Real-Time Anomaly Detection for Large-Scale Network Devices

Lei Tao, Minghua Ma, *Member, IEEE*, Shenglin Zhang, *Member, IEEE*, Junhua Kuang, Xiao-Wei Guo, Canqun Yang, and Dan Pei, *Senior Member, IEEE*

***Abstract*— With the booming of large-scale network devices, anomaly detection on multivariate time series (MTS), such as a combination of CPU utilization, average response time, and network packet loss, is important for system reliability. Although a collection of learning-based approaches have been designed for this purpose, our study shows that these approaches suffer from long *initialization time* for sufficient training data. Our previously proposed JumpStarter model stands as a MTS anomaly detection method characterized by its brief initialization time and commendable detection performance. However, it suffers from high computational cost and inappropriateness for periodic MTS. In this paper, we propose *VersaGuardian*, which introduces the *Dynamic Mode Decomposition* technique to MTS anomaly detection for diverse types of MTS in a rapidly initialized, computationally efficient manner. With real-world MTS datasets collected from three companies, our results show that *VersaGuardian* achieves an average F1 score of 94.42%, significantly outperforming the popular anomaly detection algorithms, with a much shorter initialization time of 20 minutes and detection time of 15.28 milliseconds.**

***Index Terms*— Anomaly detection, network devices, multivariate time series, dynamic mode decomposition.**

I. Introduction

IN THE context of large-scale networks, hardware replacement, code modifications, software configurations, and the introduction of new technologies are commonplace among various teams. These activities aim to deploy new functionalities, address existing errors, and optimize performance. It is crucial to recognize that despite meticulous pre-deployment testing, the complexity arising from operational scale, diverse device types, vendor heterogeneity, and intricate interactions between device and software components may lead to latent defects and functional inadequacies surfacing in a real production environment [1], [2]. Even seemingly minor upgrades can potentially result in significant failures with repercussions for end-user experience. Early detection of anomalies resulting from hardware or software changes is paramount [3], enabling release and operators to intervene in the event of erroneous alterations and facilitate timely rollback procedures to prevent broader adverse consequences [4].

To facilitate the detection of anomalies within network devices, operators engage in the ongoing collection of monitoring data of each performance metric (*e.g.*, CPU utilization, network packet loss and protocol flaps, *etc.*) at equally spaced intervals [5]. The monitoring data of a metric form a univariate time series, and thus that of network devices, which has multiple metrics, constitutes a multivariate time series (MTS). Traditional MTS anomaly detection approaches are typically based on detecting univariate time series [6]. However, the status of a specific metric does not adequately reflect the overall status that operators are more concerned about [7], [8]. The limitations of univariate time series anomaly detection in capturing complex temporal relationships among different univariate time series often result in alert storms [9]. To address this issue, recent works [7], [10], [11], [12], [13], [14], [15] have utilized deep learning techniques to build learning models for detecting anomalies in MTS.

Learning-based approaches are often impractical due to their reliance on a substantial amount of training data, which is not readily available in many real-world scenarios. As a result of hardware or software changes, the data distribution of MTS can undergo significant shifts, which are known as expected concept drift [16]. For instance, when operators deploy a service to more network devices through a software change, the metric "Requests Per Second" in each network device may experience a substantial drop, as illustrated in Figure 1. Operators anticipate this change and do not consider rolling back the software change. However, after the change, it can lead to a high number of false alarms or missed alerts since the learning-based anomaly detection models trained on data before the change become invalid. This is because the fundamental assumption in deep learning that the data distribution remains consistent between the training and test sets is violated [17].

Received 21 January 2024; revised 10 November 2024; accepted 11 January 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor Y. Liu. This work was supported in part by the Advanced Research Project of China under Grant 31511010501; and in part by the National Natural Science Foundation of China under Grant 62272249, Grant 62302244, and Grant 62072264. *(Corresponding author: Shenglin Zhang.)*

Lei Tao and Junhua Kuang are with the College of Software, Nankai University, Tianjin 300192, China (e-mail: leitao@mail.nankai.edu.cn; 2013157@mail.nankai.edu.cn).

Minghua Ma is with Microsoft, Redmond, WA 98052 USA (e-mail: minghuama@microsoft.com).

Shenglin Zhang is with the College of Software, Nankai University, Tianjin 300192, China, and also with the Haihe Laboratory of Information Technology Application Innovation (HL-IT), Tianjin 300459, China (e-mail: zhangsl@nankai.edu.cn).

Xiao-Wei Guo is with the College of Computer Science, National University of Defense Technology, Changsha 410073, China (e-mail: guoxiaowei@nudt.edu.cn).

Canqun Yang is with the College of Computer Science, National University of Defense Technology, Changsha 410073, China, and also with the National Supercomputer Center in Tianjin, Tianjin 300456, China (e-mail: canqun@nudt.edu.cn).

Dan Pei is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100190, China, and also with Beijing National Research Center for Information Science and Technology, Beijing 100190, China (e-mail: peidan@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TON.2025.3529861

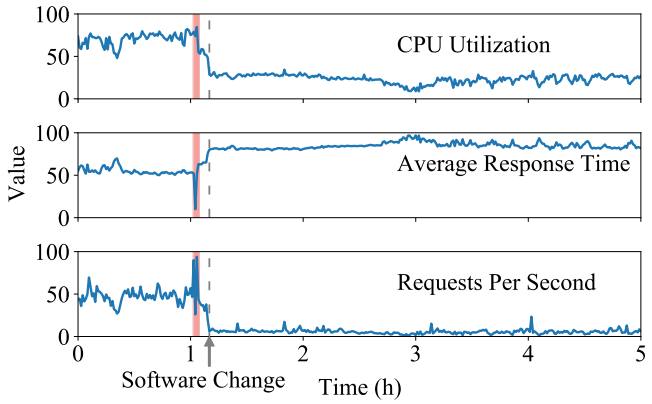


Fig. 1. The MTS (selected as examples) of a network device before and after a software change. The red segment is labeled anomalous.

To address this issue, the learning-based approaches need to be retrained to adapt to the new data distribution. However, the retraining process can be time-consuming, ranging from tens to hundreds of days [6], [7], before the models reach a steady state. This prolonged retraining duration adds to the challenges and limitations of applying learning-based methods in practice. To quantitatively measure how long it takes to "initialize" a MTS anomaly detection model, we first define initialization time, as the time lag between when the model is launched and when it becomes well trained, and conduct an empirical study on the initialization time of multiple learning-based methods.

For time series characterized by pronounced periodic patterns, the spike of local anomalies during idle periods may be much lower compared to the high values observed during busy periods [18]. Consequently, there exists a necessity for the adaptability of MTS anomaly detection algorithms to effectively accommodate periodic time series. However, usually the non-learning-based methods are not good at detecting anomalies for periodic MTS. Furthermore, the temporal duration essential for discerning potential issues directly influences user satisfaction such as our proposed method, JumpStarter [19]. Alarm strategies marked by prolonged latency often precipitate augmented financial losses.

Therefore, the endeavor to monitor a plethora of metrics within network devices undergoing frequent hardware or software changes, with the objective of promptly identifying latent issues and making informed decisions to enhance service quality, entails the following three challenges:

1) Frequent changes lead to changes in the patterns of metrics: existing methodologies fall short in concurrently achieving rapid initialization, elevated accuracy in detection, and low computational cost.
2) Unearthing anomalies hidden within periodic time series is challenging: a growing array of time series manifests periodicity, harboring an escalating count of anomalies concealed amidst localized periods.
3) In production environments, high-latency alarm strategies often increase enterprise losses: expeditious anomaly identification and immediate alerting can effectively curtail losses and mitigate the impact on the services.

In tackling these challenges, we present *VersaGuardian*, an extension of JumpStarter [19]. *VersaGuardian* is a versatile method, proficient in rapid initialization, high computational efficiency, and adept at detecting anomalies within periodic time series. Leveraging techniques such as seasonal-trend decomposition using loess (STL) [20] and dynamic mode decomposition with control (DMDc) [21], *VersaGuardian* offers a comprehensive solution.

