ADAMAS: Adaptive Domain-Aware Performance Anomaly Detection in Cloud Service Systems

Wenwei Gu*, Jiazhen Gu*, Jinyang Liu*, Zhuangbin Chen†§, Jianping Zhang*,
Jinxi Kuang*, Cong Feng‡, Yongqiang Yang‡, and Michael R. Lyu*
*The Chinese University of Hong Kong, China.
Email: {wwgu21, jyliu, jpzhang, jxkuang22, lyu}@cse.cuhk.edu.hk, jiazhengu@cuhk.edu.hk
†Sun Yat-sen University, China. Email: {chenzhb36}@mail.sysu.edu.cn
‡Huawei Cloud Computing Technology Co., Ltd, China. Email: {fengcong5, yangyongqiang}@huawei.com

***Abstract*—A common practice in the reliability engineering of cloud services involves the collection of monitoring metrics, followed by comprehensive analysis to identify performance issues. However, existing methods often fall short of detecting diverse and evolving anomalies across different services. Moreover, there exists a significant gap between the technical and business interpretation of anomalies, *i.e.*, a detected anomaly may not have an actual impact on system performance or user experience. To address these challenges, we propose ADAMAS, an adaptive AutoML-based anomaly detection framework aiming to achieve practical anomaly detection in production cloud systems. To improve the ability to detect cross-service anomalies, we design a novel unsupervised evaluation function to facilitate the automatic searching of the optimal model structure and parameters. ADAMAS also contains a lightweight human-in-the-loop design, which can efficiently incorporate expert knowledge to adapt to the evolving anomaly patterns and bridge the gap between predicted anomalies and actual business exceptions. Furthermore, through monitoring the rate of mispredicted anomalies, ADAMAS proactively re-configures the optimal model, forming a continuous loop of system improvement. Extensive evaluation on one public and two industrial datasets shows that ADAMAS outperforms all baseline models with a 0.891 F1-score. The ablation study also proves the effectiveness of the evaluation function design and the incorporation of expert knowledge.**

***Index Terms*—AutoML, Cloud System Reliability, Domain Knowledge, Cloud Service Systems**

I. Introduction

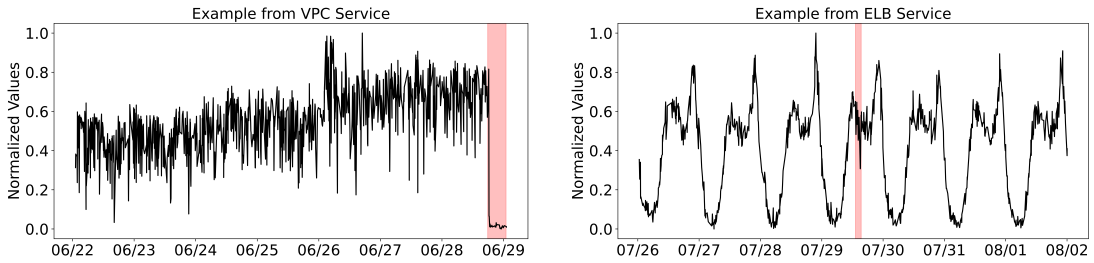
In recent years, many traditional software systems have been migrated to cloud computing platforms, which are managed effectively as *cloud services*. They serve hundreds of millions of users around the world on a 24/7 basis [1], [2], [3]. Due to their increasing scale and complexity, performance issues become inevitable [4], [5]. A common practice of reliability engineering involves the gathering of system monitoring metrics, also known as Key Performance Indicators (KPIs), from software and hardware components (*e.g.*, virtual machines and network devices). The collected data are then analyzed to identify any potential performance issues [6], [7], [8]. However, the overwhelming volume of monitoring metrics [9] in modern cloud services render manual performance analysis infeasible [10]. As a result, many automated anomaly detection methods have been proposed [11], [12], [7], [13], aiming at revealing the unusual patterns of the metrics that reflect potential performance anomalies.

Modern service technologies, *e.g.*, microservices and serverless functions, decouple software into sophisticated and fine-grained units [14]. Thus, cloud services tend to exhibit great diversity and dynamism not only in functionalities but also in anomaly patterns [15]. This situation poses two significant challenges for practical anomaly detection in production systems. First, it is difficult for a single method to effectively identify the entire spectrum of anomalies across different services, *i.e.*, the No Free Lunch Theorem [16], [17], [18]. For example, Fig. 1 presents two metrics with performance anomalies from a Virtual Private Cloud (VPC) service and an Elastic Load Balancing (ELB) service. A prevalent approach to anomaly detection involves first learning the normal pattern of a metric time series and then identifying anomalies when data points deviate from this established norm [17], [19]. While this strategy can work well in the first example, it may not be as effective for anomalies that violate time series periodicity, as illustrated in the second example, where the magnitude of deviation is not prominent. On the other hand, approaches [20], [21] that leverage the periodicity for anomaly detection face a different dilemma. The second challenge is related to the data-driven nature of anomaly detection. Existing approaches obtain the best model based on the experimental datasets. However, an anomaly detected on a technical level might not necessarily translate to a performance issue that impacts the overall business performance or customer experience. Consequently, engineers need to deal with a lot of false positives. Due to the frequent service updates and user behavior changes, the anomaly patterns of services may evolve accordingly, *i.e.*, concept drift [22], which further compounds the problem.

While efforts have been devoted to addressing these challenges, they suffer from important limitations that hinder their practical applications. For instance, [23], [24] employ the technique of Automated Machine Learning (AutoML) to prevent designing service-specific anomaly detection solutions [25], [26]. However, existing AutoML-based approaches often require a considerable amount of labeled data or cannot properly evaluate the model performance, hindering the search for the optimal model architecture and parameters. In real-world systems, obtaining sufficient labeled data for each cloud service

§ Zhuangbin Chen is the corresponding author.

1558-1225/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSE55347.2025.00084



(a) Network Interrupt Anomaly in VPC Service

(b) Packets Loss Anomaly in ELB Service

Fig. 1. Examples of anomaly patterns of different services in cloud company $\mathcal{X}$

is often challenging [27]. On the other hand, the human-in-the-loop mechanism [28], [29] has been proven effective in incorporating domain knowledge to mitigate the gap between the predicted anomalies and the actual business exceptions. However, existing approaches heavily rely on human expertise to provide high-quality feedback, which impacts the scalability and efficiency. Simply using limited feedback data to retrain the models can hardly guarantee performance.

In this paper, we propose ADAMAS (Adaptive Domain-Aware Performance Anomaly detection for cloud Services Systems), an AutoML-based anomaly detection framework adaptive to different cloud services with diverse and evolving abnormal patterns. Specifically, ADAMAS consists of two stages, *i.e.*, a *Label-free Configuration Search stage* and a *Feedback-based Adaptive Learning stage*. In the first stage, ADAMAS employs Bayesian Optimization [30] to automatically search the best model architecture with parameters for anomaly detection. During the search, ADAMAS utilizes the proposed Noise-Free Mean squared error with Kurtosis (NFMK) evaluation function to estimate model performance without labels. In the second stage, ADAMAS adopts a lightweight, adaptive learning approach to efficiently incorporate expert knowledge. Specifically, during online serving, engineers can provide feedback on whether the detected anomaly is a false positive. Given the manually labeled data, ADAMAS employs Metric Stream Clustering (MSC) to group similar anomalous patterns into clusters and leverage historical feedback to identify true performance issues or false positives, significantly reducing the feedback required. To close the loop of continuous service updates, ADAMAS triggers model retraining when the cumulative mispredicted samples exceed a threshold. In this process, the domain knowledge introduced by human feedback will guide the configuration search.

To evaluate the performance of ADAMAS, we conduct extensive experiments on one public and two industrial datasets. The results demonstrate that ADAMAS outperforms all baselines with an F1-score of 0.891. An ablation study demonstrates that compared to other evaluation functions, NFMK is more effective, and the MSC design can help ADAMAS achieve better performance with less feedback. A case study further shows how ADAMAS works in industrial cloud systems. To sum up, our main contributions are as follows:

- We design and implement ADAMAS, a domain-aware AutoML-based anomaly detection framework. It addresses two practical challenges in this field, namely, the diverse anomaly patterns across different services and the gap between the predicted anomalies and the actual performance issues. It eliminates the dependency on labels in the model searching process and significantly reduces the amount of human feedback for expert knowledge integration.
- We conduct extensive experiments with public data as well as industrial data collected from Company $\mathcal{X}$. Furthermore, our framework has been successfully incorporated into $\mathcal{X}$'s performance monitoring and anomaly detection system.

