Cross-dataset Time Series Anomaly Detection for Cloud Systems

Xu Zhang, ***Microsoft Research, Nanjing University;*** **Qingwei Lin, Yong Xu, and Si Qin,** ***Microsoft Research;*** **Hongyu Zhang,** ***The University of Newcastle;*** **Bo Qiao,** ***Microsoft Research;*** **Yingnong Dang, Xinsheng Yang, Qian Cheng, Murali Chintalapati, Youjiang Wu, and Ken Hsieh,** ***Microsoft;*** **Kaixin Sui, Xin Meng, Yaohai Xu, and Wenchi Zhang,** ***Microsoft Research;*** **Furao Shen,** ***Nanjing University;*** **Dongmei Zhang,** ***Microsoft Research***

https://www.usenix.org/conference/atc19/presentation/zhang-xu

This paper is included in the Proceedings of the 2019 USENIX Annual Technical Conference.

July 10–12, 2019 • Renton, WA, USA

ISBN 978-1-939133-03-8

Open access to the Proceedings of the 2019 USENIX Annual Technical Conference is sponsored by USENIX.

Cross-dataset Time Series Anomaly Detection for Cloud Systems

Xu Zhang[1,2], Qingwei Lin[2], Yong Xu[2], Si Qin[2], Hongyu Zhang[3], Bo Qiao[2], Yingnong Dang[4], Xinsheng Yang[4], Qian Cheng[4], Murali Chintalapati[4], Youjiang Wu[4], Ken Hsieh[4], Kaixin Sui[2], Xin Meng[2], Yaohai Xu[2], Wenchi Zhang[2], Furao Shen[1], and Dongmei Zhang[2]

[1]*Nanjing University, Nanjing, China*
[2]*Microsoft Research, Beijing, China*
[3]*The University of Newcastle, NSW, Australia*
[4]*Microsoft Azure, Redmond, USA*

Abstract

In recent years, software applications are increasingly deployed as online services on cloud computing platforms. It is important to detect anomalies in cloud systems in order to maintain high service availability. However, given the velocity, volume, and diversified nature of cloud monitoring data, it is difficult to obtain sufficient labelled data to build an accurate anomaly detection model. In this paper, we propose cross-dataset anomaly detection: detect anomalies in a new unlabelled dataset (the target) by training an anomaly detection model on existing labelled datasets (the source). Our approach, called ATAD (Active Transfer Anomaly Detection), integrates both transfer learning and active learning techniques. Transfer learning is applied to transfer knowledge from the source dataset to the target dataset, and active learning is applied to determine informative labels of a small part of samples from unlabelled datasets. Through experiments, we show that ATAD is effective in cross-dataset time series anomaly detection. Furthermore, we only need to label about 1%-5% of unlabelled data and can still achieve significant performance improvement.

1 Introduction

In recent years, we have witnessed increasing adoption of cloud service systems. Many software applications are now deployed on cloud computing platforms such as Microsoft Azure, Google Cloud Platform, and Amazon Web Services (AWS). As the cloud systems could be used by millions of users around the world on a 24/7 basis, high service reliability and availability are critical.

However, cloud systems, like other software systems, may exhibit some anomalous behaviors. These anomalies could seriously affect service availability and reliability and could even lead to huge financial loss. The anomalies could be caused by a variety of factors (such as software bugs, disk failures, memory leaks, network outage, etc.) and reflected by a variety of cloud monitoring data (such as KPI, performance counters, usage statistics, system metrics, logs, etc.). The cloud monitoring data are usually time series data, which have high velocity and enormous volume because of the scale and complexity of cloud systems. To maintain high service reliability and availability, it is important yet challenging to detect anomalies from a large amount of cloud monitoring data precisely and timely.

Specifically, anomaly detection in practice encounters several challenges due to the characteristics of cloud systems. A large cloud system is composed of a variety of services and each service is associated with some monitoring data. For some types of data, the characteristics of anomalies are common across many services. While for some other types of data, the characteristics of anomalies could differ from service to service. For example, 90% CPU utilization is normal for computation intensive services but anomalous for other services. Therefore, simple threshold-based anomaly detectors can hardly perform well for a variety of services.

Over the years, many machine-learning based anomaly detection methods have been proposed, including supervised [22, 25] and unsupervised methods [2, 41, 38]. However, it is not trivial to detect anomalies in a large and diversified set of time series data in real cloud environment where labelled data is scarce but a high detection performance is demanded. Unsupervised learning methods can deal with a large amount of data as they do not require labelled data. However, the performance achieved by these methods is rather low [13]. Although supervised learning methods can achieve higher accuracy than the unsupervised counterparts, it is time-consuming and tedious to manually label the anomalies due to the volume and diversity of cloud monitoring data. Therefore, supervised-learning based methods are difficult to be applied to anomaly detection in practice.

Facing the above challenges, to build an accurate and efficient anomaly detection model, we propose ATAD, which enables cross-dataset anomaly detection for cloud systems. The main idea of cross-dataset is to perform anomaly detection on an unlabelled dataset (the target dataset) by learning

from existing, labelled datasets (the source datasets). For example, a detector can be learned from a public dataset such as NAB [23], and then applied to an unlabelled dataset collected from a real-world system.

ATAD consists of two major components: 1) Transfer Learning, which transfers the common anomalous behavior learned from a labelled time series data to a large volume of target unlabelled dataset. Through transfer learning, the commonalities across datasets could be leveraged and the labelling effort for the target dataset could be reduced. 2) Active Learning, which improves the detection performance by labelling only a small number of selected samples in the target dataset. Through active learning, the diversified data with specific characteristics can be addressed with a small amount of labelling effort.

In particular, in the Transfer Learning component, we identify multiple features of cloud monitoring data and perform clustering to select an appropriate subset of existing labelled data as the sub source domain. Then the CORAL algorithm [37] is applied to narrow the feature difference between the source and target domain. In the Active Learning component, we utilize the UCD (Uncertainty-Context-Diversity) method to recommend informative data points to be labelled. The labelled points are used to retrain the classifier trained from the Transfer Learning component. In this way, we aim at minimizing the labelling effort and improving the performance of the detector as much as possible.

We have conducted experiments on public datasets to verify the effectiveness of our method. The experimental results show that using ATAD we can achieve cross-dataset anomaly detection with good accuracy. We test the effectiveness on both public datasets and real-world cloud monitoring data. On public datasets, ATAD shows higher accuracy than existing methods when performing anomaly detection on a target dataset (i.e. Yahoo) by the detector learned from a source dataset (Non-Yahoo, like AWS, Twitter and Artificial datasets). Furthermore, labelling about 1%-5% of unlabelled data could achieve much higher F1-score than the related methods. We also train an ATAD model using public datasets and apply the trained model to detect anomalies in real-world cloud monitoring data of Microsoft. ATAD achieves the best F1-Score, which is much higher than those achieved by other methods.

The contributions of this paper are as follows:

- We propose a new anomaly detection method called ATAD, which enables cross-dataset anomaly detection for cloud systems.
- To the best of our knowledge, we are among the first to detect anomalies in time series cloud data using a combination of transfer learning and active learning techniques.
- We have performed an extensive evaluation of the proposed approach using public and real-world datasets.

