2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE) | 979-8-3503-2996-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ASE56229.2023.00082

Maat: Performance Metric Anomaly Anticipation for Cloud Services with Conditional Diffusion

Cheryl Lee[†], Tianyi Yang[†], Zhuangbin Chen[*‡], Yuxin Su[*], and Michael R. Lyu[†]
[†]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China.
Email: cheryllee@link.cuhk.edu.hk, {tyyang, lyu}@cse.cuhk.edu.hk
[*]School of Software Engineering, Sun Yat-sen University, Zhuhai, China.
Email: {chenzhb36, suyx35}@mail.sysu.edu.cn

Abstract—**Ensuring the reliability and user satisfaction of cloud services necessitates prompt anomaly detection followed by diagnosis. Existing techniques for anomaly detection focus solely on real-time detection, meaning that anomaly alerts are issued as soon as anomalies occur. However, anomalies can propagate and escalate into failures, making faster-than-real-time anomaly detection highly desirable for expediting downstream analysis and intervention. This paper proposes Maat, the first work to address anomaly anticipation of performance metrics in cloud services. Maat adopts a novel two-stage paradigm for anomaly anticipation, consisting of metric forecasting and anomaly detection on forecasts. The metric forecasting stage employs a conditional denoising diffusion model to enable multi-step forecasting in an auto-regressive manner. The detection stage extracts anomaly-indicating features based on domain knowledge and applies isolation forest with incremental learning to detect upcoming anomalies. Thus, our method can uncover anomalies that better conform to human expertise. Evaluation on three publicly available datasets demonstrates that Maat can anticipate anomalies faster than real-time comparatively or more effectively compared with state-of-the-art real-time anomaly detectors. We also present cases highlighting Maat's success in forecasting abnormal metrics and discovering anomalies.**

Index Terms—**Could Computing, Anomaly Anticipation, Denosing Diffusion Model, Performance Metric**

I. Introduction

With the wave of cloud computing, recent years have witnessed a dramatic expansion in the scale and complexity of cloud services [1], [2], [3]. However, cloud systems are prone to performance anomalies due to the complexity of the underlying cloud system and service interaction structure [4]. Even worse, minor anomalies could magnify their impacts and escalate into serious failures, worsening end users' satisfaction and resulting in severe revenue implications [5], [6]. Thus, guaranteeing service reliability in cloud systems hinges on effectively managing performance anomalies. As cloud infrastructure scales, monitoring data increases exponentially, making manual anomaly detection unfeasible and costly.

Hence, tremendous efforts have been expended toward the automation of cloud service assurance [7], [8], [9], [10], [11]. These methods rely on run-time data, inclusive of logs, traces, and various performance metrics such as CPU usage, memory consumption, and I/O loads. However, existing studies all focus on real-time detection through alert issuance after suspicious monitoring data are detected, followed by downstream tool provision of relevant data on anomaly localization, root cause analysis, and countermeasures [7]. The downstream analysis is time-consuming, during which anomalies can propagate, cascade and evolve into significant failures. Therefore, there is a pressing need for advanced approaches in handling faster-than-real-time (FTRT) anomaly detection to trigger alarms proactively, which can expedite downstream analysis and provide opportunities for intervention to prevent further propagation and revert cloud services to a stable state.

In light of the limitations of real-time anomaly detection, we propose a novel paradigm, anomaly anticipation, to achieve FTRT anomaly detection. This two-stage paradigm consists of metric forecasting and anomaly detection based on the forecasts combined with real-time observations. Yet, achieving this paradigm of anomaly anticipation is not straightforward. We identify three key challenges: 1) *Aggressive forecasts*: Widely-used forecasting models [12], [13], [14], [15] tend to make conservative forecasts that are restricted to the range of the previous values, making it less likely to forecast anomalous patterns. 2) *Visible forecasting values*: Besides a near-feature binary output (normal or not), the anomaly anticipator needs to forecast concrete and numerical performance metric values to allow for downstream data-driven analysis. 3) *Detection of interest*: The detected anomalies of a completely data-driven and black-box detector may not correspond to the interest of cloud services. The detector should allow the integration of domain expertise to facilitate the understanding of anomalies and gain trust from operational engineers.

By addressing these challenges, we propose Maat, the first performance Metric Anomaly AnTicipation approach for cloud services. Maat employs a conditional denoising diffusion model for the forecasting stage, enabling the autoregressive generation of metrics. It extracts temporal dependencies and cross-metric relations as the condition and samples the next-time metric vector. Denoising diffusion models, inspired by non-equilibrium thermodynamics, have shown great success in high-quality generations, especially in text, images, audio, etc [16], [17], [18]. Motivated by the generative progress of diffusion models, Maat provides a novel application of the diffusion model to address cloud computing challenges by learning probability distributions of historical monitoring metrics to generate a next-time sample of the learned distribu-

[‡]Zhuangbin Chen is the corresponding author.

2643-1572/23/$31.00 ©2023 IEEE
DOI 10.1109/ASE56229.2023.00082

tion conditioned on learned dependencies. In this way, Maat realizes multi-step forecasting in an autoregressive manner and enables accurate forecasts even though the ground truths are abnormal signals with a small probability, thereby addressing the first and second challenges. The detection stage of Maat establishes a set of anomaly-indicating features and applies incremental learning with isolation forest [19]. The expertise-dependent feature extraction and the tree-based model facilitate interpretable anomaly detection. Hence, Maat represents a significant innovation in FTRT anomaly anticipation in cloud services by integrating the forecasting and detection processes, while avoiding over-complexities and total agnosticism.

We evaluate Maat on three widely-used public datasets. The experimental results demonstrate that Maat can achieve comparatively or more optimal FTRT anomaly anticipation compared with state-of-the-art real-time anomaly detectors. In particular, Maat improves the F1-score by a significant margin (15.15%∼79.95%) on average, and it is worth noting that only Maat can issue anomalies before their occurrence. Additionally, the real-time detector component of Maat also exhibits superiority over the competitors (19.17%∼86.76% higher in an average F1-score). Besides, our forecaster outperforms recent advances in probabilistic forecasting of multivariate time series. Specific cases also illustrate Maat's success in forecasting abnormal metrics and distinguishing anomalies.

In summary, the main contributions of this paper are:

- We are the first to propose and formulate anomaly anticipation as a new paradigm, which can be easily extended to achieve the anticipation of root causes, fault diagnosis, and other reliability challenges in cloud computing.
- We propose Maat, the first framework to proactively anticipate cloud-service performance anomalies based on a novel conditional diffusion model. We are the first to apply diffusion models to tackle software engineering challenges. The implementation code has been publicly released on https://github.com/BEbillionaireUSD/Maat.
- We conduct extensive experiments to demonstrate that Maat, as an FTRT anomaly anticipation approach, is comparatively or more effective than state-of-the-art real-time anomaly detectors, thereby enabling alerts in advance and saving much time for downstream analysis.

II. Background and Problem Formulation

This section introduces performance metrics, denoising diffusion models, and the formulation of anomaly anticipation.

A. Background

Herein, we introduce the data and technique, i.e., the denoising diffusion model, based on which this paper works.

1) Cloud-service Performance Metrics: Performance metrics are uniformly sampled real-valued time series measuring the system status, categorized into three main areas generally: resource utilization (e.g., CPU usage, memory usage), workload performance (e.g., error rate, throughput), and service level agreement (e.g., uptime, response time). They are responsive to performance changes and can provide meaningful insights into cloud reliability. Usually, a system has multiple monitoring metrics reflecting different aspects of the system's performance, and the number of collected values of each metric is its *length*.

2) Denoising Diffusion Model: Denoising diffusion models [16] are novel generative models inspired by non-equilibrium thermodynamics. They add noise to the input (i.e, forward) and learn to generate by eliminating the noise (i.e., reverse). In the forward process, Gaussian noise is gradually added to the input $x_0 \sim q(x_0) \in \mathbb{R}^{m \times l}$ containing m l-length metrics with the approximate posterior:

$$q(x_1, \cdots, x_T|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

Each step of the Eq. 1 is:

$$q(x_t|x_{t-1}) := \mathcal{N}(x^n; \sqrt{1-\beta_n}x^{n-1}, \beta_n \boldsymbol{I}) \tag{2}$$

where $\beta_1, \cdots, \beta_N \in (0,1)$ are learned or pre-defined variance schedules. As x_t is only related to x_{t-1}, the forward process is parameterized as a Markov chain. The end, x_T, is totally corrupted to be random noise: $p(x_T) = \mathcal{N}(x_T; \mathbf{0}; \boldsymbol{I})$.

