Combining Self-Organizing Map with Reinforcement Learning for Multivariate Time Series Anomaly Detection

Peng Su 1, Zhonghai Lu 2 and DeJiu Chen 1,*

Abstract—Anomaly detection plays a critical role in condition monitors to support the trustworthiness of Cyber-Physical Systems (CPS). Detecting multivariate anomalous data in such systems is challenging due to the lack of a complete comprehension of anomalous behaviors and features. This paper proposes a framework to address time series multivariate anomaly detection problems by combining the Self-Organizing Map (SOM) with Deep Reinforcement Learning (DRL). By clustering the multivariate data, SOM creates an environment to enable the DRL agents interacting with the collected system operational data in terms of a tabular dataset. In this environment, Markov chains reveal the likely anomalous features to support the DRL agent exploring and exploiting the state-action space to maximize anomaly detection performance. We use a time series dataset, Skoltech Anomaly Benchmark (SKAB), to evaluate our framework. Compared with the best results by some currently applied methods, our framework improves the F1 score by 9% from 0.67 to 0.73.

I. INTRODUCTION

Many Cyber-Physical systems (CPS) perceive and process multivariate time series data in different environments to operate a variety of tasks. However, unforeseen conditions (e.g., run-time variation and defects) may cause anomalous data threatening the system’s health [1]. Anomaly detection is a practical approach to condition monitoring for attaining the system’s trustworthiness [2]. To detect anomalies efficiently and effectively, identifying anomalous data is one critical step. However, the complexity of anomalies challenges the effectiveness of anomaly identification and detection [3], [4]. For example, instead of point anomalies that exhibit the discrepancies as simple outlines, there are also contextual anomalies that also depend on the specific spatial and temporal contexts where the discrepancies occur [3], [5]. The contextual anomalies could widely occur in the CPS if not being satisfactorily managed [1], [3], [4]. As one example of CPS, a modern water treatment plant [4], [6] uses multiple sensors to monitor the system’s operational status. When a component fails (e.g., due to water supply damage), such sensors detect several conditions of concern. For example, 1) The pressure sensor of water pipes records rapid descent trends. These data show significantly different values, which can be marked as point anomalies; 2) The thermal sensor detects that the water pump is cooling down. These temperature data are associated to contextual anomalies considering the correlation of the historical data. In general, it is advantageous to monitor the system’s health using multivariate data, which could contain more information about the system [6]. However, conventional methods, especially purely relying on statistical approaches, are insufficient to satisfactorily reveal the complex contextually related spatial and temporal anomalous features implied by the operational data [5]. To support effective anomaly detection of CPS, we propose a semi-supervised learning approach that benefits from the self-learning ability of Deep Reinforcement Learning (DRL). The backbone of DRL is that the agent makes an action to interact with the environment following a Markov Decision Process (MDP), which can be optimized by reward functions based on the actions and the environment’s feedback [7]. However, the tabular dataset (i.e., the operational data recorded in generic files) usually does not support such an interactive decision process since the data sequences are irrelevant to the DRL actions. To overcome this gap, we create an environment for DRL based on Self-Organizing Map (SOM) and thereby allow the DRL agent to interact with the tabular dataset. The results of SOM are used to train Markov chains, which reveal the likely anomalous features by characterizing the transitions among states implied by the operational data. In summary, the contributions of our work are summarized as follows:

• Proposing an anomaly detection framework combining SOM with DRL for multi-dimensional time series anomalous data.
• Using SOM to create an interactive environment for DRL agents conducting the anomaly classification and detection.
• Adopting Markov chains to characterize the temporal process and likely anomalous features.

The rest of this paper is organized as follows: Section II presents the related work on anomaly detection. Section III describes the design of our framework. Section IV introduces the experiment setup and the results evaluated by a public dataset. We conclude with a discussion of future work in Section V.

