MonitorAssistant: Simplifying Cloud Service Monitoring via Large Language Models

Zhaoyang Yu
Tsinghua University & BNRist
Beijing, China

Minghua Ma*
Microsoft
Redmond, USA

Chaoyun Zhang
Si Qin
Yu Kang
Microsoft
Beijing, China

Chetan Bansal
Saravan Rajmohan
Yingnong Dang
Microsoft
Redmond, USA

Changhua Pei
CNIC, CAS
Beijing, China

Dan Pei
Tsinghua University & BNRist
Beijing, China

Qingwei Lin
Dongmei Zhang
Microsoft
Beijing, China

ABSTRACT

In large-scale cloud service systems, monitoring metric data and conducting anomaly detection is an important way to maintain reliability and stability. However, great disparity exists between academic approaches and industrial practice to anomaly detection. Industry predominantly uses simple, efficient methods due to better interpretability and ease of implementation. In contrast, academically favor deep-learning methods, despite their advanced capabilities, face practical challenges in real-world applications. To address these challenges, this paper introduces MonitorAssistant, an end-to-end practical anomaly detection system via Large Language Models. MonitorAssistant automates model configuration recommendation achieving knowledge inheritance and alarm interpretation with guidance-oriented anomaly reports, facilitating a more intuitive engineer-system interaction through natural language. By deploying MonitorAssistant in Microsoft's cloud service system, we validate its efficacy and practicality, marking a significant advancement in the field of practical anomaly detection for large-scale cloud services.

CCS CONCEPTS

• **Software and its engineering**; • **Computing methodologies** → **Artificial intelligence**; **Machine learning**;

KEYWORDS

Cloud Service Monitoring, Software Reliability, Anomaly Detection, Large Language Models

ACM Reference Format:
Zhaoyang Yu, Minghua Ma, Chaoyun Zhang, Si Qin, Yu Kang, Chetan Bansal, Saravan Rajmohan, Yingnong Dang, Changhua Pei, Dan Pei, Qingwei Lin, Dongmei Zhang. 2024. MonitorAssistant: Simplifying Cloud Service Monitoring via Large Language Models. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24), July 15–19, 2024, Porto de Galinhas, Brazil.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3663529.3663826

*Minghua Ma is the corresponding author.

FSE Companion '24, July 15–19, 2024, Porto de Galinhas, Brazil

ACM ISBN 979-8-4007-0658-5/24/07
https://doi.org/10.1145/3663529.3663826

1 INTRODUCTION

In recent years, the large-scale cloud systems have diverse hardware and software components, including containers, virtual machines, switches, routers, *etc.* Ensuring the reliability of these systems, particularly in a vast cloud environment, is crucial as it directly affects user experience and overall service stability and reliability.

Monitoring metric data and conducting anomaly detection is an essential way to maintain service stability and reliability [7, 18, 20, 31, 32, 34, 39, 43]. In the real-world environment, monitoring systems often relies on relatively straightforward methods, most notably threshold rules [11, 36] to detect anomalies. These methods have proven to be efficient for monitoring systems, providing timely alerts to operators without requiring complex computations [26]. However, the academic sphere has seen the exploration of more advanced, deep-learning-based anomaly detection methods [46]. These methods have showcased impressive capabilities in theory and in laboratory settings though they usually consume more computation resources [18, 31, 39], but they do not become mainstream anomaly detection methods in industry.

Why does the real-world industry frequently resort to the traditional method, such as threshold rules and statistical method? The first reason is the practicality of implementing and maintaining these algorithms in a dynamic, ever-changing industrial environment. Consider the retraining requirements of machine learning models, which often need to be updated frequently to adapt to evolving system dynamics. This process can be time-consuming and requires a constant feed of labeled data, a resource that might not be readily available in many real-world settings [11, 16, 17, 36, 41].

Furthermore, the complexity of these models can present obstacles in comprehending and diagnosing their outputs. In many industrial situations, interpretability is crucial: operators need to understand why a particular anomaly was detected to decide on the best course of action. The "black box" nature of many machine learning methods can therefore present a barrier to their adoption in

these practical contexts. In addition, the complexity of implementing and managing these advanced models and algorithms often demands a higher level of expertise than is required for simpler methods. This can lead to higher costs, encompassing not only financial outlay but also time and human resources.

The interaction between engineers and anomaly detection models also influences the practicality. After deploying the model, engineers collect extensive feedback based on the model's detection outcomes. For traditional methods, such as fixed thresholds, engineers can easily adjust thresholds dynamically based on online feedback to further optimize the anomaly detection model. However, when dealing with deep-learning-based models, it becomes challenging for engineers to interact efficiently and specifically with the anomaly detection models.

To address the aforementioned challenges, we propose MonitorAssistant, an end-to-end practical anomaly detection system via Large Language Model (LLM), in particular GPT-4. MonitorAssistant is designed to streamline the deployment and effectiveness of both traditional and deep-learning-based models. Addressing the complexities in model configuration recommendation, MonitorAssistant automates this process, significantly reducing the burden on engineers. MonitorAssistant tackles the critical issue of alarms interpretation by summarizing the historical knowledge and supporting troubleshooting guide. Furthermore, MonitorAssistant introduces a LLM-engineer-in-the-loop workflow, enabling engineers to interact with anomaly detection models through natural language. This innovation simplifies feedback processes and model optimization, ensuring quick adaptation to service changes and reducing the need for frequent retraining. By automating key aspects of anomaly detection model deployment and enhancing engineer-model interaction, MonitorAssistant offers a practical, efficient, and user-friendly solution for anomaly detection in cloud service systems.

In summary, this paper makes the following contributions:

- We systematically reveal the distinct nature of practical anomaly detection in industry settings and provide a guidance to conduct practical anomaly detection in monitoring system.
- We propose MonitorAssistant, an end-to-end practical anomaly detection system based on LLM. MonitorAssistant automates the whole pipeline of practical anomaly detection, provide more guidance-oriented anomaly reports and a more user-friendly engineer-system interaction mechanism.
- We pilot MonitorAssistant into several Microsoft's cloud service systems, making service monitoring easier, and demonstrate that MonitorAssistant is a powerful practical monitoring system through the empirical study based on real-world cases.

2 BACKGROUND

2.1 Service Monitoring

2.1.1 Metric Data. Monitoring cloud services is a crucial way to understand the health status of services and ensure their stability and reliability. In Microsoft, metrics are the core data for monitoring. Metrics are time series data that measure the status of a system in three main areas: resource utilization (such as CPU and memory usage), workload performance (such as error rate and throughput), and service level agreement (such as down time and response time). These metrics are uniformly sampled and consist of real-valued data. They are sensitive to changes in performance and can offer valuable insights into the reliability of cloud services.

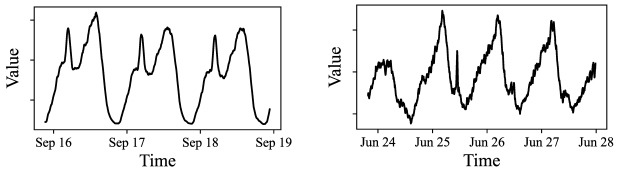


Figure 1: Examples of metric data in online service systems.

