RLAD: Time Series Anomaly Detection through Reinforcement Learning and Active Learning

Tong Wu
tong.wu96@rutgers.edu
Rutgers University
Piscataway, New Jersey

Jorge Ortiz
jorge.ortiz@rutgers.edu
Rutgers University
Piscataway, New Jersey

ABSTRACT
We introduce a new semi-supervised, time series anomaly detection algorithm that uses deep reinforcement learning (DRL) and active learning to efficiently learn and adapt to anomalies in real-world time series data. Our model – called RLAD – makes no assumption about the underlying mechanism that produces the observation sequence and continuously adapts the detection model based on experience with anomalous patterns. In addition, it requires no manual tuning of parameters and outperforms all state-of-art methods we compare with, both unsupervised and semi-supervised, across several figures of merit. More specifically, we outperform the best unsupervised approach by a factor of 1.58 on the F1 score, with only 1% of labels and up to ~4.4x on another real-world dataset with only 0.1% of labels. We compare RLAD with seven deep-learning based algorithms across two common anomaly detection datasets with up to ~3M data points and between 0.28% ~ 2.65% anomalies. We outperform all of them across several important performance metrics.

CCS CONCEPTS
• Computing methodologies → Machine learning approaches.

KEYWORDS
anomaly detection, reinforcement learning, active learning

ACM Reference Format:

1 INTRODUCTION
Time series anomaly detection is important across many application domains. Data center monitoring systems [27], sensor networks [24] and the internet of things [20], cyber-physical systems [5, 14, 36] and finance [34], among many others. However, most anomaly detection algorithms are tightly coupled to application-specific data properties. Models are designed for a specific application; setting thresholds, feature extraction processing and hyper-parameters based on strong prior knowledge or extensive experimentation. Some properties include multi-scale temporal dependencies in a multivariate time series setting, anomalies in the frequency domain, and other associated statistical moments. Deep learning techniques have recently been described in the literature. For example SR-CNN [27] casts time-series data into a visual context and spatial correlations are used to identify anomalies. Recently, there has been a new, deep-learning based semi-supervised technique, Deep-SAD [28] which uses an information theoretic criteria to find anomalies.

In addition, anomaly detection algorithms must overcome two major challenges: 1) anomalies are rare and hence, models are difficult to train and 2) many real-world applications have non-stationary properties; that is, the underlying mechanism producing a series of measurements, changes over time. In order to deal with the lack of labeled data, there have been many proposal in the literature that take an unsupervised approach [13, 27, 31, 35, 38]. However, unsupervised algorithms tend to perform quite poorly in practice and are significantly outperformed by semi and fully supervised methods [10, 15, 22, 28]. Supervised models perform very well when labels are available, however do not perform well when there are few or no labels available. Even when labels are readily available, they fundamentally assume that the underlying distribution is stationary. If the distribution changes, they must be re-trained.

To address these challenges, we must design an algorithm that can learn efficiently, using only a fraction of the labels necessary to train a supervised model. To address the second challenge, we must consider adaptive designs; algorithms that adapt to changes as they learn more about the data and the distribution of anomalies within it. We introduce a new algorithm called RLAD, combines reinforcement learning with active learning and label propagation to learn an anomaly-detection model that generalizes and that can adapt and improve dynamically, as it observes more data and learns more about anomalous samples. We show that our model quickly learns an effective anomaly detection model using very few labeled examples, outperforming several state-of-the-art, unsupervised and semi-supervised techniques. We run extensive experiments on several public datasets and show superior performance over the competing state-of-art methods.

To the best of our knowledge, we are the first to combine deep reinforcement learning for anomaly detection with active learning. There are techniques that use reinforcement learning for anomaly detection [8] but are difficult to train without a significant number of labeled samples. There are also techniques that use active learning,
showing competitive performance to semi-supervised models with a fraction of the labels [39]. Our contributions are as follows:

- We demonstrate the first combination of deep reinforcement learning and active learning for anomaly detection, RLAD.
- We run extensive experiments on 3 real-world anomaly data sets and 7 semi-supervised and unsupervised anomaly detection algorithms.
- We show that RLAD outperforms unsupervised techniques by a large margin (~4.4x the F1 score on some data sets)
- We outperform the state-of-the-art, semi-supervised technique Deep-SAD [28] by up to ~10x in our experiments.