II. Background

In this section, we first introduce some background about metric-based performance anomaly detection in modern cloud systems. An example from an industrial scenario that motivates this work is also present. Then, we give a brief introduction to the general workflow of AutoML.

A. Performance Anomaly Detection in Cloud Service Systems

In modern cloud service systems, software reliability engineers (SREs) usually collect a large number of metrics that track the health status of the services (*e.g.*, CPU utilization, memory usage and network traffic) with monitoring tools like Grafana, Splunk, *etc.* [31]. Monitoring metrics provide the most granular information, which can be used to derive other types of cloud monitoring data. For example, alerts are usually triggered when metrics cross a threshold defined by the engineers [32]. In cloud systems, there are usually redundant components that provide fault tolerance and self-healing [7] capabilities. Thus, most of the service outages manifest themselves as performance anomalies (also known as fail-slow failures [33], [34]) before fail-stop failures, which can be detected through analysis of monitoring metrics. Previous studies [35], [7] have demonstrated that performance anomalies of similar types tend to trigger similar symptoms on the monitoring metrics. Thus, a particular type of performance anomaly can be characterized as an anomaly pattern.

However, we observe that not all anomaly patterns manifest in monitoring metrics are true performance anomalies. An industrial example from Cloud Company $\mathcal{X}$ is shown in Fig. 2, where the traffic of a load balancer in the Relational Database Service (RDS) is monitored. In the red area, the metric suddenly dips to nearly zero. It persists until system maintainers replace the network devices with new ones, which

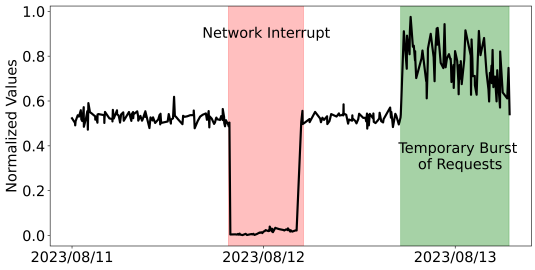


Fig. 2. An Motivating Example from Company $\mathcal{X}$

is confirmed as a network interrupt failure. In the green area, as load balancers can handle unpredictable traffic surges [36] by scaling up the hardware and software resources, the service can recover from a burst of requests due to its auto-scaling feature without manual intervention in the green area. Thus, this anomaly pattern is not a true anomaly because the fluctuating requests are expected behaviors. Due to the recurring nature of performance anomaly and failures [37], these false positives can bring meaningless trouble to engineers if not properly handled. A natural solution is to automatically differentiate the types of performance anomalies and determine the true anomaly by considering the context of the specific service.

B. Automated Machine Learning

In a large-scale cloud system, there are many services with different anomaly patterns that need different configurations to achieve optimal performance. Though there are many on-hand anomaly detection methods, in software reliability engineers' view, an "out of the box" tool is more desired. Automated Machine Learning (AutoML), the process that makes machine learning easier by avoiding tedious manual hyperparameter tuning for both machine learning experts and non-experts, provides great benefits to cloud system operators. Typically, AutoML can be formulated as a Combined Algorithm Selection and Hyper-parameter (CASH) problem [38].

A general AutoML workflow is shown in Fig. 3. The optimizer fetches model configurations based on a search space and the observations from the evaluator. Next, the evaluator measures the model trained with configuration passed from the optimizer on some objective functions, and the observation is used to update the optimizer. The model configuration refers to both the model selection and model hyperparameters. After several iterations, the model that achieves the best performance will be output as the output model.

There are extensive studies on AutoML that are dedicated to improving the efficiency and the effectiveness of the optimizer [39], [40], [41], [42]. However, to the best of our knowledge, none of the existing AutoML methods possess the merit of domain knowledge that plays a critical role in differentiating true anomalies and false positives. Furthermore, existing AutoML frameworks lack the adaptability to new anomaly patterns in online services, which hinders their application in dynamic, evolving cloud systems.

III. Methodology

In this section, we present ADAMAS, which is a Bayesian Optimization-based approach for identifying performance issues based on monitoring metrics in cloud systems. First, we will give a formal definition of the problem. Then, we give the overview of ADAMAS. We illustrate the core design of ADAMAS in detail in the remaining sub-sections. We use the term metrics in two different contexts. To avoid confusion, we use monitoring metrics to refer to the collected time series that quantifies the health status of cloud systems, *e.g.*, CPU usage. While evaluation metrics refer to the indicators measuring the quality of anomaly detection models, *i.e.*, the F1 score.

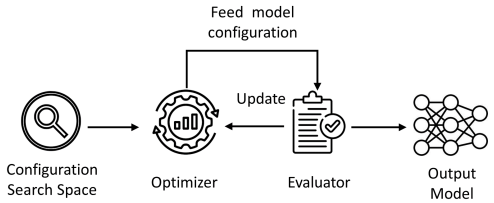


Fig. 3. A General Overview of AutoML

A. Problem Formulation

The objective of our work is to detect performance anomalies in software systems, especially large-scale cloud systems with monitoring metrics. Specifically, a monitoring metric can be seen as a time series $X \in R^n = [x_1, x_2, ..., x_n]$, where n denotes the number of observations collected, *i.e.*, the length of a time series. Typically, a sliding window of historical values $X_t^l = [x_{t-l+1}, x_{t-l+2}, ..., x_t]$ is used for modeling current observation, where l is the length of the sliding window. Our goal is to determine whether or not the given observation X_t^l is anomalous, *i.e.*, whether there is an occurrence of performance issues at the observation.

An AutoML framework typically contains several models with parameters. Given the set of monitoring metrics anomaly detection model $M = \{m_1, m_2, ..., m_k\}$, the set of parameter search space $\theta_i = \{\theta_1^i, \theta_1^i, ..., \theta_{n_i}^i\}$, we call the combination of a model m with parameter θ as a configuration $c \in \mathcal{C}$. With the objective function $\mathcal{L}(m, \theta, X)$ that evaluates the performance of configurations, the goal of AutoML is to find the optimal configuration c^* as follows:

$$c^* = \arg\max_{c \in \mathcal{C}} \mathcal{L}(c; X) \tag{1}$$

Normalization is performed on each monitoring metric to unify the range of them and improve the robustness of our method. We normalize the monitoring metrics with the Min-max normalization first.

B. Overview

The overall architecture of ADAMAS is shown in Fig. 4, which consists of two phases, namely, *label-free configuration search* and *feedback-based adaptive learning*. In the configuration search phase, we apply Bayesian Optimization, a widely-used AutoML technique, to iteratively search for new configurations and update the belief over the search space with the evaluation performance of the configuration. Bayesian Optimization typically consists of three main components: surrogate model, objective function, and acquisition function [43]. In our framework, the surrogate model that represents the

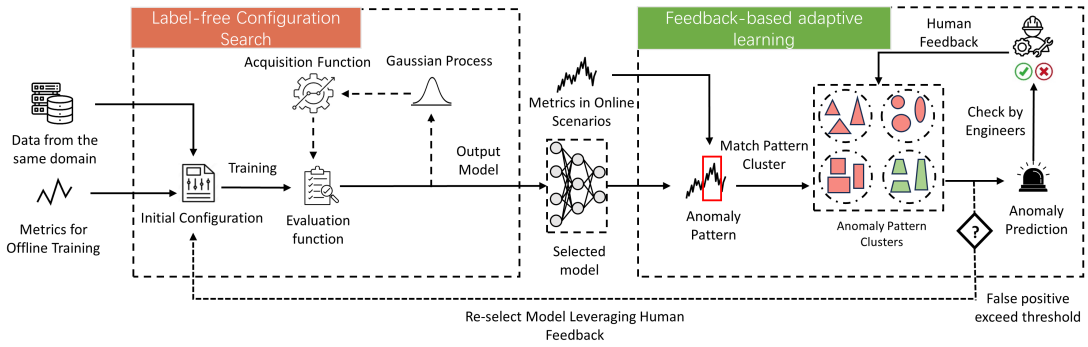


Fig. 4. An Overview of Proposed Framework ADAMAS

estimated belief of performance over the observed configurations is the most widely used Gaussian Process (GP) [44]. The objective function is utilized to evaluate the performance of configurations. In cloud systems, performance anomaly detection based on monitoring metrics is usually unsupervised due to the lack of high-quality labels. Thus, an objective function that approximates the performance without labels is needed. To this end, we propose the NFMK objective function, which is an excellent approximation of the F1 score compared with other MSE-based objective functions. The acquisition function is used to search for the next configuration to observe with the exploration-exploitation trade-off. In our case, we deploy the commonly used max-value entropy Search, an acquisition function with a fast convergence rate. The initialized configuration can be either from the optimal configurations of the same service, which serves as a warm start of BO-based search, or random configurations.

