ATS: A Fully Automatic Troubleshooting System with Efficient Anomaly Detection and Localization

Lu Yuan[1,2], Yuan Meng[3], Jiyan Sun[1,4](✉), Shangyuan Zhuang[1,2], Yinlong Liu[1,2], Liru Geng[1,2], and Weiqing Huang[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Xinjiang Aksu Public Security Bureau, Xinjiang, China
[4] Corresponding Author: sunjiyan@iie.ac.cn
`{yuanlu, sunjiyan, zhuangshangyuan, liuyinlong, gengliru, huangweiqing}@iie.ac.cn, mengyuan6677@sina.com`

Abstract. As network scale expands and concurrent requests grow, unexpected network anomalies are more frequent, leading to service interruptions and degraded user experience. Real-time, accurate troubleshooting is critical for ensuring satisfactory service. Existing troubleshooting solutions adopt ensemble anomaly detection (EAD) to detect anomalies due to its robustness. However, the fixed base classifier parameters in EAD set by expert experience may reduce the efficiency of anomaly detection when faced with different data distributions. Furthermore, the binary results fed to the secondary classifier in EAD cause information loss, leading to compromised accuracy and inaccurate root cause localization. Besides, key performance indicators (KPIs) are crucial for measuring the system performance, but relying on multiple redundant KPIs to identify the root causes of anomalies is time-consuming and error-prone.
To address the above issues, we propose a fully automatic troubleshooting system, **ATS**. A new EAD method is introduced to detect anomalies, then a module is designed to trigger the root cause localization. Specifically, the EAD method updates the parameters of base classifiers to dynamically adapt to different KPI data distributions. The ensemble of soft labels generated by base classifiers is subsequently fed into the secondary classifier to achieve information-lossless anomaly detection. Then, a heuristic module is proposed to select the most appropriate KPI data based on the metric i.e., *bilayer relative difference* to trigger the efficient root cause localization. Extensive experiments demonstrate that ATS is more than twice as fast as most state-of-the-art solutions while with higher troubleshooting accuracy.

Keywords: Troubleshooting · Ensemble Anomaly Detection · Soft Label · Bilayer Relative Difference

1 Introduction

With the continuous expansion of network scale and rapid growth of concurrent requests, unexpected network anomalies are becoming increasingly com-

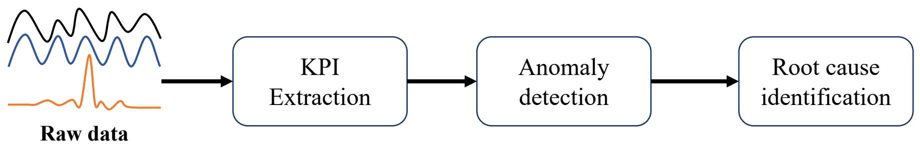


Fig. 1: Process of troubleshooting.

mon, such as unusual traffic patterns, data packet loss, and sudden spikes or drops in network traffic. If the anomalies are not addressed, they may result in service interruptions for users and significant deterioration in user experience. In online commercial services, such issues can even harm business profits [3, 10]. Therefore, a real-time and accurate troubleshooting scheme is crucial for providing satisfactory service and ensuring a seamless user experience by rapidly identifying and resolving network anomalies.

As shown in Figure 1, the troubleshooting process can be summarized in three steps: Key Performance Indicator (KPI) extraction, anomaly detection, and root cause localization (RCL). Firstly, operators extract various real-time KPIs by aggregating and calculating from the raw data. Secondly, Ensemble Anomaly Detection (EAD) algorithms [1, 4, 26, 37, 40] are typically employed to monitor system status by detecting variation of KPIs. In detail, the KPIs are initially input into the base classifiers, i.e. base learners [37], which generate preliminary classification results. These binary outcomes are then simultaneously fed into the secondary classifier to make the final classification decisions [30]. Finally, after detecting an anomaly, many Root Cause Localization (RCL) algorithms [2, 18, 22, 29, 35] are introduced to identify the anomaly root cause to specific dimensions such as province, city, or server ID. Considering the inevitable false positives of anomaly detection, they utilize all KPIs flagged as anomalous to locate the candidates of root cause individually. Subsequently, each candidate is scored and the one with the highest score is ultimately selected as the conclusive result.

Despite the strong performance of these methods, there still exist several important challenges to achieving an efficient troubleshooting method in practice.

Firstly, base learners are the foundation of EAD. By combining more accurate results from base learners, a more precise and robust result can be achieved. However, existing EAD schemes mostly concentrate on optimizing the parameters of the secondary classifier, while the parameters of the base learners are set based on expert experience and remain fixed [31, 34]. The fixed parameters of base learners lead to low accuracy in handling different data distributions, which ultimately affects the final accuracy of the secondary classifier.

Secondly, the final decision of EAD heavily depends on the preliminary classification results of base learners. Many EAD algorithms [1, 4, 37, 40] utilize the secondary classifier to make the final decision only considering the binary results of base learners. However, this approach fails to capture the full classification information provided by the base learners, resulting in significant information loss. This loss has a harmful impact on the optimization of the secondary classifier and the final performance of the system.

Thirdly, current approaches [7, 25, 29] often utilize all impacted KPI data to identify the root cause, resulting in high computational overhead and low efficiency. Moreover, including KPI data that are irrelevant with anomaly as input can lead to a misidentification of the root cause.