The collection and storage of metric data is already a mature technology; however, the analysis and decision-making regarding metrics are very challenging. On the one hand, understanding the physical meaning of metrics usually requires nontrivial expert knowledge. On the other hand, in a large cloud service system, the number of metrics is enormous (*e.g.*, millions of metrics exist in a practical service system). Therefore, efficient, accurate, and automated analysis and decision-making for metrics is the primary future direction of service monitoring in the industrial field.

2.1.2 Incident. In large-scale cloud systems, an incident refers to any unexpected disruption or decline in the quality of a service or system [8] [9, 19, 27]. This can result in service availability issues across various levels, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Generally, each incident is assigned a severity level based on its potential impact on systems, while classification standards vary among organizations (*e.g.*, Low, Medium, High, and Critical). In large-scale cloud systems, incidents typically originate from three ways. First, engineers use metric anomaly detection methods such as threshold rules or statistical method to detect anomalies in metric data, based on expert experience. These anomalies are interpretable and effectively reflect the system's state. Second, incidents are manually reported by engineers based on user feedback. Third, incidents are automatically generated when the alarm logic inherent in the service code is triggered. Figure 2 shows an incident report at Microsoft. Due to the company policy, we have to hide some details of these incidents and report relatively rough data instead.

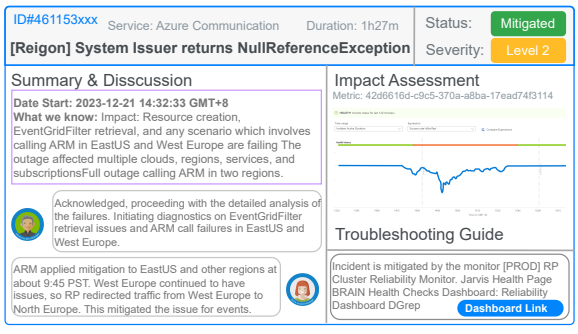


Figure 2: Example of incident data in online service systems.

2.1.3 Relationship between Metric and Incident. In the context of incident generation processes, a notable interrelation emerges between metric data and incident data. Both types of data serve as indicators of specific events within system and service operations. Metric data, characterized by its rapid response capability, often necessitates extensive expert knowledge for accurate interpretation. In contrast, incident data, while potentially subject to delays, offers greater clarity and accessibility for engineers. Incident identified through metric data are generally indicative of effective anomaly detection, whereas incidents reported manually may signal missed alarms in the anomaly detection system.

2.2 Metric Anomaly Detection

Anomaly detection for metrics is currently a fundamental and important aspect of monitoring data analysis [11, 18, 20, 31, 34, 39]. Through anomaly detection, potential anomalies in services can be discovered, and engineers can be alerted to resolve these anomalies promptly to ensure the reliability of services and prevent large-scale failures. Since metrics can be measured as time-series data, anomaly detection for metrics is usually formulated as time-series anomaly detection (TSAD) [22, 34, 39].

2.3 Motivation

Metric anomaly detection online for cloud systems is of significant importance for the stable operation of services. Although many advanced time series detection algorithms have been proposed in academia, these algorithms still struggle to find widespread application in industry. Based on our long-term practice in Microsoft's cloud service systems, we identify three main reasons why deep-learning-based detection models are not practical enough.

Model and hyperparameter selection. Choosing the model and corresponding hyperparameters is usually the first step in anomaly detection. The same model and hyperparameters cannot achieve satisfactory performance across all metrics [39], so selecting the appropriate model and parameters for different metrics is crucial to fully unleashing the performance of anomaly detection models. However, in a vast cloud system, the scale of monitor metrics is massive. Therefore, it is impractical for engineers to manually choose the appropriate model and hyperparameters for each metric.

Anomaly interpretation. In academia, accurately detecting anomalies is often the ultimate goal of anomaly detection. However, in the industrial sector, merely identifying the occurrence of anomalies is insufficient. Once an anomaly occurs, there is a substantial amount of work to be done. Therefore, helping engineers to interpret anomalies and provide insights for troubleshooting is also crucial for a practical anomaly detection detection system in the industrial context.

Engineer-system interaction. After deploying the model, engineers will collect a lot of feedback. However, with deep-learning-based models, it is difficult for engineers to interact efficiently and specifically with the anomaly detection models. Especially when service changes occur, if engineers cannot effectively feed back the information of service adjustments to the anomaly detection model, the current model may suffer significant performance degradation in the changed system, leading to the need to retrain and redeploy the anomaly detection model.

These three difficulties hinder the widespread application of the advanced time series anomaly detection models in industry. To address these challenges, we propose MonitorAssistant, automating model configuration recommendation, providing guidance-oriented anomaly reports and facilitating a more intuitive engineer-system interaction mechanism.

3 PRACTICAL ANOMALY DETECTION

Practical anomaly detection for metric data has always been a goal pursued in the industry. However, there has never been a precise answer to what constitutes practical anomaly detection or how to conduct practical anomaly detection. We are primarily interested in gaining a thorough understanding of practical anomaly detection for performance metrics in Microsoft's cloud systems that serve more than ten million users worldwide. Through our experience of conducting anomaly detection on these services, we have gained valuable insights into practical anomaly detection. In our exploration, we focus on answering the following questions:

Q1. **What are practical anomalies in cloud systems?**
Q2. **How engineers comprehend practical anomalies?**
Q3. **What capabilities should a practical anomaly detection system have?**

In the rest of this section, we analyze the practical anomaly detection in an empirical way, trying to answer the three questions mentioned above.

3.1 Practical Anomaly Definition

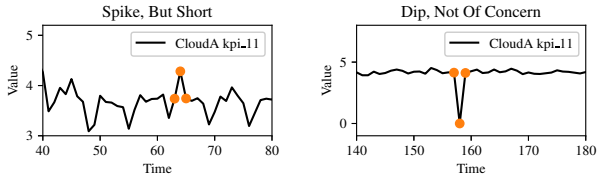
In our analysis of anomaly detection within large-scale cloud systems, we draw upon extensive experience with these complex environments. In such systems, anomalies are inevitable. The primary goal of anomaly detection models is to assist engineers in detecting potential risks that may adversely affect the system or highlight areas where enhancements can be made.

We have observed that anomalies identified by advanced models do not always align with what engineers deem significant. Occasionally, these models highlight anomalies that are not practically relevant. Thus, we propose a refined definition of a "practical anomaly" from an industrial standpoint: *Practical anomalies are those data points deviations that not only statistically diverge from the normal patterns but are also corroborated by related incidents.* These anomalies merit immediate attention and action from engineers, distinguishing them from the anomalies that might simply be statistical outliers without real-world implications.

Another noteworthy aspect concerning practical anomalies is that determining whether a metric has experienced an anomaly requires understanding the actual physical meaning of the metric. That is, anomalies that appear similar in form might be considered anomalous in some metrics but may not be in others. Therefore, the identification of such anomalies must rely on additional information. For this reason, even though deep-learning-based anomaly detection methods exhibit impressive performance, they typically cannot address all monitored metrics in a complex cloud service. Information beyond the metric values, such as descriptions of the

Table 1: Cases of practical anomalies with different shapes.

