The Analysis of Online Event Streams: Predicting the Next Activity for Anomaly Detection

Suhwan Lee[1], Xixi Lu[1], and Hajo A. Reijers[1]

Utrecht University, Utrecht, The Netherlands
{s.lee, x.lu, h.a.reijers}@uu.nl

Abstract. Anomaly detection in process mining focuses on identifying anomalous cases or events in process executions. The resulting diagnostics are used to provide measures to prevent fraudulent behavior, as well as to derive recommendations for improving process compliance and security. Most existing techniques focus on detecting anomalous cases in an offline setting. However, to identify potential anomalies in a timely manner and take immediate countermeasures, it is necessary to detect event-level anomalies online, in real-time. In this paper, we propose to tackle the online event anomaly detection problem using next-activity prediction methods. More specifically, we investigate the use of both ML models (such as RF and XGBoost) and deep models (such as LSTM) to *predict* the probabilities of next-activities and consider the events predicted unlikely as anomalies. We compare these predictive anomaly detection methods to four classical unsupervised anomaly detection approaches (such as Isolation forest and LOF) in the online setting. Our evaluation shows that the proposed method using ML models tends to outperform the one using a deep model, while both methods outperform the classical unsupervised approaches in detecting anomalous events.

Keywords: Process mining · Event stream · Anomaly detection.

1 Introduction

Information systems, empowered by blockchain [6] and IoT systems [10], allow an enormous amount of event data to be generated and logged in real-time. The data analytic techniques, such as process mining, are developed to manage the big volume of recorded real-time data. Process mining is a technique to identify and acknowledge the recorded events and gain insights to improve process execution [1]. Recently, process mining has focused on process management and analysis on online settings including process discovery [3], conformance checking [2], and process monitoring techniques [13].

Anomaly detection in process mining aims to detect anomalous behavior in event data [5,12]. Such techniques have been used to identify potential fraudulent behavior to prevent compliance violations [2]. In addition, they are also used to detect log quality issues to improve data quality [14]. Most existing unsupervised

anomaly detection techniques focus on the *case-level* or are situated in *offline* settings [5], i.e., they take as input of a batch event log that contains a set of completed cases.

In practice, detecting anomalies in *online* streaming settings has many advantages, such as being able to take action and timely counter measures. Timely detection also helps dealing with concept drift. At the same time, online anomaly detection also faces many challenges. Unlike an offline setting, which only deals with completed cases, online detection should be able to continuously handle incomplete, ongoing cases. Moreover, online detection should pinpoint anomalies at *event-level* to allow timely, concrete reactions. For example, if credit card fraud is established, the techniques should immediately detect and pinpoint which purchase events are suspicious.

In this paper, we propose to tackle unsupervised anomalous event detection by predicting which activity is next, assuming an ongoing case. More specifically, we first learn a predictive model to predict next activities by preprocessing the completed cases into feature vectors and using machine learning-based classification algorithms (such as Random Forest, XGBoost, LSTM). When a new event of an ongoing case arrives, we apply that predictive model to predict the possible activities and their probabilities using the previous events. The less likely that an activity occurs, the more likely it is an anomaly.

We conduct an evaluation and compare our approach to other approaches that simply encode the events and apply unsupervised anomaly detection algorithms (including Isolation Forest, LOF, OCSVM) and show that our approach performs better in terms of F1 scores. Therefore, this seems a promising direction for online anomaly detection, which raises many new research challenges.

The paper is organized as follows. Related work is discussed in the next section. Section 3 describes the preliminary knowledge for easy understanding on the proposed method. Section 4 introduces the proposed method and setup of evaluation is described in Section 5. The experimental results are reported in Section 6, while conclusions and challenges are drawn in Section 7.

2   Related work

In this section, we discuss event anomaly detection techniques, that are related to our approach. See Table 1. We categorize existing approaches along two dimensions, (1) *offline* versus *online* and (2) *case-level* versus *event-level*. We discuss them accordingly. Then, our contributions in relation to related works are presented.