In the reverse process, we define a probability density function $p_\theta(x_0)$ to approximate $q(x_0)$ since the original distribution of inputs is unknown. Starting from x_T, we can restore x_0 with learned Gaussian transitions by iteratively calculating:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t)) \tag{3}$$

where parameters θ are learnable and shared; $\boldsymbol{\mu}_\theta : \mathbb{R}^m \times \mathbb{N} \rightarrow \mathbb{R}^m$ and $\boldsymbol{\Sigma}_\theta : \mathbb{R}^m \times \mathbb{N} \rightarrow \mathbb{R}^+$ are functions of the corrupted input x_t and the forward step's index $t \in \mathbb{N}_+$. Thus, the joint distribution is:

$$p_\theta(x_0, \cdots, x_{T-1}|x_T) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{4}$$

Naturally, the training objective is to minimize the KL-divergence between distributions p and q. With Jensen's inequality and the speeding-up parameterization proposed in [16], the objective is derived to be:

$$\min_\theta \mathcal{L} := \min_\theta \mathbb{E}_{x_0 \sim q(x_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), t} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon_\theta}(x_t, t)\|_2^2 \tag{5}$$

where $\boldsymbol{\epsilon}$ denotes the noise added to the input in the forward process and $\boldsymbol{\epsilon_\theta}$ is a trainable model taking $x_t = \sqrt{\bar{\alpha}_t}x_0 + (1 - \bar{\alpha}_t)\boldsymbol{\epsilon}$ and the index t as inputs.

Hence, $\boldsymbol{\mu}_\theta(x_t, t)$ and $\boldsymbol{\Sigma}_\theta(x_t, t)$ in Eq. 3 are rewritten by (Detailed proofs are given by [16]):

$$\begin{aligned} \boldsymbol{\mu}_\theta(x_t, t) &:= \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon_\theta}(x_t, t) \right), \\ \boldsymbol{\Sigma}_\theta(x_t, t) &:= \sigma_t^2 \boldsymbol{I} \end{aligned} \tag{6}$$

where $\tilde{\beta} := \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t$ if $t > 1$; $\tilde{\beta} := \beta_1$ if $t = 1$.

Once trained, we can sample x_t with any t from the corrupted input $x_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, so by experimentally setting $\boldsymbol{\Sigma}_t^2$ to be $\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}$, we can estimate $\hat{x}_0$ from white noise:

$$x_{t-1} = \boldsymbol{\mu}(x_t, t) + \boldsymbol{\Sigma}_t \cdot \varkappa,\ \varkappa \begin{cases} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), & t > 1 \\ = \mathbf{0}, & t = 1 \end{cases} \tag{7}$$

Thus, given an arbitrary noise vector serving as a pseudo x_T, we can generate a new sample by estimating the $\hat{x}_0$, even if the real x_0 does not exist in a generation task.

To sum up, denoising diffusion models attempt to learn the probability distribution of the performance metrics, and the learned distribution enables sampling values as forecasts.

B. Problem Formulation

This section distinguishes between three tasks: real-time anomaly detection, early anomaly detection and our proposed anomaly anticipation, each presented with its corresponding mathematical formulation.

1) Real-time Anomaly Detection: Real-time anomaly detection is a far-reaching field in cloud computing. Anomaly detectors aim to discover abnormal signals from observed monitoring metrics reflecting various performance aspects, such as CPU usage, memory, network, and so on. Anomalies are observations containing patterns inconsistent with expectations, historical data patterns, or human knowledge. Real-time anomaly detection can be formulated as follows. Given observed data $x_{[1:t]} \in \mathbb{R}^{m \times t}$ at the time of t, where m is the number of metrics, a detector should output $\hat{y}_t \in \{0, 1\}$ to denote the existence of anomalies (normal: 0, abnormal: 1). A real-time anomaly detector is only possible to capture the anomaly after its occurrence.

2) Early Anomaly Detection: When a threat to cloud services is detected, every second counts. Thus, detecting anomalies (including failures, incidents, and outages) in real-time is a bit late in some scenarios [20], [21], [22]. That is where early anomaly detection comes in, which aims to extract the relationship between observations and the future system status. Given current monitoring metrics $x_{[1:t]}$, an early anomaly detector will output $y_{t+s} \in \{0, 1\}$ to indicate upcoming anomalies, where s is the number of advanced time steps.

3) Anomaly Anticipation: We propose anomaly anticipation, a brand-new paradigm to identify anomalies before they occur. Given the observed data $x_{[1:t]}$, the first phase aims to build a model f_1 to forecast the near-feature data, i.e., $f_1(x_{[1:t]}) = \hat{x}_{[t+1:t+s]}$, where s is the number of forecasting time steps. Then, another model f_2 detects anomalies based on the concentration of the observed and forecasted values, i.e., $f_2([x_{[1:t]}, \hat{x}_{[t+1:t+s]}]) = \hat{y}_t$, where $\hat{y}_t \in \{0, 1\}$ indicates whether a current (upcoming) anomaly exists (will occur).

Instead of mining the exclusive relationships between current metrics and upcoming anomalies, anomaly anticipation forecasts predictable metrics and pinpoints abnormal patterns s-step in advance. The first phase is self-supervised, while the second phase can be modeled as unsupervised or semi-supervised. Hence, we propose a simple but effective paradigm to decompose a challenging and not thoroughly studied problem into two known problems with in-depth research achievements. Note that the FTRT anticipation result is usually poorer than the real-time detection result, as forecasting will inevitably introduce imperfections and anomaly anticipation is much more challenging than detection.

III. Motivation and Challenges

This section first introduces the necessity and feasibility of anomaly anticipation, motivated by which we propose a novel paradigm of anomaly anticipation. Afterward, we discuss the challenges of achieving this paradigm, which calls for our solution for effective anomaly anticipation.

A. Motivation

1) Issues of Current Practice: To our best knowledge, most, if not all, current performance metric-based practices are for real-time anomaly detection, aiming to identify anomalies in a target service [11], [9], [10], [23], [24]. Though abnormality itself may not directly result in huge losses, it can cause serious failures [25]. Therefore, anomaly detection, which typically comes before more in-depth analysis, like root cause analysis and fault diagnosis, is given great importance. Despite encouraging progress, existing real-time detectors lack the ability to alert operation engineers before an anomaly occurs and escalates into a failure, leading to potential disasters during such an alert delay. Given the complexity of fine-grained diagnosis and recovery, there is a need for speeding up the prerequisite step, i.e., anomaly detection, to save valuable time for downstream slower analysis.

Early anomaly detection (§II-B2) has gradually gained attention in a broad sense, particularly in predicting incidents [20], outages [21], and disk failures [22], [26] for cloud systems. These efforts aim to extract the relationship between the current data and near-future status, whether normal or not. However, before an anomaly arises, there are usually few visible abnormal behaviors, meaning that the relationships are very subtle and unfathomable. Therefore, existing proactive anomaly detectors are mostly supervised [21], [22], [20], [26]. The lack of high-quality labels and distrust from operation teams have been bottlenecks of these approaches. Even worse, such approaches only provide binary results without providing concrete information on the underlying abnormal data. This makes it difficult to analyze the causes earlier and even prevent anomalies proactively, rendering early detection less useful.

In this context, there is a growing need for more advanced anomaly detectors that can not only detect anomalies faster than real-time but also provide insights into why the detector predicts an impending anomaly. It is crucial to provide visible metric values of the upcoming anomalies to enable earlier downstream analysis.

2) Anticipatability of Anomalies: After discussing the necessity of anticipating anomalies, we present a twofold feasibility illustration. First, due to the inherent temporal correlation within cloud services, as they adopt similar programming models and underlying systems, forecasting their performance metrics is possible. Second, as an anomaly is characterized by a significant deviation from the expected behavior, it must exhibit notable distributional differences from normal metrics, enabling detection within cloud services. Hence, considering the temporal correlation and distributional characteristics of performance anomalies in cloud services, the anticipation of performance anomalies is indeed a feasible task. The detailed illustration will be presented one by one.

Cloud services, regardless of the adopted architecture, such as vertical [27], service-oriented [28], or microservice [29], comprise multiple software components that interact with each other and operate under a defined programming logic, i.e., "do certain reactions to certain user requests or system events". For example, to process a user request "post a new blog", a cloud service needs to execute a series of tasks: a) compress the user-uploaded images, b) safely store the compressed image in a backend storage service (e.g., Amazon S3), c) store the text contents to a backend database. Note that a) is a computation-intensive task, b) and c) are network-intensive. Since metrics reflect the execution status and the execution logic of the underlying cloud system, processing such a request will incur a peak in CPU usage followed by a peak in network traffic. Hence, the intrinsic temporal relation between a cloud service's actions leads to the temporal correlation reflected in performance metrics, whose two adjacent periods are logically related. In summary, the similarity in programming models and underlying infrastructure of cloud services contributes to the temporal correlation of their monitoring metrics.

