EAD: An Efficient Anomaly Detection Algorithm for Multivariate Time Series*

Dehong Ma[1], Second Author[2,3][1111−2222−3333−4444], and Third Author[3][2222−−3333−4444−5555]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
madehong12@nudt.edu.cn
http://www.springer.com/gp/computer-science/lncs
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
{abc,lncs}@uni-heidelberg.de

Abstract. Anomaly detection based on deep learning has been widely used in IT infrastructure management. Driven by a large amount of data, deep learning (DL) based algorithms can achieve higher accuracy rate compared to rule-based algorithms. However, the computational complexity of deep learning algorithms is much higher than traditional rule-based ones, which will cost lots of time and computing resources. This limits the application of such algorithms in actual systems. Therefore, it is necessary to improve the detection efficiency of existing DL-based algorithms.

In this work, we propose an efficient detection approach to solve the occupation of computing resources by combining DL-based algorithms and rule-based algorithms. Specifically, the existing DL-based algorithm feeds all the data to the neural network for detection, which can achieve a higher rate, but consumes computing resources in turn. Actually, In anomaly detection scenarios, anomalous data only accounts for a small portion. Therefore, it is not necessary to hand all the data to the neural network for detection. Thus, we apply rule-based algorithms to filter the original data roughly and then exploit the DL-based algorithms to make the final decision, thereby greatly reducing the amount of calculation without reducing accuracy.

We evaluate our approach using a state-of-the-art deep learning anomaly detection architecture with three real-world datasets. The results show that the proposed approach can significantly improve detection efficiency, saving 60 percent of time. Keywords- anomaly detection, detection efficiency.

Keywords: Anomaly detection · Detection efficiency.

1 Introduction

DL-based anomaly detection has been widely applied in many industrial fields, such as spacecrafts control, server management, water resources dispatch, etc.

* Supported by nudt.

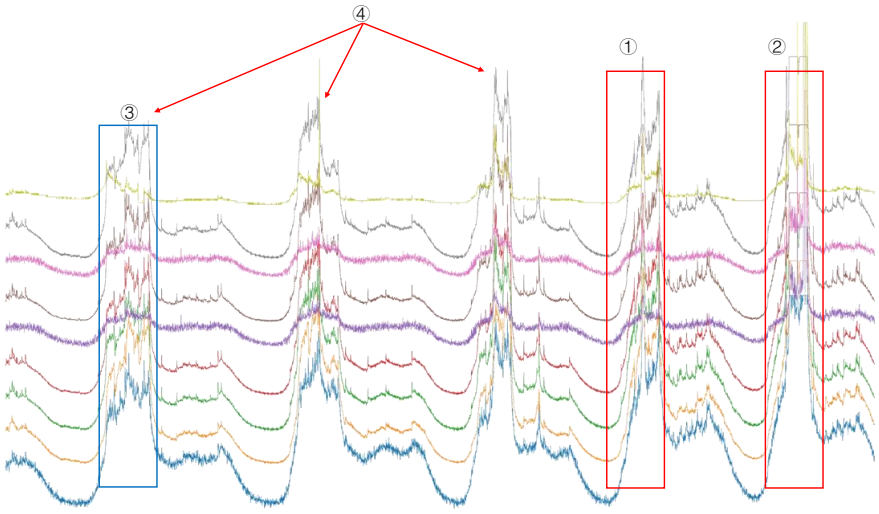

Fig. 1. Figure 1 A fragment in the SMD[1] dataset, which is a typical dataset of Anomaly detection. Anomalies are marked by red boxes①and②. It can be seen from the blue box③ that there are peaks in data of different dimensions at the same time, showing the correlation of the data. We can see the seasonality of the time series by marking ④. There are also noises in the data.

