ and $\sigma^2_\theta(z)$ are the mean and variance of the reconstructed x' respectively, θ denotes parameters).
- Prior distribution: $p(z)$, typically assumed to be a standard normal distribution.

Maximizing the variational lower bound is equivalent to minimizing the reconstruction error and the KL divergence, given by:

$$\begin{aligned}\mathcal{L}(\theta, \phi; x) &= \frac{1}{2}\sum_{i=1}^{D}\left(1 + \log(\sigma^2_\phi(x)) - \mu^2_\phi(x) - \sigma^2_\phi(x)\right) \\ &- \log p_\theta(x|z) \end{aligned} \quad (8)$$

By using the backpropagation algorithm, we can optimize the parameters of the encoder and decoder to learn the latent representation of the window.

As shown in the second block of Figure 3, the drift detection for the input consists of two stages. In the first stage, we employ the regression network VAE to reconstruct the historical window data, capturing the temporal dependencies on a single dimension and the dependencies between each dimension. In the second stage, we calculate the difference between the original window W^I_t and the reconstructed window $W^{I'}_t$ using metrics such as Kullback-Leibler divergence, Euclidean distance, cosine similarity, etc. Based on predefined rules, we determine whether the distribution has changed. If drift is detected, we update the regression network VAE. Otherwise, we continue sliding the window and repeat the detection.

Assuming at time t, the window data is $W_t = X_{t-w+1:t} \in m \times w$, and the regression function is F_t. We can train F_t and reconstruct W_t:

$$F_t = train(VAE, W_t) \tag{9}$$

$$W_t^{'} = F_t(W_t) \tag{10}$$

We can calculate the distance between W_t and $W_t^{'}$ with the euclidean distance as formula 11.

$$d_t(W_t, W_t^{'}) = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{W}(W_{t,ij} - W_{t,ij}^{'})^2} \tag{11}$$

Then we compare d_t with the predefined $threshold$ to determine if drift has occurred.

$$drift \quad = \quad \begin{cases} 1 & if \quad d_t > threshold \\ 0 & otherwise \end{cases} \tag{12}$$

By sliding the window with a step s, we obtain W_{t+s}. If drift is detected at time t, we retrain the regression function $F_{t+s} = train(VAE, W_{t+s})$. Otherwise, we keep it unchanged $F_{t+s} = F_t$. We keep this process cycling.

Finally, we can obtain the d_t of the window in real time. We can use dynamic $threshold$ based on d_t. Here, we use the 3σ method. Assuming the last drift time is t_{ld}.

$$\mu = \frac{1}{t - t_{ld} + 1}\sum_{i=t_{ld}}^{t} d_i \tag{13}$$

$$\sigma = \sqrt{\frac{1}{t - t_{ld} + 1}\sum_{i=t_{ld}}^{t}(d_i - \mu)^2} \tag{14}$$

$$threshold = 3\sigma \tag{15}$$

3.2.2 Output Windows

In reality, we can't obtain the correctness of the predictions of the anomaly detection model. So it is impossible to know the changes in the performance of the model. Inspired by the ADWIN algorithm, we divide the streaming data into real-time windows, continuously calculate the proportion of the model prediction result categories in the window. By monitoring the change in the ratio (R_i, shown in the formula 16) of abnormal labels within the window and combining it with the changes in the input data, we can determine whether drift has occurred.

$$R_i = \frac{sum(Y_{t-w+1:t} == anomaly)}{w} \tag{16}$$

Finally, we simultaneously capture the changes in both the input and output of the anomaly detection service and feed them into the drift detection module for comprehensive analysis.

3.3 Drift Interpretation

The complexity of model services often hinders understanding of the model, making diagnosis and problem-solving difficult. A better comprehension and transparency of the model is crucial for the quality of service.

Firstly, we provide a detailed definition of data drift patterns to better interpret the situation of the model. In Section 2, we defined covariate drift and concept drift. Consider two drift modes: sudden drift and incremental drift. We can define four common drift patterns. Assuming t_{span} represents is a manually set time interval.