To address the above challenges, we propose a *fully Automatic Troubleshooting System* (ATS) in this paper. The main contributions are summarized as follows:

– **Framework of Fully Automatic Troubleshooting System**. We propose a fully automatic troubleshooting system, namely ATS, which contains three primary components: *AutoDetect*, *Gunlock*, and *AutoRoot*. When AutoDetect identifies a system anomaly, Gunlock calculates the most relevant KPI data to trigger AutoRoot for efficient root cause localization(Section 4.1).
– **A Robust and Information Lossless Ensemble Anomaly Detection**. To improve the accuracy of anomaly detection, we propose a robust and information lossless ensemble scheme for anomaly detection (AutoDetect). AutoDetect dynamically updates base learner parameters using Bayesian Optimization to adapt to various KPI data distributions. It then combines the original probabilities of each base learner as *soft labels*, which are fed into the secondary classifier for information-lossless anomaly detection(Section 4.2).
– **Heuristic Trigger**. We propose a novel heuristic trigger called *Gunlock* to accelerate RCL (AutoRoot). Gunlock utilizes a metric known as *bilayer relative difference* (BRD) to identify the most proper KPI and transfer it to AutoRoot for root cause localization (Section 4.3).
– **Evaluation on real data traces**. We use a dataset from a large-scale content delivery network (CDN) to evaluate the performance of ATS. Extensive experiments demonstrate that ATS outperforms the state-of-the-art approaches. For instance, *AutoDetect* gains a high system anomaly detection performance with average 10 percent higher than state-of-art algorithms in F1-score. In addition, we note that ATS shortens troubleshooting time by half on average (Section 5).

2 Related Work

As the two critical components of online service system troubleshooting, anomaly detection and RCL have become popular research topics in recent years [22, 23].

Ensemble learning is a machine learning technique that combines multiple base learners to achieve higher accuracy than a single model. It plays an important role in the research field of anomaly detection [1, 7, 37] and is widely deployed in industry [17, 27, 36]. To create a stronger model, Paulauskas and Auskalnis [13] propose an ensemble model consisting of four different base classifiers, which depends on the idea of combining multiple weaker learners. Vanerio et al. [37] investigate different ensemble-learning approaches and find the optimal scheme to enhance anomaly detection in network measurements. Rajagopal et al. [32] provide an ensemble paradigm based on meta-classification and stacked generalization with the goal of improving prediction accuracy. In industry, EGADS [17] in Yahoo trains multiple models for different types of KPI and assembles the appropriate models as plugins. Metis [36] and Surus [27] are proposed to achieve robust anomaly detection by aggregating different anomaly detection algorithms. These studies have demonstrated that the application of ensemble methods enhances the performance of the models, such as prediction accuracy.

However, the above solutions combine the binary results of base learners. The ensemble with binary results of base learners produces information loss which leads to inaccurate results. Moreover, they optimize the model by adjusting the parameters of the secondary classifier but do not optimize the base learners. This leads to the inefficient performance of the ensemble model.

There have been some papers proposing the advanced root cause location algorithm. HotSpot [35] develops a score metric to identify the root cause nodes. Li et al. [18] and Jing et al. [15] propose improved solutions based on clustering to address the problem of long tail distribution of KPI metrics. They locate root causes using each KPI data affected by anomalies. But some changes in KPI data are not associated with anomalies. Locating the root cause with the improper input KPI data results in wrong results and wasting of computing source.

3 Preliminaries

In this section, we illustrate some important definitions used in this paper.

To enable real-time troubleshooting, a variety of KPIs such as *visit count*, *in flow rate*, and *cache hit ratio* are collected from raw logs. KPIs that are collected with multiple dimensions are referred to as multi-dimensional KPIs (**MDKPI**s). For instance, consider the CDN data of visit count in Table 1, where the first row represents an MDKPI with three dimensions: Province, ISP (Internet Service Provider), and Website. The corresponding dimension values are $Beijing$, $ChinaMobile$, $Weibo.com$, respectively.

The KPI can be calculated for each **dimension combination**, e.g., in the first row of Table 1, the KPI value for dimension combination $\{Beijing, ChinaMobile, Weibo.com\}$ is 59 (line 1). Specifically, the character $*$ represents all the possible values of the corresponding dimension. For instance, the dimension combination $\{*, ChinaMobile, Weibo.com\}$ represents all the users that come from ISP *China Mobile* and visit the website *Weibo.com*. The *visit count* value of $\{*, ChinaMobile, Weibo.com\}$ is $59+31028+370=31457$ (line 1-3).

Table 1: Example of multi-dimensional KPI.

Index	Province	ISP	Website	Value
1	Beijing	China Mobile	Weibo.com	59
2	Shanghai	China Mobile	Weibo.com	31028
3	Guangdong	China Mobile	Weibo.com	370
4	Beijing	China Mobile	QQ.com	221
5	Shanghai	China Mobile	QQ.com	10
6	Guangdong	China Mobile	QQ.com	33
7	**Beijing**	**China Unicom**	**Weibo.com**	**731**
8	**Shanghai**	**China Unicom**	**Weibo.com**	**10**
9	**Guangdong**	**China Unicom**	**Weibo.com**	**6**
10	Beijing	China Unicom	QQ.com	441
11	Shanghai	China Unicom	QQ.com	16

Especially, when there is only one dimension value and other dimensions take the value of *, the KPI is called Single-dimension KPI (**SDKPI**). For example, the KPI *visit count* with dimension *Website* is one SDKPI, which corresponds to the dimension combinations $\{*,*,Weibo.com\}$ and $\{*,*,QQ.com\}$. The visit count values of Website dimension are $(Weibo.com,32204)$ and $(QQ.com,721)$, which is calculated by $59+31028+370+731+10+6=32204$ (line 1-3, 7-9), $221+10+33+441+16=721$ (line 4-6, 10-11), respectively.