Shapes	Example Scenarios
Spike	Significant anomalies occur when service call latency unexpectedly increases, such as during a DDoS attack.
Dip	E-commerce platforms detect anomalies indicating a decline in user activity, possibly due to a service disruption.
Level Up	Sudden increase in disk usage, indicating potential log file flooding or unexpected data accumulation.
Level Down	Sudden decrease in network traffic, possibly indicating a loss of client connections.
Ramp Up	Gradual increase in CPU usage, potentially indicating a memory leak or an inefficient algorithm.
Ramp Down	Gradual decrease in response times, possibly due to improved system performance or reduced load.



(a) Impractical Spike **(b) Impractical Dip**

Figure 3: Examples of "impractical" anomaly in Microsoft's cloud system. The anomalies are marked by orange points.

physical meaning of metrics, is extremely important and necessary in practical anomaly detection.

Figure 3 shows two anomalies detected by a deep-learning-based model within Microsoft. However, in our context, both anomalies are impractical and do not necessitate triggering an incident for further engineering intervention. The anomaly in Figure 3a is a very small and short spike, yet the anomaly detection algorithm is highly sensitive to it, identifying this spike as statistically significant compared to the same historical period, and thus deeming it an anomaly. However, engineers are not concerned with such small and short spikes in this metric, as they almost have no impact on the whole system. The anomaly in Figure 3b, on the other hand, is a dip. Though intuitively it appears to significantly deviate from the normal pattern, any decline in this metric is not of concern to engineers. Therefore, even if a statistically significant dip occurs, engineers do not consider it a practical anomaly.

3.2 Practical Anomaly Interpretation

In academia, the primary goal of anomaly detection is usually to accurately identify anomalies [12, 18, 31, 34, 35, 40]. However, in industry, simply identifying whether an anomaly has occurred is far from enough. After detecting an anomaly, there is still a lot of work that needs to be done, such as prioritizing [4], triage [3], mitigation [15], and root cause analysis [5, 38]. Engineers need to interpret the anomaly to understand what happened in the system, derive knowledge from it, and then enrich subsequent incidents for the relevant engineers to troubleshoot. Therefore, precise detection of anomalies is not just crucial for a functional anomaly detection system; aiding engineers in comprehending anomalies and offering insights for troubleshooting holds significant importance, especially in the industrial setting.

The engineers who directly deal with anomalies are those responsible for the service. Typically, they are not experts in algorithms related to anomaly detection. Therefore, for service engineers, understanding and interpreting anomalies usually come from two aspects: firstly, the analysis of the anomaly **shapes**, and secondly, similar **incidents** related to this kind of anomaly. The analysis of anomaly shapes and similar incidents are often complementary, possessing hidden logical relationships. Combining both provides service engineers with a more comprehensive understanding of the specific anomaly.

3.2.1 Anomaly Characteristic. For anomaly classification within the academic sphere, prior researches have established a foundational taxonomy [13, 33]. Drawing from our empirical insights, we posit that the most salient attributes of anomalies pertain to their **duration** and **shape**. This perspective is informed by our hands-on experience, underscoring the importance of these characteristics in the practical assessment and management of anomalies.

Duration. The duration of the anomaly is a critical attribute in our anomaly classification. It's the length of the continuous segment marked as anomalous. Anomalies that manifest over a brief time span might be noisy false positives or insignificant aberrations that revert to normal levels quickly. Despite their fleeting nature, such short-lived anomalies can be crucial to some engineers who focus on immediate system response and correction. On the other hand, anomalies with longer time ranges might indicate persistent or systematic issues and are often of interest to engineers seeking to understand and address deeper, underlying problems.

Shape. The term 'Shape' describes the distinctive pattern that an anomalous sequence takes when contextualized within its surrounding normal sequences. This categorization provides critical insights for engineers, enabling the identification of the type and root cause of anomalies, as those stemming from the same origin often exhibit similar shapes. This understanding allows engineers to focus specifically on certain shapes, according to their diagnostic needs. For the purpose of our study, we categorized anomalies into six primary shape types: spike, dip, level shift up, level shift down, ramp up and ramp down as shown in Figure 4. We provide typical cases for these six shapes in Table 1.

3.2.2 Metric-Incident Pair. Incidents, as a form of high-level data, also directly assist engineers in understanding and interpreting

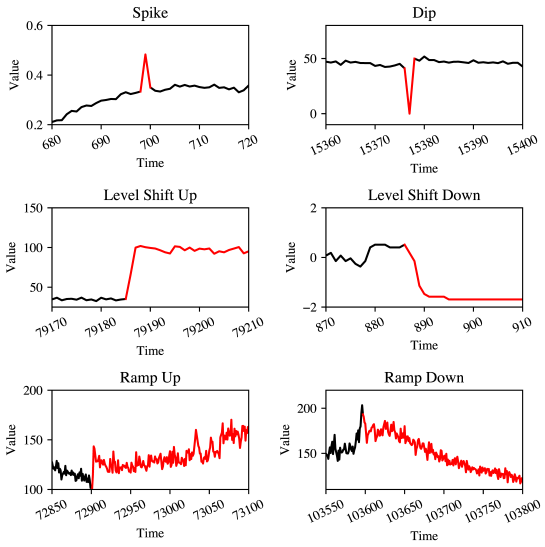


Figure 4: Illustration of the six types of anomaly shapes our engineers care most.

anomalies. When an anomaly detector reports a metric anomaly, there is no incident associated with this specific anomaly. This is because many incidents are generated in response to metric anomalies. Typically, incidents and metric anomalies are paired together to form metric-incident pairs. That means for each incident, the related metric details and the specific anomaly segment that triggered the incident are recorded. However, the absence of an incident for a current anomaly does not mean engineers cannot gain insights from historical incidents. When a metric anomaly occurs, engineers will, based on their experience, review historical incidents of similar anomalies or trends, attempting to understand the current anomaly and derive its root causes and troubleshooting recommendations from these insights. Moreover, incidents usually encompass a vast amount of expert knowledge and historical experience, which is significantly valuable for understanding practical anomalies. Academic research on anomaly detection often focuses more on the metric data itself, thus may not appear as practical in industries.

3.3 Capabilities of Practical Anomaly Detection

Based on our in-depth practice on practical anomaly detection, we conclude that a practical anomaly detection system in the industrial sector must achieve three capabilities.

First, a practical anomaly detection system should not only incorporate advanced deep-learning-based anomaly detection models, but also inherit the long-term accumulated practical experience of the industry. The core reasons are twofold. Firstly, many rule-based or classical statistical methods have been proven effective over years of practice on certain metrics. Moreover, the results obtained from these simple methods are not only highly interpretable but also usually have significant efficiency advantages compared to advanced deep-learning-based methods, which can greatly save computational resources. Therefore, completely abandoning these types of methods has no practical value for the industry. Secondly, as Section 3.2 mentioned, the historical metric-incident pairs contain a large amount of expert experience, which is crucial for engineers to explain and understand anomalies. Engineers responsible for services are not willing to see this kind of knowledge fail to be effectively inherited and utilized in anomaly detection systems.

