Journal of Big Data

BRIEF REPORT Open Access

Deep reinforcement learning for data-efficient weakly supervised business process anomaly detection

Eman Abd Elaziz*, Radwa Fathalla and Mohamed Shaheen

*Correspondence:
eman.abdelaziz@aast.edu

College of Computing and Information Technology, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt

Abstract

The detection of anomalous behavior in business process data is a crucial task for preventing failures that may jeopardize the performance of any organization. Supervised learning techniques are impracticable because of the difficulties of gathering huge amounts of labeled business process anomaly data. For this reason, unsupervised learning techniques and semi-supervised learning approaches trained on entirely labeled normal data have dominated this domain for a long time. However, these methods do not work well because of the absence of prior knowledge of true anomalies. In this study, we propose a deep weakly supervised reinforcement learning-based approach to identify anomalies in business processes by leveraging limited labeled anomaly data. The proposed approach is intended to use a small collection of labeled anomalous data while exploring a huge set of unlabeled data to find new classes of anomalies that are outside the scope of the labeled anomalous data. We created a unique reward function that combined the supervisory signal supplied by a variational autoencoder trained on unlabeled data with the supervisory signal provided by the environment's reward. To further reduce data deficiency, we introduced a sampling method to allow the effective exploration of the unlabeled data and to address the imbalanced data problem, which is a common problem in the anomaly detection field. This approach depends on the proximity between the data samples in the latent space of the variational autoencoder. Furthermore, to efficiently model the sequential nature of business process data and to handle the long-term dependences, we used a long short-term memory network combined with a self-attention mechanism to develop the agent of our reinforcement learning model. Multiple scenarios were used to test the proposed approach on real-world and synthetic datasets. The findings revealed that the proposed approach outperformed five competing approaches by efficiently using the few available anomalous examples.

Keywords: Business process, Deep reinforcement learning, Weakly supervised learning, Variational autoencoder, Long short-term memory, Self-attention, Transformers, Imbalanced data, Process mining, Process discovery, Conformance checking, Balanced accuracy

Introduction

Anomalies in business processes can result in significant losses. In 2012, the Association of Certified Fraud Examiners reported that there had been 1388 fraud cases in 96 countries, which caused $1.4 billion in losses. On average, organizations have lost a gross profit of 7% annually to fraud [1]. Manual identification of abnormalities in business process data is time-consuming, labor-intensive, and prone to human error owing to the vast volumes of data. There is a requirement for an automated system that can discover abnormalities in massive business process logs.

A business process is a set of activities or tasks triggered by an event to achieve a specific organization's goal. The process can be simple or complex, depending on the number of activities. They can be short or long. Longer processes tend to have multiple dependencies between the activities [2]. Business processes integrate multiple data sources, applications, and information technology systems, and manage public as well as confidential private information [3]. Furthermore, business processes are carried out in a variety of flexible infrastructures and networked settings where they play important roles and allow key business activities.

The event log is the main data structure of the business process data that contain information regarding what activities have been performed in a process, who executed them, at which time, etc. [4]. Each event log is separated into traces, and each trace reflects one process instance's execution.

In the context of business process analytics, an anomaly can be an unusual process execution or a noise in the event log that can be produced by system failure, data entry error, or a fraud attempt [5]. Because of its wide range of applications, such as fraud detection, intrusion detection, and outlier identification, business process anomaly detection has emerged as an important issue in recent years.

Business process data are complex because they are sequential time series data. A business process is a group of activities that occur in sequential order over a while. In addition, most corporate processes include complex interdependencies between activities. Furthermore, activities are carried out in contexts where mistakes are possible and are acknowledged as a normal part of the process. As a result, even for human professionals, detecting anomalies in business processes is a difficult task [6].

The difficulties in business process anomaly detection task are further aggravated by the difficulties in the acquisition techniques of the samples in the training set on which we apply automated learning. First, most real-life business process event logs or datasets lack an established process model that holds information regarding their process logic. Second, there is a significant expense associated with manually labeling the acquired traces, which makes acquiring large-scale labeled datasets difficult. Third, because there are numerous sources of anomalies, there is an infinite pool of anomaly classes. Anomalies are also uncommon and surprising in real-world datasets. As a result, obtaining training data that cover all probable types of anomalies is difficult.

Business process anomaly detection methods have improved and, at first, are moving toward using process mining techniques [7–13]. However, the effectiveness of process mining algorithms is highly dependent on the availability of a clean dataset and specified process model, which is usually not the case, as the business process datasets most likely contain anomalies and the predefined model is often not available. The methodologies then evolved into statistical procedures that rely on the assumption that data originate from the anticipated statistical distribution, which does not always hold in business process data [14]. Furthermore, statistical methodologies are frequently insufficient for dealing with high-dimensional data.

For these reasons, research has shifted away from process mining and statistical techniques and toward machine learning (ML) approaches, which may infer process logic from a dataset and may exploit patterns identified within the dataset to discern normal from abnormal process executions. The ML techniques published in the business process anomaly detection literature may be categorized as supervised, unsupervised, and semi-supervised approaches based on the level of label availability.

Due to the difficulty of obtaining labeled anomaly data in business process field, research based on supervised learning is very rare [1, 15, 16]. The major research efforts in this area have focused on unsupervised anomaly detection methods because they require no labeled training data and can detect diverse anomalies, as they are not restricted to specific anomaly classes [17–22]. However, these methods suffer from significant performance drop because they lack prior knowledge of true anomalies. They rely only on their assumption based on the distribution of anomalies [17–22].

A semi-supervised approach is utilized to detect anomalies in business process data [23]. However, these approaches are inapplicable to the problem of unlimited pool of classes because they are exclusively fitted to the limited labeled anomalies, ignoring the supervisory signals from the possible anomalies in the unlabeled data. The semi-supervised approach is therefore inapplicable to the detection of novel anomaly classes.

In this work, we treat the problem of business process anomaly detection by deviating from the traditional unsupervised learning setting and delving into the weakly supervised learning setting. Weakly supervised learning is an umbrella term covering various studies that attempt to construct predictive models by learning with weakly labeled training data (i.e. partially, inexactly, or inaccurately labeled data) [24].

We have created a small set of labeled anomaly data for this purpose. Exploiting those limited labeled anomaly samples without assuming that they encompass every class of anomaly will yield improvements over both supervised and semi-supervised methods. This is of important practical benefit, because anomalies are unpredictable in form and often expensive to miss. Furthermore, these labeled anomalies provide valuable prior knowledge, leading to accuracy improvements over the unsupervised methods.

The core challenge revolves around the ability to effectively exploit those limited labeled anomalous samples without confining classification to their labels. Another challenge we encounter in our weakly supervised learning mode is the high-class imbalance as we have a small set of minority class samples (anomaly samples) and a large set of majority class samples (normal samples). Overcoming this challenge is critical as this imbalance harms the performance of any classification task [25, 26].

To overcome the aforementioned challenges, we apply Deep Reinforcement Learning (DRL) to the problem of weakly-supervised anomaly detection in business process data. The proposed DRL approach is designed to exploit the small set of labeled anomalous data and explore the large set of unlabeled data, so it can detect new classes of anomalies that lie beyond the scope of the labeled anomaly data. The exploration of the DRL framework in this work is empowered via using a Variational AutoEncoder (VAE). The

VAE provides supervisory signal from the unlabeled data to encourage the unsupervised exploration of those data.

We solve the imbalanced data problem in our work through defining a sampling function that is biased towards choosing anomalous traces from the unlabeled dataset over the normal ones. This process is carried out based on the Euclidean distances between the traces in the latent space of the variational autoencoder. To simulate the sequential nature of business process data and taking the long-term dependencies between activities in consideration, the core of the DRL agent is a Long Short-Term Memory (LSTM) network combined with a self-attention mechanism.

In summary, this work aims for the following four major contributions:

- To the best of our knowledge, we are the first to investigate the problem of anomaly detection in the business process domain based on the weakly supervised paradigm.
- To the best of our knowledge, we are the first to propose a deep reinforcement learning approach for detecting anomalies in the business process domain.
- We investigate the use of an autoencoder in performing the exploration task associated with the proposed approach.
- We explore utilizing LSTMs and transformers to achieve the most robust modeling for the business process data and incorporating them in our DRL model.
- The proposed framework is instantiated into an anomaly detection model. The model is extensively evaluated on two real life and two synthetic datasets where multiple scenarios with different known anomaly classes are considered. The evaluation results confirm that the proposed model outperforms five state-of-the-art methods.

The rest of this paper is organized as follows. In section "Related work", we present an overview of related works in the field of business process anomaly detection. Section Datasets provides technical details of the proposed framework. For the evaluation of the proposed model, the results of the conducted experiments are presented and discussed in section "Proposed approach". Section Experimental evaluation summarizes the results of our study and presents the plans for future work.

Related work

This section addresses the approaches used in the existing business processes anomaly detection literature and the techniques adopted in the anomaly detection field in general. The business process anomaly detection approaches can be classified into statistical, process mining and ML approaches.

It is common in the field of process mining [7] to employ discovery methods to construct process models from event logs, followed by conformance checking to find an anomalous trace. This technique can detect the deviation between the process model and process executions [8–13]. This approach, however, has certain downsides [8–13]. For starters, it cannot use event attributes. Therefore, it can only detect anomalies at the control flow level, not at the attribute level. The second disadvantage of this approach is that it might be overly strict, resulting in high false-positive rates and the ability to identify only point anomalies. The third disadvantage is that it relies on a clean dataset, that is, no anomalous traces must be present in the dataset during discovery. However, this is rarely the case because the business process event logs usually contain anomalies. Statistical anomaly detection approaches generate a stochastic model that represents investigated behavior. If unseen behavior is mapped on the low-probability region of this stochastic model, then an anomaly was found. A statistical model is fitted (i.e., trained) to the provided normal data for this purpose. Statistical methods rely on a finely adjusted and precise assumption regarding the data distribution utilized to create the stochastic model. Getting this right might be simple for a limited set of dimensions but gets harder for multidimensional data (i.e., if multiple process perspectives should be incorporated). Such an approach is applied by Solti and Kasneci [14] to identify temporal anomalies (i.e., unexpected durations of activities as point anomalies).

The application of ML, especially deep learning methods, to identify anomalies in business processes has gained increased attention in recent years. These techniques are divided into supervised, semi-supervised, and unsupervised approaches based on the level of label availability. Supervised methods involve training a binary or multi-class classifier using labels of both normal and anomalous traces. The rule mining-based technique is an example of a supervised approach used in the field of business process anomaly detection [1, 15, 16]. This method develops rules that capture the typical behavior of traces. As a result, if the criteria fail to capture a trace, it is classified as an anomalous trace.

Sarno et al. [15] used association rule mining to uncover business process anomalies. However, this study has several drawbacks, including the fact that the rules were developed manually (e.g., a user defined the expected maximum activity time) and that the order dependencies between activities were not validated. Sarno and Sinaga [16] then offered ontologies to gather further understanding of the processes under consideration. Furthermore, researchers attempted to combine the capabilities of process mining and association rules to build a hybrid model. One of these studies is presented by Sarno et al. [1]. They proposed integrating process mining, fuzzy multi-attribute decision making and fuzzy association rule learning to detect anomalies in business processes. They used a Process mining technique to check the conformance between the event log and the process model. Then, they applied the fuzzy multi-attribute decision making to calculate the anomaly rates. Finally, they applied the fuzzy association rule learning to generate association rules that can be used to detect anomalous traces.

The advantage of the rule mining approaches proposed in the literature is that they generate a small set of rules that enables the detection of anomalous traces. However, in the event of a flexible business process, the strictness enforced by the produced rules leads to inaccurate analytical conclusions. Furthermore, the majority of the association rule-based techniques rely on human interventions and expert knowledge. To guarantee effective anomaly detection performance, experts must identify every execution trace as normal or abnormal and must analyze developed rules.

The performance of supervised approaches is not optimal because of the class imbalance data problem. Furthermore, even after monitoring system behavior for lengthy periods, it is difficult to gather training data that cover all probable classes of anomalies. Because supervised approaches are limited to the classes seen in the training data, they are unworkable.

When it comes to unsupervised approaches, some of them make use of clustering algorithms. These approaches try to identify clusters of normal and abnormal traces, with anomalous traces represented by tiny and dispersed clusters. This necessitates determining the distance or similarity of many traces. Zhu et al. [17] converted the behavior of resources into vectors to measure the distance between these vectors. Hereby, the assumption is that resources that are assigned to the same roles should behave similarly. Similarly, Hsu et al. [18] used the same approach, but the traces were grouped based on temporal viewpoint, implying that the same activity behaved similarly in all executions in terms of temporal perspective (e.g., its execution time). Folino et al. [19] applied a clustering method that mine patterns to cluster traces (e.g., based on the existing control flow patterns).