- sudden covariate drift:
  $d_{t,changed} \cap (t - t_{ld} < t_{span})$
- incremental covariate drift:
  $d_{t,changed} \cap (t - t_{ld} > t_{span})$
- sudden concept drift:
  $d_{t,unchanged} \cap R_{i,changed} \cap (t - t_{ld} < t_{span})$
- incremental concept drift:
  $d_{t,unchanged} \cap R_{i,changed} \cap (t - t_{ld} > t_{span})$

In the prior steps, we have collected essential information about the model and calculated statistical values and drift detection metrics. When drift is detected, we compare the distance d_t of each dimensional feature. We believe that features with larger distance are more likely to trigger drift. We can analyze these features by combining the raw values and statistical values to interpret the drift. We hope that through interpretation, we can facilitate the maintenance and optimization of anomaly detection services in real-world applications, in order to achieve better performance.

4. Experiments and Analysis

Firstly, we introduce the datasets and metrics used for evaluation. Then, we design experiments to answer the following research questions:

- RQ1: How effective is the drift detection method in MTS?
- RQ2: Is drift detection effective in improving the quality of anomaly detection services based on deep learning? What are the differences in the effectiveness of drift detection when applied to different datasets and different anomaly detection models?
- RQ3: Can we explain the performance of a model based on drift interpretation?

4.1 Datasets and Evaluation Metrics

We used three publicly available datasets, all of which are MTS. Their characteristics are summarized in Table I.

- Server Machine Dataset (SMD): This is a five-week long dataset of stacked traces of the resource utilizations of 28 machines from a compute cluster [34]. We use the nontrivial sequences machine-1-1 in this dataset.
- Mars Science Laboratory (MSL): This dataset corresponds to the sensor and actuator data for the Mars rover itself [32]. We consider only the three nontrivial ones (A4, C2, and T1).

- Secure Water Treatment (SWaT): This dataset is collected from a real-world water treatment plant with 7 days of normal and 4 days of abnormal operation [35].

For anomaly detection, we use precision, recall, area under the receiver operating characteristic curve (ROC/AUC), and F1 score to evaluate the performance of all models [36].

For drift detection, we use VAE for regression to reconstruct the window data, with some hyperparameters as follows: Window size = 200, Sliding stride = 100, VAE hidden_dim = 256, VAE latent_dim = 2, VAE learning_rate = 0.01.

TABLE I. The dataset statistics

Dataset	Train	Test	Dimensions	Anomalies(%)
SMD	28479	28479	38	9.46
MSL	58317	73729	55	10.72
SWaT	99000	89984	51	12.17

4.2 RQ1. The Effectiveness of Drift Detection

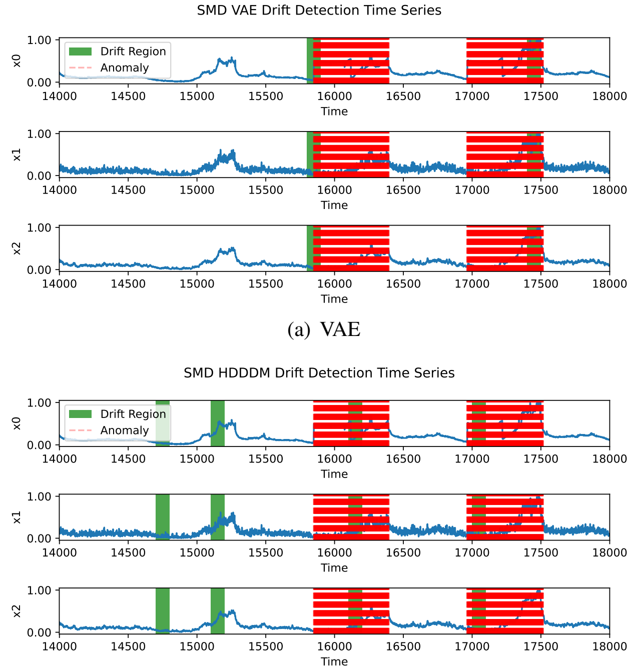
We apply HDDDM [19] (an unsupervised drift detection algorithm on multivariate time series) and our method to detect drifts in the three MTS datasets. The detected number of drifts is shown in the Table II. We found that all of them have data drift. Especially when there are more anomalies, drift occurs more frequently. The results of the two methods are shown in Figure 5. Due to the large number of features and data, we select the first 3 dimensions and 10,000 data points for plotting. It can be observed that our method has a higher threshold and is not easily to be triggered, while HDDDM detects drift more frequently.

TABLE II. The number of drifts

	Number of drifts			
	VAE(3σ)	VAE(2σ)	VAE(σ)	HDDDM
SMD	2	19	142	46
MSL	6	13	368	82
SWaT	8	25	449	65

Our drift detection algorithm can output the distance of features. We normalize these values to a range of 0-1, enabling the analysis of the impact of each feature on the drift. We visualize it using the heatmap as shown in Figure 6. The darker the color in the image, the greater the impact caused by the feature. The y-axis represents drift instances, and the x-axis represents the different features. Through the heatmap, we can easily identify the features that have a significant impact on drift.

