2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE) | 979-8-3503-1594-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISSRE59848.2023.00084

fKPISelect: Fault-Injection Based Automated KPI Selection for Practical Multivariate Anomaly Detection

Xingjian Zhang[1,2], Yinqin Zhao[1,2], Chang Liu[1], Long Wang[1,3],
Xin Yang[2], Yefei Hou[2], Zhongwen Lan[2], Xining Hu[2],
Beibei Miao[2], Ming Yang[2], Xiangyi Jing[2], Sijie Li[2]
[1]REASONS Lab, Institute for Network Sciences and Cyberspace, Tsinghua University
[2]China Telecom Cloud Technology Co., Ltd. [3]Zhongguancun Laboratory
{zhang-xj22, zyq21, chang-liu22}@mails.tsinghua.edu.cn, longwang@tsinghua.edu.cn,
{yangxin6, houyf1, lanzw12, huxining, miaobb, yangm37, jingxy1, lisj22}@chinatelecom.cn

***Abstract*—IT services are now popularly hosted in cloud systems. In order to enhance the availability of cloud services, an emerging approach for detecting failures of cloud components is to monitor Key Performance Indicators (KPIs) of the components and apply Neural Network based AI technologies to detect KPI anomalies. Multivariate Time Series Anomaly Detection (TSAD) models have been designed for this purpose. However, when applying such models directly to real-world cloud systems the anomaly detection performance is not as good. This is because the number of KPIs in real cloud systems is typically much more than the number of KPIs in the datasets used for model evaluation, and the larger number of KPIs bring about a performance loss of the models' anomaly detection. Therefore, selecting KPIs properly is essential for applying multivariant KPI data for any practical anomaly detection. This paper studies this performance loss issue when TSAD models are applied onto real-world cloud systems, and proposes fKPISelect, a mechanism of automated KPI selection based on fault injection. We implemented fKPISelect, deployed it to a real cloud system, and created a real-world KPI dataset. We conducted extensive experiments, and the experimental results show the effectiveness and practicality of fKPISelect: it improves the F1 score of anomaly detection from 0.68 to 0.91 for real-world KPI data.**

***Index Terms*—anomaly detection, cloud reliability, unsupervised learning, KPI, multivariant analysis**

I. Introduction

Nowadays many IT services are hosted in cloud platforms. Cloud services must be highly available as accidental failures of cloud services may cause revenue losses up to millions of dollars or higher to the service providers, cloud providers and/or service users [1].

There are a large number of various kinds of components involved in cloud services, including software programs, virtual machines (VMs), physical servers, network devices, etc. A common approach for detecting failures of the components is to monitor Key Performance Indicators (KPIs) of the components, such as CPU utilization, memory utilization, disk usage of the VMs or physical servers, network latency, response time and service throughput, etc.

Traditionally simple threshold-based approaches are applied to detect component failures, i.e. pre-defined threshold-based rules are created to monitor these KPI values and when certain KPI values exceed the threshold values a failure is detected[2, 3]. The approaches require users to have sophisticated expertise on the KPI values and carefully select the large number of different threshold values for so many components' various KPIs.

Recent years see the rapid adoption of Neural Network based artificial intelligence (AI), and employing deep learning models, such as multivariate Time Series Anomaly Detection (TSAD) models, for KPI-based anomaly detection is extensively studied [4–13]. Each vector of the time series data at a time point consists of multiple items with each item being the value of a KPI at that time. The number of KPIs is just the number of the vector's items or the vector dimension. Softwares like Prometheus[14] and Grafana[15] are now popularly deployed in cloud systems for collecting, monitoring and visualizing a huge amount of time series KPI data, and neural network AI models such as Variational AutoEncoder (VAE)[16], AutoEncoder (AE)[6] are trained to detect KPI anomalies. Specifically, the AI models process multivariant time series data [17, 18] and perform the tasks of reconstructing the data after the models are trained with the data collected during normal behavior; when the actual values largely deviate from reconstructed values (in terms of reconstruction loss values) an anomaly is detected.

However, although these models are demonstrated to be effective in experimental results reported in papers, applying them directly to real-world cloud services/systems does not result in as good anomaly detection performance as reported. Our experience of practically applying such models to our cloud system shows that, though their anomaly detection performances, in terms of F1 score (a metric combining both precision and recall of a detection capability), are reported to be around 0.80 or higher (0.92∼0.98 in the AnomalyTransformer model [9] and 0.79 in the USAD model [6]), the F1 score of applying them in our real cloud system is much smaller (0.68 and 0.28 for the AnomalyTransformer and USAD models, respectively, as shown in the "Node_data Full" row of Table III in Section VI).

We looked into the performance loss of anomaly detection. The number of KPIs in real cloud systems is typically much more than the number of KPIs in the datasets used for model evaluation. Popular datasets that most existing work is evaluated against are well-preprocessed, and the preprocessing includes KPI selection; as a result, the number of KPIs of the datasets is largely reduced. For example, the Server Machine Dataset (SMD) [5] has 38 KPIs, and the Course-To-Fine dataset (CTF) [10] has 49 KPIs. But real cloud systems may have far more KPIs. For example, Prometheus, a popularly deployed monitoring tool in cloud systems, provides

This work has been supported by the NSFC Project of China under Grant 62132009. Long Wang is the corresponding author.

2332-6549/23/$31.00 ©2023 IEEE
DOI 10.1109/ISSRE59848.2023.00084

various KPIs in the categories of CPU, Memory, disk, network, etc., and the number of KPIs Prometheus provides in our cloud system reaches 493. The large number of KPIs bring about much noise which undermines the models' performance (see in-depth discussion in Section III).

As a result, the practical experience of applying multivariate time series anomaly detection models onto real-world systems, e.g. [10, 11], employs a manual selection of KPIs as part of data preprocessing. However, the manual KPI selection demands operators to have rich experience and expertise in service/system behavior. Open datasets of multivariant KPI time series data do not help operators much in the manual KPI selection task. It is because currently such open datasets do not provide KPI name or detail information, i.e. what system property (like CPU utilization, memory utilization, or something else) each item of a vector in the dataset specifies. Moreover, a KPI item in an open dataset may not be exactly the same as one collected from the practical cloud system. So the operators are unable to know which ones of the large number of KPIs in the practical system correspond to the small set of KPIs in the datasets.

Therefore, KPI selection is an essential task in applying multivariant KPI time series data for any practical anomaly detection. This paper studies the problem of *KPI selection for practical KPI-based multivariate anomaly detection in cloud systems*. As far as we know, there are few, if not none, prior work that investigated this essential problem in Multivariate Time Series Anomaly Detection (TSAD). This paper focuses on studying the limitations of current KPI-based anomaly detection approaches. Now almost all KPI-based anomaly detection uses unsupervised models. So supervised models for KPI-based anomaly detection is not discussed here.

In this paper, we propose fKPISelect, a fault-injection-based automated KPI selection mechanism, after studying how the noise brought by a large number of KPIs worsens the anomaly detection performance of TSAD models. We inject errors such as network packet loss and high CPU consumption, but we use the *fault injection* terminology in this paper as it has been widely adopted for traditional reasons. Specifically, the fKPISelect methodology performs the following steps:

i) The KPI data of the cloud system are collected.

ii) A set of predefined types of errors are injected into a small set of components in a test environment of the cloud system, and the KPI data associated with the set of components are collected. Note that these data include those collected when an injected error is present and others collected when no error is present, i.e., before the fault injection or after the injected error is corrected. These KPI data are labeled as error-present or normal correspondingly.

iii) For each KPI item we use the time series data of only this KPI item collected in step i (i.e. the KPI vector dimension is 1) to train TSAD models and use the data of only this KPI item collected in step ii as the testing dataset to evaluate how much this KPI item is sensitive to the predefined error types. Though the TSAD models are typically unsupervised models, we exploit the labels in the testing dataset to evaluate the models' anomaly detection performance (precision, recall, or F1 score), and compute the KPI item's sensitivity value according to this performance. Details of the sensitivity computation are available in Section IV-A.

iv) After all KPI items' sensitivity values are computed those KPI items with values higher than a threshold are selected.