The classical approach to detect anomalies in process data is aimed at analyzing an offline event log, which has a fixed number of events. Regarding *offline*, *case-level* anomaly detection algorithms, existing works try to discover anomalous cases in an event log, which have infrequent patterns [5] [18] or statistically deviate to an event log [8]. Sani et al. [18] suggested an outlier filtering method based on observed subsequence, in which the activity below the threshold is classified as outlier according to the succeeding activity probability. Khatuya et

Table 1: Comparison of related anomaly detection approaches

	Online/Offline	Target status	Algorithm
Ghionna et al. [5]			Markov cluster
Sani et al. [18]		Case-level	Occurance proability
Khatuya et al. [8]	Offline		Ridge regression
Nguyen et al. [14]			Autoencoder
Savickas and Vasilecas [19]		Event-level	Bayesian belief network
Nolle et al. [15]			LSTM
Tavares et al. [20]			Process model conformity check
Ko and Comuzzi [9]	Online	Case-level	Leverage score calculation
Neto et al. [22]			Autocloud
Van Zelst et al. [23]		Event-level	Automaton processor

al. [8] proposed to use ridge regression to estimate an anomaly score of the individual case by obtaining a statistical distribution of event log features. Although some events in the anomalous case may be normal, these methods are focused on the case rather than individual events.

In case of *offline*, *event-level* anomaly detection, a probabilistic process model has been proposed [19]. Savickas and Vasilecas used Bayesian Belief Networks to detect an anomalous event. An event with a low probability according to the obtained table is classified as an outlier. To improve detector performance, some methods use a deep neural network [14, 15]. Nguyen et al. [14] have proposed anomalous event detection and reconstruction framework, which uses Autoencoder to extract normative features from an event log. Nolle et al. [15] recently introduced a framework to detect case and event anomalies by obtaining the probability of the next event with the deep neural network LSTM. The model based on deep neural networks outperforms existing anomaly detection techniques. However, the works that detect anomalous events assume a constant process distribution in an event log.

Regarding *online*, *case-level* anomaly detection, statistical leverage and data clustering models are used. Ko and Comuzzi [9] have proposed to use a sliding window with a recent event feature vector to calculate the statistical leverage score of the coming trace. When the sliding window is updated, a trace with a higher leverage score is classified as anomalous. Neto et al. [22] have described how to use Autocloud to detect anomalous cases from a stream of events. The model updates the data cluster and classifies anomalous data which is deviated from the existing cluster. To provide a process model, Tavares et al. [20] have proposed a framework that classifies an anomalous case by checking conformity between discovered model and the target case. The proposed methods are able to detect outliers in a streaming event log with a normative process change. Nevertheless, a streamed event is not simultaneously classified and the works are limited in taking into account case-level detection.

Specifically for *online*, *event-level* anomaly detection, Van Zelst et al. [23] have proposed automaton based filtering that uses a sliding bucket with a finite number of events. The model learns the probability of activity sequences. The

advantage of this approach is that the model classifies a streaming event as soon as the event arrives. However, the detector performance may be limited, considering that the training data takes only activity occurrence and only one or two consecutive event sequence is used.

In summary, we see an opportunity in online, event-level anomaly detection to deal with process change and possible performance improvements. The traditional anomaly detection methods are not designed to cope with a streaming event log. The detector assumes a steady process and is not updated. In the case of anomaly detection in a streaming event log, existing works have mainly focused on case-level detection that are capable of catching anomalous data after the case is finished. In addition, the work for event-level detection uses a simple probabilistic model. In this paper, we propose an approach for online anomalous event detection. The proposed approach is based on the machine learning model for performance. The arrived event is classified by a retrained model taking into account possible process change.

3   Preliminaries

Before discussing the steps of the proposed approach in detail, let us explain some required preliminaries. This involves notations for the event log, the key components of pre-processing, and some detection mechanisms. The concepts discussed in this section are implemented to develop our anomaly detection approach, which is discussed in Section 4.