Compared with traditional rule-based algorithms, DL-based algorithms have many significant advantages. Firstly, DL-based algorithms need less dependency on the operators' experience, as rule-based algorithms require manually designed rules, which are labor intensive. Secondly, it's difficult or even impossible to design rules for complex input data(Figure 1). Thirdly, modern information systems update quickly, leading to frequent rule design tasks, which makes it quite difficult to apply these algorithms to actual systems. While the DL-based algorithms solve the problems above perfectly, and many researches have achieved pretty high accuracy in their specific fields . However, some other problems have also arisen, most significant of which is the detection efficiency. In practical applications, the anomaly detection algorithms must be efficient enough to detect faults in time to reduce losses caused by them. Some public data sets demonstrate that the monitoring system usually generate millions, or even billions of time series in real time(Figure 1). If all of this data is fed to the neural network, a large amount of additional computing load will be generated, which will not only increase the burden on the machine, but also misses the timing of detection, causing greater losses. Let's consider the following two scenarios:

Scenario one: The anomaly detection system is deployed in the server of a large-scale data center to monitor the real-time status of the server to ensure that the server can provide safe and reliable services. In this scenario, precious computing resources should be used as much as possible to provide services to increase revenue. Frequent invocation of a computationally intensive detection

system is obviously not cost-effective. Scenario two: The anomaly detection sys-

Table 1. Basic information of some typical data sets

Dataset	Main indicators	Data dimension	Abnormal rate(%)	Training set size
Server machine[]	CPU load, network usage, memory usage, etc.	38	4.16	708405
Soil Moisture Active Passive satellite[]	Telemetry data: radiation, temperature,power, computational activities, etc.	25	13.13	135183
Mars Science Laboratory rover[]		55	10.72	58317
Secure Water Treatment[]	Sensor data and actuaor data of SWaT	51	11.98	496800
Water Distribution[]	Network traffic, sensor and actuator data of WADI	103	5.99	1048571

tem is deployed in the embedded computer of the spacecraft to detect whether the data of the various sensors is abnormal, and to assist the ground controller to regulate and control the aircraft in time. As we all know, aircrafts need to be as light as possible, and electrical energy is very precious in space. Therefore, the amount of calculation for anomaly detection should be reduced as much as possible to reduce the launch costs.

From the above two scenarios, in order to deploy the learning-based anomaly detection algorithms to practical applications, it is necessary to improve its detection efficiency. To solve the above problems, we start with analyzing of the input data first. According to our observation, the input data usually meets the following characteristics

Characteristic one: The statistical results show that in most data sets, the proportion of abnormal data is very small(Table 1), which means that most of the monitoring is normal. This is in line with our intuition, the system is running normally most of the time, and the anomaly is a small probability event.

Characteristic two: The monitored machine usually serves humans. Many monitoring data are seasonal data following the humans work and rest. Therefore, it is almost impossible to find static thresholds for these data.

Characteristic three: Due to the variability of the work environment, monitoring data usually contain noise in real word applications.

Characteristic four: Anomalies mostly occur in locations where the data changes drastically.

Considering the above characteristics, we propose an approach that combines learning-based algorithms and rule-based methods. Specifically, we divide the detection process into two steps. Step one, a rule-based algorithm is applied to the data for preliminary detection first. In this process, most of the apparently normal data will be filtered out efficiently. Since this step only needs to execute a few judgment codes, efficiency has been greatly improved compared with DL-

based algorithms. Then, in step two, we apply the DL-based algorithms to further detect the suspicious data remained in the previous step. We submit the result of step 2 as the final test result. Although the amount of calculation based on the deep learning algorithm is relatively large, due to our input data is relatively small, so the execution time is very short. In this way, we reduce the amount of calculation in the anomaly detection process by reducing the number of calls to complex neural networks. There are two major challenges for this detection architecture.

The first challenge is how to design universal rules which can adapt to different application scenarios. As mentioned above, designing anomaly detection rules for complex data is labor intensive. Therefore, we need to find a rule that can be applied to different scenarios without human intervention. Since step one requires a rough detection of the data, we don't need to pursue a higher accuracy rate. we only need to filter out the data that is determined to be normal. Taking into account the characteristics of seasonal changes in the data, we apply the exponentially weighted moving average (EWMA) to record overall trend of the data. Then we calculate the variance of the data, and determine the threshold of the data according to the 3-σ principle. Compared to using a fixed average as a benchmark, this algorithm is more adaptable to seasonal data. At the same time, EWMA has low computational complexity and low memory usage, making it very suitable for pre-detection of the monitoring data .

The second challenge is how detect anomalies in multi-dimensional time series, which are noisy, seasonal and correlate. Thanks to the sufficient research in this field, there are a large number of excellent detection algorithms that can be applied. We can select the appropriate DL-based algorithm according to the different application scenarios. In this paper, we mainly refer to the work of [] and apply it as the second step of the detection algorithm.

