Robust KPI Anomaly Detection for Large-Scale Software Services with Partial Labels

Shenglin Zhang[§‖], Chenyu Zhao[§], Yicheng Sui[§], Ya Su[‡*]
Yongqian Sun[§‖], Yuzhi Zhang[§‖], Dan Pei[†], Yizhe Wang[¶]
[§]Nankai University, {zhangsl, sunyongqian, zyz}@nankai.edu.cn, {zhaochenyu, suiyicheng}@mail.nankai.edu.cn
[‡]Kuaishou Technology, suya@kuaishou.com, [†]Tsinghua University, BNRist, peidan@tsinghua.edu.cn
[¶]Lanling Information Technology (Shijiazhuang) Co. Ltd, wangyizhe_2008@126.com
[‖]Tianjin Key Laboratory of Operating System

***Abstract*—To ensure the reliability of software services, operators collect and monitor a large number of KPI (Key Performance Indicator) streams constantly. KPI anomaly detection is vitally important for software service management. However, none of supervised learning methods, semi-supervised learning methods, transfer learning methods, or unsupervised learning methods achieve accurate anomaly detection for the large-scale, diverse, dynamically changing KPI streams with little labeling effort. In this paper, we propose *PUAD*, a PU learning-based method, to achieve accurate KPI anomaly detection requiring a few partial labels. It integrates clustering, PU learning, and semi-supervised learning to minimize labeling effort and improve anomaly detection accuracy simultaneously. Additionally, we propose a novel active learning method that selects the samples most likely to be positive in each iteration to avoid false alarms. We apply 208 real-world KPI streams collected from a large-scale software service provider to evaluate the performance of *PUAD*, demonstrating that it achieves a close F1-score to supervised learning methods with much fewer manual labels, and greatly outperforms semi-supervised learning methods, transfer learning methods, and unsupervised learning methods.**

***Index Terms*—Anomaly detection, Metrics, AIOps, PU learning, Active learning**

I. Introduction

With the rapid development of software technology, software services, e.g., online games, social networks, search engines, have witnessed an explosion in both scale and complexity. To ensure the reliability of software systems, millions of KPI (key performance indicator) streams, including user-perceived metrics, e.g., response delay, queries per second (QPS), failure ratio, and system-level metrics, e.g., CPU utilization, memory utilization, network throughput, are constantly monitored and collected at equal-space timestamps [1], [2]. Anomalies in KPI streams usually manifest as spikes, level shifts, ramp up/down and indicate potential service failures, e.g., software bugs, failed updates, network overload, external attacks [3]–[7]. KPI anomaly detection, which aims to find the anomalous behaviors in KPI streams, is vitally important for operators to proactively detect software failures and timely trigger failure diagnosis to mitigate loss. Because today's software services are at large scale and very complex, KPI streams have various shapes and KPI anomalies are highly diverse. Moreover, due to the continuous software updates and configuration changes, KPI streams and their anomalies can dramatically change over time [1], [2], [8], [9].

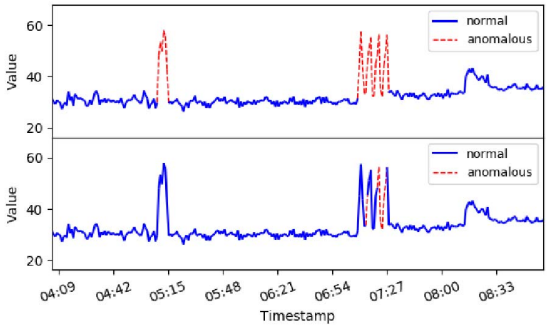


Fig. 1: The manual labels needed for semi-supervised learning methods (upper one) and PU learning methods (lower one)

A great number of KPI anomaly detection algorithms have been proposed over the years [6], [7], [10]–[16], which are mainly divided into supervised learning and unsupervised learning methods. Unfortunately, none of them are feasible to deal with the above scenario well:

(1) For supervised learning methods, *all anomalous and normal samples of each KPI stream* in the training set should be manually labeled [10], [12]. Due to the large scale and high diversity of KPI streams, this labeling work, if not impossible, is time-consuming and labor-intensive.

(2) Unsupervised learning methods require no labels. However, they either suffer from low accuracy [17] or require large amounts of training data for each new KPI stream (e.g., six months worth of data) [13]–[16], which cannot be used for the scenario where the patterns of KPI streams dynamically change over time because of software upgrades and/or configuration changes [2], [11].

Apart from the above two categories of methods, semi-supervised learning methods (e.g., ADS [18]) or transfer learning methods [11] (e.g., ATAD [11]) have also been applied for KPI anomaly detection, both of which enable accurate anomaly detection for a large number of KPI streams, without

*Ya Su is the correspondence author.

2332-6549/21/$31.00 ©2021 IEEE
DOI 10.1109/ISSRE52982.2021.00023

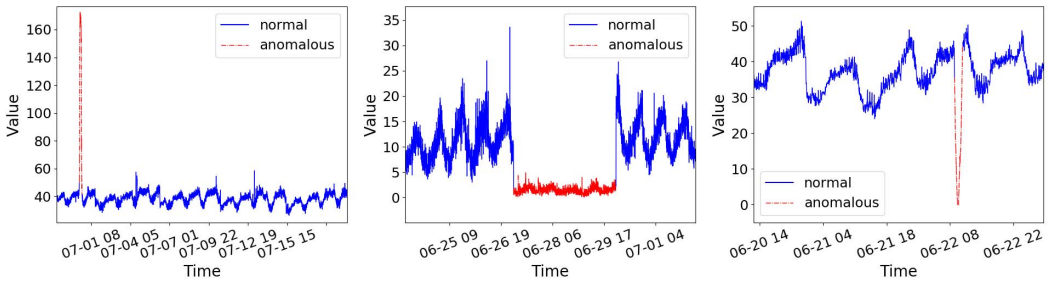


Fig. 2: Examples of anomalies in KPI streams. The red parts in the KPI stream denote anomalous segments

manual algorithm selection or parameter tuning. Moreover, they can be quickly initialized for those KPI streams whose patterns dynamically change over time. However, they still need a large number of KPI streams with high-quality ground truth. Specifically, *all the KPI streams (time series segments) in the training set are required to be carefully labeled and periodically updated*. Considering the diverse types of KPI streams and frequent updates of their patterns, it is still tedious for operators to examine the KPI streams back and forth periodically.

To solve the above problem, a natural idea is to randomly label some anomalies manually and learn anomaly patterns from these labels, which is the core idea of PU learning (Positive-Unlabeled learning) [19]. Specifically, for a KPI stream in the training set, PU learning only require labeling part of anomalous segments and does not need to label all the anomalous segments. In this way, operators' manual labeling work is minimized. For example, Fig. 1 shows the manual labels needed with semi-supervised learning/transfer learning and those with PU learning, respectively. We can see that PU learning can greatly reduce the labeling effort compared to semi-supervised learning/transfer learning.

However, applying PU learning for KPI anomaly detection faces the following two challenges:

(1) KPI streams are large in number and diverse in the pattern. On one hand, if we train a PU learning model for each KPI stream, the overall labeling effort is still very large. On the other hand, if we train a PU learning model for all KPI streams, the model will suffer from low accuracy because different KPI streams have different patterns and thus their most suitable anomaly detectors and parameters can be significantly different.