This paper is organized as follows: we first elaborate the background and motivation of our work in Section 2. In Section 3, the details of ATAD are described. Section 4 reports the experiments and corresponding results. Next, we discuss threats to validity in Section 5. We introduce the related work in Section 6, before concluding the paper in Section 7.

2 Background and Motivation

For cloud service vendors like Microsoft Azure, Google Cloud Platform, and Amazon Web Services (AWS), there are millions of servers and virtual machines providing a variety of services to users. Despite many quality assurance methods, it is difficult to avoid system failures in reality. A severe system failure can cause damage to user's operation and vendor's reputation. Recovering system from failures in time is of great importance. In order to do that, quick and accurate anomaly detection is essential.

Anomaly detection is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data [43]. Anomaly detection in cloud is usually performed on Cloud Monitoring Data (such as KPI, performance counters, CPU utilization, VM downtime, system workload, etc.). The cloud monitoring data is often presented in time series, that is a series of numerical data points recorded in time order.

Unlike a general anomaly detection problem, it is much more difficult to detect anomalies in a large-scale cloud service system. We identify the following challenges:

- Diverse characteristics of anomalies: in a large-scale cloud service system, different usage scenarios and components have different levels of tolerance to anomalies. For example, a minor system deviation occurring in a certain key component, like storage cluster, may become an anomaly and lead to the failure of the whole system [11, 21]. However, such a deviation may not cause serious problems in other components. It is difficult to set accurate thresholds of anomalies for each usage scenario and system component [11]. Therefore simple threshold-based anomaly detection methods are not suitable for cloud service systems.
- Anomaly detection in time series data: cloud monitoring data is large-scale time series data that has temporal property. Many commonly-used machine learning algorithms cannot be directly applied because the time series data does not satisfy the independent and identically distributed (i.i.d) assumption. Although some deep learning models, like LSTM [17], could capture the temporal property, they require enormous labelled data to train an accurate model. Thus, an appropriate approach to incorporate the temporal property of time series data is important.

- Unsatisfactory performance of unsupervised learning: unsupervised machine learning techniques such as Isolation Forest [26] or Seasonal Hybrid ESD [16] can be applied to anomaly detection. These methods detect anomalies by checking outliers/deviations from the normal data distribution. However, the effectiveness of unsupervised anomaly detection algorithms is often unsatisfactory [13]. The false alarm rate of unsupervised models is higher, which requires much more effort for engineers to check the status of the cloud system.

- Lacking labels for supervised learning: As mentioned above, if the temporal property of time series data can be well incorporated into the labelled data, supervised machine learning methods such as SVM or Random Forest are good to be used to learn and predict anomaly patterns [29]. However, due to the scale and complexity of a cloud service system, labelling the whole dataset requires enormous human effort and is an almost impossible task. The problem of lacking labelled data limits the application of supervised anomaly detection methods to cloud service systems.

3 Proposed Approach

In order to address the challenges mentioned above, in this paper, we propose a novel time series anomaly detection method called ATAD (Active Transfer Anomaly Detection), which combines transfer learning and active learning technologies for anomaly detection in cloud monitoring data. Fig. 1 shows the overall workflow of ATAD.

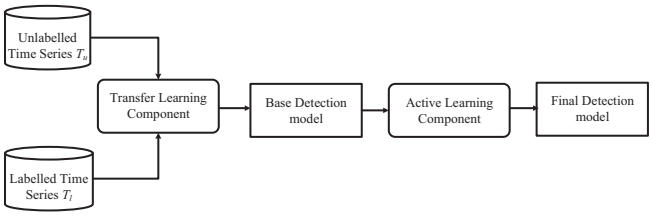


Figure 1: The overall workflow of ATAD

There are two sets of input data, one is the unlabelled time series data T_u on which anomaly detection will be conducted, the other is labelled time series data T_l collected from the public domains or the other components of the cloud system. In T_l, each point in the time series has been manually labelled as either anomaly or normal.

Our approach consists of two main components, namely *Transfer Learning component* and *Active Learning component*. In transfer learning, to incorporate temporal property of time series data, multiple general features are extracted from the raw dataset T_l and form the feature dataset F_l. Feature-based and instance-based transfer learning methods are then applied on F_l to learn a base detection model. After that, the unlabelled time series T_u goes through the same feature extraction process and forms the feature dataset F_u. The active learning component recommends a small number of informative samples from F_u for labelling through *Uncertainty and Context Diversity* (UCD) strategy. Then the labelled data is used to retrain the base detection model. After T rounds of active learning, we obtain the final anomaly detector.

We will describe more details about ATAD in the following sections.

3.1 Transfer Learning Component

Many machine learning methods assume that the distributions of labelled and unlabelled data are the same. However, transfer learning, in contrast, allows the domains, tasks, and distributions used in training and testing to be different [30]. In order to achieve knowledge transfer among different time series datasets, we utilize an efficient and effective transfer method for large-scale cloud monitoring data. For the anomaly detection problem in cloud systems, it is non-trivial to perform transfer learning directly. We need to consider the following factors when we design our transfer learning component.

- Cloud monitoring data is usually presented in the form of time series. Time series is not independent data, but a set of data points with temporal dependence. Thus the anomaly patterns of time series have contextual relevance. How to incorporate this relevance into anomaly detection is a challenge. In our work, we extract time series related features at each data point. Each data point is transformed from the original single-dimensional scalar into a high-dimensional feature vector, and the contextual information is preserved by these features. In ATAD, we not only take account of the simple descriptive features (statistical values), but also the order-aware features (forecasting error features and temporal features).

- For a time series, it is a problem what granularity transfer learning should be performed at. We can conduct transfer learning on entire time series, subseries, or discrete time points. If transfer learning is performed at a coarse-granularity (e.g. the entire time series or subseries) that contains several different anomalous patterns, it is not conducive to distinguish them. Further, coarse-granularity anomaly detection leads to the difficulty of locating and retrieving the cause of anomalies. In this work, we aim to conduct anomaly detection at a fine granularity (i.e. for each data point), and so we perform transfer learning at the level of data point.

- Transfer learning requires that the source domain and

the target domain have the underlying similarity. However, the time series generated from various components in a large-scale cloud system could be very different. We should guarantee that the source domain and the target domain come from similar services or have similar characteristics. Thus, during the transfer learning process, we need to filter out those source-domain samples that are not similar to the counterpart in the target domain.

Fig. 2 shows the workflow of the Transfer Learning Component. The following subsections describe our algorithm in detail.

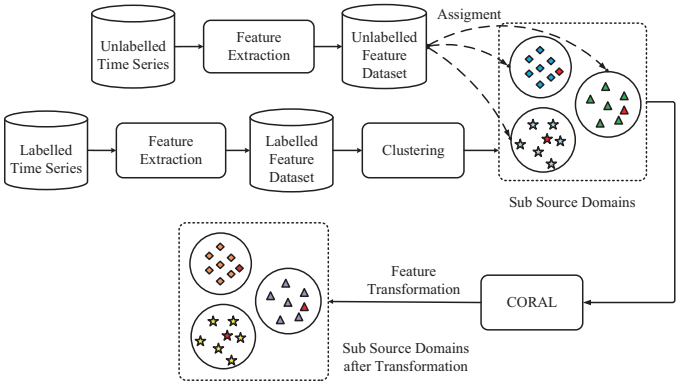


Figure 2: Transfer Learning Component

3.1.1 Feature Identification

The feature engineering process of ATAD converts each data point in a time series (T_l) into a set of features (F_l), which can capture both contextual and temporal information around the point. These features can be categorized into three groups: statistical features, forecasting error features, and temporal features.