Second, it is necessary to provide engineers with a comprehensive anomaly reports rather than detection results. The final results provided by a practical anomaly detection system should not merely indicate the presence of anomalies, but also offer further explanations and corresponding troubleshooting recommendations. As Section 3.2 mentioned, accurately detecting anomalies is only the starting point, not the end. If an anomaly detection system can only determine whether an anomaly has occurred, then its value to engineers is greatly diminished. A practical anomaly detection system needs to provide engineers with more insights, helping them better understand the anomaly and perform subsequent troubleshooting work. Therefore, a practical anomaly detection system needs to report anomalies in the form of a report. The report should cover the understanding and explanation of the anomaly, and provide engineers with as much troubleshooting insight as possible.

Third, a practical anomaly detection system should offer engineers an friendly interaction process with a low algorithmic threshold. Inevitably, anomaly detection methods will have false negatives or false positives. The traditional way is to patch the anomaly detection model, or collect a certain amount of data to fine-tune or retrain the model, thereby improving the model. In this process, a large amount of expert knowledge about anomaly detection models and algorithms is usually needed. However, in industry, engineers responsible for services are not usually directly in charge of the anomaly detection model. They need to communicate with algorithm engineers to pass feedback to the algorithm. This is a very inefficient process, as the threshold for service engineers to transfer their domain knowledge is too high, making it difficult to utilize feedback information. If the anomaly detection system itself could provide a very simple and direct way of interaction with a low algorithmic threshold, it would make the utilization of feedback information more convenient, making the system more practical.

4 MONITORASSISTANT

Through the analysis of Section 3, inheriting and transferring domain knowledge in the field of anomaly detection to achieve a practical process is indispensable for natural language capabilities, because a large amount of domain knowledge in anomaly detection within the industrial sector is inherited and transferred in the form of natural language.

To achieve practical anomaly detection, strong capabilities in processing semi-structured and unstructured data, especially in understanding natural language, are essential. With the development of Large Language Models (LLMs) in recent years, LLMs like GPT-4 [1] have demonstrated impressive natural language processing capabilities across different domains [1, 10, 14, 23, 42]. This ability to compress and distill knowledge from natural language aligns precisely with the requirements for practical anomaly detection.

Hence, we propose MonitorAssistant, an LLM-based end-to-end practical anomaly detection system. It refines and summarizes a large amount of domain knowledge in anomaly detection from

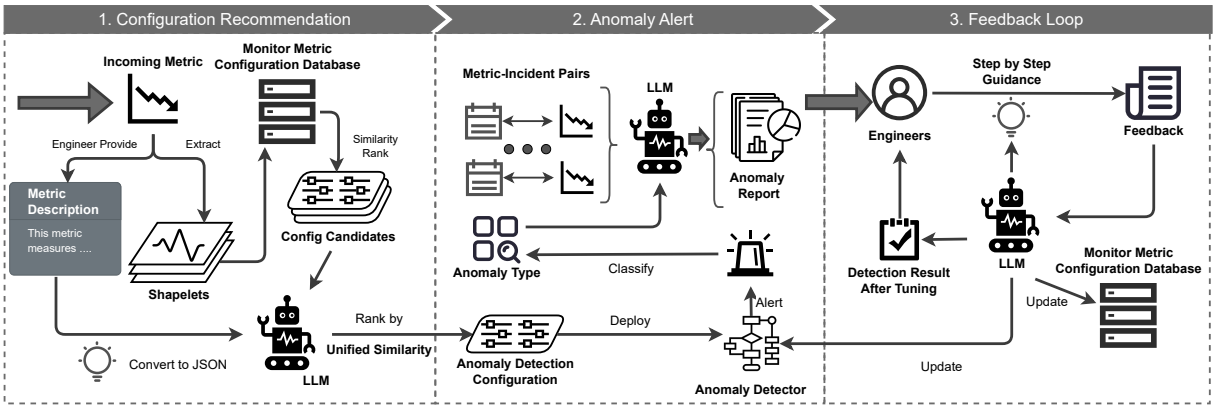


Figure 5: The overall workflow of MonitorAssistant facing an incoming metric

historical incident-metric pairs, achieving knowledge inheritance. Simultaneously, MonitorAssistant can automate the pipeline of anomaly detection and provide an informative anomaly report with actionable guidance for troubleshooting, rather than a simple determination of anomaly occurrence. Moreover, MonitorAssistant offers an interaction process with a low algorithmic threshold for engineers, making their feedback more effectively influence the anomaly detection system. MonitorAssistant has already been deployed in Microsoft's cloud service system.

In the remaining part of this section, we will detail the workflow of MonitorAssistant and the key novel technologies that enable practical anomaly detection.

4.1 Workflow

As illustrated in Figure 5, when an incoming metric is integrated, MonitorAssistant undergoes three phases: Configuration Recommendation, Anomaly Alert and Feedback Loop. In all three phases, LLM is the core component. We choose the GPT-4 Turbo model for MonitorAssistant because a variety of complex natural language tasks are designed in MonitorAssistant, and GPT-4 Turbo model achieves state-of-the-art overall performance in many natural language tasks.

The first phase is the Configuration Recommendation phase, where MonitorAssistant, based on the characteristics of the data, recommends promising configurations for the metric. In Figure 5, the main focus of configuration recommendation phase is on the online process, while the monitor metric configuration database in Figure 5 is constructed in the offline process. To implement the aforementioned process and achieve the goals of knowledge inheritance and utilization, we introduce the Monitor Configuration Infusion technique. It is the core of the entire configuration recommendation phase, and we provide a detailed description of this technique in Section 4.2.

The second phase is the Anomaly Alert phase. When the model detects an anomaly in the metric, it will generate an alert with type information and provides corresponding troubleshooting suggestions. The types of anomalies mainly include seven categories, among which six are mentioned in Section 3.2: Spike, Dip, Level Up, Level Down, Ramp Up, and Ramp Down. The rest are uniformly considered as the Other type. The classification results are provided by a trained fully connected network. As for the anomaly report, we design a practical alert generation technique, we introduce this technique in Section 4.3 in detail.

The final stage is the Feedback Loop, where engineers provide feedback based on the performance of the anomaly detection model. MonitorAssistant then adjusts the anomaly detection configuration according to the engineers' feedback, enhancing the model's performance. The detail of the way to achieve LLM-Engineer-In-The-Loop interaction are shown in Section 4.4.

4.2 Monitor Configuration Infusion

In the Configuration Recommendation phase, two core issues need to be addressed. The first is how to extract the knowledge contained in historical anomaly detection experiences to provide references for subsequent configuration recommendations. The second issue is how to utilize this knowledge to make reasonable recommendations during the recommendation process. To tackle these two issues, we propose Monitor Configuration Infusion technique.

To address the extraction of historical experiences, we built a Monitor Metric Configuration Database. This is a database for JSON files. For each metric, we extract the key-value pairs by LLM and create a JSON file, which contains key information about the metric. Two fields worth noting are shapelets and incidents.

Shapelets refer to representative time series segments within a metric [28, 37]. Since the length of a metric is constantly increasing and the total volume is enormous, directly storing the metric is impractical. Therefore, we extract shapelets from the metric using the Fast Shapelet Discovery (FSD) [28], serving as the primary representation of the metric. The incidents field records all incident IDs related to the metric. These incidents can be considered as records of historical anomalies that have occurred with the metric. This field plays an important role in subsequent anomaly report generation.