(2) The performance of PU learning methods is limited due to insufficient labels, and naturally, active learning can be applied. However, the current active learning methods, which usually label anomalous samples near classification boundary [20], [21], tend to cause normal samples to be misclassified as anomalies and thus generate many false alarms.

In this work, we propose *PUAD*, a PU learning-based anomaly detection framework, to solve the above challenges and accurately and efficiently detect KPI anomalies with a small number of partial labels. *PUAD* consists of three major components: (1) **Clustering**. Clustering KPI streams according to shape similarity. For each cluster, operators manually label some anomalous segments for its centroid KPI stream. Because the number of clusters is much smaller than that of KPI streams, operators' labeling effort is minimized. Moreover, a newly emerging KPI stream can be assigned into one existed cluster based on its shape similarity with each cluster centroid. (2) **PU learning**. For each cluster centroid, *PUAD* applies PU learning to build a binary classifier from positive (i.e., anomalous) and unlabeled samples. It then adopts a novel active learning method to obtain reliable positive samples interactively in several iterations. In this way, we can obtain reliable anomalous and normal labels for each cluster centroid KPI stream. (3) **Semi-supervised learning**. For each KPI stream, we train a semi-supervised learning-based model according to the labels of its cluster centroid KPI stream. Its anomalous behavior will be detected based on this model.

The contributions of this paper are summarized as follows:

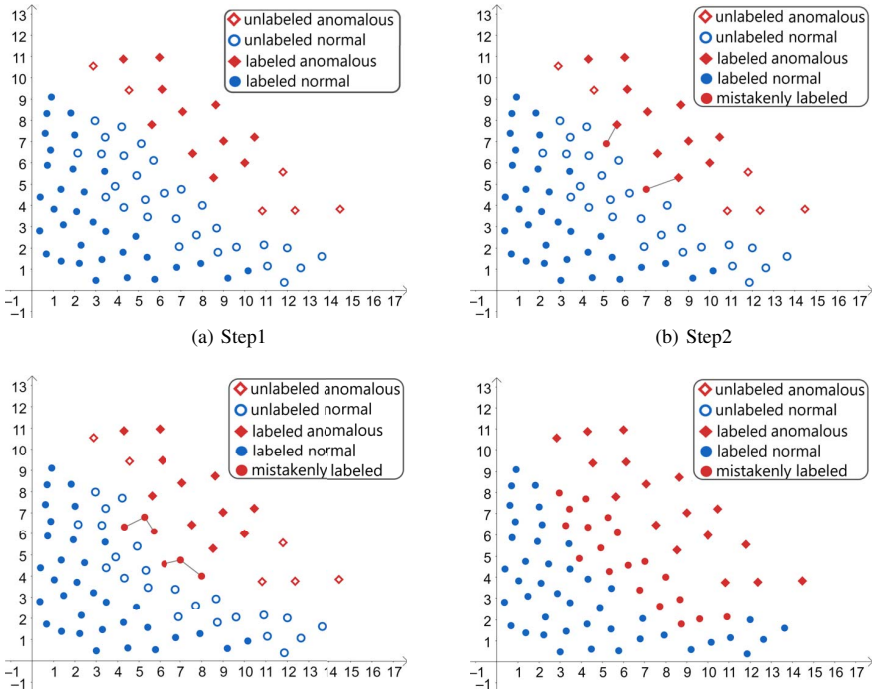
- To address the first challenge, *PUAD* integrates clustering, PU learning, and semi-supervised learning, which together minimize operators' labeling effort and achieve high accuracy simultaneously.
- To address the second challenge, we propose a simple yet effective active learning model, which selects samples that are most likely to be positive in each iteration instead of those that are close to the classification boundary. In this way, a large number of false alarms are avoided.
- We apply 208 real-world KPI streams collected from a large-scale software service provider to evaluate *PUAD*'s performance. *PUAD* achieves a close F1-score to a well-known supervised learning method and greatly outperforms two unsupervised learning-based methods, one semi-supervised learning-based method, and one transfer learning-based method by 103.7%, 786.2%, 28.7%, and 482.5%, respectively. To get readers better understand our work, we have made our code publicly available[1].

II. Background and Challenges

A. Definitions

In this section, we define some key terms about anomaly detection.

[1]https://github.com/PUAD-code/PUAD



(a) Step1 (b) Step2

(c) Step3 (d) Step4

Fig. 3: The labeling processes of an active learning method which labels samples near classification boundary

KPI streams: The rapid development of software services has brought great convenience to our daily life. To ensure the reliability of a software service, operators continuously monitor the KPIs (e.g., QPS, error ratio, CPU utilization, network throughput) of every component including microservices, databases, virtual machines, containers, physical machines, etc. A KPI stream is a time series containing the monitoring data of a KPI. It is collected at equal-space timestamps and defined as $X = x_1, x_2, \ldots, x_n$, where x_i is the observation at the ith time point and n is the length of X [22]. Because of the great number of components in large-scale software service and each component have diverse types of KPIs, a great many KPI streams are generated for each software service.

Anomaly: An anomaly in a KPI stream denotes that one of its segments deviates from normal behaviors, such as a spike, a level shift, or a ramp up/down. Fig. 2 shows three examples of anomalies in KPI streams [22]. Usually, different types of KPI streams have different patterns when they become anomalous.

Anomaly detection: Anomaly detection for KPI stream X is to determine whether a segment $x_{i-w+1:i}$ is anomalous (let $y_i = 1$ denotes an anomaly and $y_i = 0$ otherwise), where w is the length of the sliding window required for anomaly detection. For each segment, an anomaly detection method typically computes an anomaly score to indicate its anomalous possibility, and usually a threshold is set to determine whether it is anomalous or not. If its anomaly score exceeds this threshold, it will be regarded as an anomaly.

B. Challenges

Large-scale and diverse KPI streams. If we train a PU learning model for each KPI stream following [7], [10], [13]–[16], too many manual labels are needed because of the large number of KPI streams. However, if we try to train a universal PU learning model for all KPI streams, the model will suffer from low accuracy because KPI streams are highly diverse and it is very difficult, if not impossible, to find the combinations of anomaly detectors and parameters that are suitable for all KPI streams.

Active learning strategy. Naturally, we can adopt active learning to help a PU learning method obtain more normal samples. However, the existing active learning methods usually label samples near the classification boundary in each iteration [20], [21], causing some normal samples to be misclassified as anomalies and thus generating a lot of false alarms. As shown in Fig. 3, when using these active learning methods, it is likely that the PU learning method will mistakenly label some normal samples as anomalies (red circles) and more and more normal samples will be wrongly labeled as anomalies iteratively. We can see that this active learning strategy is likely to generate many false alarms.

III. Framework of PUAD

A. The Workflow of PUAD

As mentioned before, we integrate clustering, PU learning, and semi-supervised learning in *PUAD* to address the first challenge. The overall workflow of *PUAD* is shown in Fig. 4.