Anomalous metrics exhibit distributional differences from normal metrics. Previous investigations [30], [31] have identified three types of distributional differences in cloud systems: point anomalies, contextual anomalies, and collective anomalies. Point anomalies indicate global peaks in application latency or resource usage metrics, while contextual anomalies only indicate local anomalous patterns in specific execution environments. Collective anomalies occur when each metric point is normal, but the joint occurrence of metric points becomes abnormal, defined by cumulative observations. For example, compared with the past high throughput, the unexpected fringe of low throughput value is anomalous. Both contextual and collective anomalies are rooted in the intrinsic logic of cloud services. By capturing this logic and the temporal relations of performance metrics, it is possible to foresee the occurrence of contextual and collective anomalous metrics. Thus, anomaly anticipation can be achieved by detecting abnormality in forecasted performance metrics.

Insight: Due to the commonality of programming models and underlying systems in cloud services, the performance metrics exhibit an intrinsic temporal correlation, making their behaviors predictable. By applying anomaly detection techniques to forecasted performance metrics, it is possible to anticipate anomalies in cloud services.

B. Challenges and Solutions

We identify three technical challenges in achieving FTRT anomaly anticipation and provide our point-to-point solutions.

1) Aggressive Forecasting: Anticipating anomalies in cloud services requires accurate forecasted metrics, as forecasting errors can affect the accuracy of anomaly detection. Most forecasting models tend to make conservative forecasts, which restrict the forecasted value within the range of previous values. They are less likely to forecast abnormal signals with low probability. To address this issue, probabilistic generative models such as diffusion models are preferred over discriminative models. We also use a smooth L1 loss function to encourage more aggressive forecasts.

2) Visible Forecasting Values: Existing early anomaly detectors only produce a binary output indicating whether an upcoming anomaly will occur, but a binary output is far from sufficient for downstream analysis and preventive decisions, which require concrete data. To address this challenge, we propose a two-stage forecasting-detecting model that produces numerical performance metrics along with a binary anomaly indicator. This approach empowers downstream data-driven analysis and proactive actions.

3) Detection of anomalies of interest: Enabling human reasoning about the detector decisions is crucial to winning the trust of engineers as well as facilitating diagnosis and proper preventive responses. Totally data-driven methods (unsupervised) identify novel patterns as abnormal, but it does not necessarily correspond to cloud applications. For example, resource usage metrics are excepted to experience peaks during holidays for a traveling-related service. Hence, we apply feature engineering to the concatenation of observed and forecasted metrics. Leveraging domain knowledge, we manually define performance-related features of six categories that can indicate anomalies. Afterward, we build up an interpretable tree-based model to distinguish anomalies from normal ones.

IV. Methodology

Figure 1 presents the overview of Maat, an anomaly anticipation framework for cloud services. It consists of two main components: a conditional diffusion model-based forecaster and a tree-based detector. The forecaster captures the temporal and cross-metric dependencies among performance metrics by encoding them into high-dimensional embeddings as the condition, based on which a denoising diffusion model forecasts the near-feature performance metrics auto-regressively. The detector extracts anomaly-indicating features from the concatenation of observations and forecasts and then employs an incrementally trained isolation forest to anticipate existing or upcoming performance anomalies.

A. Metric Forecasting

This section introduces how we forecast by incorporating conditions to improve upon naive denoising diffusion models, which do not consider the short-term context. Denoising diffusion models, based on probabilistic generation, are selected over other time series prediction approaches based on discriminative models. They can better handle abrupt changes, volatility shifts, and irregular patterns, so they are suitable for our aggressive forecasting with anomalies. Moreover, the reasons for incorporating conditions are two-fold. First, according to the analysis in III-A and existing studies [32], [33], [11], previous metrics directly influence the following values. Second, cross-metric relations also influence performance or resources. For example, a CPU-extensive workload may also

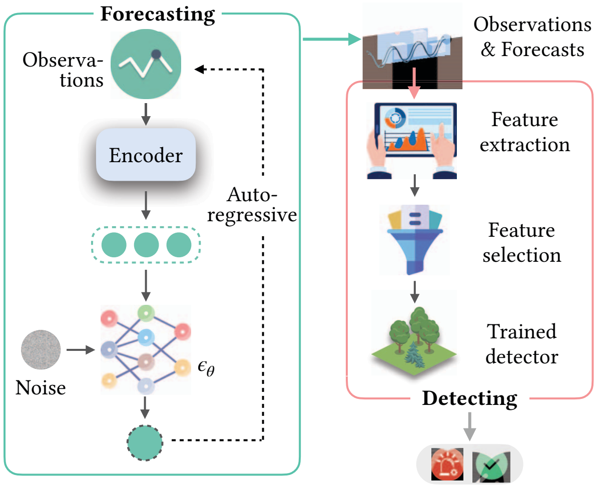


Fig. 1. Overview of Maat.

eat up much memory. As plenty of factors may influence the future's metrics, we regard the near-historical observations as conditions, drawing inspiration from successful applications of conditional time series generation [34], [35], [36]. Figure 2 shows the process of metric forecasting. We will introduce conditioning, training, inference, and model $\mathcal{M}$ in detail.

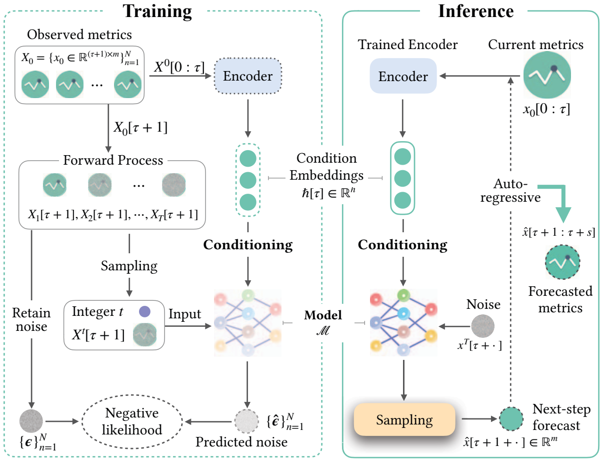


Fig. 2. Work process of the forecaster based on conditional denoising diffusion model.

Conditioning. We first embed performance metrics into a high-dimensional space by extracting meaningful information via an encoder. Various neural networks (e.g. temporal convolutional network [12]) or embedding techniques (e.g., Time2Vec [37]) can serve as the encoder to capture temporal dependencies and cross-metric correlations. We choose the gated recurrent unit (GRU) [13] due to its lightweight architecture and superior performance in signal processing. Through a multi-layer GRU network with learnable parameters θ, we obtain the metric embeddings, denoted by:

$$\hbar = \mathrm{GRU}_\theta([x_0^\tau, c^{\tau+1}], \hbar^{\tau-1}) \tag{8}$$

where $x_0^\tau \in \mathbf{R}^m$ is the τ-th value of the input x_0 with m metrics, and $c^{\tau+1}$ are next-time covariates.

Training. Similar to a naive denoising diffusion model aiming to minimize the KL-divergence between p and q (Eq. 5), the training objective conditioned on $\hbar$ is to narrow the gap between $q(z|\hbar)$ and $p_\theta(z|\hbar)$. In anomaly anticipation, the forecaster should make accurate forecasts even under anomalous contexts, so we encourage aggressive forecasts through smooth convergence by introducing the smooth L1 loss [38] to replace the commonly used L2 loss used in Eq. 5, because smooth L1 loss is more robust and less sensitive to outliers. Given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$, smooth L1 loss is $\mathcal{L}_\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{N}\sum_{n=1}^N l_n$, where

$$l_n = \begin{cases} 0.5(x_n - y_n)^2, & \text{if } |x_n - y_n| < 1 \\ |x_n - y_n| - 0.5, & \text{otherwise} \end{cases}$$

Hence, with N training windows, the training objective in Eq. 5 is further reformulated on conditions and simplified as:

$$\min_\theta \mathcal{L} := \sum_{\tau=l}^{l+s} \min_\theta \mathbb{E}_{x_0^\tau, \boldsymbol{\epsilon}, t}\left[\mathcal{L}_\mathrm{s}\left(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(x_t^\tau, t, \hbar^{\tau-1})\right)\right] \tag{9}$$

where l is the context length and s is the forecasting length, and the superscript n is omitted for simplicity.

Inference. Again, referring to the sampling of a naive diffusion model in Eq. 7, the trained model forecasts the next-step value by sampling from the learned distribution conditioned on $\hbar$:

$$x_{t-1}^{\tau+1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t^{\tau+1} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(x_t^{\tau+1}, t, \hbar^\tau)\right) + \boldsymbol{\Sigma}_t \cdot \varkappa \tag{10}$$

where the initial input $x_T^{\tau+1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \in \mathbb{R}^m$ is an arbitrary noise vector. The forecasts are inputted autoregressively to the encoder so that we can obtain multi-step (s-step) forecasts $\hat{x}^{[l+1:l+s]}$, given the l-length observation $x_0^{[1:l]}$.

Moreover, as the model is probabilistic, we sample more than once and then adopt the mean for smoothing. We can also apply maximum or minimum for more aggressive forecasts. It is up to the application scenario, e.g., CPU usage is relatively stable, while I/O metrics can fluctuate dramatically.