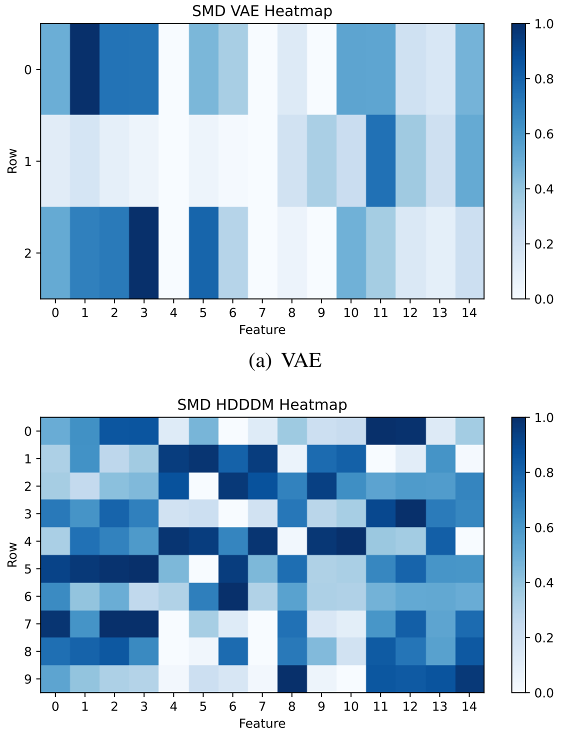
Our method performs well in detecting drift in MTS. The detected drift can be identified by analyzing the heatmap for relevant features and then locating the corresponding regions. Additionally, our method has a default threshold of 2σ (The performance is excellent in RQ1.), which can be manually adjusted to adapt to different sensitive environments.



(a) VAE

(b) HDDDM

Figure 5: SMD drift detection



(a) VAE

(b) HDDDM

Figure 6: SMD heatmap

4.3 RQ2. The Effectiveness of Monitoring AD Models

For real-time monitoring of anomaly detection models, we select data segments with high anomaly values for experimentation, as shown in Table III.

TABLE III. The selected dataset statistics

Dataset	Segment	Segment Anomalies(%)
SMD	15001-20000(5000)	43.8
MSL	5001-10000(5000)	23.3
SWaT	10001-15000(5000)	15.9

We select four deep learning models of anomaly detection, USAD [22], LSTM_AD [32], GDN [36], and MAD_GAN [37], to analyze the effect of drift detection algorithms on deep learning anomaly detection. The model is used to evaluate the performance of the drift detection algorithm. We utilize selected sets of 5000 data points for real-time analysis using a sliding window. During initialization, we train basic anomaly detection models and drift detection models VAE with the window data. When the drift is detected, we update both models using the current window data. The final evaluation metrics are computed as shown in Table IV. Original symbols are without the drift detection. Symbols with " ' " are with the drift detection added.

From the table information, we observe that the effectiveness of the drift detection on the service quality of the anomaly detection models varies across different datasets. In general, the improvement effect is shown. However, certain models perform worse on specific datasets. Firstly, because we update the anomaly detection models with window data, which may not adequately represent the data distribution over a long period. As a result, insufficient fitting may occur. Secondly, models of anomaly detection are unsupervised models and have higher requirements for data quality. Ideally, all normal values should be fully covered. This can't be achieved in the window data.