In the Feedback-based Adaptive Learning phase, we leverage the output model from the search phase to detect anomalies in a scenario where monitoring metrics arrive in streams, *i.e.*, one instance at a time. To fill in the gap that many predicted anomalies are not true anomalies and reduce the number of false positives, we incorporate expert knowledge with human feedback from on-site engineers when an anomaly happens. However, the human effort will be unaffordable if each anomaly alert is checked manually; we try to solve this with a streaming metrics cluster method MSC. In this way, the historical feedback from a similar anomaly pattern can be leveraged, which prevents additional labeling by engineers. Moreover, when the false positive exceeds a threshold, a retraining and configuration search will be triggered to make our framework adaptive to the system change.

C. Label-free Configuration Search

Generally, an AutoML framework trains different models with different parameters to search for the best configuration. Before the search process, the framework randomly selects the initial models and hyperparameters as the cold start due to the lack of prior knowledge about the monitoring metrics. If there exist explored optimal configurations from metrics in the same service, these configurations can serve as a warm start [45] as anomaly patterns in the monitoring metrics of the same service are similar. After obtaining the configurations, the objective function will evaluate the performance of the current configuration, and the surrogate model (GP in our case) updates the belief over the search space. We design an objective function, Noise-Free Mean squared error with Kurtosis (NFMK), that serves as an excellent approximation to evaluation functions like the F1 score without labels. This is because, in the case of metrics anomaly detection, performance anomalies are rare, and the labels are hard to obtain, which prohibits the exact evaluation of labels like the F1 score.

1) Objective Function: The idea of approximating an evaluation function with ground truth is based on a basic property of anomaly. If a monitoring metric segment cannot be reconstructed well by the model, it is more likely to be an anomaly [46]. Specifically, by using Mean Squared Error (MSE), we can evaluate the reconstruction ability of a model. However, collecting anomaly-free metric data is challenging and requires tedious manual efforts. So, the training data often contains noises, which manifest large errors in the MSE between the original and reconstructed monitoring metrics. To ensure the model can differentiate potential anomalies from normal patterns, we use the difference in kurtosis before and after noise removal. This helps measure the model's sensitivity to noise and its ability to discern outliers. Based on this observation, we try to capture the noise (*e.g.*, some mild performance anomaly ignored by engineers [4]) and calculate the MSE of the noise-free part. To make the noise part that potentially represents performance anomalies more distinguishable, we computed the difference in kurtosis before and after removing the noise. Specifically, as an effective estimator that measures the distance between the predicted and the raw monitoring metrics, MSE can be written as follows:

$$MSE(X, \hat{X}) = \sum_{i=1}^{n} \frac{\|X_i - \hat{X}_i\|^2}{n} \tag{2}$$

where X and $\hat{X}$ are raw and reconstructed monitoring metrics, respectively, and the length of the metric data is n, X_i and $\hat{X}_i$ are the raw and reconstructed sliding windows at the i-th point. Given an anomaly ratio r estimated by engineers according to historical observation, a threshold α is the r quantile of reconstruction error in each point. The points that exceed the threshold α are considered noise, while others are normal. The noise indicator A_t can be written as follows:

$$A_t = \begin{cases} 1, & |X_t - \hat{X}_t| \geq \alpha, \\ 0, & |X_t - \hat{X}_t| < \alpha. \end{cases} \tag{3}$$

Typically, a good anomaly detection method can fit the pattern that represents the normal behavior of monitoring metrics. Suppose the noise-free part of monitoring metric X is represented as X_N ($X_N = \{X_t | A_t = 1\}$), then the MSE term without noise data is $MSE(X_N, \hat{X_N})$. When a model has a strong ability to differentiate anomalies from normal patterns, the objective function should change significantly after noise removal. This is based on the assumption that a higher kurtosis corresponds to greater fluctuation, indicating the presence of anomalies. Thus, if a model is proficient in anomaly detection and the original kurtosis is large, there will be a dramatic drop in kurtosis after the noise is removed. In contrast, if the data has a small kurtosis, this suggests fewer outliers. In such cases, both the kurtosis before and after noise removal would be small, and the MSE term dominates the objective function. This encourages the model to fit the monitoring metric well. The formulation of kurtosis is as follows:

$$Kurt(X) = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^4}{(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2)^2} \tag{4}$$

Note that we take the reciprocal of the kurtosis part to make it unified with the MSE part. This way, we derive NFMK as our objective function and minimize it to search for the optimal configuration. The NFMK function in our framework is:

$$\begin{aligned} NFMK(X, \hat{X}) =& MSE(X_N, \hat{X_N}) + (Kurt(X - \hat{X}) \\ & - Kurt(X_N - \hat{X_N}))^{-1} \end{aligned} \tag{5}$$

2) Acquisition Function: The acquisition function is the utility function that guides the selection of the next configuration that should be explored to reach the optimum of the objective function [47]. Among many acquisition functions, entropy search-based acquisition functions [48], [49], [50] motivated by information theory are proven to achieve a fast convergence rate that dramatically improves the efficiency of the adaptive learning process [51]. In our framework, we apply the Max-value Entropy Search (MES) [52] as the acquisition function. MES uses the information about the maximum value of objective function $y^* = \mathcal{L}(c^*, X)$ at optimal configuration c^*, where the gain in mutual information between the maximum y^* and the next configuration can be approximated by evaluating the entropy of the predictive distribution as follows:

$$\begin{aligned} MES(c) &= H(p(y|D_t, c)) - \mathbb{E}(H(p(y|D_t, c, y^*))) \\ &\approx \frac{1}{K}\sum_{y^* \in Y^*} \frac{\gamma_{y^*}(c)\phi(\gamma_{y^*}(c))}{2\Phi(\gamma_{y^*}(c))} - \log\Phi(\gamma_{y^*}(c)) \end{aligned} \tag{6}$$

$$\gamma_{y^*}(c) = \frac{y^* - \mu_t(c)}{\sigma_t(c)} \tag{7}$$

where D_t represents the observations until the *i*-th iteration, H denotes the entropy, ϕ is the probability density function and Φ is the cumulative density function of normal distribution. In the first term of Equation 6, the mean and variance of $p(y|D_t, c)$ are $\mu_t(c)$ and $\sigma_t(c)$. The expectation of the second term in Equation 6 is approximated using Monte Carlo methods through sampling a set of K function maxima Y^*, where the sampling is from an approximation via a Gumbel distribution.

D. Feedback-based Adaptive Learning

After the configuration search phase, an optimal model will be applied to detect performance anomalies in online scenarios where monitoring metrics arrive in streams. However, there are inevitably many false positives as some anomaly patterns on monitoring metrics are not considered performance anomalies. It is difficult for the offline model to discriminate these false positives from true performance anomalies without domain knowledge from system experts. To this end, we propose to incorporate the knowledge of experts through human-in-the-loop to facilitate the anomaly detection process. It has been a common practice in cloud systems that on-call engineers manually verify every reported suspicious anomaly to mitigate the problem [9], [53]. Though the precision of the performance anomaly detection model can be tremendously improved with human feedback, the human effort devoted to labeling is non-negligible, especially in large-scale cloud systems that collect thousands of monitoring metrics on the fly. We observe that similar types of performance anomalies tend to trigger similar patterns on the monitoring metric, similar to [35], [7]. A straightforward idea is clustering similar anomaly patterns into one cluster and leveraging the feedback on historically similar patterns. In this way, the engineers only need to label the anomaly patterns in a cluster once.

Since the monitoring metrics are generated in a streaming manner, overwhelming data can exceed storage capacities, making clustering methods that require full data unsuitable [7]. Thus, we have no access to all metric patterns, unlike offline clustering approaches. We propose a streaming metrics clustering method, Metric Stream Clustering (MSC), that continuously clusters all incoming anomalous metric segments. Particularly, MSC is presented in Algorithm 1. First of all, noise in the NFMK term containing potential performance anomalies can be utilized to construct initialized clusters, which is denoted as X_0 for ease of presentation. We apply DBSCAN [54], a widely-used density-based clustering algorithm [55] to construct the initialized clusters. The mean vectors, cluster sizes, and radii are denoted as μ_0, S_0, R_0. Our core idea of metric stream clustering is that given a new metric segment X_t, we determine whether it can be attributed to an existing known anomaly cluster. The cluster will include X_t as an element and then update its parameters. Otherwise, a new anomaly cluster containing X_t itself will be created, an unseen metric pattern that on-site engineers should label. Particularly, we search for the index of the closest cluster idx_t and check whether the distance between the mean vector of the cluster and the metric pattern is smaller than the radius of the cluster. If yes, X_t will be considered a member of cluster idx_t. Otherwise, a new cluster containing X_t with a small radius δ represents the unseen pattern. When a new anomaly pattern is absorbed, the cluster's attributes should be updated. The mean vector and cluster size can be updated straightforwardly through the formulation in lines 11-12. However, the radius cannot be directly calculated as only the attributes of clusters are retained, which is common practice in online learning.