Then the production time series data of only the selected KPI items (production data are not involved in steps i or ii) are fed to the TSAD models for online anomaly detection.

A straightforward idea of reducing the number of KPIs is to leverage common dimensionality reduction technologies such as clustering algorithms. But our experience shows that the clustering of KPI data collected during normal behavior does not help much: the F1 score is 0.71 as shown in the "cluster" row of Table V in Section VI (when AnomalyTransformer is used). Note that the F1 score is 0.68 when all KPI data are used. When fault injection is conducted and error-present KPI data are employed for KPI selection, the anomaly detection's F1 score increases to 0.91 in the "fKPISelect" row of Table V.

This clearly demonstrates that error-present KPI data obtained via fault injection are critical for the KPI selection. It is because similarities of two KPIs during normal behavior do not mean the two KPIs still behave similarly when there is an error. We believe general knowledge of abnormal behavior of the system (like the error-present KPI data) largely boosts the performance of anomaly detection, as pointed out by [19]. So for practical anomaly detection in real-world cloud systems, we leverage fault injection to gain the knowledge, which is implied by the sensitivities of the selected KPIs. In summary, this paper's contributions are listed as follows:

- To the best of our knowledge, we are the first to investigate the issue of KPI selection in multivariate TSAD and point out the necessity of it for practical multivariate TSAD. We propose fKPISelect, a mechanism of automated KPI selection based on knowledge learned by means of fault injection.
- We investigated the performance loss issue of multivariate TSAD models when they are applied to real-world cloud systems, in particular when there are a large number of KPIs. Besides the experimental results that demonstrate the effectiveness of our fKPISelect (Table III), we conducted a theoretical analysis of the performance loss issue in Section III.
- We implemented fKPISelect, deployed it to a real cloud system, and created a real-world KPI dataset. The dataset containing the metadata of the KPIs for future KPI-engineering research and the fault injection tool are available at https://github.com/THUzxj/fKPISelect.
- We conducted a series of experiments on both open datasets and the real-world dataset. The experimental results show the F1 score of the anomaly detection with full KPI data is 0.68, the score with manual KPI selection is 0.75, and the score with fKPISelect is 0.91 (Table III) for the real-world KPI data (Node_data). The results clearly demonstrate the fKPISelect's effectiveness in improving the anomaly detection performance and making TSAD models practical for real systems.

II. Gap Between Existing Datasets and Data in Practice

We find a gap between existing open datasets and the KPI data in practical systems, which makes it difficult to select the most valuable KPIs for anomaly detection based on existing practice.

The gap includes three aspects: the number of KPIs, the type of KPIs, and the relationships between KPIs and errors.

The datasets that most existing works are evaluated on are well-preprocessed, including the KPI selection, standardization, *etc.*. SMD (Server Machine Dataset) is a dataset of 28 servers, and each server has 38 KPIs[5]. CTF published a dataset of 533 servers, and each server has 49 KPIs, including CPU (15 KPIs), Memory (10 KPIs), Sockets (6 KPIs), UDP (7 KPIs), TCP (11 KPIs)[10]. Some private datasets are also described in previous works. TC_data is a dataset of over 500 entities, and each entity has 11 KPIs such as CPU usage, memory usage, and network speed[11]. FluxRank uses 47 types of KPIs, including CPU (8 KPIs), Disk (15 KPIs), Memory (6 KPIs), Network (13 KPIs), and OS kernel (5 KPIs)[20].

However, current monitoring tools provide a huge number of KPIs, considering the complexity and diversity of hosts and services in systems. One entity may contain multiple components and each component may have multiple KPI groups providing different metrics. Each KPI group may have multiple parameters, such as network interfaces for network KPIs and CPU cores for CPU KPIs. Taking host monitoring as an example, the agent *node exporter*[21] of Prometheus[14] exports a series of OS and hardware metrics exposed by OS kernels, and the dashboard *node exporter full*[22] of Grafana[15] performs queries to Prometheus and visualize the time series for operation engineers. By default, *node exporter* provides various KPIs of Network, CPU, Memory, and *etc.*, as shown in Table I. In our scenario of monitoring servers in a cloud system, the number of KPIs reaches 493. The comparison between existing datasets and the data in our practice is shown in Table II.

TABLE I: The categories of KPIs and the numbers of KPIs in the categories provided by *node exporter* for virtual machines in our practice

Component	Examples of KPI groups	Number of KPIs
basic	CPU / Memory / Net / Disk	37
Network	Traffic (Packets, Errors, Drops, *etc.*)	145
	Sockstat (TCP, UDP, FRAG/RAW, *etc.*)	14
	Netstat (IN/OUT, Forwarding, ICMP, TCP, UDP, *etc.*)	27
CPU	Usage, Softnet	9
Memory	Basic, RAM Total	8
	Physical (Active/Inactive, Writeback, Committed, Dirty, *etc.*)	35
	Virtual (Pages In/Out, Page faults, *etc.*)	8
Storage	Basic, RootFS, SWAP Total	8
	Disk (IOps, R/W Data, *etc.*)	15
	Filesystem (Space, Error, Node size, *etc.*)	35
System	Processes (Status, Schedule, *etc.*)	10
	Misc (Context Switch/Interrupts, Load, Schedule Timeslices, *etc.*)	11
	Systemd	26
	Time Sync	8
Hardware	Temperature, Cooling, Power	4
Node Exporter	Scrape Time	93
Others	Uptime	1

TABLE II: Comparison of KPIs' number and descriptions in open datasets and research works with the data in practice

Dataset	KPI Num	KPI Description in Dataset
SMD[5]	38	CPU, network and memory usage, *etc.*
CTF[10]	49	CPU, memory, sockets, UDP, TCP
TC_data[11]	11	CPU usage, memory usage, and network speed, *etc.*
FluxRank[20]	47	CPU, Disk, Memory, Network, and OS kernel
Node Exporter	493	All KPIs provided by node exporter

Thus, the first aspect of the gap is that the whole number of KPIs is far more than the number in existing datasets. If the entire data are applied to the existing multivariate TSAD models, the scalability of the models on the KPI dimension will be a challenge, which will be discussed in Section III.

The second aspect of the gap is that the necessary metadata of KPIs, such as name, parameter, and unit, are usually missing in existing open datasets. Most previous work only focuses on refining models for mining information from given time series datasets. The KPI selection for creating the existing datasets was made by researchers or engineers manually based on their experience, with the selection procedure or standard not clearly explained. Furthermore, considering the variety of workloads and components/devices in different systems, there is a high chance that there are not exactly the same KPIs across different systems. So existing open datasets do not help much for the KPI selection task in applying TSAD models to a real system.

The third aspect of the gap is that most existing open datasets do not provide information on relationships between errors and KPIs. So it is not clear which KPIs are sensitive to what types of errors. Then, given a list of specified types of errors, these existing datasets do not provide guidance on which KPIs are more useful for the detection.

III. Performance Loss of Multivariate TSAD with Many KPIs

We encountered a phenomenon in our practical experience: current multivariate TSAD models have a performance loss when there are a huge number of KPIs. Current multivariate TSAD models were reported to work well on open datasets [6, 9]. However, in our practice, the models' accuracy decreases when the number of KPIs increases.