For making recommendations based on historical anomaly detection experiences, an intuitive idea is to recommend based on similarity. However, calculating similarity solely based on time

Algorithm 1: Calculation of Unified Similarity

Input: Incoming metric X and a history metric X_h
Input: The description data of incoming metric D and history metric D_h
Output: *usim* Unified Similarity between the incoming metric and history metric
Function `UnifiedSimilarity`(X, X_h, D, D_h)**:**

```
// Extract n shapelets from X
S = {s_1, s_2, ...S_n} ← ShapeletExtraction(X);
// Query from Monitor Metric Configuration Database
S_h = {s_1^h, s_2^h, ...S_m^h} ← ShapeletExtraction(X_h);
// Initialize the average metric similarity
msim_avg ← 0;
foreach s in S = {s_1, s_2, ...S_n} do
    // Record the maximum similarity between s and s_h
    max_sim ← 0;
    // Normalization
    s ← (s − min(s)) / (max(s) − min(s));
    foreach s_h in S_h = {s_1^h, s_2^h, ...S_m^h} do
        s_h ← (s_h − min(s_h)) / (max(s_h) − min(s_h));
        // Calculate the shapelet distance by Shape
           Based Distance (SBD) [25]
        distance ← SBD(s, s_h);
        // Get the shapelet similarity msim ∈ [0, 1]
        msim ← (max(length(s), length(s_h)) − distance) / max(length(s), length(s_h));
        if msim > max_sim then
            max_sim ← msim;
        end
    end
    msim_avg ← msim_avg + max_sim;
end
msim_avg ← msim_avg / n;
// Extract key-value pairs converting to JSON
D ← LLM(D, task = KVExtract);
D_h ← LLM(D_h, task = KVExtract);
// Get description similarity dsim ∈ [0, 1] from LLM
dsim ← LLM(D, D_h, task = SimiCal);
usim = (msim_avg + dsim) / 2;
return usim;
```

series would lose a significant amount of natural language information describing the metric. Therefore, we proposed unified similarity. Unified similarity combines the similarity of the time series itself with the similarity of related descriptive information, fully utilizing the metric's physical significance and a vast amount of descriptive information. The specific calculation process is shown in Algorithm 1. For time series similarity, we calculate based on shapelets rather than the series itself, because calculating based on the original series would take a significant amount of time, failing to meet the requirements of industrial applications. The similarity of descriptive information is calculated by LLM. The main approach is to provide LLM with manually constructed similarity cases as prompts, allowing it to output a similarity value between 0 and 1. However, since calling LLM is also very time-consuming, in MonitorAssistant, we first use time series similarity for preliminary screening, selecting the Top N metrics with the highest similarity. Afterwards, we calculate the unified similarity among these N metrics and finally recommend the configuration of the anomaly detection model for the incoming metric with the highest similarity.

4.3 Practical Alert Generation

Understanding anomalies often requires integrating a large amount of external information and engineers' long-term accumulated expert knowledge. Therefore, for a practical anomaly detection system, merely determining whether a metric is abnormal lacks practical significance for engineers. To tackle this issue, we propose the **Practical Alert Generation** mechanism. It not only can determine whether a metric is abnormal but also can combine a vast amount of historical experience to summarize an anomaly report, helping engineers better understand anomalies, reduce the cost of understanding anomalies, and assist engineers in effectively troubleshooting.

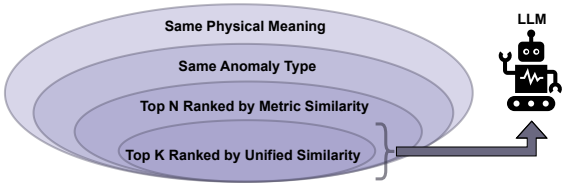


Figure 6: Filtering process of historical metric data

In the context of Practical Alert Generation, the most critical issue is how to filter valuable data from historical records for summarization. In our approach, as mentioned in Section 3.2, for engineers, duration and shape serve as the most direct entry points for understanding anomalies. However, when an anomaly is detected, it has just occurred, making the duration uncertain at that moment, but the shape is generally ascertainable. Thus, we emulate the way engineers handle anomalies by summarizing knowledge and experience from history. Engineers primarily focus on six types of anomaly shapes: Spike, Dip, Level Up, Level Down, Ramp Up, Ramp Down (see Section 3.2 for more details), simplifying the need for a complex classifier. MonitorAssistant employs a pre-trained fully connected network classifier to categorize anomalies. Any anomaly that does not fit into the aforementioned six categories is classified as "Other". Then, the anomaly category information is added as an entry in the JSON generated for that metric by the Model Configuration Infusion mechanism. Subsequently, we filter relevant data from historical metric-incident pairs using the following steps:

(1) Filter metrics with the same physical meaning based on their physical meanings.
(2) Filter metric-incident pairs that have experienced the same type of anomaly based on category.
(3) Calculate metric similarity using the current anomaly segment and the recorded anomaly segments in existing metric-incident pairs.

(4) Sort by similarity and select the Top N, then compute the unified similarity (see Algorithm 1) with the LLM to determine the final Top K.

After identifying the Top K related metric-incident pairs, we concatenate various fields and have the LLM summarize them by field. Finally, we use the LLM to organize the field-wise summaries into an anomaly report. The report includes: the physical meaning of the metric, the type of anomaly, similar past incidents' IDs, possible root causes of the anomaly, and recommendations for troubleshooting.

4.4 LLM-Engineer-In-The-Loop Interaction

To improve the process of engineer-model interaction by LLM, we firstly analyze the weakness of the traditional interaction way. In the industrial sector, engineers responsible for monitoring metric and handling anomaly detection results are typically service engineers. These individuals are not usually experts in anomaly detection algorithms; their domain knowledge is often specific to the metrics themselves. The development and maintenance of anomaly detection algorithms are usually carried out by algorithm engineers. When service engineers identify issues such as miss or false alarms in the anomaly detection system, they directly communicate with algorithm engineers, who are then responsible for addressing the issues related to the anomaly detection model. Due to algorithm engineers' lack of in-depth understanding of the metrics, it is often challenging for them to accurately grasp the service engineers' concerns. Consequently, this mode of interaction is highly inefficient, making it difficult for service engineers to effectively transfer their domain knowledge to the anomaly detection model itself.

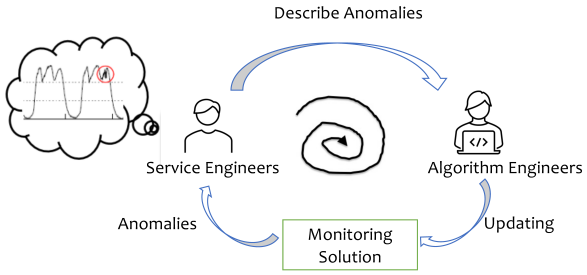


Figure 7: Traditional engineer-model interaction.

To address the aforementioned issue, we designed the LLM-Engineer-In-The-Loop interaction mechanism, allowing service engineers to interact directly with an LLM, providing a low-threshold way for transferring domain knowledge. This enables service engineers to effectively aid in improving the anomaly detection system, thereby allowing the system to continuously incorporate the service engineers' domain knowledge.