Algorithm 1 Metric Stream Clustering

Input: X_0, X_t, μ_t, S_t and R_t // Metrics and parameters of clusters

Output: μ_{t+1}, S_{t+1} and R_{t+1} // Updated parameters

1: **if** t=0 **then**
2: // Initialization Phase
3: $\mu_0, S_0, R_0 \leftarrow$DBSCAN$(X_0)$
4: **else**
5: // Continuous Clustering Phase
6: $\mu_{t+1}, S_{t+1}, R_{t+1} \leftarrow \mu_t, S_t, R_t$
7: $D_t \leftarrow$ PairWiseDistance(X_t, μ_t)
8: $idx_t \leftarrow$ MinIndex(D_t)
9: **if** $D_t[idx_t] < R_t[idx_t]$ **then**
10: // Add X_t to the nearest cluster and update parameters
11: $\mu_{t+1}[idx_t] \leftarrow (\mu[idx_t] \times S_t[idx_t] + X_t)/(S_t[idx_t] + 1)$
12: $S_{t+1}[idx_t] \leftarrow S_t[idx_t] + 1$
13: $R_{t+1}[idx_t] \leftarrow |\mu_{t+1}[idx_t] - \mu_t[idx_t]|/2 + R_t[idx_t]$
14: **else**
15: // A new cluster is created
16: $\mu_{t+1} \leftarrow$ Append μ_t with X_t
17: $S_{t+1} \leftarrow$ Append S_t with 1
18: $R_{t+1} \leftarrow$ Append R_t with δ // A small radius will be assigned to new cluster
19: **end if**
20: **end if**
21: **return** μ_{t+1}, S_{t+1} and R_{t+1}

We estimate it by analyzing the best and worst case of the radius update. On the one hand, the best case can be trivially derived in that the radius remained unchanged at $R_t[idx_t]$. On the other hand, the worst case is shown in Fig. 5, where the new radius reaches its maximum value when X_t lies on the opposite side of the cluster center μ_t with respect to the furthest point that yields the radius R_t, resulting in the largest radius update. We update the radius with the mean of the best and worst case, shown in line 13.

With the proposed clustering method MSC, we can seamlessly integrate valuable feedback from on-site engineers to ADAMAS. Specifically, when the offline model identifies an unseen suspicious anomaly pattern, it issues a query to cloud experts for confirmation. It should be noted that the number of suspicious anomalies is far less than the number of all metric streams. Moreover, the expert is only required to label once for an anomalous cluster, where all the patterns are considered anomalous if there is an element that is labeled as anomalous in the cluster. This further drastically reduces the required human effort. Consequently, the number of queries to experts is considerably lower compared to the total amount of metrics data, making our solution highly feasible and scalable. A problem with our strategy of incorporating human-in-the-loop is that though overwhelming false positives can be alleviated, the offline model remains unchanged and can not adapt to the evolving metric patterns. We try to solve this problem by setting an engineer-specified threshold that

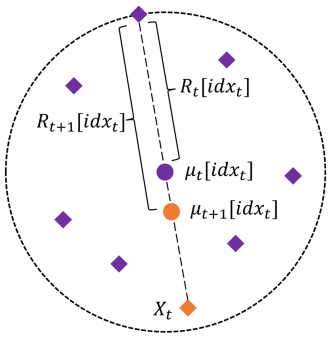


Fig. 5. The Worst Case of Cluster Radius Update

when the false positive clusters exceed it, a model retraining will be triggered. Only when many accumulated false positives cannot be properly differentiated by offline models should we make our model adaptive to these patterns through retraining; otherwise, the offline model works well toward a satisfactory performance. In this way, the computation cost that the regular retraining adopted by most frameworks consumes is significantly reduced. Specifically, we can take the incorporated domain knowledge to guide the configuration search process through the following objective function:

$$NFMK(X', \hat{X}') = MSE(X'_N, \hat{X'_N}) + (Kurt(X' - \hat{X}') - Kurt(X'_N - \hat{X'_N}))^{-1} \tag{8}$$

where X' and $\hat{X}'$ represent the raw and reconstructed online monitoring metric segments. $\hat{X'_N}$ represents the normal patterns not identified as anomalies and the false positives, which is a bit different from Equation 5. The new model is updated through training on the X' and selecting based on the objective function. It should be noted that X' represents the most recent metric segments as past metric segments are discarded due to the overwhelming volume of metrics.

IV. Experiments

In this section, we evaluate the effectiveness of ADAMAS using both a publicly available dataset and real-world monitoring metrics datasets collected from large-scale cloud systems in $\mathcal{X}$, a world-leading cloud company. In particular, we aim to answer the following research questions (RQs).

- RQ1: What is the effectiveness of ADAMAS compared with other baselines?
- RQ2: What is the effectiveness of each component in ADAMAS?
- RQ3: What is the sensitivity of ADAMAS to each hyper-parameter?

A. Experiment Settings

1) Datasets: To evaluate the effectiveness of ADAMAS, we conduct experiments on a public dataset and two industrial datasets collected from large-scale cloud systems in Company $\mathcal{X}$, which confirm its practical significance. The statistics of the datasets are summarized in Table I.

Public dataset. The AIOps18 dataset was released by an international AIOps competition in 2018, composed of multiple

TABLE I
STATISTICS OF ALL DATASETS

Dataset	#Metrics	#Points	#Ratio
AIOps18	29	5922913	2.26%
Industry A	506	5100480	4.67%
Industry B	535	5392800	3.46%

monitoring metrics collected from the web services of large-scale IT companies. Particularly, the dataset contains service metrics and machine metrics. The service metrics record the performance of the services, *e.g.*, response time and traffic, while the machine metrics reflect the health state of physical machines, including CPU usage and network throughput.

Industrial dataset. To fully evaluate the effectiveness in real-world production scenarios, we collect monitoring metrics from various services (*e.g.*, VPC, ELB, and RDS) in two availability zones (AZs) of a global cloud service provider $\mathcal{X}$. These metrics, including the service CPU usage, NIC throughput, received packets of load balancers, *etc.*, closely monitor the health status of services. For each of these metrics, we collect one week of data with a sampling interval of one minute. We rely on the corresponding issue reports of performance anomalies, which contain the start and end times of problems identified by on-site engineers or customers of an online service, to label the data.

2) Implementation: We implement four widely used metric performance anomaly detection methods by cloud system operators in our ADAMAS framework. Table II shows the details of the algorithms and hyperparameters. We run all the experiments on a Linux server with Intel Xeon Gold 6140 CPU @ 2.30GHZ and Tesla V100 PCIe GPU. The proposed model is implemented under the PyTorch and BoTorch [56] framework and runs on the GPU. The value of γ corresponds to the NFMK function and is set to 0.98, and the value of δ in MSC is 0.01. Due to the policy of company $\mathcal{X}$, we will release our data upon acceptance. Both the artifacts and data are available on https://github.com/WenweiGu/ADAMAS.

3) Evaluation Metrics: The anomaly detection problem is modeled as a binary classification problem, so the widely used binary classification measurements can be applied to evaluate the models. We employ Precision: $PC = \frac{TP}{TP+FP}$, Recall: $RC = \frac{TP}{TP+FN}$, F1 score: $F1 = 2 \cdot \frac{PC \cdot RC}{PC+RC}$. Specifically, TP is the number of abnormal samples the model correctly discovered; FP is the number of normal samples incorrectly classified as anomalies; FN is the number of anomalous samples that failed to be detected by the model. F1 score is the harmonic mean of the precision and recall, which symmetrically represents both precision and recall in a metric.

In real-world applications, anomalies will last for a while, leading to consecutive anomalies in the monitoring metrics. Therefore, it is acceptable for the model to trigger an alert for any point in a contiguous anomaly segment if the delay is within the acceptable range. Thus, we adopt the widely used evaluation strategy following [11], [46], [57], [58], [24]. In practice, different monitoring metrics have varying sampling frequencies, ranging from milliseconds (*e.g.*, latency) to minutes (*e.g.*, average load). Hence, to handle metrics with different sampling intervals, we use the number of timestamps rather than a fixed time interval. Particularly, the range is set to 10 timestamps based on the experiences of engineers.