Model $\mathcal{M}$. Having decided the diffusion model, how to predict the noise vector, $\hat{\boldsymbol{\epsilon}} \leftarrow \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\cdot)$, remains. We address this by viewing the forecaster as a special auto-encoder, with both the input and output being noises. Thus, $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\cdot)$ should be a conditional generative decoder. Particularly, we borrow Conditional PixelCNN from the domain of image generation to serve as $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\cdot)$ because it is a powerful decoder capable of efficiently generating novel high-quality images with pixel-level precision, especially considering that anomaly anticipation requires intricate detail and pattern generation. Its flexibility in incorporating different types of conditional information, such as latent embeddings provided by other neural networks, also encourages our selection.

We transformed the 2D network of PixelCNN into 1D for our application, denoted by model $\mathcal{M}$, whose architecture is illustrated in Figure 3. Model $\mathcal{M}$ consists of k residual layers with skip-connections, each with multiple 1D convolution operators, a dilated 1D convolution operator [39], and a gated activation unit that is calculated by:

$$Y_{\text{output}} = \tanh(W_1 \circledast X_{\text{input}}) \odot \sigma(W_2 \circledast X_{\text{input}}) \quad (11)$$

where W_1, W_2 are learnable parameters; $\circledast$ is the convolution operator, and $\odot$ is the element-wise product; σ and tanh are sigmoid and tanh functions, respectively. In addition, we apply the sinusoidal position embedding leveraged in Transformer [15] to embed the diffusion step's index t.

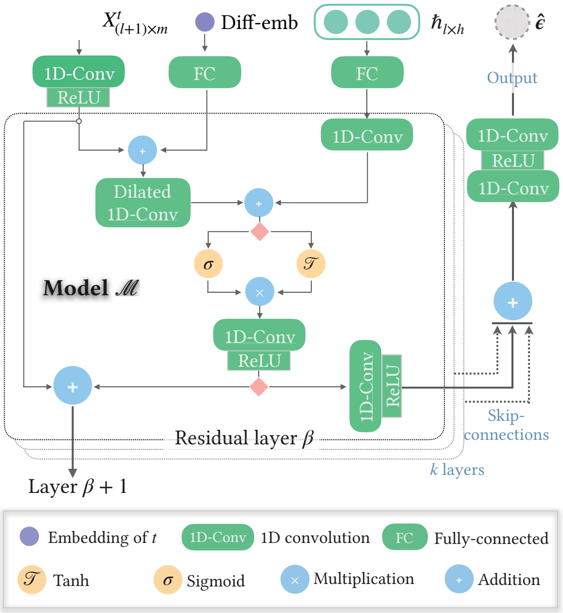


Fig. 3. The architecture of model $\mathcal{M}$ fitting $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\cdot)$, where Diff-emb is the embedding of the step's index t.

B. Faster-than-real-time Detection

Directly combining an advanced data-driven real-time anomaly detector (usually deep learning-based) with our forecaster can cause practicality, interpretability, and adaptability issues. First, overly complex models tend to be impractical for deployment by largely increasing the computation burden. Second, deep learning-based detectors are often black-box and difficult to interpret so as to lose the trust of operation engineers. Third, numerical anomalies may not always equate to operational anomalies in real-world services, making fully data-driven approaches unadaptable in some cases. For example, an inactive service that constantly uses small memory (e.g., 0.1%) and suddenly occupies zero memory may not trigger an alarm in completely data-driven approaches, as there are no spikes, jitters, or other common abnormal behaviors. However, this is an undoubted anomaly because low and no memory usage are qualitatively different. To address these issues, we propose a visible and intuitive anomaly detection algorithm. We define a set of anomaly-indicating features of performance metrics and leverage a tree-based model to achieve the final anticipation.

1) Anomaly-indicting Feature Extraction: This step extracts meaningful features from performance metrics. We systematically identify a set of candidate features that can indicate common anomalies in cloud services based on our expertise and collaboration with industry partners. Representative features are presented in Table I, while additional feature definitions and the calculation code are also available in our online repository. This can also contribute to real-time anomaly detection in the software engineering domain. These features cover typical patterns, such as spikes, jitters, and peaks, and incorporate system understanding and historical failure causes. For instance, a rise in the frequency of virtual memory usage may indicate more virtual page errors. Some features also consider human activities, such as holidays, which may impact network throughput for a ticket service. Besides our defined features, we encourage developers to extend and enhance Maat by customizing the anomaly-indicating features for specific cloud service scenarios. For example, a weather report service may experience a sudden surge in page views when a hurricane approaches, leading to an increase in network traffic. In such a case, developers may not want frequent alerts triggered by such surges, as fluctuations are expected.

2) Feature Selection: This step aims to identify the optimal subset of features by eliminating irrelevant or redundant features, thereby reducing the number of features to improve model accuracy and reduce execution time. Moreover, selecting genuinely relevant features simplifies the model and aids in understanding the data. We select features based on the importance scores of features calculated by Xgboost on a small set with annotations (i.e., the validation set). However, manual annotation is time-consuming and sometimes infeasible. For such cases, we can use the Pearson correlation coefficient or mutual information to calculate the relation between features and remove those strongly related features. Note that all existing feature selection approaches can be directly leveraged in this step, and expertise-dependent selection may perform better in the real world.

3) Anomaly Anticipation: The final step performs anomaly detection on the concatenation of observations and forecasts in an unsupervised manner. Particularly, Maat adopts isolation forest [19] to detect anomalies based on the extracted features. Isolation forest can separate anomalous instances from the rest. It is prevalent in practice due to its high effectiveness, efficiency (with a linear time complexity), and intuitive interpretability, which is important for winning the trust of operation engineers.

Specifically, we first construct an isolation forest model with observations in the training set. The training data $X_{[1:N]} = x_{[1:l]}n = 1^N$ consists of N data windows, where each window $x[1:l] \in \mathbb{R}^{m \times l}$ has m metrics with length l. Each isolation tree (denoted by $iTree$) inside the forest samples a subset of the inputted data with the size of ψ (a hyperparameter) and recursively constructs leaf nodes based on the values of the data attributes until the tree height reaches a pre-defined threshold, or all sampled data are used.

Assuming that the trained isolation forest contains γ trees,

TABLE I
DESCRIPTIONS OF SOME ANOMALY-INDICATING CANDIDATE FEATURES FOR GENERAL CLOUD SERVICES.

Category	Feature	Description	Manifestation
Point	(abs-)Min/Max	(absolute) extremums of computing resources (e.g., CPU, memory)	alarming peaks
	ZeroCount	the number of points with values being zero	dead process
	SpeDayCount	the number of holidays or festivals inside an observed window	expected peaks
	OverZCount	the number of points with the modified Z-score [40] larger than 3.5	short-lived spikes
Frequency domain	FC	Fourier coefficients of performance metrics	voilent fluctuations; jitters
	FTP-param	parameters (centroid, variance, etc.) of absolute Fourier transform spectrum	
	CPSD	cross power spectral density between metrics of the same aspect[†]	
	SK	time-varying spectral kurtosis [41] of IO Bytes	nonstationary regime
Trend	LLS-param	calculate the linear least-squares regression over the observed window and obtain the slope, intercept, standard error, etc.	horizontal/up-/down-trend; linearity;
	LLS-agg-param	calculate the linear least-squares regression over rolling sub-sequences in an observed window and obtain the mean of slopes, intercepts, standard errors, etc.	
	c3	c3 statistics [42] of computing resources measuring the non-linearity	
Temporal dependencies	ACF-μ/-σ	the mean and the variance over the autocorrelation for different lags	unpredictable volatility
	PACF-μ/-σ	the mean and the variance over the partial autocorrelation for different lags	
	margin-Σ	the sum of changes between every two neighboring points of metrics	sudden rise and fall
	(abs)-mar-Min/Max	(absolute) extremums of marginal change of computing resources	
Distribution	std	standard deviation	concept shift, staircases
	skew, kurt	skewness and kurtosis of both single-series metrics and joint multi-variate metrics	
	q-quantiles	the quantile of 10%, 50%, 90%, and the anomaly ratio (if known)	
Cross-series	CID	the complexity-invariant distance [43] between metrics of the same aspect	complexity
	corr	Pearson correlations of metrics between the same and different aspects	cross-metric and cross-aspect relations
	TLCC	time lag cross-correlations of metrics between the same and different aspects	
	MI	pointwise mutual information [44] of metrics between the same and different aspects	

[†] Metrics reflecting diverse aspects may tend to behave differently, e.g., the disk usage is steady, while the CPU usage can fluctuate dramatically without anomalies [11].