A. Multivariate TSAD

The multivariate Time Series Anomaly Detection problem is that, given the multivariate time series in the normal state as the training input, and the multivariate time series in uncertain (normal or anomalous) status as the testing input, the model should judge whether the time series is anomalous at each time point in the testing input. Rather than using univariate methods and training and maintaining a model for each KPI, we use multivariate methods for the following reasons. First, the multivariate TSAD methods can reconstruct the time series with more information of the KPIs of the entities and model the overall status of entities. Second, considering the large number of KPIs, training and maintaining a model for each KPI has a much higher overhead. Third, the relation between KPIs and the relation between KPIs and anomalies are complex and then we should define rules to determine the anomalies in the entity level from the anomalies from the KPI level[5].

Current multivariate and unsupervised TSAD models learn the features of time series from the normal time series, then distinguish anomalous time series that are different from the normal ones. As shown in Figure 1, these models employ reconstruction-based unsupervised methods. These methods are based on autoencoders (AE) or variational autoencoders (VAE) which consist of an encoder and a decoder. The specific neural network models of the autoencoders can be MLP, LSTM, or Transformer. All KPIs are normalized into the same scale by

means of methods like min-max scale or standard scale before the KPIs are fed to the autoencoder models (see Figure 1). The encoder maps the input into a latent vector Z, and the decoder decodes Z back to a reconstruction $\hat{W}_t$.

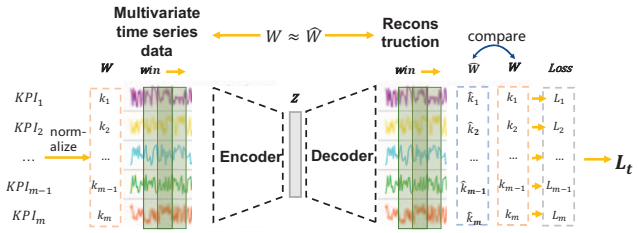

Fig. 1: The structure of a Multivariate TSAD model

The main training objective of these models is to let the reconstructed data be as close as possible to the original data by minimizing the reconstruction loss[6, 8, 9, 11–13]. The reconstruction loss of multivariate time series L_t at time t is the average of the reconstruction loss of each KPI:

$$\mathrm{L}_t = \frac{1}{m}\sum_{i=1}^{m} \mathrm{L}_{i,t} \tag{1}$$

where the number of KPIs is m, the reconstruction loss of each KPI $\mathrm{L}_{i,t}$ is:

$$\mathrm{L}_{i,t} = \|\mathbf{k}_{i,t} - \hat{\mathbf{k}}_{i,t}\|_2^2 \tag{2}$$

where $\|\cdot\|_2$ denotes the L2-norm, the origin data $\mathbf{k}_{i,t}$ is the origin data of the i-th KPI's input window at time t and $\hat{\mathbf{k}}_{i,t}$ is the corresponding reconstructed data.

In the predicting phase, the reconstruction loss is used as the anomaly score. The time points with higher scores than the threshold are considered anomalies. It is based on the assumption that the autoencoder can only reconstruct the encountered data patterns and will reconstruct worse results when encountering abnormal data patterns.

B. Performance Loss of Anomaly Detection with Many KPIs

We conducted experiments to study the performance loss phenomenon as the number of KPIs increases. We created a synthetic dataset that mimics the situation in which one KPI responds to an error, and other KPIs remain normal when the error is present, and let the AnomalyTransformer model processes the dataset (i.e. part of the dataset for training and the rest for testing) for detecting the anomaly.

As shown in Figure 2 (the horizontal axis is time), this dataset contains m time series, and each time series are samples of

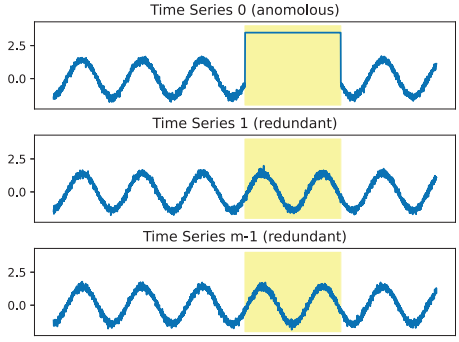

Fig. 2: Schematic diagram of the synthesized time series dataset

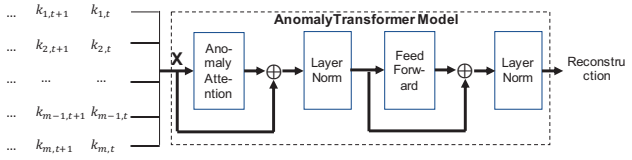

Fig. 3: How the multiple KPIs are fed into the AnomalyTransformer model as input in our experiments

a sine wave ($sin(t)$, t is the time), with noises in the normal distribution added to the samples. Noises in KPI time series data are inevitable in real systems. Then one of the m time series is injected with data to mimic a period of anomaly (yellow period in the figure), as indicated by the *Time Series 0* in the figure. All the other time series (from 1 to $m-1$, called *redundant* time series) remain normal all the time, i.e. they all show a sine wave with different noise values added. Note that *Time Series 0* here is a simplified example to show an evident error for the purpose of clearly explaining the performance loss issue.

Our experiments employ the Multivariate TSAD setup shown in Figure 1. How the KPIs are fed into the model is illustrated in Figure 3. $k_{i,t}, k_{i,t+1}$, ... is the stream of the normalized KPI k_i ($1 \le i \le m$). The streams of all m KPIs within a time window are combined as a matrix X and fed to the AnomalyTransformer model as the input. Figure 3 also shows AnomalyTransformer's structure (the encoder-decoder in Figure 1). More details on the model's handling of the input X can be found in [9].

We ran multiple experiments with m varying from 2 to 64. The AnomalyTransformer model sizes in the experiments, in terms of the number of parameters (weights), are different as the encoder's input layer and the decoder's output layer have the sizes associated with the KPI number (other layers have same sizes for all m): the encoder and decoder's sizes are 4753432 and 1026 for m=2, and 4848664 and 32832 for m=64, respectively. The training set has 9901 samples for all experiments.

Figure 4 presents the anomaly scores obtained in our experiments. We can see that when the number of KPIs of the dataset, m, is 2, the anomalies can be easily detected from the anomaly scores (the top picture of Figure 4). However, when the number of KPIs increases to 16 and 64, the anomaly scores during the error-present period become not so outstanding both in terms of absolute values and comparisons with other values, and there are much larger fluctuations of the anomaly scores caused by the accumulation of so many KPIs' noises. As a result, the number of false positives (red lines in the figure) increases when the number of redundant KPIs increases. Similar experimental results are also observed for the USAD model. We also tried experiments with larger models and larger training sets, which demonstrated the similar performance loss.

C. Explanation

The anomaly detection task is different from traditional deep learning tasks like regression and classification, whose prediction is consistent with their training objective. In the anomaly detection task, the anomaly scores are calculated as the reconstruction loss of the model output indirectly, instead of the direct output of the deep learning model.