II. RELATED WORK

Current anomaly detection methods can be classified into supervised and unsupervised/semi-supervised learning approaches according to the knowledge of the anomalous features [3], [5], [8]. The supervised learning is normally time-consuming and labour-intensive due to the need of labeling various anomalies in the CPS [3], [5]. In this paper, we focus on unsupervised/semi-supervised learning [8]–[21].
Without the need of explicit data labeling, anomaly detection methods can be classified based on two categories of models, 1) Sequential models, including Long Short-Term Memory (LSTM) in [9], Hidden Markov chains (HMM) in [10], and Gated Recurrent Unit (GRU) in [11], are learned to predict time series sequences based on training data. The discrepancy between the predicted results and the actual values indicates possible anomalies. However, to accurately detect anomalies, the quantification of the discrepancy replies on large amount training data [13]. 2) Generative models including Variational AutoEncoder (VAE) [12], [13] and Generative Adversarial Network (GAN) [14], [15] learn generative models for input data reconstruction. Deviations between the reconstructed results and the ground truths could be the criteria to indicate the possible of anomalies. The methods usually assume the anomalous data exhibits features which rarely occur in the training data. However, sun an assumption is insufficient for detecting anomalies that always exist in the models implied by the collected data (e.g., contextual anomalies).

We also investigate DRL-based anomaly detection methods [16]–[21]. According to the RL agent’s environments, these methods can be divided into two categories: 1) The RL agent is employed in artefacts (e.g., motors [20], [21]), which act as an environment to interact with the RL agent. For instance, input data are defined by artefacts’ states (e.g., overheating or low battery), which can be altered by the DRL agent’s actions. Such states are characterized by prior knowledge, thus decreasing the complexity of anomalous features. Furthermore, the agent’s actions directly influence the environmental conditions, supporting a MDP to train the DRL agent. 2) The DRL agent detects anomalies from tabular datasets in [16]–[19]. To simulate a MDP for training the agent, some of them (e.g., [17]), [22]) use active learning, a semi-supervised method by querying ground truth of data. These approaches still rely on domain experts to label anomalies, demanding the automatic interaction between anomaly detectors with the tabular dataset.

Based on the delimitation of these current works, we propose a semi-supervised learning approach with a limited label dataset and a large-scale unlabeled dataset to detect multivariate time series anomalies by combining SOM with RL.

### III. METHODOLOGY

#### A. Problem Statement

Considering a wide range of possible anomalies, it is laborious and expensive to acquire all the labels of anomalous data. Therefore, we use a limited labeled dataset \( D^a \) and a large-scale unlabeled dataset \( D^u \), \( (D^a \cap D^u = \emptyset) \), to train the anomaly detector. The design of a DRL agent, including the rewards and actions, is introduced in Section III-B. In Section III-C, we propose an interaction environment for DRL by using SOM to create a basis for the state transition dynamics, through which the Markov chains are trained for revealing anomalous features. The overall workflow (Fig. 1) supported by our framework can be summarized as follows: 1) Implementing the anomaly detection agent by DRL, including the rewards functions, states and action spaces; 2) Generating an interactive environment between the DRL agent and the tabular data. This environment is created by SOM for handling the unlabeled dataset \( D^u \) and a Markov chain for revealing anomalous features.

#### B. Implementing the Anomaly Detection Agent by DRL

DRL is a reward-driven approach to motivate the DRL agent to interact its actions with the environment. By observing the consequences (e.g., rewards) of these actions (Fig. 2), the agent follows an optimization process to alter its behaviors. We use a Markov Decision Process to describe such behaviors [17], [22], which consists of \( \langle S, A, T, r, \gamma \rangle \). In our case, \( S \) represents to a finite set of states, while arbitrary state \( s_t \) is a multivariate input data \( (s_t \in S) \). \( A \) refers to an action space of the agent. This space contains two actions defined as \( \{a^1, a^0\} = A \), where \( a^1, a^0 \) refer to the actions of labelling anomalous and nominal conditions respectively based on the observing states. Depending on the actions made by the agent, \( T(s_{t+1} | s_t, a_t) \) describes transition dynamics from the current states \( s_t \) to the next states \( s_{t+1} \). We define \( r \) as rewards, which consist of external rewards \( r^e \) and internal rewards \( r^i \); \( r^e \) refers to the rewards based on state-action pairs, while \( r^i \) refers to rewards related to the transitions dynamics \( T; \gamma \) refers to a discount factor, which reflects to the importance of the reward \( r \) in a long term. The agent uses its action \( a_t \) (anomalous and nominal conditions) to classify input states from \( D^a \) or \( D^u \). To maximize a future reward \( R_t \), a policy \( \pi \) is proposed to predict an action \( a_t \) given by a state \( s_t \), that is, the actions depend on the given state. Thus, we define the optimal action-value function \( Q^*(s, a) \) of the future reward \( R_t \):