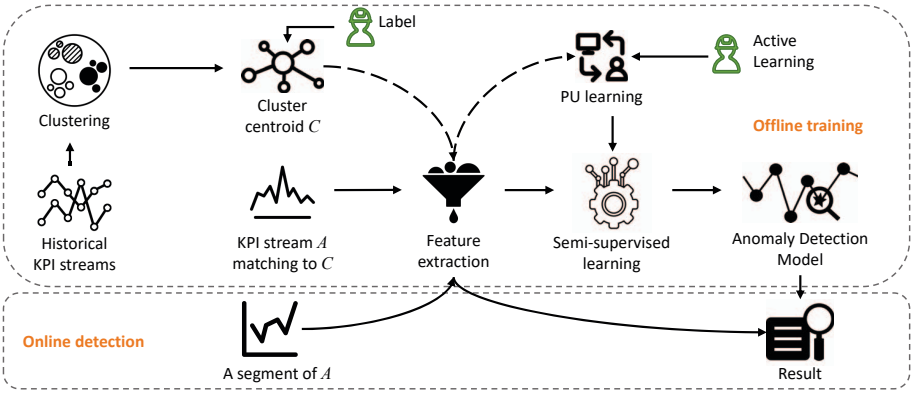


Fig. 4: The framework of *PUAD*

In the offline training process, *PUAD* clusters historical KPI streams, labels a few potential anomalies, and extracts the features of cluster centroid KPI streams. To obtain more and enough credible labeled samples, *PUAD* exploits PU learning based on the prior extracted features and labels. For a newly emerging KPI stream *A*, *PUAD* firstly assigns it into an existing cluster by comparing its shape similarities with all cluster centroids and then extracts its features. Finally, *PUAD* trains a model for it through semi-supervised learning with its features as well as the features and the labels of its corresponding cluster centroid.

In the online detection process, for a newly arrived data point (i.e., x_i) of the KPI stream *A*, its features would be firstly extracted according to data points in segment $x_{i-w+1:i}$. Then the features would be fed into the trained model to get an anomaly score for $x_{i-w+1:i}$. *PUAD* will determine whether it is an anomaly according to the anomaly threshold. In this paper, we apply the threshold with which the labeled samples in the training set reach the best F1-score.

In summary, the combination of clustering, PU learning, and semi-supervised learning not only greatly reduces the labeling effort, but also improves the accuracy of anomaly detection.

1) Clustering: After observation on a large number of KPI streams, we find that despite the diversity of KPIs, many of them are similar because of their implicit associations and similarities. If similar KPI streams can be grouped into a few clusters, we can train an anomaly detection model for each cluster using a few manual labels, and "transfer" the trained model within each cluster. Let ϵ be the number of clusters. Because ϵ is significantly smaller than that of KPI streams, we can greatly reduce the labeling overhead. Therefore, we cluster KPI streams and obtain a few labels for each cluster centroid KPI stream, which is the most representative one in the cluster.

PUAD adopts ROCKA [23] for clustering, which has been verified to be a robust and rapid time series clustering algorithm. Besides, it uses SBD (shape-based distance) [24] to measure the time series similarity, and exploits DBSCAN [25] to cluster KPI streams, and chooses the centroid for every cluster. Operators will randomly label a small number of anomalous segments for each cluster centroid. For a new KPI stream, it will be assigned into the most similar cluster by calculating its SBD with these centroid KPI streams.

Please note that the choice of clustering method is flexible, and applying ROCKA for clustering is not the main contribution of this work.

2) Feature extraction: Motivated by [11], [18], *PUAD* extracts and categorizes the features into two groups: temporal features and forecasting error features.

Temporal Features: In general, dramatic changes in time series are likely to be anomalous. To indicate the changes of KPI streams in a short period, *PUAD* extracts 19 temporal features, as shown in Table I. These features are calculated according to a sliding window of data points, i.e., $x_{i-w+1:i}$.

Forecasting Error Features: Following [11], we utilize a set of error indicators generated by time series forecasting methods as features. For a data point, if its actual value differs too much from its prediction, it may be an anomaly generally. *PUAD* adopts three classical time series prediction techniques, i.e., Holt [26], STL [27], and Holt-Winters [26], to extract three forecasting error features. Through these techniques, *PUAD* can predict the short-term and long-term trends of a time series, respectively.

In total, *PUAD* eventually extracts 22 features through the above methods.

3) PU learning: After extracting the above features, *PUAD* adopts the feature samples of each cluster centroid as a training set, in which only a few anomalies are labeled. Let θ be the number of initially labeled samples. Then, the training set consisting of positive samples (i.e., labeled anomalies) and unlabeled samples will be input together into the PU learning component. Let $\Omega(P)$, $\Omega(U)$, and $\Omega(N)$ represent the set of positive samples, unlabeled samples, and negative samples in the training set, respectively. Hereinafter, we use "positive" to refer to anomalous, and "negative" to refer to normal. Each sample is a feature vector. Section III-B introduces this component in detail.

4) Semi-supervised learning: After obtaining enough labels by PU learning, each cluster centroid currently contains positive samples $\Omega(P')$, negative samples $\Omega(N')$, and unlabeled

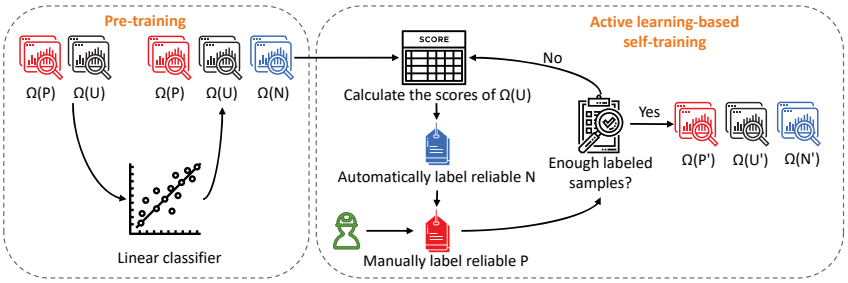


Fig. 5: The framework of PU learning and active learning

TABLE I: Temporal features extracted by *PUAD*.

Feature	Description
Slope_ratio	Slope between two consecutive points.
Sum_ratio	Slope between two consecutive windows.
Cv_delta	Difference of coefficient of variation between two consecutive windows.
Cv_slope	Slope of coefficient of variation between two consecutive windows.
Ping_delta	Difference between two consecutive points.
Sum_delta	Difference between two consecutive windows.
Long_time_delta	Difference between the current point and the previous long time window.
Long_time_slope	Slope between the current point and the previous long time window.
Block_delta	Difference between two adjacent windows.
Block_slope	Slope between two adjacent windows.
Block_dping_delta	Difference between the current point and the previous window.
Shift_block_dping_delta	Shift between the current point and previous window.
Std	Standard deviation in a window.
Std_delta	Difference of standard deviation between two consecutive windows.
Max_level_shift	Max trimmed mean between two consecutive windows.
Max_var_shift	Max variance shift between two consecutive windows.
Max_KL_shift	Max shift in Kullback-Leibler divergence between two consecutive windows.
Lumpiness	Changing variance in the remainder.
Flatspots	Discretize time series values into ten equal-sized intervals. Find maximum run length within the same bucket.

samples $\Omega(U')$. A newly emerging KPI stream would be assigned into an existing cluster firstly. Next, we extract its features and treat them as a new unlabeled dataset $\Omega(U_{new})$. Then, we aim to train a model based on the dataset consists of $\Omega(P')$, $\Omega(N')$, $\Omega(U')$, and $\Omega(U_{new})$. In this work, we adopt a semi-supervised learning method called CPLE (Contrastive Pessimistic Likelihood Estimation) [28]. Verified by [18], CPLE is more robust than other semi-supervised learning algorithms because it needs no strong assumptions. Besides, we apply random forest as the base model of CPLE following [18]. After the semi-supervised learning process, *PUAD* gets an anomaly detection model for each KPI stream.