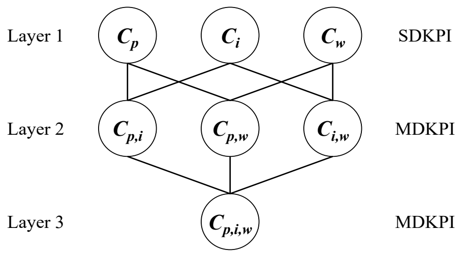


Fig. 2: Root Cause Search Tree.

As shown in Figure 2, we assume that the search space of multi-dimensional root causes is a tree-like structure. Each node in the tree indicates a dimensional combination. A specific dimension value represents the corresponding potential fault location of it. Especially, a **leaf node** is a dimension combination without any wildcard $*$, e.g., $\{Beijing,ChinaUnicom,Weibo.com\}$. A **cabin** denotes the set of dimension combinations with the same non-wildcard dimensions. The **layer** is the number of non-wildcard dimensions of a cabin. Let the terms p, i, w denote the dimensions *Province*, *ISP*, *Website*, respectively. $\{Beijing,ChinaUnicom,*\}$ and $\{Shanghai,ChinaMobile,*\}$ belong to the cabin $C_{p,i}$, and they are in the layer 2. All leaf nodes are in the cabin $C_{p,i,w}$ of Layer 3.

4 System Design

4.1 Overview of Framework

We propose a *fully automatic troubleshooting system* (ATS) shown in Figure 3. The ATS mainly consists of four parts: KPI extraction, *AutoDetect* for anomaly detection, *Gunlock* for trigger AutoRoot, and *AutoRoot* for RCL.

To **extract** KPI data, we preprocess the raw data, which includes *data filling*, *data smoothing*, and *KPI extraction*. Firstly, we fill incomplete data with the mean of the surrounding context and smooth KPI data using exponential mean average [16] to eliminate noise to some extent. Then we normalize the values to remove the influence of the order-of-magnitude differences between KPIs. We filter out KPIs with a variance close to 0 and extract SDKPI with website dimension for anomaly detection and MDKPI for RCL.

To enhance the performance of anomaly detection, **AutoDetect** is proposed which is a robust and information lossless ensemble anomaly detector (Section 4.2). AutoDetect's extensible first layer comprises multiple base learners elaborately selected. To accommodate diverse data distributions, we employ Bayesian optimization to automatically select the best parameters for base learners. Subsequently, the *soft labels* from the base learners are integrated and fed to the secondary classifier to avoid information loss and improve the efficiency of anomaly detection.

After detecting an anomaly, **Gunlock** (Section 4.3) computes the heuristic metric BRD to identify the most suitable KPI, referred to as *Trigger_ KPI*. Then

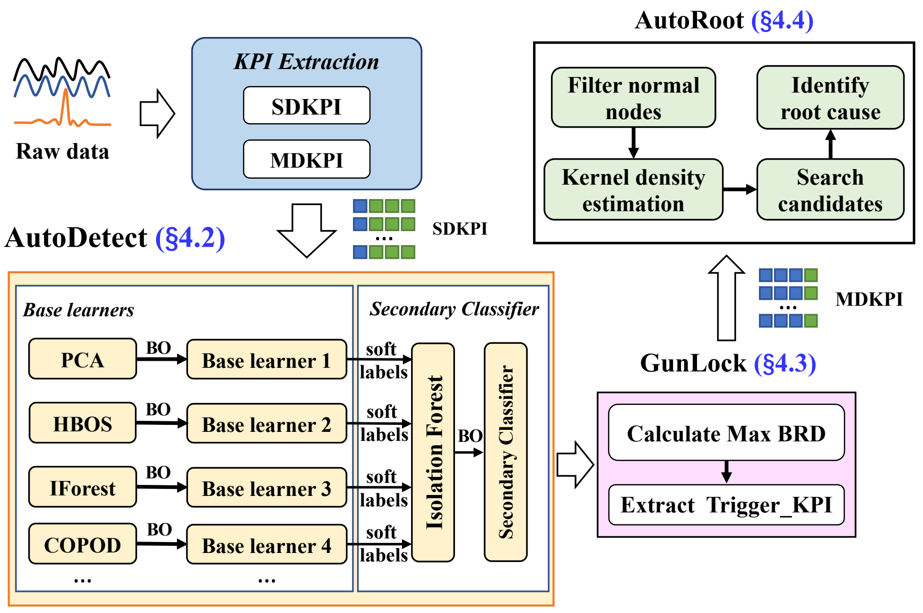


Fig. 3: The framework of ATS. The blue square denotes the KPI dimension and the green square denotes the KPI category. SDKPI consists of different types of KPI and one dimension. MDKPI includes multiple dimensions and one KPI.

it extracts the MDKPI of Trigger_KPI and employs it to trigger **AutoRoot** to locate the root cause.