The contributions of this paper are summarized as follows:

- To the best of our knowledge, *VersaGuardian* stands as a pioneering approach, employing dynamic mode decomposition to extract essential low-dimensional primary modes and temporal dynamics from intricate high-dimensional MTS. This strategic extraction facilitates the prediction of time series evolution, ultimately leading to rapid online anomaly detection and the attainment of low-latency alerting capabilities.
- *VersaGuardian* introduces a novel integration by embedding both the seasonal and remainder components, derived from the MTS' seasonal-trend decomposition, into the DMD framework. This ingenious fusion empowers *VersaGuardian* with the competence to identify anomalies within periodic time series, thereby enhancing the accuracy of anomaly detection.
- *VersaGuardian* represents a data-driven approach to MTS anomaly detection, seamlessly integrating rapid initialization, minimal computational overhead, and the proficiency to identify anomalies within periodic time series.
- We conducted a comprehensive study to evaluate the performance of *VersaGuardian* based on four datasets from three companies, including China Mobile Communications Corporation (CMCC), a top-tier global Internet Service Provider (ISP). The average F1 score of *VersaGuardian* is 94.42%, while those of the other fice approaches are 38.87%, 82.61%, 53.42%, 83.34%, and 86.22% respectively. *VersaGuardian* achieves good accuracy with an initialization time of 20 minutes and a detection time as short as 15.28 milliseconds, open sourced at `https://github.com/stonebegin/VersaGuardian`. Engineered to cater to real-time anomaly detection within large-scale network devices, *VersaGuardian* embodies a robust solution.

II. Related Work and Preliminaries

A. MTS Anomaly Detection

MTS anomaly detection has gained significant attention in various domains due to its importance in detecting and mitigating abnormal patterns and events. Over the years, researchers have proposed several approaches to tackle this challenging problem. We can classify these approaches into three categories.

Traditional method. These methods leverage statistical techniques such as mean, standard deviation, and probability distributions to model the normal behavior of time series data. Deviations from these statistical models are then considered as anomalies. Popular statistical methods include the

time series analysis [22], RRCF [23] and clustering-based LESINN [24],which do not need training data thus the initialization time is short.

Deep learning-based method. Deep learning-based methods in time series anomaly detection aim to identify anomalies based on the inherent structure and patterns present in the data. MSCRED [10], USAD [11], DOMI [25], OmniAnomaly [7] and InterFusion [13] build anomaly detection models by learning the anomaly patterns using a large span of historical data. These methods suffer from long initialization time for training the model.

Others. Some anomaly detection methods employ rule-based or alternative methods [26]. One representative work is our previously proposed method JumpStarter [19], which introduces the compressed sensing technique to reconstruct data. It is known for its fast initialization, but it requires significant computational overhead and lacks the ability to detect anomalies within periodic MTS.

B. Dynamic Mode Decomposition

The DMD is an innovative algorithm within the field of dynamical systems, initially introduced in the fluid dynamics community [27]. It provides the eigenvalues and eigenvectors of the best-fit linear system that connects a snapshot matrix with its time-shifted version at a later time. DMD excels in handling voluminous and streaming datasets [28], effectively extracting modal structures from both numerical simulations and experimental data [29]. Its ability to model and predict high-dimensional data further enhances its versatility [30]. Moreover, DMD holds promise for potential application in the domain of fault diagnosis in rotating machinery [31].

Consider the following data snapshot matrices:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}, \mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the k_{th} snapshot and typically $n < m$. The DMD involves the decomposition of the best-fit linear operator $\mathbf{A}$ relating the matrices above [27]:

$$\mathbf{X}' = \mathbf{A}\mathbf{X} \tag{1}$$

Here we provide a quick introduction to the time series prediction step in DMD, using this anomaly detection problem as the context. The objective of DMD is to "solve" evolution matrix $\mathbf{A}$, and the calculation steps are as follows [32]:

1) Find the truncated SVD of $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

2) Compute $\tilde{\mathbf{A}}$, the projection of the full matrix $\mathbf{A}$ onto $\mathbf{U}$:

$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}$$

3) Compute the eigenvalues and eigenvectors of $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$$

4) Solve for the dynamic modes of $\mathbf{A}$:

$$\mathbf{A}\mathbf{\Phi} = \mathbf{\Phi}\mathbf{\Lambda}, \mathbf{\Phi} = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$$

5) Predict the next time step:

$$\mathbf{X_t}' = \mathbf{A}\mathbf{X_t} \tag{2}$$

III. Background and Motivation

A. Background

Multivariate time series. In large-scale network devices, operators continuously collect monitoring data of multiple metrics or extract numerical values from logs [33]. An network performance metric (*e.g.*, line card crashes), or network health metric (*e.g.*, CPU utilization, memory utilization) [5], is usually collected by equal interval, forming a univariate time series. Any univariate time series alone, however, cannot capture performance issues across all devices [7]. Because a device typically has a collection of monitoring metrics, it can be denoted as a MTS [34], which includes diverse types of univariate time series and thus track various aspects of performance issues. With the scale and complexity of the device increasing, it is becoming more difficult to manually inspect device anomalies. Therefore, MTS anomaly detection is of great importance [7], [10]. We denote a MTS at time t as $\mathbf{X}_t = [\mathbf{x}_t^1, \mathbf{x}_t^2 \ldots, \mathbf{x}_t^n]^{\mathrm{T}}$, where $\mathbf{x}_t^i = [x_{t-w+1}^i, x_{t-w+2}^i, \ldots, x_t^i]$ is the univariate time series of the i^{th} monitoring metric, n is the number of metrics, and w is the observation window size. We apply the sliding window, which is a common practice in time series anomaly detection [6], to construct $\mathbf{x}_t$.

Anomaly detection. Anomaly detection using MTS [7] is important in large-scale network devices. In previous anomaly detection works [7], [10], [14], [35], operators have a rough consensus on the following points: 1) A MTS anomaly is a data point or a data segment that significantly deviates from operators' expectations of normal behavior, and it can be visually observed (*e.g.*, in Figure 1). 2) An anomaly indicates something might have gone wrong, although further investigation may still be needed for verification. 3) Anomaly detection is often used as a failure discovery mechanism. Formally, we define MTS anomaly detection: for time t, given its MTS $\mathbf{X}_t$, we determine whether an anomaly occurs (*e.g.*, jitter, sudden drop or surge), which is denoted by $y_t = 1$ if yes and $y_t = 0$ otherwise.

B. Motivation for Initialization Time

Anomaly detection initialization time. With a new network device being deployed or updated, operators usually launch an anomaly detection approach for it. The initialization time of the anomaly detection approach is the time lag between when it is launched (t_1) and when it becomes effective (t_2), as shown in Figure 2. Many prior approaches, *e.g.*, [7], [10], and [35], use a learning-based workflow to detect anomalies. Typically, they are periodically trained based on historical data [36]. The initialization time of these approaches, *e.g.*, tens of days, is relatively long, because they usually need to offer a lot of historical data for training. In Table I we list the suggested initialization time of five learning-based anomaly detection approaches on different datasets used in their evaluation experiments. For example, OmniAnomaly [7] used two robot system datasets

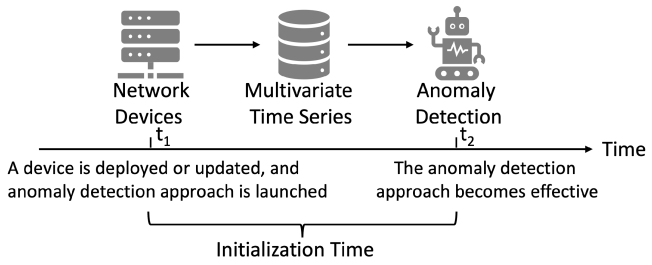


Fig. 2. The initialization time of an anomaly detection approach.