Before computing the value of these features, we use Discrete Fourier Transform (DFT) to estimate the period p of the most dominant frequency. Different periods determine different sizes of the sliding window used in the following process.

Statistical features: Statistical features describe some basic characteristics around each data point in time series. We hold a view that the statistical features are able to describe the basic characteristics of different time series generated from various sources in cloud systems, and it is conducive to detecting anomalies that violate the basic characteristics. For example, in an active-running computation intensive service, if the average CPU utilization over a time window tends to be low, it may be an indicator that part of the computation process on it halts unexpectedly.

We identify features for depicting statistical characteristics of time series data (as listed in Table 1). These descriptive features are all calculated in a rolling window derived from the period p. They can represent short-period aspects of time series data such as mean, variance, autocorrelation, trend, remainder, and stationary test.

Table 1: Statistical Features

Feature	Description
Mean	Mean.
Var	Variance.
Crossingpoint[18]	The number of crossing points.
ACF1	First order of autocorrelation.
ACFremainder	Autocorrelation of remainder.
Trend	Strength of trend.
Linearity [18]	Strength of linearity computed on trend of STL [5] decomposition.
Curvature [18]	Strength of curvature computed on trend of STL [5] decomposition.
Entropy [18]	Spectral entropy [12].
ARCHtest.p [9]	P value of Lagrange Multiplier (LM) test for ARCH model [8].
GARCHtest.p [24]	P value of Lagrange Multiplier (LM) test for GARCH model [8].

Forecasting error features: Following the prior work [22], we use a set of error metrics resulted from time series forecasting as features. The intuition is that if the value of current point deviates from the forecasting result, there is more likely to be an anomaly. We use ensemble models to carry out forecasting and apply different models based on the seasonality of the time series. The models leverage classical forecasting techniques, namely SARIMA [27], Holt [19], Holt-Winters [19], and STL [5] for seasonal data, and SARIMA, Holt, Holt-Winters, and Polynomial Regression [15] for non-seasonal data, respectively. The metric *RMSE* (Root Mean Squared Error) is used to endow different forecasting methods with different weights at a fixed time. More weight should be assigned to more precise forecasting model. The weighted prediction result of the ensemble model from M models at time t is calculated by:

$$\hat{Y}_t = \sum_{m=1}^{M} \frac{\hat{Y}_{m,t}}{M-1} \cdot \left(1 - \frac{RMSE_{m,t}}{\sum_{n=1}^{M} RMSE_{n,t}}\right) \quad (1)$$

where $\hat{Y}_{m,t}$ is the prediction by model m at time t, $RMSE_{m,t}$ is the prediction error of model m at time t, $\hat{Y}_t$ is the ensemble prediction at time t.

After gaining the ensemble prediction $\hat{Y}_t$, we calculate 5 metrics on 3 rolling time windows to measure the bias between predicted and actual values. The metrics are shown

in Table 2, where $\hat{Y}$ is the predicted value, Y represents the actual value, and N is the size of the time window.

Table 2: Metrics used as forecasting error features

Features	Formula	Description
ME	$\frac{\sum(Y_i-\hat{Y}_i)}{N}$	Mean Error.
RMSE	$\sqrt{\frac{\sum(Y_i-\hat{Y}_i)^2}{N}}$	Root Mean Squared Error.
MAE	$\frac{\sum\lvert Y_i-\hat{Y}_i\rvert}{N}$	Mean Absolute Error.
MPE	$\frac{1}{N}\cdot\sum\frac{Y_i-\hat{Y}_i}{Y}$	Mean Percentage Error.
MAPE	$\frac{1}{N}\cdot\sum\frac{\lvert Y_i-\hat{Y}_i\rvert}{Y}$	Mean Average Percentage Error.

Temporal features: Generally speaking, the drastic changes of system metrics are likely to be anomalies. For example, the sharp decline of disk I/O traffic rate may be caused by the hardware failure in the disk array. To understand the changes of time series data over time, we identify temporal features (as shown in Table 3) by comparing data in two consecutive windows. We also compute the difference between current values and previous w values (e.g., the difference between x_{n-w} and x_n). In our implementation, we set $w = p/2, p, 2p, w_{pr}, w_{pr}/2$, respectively to get the corresponding different values (Diff-w). w_{pr} is selected according to some prior knowledge. For example, if a time series is recorded by hours, we can set $w_{pr} = 24$.

Table 3: Temporal Features

Features	Description
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 remainder.
Flatspots	Discretize time series values into ten equal-sized intervals. Find maximum run length within the same bucket. [18].
Diff-w	The differences between the current value and the w-th previous value.

In summary, with original time series value, we extract total 37 features used to capture the characteristics of time series data. It is also worth mentioning that all those features are normalized in order to make them comparable among different time series.

3.1.2 The Transfer between Source Domain and Target Domain

In order to transfer knowledge between the source and target domains, it is necessary to narrow the difference between the two domains. Considering the effectiveness and efficiency requirements in the anomaly detection task, we propose a transfer method combining the instance-based transfer learning and feature-based transfer learning.

In transfer learning, we should guarantee that the source domain and the target domain come from similar fields (such as similar monitoring data) or own similar characteristics (such as trend or period). However, the source domain may consist of various time series data. Thus, the first step of transfer learning is to collect the time series data from the source domain that are similar to the data in the target domain. In ATAD, we use instance-based method to filter out those source-domain samples which are not similar to the counterpart in the target domain.

The idea of instance-based transfer learning is to select the source-domain samples which are similar to samples in the target domain so that the difference between two domains can be reduced. For source domain, after identifying features and converting T_l into F_l, we perform *K-means* [29] algorithm on F_l to build K clusters. Each cluster F_l^i $i \in [1,K]$ is a subset of F_l without overlap, which can be regarded as a sub source domain. To select the similar samples, firstly, the same feature extraction process is applied to the unlabelled time series data T_u to form feature dataset F_u. Then we calculate the Euclidean distance between each unlabelled sample and the central point of each cluster. Each target-domain sample will be assigned to the nearest sub source domain F_l^i. We denote the testing samples which are assigned into the same cluster as F_u^i $i \in [1,K]$.

After the instance-based transfer, we need to further narrow the difference between target domains and corresponding sub source domains from the perspective of feature space, because the distribution of features may still remain different. In ATAD, we conduct CORrelation ALignment (CORAL) [37] on each cluster, which is the idea of the feature-based transfer learning process. CORAL is a domain adaption algorithm, which can align the second-order statistics, namely, the co-variance of the source and target features in an unsupervised manner. Specifically, CORAL aims to minimize the Frobenius norm between co-variance matrices of the target and source domains, as shown in Eq. 2.

$$\min_A \left\| A^T C_l^i A - C_u^i \right\|_F^2 \tag{2}$$

where A is a linear transformation matrix, C_l^i is the co-variance matrix of labelled data in cluster i, and C_u^i is the co-variance matrix of unlabelled data in cluster i. We can get the optimal solution of A by whitening source data and re-coloring with target co-variance method, i.e. CORAL. More

details can be referred in [37]. Finally, we can get the new sub source domain features data $\hat{F}_l^i$ after transformation.