Note that the choice of semi-supervised methods is flexible, and applying CPLE as the semi-supervised method is not the main contribution of our work.

B. PU Learning

[19] improved PU learning method and applied it for time series anomaly detection. Following [19], we utilize PU learning to label anomalies as few as possible in *PUAD*.

Fig. 5 shows an overview of PU learning with two steps. The first step is to initialize the $\Omega(N)$ through pre-training. In the second step, positive samples $\Omega(P)$ and negative samples $\Omega(N)$ are iteratively extended through self-training and active learning.

Step 1. pre-training. In the beginning, the training set consists of several positive samples (i.e., $\Omega(P)$) and a great many unlabeled samples (i.e., $\Omega(U)$). To find reliable negative samples more cautiously from $\Omega(U)$, employing a linear model as a classifier will be helpful [19]. Note that the choice of the linear model is flexible, and we adopt Elastic Net [29] in this paper, which has been proved to identify possible directional edges even in the presence of highly correlated data. After training the linear model, we can obtain, for each sample in $\Omega(U)$, a prediction score indicating the probability of being positive. *PUAD* selects the top s samples with the lowest scores from $\Omega(U)$ to initialize $\Omega(N)$ (we set $s = 0.20$ in our scenario).

Step 2. active learning-based self-training. After pre-training, the training set now consists of a few positive samples (i.e., $\Omega(P)$) and negative samples (i.e., $\Omega(N)$), and a large number of unlabeled samples (i.e., $\Omega(U)$). To make better use of the unlabeled samples, we adopt a self-training method to label more unlabeled samples in $\Omega(U)$ iteratively.

As shown in Fig. 3, we need to ensure that positive samples determined by the self-training method are indeed anomalies. Consequently, we adopt active learning in this self-training step, called active learning-based self-training. Algorithm 1 describes the details of this process.

Firstly, *PUAD* trains a random forest model classifier on $\Omega(P)$ and $\Omega(N)$ to obtain the prediction scores of $\Omega(U)$ in each iteration, which is shown from lines 1 to 2.

Then, *PUAD* labels some unlabeled samples iteratively. In each iteration, let λ be the speed denoting the number of newly added unlabeled samples, π be the class prior representing the proportion of positive samples labeled by operators. For all iterations, p be the proportion of labeled samples by the PU learning method consisting of pre-training and self-training. *PUAD* sorts $\Omega(U)$ according to the above scores, which is

Algorithm 1 Active-learning-based self-training process

Input: training data: $\Omega(P)$, $\Omega(U)$, $\Omega(N)$; π: positive class prior; λ: speed; p: the proportion of labeled samples by PU learning; s: the proportion of labeled samples by pre-training in PU learning;
Output: updated $\Omega(P')$, $\Omega(U')$, $\Omega(N')$;

1: Model $M = Random\ Forest\ Model(\Omega(P),\ \Omega(N))$;
2: $score = predict(M,\ \Omega(U))$;
3: $\Omega(N') = \Omega(N)$;
4: $\Omega(P') = \Omega(P)$;
5: **for** $|\Omega(N')| - |\Omega(N)| < (p-s)\times(1-\pi)\times|\Omega(U)|/\lambda$ **do**
6: $I = rank(score)$ in ascending order;
7: $P_{candidate} = \{x_i \mid (score(i) > I(end - \lambda\times \pi))\}$;
8: $P_{real},\ N_{real} = manually\ check(P_{candidate})$;
9: $N_{add} = \{x_i \mid (score(i) < I(\lambda))\}$;
10: $\Omega(N') = \Omega(N') \cup N_{add} \cup N_{real}$;
11: $\Omega(P') = \Omega(P') \cup P_{real}$
12: $\Omega(U') = \Omega(U') - N_{add} - N_{real} - P_{real}$;
13: Model $M = Random\ Forest\ Model(\Omega(P'), \Omega(N'))$;
14: $score = predict(M,\ \Omega(U'))$;
15: **end for**
16: **return** $\Omega(P'), \Omega(U'), \Omega(N')$

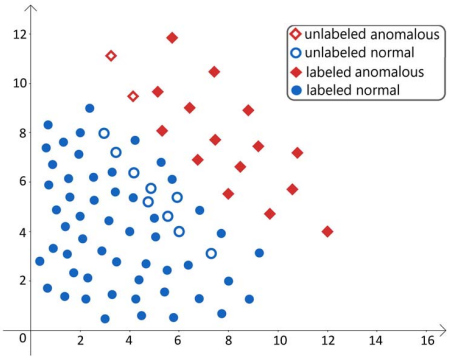


Fig. 6: Expected results of active learning-based self-training

shown in line 6. For the unlabeled samples with the highest scores, *PUAD* sets them as candidate positive samples. Next, operators manually label them to guarantee that the samples labeled as positive are truly positive. For the unlabeled samples with the lowest scores, *PUAD* labels them as negative samples directly. After that, as shown from lines 10 to 13, *PUAD* updates $\Omega(P)$, $\Omega(N)$, and $\Omega(U)$, and then start the next iteration until enough unlabeled samples are labeled.

Finally, as shown in Fig. 6, we can guarantee that there are few misclassified negative samples in the set of updated positive samples. These reliable positive samples ensure the accuracy of the anomaly detection model. Evaluated in Section IV-C1, the proposed active learning method significantly improves the accuracy of *PUAD*.

IV. Evaluation

A. Experimental Design

1) Dataset: We obtained a total of 208 KPI streams from a large-scale software service provider. Moreover, we selected 128 of them for clustering and the other 80 KPI streams as newly emerging KPI streams.

In *PUAD*, we apply ROCKA [23] to cluster the 128 historical KPI streams firstly. As evaluated in Section IV-D, ϵ is chosen with a higher F1-score, i.e., 9. Then, manually labeling several potential anomalies and extracting the features of the 9 cluster centroids. Finally, we assign the 80 new KPI streams into the existing clusters and extract their features meanwhile. In summary, we have 128 time series for clustering, 9 cluster centroids with random labels, and 80 streams for evaluation (the former 40% with its entire cluster centroid for training, the latter 60% for testing).

2) Evaluation Metrics: *PUAD* uses a simple strategy following [14]. If any data point in an anomalous segment can be detected by a chosen threshold, it is said that this segment can be detected correctly. The points outside the anomalous segment are treated as normal. The F1-score, precision, and recall can be computed accordingly [11]. We apply this strategy because operators usually do not care about point-wise metrics, and it is acceptable for a method to issue an alarm for any time point in a contiguous anomalous segment when the delay is not too long. The best F1-score indicates the best performance of the model on a specific test set. Therefore, we evaluate the effectiveness of anomaly detection methods using the best F1-score.

In addition to the F1-score, we also apply Matthew's correlation coefficient (MCC), which takes true negatives into account, to evaluate every methods' effectiveness [30]. The value of MCC ranges from -1 to 1, and a larger value represents higher effectiveness.