On one hand, the anomaly scores can only be calculated as the averages of the reconstruction loss values of all KPIs (equation

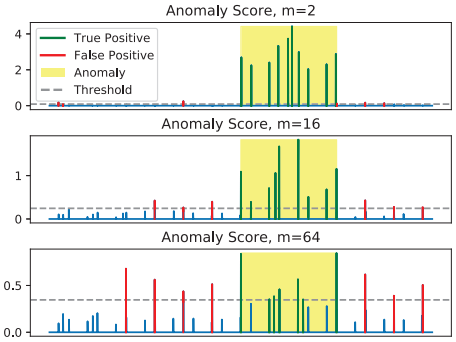

Fig. 4: Anomaly Detection results of AnomalyTransformer with different redundant KPI numbers m. The dash lines indicate the thresholds of the reconstruction loss for anomaly detection. The scale of the score (the vertical axis) decreases as m increases because of the definition of the reconstruction loss in formula (1). With more KPIs the denominator increases while the numerator does not increase proportionally (most KPIs have low reconstruction loss).

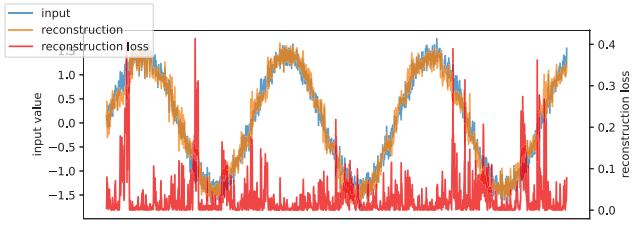

Fig. 5: Reconstruction and reconstruction loss of one sine time series with noises by AnomalyTransformer

1). Existing unsupervised learning-based models assume that the user does not have anomaly data in the training phase because of the scarcity of anomaly data and the difficulty of labeling anomalies[6, 9]. Thus the models are only trained with the data with normal data and not instructed to place attention onto specific KPIs. As a result, the models do not know the importance of individual KPIs for anomaly detection.

On the other hand, deviations exist in the reconstruction loss of each KPI due to the existence of noises. Ideally, the reconstruction loss should be very low, near zero, when reconstructing normal data whose patterns have been seen by the autoencoder. However, many KPIs have high-frequency noises in random distributions. We let the AnomalyTransformer model process time series data with a single KPI, the sine curve, and added noises, and obtained the model's reconstruction of the data as well as the reconstruction loss at every time point. No error is injected in this process. Figure 5 charts the input time series data, the reconstruction of the data, and the reconstruction losses along the time. The figure shows the reconstruction loss fluctuates a lot and sometimes has quite high values, and note that there is only a single KPI in this figure. The result shows the noises are hard for the autoencoder-based multivariate TSAD models to reconstruct precisely, even when reconstructing the time series in the normal state. As noises are inevitable in KPIs of practical systems, the deviations of the reconstruction loss of KPIs are also inevitable.

When the number of KPIs is not so large, certain existing models are still reported to work well[6, 9] with deviations in reconstruction losses. This is because the change of reconstruction loss in an anomaly state is significantly greater than the fluctuation range of the deviations when the number of KPIs is small. As the number of KPIs increases, the deviations are accumulated by adding up all KPI's reconstruction loss values and finally go beyond the anomaly-caused reconstruction changes. Assuming the deviations n_i are in the normal distribution, $n_i \sim N(\mu_i, \sigma_i^2)$, the sum of the deviations $n = \sum_{i=0}^{m} n_i$ is also in the normal distribution,

$$n = \sum_{i=0}^{m} n_i \sim N(\sum_{i=0}^{m} \mu_i, \sum_{i=0}^{m} \sigma_i^2) \tag{3}$$

which has an accumulated variance. With the accumulated deviation comparable with or larger than the anomaly score caused by errors, the models fail to detect the anomalies as accurately as in the cases with a limited number of KPIs.

A straightforward way to deal with this performance loss is to design an anomaly detection model for each KPI and vote among them. However, this design may result in hundreds of neural models and is not practical at all. Moreover, KPI correlations may be more complicated than a simple voting of multiple KPIs' anomaly detection results. A single neural model may better exploit such correlations. Therefore, properly selecting KPIs for anomaly detection is necessary to ensure TSAD models' performance.

This paper focuses on one particular issue of current KPI-based anomaly detection approaches, KPI selection, and we studied it in normal-load behavior. Intermittent heavy loads may result in KPI values drastically different from their normal values, and such situations may be flagged as anomalies incorrectly. Continuous machine learning may help deal with such heavy-load caused false positives, but it is not in this paper's scope.

IV. Proposed Method

This section introduces the proposed Fault-Injection-Based Automatic Selection of KPIs (fKPISelect), shown in Figure 6. fKPISelect proposes a flow path of KPI selection from predefined error types to the error-sensitive KPI (the KPIs that respond to the errors) list for anomaly detection, solving the gap between existing datasets (Section II) and letting the multivariate TSAD models avoid the performance loss when dealing with many KPIs (Section III). fKPISelect leverages fault injection to gain knowledge of sensitive KPIs to specified errors and anomaly detection models to measure the sensitivity of KPIs. fKPISelect is compatible with existing multivariate TSAD methods, replacing human work with automated work in the data processing of anomaly detection models.

A. Criterion Measuring Sensitivity of KPIs

The sensitivity of KPI is defined as whether the KPIs respond to the errors. Different from examining the huge amount of KPIs by humans through visualization, a quantitative criterion measuring the sensitivity of KPIs is required for automatic selection. However, directly accessing KPIs' sensitivity to the errors is a complex problem. Thus we use fault injection and convert the criterion *Is the KPI sensitive to the errors?* to the question *Does the KPI behave anomalously when the errors occur?*. Then as anomaly detection models are a ready tool to judge whether there are anomalies, the question is converted to

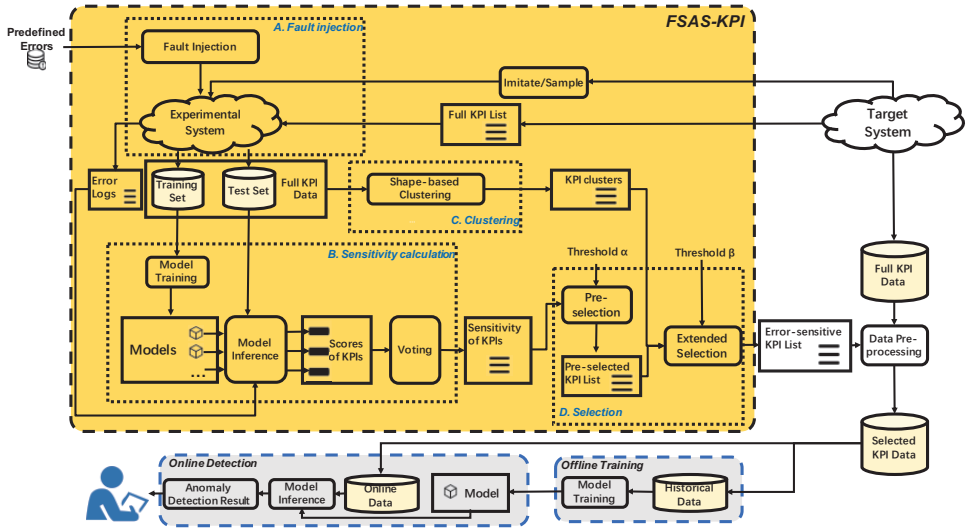

Fig. 6: The overview of fKPISelect and its interaction with anomaly detection system

Do the anomaly detection models recognize the anomalies in the KPI when the errors occur?.