\[
Q^*(s, a) = \max_{\pi} \mathbb{E} \left[ R_t | s_t = s, a_t = a, \pi(s, a) \right]
\]

To select a suitable algorithm to reach this Q-function, we treat the anomaly detection as a one-class prediction, where the action and state space \( A, S \) are discrete, and...
the reward \( r^e \) is explicit and numerical. Thus, we use Deep Q-network (DQN), a model-free and off-policy algorithm, to approximate \( Q^*(s, a) \) [7]. This Q-network is trained by minimizing loss function \( L_i(\theta_i) \) at each iteration \( i \)

\[
L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[ (r^e + \gamma \cdot Q(s\prime, a\prime; \theta_i) - Q(s, a; \theta_i))^2 \right]
\] (2)

where \( \rho(\cdot) \) refers to the probability distribution of different state-action pairs. The agent stores its experience \( e_t = (s_t, a_t, r^e_t, s_{t+1}) \) from dataset \( D^a \) and \( D^u \). In this case, \( 1 \leq t \leq N \), \( N \) is a predefined parameter, referring to the size of the replay memory [23]. The replay memory breaks the correlation of states sequences and improve the sample efficiency. For DQN training stability, two neural networks (target and policy nets) with the same configurations are used in DQN to update their parameters with specific iterations.

**TABLE I**  
**CONFUSION MATRIX**

<table>
<thead>
<tr>
<th>Actual Actions</th>
<th>Predicted Actions (a)</th>
<th>Anomaly ((a^1))</th>
<th>Nominal ((a^0))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Anomaly</td>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td></td>
<td>Nominal</td>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

According to the confusion matrix (Tab. I) of the one-class predication [17], we configure the internal reward \( r^e \) as follows:

\[
r^e = \begin{cases} 
\alpha & \text{if the action is TP} \\
-\beta & \text{if the action is FP} \\
0 & \text{if the action is TN} \\
-\xi & \text{if the action is FN}
\end{cases}
\] (3)

where \( \alpha \) is the reward for detecting actual anomalous data. \( \beta \) and \( \xi \) are non-negative constants, which reflect the tolerance of False Positive (FP) and False Negative (FN). \( \beta = \xi \) indicates that the agent has equal metrics for FP and FN. Otherwise, these parameters can be defined depending on the actual requirements e.g., The safety-critical systems adopt the graceful degradation by leveraging False Positive and False Negative [1].

**C. Generating an Interactive Environment between the DRL Agent and the Tabular Data**

As we mentioned in the previous sections, transition dynamics \( T \) map a state-action pair \((s_t, a_t)\) to next states \( s_{t+1} \). However, if the agent’s actions \((a^0_t, a^1_t)\) cannot impact on the next states, the transitions are irrelevant to the state-action pairs. To solve this issue, we model the transition dynamics \( T(s_t+1|s_t, a_t) \) by interacting with the environments (e.g., tabular dataset) and the agent. Specifically, when the agent receives data from the labeled dataset \( D^a \), the transition dynamics follow a uniform distribution to sample next states \( s_{t+1} \), providing an equal probability for the agent to access every labeled data. Such a method exploit the agent’s learning capability to optimize its actions based on the labeled data and its anomalous behaviors. The transition dynamics need to support the agent to explore the dataset \( D^u \) where the data contain anomalous features implied by the transition probabilities. Therefore, a transition matrix is used to quantify the probabilities of moving from \( s_t \) to \( s_{t+1} \). However, it is difficult to derive this matrix by using multivariate dimensional data [24]. To solve this issue, we propose the following steps by modeling the transition dynamics (Fig. 3): 1) Self-Organizing Map (SOM) clusters input states to a low-dimensional space \( U \). 2) A Markov chain is used for quantifying the transition matrix and its transition probabilities. Therefore, we formulate the transition dynamics as follows:

\[
T(s_{t+1}|s_t, a_t) = f_u(s_{t+1}|s_t, a_t), \forall s_t, s_{t+1} \in D^u
\] (4)