TABLE I

COMPARISON OF THE INITIALIZATION TIME (DAYS) ON DIFFERENT DATASETS (S1∼S3) USED IN THEIR EVALUATION EXPERIMENTS. * DENOTES UNIVARIATE TIME SERIES ANOMALY DETECTION APPROACH, WHICH CAN BE USED FOR MTS BY COMBINING IT WITH MAJORITY VOTE [7]

Approach	S1	S2	S3	Avg.
MSCRED [10]	7	13	-	10
OmniAnomaly [7]	17	15	17	16.3
LSTM-NDT [35]	69	36	-	52.5
* Opprentice [36]	56	56	56	56
* Donut [6]	102	110	99	103.6

(denoted as S1, S2, respectively) and a server dataset (S3, which also used in our experiment as D1). From the last column of Table I, we can see that the average initialization time of these approaches ranges from 10 days to more than one hundred days, indicating that it is unsuitable to use these approaches for newly deployed or updated systems.

Incremental retraining. Considering the long initialization time of learning-based anomaly detection approaches, one may suggest incremental retaining, *i.e.*, gradually (incrementally) adding a short-period (say one day) of data to train these approaches. In this way, we can improve the performance of these approaches step by step. Adding one day's data each time is because these learning-based approaches need at least thousands of data points to converge [6]. We then try to apply incremental retraining to two popular MTS anomaly detection approaches, *i.e.*, OmniAnomaly [7] and MECRED [10]. The dataset is the same as what is used in OmniAnomaly (see section V for more details). We gradually enlarge the training set from one day's data to 13 days' data (*i.e.*, the largest training set of this dataset), and the testing set remains as the data collected after the 13th day.

This sounds ideal, but anomaly detection using incremental retraining cannot ensure satisfactory performance. Figure 3 shows the average F1 score and training time of OmniAnomaly and MECRED as the period of training data increases (day by day), respectively. From Figure 3(a), we can see that the average F1 scores of both OmniAnomaly and MECRED increase along with more training data being used, and they do not converge until 10 days' data is used for training. One primary reason is that these learning-based approaches have to *explicitly* learn the probability distribution of a MTS from a large amount of training data to capture its normal behavior. Figure 3(b) shows that the training time of both OmniAnomaly and MECRED increases linearly with the size

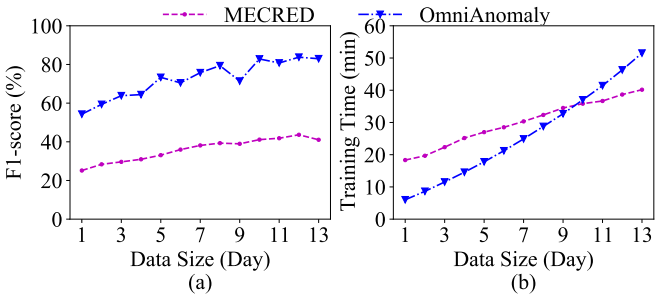


Fig. 3. Performance of OmniAnomaly [7] and MECRED [10] by incremental retraining.

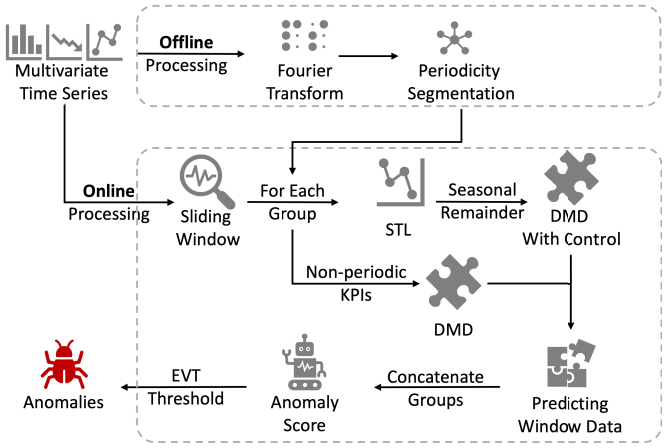


Fig. 4. *VersaGuardian* approach consists of offline processing and online processing, of which output is whether anomaly or not.

of training data. When the training dataset contains 10 days of data, it takes about 35 minutes to train OmniAnomaly or MECRED. Therefore, these approaches are not suitable for newly deployed or updated systems due to their non-robustness and considerable training cost.

IV. *VersaGuardian* APPROACH

A. Overview of VersaGuardian

The *VersaGuardian* methodology, depicted in Figure 4, encompasses both offline and online processing procedures. In the offline processing phase, a periodicity segmentation method is adopted to enhance the accuracy of time-series decomposition and optimize anomaly detection efficiency. Each univariate time series within the MTS undergoes Fourier transformation, followed by grouping into multiple clusters based on their respective periods. In the online anomaly detection phase, for non-periodic time series, DMD is directly employed for forecasting the subsequent time step. Conversely, for time series manifesting periodic patterns, a seasonal-trend decomposition using loess (STL) is performed based on their respective periods, yielding trend components, seasonal components, and remainder components. These seasonal and remainder components are subsequently integrated as control factors into the DMD for time series forecasting. Subsequently, the forecasted time series are concatenated, the disparity between the original and forecasted MTS is quantified as an

anomaly score, and anomaly detection is executed through *EVT threshold* applied to the anomaly score.

B. Periodicity Segmentation

Before detecting anomalies concealed within periodic time series, it is imperative to distinguish whether the time series exhibits periodicity. Periodicity segmentation enables the formation of MTS with similar patterns by grouping univariate time series with comparable period sizes. This not only preserves the overall state of network devices but also enhances the efficiency of subsequent anomaly detection.

The Fourier transform (FT) [37] is a technique employed for the conversion of time-domain data into frequency-domain data. In periodicity detection, FT identifies the primary frequency peaks within a signal's spectrum, which correspond to the dominant cycles or periodic elements of the data. By examining these frequency peaks, the signal's main period can be calculated as the inverse of the peak frequency. This method is widely used in applications such as failure detection, vibration analysis, and speech recognition for analyzing and extracting periodic patterns [38].

The application of the FT serves to ascertain the presence of periodicity in time series and facilitates the computation of period sizes. Univariate time series within the multivariate dataset devoid of periodic patterns are grouped into one cluster, while those manifesting periodicity are grouped into distinct clusters contingent upon the sizes of their respective periods.

C. Seasonal-Trend Decomposition

In order to enhance our understanding of the patterns and characteristics underlying the temporal evolution of time series, facilitating more accurate predictions and analyses, it is essential to decompose MTS. The components obtained through decomposition provide clearer guidance for anomaly detection [18].

Performing STL on each time series based on their respective periods, we decompose them into trend components $\tau_{\mathbf{t}}$, seasonal components $\mathbf{s_t}$, and remainder components $\mathbf{r_t}$. The decomposition of the univariate time series $\mathbf{x}_t^i$ is shown as follows:

$$\mathbf{x_t^i} = \tau_{\mathbf{t}}^{\mathbf{i}} + \mathbf{s_t^i} + \mathbf{r_t^i} \tag{3}$$

As shown in Figure 5, the original time series acquired from large-scale network devices is illustrated as the composite of three components. Typically, the trend component of a MTS is characterized by its inherent smoothness, with anomalies often discernible in the remainder. This property proves advantageous in facilitating subsequent anomaly detection processes. Against the backdrop of network devices, where these time series originate, this analytical approach becomes particularly relevant in discerning anomalies within the system.

As shown in Figure 6, we utilize the grouping obtained from the periodicity segmentation along with their corresponding period sizes as inputs to STL. The decomposition process generates components, including trend, seasonal, and remainder. Subsequently, the remainders that potentially contain extreme values are subjected to a smoothing operation,

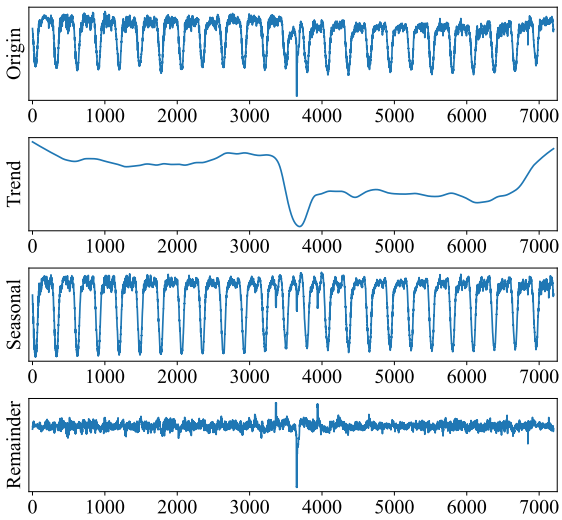


Fig. 5. An example of using STL to decompose the original time series into trend component, seasonal component, and remainder component.

