Rapid Deployment of Anomaly Detection Models for Large Number of Emerging KPI Streams

Jiahao Bu[†¶], Ying Liu[†¶], Shenglin Zhang[‡*], Weibin Meng[†¶], Qitong Liu[§], Xiaotian Zhu[§], Dan Pei[†¶]
[†]Tsinghua University [‡]Nankai University [§]Tencent
[¶]Beijing National Research Center for Information Science and Technology (BNRist)
Email: bjh16@mails.tsinghua.edu.cn, liuying@cernet.edu.cn, zhangsl@nankai.edu.cn, mwb16@mails.tsinghua.edu.cn
{cloudliu, markxtzhu}@tencent.com, peidan@tsinghua.edu.cn

***Abstract*—Internet-based services monitor and detect anomalies on *KPIs* (*Key Performance Indicators*, say CPU utilization, number of queries per second, response latency) of their applications and systems in order to keep their services reliable. This paper identifies a common, important, yet little-studied problem of KPI anomaly detection: *rapid deployment of anomaly detection models for large number of emerging KPI streams, without manual algorithm selection, parameter tuning, or new anomaly labeling for any newly emerging KPI streams*. We propose the first framework *ADS* (*Anomaly Detection through Self-training*) that tackles the above problem, via clustering and semi-supervised learning. Our extensive experiments using real-world data show that, with the labels of only the 5 cluster centroids of 70 historical KPI streams, *ADS* achieves an averaged best F-score of 0.92 on 81 new KPI streams, almost the same as a state-of-art supervised approach, and greatly outperforming a state-of-art unsupervised approach by 61.40% on average.**

I. Introduction

Internet-based services (*e.g.*, online games, online shopping, social networks, search engine) monitor *KPIs* (*Key Performance Indicators*, say CPU utilization, number of queries per second, response latency) of their applications and systems in order to keep their services reliable. Anomalies on KPI (*e.g.*, a spike or dip in a KPI stream) likely indicate underlying failures on Internet services [1]–[5], such as server failures, network overload, external attacks, and should be accurately and rapidly detected.

Despite the rich body of literature in KPI anomaly detection [2], [6]–[12], there remains one common and important scenario that has not been studied or well-handled by any of these approaches. Specifically, when large number of KPI streams emerge *continuously and frequently*, operators need to deploy accurate anomaly detection models for these new KPI streams as quickly as possible (*e.g.*, within 3 weeks at most), in order to avoid that Internet-based services suffer from false alarms (due to low precision) and/or missed alarms (because of low recall) and in turn impact on user experience and revenue. Large number of new KPI streams emerge due to the following two reasons. First, new products can be frequently launched, such as in gaming platform. For example, in a top gaming company G studied in this paper, on average over ten new games are launched per quarter, which results in more than 6000 new KPI streams per 10 days on average. Second, with the popularity of DevOps and micro-service, software upgrades become more and more frequent [13], many of which result in the pattern changes of existing KPI streams, making the previous anomaly detection algorithms/parameters outdated.

Unfortunately, none of the existing anomaly detection approaches, including *traditional statistical algorithms, supervised learning, and unsupervised learning*, are feasible to deal with the above scenario well. For *traditional statistical algorithms* [6]–[9], to achieve the best accuracy, operators have to manually select an anomaly detection algorithm and tune its parameters for *each KPI stream*, which is infeasible for the large number of emerging KPI streams. *Supervised learning* based methods [2], [10] require manually labeling anomalies for each new KPI stream, which is not feasible for the large number of emerging KPI streams either. *Unsupervised learning* based methods [11], [12] do not require algorithm selection, parameter tuning, or manual labels, but they either suffer from low accuracy [14] or require large amounts of training data for each new KPI stream (*e.g.*, six months worth of data) [12], which do not satisfy the requirement of rapid deployment (*e.g.*, within 3 weeks) of accurate anomaly detection.

In this paper, we propose *ADS*, the first framework that enables the rapid deployment of anomaly detection models (say at most 3 weeks) for large number of emerging KPI streams, without manual algorithm selection, parameter tuning, or new anomaly labeling for any newly emerging KPI streams.

Our idea of *ADS* is based on the following two observations. (1) In practice, many KPI streams (*e.g.*, the number of queries per server in a well load balanced server cluster) are similar due to their implicit associations and similarities, thus potentially we can use the similar anomaly detection algorithms and parameters for these similar KPI streams. (2) Clustering methods such as ROCKA [15] can be used to cluster many KPI streams into clusters according to their similarities. The number of clusters are largely determined by the nature of the service (*e.g.*, shopping, gaming, social network, search) and the type of KPIs (*e.g.*, number of queries, CPU usage, memory usage), but not by the scale of the entire system. Thus for a given service, the number of clusters can be orders of magnitude smaller than the number of KPI streams, and there is a good chance that a newly emerging KPI stream falls

* Shenglin Zhang is the corresponding author.

978-1-5386-6808-5/18/$31.00 ©2018 IEEE

into one of the existing clusters resulted from historical KPI streams.