3.1   Event log

An event log contains cases, which consist of a sequence of events. An event consists of multiple attributes including case id, activity, and timestamp. Let $\mathcal{G}_{att} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$ be a set of all possible attributes and $\mathcal{D}_i$ be a set of all possible values for the attribute i. Attributes could be numerical or continuous values. For example, a timestamp takes a numerical value within an interval from beginning to end of an event log. The categorical attribute takes a value from a given set, e.g., an activity with a string data type is assigned within the labels $\{a_1, a_2, ..., a_n\}$. Therefore, we can express an event as a tuple $e = \langle c, act, tst \rangle$, where c, act, tst are case id, one label from set of activity, and a point of time in an event log for timestamp, respectively.

3.2   Next activity prediction

Next activity prediction is one of the techniques in process mining to predict a following activity of running case. Predictions are made using a *classifier* that takes a fixed number of independent features as input. The classifier learns mathematical function to estimate target variables. This means that a classifier extracts features and predicts a probability of following activity from previous events of a case. The data in an event log and previous events of a target case are used as

input for training and testing of the classifier, respectively. Both input data are pre-processed and encoded to a feature vector of equal size. The output from the classifier consists of possible activity labels and probability of the candidates. The activity with the highest probability is selected as a prediction.

3.3   Unsupervised anomaly detection

The models used in unsupervised anomaly detection extract information from the data and map input matrix to a feature space. During a classifier training phase, the data in an event log is transformed and allocated to feature space. The classifier takes events from a running case, including target event, and calculates a distance between the running case and training data mapped into the feature space. If the target event sufficiently deviates from the training data, above the *anomaly threshold*, the event is denoted as anomalous. Otherwise, the event is normal.

4   Approach

This section presents in detail the proposed approach for detecting anomalous activities in streaming event logs: *predictive anomaly detection*. The steps of the proposed approach are shown in Fig. 1. We first explain our approach in an offline setting in Section 4.1. Next, we discuss how the approach can be adapted in an online setting in Section 4.2.

4.1   Predictive Anomaly Detection (PAD)

The proposed approach uses machine learning classification methods to predict next activities for detecting anomalous events. We divide the approach into five steps, as shown in Fig. 1. In the first two steps, we *pre-process* the event log and *train* a model to predict next activity for an event. In the step 3-5, we classify an event e to be an anomaly or not by first *retrieving* its previous event e'. Next, using the trained model, we *predict* the next activities of e' and their probabilities. We then *detect* whether the probability of e is above a threshold. In the following, we explain the approach in depth.

Step 1 and 2: Pre-processing and Model Training To deploy machine learning techniques, which requires feature space in regular format, we transform the finished cases into a training dataset. The dataset is suitable for model training by using *prefix-bucketing* and *feature encoding*. Since our approach relies on the output of a next activity prediction approach, pre-processing methods are adopted from existing predictive process monitoring approaches [21].

Cases used for model training may have a different sequence of events. A common choice is to group cases into buckets by the same prefix length and separately pre-process the bucket for individual anomalies detector [11]. Multiple
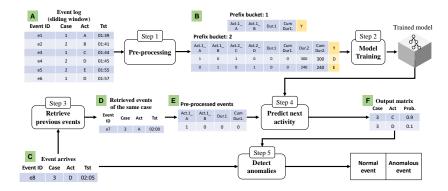
Fig. 1: Online anomalous event detection procedure: Proposed approach

classifiers are trained to learn features from respective bucket which contains cases. Prefix bucket n contains events from first to nth event of each case.

For example, let us consider two cases in an event log, both of which have 3 events (see Fig. 1A). Two prefix length buckets are obtained, one with all prefixes of length 1, and the other with all prefixes of length 2. For each event in the buckets, its next activity is used as output label, Y, for training a classifier (Fig. 1B).

