2022 IEEE 15th International Conference on Cloud Computing (CLOUD) | 978-1-6654-8137-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CLOUD55607.2022.00039

Applying Value-Based Deep Reinforcement Learning on KPI Time Series Anomaly Detection

1st Yu Zhang
School of Cyber Science and Technology
Beihang University
Beijing, China
Bigbigzy@buaa.edu.cn

2nd Tianbo Wang*
School of Cyber Science and Technology
Beihang University
Beijing, China
wangtb@buaa.edu.cn

***Abstract*—Time series anomaly detection has become more critical with the rapid development of network technology, especially in cloud monitoring. We focus on applying deep reinforcement learning (DRL) in this question. It is not feasible to simply use the traditional value-based DRL method because DRL cannot accurately capture important time information in time series. Most of the existing methods resort to the RNN mechanism, which in turn brings about the problem of sequence learning. In this paper, we conduct progressive research work on applying value-based DRL in time series anomaly detection. Firstly, because of the poor performance of traditional DQN, we propose an improved DQN-D method, whose performance is improved by 62% compared with DQN. Second, for RNN-based DRL, we propose a method based on improved experience replay pool (DRQN) to make up for the shortcomings of existing work and achieve excellent performance. Finally, we propose a Transformer-based DRL anomaly detection method to verify the effectiveness of the Transformer structure. Experimental results show that our DQN-D can obtain performance close to RNN-based DRL, DRQN and DTQN perform well on the dataset, and all methods are proven effective.**

***Index Terms*—Time series, KPI, Anomaly detection, Deep reinforcement learning, DQN**

I. Introduction

In recent years, with the rapid development of network technology and application services, the challenges encountered by data providers have become more and more difficult. They need to ensure the healthy operation of large-scale and complex systems to meet the high-quality service needs of users. For example, hosting software services on cloud computing platforms is a hot trend to reduce operating costs. However, cloud computing platforms usually include thousands of computing nodes, various container technologies, and microservice applications. Once a component fails, it will affect the service operation and even huge asset losses. Cloud service providers usually deploy anomaly detectors and collect and analyze the key performance indicators(KPI) of platform components for continuous monitoring to ensure the system's regular operation. KPIs are usually time series that are recorded periodically, and their values represent the parameters of the monitored target (such as CPU usage, memory usage, etc.). KPIs out of the normal range often indicate an impending or ongoing failure/abnormal attack.

*Corresponding author

Therefore, KPIs reflect the system's health, and how to efficiently analyze KPIs for anomaly detection is the focus of research. Some supervised learning methods have been proposed [1]–[4]. Since the dataset and real-world data are extremely unbalanced (anomalies rarely occur), many unsupervised learning methods for anomaly detection by learning standard patterns have been proposed [5]–[8]. Deep reinforcement learning (DRL) is one of the hot research directions in recent years, and its learning model is more human-like. Some studies combine reinforcement learning and time series research work [9]–[13], but this is not easy, because this is not a typical problem that DRL is suitable for solving: First, anomalies in time series show strong instability, some anomalies like affected by the impulse function, the KPI value suddenly increases, which has nothing to do with the previous moment. Some appear as long-term changes, with multiple previous temporal states associated with the current abnormal state. Therefore, the problem exhibits a non-Markovian character, requiring information from multiple past states or nothing at all. Second, the actions required by DRL will affect the state and reward obtained in the future, and in this problem, after we evaluate whether a state is normal and give a judgment, it will not affect the state obtained by the model in the future. DRL usually introduces an RNN mechanism to help the algorithmic model capture information from more distant states in the past to deal with the former. At the same time, the latter means controlling the value of crucial discount factors in DRL.

This paper systematically studies the value-based DRL algorithm model to solve the time series anomaly detection. We use various structures and techniques to help DRL models obtain more temporal information from time series to improve performance. Experimental results show that our proposed RNN structure-based DRL model (DRQN) achieves excellent performance by tuning the model and improving the experience replay pool works. At the same time, our proposed DQN-based lightweight DQN-D method also achieves good performance. The contributions of this paper are as follows: First, we use a double-layer frame stitching technique and adjust the structure to improve the performance of existing models, including DQN and DQN with LSTM. Second, we propose a Transformer-based DQN model (DTQN) for time series anomaly detection and test its performance in experi-

2159-6190/22/$31.00 ©2022 IEEE
DOI 10.1109/CLOUD55607.2022.00039

ments.