Utilizing the above two observations, *ADS* proposes to (1) cluster all existing/historical KPI streams into clusters, (2) manually label the anomalies of all cluster centroids, (3) assign each newly emerging KPI stream into one of the existing clusters, and (4) combine the data of the new KPI stream (unlabeled) and it's cluster centroid (labeled) and use *semi-supervised learning* [16] to train a new model for each new KPI stream. During semi-supervised learning, *ADS*'s base model is supervised learning such as [2], thus is able to avoid algorithm selection and parameter tuning. Semi-supervised learning can train a new model for a new KPI stream using an existing labeled KPI stream as long as these two KPI streams have similar data distribution, which is the case since they are in the same cluster. This way, *ADS* enjoys the benefits of supervised learning, yet only needs to label a much smaller number of historical KPI streams (*i.e.*, cluster centroids) without labeling new KPI streams. Unsupervised learning based methods are not selected as the base model of *ADS* due to low accuracy [14] or the requirement of long period of data for each new KPI stream (*e.g.*, six months worth of data) [12], but they are nonetheless compared with *ADS* in Section IV.

The contributions of this paper are summarized as follows:

- To the best of our knowledge, this paper is the first to identify the common and important problem of *rapid deployment of anomaly detection models for large number of emerging KPI streams, without manual algorithm selection, parameter tuning, or new anomaly labeling for any newly generated KPI streams*, and proposes the first framework *ADS* that tackles this problem.
- To the best of our knowledge, this paper is the first to apply semi-supervised learning to the KPI anomaly detection problem. We adopt a robust semi-supervised learning model, contrastive pessimistic likelihood estimation (CPLE), which is suitable for KPI anomaly detection and only requires *similar* (not necessarily *the same*) data distribution between the existing labeled KPI stream and the new KPI stream.
- We conduct extensive experiments using 70 historical KPI streams and 81 new KPI streams from a top global online game service G. With the labels of only the 5 cluster centroids of 70 historical KPI streams, *ADS* achieves an averaged best F-score of 0.92 on 81 new KPI streams, almost the same as the state-of-art supervised approach [2] which requires the labels for all 81 new KPI streams, and greatly outperforms an unsupervised approach Isolation Forest [14] by 360% and the state-of-art unsupervised approach Donut [12] by 61.40% on average.

The rest of this paper is organized as follows. In Section II, we review the background, related works and motivation. The framework of *ADS* is introduced in Section III. We report the experimental results of evaluating *ADS* in Section IV. Finally, we give a conclusion of this paper in Section V.

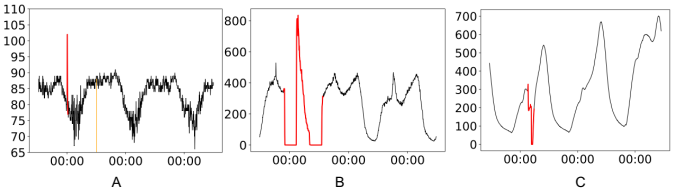

Fig. 1: Examples of anomalies in KPI streams. The red parts in the KPI stream denote anomalous points, and the orange part denotes missing points (filled with zeros).

II. Background

A. Anomaly Detection for KPI Streams

A KPI stream of an Internet-based service is a time series with the format of (timestamp, value). It is essentially monitoring data collected from Simple Network Management Protocol (SNMP), syslogs, web access logs or other data sources [17]–[19]. It can be denoted as $x_{t-m+1}, \ldots, x_t$, where x_i is a monitoring value at time $i, i \in [t-m+1, t]$, t is the present time, and m is the length of the KPI stream.

Anomalous data points of a KPI stream usually have different data characteristics from those of normal data points. For example, a spike, a level shift or a dip in a KPI stream likely indicates an anomaly. Figure 1 shows three examples of anomalies in KPI streams. Anomaly detection for the KPI stream $x_{t-m+1}, \ldots, x_t$ is to determine whether x_t is an anomalous data point (let $y_t = 1$ denote an anomalous data point and $y_t = 0$ denote a normal one).

Most anomaly detection algorithms, including traditional statistical algorithms, supervised learning based methods and unsupervised learning based methods, compute an anomaly score for a data point to denote how likely this data point is anomalous. Operators then set a threshold to determine whether each data point is anomalous or not. That is, only if the anomaly score at time t exceeds this threshold, x_t will be regarded as an anomalous data point.

From the above definition, we can see that anomaly detection for a KPI stream is essentially a two-class classification problem – classifying a data point into an anomalous data point or a normal one. Consequently, we can use the intuitive classification metrics of two-class classification methods, including precision, recall, and F-score, to evaluate the performance of anomaly detection algorithms.

