IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 30 April 2024, accepted 17 May 2024, date of publication 30 May 2024, date of current version 30 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3406633

RESEARCH ARTICLE

Business Process Anomaly Detection and Root Cause Analysis Using BLSTM-VAE With Attention

EMAN ABD EL-AZIZ[1], RADWA FATHALLA[1], YASSER ISMAIL[2], AND MOHAMED SHAHEEN[1]
[1]College of Computing and IT, Arab Academy for Science, Technology and Maritime Transport, Alexandria 21625, Egypt
[2]Electrical Engineering Department, Southern University and A&M College, Baton Rouge, LA 70807, USA

Corresponding author: Eman Abd El-Aziz (eman.abdelaziz@aast.edu)

ABSTRACT Detecting anomalous executions in business process data is crucial for safeguarding the efficiency and success of an organization. Unsupervised approaches are commonly used for business process anomaly detection because of the scarcity of labeled anomaly data. However, these approaches often encounter a notable decline in performance because they lack prior knowledge about the anomalies. Additionally, most of them do not perform root cause analysis on the detected anomalies. This study proposes a variational autoencoder-based approach to overcome the performance limitations of existing unsupervised methods and determine the root causes of the detected anomalies. The learning of the variational autoencoder from unlabeled business process data is enhanced in the proposed approach by leveraging different architectural components, namely, the entity embedding technique, the bidirectional long short-term memory network, and the self-attention mechanism. Combining these architectural components in the variational autoencoder architecture leads to learning high-level representations from the business process data and thus improving the reconstruction capability of the variational autoencoder. Furthermore, this study suggests feeding the reconstruction error provided by the variational autoencoder into the logistic regression classifier to improve the accuracy of anomaly detection. The performance of the proposed model was evaluated on real-life and synthetic datasets. The experimental findings indicate that the proposed model outperforms six existing anomaly detection models in terms of precision, recall, and F1-score metrics.

INDEX TERMS Anomaly detection, bidirectional long short-term memory, business process, root cause analysis, entity embedding, logistic regression, process-aware information systems, self-attention, variational autoencoder.

I. INTRODUCTION

Process-aware information systems (PAISs) have emerged as a highly relevant subject in the information system domain. Currently, several companies depend on PAISs to enhance their business operations [1]. The increasing adoption of PAISs has sparked significant attention to the data generated by them. The log files stored by the PAISs can be utilized to extract the executed events and generate event log files.

An event log can be described as the primary data structure for business process data, providing details about the activities performed, the individuals involved, the timestamps, etc. [2]. Each event log is further divided into traces, with each trace representing a single process execution. Essentially, the event log captures the activities that occurred in each process execution, enabling process analysts to explore and analyze the underlying processes [1].

A business process can be defined as a series of activities initiated by an event aimed at accomplishing specific organizational objectives. The complexity of a business process varies according to the number of unique activities involved. It can be relatively short or lengthy. Longer processes often exhibit more interdependencies between activities [1], [2], [3]. Within the realm of business process analysis, anomalies can manifest as atypical process executions or noise present in event logs. These anomalies can arise from various sources,

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

including system errors, data entry mistakes, or fraudulent activities [4].

The presence of anomalous business process executions can result in substantial financial losses. According to a report by the Association of Certified Fraud Examiners in 2022, there were more than 2,110 fraudulent business process executions across 133 countries, leading to a cumulative loss of more than $3.6 billion. Organizations experienced a 5% reduction on average in their gross profit annually due to fraud [5]. However, manually identifying anomalies in business process event logs is a time-consuming and labor-intensive task that is susceptible to human errors given the large amount of data involved. Consequently, there is a pressing need for an automated approach capable of detecting anomalies within extensive business process logs.

The interpretability of an automated anomaly detection model is crucial for users to understand and trust its predictions. This is particularly important in the field of business process analytics, where there is a demand for an interpretable model that can conduct a root cause analysis on detected anomalies. By providing direct insights into why a specific trace is flagged as anomalous, such a model can help prevent major future problems. Additionally, the availability of such models can significantly reduce the manual effort needed by process analysts for verification [6].