![Fig. 2. The design of DRL-based anomaly detector. According to \( s_t \), the output of the DRL agent is probabilities for the anomalous \( a^1 \) and nominal \( a^0 \) actions. Depending on the actions, the next states \( s_{t+1} \) are sampled from \( D^a \) or \( D^u \).](image)

![Fig. 3. Multivariate unlabeled data map to a two dimensional SOM space. The sample intervals between samples are equal. The length of sampling window is predefined by parameters. The dash line in the SOM space represents the transition probability. Different colors refer to different kernel units.](image)
1) Using Self-Organizing Map to Cluster Multivariate States: Self-Organizing Map (SOM) [25] formulates the input states \(s_i\) to a low-dimensional unit \(u_i \in \mathcal{U} = \{u_1, \ldots, u_t\}\). Each unit \(u_i\) represents a weight vector \(w_i\), which owns the same dimensional as \(s_i\). The similarity between the unit \(u_i\) and the states \(s_i\) is measured by the Euclidean distance \(d(s_i, u_i) = \|s_i - w_i\|\). SOM follows the binary winner-takes-all-decision, selecting a best matching unit (bmu) which has the minimum distance towards the input states \(s_i\). During the training time \(t\), the bmu updates its weight vector as follows:

\[
\mathbf{w}_i(t + 1) = \mathbf{w}_i(t) + \alpha(t) \cdot h(bmu(s_i), \mathbf{u}_i) \cdot d(s_i, \mathbf{u}_i) \tag{5}
\]

where \(bmu(s_i) = \arg\min_{u \in \mathcal{U}} \{d(s_i, u_i)\}\), \(\alpha\) represents a learning rate. \(h(\cdot)\) is a Gaussian kernel function (Eq.6) that indicates an impact-factor in respect to the distance from the bmu. \(\sigma^2(t)\) is a coefficient deceasing monotonically in time.

\[
h(bmu(s_i), \mathbf{u}_i) = \exp\left(-\frac{\|bmu(s_i) - \mathbf{u}_i\|^2}{2\sigma^2(t)}\right) \tag{6}
\]

We define the quantization error [25] to estimate the quality of the output space \(\mathcal{U}\) as follows:

\[
\mathcal{L}_{\text{SOM}} = 1/N_u \sum_{m=1}^{N_u} d(s_i - bmu(s_i)) \tag{7}
\]

where \(N_u\) refers to the amount of states in \(\mathcal{U}\). The quantization error indicates the average distance between every input state and its bmu.

2) Revealing Anomalous Features by using the Markov Chain: A Markov chain is a sequential process that satisfies the Markov property, referring to the condition where a future state is independent of specific past states given the present state [22]. The transition probabilities indicate the temporal dependency among the states. However, due to the binary winner-takes-all-decision strategy, the available units by collected data could be inefficient for the training of Markov chain. In Fig. 3, an illustrative example is shown as the blue cubes, where the states with comparable values may choose different bmus since they only select the most nearby units. This issue can be solved by improving robustness of SOM. We define a kernel unit for generalizing the bmu of an input state \(s_i\) as follows:

\[
\phi_k = \{d(u_i, bmu(s_i)) \leq \delta\} \tag{8}
\]

where \(\phi_k \in \mathcal{M}, \mathcal{M} = \{\phi_0, \ldots, \phi_q\}\), \(q < j\) refers to a subspace of the output space \(\mathcal{U}\). \(\delta\) refers to a threshold of the distance between the bmu of \(s_i\) and \(u_i\). A Markov chain is modeled by the transition path \(\psi_s(1, T) = \{\phi_0, \ldots, \phi_q\}\), where \(T\) is the terminal states of the training dataset from \(\mathcal{D}^u\). This path consists of a transition matrix \(\mathcal{P}\), which represents as follows:

\[
\mathcal{P} = \begin{pmatrix}
P_{00} & \ldots & P_{0q} \\
0 & \ldots & 0 \\
P_{q0} & \ldots & P_{qq}
\end{pmatrix} \tag{9}
\]