Furthermore, a clustering/likelihood graph-based approach was proposed by Böhmer and Rinderle-Ma [20]. A reference model that enables the calculation of the likelihood of business process traces was generated. If a trace is unlikely compared to other traces, then the trace is marked as anomalous.

With the developments in the field of deep learning, recent publications propose the use of deep neural networks to detect anomalies in business process data. Nolle et al. [21] applied a neural network based denoising autoencoder to the detection of anomalies in business process data. Through their training, the autoencoder could reproduce the input trace that should be analyzed. If the input and the reproduced trace are dissimilar then the trace is considered an anomaly.

Subsequently, an unsupervised business process anomaly detection approach based on neural networks was introduced by Nolle et al. [5]. They divided the log based on control flow and data perspectives. These two perspectives were used as inputs for their neural network architecture aimed at predicting both perspectives for the next event. Anomalies are detected in case of discrepancies between the predicted and the actual next event. However, this method is incapable of handling many attributes in the data perspective as the learning phase is unable to cope with unseen values (both in the activity and data perspective).

Recently, Krajsic and Franczyk [22] presented an approach that uses variational autoencoders and support vector machines for business process anomaly detection.

The semi-supervised approaches, adopted in the business process anomaly detection domain, are very rare. Krajsic and Franczyk [23] presented a deep learning model to optimize the use of available labeled normal or anomalous traces in the detection of anomalies. The proposed model was built using a bidirectional LSTM variational autoencoder to filter anomalous traces from the business process data using the reconstruction error. The main drawback of this approach is that the model is exclusively fitted to the limited labeled anomalies, ignoring the supervisory signals from the possible anomalies in the unlabeled data. Therefore, this approach is inapplicable to the detection of novel anomaly classes.

Anomaly detection has been studied also in various other real-world application areas including intrusion detection in cybersecurity, credit card fraud detection, mechanical defect detection, anomaly detection in smart phone applications, online social networks, internet of things, and medical diagnoses [27, 28]. As expected the research in this domain has evolved in similar directions of statistical and]. As expected, the research in this domain has evolved in similar directions where statistical and machine learning approaches have been employed.

Recently, many deep learning architectures have been introduced for time-series data anomaly detection. The most common of these architectures are based on recurrent neural network (RNN) or LSTM [29–41] because LSTM is capable of handling sequential time series data. In addition, some autoencoder-based deep learning architectures have been proposed for time-series anomaly detection [42–46]. Some studies have combined the efficiency of LSTM neural networks with the reconstruction properties of autoencoders and developed a combined model (LSTM-AE) for the time-series anomaly detection task [29, 47, 48]. A few approaches have used generative adversarial networks (GANs) for this purpose [49–51].

Lately, deep reinforcement learning has been taken into consideration for solving the time series anomaly detection problems. Xu [52] proposed an anomaly detection method based on a reinforcement learning algorithm referred to as TD learning, where the reward function is based on the Markov decision process. In that work, the anomaly detection problem of multistage cyberattacks is considered as an application case.

A framework that uses deep Actor–Critic reinforcement learning for sequential testing of anomaly detection in sensor networks was proposed by Zhong et al. [53]. The proposed method focused on dynamic detection of anomalies based on posterior probabilities.

Huang et al. [54] presented a deep reinforcement learning time series anomaly detector with a deep Q-network algorithm in supervised settings. Furthermore, Yu and Sun [55] presented a generic policy-based reinforcement learning framework to address the time series anomaly detection in Internet of Things (IOT) data. Malla et al. [56] also proposed an end-to-end framework for sequential anomaly detection of IOT data using inverse reinforcement learning (IRL). Recently, Oh and Iyengar [57] applied IRL to sequential anomaly detection in unsupervised settings.

The work described in this paper is very different from the aforementioned studies [52–56]. Previous studies were confined to supervised settings and were unable to generalize to the weakly supervised settings as in our case. Therefore, they are unable to satisfy most of real world applications in general and business process application in particular because they require fully labeled datasets. The proposed work is also different from [57], which can handle only the unsupervised settings and was unable to handle the weakly supervised settings that our work considers. Moreover, they focused on learning a reward function whereas we focus on leveraging predefined reward functions to encourage the agent in exploitation of the labeled anomalous data and exploration of the unlabeled data.

To sum up, the proposed reinforcement learning based anomaly detection approaches are insufficient in the current literature. Furthermore, they are not directly applicable to our problem because they cannot work in weakly-supervised learning settings, cannot detect novel classes of anomalies, and cannot handle the imbalanced data problem. Therefore, there is a room for improvement to make the reinforcement learning applicable for our task.

Datasets

The event log is the key data structure of the business processes in the discipline of business process modeling. An event log is made up of traces, each of which is made up of events that occurred within a process. An activity name is assigned to each event.

Definition 1 The event, trace, and event log. Let E be the set of all events. A trace is a sequence of events $t \in E^*$, where E^* is the set of all sequences over E. Let T be the set of all traces. An event log is a set of traces $L \subseteq T$.

To assess the proposed model, two synthetic event logs were generated from random process models of different complexities using Processes and Logs Generator 2 (PLG2) [58]. The amount of activities determines the model's complexity. Two real life event logs from the Business Process Intelligence Challenge, notably BPIC12 and BPIC17, were utilized in addition to the synthetic logs. Refer to Table 1 for the information regarding the datasets.

It is a common practice to apply artificial anomalies to synthetic logs [5, 11, 20, 21] as well as real-life logs [5, 20], with the assumption that the real-life logs already contain few natural anomalies. As a result, the performance of the algorithms can be measured on the real-life logs to show feasibility whereas the synthetic logs were used to assess accuracy.

When applying artificial anomalies, we can gather a ground truth dataset. We use these ground truth labels to turn our problem from the unsupervised setting into the weakly supervised setting.

Weakly supervised learning has been proven to provide better performance over unsupervised learning in the anomaly detection field in many studies [59, 60]. The labeled anomaly data were used in these experiments to build expressive representations of normality/abnormality for more accurate anomaly identification. In the case of weakly supervised learning, it is clear that we needed to incorporate certain techniques that could handle unlabeled dataset as well as others that could use labeled dataset. We used the most commonly applied artificial anomalies in the business process field.

These anomalies can be defined as follows:

- Rework: an activity has been executed twice.
- Skip: an activity has not been executed.
- Switch: two events have been swapped during execution.

Table 1 Overview showing dataset information

Name	#Traces	#Activities
Dataset 1	10K	13
Dataset 2	10K	20
BPIC12	13K	36
BPIC17	31K–43K	8–26

Proposed approach

This study presents a deep reinforcement learning approach for weakly supervised anomaly detection in business process data. In the proposed approach, a VAE is used to empower the exploration process. The VAE generates a complementary supervisory signal from the unlabeled data, allowing the DRL agent to explore rare and novel anomalies in the unlabeled data efficiently.

Preprocessing

We initiate processing by transforming the event logs into a numerical representation to be input to the Neural Networks. To accomplish this, each event in each trace is encoded using one-hot vector encoding. It encodes each event by a K-dimensional vector, where K is the number of unique activities in the dataset, We take a consistent ordering over the set of activities A, and use index $\in \{1,..., K\}$ to indicate the position of an activity in it. To encode an event, we set the corresponding index of the one-hot vector to a fixed value of one and all other indices to zero.

We represent the event logs as third-order tensors. Each event is a first order tensor $e \in R^A$. Each trace is then represented as a second-order tensor $t \in R^{M \times A}$; with M is the maximum trace length of all traces in the log L. To force all traces to have the same size, we pad all shorter traces with event tensors only containing zeros. Thereafter, the log L is represented as a third-order tensor $L \in R^{N \times M \times A}$; with N is the number of traces in log L.

The synthetic and real life datasets were split into training and testing sets. Thereafter, the training set was split into a small set D_1 and a large set D_2. We apply artificial anomalies to all the traces in D_1 for which the ground truth is available, to form the small labeled anomalous dataset D_{la}. The aim is to encourage the DRL model to learn a broad spectrum of anomalies. We further apply artificial anomalies to a few traces in D_2 without retaining their label in order to form an unlabeled dataset D_u that mimics the natural phenomenon of anomalous business processes. In addition, artificial anomalies are applied to a fixed percentage of traces in the testing set (while retaining their labels) to enable accuracy evaluation of the proposed model.

D_{la} consists of labeled anomalous traces from a group of known anomaly classes $\{C_1, C_2, \ldots, C_i\}$. Similarly, D_u consists of normal traces and some anomalous traces of anomaly classes $\{C_1, C_2, \ldots, C_i, C_{i+1}, C_{i+2}, \ldots, C_{i+j}\}$, where $\{C_{i+1}, C_{i+2}, \ldots, C_{i+j}\}$ are unknown anomaly classes.

Deep reinforcement learning-based anomaly detection framework

We chose to use reinforcement learning (RL) for building our framework because of its generalization properties and incremental self-learning property [61].

Our DRL framework was designed to make a balance between exploiting the small labeled dataset D_{la} and exploring for new classes of an anomaly in the large unlabeled dataset D_u. It was designed to leverage labeled anomalous data to enhance detection accuracy without limiting the set of anomalies searched for to the given anomaly examples.

The exploration of our DRL framework was carried out using a VAE. A VAE provides a supervisory signal from the unlabeled dataset D_u to encourage the unsupervised exploration of those data. Many earlier research has employed autoencoders as unsupervised deep anomaly detection techniques [22, 42–46]. Autoencoders are implemented as sequence-to-sequence models. They are made up of an encoder and a decoder network that can be trained end-to-end [62]. The encoder takes the original input and generates a fixed-sized representation that is substantially smaller in dimension than the input. Latent representation is also sent into the decoder network, which has been trained to reconstruct the original input. The VAE in our framework is trained on normal data. When the input sequence is different from the normal data, the VAE will not be able to reconstruct the sequence with equal quality compared to the reconstruction of normal data. As a result, an anomaly score can be calculated based on the difference between the reconstructed sequence and original input sequence. The VAE is at the heart of the weakly-supervised learning functionality of our work, because it has the ability to signal that something went wrong without being trained on the type of anomaly.

The proposed framework's goal is to teach the DRL agent by interacting with an environment created from training data. We designed the environment to empower the agent to exploit the small set of labeled anomalies and investigate the unlabeled data for any potential anomalies that lie outside the scope of this set. A mixed reward function is intended to produce balanced exploitation–exploration by utilizing a synthesis of supervisory signals from both labeled and suspicious unlabeled anomalies. The training phase's exploitation and exploration procedures allow the agent to learn regarding abnormality and to determine if a new trace is anomalous or not.

The major key components of the adopted RL in our framework are the agent, environment, and reward [63]. The process starts at time t with an agent taking an action a_t (classifying as normal or anomaly) based on the current state s_t and policy π. Following that, the environment responds to the taken action in the form of a reward r_t. In every time step t, the agent receives a new state and reward and eventually learns to analyze the policy and perform the optimal action. The goal of the agent is to learn an optimal policy that maximizes the discounted accumulated rewards, which can be defined as follows:

$$\pi^* = \text{argmax}_\pi \mathrm{E}^\pi \left\{ \sum_{t=0}^{\infty} \gamma^t r_t \right\} \tag{1}$$

which is the maximum expected return starting from a state s, taking the action $a \in \{a_0, a_1\}$, and thereafter following a behavior policy π, with the return defined as the sum of rewards r_t discounted by a factor γ at each time step t. $\gamma \in [0, 1]$ is a discount factor reflecting the decreasing value of the current reward on future ones.

To derive a policy, the state–action–value function (Q-value function) can assess how good it is to choose a particular action a when in a state s under a stochastic policy π, which can be defined as follows:

$$Q^\pi(s, a) = \mathrm{E}^\pi [\textstyle\sum_{T=t}^{\infty} \gamma^t r_t | s_t = s, a_t = a] \tag{2}$$

where r_t is the immediate reward of executing an action a_t on state s_t. Given the Q-value function, a policy can be derived as follows:

$$\pi(s) = \text{argmax}_a Q^{\pi}(s,a) \tag{3}$$

When the update equation converges, the optimal Q-value, as well as an optimal policy is obtained.

Q-learning [63] is one of the RL algorithms that focus on estimating the value function. Q-learning has received a lot of attention because of its ease of use and efficacy. Q-learning in a discrete state space uses an online off-policy update as follows:

$$Q^{\pi}(s,a) = Q^{\pi}(s,a) + \alpha\left(r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right) \tag{4}$$

where s represents the current state, a represents the current action, $s^{'}$ represents the next state, $a^{'}$ represents the next action, γ represents the discount factor, α represents the learning rate, and r represents the current reward.

However, a Q-learning-based system performs poorly in real applications because of the high-dimensional (possibly continuous) state–action space [64]. Calculating the Q-value of each state–action pair in these circumstances becomes computationally impossible as the complexity of the environment rises.