TABLE II
IMPLEMENTED ALGORITHMS AND THEIR HYPERPARAMETERS

Algorithm	Parameters	Range
Autoencoder	encoding dimension	10∼50
	hidden dimension	100∼200
	decoding dimension	100∼200
VAE	hidden dimension	20∼200
	latent dimension	10∼50
LSTM	hidden dimension	50∼200
Transformer	hidden dimension	20∼500
	head number	1∼8
	layer number	1∼6
Common	window length	100∼300
	batch size	128∼2048
	learning rate	$1^{-5} \sim 1^{-2}$

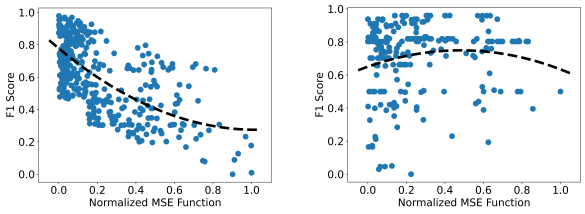
B. RQ1: The Effectiveness of ADAMAS

To study the effectiveness of ADAMAS, we compare its performance with various state-of-the-art baselines on a public dataset and two industrial datasets collected from industry. These baselines include (1) Traditional machine learning-based approaches, isolation forest (IForest) [59] and ADSketch [7]; (2) Deep learning-based approaches, DAGMM [60], VAE [11], LSTM [12], [61], SCRNN [20] and Maat [13]; (3) AutoML-based approaches, Hyperband [39], AutoAD [23] and AutoKAD [24]. For a fair comparison, the models and the hyperparameter space of the Hyperband, AutoAD, and AutoKAD are the same as ADAMAS.

The results are shown in Table III, where the best F1 scores are marked with boldface. We can see the average F1 score of ADAMAS outperforms all baseline methods in three datasets. We observe that the improvement of effectiveness achieved by ADAMAS is more significant in two industrial datasets, especially on precision. This is because modern large-scale cloud systems widely employ microservices architectures [62] and possess the ability of fault tolerance and self-healing [63], and some anomalies manifested in monitoring metrics can be mitigated without intervention, which is not considered as performance anomalies. Based on a random sampling of 100 metrics from the datasets, software reliability engineers manually analyzed these samples and determined that 28.7% of the anomalies were data anomalies but not business anomalies. These data anomalies are non-negligible due to their potential to generate a significant number of false positives, which may distract the SREs. Through seamlessly incorporating human feedback, ADAMAS drastically reduces the false positives that waste the human effort of engineers. We also notice that though ADAMAS achieves near-perfect precision in both industrial datasets, there exists a few false positives in the AIOps18 dataset. The reason is that we apply MSC in order to reduce the effort of human labeling, which assigns all the

TABLE III
EXPERIMENTAL RESULTS OF DIFFERENT ANOMALY DETECTION METHODS

Methods	AIOps18			Dataset A			Dataset B		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
IForest	0.581	0.467	0.517	0.483	0.787	0.541	0.445	0.802	0.515
DAGMM	0.530	0.608	0.535	0.521	0.777	0.535	0.376	0.771	0.440
VAE	0.592	0.483	0.543	0.504	0.810	0.566	0.418	0.779	0.465
LSTM	0.598	0.706	0.530	0.621	0.730	0.638	0.534	0.726	0.553
ADSketch	0.744	0.670	0.677	0.691	0.732	0.658	0.583	0.745	0.618
Maat	0.753	0.687	0.692	0.595	0.818	0.656	0.568	0.856	0.641
SRCNN	0.741	0.656	0.674	0.619	0.707	0.640	0.597	0.756	0.636
Hyperband	0.836	0.647	0.732	0.640	0.743	0.673	0.615	0.764	0.683
AutoAD	0.798	0.665	0.710	0.611	0.784	0.662	0.539	0.848	0.614
AutoKAD	0.861	0.694	0.769	0.675	0.798	0.685	0.662	0.846	0.692
ADAMAS	**0.975**	**0.763**	**0.848**	**0.997**	**0.832**	**0.897**	**0.998**	**0.878**	**0.929**
Effect size	4.472	2.237	3.091	12.034	0.883	8.316	13.170	1.395	9.292



(a) Correlation of NFMK (b) Correlation of MSE

Fig. 6. The correlation between two evaluation functions and F1 score

metric patterns in a cluster with the same label. The symptoms in the two industrial datasets are similar for the same types of performance anomalies. In contrast, the AIOps18 dataset may contain different types of performance issues that demonstrate the same patterns, resulting in imperfect precision. However, we believe that ADAMAS will not introduce much burden on the cloud operators as only 2.5% of the prediction results are false positives. Furthermore, we find that the recall on two industrial datasets is consistently higher compared with the AIOps18 dataset. We attribute this phenomenon to the good monitoring mechanism established in Company $\mathcal{X}$ that makes most of the performance anomalies perceptible through analysis of monitoring metrics. We also calculated Cohen's d to measure the effect size of the differences between our approach and baselines.

In terms of baselines, we observe that AutoML-based baselines typically achieve better results compared with others, consistent with recent studies [23], [24]. This indicates that in cloud systems, there is no golden algorithm that performs consistently best on all data from various services, and we should employ AutoML to achieve better performance rather than rely on a single algorithm. Among the machine learning-based and deep learning-based baselines, ADSketch and Maat perform best across three datasets with average F1 scores of 0.647 and 0.663. Specifically, ADSketch detects performance anomalies through pattern sketching, and Maat is based on the denoising diffusion model that resists noisy data during the training phase, both achieving relatively good performance. However, all these baselines purely mine the abnormal behaviors from the monitoring metrics without incorporating the crucial domain knowledge from on-site engineers, resulting in suboptimal results.

C. RQ2: The Effectiveness of Each Components of ADAMAS

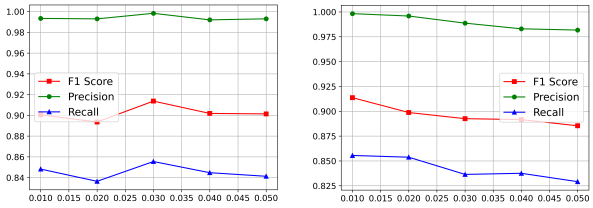
In this research question, we conduct an ablation study on ADAMAS to show the effectiveness of its components. In particular, we derive seven baseline models based on removing the feedback from MSC, replacing the NFMK function with other widely used proxy functions from the AutoML community, and replacing it with random feedback. We further visualize the correlation between NFMK and F1 score to demonstrate the contribution of our design.

- **ADAMAS-OFF** This baseline removes the human feedback provided by on-site engineers and the retraining of ADAMAS. We merely utilize the model output by AutoML in the configuration search phase to detect the anomalies.
- **ADAMAS-Jacob, Snip, Synflow** These three variants replace the NFMK function with three widely used proxy functions, namely, Jacob Covariant [64], Snip [65] and Synflow [66]. Specifically, the original loss function value required by Snip is MSE loss in our context.
- **ADAMAS-F05, F15, F30** These three baselines randomly sample 5%, 15%, and 30% of the whole testing metric. Feedback on these sampled metric segments is utilized.

Table IV shows the experimental results of ADAMAS and its variants. On one hand, we can observe performance drops when replacing our NFMK function with other proxy functions. Among these three proxies, Jacob Covariant performs the best. The assumption behind Jacob Covariant is that a lower correlation indicates a better network, which works well when the input data show various anomaly patterns. However, it may fall short when the input batch contains similar anomaly features. Another proxy, Snip, requires the original loss function, which in our case is MSE loss. This proxy metric approximates the loss change when a certain parameter is removed and encourages searching for the model with the lowest loss. In our scenario, where training data may contain noise, models with low MSE can still be ineffective in differentiating anomaly patterns. As a result, we can observe a significant drop on the recall. Similarly, the Synflow function computes the loss as the product of all parameters without requiring the original loss. However, it does not inherently

TABLE IV
EXPERIMENTAL RESULTS OF THE ABLATION STUDY

Methods	AIOps18			Dataset A			Dataset B		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
ADAMAS-OFF	0.843	0.741	0.778	0.728	0.793	0.730	0.715	0.845	0.751
ADAMAS-Jacob	0.956	0.715	0.813	0.972	0.753	0.857	0.976	0.813	0.870
ADAMAS-Snip	0.941	0.696	0.807	0.963	0.722	0.835	0.967	0.769	0.851
ADAMAS-Synflow	0.923	0.704	0.785	0.952	0.736	0.829	0.956	0.794	0.862
ADAMAS-F05	0.876	0.757	0.791	0.897	0.824	0.841	0.881	0.859	0.858
ADAMAS-F15	0.904	0.760	0.815	0.921	0.828	0.858	0.921	0.854	0.877
ADAMAS-F30	0.935	0.752	0.823	0.946	0.827	0.865	0.954	0.857	0.883
ADAMAS	**0.975**	**0.763**	**0.848**	**0.997**	**0.832**	**0.897**	**0.998**	**0.878**	**0.929**



(a) Parameter Sensitivity of γ (b) Parameter Sensitivity of δ

Fig. 7. Parameter Sensitivity of ADAMAS

consider the specific pattern of input data. Generally speaking, these proxies are not specially designed for anomaly detection, and they neglect the unique characteristics of the task, thus leading to unsatisfactory performance.