In order to train a classifier, the collected events in the buckets are required to be modified as feature vector with fixed size. For feature encoding, we transform the attributes (e.g., activity and timestamp) into suitable features with index-base encoding method to maintain the order of events. The categorical attribute, such as activity label, is encoded using a one-hot encoding scheme by considering the order of events [11]. An encoded activity label a for case c at event i is:

$$c_{i,act} = \begin{cases} 1 & \text{if } c_{i,a} = Act.i\_a \\ 0 & \text{Otherwise} \end{cases}$$

Regarding a timestamp attribute, a point of the event occurrence is transformed into duration and cumulative duration of an event and a case, respectively (see Fig. 1B). Event duration $Dur.i$ is elapsed time between a preceding event $i-1$ and a current event i, while cumulative duration $CumDur.i$ at event i is aggregated event duration since the start of the case. Given an event timestamp, the obtained duration and cumulative duration respectively are: $Dur.i = e_{tst,i} - e_{tst,i-1}$ and $CumDur.i = \sum_{n=1}^{i} Dur.n$

After prefix-bucketing and feature encoding, the pre-processed rows and columns are concatenated together as an input feature matrix for model training.

The objective of this phase is to learn a model (i.e., a classifier) that uses previous events to predict the activity of next event. In addition to the possible next activities, we also retrieve the probabilities of each possible activity from the model to determine how *likely* or *unlikely* the next activity is. Note that

we use the *completed* cases for training a model. We then apply the model to predict the next coming event of *running* cases in step 3-5.

Step 3-5: Retrieve, Predict, and Detect We have trained a model that predicts the next possible activities. When a new event e of a running case arrives (Fig. 1C), we first retrieve its previous event e' (Fig. 1D) and the encoded event (Fig. 1E). We use this encoded event as input for the trained model to predict the possible next activities and their probability, which we wrapped into an *output matrix*(Fig. 1F).

We then use this output matrix to classify the new event e. If the activity of event e is listed on the possible activities and its probability is above a sufficient level, i.e., the *anomaly threshold*, this means that the target event occurs commonly after e' and, therefore, is classified as normal. Otherwise, we classify e as an anomaly.

As an example, let us assume the anomaly threshold is 0.15. Event e_8 in Fig. 1 arrives as the second event of Case *3* and has an activity label D. We first retrieve the previous events of Case *3*. We then apply the trained model to predict the possible next activities. According to the output matrix of the trained model, the probability of e_8 being activity D is only 0.1. Due to this probability lower than the anomaly threshold, this event classified as anomalous.

4.2   Online PAD using Sliding Window and Retraining Interval

We have explained the proposed approach in an offline setting. In this section, we explain how the approach handles streaming events.

A *sliding window* is one approach to learn features from new observations of sequential data via updated window [7]. Before the pre-processing step, a sliding window collects a number of most recent completed cases from the streaming events and manages the data to be transformed. As a newly finished case from event streams arrives, a sliding window takes it as an input and places it at the beginning of the window. A case located at the end of the window is removed to maintain the fixed sliding window size (W). As long as W is relatively small to the total number of cases, this procedure allows the machine learning model to be agilely retrained considering possible normal behavior changes of the dataset by collecting the information from more recent observations. However, a small sliding window size may be insufficient to properly train due to the lack of information on normal behavior. As well as issues on size for model training, a small sliding window size requires frequent retraining on the model, which leads to unnecessary recalculation of the outcome. We apply multiple parameters to investigate the sensitivity of window size to detect the optimal performance of the proposed approach in our evaluation.

Besides using the sliding window, we implement a parameter called *retraining interval R* to control the retraining frequency of the model. In essence, after retraining the model at a certain point in time t, we pause to retrain the model again until R number of new cases are completed and updated in the sliding

window. For example, if the absolute value of R is 1, then the model is retrained after each case is completed. If $R = W$, then the model is retrained after all the cases in the sliding window are updated. For such a retraining interval, we use a relative size to a sliding window, e.g., $R = 10\%$ of W. If the number of new cases inserted into the sliding window satisfies the retraining interval size condition, the anomaly detection model is retrained.
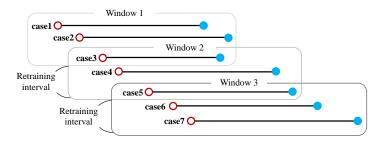

Fig. 2: The architecture of sliding window and retraining interval

Fig. 2 exemplifies how the sliding windows are updated along with a retraining interval. Let us assume there are 30 cases in an event log and 10% of sliding window size with 66.7% of retraining interval size. The sliding window takes 3 cases to train a classifier. The window is updated when 2 newly finished cases are inserted which satisfies retraining interval size. Window 1 contains case1 to case3. The window is updated after two new cases, case4 and case5, are inserted into a window. After updating the window, we retrain the model and use the model for detecting anomalies in newly arrived events (e.g., case6 and case7).