We summarize our main contributions as below.

(1) We propose a detection architecture that combines rule-based algorithms and DL-based algorithms. This structure can take full advantage of the two types of algorithms, ensuring high execution efficiency and detection accuracy. What's more, this two-stage detection architecture can be easily applied to most of the current DL-based detection algorithms to improve the detection efficiency.

(2) We have carefully selected a rule-based algorithm and a DL-based algorithm, whose combination can achieve higher performance without modifying the hyperparameters.

(3) We feed the data instance one by one to the detector rather than batch input like other methods, which is more similar to the actual testing environment.

(4)We have integrated and packaged the code into docker, which is convenient for everyone to discuss and exchange.[†]

[†] https://github.com/NetManAIOps/label-tool

2 Related Work

Anomaly detection is a hot topic that has been widely studied for many years. At early stages, due to the relatively small amount of data, the rule-based method was widely used. One of the most common methods is the 3-σ principle, which assumes that the data samples obey independently identically Gaussian distribution, and regards samples with a variance of 3 times higher or lower than the mean value as anomalies. Furthermore, researchers proposed distance-based and probability-based algorithms based on the extraction of data features. This type of algorithms can utilize the advantages of big data, compared with rule-based ones, performance has been improved to a certain extent. Opprentice is the classic work of applying machine learning algorithms to anomaly detection, which converts the anomaly detection problem into a supervised classification problem. The workflow is as follows, the staff label the historical data through the labeling tool, and the labeled data is trained with random forest, and then they use the trained model to detect the new data. Through supervised learning, the accuracy of this method can be very high, but the problem of this method is also very obvious. On the one hand, labeling large-scale monitoring data is not cost-effective. On the other hand, for data sets with unbalanced positive and negative samples, the accuracy of the classification model will be reduced. While in the current industrial field, the most common monitoring data is MTS, and the Recurrent Neural Network - Variational AutoEncoder (RNN-VAE) architecture is the state-of-the-art method for MTS anomaly detection.The core idea of these algorithm is to capture the normal patterns of MTS through RNN with stochastic variable. Since the RNN can capture the context of time series, and the stochastic variable can increase the robustness of the model, these algorithms achieve high performance in the field of anomaly detection for MTS. There are many researches using this architecture or its variants. [] applies LSTM []-VAE model to achieve anomaly detector task for robot-assisted feeding [] . [] applies bi-LSTM and attention mechanism to better capture contextual features. OmniAnomaly [] further applies techniques such as stochastic variable connection, planar NF to model the temporal dependence between stochastic variables better. RNN-VAE is a series of unsupervised algorithm, and it does not need labeled data for training, which is an important advantage. However, this architecture includes twice neural network calls—an encoding network and a decoding network, and the dimensionality of the input data is usually high, so the amount of calculation is large, which limits its application in actual scenarios. The main work of this paper is to improve the detection efficiency.

3 Preliminaries

In this section, we present the problem statement of anomaly detection for MTS in detail, and the necessary components of our EAD.

3.1 Problem Description

In modern software monitoring systems, MTS is one of the most common data types. Take the server machine as an example, each machine is monitored at a fixed period with many key performance indicators (KPIs) (e.g., CPU, memory, and network traffic, etc.). Each KPI can be formulated as a univariate time series, and thus, each machine can be represented as an MTS . For more convenient expression, we formalize MTS as follows. At time t, we want to detect

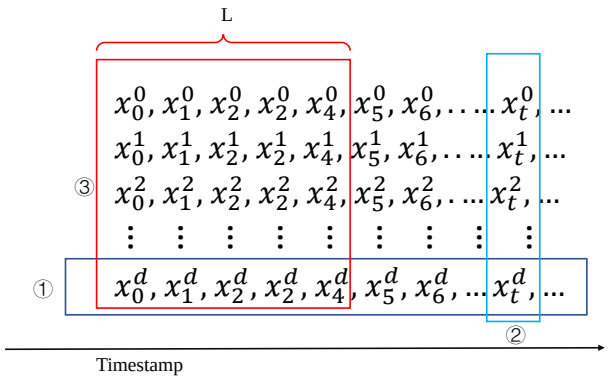

Fig. 2. As shown in the figure,x_t^d is the value of the d dimension at time t. The values in box 1 are the sequence formed by NO. d indicator, expressed as $\boldsymbol{x^d}$. The values in box 2 are all the indicators values at time t, expressed as $\boldsymbol{x_t}$. All the values in box 3 form a data instance, which is the sample input to the neural network, denoted as $\boldsymbol{X_t}$. L is the length of the sliding window.

whether the current observation $\boldsymbol{x_t}$ is anomalous or not. To achieve this, a piece of previous data needs to be considered, which will be specifically introduced in the next section.