In the last step, we train a base supervised model, like Random Forest (RF) or Support Vector Machine (SVM), on each sub source domain $\hat{F}_l^i$. In the end, we can get K independent base models.

Some notes about the proposed transfer learning component:

- Base model: we use Random Forest as the supervised machine learning model (i.e. the base model) in our implementation. Random Forest can be implemented in a parallel way thus it owns high efficiency.
- Computational framework: we emphasize that this component can be regarded as a computational framework. In fact, the choices of distance measurement, clustering method, and base model are flexible.
- Assignment complexity analysis: when assigning unlabelled samples, we need to calculate the distance between each sample in the unlabelled set and the center points of all clusters (sub source domains). This time complexity is $\mathcal{O}(m \cdot K)$, where K is the number of clusters and m is the size of F_u. K is generally much less than m and can be regarded as a constant, so $\mathcal{O}(m \cdot K) \approx \mathcal{O}(m)$. Therefore, our assignment process has linear complexity.
- Parallel processing: it is worth noting that the sub source domains are completely independent to each other, which means that the follow-up processes for each sub source domain could be conducted in a parallel manner. This can help improve the efficiency of anomaly detection.

3.2 Active Learning Component

Due to the high complexity of the cloud service systems, the time series generated from different components are characterized by great diversity. Thus, transfer learning technique is not enough to achieve satisfactory results on various time series in cloud. In ATAD, leveraging active learning method, the diversified data with specific characteristics can be addressed with a small amount of labelling effort.

Active learning focuses on minimizing the labeling effort of users and improving the accuracy of the prediction model. In this work, we utilize an active learning method that considers *Uncertainty* and *Context Diversity* of samples during sampling. We call it UCD for short.

3.2.1 Uncertainty

Most active learning methods use *uncertainty* as the principle to select samples for labelling [33] because it is believed that if a model is less certain about the classification results of some samples, labelling such samples would be more helpful to the base model. In our approach, we use the base model (Random Forest) to estimate the probability of an unlabelled data to be normal or anomalous. We then use the following formula to calculate the uncertainty for unlabelled samples:

$$Uncertainty = -\left|Prob\left(Normal\right) - Prob\left(Anomaly\right)\right| \quad (3)$$

where *Prob* represents the probability given by the base model.

We calculate the uncertainty according to Eq. 3 and sort them in descending order. The larger the uncertainty, the more it needs to be labelled.

3.2.2 Context Diversity

To recommend samples to be labelled, *diversity* is also an important factor to be considered. Sometimes, two samples are very similar or may belong to the same anomaly pattern. It is unnecessary to label both of them.

Traditional diversity methods are generally based on clustering [7], which do not consider the context of samples in time series scenario. In cloud systems, time series of system metrics, such as CPU utilization or traffic load, are continuous without drastic breakpoints. Thus, samples that are adjacent in time series tend to be similar. Our active learning algorithm makes full use of this property in time series.

Specifically, we sort all samples by uncertainty and scan them sequentially. If a new sample we scanned is in the context of another sample in the candidate set, i.e. these two samples are adjacent to each other, we hold a view that the information embedded in them could also be similar. We thus ignore the new sample because it may contain redundant information. If the new sample does not appear in the context of all samples in the candidate set, it is added to the candidate set. In our work, the context of sample x_t is controlled by a parameter α, which represents a range from $x_{t-\alpha}$ to $x_{t+\alpha}$ in a time series. Fig. 3 illustrates the concept of context diversity in a time series. More details can be found in Algorithm 1.

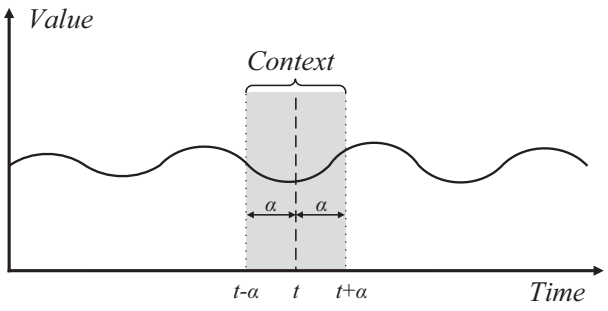


Figure 3: Context in time series

For each source domain, we perform active learning on its own testing data F_u^i. We recommend d diverse and uncertain samples to be labelled, and add the labelled samples into the training set to retrain the base model. After repeating T rounds of this process, we obtain the final detection model.

Algorithm 1: Active Learning Component

Input:
labelled feature data from source domain F_l;
unlabelled feature data from target domain F_u;
base model M_{base} obtained from F_l;
the number of samples at each round d;
context parameter α;
the number of rounds T;

Output: Final model M_{final};

$M = M_{base}$
for $i = 1$ **to** T **do**
  Candidate Set, $S = \varnothing$
  $Prob(Normal), Prob(Anomaly) = M(F_u)$
  $Uncertainty = -|Prob(Normal) - Prob(Anomaly)|$
  $Uncertainty\_Candidate = argsort(Uncertainty)$
  for $j = 1$ **to** $S.size()$ **do**
    if $S == \varnothing$ **then**
      $S = S \cup Uncertainty\_Candidate[j]$
      continue
    end
    if $S.size() > d$ **then**
      break
    end
    if $Uncertainty\_Candidate[j] \notin [x_{t-\alpha}, x_{t+\alpha}], \forall x_t \in S$ **then**
      $S = S \cup Uncertainty\_Candidate[j]$
    end
  end
  label the S as F_{fed}
  $F_l = F_l \cup F_{fed}$
  $M = M.train(F_l)$
end
return M *as* M_{final}

3.3 Usage of ATAD

Transfer learning and active learning are only conducted in the training process. Once the training of ATAD finished, a classifier will be generated and further applied to anomaly detection task in practice. The detection process of ATAD is as follows: Firstly, we feed the time series data to be detected into the feature extraction component and extract features as the training process does. After that, the features are input to the trained classifier to get the anomaly probabilities. Finally, the points whose probabilities are higher than a pre-specified threshold are predicted as anomalies. This pre-specified threshold can be treated as the sensitivity parameter for adapting to different requirements of various users and scenarios.

4 Experiments

In this section, we evaluate the effectiveness of our approach ATAD through a series of experiments. We aim to answer the following research questions in evaluation:

RQ1: How effective is the proposed ATAD approach?
RQ2: How effective is the Transfer Learning component?
RQ3: How effective is the Active Learning component?
RQ4: How effective is ATAD in detecting anomalies in a company's local dataset based on public datasets?