The core of the LLM-Engineer-In-The-Loop interaction mechanism is the understanding of the engineers' intentions. Only by clearly understanding the service engineers' intentions can the anomaly detection model be accurately improved according to their ideas. However, for engineers who are unfamiliar with LLM, their untrained, unstructured natural language inputs present an additional understanding barrier for the LLM. Therefore, we designed a step-by-step guidance mechanism illustrated by Figure 8, allowing the LLM to ask engineers questions gradually, constructing a semi-structured context environment step by step. Finally, the LLM organizes each of the engineer's responses according to our designed template into feedback with a clear logical structure.

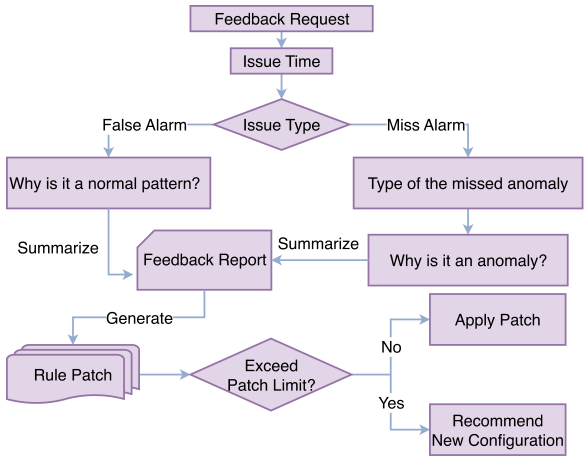


Figure 8: The workflow of the step-by-step guidance in LLM-Engineer-In-The-Loop Interaction.

After guiding engineers to complete the feedback process, MonitorAssistant first generates rule-based patches targeting the feedback. This rule-based patching approach is highly computationally efficient. Furthermore, by patching based on the raw results of anomaly detection models, this method minimizes the likelihood of affecting the original performance of the models.

When the patches to this model exceed a certain threshold, MonitorAssistant will, based on existing information, initiate a configuration recommendation process to prevent excessive patching from affecting anomaly detection performance. It will recommend a more suitable anomaly detection model for the metric based on the available information. This behavior essentially mimics the approach human engineers take to address such issues. In real-world environments, when individual cases of false alarms or miss alarms are encountered, the intuitive response is not to directly fine-tune the model with this data, but rather to first use rule-based patches to address similar issues.

5 CASE STUDY

To evaluate the effectiveness of MonitorAssistant in the real-world environment, we deployed MonitorAssistant in Microsoft's cloud service system and integrated real performance metric data into MonitorAssistant. During this process, we collected feedback from several engineers regarding MonitorAssistant and gathered real-case studies to analyze the effectiveness of MonitorAssistant in practical anomaly detection.

5.1 Configuration Recommendation

To validate the effectiveness of model recommendation, we collected workload metric from a service in Microsoft's production environment and provided a brief description of this metric as input to MonitorAssistant. Figure 9 shows the process by which

MonitorAssistant selects models for this metric and the effect of anomaly detection.

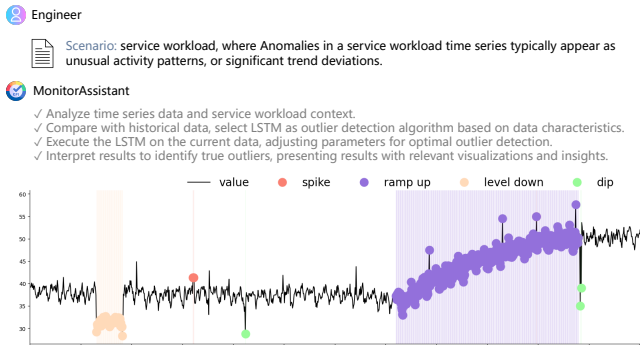


Figure 9: Case for configuration recommendation. Ramp up anomalies are marked by purple, level shift down anomalies are marked by yellow and spike anomalies are marked by red points.

In this scenario, MonitorAssistant selected an LSTM-based anomaly detection model for the given metric because LSTM-based models are sensitive to trend-like anomalies [46], which aligns with the engineers' highlighted concern for the metric's trend in the description. Moreover, MonitorAssistant detected anomalies such as level downs. The initial recommendation for the anomaly detection model by MonitorAssistant did not have any miss alarms, detecting all practical anomalies of concern to the engineers.

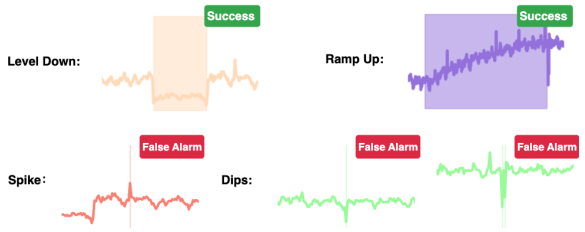


Figure 10: All alarms in the case of Figure 9.

However, the initial recommended model still had some flaws. Due to the LSTM's heightened sensitivity to local information [12], the model produced some false alarms, as shown in Figure 10, a frequent issue with deep-learning-based models like LSTM [39, 46]. In Section 5.3, we will explain how MonitorAssistant resolves these flaws through a feedback loop.

5.2 Anomaly Report

In the Section 5.1, it is demonstrated how to ascertain whether a metric anomaly has occurred. However, MonitorAssistant does not stop there. To facilitate engineers' comprehension, MonitorAssistant provides a more practically meaningful anomaly report by summarizing historical experiences. Table 2 shows an anomaly report about the ramp up anomaly in Figure 10.

From Table 2, we can see that MonitorAssistant presents multiple possible scenarios to assist engineers in conducting a deeper investigation into the anomaly by integrating the physical meaning of the metric and historical anomaly detection knowledge . Moreover, in the troubleshooting guide, MonitorAssistant offers engineers preliminary troubleshooting suggestions, which can significantly expedite the resolution of the anomaly.

5.3 Feedback Loop

Due to the complexity of anomaly detection, deployed anomaly detection models are prone to issues such as false alarms or miss alarms during operation. False alarms, in particular, pose a significant challenge [12, 18, 31, 33, 46]. In the case in Section 5.1, MonitorAssistant experienced false alarms, due to the deep-learning-based model's sensitivity to local information [12]. When engineers encounter instances of false alarms, they can directly submit feedback to MonitorAssistant, bypassing the need to involve algorithm engineers. Figure 11 illustrates this process.

Figure 11: Case for feedback loop. Ramp up anomalies are marked by purple, level shift down anomalies are marked by yellow.

Once engineers submit all feedback, MonitorAssistant selectively adjusts the anomaly detection model's strategies. It then tests the improved model on previous data and provides the results for engineer confirmation.

It is noteworthy that the original anomaly detection model produced three false alarm dips, but the engineers provided only one example. Nonetheless, MonitorAssistant successfully resolved all false alarm dips indicating that MonitorAssistant can deeply understand the engineers' intentions, rather than merely addressing the cases precisely described by the engineers.

However, if engineers have additional feedback, the process continues iteratively. Our real-world case study shows that MonitorAssistant solve the false alarm issues for the service engineer after one iteration of the feedback loop, demonstrating that

Table 2: Anomaly Report of a Ramp Up Anomaly in Service A Workload.