B. Anomaly Detection Methods for KPI Streams

Anomaly detection for KPI streams deals with the task of recognizing unexpected data points from normal behavior. Over the years, diverse *traditional statistical algorithms* have been applied for KPI anomaly detection, including SVD [6], Wavelet [7], ARIMA [8], Time Series Decomposition [1], Holt-Winters [9], *etc.* Each of the above algorithms computes an anomaly score for each data point in a KPI stream on the basis of simple statistical assumptions. To achieve the best accuracy, operators have to manually select an anomaly detection algorithm and tune its parameters for *each KPI stream*. Since it is often the case that a large number of newly emerging KPI streams have to be carefully monitored [13],

manual **algorithm selection and parameter tuning** for every newly emerging KPI stream becomes infeasible.

To address the problem posed by algorithm selection and parameter tuning, several *supervised learning* based methods such as EGADS [10] and Opprentice [2] have been proposed. For each KPI stream, these methods learn (traditional statistical) algorithm selection and parameter tuning from operators' manual labels of KPI anomalies. Obviously, manual **anomaly labeling** for a large number of newly emerging KPI streams is not feasible either.

Unsupervised learning has emerged as a promising field in KPI anomaly detection. For example, isolation based methods [11] and variational autoencoders (VAE) [12] are applied in detecting anomalies in (KPI) streams. These methods are trained without manual labels, and thus they can be applied for large volume of KPI streams. However, isolation based methods suffer from low accuracy [14] (see Section IV-C for more details). In addition, Donut [12], which is based on VAE, requires a long period (say six months) of training data for newly emerging KPI streams. During this period, the Internet-based services may suffer from false alarms (due to low precision) and/or missed alarms (because of low recall) and in turn impact user experience and revenue.

As discussed above, existing anomaly detection algorithms for KPI streams suffer from manual algorithm selection and parameter tuning (traditional statistical algorithms), or manual anomaly labeling (supervised learning based methods), or low accuracy in real-world Internet-based service (unsupervised learning based methods like isolation forests [11]), or long period of training time for newly generated KPI streams (unsupervised learning based methods like Donut [12]).

Semi-supervised learning [16] is halfway between supervised and unsupervised learning. It uses unlabeled data to modify either parameters or models obtained from labeled data alone to maximize the learning performance. [20]–[22] use semi-supervised learning for anomaly detection in other domains, but are not designed for KPI streams (time series).

C. Clustering of KPI Streams

Millions of KPI streams bring a huge challenge to KPI anomaly detection. Luckily, many KPI streams are similar because of their implicit associations and similarities. For example, Figure 2 shows two KPI streams of response latency collected from two different applications, and these two KPI streams are quite similar in shape. If we can identify similar KPI streams, and group the huge volume of KPI streams into a few clusters, we can reduce the overhead in anomaly detection.

Time series clustering is a popular field which has caught lots of attention in the past 20 years. [23] summarized a large number of methods on this topic, most of which are designed for smooth and idealized data. However, the large number of spikes, dips and level shifts in KPI streams can significantly change the shape of KPI streams. Therefore, the above methods do not perform good for KPI streams.

In this work, we adopt ROCKA [15], a rapid clustering algorithm for KPI streams based on their shapes. It applies moving average to extract baselines which successfully reduce the biases of noises and anomalies. In addition, it uses shape-based distance as the distance measure, and reduces its algorithm complexity to $O(m\log(m))$ using Fast Fourier Transform. Finally, it uses DBSCAN to cluster KPI streams and chooses the centroid for every cluster. Extensive experiments in [15] have demonstrated ROCKA's superior performance in clustering KPI streams for large Internet-based services. Please note that applying ROCKA to cluster KPI streams is not our contribution.

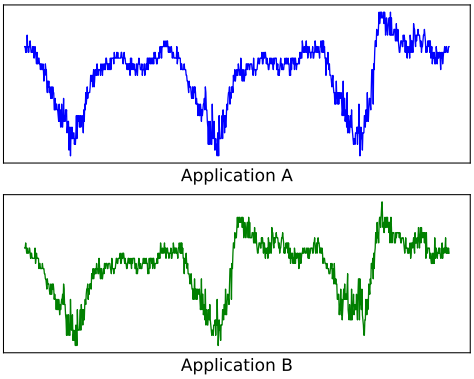

Fig. 2: The response latency streams of two different applications.

III. Framework of ADS

As aforementioned, we propose *ADS*, a semi-supervised learning based anomaly detection framework for KPI streams, to tackle the problem of rapid deployment of anomaly detection models for large number of emerging KPI streams, without manual algorithm selection, parameter tuning, or new anomaly labeling for any newly generated KPI streams.

Figure 3 shows the framework of *ADS*. For historical KPI streams, *ADS* preprocesses them with filling missing points and standardization, clusters the preprocessed KPI streams using ROCKA (Section III-A), and extracts features (namely, output results of different anomaly detection algorithms and parameters) for the KPI streams on cluster centroids (Section III-B). Similarly, when a new KPI stream is generated, *ADS* preprocesses it, and classifies it into one of the above clusters (Section III-A), after which the features of this new KPI stream are extracted (Section III-B). Based on the features of the new KPI stream, and the features and labels of the new KPI stream's cluster centroid, *ADS* trains a semi-supervised learning model using the CPLE algorithm (Section III-C). Finally, *ADS* detects anomalies in the new KPI stream based on the above model and the severity threshold (aThld henceforth).