4.1 Dataset and Setup

We use two public time series anomaly detection datasets, NAB [23] and Yahoo [22], to evaluate our proposed method. NAB is a novel benchmark for evaluating anomaly detection algorithms in streaming, real-time applications. It contains datasets collected from different fields, including AWS, Twitter, and Artificial, etc. Each dataset contains several time series of variable length. The AWS dataset contains different server metrics, such as CPU utilization, network traffic, disk write bytes, etc, collected by the Amazon CloudWatch service. The Artificial dataset contains artificially generated time series data with various types of anomalies, while the anomaly patterns are much simpler. The Twitter dataset is the collection of Twitter mentions of large publicly-traded companies such as Google and IBM. The Yahoo dataset consists of metrics of various Yahoo services, which reflects the status of Yahoo system. All datasets are given in time series form and every data point is manually labelled. These time series range in length from hundreds to thousands. The proportion of anomaly is about $1\% \sim 5\%$.

In the experiments, we use Yahoo, AWS, Artificial and Twitter datasets as the testing set (target domain). There are two reasons for this choice. First, these datasets are related to cloud monitoring data. Second, the scale of these datasets are relatively large, or the anomalous points are also much more than other datasets. More details about datasets are shown in Table 4. The first column is the average length of time series. The second and third columns are the total number of data points and the number of anomalies, respectively. We also show the percentage of anomaly data points in the last column.

We perform cross-dataset anomaly detection according to our setup. The experiments are conducted on four pairs of datasets, including non-Yahoo→Yahoo, non-AWS→AWS, non-Twitter→Twitter, and non-Artificial→Artificial. The right side of the arrow represents the unlabelled testing

Table 4: Summary of datasets

Dataset	time series mean length	#data points	#anomaly points	%anomaly
Yahoo	1415	92016	1617	1.76%
AWS	3985	67740	3097	4.57%
Artificial	4032	16128	624	3.87%
Twitter	15862	142765	217	0.15%

dataset, i.e. target domain and the left side of the arrow represents the labelled dataset from other fields. The labels of the target domain are not used during the training and transfer learning process. They are only used during active learning and evaluation.

4.2 Evaluation Metric

We evaluate the accuracy of anomaly detection methods using F1-Score, which is defined as follows:

$$F1 = \frac{2 \cdot P \cdot R}{P+R}, \quad P = \frac{TP}{TP+FP}, \quad R = \frac{TP}{TP+FN} \tag{4}$$

where P and R denote the precision and recall, respectively. In addition, TP, FP, FN, and TN are referred to as true positive, false positive, false negative, and true negative, respectively. We might fail to detect potential anomalies if only focus on the precision. On the other hand, a couple of false positives might be received when we solely pay attention to the recall. F1-Score builds up the balance of the precision and recall, is therefore used as the main evaluation metric in our experiments.

Under an acceptable recall, we expect the anomaly detector to achieve as high precision as possible. The more precise the detector is, the less amount of false alarms will be reported, and thus less human effort is required to investigate. Therefore, precision can be regarded as an important metric, which reflects the automation degree of anomaly detection systems.

4.3 Results

4.3.1 RQ1: How effective is ATAD?

In this section, we evaluate the effectiveness of our proposed approach, ATAD. First, we compare ATAD with some commonly-used anomaly detection algorithms to examine the superior performance of the proposed approach. Second, we present the advantages of ATAD in saving labelling cost, as a comparison with supervised learning based anomaly detectors.

The comparative anomaly detection algorithms are developed based on Isolation Forest (iForest) [26], K-Sigma [14], Seasonal Hybrid ESD (S-H-ESD) [16] and Random Forest (RF). The iForest model ensembles random split tree models to identify which points are isolated. K-Sigma is a common statistics-based method, in which the samples are taken as anomalies whose values deviate more than k times of the variance of samples from the corresponding mean. S-H-ESD builds upon the Generalized ESD test [32], and is able to detect both global and local anomalies. This algorithm is incorporated in the well-known *AnomalyDetection* R package [39] and thus is widely used. Because the RF is used in ATAD as the based learning classifier, we exploit a classical RF based supervised model as a comparison to demonstrate the superior performance of ATAD. The RF and iForest models use the same features as the ATAD. The K-Sigma and S-H-ESD are performed on the raw time series.

In the experiments, we evaluate the metrics under different settings and present the best results in terms of F1-Score in the following. Towards this end, the proportion of anomalies in iForest is set to 0.01, 0.05, 0.10 and 0.20, respectively, and the integer k varies from 1 to 3 in K-Sigma. In addition, the maximum proportion of anomalies in S-H-ESD is set to 0.01 and 0.05, respectively. We labelled the same proportion of samples for RF as the counterpart in ATAD used in the target domain. In ATAD, the K in the Transfer Learning component changes from 3 to 5, whereas the number of rounds T and the labelling ratio are fixed to 3 and 1%, respectively, in the Active Learning component. The probability threshold is set to $0.6 \sim 0.8$.

The results are shown in Table 5. It is clear that the ATAD can achieve much higher F1-Scores than other approaches on all datasets. Particularly, though the supervised learning method (RF) achieves better performance than those unsupervised methods, the ATAD outperforms the RF when given the same number of labels.

In order to demonstrate the advantages of ATAD in saving labelling efforts, we compare the number of labelled samples of ATAD and RF under the similar F1-Scores. The relevant results are presented in Table 6, whose first and third column depicted the F1-Scores of the supervised model (RF) and the ATAD, respectively. Their corresponding quantities of labelled data are included in the second and fourth column of Table 6. It is evident that the supervised model generally takes $3\sim10$ times more labels than the ATAD to achieve comparable results. The superior performance benefits the transferred knowledge from the source domain in the Transfer Learning component. As a consequence, a small number of labels is required in the target domain. Moreover, the UCD method helps to further reduce the number of labels because the performance can be improved rapidly and significantly by the extraordinary informative recommendations in the Active Learning component.

Table 5: Results of Comparative Methods

Dataset	Method	Precision	Recall	F1-Score
Non-Yahoo → Yahoo	iForest	0.3832	0.2183	0.2781
	K-Sigma	0.6499	0.3364	0.4433
	S-H-ESD	0.2779	0.6215	0.3840
	RF	0.8668	0.2075	0.3348
	ATAD	0.8847	0.4040	**0.5547**
Non-AWS → AWS	iForest	0.1523	0.0491	0.0743
	K-Sigma	0.6899	0.1992	0.3091
	S-H-ESD	0.5382	0.7100	0.6123
	RF	0.9999	0.6226	0.7674
	ATAD	0.9195	0.8142	**0.8637**
Non-Artificial → Artificial	iForest	0.3477	0.9006	0.5017
	K-Sigma	1.000	0.1730	0.2950
	S-H-ESD	0.7888	0.4568	0.5785
	RF	0.9182	0.9301	0.9241
	ATAD	0.9990	0.9850	**0.9924**
Non-Twitter → Twitter	iForest	0.4685	0.3087	0.3722
	K-Sigma	0.2608	1.0000	0.2608
	S-H-ESD	0.7481	0.4654	0.5739
	RF	0.7285	0.4811	0.5795
	ATAD	0.8769	0.6951	**0.7755**

Table 6: Supervised Model (Random Forest) vs. ATAD

	RF	#labels	ATAD	#labels
Non-Yahoo → Yahoo	0.5440	4600	0.5547	920
Non-AWS → AWS	0.8403	763	0.8637	254
Non-Artificial → Artificial	0.9755	1612	0.9924	161
Non-Twitter → Twitter	0.7126	17131	0.7755	1427

4.3.2 RQ2: How effective is the Transfer Learning Component?

We evaluate the effectiveness of our Transfer Learning Component from the following two aspects:

- The effectiveness of identified features, including forecasting error, statistical, and temporal features.
- The effectiveness of our proposed transfer method.

