Anomaly Correction of Business Processes Using Transformer Autoencoder

Ziyou Gong[1,2][0000−0003−1724−1364], Xianwen Fang[1,2][0000−0001−8531−7215], and Ping Wu[1][0009−0009−6631−1032]

[1] Anhui University of Science and Technology, Huainan, China
`gzy409338163@163.com`
[2] Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, Huainan, China

Abstract. Event log records all events that occur during the execution of business processes, so detecting and correcting anomalies in event log can provide reliable guarantee for subsequent process analysis. The previous works mainly include next event prediction based methods and autoencoder-based methods. These methods cannot accurately and efficiently detect anomalies and correct anomalies at the same time, and they all rely on the set threshold to detect anomalies. To solve these problems, we propose a business process anomaly correction method based on Transformer autoencoder. By using self-attention mechanism and autoencoder structure, it can efficiently process event sequences of arbitrary length, and can directly output corrected business process instances, so that it can adapt to various scenarios. At the same time, the anomaly detection is transformed into a classification problem by means of self-supervised learning, so that there is no need to set a specific threshold in anomaly detection. The experimental results on several real-life event logs show that the proposed method is superior to the previous methods in terms of anomaly detection accuracy and anomaly correction results while ensuring high running efficiency.

Keywords: business process · event log · anomaly detection · anomaly correction · Transformer autoencoder · self-supervised learning.

1 Introduction

Anomaly detection in business processes is an important means to guarantee the correct execution and timely correction of business processes. At the same time, the execution of business processes will record event logs in the information system, on which many process analysis methods depend, and the repair of low-quality event logs, i.e., anomaly correction, also facilitates the subsequent analysis of business processes [1,11].

Previous anomaly detection methods of business process based on predictive process monitoring first predict the next event of business process through machine learning method, and set a threshold to determine whether the business

process instance is anomalous by whether it exceeds or falls below this threshold [16,13]. Most of the machine learning models for predicting the next event are LSTM [6,25,4], but the feature extraction capability of LSTMs and the parallelism of model training are difficult to satisfy all business process instances. And this methods requires iteratively predicting the next event for each process instance, which is less efficient in offline scenarios.

Autoencoder-based anomaly detection and anomaly correction trains a denoising autoencoder by injecting noise into the normal event log, and ensures that the output is close to the original normal event log by minimizing the event log reconstruction error [17,8,12,15]. The autocoder-based approach inputs a complete business process instance with anomalies and outputs a complete business process instance without anomalies, which can be more efficient compared to predicting the next event. However, previous autoencoder-based methods can only handle a limited number of anomalies and all of them set the threshold and thus discriminate the anomalies by averaging the reconstruction error, which relies heavily on the model to learn the difference between the characteristics of normal and anomalous cases.

In recent years Transformer [26] has shown excellent performance in the field of natural language processing and computer vision. Compared with LSTM, it can process all elements of the input sequence at the same time, so the model has higher parallelization and is more suitable for large-scale scenarios. Therefore, Transformer have also been noticed by researchers in the field of process mining [20,3]. The mask pre-training models developed on this basis utilize Transformer's excellent feature extraction capability to achieve the best performance in various tasks in the fields of natural language processing and computer vision [5,7]. It enhances the feature learning capability of the model by predicting the masked parts of input sentences by context.

In this paper, we propose a business process anomaly detection and anomaly correction method based on Transformer autoencoder (TransformerAE), which utilizes the Transformer autoencoder and self-supervised learning on original event logs. Compared with previous methods, we transform the anomaly detection problem into a classification problem so that it does not need to set a threshold for anomaly determination, and leverages transformer's structure to combine it with anomaly correction tasks. Evaluation results of anomaly detection and anomaly correction task on 4 real-life event logs comparing with baseline methods show that our method has a better balance of detection and correction results and running speed.