3.2 EWMA

Since most of the monitoring data is seasonal data (figure), we cannot use the average value as the benchmark for detection. EWMA is a common method for processing sequence data. For univariate time series $\boldsymbol{x(t)}$, its calculation formula is as follows.

$$EWMA(t) = \beta \boldsymbol{x(t)} + (1-\beta)EWMA(t-1) \tag{1}$$

Where the weight β controls the smoothness of EWMA(t) EWMA can describe the overall trend of the sequence, and anomalies usually occur in locations where the data is violently jittered. Therefore, the value that obviously deviates from the EWMA is the most likely to be outliers. In the process of calculating EWMA, only one hyperparameter β needs to be set, thus we don't need to adjust the hyperparameter for different input data, making this algorithm can meet various application scenarios.

3.3 RNN-VAE Architecture

Time series data have contextual relations, to determine whether the current data is abnormal, the previous data needs to be considered, thus it is better to use RNN and its variants to process it. Through adopting deterministic hidden variables, RNN can better model time series. LSTM [] and GRU [] are important variants of RNN, which can better learn the long-term dependence of sequence by using gating mechanisms . Therefore, more and more works use GRU or LSTM to model time series.

VAE is a deep Bayesian model [], DOUNUT [] has applied it to solve seasonal univariate time series detection task successfully.VAE maps a high-dimensional input x_t to a latent representation z_t with a reduced dimension, and then reconstructs x_t by z_t. These two processes are represented by two neural networks $p_\theta(\mathbf{x_t} \mid \mathbf{z_t})$ and $q_\phi(\mathbf{z_t} \mid \mathbf{x_t})$ with parameters θ and ϕ.Stochastic Gradient Variational Bayes (SGVB) [7] is a variational inference algorithm often used in VAE to train the parameters and by maximizing the evidence of lower bound (ELBO), $\mathcal{L}(x_t)$:

$$\begin{aligned}\mathcal{L}(\mathbf{x_t}) &= \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x_t})}\left[\log\left(p_\theta(\mathbf{x_t} \mid \mathbf{z_t})\right)\right] - D_{KL}\left[q_\phi(\mathbf{z_t} \mid \mathbf{x_t}) \,||\, p_\theta(\mathbf{z_t})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x_t})}\left[\log\left(p_\theta(\mathbf{x_t} \mid \mathbf{z_t})\right) + \log\left(p_\theta(\mathbf{z_t})\right) - \log\left(q_\phi(\mathbf{z_t} \mid \mathbf{x_t})\right)\right]\end{aligned} \tag{2}$$

Monte Carlo integration [] can be used to compute the above expectation, as shown in Eq. (3), where t,l = 1,2...L is sampled from the

$$\mathcal{L}(\mathbf{x_t}) \approx \frac{1}{L}\sum_{l=1}^{L}\left[\log\left(p_\theta\left(\mathbf{x_t} \mid \mathbf{z_t}^{(l)}\right)\right) + \log\left(p_\theta\left(\mathbf{z_t}^{(l)}\right)\right) - \log\left(q_\phi\left(\mathbf{z_t}^{(l)} \mid \mathbf{x_t}\right)\right)\right] \tag{3}$$

The advantage of VAE is that it is an unsupervised learning algorithm that does not require labeled data. In the training phase, it only needs enough normal data, which is easy to satisfy in actual scenarios. The training process essentially stores the characteristics of the training data in the neural network. In the detection stage, we compare whether the current input data sample is similar to the characteristics of the training data. If not, we consider the current data to be abnormal data.