Effectiveness of the identified features Our proposed transfer learning is based on many time series features including forecasting error, statistical, and temporal features. The conventional transfer learning is only based on statistical features such as averages and variances [30]. The statistical features are simple descriptive values that are independent of the context of time series.

In order to evaluate the validity of features used in ATAD, we perform an experiment on four dataset pairs. We compare ATAD using statistical features alone and ATAD using order-aware features (forecasting error features and temporal features).

Experimental results are shown in Table 7. It can be seen that using statistical features alone for ATAD leads to poor results, and when order-aware features are added, the performance becomes better. The reason is that the transfer learning needs to narrow the differences between the source domain and the target domain. If only statistical features are extracted and the characteristics of time series are ignored, the features do not reflect the context property in time series. Thus the resulting performance is less satisfactory.

Table 7: The effectiveness of features (F1-Score)

Features	Yahoo	AWS	Artifical	Twitter
Statistical	0.2956	0.7387	0.7441	0.6937
Order-aware	0.4200	0.8441	0.7569	0.6622
All features	**0.5781**	**0.8637**	**0.9924**	**0.7755**

Random Forest also provides a popular approach to feature ranking. Here we use the Random Forest classifier to evaluate the importance of the features used in ATAD. The importance of a feature can be ranked according to the mean decrease impurity [3]. In Table 8, we show the top-10 most important features of the RF models in ATAD. The definitions of these features can be found in Table 1, Table 2 and Table 3. We can see that the forecasting error features and the original time series are much more informative for anomaly detection. In addition, some temporal features, such as Diff-*w* and Flatspots, also play an important role in the model. It is also worth noting that the important features of each dataset are different. It implies that we should consider all the features comprehensively when conducting transfer learning between different datasets.

Effectiveness of the proposed transfer method In our transfer learning, we create multiple independent sub source domains through clustering and conduct the CORAL algorithm to transform the features of sub source domains. We expect the transfer learning can reduce the labelling effort for users, because we can activate the Active Learning component directly based on the transfer model, without any manual labels in the testing set. We validate the effectiveness of our Transfer Learning component by comparing with

Table 8: Feature Importance Evaluation

Dataset	Important Features
Yahoo	Original_data, RMSE, MAE, ME, Mean, MPE, Diff-p, Diff-$w_{pr/2}$, Diff-2p, MAPE
AWS	Original_data, RMSE, MAE, ME, Mean, ACF1, Diff-2p, Curvature, Flatspots, Diff-p
Artificial	Original_data, Mean, Diff-p, MPE, Flatspots, RMSE, MAE, Diff-2p, Diff-w_{pr}, Lumpiness.
Twitter	Var, RMSE, MAE, ARCHtest.p, Mean, Flatspots, Original_data, Max_level_shift, MPE, Original Data

naive active learning applied directly on target domain without transfer methods.

Specifically, in the naive method, we pre-fetch a part of samples in the testing dataset to train a base model. After that, we use this base model to conduct the active learning process. This naive active learning approach needs no auxiliary public labelled data as source domain and transfer learning technique, but it needs more labelling effort for building the base model, as illustrated in Table. 9.

Table 9: Comparative Experiment of ATAD and Naive Active Learning without Transfer Learning (F1-Score)

	Naive		ATAD	
	F1-Score	#labels	F1-Score	#labels
Yahoo	0.5691	1380	0.5697	920
AWS	0.8589	381	0.8637	254
Artificial	0.9815	322	0.9924	161
Twitter	0.6164	2855	0.7755	1427

In Table 9, we fix the labelling ratio of the active learning process to 1% in both Naive method and ATAD. For fairness, in the naive method, we pre-fetch samples from the testing set in stratified style to train the base model. From the table, it can be seen that the number of samples required for the naive method without transfer learning component is, in average, 1.56 times than that of ATAD, but its results are still slightly lower than those achieved by ATAD. We also illustrate the number of labels required by the supervised model, naive active learning model without transfer methods, and ATAD in Fig 4. It is clear that active learning can significantly save labelling effort and ATAD can further reduce the labelling cost by introducing the transfer learning component.

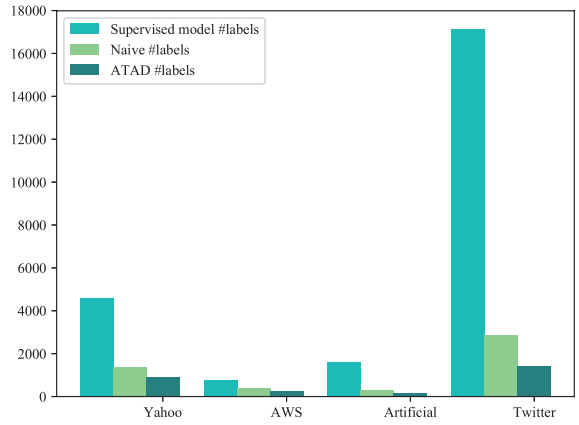


Figure 4: The number of labels required by Supervised Model, Naive Active Learning without transfer learning and ATAD

Table 10: The experimental result with active learning alone (F1-Score)

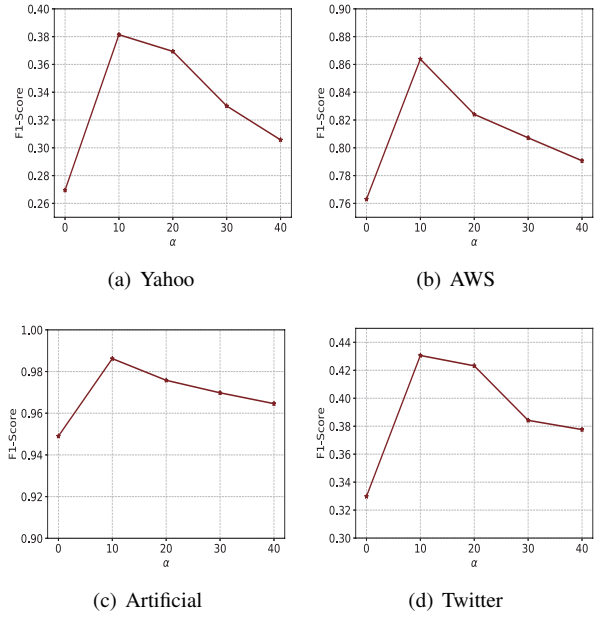
Dataset	Method	Base	Round1	Round2	Round3
Yahoo	U	0.2090	0.2437	0.2839	0.2695
	UCD	0.2090	0.2383	0.3032	**0.3814**
	random	0.2090	0.2275	0.2256	0.2196
AWS	U	0.0443	0.1932	0.6838	0.6799
	UCD	0.0443	0.7550	0.8092	**0.8879**
	random	0.0443	0.3227	0.5164	0.6604
Artificial	U	0.6239	0.7272	0.8737	0.9006
	UCD	0.6239	0.7340	0.8780	**0.9715**
	random	0.6239	0.6446	0.6275	0.5000
Twitter	U	0.1647	0.2916	0.2959	0.3298
	UCD	0.1999	0.3070	0.3703	**0.4232**
	random	0.1647	0.1798	0.1944	0.1689

4.3.3 RQ3: How effective is the Active Learning component?

To evaluate the effectiveness of the Active Learning component, we use total labelled datasets F_l to train the base model and do not apply transfer learning on them. We compare the UCD method with the conventional Uncertainty method (U) and the random selecting method (random). In all experiments, we conduct 3 rounds of active learning and select 60 samples for labelling at each round. In order to avoid the data leakage problem [20], all samples labelled by the active learning component are removed from the testing set. The experimental results are shown in Table 10.