5   Empirical evaluation

The objectives of the evaluation are twofold. Firstly, we investigate how well our approach performs to detect anomalies in comparison to classical unsupervised anomaly detection as baseline. Secondly, we investigate the influence of the parameters on the detection performance, specifically the anomaly threshold, sliding window, and retraining interval. For these objectives, we implemented the proposed approach and the experiments in Python. The code to reproduce the experiments is publicly available on Github[1]. In the following, we first explain the experiment settings, which include the dataset used, the chosen techniques, and the parameter settings. Subsequently, we discuss the results of our evaluation.

5.1   Setup of evaluation

We have used synthetic logs with anomalous events that were used in [23]. The event logs are generated from the gathered event streams based on the 21 vari-

---

[1] https://github.com/ghksdl6025/streaming_anomaly_detect

ations of the loan application process [4]. Then 6 anomalous event logs are created by randomly injecting the infrequently occurred events in various probability [23]. Each log follows a different probability range from 2.5% to 15% in steps of 2.5%. In this paper, the probability of generated anomalous events is denoted as *Noise level*. Each log comprises 500 cases and approximately 7600 to 8600 events depends on the noise levels. Every log has 18 different activity labels. Table 2 shows descriptive statistics of the used event logs.

Table 2: Descriptive statistics of the noise imputed event log

Noise level	Cases	Events	Activity labels
2.50%	500	7410	18
5%	500	7630	18
7.50%	500	8542	18
10%	500	7922	18
12.50%	500	7888	18
15%	500	8159	18

As discussed in Section 4, sliding window and retraining interval are parameters to respectively control the size of the training data and the retraining rate in the streaming setting. Fig. 3 shows evaluation settings to check the influence of the sliding window and the retraining interval size. We consider 5%, 10%, and 20% as a ratio to the total number of cases in the log as sliding window size. For the retraining interval, we tested with 6 different parameters from 0% to 50% in steps of 10% as a ratio of retraining interval size to the sliding window. The influence of the sliding window is examined through changing sliding window sizes with fixed retraining interval, and vice versa. The threshold level is a proportion of anomalies in the dataset, which is used as a criteria in the anomaly detection phase. We experiment with 6 different thresholds, which have a range from 0.01 to 0.25, to investigate the influence of threshold on detection performance.

In this evaluation, we consider both a machine learning model and a deep neural network to classify anomalous events, which are typically adopted in data mining field. As a detection model for the proposed approach, we experiment with two machine learning models and one deep neural network typically adopted in next activity prediction [21]: random forest (RF), extreme gradient boosting (XGB), and long short term memory (LSTM). For the baseline, we experiment with isolation forest (IForest), local outlier factor (LOF), one-class support vector machine (OCSVM), and autoencoder (AE), which are unsupervised anomalous data detection algorithms. For machine learning models, we use the classifiers provided by the Python packages Scikit-learn [17].

For the deep neural network based detection model, the experiment is conducted with the Python package Pytorch [16] with different layer sizes and a number of layers by LSTM and AE. Regarding the LSTM model for the proposed method, two LSTM layers are stacked to obtain hidden feature vectors.

		W: 5% R: 20%			
W: 10% R: 0%	W: 10% R: 10%	W: 10% R: 20%	W: 10% R: 30%	W: 10% R: 40%	W: 10% R: 50%
		W: 20% R: 20%			

Fig. 3: Evaluation settings. **W** indicates the percentage of total number of cases and **R** the percentage of sliding window

Multiple linear layers are followed to predict next activity labels. We implement autoencoder structure including multi-linear layers for latent variables as presented in [14].