Field	Description
Metric ID	ffb82d38-5f00-37db-abc0-5d2e4e4cb6aa
Physical Meaning	The workload of Service A
Anomaly Type	Ramp Up
Anomaly Range	2023-11-19 14:43:20 to 2023-11-19 20:43:20
Similar Incident IDs	414072, 421105, 443790
Root Cause Analysis	• **Sudden Increase in User Traffic**: A rapid increase in user traffic, possibly due to marketing campaigns, new feature releases, or viral content. • **Service Dependencies**: Issues in dependent services or APIs, such as increased latency or errors, can cause retries or back-off mechanisms to kick in. • **Resource Limitations**: Hitting the limits of allocated resources (CPU, memory, database connections, etc.) can create bottlenecks, making the system less efficient. • **Configuration Changes**: Configuration updates, including those in the infrastructure or in the application (like feature flags), can alter the system's behavior and impact its workload. • **Cache Evictions or Failures**: If a caching layer fails or experiences a high rate of cache misses, the increased load will fall back to the primary data stores or computation resources.
Troubleshooting Guide	Steps include: reviewing recent changes, understanding user behavior changes, use APM tools to drill down into specific transactions., implementing scalability best practices, conducting load testing, and apply rate limiting to control the traffic volume from individual users or IPs.

MonitorAssistant effectively interfaces with service engineers, streamlining the process for them to directly apply their expertise to the anomaly detection models.

6 RELATED WORK

Cloud service monitoring, particularly in the context of metric data, has garnered significant attention in both academia and industry due to its critical applications in various domains. Monitoring strategies commonly encompass anomaly detection, alerting mechanisms, and incident management to ensure the reliability and performance of cloud services.

Anomaly detection: Early approaches to time series anomaly detection often relied on statistical techniques such as moving averages, standard deviation analysis, and z-score normalization [24, 36, 41]. Recent advancements in deep learning have led to the development of neural network-based models tailored for time series anomaly detection [12, 18, 20–22, 29, 31, 34, 35, 39, 40, 45]. However, effectively applying anomaly detection algorithms to large-scale cloud service systems remains a significant challenge, particularly in achieving high accuracy, interpretability, and friendly engineer-system interaction [39] .

Alerting mechanisms: In industry settings, the alerting mechanisms typically relies on hand-crafted rules based on domain knowledge, incorporating approaches like alert aggregation [44], alert correlation [6], and alert ranking [4]. However, these alert strategies not only have a high algorithmic threshold but also struggle to provide engineers with additional context or guiding suggestions beyond metric data, which fails to effectively assist engineers in improving their efficiency in resolving anomalies.

LLM-based service monitoring: In recent years, the emergence of Large Language Models (LLMs) has opened new avenues in the realm of cloud service monitoring. For instance, GPT models have been leveraged to suggest root causes and mitigation strategies to streamline cloud incident handling [2, 5, 42], conduct auto-remediation [30], as well as to generate outage summaries [15]. Our work diverges from these efforts, aiming to bridge the gap between the capabilities of LLMs and the practical monitoring requirements.

7 CONCLUSION

This paper proposes MonitorAssistant, an end-to-end practical anomaly detection system that leverages LLM to bridge the gap between academic research and industrial needs in the realm of cloud service monitoring. By integrating model configuration recommendation, providing guidance-oriented anomaly reports and LLM-engineer-in-the-loop interaction mechanism, MonitorAssistant simplifies the anomaly detection process, making it more accessible, interpretable and friendly for engineers. Our deployment of MonitorAssistant within Microsoft's cloud service system has not only demonstrated its practical effectiveness but also its potential to significantly enhance the reliability and stability of large-scale cloud services. The success of MonitorAssistant underscores the importance of developing technology that is both advanced and user-friendly, paving the way for future innovations in the monitoring and maintenance of large-scale cloud systems.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant 62072264, and the Beijing National Research Center for Information Science and Technology (BNRist) key projects. We would also like to acknowledge Bingchen Shi for his contribution to the study.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. 2023. Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models. *arXiv preprint arXiv:2301.03797* (2023).

[3] Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. Continuous incident triage for large-scale online service systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 364–375.

[4] Junjie Chen, Shu Zhang, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Yu Kang, Feng Gao, Zhangwei Xu, Yingnong Dang, et al. 2020. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 373–384.

[5] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2024. Automatic Root Cause Analysis via Large Language Models for Cloud Incidents. (2024).

[6] Yujun Chen, Xian Yang, Hang Dong, Xiaoting He, Hongyu Zhang, Qingwei Lin, Junjie Chen, Pu Zhao, Yu Kang, Feng Gao, et al. 2020. Identifying linked incidents in large-scale online service systems. In *Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*. 304–314.

[7] Yuhang Chen, Chaoyun Zhang, Minghua Ma, Yudong Liu, Ruomeng Ding, Bowen Li, Shilin He, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. ImDiffusion: Imputed Diffusion Models for Multivariate Time Series Anomaly Detection. *Proc. VLDB Endow.* 17, 3 (2023), 359–372.

[8] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, Yingnong Dang, Feng Gao, Pu Zhao, Bo Qiao, Qingwei Lin, Dongmei Zhang, and Michael R. Lyu. 2020. Towards intelligent incident management: why we need it and how we make it. In *ESEC/SIGSOFT FSE*. ACM, 1487–1497.

[9] Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Xiaomin Wu, Meng Zhang, Qingjun Chen, Xin Gao, Xuedong Gao, et al. 2023. TraceDiag: Adaptive, Interpretable, and Efficient Root Cause Analysis on Large-Scale Microservice Systems. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1762–1773.

[10] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines* 30 (2020), 681–694.

[11] Vaibhav Ganatra, Anjaly Parayil, Supriyo Ghosh, Yu Kang, Minghua Ma, Chetan Bansal, Suman Nath, and Jonathan Mace. 2023. Detection Is Better Than Cure: A Cloud Incidents Perspective. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1891–1902.

[12] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Söderström. 2018. 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*, Yike Guo and Faisal Farooq (Eds.). ACM, 387–395.

[13] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.

[14] Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Xpert: Empowering Incident Management with Query Recommendations via Large Language Models. arXiv:2312.11988 [cs.SE]

[15] Pengxiang Jin, Shenglin Zhang, Minghua Ma, Haozhe Li, Yu Kang, Liqun Li, Yudong Liu, Bo Qiao, Chaoyun Zhang, Pu Zhao, et al. 2023. Assess and Summarize: Improve Outage Understanding with Large Language Models. (2023).

[16] Haozhe Li, Minghua Ma, Yudong Liu, Si Qin, Bo Qiao, Randolph Yao, Harshwardhan Chaturvedi, Tri Tran, Murali Chintalapati, Saravan Rajmohan, et al. 2023. CODEC: Cost-Effective Duration Prediction System for Deadline Scheduling in the Cloud. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 298–308.

[17] Liqun Li, Xu Zhang, Shilin He, Yu Kang, Hongyu Zhang, Minghua Ma, Yingnong Dang, Zhangwei Xu, Saravan Rajmohan, Qingwei Lin, et al. 2023. Conan: Diagnosing batch failures for cloud systems. In *IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 138–149.