The rest of the paper is organized as follows: Section 2 focuses on some relate work on business process anomaly detection and business process anomaly correction. Section 3 focuses on the Transformer autoencoder-based approach for business process anomaly detection and anomaly correction. Section 4 describes the data used for the experiments and the evaluation and comparison of the methods in this paper. Section 5 summarizes the paper as well as provides an outlook for future work.

2   Related work

Business process anomaly detection methods are usually closely related to business process anomaly correction methods, and a lot of previous work has been proposed, which can be mainly divided into classical methods and machine learning methods. Classical methods mainly focus on statistical analysis of event logs and manual feature extraction. For business process anomaly detection, Böhmer et al. [2] proposed a multi-angle anomaly detection method, which can deal with unexpected situations of process model execution events and combine multiple events to detect collective anomalies. Sarno et al. [22] proposed an anomaly detection method based on fuzzy multi-attribute decision making and fuzzy association rule learning. Van Zelst et al. [28] proposed a general event flow filter, which detects and removes infrequent behaviors from the event stream by constructing a probabilistic automaton set and dynamically updating it to filter anomalous events. Ko et al. [9] propose a statistical lever-based process anomaly scoring method that uses three different methods to set anomaly detection thresholds. It can be integrated into an online framework to handle anomalies in an online event stream [10]. For business process anomaly correction, Suriadi et al. [23] proposed a pattern-based approach to record common event log quality problems, developed a set of components to describe event log quality problems as patterns, and used the pattern library to identify and repair event log quality problems. Fani Sani et al. [21] used a probabilistic approach to identify anomalous behavior based on the behavioral context of the process, i.e. the sequence of fragments of activity that occurred before and after the potentially anomalous behavior, and then replaced them with behaviors that were more likely to occur in the context in which the anomalous behavior occurred. Liu et al. [14] proposed a missing activity repair method based on activity sequence relationships in event logs, which used activity relationship matrix to represent event logs and cluster them.

Machine learning methods make feature extraction easier and make end-to-end anomaly detection and correction easier. For business process anomaly detection, Pauwels et al. [19] proposed a model that uses dynamic bayesian networks to model the normal behavior in log files and point out the root cause of the anomaly. Tavares et al. [24] proposed an online anomaly detection method for business processes, which extracted case descriptors from event streams and applied density-based clustering techniques to detect outliers. Autoencoder-based approaches learn the representation of the underlying model to detect anomaly traces and abnormal activity [17,12], or set an anomaly detection threshold by averaging reconstruction losses [15,8]. Lee et al. [13] solve the problem of anomaly detection of online events by proposing a method that adopts next stage activity prediction, which uses a machine learning model to predict the probability of the next activity and treats the unpredictable events as anomalies. Binet [16] is a neural network architecture for anomaly detection of multi-view business process event logs, and a set of heuristic algorithms for automatically setting the threshold of anomaly detection algorithms. For business process anomaly correction, [15] proposed an autoencoder-based event log repair method to pre-

dict missing or incorrect events by training the autoencoder on the event log. Nolle et al. [18] proposed a multi-view process anomaly correction method based on recurrent neural networks and bidirectional beam search, using two trained recurrent neural networks to predict event sequence from left to right and calculate event sequence alignment from right to left, as a way to detect and correct event sequence anomalies.

Few previous works combine anomaly detection and anomaly correction of business processes together, and cannot achieve end-to-end anomaly detection and anomaly correction. Moreover, most anomaly detection methods rely on specific threshold Settings, and different thresholds need to be designed for different event logs.

3   Methodology

In this section, we first give the overview of proposed methods. As shown in Fig. 1, the input of the model is an event log containing anomalies constructed from the original nomal event log. The procedure for creating an event log containing anomalies will be described in detail in Section 3.1. The model will classify each trace to determine whether it is an anomaly trace, and output the corrected input trace. The detailed model structure and training process will be presented in Section 3.2.
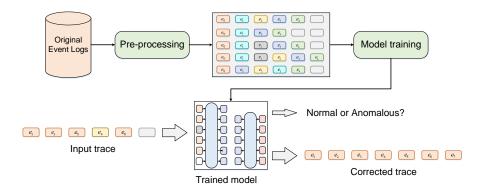

Fig. 1. Overview of proposed method.