To performance of the anomaly detection task is separately evaluated using the F-score, ranged from 0 to 1, for both normal and anomalous events. The F-score is a classification model accuracy indicator calculated from precision and recall, i.e., correct rate among positive predictions and correct decision rate among true items, respectively. The high f-score, close to 1, indicates accurate identification of the events.

5.2   Baseline - Unsupervised Anomalous Event Detection

We propose to use classical unsupervised anomaly detection as the baseline approach. The approach is shown in Fig. 4. We perform the same pre-processing step. However, the model detects the anomaly of an arrived event without predicting the next activity. This way of anomalous event detection is close to traditional outlier classification in data mining as explained in Section 3.3. The same pre-processing method is applied to encode input feature vector, as well as sliding window mechanism for streaming data. In the detection step, the classifier takes events of the running case with the target event as input and calculates deviated distance to the feature space. The model distinguishes the anomaly of an arrived event using an anomaly threshold.

6   Result and discussion

Detection performance of proposed approach Fig. 5 shows the F-score of normal and anomalous events detection for the proposed approach and the baseline. From the results, we find that (i) the proposed approach outperforms the baseline on every noise level and (ii) the deep neural network shows a lower performance than machine learning models.
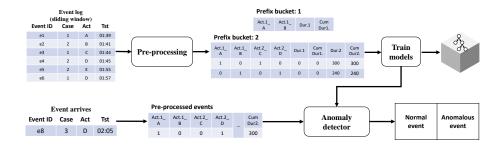
Fig. 4: Online anomalous event detection procedure: Unsupervised method

The proposed approach detects anomalous events more effectively than baseline for all every noise levels. Deep neural network based models in both approaches show lower performance than other classifiers. One possible reason for the performance gap is an issue on training data size. The sliding window size may not be big enough for proper deep learning model training.

Effects of different anomaly threshold Fig. 6 shows the F-score of anomalous event detection for the proposed approach by different anomaly threshold levels. According to Fig. 6a, the F-score of both normal and anomalous event decreases with the high threshold in next activity prediction, i.e., all classifiers perform the highest capability on detection at a 0.01 threshold level, except high noise level with RF. We can observe that each classifier requires a separate threshold level for optimal performance. Moreover, RF shows the highest performance at a 0.05 threshold with high noise event log, unlike the other classifiers.

In case of the unsupervised approach, we observe two remarkable points from the analysis in Fig. 6b. These are (i) different performance patterns among implemented algorithms and (ii) the influence of high threshold on anomalous event detection. Regarding performance volatility within models for unsupervised approach, the detection ability of IForest and LOF increases with high threshold while OCSVM and AE are relatively stable at all noise levels. In contrast to the first proposed approach, low threshold to distinguish anomalies could not be applied to the unsupervised method.

Effects of sliding windows and retraining intervals We evaluate the anomaly detection performance of the proposed approach by different training windows and retraining interval sizes.

We find that (i) the sliding window size has a positive influence on better model performance, (ii) the proposed approach is more sensitive to window size change than the unsupervised approach, and (iii) the performance change slope is getting flattened with window size. Fig. 7a and 7b show the F-score of normal and anomalous event detection for the proposed approach and the unsupervised approach with sliding window change, respectively. Despite that, if both

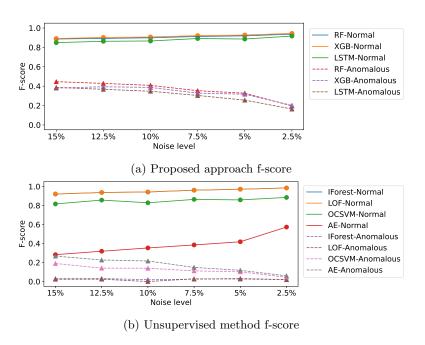(a) Proposed approach f-score

(b) Unsupervised method f-score

Fig. 5: F-score of normal and anomalous events detection by noise level

approaches were trained using the same window size, the sliding window size does not influence the performance improvement for the unsupervised approach. More specifically, the results regarding OCSVM seem to show that the increase of training window size even has a negative impact on detecting the anomalies.