A. OVERVIEW

Because of the absence of labeled anomalous data in the business process domain, most anomaly detection methods used are unsupervised. However, these methods often struggle to perform effectively because they lack prior knowledge of the anomalies. As a result, there is a need for a new method that can effectively operate within the constraints of available data and achieve higher accuracy than existing methods.

Additionally, most of the current methods for detecting anomalies in business process data do not determine the cause of the anomalous behaviors that occurred in the data. Instead, they either display to the user the executions that are identified as anomalous or provide an anomaly score that indicates the extent of deviation from the anticipated behavior. However, this approach may not be adequate for determining the exact cause of anomalous execution to the end user. Therefore, it is crucial to interpret the detected anomalies to analyze them in terms of the specific event during a process execution that is responsible for the anomaly.

The aim of this study is to propose an innovative approach to overcome the accuracy limitations of the current methods and enable it to determine the root causes of the detected anomalies. To achieve that aim, we must address several challenges in the business process anomaly detection and root cause analysis areas. These challenges are as follows:

- Efficiently processing business process data is challenging due to the complex sequential nature of such data and the numerous direct short-term dependencies between activities within each trace [1], [2], [3], [4].
- Reasoning about anomalies is further complicated by the long-term spans of dependencies in many cases.
- Anomalies in business processes are difficult to detect even for human experts because activities are performed in the context of potential errors, which are considered a normal part of the process [6].
- Acquiring labeled anomalous data in the business process domain is challenging because the available business process datasets are typically unlabeled.
- Many real-life datasets either do not have a predefined process model that includes information about their process logic or, if they do, are often outdated and not regularly updated.
- Obtaining comprehensive data that encompasses all possible classes of anomalies is challenging. Therefore, the ability to generalize to unseen anomalies in training data is crucial, as anomalies are unpredictable and can be costly to overlook.
- Identifying meaningful features to represent an event is difficult because a single event in a sequence lacks sufficient information to support anomaly detection and root cause analysis tasks. Additionally, an anomalous event can result from a breakdown in the correlation between events, necessitating the consideration of the entire sequence to identify anomalous events within their context.

B. CONTRIBUTIONS

We introduce a novel approach based on a variational autoencoder (VAE) to overcome the limitations of the current approaches and the challenges of the research area. We chose VAEs for building our approach because they can be used as generative models [7], [8]. They have an important feature, which is that they can be trained without using class labels and can learn from unlabeled data, thus, overcoming the unavailability of labeled data challenge in the business process domain. Moreover, training the VAE in an unsupervised way enables it to generalize to anomalies not seen in the training data and detect various anomalies, as it is not limited to specific anomaly classes.

Additionally, The VAE is trained in our study to derive the process logic from the event log itself and leverages the patterns found in the event log to distinguish normal traces from anomalous traces and identify root causes of the anomalous traces, thereby overcoming the lack of a predefined process model.

We employ a bidirectional long short-term memory (BLSTM) network in the VAE architecture because of its ability to model the sequential nature of business process data effectively and learn the long-term dependencies between the activities. We further enhance the learning of the long-term dependencies and improve our model's focus on the exact event that caused the anomaly by combining the self-attention mechanism with the BLSTM. The self-attention mechanism enables deep neural networks to capture global dependencies between sequences of events and corresponding outputs for powerful general-purpose representation learning [9].

Furthermore, we apply the entity embedding technique to represent the events in the business process data by mapping them to highly informative vectors to enable the proposed approach to capture the relations between the activities and their respective process behaviors. Moreover, we suggest feeding the output provided by the VAE into the logistic regression (LR) classifier to separate anomalous events from normal events and anomalous traces from normal traces effectively and reduce the false positives.