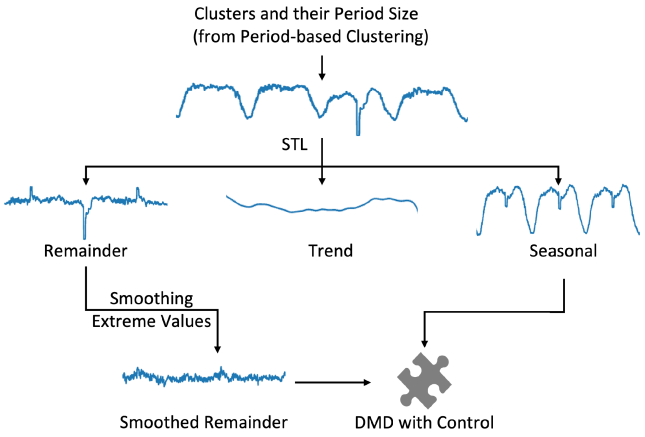


Fig. 6. The flowchart of STL used in *VersaGuardian*.

and the smoothed remainders, along with the seasonals reflecting periodic variations, are saved as intermediate outputs. The seasonal components and smoothed remainders obtained from STL enable subsequent DMD to generate the predicted MTS with enhanced accuracy.

D. Dynamic Mode Decomposition With Control

DMD adeptly isolates temporal patterns, enabling the identification of dominant dynamics in MTS. By inherently reducing data dimensionality through the identification of significant modes, DMD streamlines analysis and bolsters computational efficiency. This technique captures the temporal evolution of MTS, offering insights into how different indicators change over time. As a data-driven method, DMD unveils inherent patterns and structures without reliance on predefined models, rendering it suitable for diverse and complex MTS. By computing the eigenvectors and eigenvalues, the frequencies and amplitudes of the modes can be determined and utilized for data reconstruction and future behavior prediction.

Algorithm 1 DMDc Used in *VersaGuardian*

Input: $\mathbf{D_t}(k*w)$: original time series obtained through sliding window, $\mathbf{S_t}(k*w)$: seasonal component obtained through STL of $\mathbf{D_t}$, $\mathbf{R_t}(k*w)$: smoothed remainders obtained through STL of $\mathbf{D_t}$, $\mathbf{r}$: the number of main modes that DMD needs to retain.

Output: $\mathbf{A}$: evolution matrix for predicting the next time step MTS.

$\mathbf{X_1} \leftarrow \mathbf{S_t}[:, :-1]$
$\mathbf{B}(k*k) \leftarrow$ Identity matrix
$\mathbf{\Upsilon} \leftarrow \mathbf{R_t}[:, :-1]$
$\mathbf{X_2} \leftarrow \mathbf{D_t}[:, 1:] - \mathbf{B\Upsilon}$
U, Σ, $V^* \leftarrow \mathbf{SVD}(\mathbf{X_1})$
$\tilde{A} = U[:, :\mathbf{r}]^* \mathbf{X_2} V[:\mathbf{r}, :]\Sigma[:\mathbf{r}]^{-1}$
Λ, W $\leftarrow$ Perform eigenvalue decomposition on $\tilde{A}$
$\Psi = \mathbf{X_2} V[:\mathbf{r}, :]\ Diag(\Sigma[:\mathbf{r}])\mathrm{W}$
$\mathbf{A} = \Psi Diag(\Lambda)\Psi^{\dagger}$
return A

Therefore, DMD aids in enhancing our understanding and forecasting of MTS behavior.

Due to the inherent nature of DMD in predicting MTS based on primary modes, it tends to overlook anomalies concealed within periodic time series. In such instances, a more precise control over the prediction methodology of DMD is required, referred to as DMDc [21]. DMDc is an extension of the standard DMD method that incorporates control inputs or external forcing into the analysis. It aims to capture the influence of these control inputs on the dynamics of a MTS. Eq. (1) has been rewritten as follows:

$$\mathbf{X}' = \mathbf{AX} + \mathbf{B\Upsilon} \tag{4}$$

Here, we set $\mathbf{X}$ as the seasonal components, $\mathbf{B}$ as the identity matrix, and $\mathbf{\Upsilon}$ as the remainder components obtained from STL, which has undergone extreme values smoothing. Setting B as the identity matrix in DMDc simplifies the modeling process by allowing each control input in $\mathbf{\Upsilon}$ to directly influence the corresponding state variable in $\mathbf{X}$ without any transformation or scaling. This choice is particularly beneficial in scenarios where $\mathbf{\Upsilon}$ represents the direct influence of anomalies or external factors on the primary dynamics of the time series (captured by $\mathbf{X}$). Smoothing extreme values is a common procedure in handling MTS [19], [39]. Specifically, it involves removing the top 5% of data points with the largest deviations from the mean and employing linear interpolation to fill the gaps. We describe the specific calculation steps in Algorithm 1. This approach enables accurate prediction of the evolution of normal mode time series while suppressing anomalies in periodic time series. It amplifies the discrepancies between real MTS and predicted MTS, thereby enhancing the accuracy of anomaly detection.

Anomaly score. We first reconstruct the time series for each cluster of univariate time series to approximate the original MTS. We then concatenate these reconstructed univariate time series to form a new multivariate time series $\mathbf{X'}_t$. Note that the original and predicted MTS have the same order of univariate time series. Intuitively, an anomaly score is needed to measure the similarity between the original and the predicted MTS. We measure the differences of the n time series between $\mathbf{X}_t$ and $\mathbf{X'}_t$ using euclidean distance [36]: $\mathbf{d}_t^i = |\mathbf{x}_t^i - \mathbf{x'}_t^i|$, where $\mathbf{x'}_t^i$ is the predicted univariate time series of $\mathbf{x}_t^i$. To avoid an anomaly score being dominated by a single significant spike in a univariate time series, we calculate s_t using the harmonic mean of $\mathbf{d}_i$, *i.e.*, $s_t = n/(\sum_{i=1}^{n} \mathbf{d}_t^{i\,-1})$.

Choosing threshold. To properly generate anomaly alerts, we need to accurately choose a threshold to determine whether an anomaly score is high enough to trigger an alert. A static threshold does not work well since the data distribution changes over time. Because an extreme value of the anomaly score generated by *VersaGuardian* usually represents an anomaly, we adopt the widely used Extreme Value Theory (EVT) [40] to tailor the anomaly threshold automatically. EVT is a statistical theory aiming to find the law of extreme values, and it does not assume data distribution. It has been demonstrated to accurately choose the threshold for anomaly detection methods [7], [34]. Note that using EVT for choosing threshold is not the main contribution of our work.

V. Experiment

In the study, we address the following research questions:

RQ1: How well does *VersaGuardian* perform in MTS anomaly detection?

RQ2: How *VersaGuardian* copes with the challenge of computational efficiency?

RQ3: How does *VersaGuardian* detect anomalies hidden in periodicity?

A. Experimental Setup

1) Dataset: We conduct experiments on four real-world datasets, including one public dataset[1] – D1 from the production environment of a top-tier global content provider $\mathcal{A}$ offering services for over 500 million daily active users, two datasets (D2, D3) collected from the production environment of a top-tier global content platform $\mathcal{B}$ providing services for over 800 million daily active (over 1 billion cumulative) users, and one dataset – D4 from the production environment of a top-tier global Internet Service Provider (ISP) $\mathcal{C}$.

Specifically, D1 is a five-week-long dataset collected from 28 servers, and it is sampled once per minute. D2 and D3 are two datasets collected from 30 servers over two different seven-week-long periods, respectively. They are both sampled once every five minutes. D4 is collected from 200 wireless base stations over fourteen-day-long periods and sampled once every fifteen minutes.