The performance of the proposed approach improves for both normal and anomalous events detection with a large window, i.e., more useful information is collected with a bigger window size. Along with the positive relation between model improvements and training data size, we also observe a marginal effect of increasing training window and detecting ability improvements. As in Fig. 7a, the slope between sliding window size and F-score is not linear, i.e., the obtained useful information for detecting anomalous by increasing training dataset is limited. Therefore, the efficiency of managing the training window depends on a balance between information gain and the cost of increasing the dataset.

Retraining interval size does not influence the anomaly detection ability for either the proposed approach or unsupervised method. Fig. 8a and 8b show the F-score of normal and anomalous event detection for the proposed approach and the unsupervised algorithm with retraining interval changes, respectively. Generally, we can observe that the retraining interval is irrelevant to the model training phase. The performance lines across all detecting models are stable over noise level.
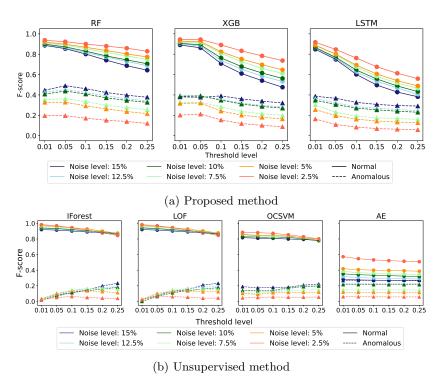
(a) Proposed method

(b) Unsupervised method

Fig. 6: Performance comparison by anomaly threshold

7 Conclusion and challenges

We have presented an approach for online event anomaly detection via next activity prediction using machine learning models. The proposed approach uses a sliding window to (re)train the model with the recently recorded cases on the event stream. The probability of possible next activities obtained from the ML models is used to classify a new event as anomalous or not.

The method uses well-known machine learning and deep learning algorithms, which can give flexibility by changing into other models later. Even though the performance of deep neural networks is lower than machine learning models, we have shown that the next activity prediction method outperforms the classical unsupervised anomaly detection method when applied to event logs.

As a relatively new research area, anomalous event detection in event streams has open challenges in several perspectives. We identify multiple challenges on online anomaly detection that still need to be addressed. Firstly, in an online setting, an event classified as (potential) anomalous may be changed into normal behavior later after the model update or after new events arrives. *(RC1)* *How and when can we confirm that a potential anomalous event is a definitive anomaly?* The approach proposed in this paper is designed to not updating the prediction. Nevertheless, taking such changes into account may be very infor-

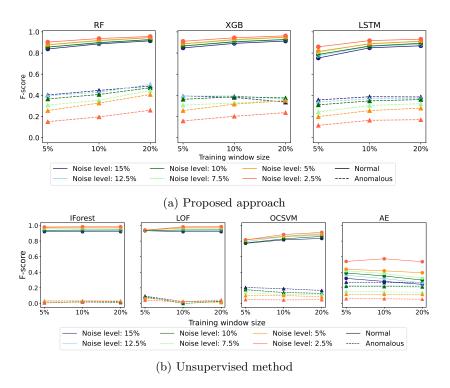(a) Proposed approach

(b) Unsupervised method

Fig. 7: Performance comparison by sliding window size

mative for suggesting user actions and is a challenge itself and worthy of further investigation. Another interesting challenge related to changes is *(RC2) how can anomalies be detected while taking into account concept drift?* Finally, the online detection of potential anomalies leads to the possibility of building an online decision-making system that suggests follow-up actions for the detected anomalies. This possibility leads to the third research challenge: *(RC3) how can we build a decision-making system with the domain experts for the further investigation of the potential anomalies?*

The work presented here can be extended in several ways. In addition to solving the research challenges, we will study the post-hoc analysis on the online anomaly detecting model to provide an explanation of the cause of anomalous events. Finally, we are planning to develop the interactive online detecting model by implementing the feedback from users who select anomalous events. The collected feedback would help the model to give concrete reasoning on the predicted anomalies and improve the model performance.