In summary, this paper has the following four major contributions:

1) A novel framework that combines the VAE with the LR classifier for anomaly detection and root cause analysis in the business process domain.
2) An enhanced VAE architecture that incorporates different architectural components, such as entity embedding, BLSTM, and self-attention, is proposed for learning high-level representations from the business process data.
3) We suggested utilizing the LR classifier to find the best threshold automatically and maximize the accuracy of anomaly detection.
4) The proposed framework is instantiated into an automated model. The model is evaluated on real-life and synthetic datasets, and the results demonstrated that the proposed model outperformed all the other methods in terms of performance.

II. RELATED WORK

This section reviews the studies conducted on both business process anomaly detection and root cause analysis of the detected anomalies.

A. BUSINESS PROCESS ANOMALY DETECTION RELATED WORK

The anomaly detection methods utilized in the business process domain can be categorized into three main categories: process mining, statistical, and machine learning methods.

Process mining, as described in [10], often involves utilizing discovery techniques to extract process models from business process data. Subsequently, conformance checking is performed to detect anomalies by comparing the process model with the actual executions [11], [12], [13], [14], [15], [16]. However, this approach has certain drawbacks. First, this approach can be excessively stringent, resulting in a high rate of false positives. Second, its effectiveness relies on the availability of clean datasets, meaning that no anomalies are present during the discovery phase. Unfortunately, this is rarely the case because business process datasets frequently have anomalies.

Subsequently, the approaches evolved into statistical methods, assuming that the data originated from a theoretical statistical distribution [17]. However, this assumption does not always hold for business process data. Additionally, statistical methods are often inadequate for handling multidimensional data. The statistical methods for anomaly identification provide a stochastic model that represents the behavior being analyzed. If a previously unpredictable behavior falls into the low-probability region of the stochastic model, it is considered an anomaly. To train the statistical model, normal data were utilized. Solti and Kasneci [17] employ this approach to detect temporal anomalies.

Numerous research studies have employed machine learning techniques, including rule mining and clustering, to detect abnormal executions in business process data. Approaches based on rule mining generate rules that extract the behavior of normal process executions. Consequently, if the tested trace does not conform to these rules, it is classified as an anomaly [4], [18], [19]. Sarno et al. [18] utilized association rule mining to identify anomalies in business processes. However, this research has certain limitations, such as the reliance on manually constructed rules (such as user-defined maximum expected duration of activity) and the absence of verification of dependencies between activities. Additionally, Sarno and Sinaga [19] proposed the use of an ontology to provide an understanding of the typical behavior of the business process.

Additionally, several researchers have combined process mining and association rule learning to develop a combined model. One such study was proposed by Sarno et al. [4]. They presented combining process mining, multiattribute fuzzy decision making, and fuzzy association rule learning for identifying anomalies in business process data. The researchers employed the conformance-checking technique, a process mining method, to evaluate the consistency between the process model and the event log. Subsequently, they utilized multiattribute fuzzy decision-making to compute the anomaly rate. Then, fuzzy association rule learning was applied to determine the association rules that could be utilized for anomaly detection.

Rule mining methods offer the advantage of generating a concise set of rules that can be employed for anomaly detection. However, these methods have a disadvantage in the context of flexible business processes, as the generated rules may enforce strictness, resulting in inaccurate outcomes. Moreover, most association rule learning approaches depend on expert intervention. To achieve accurate anomaly detection, experts should assess whether each trace is normal or anomalous and analyze the generated rules.

Due to the unavailability of labeled anomaly data, most machine learning studies focused on detecting anomalies in business processes have employed unsupervised techniques. Among these techniques, clustering algorithms have been commonly utilized. These studies aimed to detect normal and anomalous trace clusters by calculating the distance between multiple traces. Furthermore, Zhu et al. [20] mapped resource behaviors into vectors to calculate the distance between them. They assumed that resources assigned to the same roles would exhibit similar behavior.

Likewise, Hsu et al. [21] employed a similar approach, but they clustered traces based on the temporal perspective, assuming that the same activities exhibit consistent behavior across all traces in terms of their execution time. Additionally, Folino et al. [22] utilized a clustering technique that extracts patterns for trace clustering, such as control flow patterns. Additionally, Böhmer and Rinderle-Ma [23] introduced a clustering likelihood graph-based method. This method produces a reference model that calculates the likelihood of a trace. If the tested trace is not the same as the others, it is identified as an anomaly. The downside of this method is that it needs a clean business process dataset (i.e., no anomalous traces in the dataset).