3.1   Pre-processing

We first introduce some basic concepts of event logs. Let A be the set of h distinct activities, T be the time domain, and C be the set of case identifiers. An event is a tuple $\varepsilon = (c, a, t)$ where $c \in C$, $a \in A$, $t \in T$. A trace $\sigma = \langle \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m \rangle$ is the execution trajectory of an instance of a business process. It is a finite sequence

of m distinct events with the same case identifier that is non-decreasing in time. So the event log $L = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ is a set of n different traces of all records of business process execution.

Anomaly detection in business processes requires event logs with labeled traces (normal or anomaly), which is usually not available in practice. Therefore, previous research usually inject different types of anomalies in the event log and label each trace [9,13,2]. We redefines 6 types of anomalies based on the previous research [9,13,2] which are Missing, Skip, Replace, Insert, Early and Late. The 6 types of anomalies are described as follows:

- **Missing**: An event in the process instance is recorded in the event log, but the activity identifier is missing.
- **Skip**: An event in the process instance is skipped.
- **Replace**: An event in the process instance is replaced with an arbitrary other event.
- **Insert**: Insert a random event into the process instance.
- **Early**: an event in the process instance occurs early.
- **Late**: an event in the process instance is postponed.

Then we adopt the masking operation similar to that in BERT to inject at least one kind of anomaly into a certain proportion of traces, which is determined by r_{case}, and the maximum number of anomaly categories is determined by r_{act}. Since Skip and Insert anomalies will change the length of traces, the position of each injected anomaly will be restricted in order to prevent some injected anomalies from affecting other anomalies. A limit on the maximum length of the input sequence is required to ensure a same length of the input sequence, which is also determined by the proportion r_{act} and the anomaly type, i.e., the maximum length of the input sequence is all anomalies injected in the maximum length traces are of Insert anomaly type. In addition, it is known from the log behavior profile [27] that the traces created by some of the above anomalies may be normal. For example, if two events have a interleaving order relation, the injection of the Early and Late anomalies results in an exchange of the order of these two events in the business process instance that does not result in an anomaly. Therefore for this case it needs to be labeled as normal. The anomaly trace detection problem will be transformed into a classification problem, that is, let the classification model learn the difference between normal traces and anomalous traces.

3.2 Anomaly Detection and Anomaly Correction with TransformerAE

The structure of the TransformerAE model proposed in this paper is shown in Fig. 2. The model consists of two parts, namely Encoder and Decoder, both Encoder and Decoder are stacked based on Transformer Block, and the Transformer Block used is the Encoder of the original Transformer. Suppose that the input sequence is X, the multi-head attention mechanism on X is computed as follows:

$$\text{MultiHeadAttention}(X) = \text{Contact}(\text{head}_1, \ldots, \text{head}_n)W^O$$
$$\text{where head}_i = \text{SelfAttention}(X)$$
$$= \text{Attention}(X, X, X) \tag{1}$$
$$= \text{softmax}(\frac{XW_i^Q(XW_i^K)^T}{\sqrt{d_k}})XW_i^V$$

Where W_i^Q, W_i^K, W_i^V and W_i^O is the learnable weights. Then the Transformer Block is defined as follows:

$$Z = \text{Norm}(\text{MultiheadAttention}(X) + X)$$
$$\text{Transformer Block} = \text{Norm}(\text{FFN}(Z) + Z) \tag{2}$$