3) Baseline Methods: To examine the performance of *PUAD*, we selected several commonly-used anomaly detection algorithms as baseline methods, summarized in Table III.
(1) For supervised learning based methods, Opprentice utilizes the labels to train a random forest classifier. EGADS uses a collection of anomaly detection and forecasting models with an anomaly filtering layer for detection.
(2) For unsupervised learning based methods, Donut [14] and iForest [31] are chosen as baseline methods. Donut is a VAE-based method for seasonal KPI streams with local variations. iForest detects anomalies based on isolation forest.
(3) For semi-supervised learning based methods, ADS [18]. enables the rapid deployment of anomaly detection models for a large number of emerging KPI streams, through clustering and semi-supervised learning.
(4) Compared with the methods that also reduce the amount of labeling, we select ATAD [11], which combines transfer learning with active learning and achieves a balance between labeling effort and performance.

4) Experimental Settings: In the experiment, we set *PUAD*'s hyper-parameters experimentally (more details can

TABLE II: The detailed information of the dataset collected from a large-scale software service provider

Process	number of KPI streams	Interval (minute)	Total points	Anomaly points	Anomaly ratio (%)
Clustering	128	5	1024664	8318	0.81%
Anomaly Detection	80	5	643593	6839	1.06%

TABLE III: The detailed information of the baseline methods

Name	Context	Dataset	Algorithm	Performance	Labeling effort
Opprentice	Internet-based service	PV, #SR, SRT	Random Forest	recall≥0.66, precision≥0.66	Full labels
EGADS	A Universal Anomaly Detection System of Yahoo	Yahoo Membership Login (YML) data, Synthetic data	A collection of anomaly detection and forecasting models with an anomaly filtering layer	between 0 and 0.8	Full labels
Donut	Internet-based service	KPI streams from Internet companies	VAE	range from 0.75 to 0.90	None labels
iForest	General anomaly detection scenario	11 natural datasets, a synthetic dataset	A binary tree structure called isolation tree (iTree)	0.67≤AUC≤1.0	None labels
ADS	Internet-based service	KPI streams from Internet companies	Rocka and CPLE	the average best F1-score is 0.92	Cluster centroids full labels
ATAD	Cloud service systems	NAB, Yahoo	CORrelation ALignment(CORAL) and active learning	between 0.5 and 1	1%-5% labels

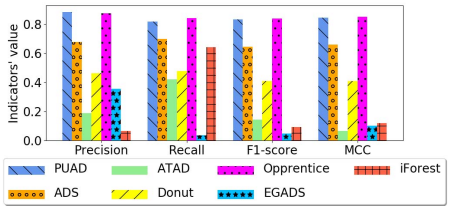


Fig. 7: The effectiveness of different methods

be seen in Section IV-D). In the clustering process, we set the cluster radius as 0.05 following ROCKA [15] and set the number of clusters, i.e., $\epsilon = 9$. For feature extraction, we set the length of each sliding window, i.e., $w = 6$. In the PU learning process, we set the number of initially labeled samples, i.e., $\theta = 8$, the positive class prior, i.e., $\pi = 0.015$, with an estimation of 0.01, the speed, i.e., $\lambda = 200$, the proportion of samples for initializing $\Omega(N)$, i.e., $s = 0.200$, and the overall proportion of labeled samples by PU learning, i.e., $p = 0.90$.

5) Implementation: *PUAD* is implemented using Python. We conduct the evaluation experiments on a PowerEdge R730 server with 48 Intel Xeon E5-2650 CPUs and 128GB memory.

B. Evaluation of The Overall Performance

1) Effectiveness.: Fig. 7 shows the average precision, recall, and best F1-score of all 80 new KPI streams using the above seven methods. We can see that both *PUAD* and Opprentice perform superior in KPI anomaly detection, with the average best F1-score of 0.833 and 0.840, respectively. In contrast, ADS performs a little worse than them whose average best F1-score is 0.647. The performance of the above three models is much better than that of ATAD, Donut, iForest, and EGADS, for the following reasons: (1) The clustering method of ATAD is based on time data points rather than time series, and it only pays attention to KPIs' diversity but ignores the similarity of KPI streams. Besides, ATAD requires a high anomaly labeling ratio (ranges from 1% to 5%), but this ratio is not that high ($< \%1$) in the dataset. (2) Donut is a deep Bayesian model requiring a long period of training data (say six months' worth of data). However, the period of the KPI streams in the dataset is not that long, which affects the effectiveness of Donut. (3) iForest cannot effectively extract features from KPI streams, and it is sensitive to noises, which is not rare in the dataset. (4) EGADS heavily relies on high-quality labels and could be biased towards certain data sets [32]. Furthermore, it proves that *PUAD* is significantly better than ADS. The reason is that we adopt active learning to check samples and ensure label correctness. It can be seen that the average best F1-score of *PUAD* is close to that of Opprentice, a state-of-the-art supervised model.

Fig. 7 shows the average MCC of *PUAD* and the six baseline methods on the 80 new KPI streams. We can find that the MCCs of *PUAD* and Opprentice are significantly higher than those of other methods, which is consistent with the evaluation results measured through the F1-score.

2) Computational efficiency.: The complexity is analyzed by comparing the time required to detect each data point online. When a data point is generated online, *PUAD*, ADS, ATAD, and Opprentice need to extract its features firstly. The complexity of these algorithms is mainly reflected in the feature extraction process. According to statistics, it takes 0.501s for *PUAD*, 0.488s for ADS, 3.243s for ATAD, and 0.488s for Opprentice. The prediction time of other baselines is 1.219s for Donut, 0.009s for iForest, and 0.011s for EGADS. Since the data point arrives at an interval of 5 minutes online, each baseline can complete the detection within this time. Furthermore, *PUAD* can achieve a satisfactory detection result by only labeling a small number of samples, which is competitive considering both effectiveness and labeling cost.

TABLE IV: The detailed information of the NAB dataset

Dataset	Interval (minute)	Total points	Anomaly points	Anomaly Ratio (%)
AWS	5	67740	3097	4.57%
Artificial	5	16128	624	3.87%
Twitter	5	142765	217	0.15%

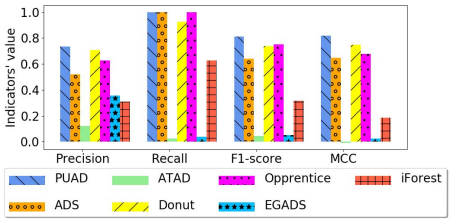


Fig. 8: The effectiveness of different methods on the NAB dataset

3) Generality.: To further evaluate the generality of *PUAD*, we apply an open-source dataset, Numenta anomaly benchmark (NAB) [33], which consists of the time series data collected from different scenarios including AWS, Twitter, and Artificial, etc. All the data points have been manually labeled. We select AWS, Artificial, and Twitter datasets in our experiments as they are large in scale and contain more anomalies than other datasets [11]. The detailed information of the dataset is listed in Table IV. Fig. 8 shows the effectiveness of all methods on the NAB dataset, and *PUAD* outperforms the six baseline methods in terms of F1-score and MCC. We can observe that when the best F1-scores are achieved, the recall values of *PUAD*, ADS, Opprentice are high. It is because *PUAD*, ADS, Opprentice all can easily detect most of the anomalous segments for the following two reasons. (1) The labeled anomalies in the NAB dataset all have significant patterns. (2) As mentioned in Section IV-A2, we apply a practical strategy for calculating precision, recall, and F1-score following [11], [14], [15], where if a data point of an anomaly segment labeled by operators is detected by a method, we determine this segment is accurately detected. However, both ADS and Opprentice generate much more false alarms and thus suffer from low precision. Overall, the evaluation experiments demonstrate that *PUAD* is general to other scenarios.