The remainder of this article is as follows: In Section II, we enumerate related work and algorithmic models. Section III elaborates on our modeling process and ideas. Section IV presents the relevant experimental results and concludes. In Section V, we summarize our work and outline potential future work.

II. Related Work

We mainly investigate recent research on time series processing, including time series anomaly detection, time series forecasting. We classify those research according to traditional models, unsupervised methods, and supervised methods and separate DRL-related columns as a class.

A. Traditional Method

Most of the early traditional methods are statistical and similarity measurement methods, such as 3-sigma [14], ARIMA [15], KNN based on distance metric [16], LOF based on density metric [17], one-class SVM [18]. These methods have unique advantages, such as low complexity, fast computation speed, strong generality, and no need for large-scale training data. However, their accuracy is mediocre, and many have strong assumptions (such as the data satisfying a normal distribution), which do not meet this requirement in practical scenarios, so they cannot meet the needs of the real world.

B. Supervised Method

Deep supervised learning methods mainly learn classification models and prediction functions from labeled training data to identify anomalies. EDGAS [1] and Opprentice [2] are typical examples. The former is Yahoo's open-source and scalable general anomaly detection system, including a time series construction module and an anomaly detection module. Each module contains a variety of algorithms and rules, which can detect outliers, fluctuation points, and anomalous sequences. The latter uses machine learning algorithms to extract features and train to generate random forest-based classification models. Its advantage is that it can automatically select the appropriate detector parameters and thresholds, significantly improving the operator's work efficiency. It is worth noting that these methods usually rely on many manually annotated datasets covering various anomalies, which are extremely difficult to achieve in practical scenarios.

C. Unsupervised Method

In practical scenarios, abnormal data and standard data are usually highly unbalanced (there are very few abnormal data), and at the same time, the cost of manually annotating abnormal labels is also prohibitive. Therefore, more and more research on anomaly detection is based on deep learning generative models. Many works are based on VAE [7], [8], [19], such as donut [5] and bagel [6] methods, and some GAN-based methods, such as MAD [20]. The main idea of these unsupervised methods is to learn standard behavioral models from routine data and, at detection time, through model reconstruction to identify anomalies that do not conform to standard patterns. They do not require human-labeled data, but only a large amount of data conforming to standard behavioral patterns, bringing unparalleled advantages. Nevertheless, they all require a long training time and face the problem of choosing an appropriate reconstruction error threshold.

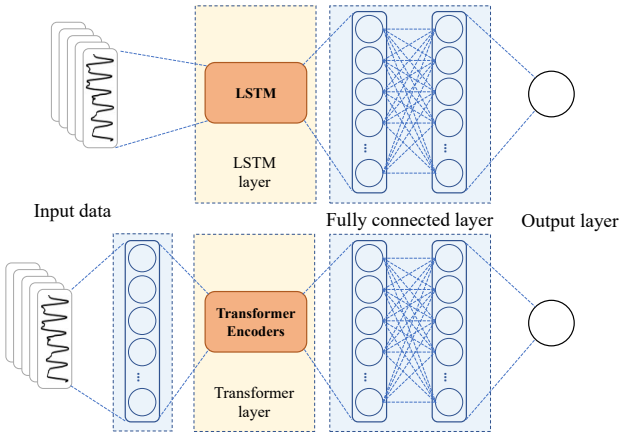

Fig. 1. Schematic diagram of DRQN and DTQN structures. Most of their network structures are simple fully connected networks.

D. Reinforcement learning based Method

Recently, many research works have been using reinforcement learning to process time series, mainly forecasting and classification. [21] uses DRL to find the best strategy for profit in the stock game, and [22] uses DRL to process electrocardiogram time series and predict the occurrence of epilepsy. In the field of time series detection, [9] introduced the policy-based DRL method to deal with the problem of time series anomaly detection. In contrast, [11], [13] introduced the value-based method to solve the problem, and [23] tried a novel IRL method to work.