The rest of the paper discussed some related work in section 2 and introduce preliminary knowledge in section 3. In section 4, we present an overview of our anomaly detection system. Next, we give a more detailed description of our approach in section 5. In section 6, we provide the experiments of RLAD and evaluation with other approaches. We conclude our work in section 7.

2 RELATED WORK

Most traditional time series anomaly detection approaches are usually simple machine learning algorithms based on distance and density such as K-nearest-neighborhood (KNN) [1], local outlier factor (LOF) [2], local correlation integral (LOCI) [23], isolation forest (iForest) [16]. And others like one-class support vector machine (OC-SVM) [25], principle component analysis (PCA) [30] are commonly used in this area. These approaches are usually time efficient but not able to achieve high performance in real practice.

Recently, deep learning has achieved much success in anomaly detection area. Most deep learning based approaches usually extract features from data and build a classification model to identify anomalies. Supervised deep anomaly detection involves training a deep supervised binary or multi-class classifier, using labels of both normal and anomalous data instances. For instance, [10] is the anomaly detection system of Yahoo, called EGADS. It uses a collection of anomaly detection and forecasting models with an anomaly filtering layer for accurate and scalable anomaly detection on timeseries. There are also general anomaly detection supervised algorithms which are not specific for time series but can be applied to this problem. Oppenentice [15] used operators’ periodical labels on anomalies to train a random forest classifier and automatically select parameters and thresholds. However, these methods rely on tons of labels to train their models, which is quite hard to get in real world.

Advanced unsupervised learning techniques become more popular in this area. They do not require labels and assume the abnormal points have larger deviation from the normal distribution. LinkedIn developed their anomaly detection system Luminol[13], which is a light library of Python. Twitter proposed TwitterAD [35] which is written in R and employs statistical learning to detect anomalies in both application as well as system metrics. Xu, et al. proposed DONUT [38], an unsupervised anomaly detection system based on Variational Auto Encoder (VAE). Autoencoder[7] uses replicator neural networks (RNNs) with three hidden layers to provide a measure of the outlyingness of data records. SR-CNN [27] from Microsoft which borrows the SR model from visual saliency detection domain to time-series anomaly detection and combines with CNN to improve performance. In 2017, SOPT and DSPOT are proposed [31] on the basis of Extreme Value Theory[4]. DAGMM [40] utilizes a deep autoencoder to generate a low-dimensional representation and reconstruction error for each input data point, which is further fed into a Gaussian Mixture Model. However, the performance achieved by these is rather low[6] since they are difficult to leverage prior knowledge (e.g., a few labeled anomalies) when such information is available as in many real-world anomaly detection applications or there is availability of human expert. Moreover, they require the assumption that normal pattern is stationary and are not able to adapt to shifts in input distributions.

Semi-supervised approaches use a fraction of labeled data for training. They are commonly used in classification [3, 9, 19, 26]. Only a few of them have been proposed for anomaly detection problem. [17] and [33] used label propagation process to achieve anomaly detection but only applicable for graph data. [21] introduced REPEN which leverages a few labeled anomalies to learn more relevant features. Deep-SAD [28] used an information-theoretic perspective on anomaly detection and an autoencoder model for pretraining. It uses a hyper-parameter to control the balance of training data. [22] proposed a neural deviation learning based approach and leveraged a few labeled anomalies with a prior probability to enforce the deviation of anomalies in normal distribution. However, these methods highly rely on the number of labeled anomalies in training data. But anomalies are rare in anomaly detection problem. Given a set of unknown samples, it has to inspect a large amount of unknown samples to filter required amount of anomalies. It makes labels more expensive.

In the past three years, reinforcement learning attempted to be introduced in anomaly detection. It could solve problem that sequences without a clear normal pattern because of its self-improving characteristic. [8] proposed an idea to apply deep reinforcement learning on time-series anomaly detection. However, the performance is not able to satisfy real world applications and fully-labeled data is required. [18] applied inverse reinforcement learning technique but it is not specified for time series and need fully-labeled data.