Unlike existing methods like clustering[23] that purely reduce KPI numbers, fKPISelect focuses on reducing KPI types. Fault injection is used for inspecting the relationship between errors and KPIs [24]. Univariate anomaly detection models are used to judge whether there are anomalies in each KPI when the errors occur. Because the univariate anomaly detection models can not accurately recognize the anomaly every time for the randomness of the model and the KPI data, to achieve more accurate sensitivity calculation, fault injection should be performed several times with the same or different configurations for one error type. After getting the predicted labels of the models, whether the error events have been detected by the model is determined by whether there is at least one positive label during each error event. Thus the sensitivity of a KPI is defined as the recall value of the error events,

$$\text{sensitivity}_i = \frac{num_{hit\_errors,i}}{num_{errors}} \tag{4}$$

where $num_{hit\_errors,i}$ is the number of errors correctly predicted by the model for i-th KPI, and num_{errors} is the total number of injected errors. Then we can select KPIs with higher sensitivity values by setting a threshold.

Other potential choices of the sensitivity of KPIs include the F1 score of the models (a widely used evaluation measure for classification tasks that is the harmonic mean of precision and recall) and the output anomaly scores by the models. However, compared with the F1 score that has various definitions with adjustment for time series[25], recall of the error events enables us to further apply the voting mechanism (discussed in Section IV-B2). And as the number of positive points is equal when we fix the anomaly ratio, a higher recall roughly leads to a higher F1 score. Moreover, anomaly scores rely on anomaly detection models and vary in different KPIs, thus can not be used for comparison. So, using recall is reasonable and practical.

B. fKPISelect Mechanism

1) Fault Injection: As shown in part A in Figure 6, first, a set of concerned types of errors should be specified as the input of fKPISelect. Fault injection tools[26–28] are used to inject various types of errors into services and servers. Then, plan several error configurations in detail for each specified error type. The error configurations include intensity, target, random changes, *etc.*, varying in different error types. After that, perform the fault injections in the experimental system that imitates the target system or is sampled from the target system and has similar workloads with the target. The fault injections are organized in fault injection campaigns [29]. Each fault injection lasts for a period of time to wait for the spread of the errors. The fault injection logs from tools are recorded to label the anomalies. Finally, collect the KPI data during the fault injection for the test set and the error labels within the error logs. An example of how the fault injections are planned and performed in our practice is introduced in Section VI-B.

2) KPI Sensitivity Calculation: As shown in part B in Figure 6, we calculate the KPI sensitivity and perform KPI selection on it. Univariate anomaly detection models are trained with the normal data in the training set collected in the experimental system for each KPI. Then the test data are input to perform the model inference. The output anomaly labels of the models and the error labels from error logs are used to calculate the sensitivity introduced in Section IV-A. Instead of finding the best threshold for each model, we fix the anomaly ratio of anomaly scores to one global value across different KPIs for comparison of sensitivity.

As fKPISelect uses the anomaly detection results of the injected errors for KPI selection, the accuracy of KPI sensitivity depends on the performance of the anomaly detection models, thus the inaccuracy of the models will lead to the inaccuracy of sensitivity calculation. One type of the inaccuracy is false negative: a KPI responds to the error, but the anomaly detection model fails to flag the anomaly. In this situation the KPI's response is not intense enough for this anomaly detection model. The other type is false positive: during an error, a KPI doesn't respond to the error, but the model flags an anomaly falsely. This leads us to wrongly select an insensitive KPI as a sensitive one. Note that the *false negative* and *false positive* terms used here

are for the KPI selection, and are not for analyzing final anomaly detection results.

Inspired by ensemble learning[30], which uses multiple algorithms to reach a better performance than one alone algorithm, we apply a voting mechanism to identify the detected anomalies during the error events. We use multiple models to give a result of the detected anomalies during the error events separately. If over half the number of the models recognize the anomaly in one KPI during one error event, we decide that the models recognize the anomaly in this KPI during the error event, otherwise, the models don't recognize the anomaly in this KPI during the error event. Finally, the voted results lead to better KPI selection.

3) KPI Clustering: One more problem is the false ignorance of possibly error-sensitive KPIs. Although fault injections can efficiently reveal the KPIs related to the injected errors, we can only perform part of the error configurations of one specified error type. Directly selecting KPIs for the performed fault injection configurations leads to a risk of ignoring error-sensitive KPIs that are also relative to the error type but not tested, which may reduce the generalization performance of the anomaly detection model.

To address this challenge, when we have selected some KPIs, KPIs that have similar properties to the selected KPIs are also should be selected. To find the KPIs with similar properties, selecting the KPIs which have similar shapes to them should also be selected. Because some KPIs with similar properties are similar in shape as the same workload of the entity influences them. A shape-based clustering[23] method is used to find similar KPIs, as shown in part C in Figure 6. Then, the cluster results will be used for an extended selection of KPIs in the next step. If the data has information on how KPIs are clustered, predefined clusters can also be input to the cluster method.

The shape-based distance of 2 KPIs is the max normalized cross-correlation of the selected windows of 2 KPIs. HDBSCAN is adopted for it only depends on the input distance matrix and parameter *minPts* (the minimum number of points within one cluster).

4) KPI Selection: As shown in part D in Figure 6, the preselection and extended selection are done to generate the final error-sensitive KPI list. First, a threshold of sensitivities of KPIs α is defined, and the KPIs with sensitivities higher than the threshold are selected. Then, a selecting ratio of each cluster is calculated as the proportion of the selected KPIs to the total KPIs in the cluster. A threshold of selecting ratio β is set and the KPIs in clusters that have higher selecting ratios are added to the extended list of KPIs.

The intuition behind the two thresholds is below. We first select a set A of KPIs based on fault injection results by using the threshold α, and then select another set B whose properties (curve shapes) are sufficiently resembling those in A by using the threshold β. The final selected KPIs are A+B.

5) Model Training and Online Anomaly Detection: As shown in Figure 6, with the error-sensitive KPI list applied to the data processing, users can train the multivariate TSAD model on the selected KPI data and apply the model to online anomaly detection. The knowledge of selected KPIs is directly transferred from the experimental systems to the production systems in our practice because the running logic and KPI types are consistent across the experimental system and the target.

V. Implementation

A fault injection tool is implemented with Ansible (An IT Automation tool)[31] as the controller and `tc` (traffic control)[26] and `stress-ng`[27] as the executors. The controller manages the fault injection campaign, generates fault injection commands, and sends them to executors. The executors execute the commands in the target machines.

fKPISelect is implemented based on PyTorch[32] and Numpy[33]. USAD[6] and AnomalyTransformer[9] were used in fKPISelect to calculate the sensitivity of KPIs, then the two models voted for the selected KPIs.

VI. Evaluation

A. Experimental Setup

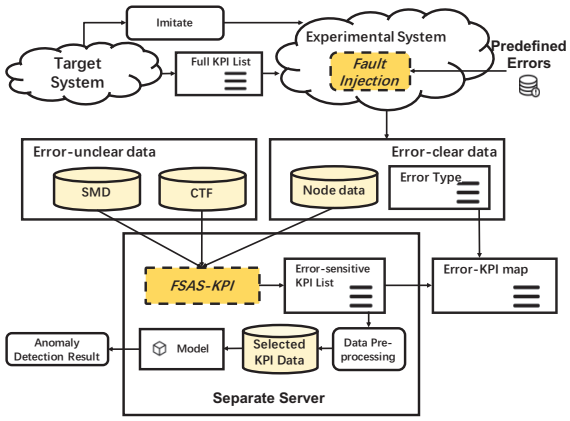

Fig. 7: The overview of the Experimental Setup

As shown in Figure 7, first, we build an experimental system that imitates the target system. Second, we perform fault injection in the experimental system and collected the data. Then fKPISelect is deployed to select error-sensitive KPIs. Subsequently, we use the error-sensitive KPI list to preprocess the dataset. Finally, the data preprocessed is used for anomaly detection and the anomaly detection results were analyzed. Experiments are performed on SMD, CTF, and Node_data (introduced in Section VI-B) separately.