A transition probability in \(\mathcal{P}\) defines as follows:

\[
P_{ab} = P(\phi_b|\phi_a) = \frac{c_{ab}}{\sum_{t=0}^{q} c_{ai}} \tag{10}
\]

where \(c_{ab}\) refers to a counter that indicates transitions \(\phi_a\) to \(\phi_b\). \(\sum_{t=0}^{q} c_{ai}\) refers to the total occurrence moving \(\phi_a\) to any other kernel units. The transition matrix \(\mathcal{P}\) uses kernel units to represent the temporal dependency of input states. Depending on the transition matrix, those low probability transitions could imply anomalies. Therefore, the transition dynamics can be formalized (Eq.4) as follows:

\[
f_u(s_{t+1}|s_t, a_t) = \begin{cases}
f_{\text{min}}(P(\phi(s')|\phi(s_i); \mathcal{U})) & \text{if } a_t = a^0 \\
f_{\text{max}}(P(\phi(s')|\phi(s_i); \mathcal{U})) & \text{else}
\end{cases} \tag{11}
\]

where \(f_{\text{min}}\) and \(f_{\text{max}}\) refer to the \(\text{argmin}\) and \(\text{argmax}\) functions which indicate the next states in the output space \(\mathcal{M}\). Considering the above transition dynamics, the internal reward \(r^i\) (Eq. 12) encourages the DRL agent to sample the anomalous states which is rare occurred in this sequence.

\[
r^i_t = \begin{cases}
\exp(-P(\phi(s_{t+1})|\phi(s_t); \mathcal{U})) & \text{if } s_t, s_{t+1} \in \mathcal{D}^u; \\
0 & \text{else}
\end{cases} \tag{12}
\]

For any state \(s_t\) from \(\mathcal{D}^u\), the agent receives an overall reward defined as \(r_t = r^i_t + r^v_t\). This reward helps the DRL agent explore and exploit the state-action space to maximize anomaly detection performance.

![Fig. 3. An illustrative example is shown as the blue cubes, where the states with comparable values may choose different bmus since they only select the most nearby units.](image)

Fig. 3. An illustrative example is shown as the blue cubes, where the states with comparable values may choose different bmus since they only select the most nearby units. This issue can be solved by improving robustness of SOM. We define a kernel unit for generalizing the bmu of an input state \(s_i\) as follows:

\[
\phi_k = \{d(u_i, bmu(s_i)) \leq \delta\} \tag{8}
\]

where \(\phi_k \in \mathcal{M}, \mathcal{M} = \{\phi_0, \ldots, \phi_q\}\), \(q < j\) refers to a subspace of the output space \(\mathcal{U}\). \(\delta\) refers to a threshold of the distance between the bmu of \(s_i\) and \(u_i\). A Markov chain is modeled by the transition path \(\psi_s(1, T) = \{\phi_0, \ldots, \phi_q\}\), where \(T\) is the terminal states of the training dataset from \(\mathcal{D}^u\). This path consists of a transition matrix \(\mathcal{P}\), which represents as follows:

\[
\mathcal{P} = \begin{pmatrix}
P_{00} & \ldots & P_{0q} \\
0 & \ldots & 0 \\
P_{q0} & \ldots & P_{qq}
\end{pmatrix} \tag{9}
\]