TABLE IV. Results of anomaly detection models

Method	SMD							
	P	P'	R	R'	AUC	AUC'	F1	F1'
USAD	0.6808	0.4050	0.9973	0.8915	0.8929	0.7745	**0.8092**	0.5570
LSTM_AD	0.7731	0.8644	0.9820	0.9844	0.9102	0.9406	0.8651	**0.9205**
GDN	0.2648	0.4233	0.9983	0.9298	0.8083	0.7988	0.4186	**0.5817**
MAD_GAN	0.6169	0.2954	0.6127	0.5312	0.6447	0.5502	**0.6148**	0.3797
	MSL							
	P	P'	R	R'	AUC	AUC'	F1	F1'
USAD	0.9039	0.9047	0.6610	0.7350	0.8130	0.8510	0.7636	**0.8111**
LSTM_AD	0.9056	0.9056	0.7404	0.8460	0.8539	0.9075	0.8147	**0.8748**
GDN	0.9923	0.9888	0.5862	0.7042	0.7915	0.8500	0.7370	**0.8226**
MAD_GAN	0.9099	0.9082	0.5400	0.4008	0.7515	0.6756	**0.6777**	0.5561
	SWaT							
	P	P'	R	R'	AUC	AUC'	F1	F1'
USAD	0.1613	0.1613	0.6332	0.7967	0.7341	0.8167	0.2571	**0.2683**
LSTM_AD	0.1613	0.3226	0.6360	0.1618	0.7355	0.4797	**0.2573**	0.2155
GDN	0.1613	0.3226	0.6170	0.3062	0.7259	0.5741	0.2557	**0.3142**
MAD_GAN	0.1613	0.3226	0.6277	0.2871	0.7313	0.5632	0.2566	**0.3038**

4.4 RQ3. The Interpretability of the Model Performance

We selected the LSTM_AD on the SMD dataset in RQ2 for analysis. Firstly, based on the results, we found that there are four drift alerts triggered among the 5000 time series, as shown by the green regions in Figure 8. We first plot a heatmap (Figure 7) of all drift points, which provides an intuitive view. It is evident that Feature 22 had the most significant continuous impact, making it a key feature for analysis. We synthesize the statistical values of Feature 22 obtained from real-time calculations, including the mean (blue), standard deviation (orange), the proportion of anomaly detection model predictions (Red R_i), and the occurrence time points of data drift (green span). All these aspects are combined in Figure 8. Overall, the trend of the curve in the figure aligns with our expectations. We can capture the dynamic changes from one distribution to another. These validate the practicality of our online data drift detection method.

Secondly, based on Figure 8, we can interpret the quality of the model's service. We can observe that all four detected drifts were due to sudden covariance drift. They all become anomalies due to changes in input features. Then it is detected by the anomaly detection model, resulting in an increase in R_i. This can also be inferred from the mean and standard deviation of Feature 22, where the model considers them as outliers during input fluctuations. However, there are two drift segments that are not detected by our algorithm, likely due to the strict default threshold we set. It can be adjusted manually according to the specific usage environment.

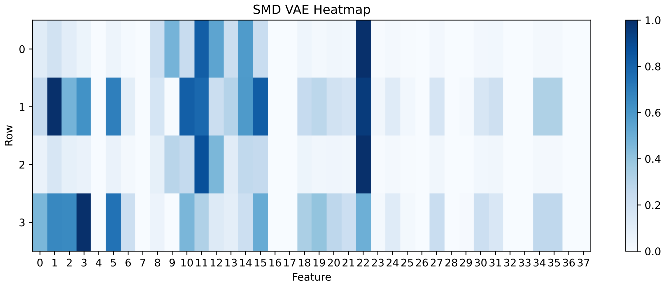


Figure 7: LSTM_AD heatmap

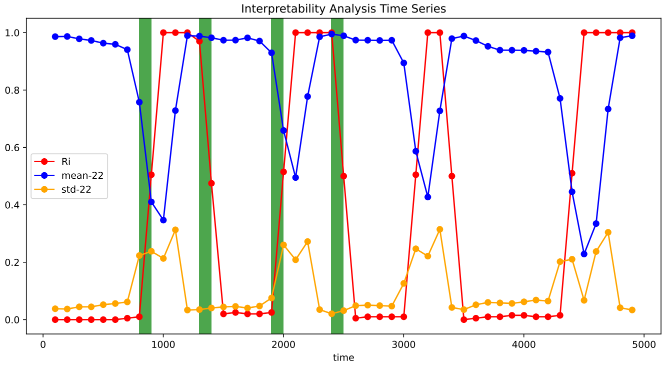


Figure 8: LSTM_AD interpretability analysis

5. Related Work

Anomaly Detection in Multivariate Time Series. Detecting anomalies in time-series data has been extensively studied [38]. Traditional statistical models, including ARIMA [39], and SVM [40], have been widely used for anomaly detection in both univariate and multivariate time series. Other techniques such as wavelet analysis [41], pattern-based approaches [42], and distance-based methods [43], have also been employed. At the same time, deep learning frameworks have gained popularity due to their ability to handle high-dimensional temporal data without assuming stationarity.