On the other hand, human feedback through MSC helps to improve the effectiveness of ADAMAS as it performs the best compared with the variants without feedback or with random feedback. We observe that in the variant without human feedback, not only does the precision drop markedly, but also the recall decreases a bit. We attribute this to the design of retraining during the adaptive learning phase because the human-labeled evolving anomaly patterns are utilized to update our searched model, enhancing its ability to capture unseen anomalies. Compared with the results in Table III, we find that even without human feedback, our method still outperforms all the baselines, which shows the effectiveness of the design of our configuration search phase. It should be noted that, according to our experiments, the feedback ratio required by ADAMAS for three datasets is 5.9%, 6.7%, and 4.3%, respectively. Compared to baselines with random feedback, ADAMAS can achieve higher performance with less human feedback than utilizing even 30% feedback of the whole metric, which demonstrates the effectiveness of our design of the MSC algorithm, where anomaly patterns in the same cluster will only be labeled by engineers once.

Our proposed NFMK function tries to fit the noise-free metric patterns and prevent overfitting through a regularization term. To demonstrate this, we visualize the correlation of the NFMK function versus the F1 score compared with MSE, shown in Fig. 6. For the MSE function, we can observe a trend of decline in the correlation when the MSE is small enough. This indicates the model overfits the metrics, *i.e.*, it fits both the normal and anomalous pattern well, resulting in a drop in performance. In fact, an effective performance anomaly detection method generally reconstructs normal patterns well while differentiating anomalies with high reconstruction errors, while our NFMK function takes this into account. According to our assumption, the smaller the NFMK function is, the better the performance is. We can observe an obvious negative correlation between the NFMK function and the F1 score, which demonstrates the effectiveness of the NFMK function as an estimator of the F1 score.

D. RQ3: Parameter Sensitivity of ADAMAS

In ADAMAS, there are only two parameters to tune, namely, the threshold γ that determines the ratio of noise in the NFMK function and δ that represents the radius of a new cluster in MSC. We hereon evaluate the sensitivity of ADAMAS to these two parameters on two industrial datasets. We change the value of γ and δ while keeping all other parameters unchanged in our experiments to guarantee fairness. Specifically, we choose the value of γ in the range from 0.95 to 0.99, while the value of δ is selected, ranging from 0.01 to 0.05. Fig. 7 presents the experimental results, where we observe that ADAMAS is relatively stable under different settings. Therefore, ADAMAS exhibits robustness to these two parameters, eliminating the need for meticulous parameter tuning. This aligns well with our objective of sparing non-ML expert engineers' effort in parameter tuning. It should be noted that there is a consistent decline in performance with the increase of the δ. This can be attributed to the fact that when the radius of a new pattern becomes excessively large, it results in numerous new anomalous patterns merging into a single cluster. However, these patterns may not represent the same type of performance anomaly, thus reducing the accuracy.

E. Case Study

Since October 2023, ADAMAS has been effectively incorporated into the cloud service systems of Company $\mathcal{X}$, a leading company that provides cloud service to millions of users worldwide. ADAMAS can be seamlessly integrated into the existing cloud monitoring data analytics pipeline, such as Apache Kafka [7], InfluxDB [67], Datadog [68]. We focus on the deployment of ADAMAS in the Object Storage Service (OBS), which is a scalable solution offering cloud storage through a RESTful web services interface. Fig. 8 reflects the feedback required by ADAMAS over a 25-week deployment period, according to the weekly troubleshooting reports produced by SREs. It can be observed that the required feedback experiences a significant decrease after the initial week, thanks

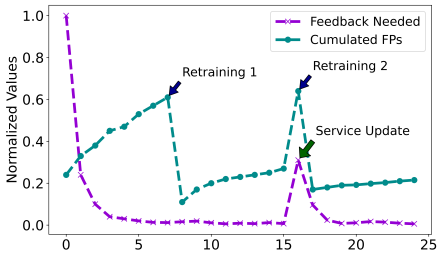


Fig. 8. Industrial deployment in OBS service

to our design of MSC. Following a service update in the sixteenth week, new anomaly patterns begin to emerge, resulting in an increase in required feedback. However, within a week, the required feedback decreases to a relatively low level again, effectively reducing the workload for SREs. Furthermore, we notice two drops in the accumulated false positives. This occurs due to our design of retraining when the accumulated false positives exceed a predetermined threshold (0.6 in our normalized values), which makes ADAMAS adaptive to the evolving anomaly patterns of the cloud system.

Fig. 9 presents a case study illustrating how ADAMAS harnesses human feedback to minimize false positives. This example focuses on a metric monitoring the data transfer rate within the OBS provided by Company $\mathcal{X}$. A transient spike in the metric may be attributed to expected events like an influx in user requests or planned data migration. These events are not typically categorized as performance anomalies. On the contrary, a sudden dip to near zero in the data transfer rate could suggest network congestion or service malfunction, necessitating prompt troubleshooting. In this case, the green areas highlight four false positives; nonetheless, only the first anomaly pattern will undergo manual inspection by engineers, as these four false positives share a similar pattern and would be clustered together. When encountering a true performance anomaly, as denoted by the red area, an alert will be triggered. Engineers can then confirm that the service is experiencing performance degradation. When this anomaly pattern recurs again, engineers can recognize performance anomalies even without the effort of manually checking. Thus, we posit that our design of ADAMAS can reduce false positives and facilitate maintaining cloud systems.

V. Discussion

In this section, we discuss the reduction of human efforts with ADAMAS, its overhead, and some potential threats to the validity.

A. The Reduction of Human Efforts

We give an estimation of how much time ADAMAS can reduce the amount of human effort. The feedback ratio required by ADAMAS for the three datasets is 5.9%, 6.7%, and 4.3%. Based on interviews with software engineers who routinely maintain the systems, about 30% of data should be reviewed manually to maintain accurate anomaly detection without ADAMAS. We estimate that without ADAMAS, SREs should manually check approximately six times the cases. According to the interview with SREs, each SRE typically

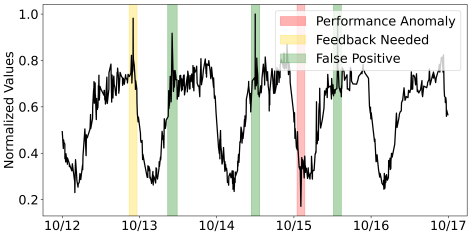


Fig. 9. Case Study in OBS service

checks around 100 anomalies per week and spends about 2-3 minutes per anomaly, amounting to approximately 4 hours per week in total. ADAMAS can potentially reduce this effort to around 40 minutes. Thus, ADAMAS can reduce more than 3 hours of manual effort for each SRE per week. We aim to gather more data to accurately assess the exact time savings, which will be the focus of our future work.

B. The overhead of ADAMAS

The execution time for ADAMAS mainly consists of three components: the computation time for searching optimal configuration (the most time-consuming one), the time spent by humans providing feedback, and the time required for clustering. According to our investigation, when we search for 20 iterations and train each model with 50 epochs, the average execution times on three datasets for ADAMAS and Hyperband are 937s and 354s, respectively. It should be noted that the efficiency can be enhanced when utilizing the computational power available in cloud service systems. Furthermore, since this phase is conducted offline, its impact on real-time operations is insignificant. Thus, the overhead is affordable for industrial deployment.

C. Threats to Validity

Internal threats. The implementation of baselines constitutes one of the internal threats to our study's validity. To address the threat of implementation, we directly utilized the open-sourced code released by the authors of the papers. As for our proposed approach, the source code undergoes rigorous peer code review by the authors and experienced engineers to minimize the risk of errors in our results. To mitigate the parameter setting threat, we fine-tuned the baseline methods utilizing a grid-search approach and derived the optimal results. To make our results reproducible, we have also made our code and partial data available.

External threats. The external threats to the validity of our study mainly lie in the generalizability of our experimental results. We conduct experiments on large-scale cloud systems within a prominent cloud service company. In addition to this, our approach is also evaluated on a publicly available dataset containing monitoring metrics from the web services of a large-scale IT company, further expanding the scope of our evaluation. Therefore, we believe the evaluation is representative and convincing, demonstrating that our framework ADAMAS could be applied to typical cloud systems.

