Practical and White-Box Anomaly Detection through Unsupervised and Active Learning

Yao Wang[†¶], Zhaowei Wang[‡], Zejun Xie[‡], Nengwen Zhao[†¶], Wenchi Zhang[‡],
Junjie Chen[§], Kaixin Sui[‡], Dan Pei[†¶*]
[†]Tsinghua University, 100084 [‡]BizSeer, 100083 [§]Tianjin University, 300354
[¶]Beijing National Research Center for Information Science and Technology (BNRist), 100084
[†]{wangyao18, znw17}@mails.tsinghua.edu.cn, peidan@tsinghua.edu.cn [§]junjiechen@tju.edu.cn
[‡]1160300903@stu.hit.edu.cn, 2017201625@ruc.edu.cn, {zhangwenchi, suikaixin}@bizseer.com

***Abstract*—To ensure quality of service and user experience, large Internet companies often monitor various Key Performance Indicators (KPIs) of their systems so that they can detect anomalies and identify failure in real time. However, due to a large number of various KPIs and the lack of high-quality labels, existing KPI anomaly detection approaches either perform well only on certain types of KPIs or consume excessive resources. Therefore, to realize generic and practical KPI anomaly detection in the real world, we propose a KPI anomaly detection framework named *iRRCF-Active*, which contains an unsupervised and white-box anomaly detector based on Robust Random Cut Forest (RRCF), and an active learning component. Specifically, we novelly propose an improved RRCF (iRRCF) algorithm to overcome the drawbacks of applying original RRCF in KPI anomaly detection. Besides, we also incorporate the idea of active learning to make our model benefit from high-quality labels given by experienced operators. We conduct extensive experiments on a large-scale public dataset and a private dataset collected from a large commercial bank. The experimental resulta demonstrate that *iRRCF-Active* performs better than existing traditional statistical methods, unsupervised learning methods and supervised learning methods. Besides, each component in *iRRCF-Active* has also been demonstrated to be effective and indispensable.**

***Index Terms*—Key Performance Indicators, Unsupervised Anomaly Detection, Time Series, Active Learning, RRCF**

I. Introduction

To provide high-quality services and guarantee user experience, technology and financial corporations have been closely monitoring various **Key Performance Indicators (KPIs)**, including business KPIs (high-level) and server KPIs (low-level). Abnormal patterns or behaviors in KPIs may indicate potential failures of systems or applications. Therefore, detecting KPI anomalies is crucial for immediate failure discovery and troubleshooting. However, designing a well-performed KPI anomaly detection in practice faces the following challenges. First, based on our observation, the numbers of KPIs that are required to be monitored is huge in large companies [1], [2]. Take a commercial bank as an example, there are millions of indicators that need to be monitored in a single business line. Besides, diverse KPI patterns exist in the real world, for example, seasonal, stationary and variable. Fig. 1 presents some typical patterns from our datasets (to be introduced in §IV-A). Furthermore, some seasonal KPIs are also influenced by irregular holidays (e.g. KPI e in Fig. 1), and consequently, manual intervention is necessary to correct the detection results in order to avoid future mistakes.

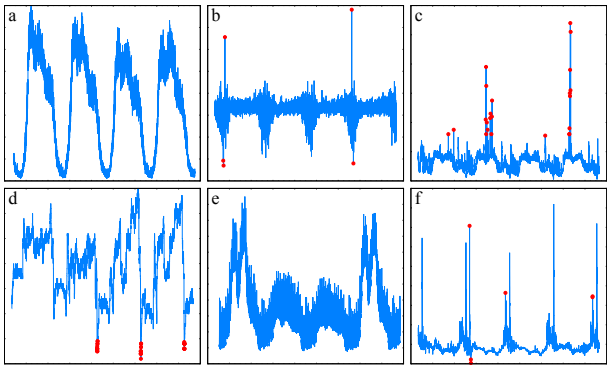


Fig. 1. Different KPI patterns in our datasets. The three KPIs in the first row are from dataset α, and those in the second row are from dataset β. Red points are anomalies labeled by operators.

In recent years, a great deal efforts have been devoted into the research of KPI anomaly detection. Existing methods can be classified into three categories, i.e., statistical regression based methods (e.g. Holt-winters [3], ARIMA [4]), ensemble supervised learning based methods (e.g. Opprentice [5]), unsupervised learning based methods (e.g., Donut [6], Bagel [7], Buzz [8]). However, taking accuracy, generality, efficiency and interpretability into consideration, which are the concerns of engineers, the performance of these methods are far from satisfying in practice.

Based on our domain knowledge collected from practice and discussions with engineers from several large corporations, we conclude that a practical KPI anomaly detector should satisfy the following requirements:

1. Unsupervised. It is difficult to obtain high-quality labeling for a large number of KPIs, which requires rich domain knowledge. Therefore, an unsupervised method is desired.
2. Generic. The algorithm must perform well on various KPIs.
3. Interpretable. The algorithm should be friendly to engineers, so it should avoid a large number of parameter tuning processes. In addition, users should be able to perceive the internal details of the model, and can adjust the model according to the actual situation to avoid repeated false

* Dan Pei is the corresponding author.

positives or false negatives.

4. Efficient. The resource requirements should be satisfiable and reasonable in commodity setups. The amount of data to be detected will increase year by year, but resources used for anomaly detection will not increase linearly, thus a large number of KPIs should be trained and detected with limited resources. Unsupervised methods based on deep generative models are not promising in this aspect.