A. Preprocessing and Clustering

In Internet-based services, monitoring system malfunctions and/or operator misconfigurations, though occur infrequently, can lead to missing points in KPI streams. These missing points can bring significant biases to *ADS*. Therefore, we fill these missing points using linear interpolation following [15].

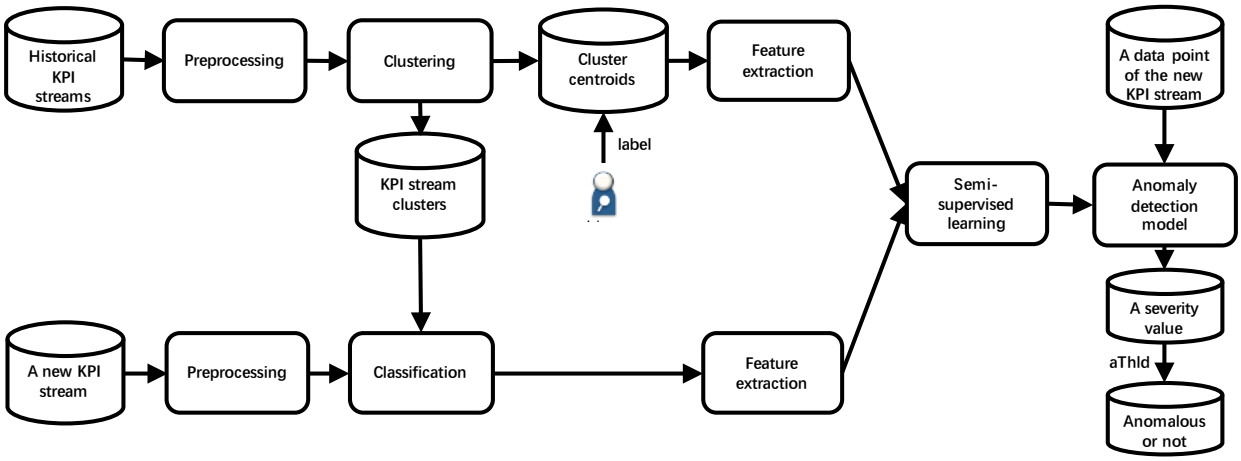

Fig. 3: The framework of *ADS*

In addition, to make KPI steams of different amplitudes and/or length scales comparable, we also standardize KPI streams. This way, different KPI streams are clustered based on their shapes, rather than the absolute values of amplitudes and/or length scales.

As discussed in section II-C, *ADS* adopts ROCKA [15] to group KPI streams into a few clusters, and obtains a centroid KPI stream for each cluster.

B. Feature Extraction

When training based on historical data, since the KPI stream on the centroid of each cluster represents the cluster's characteristics, we extract the features of the KPI stream on each cluster centroid. In addition, when training based on new data, we extract the features of the newly generated KPI stream. The features of the new KPI stream, the features and the anomaly labels of the new KPI stream's cluster centroid, together form the training set.

Here we introduce how to extract features for KPI streams. As is with [2], we use the output results of different anomaly detectors (namely, the anomaly severities measured by anomaly detectors) as the features of KPI streams. This way, each detector serves as a feature extractor.

When an anomaly detector receives an incoming data point of a KPI stream, it internally produces a non-negative value, called *severity*, to measure how anomalous that data point is. For example, historical average [24] applies how many times of standard deviations the point is away from the mean as the severity, based on the assumption that the KPI stream data follows Gaussian distribution; Holt-Winters [9] uses a residual error (namely, the absolute difference between the actual value and the forecast value of each data point) to measure the severity. In addition, most anomaly detectors are parameterized and have a set of *internal parameters* (say, historical average has one parameter of window length, and Holt-Winters has three parameters $\{\alpha, \beta, \gamma\}$). As a result, both detectors and their internal parameters decide the severity of a given data point.

In this work, for each (parameterized) anomaly detector, *ADS* samples its parameters to generate one or more "fixed" anomaly detectors. This way, an anomaly detector with specific sampled parameters acts as a *feature extractor* as follows:

$$\text{A data point} \xrightarrow{\text{anomaly detector + sampled parameters}} \text{feature}$$

The feature extraction, training, and classification (detection) in *ADS* are all designed to work with individual data points, not anomaly windows, so that the semi-supervised learning algorithm can have enough data for training. Another benefit of this design choice is that the classifier can detect anomalies fast on each data point.

TABLE I: Detectors and sampled parameters used in *ADS*. Some abbreviations are MA (moving average), EWMA (exponentially weighted MA), TSD (time series decomposition), SVD (singular value decomposition), win(dow), and freq(uency).