VI. Related Work

Performance Anomaly Detection. Metrics anomaly detection approaches can be divided into machine learning-based and deep learning-based. A representative method of machine learning is ADSketch [7], an online performance anomaly detection approach based on pattern sketching that compares anomaly patterns with the historically identified patterns and updates the cluster synopsis with the evolving data. Another widely-used method is Isolation Forest (IForest) [59], which assumes that anomalous data can be more easily isolated through multiple binary trees. Among deep learning-based approaches, DAGMM [60] is a deep auto-encoding Gaussian mixture model without considering the temporal pattern in monitoring metrics. Another method is VAE [11], which detects anomalies by the reconstruction error from the normal hidden state of monitoring metrics. LSTM [61], [12] leverages the Long Short-Term Memory (LSTM) network for anomaly detection in monitoring metrics based on prediction errors. SRCNN [20] is a metric anomaly detection service deployed at Microsoft, widely used by teams including Bing, Office, and Azure. It is based on the combination of Spectral Residual (SR) and Convolutional Neural Network (CNN). Maat [13] is a conditional denoising diffusion-based model, which defines anomaly-indicating features and then distinguishes performance anomalies from normal metric segments.

AutoML-Based Anomaly Detection. In large-scale cloud systems, manual model selection and hyperparameters are time-consuming and error-prone; thus, engineers resort to utilizing the AutoML technique. One of the most popular solutions is HyperBand [39], which combines the randomized search procedure Successive Halving with the early-stopping mechanism. However, the random nature and the neglect of historical information from previously explored Hyperband configurations result in suboptimal results. ProxyBO [69] is another efficient AutoML-based framework that utilizes three proxy functions to accelerate neural architecture search. However, these proxies are not particularly designed for anomaly detection. Besides, several AutoML-based solutions specially designed for anomaly detection are proposed. Among them, AutoAD [23] is the first AutoML framework that utilizes an unsupervised measurement for model evaluation. However, AutoAD adopts MSE as the evaluation function, which cannot accurately approximate the performance. AutoKAD [24] is another AutoML framework that applies Bayesian Optimization [56], which iteratively searches for the optimal configurations with MSE-NF function. Both methods capture the relationship between the performance and the hyperparameter settings and search for the best performance configuration. However, they neglect the difference between true anomaly and predicted anomaly without considering the domain knowledge of cloud system experts. Thus, the overwhelming false positives will add burdens on cloud operators. Besides, existing AutoML-based anomaly detection frameworks are offline, which makes it difficult to apply them to online scenarios in which software changes and anomaly patterns evolve.

Online Stream Clustering. As the scale of cloud systems grows increasingly large, monitoring platforms are gathering an overwhelming amount of data on the fly. Thus, the historical metrics data required by offline clustering methods are unavailable as past data are discarded due to the limited storage. As such, online stream clustering has gained more popularity in large-scale cloud service systems, where only some synopsis of the historical metrics are retained [70]. Among these stream clustering methods, CluStream [71] is a partitioning-based method based on the concept of microclusters, which summarizes a set of instances from the metric stream. However, the number of clusters should be predefined, limiting its robustness [72]. Another representative method is E-Stream [73], a hierarchical method that uses a data structure named the α-bin histogram for saving summary statistics. However, this category of methods typically bears high complexity and is sensitive to outliers. MuDi-Stream [74] is a density-based method that falls into the same category as our stream clustering method, based on the combination of density-based clustering and grid-based outlier detection. However, it requires much more tuning of parameters. In a word, both CluStream and MuDi-Stream contradict our objective of minimizing the need for manual parameter tuning. The E-Stream method falls short of meeting the requirements of real-time analysis essential in online scenarios. Thus, these methods were not adopted in our scenario.

VII. Conclusion

In this work, we propose ADAMAS, an adaptive AutoML-based performance anomaly detection framework incorporated with domain knowledge. Specifically, ADAMAS consists of a configuration search stage and a feedback-based Adaptive Learning stage. In the first stage, a novel unsupervised evaluation function, NFMK, is proposed to guide the configuration search. In the second stage, we incorporate human feedback to differentiate true anomalies from all predicted anomalies. A streaming metrics clustering algorithm, MSC, is proposed to leverage the historical feedback from on-site engineers to decrease the human effort. Furthermore, when the number of mispredicted anomalies exceeds a threshold, retraining will be triggered to make our framework adaptive to evolving patterns due to rapid software updating. Extensive experiments on a public dataset and two industrial datasets show that ADAMAS achieves 0.891 F1-Score on anomaly detection, outperforming all the baselines. Furthermore, a case study demonstrates how ADAMAS is deployed into the cloud service system of Company $\mathcal{X}$.

VIII. Acknowledgments

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14206921 of the General Research Fund) and Fundamental Research Funds for the Central Universities, Sun Yat-sen University (No. 76250-31610005).

REFERENCES

[1] J. Gu, J. Wen, Z. Wang, P. Zhao, C. Luo, Y. Kang, Y. Zhou, L. Yang, J. Sun, Z. Xu *et al.*, "Efficient customer incident triage via linking with system incidents," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1296–1307.

[2] J. Gu, C. Luo, S. Qin, B. Qiao, Q. Lin, H. Zhang, Z. Li, Y. Dang, S. Cai, W. Wu *et al.*, "Efficient incident identification from multi-dimensional issue reports via meta-heuristic search," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 292–303.

[3] J. Liu, Z. Jiang, J. Gu, J. Huang, Z. Chen, C. Feng, Z. Yang, Y. Yang, and M. R. Lyu, "Prism: Revealing hidden functional clusters from massive instances in cloud systems," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 268–280.

[4] W. Gu, J. Liu, Z. Chen, J. Zhang, Y. Su, J. Gu, C. Feng, Z. Yang, Y. Yang, and M. R. Lyu, "Identifying performance issues in cloud service systems based on relational-temporal features," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2024.

[5] W. Gu, X. Sun, J. Liu, Y. Huo, Z. Chen, J. Zhang, J. Gu, Y. Yang, and M. R. Lyu, "Kpiroot: Efficient monitoring metric-based root cause localization in large-scale cloud systems," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2024, pp. 403–414.

[6] Z. Chen, J. Liu, Y. Su, H. Zhang, X. Wen, X. Ling, Y. Yang, and M. R. Lyu, "Graph-based incident aggregation for large-scale online service systems," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 430–442.

[7] Z. Chen, J. Liu, Y. Su, H. Zhang, X. Ling, Y. Yang, and M. R. Lyu, "Adaptive performance anomaly detection for online service systems via pattern sketching," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 61–72.

[8] R. Zhong, Y. Li, J. Kuang, W. Gu, Y. Huo, and M. R. Lyu, "Automated defects detection and fix in logging statement," *arXiv preprint arXiv:2408.03101*, 2024.

[9] Z. Chen, Y. Kang, L. Li, X. Zhang, H. Zhang, H. Xu, Y. Zhou, L. Yang, J. Sun, Z. Xu *et al.*, "Towards intelligent incident management: why we need it and how we make it," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1487–1497.

[10] L. Li, X. Zhang, S. He, Y. Kang, H. Zhang, M. Ma, Y. Dang, Z. Xu, S. Rajmohan, Q. Lin *et al.*, "Conan: Diagnosing batch failures for cloud systems," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2023, pp. 138–149.

[11] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 187–196.

[12] G. Zhao, S. Hassan, Y. Zou, D. Truong, and T. Corbin, "Predicting performance anomalies in software systems at run-time," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 3, pp. 1–33, 2021.

[13] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Maat: Performance metric anomaly anticipation for cloud services with conditional diffusion," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 116–128.

[14] X. Liu, J. Wen, Z. Chen, D. Li, J. Chen, Y. Liu, H. Wang, and X. Jin, "Faaslight: General application-level cold-start latency optimization for function-as-a-service in serverless computing," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 5, pp. 1–29, 2023.

[15] N. Zhao, H. Wang, Z. Li, X. Peng, G. Wang, Z. Pan, Y. Wu, Z. Feng, X. Wen, W. Zhang *et al.*, "An empirical investigation of practical log anomaly detection for online service systems," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1404–1415.

[16] J. Paparrizos, Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, and M. J. Franklin, "Tsb-uad: an end-to-end benchmark suite for univariate time-series anomaly detection," *Proceedings of the VLDB Endowment*, vol. 15, no. 8, pp. 1697–1711, 2022.

[17] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: a comprehensive evaluation," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, pp. 1779–1797, 2022.