A transition probability in \(\mathcal{P}\) defines as follows:

\[
P_{ab} = P(\phi_b|\phi_a) = \frac{c_{ab}}{\sum_{t=0}^{q} c_{ai}} \tag{10}
\]

where \(c_{ab}\) refers to a counter that indicates transitions \(\phi_a\) to \(\phi_b\). \(\sum_{t=0}^{q} c_{ai}\) refers to the total occurrence moving \(\phi_a\) to any other kernel units. The transition matrix \(\mathcal{P}\) uses kernel units to represent the temporal dependency of input states. Depending on the transition matrix, those low probability transitions could imply anomalies. Therefore, the transition dynamics can be formalized (Eq.4) as follows:

\[
f_u(s_{t+1}|s_t, a_t) = \begin{cases}
f_{\text{min}}(P(\phi(s')|\phi(s_i); \mathcal{U})) & \text{if } a_t = a^0 \\
f_{\text{max}}(P(\phi(s')|\phi(s_i); \mathcal{U})) & \text{else}
\end{cases} \tag{11}
\]

where \(f_{\text{min}}\) and \(f_{\text{max}}\) refer to the \(\text{argmin}\) and \(\text{argmax}\) functions which indicate the next states in the output space \(\mathcal{M}\). Considering the above transition dynamics, the internal reward \(r^i\) (Eq. 12) encourages the DRL agent to sample the anomalous states which is rare occurred in this sequence.

\[
r^i_t = \begin{cases}
\exp(-P(\phi(s_{t+1})|\phi(s_t); \mathcal{U})) & \text{if } s_t, s_{t+1} \in \mathcal{D}^u; \\
0 & \text{else}
\end{cases} \tag{12}
\]

For any state \(s_t\) from \(\mathcal{D}^u\), the agent receives an overall reward defined as \(r_t = r^i_t + r^v_t\). This reward helps the DRL agent explore and exploit the state-action space to maximize anomaly detection performance.

![Fig. 4. The architecture of the Skoltech test bench. Water pump cycles water from the tank. There are multiple sensors to collect the cycle process. When the valves are close or pump are damaged, the collected data are marked as anomalous.](image)

Fig. 4. The architecture of the Skoltech test bench. Water pump cycles water from the tank. There are multiple sensors to collect the cycle process. When the valves are close or pump are damaged, the collected data are marked as anomalous.

IV. CASE STUDY

A. Dataset Description

We use the Skoltech Anomaly Benchmark (SKAB) [26], a multivariate time series dataset, to evaluate the performance of the proposed DRL anomaly detector. This benchmark collects data from a water circulation platform where the system status are monitored by different sensors, such as temperature sensors, voltage/current measurements, vibration sensors, and pressure sensors (Fig. 4). Compare with other
multivariate datasets [4], [14], the Skoltech dataset has following advantages: 1) The environment represents one type of CPS; 2) It is difficult to detect anomalies relying on one dimensional data in this dataset; 3) The anomalies exhibit various features (e.g., point and contextual anomalies).

B. Experiments Setup

The SKAB benchmark, which includes 727 anomalous data and 7985 nominal data, is divided into the labeled dataset $D^a$ and unlabeled dataset $D^u$. The labeled dataset is a part of anomalous data which are marked in the raw files. With the help of the labeled dataset, the DQN exploits the state-action pairs for reaching optimal actions. Meanwhile, the unlabeled dataset $D^u$ contains the nominal data and the rest of the anomalies, supporting the DRL agent to explore various anomalies with different state-action pairs.

The first step for training our anomaly detector is to create the transition dynamics: We set a $8 \times 8$ output space $U$ to map the unlabeled data, while 10 kernel units are used for training the Markov chain. A random seed $p_{\text{sample}}$ is used for determining the sampling between $D^u$ and $D^a$, which represents as follows:

$$T(s_{t+1}|s_t, a_t) = \begin{cases} U(D^a), & \text{if } p_{\text{sample}} \leq 0.5; \\ f_u(s_{t+1}|s_t, a_t), & \text{else} \end{cases}$$

(13)

Secondly, we build a DQN with three hidden layers, while the input layer contains 8 units responsible for processing the multivariate data. With the softmax layer, the output layer indicates the probability of two actions (anomalous or nominal conditions). To train this DQN, we set the learning rate $lr = 0.00025$, the discount factor $\gamma = 0.85$ and the minibatch size as 32. For every training episode, there are 1000 training steps. At the start of training the DQN, the probability $p_\epsilon$ of $\epsilon$-greedy algorithm is equal to 0.5. After 800 episodes, we set $p_\epsilon = 0.25$ to exploit the optimal action. The external rewards $\alpha, \beta, \xi$ are equal to 1. We use F1 score determined by the precision and recall to evaluate the performance of the anomaly detector.