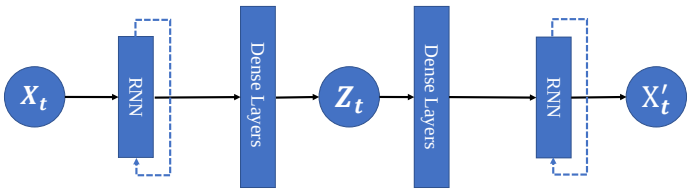

Fig. 3. RNN-VAE architecture

RNN-VAE architecture (Figure 3) can give play to the advantages of the both, so it is widely used in the field of anomaly detection for MTS. Its structure

is shown in the figure. Specifically, the architecture uses RNN as an encoding network and a decoding network. After the RNN network, a fully connected layer will be added to obtain the data of the corresponding dimension.

4 Design

In this section, we introduce in detail the workflow of EAD.

4.1 Overall Structure

The ADC workflow mainly includes the following four processes, the overall structure is shown in the Figure 4, and the following will be introduced in turn.

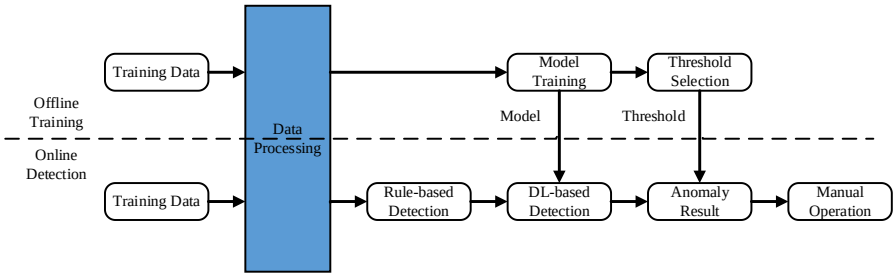

Fig. 4. The workflow of EAD

4.2 Data preprocessing

The original monitoring data needs to be preprocessed before it can be input to the neural network. The process mainly includes the following steps. Filling in missing data. In real application scenarios, monitoring data may be missing due to reasons such as data collection and network transmission. When a value is missing, we use its previously observed value to fill in its position . In addition, we can also fill with random values or fixed values (e.g., 0), but generally we don't use the median to fill in. This is because the missing value itself may be caused by a system failure. Filling it with the median is equivalent to smoothing the series, leading to false negatives.

Data normalization. Since the values of different dimensions of MTS have different scales, we must normalize the data before feeding it to the neural networks to ensure it performing better. Commonly used normalization methods include Z-score normalization and Max-Min normalization.

$$x^* = \frac{x - x_{\text{mean}}}{x_{\text{std}}} \tag{4}$$

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{5}$$

Data Slice. In order to provide previous information, we need to input the previous data together with the current data into the neural network. We divide the data into data instances through a sliding window. The length of the sliding window L should be appropriate, neither too long nor too short. A sliding window that is too long will increase the amount of calculation, while too short will not be able to capture the characteristics of outliers.

4.3 Offline Training

The purpose of the offline training phase is to train the model based on the data that has been obtained so far. The research in this area has been very sufficient, so we refer to the model architecture of Omninly, which is a state-of-the-art method in this field. The mean and variance of each dimension of the training data should also be calculated for use in the detection stage.

4.4 Online detection

The online detection stage is divided into two parts, namely rule-based detection and learning-based detection. Rule-based detection will compare the value of all dimensions at the current moment with its corresponding benchmark (EWMA value). The EWMA value is constantly updated with the input of the data, and the variance is the variance of the training set. When the value of any dimension has a large deviation from the benchmark (more than three times the variance), we consider the current data to be suspicious data, and feed sequence end with it to the neural network for further detection. The trained model can calculate the anomaly score of the input data. We judge whether the data is outlier according to the anomaly score. In actual deployment, we provide two design options, as shown in Figure 5.

Option one, Distributed detection. This option is to deploy all the detection system directly on the monitored machine, and then directly submit the detection results to the operation and operator. The advantage of this method is that it can reduce unnecessary network transmissions and we can get the results in the first time. The disadvantage of this method is that it will occupy the computing resources of the monitored machine, reducing its performance to a certain extent. Therefore, this option is more suitable for deployment on a single machine with certain computing power.