Where Norm is the normalization layer and FFN is the feed forward neural network. Based on this, TransformerAE is composed of transformer block encoder and decoder, and input sequence first will be processed by an embedding layer consisting of a learnable token embedding and a learnable position embedding before being fed into encoder and decoder:

$$Y = \text{Embedding}(X) + \text{PositionEmbedding}(X)$$
$$\text{Encoder}(Y) = \text{Transformer Block}(\ldots\text{Transformer Block}(\text{Transformer Block}(Y)))$$
$$\text{Decoder}(Y) = \text{Transformer Block}(\ldots\text{Transformer Block}(\text{Transformer Block}(Y)))$$
$$\text{TransformerAE} = \text{Decoder}(\text{Encoder}(X))$$
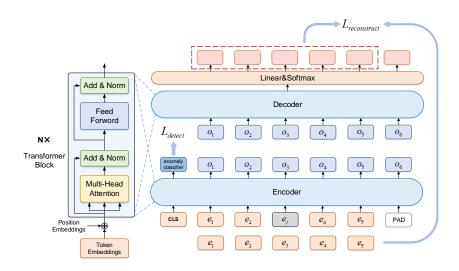$$\tag{3}$$

Fig. 2. Architecture of TransformerAE.

The trace is represented as a sequence of events after preprocessing, then the classification token "[CLS]" is added to the head of the event sequence and the token "[PAD]" is added to the tail of the event sequence to fill the event sequence into the same length. The event sequence is encoded by one-hot encoding and then fed into Encoder, which obtains the embedding information of the event sequence through token embedding and position embedding. Decoder's input also consists of token embedding and position embedding, the difference from Encoder is that the first position of the sequence does not have the classification token "[CLS]". Decoder's output consists of the linear layer and Softmax activation function to obtain the classification result of each element of the event sequence, i.e. corrected trace. In this way, the corrected trace can be output at one time, which is more efficient than the autoregressive output.

Since there is no way to know whether there is any anomaly in the event log in the existing dataset, it is necessary to inject random anomalies into the original event log which is assumed to be free of anomalies in order to simulate possible anomalies in the real world, so as to test the performance of the trained model. Assuming an original event log $L = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ with no anomalies, and injecting anomalies into it to get an event log $L' = \{\sigma'_1, \sigma'_2, ..., \sigma'_n\}$ containing anomalous events, then the anomaly correction task in this paper is to make $\hat{L} = \text{Decoder}(\text{Encoder}(L'))$ as close as possible to L by using an Encoder and a Decoder. The anomaly detection task is also based on the output of the Encoder, i.e., defining a detection function f, there:

$$f(\text{Encoder}(\sigma_i)[0]) = \text{sigmoid}(\text{Linear}(\text{Encoder}(\sigma_i)[0])) \qquad (4)$$

Where [0] denotes the first element of the taken sequence, corresponding to the first element of the input sequence, i.e., the classification token "[CLS]", Linear denotes the linear mapping, and sigmoid is the activation function.

During model training, the training samples created from the original traces are input into Encoder, then the output of the position corresponding to the classification token "[CLS]" is fed into an anomaly classification layer to get the classification result of the input trace, and the loss function for the binary classification of normal and anomalies is $L_{detect} = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$. The rest of the output of Encoder is then fed into Decoder. For the correction of anomaly traces, unlike the autocoder-based anomaly correction which uses a squared loss function, here the cross-entropy loss is computed from the Decoder's output and the original input sequence, i.e., the reconstructed loss function is $L_{reconstruct} = -\sum_{i=1}^{h+2} y_i \log \hat{y}_i$, where h is the total number of activity categories ,because the "[CLS]" and "[PAD]" tokens are added, and the total loss is $L = L_{detect} + L_{reconstruct}$. Since correcting some specific anomalies changes the length of the event sequence, the complete input sequence and output sequence need to be considered in computing the reconstruction loss, which introduces unnecessary filling of tokens, but this effect was found to be negligible in the experiments.

4   Evaluation

4.1   Experimental data and experimental setup

For the experiments, we use four real-life event log datasets from different domains which are available at 4TU.ResearchData (https://data.4tu.nl/). All the event log statistics are shown in Table 1.

Table 1. Event log dataset statistics.