From Table 10, it can be seen that the UCD method achieves the best results on all datasets, confirming the usefulness of incorporating time series context diversity. We

also find that as the number of rounds increases, the F1-Scores achieved by UCD are also steadily improved. However, the random selection method cannot guarantee such trend. The results confirm the validity of the active learning method.



(a) Yahoo (b) AWS

(c) Artificial (d) Twitter

Figure 5: Experimental results with different α

The hyper-parameters in the Active Learning component include the number of samples to be labelled per round d, the number of rounds T, and the context parameter α. Obviously, if T and d are much larger, the entire algorithm will be closer to supervised learning. Therefore the accuracy will be improved gradually.

We conduct an experiment to explore the impact of different α values on the final results. In Fig. 5, we can see that as the α value increases, the F1-Score rises rapidly first and then falls gradually. The larger α means the wider range of context and more attention to Context Diversity because the wider context will cause more uncertain candidate samples to be discarded. The smaller α means the narrower range of context and more attention to Uncertainty. If $\alpha = 1$, UCD becomes purely U. Therefore, α can be treated as a parameter which trades off between Context Diversity and Uncertainty. Too large or too small α value will result in loss of accuracy. In conclusion, choosing an appropriate α can make full use of both kinds of information. Through our practice on all experimental datasets, we empirically set $\alpha = 10$.

4.3.4 RQ4: How effective is ATAD in detecting anomalies in a company's local dataset based on public datasets?

In RQ1, we examined the performance of ATAD on public datasets. In this RQ, we evaluate its effectiveness by using practical industrial data from the large-scale cloud system in Microsoft. More specifically, we hourly record IOPS (I/O Operations Per Seconds) of the storage service in a cluster and collect the corresponding time series data in the past few months. The data is labelled by domain experts in the operation team. All public datasets are utilized as the source domain, and then the ATAD is applied to the target domain dataset, i.e., the IOPS data, for anomaly detection. The experiment is conducted as the similar process in RQ1.

Table 11: Experimental result on IOPS dataset of Microsoft

	Precision	Recall	F1-Score
iForest	0.2886	0.3988	0.3349
K-Sigma	0.8170	0.1882	0.3059
S-H-ESD	0.9117	0.1741	0.2924
RF	0.5213	0.6724	0.5873
ATAD	0.8082	0.6188	**0.7009**

The results are shown in Table 11. It is noted that the traditional unsupervised methods cannot work well due to the low recalls and the resulting F1-Scores. The extremely low recalls exhibit a high probability in failing to detect potential anomalies, which might cause a huge customer and financial loss. RF can achieve a higher recall but lower precision, which means lots of false alarms. In contrast, ATAD can achieve a higher F1-Score, which is beneficial to improving the service availability and detection automation.

5 Threats to Validity

- Data quality: in this work, we use public datasets in evaluation. The labels about anomalies are provided with the datasets. Although these dataset are of high quality and are used by several other studies [23, 22], it is possible that they contain a small degree of noise. Furthermore, their data volume is also limited. We will experiment with larger scale datasets in our future work.

- Correctness of labelling: the Active Learning component in ATAD requires users to manually label a few percentages of data and assumes that the labels are correct. However, in reality, the quality of labeling may vary (i.e., users may incorrectly label a data point).

- Data leakage: in order to avoid the data leakage problem, we remove the samples labelled in the active learning process from the testing set. Since active learning

may recommend different samples to be labelled at different rounds, the resulting testing set could be slightly different in each experiment, which may cause bias in comparisons. However, the proportion of samples to be labelled is very small, so we can ignore this influence on the final results.

6 Related Work

In recent years, there have been many studies on anomaly detection problem for time series data. The proposed methods can be largely divided into supervised methods, semi-supervised methods, unsupervised methods and statistical-based methods [4].

Unsupervised methods do not require manual labelling. These methods assume that normal instances are far more frequent than anomalies and anomalies deviate from the normal data distribution. For example, Ahmad, et al. proposed an unsupervised online sequence memory algorithm called Hierarchical Temporal Memory to detect anomaly in streaming data [2]. Xu, et al. proposed Donut, an unsupervised anomaly detection algorithm based on Variational Auto Encoder (VAE) [41].

Supervised methods aim to build a classification model for normal and anomaly classes. General supervised machine learning models can be applied to this problem. Steinwart, et al. interpreted anomaly detection problem as a binary classification task and proposed a supervised framework [36]. The most famous is the anomaly detection system of Yahoo, called EGADS [22], which used a collection of anomaly detection and forecasting models with an anomaly filtering layer for accurate and scalable anomaly detection on time series. Opperentice [25] used operators' periodical labels on anomalies to train a random forest classifier and automatically select parameters and thresholds.

Semi-supervised methods assume that the training data has labelled instances for only the normal class. Malhotra, et al. used stacked LSTM networks trained on non-anomalous data as a predictor to detect anomaly [28]. Erfani, et al. used a combination of one-class SVM model and deep learning model to detect anomaly [10]. Daneshpazhouh, et al. presented an entropy-based method that consists of two phases, including reliable negative examples extraction and entropy-based outlier detection [6].

Statistical-based methods are built up based on the statistical theory. The most famous method is K-Sigma method [14], in which the samples are taken as anomalies whose values deviate more than k times of the variance of samples from the corresponding average. Recently, more advanced methods using Extreme Value Theory [34, 42] were proposed. Compared with K-Sigma method, these methods do not require prior assumption on the data distribution.

From the view of effectiveness, supervised and semi-supervised methods generally perform better than unsupervised methods and statistical-based methods. However, due to the high labelling cost, they are difficult to be applied in the real world when the scale of dataset is very large. More recently, transfer learning and active learning techniques are applied to deal with this important problem. Transfer learning has been widely applied in many fields like classification, regression, and forecasting [30]. For example, Spiegel, et al. embeds a given set of labelled data into dissimilarity space, leading to enriched feature representations that facilitate statistical learning procedures [35]. Vercruyssen, et al. transferred labelled examples from a source domain to a target domain where no labels are available and constructed a nearest-neighbor classifier in the target domain with DTW measure [40]. Another technique to reduce the labelling effort is active learning. For example, Abe, et al. used a selective sampling mechanism based on active learning to the reduced classification problem of outlier detection [1]. Pelleg, et al. proposed a novel active learning method to identify rare category records in an unlabelled noisy set with a small budget of data points that they are prepared to categorize [31].

In our work, we combine transfer learning and active learning methods so that we could achieve a balance between labelling effort and performance. On the one hand, unlike unsupervised models, ATAD could achieve a high F1-Score. On the other hand, we only need to label a few number of samples from the unlabelled dataset. These advantages are not possessed by the existing methods mentioned above.