Detectors / #Configurations	Sampled Parameters
Simple threshold [25] / 1	none
Diff / 3	last-point, last-day, last-week
Simple MA [26] / 5	win = 10, 20, 30, 40, 50 points
Weighted MA [27] / 5	
MA of diff / 5	
EWMA [27] / 5	α = 0.1, 0.3, 0.5, 0.7, 0.9
TSD [1] / 1	win = 1 week
TSD MAD / 1	
Historical average [24] / 1	
Historical MAD / 1	
Holt-Winters [9] / $4^3 = 64$	α, β, γ = 0.2, 0.4, 0.6, 0.8
SVD [6] / $5 \times 3 = 15$	#row =10, 20, 30, 40, 50 points, #column =3, 5, 7
Wavelet [7] / $3 \times 3 = 9$	win = 3, 5 ,7 days, freq = low, mid, high
ARIMA [8] / 1	Estimation from data
In total: 14 detectors / 117 configurations	

Following [2], we implement 14 widely used anomaly detectors in *ADS*. All the 14 anomaly detectors and their sample parameters are shown in Table I.

C. Semi-Supervised Learning

After extracting features of anomaly detectors for both the *labeled* KPI streams on cluster centroids and the *unlabeled*

new KPI stream, we try to learn a model which is based on both labeled and unlabeled data, namely a semi-supervised learning model.

As is summarized in [16], different semi-supervised learning models have different advantages and disadvantages. Among these methods, self-training based methods [20] apply an existing model to "label" unlabeled data, and employ the newly labeled data together with the actual labeled data to retrain the model until the prediction result no longer changes or iteration ends.

In this work, we adopt CPLE [28], an extension model of self-training. CPLE is a resilient semi-supervised learning framework for any supervised learning classifier (base-model henceforth) including random forest, SVM, decision tree, *etc.* It takes the prediction probabilities of base-model as input to fully utilize the unlabeled data. CPLE has the three following advantages:

- CPLE is flexible to change base-model, so we can set its base-model to achieve the best accuracy in anomaly detection.
- CPLE needs low memory complexity, as opposed to graph-based methods [29] which needs $O(n^2)$ memory complexity.
- CPLE is more robust than other semi-supervised learning algorithms because it needs no strong assumptions such as (1) the accurate estimation for data distribution of labeled and unlabeled data (as required by expectation maximization algorithms with generative mixture models [30]) and (2) low density (as required by transductive SVM [31]).
- CPLE supports incremental learning. Therefore, when more and more data points are added to a KPI stream, we can continuously train *ADS* to improve its accuracy.

ADS applies CPLE to detect anomalies for KPI streams as follows. For a *base-model* (a supervised learning based binary classifier) $f(x)$ with parameter vector θ, it can be denoted with the form $f(x;\theta) \in \{0,1\}$. Then the probability that a base-model deems a data point as anomalous is $g(x;\theta) = p(f = 1|x,\theta)$, where $g(x;\theta) \in [0,1]$. In addition, the negative log loss for binary classifiers takes on the general form:

$$\begin{aligned} J(\mathbf{y},\mathbf{p}) &= \log \mathbf{p}(\mathbf{y}|\mathbf{p}) \\ &= \frac{1}{N}\sum_{i=1}^{N}[y_i \log p_i + (1-y_i)\log(1-p_i)] \end{aligned} \tag{1}$$

where N is the number of the data points in the KPI streams of training set, y_i is the label of the i-th data point and p_i is the i-th discriminative likelihood (DL) [32]. Usually, a machine learning model is aiming to maximize the negative log loss.

For the unlabeled data points in the training set (namely, the data points of the newly generated KPI stream), we randomly assign a weight q_i to the i-th data point. The objective of CPLE is to minimize the function

$$E(\mathbf{q},\theta|\mathbf{X},\mathbf{U}) = J(\mathbf{y}',g(\mathbf{U};\theta)) - J(\mathbf{y},g(\mathbf{X};\theta)) \tag{2}$$

where $\mathbf{X}$ is the data set of labeled data points, $\mathbf{U}$ is the one of unlabeled data points, and $\mathbf{y}' = H(\mathbf{q})$, where

$$H(q_i) = \begin{cases} 1 & \text{if } q_i \geq 0.5 \\ 0 & \text{if } q_i < 0.5 \end{cases} \tag{3}$$

where $i \in \{1,...,|U|\}$.

This way, (the parameter vector θ of) the base-model, which serves as the anomaly detection model, is trained based on $(\mathbf{X} \cup \mathbf{U})$ using actual and hypothesized labels $(\mathbf{y} \cup \mathbf{y}')$, as well as the weights of data points $\mathbf{w}$, where

$$w_i = \begin{cases} 1 & \text{if } x_i \in \mathbf{X} \\ 0 & q_i \text{ otherwise} \end{cases} \tag{4}$$

where $i \in \{1,...,|U|\}$.

In this work, we apply random forest as the base-model of CPLE because of its simplicity, parallelization, and low memory usage, similar to Opprentice [2]. We do not claim the adoption of random forest model as our contribution.