The multi-dimensional root cause locator AutoRoot [15] is applied to RCL. Firstly, it calculates the deviation score of each node to filter normal nodes. Secondly, *Kernel Density Estimation* (KDE) with Gaussian kernel is employed to group leaf nodes into clusters. Next, the candidate of each cluster is identified by calculating the *root score* (RS). Finally, the candidates are merged and unnecessary ones are removed to obtain a precise set of root causes (Section 4.4).

4.2 AutoDetect: A Novel Ensemble Anomaly Detection

According to the data analysis, we observe that an anomaly will be reflected in the value of multiple KPIs. To better capture the various correlations between KPI data and anomalies, we have developed four base learners and integrated them to create a more effective anomaly detection system.

Base Learners The base learners can be machine learning-based classifiers (MLCs) [20] and deep learning-based classifiers (DLCs) [28, 33]. While both are effective at analyzing static data, our experiments have shown that DLCs require more time to train than MLCs (see the comparison in Section 5.3). Since KPI data distributions in online systems are constantly varying, it is essential that our anomaly detection model be retrained and updated frequently. To minimize the time required for these processes, we choose four MLCs as our base learners due to their low training overhead. It is worth noting that this stage is scalable,

Table 2: Comparison of the ensemble with soft labels and binary results.

Soft Labels					Binary Results					Label
PCA	HBOS	IForest	COPOD	Result	PCA	HBOS	IForest	COPOD	Result	
0.0061	0.0187	0.0314	0.1404	**1**	0	0	0	0	**0**	**1**
0.0079	0.0226	0.0332	0.1406	**1**	0	0	0	0	**0**	**1**
0.0091	0.0205	0.0147	0.1704	**1**	0	0	0	0	**0**	**1**
0.0080	0.0229	0.030	0.1612	**1**	0	0	0	0	**0**	**1**

so the base learners are not confined to the selected learners and other learners can be added.

To examine the KPI data from multiple angles, we choose several effective techniques as base learners. These include the proximity-based approach known as Histogram-based Outlier Score (HBOS) [8], the refactoring-based method called Principal Components Analysis (PCA) [11], the efficient classifier known as Isolation Forest (IForest) [21], and a newly developed technique called Copula-based Outlier Detection (COPOD) [19]. All of these models excel at analyzing data with high dimensions [6].

Automatic Parameter Selection To achieve optimal performance of base learners in varying data distributions, we apply the Bayesian Optimization (BO) [9] method to optimize the base learners during training. By utilizing BO to select the most appropriate parameters, the performance of the base learners is significantly improved. It can be expressed as:

$$\lim_{p \in P} F(Clf(p); P) \tag{1}$$

where P is the set of the possible values of p. $Clf(\cdot)$ denotes the classifiers. $F(\cdot)$ is the objective function that is usually a user-defined function using different classifier metrics. In this paper, the objective function is the negative number of the F1-score of the classifier. Therefore, the objective function is represented as follows:

$$\min_{p \in P} \sum_{m=1}^{M} -F_1 Score(C_m(p); P) \tag{2}$$

where C_m represents the base learners, and M represents the number of base learners which is 4 in this paper. Equation 2 represents that each base learner C_m chooses the appropriate parameters p to optimize the performance of the classifier.

Information Lossless Ensemble Existing ensemble strategies [4, 26, 37] combine the binary results of base learners for comprehensive making decisions to gain better performance. The results shown in Table 2 indicate that all of the base learners generate binary results of 0 (columns 6-9). As a result, the final result of the ensemble classifier is also 0 (column 10), despite the fact that the true labels indicate a value of 1 (column 11).

It is obvious that information loss occurs during the generation of binary results by the base learners, leading to incorrect outcomes when these results are fed into the secondary classifier, ultimately compromising its accuracy.

Instead of generating binary results, the base learners of AutoDetect output the probability of data being anomalous. This probability retains all the learning information and is instrumental in identifying anomalies. We refer to this probability as the *soft label*. The soft labels then are fed into the secondary classifier to make the final decision. As shown in Table 2, the final decisions based on the soft labels of the base learners (column 5) are identical to the true labels, demonstrating the effectiveness of our approach.

Additionally, the choice of the secondary learning algorithm has a significant impact on the generalization performance of stacking integration. Therefore, after comparing with typical algorithms, we have selected Isolation Forest [21] as our secondary classifier(Section 5). To further improve the model's performance, Bayesian optimization (BO) is applied to the training of the secondary classifier.

Training of AutoDetect As Algorithm 1 shows, the process of *AutoDetect* training is summarized as follows:

– The first step is to train individual base learners (Step 1-4). The extracted SDKPI is input into the base learners. And BO is applied to choose appropriate parameters to enhance the performance of each learner.
– The second step is to use the base learners to predict the test data and aggregation of soft labels is used as the secondary training set, which is the training set of the secondary classifier (Step 5-11).
– The final step is to train the secondary classifier (Step 12-13). To thoroughly analyze the soft labels, we adopt the stacking combination strategy and select Isolation Forest as our secondary classifier. Additionally, we utilize BO during the training process of the secondary classifier.