B. BUSINESS PROCESS ANOMALY DETECTION AND ROOT CAUSE ANALYSIS RELATED WORK

The methods proposed for detecting anomalies and conducting root cause analysis in the business process field can be classified into process mining and machine learning approaches. Only a few studies have utilized process mining techniques to address both anomaly detection and root cause analysis tasks. for instance, Calderón-Ruiz et al. [24] presented an approach that involves comparing behavioral patterns extracted from two event logs: one containing successful traces and the other containing failed traces. This comparison is performed from both control flow and time perspectives through the sequence diagram analysis algorithm. Any differences identified in these patterns can indicate potential causes of business process failures.

Machine learning approaches can be categorized into classification approaches, rule mining approaches, and unsupervised deep learning approaches. Many classification-based methods employ decision trees to identify anomalies and their causes in business process data. One such study was proposed by Suriadi et al. [25] to enhance event logs for root cause analysis using a classification algorithm. They utilized decision trees specifically to detect the causes of overtime errors.

Vasiliev et al. [26] presented an approach to detect the underlying cause of delays in business process datasets. This method utilizes a logical representation of the dataset and applies decision tree induction to classify traces based on their duration. Similarly, Ferreira et al. [27] focused on uncovering the causes of process delays using logical decision trees. Gupta et al. [28] also proposed a method to identify the cause of anomalies in business process datasets. They initially employed a window-based technique to detect anomalous traces and subsequently applied a decision tree classifier to generate rules that describe these traces and can be utilized to explain the causes of the anomalies. However, the classification methods have a limitation which is that they are based on correlation rather than causal relationships, resulting in suboptimal performance.

Rule mining techniques have been utilized in various studies to detect anomalies and their causes in business processes. Böhmer and Rinderle-Ma [29], [30] proposed utilizing association rule mining to discover interesting relationships between activities. Initially, a group of association rules is extracted from the business process dataset, ensuring that each trace in the dataset satisfies these rules. To detect anomalous traces, a new trace is compared to all traces in the dataset, and the most similar trace in terms of control flow is identified. The rules associated with the most similar trace are then applied to the new trace. If the support of the new trace is less than the support of the most similar trace, it is considered an anomaly. One advantage of rule mining techniques is that they explain why a particular trace is classified as an anomaly. However, manual analysis by an expert is still needed to identify the point of divergence.

Deep unsupervised approaches for detecting anomalous business process executions have recently been used because of the unavailability of labeled data and their ability to learn from unlabeled data [31], [32], [33]. Nolle et al. [31] introduced an unsupervised approach based on deep neural networks. The authors divided the datasets from both control flow and data perspectives. Both perspectives are utilized as inputs to the architecture, which aims to predict the next event from both perspectives. Anomalous traces are identified when there is a difference between the encountered next event and the predicted next event. This approach has a limitation, which is that it cannot address many attributes because the learning cannot handle unseen values from both the control flow and data perspectives.

Nguyen et al. [32] employed three distinct autoencoder architectures to reconstruct missing events in the dataset and eliminate anomalies. The first architecture is a standard autoencoder (AE). The second architecture is a variational autoencoder (VAE). In the third architecture, recognizing that business process data are temporal data, the authors incorporated an LSTM network into the standard autoencoder structure, which they named the LSTM autoencoder (LAE).

Furthermore, Nolle et al. [33] introduced an approach for detecting anomalous traces in business process executions using a denoising autoencoder (DAE). This approach also aimed to determine which event in the trace was anomalous relative to the entire trace. The authors trained the autoencoder on an unlabeled dataset and subsequently used it to reconstruct traces in the test set. To detect anomalies, they computed the reconstruction error, which was the mean squared error between the input trace and the reconstructed trace. The assumption behind this approach was that the autoencoder would reconstruct normal traces with lower reconstruction errors than would reconstruct anomalous traces, as the distributions of the normal and anomalous traces in the dataset differed. By setting a threshold, traces with a reconstruction error exceeding this threshold were considered anomalies. The authors further improved the approach by modifying the computation of the reconstruction error from trace-based to event-based, enabling the determination of which event in the trace caused the anomaly.