C. Results Analysis

From Fig. 5.(a), we highlight the anomalous data as the red zones where the DRL agent efficiently detects these anomalies. The agent’s actions according to a test trajectory is shown in Fig. 5.(b). To further verify the performance of anomaly detection, we also compare our framework with benchmarks given by [26] (Tab.II). Our method performs a better recall rate than the isolated frost, referring to a distance-based method to detect point anomalies. Such an improvement implies the benefit from the reward mechanism by specifying the penalty of false negatives. We also compare our work with anomaly detectors based on DNN techniques. Our detector obtains better performance than LSTM and Autoencoder. These methods define the anomalous data by a threshold of a discrepancy between the predicted data and the ground truth, while our anomaly detector models various anomalous features via the Markov chains. The comparison between our work and the DRL-based methods in [16], [17] indicates that our agent performs better under the same amount of labeled dataset. Moreover, compared with these current methods, the agent supports a flexible scenario where the reward can be adjusted according to requirements of precision or recall. As an example, we switch the rewards of FP from -1.0 to -1.8, thus the precision rate arises from 0.62 to 0.94.

To sum up, our approach has the following advantages: 1) Compared with threshold-based methods of unsupervised/semi-supervised learning approaches, which rely on prior knowledge to label the features of anomalous data, we use transition dynamics to describe the multivariate data context, improving the explainability of the anomaly detection; 2) Compared with supervised anomaly detectors, our approach can deal with different anomalies features, benefiting from RL’s self-learning ability; 3) Compared with
other anomaly detection methods based on DQN in [16]–[19], our work interacts the agent with tabular dataset to avoid querying the ground truths.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>F1 Score</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect detector</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t-squared+k (PCA)</td>
<td>0.67</td>
<td>0.86</td>
<td>0.36</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.64</td>
<td>0.85</td>
<td>0.61</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>0.45</td>
<td>0.93</td>
<td>0.35</td>
</tr>
<tr>
<td>Isolation forest</td>
<td>0.4</td>
<td>0.94</td>
<td>0.28</td>
</tr>
<tr>
<td>Our Work1</td>
<td>0.61</td>
<td>0.62</td>
<td>0.82</td>
</tr>
<tr>
<td>Our Work2</td>
<td>0.73</td>
<td>0.94</td>
<td>0.60</td>
</tr>
<tr>
<td>DQN</td>
<td>0.51</td>
<td>0.67</td>
<td>0.41</td>
</tr>
</tbody>
</table>

1 We set the reward of FP and FN as -1.
2 We set the reward of FP and FN as -1.8 and -1.

V. CONCLUSION AND FUTURE WORK

This paper proposes an anomaly detection framework by combining SOM with DRL through DQN. This DRL agent detects anomalies by exploiting the labeled dataset and exploring the unlabeled dataset. A SOM-based Markov chain characterizes the temporal dependency and constitutes the basis for justifying the rewards. By using the semi-supervised learning, our approach decreases the requirements for explicitly labelling the data. As future work, we will investigate the opportunities by further optimizations of the hyper-parameters, especially for the SOM based environment. Another future task is related to a further investigation of the correlation between the internal rewards and the F1 score.

ACKNOWLEDGMENT

This work is supported by 1. KTH Royal Institute of Technology with the industrial research project ADinSOS (2019065006); and 2. the Swedish government agency for innovation systems (VINNOVA) with the cooperative research project Trust-E (Ref: 2020-05117) within the programme EUREKA EURIPIDES.

REFERENCES