Event logs	Num. of trace	Num. of activity	Avg. case length	Max. case length
BPIC2012W	9658	6	7	79
BPIC2012	13087	24	20	175
BPIC2013in	7554	7	8	123
BPIC2017	31509	26	38	180

In the experiment, 80% of the entire event log is used for training, and the remaining is used to test the effect of anomaly classification and anomaly correction. During training, case anomaly percentage is set to 0.5 for class balance, and event anomaly percentage is set ot 0.3, 0.5 and 0.7 respectively (the three anomaly pecentage only represent fewer, balanced, and more anomalous events, and the best results can be obtained through more fine-tuning in real-life scenario), and for testing, the case anomaly pecentage is selected as 0.1, 0.3, 0.5, and 0.7, while the event anomaly percentage is at 0.1, 0.3, 0.5 and 0.7 on the basis of increasing the number of anomalous events for 1 and 2 cases, used to detect the performance of all methods in the most extreme cases, note that some event logs or part of the maximum length of the trace is small, 1 or 2 events may also account for the proportion of the total length of the trace is more than 0.1, so increase the number of anomalous events for the case of 1 and 2 in order to try to simulate all the cases in reality.

We compare our method with four neural network architectures, including BINet [16], Autoencoder (AE), LSTM Autoencoder (LSTMAE), and BERT. BINet is a time-step prediction-based anomaly detection method, which has three versions respectively. Since no other event attribute except activity is used in this paper, only the BINetv1 is compared. The anomaly detection method based on AE and LSTMAE is based on the [15], which sets a threshold of anomaly detection by the average reconstruction loss of the whole event log, and then discriminates whether the trace is anomalous or not, and the reconstruction results of AE and LSTMAE are corrected traces. In this paper, the hyperparameters of the LSTMAE are adjusted (the dimension of the hidden layer is adjusted to 100, and the number of the LSTM layers is adjusted to 2), so as to achieve a better performance. The BERT-based method replaces the decoder of TransformerAE with a fully connected neural network, and the procedure of anomaly

detection is the same as TransformerAE. The hyperparameter settings for BERT and TransformerAE in Transformer Block are shown in Table 2.

Table 2. Transformer Block hyperparameter settings.

	BERT	TransformerAE
Num. of encoder attention heads	8	8
Num. of decoder attention heads	-	8
Num. of Transformer Block layers of the encoder	2	2
Num. of Transformer Block layers of the decoder	-	2
Dimension of FFN	64	64

4.2   Evaluation metrics

For the results of case-level anomaly detection, we use the F-score for case classification to evaluate the proposed method. At the event level, the same evaluation metric as for anomaly correction are used because some anomalies lead to changes in the length of the traces and the position of the events, which makes it difficult to accurately measure them by the accuracy of event prediction. The position and type of anomalous events are determined by comparing the corrected traces with the input traces.

For anomaly correction results, some works usually only repairs missing and replaced activities so only the prediction accuracy of missing and replaced activities is evaluated. For real-life anomaly event logs, however, not only missing and replaced activities are included, but also missing events or changes in event locations and redundant event records may be included, so each complete trace in the event log needs to be compared during anomaly correction. In view of this, we consider using similarity based on Damerau-Levenstein distance (edit distance) to evaluate the results of anomaly correction, i.e., we compute the similarity between the anomalous traces and the corrected traces.

$$S_{DL}(s_1, s_2) = 1 - \frac{\text{DL\_distance}(s_1, s_2)}{\max(\text{len}(s_1), \text{len}(s_2))} \tag{5}$$

where $\text{DL\_distance}(s_1, s_2)$ is the edit distance, and $\text{len}(s)$ is the length of the sequence s. A similarity of 0 means completely different and a similarity of 1 means completely identical.