Both the approach presented by Nolle et al. [33] and our approach detect anomalies on the trace level and the event level in terms of reconstruction errors. However, there are essential distinctions between their approach and our approach. First, we used a different autoencoder model from theirs. Second, in our autoencoder‘s architecture, we employed various architectural components that they did not use such as entity embedding, BLSTM, and self-attention. These design choices have led to an increase in accuracy, as verified by empirical evaluation. Third, the method of calculating the reconstruction error based on the trace level and the event level in our study is more reliable than the method used in [33] since we compute the event reconstruction error using the cross-entropy loss and the trace reconstruction error as the means of the event reconstruction errors.

The accuracy of deep unsupervised learning approaches in detecting anomalies and performing root cause analysis is a common limitation. Another limitation is the reliance on manual threshold setting to differentiate normal events from anomalous events and normal traces from anomalous traces as well, which can be unreliable when applied to diverse datasets. As a reference, we provide a taxonomy of the approaches adopted for the business process anomaly detection and root cause analysis tasks and their limitations in Table 1.

TABLE 1. A summary of related work.

Approaches		Limitations
Process mining approaches [24]		-Their performance is dependent on the availability of a clean dataset and a predefined 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. -They use signatures of known anomalies that can occur in the system, so they cannot recognize a new type of anomaly.
Machine learning approaches	Classification approaches [25-28]	-They are based on correlation and not causality therefore the performance of these approaches cannot be guaranteed.
	Rule mining approaches [29, 30]	-The generated rules still have to be analyzed by an expert to find the point of divergence to detect the cause of the anomaly.
	Deep unsupervised approaches [31-33]	-They suffer from relatively low performance because they do not have prior knowledge about the anomalies. -They rely on manual threshold setting to distinguish normal events from anomalous events and normal traces from anomalous traces, so they strive to set a threshold for all datasets they use.

III. DATASETS

In the business process modeling principle, the fundamental data structure is the event log, which is composed of traces containing events that occur in a business process. Each event is assigned an activity name, and a sample of an event log is displayed in Table 2, representing a purchase-to-pay (P2P) business process. The event log should contain a minimum of three columns: a trace ID for unique activity assignments, a timestamp for sorting activities, and an activity name for differentiation purposes.

TABLE 2. A part of a P2P process event log.

Trace ID	Timestamp	Activity
1	2015-03-21 12:38:39	PR created
1	2015-03-28 07:09:26	PR released
1	2015-04-07 22:36:15	PO created
1	2015-04-08 22:12:08	PO released
1	2015-04-21 16:59:49	Goods receipt
2	2015-05-14 11:31:53	SC created
2	2015-05-21 09:21:26	SC purchased
2	2015-05-28 18:48:27	SC approved
2	2015-06-01 04:43:08	PO created

To assess our model, we utilized two real-life event logs sourced from the business process intelligence challenge: BPIC12 [34] and BPIC17 [35]. Alongside the real-life event logs, we generated two synthetic event logs using Processes and Logs Generator 2 (PLG2) [36]. These synthetic logs were based on process models of varying complexity. The first event log was derived from a P2P process model, shown in Fig. 1 as a BPMN model. The second event log was generated from a random process model, with the complexity determined by the number of activities within each model.

Information regarding the datasets (i.e., the event logs) is shown in Table 3.

TABLE 3. Datasets information.

Name	#Traces	#Activities
Dataset1	10K	13
Dataset2	10K	20
BPIC12	13K	36
BPIC17	31K-43K	8-26

Adding artificial anomalies to real-life and synthetic event logs is a common practice [14], [23], [31], [33]. This is done under the assumption that real-life event logs often contain a few anomalous traces. Hence, we utilized real-life event logs to evaluate the feasibility of the proposed model and synthetic event logs to evaluate its accuracy.