In our scenario, the target system is a production cloud system containing thousands of nodes. However, labeling the data collected from the target system is painstaking, and at the same time, there are too few anomaly samples in the target system's data. Therefore, we build an experimental system with real workloads, and we monitor the experimental system using the same KPIs as in the target system. The experimental system is a 5-node distributed system with a container cloud platform Kubernetes[34] cluster deployed on it. Each node has 8 logical CPU cores and 16GB memory with the Ubuntu 20.04 LTS. We generate workloads on the experimental system similar to the target system. Sock-shop microservice [35] is deployed. We use the load testing tool Locust[36] running out of the experimental system to manage the users with the number changes among 50-150 periodically and 100 queries per second to send HTTP requests to Sock-shop. We use Prometheus[14] for service and

host metric collection. To maintain consistency with the target system, we employed the *node exporter* [21] software to collect data for the same set of KPIs.

The KPI selection and Anomaly detection experiments were performed on the Separate Server. The server has a GPU of GeForce RTX 3090, a CPU of Intel Xeon Silver 4214R 2.4G, which has 48 logical CPU cores, and 188GB memory with the Ubuntu 22.04 LTS Linux system.

Here we describe the hyperparameters for USAD and AnomalyTransformer training. For both models, the number of training epochs is 10 and the window size is 5. For the USAD model, the batch size is 10000. For the AnomalyTransformer model, the batch size is 256. During training, an early stop strategy is applied according to the validation loss at the end of each epoch.

B. Datasets

As shown in Figure 7, we used Node_data, SMD, and CTF datasets in the experiments. The existing datasets, SMD and CTF, are introduced in Section II, while the real-world dataset Node_data is introduced here.

The data collected from the experimental system with fault injection were called Node_data, which is a dataset that has 493 KPIs. The training set and testing set both contain 7 days of KPI data from 5 machines with an exporting interval of 15s. We injected four types of anomalies that widely exist in common computing systems[37], described below:

Network Anomalies. Network anomalies, including network packet delay, network packet loss, and network packet duplicate, are injected by `tc` to emulate the s of network interfaces, the operating systems, and the interconnection of the network.

High CPU consumption. Anomalous high CPU consumptions are injected by `stress-ng` to emulate the situation that anomalous programs fell into infinite loops, busy waits, or deadlocks.

Memory leaks. Anomalous memory leaks are injected by `stress-ng` to emulate the situation that allocated chunks of memory are not freed after use and then the accumulation of memory leaks causes memory shortage and system failures.

Anomalous number of disk access. Anomalous high number of disk accesses is injected by `stress-ng` to emulate the situation that a high number of disk retries is caused by disk access failures or a high number of disk accesses from anomalous programs.

Each fault injection campaign contains one fault injection experiment. In each fault injection campaign, the workload runs for 40 minutes. The error is injected in the period between 20 minutes and 23 minutes. As the anomalies are rare in practical systems, to ensure that the detection of the errors is not influenced by the near errors, a 37-minute gap between two injections can successfully prevent models with high window sizes such as 100 (the duration of a window is 100×15 seconds or 25 minutes).

In our evaluation, both KPI selection and anomaly detection require datasets. The datasets for KPI selection and for anomaly detection should not overlap. Therefore, we split a dataset into two parts, one for KPI selection and the other for anomaly detection, accounting for 30% and 70% respectively. As shown in Figure 8, a dataset consists of time series data of n nodes, we divide the original training set and test set according to the time, and then recombine the training set and the test set, T_0+T_2 time period data for KPI selection, and T_1+T_3 time period data anomaly detection.

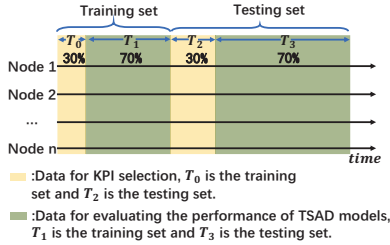

Fig. 8: Dataset segmentation for KPI selection and performance evaluation

C. Evaluation Measures

The measure to evaluate the performance of TSAD models should be defined because of the special properties of time series. The anomaly detection models give an anomaly label to each time point. Anomaly detection can be treated as a classification task and evaluated by precision, recall, and F1 score. Precision measures the proportion of true positive predictions out of all positive predictions, recall measures the proportion of true positive predictions out of all actual positive instances, and F1 score is the harmonic mean of precision and recall. Previous works widely use the Point Adjustment strategy: if any point in an anomaly segment in the ground truth can be detected by a chosen threshold, then say this segment is detected correctly, and all points in this segment are treated as if they can be detected by this threshold[5, 6, 9, 38]. For threshold selection, we don't check all possible thresholds and search for the best F1 score because in practice there are no ground truth labels for searching for the best result. Instead, we select a suitable fixed anomaly ratio for each model.

D. Comparisons

We conduct a series of experiments on SMD, CTF, and Node_data. We first apply fKPISelect (using both USAD and AnomalyTransformer for voting) to preprocess each dataset, and then employ USAD or AnomalyTransformer for anomaly detection on the preprocessed data. We do not apply other filtering of KPIs such as pruning of constant-value KPIs. Because certain constant-value KPIs may indicate errors, e.g. "Network Traffic Errors, Network Traffic Drop, TCP Errors" in our cloud's KPIs remain as 0 for a long time and their value changes indicate certain errors. So, directly pruning them away automatically may not be good.

Table III presents the performance of anomaly detection using USAD and AnomalyTransformer on SMD, CTF, and Node_data datasets after different data preprocessing techniques. The threshold α is 0.08 and the threshold β is 0.80 in the experiments. For each dataset, we compared the performance of using all KPIs (Full), manually selected KPIs (Manual), and KPIs automatically selected through fKPISelect preprocessing. For the Node_data dataset in Table III, the KPIs associated with the *Full* preprocessing are all of the 493 KPIs directly collected from our cloud system via Prometheus. fKPISelect selected 49 KPIs from these 493 KPIs. To compare fKPISelect with manual KPI selection,

TABLE III: Comparison of Anomaly Detection Performance

Dataset	Preprocess	USAD			AnomalyTransformer		
		Precision	Recall	F1 score	Precision	Recall	F1 score
Node_data	Full	0.3847	0.4088	0.2756	0.5625	0.8571	0.6792
	Manual	0.4610	0.7976	0.5843	0.6296	0.9285	0.7504
	fKPISelect	**0.4702**	**0.8069**	**0.5942**	**0.8461**	**0.9897**	**0.9123**
SMD	Full	0.6509	0.6533	0.6521	0.9414	0.8901	0.9151
	Manual	**0.6800**	0.7100	0.6946	**0.9424**	0.8354	0.8856
	fKPISelect	0.6082	**0.8110**	**0.6951**	0.9421	**0.9247**	**0.9338**
CTF	Full	0.2976	0.3439	0.3190	0.9047	1.0000	0.9500
	Manual	0.2976	0.3440	0.3191	0.9051	1.0000	0.9502
	fKPISelect	**0.4534**	**0.5598**	**0.5010**	**0.9070**	**1.0000**	**0.9646**

TABLE IV: Comparison of the time and space cost

Dataset	Preprocess	KPI Numbers	USAD			AnomalyTransformer		
			Training time (sec)	Detection time per sample (sec)	Model size (KB)	Training time (sec)	Detection time per sample (sec)	Model size (KB)
Node_data	Full	493	292.75	8.54E-3	45250.67	361.34	1.56E-2	32525.59
	fKPISelect	49	59.11	5.16E-4	518.92	63.28	1.47E-3	28971.91

experts select KPIs based on the visualization of the KPIs in the training set and the test set, and the KPI metadata (only in Node_data). We kept the number of manually selected KPIs consistent with the number of automatically selected KPIs.