This issue is addressed by a deep Q-network (DQN), which is a networked Q-learning algorithm and DRL method that combines RL with a class of artificial neural networks known as a deep neural network (DNN) [64]. The DNN gets input from the environment as a state, calculates action based on the weights, and communicates it with the environment. The environment verifies the action and provides a positive reward if the action is favorable otherwise penalize it with a negative reward. Furthermore, this reward is utilized to update the DNN's weights to increase its performance.

Nevertheless, DQN's performance does not rely solely on the use of a neural network approximator function. Some learning stability problems prevent neural networks as nonlinear function approximators from converging. A deep double Q-network (DDQN) was presented as a solution to overcome learning stability issues and increase DQN performance [64].

The DDQN method employs a target Q-network to stabilize learning and reduce the variance of the approximated function. Target Q-network only copies the parameters from the original trained Q-network after hundreds or thousand training steps and therefore does not change quickly and allows the algorithm to learn stable long-term dependencies.

The use of experience replay is one of the key concepts in RL. This approach stores the agent's experience (transitions between states in the past and the corresponding actions and rewards) at each recorded time step $\varepsilon = (s, a, r, s^{'})$ in a replay memory to be able to calculate the loss correctly at any time step in the future. The weight updates for DQN training are then performed using minibatches of experiences picked at random Q-learning, with experience replay offering many advantages over Q-learning's usual form. For starters, each step of experience is potentially used in many weight updates, which allows for greater data efficiency. Second, learning directly from consecutive samples is inefficient, owing to the strong correlations between the samples; randomizing the samples breaks these correlations and therefore reduces the variance of the updates [64, 65].

The exploration vs. exploitation dilemma is crucial in RL [61]. To converge to the optimum policy, the learning algorithm must try out all possible alternatives. Exploitation means taking action that appears best according to the knowledge already learned, whereas exploration means trying new things out in the hope that better actions will be found. To maximize the reward, the agent has to exploit and choose the best action that he knows. However, exploitation does not ensure the greatest long-term solution because the actions chosen may not be the best ones, and there can be other unexplored actions that result in better long-term reward. The agent must do a trade-off between acting optimally in the context of current knowledge and acting to obtain more knowledge. Figure 1 shows the proposed framework. Agent $\boldsymbol{A}$ selects an action out of two actions, which are a_0 and a_1, and it selects an action a_0 for the normal state and an action a_1 for the anomalous state. The state denotes the trace in our framework, so the term "trace" and "state" are used interchangeably throughout the paper. The environment $\boldsymbol{E}$ is composed of a trace sampling function $\boldsymbol{S}$ and an extrinsic reward function $\boldsymbol{X}$.

Agent $\boldsymbol{A}$ in each time step t interacts with the environment. The trace sampling function $\boldsymbol{S}$ picks a trace from D_{la} and D_u alternately to train the agent. When the agent receives the trace or state s_t, it takes action a_t and then receives an extrinsic reward r_e from the extrinsic reward function $\boldsymbol{X}$. The agent also receives an intrinsic reward r_i from a VAE trained on the unlabeled dataset, which rewards the agent based on the abnormality of the trace to encourage the unsupervised exploration of D_u for identifying novel unlabeled anomalous traces. During the training phase, the agent is trained to maximize the sum of those two rewards.

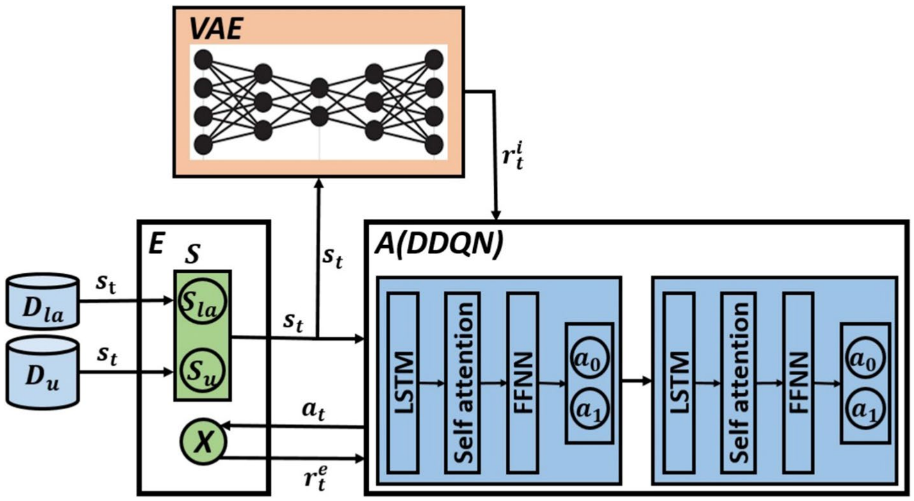

Fig. 1 The proposed variational auotencoder-based weakly supervised deep reinforcement learning framework

The main components of the proposed framework are described as follows:

A) *Agent A*

The internal architecture of the agent ***A*** is illustrated in Fig. 2. The goal of the agent in our framework is to learn an optimal policy that maximizes the discounted accumulated rewards. We use the state-action value function (Q-value function) for this purpose.

During training, the agent's objective is to learn an optimal action-value function (i.e., Q-value function), which can be approximated as:

$$Q^*(s,a) = \text{argmax}_\pi \text{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi] \tag{5}$$

This is the maximum expected return starting from a state s, taking the action $a \in \{a_0, a_1\}$, and thereafter following a behavior policy π. The return is defined as the sum of rewards r_t discounted by a factor γ at each time step t.

In this work, we choose DDQN to learn the optimal Q-value function $Q^*(s,a)$ due to its aforementioned advantages [54, 66, 67]. DDQN leverages deep neural networks as the function approximator with the parameters θ: Q (s, a; θ) = Q^* (s, a) and learns the parameters θ by minimizing the loss iteratively as follows:

$$L_i(\theta_i) = \text{E}_{s,a,r,s' \sim U(\varepsilon)}\left[\left(r + \gamma \max_{a'} Q\left(s', a'; \theta^-\right) - Q(s,a;\theta_i)\right)\right] \tag{6}$$

where, Ɛ is a group of the agent's learning experiences with each state or trace stored as (s, a, r, s'). Minibatches of the traces are randomly taken from the stored experience, and the loss is calculated using these batches.θ_i are the parameters of the main Q-network, and θ_i^- are the parameters of the target network at iteration i. θ_i^- are updated with θ_i every N steps. The gradient update of the main network is given as follows:

$$\nabla_{\theta_i} L_i(\theta_i) = \text{E}_{s,a,r,s' \sim U(\varepsilon)}[(r + \gamma \max_{a'} Q\left(s', a'; \theta^-\right) - Q(s,a;\theta_i)\nabla_{\theta_i} Q(s,a;\theta_i))] \tag{7}$$

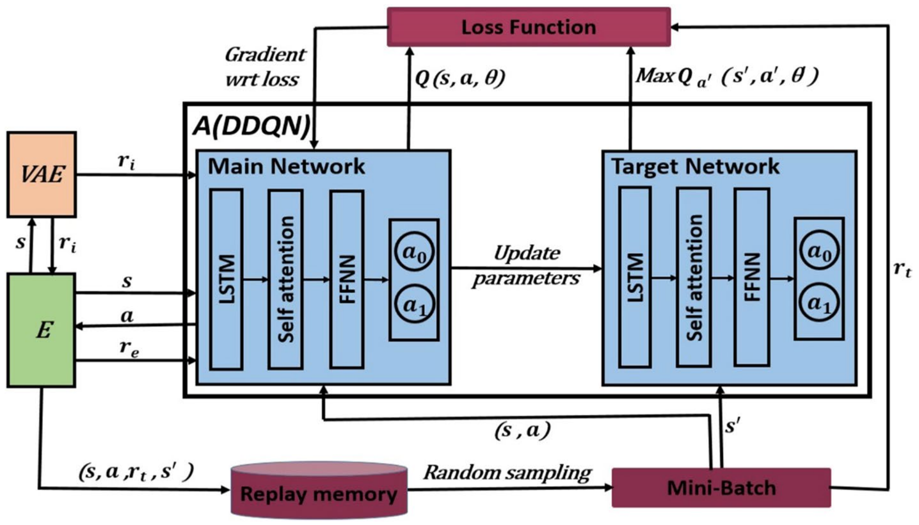

Fig. 2 The DDQN agent of the proposed deep reinforcement learning framework

Business process data has a special nature; it is a sequential data with temporal properties, it is represented as sequences with various dependencies between the activities. For this reason, we chose to use RNNS to model the business process data. RNNs can model sequential temporal data more effectively than most other networks because RNNs can maintain the temporal information through the recurrence mechanism, which feeds the current recurrent layer's output back to input layer, thereby including each current output to the subsequent input and forming a temporal chain of causality by maintaining an internal state (i.e. "memory") [68, 69]. LSTM is chosen in this study to be the core of the Q-networks of our DRL model because it is one of the most robust of the RNN architectures [69, 70]. Additionally, many studies have proven that LSTMs are excellent at modeling the sequential data and capturing its local temporal properties [23, 29–41, 47, 48].

Transformers were recently proposed by Vaswani et al. [71] as an architecture for modeling sequences. Transformers have been shown to perform well in some sequence modeling tasks such as text translation [71] and image classification [72, 73]. However, these tasks all have one thing in common; they are inherently non-temporal. Transformers rely solely on attention mechanisms, and eliminates RNN component or the recurrence mechanism. Many studies have proved that the lack of recurrence modeling prevents transformers from preserving the temporal information and capturing several important properties of the input sequence [74–76].

To achieve the most robust modeling for the business process data and to allow the LSTM network to model the long-term dependencies more efficiently, we combined the LSTM with an attention mechanism.

When modeling the data with an attention component, we are attending more or less to portions of a sequence. Therefore, if the sequence is very long, the model degrades to attention only due to the LSTM's decreased ability to model time-steps too far in the past [69, 77].

Self-attention, also known as intra-attention, is a type of the attention mechanisms that relate different positions of a sequence in order to model dependencies between different parts of the sequence [71].

The self-attention mechanism applied in this study is the same as the self-attention mechanism applied in the transformers, which is the central mechanism of the transformer architecture.

Figure 3 shows the internal architecture of the main and the target networks of the DDQN agent. Both networks are composed of an LSTM network, self-attention layer and a feedforward neural network (FFNN) for outputting two estimated Q-values associated with taking actions a_0 and a_1, respectively, for each input trace. The agent selects the action a_i with the highest $Q(a_i)$. In both cases, the agent feeds the selected action back to the environment.

As shown in Fig. 3, The main network receives a trace or a sequence of events $s = (e_1, e_2, \ldots, e_n)$ as input, where n is the trace length. This input sequence is passed to the LSTM network. Then, LSTM produces the hidden vector sequence $h = (h_1, h_2, \ldots, h_n)$ and output $y = (y_1, y_2, \ldots, y_n)$ of the same length, by iterating the following equations from $t = 1$ to n.

$$h_t = H(W_{eh} e_t + W_{hh} h_{t-1} + b_h) \tag{8}$$

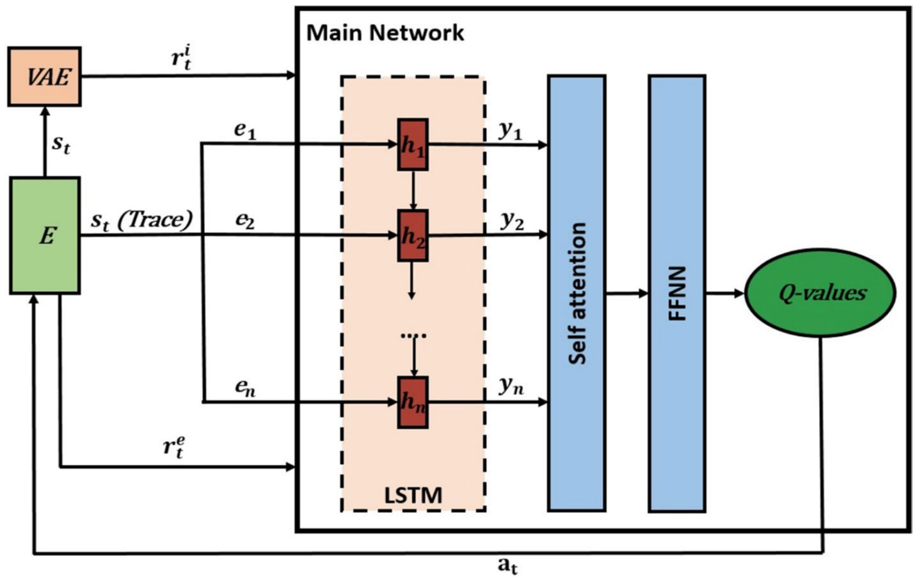

Fig. 3 The main network architecture of the proposed DRL agent

$$y_t = W_{hy}h_t + b_y. \tag{9}$$

where W represents the weight matrices, b represents the bias vectors, and H is the hidden layer function. The self-attention layer takes this hidden vector sequence as input and gives a sequence of context vectors, which are determined as weighted sum of all input vectors.