Algorithm 1: Incrementally training isolation forest.

```
Input: $X^{\text{cat}}_{[1:N]}$, $\gamma$, $\psi$, $F_{pre}$ - previously trained forest
Output: A new forest $F$ consisting of $\gamma$ trees and $F_{pre}$
Initialize $F$
$i \leftarrow 1$ while $i \leq \gamma$ do
    $X' \leftarrow sample(X^{\text{cat}}_{[1:N]}, \psi)$
    $X'_{iso} \leftarrow F_{pre}(X')$ // Keep the samples "isolated" by $F_{pre}$
    $F \leftarrow F \cup iTree(X_{iso})$
end
return $F$
```

we apply incremental learning to further train the model with forecasts, whose process is described in algorithm 1. The forecasted results $\hat{X}[1:N] = \hat{x}[l+1:l+s]n = 1^N$ are concatenated with the observed data, denoted by $X^{\text{cat}}[1:N] = [x_{[s:l]}; \hat{x}[l+1:l+s]n = 1^N$. The concatenated data are then fed into the established forest. If a concatenation is isolated, we then remove it from the existing forest and build up a new isolation tree until the number of trees reaches the pre-defined threshold γ. Finally, 2γ trees are created in the isolation forest. In this way, only the extremely abnormal concatenated samples are isolated. The idea is based on the fact that anomalies are rare and most of the concatenated samples should be normal, even though there are slight differences between concatenated ones and the original observations $X_{[1:N]}$.

V. EVALUATION

We evaluate Maat by answering three research questions:

- RQ1: How effective is Maat in anomaly anticipation?
- RQ2: How effective is the forecaster of Maat?
- RQ3: How much time can Maat advance anomaly alarm?

A. Experiment Settings

1) Comparative Approaches: As Maat is the first work to address anomaly anticipation, we have to compare it with real-time anomaly detectors. Following the most recent anomaly detection papers [11], [10], we choose eight state-of-the-art baselines: Dount [24], SR-CNN [45], Adsketch [10], Telemanom [46], LSTM-VAE [47], MTAD-GAT [23], DAGMM [48], and OmniAnomaly [9]. Note that Maat anticipates anomalies in advance, whereas the competitors issue alerts after their occurrences. We also evaluate Maat's detector individually by removing the forecaster and only retaining the detector before incremental training to derive its real-time version, Maat-rt, trained and tested on observations as baselines do.

As far as we know, no existing studies directly target performance metric forecasting. Thus, we compare Maat's forecaster with general-purpose deterministic baselines (GRU [13], TCN [12], and Transformer [15]) and advanced probabilistic generation-based methods for multivariate time series (DeepVAR [14], GRU-MAF, and Transformer-MAF [49]). The latter category regards the mean of output as the final forecast.

2) Datasets: We use three wildly used datasets containing complex anomaly patterns. Table II summarizes the statistics of these datasets, where #MetricNum denotes the number of metrics in the dataset, and #MetricLen Avg. denotes the average number of sampled points of each metric. All of them are publicly available with brief introductions as follows.

- *AIOps18* [50] is a real-world dataset consisting of business and performance metrics (inside this paper's scope) collected from web services of a large-scale IT company. The metric interval is either one minute or five minutes.
- *Hades* is a recently released dataset for anomaly detection [11], containing multivariate performance metrics and logs collected from Apache Spark with annotations. It covers various workloads and 21 typical types of faults. The metrics are all equally spaced one second apart.
- *Yahoo!S5* is a benchmark dataset for metric-based anomaly detection [51]. We only consider its real-world metrics sampled every hour, whose anomalies are labeled by hand.

AIOps18 and Hades are multivariate, while Yahoo!S5 is univariate. We choose these different datasets because the real-world needs of engineers may vary from holistic detection over multiple metrics to focusing on a single essential metric.

TABLE II
DATASET STATISTICS.

Dataset	#MetricNum	#MetricLen Avg.	Anomaly Ratio
AIOps18	29	204238.38	2.26%
Hades	11	64,422	24.47%
Yahoo!S5	67	1415.91	1.76%

The used datasets are split based on their collection time: the first 60% is for training, and the subsequent 10% and 30% are for validation and testing, respectively. This splitting guarantees no data leakage with respect to time. Moreover, all methods, except Adsketch, are unsupervised, so only Adsketch requires normal data for training. We first follow Adsketch [10]'s original paper to obtain the training and test sets for AIOps18 and Yahoo!S5. Then we follow Hades's paper [11] to train Adsketch with fault-free data. The validation and testing sets are randomly sampled from the previously split datasets to make the final ratio 6:1:3.

3) Evaluation Measurements: The final results of anomaly anticipation are binary (normal or not), so we adopt the widely-used evaluation measurements of binary classification to gauge Maat:

$$Rec = \frac{TP}{TP + FN}, Pre = \frac{TP}{TP + FP}, F1 = \frac{2 \cdot TP}{2 \cdot TP + FN + TP} \tag{12}$$

where *TP* is the number of correctly detected abnormal samples; *FP* is the number of normal samples incorrectly triggering alarms; *FN* is the number of undetected abnormal samples. Performance metrics in one observation window form a univariate/multivariate sample.

Additionally, we evaluate our forecaster separately with two measurements: mean absolute error and symmetric mean absolute percentage error:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2), sMAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{(|\hat{y}_i| + |y_i|)/2}) \tag{13}$$

where y_i and $\hat{y}_i$ are the real and forecasted values at time i, respectively; N denotes the number of samples. y_i is a float if the input is univariate, and otherwise, for a m-variate input, $y_i \in \mathbb{R}^m$ is a vector. Absolute error-based measurements are preferred because we intend to encourage aggressive forecasting but squared error-based measurements, such as mean squared error (MSE), tend to punish extreme forecasted values. Moreover, MAE is simple to understand, engaging its prevalence in practice; sMAPE is expressed as a percentage, which is scale-independent and suitable for comparing forecasts on different scales.

4) Implementations and Hyperparameters: Our implementation is based on Python 3.8 with open-source packages such as Scikit-learn, Pytorch, TSfresh [52], and Gluonts [53]. We directly deploy the original open-source implementations of all compared methods. In particular, we converted point-wise detection results into segment-level results for fairness by considering the entire window as abnormal once an anomalous data point is detected, which is presented by [24] and applied in many baselines' original papers [24], [23], [48], [9]. All of the experiments are performed on three NVIDIA GeForce GTX 1080 GPUs. We use an open-source toolkit called NNI [54] to optimize the hyperparameters of all approaches automatically, and the best validation *F1* corresponds to the optimal hyperparameter combination.

As for hyperparameters in Maat, the hidden size of metric embeddings is 64, and the number of sampling time steps is 100. We adopt an early-stop mechanism in the forecaster, and the maximum epoch is 50. In each epoch, 100 batches are randomly selected, and the batch size is 256. We use the Adam optimizer with the initial learning rate set to 0.0005. To address the large number of anomaly-indicating features induced by high-dimensional metrics, we apply Xgboost [55] on the validation set to filter features before detection. The detector hyperparameters are default values in Scikit-learn.

B. RQ1: Overall Effectiveness of Maat

From Table III, we can conclude that Maat achieves comparable or even superior effectiveness as an FTRT anomaly anticipator compared to real-time detectors. Maat outperforms all other approaches (except Maat-rt) in terms of the major evaluation measurement *F1*, with *F1* scores being 15.15% to 79.95% higher than competitors on average. Maat's *F1* is only slightly lower than Maat-rt. Note that it is reasonable and expected that Maat's detection is slightly poorer than its real-time version because the forecaster is inevitable to bring imperfections. The slight inferiority actually indicates Maat can make accurate forecasts and deliver results effectively in advance. This finding is important as it suggests that Maat can be a viable approach for anticipating anomalies before they occur in cloud services without sacrificing detection accuracy.

In particular, the superiority of Maat is more significant in Hades and Yahoo!S5, with *F1* being 20.30% to 139.62% higher than competitors for Hades and 12.01% to 97.39% higher for Yahoo!S5. Moreover, Maat's *F1* is only 1.60% lower than the best competitor, Adsketch, for AIOps18. Notably, Adsketch is a semi-supervised approach that requires normal data as training inputs, while Maat can resist noisy

TABLE III
OVERALL PERFORMANCE COMPARISON (%)*.