IV. Evaluation

To evaluate the performance of *ADS*, we have conducted extensive experiments using real-world data collected from a top global online game service. We first introduce the data set (Section IV-A) and metrics (Section IV-B) used for the evaluation experiments. Then, we compare the performance of *ADS* with that of supervised learning based method such as Opprentice, and unsupervised learning based method including iForest and Donut (Section IV-C). Finally, to highlight the importance of semi-supervised learning, we compare *ADS* with the combination of ROCKA and Opprentice (Section IV-D).

A. Data Set

We randomly pick 70 historical KPI streams for clustering and 81 new ones for anomaly detection from a top global online game service. These KPI streams are of the most important three KPIs, including success rate, number of online players and latency. Table II lists the detailed information of the 81 new KPI streams, including the number, interval and length of the KPI streams of each KPI. In addition, it also lists the averaged number and percentage of anomalous data points per KPI stream of each KPI, respectively.

Note that the KPI streams of the same KPI may have very different shapes and thus belong to different clusters. Therefore, experienced operators first manually group the 70 historical KPI streams into five clusters according to their shapes, and then classify the new 81 KPI streams into the five clusters. This serves as the ground truth for clustering.

As mentioned in Section III-A, in this work we apply ROCKA to cluster KPI streams. Based on the manual clustering results by operators, we find that ROCKA accurately groups *all* the 70 historical KPI streams into the *right* cluster. In addition, ROCKA successfully classifies *all* the 81 new KPI streams into the *correct* cluster.

To evaluate the performance of anomaly detection methods, operators also manually label anomalous data points for the

TABLE II: Description of the new 81 KPI streams

KPI	# KPI streams	Interval (minute)	Length (month)	#/percentage of anomalous data points per KPI stream
Latency	19	5	1	150/1.7%
# online players	58	5	1	72/0.83%
Success rate	4	5	1	84/0.97%

81 new KPI streams, as well as the 5 historical KPI streams that are on cluster centroids (calculated using ROCKA). Note that we focus on anomaly detection on newly emerging KPI streams in this work, and thus only the new 81 KPI streams are used for the following evaluation experiments. Therefore, there is no need to manually labeling the remaining 65 historical KPI streams.

As aforementioned, more than 6000 new KPI streams are produced per 10 days. However, manually clustering and labeling anomalies for thousands of KPI streams is infeasible, considering the long period of KPI streams (one month). Therefore, we randomly selected 151 KPI streams in our evaluation. We believe that the 151 KPI streams are sufficient to evaluate *ADS*'s performance.

B. Evaluation Metrics

In real applications, the human operators generally do not care about the point-wise metrics. Therefore, we use a simple strategy following [12]: if any point in an anomaly segment in the ground truth can be detected by a chosen threshold, we say this segment is detected correctly, and all points in this segment are treated as if they can be detected by this threshold. Meanwhile, the points outside the anomaly segments are treated as usual. The precision, recall, F-score and best F-score are then computed accordingly.

As is with [12], we apply the *best F-score* as the metric to evaluate anomaly detection methods. The best F-score indicates the best possible performance of an anomaly detection method on a particular testing set, given an optimal global threshold. In practice, the best F-score is mostly consistent with Area Under the (ROC) Curve (AUC).

Note that in Section IV-C and Section IV-D, we tune aThld based on the labels of the testing set (namely, the back 40% of every new KPI stream) to obtain the *best F-score*. Although it is not entirely practical in real-world applications, it fully compares the *best* performance of *ADS*, iForest, Donut, Opprentice and the combination of ROCKA and Opprentice.

C. Evaluation of The Overall Performance

To evaluate the performance of *ADS* in anomaly detection for KPI streams, we calculate its best F-score, and compare it with that of iForest [11], Donut [12] and Opprentice [2].

For each of the 81 new KPI streams, we train *ADS* using the features extracted from the front 60% of it, as well as the features and manual labels of its cluster centroid (KPI stream). Then we use this *ADS* model to detect anomalies on the back 40% of this new KPI stream. As for iForest, Donut and Opprentice, we divide each new KPI stream (of the 81 KPI streams) into training set and testing set, whose ratios are (front) 60% and (back) 40%, respectively.

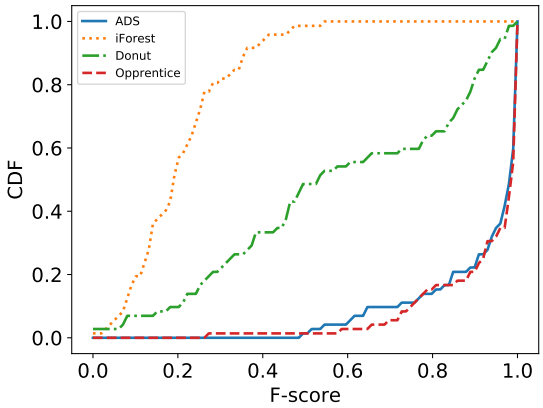

Fig. 4: CDFs of the best F-scores of each new KPI stream using *ADS*, iForest, Donut and Opprentice, respectively.