To achieve the aforementioned design goals, in this paper, we propose a novel and practical KPI anomaly detection method based on two observations. First of all, after investigating many technology and financial corporations, we found that although corporate users are unwilling to learn parameter tuning, they can still accept a small amount of labeling overhead. For example, users are willing to spend 2-3 minutes labeling 20-30 anomalies to get more accurate results. Also, users expect to be able to automatically adjust the model through label feedback when a false positive occurs, to prevent similar situations from happening again. Second, inspired by IF-Active [9], which incorporates feedback and tree-based anomaly detectors, we believe that white-box and unsupervised algorithms, coupled with active learning, can improve accuracy. White-box algorithms such as Isolation Forest, mean that we know exactly how each anomaly is generated, thus we can accurately analyze which labels are more valuable for model improvement. By actively recommending curve segments with higher uncertainty, the same number of labels can provide richer feedback information, thereby improving the efficiency of model tuning.

Therefore, for anomaly detection to perform better in this context, we need an unsupervised, white-box, stable anomaly detection algorithm without parameter tuning. This algorithm is assisted by active learning to improve feedback efficiency and it can handle a large number of KPIs while requiring limited resources.

During the design of this solution, we encountered the following challenges.

1. Unsupervised white-box algorithms can be divided into two main categories. Basic statistical methods (such as ARIMA, 3-sigma, Holt-Winters) [3], [4] cannot be dynamically adjusted based on user feedback. Algorithms of constructing complex models (such as Isolation Forest) [10] need operators to design adjustment solutions based on the characteristics of the algorithms. We chose Robust Random Cut Forest (RRCF) [11], which was published later with better theoretical results. RRCF performs normally on our dataset, and its capacity must be lifted.
2. The RRCF algorithm frequently optimizes the tree model based on the detected outliers, so the performance is poor when processing a large number of indicators. We need to improve the speed of RRCF so that it can cope with real-world situations.
3. Labels are just pieces of outliers, and the overhead of labeling is quite heavy, so we need a carefully designed solution to convert labels into an optimization strategy for the model and to improve the efficiency of this transformation.

In this paper, we propose *iRRCF-Active* on top of RRCF, on which we made the following improvements to solve the above-mentioned challenges.

1. The algorithm capacity is improved because of several improvements to RRCF, which include redesigning $CoDisp$ calculation methods, optimizing the defects of the tree structure, and strengthening the feature selection stage. With these adjustments, *iRRCF-Active* can perform well on a variety of curves elaborated later in §IV.
2. The original RRCF is slow so it needs to be optimized. We improved the automatic update process of the original RRCF, and cached outliers from typical KPIs that are valuable for learning, thus the frequency and time of model updates are reduced.
3. RRCF itself cannot learn from feedback. We designed a label feedback strategy for RRCF. Besides, our approach proactively recommends the most valuable possible segments to users, to boost user's labeling efficiency.

The contributions of this paper can be summarized as follows:

1. To the best of our knowledge, *iRRCF-Active* is the first to propose a framework that uses a white-box unsupervised tree-model with the help of active learning in the field of time series anomaly detection.
2. Our implementation of this framework by improving RRCF results in lower latency and higher accuracy. *iRRCF-Active* improves the anomaly detection best F1-score by 0.19 to 0.5 for datasets α and β, and reduces the detection time by up to 58%.
3. We design an active learning solution for time series anomaly detection, which can actively recommend segments for labeling based on uncertainty, thereby enhancing the efficiency of transforming labels into model improvement.

II. Background

A. Key Performance Indicator

A key performance indicator(KPI) is a time series composed of $(timestamp, value)$ pairs. A KPI can be denoted as $\{x_1, x_2, \ldots, x_n\}$, where x_i is the value corresponding to time index t for $t \in \{1, 2, \ldots, n\}$, n is the timestamp of the last time this time series was monitored. KPIs can be collected from many sources, such as networks, transaction links, request logs, and other monitoring systems [5]. Compared with traditional time series, KPIs contain a larger amount of data which can reflect many applications and business changes, and KPIs with similar collection sources often have similar characteristics [12]. For example, our dataset contains many types of KPIs, as shown in Fig. 1. Among these types, Internet traffic KPIs often demonstrate seasonal patterns, and low-frequency trading KPIs usually have more glitches.

B. KPI Anomaly Detection

An anomaly in a KPI often appears as a sudden spike, sudden dip, or jitter, etc., as shown in red in Fig. 1, which

indicates an unexpected pattern that violates expectations based on history. Operators can deploy detectors according to their domain knowledge to identify KPI anomalies, but a KPI anomaly is difficult to be described with pre-defined rules [5], since it depends on the context in which standard formulas cannot represent them [13]. Consistent with the general definition of outliers [14], KPI anomalies are observations that differ significantly from other observations [15]. When labeling anomalies, the domain expert will visually scan the KPI to determine its normal pattern, and consider those points that are significantly different from the normal counterparts as abnormal [16]. Therefore, understanding the distribution of the normal data is the key to unsupervised anomaly detection.

Existing time series anomaly detection methods can be classified into three categories, i.e., statistical regression based methods, ensemble supervised learning based methods, unsupervised learning based methods.

1) Statistical regression based methods (e.g. Holt-winters, 3-sigma, etc.) [3], [4], [17], [18] have been proposed and improved, but each method fits to only KPIs with specific patterns. For example, Moving Average (MA) can only deal with KPIs that do not change drastically, and 3-sigma can only deal with KPIs with a concentrated distribution. Thus these methods are difficult to be generalized. Besides, after many iterations of improvements, these methods still demand extensive manual fine-tuning.

2) Supervised-learning based methods (e.g. Opprentice) [5], [19] ensemble several traditional statistical methods into a powerful classification model, which heavily rely on labeled data and the quality of labeling. However, it is time-consuming and tedious to acquire high-quality labels, considering the amount of KPIs and the requirement of expert knowledge for labeling [16], which prevents supervised learning based methods from being deployed in scale.