This work studies metrics of network devices. These metrics are equally important and have no hierarchy among them. The ground truth of anomalies in all the four datasets are manually labeled by operators based on performance issues and failure tickets. The point-wise anomaly rates ($\frac{\#\ anomaly\ data\ points}{\#\ total\ data\ points}$) are diverse in these datasets. For example, the anomaly rate of D3 (20.26%) is much higher than those of D1 (4.16%), D2 (5.25%) and D4 (5.18%), mainly because D3 contains

[1] https://github.com/NetManAIOps/OmniAnomaly

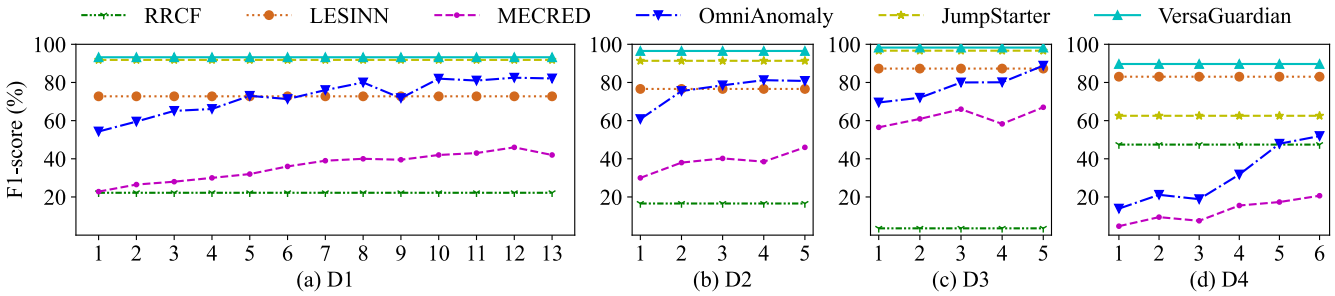


Fig. 7. The average F1 score for the four datasets D1, D2, D3, and D4 as a function of the training dataset size in segments.

TABLE II
THE DETAILED INFORMATION OF THE DATASETS

Dataset	# Servers # Base Stations	n	# Training Days	# Test Days	Anomaly ratio
D1	28	38	13	13	4.16
D2	30	19	20	25	5.25
D3	30	19	20	25	20.26
D4	200	25	14	14	5.18

a severe outage that lasted a long time. Table II lists the detailed information of each dataset, including the number of metrics (n) in each MTS, the scale of the training and test sets, and the anomalies ratio. For each network device, the monitoring metrics constitute its MTS. The number of metrics monitored in D1, D2,D3 and D4 are 38, 19, 19 and 25 respectively. Monitoring tens of metrics is a typical setting for large-scale network devices.

2) Compared Approaches: We compare *VersaGuardian* with two learning-based unsupervised approaches for anomaly detection in MTS: MSCRED [10] and OmniAnomaly [7]. Additionally, we evaluate *VersaGuardian* against three other anomaly detection algorithms: robust random cut forest (RRCF [23]), least similar nearest neighbors (LESINN [24]) and JumpStarter [19]. We exclude the comparison with baseline methods designed for univariate time series, as it has been demonstrated that these approaches are not suitable for detecting anomalies in MTS [7]. We implement these baselines using the default hyperparameters provided in the corresponding open-source code.

3) Implementation: We implement *VersaGuardian* and baselines methods with Python 3.8, and run them on a Dell R420 server with 16 * Intel Xeon E5-2420 CPUs and a 64GB memory

4) Evaluation Metrics: The output of a MTS anomaly detection approach for a specific timestamp is either anomalous or not. We use True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) to label an anomaly detection result according to the ground truth. A TP is an anomaly both confirmed by operators and detected by the approach. If an anomaly is labeled by operators but not detected by the approach, we label the item as an FN. An FP is an "anomaly" that is detected by the approach but is actually normal. We use three metrics for evaluating the performance of *VersaGuardian* and related approaches: Precision = TP / (TP + FP), Recall = TP / (TP + FN), F1 score = 2 * Precision * Recall / (Precision + Recall). The accounting of the three metrics is point-adjusted. That is, if any point in an anomalous segment in the ground truth is detected, we consider the entire segment, or all anomalous points therein, as detected correctly. Point-adjusted metrics are widely adopted in anomaly detection [6], [7], since operators care more about anomalies in a contiguous segment than point-wise anomalies.

B. RQ1: Performance of VersaGuardian

We evaluate two aspects of the performance of anomaly detection each using a different partitioning of training and test sets. First, we conduct network device anomaly detection in the online mode and evaluate it as an online experiment. Second, in the offline experiment, we adopt the same experiment settings as used in previous work [7], [10], [19], [23], [24].

Online experiment. We evenly split the training set of D1 into 13 segments (1)-day-long data per segment and each has a similar number of anomalies), D2 and D3 each into 5 segments (4)-day-long data per segment and each has a similar number of anomalies), and D4 into 7 segments (2)-day-long data per segment and each has a similar number of anomalies). D2 and D3 have a longer segment because they have fewer anomalies per day. For each dataset, the test set remains the same for a fair comparison (see Table II). Figure 7 shows the average F1 score of *VersaGuardian* and five baseline methods as the amount (scale) of training data increases from 1 segment to 13 consecutive segments for D1, to 5 consecutive segments for D2 and D3, and to 7 consecutive segments for D4. As for *VersaGuardian*, RRCF, LESINN and JumpStarter, they conduct anomaly detection without any training data. Therefore, their performance stays the same when the scale of training data varies.

We can see that *VersaGuardian* and JumpStarter performs significantly better than the four baseline approaches across all segments on all the four datasets. RRCF is less accurate than the other approaches because it aims to detect the anomalous behavior of *a single data point*, which is not suitable in our scenario where the anomalous behavior of *a time series segment* is studied. The F1 scores of learning-based approaches, namely OmniAnomaly and MSCRED, increase as the scale of training set increases, and approach 90% and 60% respectively toward the end. OmniAnomaly achieves higher accuracy than LESINN when the amount of training data is sufficient.

TABLE III
AVERAGE PRECISION (P), RECALL (R), AND F1 SCORE (F) OF *VersaGuardian* AND BASELINE METHODS

Method	D1			D2			D3			D4			Avg.		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
RRCF	40.26	54.45	39.54	44.06	39.02	30.10	28.33	75.33	38.38	44.41	66.73	47.44	39.27	58.88	38.87
LESINN	75.49	77.40	76.43	77.50	87.15	82.04	87.02	90.95	88.94	**93.68**	78.58	83.02	83.42	83.52	82.61
MSCRED	46.19	56.16	50.69	46.91	58.26	51.97	68.21	86.47	76.26	28.11	60.35	34.76	47.36	65.31	53.42
OmniAnomaly	78.19	**95.03**	85.79	77.24	85.84	81.31	89.59	95.02	92.23	64.26	87.26	74.01	77.32	90.79	83.34
JumpStarter	90.35	94.31	92.29	92.05	94.51	93.26	94.14	**99.60**	96.79	63.54	69.87	62.54	85.02	89.57	86.22
VersaGuardian	**94.37**	93.06	**93.21**	**96.57**	**96.57**	**96.51**	**97.26**	99.43	**98.28**	84.72	**96.05**	**89.68**	**93.23**	**96.28**	**94.42**

TABLE IV
THE AVERAGE INITIALIZATION TIME (IT) AND DETECTION TIME (DT) OF *VersaGuardian* AND BASELINE APPROACHES

Approach	RRCF	LESINN	MSCRED	OmniAnomaly	JumpStarter	*VersaGuardian*
IT (min)	20	20	>86400	>86400	20	20
DT (ms)	41.24	118.63	122.82	191.86	127.13	15.28

MSCRED does not perform well because it mainly focuses on the inter-correlations rather than the overall performance of MTS. The F1 Scores of *VersaGuardian* on D1, D2, and D3 are slightly higher than those of JumpStarter, while on D4, the F1 Score is significantly superior to JumpStarter. This discrepancy arises from the fact that D4 comprises a greater number of MTS with periodicity, a capability that JumpStarter lacks in detecting anomalies concealed within periodic MTS. For all approaches except RRCF, they achieve the best performance on D3 because the anomalous patterns in D3 are easier to capture than those in the other three datasets.