Algorithm 1 AutoDetect training

```
Input: The SDKPIs training dataset $D = (X_1, y_1), (X_2, y_2), ..., (X_n, y_n)$, base learners
    $\mathcal{L}_1, \mathcal{L}_2, ...\mathcal{L}_m$, secondary learner $\mathcal{L}$
Output: Predicting result $H(X) = h'(h(X_1, X_2, ..., X_n))$
1:  for $i = 1, 2, ..., m$ do
2:      $p = BayesianOptimization(\mathcal{L}_i)$.
3:      $h_i = \mathcal{L}_i(D, p)$
4:  end for
5:  $D' = \varnothing$
6:  for $i = 1, 2, ..., n$ do
7:      for $j = 1, 2, ..., m$ do
8:          $z_{ij} = h_j(X_i)$
9:      end for
10:     $D' = D' \cup ((z_{i1}, z_{i2}, ..., z_{im}), y_i)$
11: end for
12: $p' = BayesianOptimization(\mathcal{L})$.
13: $h' = \mathcal{L}(D', p')$
```

4.3 Gunlock: Trigger

In real-world systems, a single anomaly can impact multiple KPIs. However, analyzing each affected KPI is computationally costly, and not all changes in KPI values indicate the anomaly. This wastes computing resources. To optimize efficiency and accelerate localization, it is crucial to select the most appropriate KPI called *Trigger_KPI* as the sole input for RCL.

Existing solutions for identifying system anomalies focus on KPIs with the largest fluctuation, but this may not always be relevant or indicate anomalies. Some KPIs have a wide normal range of fluctuation without significant impact on the average value over time, leading to relative differences that appear large but are not anomalous.

To find the most proper KPI data for RCL, we propose a novel heuristic metric named *bilayer relative difference* (BRD) to evaluate the degree of change in KPIs. It considers the historical change in KPI data as follows:

$$\begin{aligned} BRD_\tau^i &= \frac{k_\tau^i - k_{mean}^i}{k_{mean}^i} \\ k_{mean}^i &= avg(med_1, med_2) \end{aligned} \tag{3}$$

where k_τ^i represents the i-th KPI at time τ and $i = 1, 2, ..., n$. k_{mean}^i denotes the median mean of the i-th KPI in the last week. We calculate the median m_1 of the de-duplicated values of the i-th KPI data for a week. Then, the weekly data is split into two groups based on whether the values are greater than or less than m_1. med_1 and med_2 represent medians of these groups respectively. k_{mean}^i is the average of med_1 and med_2.

When an anomaly is detected, Gunlock analyzes the impact on each KPI by calculating its BRD. This helps determine which KPIs are most affected by the anomaly. Gunlock then identifies the KPI with the highest BRD as the Trigger_KPI. Finally, Gunlock extracts the MDKPI for the Trigger_KPI and sends the data to AutoRoot, which locates the root cause of the anomaly.

4.4 AutoRoot: Root Cause Localization

We implement the AutoRoot proposed in [15] to localize the root cause due to the parameter-free clustering. It is beneficial for constructing a fully automatic system.

Fliter normal nodes The forecast value of the Trigger_KPI is calculated from historical data according to the ARMA (Auto-regressive moving average model) algorithm [24]. By comparing the real and forecast values, the *deviation score* for each leaf node is computed to enable differentiation between anomalous and normal nodes. The *deviation score* is defined as:

$$d(e) = \frac{f(e) - v(e)}{f(e) + v(e)} \tag{4}$$

where $f(\cdot)$ and $v(\cdot)$ are forecast value and real value functions of dimension combinations. The deviation scores of normal leaf nodes are far less than that of abnormal ones according to the GRE principle [18, 35]. We use a proper range of deviation scores [-0.1,0.1] to filter out normal leaf nodes according to the observation of the real dataset [18].

Kernel Density Estimation based clustering We use the KDE with the Gaussian kernel to obtain the distribution density function from the deviation score array and the relative maxima and minima of this function. The relative maxima are the centers of each cluster, while the nearby relative minima are the boundaries of the clusters. Consequently, we obtain distinct clustering intervals, and by categorizing the deviation score arrays into these intervals, we can allocate the anomalous leaf nodes inherited from the same root cause into the same clusters.

Search candidates The root score (RS) of each dimension combination is calculated following Equation 5 to find the candidates for the root causes.

$$RS = avg(NPS + LF + CF) \tag{5}$$

$$NPS = 1 - \frac{avg(\frac{|v(e_{rc}) - a(e_{rc})|}{v(e_{rc})}) + avg(\frac{v(e_l) - f(e_l))}{v(e_l)})}{avg(\frac{|v(e_{rc}) - f(e_{rc})|}{v(e_{rc})}) + avg(\frac{v(e_l) - f(e_l))}{v(e_l)})} \tag{6}$$

$$a(e) = f(e) - f(e)\frac{f(S) - v(S)}{f(S)} = f(e)\frac{v(S)}{f(S)} \tag{7}$$

where the new potential score (NPS) is used to evaluate the probability of a node rc being the root cause. CF is denoted as the rate of descending leaf nodes in the cluster and LF is the descending rate of all its leaf nodes. The notation $avg(\cdot)$ is an average function. As shown in Equation 6, NPS follows that if a dimension combination is a root cause, the difference between its real value and forecast value should be assigned proportionally to its leaf nodes. $v(e_{rc})$ and $a(e_{rc})$ are the real value and expected value of the leaf nodes which are inherited from the assumed root cause node rc respectively. $v(e_l)$ and $f(e_l)$ represent the real value and forecast value of all the remaining leaf nodes, respectively. Equation 7 defines the expected value $a(e)$ where S denotes a non-leaf node and leaf node e is inherited from S.