4.3   Experimental results

The experimental results of the anomalous case classification on the event log BPIC2012 are shown in Fig. 3, each subimage corresponds to the event anomaly proportion r_{act} is 0.1, 0.3, 0.5 and 0.7 respectively during traing. Where each

curve represents the different case anomaly proportion r_{case} at the time of testing and the horizontal coordinate is the different event anomaly proportion r_{act} at the time of testing. From the figure, it can be seen that injecting different event anomaly proportion during training has a greater impact on the results, and for the lower the number of anomalous events, the worse the classification effect is, so a lower event anomaly proportion is needed to select during training.
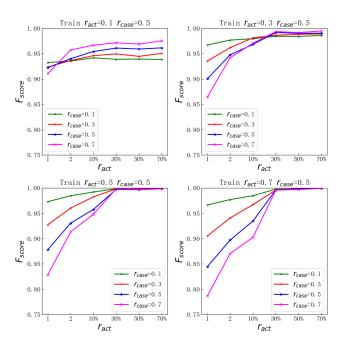

Fig. 3. Results of anomalous case classification task on BPIC2012.

The anomalous proportion setting in the experiment of anomaly correction is the same as the experiment of anomalous case classification, and the results are as follows Fig. 4. The normal (dotted dashed line) indicates the change to the traces originally labeled as normal, and it can be found that TransformerAE almost does not change the normal traces, i.e., it can distinguish the normal traces from the anomalous traces well, and at the same time the effect on the normal traces can be almost negligible even though the model may make a mistake in discriminating the normal traces. The original (short dashed line) indicates the similarity between the anomalous traces and the traces before anomalies are injected, and the solid line is the similarity between the corrected traces and the traces before anomalies are injected. This can be shown that TransformerAE has some ability to correct the anomalous traces, and the larger the event anomaly proportion is, the more obvious is the effect of correction. When the event anomaly proportion is 0.3 during training, the result of anomaly cor-

rection decreases more dramatically for higher event anomaly proportion (70%) during testing, while the result of testing decreases less with event anomaly proportion of 0.5 and 0.7 for training. This shows that for the anomaly detection task and the anomaly correction task a balance needs to be found so that both achieve better results.
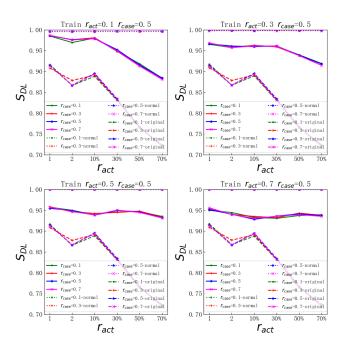

Fig. 4. Results of anomalous correction task on BPIC2012.

The comparison with the baseline methods in the anomaly detection task is shown in the Table 3, in order to make the experimental setting close to BINet, therefore the event anomaly proportion are set to 0.3 for both training and testing. It can be seen that BINetv1 shows similar performance on all logs and are lower than the autocoder-like methods, which may be due to the fact that BINet is designed to be used on the conditions that the anomalies are unknown in the original logs, and therefore the detection is less effective. In the anomaly detection task, the results of TransformerAE and BERT are basically the same, this is due to the fact that the anomaly detection part of TransformerAE is same to BERT. The results of AE are much worse than that of other models. Combined with the result in Fig. 5, this may be because AE changes both anomalous traces and normal traces, resulting in close reconstruction errors between normal traces and anomalous traces, so that it is hard to distinguish them well. The results of LSTMAE are better than AE because the feature learning ability of LSTM for event sequences is much better than that of fully connected neural networks.
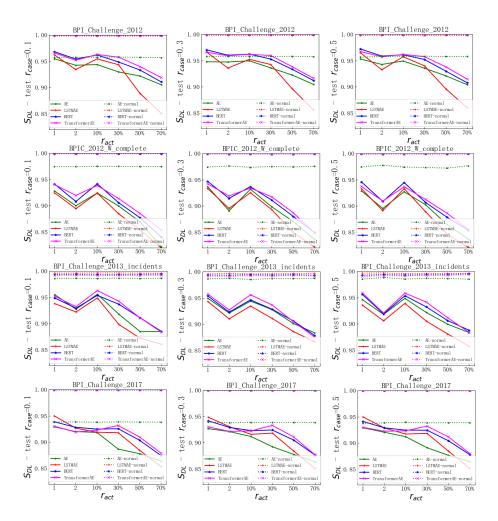
However, the threshold setting based on the average reconstruction error can only give slightly better results when the proportion of normal traces and anomalous traces is close to each other, and the normal traces and anomalous traces can't be well distinguished in many cases.