References

1. van der Aalst, W.M.P.: Data science in action. In: Process mining, pp. 3–23. Springer (2016)
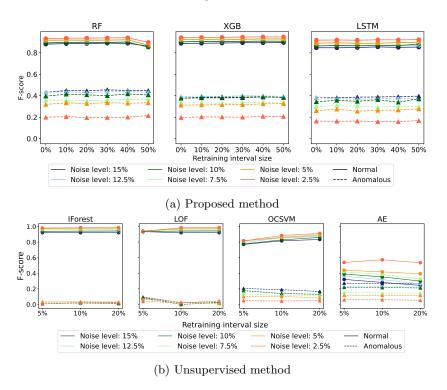
(a) Proposed method

(b) Unsupervised method

Fig. 8: Performance comparison by retraining interval size

2. Burattin, A., Carmona, J.: A framework for online conformance checking. In: International Conference on Business Process Management. pp. 165–177. Springer (2017)

3. Burattin, A., Sperduti, A., van der Aalst, W.M.P.: Heuristics miners for streaming event data. arXiv preprint arXiv:1212.6383 (2012)

4. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013)

5. Ghionna, L., Greco, G., Guzzo, A., Pontieri, L.: Outlier detection techniques for process mining applications. In: International symposium on methodologies for intelligent systems. pp. 150–159. Springer (2008)

6. Guo, H., Meamari, E., Shen, C.C.: Blockchain-inspired event recording system for autonomous vehicles. In: 2018 1st IEEE international conference on hot information-centric networking (HotICN). pp. 218–222. IEEE (2018)

7. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 97–106 (2001)

8. Khatuya, S., Ganguly, N., Basak, J., Bharde, M., Mitra, B.: Adele: Anomaly detection from event log empiricism. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. pp. 2114–2122. IEEE (2018)

9. Ko, J., Comuzzi, M.: Online anomaly detection using statistical leverage for streaming business process events. In: International Conference on Process Mining. pp. 193–205. Springer (2020)

10. Kolozali, S., Bermudez-Edo, M., Puschmann, D., Ganz, F., Barnaghi, P.: A knowledge-based approach for real-time iot data stream annotation and processing. In: 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom). pp. 215–222. IEEE (2014)

11. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: International Conference on Business Process Management. pp. 297–313. Springer (2016)

12. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Detecting deviating behaviors without models. In: Business Process Management Workshops. Lecture Notes in Business Information Processing, vol. 256, pp. 126–139. Springer (2015)

13. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. SCC **17**, 1–8 (2017)

14. Nguyen, H.T.C., Lee, S., Kim, J., Ko, J., Comuzzi, M.: Autoencoders for improving quality of process event logs. Expert Systems with Applications **131**, 132–147 (2019)

15. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Binet: Multi-perspective business process anomaly classification. Information Systems **103**, 101458 (2022)

16. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32**, 8026–8037 (2019)

17. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)

18. Sani, M.F., van Zelst, S.J., van der Aalst, W.M.P.: Improving process discovery results by filtering outliers using conditional behavioural probabilities. In: International Conference on Business Process Management. pp. 216–229. Springer (2017)

19. Savickas, T., Vasilecas, O.: Belief network discovery from event logs for business process analysis. Computers in Industry **100**, 258–266 (2018)

20. Tavares, G.M., Ceravolo, P., Da Costa, V.G.T., Damiani, E., Junior, S.B.: Overlapping analytic stages in online process mining. In: 2019 IEEE International Conference on Services Computing (SCC). pp. 167–175. IEEE (2019)

21. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD) **13**(2), 1–57 (2019)

22. Vertuam Neto, R., Tavares, G., Ceravolo, P., Barbon, S.: On the use of online clustering for anomaly detection in trace streams. In: XVII Brazilian Symposium on Information Systems. pp. 1–8 (2021)

23. van Zelst, S.J., Sani, M.F., Ostovar, A., Conforti, R., La Rosa, M.: Filtering spurious events from event streams of business processes. In: International Conference on Advanced Information Systems Engineering. pp. 35–52. Springer (2018)