III. METHODOLOGY

In the following, we describe the specific technical details of our approach, including the modeling approach and modifications to existing models

A. Proposed Model

Traditional Q-learning methods are often unable to deal with complex real-world problems because these problems often bring the curse of dimensionality to Q-learning. Therefore, DQN uses the idea of value function approximation and neural networks to help express the Q function so that q-learning can be applied to more real-world problems. The DQN model accepts the current environment state s_t and returns the Q value corresponding to each possible action, selects the action a_t to interact with the environment through the epsilon greedy policy, obtains the next state s_{t+1} and the reward r_t, and stores its composition entry $< s_t, a_t, r_t, s_{t+1} >$ in the experience

replay pool. The most critical value function iteration equation in Q-learning is as follows:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \quad (1)$$

DQN uses an ingenious dual network structure to solve the network training problem through the loss function formula:

$$L(w) = \mathbb{E}[(r + \gamma \max_{a'} Q(s',a',w) - Q(s,a,w))^2] \quad (2)$$

The term $r + \gamma \max_{a'} Q(s',a',w)$ in formula 2 is the *target*. Taking the Q value in the target network as a "label," let the updated network's Q value approach the target network's Q value and exchange network parameters after the set number of rounds to complete the training. DQN usually works better with the help of frame stitching tricks, but in this problem, this way also does not perform well, so people usually resort to more complex RNN structures. We find that DQN still has the potential to be explored, and we propose a double-layer frame stitching technique to be applied to DQN (we call it the DQN-D method), and the details will be given in Section III-B. In experiments, we compare the performance of DQN with DQN-D.

Recurrent Neural Networks (RNN) are typically models used to process sequence data. Adding the RNN structure to the DQN can help the agent capture more temporal information on time series, which is a more advanced practice in time series anomaly detection. It also trains the network based on equation 2. It is worth noting that, unlike traditional DQN, the input to DRQN is usually a sequence of states rather than a single state. We consider adding LSTM to the structure of DQN, excellent work is done by [13], but there are still some shortcomings. Specifically, [13] uses an additional binary tree strategy to add s^0_{t+1}, s^1_{t+1} obtained by performing different actions on the state s_t to the experience replay pool . It sets four different rewards according to environmental feedback: TP, TN, FP, and FN. We simplify these practices. At the same time, [13] does not have much improvement on the original DQN replay pool when storing data in the pool and extracting data from the pool and still adopts the method of simple storage and random sampling. Because of the existence of the LSTM mechanism in the network, such a data extraction method may not conform to the serialized learning method. We modified the storage method of the pool to be serialized storage and applied the Bootstrapped Random Update method [24], that is, randomly extracting segments from episodes for an update(shows in Section III-B). We call our method DRQN and verify its performance in experiments.

Transformer has achieved great success in natural language processing (NLP) tasks. The idea behind it is to use self-attention mechanism and position encoding to replace RNN. It includes two modules: encoder and decoder. Transformer has more advantages in terms of long-distance feature capture ability, parallel computing ability, and computing efficiency. Inspired by this, we want to join Transfomer to help DQN improve its performance. However, Transformer is designed to solve the seq2seq problem, and our requirement is to let it help DQN extract more meaningful features, so we only use the encoder module in the transformer. At the same time, we do not retain the positional encoding (we don't care about location information). Figure 1 shows a schematic diagram of the structure.

B. Implementation

State and data preparation. When interacting with the environment to explore and collect learning samples, DRL algorithms usually act on actual or simulated environments in real-time. In our problem, there seems to be no such specific environment. We abstractly simulate an environment with the help of actual labels: every action the agent makes will be compared to the labels and rewarded accordingly. Our state cannot simply be set to a single value because it carries too little information in a single-dimensional KPI. The usual practice is to draw on the skill of frame splicing and use a sliding window to splice multiple adjacent time points together as a state. We take a time series file as an episode; In the DQN-D experiments, we use double-layer frame stitching to help the agent obtain more temporal information. Specifically, double-layer frame stitching uses sliding windows (first stitching) to stitch adjacent windows together again as a state (secondary stitching).

Action and reward. Our goal is that for a given state s_t, our agent can determine whether there is an abnormality in this state s_t without any other actions for suspicious or uncertain alerts. Therefore, we set the actions to 0 and 1 to indicate no abnormality in this state and this state is abnormal, respectively. The reward is obtained according to the comparison between the action and the actual label. If the label and the action are inconsistent, a wrong judgment has occurred. The agent will be given a punishment reward of -1. On the contrary, if they are consistent, the agent's judgment is correct. It will be given a positive reward of +1. Our goal is to maximize the reward accumulated by the agent during an episode.