C. Ablation Study

In this section, we evaluate the effectiveness of active learning, clustering, and semi-supervised learning, respectively, to examine the contributions of the three methods.

1) Active learning: During active learning, we label the samples that are likely to be anomalies instead of those are near the classification boundary. We believe that this strategy is more effective. To evaluate the effectiveness of this strategy, we conduct the following experiments:

- Remove active learning from *PUAD*.

TABLE V: The best F1-scores on 9 clusters

Clusters	W/o active learning	With random labels	Label boundary	*PUAD*
1	0.612	0.697	0.812	**0.920**
2	0.545	0.576	0.667	**0.967**
3	0.819	0.850	**0.860**	0.851
4	0.745	0.720	0.860	**0.872**
5	0.596	0.714	0.785	**0.839**
6	0.458	0.664	0.638	**0.737**
7	0.871	0.900	0.872	**0.921**
8	0.714	0.762	0.793	**0.812**
9	0.587	0.589	0.673	**0.675**
Average	0.636	0.719	0.772	**0.833**
Increase ratio	31.0%	15.9%	7.9%	–

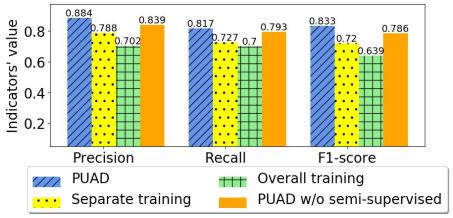


Fig. 9: The effectiveness of *PUAD*, *PUAD* without clustering (*overall training* and *separate training*), and *PUAD* without semi-supervised learning

- Remove active learning from *PUAD*, and add the same number of *random* labels as that used for active learning.
- Label the same number of positive samples (i.e., $\lambda \times \pi$) that are near the classification boundary in each iteration during active learning.

TABLE V lists the best F1-scores on the nine clusters using these three baseline methods and *PUAD*. We can observe that active learning indeed improves the effectiveness of *PUAD* (the second and third columns), and labeling the samples that are likely to be anomalies is more effective than labeling the samples near the classification boundary (the fourth column).

2) Clustering: Utilizing clustering to preprocess the historical KPI streams as the first step because: (1) If training only one model for all KPI streams, the detection result will be inaccurate because of the diversity of KPI streams. (2) If training an anomaly detection model separately for each KPI stream, the overhead of anomaly detection is huge when facing large-scale data.

To verify the effectiveness of clustering, we design the following two methods:
(1) Train one model for all new KPI streams. We randomly select nine (consistent with the number of clusters in *PUAD*) of all historical KPI streams as the training set, called *overall training*. For a new KPI stream, we randomly select one of the nine streams together with its former 40% data points to constitute the training set and apply its latter 60% data points as the testing set.
(2) Train a model for each new KPI stream separately. We utilize the former 40% data points of each new KPI stream to train a model separately and detect anomalies for its latter

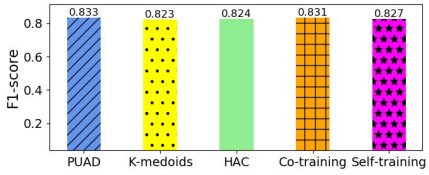


Fig. 10: The effectiveness of *PUAD* when replacing its clustering methods with K-medoids and HAC, respectively, and its semi-supervised learning methods with co-training and self-training, respectively

60% data points called *separate training*.

Fig. 9 shows the best average F1-scores of the above two methods. We find that clustering indeed improves the effectiveness of *PUAD* in terms of precision, recall, and F1-score. This is because: (1) KPI streams are diverse in the pattern. If we train only one model for all KPI streams, the pattern of the KPI streams in the training set may be completely different from that of the new KPI streams. Therefore, the trained model cannot be used for the new KPI streams. (2) If we train a separate model for each new KPI stream, it is computationally inefficient.

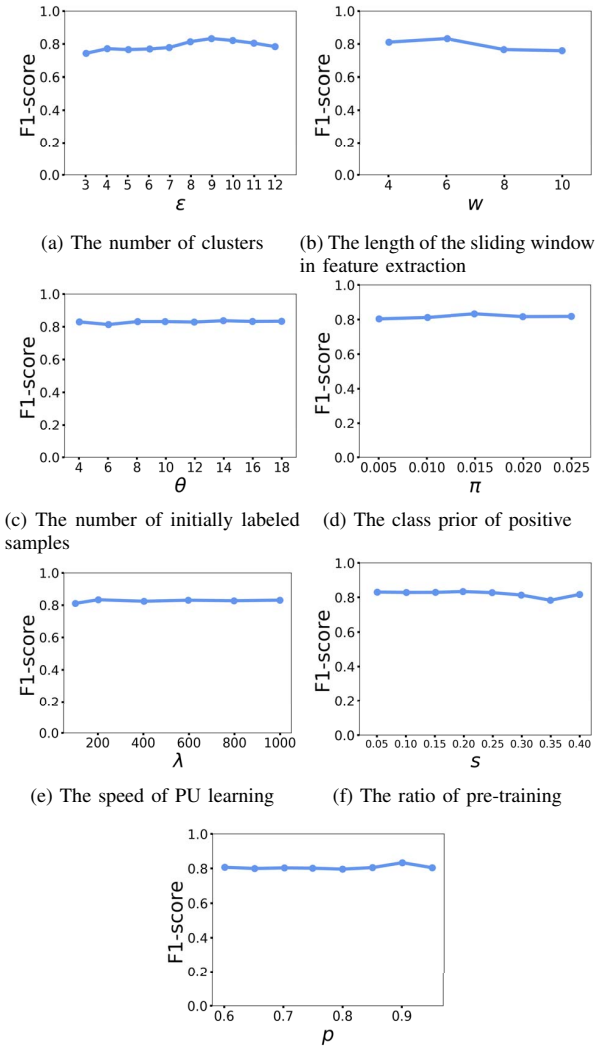
As mentioned in Section III-A1, the choice of a clustering method for *PUAD* is flexible. To verify this point, we replace the clustering method, ROCKA [23], with K-medoids [34], and Hierarchical Agglomerative Clustering (HAC) [35], respectively, and evaluate the effectiveness, respectively. The evaluation results are shown in Fig. 10, and we can find that both K-medoids and HAC achieve approximate F1-scores to ROCKA, respectively, showing that *PUAD* is robust to various clustering methods.

3) Semi-supervised Learning: Each cluster centroid KPI stream consists of labeled and unlabeled samples after PU learning. When a new KPI stream is assigned to a particular cluster, the semi-supervised learning component is applied to train the anomaly detection model. To evaluate the effectiveness of the semi-supervised learning component, we remove it and only utilize the existing labeled samples to train the anomaly detection model. The average best F1-score is shown in Fig. 9. We can find the removal degrades the performance of *PUAD* because it leads to the unlabeled samples cannot be fully utilized.