Node_data, containing a larger number of KPIs than other datasets, exhibited the most significant performance improvement after KPI selection. Compared to using all KPIs for anomaly detection, with fKPISelect preprocessing, the F1 score is improved from 0.28 (0.2756 in Table III) to 0.59 for USAD and from 0.68 to 0.91 for AnomalyTransformer. Although SMD and CTF datasets were preprocessed before being published, applying fKPISelect preprocessing still lead to performance improvements.

We also ran experiments of random KPI selection on the SMD dataset while using the AnomalyTransformer model. The random selection achieved the 0.82 F1-score, much worse than the three selections compared in Table III) (all over 0.88).

Table IV presents a notable decrease in the time and space cost of USAD and AnomalyTransformer after applying fKPISelect. The Node_data dataset is used for evaluation, with two preprocessing approaches: Full, representing the use of all 493 KPIs, and fKPISelect, where 49 selected KPIs are utilized. The Training Time (sec) refers to the duration required for model training. The Detection time per sample (sec) represents the time that the anomaly detection model process each individual data sample. Model size (KB) denotes the storage occupied by the models. The reduction of the input number of KPIs leads to a smaller model size and shorter training and predicting time. The USAD's model size is reduced by 98.85% (from 45250.67 to 518.92) because USAD leverages full connect layers with the input and output shape proportional to the number of KPIs.

E. Ablation Study

To validate the effectiveness of our fKPISelect design, we conducted a series of experiments on the Node_data dataset, including: (i) cluster: Using the straightforward method of clustering to reduce the number of KPIs, as introduced in Section I. (ii) -vote: Removing the voting mechanism. (iii) -cluster: Removing the clustering mechanism. (iv) -vote,F1: Using F1 score as the sensitivity of KPIs, at the same time removing the voting mechanism because of incompatibility.

The Node_data dataset can be clustered into 8 classes by their shapes, in experiment (i), we randomly select 6 KPIs in each cluster. In experiments (ii) and (iv), the AnomalyTransformer model was used to calculate the Sensitivity of KPIs. In experiment (iii), KPIs were directly selected based on their Sensitivity rankings without extended selection. The KPI numbers of the experiments (ii), (iii) and (iv) are all 49.

As shown in Table V, the experiment that uses the straightforward method of clustering to reduce the number of KPIs has the lowest F1 score, since the algorithm only clusters KPIs and cannot select useful KPIs for anomaly detection. The removal of the voting mechanism (-vote) has a greater impact than the clustering mechanism (-cluster), which demonstrates that the voting mechanism plays a more important role in KPI selection, and the anomaly detection ability of different models is a more important factor to consider. The result of -vote, F1 shows that using F1 score as the sensitivity has a similar effect as using recall, but using F1 score is incompatible with the voting mechanism.

TABLE V: Anomaly Detection Performance of different design of fKPISelect

Preprocess	USAD			AnomalyTransformer		
	Precision	Recall	F1 score	Precision	Recall	F1 score
cluster	**0.4873**	0.2765	0.3528	0.6101	0.8571	0.7128
-vote	0.4264	0.4795	0.4514	0.6605	0.8861	0.7569
-cluster	0.4864	0.6222	0.5460	0.7066	0.9518	0.8246
-vote,F1	0.4542	0.5530	0.4988	0.5726	0.8874	0.6961
fKPISelect	0.4702	**0.8069**	**0.5942**	**0.8461**	**0.9897**	**0.9123**

F. Sensitivity Study

In Figure 9, we investigate the effect of the number of injected errors of one error type on the KPI selection result. We take the KPI list's overlap ratio with the top-K sensitive KPI list calculated with an error number of 100 as the criterion. When the number of error events is 10, the overlap ratio of top-K-sensitive KPIs (except top-100) is over 0.7. When the number of error events is 10, the overlap ratio of top-K-sensitive KPIs (except top-100) is over 0.8. Thus, if time is limited, performing 10 or 20 fault injections can reach similar results with higher numbers of fault injections.

G. Study on the Result of fKPISelect

This section introduces a study on the result of fKPISelect, including the relationships between KPIs and errors, and the analysis of properties of selected and unselected KPIs.

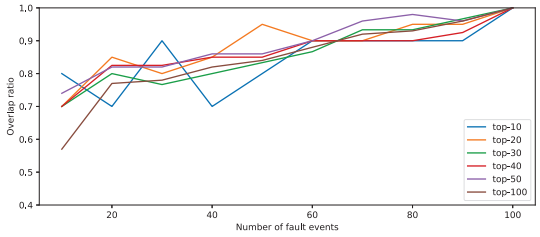

Fig. 9: The overlap ratio of top-K KPIs calculated with different error numbers and those calculated with an error number of 100

TABLE VI: The samples of selected KPIs sensitive to different types of errors

Error Type	Samples of Selected KPIs
Network	Network Traffic, CPU Usage, Network Errors, *etc.*
CPU	CPU Usage, Seconds spent by processes waiting for CPU, Network Traffic, *etc.*
Memory	RAM Usage, Memory used by user-space applications, Memory Page Faults, *etc.*
Disk	CPU Usage, Disk Queue Size, Disk IOps Completed, *etc.*

1) Sensitive KPI to Different Error Types: We used fKPISelect to select KPIs from Node_data. The samples of selected KPIs sensitive to the four injected types of errors are shown in Table VI. From the results of the automatic KPI selection, here is some inspiration we summarize for host monitoring:

- **Resource usage is generally sensitive to errors.** For common workloads and error types, KPIs of resource usage (CPU usage, RAM usage, and Network traffic which can be seen as bandwidth usage) is important metrics for monitoring the system, which is consistent with prior experience.
- **Monitor multiple components' KPIs in one entity.** For each error type, the sensitive KPIs are related to various KPIs. As shown in Table VI, errors in CPUs can lead to anomalies in Network Traffic and errors in disk and network can lead to anomalies in CPU usage. The reason is that the workload is processed with the cooperation of each part. Thus anomaly detection of multivariate time series of multiple components' KPIs is worthwhile to monitor the overall status of entities.
- **Involve process data in host monitoring.** Some errors can be revealed by KPIs related to processes, such as the time spent by processes waiting for CPU. Although this type of KPI is ignored by previous datasets[5, 10], the statistics of processes provided by systems are also valuable to be monitored for host monitoring.

2) Comparison Between Selected and Unselected KPIs: Figure 10 shows the selected and unselected KPIs separately, where the x-axis is the time, the y-axis is the value of the KPI, and the yellow region indicates that anomalies occurred during the time period. Two selected error-sensitive KPIs to memory errors are shown in Figures 10a and 10b, where both KPIs suddenly increase when the memory error occurs and fall back when the error recovers. Figures 10c and 10d show the shapes of two clusters of unselected KPIs, and we select ten KPIs from each cluster to draw them in different colors. Figure 10c is the cluster of KPIs that fluctuate very little during both normal and anomaly periods, and Figure 10d is the cluster of KPIs that fluctuates greatly during the two periods. In general, fKPISelect tends to select KPIs that fluctuate obviously during anomaly periods and disregard those not changing much relative to normal fluctuations.