Experience replay pool. Experience replay pool is an essential component in DQN, which can break the solid temporal correlation between observation data, thereby improving the learning performance of the neural network. It stores the data obtained from each exploration in the pool in the structure of $Transitions = [s_t, a_t, r_t, s_{t+1}]$ (if the Pool is full, it replaces the premature data). In each learning process, a batch of data is extracted from the Pool to update the network according to formula 2. Since the input of the LSTM and Transformer structures is sequence data, the structure of the experience replay pool needs to be changed after we add these structures. At the start of each episode, the Pool will create a new temporary stack. As the sliding window advances, the stack will be filled with $Transitions$ data until the end of the episode. The Pool will put the entire stack into it as a unit. During training, select an episode data from the buffer randomly, and then randomly select a moment in the episode, select several steps to learn. The state of the LSTM hidden

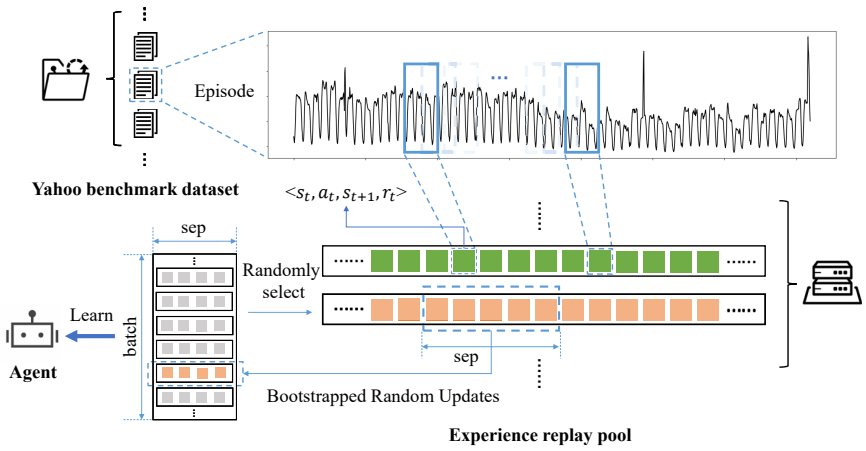

Fig. 2. Schematic diagram of the experience replay pool applying the serialized storage method.

layer will be set to 0 before each training. Figure 2 and Algorithm 1 give some specific details.

Discount factor γ. Unlike the common discount factor γ parameter setting in other DRL problems, in this problem, γ cannot be set too high, so we need to make the agent "short-sighted." As a simple example, if we want the agent to consider the available rewards of T steps in the future at time s_t, we let the reward after T steps account for 0.1 of the Q value at time s_t, that is, $0.1 = \gamma^T$. If roughly want the agent is to consider the following five steps, γ is about 0.6. One of the reasons we need a small γ is that the agent's action does not determine the state it will feedback in the future. In other words, no matter what action a_t the agent makes in the current state s_t, the sliding window is only mechanically forward. Therefore, it is not sensible to consider the reward after the lengthy step.

IV. Experiments

This section introduces our experiments' dataset and some specific parameter settings and gives the relevant experimental results.

A. Dataset and Metrics

We used the *Yahoo benchmark dataset*, which is a webscore dataset released by Yahoo for time series anomaly detection for academic research. It contains four categories, of which the A1 benchmark is the real traffic business data from Yahoo, and the other three benchmarks are based on synthetic time series. Each file contains 1400-1600 timestamps, all with detailed labels. A1 benchmark and A2 benchmark were mainly used in our experiments. In anomaly detection, the standard measurement method for evaluating the quality of models is F1-score, which we also choose as our evaluation method. It is worth noting that we use the sliding window to bring the diffusion of abnormality determination rules. For example, if the sliding window length is eight and the step is 1 if the timestamp T_t is an abnormal point, then $< [T_{t-7}, ..., T_t], ..., [T_t, ..., T_{t+7}] > 8$

Algorithm 1: Experience replay buffer

```
Class Experience replay pool:
    Function init(capacity, collection step,
     collection size, sequence length):
        Set pool capacity
        Set sampling collection step and
         collection size
        Set sequence length
        Set seq memory and memory
    Function push(Transtions, done):
        Push Transtions into seq memory
        if Episode is done then
            while len(seq memory) <
             sequence length do
                Push 0 Transtions into seq memory
            Push seq memory into memory
            Clear seq memory
    Function sample():
        Initialize batch
        Randomly select indexes of size
         collection size form memory
        for index in indexes do
            Randomly choose a starting point from
             memory[index]
            for _ to collection step do
                push Transtions form start to
                 start + sequence length into batch
                start ← start + sequence length
        return batch
```

windows should be judged as abnormal. That is to say, as long as the sliding window contains the abnormal timestamp, the sliding window is abnormal. In addition, [5], [6] mentioned that when it comes to detecting continuous abnormal intervals of time series, in actual operation, the operator does not care whether the detection program can alarm point by point. Therefore, we apply two evaluation metrics: a Harsh f-score, where all windows containing anomalies should be detected. In the Revised f-score, an anomaly is detected at any point in the continuous anomaly interval; it is considered that this interval has been fully detected.