3 PRELIMINARIES

3.1 Reinforcement learning

We consider our anomaly detection problem as a Markov decision process (MDP) which could be expressed by a tuple of \( < S, A, P_a, R_a, γ > \). \( S \) represents the set of environment states. \( A \) is the set of actions taken by RL agent. \( P_a(s, s') \) means the probability that action \( a \) is performed at the state \( s \) and will lead to state \( s' \). \( R_a(s, s') \) is the instant reward received by agent from taking the action \( a \). \( γ \in [0, 1] \) is the discount factor. The value function is defined as \( V_π(s) = \mathbb{E}[\sum_{t=0}^{T} γ^t R_t | s_0 = s] \) represents the expected reward return starting from state \( s \). In MDP, the agent is trying to learn a control policy \( π : S \rightarrow A \) that maximizes the accumulated future reward \( \sum_{t=0}^{T} γ^t R_t \).

3.2 Deep Q Network

Q learning is a famous value-based algorithm. Agents learns the action-value function \( Q(s, a) \) and predicts how good to take an
action as a specific state. The target value is defined by:
\[ \text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \]
And the Q function is updated by:
\[ Q_{k+1}(s, a) \leftarrow (1 - \alpha)Q_k(s, a) + \alpha \text{target}(s') \].

Q learning is unstable or even divergent when action value function is approximated with a nonlinear function like neural networks.[11] Thus, a method called Deep Q network(DQN) that combines reinforcement learning with deep neural network is proposed by Deepmind and approved to be able to handle more complicated problems. It stabilizes the training of action value function approximation into a deep neural network. Experience replay[12] reduces the correlation between samples and improves data efficiency and target network fix target values temporarily in order to speed training process.

### 3.3 Active learning

Active learning is a specific machine learning algorithm which allows the model actively interacts with user to obtain the desired learning experience. The machine(learner) could ask the desired data used in learning process.

Assume we have a training set \( L = (X, Y) \) and an unlabeled instance pool \( U = (x_1, x_2, \ldots, x_n) \). The unlabeled data is normally cheaper than labeled data. Thus, the \( U \) can be very large. Active learning is going to select the unlabeled samples from \( U \) and ask a human expert to label them manually through query function \( Q \). Typically, the chosen samples have more valuable information to the current model \( C \). The learner will get most training benefits from the new labeled instances. The new labeled instance set \( L_{\text{new}} = (X_{\text{new}}, Y_{\text{new}}) \) will be added to the training set \( P \) and used for training in the next iteration. The goal of active learning is to achieve a good classifier model \( C \) with a reasonable number of queries.

### 4 SYSTEM OVERVIEW

In this section, we provide an overview of our system. We proposed a time-series anomaly detection system RLAD which combines reinforcement learning with active learning. Our system contains five modules. Data Pre-processing standardizes values and segments time-series through sliding windows. Also, we initialize label and pseudo label for each timestamp. Then, the preprocessed data will go through Warm Up Module. During warming up, an Isolation Forest (iForest)[16] classification model \( \Theta_w \) is trained. iForest is an unsupervised algorithm which will return anomaly scores for input instances based on \( \Theta_w \).

A fraction of instances are used for replay memory initialization. Considering the diversity of replay memory, we select amount of \( N_u \) instances with highest anomaly scores, \( N_w \) instances with lowest anomaly scores and \( N_{aw} \) instances with anomaly scores closed to 0.5. In Label Propagation module, a label propagation model \( \Theta_{lp} \) will be trained by these instances. It propagate labels of these instances and generate pseudo labels for next stage. Then, we start training reinforcement learning model \( \Theta_{rl} \) based on the true labels and pseudo labels. The Active Learning module will receive the amount of \( N_{al} \) selected instances by model \( \Theta_{rl} \) based on prediction uncertainty and ask human expert for manually labeling. Again, the new labeled instances are used for label propagation and then throw to training data pool for next iteration training.

### 5 METHODOLOGY

In this section, we describe each component of our approach in detail. We first introduce the warm up algorithm, followed by the structure of deep reinforcement learning network, label propagation and active learning.