For this purpose, the relevance of the pairwise hidden states of the LSTM h_i and h_j of dimension d_h is determined using the scaled dot-product [71]. A softmax function is applied to this determined score, which is used to determine the weights of the values within the input vector:

$$Score(h_i, h_j) = Softmax\left(\frac{(h_i)^n h_j}{\sqrt{d_h}}\right) \tag{10}$$

Then, the determined context vectors are passed to the FFNN for predicting the Q-values.

B) *Environment E*

An environment $\boldsymbol{E}$ is created to enable exploitation of D_{la} and exploration of D_u by the agent. $\boldsymbol{E}$ is composed of a trace sampling function $\boldsymbol{S}$ and an extrinsic reward function $\boldsymbol{X}$.

- Trace sampling function S

The sampling function S is composed of two sampling functions, the random sampling function $\boldsymbol{S_{la}}$ and the distance-based sampling function $\boldsymbol{S_u}$, which we formulate

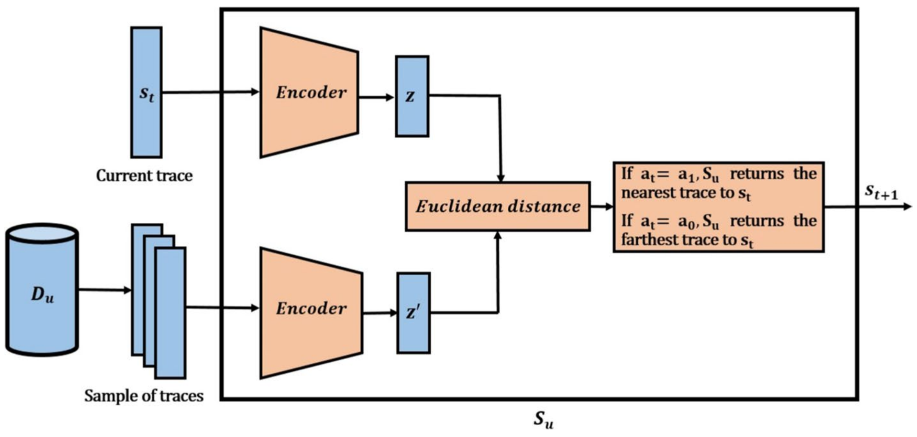
s_t
Current trace
D_u
Sample of traces
Encoder
Encoder
z
z'
Euclidean distance
If $a_t = a_1$, S_u returns the nearest trace to s_t
If $a_t = a_0$, S_u returns the farthest trace to s_t
s_{t+1}
S_u

Fig. 4 The distance-based sampling function $\mathbf{S_u}$ mechanism

for treatment of the imbalanced dataset problem. $\boldsymbol{S_{la}}$ samples the subsequent trace s_{t+1} randomly from D_{la} to allow the equivalent opportunity for each labeled anomalous trace to be exploited. $\boldsymbol{S_u}$ samples s_{t+1} from D_u according to the Euclidean distances between the current trace $\boldsymbol{s_t}$ and the other D_u traces in the latent space of the variational autoencoder to enable the agent to explore D_u in an efficient and effective way. $\boldsymbol{S_u}$ is defined as:

$$S_u(s_{t+1}|s_t, a_t; \theta) = \begin{cases} \text{argmin}\, d(s_t, s; \theta)\ \textit{if}\ a_t = a_1 \\ \text{argmax}\, d(s_t, s; \theta)\ \textit{if}\ a_t = a_0 \end{cases} \tag{11}$$

where, $s \in S$ and S is a random subset of traces from D_u, θ are the parameters of the encoder, and d returns the Euclidean distance between s_t and s in the latent space of the variational autoencoder.

$\boldsymbol{S_u}$ returns the closest trace to the current trace when the agent selects the anomaly action a_1. This permits the agent to investigate traces that are comparable to the suspicious anomalous traces. It also returns the farthest trace to the current trace when the agent selects the normal action a_0. In so doing, the agent can explore more possible anomalous traces. The proximity is computed based on the Euclidean distances between the encoded traces in the latent space of the VAE as shown in Fig. 4.

The proximity computation is performed on a subset of D_u for efficiency reasons. The used subset is 15% of D_u. By this way, $\boldsymbol{S_u}$ obtains anomaly samples more than normal samples without replacement so the proposed model overcome the problem of imbalanced data. Both of the sampling functions $\boldsymbol{S_{la}}$ and $\boldsymbol{S_u}$ are used with the probability 0.5 to make a balance between the exploration and the exploitation.

There are other sampling methods that have been developed to handle the imbalanced dataset problem, such as random over sampling (ROS), random under sampling (RUS), and synthetic minority over-sampling technique (SMOTE). The ROS method replicates a set of minority samples to obtain a balanced dataset [78]. The drawback of this technique is that it does not always improve minority class predictions because making exact copies of minority class samples may lead to the overfitting problem as the decision region becomes too specific.

In contrast, the RUS method balances the dataset by randomly removing some samples from the majority class [78]. The shortcoming of this technique is that potentially useful majority samples are dropped out during the process. Instead of duplicating a set of samples from minority class as in the ROS method or, removing samples from the majority class as in the RUS method, the SMOTE method overcomes the imbalance property by generating synthetic minority samples. This technique arbitrarily interpolates new minority samples in between several samples of a given minority group, which can be found through K-nearest neighbors (KNN) [79, 80].

In the SMOTE method, for each sample in the minority classx_i, k nearest neighbors (samples) are randomly selected, and a synthetic sample can be generated as follows,$x_{new} = x_i + r_{ij} * \delta = x_i + (\widehat{x_{ij}} - x_i) * \delta$, where x_i is a minority class sample, $\widehat{x_{ij}}$ is one of the k-nearest neighbors forx_i, j=1, 2,..., k, k is the number of chosen neighbors, $\delta \in [0, 1]$ is a random number, and x_{new} is a point (sample) along the line segment joining x_i and$\widehat{x_{ij}}$. In other words, a synthetic sample is created by randomly selecting one of the k nearest neighbors $\widehat{x_{ij}}$, multiplying a random number δ by the corresponding feature vector difference r_{ij} and then adding this vector to x_i [79].

An example of the SMOTE method is shown in Fig. 5 [79]. Here, assuming k=4, the circles represent the majority class samples and the squares represent the minority class samples. The green squares represent the generated samples along the line between x_i and $\widehat{x_{ij}}$.

- *Extrinsic reward function X*

The reward function X gives an extrinsic reward r_e to the agent as follows:

$$r_e = x(s, a) = \begin{cases} 1 & if\ a = a_1, s \in D_{la} \\ 0 & if\ a = a_0, s \in D_u \\ -1 & \text{Otherwise} \end{cases} \tag{12}$$

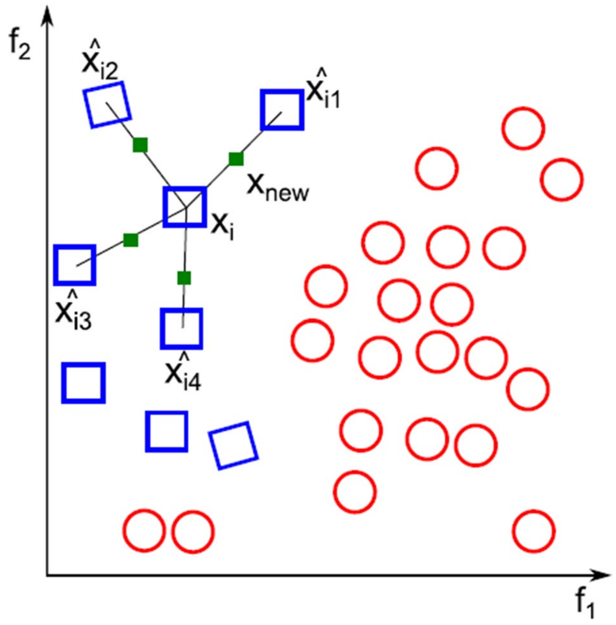

Fig. 5 An example of the SMOTE method

This function gives the agent a positive reward only when the agent correctly labels the known anomalous trace as an anomaly and penalizes any misclassification of the labeled group with a value of − 1. Therefore, this function encourages the agent to fully exploit the labeled dataset D_{la}. In case ofD_u, the agent is neutral. VAE will take over the task of encouraging the agent in the exploration of the unlabeled datasetD_u.

C) *Variational autoEncoder*

In addition to the extrinsic reward r_e, the agent receives an intrinsic reward r_i from the VAE to encourage the unsupervised exploration of the potential anomaly traces in the unlabeled datasetD_u. The VAE used in the proposed framework is shown in Fig. 6.

The reconstruction error that the autoencoder provides for each trace is an indicator of the abnormality of that trace, and hence, in the proposed framework this error is used as the intrinsic reward.

The loss function of the VAE is computed from the reconstruction loss or expected negative log-likelihood of the ith datapoint in the first term of Eq. 10. The expectation is made with respect to the encoder's distribution across the representations by taking a few samples. This term encourages the decoder to learn data reconstruction using samples from the latent distribution. A large reconstruction error indicates that the decoder achieved poor reconstruction of the data. The goal is to minimize this error such that $x \approx x'$. The second term represents the Kullback–Leibler divergence between the encoder's distribution $q_\theta(z|x)$ and $p(z)$. This divergence measures the amount of information lost when q is used to represent a prior over z and encourages its values to be Gaussian [81].

$$L_i(\theta, \varphi) = -\mathrm{E}_{z \sim q_\theta(z|x_i)}\left[\log p_\varphi(x_i|z)\right] + KL(q_\theta(z|x_i)||p(z)) \tag{13}$$

The difference between the original input x and the reconstructed input x' is referred to as the reconstruction error, which is calculated as:

$$||x - x'||^2 \tag{14}$$

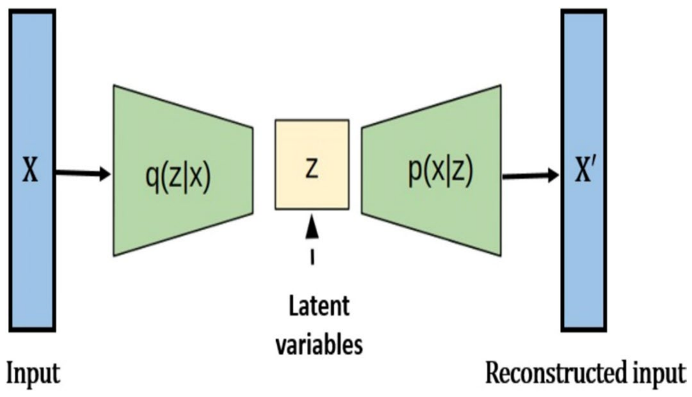

Fig. 6 The Variational AutoEncoder framework

The reconstruction error is rescaled into the range [0, 1] by applying the Min–Max normalization method; therefore, the probability of an abnormality occurring for the trace increases with increasing value of r_i. To account for both the exploitation of D_{la} and the exploration of D_u, the total reward the agent receives for each trace is defined as follows:

$$r_t = r_e + r_i \tag{15}$$

Figure 7 illustrates the method of calculating the total reward from the extrinsic and the intrinsic rewards in different cases.

After the training phase, the optimal Q-value function $Q(s, a, \theta^*)$ is obtained. Subsequently, in the inference stage where a given a test trace s' is considered, the proposed model outputs two Q-values associated with taking actions a_0 and a_1, respectively, for this trace. The model selects the action a_i with the highest $Q(a_i)$. See Appendix for algorithmic description of the proposed approach.