(a) RAM Usage (b) Memory Page Fault

(c) cluster of unselected KPIs 1 (d) cluster of unselected KPIs 2

Fig. 10: Two examples of selected error-sensitive KPIs to memory errors, and two example clusters of unselected KPIs

VII. Related Work

In recent years, numerous works have researched Multivariate Time Series Anomaly Detection (TSAD). The mostly concerned research points include mining more information in general TSAD tasks, such as OmniAnomaly[5], USAD[6], SDF-VAE[7], InterFusion[8], AnomalyTransformer[9]; improving the scalability of the entity monitoring, such as CTF[10], UniAD[11]. The detailed design of these methods is discussed in section III. However, none of them take how to select suitable time series to represent a whole entity as a question at the preprocessing stage. Another problem with these works is that these improvements are incompatible with others. Currently, some research works focus on the dataset quality and evaluation methods to make anomaly detection research more practical to real-world systems[39, 40].

Several works focus on the engineering of KPIs for anomaly detection. The most popular methods include applying clustering algorithms to KPIs. ROCKA[41] proposes a KPI clustering algorithm on the shape-based distance of KPIs to reduce the model numbers for anomaly detection of many machines. ADS[5] combines clustering and semi-supervised learning to introduce a framework for automatically emerging new KPI streams. These approaches work with univariate TSAD methods. However, currently, there is no systematic KPI selection research to improve Multivariate TSAD for many KPIs.

VIII. Conclusion

To solve the problem that existing studies have overlooked the significance of selecting KPIs before applying anomaly detection models, we proposed a fault-injection-based automated KPI selection mechanism (fKPISelect), which leverages fault injection for knowledge of the relationship between KPIs and errors, and anomaly detection models for sensitivity calculation of KPIs. Through our extensive experimentation and analysis, we demonstrated the effectiveness and practicality of fKPISelect in addressing the challenges associated with applying TSAD models in cloud systems.

References

[1] W. Jackson. (2018) Lloyd's estimates the impact of a u.s. cloud outage at $19 billion. [Online]. Available: http://www.eweek.com/cloud/lloyd-s-estimates-the-impact-of-a-u.s.-cloud-outage-at-19-billion

[2] L. Wang, R. A. Hosn, and C. Tang, "Remediating Overload in Over-Subscribed Computing Environments," in *2012 IEEE Fifth International Conference on Cloud Computing*, pp. 860–867.

[3] Z. Shang, Y. Zhang, X. Zhang, Y. Zhao, Z. Cao, and X. Wang, "Time series anomaly detection for kpis based on correlation analysis and hmm," *Applied Sciences*, vol. 11, no. 23, p. 11353, 2021.

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

[5] 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.

[6] 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*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 3395–3404.

[7] L. Dai, T. Lin, C. Liu, B. Jiang, Y. Liu, Z. Xu, and Z.-L. Zhang, "Sdfvae: Static and dynamic factorized vae for anomaly detection of multivariate cdn kpis," in *Proceedings of the Web Conference 2021*, 2021, pp. 3076–3086.

[8] 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.

[9] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=LzQQ89U1qm_

[10] N. Authors, "Netmanaiops/ctf_data: Data of paper "ctf: Anomaly detection in high-dimensional time series with coarse-to-fine model transfer"," 2021, accessed: 2021-12-14. [Online]. Available: https://github.com/NetManAIOps/CTF_data

[11] Z. He, P. Chen, and T. Huang, "Share or not share? towards the practicability of deep models for unsupervised anomaly detection in modern online systems," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2022, pp. 25–35.

[12] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network." [Online]. Available: http://arxiv.org/abs/2009.02040

[13] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data." [Online]. Available: http://arxiv.org/abs/1811.08055

[14] P. Authors, "Overview — prometheus," 2021, accessed: 2021-12-14. [Online]. Available: https://prometheus.io/docs/introduction/overview/

[15] G. Labs, "Grafana — query, visualize, alerting observability platform," 2021, accessed: 2021-12-14. [Online]. Available: https://grafana.com/grafana/

[16] L. Li, J. Yan, H. Wang, and Y. Jin, "Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 3, pp. 1177–1191, 2020.

[17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[18] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, "An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2508–2517, 2021.

[19] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *2010 IEEE Symposium on Security and Privacy*, pp. 305–316.

[20] P. Liu, Y. Chen, X. Nie, J. Zhu, S. Zhang, K. Sui, M. Zhang, and D. Pei, "FluxRank: A Widely-Deployable Framework to Automatically Localizing Root Cause Machines for Software Service Failure Mitigation," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 35–46.

[21] P. Authors, "prometheus/node_exporter: Exporter for machine metrics," 2021, accessed: 2021-12-14. [Online]. Available: https://github.com/prometheus/node_exporter

[22] G. Labs, "Node exporter full — grafana labs," 2021, accessed: 2021-12-14. [Online]. Available: https://grafana.com/grafana/dashboards/1860-node-exporter-full/

[23] Z. Li, Y. Zhao, R. Liu, and D. Pei, "Robust and rapid clustering of kpis for large-scale anomaly detection," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–10.

[24] R. Natella, D. Cotroneo, and H. S. Madeira, "Assessing Dependability with Software Fault Injection: A Survey," vol. 48, no. 3, pp. 44:1–44:55. [Online]. Available: https://dl.acm.org/doi/10.1145/2841425

[25] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich. Precision and Recall for Time Series. [Online]. Available: http://arxiv.org/abs/1803.03639

[26] R. Hat, "Chapter 27. linux traffic control red hat enterprise linux 9," 2021. [Online]. Available: https://access.redhat.com/documentation/zh-tw/red_hat_enterprise_linux/9/html/configuring_and_managing_networking/linux-traffic-control_configuring-and-managing-networking

[27] Ubuntu, "Kernel/reference/stress-ng - ubuntu wiki," 2021. [Online]. Available: https://wiki.ubuntu.com/Kernel/

Reference/stress-ng
[28] "Chaos-mesh/chaos-mesh," Chaos Mesh. [Online]. Available: https://github.com/chaos-mesh/chaos-mesh
[29] L. Feinbube, L. Pirl, P. Tröger, and A. Polze, "Software Fault Injection Campaign Generation for Cloud Infrastructures," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 622–623.
[30] O. Sagi and L. Rokach, "Ensemble learning: A survey," vol. 8, no. 4, p. e1249. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1249
[31] A. Hat, Red. Ansible is Simple IT Automation. [Online]. Available: https://www.ansible.com
[32] PyTorch, "Pytorch," 2021. [Online]. Available: https://pytorch.org/
[33] NumPy, "Numpy," 2021. [Online]. Available: https://numpy.org/
[34] Production-Grade Container Orchestration. Kubernetes. [Online]. Available: https://kubernetes.io/
[35] "Sock Shop : A Microservice Demo Application," Microservices Demo. [Online]. Available: https://github.com/microservices-demo/microservices-demo
[36] Locust.io. [Online]. Available: https://locust.io/
[37] C. Sauvanaud, K. Lazri, M. Kaâniche, and K. Kanoun, "Anomaly Detection and Root Cause Localization in Virtual Network Functions," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 196–206.
[38] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. International World Wide Web Conferences Steering Committee, pp. 187–196. [Online]. Available: https://dl.acm.org/doi/10.1145/3178876.3185996
[39] K.-H. Lai, D. Zha, Y. Zhao, G. Wang, J. Xu, and X. Hu, "Revisiting Time Series Outlier Detection: Definitions and Benchmarks."
[40] R. Wu and E. Keogh, "Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress," pp. 1–1. [Online]. Available: https://ieeexplore.ieee.org/document/9537291/
[41] J. Bu, Y. Liu, S. Zhang, W. Meng, Q. Liu, X. Zhu, and D. Pei, "Rapid deployment of anomaly detection models for large number of emerging kpi streams," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–8.