Mode	Methods	AIOps18†			Hades			Yahoo! S5			**Average**		
		F1	*Rec*	*Pre*	*F1*	*Rec*	*Pre*	*F1*	*Rec*	*Pre*	*F1*	*Rec*	*Pre*
Real-time	Dount	36.60	43.06	31.82	49.17	47.49	50.97	58.30	65.77	52.36	48.02	52.11	45.05
	SR-CNN	44.81	**71.91**	32.54	34.25	61.43	23.74	41.06	61.81	30.74	40.04	65.05	29.01
	Adsketch	64.82	64.28	65.37	65.35	57.47	75.73	58.08	67.28	51.09	62.75	63.01	64.06
	Telemanom	49.49	60.10	42.06	46.75	66.29	36.10	54.10	**77.43**	41.57	50.11	67.94	39.91
	LSTM-VAE	46.35	54.57	40.29	36.89	69.07	25.16	62.77	63.35	62.20	48.67	62.33	42.55
	MTAD-GAT	37.85	46.24	32.04	56.90	55.40	58.48	35.62	31.86	40.38	43.46	44.50	43.63
	DAGMM	53.52	58.08	49.63	62.10	55.62	70.29	57.33	51.70	64.33	57.65	55.13	61.42
	OmniAnomaly	57.40	66.82	50.31	68.17	78.81	60.06	53.13	76.75	40.63	59.57	74.13	50.33
	Maat-rt	**66.75**	64.12	**69.60**	**85.30**	84.35	**86.28**	**72.28**	74.65	70.06	**74.78**	**74.37**	**75.31**
FTRT	**Maat**	63.78	58.94	69.48	82.07	**88.77**	76.31	70.31	69.15	**71.51**	72.05	72.29	72.43

[*] Each value is averaged over three independent runs with different random seeds under the optimal hyperparameters.
[†] In AIOps18, only metrics collected at the intersecting timestamps are considered, resulting in a 16-variate time series.

training data containing anomalies. Furthermore, Maat shows superiority over Adsketch in AIOps18 and Hades, as Yahoo!S5 contains relatively simple data patterns with a low anomaly ratio, where noisy training data has a lesser impact on detection.

In terms of *Pre*, Maat surpasses all other real-time methods by 13.07%∼149.67% on average, indicating that it is effective in targeting anomalies. Some methods achieve higher *Rec* scores on specific datasets, such as SR-CNN and OmniAnomaly on AIOps18 and Telemanom on Yahoo!S5. However, they all perform an imbalance between *Rec* and *Pre*. Note that AIOps18 is a complex dataset with diverse data patterns, making it difficult to model essential normal patterns, resulting in more false positives. In the case of SR-CNN, Spectral Residual (SR) labeling may cause more false positives by regarding complicated data as abnormal, while OmniAnomaly's assumption of generalized Pareto distribution and Gaussian distribution may not hold true in real-world scenarios. Telemanom, on the other hand, predicts the next-time metric and can mistake the inaccurate prediction for anomalies, leading to low *Pre*. Overall, Maat achieves a balance between *Pre* and *Rec*, making it a robust and effective anomaly anticipator for various real-world scenarios.

Moreover, Maat-rt outperforms all competitors by 19.17%∼86.76% on averaged *F1*. Unlike other fully data-driven techniques, Maat-rt incorporates human expertise to design anomaly-indicating features, able to detect suspicious metric segments more in line with operation engineers' desire. In addition, the isolation forest used in Maat-rt fits the most concentrated regions of the training data, regardless of some anomalies, thereby being robust to training noise that wildly exists in real-world data. Besides, it does not assume a priori distributions, enabling Maat-rt to handle different data distributions in practice.

A1: Maat, able to provide the FTRT anomaly anticipation relying on forecasts, performs as well as or better than state-of-the-art real-time detectors based on real observations.

C. RQ2: Effectiveness of Forecasting

Table IV presents the evaluation on Maat's forecaster(Maat-$\mathcal{F}$), demonstrating that Maat-$\mathcal{F}$ outperforms baselines significantly, particularly on complex datasets (AIOps18 and Hades), reducing *MSE* by 44.73%∼89.81% and *sMAPE* by 30.76%∼65.87% on average.

TABLE IV
COMPARISON FOR PERFORMANCE METRIC FORECASTING.

Methods	AIOps18		Hades		Yahoo!S5	
	MSE	*sMAPE*	*MSE*	*sMAPE*	*MSE*	*sMAPE*
GRU	6.170	1.256	3.368	1.957	1.422	1.448
TF*	5.627	1.400	5.628	1.492	1.717	1.443
TCN	4.610	1.230	3.622	0.835	1.111	1.498
DeepVAR	0.428	0.677	1.250	0.692	0.714	1.022
GRU-MAF	2.607	1.451	6.739	1.959	1.180	1.439
TF-MAF	3.235	1.470	2.091	1.677	1.226	1.505
Maat-$\mathcal{F}$	**0.298**	**0.566**	**0.597**	**0.487**	**0.426**	**0.602**

[*] TF is Transformer.

The success of Maat-$\mathcal{F}$ can be attributed to three reasons: 1) The diffusion model's flexibility allows it to forecast abnormal values exceeding the value limit of the contexts, resulting in better and more aggressive forecasting, especially anomalous segments. In contrast, classical baselines tend to make conventional forecasting limited to the range of inputs by directly projecting the historically observed data into the output space, failing to forecast abnormal values with a low probability. 2) Unlike flow-based approaches that are constrained by the invertibility of a sequence of transformations, which limits the expressiveness of the model, our diffusion design can better model the predictive distribution. This explains why MAF-based approaches perform better than traditional deep learning models but still suffer inaccuracy for complex data. Moreover, their failure to estimate the likelihood of out-of-distribution samples makes them even worse, since our data contain multiple anomalies falling off the assumed distributions. 3) Maat-$\mathcal{F}$ is more robust to input noise as it incorporates metric values through the likelihood term rather than directly modeling these values, which is adopted in DeepVAR. Maat-$\mathcal{F}$ demonstrates the greatest improvement in Hades dataset, which contains more anomalous metric segments, with a 52.19% reduction in *MSE* and a 29.60% reduction in *sMAPE*. These results highlight the significant contribution of Maat-$\mathcal{F}$ in improving accuracy for complex and anomalous datasets.

A2: Maat's forecaster performs effectively in metric forecasting, especially in anomalous contexts. Such accuracy enables Maat to attain effective anomaly anticipation results.

D. RQ3: Time Advancing of Maat

Table V presents the overhead of each phase in Maat's inference, including forecasting (#ForeT), feature extraction (#FeatT), and detection (#DeteT), where #ForeL denote the observation length and the forecast length, respectively. #ALen indicates how far in advance can Maat alarm an upcoming anomaly, and the advanced time equals the production of #ALen and the metric sampling interval. For example, Yahoo!S5 is sampled every hour, and Maat can report anomalies 3 hours faster than real-time under our setting, though the potential has not been fully exploited.

TABLE V
TIME CONSUMPTION OF MAAT (UNIT: SECOND).

Dataset	#ALen	#PredT	#FeatT	#DeteT	Total
AIOps18	5	3.031	1.320	0.035	4.386
Hades	3	1.922	0.976	0.036	2.934
Yahoo!S5	3	1.915	0.238	0.036	2.189

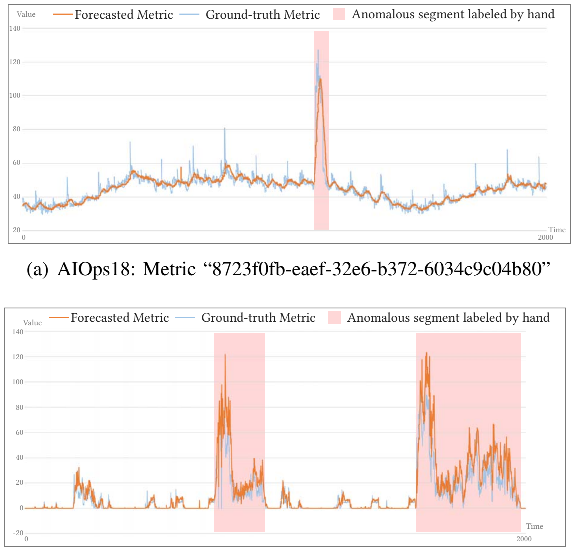
The experimental results show that Maat can effectively anticipate upcoming anomalies 3∼5 time points in advance, saving significant time for downstream analysis. This is because the interval of metric sampling is usually on the order of minutes or hours in practice, while the anticipation time is just seconds, almost negligible in contrast. In addition, the anticipation paradigm of Maat has the potential to prevent anomalies and even serious failures before their occurrence by providing forecasts for downstream automated analysis.

A3: Maat can anticipate anomalies several minutes or hours in advance, with only a few seconds needed for inference, thus imposing negligible overhead on the system. This means that Maat saves a lot of time for downstream analysis and has the potential to prevent anomalies and more serious failures.

E. Successful metric forecasts and feature extraction

We present cases highlighting Maat's success in forecasting abnormal metrics and utilizing anomaly-indicating features. Specifically, Figure 4 displays that Maat can accurately forecast performance metrics, even in abnormal contexts. Though Maat may not be able to forecast the exact same values, and it is almost impossible to do so, it successfully forecasts the trend of suspicious plunges or drops, which implies its ability to issue effective warnings ahead of anomalies' occurrence.