Experimental evaluation

Dataset preparation

We split the synthetic and real-life datasets into training and test sets, with 80% in the training set and the other 20% in the test set. Then, we split the training dataset into a small dataset D_1 and large dataset D_2 with $M_1 = 0.5\%$ and $M_2 = 99.5\%$, respectively. We added artificial anomalies to all traces in D_1 to be the labeled anomaly dataset D_{la}. The unlabeled dataset D_u was produced by adding artificial anomalies (contamination rate $C = 2\%$) to D_2 because anomalous traces are rare traces in business processes and by

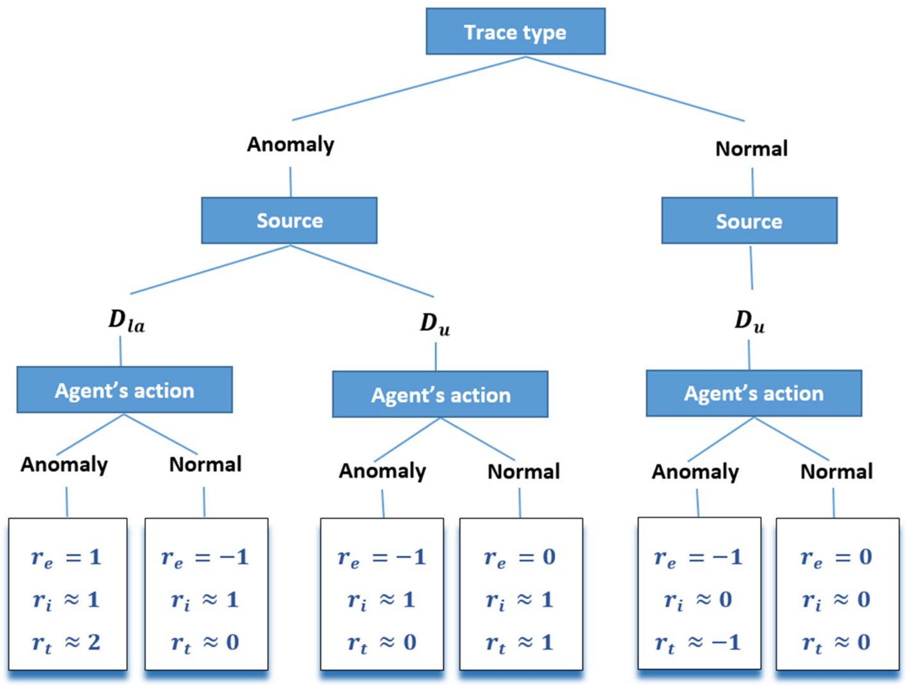
Trace type
Anomaly
Normal
Source
Source
D_{la}
D_u
D_u
Agent's action
Agent's action
Agent's action
Anomaly
Normal
Anomaly
Normal
Anomaly
Normal
$r_e = 1$, $r_i \approx 1$, $r_t \approx 2$
$r_e = -1$, $r_i \approx 1$, $r_t \approx 0$
$r_e = -1$, $r_i \approx 1$, $r_t \approx 0$
$r_e = 0$, $r_i \approx 1$, $r_t \approx 1$
$r_e = -1$, $r_i \approx 0$, $r_t \approx -1$
$r_e = 0$, $r_i \approx 0$, $r_t \approx 0$

Fig. 7 Total reward calculation mechanism

retaining the remaining normal traces. This means that the number of labeled anomalies in D_{la} (i.e., known anomalies) accounts for only $U = 25\%$ of the unlabeled anomalies in D_u (i.e., unknown anomalies) and $V = 20\%$ of all anomalies in the training set to mimic the limited number of anomaly samples available in real life business process datasets. These ratios are clarified in Eq. 5. In addition, artificial anomalies were added to 2.5% only of the testing set.

$$\boldsymbol{V} = \frac{\boldsymbol{P} * \boldsymbol{U} \in \boldsymbol{C}}{\boldsymbol{P} * \boldsymbol{U} * \boldsymbol{C} + \boldsymbol{P} \in \boldsymbol{M}_2 * \boldsymbol{C}} \tag{16}$$

$$\boldsymbol{V} = \frac{\boldsymbol{U}}{\boldsymbol{U} + \boldsymbol{M}_2}$$

where P is the fraction of the samples of the dataset comprising the training set in our case.

Experimental scenarios

For each training dataset D composed of D_{la} andD_u, D_{la} is a small set of labeled anomalous traces belonging to a subset of the following anomaly classes $\{C_1, C_2, C_3\}$, whereC_1 ,C_2, and C_3 refer to the skip, rework, and switch anomaly classes, respectively, whereas D$_u$ is an unlabeled dataset composed of normal traces and some anomalous traces from the whole set of anomaly classes. The performance of the proposed model was assessed using four scenarios as follows:

- All scenario The artificial anomalies that we add to D_{la} are from all three known anomaly classes (skip, rework, and switch), and the artificial anomalies added to D_u and the test set are from all three known anomaly classes as well.
- *Rework* scenario The artificial anomalies added to D_{la} are from only the known rework anomaly class; whereas the artificial anomalies added to D_u and the test set belong to all three known classes of anomalies.
- *Skip* scenario The artificial anomalies added to D_{la} are from only the known skip anomaly class; whereas the artificial anomalies added to D_u and the test set belong to all three known classes of anomalies.
- *Switch* scenario The artificial anomalies added to D_{la} are from only the known switch anomaly class; whereas the artificial anomalies added to D_u and the test set belong to all three known classes of anomalies.

Competing methods

We evaluated the proposed model by comparing it to four ML models and one process mining model. Three of the ML models were unsupervised models: denoising autoencoder [21], BINET [5], and VAE [22], and the fourth model was semi-supervised: Bi-LSTM VAE [23].

The process mining model was built in this study using a process mining library called Process Mining for Python (PM4Py) [82]. The model performs a process discovery process to generate the process model of a given dataset and then perform the conformance checking process to measure the conformance between the process model and the contaminated dataset. In this study, the Heuristics Miner [83] was used to execute the process discovery process.

Implementation setup

All implementations were performed using Python, with a denoising autoencoder and BINET taken from the authors' available implementation source codes [5, 21]. Regarding the VAE and Bi-LSTM VAE models, we preserved the original architecture and hyperparameters employed in [22, 23] to unify conditions, thereby ensuring fairness when conducting comparisons.

The main libraries used in this work are defined as follows [84]:

- *PyTorch* is an open source machine learning library used for developing and training neural network based deep learning models.
- *Scikit-learn* is an open-source Python library that implements a set of machine learning, preprocessing, cross-validation, and visualization algorithms through a unified interface.
- *Imbalanced-learn* is a python package that provides a set of re-sampling techniques commonly used on datasets showing strong between-class imbalance.
- *Numpy* is a python library for defining a multi-dimensional arrays and associated fast math functions that operate on it.
- *Collections* A Python built-in module that implements special container datatypes that provide alternatives to Python's general purpose built-in containers such as dict, list, set, and tuple.
- *Random* A built-in Python module used for generating random numbers within a given range.
- *Matplotlib* is the most common data exploration and visualization library used for creating basic graphs like line plots, histograms, scatter plots, bar charts, and pie charts.

The VAE used in the proposed model was trained for a fixed number of 500 epochs using a mini batch size of 32. As an optimizer, Adam optimizer is used with a learning rate of 0.001. Both the encoder and the decoder consist of an input and an output layer with linear units, and two hidden layers with rectified linear units. The size of the input layer in the encoder is adapted based on the dataset. The number of neurons in this layer is set to the trace length multiplied by the number of unique activities in the dataset. For example, the synthetic dataset 1 contains 12 different activities and the maximum length of all traces in the event log is 14. After the padding, every trace will have a fixed size of 14, and each activity is encoded by a 12-dimensional one-hot vector, yielding a size of 168 for the one-hot vector of each trace.

The number of neurons in the decoder input layer and the number of neurons in the hidden layers and the output layers are listed in Table 2. The activation functions used in the encoder and the decoder are also provided in the table.

The proposed DRL model is trained with 15 episodes, and each episode consists of 2,000 steps using a mini batch size of 32. The target network is updated every five episodes.

Gamma γ or the discount factor is a reflection of how we value our future reward. We utilized the most common gamma value, which is 0.99 because if we select gamma value = 0, this would indicate that we are going for a greedy policy in which the learning agent has no regard for the future. In contrast, selecting the gamma value of 1 would indicate that we have no preference regarding when we should receive the reward; that is, if we receive a reward in time step 4 or time step 8, for instance, it will be of equal value to our agent. We would prefer to have something in the middle to find the optimal policy.

The Epsilon ∈ used in training phase decreased from 1.0 to 0.1. Epsilon is the greedy exploration hyperparameter. We would want the agent to move randomly, since our initial Q values are random, thus we do not want the agent to strictly follow it. Therefore, at first, we would want our agent to move randomly, and as our Q values actually become useful, in that we hopefully have some idea of where rewards are, then we want to search those paths more often, eventually to the point where the Q values converge. Thus, as we go through the episodes, we want to follow the Q values more and move randomly less. To implement this, we set epsilon = 1.0 at first to move randomly. To lower epsilon, we multiply it by the epsilon Decay = 0.99 until the epsilon reaches to 0.1.

The size of the experience replay memory is set to 100,000. The optimization algorithm used for the DRL model is RMSprop (Root Mean Squared Propagation) optimizer with a learning rate of 0.001. The Q-network hyperparameters used in our work are shown in Table 3. The size of the LSTM's input layer is adapted according

Table 2 VAE hyperparameter settings

	Encoder	**Decoder**
Input layer size	Trace length * Number of activities	5
No. of hidden layers	2	2
1st hidden layer size	Input layer size/2	Input layer size/4
2nd hidden layer size	Input layer size/4	Input layer size/2
Output layer size	5	Encoder input layer size
Activation functions	Relu	Relu

Table 3 Q-network hyperparameter settings

	LSTM	**Self-attention**	**FFNN**
Input layer size	Trace length * number of activities	Trace length * 512	Trace length * 512
No. of hidden layers	1	0	1
hidden layer size	512	–	Trace length * 1024
Output layer size	Trace length * 512	Trace length * 512	2
Activation functions	Relu	Softmax	Relu

to the dataset like the encoder's input layer size. We have found experimentally that embedding FFNN input into a higher space yielded better results. The choice of the VAE and the Q-network hyperparameters is made empirically after conducting preliminary experiments with different hyperparameter values but we found that they work sufficiently well with these values.

Results and analysis

To evaluate our model and all the other competing models, we run each experiment in this study multiple times with 5 random initialization seeds and reported the average and the standard deviation in Tables 4, 5, 6 and 7.

To assess the performance of our model and the competing models, we used two performance measures, namely, balanced accuracy and F-measure (F1-score). The F1-score represents the mean of precision and recall [85]. It is the most commonly used metric in the anomaly detection literatures [5, 21, 23, 86, 87], whereas balanced accuracy is defined as the average of sensitivity and specificity [88]. We chose it as a performance measure because it has the ability to capture how well the examined model can deal with the imbalanced data classification problem. Balanced accuracy has been used in several imbalance learning articles [89–91].

We evaluated the proposed DRL model when the core of the main network is standalone LSTM, LSTM combined with self-attention, against self-attention alone, i.e. the transformer to achieve the optimal modeling for the business process data.

The transformer used for comparison is directly built from the transformer described in [71]. However, transformers were designed to work for sequence to sequence (seq2seq) tasks such as automatic speech recognition or language translation. However, in this work there is no sequence to sequence task. Instead, we need to predict the Q values given the input trace. Therefore, we use only the transformer

Table 4 F1-score and balanced accuracy results (%) of the proposed model with LSTM only, LSTM combined with self-attention, against the transformer over all datasets using the first scenario

Datasets	Balanced accuracy			F1-score		
	LSTM	LSTM w/Self Att	Transformer	LSTM	LSTM w/Self Att	Transformer
DS1	95.67 ± 0.02	98.90 ± 0.07	94.34 ± 0.02	92.64 ± 0.11	97.41 ± 0.05	92.54 ± 0.15
DS2	95.46 ± 0.01	97.02 ± 0.05	93.88 ± 0.05	91.90 ± 0.09	95.60 ± 0.07	91.10 ± 0.04
BPIC12	85.96 ± 0.01	93.37 ± 0.06	83.24 ± 0.03	82.32 ± 0.04	87.56 ± 0.01	79.96 ± 0.02
BPIC17	90.61 ± 0.15	95.04 ± 0.04	88.76 ± 0.04	87.43 ± 0.04	91.40 ± 0.05	84.23 ± 0.10

Table 5 The balanced accuracy results (%) of the proposed model with our sampling method, against the SMOTE sampling method over all datasets using the first scenario

Datasets	Our sampling method	SMOTE
DS1	98.90 ± 0.07	93.06 ± 0.06
DS2	97.02 ± 0.05	90.48 ± 0.07
BPIC12	93.37 ± 0.06	84.53 ± 0.08
BPIC17	95.04 ± 0.04	88.91 ± 0.12