Offline experiment. Now we study the potential performance in the offline setting using best F1 score. Since we need a long time to train models of learning-based anomaly detection approaches, we split the dataset into training and test set following the settings in [7]. Then, we calculate the best F1 score of each approach by grid searching their parameters and anomaly thresholds. Table III lists the average best F1 scores of *VersaGuardian* and baseline approaches on each dataset, as well as their corresponding Precision and Recall. The average best F1 score of *VersaGuardian* across the four datasets is 94.42%, significantly higher than those of the other five approaches, which are 86.22%, 83.34%, 53.42%, 82.61%, and 38.87%, respectively. This is because D2 contains a large quantity of noises, and none of the other four approaches besides *VersaGuardian* and JumpStarter is robust to such noises. In contrast, *VersaGuardian* and JumpStarter reconstruct an anomaly-free time series with smoothing extreme values, making it robust to such noises in each dataset. In addition, D4 contains a significant number of anomalies hidden within periodic patterns, while the other datasets contain only a small number of similar anomalies. The capability of *VersaGuardian* in detecting anomalies for periodic metrics gives it a better performance on D1, D2 and D3 compared to JumpStarter. Moreover, *VersaGuardian* outperforms the other five methods, including JumpStarter, by a significant margin on D4.

Efficiency. Table IV lists the average initialization time and detection time of the five approaches across the four datasets. The initialization time of *VersaGuardian* is only twenty minutes, much shorter than those of the deep learning-based methods, *i.e.*, OmniAnomaly and MSCRED. Although RRCF, LESINN and JumpStarter achieve the same initialization time as *VersaGuardian*, they suffer from lower accuracy as shown in Table III. The detection time of *VersaGuardian* for each MTS window is 15.28 ms, which is significantly shorter than the detection time of other approaches. The specific reasons for the efficiency improvement will be explained in section V-C. Among the six approaches, only *VersaGuardian* simultaneously achieves rapid initialization, high detection accuracy, and short detection time.

TABLE V
THE DETECTION TIME (ms) OF *VersaGuardian* AND JUMPSTARTER

Method	D1	D2	D3	D4	Avg.
JumpStarter	172.71	89.75	97.92	148.14	127.13
VersaGuardian	32.09	9.84	10.62	8.57	15.28

C. RQ2: Secret to Accelerating Detection Speed

From Table III, it can be observed that JumpStarter's performance is slightly worse than *VersaGuardian*, and JumpStarter can also achieve rapid initialization, within 20 minutes, just like *VersaGuardian*. However, as shown in Table V, *VersaGuardian* requires significantly less online detection time compared to JumpStarter. The main reason for JumpStarter's slow online detection speed is the relatively slow speed of the compressive sensing reconstruction it utilizes. Additionally, as the time detection window increases, the likelihood of reconstruction failure also increases, further slowing down the detection speed.

In contrast, *VersaGuardian* utilizes DMD, which is a dimensionality reduction technique that extracts low-dimensional structures from high-dimensional data. It inherently tackles the challenge of slow detection speed and has enhanced the detection speed by a factor of 8 compared to JumpStarter. This significant improvement enables the realization of low-latency alarms, making it possible to achieve rapid alerting.

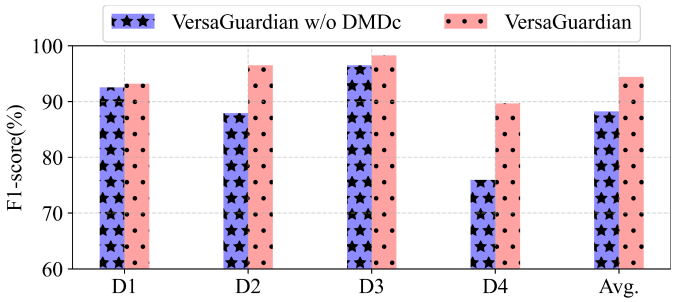


Fig. 8. The F1 Score of *VersaGuardian* when the DMDc is removed.

D. RQ3: Treatment on Anomalies Hidden in Periodicity

The MTS obtained from network devices may exhibit varied temporal characteristics, including the presence of periodic patterns, thereby introducing challenges in the detection of concealed anomalies within periodic MTS. *VersaGuardian* mitigates this challenge by employing STL on time series with periodic patterns, where anomalies are frequently discerned in the remainder component. The seasonal component and the smoothed remainders (obtained after extreme values smoothing) serve as control inputs for DMDc.

By effectively suppressing anomalies hidden in periodic time series while accurately predicting the evolution of normal mode time series, *VersaGuardian* significantly enhances anomaly detection performance. As shown in Figure 8, we compared *VersaGuardian* with and without the DMDc component, which is responsible for detecting anomalies for periodic MTS data. The results indicate that DMDc achieved a 6.18% increase in F1 Score. The incorporation of DMDc and STL significantly improves the accuracy of *VersaGuardian*.

VI. Deployment and Discussion

A. Success Story

Case study. We applied *VersaGuardian* into 30 servers in ByteDance, a top-tier global content platform having more than 800 million daily active users around the world. Operators can register a MTS monitoring task by extracting these time series from influxDB and Kafka. From the monitoring dashboard of these devices, we can see the MTS. Figure 9 shows some time series of two technical outages.

Case I: Long response time caused by network issue. As illustrated in Figure 9a, we observed jitters in time series `tcpext_listendrops` and `tcp_attemptfails`. In the meantime, time series such as `cpu_user` and `load_one` drastically dropped. *VersaGuardian* successfully pinpointed this anomaly and generated an alert. After diagnosing the anomaly, operators found that it was a network issue of a database node.

Case II: Service hang-up due to software change. As illustrated in Figure 9b, time series `tcp_retrans_percentage` witnessed significant jitters and `cpu_idle` plunged to zero. After that, `cpu_sintr`, `cpu_ctxt`, `rx_byptes_eth0` and `tx_pkts_eth0` increased significantly. *VersaGuardian* detected and reported this anomaly to operators. Operators conducted software changes and a configuration error occurred in the new version. Thanks to *VersaGuardian*, operators found out this error in time and quickly rolled out the software change.

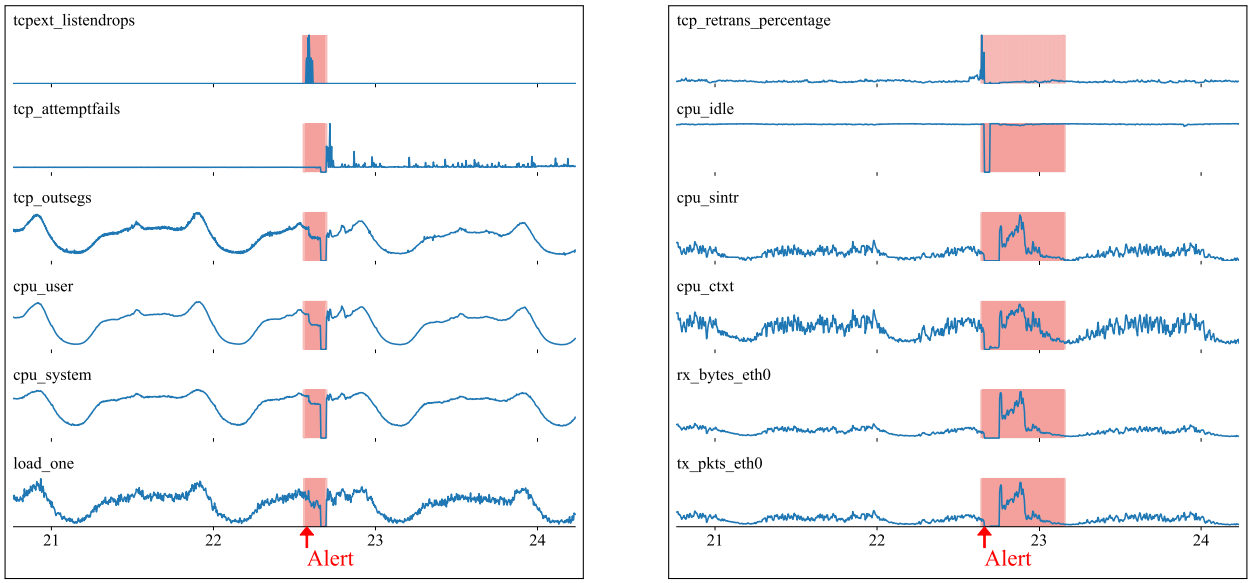
Case III: Request failure due to link switching issue. As illustrated in Figure 10, all these time series exhibit periodicity. Time series `erab_nbrsuccestab_1` and `ho_attoutintraenb` exhibit a significant increase in the middle period compared to others. *VersaGuardian* rapidly detected and reported this anomaly to operators. Operators promptly identify network devices with associated anomalies and optimize link switching.