3) Unsupervised-learning based methods is pretty popular in recent years, for example Donut [6] based on deep generative models achieved outstanding results in seasonal KPI anomaly detection. Nevertheless, with the adoption of neural networks, the resources required for training are much more than that required by traditional methods, and GPU participation is often required. This amount of resource consumption is usually impractical in production environments with millions of KPIs to be monitored.

Other unsupervised learning based methods [10], [11], [20], [21] generally outperform statistical regression based methods, but are surpassed by deep generative model based methods [6], [8], [15], [22] in certain cases, such as KPIs with obvious periodicity. Note that even with the help of KPI clustering algorithms like ROCKA [23], unsupervised deep generative models still cannot handle a large number of KPIs. This is because ROCKA cannot guarantee a fixed number of KPI clusters. Once the number of clusters is specified to limit resource usage, the effectiveness of the algorithm may be reduced due to the diversity of the curves within the cluster. Therefore, clustering can only be used as an auxiliary means of improving resource utilization, not to mention that neural networks require additional GPU resources unlike other algorithms.

A comparative summary is shown in TABLE I, which will be detailed later in §IV. Therefore, none of the current algorithms can meet all the requirements above, and production anomaly detection has not been implemented in enterprise-level practice.

Yet, we notice that those unsupervised algorithms that do not use neural networks are not flawed in the above three characteristics, but they are not excellent enough to compete with unsupervised algorithms that use neural networks. However, due to the use of labels, the effects of supervised algorithms such as Opprentice are close to methods using neural networks like Donut. Obviously, one disadvantage of unsupervised learning is that they are unable to learn feedback from expert operators, in addition to the difficulty of promotion without enough apparent outliers. If they can somehow learn at least some expert knowledge, their results may be greatly improved, with fewer additional resources required than neural networks do.

TABLE I
COMPARISON AMONG ANOMALY DETECTION APPROACHES

With characteristic	1	2	3	4	*iRRCF-Active*
High Capacity	No	Yes	Some	Yes	Yes
Avoid parameters tuning	No	Yes	Yes	Some	Yes
Avoid Labeling	Yes	No	Yes	Yes	Yes
High detection speed	Yes	Some	Some	Some	Yes
Train with limited resources	Yes	Some	Yes	No	Yes
Adjustable	No	Yes	Yes	No	Yes

1: traditional statistic method, e.g., time series decomposition, ARIMA
2: supervised ensemble method, e.g., Opprentice
3: unsupervised method, e.g., RRCF
4: unsupervised method with deep generative model, e.g. Donut

C. Active Learning with Anomaly Detection

Active learning [24] utilizes a strategy to query the most valuable unlabeled samples and passes them to the experts for labeling, and trains a classification model based on this to improve accuracy. It is closely related to the problem we are trying to solve. In the time series anomaly detection scenario, labeling can improve the algorithm's effect, but it is relatively difficult to obtain. IF-Active [9] takes advantage of the Isolation Forest supplemented by a simple active learning strategy, which has produced good results, but it is not designed for time series anomaly detection scenarios. In this paper, we present three ideas for actively recommending labeling of time series, and we offer specific implementations based on RRCF. More complex active learning strategies may lead to better results and we will leave them for future work.

D. Clustering of KPIs

Millions of KPIs incur huge challenges for anomaly detection. Fortunately, many KPIs are similar becasue of their implicit associations and similarities [23]. For example, Fig. 2 shows the TPS metric collected from two different hosts that are quite similar in shape. If we can identify similar KPIs and group them into several clusters, the overhead that originates from KPIs quality can be reduced.

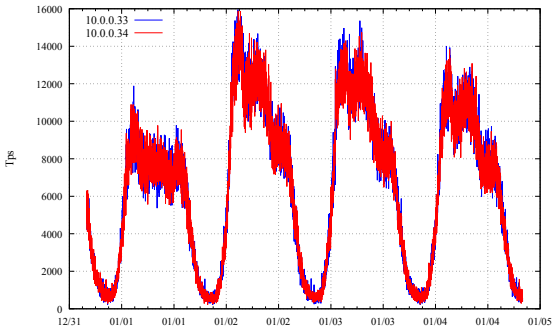


Fig. 2. The TPS metric of two different hosts

Time series clustering is a popular field that has attracted a lot of attention in the past 20 years. [25] summarizes a number of approaches on this topic. In this paper, we use Dynamic Time Wrapping (DTW) [26] to compute the similarity between KPIs, and use DBSCAN [27] to pre-cluster the KPIs. DBSCAN does not require a predefined number of categories, and can control the degree of similarity within categories by adjusting the clustering radius, which is suitable for our scenario. Therefore, we can train a model for a KPI group instead of a single KPI if possible, which will reduce training resources requirements.

E. Robust Random Cut Forest

Robust Random Cut Forest(RRCF) [11] was proposed to improve Isolation Forest [10]. RRCF is designed to handle large volumes of streaming data. It reduces the influence of irrelevant dimensions in the input data, and gracefully handles duplicates and near-duplicates that could otherwise mask the presence of outliers. In the process of constructing the tree, RRCF tends to select features with very large differences as the cut points. RRCF also maintains a rolling pool and utilizes each detected data to adjust the model. RRCF has already been implemented and used for anomaly detection in *Amazon Kinesis* real-time analytic engine [28]. Users can write SQL-like statements to implement anomaly detection for a single stream, but it still cannot process a large number of KPIs with low latency in production.