Option two, Centralized detection. The option is to collect the data and transmit it to a special machine for detecting. In doing so, EAD's advantages can be further utilized. The rule-based detection process is executed immediately after the monitoring data is generated, and only suspicious data is transmitted to the central server, which can reduce the amount of network transmission. What's more, we can also optimize the central server according to the characteristics of neural network computing, using high-performance GPUs to implement neural

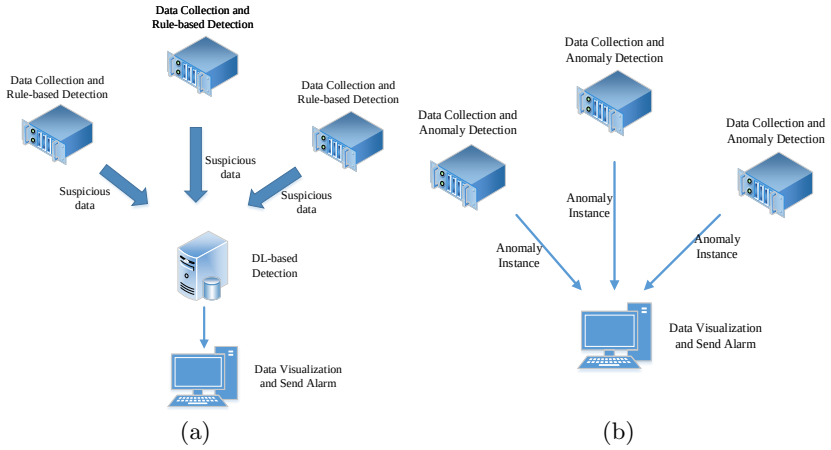

Fig. 5. Two options of deployment,Centralized detection(a),Distributed detection(b)

network calculations, which can significantly improve resource utilization. This option is suitable for deployment in large-scale data centers, where the network transmission is smooth, or in the cloud-edge architecture, where the computing power of edge devices is very limited, not enough to run deep learning algorithms.

5 Evaluation

In this section, we introduce the experimental settings, the data sets used and performance evaluation indicators.

5.1 Datasets and Performance Metrics

To demonstrate the effectiveness of EAD, we conduct experiments on three datasets: SMD (Server Machine Dataset), SMAP (Soil Moisture Active Passive satellite) and MSL (Mars Science Laboratory rover) .We use two indicators to evaluate the performance of EAD, namely accuracy rate and execution time. For the accuracy rate, we measure it more specifically through F1-score, Precision, and Recall. [] and [] give methods to automatically determine the threshold using extreme value theory [], but in practice, using a fixed threshold does not work well. Too high a threshold will cause the detector to detect very few abnormalities and the staff has nothing to do, while a too low threshold will cause too many alarms, which the staff cannot deal with one by one. This ultimately results in the anomaly detection system being unable to provide useful information to the application, reducing its usefulness. Refer to the work of 3, we directly feedback the abnormal scores to the staff, and the staff will check the abnormal scores in order, which can greatly improve the work efficiency as high as possible. Therefore, in this experiment, we evaluate the effect by comparing f,

that is, taking the threshold which can get the highest f. In addition, we noticed that for an abnormal segment, if we can detect an abnormality in any of its subsets, the staff can solve the fault in time. Therefore, in the evaluation stage, similar to the method of [], we make a little adjustment to the prediction result according to the label and then calculate the F1-score.

In the past works, in order to conduct experiments efficiently, people usually feed the test data into the neural network in batches in the testing phase,which does not match the actual scenarios. Specifically, for the same machine, the data is generated vector (a vector represents each data indicators at the same time) by vector.Therefore, in the detection process, the data is also input to the neural network instance by instance, whose efficiency is much slower than batch input.Therefore, in order to better meet the actual scenario, although it's more inefficient, we still feed the data one by one to test the running time.

5.2 Result and Analysis

Accuracy rate. To demonstrate the performance of EAD, we take a state-of-the-art unsupervised approach OmniAnomaly [] as baseline. The precision, recall and F1 of EAD and OmniAnomaly on the three datasets are shown in Table 2. In the SMD dataset and MSL dataset, the F1-score of our approach is slightly higher than the baseline, and slightly lower in the SMAP dataset. Overall, our method can basically guarantee that the accuracy rate will not decrease compare to the baseline, that is, adding a rule-based algorithm will almost not reduce the accuracy of detection.