The dimension combination with the largest NPS is denoted as a candidate. The dimension combinations with the same value are also called sets in each cabin. The candidate is the most potential root cause in each set. Then we sort all the candidates by root cause score RS and extracted the candidate with the largest RS as the potential root cause in the cluster.

Identify root cause We use Occam's Razor to merge all the most potential root causes to lite recommendation results. If a root cause belongs to another root cause, then the one with the wider scope is retained. For example, if {*Beijing, *,*

**}* and {*Beijing, *, Weibo.com*} are two recommended root causes from different clusters, We will eliminate {*Beijing, *, Weibo.com*} to make the result more concise.

5 Evaluation

In this section, we first describe the experimental dataset and performance metrics. Then, we conduct experiments on the dataset to assess the performance of ATS.

5.1 Dataset

The dataset used in this paper is collected from a real-world large-scale CDN system by ourselves. It is a synthetic dataset based on real data ranges from January 2019 to September 2020, which is collected from the CDN system. It includes five dimensions, which are *Province*, *Website*, *Operator System*, *Network Type*, and *Caching Server*. It has 30768 leaf dimension combinations. For *Website*, there are multiple system-level KPIs, such as *in_flow*, *out_flow*, *CDN_ttfb*, etc. Besides, we utilize KPIs of three websites to test the performance of *AutoDetect*. These websites are the top three video websites with massive visits and extensive traffic. The first Website (Website 1) is a live video website that lives at regular intervals.

5.2 Baseline and Evaluation Metrics

We compare the performance of anomaly detection (*AutoDetect*) with four ensemble anomaly detection (EAD) schemes as baseline. Recent research [37] mentions three ensemble schemes named *MVuniform*, *MVscore* and *MVexp*. *MVuniform* gives the same weight $(1/n)$ to each learner, where n is the number of base learners and it implements simple majority voting. *MVscore* assigns weights $w_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$ to the prediction of learner i, being f_i the F1-score of the learner. *MVexp* computes weights with an exponential classification F1-score, $w_i = \frac{e^{\lambda f_i}}{\sum_{i=1}^{n} e^{\lambda f_i}}$, where λ is selected to reduce the influence of low F1-score predictors. We take $\lambda = 10$ for such an effect. We also implement a baseline algorithm *Squeeze* [18] as the comparative solutions of ATS.

We apply *Precision*, *Recall* and *F1-score* to evaluate the performance of ATS. Additionally, we add *Time_cost* based on the metrics above to evaluate the performance of different MDRCL schemes. The key performance metrics *Precision* is the rate of correctly troubleshooting anomalies to the total number of anomalies troubleshooting and *Recall* is the ratio of correctly troubleshooting anomalies to all anomalies. *F1-score* is a harmonic average of *Precision* and *Recall*, which is usually used to measure the efficiency of an algorithm, denoted as $F1\_score = \frac{2*Precision*Recall}{Precision+Recall}$. We evaluate the average consumption time of each algorithm on the same Linux server.

Table 3: Performance comparison of AutoDetect with different classifiers based on Precision, Recall and F1-score.

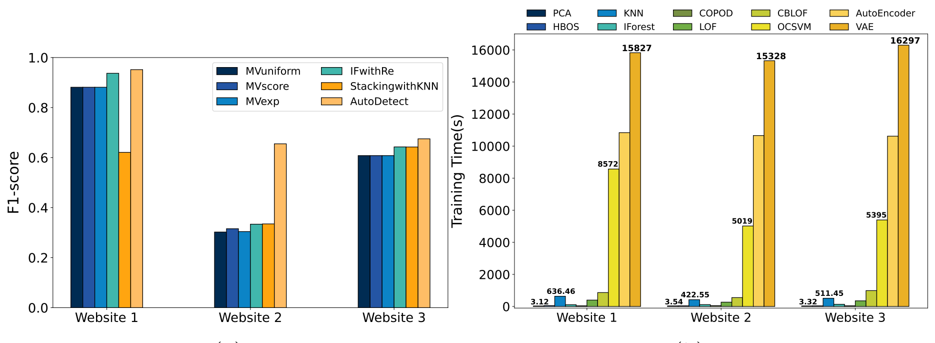
Methods	Website1			Website2			Website3		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
PCA	0.924	0.956	0.940	0.668	0.226	0.337	0.863	0.524	0.652
HBOS	0.930	0.941	0.935	0.736	0.300	0.426	0.853	0.544	0.664
IsolationForest	0.936	0.922	0.929	0.751	0.229	0.351	0.883	0.530	0.663
COPOD	0.915	0.444	0.598	0.763	0.783	0.773	0.849	0.361	0.506
KNN	0.866	0.968	0.914	0.345	0.472	0.399	0.883	0.546	**0.675**
LOF	0.833	0.791	0.811	0.567	0.557	0.562	0.620	0.320	0.422
CBLOF	0.860	0.908	0.883	0.372	0.371	0.372	0.700	0.398	0.507
OCSVM	0.735	0.907	0.812	0.911	0.712	**0.799**	0.454	0.701	0.551
AutoDetect	0.929	0.976	**0.952**	0.795	0.786	**0.790**	0.883	0.546	**0.675**

5.3 Performance of Anomaly Detection

In this section, we describe the performance of *AutoDetect* in dataset with KPIs of three websites and compare with the performance of baseline schemes.