III. Architecture

The overall architecture of our proposed *iRRCF-Active* is demonstrated in Fig. 3. Obviously, our approach contains three components, i.e., training, detection and feedback. Specifically, in the stage of training, we first preprocess KPIs and extract powerful features from each KPI, then we novelly propose an improved RRCF (iRRCF) algorithm as outlier detector to overcome the drawbacks of original RRCF, and use the extracted features to train an initial iRRCF model. In the stage of detection, iRRCF could provide an anomaly score for each point, which characterizes the degree of anomaly. To enhance the detection performance, *iRRCF-Active* leverages the technique of active learning and provides some important KPI segments to operators, so that they could spend little time labeling some key points, which can significantly improve detection performance. In the stage of feedback, operators can check the anomalous points given by iRRCF and provide feedback (false positives and false negatives), so that the iRRCF model can be further enhanced.

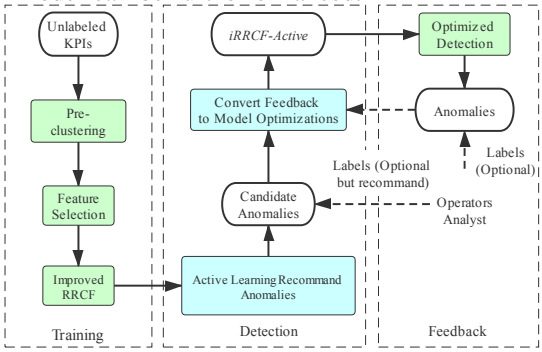


Fig. 3. Overall architecture of *iRRCF-Active*

A. Training

1) Preprocessing: Based on our observation, the number of KPIs is overwhelming in large companies, which brings great challenge for detecting anomalies for each KPI. A common practice is adopting the technique of time series clustering to cluster similar KPIs together [12]. In this way, similar KPIs with similar patterns could adopt a same anomaly detection algorithm. In our approach, we adopt Dynamic Time Warping (DTW) [26] to compute the similarity between two KPIs, and employ DBSCAN [27] to cluster the large number of KPIs. It is because DTW is state-of-the-art time series distance measurement, and DBSCAN does not require the number of clusters as prior knowledge. Besides, before KPI clustering, we need to preprocess each single KPI, including missing value imputation with linear interpolation and min-max normalization.

Notice that KPI clustering is an optional component in *iRRCF-Active*, and the main purpose is to save the time cost of training a model for each KPI individually. Its effect depends on the characteristics of the dataset. Therefore, the clustering step can be omitted on datasets without obvious commonalities.

2) Improved RRCF: The capacity of original RRCF is not enough for production, and its detection speed needs to be improved. In our *iRRCF-Active* framework, we novelly propose an improved RRCF algorithm (iRRCF), which can overcome some drawbacks of the original RRCF, and the major differences are displayed in TABLE II. In the following, we will introduce each improvement in detail.

Feature Representation. Original RRCF uses the most recent points in history as its features, so it is difficult to obtain time-related characteristics (such as periodicity, trend changes, etc.) of the data. We selected six classical and common time series features as candidate sets, as shown in TABLE III. These features cover general time series characteristics and are easy to calculate. Since different KPIs often have distinct statistical characteristics, these characteristics may not be suitable for or applicable to each KPI. Therefore, we design a set of statistical indicators and calculate their values for each KPI. Next, we determine which characteristics are appropriate to the KPI based on this set of indicators. The effectiveness of feature extraction and feature selection will be demonstrated in §IV.

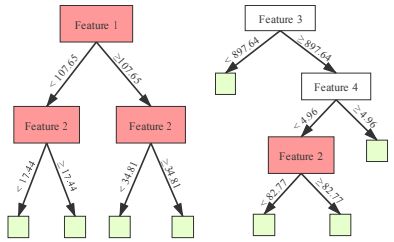
TABLE II
DIFFERENCES BETWEEN OUR PROPOSED IRRCF AND ORIGINAL RRCF

Process	Original RRCF	iRRCF
Features representation	Recent points in history	Time series statistical features with feature selection
Node cut dimension selection	Only consider range of dimensions	Also consider maximum distance of dimensions
Node cut threshold selection	Cut at randomly selected interval	Cut at the most sparse interval
Anomaly score calculation	Only consider siblings	Also take node depth into consideration

TABLE III
FEATURES AND THEIR APPLICABLE CONDITIONS

Feature	Applicable Condition
Median	Few spikiness [29]
Standard deviation	Stationary
Difference from previous period point	Seasonal
Difference from previous point	Trendiness
Second-order difference from the first point	Meet the above 4 points
Third-order exponential average	Predictability

Node Cut Dimension Selection. After extracting the multi-dimensional features of the training set, the original RRCF will randomly select dimensions and cut on them to build multiple decision trees. The tree in Fig. 4(a) first selects more distinguishing features (marked in red), so it only needs two layers to distinguish all samples. Due to the poor feature discrimination, the level of the tree in Fig. 4(b) is higher. This difference will be more obvious when the number of samples is larger. The original RRCF algorithm tends to choose the feature with greater range as the criterion for cut. That is, the probability of choosing the i-th dimension with feature set S is $\frac{l_i}{\sum_j l_j}$, where $l_i = \max_{x \in S} x_i - \min_{x \in S} x_i$ and S is in ascending order. Such a basis, however, does not well reflect the distinguishing degree of features. A series that exhibits periodic characteristics tends to have a larger range than that of those with stable patterns. Some features may have a large range, but normal data is evenly distributed. Some features have a small range, but abnormal points and normal points are clearly gathered respectively, and cut in these features can help distinguish anomalies more directly. In order to increase the ability of feature discrimination in iRRCF, we introduce the maximum distance $g_i = max_{x \in S} x_j - x_{j-1}$ for the dimension i. We choose an arbitrary dimension that is proportional to $p_i = \frac{g_i}{\sum_j g_j} + \frac{l_i}{\sum_j l_j}$.



(a) Fig. A (b) Fig. B

Fig. 4. Different node cut dimension selection strategies lead to trees with distinct structures and accuracy. Distinguishing features are marked in red.