Figure 4 shows the cumulative distribution functions (CDFs) of the best F-scores of each KPI stream (of the new 81 KPI streams) using the above four methods. Note that in this CDF figure, being closer to the upper left means worse performance, and being closer to the bottom right means better performance. We can see that both *ADS* and Opprentice perform superior in detecting anomalies for KPI streams, with 80% of best F-scores over 0.8. Their performance is much better than that of iForest and Donut, for the following reasons: (1) iForest is sensitive to noises in the training set because it does not utilize any labels [11]. (2) Donut is a deep Bayesian model, which requires a lot of training data to get good results (say six months worth of KPI streams) [12]. However, as Table II shows, we have to train the model using 60% of one month, namely 18 days, worth of newly emerging KPI streams, which is a too small amount of training data for Donut.

To intuitively compare the best F-scores of *ADS*, iForest, Opprentice and Donut, we list the average best F-scores of the above four methods on the five clusters in TABLE III, respectively. *ADS* and Opprentice perform well across all the five clusters and much better than iForst and Donut, demonstrating that it is important to utilize anomaly labels in anomaly detection for newly emerging KPI streams. Specifically, *ADS* improves the average best F-score by 61.40% (as opposed to Donut) to 360% (as opposed to iForest).

Although the supervised learning based method, Opprentice, performs similarly to *ADS*, it needs much more labeling works. For example, in this setting, *ADS* is trained based on only *five* labeled KPI streams on cluster centroids, while Opprentice is trained using all the 81 labeled KPI streams. As aforementioned, over 6000 new KPI streams are produced per 10 days in the studied online gaming service. Manually

TABLE III: Average best F-scores of *ADS*, iForest, Opprentice, Donut and ROCKA+Opprentice in the five clusters, respectively

Cluster	# KPI streams	*ADS*	iForest	Donut	Opprentice	ROCKA + Opprentice
A	7	0.91	0.33	0.42	0.90	0.67
B	9	0.91	0.21	0.37	0.91	0.88
C	8	0.95	0.22	0.28	0.98	0.94
D	53	0.93	0.19	0.67	0.94	0.90
E	4	0.67	0.13	0.45	0.71	0.66
Overall	81	0.92	0.20	0.57	0.93	0.87

TABLE IV: The new KPI streams where *ADS* performs significantly better than ROCKA + Opprentice

KPI stream ID	*ADS*	ROCKA + Opprentice
α	0.86	0.62
β	0.91	0.20
γ	0.72	0.46
δ	0.80	0.55
...	...	...

labeling all the newly emerging KPI streams to detect KPI anomalies is infeasible in practice. Consequently, Opprentice is not appropriate for our scenario.

D. Evaluation of CPLE

To the best of our knowledge, this is the first work to apply semi-supervised learning to the KPI anomaly detection problem. We adopt a robust semi-supervised learning model, CPLE, which is suitable for KPI anomaly detection and only requires *similar* (not necessarily *the same*) data distribution between the existing labeled KPI stream and the new KPI stream.

To evaluate the performance of CPLE, we compare the performance of *ADS*, which is the combination of ROCKA and CPLE, to that of the combination of ROCKA and a state-of-art supervised learning method – Opprentice [2] (ROCKA + Opprentice henceforth). We set up ROCKA + Opprentice as follows. We first apply ROCKA to group the 70 historical KPI streams into five clusters, and classify the 81 new KPI streams into these clusters. For each cluster, we train Opprentice using the features and manual labels of its centroid KPI streams. After that, we detect anomalies for the back 40% of each new KPI stream using the Opprentice model trained based on this new KPI stream's cluster centroid.

TABLE III compares the average best F-scores of *ADS* and ROCKA + Opprentice on each cluster. We can see that *ADS* outperforms ROCKA + Opprentice on every cluster, and greatly outperforms it by 35.82% on cluster A. TABLE IV lists the new KPI streams where *ADS* performs significantly better than ROCKA + Opprentice, and the best F-scores of the above two methods on these KPI streams, respectively.

Here we explain why *ADS* performs better than ROCKA + Opprentice. KPI stream clustering methods such as ROCKA usually extract baselines (namely underlying shapes) from KPI streams and ignore fluctuations. However, the fluctuations of KPI streams can impact anomaly detection. For example, Figure 5 shows the new KPI stream α, and the KPI stream on the its cluster centroid. KPI stream α and the centroid KPI stream have very similar baselines, but they have different

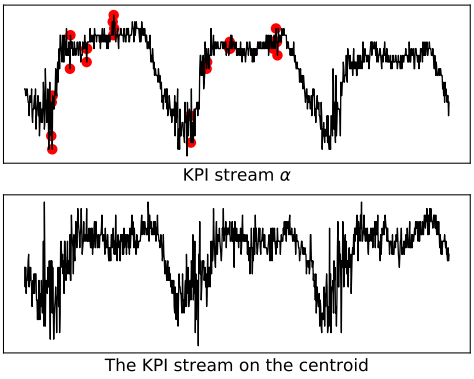

Fig. 5: The anomaly detection results of ROCKA + Opprentice on KPI stream α, and α's cluster centroid KPI stream. The red data points are anomalous determined by ROCKA + Opprentice while in actual they are normal.

fluctuation degrees, which is not uncommon in practice. This lead to that ROCKA + Opprentice, which is trained based only on the centroid KPI stream, generates a lot of false alarms.