Table 3 reports the *Precision*, *Recall* and *F1-score* of the *AutoDetect* and the typical machine learning classifiers (MLCs) on our dataset, where the best *F1-scores* for all methods are highlighted in boldface. All of the MLCs are implemented by Pyod [39] and their parameters are optimized as same as base learners in AutoDetect. It is obvious that *AutoDetect* is the most stable and efficient. The four base learners (PCA, HBOS, IForest, COPOD) are relatively efficient among the others. HBOS, KNN [5] and LOF [14] belong to proximity-based methods, but HBOS performs much stabler and less time-consuming than others in anomaly detection. Although OCSVM [38] and CBLOF [12] have relatively good performance, we do not choose them due to time consumption.

As shown in Figure 4a, we compare the *F1-score* of five anomaly detection ensemble schemes in three datasets. It is observed that *AutoDetect* performs best. We observe that *AutoDetect* outperforms Isolation Forest with the result of base learners (IFwithRe). It means that information loss impacts the performance of ensemble. Using the *soft labels* of base learners avoids the loss of



(a) (b)

Fig. 4: (a) Performance comparison of ensemble schemes, (b) Training time comparison of learners.

information. Compared with different ensemble schemes (MVuniform, MVscore, MVexp, StackingwithKNN), *AutoDetect* achieves higher *F1-score* and more stability.

In addition, we compare the cost of time between classical MLCs and deep learning classifiers (DLCs). Figure 4b reports the result. We notice that the costs of PCA, HBOS and COPOD are the lowest, which are always lower than 5 seconds. The four base learners that we select are the four with the least training time. By contrast, the time cost of Autoencoder and VAE is around 3 or 4 hours, respectively. Apart from OCSVM, the training time of MLCs is in seconds or minutes, while that of DLM is in hours. The cost of DLM is hundreds to thousands of times than MLCs brings about failure to adapt to the time variation of the online system.

5.4 Performance of ATS

In this section, we introduce the performance of the fully automatic troubleshooting system (ATS) in terms of time consumed.

We compare the cost time of ATS, Squeeze [18], and ATS without Gunlock. As shown in Figure 5, the average time cost by Squeeze and ATS is 46.27 and 9.75 seconds, respectively. It reveals ATS achieves almost 5x improvement in time cost compared to Squeeze. Since the effective parameters optimization of ATS and the reduction of search space, ATS eliminates unnecessary computations and thus achieves fast troubleshooting. Moreover, the average time cost of ATS without Gunlock (AD+AR) is more than twice as much as that of ATS. This is due to the Gunlock finding the unique *Trigger_KPI* and decreasing the computational cost.

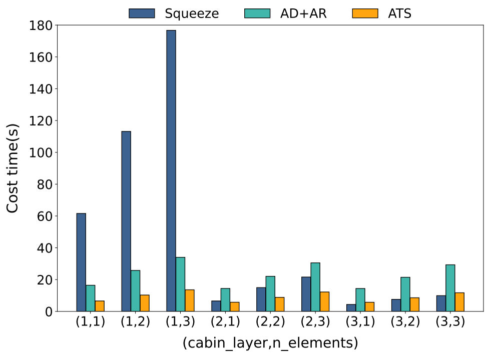


Fig. 5: Comparison of cost time.

6 Conclusion

In this paper, we propose the ATS which is a fully automatic troubleshooting system. It accomplishes troubleshooting by employing efficient ensemble soft-labels-based anomaly detection *AutoDetect*, a heuristic trigger (*Gunlock*), and fast multi-dimensional root cause localization *AutoRoot*. Furthermore, we discussed the components of the framework in detail and evaluated it in terms of anomaly detection and runtime performance. Extensive experiments on one real data trace demonstrate that ATS is more accurate and faster than traditional manual diagnostics and state-of-the-art solutions.

Acknowledgements This work is supported by the National Key Research and Development Program of China (No. 2021YFB2910108).