#### 5.1 Preprossesing and Warm up

Given a time series sequence \( TS = [t_1, t_2, \ldots, t_n] \), we separate it into a set of states:
\[ S = [(t_1, t_2, \ldots, t_0), (t_3, t_4, \ldots, t_{n-1}), \ldots, (t_{n-\omega}, t_{n-\omega+1}, \ldots, t_n)] \]
through a sliding window with size of \( \omega \). Each state(sample) has size of \( \omega \) and the label of state \( S_i \) is determined by the last time point \( t_n \), where 0 represents non-anomaly, 1 represents anomaly and -1 marked as unlabeled. All values are normalized into [0,1] by the MaxMinScaler[32].

Replay memory is a key component of deep reinforcement learning. RL agent randomly select a mini batch from replay memory for loss minimization, which will reduce the correlation between samples. However, the normal replay memory has normal distribution and lacks of data diversity. We build a isolation Forest classification model based on sklearn[32] to take a tentative outlier detection. iForest helps us select the most representative anomaly samples and normal samples. We send these samples to RL agent and help it initiate replay memory.

#### 5.2 Deep reinforcement learning

For anomaly detection problem, we consider the instance as the state. Action is the prediction of RL agent for whether it is an anomaly or not. \( a = 1 \) represents anomaly prediction and \( a = 0 \) for non-anomaly prediction. Reward function is set as \( (r_1, r_2, -r_1, -r_2) \) for true positives, true negatives, false negatives and false positives.

We adopted a Long Short Term Memory (LSTM) neural network EvalSy as the brain of RL agent to generate Q-value when a state \( S \) is received, which follows \( Q \) function. We build another neural network TargetSy which has the exactly same architecture with EvalSy. The later network has lower update rate than the former and we could consider TargetSy always fixes its parameters. For exploration and exploitation trade-off, an epsilon decay strategy is adopted. The greedy factor \( \epsilon \) is used to decide whether our RL agent should take actions based on \( Q \) function or randomly take actions to explore the entire environment space. The epsilon decay policy tries to decrease the percentage dedicated for exploration as time goes by. This can give optimal regret. [29]

At the end of the Warm up process, the initial reinforcement learning model \( \Theta_{rl} \) is trained. As shown in Algorithm 1, for each iteration, DQN receives the set of states \( S \) and picks up the labeled ones for training reinforcement learning model \( \Theta_{rl} \). The unlabeled instances will be sent to the Active Learning component and Label propagation component which will return a set of new labeled instances for next iteration’s training. The transitions will be stored in replay memory. In each epoch, a mini batch of replay memory is
sampled and the parameters in $E_{val_N}$ will be copied to $Target_N$ for every $r$ epochs. A gradient descent is performed for model updating as shown in Figure 1.

5.3 Active learning

Since fully labeled data is expensive in the real world, we introduced active learning component into our system. It gives our RL agent the capability to not only explore the environment and learn experience, but also ask questions/queries based on its experience during the exploration.

We choose margin sampling as our active learning strategy. The smaller means our model is more uncertain to identify whether a sample is anomaly or non-anomaly. For each episode, active learning receives the unlabeled instance set $S_{unlabeled}$ from deep reinforcement learning. In each epoch, for a state $s$, the RL agent has two action options: $a_0$ for non-anomaly and $a_1$ for anomaly. Their $q$ values and minimum margin could be formulated as:

$$q_0, q_1 = W * s + b$$  

(1)

$q$ values reflect the potential reward the RL agent could get when it takes certain action.

$$\text{min}_\text{margin} = \min |q_0 - q_1|$$  

(2)

We calculate the margin by Equation 2 and sort them in descending order. Amount of $N_{al}$ with the smallest $\text{min}_\text{margin}$ will be inspected by a human expert. We assume the expert does not make mistakes and the labels are all correct. The new labeled samples will be sent to propagation and finally thrown into sample pool for next iteration’s training. Since this process needs human effort, we count these labels to the total amount labels we used.

We adopted label propagation algorithm to make fully use of labels inspected by human. Label propagation is a semi-supervised algorithm to propagate labels through the dataset along high-density areas defined by unlabeled data [37].

In our system, the label propagator is trained by data with given labels. When a new instance in a time-series is labeled manually, the propagator will generate pseudo-labels to the data instances with similar distributions based on the entropies of transduced label distributions with probability of $pk$:

$$S = - \sum (pk * \log pk)$$  

(3)

Since these label are generated by label propagation model, we consider them as pseudo-labels and do not count them to the total labels we use.