By introducing artificial anomalies, we were able to establish the ground truth of the dataset, which served as the basis for testing our model. In our study, we incorporated commonly used artificial anomalies in business processes,

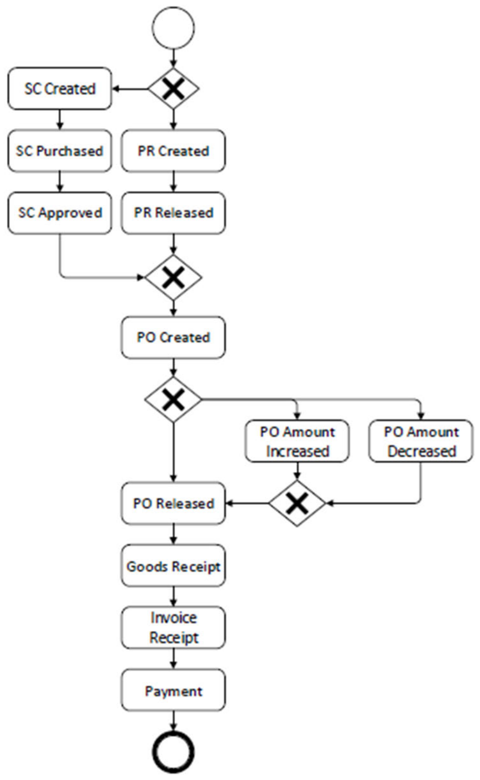


FIGURE 1. **BPMN model of a P2P process.**

as identified in previous studies [14], [23], [31], [33]. These artificial anomalies are:

- Rework: An activity is executed twice.
- Skip: An activity is not executed.
- Switch: Two activities are swapped during execution.

IV. PROPOSED APPROACH

In this section, we introduce a VAE approach to solve the low-performance problem of the current unsupervised techniques and determine the root causes of the detected anomalies.

A. PREPROCESSING

We converted the datasets into a numerical format suitable for the neural network. We accomplish this by employing label encoding, which assigns a positive integer to each unique activity. In cases where the trace length is shorter than the maximum trace length in the dataset, we pad the trace with zeros to match the maximum trace length. This approach is necessary to ensure the proper functioning of the model. Subsequently, we partitioned the real-life and synthetic datasets into training and testing sets. A certain percentage of the traces in the test set were randomly selected and introduced with artificial anomalies while retaining their original labels. This allowed us to assess the performance of our model.

B. PROPOSED FRAMEWORK

We chose the VAE to be the base of our framework because several previous studies have demonstrated the effectiveness of autoencoders as deep unsupervised anomaly detection methods [37], [38], [39], [40], [41]. An autoencoder is a kind of neural network that learns to encode data from a dataset. It comprises an encoder and a decoder [37]. The encoder network receives the input and provides a compressed representation Z with a lower dimensionality than the original input. This latent representation Z is subsequently fed into the network of the decoder, which is trained to reconstruct the input.

The VAE is a type of autoencoder that incorporates additional constraints on the latent representation Z [7], [8]. These constraints force the learned features of Z to roughly follow a specific probability distribution. This property is particularly useful when using VAE as a generative model, as it allows for the generation of novel outputs that are similar to the original input by sampling values from the distribution and feeding them into the decoder.

It is a common practice for autoencoders-based anomaly detection methods to train the autoencoders using only normal samples [32], [33], [42]. The advantage of using autoencoders for anomaly detection is that they can learn in unsupervised settings.

This implies that they do not need labeled data that specifically points out anomalies during the training phase. Instead, they learn from the distribution of the normal data, enabling them to identify anomalies that may not have been encountered during training.

This fundamental idea underlies the utilization of autoencoders for anomaly detection. Throughout the training phase, the autoencoder is provided with a set of normal data, then it learns to represent the normal patterns of this data in the latent space. It becomes adept at accurately encoding and decoding normal data. In essence, the autoencoder learns the underlying patterns and structure of the normal data to generate a reliable reconstruction.