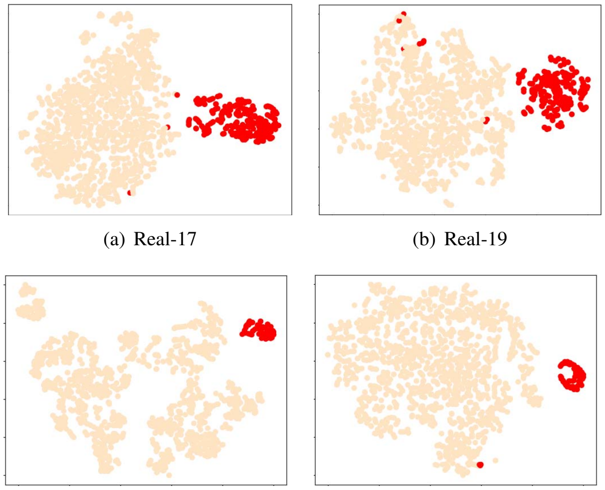
Moreover, we find that using the defined anomaly-indicating features enables the effective isolation of anomalies from normal samples. To illustrate this, Figure 5 showcases the feature distributions of four metrics (named by Real-*) from Yahoo!S5. The figure is constructed by t-SNE [56] with principal component analysis as the inner dimension reduction technique to project the features into a 2D space. Notably, the extracted anomaly-indicating features directly facilitate easy discrimination between anomalies and normal samples. Thus, we posit that properly designed features contribute to the success of Maat in anomaly detection.



(a) AIOps18: Metric "8723f0fb-eaef-32e6-b372-6034c9c04b80"

(b) Hades: Metric "CPU iowait"

Fig. 4. Successful cases of Maat in forecasting metrics with anomalous segments on AIOps18 and Hades.



(a) Real-17

(b) Real-19

(c) Real-22

(d) Real-42

Fig. 5. Distributions of the anomaly-indicating features of four metrics in Yahoo!S5, wherein the bisque points represent normal samples and the red points represent abnormal samples.

VI. DISCUSSION

A. Limitations

Limited generalizability beyond cloud service performance. Maat's anomaly anticipation is tailored to cloud-service performance metrics with predictable anomalies due to their well-explored temporal relations and similar underlying infrastructures. Its effectiveness for other types of time series or external anomalies is uncertain as other anomalies beyond cloud-service performance may be unpredictable.

Pre-defined advancing time. Engineers must decide the time horizon for anomaly anticipation at the very beginning, and Maat cannot adapt to changes once trained. This also leads to an insufficient explosion of Maat's potential since it may also work with an extended horizon, but the effectiveness under such a setting is unknown. Additionally, one trained model can not fit all services, though a one-size-fits-all solution is desirable to handle diverse situations in practice.

Limited applicability in high-frequency sampling. Maat only needs a few seconds for inference and thereby is efficient for most real-world cloud systems with a moderate sampling frequency, allowing anticipation a few minutes or hours ahead. Though most systems apply minutely or hourly sampling frequency due to computational and storage resource limitations, with extremely high sampling frequency, Maat's forecasting inference may become slower than real-time observation, making it impractical for anomaly anticipation.

B. Threats to Validity

Internal. Despite Maat's success in anticipating anomalies, there is room for improvement in forecasting spikes. The design of the learning rate decay function and loss function may still need refinement to prevent the coverage at a local optimum or overly conservative forecasting. To alleviate this threat, we choose the smooth L1 loss instead of the most commonly used L2 loss to encourage aggressive forecasting, whose usefulness is demonstrated in the encouraging experimental results and case studies. On the other hand, occasional unpredictable spikes may not significantly threaten the validity of Maat as it will issue an alert at the anomaly's onset if the anomaly has a lasting impact and thereby forms an anomalous segment. Otherwise, real-time detection is acceptable if the spike's cause only incurs instantaneous effects.

External. The effectiveness of Maat on other datasets is yet unknown. However, we carefully select three representative datasets for evaluation. AIOps18 and Yahoo!S5 are real-world collections from big companies, widely used in existing comparable studies. Hades covers multiple workloads and typical fault types derived from a large-scale company, thereby supporting its representativeness.

VII. Related Work

A. Early Anomaly Detection

Early anomaly detection is a challenging and emerging problem in various domains, including cloud computing. Existing studies (including but not limited to the cloud computing domain) focus on mining the relationships between observed metrics and future status. Some methods directly predict the binary status (normal or abnormal) in the future [57], [20], [58], [22], [59], [26], while others expand real-time binary classification (normal or abnormal currently) into a triple-state classification by adding the state of anomalies about to occur [60], [61]. Despite their success, these methods by means of supervised learning require large amounts of high-quality labels and suffer from imbalanced data, making them difficult to apply in reality. Practically, the number of abnormal or alerting data samples is often much smaller than that of the normal ones. Additionally, they struggle to generate high-quality intermediate forecasts for downstream analysis after anomaly detection, limiting their potential to take full advantage of early detection and steer services from failure.

B. Real-time Anomaly Detection

Existing anomaly detection approaches fall into two categories: statistical or machine learning-based [62], [10], [63] and deep learning-based [24], [9], [23], [48], [47], [46], [64], [65], [11], [66]. Deep learning-based approaches model temporal dependencies and cross-metric relationships, and are mostly reconstruction-based or forecasting-based. Reconstruction-based approaches reconstruct the probability distribution of normal data via the family of auto-encoders [24], [9], [47]. Forecasting-based approaches [46], [64] assume that the forecasted values conform to the distribution of the observations. This assumption is both plausible and flawed, especially when it comes to complex distributions. MTAD-GAT [23] combines both, improving the detection accuracy while dramatically increasing the approach's complexity. Moreover, some studies incorporate multi-modal information like logs [11], [66], [65], and promising detection performance has been achieved. However, traditional real-time anomaly detectors are reactive and can only alarm anomalies after occurrences, which may be a bit late. Hence, we propose a novel anomaly anticipation approach to issue anomalies in advance, allowing for pre-anomaly intervention and saving time for downstream analysis before anomalies escalate into serious failures.

VIII. Conclusion

This paper proposes and formulates the paradigm anomaly anticipation for improving cloud service reliability for the first time, which consists of two stages: performance metric forecasting and anomaly detection on forecasts. We propose Maat, the first framework to address this problem. Maat presents a novel conditional denoising diffusion model and defines anomaly-indicating features that facilitate distinguishing anomalies from normal metric segments. Experiments on three large-scale datasets demonstrate the effectiveness of Maat in faster-than-real-time anomaly anticipation, and cases also show that Maat can make accurate forecasts even under anomalous contexts. Lastly, we release our code, hoping to provide a useful tool for practitioners and lay the foundation for future research in faster-than-real-time cloud operation.

Acknowledgement

The work described in this paper was supported by the Key-Area Research and Development Program of Guangdong Province(No. 2020B010165002) and the Key Program of Fundamental Research from Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109113403826). It was also supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14206921 of the General Research Fund) and the National Natural Science Foundation of China (No. 62202511).

References

[1] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 130:1–130:37, 2022. [Online]. Available: https://doi.org/10.1145/3460345

[2] Z. Chen, Y. Kang, L. Li, X. Zhang, H. Zhang, H. Xu, Y. Zhou, L. Yang, J. Sun, Z. Xu, Y. Dang, F. Gao, P. Zhao, B. Qiao, Q. Lin, D. Zhang, and M. R. Lyu, "Towards intelligent incident management: why we need it and how we make it," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*. ACM, 2020, pp. 1487–1497. [Online]. Available: https://doi.org/10.1145/3368089.3417055

[3] Z. Zheng, X. Li, M. Tang, F. Xie, and M. R. Lyu, "Web service qos prediction via collaborative filtering: A survey," *IEEE Trans. Serv. Comput.*, vol. 15, no. 4, pp. 2455–2472, 2022. [Online]. Available: https://doi.org/10.1109/TSC.2020.2995571

[4] K. Ye, "Anomaly detection in clouds: Challenges and practice," in *Proceedings of the first Workshop on Emerging Technologies for software-defined and reconfigurable hardware-accelerated Cloud Datacenters, ETCD@ASPLOS 2017, Xi'an, China, April 8, 2017*. ACM, 2017, pp. 6:1–6:2. [Online]. Available: https://doi.org/10.1145/3129457.3129497

[5] C. Delimitrou and C. Kozyrakis, "Hcloud: Resource-efficient provisioning in shared cloud systems," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2016, Atlanta, GA, USA, April 2-6, 2016*. ACM, 2016, pp. 473–488.

[6] T. Yang, J. Shen, Y. Su, X. Ling, Y. Yang, and M. R. Lyu, "AID: efficient prediction of aggregated intensity of dependency in large-scale cloud systems," in *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*. IEEE, 2021, pp. 653–665. [Online]. Available: https://doi.org/10.1109/ASE51524.2021.9678534

[7] T. Yang, J. Shen, Y. Su, X. Ren, Y. Yang, and M. R. Lyu, "Characterizing and mitigating anti-patterns of alerts in industrial cloud systems," *CoRR*, vol. abs/2204.09670, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.09670

[8] Y. Li, X. Zhang, S. He, Z. Chen, Y. Kang, J. Liu, L. Li, Y. Dang, F. Gao, Z. Xu, S. Rajmohan, Q. Lin, D. Zhang, and M. R. Lyu, "An intelligent framework for timely, accurate, and comprehensive cloud incident detection," *ACM SIGOPS Oper. Syst. Rev.*, vol. 56, no. 1, pp. 1–7, 2022. [Online]. Available: https://doi.org/10.1145/3544497.3544499

[9] 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, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 2828–2837.