Node Cut Threshold Selection. Original RRCF randomly chooses a value between maximum and minimum feature value as the cut threshold. For a feature with high discrimination, its data often has aggregated distribution, so the result of cutting in sparse is much better than that in dense. Therefore, in iRRCF, after selecting the dimensions for cut, we focus on the data distribution specifically in this dimension and we increase the probability of cutting in the sparse distribution. Formally, the process can be summarized as follows:

1) Divide the feature extracted from the training set into N intervals $[l_0, h_0, l_1, h_1, \ldots, l_{N-1}, h_{N-1}]$
2) Compute the density of feature in every interval $d_i = Count(p, p \in [l_i, h_i])$
3) Choose a random interval i proportional to $\frac{d_i}{\sum_j d_j}$
4) Choose $X_i \sim Uniform[l_i, h_i]$

Anomaly Score Calculation. In RRCF, each sample will fall to a leaf node in the tree and original RRCF forest will calculate an anomaly score $CoDisp$ for each sample to characterize the degree of anomaly. The $CoDisp$ is calculated according to the following steps:

1) Find the leaf node $Node$ of the sample x_i in each tree T
2) Count the number of samples in the subtree rooted by sibling and parent of $Node$, denoted by $S_{Node.sibling}$ and $S_{Node.parent}$, and calculate $CoDisp_{Node} = \frac{S_{Node.sibling}}{S_{Node.parent}}$
3) Go up one level in the tree, $Node = Node.parent$
4) Repeat 2 to 3 for N times, where N is the number of features used
5) $CoDisp_T$ is the max value of each $CoDisp_{Node}$
6) The final $CoDisp_{x_i}$ for sample x_i is $\overline{CoDisp_T}, T \in forest$

Due to space limit, more details about anomaly score calculation of original RRCF can be found in [11]. Obviously, the samples represented by sibling nodes are different from the current samples in specific dimensions. If the number of sibling samples is larger, the probability that the current sample is an outlier is also higher. However, such a strategy does not take the depth of the nodes in the tree into account. Usually, the greater the depth, the larger the number of cuts and the lower the possibility of anomalies. Therefore, in iRRCF, we incorporate the node depth when calculating the anomaly score, that is, the $CoDisp_{Node}$ in step 2) above is adjusted to $CoDisp_{Node} = \frac{S_{Node.sibling}}{S_{Node.parent} \times Node.depth}$.

B. Detection

After the stage of model training, iRRCF model will provide an anomaly score for each point including both historical points and incoming online data points. Intuitively, incorporating some labels into our unsupervised learning models can

significantly improve the performance, since labels can instruct the model to know what are anomalies accurately. Therefore, to enhance our approach, we propose to incorporate some labeling into iRRCF, so that iRRCF can be optimized to obtain better detection performance.

1) Candidate Labels Recommendation: In order to collect user feedback and improve labeling efficiency, in *iRRCF-Active*, we propose an active learning method to recommend to operators those anomalous fragments that can enhance model optimization efficiency. We sort the training set based on anomaly scores given by iRRCF, and select anomalous fragments at different positions according to three strategies. The anomalies of time series data usually appear in the form of continuous time segments. Hence, instead of letting the operator give feedback to a single point, we recommend continuous abnormal points to the operator as an abnormal segment. There are three popular recommendations strategy in the field of active learning, which are introduced as follows.

a) Select 30 the most abnormal segments. Obtaining such labels can further confirm obvious anomalies and eliminate false positives.
b) Select 30 the most uncertain abnormal fragments. Anomaly detection of time series is a binary classification problem. Gaining such labels can further improve the boundaries of the classification result and can also improve the accuracy of identifying ambiguous anomalies.
c) Divide the data into 10 groups based on anomaly scores, and select 3 abnormal fragments in each group with a medium probability. Obtaining such labels can capture the preferences of operators towards different degrees of abnormality produced by the algorithm evaluation, and then can help determine which group is more possible to be the boundary point between abnormal and normal cases.

Detailed implementations of these strategies in iRRCF are displayed in Fig. 5. Based on the experiments in Section §IV, strategy a is demonstrated to be more effective than other two strategies. Therefore, we adopt strategy a in our model.

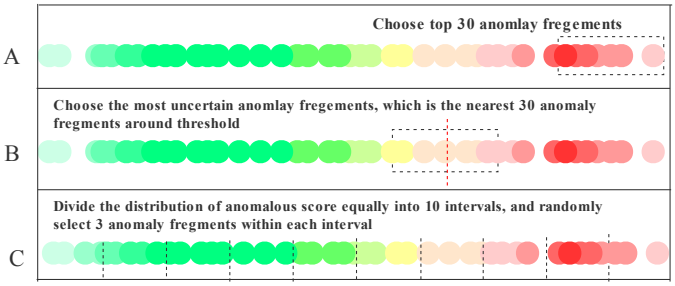


Fig. 5. Candidate labels recommendation process

2) Model Optimization: Original RRCF maintains a dynamic tree collection based on real-time data. When a new data point arrived, RRCF performs an insertion process and uses this data point to update each tree in the model. This process incurs additional calculations which slow down the inspection process. Considering the number of anomalies is relatively small in practice, the tree can be selectively updated according to the degree of abnormality of the data. In order to ensure that the model can cover the changes of extreme features in time, two types of data points need to be updated. One type is the first anomalous point in a continuous anomalous segment, and the other is a point that is judged to be normal but extremely abnormal in a specific dimension. We collectively refer to these two types of points as $extreme$. In addition, in order to obtain the slowly changing trend of some special curves, *iRRCF-Active* will also update the model with normal points at a lower frequency when the two types of data points have not appeared for a long time. Our scheme is shown in Algorithm 1.