6 EXPERIMENTS

In this section, we introduce our experiments in detail. First, we present the data sets we used and then we evaluate our technique and compare with other anomaly detection algorithms.

6.1 Dataset

We used two datasets for our evaluation. Yahoo Benchmark and KPI. They are commonly used for time-series anomaly detection. We provide an overview in Table 1 and give a more detailed description then.

6.1.1 Yahoo Benchmark. Yahoo published its Webscope data set for time-series anomaly detection. It contains real traffic communication of Yahoo services and synthetic curves. In our experiments, we evaluate on both real time series data set $A1Benchmark$ which shows the Yahoo membership login data and synthetic one $A2Benchmark$ which contains the anomalies of single outliers. $A1Benchmark$ includes 67 curves with labels for each timestamp. Each sequence has 1400-1600 data points. $A2Benchmark$ has 100 synthetic time series with 1400-1600 points.

6.1.2 KPI data set. Another one is KPI data set from AIOPs competition. It is collected from several internet companies such as Tencent, ebay and Sogou. There are 3,004,065 data points with timestamps and labels.

6.2 Experiment Setup

We set sliding window size $\omega$ to 25. The LSTM architecture is composed by an input layer, an output layer and a hidden layer. The forget bias is set to 1.0. Size of replay memory $N$ to 1000, the decay ratio of greedy factor $\epsilon$ is $\frac{1}{50000}$, reward constants $r_1 = 5$, $r_2 = 1$, discount factor $\gamma$ is set to 0.8 for Yahoo and 0.98 for KPI. The warm up number $N_{w}$ is set to 5 and query number for active learning $N_{al}$ is in range of $[1,10]$ for each iteration. We split training and test set as 0.8: 0.2 for our data sets. Since the difference of data balance between these datasets, we examined Yahoo dataset with 1, 5 and 10 active queries each episode while 5 and 10 queries for KPI in each episode. During the experiment, RLAD needs 1000 episodes to converge so that we manually labeled 1000(1%), 5000(5%) and 10000(10%) samples during active learning. RLAD converges faster on KPI with 300 episodes so that we used 15000(0.05%) and 30000(0.1%) labels. We evaluated our model on Precision, Recall and F1-score.

6.3 Results

We compared RLAD with state-of-art unsupervised and semi-supervised time series anomaly detection techniques mentioned in related

<table>
<thead>
<tr>
<th></th>
<th>Yahoo A1</th>
<th>Yahoo A2</th>
<th>KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>total points</td>
<td>94866</td>
<td>142100</td>
<td>3004066</td>
</tr>
<tr>
<td>anomalies</td>
<td>1669(1.76%)</td>
<td>400(0.28%)</td>
<td>79554(2.65%)</td>
</tr>
</tbody>
</table>

Table 1: Overview of data sets
work. Note that SR-CNN, DAGMM, Autoencoder and Deep-SAD return anomaly scores instead of binary prediction. We applied Bayesian Optimization algorithm to search the threshold which could reach the highest F1-score. The results on Yahoo dataset are shown in Table 2. It shows that RLAD outperforms all unsupervised approaches on both A1Benchmark and A2Benchmark. Specifically, with only 1% labels, our F1-score is 59% higher than SPOT which performs best over all unsupervised approaches on A1Benchmark and 45% higher than SPOT on A2Benchmark. With 10% labels, RLAD is able to achieve nearly 0.8 F1-score on A1Benchmark and 1.0 on A2Benchmark. When compared with semi-supervised approaches, it is evident that RLAD can achieve much better performance than others when given the same number of labels. As shown in Table 3, RLAD is able to achieve 3x higher f1-score than all of the unsupervised methods on KPI dataset by labeling 0.1% samples.

7 CONCLUSION

In this paper, we introduced RLAD for time series anomaly detection. It is the first attempt on anomaly detection area to combine deep reinforcement learning and active learning to reduce reliability of normal pattern assumption and availability of labels. During experiments, RLAD achieved outstanding performance by labeling a tiny fraction of samples on both real and synthetic dataset. Moreover, we outperform the state-of-art both unsupervised and semi-supervised techniques in our experiments. In the future, we plan to explore the possibility for deploying RLAD on real world and real-time applications.