[18] E. Sylligardos, P. Boniol, J. Paparrizos, P. Trahanias, and T. Palpanas, "Choose wisely: An extensive evaluation of model selection for anomaly detection in time series," *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 3418–3432, 2023.

[19] J. Liu, T. Yang, Z. Chen, Y. Su, C. Feng, Z. Yang, and M. R. Lyu, "Practical anomaly detection over multivariate monitoring metrics for online services," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 36–45.

[20] 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*, 2019, pp. 3009–3017.

[21] F. Liu, X. Zhou, J. Cao, Z. Wang, T. Wang, H. Wang, and Y. Zhang, "Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional lstm-cnn," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2626–2640, 2020.

[22] L. Wang, S. Chen, and Q. He, "Concept drift-based runtime reliability anomaly detection for edge services adaptation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[23] A. Putina, M. Bahri, F. Salutari, and M. Sozio, "Autoad: an automated framework for unsupervised anomaly detection." in *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2022, pp. 1–10.

[24] Z. Yu, C. Pei, S. Zhang, X. Wen, J. Li, G. Xie, and D. Pei, "Autokad: Empowering kpi anomaly detection with label-free deployment," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 13–23.

[25] L. K. Shar, A. Goknil, E. J. Husom, S. Sen, Y. N. Tun, and K. Kim, "Autoconf: Automated configuration of unsupervised learning systems using metamorphic testing and bayesian optimization," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 1326–1338.

[26] C. Wang, Z. Chen, and M. Zhou, "Automl from software engineering perspective: Landscapes and challenges," in *Proceedings of the 20th International Conference on Mining Software Repositories*. MSR, 2023.

[27] C. Duan, T. Jia, H. Cai, Y. Li, and G. Huang, "Afalog: A general augmentation framework for log-based anomaly detection with active learning," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 46–56.

[28] M. A. Siddiqui, A. Fern, T. G. Dietterich, R. Wright, A. Theriault, and D. W. Archer, "Feedback-guided anomaly discovery via online optimization," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2200–2209.

[29] C. Chai, L. Cao, G. Li, J. Li, Y. Luo, and S. Madden, "Human-in-the-loop outlier detection," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 19–33.

[30] M. N. Injadat, F. Salo, A. B. Nassif, A. Essex, and A. Shami, "Bayesian optimization with machine learning algorithms towards anomaly detection," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[31] S. Agarwal, S. Chakraborty, S. Garg, S. Bisht, C. Jain, A. Gonuguntla, and S. Saini, "Outage-watch: Early prediction of outages using extreme event regularizer," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 682–694.

[32] N. Zhao, P. Jin, L. Wang, X. Yang, R. Liu, W. Zhang, K. Sui, and D. Pei, "Automatically and adaptively identifying severe alerts for online service systems," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2420–2429.

[33] B. Panda, D. Srinivasan, H. Ke, K. Gupta, V. Khot, and H. S. Gunawi, "{IASO}: A {Fail-Slow} detection and mitigation framework for distributed storage services," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 47–62.

[34] R. Lu, E. Xu, Y. Zhang, F. Zhu, Z. Zhu, M. Wang, Z. Zhu, G. Xue, J. Shu, M. Li *et al.*, "Perseus: A {Fail-Slow} detection framework for cloud storage systems," in *21st USENIX Conference on File and Storage Technologies (FAST 23)*, 2023, pp. 49–64.

[35] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu *et al.*, "Diagnosing root causes of intermittent slow queries

in cloud databases," *Proceedings of the VLDB Endowment*, vol. 13, no. 8, pp. 1176–1189, 2020.

[36] Z. Guo, Y. Xu, Y.-F. Liu, S. Liu, H. J. Chao, Z.-L. Zhang, and Y. Xia, "Aggreflow: Achieving power efficiency, load balancing, and quality of service in data center networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 17–33, 2020.

[37] Z. Li, N. Zhao, M. Li, X. Lu, L. Wang, D. Chang, X. Nie, L. Cao, W. Zhang, K. Sui *et al.*, "Actionable and interpretable fault localization for recurring failures in online service systems," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 996–1008.

[38] M. Bahri, F. Salutari, A. Putina, and M. Sozio, "Automl: state of the art with a focus on anomaly detection, challenges, and research directions," *International Journal of Data Science and Analytics*, vol. 14, no. 2, pp. 113–126, 2022.

[39] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[40] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "{CherryPick}: Adaptively unearthing the best cloud configurations for big data analytics," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 469–482.

[41] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[42] Y. Zhao, R. Rossi, and L. Akoglu, "Automatic unsupervised outlier model selection," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4489–4502, 2021.

[43] J. Wilson, F. Hutter, and M. Deisenroth, "Maximizing acquisition functions for bayesian optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[44] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," *Advances in neural information processing systems*, vol. 31, 2018.

[45] V. Perrone, R. Jenatton, M. W. Seeger, and C. Archambeau, "Scalable hyperparameter transfer learning," *Advances in neural information processing systems*, vol. 31, 2018.

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

[47] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, "Recent advances in bayesian optimization," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–36, 2023.

[48] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization." *Journal of Machine Learning Research*, vol. 13, no. 6, 2012.

[49] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," *Advances in neural information processing systems*, vol. 27, 2014.

[50] M. W. Hoffman and Z. Ghahramani, "Output-space predictive entropy search for flexible global optimization," in *NIPS workshop on Bayesian Optimization*, 2015, pp. 1–5.

[51] S. Belakaria, A. Deshwal, and J. R. Doppa, "Max-value entropy search for multi-objective bayesian optimization," *Advances in neural information processing systems*, vol. 32, 2019.

[52] Z. Wang and S. Jegelka, "Max-value entropy search for efficient bayesian optimization," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3627–3635.

[53] Y. Wang, G. Li, Z. Wang, Y. Kang, Y. Zhou, H. Zhang, F. Gao, J. Sun, L. Yang, P. Lee *et al.*, "Fast outage analysis of large-scale production clouds with service correlation mining," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 885–896.

[54] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[55] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[56] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "Botorch: A framework for efficient monte-carlo bayesian optimization," *Advances in neural information processing systems*, vol. 33, pp. 21 524–21 538, 2020.

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

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

[59] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[60] 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 *International conference on learning representations*, 2018.

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

[62] G. Yu, P. Chen, and Z. Zheng, "Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach," *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 1100–1116, 2020.

[63] P. Garraghan, R. Yang, Z. Wen, A. Romanovsky, J. Xu, R. Buyya, and R. Ranjan, "Emergent failures: Rethinking cloud reliability at scale," *IEEE Cloud Computing*, vol. 5, no. 5, pp. 12–21, 2018.

[64] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *International conference on machine learning*. PMLR, 2021, pp. 7588–7598.

[65] N. Lee, T. Ajanthan, and P. Torr, "Snip: single-shot network pruning based on connection sensitivity," in *International Conference on Learning Representations*. Open Review, 2019.

[66] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in neural information processing systems*, vol. 33, pp. 6377–6389, 2020.

[67] A. Visheratin, A. Struckov, S. Yufa, A. Muratov, D. Nasonov, N. Butakov, Y. Kuznetsov, and M. May, "Peregreen–modular database for efficient storage of historical time series in cloud environments," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 589–601.

[68] Y. He, R. Guo, Y. Xing, X. Che, K. Sun, Z. Liu, K. Xu, and Q. Li, "Cross container attacks: The bewildered {eBPF} on clouds," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5971–5988.

[69] Y. Shen, Y. Li, J. Zheng, W. Zhang, P. Yao, J. Li, S. Yang, J. Liu, and B. Cui, "Proxybo: Accelerating neural architecture search via bayesian optimization with zero-cost proxies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, 2023, pp. 9792–9801.

[70] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. d. Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–31, 2013.

[71] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings 2003 VLDB conference*. Elsevier, 2003, pp. 81–92.

[72] A. Zubaroğlu and V. Atalay, "Data stream clustering: a review," *Artificial Intelligence Review*, vol. 54, no. 2, pp. 1201–1236, 2021.

[73] K. Udommanetanakit, T. Rakthanmanon, and K. Waiyamai, "E-stream: Evolution-based technique for stream clustering," in *Advanced Data Mining and Applications: Third International Conference, ADMA 2007 Harbin, China, August 6-8, 2007. Proceedings 3*. Springer, 2007, pp. 605–615.

[74] A. Amini, H. Saboohi, T. Herawan, and T. Y. Wah, "Mudi-stream: A multi-density clustering algorithm for evolving data stream," *Journal of Network and Computer Applications*, vol. 59, pp. 370–385, 2016.