Algorithm 1 Obtain Anomaly Score and Update Model

Require: New data point p, the original tree $tree$

function GET SCORE(p, $tree$)
  $Node \leftarrow tree.root$
  if $isinstance(Node, LEAF)$ **then**
    if p is $extreme$ or already skip 100 points **then**
      INSERT_POINT(p, $tree$)
    end if
    return CODISP(p)
  end if
  if $p[Node.dim] <= Node.threshold$ **then**
    return GET SCORE(p, $Node.left$)
  else
    return GET SCORE(p, $Node.right$)
  end if
end function

C. Feedback

In order to benefit *iRRCF-Active* from user feedback, we design two model optimization strategies based on labels. The use of labeling is not limited to the training phase. Users may apply a set of historically accumulated labels in the training phase. After starting real-time detection, they may label a single case due to poor results. Therefore, the feedback strategy must be simultaneously applicable to a group or a single label.

a) Original RRCF includes multiple trees that are randomly constructed, and their classification accuracy scores are different. With the help of user feedback, *iRRCF-Active* evaluates which trees are more reliable in terms of classification, and accordingly give higher weights to these trees. The process of adjusting model based on a set of labels is described in Algorithm 2.
b) Original RRCF requires that the proportion of the normal data is much larger than that of the anomalous data. In *iRRCF-Active*, the influence of normal samples is boosted by repeatedly applying the normal samples 10 times to update the model. At the same time, for samples marked as abnormal, the node corresponding to the sample in each tree can be marked as abnormal. In subsequent detection, if a sample corresponds to abnormal nodes in more than half of the trees, it can be determined as abnormal. Therefore, *iRRCF-Active* does not update with a point that has been determined to be abnormal. When there are many abnormal points in labels, the data requirements of original RRCF

will not be destroyed, and the model can also learn the characteristics of the abnormality.

Algorithm 2 Update Weight of Trees based on Labels

Require:

Number of trees in forest m
Number of labels n
Anomaly score metric $CoDisp\_M[i][j]$ for label i calculated from tree j
labels provided by users $label$
function GET TREE WEIGHT(m, n, $CoDisp\_M$, $label$)
    $tw \leftarrow Zeros(1, m)$
    for $i = 0 \rightarrow n$ **do**
        if $label[i] == 1$ **then**
            $tw[:] \leftarrow tw[:] + \delta \times CoDisp\_M[i, :]$
        end if
    end for
    return tw
end function

Based on the experiments in Section §IV, Strategy a is a better fit for our improved RRCF.

IV. EVALUATION

A. Datasets

Our experiments are conducted on two datasets. Dataset α is a public dataset(AIOps competition [30], [31]) with 29 KPIs and 5,922,913 points that consist of 79,554 anomaly points, corresponding to 1,212 consecutive anomalous segments. Dataset β contains 10 well-maintained KPIs from a large commercial bank with 375,847 points that consist of 701 anomaly points, corresponding to 43 consecutive anomalous segments. The sampling interval for KPIs in these two datasets is 1 minute, the time span of dataset α is 7 months, and the time span of dataset β is 1 month. The labels of each KPI in two datasets are manually confirmed by experienced operators.

B. Evaluation Metrics

Following [6], [7], we adopt the revised F1-score proposed by [6] as the evaluation metric. In the KPI anomaly detection scenario, the real failure is often a continuous anomalous segment, and the operator is more concerned about whether the fault notification is timely. Therefore, as long as the anomaly detection method can trigger an alarm within a short period of time after the failure starts (*i.e.*, before the maximum allowable delay), the modified F1-score considers that the entire anomalous segment is successfully detected. Similarly, if the first alarm given by the algorithm for a fault happens later than the maximum allowable delay, the entire anomalous segment corresponding to the fault is considered as false negatives. Figure 6 illustrates the way of measurements described above. For convenience, we still refer to the modified F1-score as F1-score.

Similar to [6], [7], [32], in order to show the best potential performance of the model, we use the best F1-score, which is calculated from the threshold to obtain the best performance on the **test** set. In other words, we use all possible thresholds to calculate the improved F1-score on the test set and use the best score as our evaluation indicator.

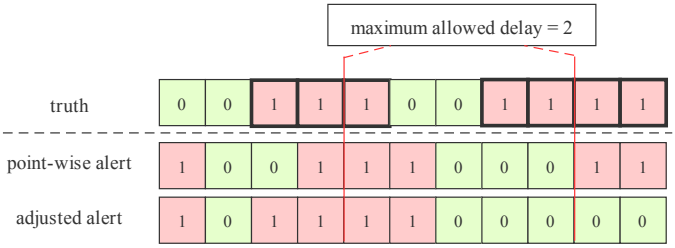


Fig. 6. Illustration of the evaluation metric

During the experiment, we divided each KPI equally with respect to time. The first half is used as the training set and is not used for testing. The second half is used as the test set and is unknown during the training process. During the test, the second half of each KPI is used to simulate an online detection process, and no information of the next data point is utilized when detecting the previous point.

All experiments are performed on all KPIs and repeated for 10 times. They were run on a same machine with each KPI evaluated serially. The machine has 22 CPU cores (2.2 GHz per core) with 48GB RAM.

All time units of training and detection in the experiment are the average minutes consumed by each KPI in the dataset.

C. Overall Performance

To demonstrate the effectiveness of our proposed *iRRCF-Active* in KPI anomaly detection, we compare it with several state-of-the-art anomaly detection methods, i.e., IF-Active [9], original RRCF (traditional unsupervised algorithm), Random Forest (RF, classic supervised method), and Donut (unsupervised algorithm based on deep generative model). The reason why we choose RF instead of Opprentice [5] is because the latter needs to calculate a large number of features, which is extremely time-consuming and unrealistic to apply in practice. The comparison results are displayed in TABLE IV.