REFERENCES


<table>
<thead>
<tr>
<th>Techniques</th>
<th>Yahoo A1</th>
<th></th>
<th></th>
<th>Yahoo A2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1-score</td>
<td>precision</td>
<td>recall</td>
<td>F1-score</td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>unsupervised</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luminol</td>
<td>0.177</td>
<td>0.261</td>
<td>0.258</td>
<td>0.277</td>
<td>0.314</td>
<td>0.266</td>
</tr>
<tr>
<td>SR-CNN</td>
<td>0.264</td>
<td>0.174</td>
<td>0.540</td>
<td>0.057</td>
<td>0.048</td>
<td>0.069</td>
</tr>
<tr>
<td>SPOT</td>
<td>0.416</td>
<td>0.513</td>
<td>0.394</td>
<td>0.689</td>
<td>0.53</td>
<td>0.991</td>
</tr>
<tr>
<td>DSPOT</td>
<td>0.423</td>
<td>0.421</td>
<td>0.426</td>
<td>0.525</td>
<td>0.685</td>
<td>0.425</td>
</tr>
<tr>
<td>DAGMM</td>
<td>0.122</td>
<td>0.139</td>
<td>0.109</td>
<td>0.018</td>
<td>0.009</td>
<td>0.490</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>0.026</td>
<td>0.013</td>
<td>0.374</td>
<td>0.381</td>
<td>1.0</td>
<td>0.235</td>
</tr>
<tr>
<td>semi-supervised</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep SAD(1%)</td>
<td>0.042</td>
<td>0.022</td>
<td>0.459</td>
<td>0.112</td>
<td>0.134</td>
<td>0.096</td>
</tr>
<tr>
<td>Deep SAD(5%)</td>
<td>0.048</td>
<td>0.025</td>
<td>0.568</td>
<td>0.14</td>
<td>0.304</td>
<td>0.091</td>
</tr>
<tr>
<td>Deep SAD(10%)</td>
<td>0.047</td>
<td>0.025</td>
<td>0.564</td>
<td>0.11</td>
<td>0.376</td>
<td>0.064</td>
</tr>
<tr>
<td>RLAD(1%)</td>
<td>0.708</td>
<td>0.652</td>
<td>0.781</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>RLAD(5%)</td>
<td>0.752</td>
<td>0.71</td>
<td>0.8</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>RLAD(10%)</td>
<td>0.797</td>
<td>0.733</td>
<td>0.922</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 2: Comparison between RLAD with other approaches on Yahoo dataset

<table>
<thead>
<tr>
<th>Techniques</th>
<th>KPI F1-score</th>
<th>precision</th>
<th>recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsupervised</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luminol</td>
<td>0.032</td>
<td>0.111</td>
<td>0.063</td>
</tr>
<tr>
<td>SR-CNN</td>
<td>0.166</td>
<td>0.195</td>
<td>0.145</td>
</tr>
<tr>
<td>SPOT</td>
<td>0.033</td>
<td>0.543</td>
<td>0.017</td>
</tr>
<tr>
<td>DSPOT</td>
<td>0.049</td>
<td>0.023</td>
<td>0.911</td>
</tr>
<tr>
<td>DAGMM</td>
<td>0.177</td>
<td>0.198</td>
<td>0.16</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>0.17</td>
<td>0.094</td>
<td>0.858</td>
</tr>
<tr>
<td>semi-supervised</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep-SAD(0.05%)</td>
<td>0.088</td>
<td>0.046</td>
<td>0.863</td>
</tr>
<tr>
<td>Deep-SAD(0.1%)</td>
<td>0.128</td>
<td>0.055</td>
<td>0.653</td>
</tr>
<tr>
<td>RLAD(0.05%)</td>
<td>0.709</td>
<td>0.681</td>
<td>0.897</td>
</tr>
<tr>
<td>RLAD(0.1%)</td>
<td>0.778</td>
<td>0.827</td>
<td>0.879</td>
</tr>
</tbody>
</table>

Table 3: Comparison between RLAD with other approaches on KPI dataset

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) 6, 1 (2012), 1–39.