Help with root cause diagnosis. *VersaGuardian* can help with root cause diagnosis in two aspects. First, hundreds of MTS need to be monitored. After periodicity segmentation in *VersaGuardian*, operators can focus on limited time series groups with a similar period size. Second, *VersaGuardian* respectively calculates the distance between the original univariate time series and the reconstructed ones. Therefore, it can output a rank list of time series' contributions to the overall anomaly. For example, `tcpext_listendrops` in Figure 9 is detected as the most anomalous time series in this figure. It can explicitly indicate the issue was caused by the network component.

B. Lessons Learned

Different network devices may prefer precision and recall differently. With collaboration with different operators, we found that their preferences on precision and recall are diverse. For example, recall weighs more than precision does in a device responsible for user communication since operators do not want to miss any potential anomaly that can negatively impact the user experience. In addition, precision is more valuable in a data analysis job because operators would better detect anomalies precisely than to obtain a lot of false alerts. Therefore, the F1 score alone is not a suitable metric for all network devices. Going forward, we can provide operators with an interface to choose their precision and recall preference level. Specifically, *VersaGuardian* can accordingly set the anomaly score using different parameters of the EVT algorithm to guide the sensitivity of detection output. We also observe that severe faults (examples in Figure 9) rarely happen in network devices but performance issues do happen a lot. We aim to reduce mean time to restore for severe failures in future work.

Alert system is not just anomaly detection. Our *VersaGuardian* for robust and quickly initialized anomaly detection is not the end of the story. Building an intelligent alert system based on anomaly detection results is also a complex task in both engineering and academic aspect. Some anomalies may have no or little signal in the monitored time series. Therefore, *VersaGuardian* may miss these anomalies. For this scenario, we aim to collect more types of monitoring data, *e.g.*, logs, traces, to build a more comprehensive anomaly detection model. We believe *VersaGuardian* can be easily extended for localizing the anomalous metrics, however, there is a significant gap between anomalous metrics and the root causes of anomalies [41], [42]. An intelligent alert system needs to merge similar anomalous cases, pinpoint more sharply to the



(a) Network Issue (b) Software Change

Fig. 9. Two anomaly cases with selected time series (service performance metrics, *e.g.*, average response time, error rate, are hidden for the confidential reason). Time (X-axis) is shown in days. The alert is generated by *VersaGuardian*.

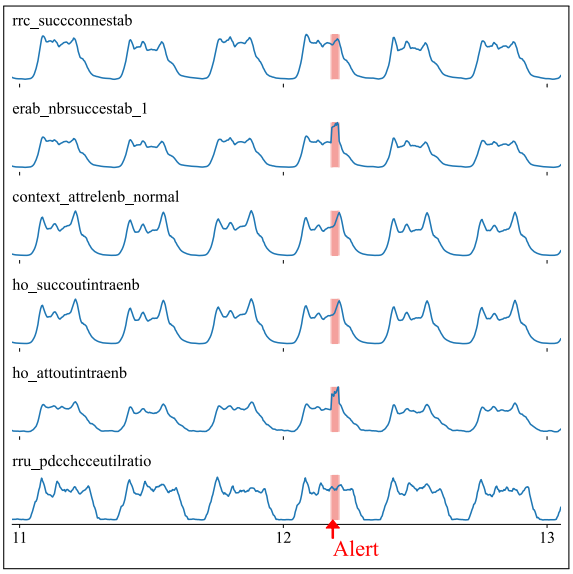


Fig. 10. An anomaly case of periodic time series collected from CMCC. Time (X-axis) is shown in days. The alert is generated by *VersaGuardian*.

root causes of anomalies. It also had better learn the priority of different anomalous cases [9]. Besides, adeptly integrating domain knowledge into the alert system is also of great importance since the system needs feedback from operators. Therefore, apart from *VersaGuardian*, we will improve the alert system behind it.

C. Threats to Validity

Anomaly labeling. In this work, we use four datasets from real-world network devices in production environment. All the labels in these datasets are provided by operators based on performance issues and incident reports. Manually labeling anomaly points in the timeline may introduce noise (false positives or negatives) because no clear boundaries lie in anomalies and normal patterns. However, domain operators with profound experience suggest the noise in those labels accounts for a very small portion. Besides, operators design evaluation metrics that utilize contiguous anomaly segments instead of point-wise anomalies. Adopting these widely used metrics [6], [7], [34], we can also eliminate labeling noises.

Subject systems. In our experiments, we use D1, D2, D3 and D4 from large-scale real-world network devices. The granularity of the time series of these datasets is one, five, five and fifteen minutes, respectively. The efficacy of our algorithm is not influenced by the granularity. With fine-grained granularity, say one second, we believe our algorithm can still work without additional efforts. Since *VersaGuardian*'s versatility has been demonstrated using four datasets collected from 288 different network devices, it should be easy for *VersaGuardian* to work with a new dataset. Admittedly, the number of subject services is still limited. We will test *VersaGuardian* on a variety of network devices in the future.

VII. Conclusion

In this paper, we present an end-to-end approach named *VersaGuardian* for anomaly detection in MTS. This approach is designed for rapid initialization, robust performance and high computational efficiency, aiding operators in detecting anomalies in network devices. *VersaGuardian* employs Fourier transform and periodicity segmentation for periodicity detection and categorization, leveraging clustering to enhance the efficiency of subsequent anomaly detection. By comparing the predicted MTS from DMDc with the original sequence, *VersaGuardian* identifies anomalies. This approach significantly accelerates anomaly detection, achieving low-latency alerts. The incorporation of seasonal and remainder components, extracted through STL of MTS data, enriches *VersaGuardian*'s DMD. This enhancement empowers the method with the

ability to detect anomalies hidden in periodic time series, further elevating its detection accuracy. One remarkable aspect of *VersaGuardian* is its reduced dependency on extensive training data. It rapidly detects deviations from expected patterns using available data. As a result, *VersaGuardian* ensures prompt anomaly detection and facilitates stable large-scale network devices. The effectiveness and efficiency of *VersaGuardian* are proved using four real-world datasets from production environment. *VersaGuardian* achieves superior performance compared to five popular MTS anomaly detection methods.

REFERENCES

[1] M. A. Qureshi, L. Qiu, A. Mahimkar, J. He, and G. Baig, "Multi-dimensional impact detection and diagnosis in cellular networks," in *Proc. 16th Int. Conf. Mobility, Sens. Netw. (MSN)*, Tokyo, Japan, Dec. 2020, pp. 561–568.

[2] L. Tao et al., "Diagnosing performance issues for large-scale microservice systems with heterogeneous graph," *IEEE Trans. Services Comput.*, vol. 17, no. 5, pp. 2223–2235, Oct. 2024.

[3] V. Ganatra et al., "Detection is better than cure: A cloud incidents perspective," in *Proc. 31st ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, San Francisco, CA, USA, S. Chandra, K. Blincoe, and P. Tonella, Eds., Nov. 2023, pp. 1891–1902.