Table 6 Balanced accuracy results (%) over all datasets using all the scenarios

Data		Proposed model	Denoising Autoencoder [21]	Variational autoencoder [22]	BINET [5]	Bi-LSTM VAE [23]	Process mining model
Datasets	Scenarios						
DS1	All	98.90 ± 0.07	86.62 ± 0.12	89.38 ± 0.18	93.67 ± 0.15	94.95 ± 0.14	92.98
	Rework	95.33 ± 0.07	82.89 ± 0.05	84.51 ± 0.12	90.02 ± 0.07	90.86 ± 0.05	88.23
	Skip	94.41 ± 0.02	81.30 ± 0.01	83.44 ± 0.08	88.45 ± 0.14	89.73 ± 0.07	86.79
	Switch	91.09 ± 0.08	71.47 ± 0.14	72.32 ± 0.06	79.63 ± 0.03	80.24 ± 0.09	78.65
DS2	All	97.02 ± 0.05	84.13 ± 0.15	87.27 ± 0.15	90.54 ± 0.08	93.71 ± 0.15	90.12
	Rework	93.18 ± 0.02	81.67 ± 0.08	82.64 ± 0.09	88.33 ± 0.01	89.87 ± 0.05	86.49
	Skip	92.72 ± 0.04	80.21 ± 0.01	82.04 ± 0.13	85.76 ± 0.00	89.04 ± 0.12	85.87
	Switch	89.61 ± 0.02	57.86 ± 0.05	69.10 ± 0.02	77.15 ± 0.14	77.93 ± 0.09	74.52
BPIC12	All	93.37 ± 0.06	58.03 ± 0.01	60.22 ± 0.05	64.68 ± 0.01	76.00 ± 0.01	74.98
	Rework	88.12 ± 0.08	53.42 ± 0.03	53.58 ± 0.14	61.32 ± 0.12	71.82 ± 0.03	70.63
	Skip	86.98 ± 0.05	52.34 ± 0.09	52.13 ± 0.03	59.93 ± 0.08	69.67 ± 0.00	69.84
	Switch	83.02 ± 0.03	42.97 ± 0.18	42.98 ± 0.17	48.75 ± 0.09	60.40 ± 0.14	58.60
BPIC17	All	95.04 ± 0.04	65.33 ± 0.02	63.46 ± 0.04	67.42 ± 0.19	78.64 ± 0.09	77.42
	Rework	91.84 ± 0.00	61.29 ± 0.17	59.78 ± 0.09	63.33 ± 0.07	74.97 ± 0.14	73.39
	Skip	90.69 ± 0.02	59.81 ± 0.03	57.91 ± 0.03	62.79 ± 0.01	72.91 ± 0.08	71.28
	Switch	87.71 ± 0.12	51.34 ± 0.01	49.11 ± 0.09	52.51 ± 0.15	63.86 ± 0.03	60.74

Table 7 F1-score results (%) over all datasets using all the scenarios

Data		Proposed model	Denoising Autoencoder [21]	Variational autoencoder [22]	BINET [5]	Bi-LSTM VAE [23]	Process mining model
Datasets	Scenarios						
DS1	All	97.41 ± 0.00	83.12 ± 0.18	85.09 ± 0.12	90.30 ± 0.08	91.22 ± 0.08	89.00
	Rework	96.66 ± 0.07	81.58 ± 0.03	83.10 ± 0.18	88.50 ± 0.05	89.30 ± 0.02	86.85
	Skip	95.47 ± 0.01	79.19 ± 0.06	81.34 ± 0.02	85.91 ± 0.01	86.84 ± 0.06	84.10
	Switch	92.51 ± 0.12	68.84 ± 0.18	70.63 ± 0.02	76.74 ± 0.03	77.40 ± 0.01	75.68
DS2	All	95.60 ± 0.07	80.69 ± 0.08	83.15 ± 0.04	87.42 ± 0.17	90.46 ± 0.08	86.12
	Rework	92.22 ± 0.05	79.71 ± 0.02	81.32 ± 0.03	86.64 ± 0.08	88.49 ± 0.12	84.91
	Skip	91.67 ± 0.07	78.22 ± 0.04	79.47 ± 0.06	82.99 ± 0.07	86.98 ± 0.01	83.23
	Switch	88.06 ± 0.05	54.93 ± 0.17	66.89 ± 0.12	74.86 ± 0.18	75.58 ± 0.15	72.70
BPIC12	All	87.56 ± 0.01	54.23 ± 0.12	55.97 ± 0.06	61.38 ± 0.05	72.11 ± 0.12	71.23
	Rework	85.84 ± 0.14	52,00 ± 0.14	52.45 ± 0.01	59.30 ± 0.14	69.91 ± 0.18	68.89
	Skip	83.12 ± 0.08	49.87 ± 0.18	50.66 ± 0.03	56.89 ± 0.05	67.44 ± 0.01	66.68
	Switch	80.39 ± 0.02	40.20 ± 0.03	41.71 ± 0.18	45.43 ± 0.08	57.50 ± 0.02	56.74
BPIC17	All	91.40 ± 0.05	57.51 ± 0.15	58.54 ± 0.06	64.76 ± 0.12	75.63 ± 0.05	73.64
	Rework	89.98 ± 0.07	55.14 ± 0.02	56.86 ± 0.07	63.94 ± 0.04	72.95 ± 0.08	70.87
	Skip	89.03 ± 0.01	52.67 ± 0.05	54.43 ± 0.12	60.46 ± 0.02	69.75 ± 0.03	67.44
	Switch	86.78 ± 0.12	42.29 ± 0.03	46.95 ± 0.01	50.21 ± 0.03	58.42 ± 0.01	49.26

encoder module to extract the features from input trace which are further used to predict the Q-values. As shown in Fig. 8, the encoder module consists of the multi-head attention module, normalization and feed-forward components. The multi-head attention module refers to the self-attention layer. More details about the transformer architecture can be found in [71].

The balanced accuracy and F1-score results are presented in Table 4. The results show that the performance of the proposed DRL model with stand-alone LSTM

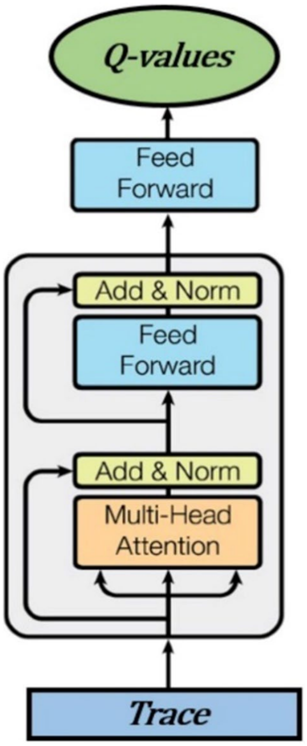

Fig. 8 Transformer encoder architecture

exceeds or matches its performance with the transformer. However, the performance of the DRL model with LSTM combined with self-attention is superior to the performance with the transformer on all the datasets.

Furthermore, the results demonstrate that the Q networks made of LSTM combined with self-attention has the capability to outperform the Q networks built using LSTM and transformer and generate better trained agents in our problem.

From this observation, we can conclude that neither self-attention nor recurrence is all we need, but rather it is recurrence combined with self-attention which provides the most robust modeling for the business process data.

To further validate the efficiency of the proposed sampling method to deal with the imbalance problem in the unlabeled set D_u, we conducted more experiments on our DRL model with the SMOTE sampling method. We applied this sampling method to each unlabeled set using the imbalanced-learn python package [92] with default parameters.

We reported the balanced accuracy results in Table 5, since the balanced accuracy has the ability to capture how well the model can deal with the imbalanced data problem. It can be observed that the obtained performance with the proposed sampling method on the four datasets is higher than the performance obtained with the other sampling methods. This is because when we used the SMOTE method, the same number of synthetic minority samples (anomaly traces) were generated for each original minority sample without taking into consideration neighboring samples, increasing the likelihood of overlapping classes. Moreover, the process of generating synthetic anomaly traces is a major source of overgeneralization problems.

The good performance obtained with the proposed sampling method can be explained by its simplicity. This is because we have improved the anomaly class detection without adding new anomaly traces and complexity to the problem itself. Additionally, because it preserved all traces, so no important information is lost, making the decision region broader.

The balanced accuracy and the F1-score results of the comparison between the proposed model and competing methods on the four datasets using the scenarios clarified in section Experimental scenarios are depicted in Tables 6 and 7, respectively. On the basis of the results, the proposed model achieved the best performance in all scenarios. The results also revealed that all of the models performed better in the first scenario, where anomalies from all of the known anomaly classes appeared in the labeled part of the training data. It is reasonable because this scenario gives additional supervision information compared to the other scenarios, allowing the models to anticipate the different types of anomalies seen in the test set.

Figure 9 shows the reconstruction error distributions of the normal, rework, skip, and switch anomalous traces produced by the VAE. It can be observed that the rework anomalies yielded the highest reconstruction errors, followed by the skip anomalies, with the switch anomalies having the lowest reconstruction errors. Furthermore, the reconstruction errors of the normal and switch anomalies were highly comparable and distinct from the reconstruction errors of the rework and skip anomalies, indicating that the normal and switch anomalous traces originated from similar distributions.

These reconstruction errors explained the balanced accuracy and the F1-score results shown in Tables 4 and 5, respectively. The relatively high reconstruction errors of the rework anomalies showed their highly anomalous behavior, and thus, they can provide stronger supervision information to the DRL model than the other anomalies that will

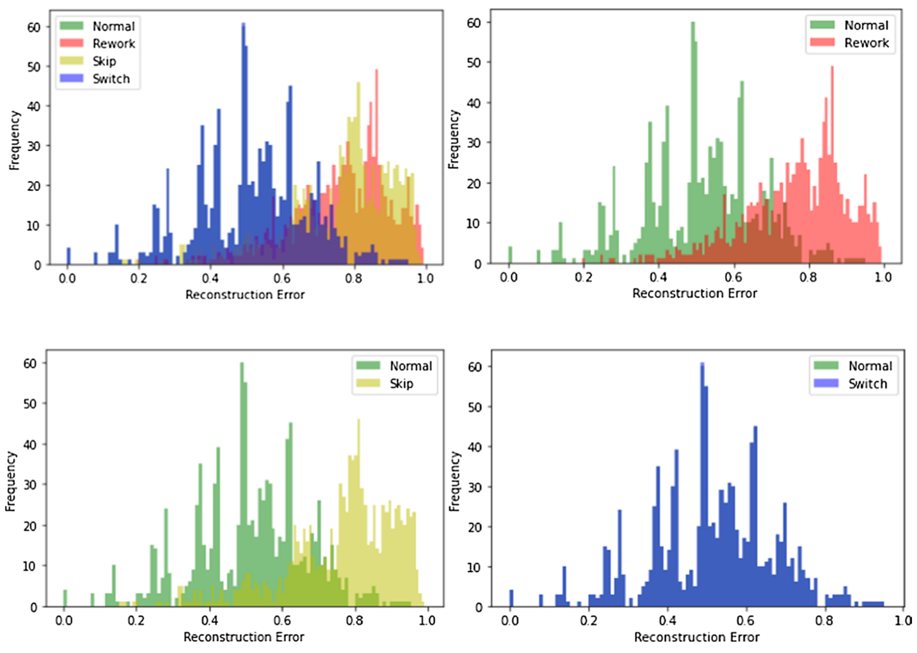

Fig. 9 The reconstruction errors of normal traces, rework, skip, and switch anomalous traces

result in better normality/abnormality learning and will yield good results after the ***All*** scenario results. The reconstruction errors of the skip anomalies were less than the reconstruction errors of the rework anomalies, so they provided less supervision. Thus, learning may be less effective than the learning in the rework scenario. The switch anomalies' reconstruction errors then explained why they provided the worse outcomes. They were close to the normal traces in the distribution and had tiny behavior deviations. Therefore, they sent a poor supervisory signal to the DRL model, and the model will not learn effectively from these anomalies.

Figure 9 also demonstrates that few normal traces assigned relatively high reconstruction errors, which is a problem that the state-of-the-art methods cannot handle. Our approach was built to tackle this problem and detect normal traces even when they have behavior deviations. It may do so using the combined supervisory signals provided by the ensemble of machine learning techniques: LSTM, FFNN and the VAE, integrated in the RL framework. The techniques rely on the labeled anomalous data D_{la} and the unlabeled data D_u. Hence, the model is more capable of separating the normal class from the anomaly classes, as the techniques with various data sources are capable of compensating for the shortcomings of one another. This is further proven by the scatterplots in Fig. 10 and Tables 7 and 8.

The Q-value scatter plots for normal traces; rework, skip, and switch anomalous traces are shown in Fig. 10. The highest anomaly Q-values are obtained for the rework anomalous traces confirming that these traces exhibit the greatest behavior deviation from the normal traces; the second-highest and lowest Q-values are obtained for the skip anomalous traces and the switch anomalous traces.