B. Experimental Setups and Comparing Methods

In DQN-R, we set the window size to 4 and the sequence length to 4 (i.e., the input size is 16). Its network consists of only two fully connected layers. In DRQN and DTQN, the window size is 8, and the sequence length is also 8. $hiddensize$ of LSTM in DRQN is set to 128. DTQN also has a fully connected layer before the Transformer-encoder module. The number of layers of the encoder in DTQN is three layers, and the $nhead$ parameter is 4. We consider the following algorithms for comparison:

Twitter [25]. A general open-source algorithm based on Seasonal Hybrid ESD (S-H-ESD) has excellent performance on most seasonal time series, and many related studies use it as a benchmark.

EXP [13]. An LSTM-based DQN anomaly detector that uses a binary tree strategy to store multiple action values under a state. It randomly samples data from the entire buffer every time it is updated and sets four kinds of rewards according to the environmental feedback of different actions: TP reward, TN reward, FP reward, FN reward.

The model parameters are set according to recommendations in the paper or instructions in open source repositories.

C. Experimental Results

We compare the performance of Twitter, EXP, DQN, DQN-D, DRQN, DTQN on yahoo A1, A2 benchmark datasets. Twitter Anomaly Detector does not require training. We sampled 10 time series in each dataset as the test part and the rest as the training part.

Table I presents the performance of all methods on the Yahoo dataset, including experimental results using two different evaluation methods. First of all, the traditional DQN basically cannot work, and our proposed DQN-D has a considerable performance improvement of 62% compared with DQN. Its performance is almost close to the EXP algorithm with RNN. This result shows that DQN-D can use more time information without using RNN. In particular, the network structure of DQN-D is far less complicated than that of EXP, and it is only a lightweight and straightforward fully connected network, so it has more advantages in terms of speed and time. Second, our proposed DRQN method achieves the best results among all the results, with a revised F1 score as high as 0.98 on the A1 benchmark, which proves its excellent performance. Finally, DTQN, which we have high hopes for, also works well, with a corrected F1 score close to 0.9 on the A1 benchmark, which is higher than the EXP method. Transformer-based models have achieved dominance in most scenarios, and their potential is enormous. One reason DTQN is not as good as DRQN maybe that anomalies appear more in local locations in this anomaly detection scenario, and DTQN's local information acquisition ability is not as strong as DRQN's. At the same time, the magnitude of the yahoo dataset we use is too small, which may be another critical factor limiting the performance of DTQN. Figure 3 shows the performance of all methods on a test file.

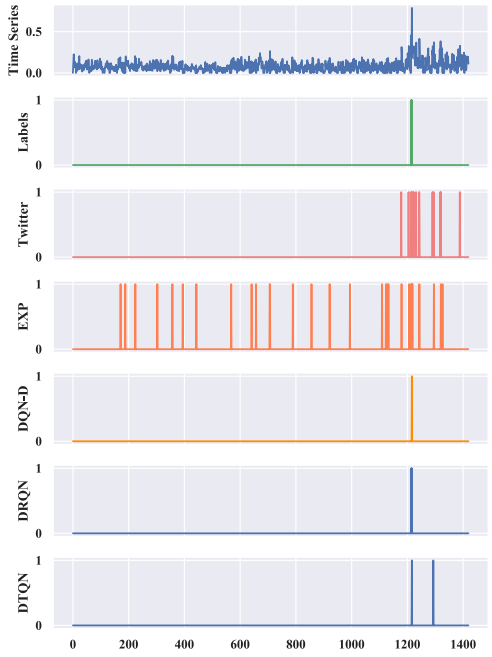

Fig. 3. A figure shows the performance of all methods on a test file. Since the performance of DQN is too poor, it is not drawn.

Further, we also compared the effects of different size discount factors on the experiments, and the results are shown in Figure 4. It can be clearly seen that the experimental results confirm the correctness of our reasoning. In this problem, the discount factor needs to be set in an appropriate range, and an excessive discount factor will affect the method's performance.

V. Conclusions

In this paper, we make value-based DRL work better in anomaly detection scenarios through multiple comprehensive works: we propose a DQN-D based on double-layer frame concatenation that can be used on a lightweight network to obtain performance close to the LSTM-based baseline method. We propose an improved experience replay pooling strategy to help RNN-based DRL methods perform better sequential learning, outperforming benchmark datasets. Inspired by the Seq2Seq task, we set our sights on the fiery transformer model, propose a transformer-based DRL method, and verify its performance in experiments. Our work is a progressive system,

TABLE I
COMPARISON OF THE PERFORMANCE OF ALL METHODS.