TABLE IV
OVERALL PERFORMANCE

Method	Best F1-score		Train Time		Detection Time	
	α	β	α	β	α	β
Donut	0.72	0.46	10.9	1.7	2.1	0.43
IF-Active	0.76	0.58	1.4	0.43	1.2	0.25
Isolation Forest	0.58	0.28	**0.17**	**0.09**	**0.11**	**0.06**
Random Forest	0.69	0.44	0.4	0.14	0.5	0.07
Original RRCF	0.70	0.31	0.48	0.37	12.1	3.8
iRRCF-Active	**0.89**	**0.81**	0.93	0.88	7.8	1.6

Obviously, our *iRRCF-Active* can achieve the best performance, with 0.89 and 0.81 F1-score on average on each dataset, larger than compared methods. The training time of *iRRCF-Active* is slightly longer than other unsupervised methods, but it is still acceptable and far below Donut. The detection time of *iRRCF-Active* is reduced by up to 58% compared to the original RRCF, which further improves the processing capacity of streaming data.

D. Feature Selection and Clustering

In TABLE V, we compare the effect of feature selection on RRCF. Experiments show that the six features we selected and the feature selection stage all improve the effectiveness of RRCF.

We also tried the pre-clustering introduced in §III-A1 on both datasets, and used the features selected by the KPI closest to the cluster center as the features selected by the cluster. The 29 KPIs of dataset α are clustered into 12 groups, and the 10 KPIs of dataset β are clustered into 6 groups. As the number of models to be trained is reduced, the time consumed for training is also significantly reduced more than 50%. Large internet companies or banks will have far more KPIs than our dataset. Creating a separate model for each cluster can significantly reduce modeling costs, and the gains achieved in reality will far exceed the proportion we evaluate in experiments.

TABLE V
EFFECTIVENESS OF FEATURE SELECTION AND CLUSTERING

Method	Best F1-score		Train Time		Detection Time	
	α	β	α	β	α	β
Original RRCF	0.70	0.31	0.48	0.37	12.1	3.8
1	0.73	0.44	0.51	0.44	13.9	4.0
2	**0.74**	**0.47**	0.41	0.32	11.2	3.6
3	0.71	0.40	**0.2**	**0.13**	**10.9**	**3.5**

1: Original RRCF with all 6 features mentioned in §III-A2
2: Method 1 with feature selection
3: Method 2 with pre-clustering stage in §III-A1

However, since the datasets we use are not collected from a single source, and the internal KPIs are not quite similar, the trade-off after clustering is also high. Therefore, to obtain better results in the following experiments, we did not choose to turn on the optional stage.

E. Improved RRCF

We evaluated the RRCF optimizations mentioned in §III-A2, and the results are presented in TABLE VI. As can be seen, most of the optimizations for the algorithm have achieved good results. However, comparing the results using all optimizations with each individual optimization, it can be seen that the improvement in performance is not superimposed. In dataset α, the optimization effect of §III-A2 has approached the effect of overall optimization. This does not mean that only this optimization point is the most effective. In dataset β, the improvement of §III-A2 is the only one of all optimizations that does not increase, but the overall optimization improvement is still outstanding.

The improvement of the detection process in §III-B2 significantly reduces the time consumed for detection, and this reduction in dataset β has even reached 65%. At the same time, because the frequency of using normal points to update the tree is greatly reduced, the algorithm's sensitivity to some trend changes that fall within the normal range will decrease, so the effect is slightly reduced.

It is worth noting that the effect of §III-B2 in dataset β has been improved compared to that before improvement. Our analysis found that there are many slight spikes and dips in dataset β. Most of them are in the normal range, but a few abnormal extreme values are mixed in these normal points. In the original RRCF, all points are used to update the tree, thus the recall rate of the model is reduced compared to when the model is only updated by those extreme values.

TABLE VI
EFFECTIVENESS OF RRCF

Method	Best F1-score		Train Time		Detection Time	
	α	β	α	β	α	β
1	0.74	0.47	**0.41**	**0.32**	11.2	3.5
2	0.75	0.48	0.51	0.37	11.9	4.1
3	0.78	0.43	0.58	0.44	11.2	4.0
4	0.76	0.48	0.86	0.48	12.0	4.3
5	**0.79**	0.59	0.93	0.74	12.0	4.3
6	0.78	**0.65**	0.93	0.75	**7.8**	**1.5**

1: Original RRCF with feature selection
2: Method 1 with $CoDisp$ calculation optimization in §III-A2
3: Method 1 with node cut dimension selection optimization in §III-A2
4: Method 1 with node cut threshold selection optimization in §III-A2
5: Method 1 with all optimizations above
6: Method 5 with model update progress optimization in §III-B2

F. Feedback Strategy

We propose two different user feedback strategies in §III-C. TABLE VII shows the improvement effects of these two strategies and their combinations after applying all the algorithmic optimizations. The number of labels used for model optimization is consistent with that of actively recommended labels in the overall scheme of *iRRCF-Active*, and the labels used are randomly selected from the label set. The results demonstrate that Strategy a in §III-C is more suitable for the improved RRCF, since it performs better than the other two strategies with same labels on both datasets.

TABLE VII
USER FEEDBACK STRATEGY

Method	Best F1-score		Train Time		Detection Time	
	α	β	α	β	α	β
1	0.78	0.65	0.93	**0.75**	**7.8**	**1.5**
2	**0.84**	**0.74**	0.96	0.83	8.0	1.6
3	0.83	0.71	**0.89**	0.77	8.1	1.7
4	0.82	0.69	0.93	0.78	8.3	1.7

1: *iRRCF-Active* without any user label feedback strategy
2: *iRRCF-Active* with label feedback strategy a in §III-C
3: *iRRCF-Active* with label feedback strategy b in §III-C
4: *iRRCF-Active* with both strategies

G. Label Recommendation Strategy

We propose three points of active recommendation labeling and specific implementations for RRCF in §III-B1. TABLE VIII compares the labeling quality recommended by these three strategies under the same optimization conditions and the best labeling feedback strategy a in §III-C. The results illustrate that the implementation of Strategy a in §III-B1 is more suitable for the improved RRCF.