[18] Zhihan Li, Youjian Zhao, Jiaqi Han, Ya Su, Rui Jiao, Xidao Wen, and Dan Pei. 2021. Multivariate Time Series Anomaly Detection and Interpretation using Hierarchical Inter-Metric and Temporal Embedding. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, 3220–3230.

[19] Qingwei Lin, Tianci Li, Pu Zhao, Yudong Liu, Minghua Ma, Lingling Zheng, Murali Chintalapati, Bo Liu, Paul Wang, Hongyu Zhang, et al. 2023. Edits: An easy-to-difficult training strategy for cloud failure prediction. In *Companion Proceedings of the ACM Web Conference 2023*. 371–375.

[20] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. 2015. Opprentice: Towards Practical and Automatic Anomaly Detection Through Machine Learning. In *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, Kenjiro Cho, Kensuke Fukuda, Vivek S. Pai, and Neil Spring (Eds.). ACM, 211–224.

[21] Jinyang Liu, Wenwei Gu, Zhuangbin Chen, Yichen Li, Yuxin Su, and Michael R Lyu. 2024. MTAD: Tools and Benchmarks for Multivariate Time Series Anomaly Detection. *arXiv preprint arXiv:2401.06175* (2024).

[22] Jinyang Liu, Tianyi Yang, Zhuangbin Chen, Yuxin Su, Cong Feng, Zengyin Yang, and Michael R. Lyu. 2023. Practical Anomaly Detection over Multivariate Monitoring Metrics for Online Services. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. 36–45. https://doi.org/10.1109/ISSRE59848.2023.00045

[23] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. GPT understands, too. *AI Open* (2023).

[24] H Zare Moayedi and MA Masnadi-Shirazi. 2008. Arima model for network traffic prediction and anomaly detection. In *2008 international symposium on information technology*, Vol. 4. IEEE, 1–6.

[25] John Paparrizos and Luis Gravano. 2015. k-Shape: Efficient and Accurate Clustering of Time Series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (Melbourne, Victoria, Australia) *(SIGMOD '15)*. Association for Computing Machinery, New York, NY, USA, 1855–1870. https://doi.org/10.1145/2723372.2737793

[26] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S Tsay, Themis Palpanas, and Michael J Franklin. 2022. TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection. *Proceedings of the VLDB Endowment* 15, 8 (2022), 1697–1711.

[27] Xiaoting Qin, Minghua Ma, Yuheng Zhao, Jue Zhang, Chao Du, Yudong Liu, Anjaly Parayil, Chetan Bansal, Saravan Rajmohan, Íñigo Goiri, et al. 2023. How different are the cloud workloads? characterizing large-scale private and public cloud workloads. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 522–530.

[28] Thanawin Rakthanmanon and Eamonn Keogh. 2013. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 668–676.

[29] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 3009–3017.

[30] Komal Sarda, Zakeya Namrud, Raphael Rouf, Harit Ahuja, Mohammadreza Rasolroveicy, Marin Litoiu, Larisa Shwartz, and Ian Watts. 2023. Adarma auto-detection and auto-remediation of microservice anomalies by leveraging large language models. In *Proceedings of the 33rd Annual International Conference on Computer Science and Software Engineering*. 200–205.

[31] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 2828–2837.

[32] Zexin Wang, Changhua Pei, Minghua Ma, Xin Wang, Zhihan Li, Dan Pei, Saravan Rajmohan, Dongmei Zhang, Qingwei Lin, Haiming Zhang, Jianhui Li, and Gaogang Xie. 2024. Revisiting VAE for Unsupervised Time Series Anomaly Detection: A Frequency Perspective. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*. ACM, 3096–3105.

[33] Canhua Wu, Nengwen Zhao, Lixin Wang, Xiaoqin Yang, Shining Li, Ming Zhang, Xing Jin, Xidao Wen, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2021. Identifying Root-Cause Metrics for Incident Diagnosis in Online Service Systems. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. 91–102.

[34] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 world wide web conference*. 187–196.

[35] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. In *ICLR*. OpenReview.net.

[36] Xiaohan Yan, Ken Hsieh, Yasitha Liyanage, Minghua Ma, Murali Chintalapati, Qingwei Lin, Yingnong Dang, and Dongmei Zhang. 2023. Aegis: Attribution of Control Plane Change Impact across Layers and Components for Cloud Systems. In *IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 222–233.

[37] Lexiang Ye and Eamonn Keogh. 2009. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 947–956. https://doi.org/10.1145/1557019.1557122

[38] Zhaoyang Yu, Qianyu Ouyang, Changhua Pei, Xin Wang, Wenxiao Chen, Liangfei Su, Huai Jiang, Xuanrun Wang, Jianhui Li, and Dan Pei. 2024. Causality Enhanced Graph Representation Learning for Alert-Based Root Cause Analysis. In *2024 IEEE/ACM 24rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE.

[39] Zhaoyang Yu, Changhua Pei, Shenglin Zhang, Xidao Wen, Jianhui Li, Gaogang Xie, and Dan Pei. 2023. AutoKAD: Empowering KPI Anomaly Detection with Label-Free Deployment. In *ISSRE*. IEEE, 13–23.

[40] Zhaoyang Yu, Shenglin Zhang, Mingze Sun, Yingke Li, Yankai Zhao, Xiaolei Hua, Lin Zhu, Xidao Wen, and Dan Pei. 2024. Supervised Fine-Tuning for Unsupervised KPI Anomaly Detection for Mobile Web Systems. In *Proceedings of the ACM on Web Conference 2024*. 2859–2869.

[41] Zhengran Zeng, Yuqun Zhang, Yong Xu, Minghua Ma, Bo Qiao, Wentao Zou, Qingjun Chen, Meng Zhang, Xu Zhang, Hongyu Zhang, et al. 2023. Traceark: Towards actionable performance anomaly alerting for online service systems. In *IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 258–269.

[42] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. 2024. Automated Root Causing of Cloud Incidents using In-Context Learning with GPT-4. arXiv:2401.13810 [cs.CL]

[43] Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang, Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, Yuzhi Zhang, et al. 2023. Robust multimodal failure detection for microservice systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5639–5649.

[44] Nengwen Zhao, Junjie Chen, Xiao Peng, Honglin Wang, Xinya Wu, Yuanzong Zhang, Zikai Chen, Xiangzhong Zheng, Xiaohui Nie, Gang Wang, et al. 2020. Understanding and handling alert storm for online service systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*. 162–171.

[45] Nengwen Zhao, Junjie Chen, Zhaoyang Yu, Honglin Wang, Jiesong Li, Bin Qiu, Hongyu Xu, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2021. Identifying bad software changes via multimodal 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*. 527–539.

[46] Zhenyu Zhong, Qiliang Fan, Jiacheng Zhang, Minghua Ma, Shenglin Zhang, Yongqian Sun, Qingwei Lin, Yuzhi Zhang, and Dan Pei. 2023. A Survey of Time Series Anomaly Detection Methods in the AIOps Domain. *arXiv preprint arXiv:2308.00393* (2023).

Received 2024-02-08; accepted 2024-04-18