	A1		A2	
	Harsh F1	Revised F1	Harsh F1	Revised F1
Twitter	0.62	0.77	0.51	0.52
EXP	0.72	0.85	0.83	0.84
DQN	0.49	0.51	0.49	0.51
DQN-D	0.71	0.83	0.82	0.82
DRQN	**0.87**	**0.98**	**1.0**	**1.0**
DTQN	0.71	**0.89**	0.74	0.82

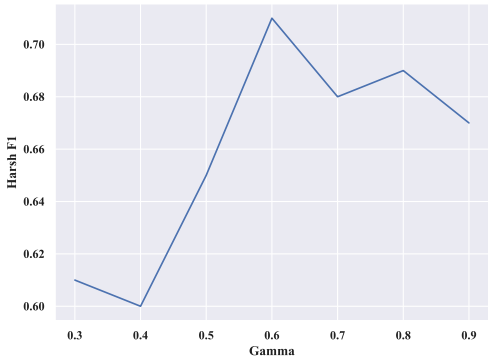

Fig. 4. The effect of different gamma parameters on the performance of the DQN-D method (A1 test dataset).

and we have made multi-directional attempts around the core point of how to help the value-based DRL to obtain more timely information when it acts on the time series. Subject to the scale of the dataset, the potential of our methods has not been fully exploited. Our future work is to consider conducting experiments and research on more realistic datasets.

REFERENCES

[1] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1939–1947.

[2] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proceedings of the 2015 internet measurement conference*, 2015, pp. 211–224.

[3] P. Malhotra, L. Vig, G. Shroff, P. Agarwal *et al.*, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89, 2015, pp. 89–94.

[4] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, vol. 2. IEEE, 2002, pp. 1702–1707.

[5] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 187–196.

[6] Z. Li, W. Chen, and D. Pei, "Robust and unsupervised kpi anomaly detection based on conditional variational autoencoder," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–9.

[7] S. Yan, B. Tang, J. Luo, X. Fu, and X. Zhang, "Unsupervised anomaly detection with variational auto-encoder and local outliers factor for kpis," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 2021, pp. 476–483.

[8] W. Chen, H. Xu, Z. Li, D. Pei, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised anomaly detection for intricate kpis via adversarial training of vae," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019.

[9] M. Yu and S. Sun, "Policy-based reinforcement learning for time series anomaly detection," *Engineering Applications of Artificial Intelligence*, vol. 95, p. 103919, 2020.

[10] T. Wu and J. Ortiz, "Rlad: Time series anomaly detection through reinforcement learning and active learning," *arXiv preprint arXiv:2104.00543*, 2021.

[11] C. Martinez, G. Perrin, E. Ramasso, and M. Rombaut, "A deep reinforcement learning approach for early classification of time series," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 2030–2034.

[12] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. 112963, 2020.

[13] C. Huang, Y. Wu, Y. Zuo, K. Pei, and G. Min, "Towards experienced anomaly detector through reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] S. Son, M.-S. Gil, Y.-S. Moon, and H.-S. Won, "Anomaly detection of hadoop log data using moving average and 3-sigma," *KIPS Transactions on Software and Data Engineering*, vol. 5, no. 6, pp. 283–288, 2016.

[15] E. H. Pena, M. V. de Assis, and M. L. Proença, "Anomaly detection using forecasting methods arima and hwds," in *2013 32nd international conference of the chilean computer science society (sccc)*. IEEE, 2013, pp. 63–66.

[16] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European conference on principles of data mining and knowledge discovery*. Springer, 2002, pp. 15–27.

[17] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[18] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[19] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[20] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 703–716.

[21] X. Gao, "Deep reinforcement learning for time series: playing idealized trading games," *arXiv preprint arXiv:1803.03916*, 2018.

[22] S. Wang, W. A. Chaovalitwongse, and S. Wong, "Online seizure prediction using an adaptive learning approach," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 12, pp. 2854–2866, 2013.

[23] M.-h. Oh and G. Iyengar, "Sequential anomaly detection using inverse reinforcement learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & data mining*, 2019, pp. 1480–1490.

[24] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[25] Twitter, "Twitter anomaly detection," https://github.com/twitter/AnomalyDetection/releases, 2015.