Table 3. Comparison of anomaly detection tasks with baseline methods (F-score).

Event logs	BINet	AE	LSTMAE	BERT	TransformerAE
BPIC2012	0.5725	0.5874	0.9215	0.9958	0.9958
BPIC2012W	0.5898	0.7633	0.9661	0.9484	0.9479
BPIC2013in	0.5767	0.6287	0.8533	0.9375	0.9375
BPIC2017	0.5636	0.6778	0.9842	0.9997	0.9997

The comparison with the baseline methods in the anomaly correction task is shown in Fig. 5, where the dashed line is the correction result of the trace originally labeled normal. From the figure, it can be seen that the model based on the self-attention mechanism is better than LSTM and fully-connected neural networks in learning sequence context features, and Transformer-AE has the best correction results for anomalous cases when the events anomaly proportion is more than 10%, and the results of BERT are slightly lower than those of TransformerAE, whereas BERT's results are better than TransformerAE's results when the events anomaly proportion is extremely low, e.g., only have 1 or 2 anomalous events, and the results of both TransformerAE and BERT are better than AE and LSTMAE. For all event logs, AE changes the normal trace greatly, and the result is much lower than the other three models, which also affects the anomaly detection accuracy of the threshold setting method based on average reconstruction loss. That is, AE cannot learn the difference between normal and anomalous trace features well. Because the detection accuracy cannot reach 100%, the traces that are classified as normal cannot simply be recorded directly into the information system without any changes, but the output of normal traces should be as close as possible to the original traces. In contrast, the other three models all make very small changes to the normal traces, and therefore have low reconstruction errors, meaning that they all learn the differences between normal and anomalous trace features better.

The time spent by each method to process all the test cases in the anomaly detection task is as follows Table 4 shows. From the results, it can be seen that the autoencoder-like methods is much more efficient than the detection method based on time step prediction (BINet), and the models based on the self-attention is also more efficient than the LSTM. In addition TransformerAE has half more Transformer Block than BERT for improving the effect of anomaly correction, so the running time in the anomaly detection task, the two is same.

Fig. 5. Comparison of anomaly correction tasks with baseline methods.

Table 4. Comparison of time spent on each event log in the anomaly detection task (seconds).

Event logs	BINet	AE	LSTMAE	BERT	TransformerAE
BPIC2012	8.6565	0.0026	0.1174	0.0123	0.0225
BPIC2012W	4.8477	0.0013	0.0266	0.0057	0.0104
BPIC2013in	3.1322	0.0013	0.0437	0.0059	0.0108
BPIC2017	19.3421	0.0055	0.2633	0.0260	0.0456

5   Conclusion

In order to achieve efficient detection of anomalies and at the same time correct the anomalies in them. This paper proposes a method for business process anomaly detection and anomaly correction based on Transformer autoencoder by self-supervised learning. Firstly, we create anomalous trace samples through original traces, transform the anomaly detection problem into a classification problem so that we do not need to set the threshold for anomaly detection, utilize the self-attention mechanism to process the complete trace at one time, and can better learn the features of the event sequence context and combine the anomaly detection and anomaly correction tasks based on the structure of the Transformer, and achieve the end-to-end modeling in the anomaly detection and its correction at the same time. Experimental results on four event logs of different sizes show that the proposed method outperforms achieving state-of-the-art in both anomaly detection and anomaly correction tasks. In the future, we will consider event attributes of business process for multi-view business process anomaly detection.