As mentioned in Section III-A4, *PUAD* is also robust to different semi-supervised learning methods. To demonstrate this point, we replace CPLE, which is the semi-supervised learning method used in *PUAD*, with co-training [36] and self-training [37], respectively, and evaluate their effectiveness. As shown in Fig. 10, when co-training or self-training is applied for semi-supervised learning, *PUAD* achieves close F1-scores to when CPLE is applied. It proves that the choice of semi-supervised method for *PUAD* is indeed flexible.

D. Evaluation of Hyper-Parameters

To evaluate the effectiveness of *PUAD*'s hyper-parameters, we calculate the average best F1-score of *PUAD* as the values



(a) The number of clusters

(b) The length of the sliding window in feature extraction

(c) The number of initially labeled samples

(d) The class prior of positive

(e) The speed of PU learning

(f) The ratio of pre-training

(g) The proportion of labeled samples by PU learning

Fig. 11: The F1-scores of *PUAD* as its parameters vary

of these hyper-parameters vary, as shown in Fig. 11. More specifically, we increase ϵ from 3 to 12, w from 4 to 10, θ from 4 to 18, π from 0.005 to 0.025, λ from 200 to 1000, s from 0.05 to 0.40, and p from 0.60 to 0.95. We can find that *PUAD* achieves approximate average best F1-scores when these hyper-parameters vary, and it is robust to all of these parameters. Specifically, we can observe that the value of π does not significantly impact *PUAD*'s performance. Therefore, when fewer positive samples are labeled by operators during active learning, *PUAD*'s performance will not significantly change. Therefore, considering both effectiveness and efficiency, we set their parameters when *PUAD* achieves relatively higher average best F1-score, i.e., $\epsilon = 9, w = 6, \theta = 8, \pi = 0.015, \lambda = 200, s = 0.20, p = 0.90$.

E. Threats to Validity

Data quality. In this work, we use two datasets, one is public and another is collected from a real Internet company. The ground truth is based on performance issues and incident reports. Although operators have a wealth of experience, manual labeling of anomalous points may still contain a small degree of noise. We believe that the noise in these labels is only a small percentage. Besides, we adopt widely used evaluation indicators (*e.g.* [14], [18]) to detect continuous abnormal segments instead of point-wise anomalies, eliminating part of the label noise.

Granularity influence. In our experiment, the granularity of the time series is 5 minutes, but the effectiveness of the algorithm is not affected by the granularity. For fine-grained data, we believe that the algorithm can still work without extra effort. Of course, the amount of data in our experiments is still limited. We will experiment *PUAD* with a larger scale of datasets on more software service systems in the future.

V. Related Work

A. Clustering KPI streams

According to [38], widely used clustering algorithms can be roughly divided into several groups, including Partitioning, Hierarchical, Grid-based, Model-based, and Density-based algorithms.

Hierarchical-based algorithms require that the time series have the same length, and they are difficult to get a satisfactory result [39] when the data length is too long. Partitioning algorithms such as K-Means [40] and K-Medoids [41] need some hyper-parameters like the number of clusters, which are hard to be predetermined definitely. Grid-based methods are rarely used in time series clustering because they either run very slowly or they are inaccurate on large datasets [38]. Model-based methods assume that they include statistical methods (e.g., COBWEB [42]) or neural network methods (e.g., ART [43]) for each cluster and find the data that best fits the models. However, these methods often follow strong assumptions (e.g., a Gaussian mixture can model the time series [44]), which are difficult to apply on complex datasets. Density-based clustering algorithms (e.g., DBSCAN [25]) are carried out according to the density distribution of samples. These algorithms do not need to specify the number of clusters and are more suitable for our scenario.

B. PU Learning Method

Traditionally, PU learning is suitable for lots of applications such as text detection [45]–[47] and bioscience [48], [49]. [50] proposes UPTAN (Uncertain Positive Tree Augmented Naive Bayes), a Bayesian network algorithm, to utilize the dependence information among uncertain attributes to create a classifier. [51] views pattern detection through PU learning, analyzes time series to learn relevant indirect contexts, and then creates a probabilistic model.

C. Anomaly Detection Algorithms

Over the years, various algorithms have been applied to KPI anomaly detection, including supervised learning, unsupervised learning, and semi-supervised learning methods.

Supervised learning methods: EGADS [12] uses AdaBoost [52] to select anomalies, whose architecture mainly includes three components: forecasting, anomaly detection, and alerting. Opprentice [10] ensembles 14 widely-used statistical algorithms to extract features for data points and then trains a classifier using a Random Forest. Such methods need a fully-label dataset. However, manual labeling for a large number of emerging KPI streams is not feasible.

Unsupervised learning methods: The iForest [31] model assumes that anomalies are few and different, and it constructs a tree structure to separate data points. For the points, the closer to the root of the tree, the more anomalous. Donut [14], Buzz [16] and Bagel [15] are VAE based models, which focus on learning the patterns of normal data rather than anomalies and computing reconstruction probabilities for detection. Although these algorithms do not require data labels, they cannot achieve satisfactory effectiveness.

Semi-supervised learning methods: [53] trained a stacked LSTM network on non-anomalous data and used it as a predictor over a number of time steps. ADS [18] used clustering and CPLE (Contrastive Pessimistic Likelihood Estimation) [28] to detect anomalies in time series. Semi-supervised learning methods use unlabeled data to modify either parameters or models obtained from labeled data alone to maximize the learning performance. They do reduce the cost of manually labeling, however, it is still difficult for operators to label and examine all anomalies in those specified time series segments.

VI. Conclusion

This paper proposes *PUAD*, a PU learning-based method to accurately detect anomalies with a small number of partial labels for large-scale KPI streams. Clustering, PU learning, active learning, and semi-supervised learning are combined to achieve accurate anomaly detection and small labeling effort at the same time. *PUAD* applies a novel transfer learning strategy to avoid false alarms. Extensive evaluation experiments using real-world KPI streams from a top-tier software service provider demonstrate that *PUAD* achieves close accuracy to supervised methods, and significantly outperforms existing semi-supervised learning-based, transfer learning-based, and unsupervised learning-based methods. In the future, we will evaluate *PUAD* in more real-world scenarios and gain more insights from case studies.

VII. Acknowledgment

We thank the anonymous reviewers for their valuable feedback. The work was supported by the National Key R&D Program of China (Grant 2019YFB1802504), National Natural Science Foundation of China (Grant No. 61902200 and 62072264), The China Postdoctoral Science Foundation (2019M651015), and Beijing National Research Center for Information Science and Technology (BNRist).

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, 2018.

[2] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2018.

[3] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang, "Automap: Diagnose your microservice-based web applications automatically," in *Proceedings of The Web Conference 2020*, 2020, pp. 246–258.

[4] M. Ma, W. Lin, D. Pan, and P. Wang, "Self-adaptive root cause diagnosis for large-scale microservice architecture," *IEEE Transactions on Services Computing*, 2020.

[5] J. Lin, P. Chen, and Z. Zheng, "Microscope: Pinpoint performance issues with causal graphs in micro-service environments," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 3–20.