TABLE VIII
LABEL RECOMMENDATION STRATEGY

Method	Best F1-score		Train Time		Detection Time	
	α	β	α	β	α	β
1	0.78	0.65	0.93	**0.75**	7.8	**1.5**
2	**0.89**	**0.81**	0.93	0.88	7.8	1.7
3	0.86	0.78	**0.89**	0.82	**7.6**	1.6
4	0.83	0.77	0.90	0.81	7.9	1.6

1: *iRRCF-Active* without any user label feedback strategy
2: *iRRCF-Active* with label recommendation strategy a in §III-B1
3: *iRRCF-Active* with label recommendation strategy b in §III-B1
4: *iRRCF-Active* with label recommendation strategy c in §III-B1

The experimental results show that our active learning phase improves the best F1-score by at least 0.1 on both datasets.

V. Conclusion

This paper proposes a white-box and unsupervised KPI anomaly detection method based on RRCF model and active learning. Specifically, we novelly design several improvements to overcome the drawbacks of original RRCF in our scenario. Besides, due to the labeling overhead, we leverage the technique of active learning and operators only need to a few data, which can significantly improve the performance of our method. Extensive experiments on a large-scale public dataset and a private dataset from a large commercial bank demonstrate that our proposed *iRRCF-Active* outperforms various state-of-the-art methods, and the best F1-score can achieve 0.81 and 0.89 on two datasets. Also, each component in *iRRCF-Active* has also been demonstrated to be effective. In summary, our proposed *iRRCF-Active* is indeed practical and can assist the real deployment of KPI anomaly detection in practice.

VI. Acknowledgment

We appreciate Lingzhi Wang, Christopher Zheng and Jing Zhu for their insightful suggestions and support. Suggestions and valuable feedback from the reviewers also helped us improve our work. This work has been supported by the National Key R&D Program of China(2019YFB1802504) and the Beijing National Research Center for Information Science and Technology (BNRist) key projects.

References

[1] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, Z. Zang, X. Jing, and M. Feng, "Funnel: Assessing software changes in web-based services," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 34–48, 2016.

[2] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu *et al.*, "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," *IEEE Access*, vol. 6, pp. 10 909–10 923, 2018.

[3] C. Chatfield, "The holt-winters forecasting procedure," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 27, no. 3, pp. 264–279, 1978.

[4] H. Z. Moayedi and M. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," in *2008 International Symposium on Information Technology*, vol. 4. IEEE, 2008, pp. 1–6.

[5] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 211–224.

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

[7] Z. Li, W. Chen, and D. Pei, "Robust and unsupervised kpi anomaly detection based on conditional variational autoencoder," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–9.

[8] W. Chen, H. Xu, Z. Li, D. Peiy, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised anomaly detection for intricate kpis via adversarial training of vae," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1891–1899.

[9] M. A. Siddiqui, A. Fern, T. G. Dietterich, R. Wright, A. Theriault, and D. W. Archer, "Feedback-guided anomaly discovery via online optimization," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2200–2209.

[10] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[11] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *International conference on machine learning*, 2016, pp. 2712–2721.

[12] 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.

[13] M. Thill, W. Konen, and T. Bäck, "Online anomaly detection on the webscope s5 dataset: A comparative study," in *2017 Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, 2017, pp. 1–8.

[14] A. C. Atkinson and D. M. Hawkins, "Identification of outliers," *Biometrics*, vol. 37, no. 4, p. 860, 1981.

[15] G. Yu, Z. Cai, S. Wang, H. Chen, F. Liu, and A. Liu, "Unsupervised online anomaly detection with parameter adaptation for kpi abrupt changes," *IEEE Transactions on Network and Service Management*, 2019.

[16] N. Zhao, J. Zhu, R. Liu, D. Liu, M. Zhang, and D. Pei, "Label-less: A semi-automatic labelling tool for kpi anomalies," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1882–1890.

[17] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang, "A provider-side view of web search response time," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 243–254.

[18] F. Knorn and D. J. Leith, "Adaptive kalman filtering for anomaly detection in software appliances," in *IEEE INFOCOM Workshops 2008*. IEEE, 2008, pp. 1–6.

[19] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1939–1947.

[20] Y. Zhang, S. Debroy, and P. Calyam, "Network-wide anomaly event detection and diagnosis with perfsonar," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 666–680, 2016.

[21] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[22] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, 2016.

[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] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[25] T. W. Liao, "Clustering of time series data—a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[26] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[27] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[28] M. Bartos, A. Mullapudi, and S. Troutman, "rrcf: Implementation of the robust random cut forest algorithm for anomaly detection on streams," *Journal of Open Source Software*, vol. 4, no. 35, p. 1336, 2019.

[29] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale unusual time series detection," in *2015 IEEE international conference on data mining workshop (ICDMW)*. IEEE, 2015, pp. 1616–1619.

[30] http://iops.ai/dataset_detail/?id=10.

[31] http://iops.ai/competition_detail/?competition_id=5&flag=1.

[32] N. Zhao, J. Zhu, Y. Wang, M. Ma, W. Zhang, D. Liu, M. Zhang, and D. Pei, "Automatic and generic periodicity adaptation for kpi anomaly detection," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1170–1183, 2019.