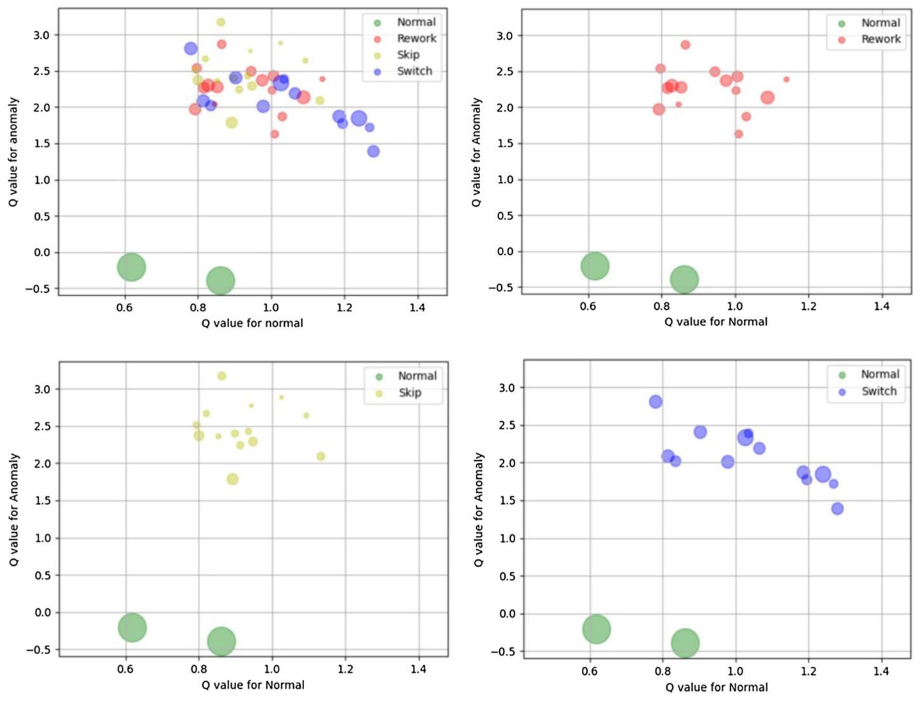

Fig. 10 Q-values scatter plot for normal traces, rework, skip, and switch anomalous traces

Table 8 Inter-class distance matrix of the four classes in the DRL feature space using the average-linkage approach

Classes	Normal	Rework	Skip	Switch
Normal	0.0	8.592794	9.1943846	7.9583769
Rework	8.592794	0.0	0.3946912	0.44282836
Skip	9.1943846	0.3946912	0.0	0.5227502
Switch	7.9583769	0.44282836	0.5227502	0.0

Table 9 Inter-class distance matrix of the four classes in the VAE feature space using the average-linkage approach

Classes	Normal	Rework	Skip	Switch
Normal	0.0	3.3478773	2.941043	1.8407798
Rework	3.3478773	0.0	2.4579616	3.2041183
Skip	2.941043	2.4579616	0.0	3.4259584
Switch	1.8407798	3.2041183	3.4259584	0.0

In addition, the scatter plots of normal traces are quite distinct from the switch anomalous traces in the feature space of our DRL model despite the similarity between the reconstruction errors of the traces. This confirms that, compared with VAE alone, the proposed model is able to learn better separation between normal and switch anomalous traces and is able to predict anomalies even in the case of small behavior deviations. Moreover, the normal traces yield high normal Q-value and low anomaly Q-value, confirming that our model can predict the normal traces, even with behavior deviations.

To verify these observations, we compute the inter-class and the intra-class distances. The inter-class distance is the distance between the classes and the intra-class distance is the distance between the data points in a given class. These distances may be determined through different approaches such as the single-, complete-, average-, and centroid-linkage approaches [93]. We determine the proximity between the classes by using the average-linkage approach to measure the inter-class distances based on Euclidian distances. The inter-class distances between the normal class and the anomaly classes in the DRL model feature space and the VAE feature space are shown in Tables 8 and 9, respectively.

The cells in the above matrices contain the Euclidean distances between the classes. Each class's distance from itself is set to 0 as shown in the main diagonal of the matrices. The inter-class distances between the classes verify that our DRL model achieves a high inter-class separability between the normal class and the other anomaly classes compared to the VAE as the distances between the normal class and the anomaly classes is large compared to these distances in the VAE feature space. In addition, the distances between the anomaly classes in the DRL feature space is small compared to the distances between them in the VAE feature space, which show that our DRL model achieves a high compactness between the anomalies.

Table 10 Intra-class distances of the four classes in the DRL model feature space using the average- and centroid-linkage approaches

Class	Average-linkage	Centroid-linkage
Normal	0.15280631	0.15280631
Rework	0.43417037	0.29623094
Skip	0.4467089	0.33093923
Switch	0.31904405	0.22662014

Table 11 Intra-class distances of the four classes in the VAE feature space using the average- and centroid-linkage approaches

Class	Average-linkage	Centroid-linkage
Normal	5.2491493	4.6055837
Rework	2.543555	1.8195286
Skip	1.9299569	1.3724583
Switch	3.3270447	2.450689

We computed the intra-class distances using the average- and centroid-linkage approaches. The results shown in Tables 10 and 11 represent the distances in the DRL model feature space and the VAE feature space, respectively. These results indicate that the distances within each class in the DRL feature space are significantly smaller than those in the VAE feature space. This shows that the compactness of the classes in the DRL model feature space is better than that of the classes in the VAE space.

Figure 11 shows the scatter point representations of some normal and rework, skip, and switch anomalous traces in the latent space of the VAE. The results shown in this figure suggest that the sampling strategy employed in our work (which is based on the proximity between traces in the latent space of the VAE) is a good choice. That is the rework, skip, and switch anomalous traces are very close in the latent space and are distinct from the normal traces.

Conclusion

This paper presented a DRL model to address the problem of anomaly detection in business processes with a limited number of labeled anomalies available. The specialized DRL framework for business processes weakly supervised anomaly detection is what distinguishes the proposed approach. The proposed model differs from the previous models offered for anomaly detection. Our model can exploit the labeled anomaly data to enhance detection accuracy without limiting the set of anomalies sought to those given anomaly examples and can investigate rare and novel anomalies in the unlabeled data effectively as the VAE used in our model provides a complementary supervisory signal to the improvement learning model. Furthermore, in the core of the DRL model, an LSTM recurrent neural network combined with a self-attention mechanism was employed to handle the problem of long-term dependencies between business process

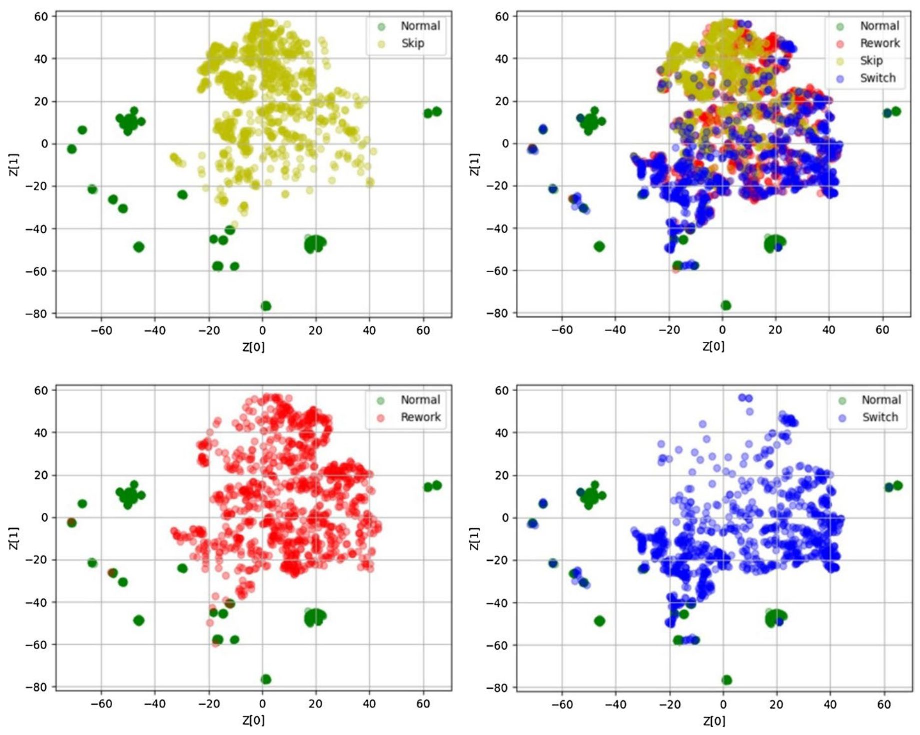

Fig. 11 Scatter plots of normal traces, rework, skip, and switch anomalous traces in the latent space

events. Furthermore, we suggested an effective sampling strategy for overcoming the significant dataset imbalance difficulty that is frequently faced in the field of anomaly detection. The performance of the proposed model was assessed using different scenarios on both real-life dataset and synthetic ones. The experimental results showed the superiority of the proposed model over the state-of-the-art methods. We are interested in expanding the proposed model to characterize the deviation that has occurred in the anomalous business process execution, rather than only detect it, as future work.

Appendix

Algorithm 1 (Training)

Appendix A: Algorithm 1 (Training)

Input: $D = \{ D_{la}, D_u \}$

Output: Q(s, a,θ^*)

```
1.  Initialize main network with random weights $\theta$
2.  Initialize target network with weights $\theta' = \theta$
3.  Initialize the size of experience set E to M
4.  for j = 1 to n_episodes do
5.    Initial first trace $s_1$ by random from $D_u$
6.    for t = 1 to n_steps do
7.        With probability epsilon select a random action $a_t$ from $\{a_0, a_1\}$
8.        Otherwise select $a_t = arg\ max_a$ Q ($s_t$, a, $\theta$)
9.        With probability p the environment returns $s_{t+1}$ randomly from $D_{la}$
10.       Otherwise return $s_{t+1}$ from $D_u$ based on the autoencoder latent space
11.       Calculate extrinsic reward x ($s_{t+1}$)
12.       Calculate intrinsic reward from the autoencoder
13.       Calculate reward $r_t = r_t^e + r_t^i$
14.       Store experience ($s_t, a_t, r_t, s_{t+1}$) in E
15.       Randomly sample a minibatch of experience records from E
16.       Set y = r if episode terminates at this step
17.       Otherwise y= r + $\gamma$ max Q^ ($s_{t+1}$, $a'$, $\theta'$)
18.       Calculate loss (y – Q ($s_t$, a, $\theta$))$^2$
19.       Update the main network weight parameters $\theta$ every N steps
20.       Update the target network weight parameters $\theta'$ every K steps
21.   end for
22. end for
23. return Q values
```

Algorithm 2 (Anomaly detection)

Algorithm 2 (Anomaly detection)

```
Input: Testing data T, $Q(s, a, \theta^*)$
Output: Testing data classification A
for i= 1 to |T| do
    $a^i = arg\ max_a\ Q(s^i, a, \theta^*), s^i \in T$
end for
return A
```

Abbreviations

ML	Machine learning
RL	Reinforcement learning
DRL	Deep reinforcement learning
IRL	Inverse reinforcement learning
VAE	Variational autoEncoder
RNN	Recurrent neural network
LSTM	Long short-term memory network
GAN	Generative adversarial network
IOT	Internet of Things
BPIC	Business Process Intelligence Challenge
DNN	Deep neural network
DQN	Deep Q-network
DDQN	Deep double Q-network
FFNN	Feedforward neural network
ROS	Random over sampling
RUS	Random under sampling
SMOTE	Synthetic minority over-sampling technique
KNN	K-nearest neighbors
PM4Py	Process Mining for Python
RMSprop	Root Mean Squared Propagation

Acknowledgements
Not applicable.

Author contributions
EA, RF, and MS discovered business process anomaly detections by using reinforcement learning. Then, EA conducted the experiments and composed the paper. RF and MS took on a supervisory role and oversaw the completion of the work by reviewing and editing the manuscript. All authors read and approved the final manuscript.

Funding
Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB). The article processing charge is funded by Arabic Academy for Science, Technology and Maritime Transport, Egypt.

Availability of data and materials
The used real-life dataset in this research is publicly available. For the synthetic datasets, readers can contact the author if they want to access the data.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 3 June 2022 Accepted: 28 February 2023
Published online: 12 March 2023