References

1. van der Aalst, W.M., Carmona, J.: Process mining handbook. Springer Nature (2022)
2. 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: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings. pp. 80–98. Springer (2016)
3. Bukhsh, Z.A., Saeed, A., Dijkman, R.M.: Processtransformer: Predictive business process monitoring with transformer network. arXiv preprint arXiv:2104.00721 (2021)

4. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate lstm models of business processes. In: Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17. pp. 286–302. Springer (2019)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems **100**, 129–140 (2017)
7. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
8. Huo, S., Völzer, H., Reddy, P., Agarwal, P., Isahagian, V., Muthusamy, V.: Graph autoencoders for business process anomaly detection. In: Business Process Management: 19th International Conference, BPM 2021, Rome, Italy, September 06–10, 2021, Proceedings 19. pp. 417–433. Springer (2021)
9. Ko, J., Comuzzi, M.: Detecting anomalies in business process event logs using statistical leverage. Information Sciences **549**, 53–67 (2021)
10. Ko, J., Comuzzi, M.: Keeping our rivers clean: Information-theoretic online anomaly detection for streaming business process events. Information Systems **104**, 101894 (2022)
11. Ko, J., Comuzzi, M.: A systematic review of anomaly detection for business process event logs. Business & Information Systems Engineering pp. 1–22 (2023)
12. Krajsic, P., Franczyk, B.: Variational autoencoder for anomaly detection in event data in online process mining. In: ICEIS (1). pp. 567–574 (2021)
13. Lee, S., Lu, X., Reijers, H.A.: The analysis of online event streams: Predicting the next activity for anomaly detection. In: International Conference on Research Challenges in Information Science. pp. 248–264. Springer (2022)
14. Liu, J., Xu, J., Zhang, R., Reiff-Marganiec, S.: A repairing missing activities approach with succession relation for event logs. Knowledge and Information Systems **63**(2), 477–495 (2021)
15. 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)
16. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Binet: Multi-perspective business process anomaly classification. Information Systems **103**, 101458 (2022)
17. Nolle, T., Seeliger, A., Mühlhäuser, M.: Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In: Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings 19. pp. 442–456. Springer (2016)
18. Nolle, T., Seeliger, A., Thoma, N., Mühlhäuser, M.: Deepalign: alignment-based process anomaly correction using recurrent neural networks. In: International conference on advanced information systems engineering. pp. 319–333. Springer (2020)
19. Pauwels, S., Calders, T.: An anomaly detection technique for business processes based on extended dynamic bayesian networks. In: Proceedings of the 34th ACM/SIGAPP symposium on applied computing. pp. 494–501 (2019)
20. Philipp, P., Jacob, R., Robert, S., Beyerer, J.: Predictive analysis of business processes using neural networks with attention mechanism. In: 2020 International conference on artificial intelligence in information and communication (ICAIIC). pp. 225–230. IEEE (2020)

21. Sani, M.F., van Zelst, S.J., van der Aalst, W.M.: Repairing outlier behaviour in event logs using contextual behaviour. Enterprise Modelling and Information Systems Architectures (EMISAJ) **14**, 5–1 (2019)
22. Sarno, R., Sinaga, F., Sungkono, K.R.: Anomaly detection in business processes using process mining and fuzzy association rule learning. Journal of Big Data **7**, 1–19 (2020)
23. Suriadi, S., Andrews, R., ter Hofstede, A.H., Wynn, M.T.: Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information systems **64**, 132–150 (2017)
24. Tavares, G.M., da Costa, V.G.T., Martins, V.E., Ceravolo, P., Barbon Jr, S.: Leveraging anomaly detection in business process with data stream mining. iSys-Brazilian Journal of Information Systems **12**(1), 54–75 (2019)
25. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with lstm neural networks. In: Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings 29. pp. 477–492. Springer (2017)
26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
27. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J.: Process compliance measurement based on behavioural profiles. In: Advanced Information Systems Engineering: 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings 22. pp. 499–514. Springer (2010)
28. van Zelst, S.J., Sani, M.F., Ostovar, A., Conforti, R., La Rosa, M.: Detection and removal of infrequent behavior from event streams of business processes. Information Systems **90**, 101451 (2020)