References

1. Aburomman, A.A., Reaz, M.B.I.: A survey of intrusion detection systems based on ensemble and hybrid classifiers. Computers & security **65**, 135–152 (2017)
2. Ahmed, F., Erman, J., et al.: Detecting and localizing end-to-end performance degradation for cellular data services based on tcp loss ratio and round trip time. IEEE/ACM Transactions on Networking **25**(6), 3709–3722 (2017)
3. Amazon: Amazon found every 100ms of latency cost them 1% in sales. http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-costthem-1-in-sales/, Aug. (2008)
4. Araya, D.B., Grolinger, K., ElYamany, H.F., Capretz, M.A., Bitsuamlak, G.: An ensemble learning framework for anomaly detection in building energy consumption. Energy and Buildings **144**, 191–206 (2017)
5. Chaovalitwongse, W.A., et al.: On the time series k-nearest neighbor classification of abnormal brain activity. T-SMCA **37**(6), 1005–1016 (2007)
6. Chen, Z., et al.: Combining mic feature selection and feature-based mspca for network traffic anomaly detection. In: DIPDMWC. pp. 176–181. IEEE (2016)
7. Folino, G., Sabatino, P.: Ensemble based collaborative and distributed intrusion detection systems: A survey. Journal of Network and Computer Applications **66**, 1–16 (2016)
8. Goldstein, M., Dengel, A.: Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. KI-2012: poster and demo track **9** (2012)
9. Golovin, D., Solnik, B., et al.: Google vizier: A service for black-box optimization. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1487–1495 (2017)
10. Google: http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html. (2006)
11. Groth, D., Hartmann, S., Klie, S., Selbig, J.: Principal components analysis. In: Computational Toxicology, pp. 527–547 (2013)
12. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern recognition letters **24**(9-10), 1641–1650 (2003)
13. Jabbar, M.A., Aluvalu, R., Reddy, S.S.S.: Cluster based ensemble classification for intrusion detection system. In: ICMLC. pp. 253–257 (2017)
14. Jin, W., Tung, A.K., Han, J., Wang, W.: Ranking outliers using symmetric neighborhood relationship. In: Pacific-Asia conference on knowledge discovery and data mining. pp. 577–593. Springer (2006)
15. Jing, P., Han, Y., Sun, J., Lin, T., Hu, Y.: Autoroot: A novel fault localization schema of multi-dimensional root causes. In: WCNC. pp. 1–7. IEEE (2021)
16. Klinker, F.: Exponential moving average versus moving exponential average. Mathematische Semesterberichte **58**(1), 97–107 (2011)
17. Laptev, N., Amizadeh, S., Flint, I.: Generic and scalable framework for automated time-series anomaly detection. In: SIGKDD. pp. 1939–1947 (2015)
18. Li, Z., Luo, C., et al.: Generic and robust localization of multi-dimensional root causes. In: ISSRE. pp. 47–57. IEEE (2019)
19. Li, Z., Zhao, Y., et al.: Copod: copula-based outlier detection. In: ICDM. pp. 1118–1123. IEEE (2020)

20. Liu, D., Zhao, Y., et al.: Opprentice: Towards practical and automatic anomaly detection through machine learning. In: IMC. pp. 211–224 (2015)
21. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 eighth ieee international conference on data mining. pp. 413–422. IEEE (2008)
22. Luglio, M., Romano, S.P., Roseti, C., Zampognaro, F.: Service delivery models for converged satellite-terrestrial 5g network deployment: A satellite-assisted cdn use-case. IEEE Network **33**(1), 142–150 (2019)
23. Ma, M., Yin, Z., Zhang, S., Wang, S., Zheng, C., Jiang, X., Hu, H., Luo, C., Li, Y., Qiu, N., et al.: Diagnosing root causes of intermittent slow queries in cloud databases. Proceedings of the VLDB Endowment **13**(8), 1176–1189 (2020)
24. McLeod, A.I., Li, W.K.: Diagnostic checking arma time series models using squared-residual autocorrelations. Journal of time series analysis **4**(4), 269–273 (1983)
25. Meng, Y., Zhang, S., et al.: Localizing failure root causes in a microservice through causality inference. In: IWQoS. pp. 1–10. IEEE (2020)
26. Mirza, A.H.: Computer network intrusion detection using various classifiers and ensemble learning. In: SIU. pp. 1–4. IEEE (2018)
27. Netflix: https://github.com/netflix/surus (2019)
28. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. ACM Computing Surveys (CSUR) **54**(2), 1–38 (2021)
29. Persson, M., Rudenius, L.: Anomaly detection and fault localization an automated process for advertising systems. Master's thesis (2018)
30. Pham, N.T., Foo, E., et al.: Improving performance of intrusion detection system using ensemble methods and feature selection. In: ACSW. pp. 1–6 (2018)
31. Rahman, M.A., Shoaib, S., et al.: A bayesian optimization framework for the prediction of diabetes mellitus. In: ICAEE. pp. 357–362. IEEE (2019)
32. Rajagopal, S., Kundapur, P.P., Hareesha, K.S.: A stacking ensemble for network intrusion detection using heterogeneous datasets. Security and Communication Networks **2020**, 1–9 (2020)
33. Su, Y., Zhao, Y., et al.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: SIGKDD. pp. 2828–2837 (2019)
34. Sun, S., Jin, F., et al.: A new hybrid optimization ensemble learning approach for carbon price forecasting. Applied Mathematical Modelling **97**, 182–205 (2021)
35. Sun, Y., Zhao, Y., et al.: Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes. IEEE Access **6**, 10909–10923 (2018)
36. Tencent: https://github.com/tencent/metis (2019)
37. Vanerio, J., Casas, P.: Ensemble-learning approaches for network security and anomaly detection. In: Big-DAMA@SIGCOMM. pp. 1–6 (2017)
38. Wang, Z., Fu, Y., Song, C., Zeng, P., Qiao, L.: Power system anomaly detection based on ocsvm optimized by improved particle swarm optimization. IEEE Access **7**, 181580–181588 (2019)
39. Zhao, Y., Nasrullah, Z., Li, Z.: Pyod: A python toolbox for scalable outlier detection. Journal of Machine Learning Research **20**(96), 1–7 (2019), http://jmlr.org/papers/v20/19-011.html
40. Zhong, Y., Chen, W., et al.: Helad: A novel network anomaly detection model based on heterogeneous ensemble learning. Computer Networks **169**, 107049 (2020)