7 Conclusion

In this paper, we propose a novel anomaly detection method ATAD for cloud service systems. ATAD combines transfer learning and active learning techniques. In transfer learning, we use an existing labelled dataset as the source dataset. We extract multiple features, construct source domains, and use the labelled data in each source domain to train base models for a target, unlabelled dataset. In active learning, we use the UCD method to recommend informative samples in the target dataset to label and retrain the base models. Our experiments on cross-dataset anomaly detection show that we can achieve satisfactory detection accuracy by labeling only a small number of samples in the target dataset. We have also evaluated the effectiveness of ATAD using real-world data collected from a production cloud system in Microsoft.

8 Acknowledgement

We thank Professor Mickey Gabel (University of Toronto) for the valuable and constructive suggestions on this paper.

References

[1] ABE, N., ZADROZNY, B., AND LANGFORD, J. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), ACM, pp. 504–509.

[2] AHMAD, S., LAVIN, A., PURDY, S., AND AGHA, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing 262* (2017), 134–147.

[3] BREIMAN, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[4] CHANDOLA, V., BANERJEE, A., AND KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 15.

[5] CLEVELAND, R. B., CLEVELAND, W. S., AND TERPENNING, I. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics 6*, 1 (1990), 3.

[6] DANESHPAZHOUH, A., AND SAMI, A. Entropy-based outlier detection using semi-supervised approach with few positive examples. *Pattern Recognition Letters 49* (2014), 77–84.

[7] DASGUPTA, S., AND HSU, D. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 208–215.

[8] ENGLE, R. Garch 101: The use of arch/garch models in applied econometrics. *Journal of economic perspectives 15*, 4 (2001), 157–168.

[9] ENGLE, R. F. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica 50*, 4 (1982), 987–1007.

[10] ERFANI, S. M., RAJASEGARAR, S., KARUNASEKERA, S., AND LECKIE, C. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition 58* (2016), 121–134.

[11] GABEL, M., SCHUSTER, A., BACHRACH, R.-G., AND BJØRNER, N. Latent fault detection in large scale services. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)* (2012), IEEE, pp. 1–12.

[12] G.GOREG. Forecastable component analysis. 64–72.

[13] GÖRNITZ, N., KLOFT, M., RIECK, K., AND BREFELD, U. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research 46* (2013), 235–262.

[14] GRAFAREND, E. W. Linear and nonlinear models: Fixed effects, random effects, and mixed models. *Walter De Gruyter* (2006).

[15] HEIBERGER, R. M., AND NEUWIRTH, E. *Polynomial Regression*. Springer New York, New York, NY, 2009, pp. 269–284.

[16] HOCHENBAUM, J., VALLIS, O. S., AND KEJARIWAL, A. Automatic anomaly detection in the cloud via statistical learning. *arXiv preprint arXiv:1704.07706* (2017).

[17] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[18] HYNDMAN, R. J., WANG, E., AND LAPTEV, N. Large-scale unusual time series detection. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on* (2015), IEEE, pp. 1616–1619.

[19] KALEKAR, P. S. Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi School of Information Technology 4329008* (2004), 1–13.

[20] KAUFMAN, S., ROSSET, S., PERLICH, C., AND STITELMAN, O. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD) 6*, 4 (2012), 15.

[21] KAVULYA, S. P., DANIELS, S., JOSHI, K., HILTUNEN, M., GANDHI, R., AND NARASIMHAN, P. Draco: Statistical diagnosis of chronic problems in large distributed systems. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)* (2012), IEEE, pp. 1–12.

[22] LAPTEV, N., AMIZADEH, S., AND FLINT, I. 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* (2015), ACM, pp. 1939–1947.

[23] LAVIN, A., AND AHMAD, S. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on* (2015), IEEE, pp. 38–44.

[24] LEE, J. H. H. A lagrange multiplier test for garch models. *Economics Letters 37*, 3 (1991), 265–271.

[25] LIU, D., ZHAO, Y., XU, H., SUN, Y., PEI, D., LUO, J., JING, X., AND FENG, M. Opprentice: Towards practical and automatic anomaly detection through machine learning. In *Proceedings of the 2015 Internet Measurement Conference* (2015), ACM, pp. 211–224.

[26] LIU, F. T., TING, K. M., AND ZHOU, Z.-H. Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on* (2008), IEEE, pp. 413–422.

[27] MAKRIDAKIS, S., AND HIBON, M. Arma models and the box–jenkins methodology. *Journal of Forecasting 16*, 3 (1997), 147–163.

[28] MALHOTRA, P., VIG, L., SHROFF, G., AND AGARWAL, P. Long short term memory networks for anomaly detection in time series. In *Proceedings* (2015), Presses universitaires de Louvain, p. 89.

[29] NASRABADI, N. M. Pattern recognition and machine learning. *Journal of electronic imaging 16*, 4 (2007), 049901.

[30] PAN, S. J., AND YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering 22*, 10 (2010), 1345–1359.

[31] PELLEG, D., AND MOORE, A. W. Active learning for anomaly and rare-category detection. In *Advances in neural information processing systems* (2005), pp. 1073–1080.

[32] ROSNER, B. Percentage points for a generalized esd many-outlier procedure. *Technometrics 25*, 2 (1983), 165–172.

[33] SETTLES, B. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning 6*, 1 (2012), 1–114.

[34] SIFFER, A., FOUQUE, P.-A., TERMIER, A., AND LARGOUET, C. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), ACM, pp. 1067–1075.

[35] SPIEGEL, S. Transfer learning for time series classification in dissimilarity spaces. *Proceedings of AALTD* (2016), 78.

[36] STEINWART, I., HUSH, D., AND SCOVEL, C. A classification framework for anomaly detection. *Journal of Machine Learning Research 6*, Feb (2005), 211–232.

[37] SUN, B., FENG, J., AND SAENKO, K. Return of frustratingly easy domain adaptation. In *AAAI* (2016), vol. 6, p. 8.

[38] TERZI, D. S., TERZI, R., AND SAGIROGLU, S. Big data analytics for network anomaly detection from netflow data. In *Computer Science and Engineering (UBMK), 2017 International Conference on* (2017), IEEE, pp. 592–597.

[39] TWITTER. Anomalydetection. `https://github.com/twitter/AnomalyDetection`, 2015.

[40] VERCRUYSSEN, V., MEERT, W., AND DAVIS, J. Transfer learning for time series anomaly detection. *IAL@ ECML PKDD 2017*, 27.

[41] XU, H., CHEN, W., ZHAO, N., LI, Z., BU, J., LI, Z., LIU, Y., ZHAO, Y., PEI, D., FENG, Y., ET AL. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. *arXiv preprint arXiv:1802.03903* (2018).

[42] YUANYAN, L., XUEHUI, D., AND YI, S. Data streams anomaly detection algorithm based on self-set threshold. In *Proceedings of the 4th International Conference on Communication and Information Processing* (2018), ACM, pp. 18–26.

[43] ZIMEK, A., AND SCHUBERT, E. Outlier detection. *Encyclopedia of Database Systems* (2017), 1–5.