[6] Z. He, P. Chen, X. Li, Y. Wang, G. Yu, C. Chen, X. Li, and Z. Zheng, "A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[7] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 3009–3017.

[8] Z. Li, Q. Cheng, K. Hsieh, Y. Dang, P. Huang, P. Singh, X. Yang, Q. Lin, Y. Wu, S. Levy *et al.*, "Gandalf: An intelligent, end-to-end analytics service for safe deployment in large-scale cloud infrastructure," in *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, 2020, pp. 389–402.

[9] A. Gartziandia, "Microservice-based performance problem detection in cyber-physical system software updates," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2021, pp. 147–149.

[10] 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 ACM Conference on Internet Measurement Conference*. ACM, 2015, pp. 211–224.

[11] X. Zhang, Q. Lin, Y. Xu, S. Qin, H. Zhang, B. Qiao, Y. Dang, X. Yang, Q. Cheng, M. Chintalapati, Y. Wu, K. Hsieh, K. Sui, X. Meng, Y. Xu, W. Zhang, F. Shen, and D. Zhang, "Cross-dataset time series anomaly detection for cloud systems," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA: USENIX Association, Jul. 2019, pp. 1063–1076. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/zhang-xu

[12] 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.

[13] 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*, vol. 17, no. 3, pp. 1294–1308, 2019.

[14] 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 on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 187–196.

[15] 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.

[16] 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.

[17] Y.-L. Zhang, L. Li, J. Zhou, X. Li, and Z.-H. Zhou, "Anomaly detection with partially observed anomalies," in *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018, pp. 639–646.

[18] 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)*, 2018, pp. 1–8.

[19] J. Zhang, Z. Wang, J. Yuan, and Y.-P. Tan, "Positive and unlabeled learning for anomaly detection with multi-features," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 854–862. [Online]. Available: https://doi.org/10.1145/3123266.3123304

[20] G. He, Y. Duan, Y. Li, T. Qian, J. He, and X. Jia, "Active learning for multivariate time series classification with positive unlabeled data," in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2015, pp. 178–185.

[21] M. Huijser and J. C. van Gemert, "Active decision boundary annotation with deep generative models," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5286–5295.

[22] 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*, 2019, pp. 1882–1890.

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

[24] J. Paparrizos and L. Gravano, "k-shape: Efficient and accurate clustering of time series," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1855–1870.

[25] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time superpixel segmentation by dbscan clustering algorithm," *IEEE Transactions on Image Processing*, vol. 25, pp. 5933–5942, 2016.

[26] O. Trull, J. C. García-Díaz, and A. Troncoso, "Initialization methods for multiple seasonal holt–winters forecasting models," *Mathematics*, vol. 8, no. 2, p. 268, 2020.

[27] L. Kegel, M. Hahmann, and W. Lehner, "Feature-based comparison and generation of time series," in *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, ser. SSDBM '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3221269.3221293

[28] M. Loog, "Contrastive pessimistic likelihood estimation for semi-supervised classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 462–475, 2016.

[29] P. A. Traganitis, Y. Shen, and G. Giannakis, "Network topology inference via elastic net structural equation models," *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 146–150, 2017.

[30] K. A. Alharthi, A. Jhumka, S. Di, F. Cappello, and E. Chuah, "Sentiment analysis based error detection for large-scale systems," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021, pp. 237–249.

[31] R. Wang, F. Nie, Z. Wang, F. He, and X. Li, "Multiple features and isolation forest-based fast anomaly detector for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 9, pp. 6664–6676, 2020.

[32] T. Buda, B. Caglayan, and H. Assem, *DeepAD: A Generic Framework Based on Deep Learning for Time Series Anomaly Detection*, 06 2018, pp. 577–588.

[33] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 38–44, 2015.

[34] M. Tiwari, M. J. Zhang, J. Mayclin, S. Thrun, C. Piech, and I. Shomorony, "Banditpam: Almost linear time k-medoids clustering via multi-armed bandits," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 10 211–10 222. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/73b817090081cef1bca77232f4532c5d-Paper.pdf

[35] S. Naumov, G. Yaroslavtsev, and D. Avdiukhin, "Objective-based hierarchical clustering of deep embedding vectors," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, pp. 9055–9063, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17094

[36] W. Zhan and M.-L. Zhang, "Inductive semi-supervised multi-label learning with co-training," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and*

Data Mining, ser. KDD'17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1305–1314. [Online]. Available: https://doi.org/10.1145/3097983.3098141

[37] J.-C. Feng, F.-T. Hong, and W.-S. Zheng, "Mist: Multiple instance self-training framework for video anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 14 009–14 018.

[38] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering – a decade review," *Information Systems*, vol. 53, pp. 16 – 38, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437915000733

[39] K. Putri and S. Halim, "Currency movement forecasting using time series analysis and long short-term memory," 2020.

[40] J. Zhu, Z. Jiang, G. D. Evangelidis, C. Zhang, S. Pang, and Z. Li, "Efficient registration of multi-view point sets by k-means clustering," *Information Sciences*, vol. 488, pp. 205–218, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519302221

[41] C. Oktarina, K. A. Notodiputro, and I. Indahwati, "Comparison of k-means clustering method and k-medoids on twitter data," *Indonesian Journal of Statistics and Its Applications*, vol. 4, no. 1, pp. 189–202, 2020.

[42] D. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139–172, 09 1987.

[43] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, no. 1, pp. 54 – 115, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0734189X87800142

[44] M. Marbac and M. Sedki, "Variable selection for model-based clustering using the integrated complete-data likelihood," *Statistics and Computing*, vol. 27, pp. 1049–1063, 2017.

[45] J. Han, W. Zuo, L. Liu, Y. Xu, and T. Peng, "Building text classifiers using positive, unlabeled and 'outdated'examples," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 13, pp. 3691–3706, 2016.

[46] K. Ren, H. Yang, Y. Zhao, W. Chen, M. Xue, H. Miao, S. Huang, and J. Liu, "A robust auc maximization framework with simultaneous outlier detection and feature selection for positive-unlabeled classification," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 10, pp. 3072–3083, 2018.

[47] M. Li, S. Pan, Y. Zhang, and X. Cai, "Classifying networked text data with positive and unlabeled examples," *Pattern Recognition Letters*, vol. 77, pp. 1–7, 2016.

[48] M. Claesen, F. De Smet, J. A. Suykens, and B. De Moor, "A robust ensemble approach to learn from positive and unlabeled data using svm base models," *Neurocomputing*, vol. 160, pp. 73–84, 2015.

[49] J. Hernández-González, I. Inza, and J. A. Lozano, "Learning from proportions of positive and unlabeled examples," *International Journal of Intelligent Systems*, vol. 32, no. 2, pp. 109–133, 2017.

[50] H. Gan, Y. Zhang, and Q. Song, "Bayesian belief network for positive unlabeled learning with uncertainty," *Pattern Recognition Letters*, vol. 90, pp. 28–35, 2017.

[51] V. Vercruyssen, W. Meert, and J. Davis, *"Now you see it, now you don't!" Detecting Suspicious Pattern Absences in Continuous Time Series*, 01 2020, pp. 127–135.

[52] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?" *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4918–4927, 2019.

[53] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *ESANN*, 2015.