Deep learning models for multivariate time-series anomaly detection often combine RNN with other architectures such as CNN, VAE, or GAN. RNN is used to capture temporal dependencies, while CNN, VAE, and GAN capture relationships among the variables [44]. For a more comprehensive exploration of deep learning techniques for time series anomaly detection, readers are encouraged to refer to the latest survey [45].

Drift Detection. There is extensive research on drift detection [7], [46], [47], [48], [49]. Common practices include using custom tests to check if feature values fall within specified ranges [50]. Various statistical hypothesis testing and confidence interval-based approaches have been proposed to evaluate whether two sets of samples are drawn from the same distribution. (e.g., Kolmogorov-Smirnov test [17], Maximum Mean Discrepancy [51]). However, these tests often require careful tuning (e.g., appropriate kernels and hyperparameters). As a result, confidence interval-based approaches are frequently employed for practical drift detection [4]. In addition to model-agnostic techniques, specialized methods that leverage the internal workings of the model and training data have been proposed [52], [53], [54].

6. Conclusion

In this paper, we propose an online data drift detection method for deep learning models in the field of multivariate time series anomaly detection to ensure model reliability, availability, and interpretability. We leverages real-time statistical values, reconstructed data differences, ratio of output categories to evaluate model service quality. We validate the effectiveness and interpretability of our drift detection approach with real-world datasets and different structured anomaly detection models. Future research directions may include exploring additional drift detection algorithms, incorporating feedback mechanisms for continuous model improvement, and extending the approach to different domains.

Acknowledgment

The research is supported by the Key-Area Research and Development Program of Guangdong Province (No. 2020B010165002), the Guangdong Basic and Applied Basic Research Foundation (No. 2023B1515020054), the Natural Science Foundation of China (No. 62272495), the Tencent Rhino-Bird Joint Research Program. The corresponding author is Pengfei Chen.

References

[1] C. Ding, S. Sun, and J. Zhao, "Mst-gat: A multimodal spatial–temporal graph attention network for time series anomaly detection," *Information Fusion*, vol. 89, pp. 527–536, 2023.

[2] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *arXiv preprint arXiv:2211.05244*, 2022.

[3] G. Li and J. J. Jung, "Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges," *Information Fusion*, 2022.

[4] D. Nigenda, Z. Karnin, M. B. Zafar, R. Ramesha, A. Tan, M. Donini, and K. Kenthapadi, "Amazon sagemaker model monitor: A system for real-time insights into deployed machine learning models," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 3671–3681.

[5] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas, "Mlops-definitions, tools and challenges," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022, pp. 0453–0460.

[6] A. Flórez, I. Rodriguez-Moreno, A. Artetxe, I. G. Olaizola, and B. Sierra, "Catsight, a direct path to proper multi-variate time series change detection: perceiving a concept drift through common spatial pattern," *International Journal of Machine Learning and Cybernetics*, pp. 1–20, 2023.

[7] E. Breck, N. Polyzotis, S. Roy, S. Whang, and M. Zinkevich, "Data validation for machine learning." in *MLSys*, 2019.

[8] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[9] M. Barry, A. Bifet, and J.-L. Billy, "Streamai: Dealing with challenges of continual learning systems for serving ai in production," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2023, pp. 134–137.

[10] D. A. Tamburri, "Sustainable mlops: Trends and challenges," in *2020 22nd international symposium on symbolic and numeric algorithms for scientific computing (SYNASC)*. IEEE, 2020, pp. 17–23.

[11] D. Xin, H. Miao, A. Parameswaran, and N. Polyzotis, "Production machine learning pipelines: Empirical analysis and optimization opportunities," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2639–2652.

[12] S. Alla, S. K. Adari, S. Alla, and S. K. Adari, "What is mlops?" *Beginning MLOps with MLFlow: Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure*, pp. 79–124, 2021.

[13] S. Horchidan, E. Kritharakis, V. Kalavri, and P. Carbone, "Evaluating model serving strategies over streaming data," in *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*, 2022, pp. 1–5.

[14] L. Piano, F. Garcea, V. Gatteschi, F. Lamberti, and L. Morra, "Detecting drift in deep learning: A methodol-

ogy primer," *IT Professional*, vol. 24, no. 5, pp. 53–60, 2022.

[15] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[16] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. M. Dos Santos, "An overview of unsupervised drift detection methods," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 6, p. e1381, 2020.

[17] D. M. Dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1545–1554.

[18] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 935–944.

[19] G. Ditzler and R. Polikar, "Hellinger distance based drift detection for nonstationary environments," in *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*. IEEE, 2011, pp. 41–48.