[10] Z. Chen, J. Liu, Y. Su, H. Zhang, X. Ling, and M. R. Lyu, "Adaptive performance anomaly detection for online service systems via pattern sketching," in *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 2022, pp. 61–72.

[11] C. Lee, T. Yang, Z. Chen, Y. Su, Y. Yang, and M. R. Lyu, "Heterogeneous anomaly detection for software systems via semi-supervised cross-modal attention," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 1724–1736.

[12] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 1003–1012.

[13] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*. Association for Computational Linguistics, 2014, pp. 103–111.

[14] D. Salinas, M. B. Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank gaussian copula processes," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 6824–6834.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, December 4-9, Long Beach, CA, USA*, 2017, pp. 5998–6008.

[16] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12, virtual*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

[17] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/92c3b916311a5517d9290576e3ea37ad-Abstract.html

[18] ——, "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 11 895–11 907. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html

[19] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 3:1–3:39, 2012.

[20] N. Zhao, J. Chen, Z. Wang, X. Peng, G. Wang, Y. Wu, F. Zhou, Z. Feng, X. Nie, W. Zhang, K. Sui, and D. Pei, "Real-time incident prediction for online service systems," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*. ACM, 2020, pp. 315–326.

[21] Y. Chen, X. Yang, Q. Lin, H. Zhang, F. Gao, Z. Xu, Y. Dang, D. Zhang, H. Dong, Y. Xu, H. Li, and Y. Kang, "Outage prediction and diagnosis for cloud service systems," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. ACM, 2019, pp. 2659–2665. [Online]. Available: https://doi.org/10.1145/3308558.3313501

[22] C. Luo, P. Zhao, B. Qiao, Y. Wu, H. Zhang, W. Wu, W. Lu, Y. Dang, S. Rajmohan, Q. Lin, and D. Zhang, "NTAM: neighborhood-temporal attention model for disk failure prediction in cloud platforms," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2021, pp. 1181–1191.

[23] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*. IEEE, 2020, pp. 841–850.

[24] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 2018, pp. 187–196.

[25] P. Huang, C. Guo, L. Zhou, J. R. Lorch, Y. Dang, M. Chintalapati, and R. Yao, "Gray failure: The achilles' heel of cloud-scale systems," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems, HotOS 2017, Whistler, BC, Canada, May 8-10, 2017*. ACM, 2017, pp. 150–155.

[26] Y. Xu, K. Sui, R. Yao, H. Zhang, Q. Lin, Y. Dang, P. Li, K. Jiang, W. Zhang, J. Lou, M. Chintalapati, and D. Zhang, "Improving service availability of cloud systems by predicting disk error," in *2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*. USENIX Association, 2018, pp. 481–494. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/xu-yong

[27] W3C, "Vertical applications," 2015. [Online]. Available: https://www.w3.org/standards/semanticweb/applications.html

[28] IBM, "What is service-oriented architecture (soa)?" 2023. [Online]. Available: https://ibm.com/topics/soa

[29] Microsoft, "Microservices architecture style," 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

[30] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[31] T. Hagemann and K. Katsarou, "Reconstruction-based anomaly detection for the cloud: A comparison on the yahoo! webscope S5 dataset," in *ICCBDC '20: 2020 4th International Conference on Cloud and Big Data Computing, Virtual United Kingdom, August, 2020*. ACM, 2020, pp. 68–75.

[32] Y. Gan, M. Liang, S. Dev, D. Lo, and C. Delimitrou, "Sage: practical and scalable ml-driven performance debugging in microservices," in *ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021*. ACM, 2021, pp. 135–151.

[33] Y. Gan, Y. Zhang, K. Hu, D. Cheng, Y. He, M. Pancholi, and C. Delimitrou, "Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*. ACM, 2019, pp. 19–33.

[34] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," in *Proceedings of the 38th International Conference on Machine Learning, ICML, 18-24 July, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 8857–8868. [Online]. Available: http://proceedings.mlr.press/v139/rasul21a.html

[35] G. Papamakarios, I. Murray, and T. Pavlakou, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, December 4-9, Long Beach, CA, USA*, 2017, pp. 2338–2347.

[36] V. L. Guen and N. Thome, "Probabilistic time series forecasting with shape and temporal diversity," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12, 2020, virtual*, 2020.

[37] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. A. Brubaker, "Time2vec: Learning a vector representation of time," *CoRR*, vol. abs/1907.05321, 2019.

[38] R. B. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1440–1448.

[39] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[40] I. Boris and H. D. C, *how to detect and handle outliers*. Quality Press, 1993, vol. 16.

[41] J. Wodecki, "Time-varying spectral kurtosis: Generalization of spectral kurtosis for local damage detection in rotating machines under time-varying operating conditions," *Sensors*, vol. 21, no. 11, p. 3590, 2021.

[42] T. Schreiber and A. Schmitz, "Discrimination power of measures for nonlinearity in a time series," *Phys. Rev. E*, vol. 55, pp. 5443–5447, May 1997.

[43] G. Batista, X. Wang, and E. Keogh, "A complexity-invariant distance measure for time series," in *Proceedings of the 2011 SIAM international conference on data mining*, 04 2011, pp. 699–710.

[44] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[45] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 3009–3017.

[46] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. ACM, 2018, pp. 387–395.

[47] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *CoRR*, vol. abs/1607.00148, 2016. [Online]. Available: http://arxiv.org/abs/1607.00148

[48] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[49] K. Rasul, A. Sheikh, I. Schuster, U. M. Bergmann, and R. Vollgraf, "Multivariate probabilistic time series forecasting via conditioned normalizing flows," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[50] N. L. of Tsinghua University. 2018 international aiops challenge. [Online]. Available: http://iops.ai/competition_detail

[51] Y. Research. A benchmark dataset for time series anomaly detection. [Online]. Available: https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly

[52] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh - A python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018. [Online]. Available: https://doi.org/10.1016/j.neucom.2018.03.067

[53] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang, "GluonTS: Probabilistic and Neural Time Series Modeling in Python," *Journal of Machine Learning Research*, vol. 21, no. 116, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/19-820.html

[54] Microsoft. (2021, 1) Neural Network Intelligence. [Online]. Available: https://github.com/microsoft/nni

[55] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[56] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[57] X. Gu and H. Wang, "Online anomaly prediction for robust cluster systems," in *2009 IEEE 25th International Conference on Data Engineering*, 2009, pp. 1000–1011.

[58] S. Tuli, G. Casale, and N. R. Jennings, "Pregan: Preemptive migration prediction network for proactive fault-tolerant edge computing," in *IEEE INFOCOM - IEEE Conference on Computer Communications, London, United Kingdom, May 2-5, 2022*. IEEE, 2022, pp. 670–679.

[59] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann, "Predicting disk replacement towards reliable data centers," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 39–48. [Online]. Available: https://doi.org/10.1145/2939672.2939699

[60] S. Huang, C. J. Fung, K. Wang, P. Pei, Z. Luan, and D. Qian, "Using recurrent neural networks toward black-box system anomaly prediction," in *24th IEEE/ACM International Symposium on Quality of Service, IWQoS 2016, Beijing, China, June 20-21, 2016*. IEEE, 2016, pp. 1–10. [Online]. Available: https://doi.org/10.1109/IWQoS.2016.7590435

[61] Y. Tan, X. Gu, and H. Wang, "Adaptive system anomaly prediction for large-scale hosting infrastructures," in *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, PODC 2010, Zurich, Switzerland, July 25-28, 2010*. ACM, 2010, pp. 173–182.

[62] M. J. Chambers and M. A. Thornton, "Discrete time representation of continuous time arma processes," *Econometric Theory*, vol. 28, no. 1, pp. 219–238, 2012.

[63] G. Pang, K. M. Ting, and D. W. Albrecht, "Lesinn: Detecting anomalies by identifying least similar nearest neighbours," in *IEEE International Conference on Data Mining Workshop, ICDMW 2015, Atlantic City, NJ, USA, November 14-17, 2015*. IEEE Computer Society, 2015, pp. 623–630.

[64] N. Ding, H. Gao, H. Bu, H. Ma, and H. Si, "Multivariate-time-series-driven real-time anomaly detection based on bayesian network," *Sensors*, vol. 18, no. 10, 2018.

[65] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Eadro: An end-to-end troubleshooting framework for microservices on multi-source data," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 1750–1762.

[66] N. Zhao, J. Chen, Z. Yu, H. Wang, J. Li, B. Qiu, H. Xu, W. Zhang, K. Sui, and D. Pei, "Identifying bad software changes via multimodal anomaly detection for online service systems," in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*. ACM, 2021, pp. 527–539.