References

1. Sarno R, Sinaga F, Sungkono K. Anomaly detection in business processes using process mining and fuzzy association rule learning. J Big Data. 2020;7(1):1–19.
2. What is Business Process. https://www.comidor.com/knowledge-base/business-process-management-kb/business-process-definition/. Accessed 18 Apr 2020.
3. Jung J, Hur W, Kang SH, Kim H. Business process choreography for b2b collaboration. IEEE Internet Comput. 2004;8(1):37–45.
4. Maaradji A, Dumas M, Rosa ML, Ostovar A. Fast and Accurate Business Process Drift Detection. Proc. of BPM. 2015.
5. Nolle T, Seeliger A, Mühlhäuser M. Binet: multivariate business process anomaly detection using deep learning. In: Lecture notes in computer science international conference on business process management. 2018. p. 271–287.
6. Hao MC, Keim DA, Dayal U, Schneidewind J. Business process impact visualization and anomaly detection. Inf Vis. 2006;5(1):15–27.
7. vander Aalst WMP. Process mining: data science in action. Berlin: Springer; 2016.
8. Aalst W, Medeiros A. Process mining and security: detecting anomalous process executions and checking process conformance. Electr Notes Theor Comput Sci. 2005;121:3–21.
9. Wen L, Aalst W, Wang J, Sun J. Mining process models with non-free-choice constructs. Data Min Knowl Disc. 2007;15(2):145–80.
10. Bezerra F, Wainer J. Anomaly detection algorithms in logs of process aware systems. Proceedings of the 2008 ACM symposium on Applied computing; 2008. p. 951–952.
11. Bezerra F, Wainer J, Aalst W. Anomaly detection using process mining. In Enterprise, business-process and information systems modeling. Springer; 2009. p. 149–161.
12. Swinnen J, Depaire B, Jans M. A process deviation analysis—a case study. Business Process Management Workshops, LNBIP, Springer, Heidelber; 2012; 99: 87–98.
13. Li G, van der Aalst WMP. A framework for detecting deviations in complex event logs. Intell Data Anal. 2017;21(4):759–79.
14. Rogge-Solti A, Kasneci G. Temporal anomaly detection in business processes. In: Business Process Management. Springer; 2014. p. 234–249.
15. Sarno R, Dewandono RD, Ahmad T, Naufal MF, Sinaga F. Hybrid association rule learning and process mining for fraud detection. Int J Comput Sci. 2015;42(2):59–72.
16. Sarno R, Sinaga FP. Business Process anomaly detection using ontology-based process modelling and multi-level class association rule learning. In: International Conference on Computer, Control, Informatics and Its Applications (IC3INA). Bandung; 2015. p. 12–7.
17. Zhu T, Guo Y, Ma J, Ju A. Business process mining based insider threat detection system. In: Xhafa F, Barolli L, Amato F (eds). Advances on P2P, parallel, grid, cloud and internet computing. International conference on P2P parallel grid cloud and internet computing Lecture notes on data engineering and communications technologies. Vol 1. Springer, Cham; 2017. p. 467–478.
18. Hsu PY, Chuang YC, Lo YC, He SC. Using contextualized activity-level duration to discover irregular process instances in business operations. Inf Sci. 2016;391–392:80–98.
19. Folino F, Greco G, Guzzo A, Pontieri L. Mining usage scenarios in business processes: outlier-aware discovery and run-time prediction. Data Knowl Eng. 2010;70:1005–29.
20. Böhmer K, Rinderle Ma S. Multi-perspective anomaly detection in business process execution events. In: On the move to meaningful internet systems: OTM 2016 conferences. Lecture notes in computer science, vol 10033. Springer, Cham; 2016. p. 80–98.
21. Nolle T, Seeliger A, Mühlhäuser M. Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In: Lecture notes in computer science international conference on discovery science. Springer, Berlin; 2016. p. 442–456.
22. Krajsica P, Franczyk B. Variational autoencoder for anomaly detection in event data in online process mining. In: Proceedings of the 23rd International Conference on Enterprise Information Systems; 2021. p. 567–574.
23. Krajsic P, Franczyk B. Semi-supervised anomaly detection in business process event data using self-attention based classification. In: Procedia Computer Science, 25th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems; 2021. p. 39–48.
24. Zhou Z. A brief introduction to weakly supervised learning. Natl Sci Rev. 2018;5:44–53.
25. Liu Z, Miao Z, Zhan X, Wang J, Gong B, Yu SX. Largescale long-tailed recognition in an open world. In: IEEE conference on computer vision and pattern recognition; 2019. p. 2537–2546.
26. Salakhutdinov R, Torralba A, Tenenbaum J Learning to share visual appearance for multiclass object detection. In: IEEE conference on computer vision and pattern recognition (CVPR); 2011.
27. Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. ACM Comput Surv. 2009;41:1–58.
28. Aggarwal CC. Outlier analysis. Berlin: Springer; 2017.
29. Malhotra P, Vig L, Shroff G, Agarwal P. Long short term memory networks for anomaly detection in time series. In: 23rd ESANN—European symposium on artificial neural networks, computational intelligence and machine learning; 2015.
30. Chauhan S, Vig L. Anomaly detection in ECG time signals via deep long short-term memory networks. Proceedings of the IEEE international conference data sci. adv anal (DSAA); 2015. p. 1–7.
31. Munir M, Siddiqui SA, Dengel A, Ahmed S. Deepant: a deep learning approach for unsupervised anomaly detection in time series. IEEE Access. 2019;7:1991–2005.

32. Lopez-Martin M, Carro B, Sanchez-Esguevillas A. Lloret J Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. IEEE Access. 2017;5:18042–50.
33. Ergen T, Kozat SS. Unsupervised anomaly detection with LSTM neural networks. IEEE Trans Neural Netw Learn Syst. 2019;31:3127–41.
34. Guo T, Xu Z, Yao X, Chen H, Aberer K, Funaya K. Robust online time series prediction with recurrent neural networks. In 2016 IEEE international conference on data science and advanced analytics (DSAA), Montreal, QC, Canada. IEEE; 2016. p. 816–825
35. Evermann J, Rehse JR, Fettke P. Predicting process behaviour using deep learning. Decis Support Syst. 2017;100:129–40.
36. Bontemps L, Cao VL, McDermott J, Le-Khac N. Collective anomaly detection based on long short term memory recurrent neural network. In: Lecture notes in computer science international conference on future data and security engineering. Springer; 2016. p. 141–152.
37. Lipton ZC, Kale DC, Elkan C, Wetzell R. Learning to diagnose with LSTM recurrent neural networks. 2016.
38. Buda TS, Caglayan B, Assem H, DeepAD: a generic framework based on deep learning for time series anomaly detection. In: Lecture notes in computer science conference on knowledge discovery and data mining. Berlin: Springer; 2018. p. 577–588.
39. Hundman K, Constantinou V, Laporte C, Colwell I, Soderstrom T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In. Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, ser. Acad Med. KDD'18; 2018. p. 387–395.
40. Du M, Li F, Zheng G, Srikumar V. Deeplog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the acad med ACM SIGSAC conference on computer and communications security; 2017. p. 1285–1298.
41. Tuor A, Kaplan S, Hutchinson B, Nichols N, Robinson S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. 2017.
42. Amarbayasgalan T, Jargalsaikhan B, Ryu KH. Unsupervised novelty detection using deep autoencoders with density based clustering. Appl Sci. 2018;8:1468.
43. Schreyer M, Sattarov T, Borth D, Dengel A, Reimer B. Detection of anomalies in large scale accounting data using deep autoencoder networks. In: CoRR; 2017. p. 1–19.
44. Luo T, Nagarajan SG. Distributed anomaly detection using autoencoder neural networks in WSN for IoT. In: IEEE international conference on communications (ICC);2018. p 1–6.
45. Kieu T, Yang B, Jensen CS. Outlier detection for multidimensional time series using deep neural networks. In: 2018 19th IEEE international conference on mobile data management (MDM); 2018. p. 125–134.
46. Zhou C, Paffenroth RC. Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. Acad Med; 2017. p. 665–674.
47. Malhotra P, Ramakrishnan A, Anand G, Vig L, Agarwal P, Shroff G. LSTM-based encoder-decoder for multi-sensor anomaly detection. 2016.
48. Omeara C, Schlag L, Wickler M. Applications of deep learning neural networks to satellite telemetry monitoring. In: 2018 SpaceOps conference; 2018.
49. Laptev N. Anogen: deep anomaly generator. In: Outlier Detection De-constructed (ODD) Workshop; 2018.
50. Lim SK, Loo Y, Tran NT, Cheung NM, Roig G, Elovici Y. Doping: generative data augmentation for unsupervised anomaly detection with GAN. In: 2018 IEEE international conference on data mining (ICDM). 2018. p. 1122–1127.
51. Li D, Chen D, Goh J, Ng Sk. Anomaly detection with generative adversarial networks for multivariate time series. 2018.
52. Xu X. Sequential anomaly detection based on temporal-difference learning: principles, models and case studies. Appl Soft Comput. 2010;10(3):859–67.
53. Zhong C, Gursoy MC, Velipasalar S. Deep actor-critic reinforcement learning for anomaly detection. 2019.
54. Huang C, Wu Y, Zuo Y, Pei K, Min G. Towards experienced anomaly detector through reinforcement learning. In: Proceedings of the thirty-second aaai conference on artificial intelligence, Riverside, New Orleans. 2018.
55. Yu M, Sun S. Policy-based reinforcement learning for time series anomaly detection. Eng Appl Artif Intell. 2020;95: 103919.
56. Malla DB, Kimura T, Sogabe M, Sakamoto K, Sogabe T. Development of anomaly prediction detection method using bayesian inverse reinforcement. In: Learning 34th annual conference of the Japanese society for artificial intelligence. 2020.
57. Oh M, Iyengar G. Sequential anomaly detection using inverse reinforcement learning. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining; 2019. p. 1480–1490.
58. Burattin A, PLG2: Multiperspective Processes Randomization and Simulation for Online and Offline Settings. 2015.
59. Tamersoy A, Roundy K, Chau DH. Guilt by association: Large scale malware detection by mining file-relation graphs. In KDD; 2014. p. 1524–1533.
60. Sultani W, Chen C, Shah M. Real-world anomaly detection in surveillance videos. In CVPR; 2018. 6479–6488.
61. Sutton RS, Barto AG. Reinforcement learning: an introduction. Cambridge: MIT Press; 1998.
62. Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks, In Advances in neural information processing systems; 2014. p. 3104–3112.
63. Hellaby Watkins CJC. Learning from delayed rewards. Ph.D. Dissertation. King's College, Cambridge. 1989.
64. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–33.
65. O'Neill J, Bouverie BP, Dupret D, Csicsvari J. Play it again: reactivation of waking experience and memory. Trends Neurosci. 2010;33:220–9.
66. Ma X, Shi W. AESMOTE: adversarial reinforcement learning with SMOTE for anomaly detection. IEEE Trans Netw Sci Eng. 2021;8(2):943–56.
67. Alavizadeh H, Alavizadeh H, Jang-Jaccard J. Deep Q-learning based reinforcement learning approach for network intrusion detection. Computers. 2022;11(3):41.

68. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9:1735–80.
69. Cheng J, Dong L, Lapata M. Long short-term memory-networks for machine reading. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016. p. 551–561.
70. Rahman, L., Mohammed, N., Al Azad, A.K.: A new lstm model by introducing biological cell state. In: 2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT); 2016. p. 1–6.
71. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser,L., Polosukhin, I.: Attention is all you need. 31st Conference on Neural Information Processing Systems (NIPS 2017).
72. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: Cnn-rnn: A unified framework for multi-label image classification. 2016 IEEE Conference on Computer Vision and Pattern Recognition .2016.
73. Zhao H, Jia J, Koltun V. Exploring self-attention for image recognition. p. 10073–10082 .2020.
74. Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar,Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In ACL. 2018.
75. Dehghani M, Gouws S, Vinyals O, Uszkoreit J, Kaiser L. Universal transformers. In ICLR. 2019.
76. Fan, Angela, et al. "Addressing Some Limitations of Transformers with Feedback Memory." arXiv preprint arXiv:2002.09402. 2020.
77. Luong MT, Pham H, Manning, C. Effective approaches to attention-based neural machine translation. 2015.
78. Kamei Y, Monden A, Matsumoto S, Kakimoto T, Matsumoto K. The effects of over and under sampling on fault-prone module detection. In: First international symposium on empirical software engineering and measurement (ESEM 2007), Madrid, Spain. 2007.
79. Sayed GI, Tharwat A, Hassanien AE Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection. Appl Inte0ll 2019. 49(1):p. 188–205.
80. Phung SL, Bouzerdoum A, Nguyen GH. Learning pattern classification tasks with imbalanced data sets. 2009.
81. Kingma DP, Welling M. An introduction to variational autoencoders. 2019.
82. Berti A, Zelst v, Aalst v. d. Process Mining for Python (PM4Py): Bridging the Gap Between Process-and Data Science. p. 13–16. 2019.
83. Weijters A, Ribeiro J. Flexible heuristics miner (fhm). In: Proceedings of the 2011 IEEE Symposium on Computational Intelligence and Data Mining – CIDM'11. IEEE; 2011. p. 310–317.
84. The Python Standard Library. https://docs.python.org/3/library/. Accessed 31 Jul 2021.
85. Sasaki Y, The truth of the F-measure, 2007.
86. Kittisak K, Nittaya K. A data mining approach to automate fault detection model development in the semiconductor manufacturing process. Int J Mech. 2011;5(4):336–44.
87. Martino MD, Decia F, Molinelli J, Fernandez A. Improving electric fraud detection using class imbalance strategies. International conference on pattern recognition applications and methods, ICPRAM, 2012.
88. Bekkar M, Djemaa HK, Alitouche TA. Evaluation measures for models assessment over imbalanced data sets. J Inf Eng Appl. 2013;3(10).
89. Velez DR, White BC, Motsinger AA, Bush WS, Ritchie MD, Williams SM, et al. A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. Genet Epidemiol. 2007;31(4):306–15.
90. Zhang D, Lee W S. Learning classifiers without negative examples: a reduction approach. In: 3rd International Conference on Digital Information Management, ICDIM; 2008. p. 638–643.
91. Dogo EM, Nwulu NI, Twala B, Aigbavboa C. Accessing imbalance learning using dynamic selection approach in water quality anomaly detection. Symmetry. 2021;13(5):818.
92. Imbalanced learn documentation. https://imbalanced-learn.org/stable/. Accessed 2 Jan 2023.
93. Leal Piedrahita EA. Hierarchical clustering for anomalous traffic conditions detection in power substations. Cienciae Ingeniería Neogranadina. 2019;30(1):75–88.

Publisher's Note