[20] A. Abbasi, A. R. Javed, C. Chakraborty, J. Nebhen, W. Zehra, and Z. Jalil, "Elstream: An ensemble learning approach for concept drift detection in dynamic social big data stream learning," *IEEE Access*, vol. 9, pp. 66 408–66 419, 2021.

[21] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[22] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3395–3404.

[23] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[24] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1409–1416.

[25] Z. Xu, R. Wang, G. Balaji, M. Bundele, X. Liu, L. Liu, and T. Wang, "Alertiger: Deep learning for ai model health monitoring at linkedin," *arXiv preprint arXiv:2306.01977*, 2023.

[26] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with lstm neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 8, pp. 3127–3141, 2019.

[27] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[28] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, and D. Pei, "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 3220–3230.

[29] N. Gugulothu, P. Malhotra, L. Vig, and G. Shroff, "Sparse neural networks for anomaly detection in high-dimensional time series," in *AI4IOT workshop in conjunction with ICML, IJCAI and ECAI*, 2018, pp. 1551–3203.

[30] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *Ieee Access*, vol. 7, pp. 1991–2005, 2018.

[31] P. Filonov, A. Lavrentyev, and A. Vorontsov, "Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model," *arXiv preprint arXiv:1612.06676*, 2016.

[32] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 387–395.

[33] Ö. Gözüaçık and F. Can, "Concept learning using one-class classifiers for implicit drift detection in evolving data streams," *Artificial Intelligence Review*, vol. 54, pp. 3725–3747, 2021.

[34] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2828–2837.

[35] A. P. Mathur and N. O. Tippenhauer, "Swat: A water treatment testbed for research and training on ics security," in *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE, 2016, pp. 31–36.

[36] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4027–4035.

[37] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International conference on artificial neural networks*. Springer, 2019, pp. 703–716.

[38] P. Boniol, M. Linardi, F. Roncallo, T. Palpanas, M. Mef-

tah, and E. Remy, “Unsupervised and scalable subsequence anomaly detection in large data series,” *The VLDB Journal*, pp. 1–23, 2021.

[39] A. Arning, R. Agrawal, and P. Raghavan, “A linear method for deviation detection in large databases.” in *KDD*, vol. 1141, no. 50, 1996, pp. 972–981.

[40] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, “Sand: streaming subsequence anomaly detection,” *Proceedings of the VLDB Endowment*, vol. 14, no. 10, pp. 1717–1729, 2021.

[41] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[42] M. Braei and S. Wagner, “Anomaly detection in univariate time-series: A survey on the state-of-the-art,” *arXiv preprint arXiv:2004.00433*, 2020.

[43] M. A. Bashar and R. Nayak, “Tanogan: Time series anomaly detection with generative adversarial networks,” in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 1778–1785.

[44] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection for discrete sequences: A survey,” *IEEE transactions on knowledge and data engineering*, vol. 24, no. 5, pp. 823–839, 2010.

[45] K. Choi, J. Yi, C. Park, and S. Yoon, “Deep learning for anomaly detection in time-series data: review, analysis, and guidelines,” *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021.

[46] G. Cormode, Z. Karnin, E. Liberty, J. Thaler, and P. Veselỳ, “Relative error streaming quantiles,” in *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2021, pp. 96–108.

[47] Z. Karnin, K. Lang, and E. Liberty, “Optimal quantile approximation in streams,” in *2016 ieee 57th annual symposium on foundations of computer science (focs)*. IEEE, 2016, pp. 71–78.

[48] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, “Characterizing concept drift,” *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[49] I. Žliobaitė, M. Pechenizkiy, and J. Gama, “An overview of concept drift applications,” *Big data analysis: new algorithms for a new society*, pp. 91–114, 2016.

[50] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, and A. Grafberger, “Automating large-scale data quality verification,” *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 1781–1794, 2018.

[51] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[52] S. Garg, Y. Wu, S. Balakrishnan, and Z. Lipton, “A unified view of label shift estimation,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 3290–3300, 2020.

[53] Z. Lipton, Y.-X. Wang, and A. Smola, “Detecting and correcting for label shift with black box predictors,” in *International conference on machine learning*. PMLR, 2018, pp. 3122–3130.

[54] Y. Wu, E. Winston, D. Kaushik, and Z. Lipton, “Domain adaptation with asymmetrically-relaxed distribution alignment,” in *International conference on machine learning*. PMLR, 2019, pp. 6872–6881.