ADS addresses the above problem effectively using semi-supervised learning. In other words, it learns not only from the labels of the centroid KPI stream, but also from the fluctuation degree of the new KPI stream. This is consistent with the observation that the model trained based on both labeled and unlabeled data should not be worse than the one trained based only on the labeled data [28].

The experiment results strongly demonstrate *ADS*'s robustness in KPI anomaly detection.

V. Conclusion

To the best of our knowledge, this paper is the first to identify the common and important problem of *rapid deployment of anomaly detection models for large number of emerging KPI streams, without manual algorithm selection, parameter tuning, or new anomaly labeling for any newly generated KPI streams*. We propose the first framework *ADS* that tackles this problem via clustering and semi-supervised learning, which is the first time that semi-supervised learning is applied to KPI anomaly detection. Our extensive experiments using real-world data show that, with the labels of only the 5 cluster centroids of 70 historical KPI streams, *ADS* achieves an averaged best F-score of 0.92 on 81 new KPI streams, almost the same as the state-of-art supervised approach [2], and greatly outperforms an unsupervised approach Isolation Forest [14] by 360% and the state-of-art unsupervised approach Donut [12] by 61.40% on average.

We believe that *ADS* is a significant step towards practical anomaly detection on large-scale KPI streams in Internet-based services. In the future, we plan to adopt more advanced techniques (*e.g.* transfer learning [33]) to further improve *ADS*'s performance.

VI. ACKNOWLEDGEMENTS

We thank Shuai Yang, Tianheng Zuo for their helpful suggestions. The work was supported by National Natural Science Foundation of China (NSFC) under grant No. 61402257, No. 61472214, No. 61472210 and No. 61772307, and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

REFERENCES

[1] 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.

[2] 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.

[3] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, and Z. Zang, "Rapid and robust impact assessment of software changes in large internet-based services," in *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2015, pp. 1–13.

[4] 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.

[5] 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.

[6] A. Mahimkar, Z. Ge, J. Wang, J. Yates, Y. Zhang, J. Emmons, B. Huntley, and M. Stockert, "Rapid detection of maintenance induced changes in service performance," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11. New York, NY, USA: ACM, 2011, pp. 13:1–13:12. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079309

[7] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 71–82.

[8] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 30–30. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251086.1251116

[9] H. Yan, A. Flavel, Z. Ge, A. Gerber, D. Massey, C. Papadopoulos, H. Shah, and J. Yates, "Argus: End-to-end service anomaly detection and localization from an isp's point of view," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2756–2760.

[10] 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.

[11] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013.

[12] 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.

[14] 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.

[13] 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 Service Computing*, 2016.

[15] 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.

[16] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[17] W. Meng, Y. Liu, S. Zhang, D. Pei, H. Dong, L. Song, and X. Luo, "Device-agnostic log anomaly classification with partial labels," *Quality of Service (IWQoS)*, pp. 1–10, 2018.

[18] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, J. Xu, Y. Zhang *et al.*, "Prefix: Switch failure prediction in datacenter networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, p. 2, 2018.

[19] S. Zhang, W. Meng, J. Bu, S. Yang, Y. Liu, D. Pei, J. Xu, Y. Chen, H. Dong, X. Qu *et al.*, "Syslog processing for switch failure diagnosis and prediction in datacenter networks," in *Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on*. IEEE, 2017, pp. 1–10.

[20] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," 2005.

[21] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017.

[22] K. Noto, C. Brodley, and D. Slonim, "Frac: a feature-modeling approach for semi-supervised and unsupervised anomaly detection," *Data mining and knowledge discovery*, vol. 25, no. 1, pp. 109–133, 2012.

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

[24] S.-B. Lee, D. Pei, M. Hajiaghayi, I. Pefkianakis, S. Lu, H. Yan, Z. Ge, J. Yates, and M. Kosseifi, "Threshold compression for 3g scalable monitoring," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1350–1358.

[25] "Amazon cloudwatch alarm," http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/ConsoleAlarms.html.

[26] D. R. Choffnes, F. E. Bustamante, and Z. Ge, "Crowdsourcing service-level network event monitoring," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. ACM, 2010, pp. 387–398. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851228

[27] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003, pp. 234–247.

[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] G. Camps-Valls, T. V. B. Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 10, pp. 3044–3054, 2007.

[30] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[31] T. Joachims, "Transductive inference for text classification using support vector machines," in *ICML*, vol. 99, 1999, pp. 200–209.

[32] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, 2002, pp. 841–848.

[33] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.