[4] Z. Li et al., "Gandalf: An intelligent, end-to-end analytics service for safe deployment in large-scale cloud infrastructure," in *Proc. 17th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Santa Clara, CA, USA, Feb. 2020, pp. 389–402.

[5] Y. Chen et al., "ImDiffusion: Imputed diffusion models for multivariate time series anomaly detection," 2023, *arXiv:2307.00754*.

[6] H. Xu et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in Web applications," in *Proc. World Wide Web Conf.*, 2018, pp. 187–196.

[7] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2019, pp. 2828–2837.

[8] L. Tao et al., "Giving every modality a voice in microservice failure diagnosis via multimodal adaptive optimization," in *Proc. 39th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2024, pp. 1107–1119.

[9] Y. Chen et al., "Identifying linked incidents in large-scale online service systems," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Nov. 2020, pp. 304–314.

[10] C. Zhang et al., "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1409–1416.

[11] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: Unsupervised anomaly detection on multivariate time series," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds., Aug. 2020, pp. 3395–3404.

[12] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. 35th AAAI Conf. Artif. Intell., 33rd Conf. Innov. Appl. Artif. Intell., (IAAI), 11th Symp. Educ. Adv. Artif. Intell., (EAAI)*, May 2021, pp. 4027–4035.

[13] Z. Li et al., "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Singapore, F. Zhu, B. C. Ooi, and C. Miao, Eds., Aug. 2021, pp. 3220–3230.

[14] H. Ren et al., "Time-series anomaly detection service at Microsoft," in *Proc. 25th SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 3009–3017.

[15] C. Zhao et al., "Robust multimodal failure detection for microservice systems," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Long Beach, CA, USA, A. K. Singh, Y. Sun, L. Akoglu, D. Gunopulos, X. Yan, R. Kumar, F. Ozcan, and J. Ye, Eds., Aug. 2023, pp. 5639–5649.

[16] E. B. Gulcan and F. Can, "Unsupervised concept drift detection for multi-label data streams," *Artif. Intell. Rev.*, vol. 56, no. 3, pp. 2401–2434, Mar. 2023.

[17] Q. Xiang, L. Zi, X. Cong, and Y. Wang, "Concept drift adaptation methods under the deep learning framework: A literature review," *Appl. Sci.*, vol. 13, no. 11, p. 6515, May 2023.

[18] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu, "RobustSTL: A robust seasonal-trend decomposition algorithm for long time series," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, Jul. 2019, pp. 5409–5416.

[19] M. Ma et al., "Jump-Starting multivariate time series anomaly detection for online service systems," in *Proc. 2021 USENIX Annu. Tech. Conf. (USENIX ATC)*, Jul. 2021, pp. 413–426.

[20] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition procedure based on loess," *J. Off. Statist.*, vol. 6, no. 1, pp. 3–73, 1990.

[21] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM J. Appl. Dyn. Syst.*, vol. 15, no. 1, pp. 142–161, 2016.

[22] D. R. Choffnes, F. E. Bustamante, and Z. Ge, "Crowdsourcing service-level network event monitoring," in *Proc. ACM SIGCOMM Conf.*, Aug. 2010, pp. 387–398.

[23] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2712–2721.

[24] G. Pang, K. M. Ting, and D. Albrecht, "LeSiNN: Detecting anomalies by identifying least similar nearest neighbours," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Nov. 2015, pp. 623–630.

[25] Y. Su et al., "Detecting outlier machine instances through Gaussian mixture variational autoencoder with one dimensional CNN," *IEEE Trans. Comput.*, vol. 71, no. 4, pp. 892–905, Apr. 2022.

[26] D. Li et al., "An empirical analysis of anomaly detection methods for multivariate time series," in *Proc. IEEE 34th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2023, pp. 57–68.

[27] S. L. Brunton, J. L. Proctor, J. H. Tu, and J. N. Kutz, "Compressed sensing and dynamic mode decomposition," *J. Comput. Dyn.*, vol. 2, no. 2, pp. 165–191, Jan. 2015.

[28] M. S. Hemati, M. O. Williams, and C. W. Rowley, "Dynamic mode decomposition for large and streaming datasets," *Phys. Fluids*, vol. 26, no. 11, Nov. 2014, Art. no. 111701.

[29] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *J. Fluid Mech.*, vol. 656, pp. 5–28, Jul. 2010.

[30] Z. Bai, E. Kaiser, J. L. Proctor, J. N. Kutz, and S. L. Brunton, "Dynamic mode decomposition for compressive system identification," *AIAA J.*, vol. 58, no. 2, pp. 561–574, Feb. 2020.

[31] Y. Lei, J. Lin, Z. He, and M. J. Zuo, "A review on empirical mode decomposition in fault diagnosis of rotating machinery," *Mech. Syst. Signal Process.*, vol. 35, nos. 1–2, pp. 108–126, Feb. 2013.

[32] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *J. Comput. Dyn.*, vol. 1, no. 2, pp. 391–421, 2014.

[33] X. Zhang et al., "Robust log-based anomaly detection on unstable log data," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 807–817.

[34] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *Proc. IEEE 29th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2018, pp. 13–24.

[35] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using LSTMs and non-parametric dynamic thresholding," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Jul. 2018, pp. 387–395.

[36] D. Liu et al., "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proc. Internet Meas. Conf.*, Oct. 2015, pp. 211–224.

[37] E. Koç and A. Koç, "Fractional Fourier transform in time series prediction," *IEEE Signal Process. Lett.*, vol. 29, pp. 2542–2546, 2022.

[38] R. Bracewell, *The Fourier Transform and its Applications*. New York, NY, USA: McGraw-Hill, 2000.

[39] S. Zhang et al., "Robust system instance clustering for large-scale Web services," in *Proc. ACM Web Conf.*, Lyon, France, F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman, and L. Médini, Eds., Apr. 2022, pp. 1785–1796.

[40] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1067–1075.

[41] M. Ma et al., "Diagnosing root causes of intermittent slow queries in cloud databases," *Proc. VLDB Endow.*, vol. 13, no. 8, pp. 1176–1189, 2020.

[42] Z. Ren, C. Liu, X. Xiao, H. Jiang, and T. Xie, "Root cause localization for unreproducible builds via causality analysis over system call tracing," in *Proc. 34th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2019, pp. 527–538.

Lei Tao received the M.S. degree in software engineering from Nankai University, Tianjin, China, in 2022, where he is currently pursuing the Ph.D. degree with the College of Software. His research interests include anomaly detection and failure diagnosis.

Minghua Ma (Member, IEEE) received the Ph.D. degree in computer science from Tsinghua University. He is currently a Senior Researcher with the Microsoft 365 Research Team, Microsoft. His research spans software engineering, systems, and artificial intelligence. His research interests are primarily focused on AI for IT Operations (AIOps)/ cloud intelligence.

Shenglin Zhang (Member, IEEE) received the B.S. degree in network engineering from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2012, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2017. He is currently an Associate Professor with the College of Software, Nankai University, Tianjin, China. His current research interests include failure detection, diagnosis, and prediction for service management.

Junhua Kuang is currently pursuing the degree with the School of Software, Nankai University. He is participating in the AIOps Laboratory, Nankai University, and has been awarded the National Scholarship.

Xiao-Wei Guo received the master's and Ph.D. degrees from the National University of Defense Technology in 2011 and 2015, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, National University of Defense Technology. His research focuses on interdisciplinary studies, such as computational fluid dynamics, parallel algorithms, and AI for science (AI4S).

Canqun Yang received the M.S. degree from the National University of Defense Technology in 1995 and the Ph.D. degree in 2008. He is currently a Researcher with the College of Computer Science, National University of Defense Technology. He is also the Director of the National Supercomputing Center in Tianjin. His primary research interests include high-performance computing and industrial software.

Dan Pei (Senior Member, IEEE) received the B.E. and M.S. degrees in computer science from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree in computer science from the Computer Science Department, University of California at Los Angeles (UCLA), in 2005. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include network and service management in general. He is a Senior Member of ACM.