Table 2. Accuracy of EAD and the baseline approach

Methods	SMD			SMAP			MSL		
	P	R	F1	P	R	F1	P	R	F1
OmniAnomaly[1]	0.8311	0.9688	0.8352	0.6670	**0.9743**	**0.7919**	**0.9205**	0.8542	0.8861
EAD	**0.8360**	**0.9701**	**0.8662**	**0.9313**	0.5604	0.6997	0.9161	**0.8647**	**0.8896**

Execution time. Further, we compared the execution time per timestamp of the two approaches. In order to get closer to the actual scenario, we feed the data into the detector instance by instance instead of by batch as mentioned in the in section 4.1. The results are shown in Figure 6. It can be seen that the execution time of EAD on the three data sets is significantly lower than the baseline, which greatly improves the detection efficiency. Specifically, in the SMAP data set, the execution time of our approach is only 5.44% of the baseline,efficiency improvement is the most obvious. The execution time on the other two data sets is also reduced to 19.44% and 20.11%, respectively, and the efficiency improvement is also very obvious.

Analyzing the detection process, our method can quickly filter out about 80%∼95% (Table 3) of the data in the first stage just as we expected. As we have introduced in Section 2, most of the monitoring data are normal data,

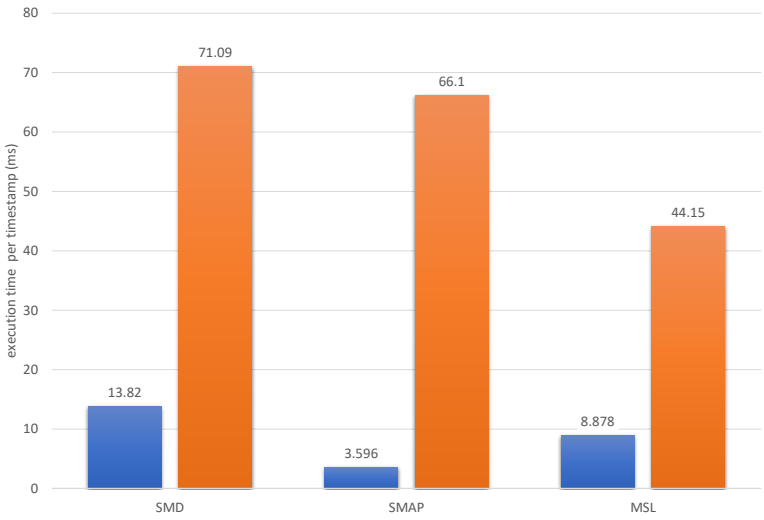

Fig. 6. Execution time of EAD (the blue blockes) and the baseline approach(the orange blockes)

therefore, it is unnecessary to apply the DL-based algorithms to detect all the data. According to this characteristic, our improved method can greatly reduce the time cost and calculation cost. In summary, our approach can significantly improve the execution efficiency under the premise of ensuring that the accuracy is basically the same, making the anomaly detection algorithms more practical.

Table 3. The ratio of neural network calls

	SMD	SMAP	MSL
Test set size	705648	427536	73630
NO. of NN calls	136426	21461	14823
Ratio of NN calls(%)	19.33	5.02	20.13

6 Discussion

Intuitively, our approach is possible to increase false positives indeed, but since the positives usually appear in the position where the value fluctuates sharply, the rule-based detection process hardly causes false positives, and the experimental results also verify this. In contrast, our approach can sometimes slightly improve F1-scores because it may reduces false positives sometimes. The advantages of using this method are very obvious. Experiments have verified that

our algorithm can significantly reduce the amount of calculation and save about 60%of the time.

7 Conclusion

Efficient and accurate anomaly detection is of great significance to ensure the safe and stable operation of information systems. Although the accuracy of the anomaly detection algorithms that apply complex deep learning has been greatly improved, the efficiency has also been reduced in turn. In this paper, we propose EAD, a novel anomaly detection architecture that can significantly improve detection efficiency. We have verified through experiments that this method can significantly improve the detection efficiency under the premise that the accuracy rate is basically unchanged. What's more, this idea can be applied in various DL-based anomaly detection algorithms to improving efficiency, making them more practical and cost-effective.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, `http://www.springer.com/lncs`. Last accessed 4 Oct 2017