During the testing phase, if the autoencoder encounters anomalous data, it will be unable to reconstruct it as precisely as the normal data. Consequently, the reconstruction error, which quantifies the difference between the original input and the reconstructed output, can serve as the anomaly score. It typically exhibits a notable increase for anomalous data in comparison to normal data. As a result, data samples with high reconstruction errors are considered to be anomalous samples.

In the same way, we train the VAE solely on normal traces from the event log to learn the normal behavior of the process. Once trained, the VAE can be used to detect anomalies. When an input trace deviates from a normal trace, the VAE fails to reconstruct the input trace of the same quality as the normal trace.

Therefore, the anomaly score or reconstruction error can be computed as the difference between the input trace and

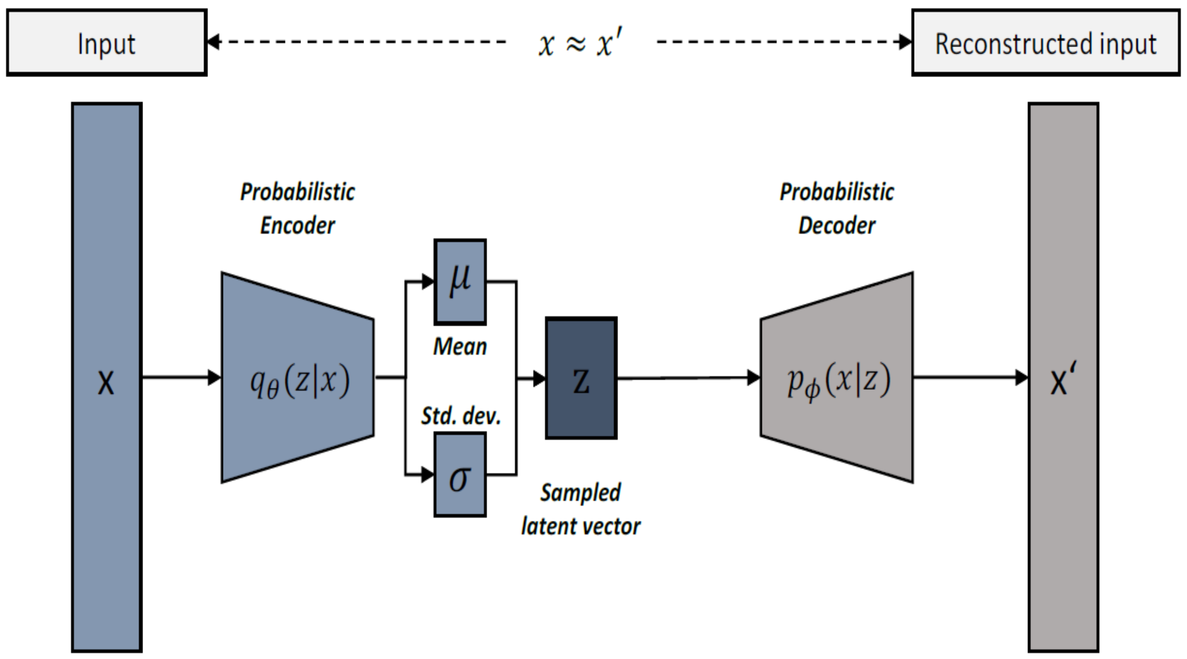


FIGURE 2. The VAE framework.

the reconstructed trace. In this way, the VAE can indicate that something is amiss without needing to be specifically trained on anomalous traces.

Fig. 2 [8] illustrates the structure of the VAE, which consists of two main components:

- Probabilistic Encoder: The encoder receives a data point X and generates a hidden representation Z using weights and biases θ. The goal of the encoder is to learn an effective compression of the data into a compressed representation Z. The encoder is denoted as $q_\theta(z|x)$. It outputs parameters for a Gaussian probability distribution, capturing the mean μ and the variance σ of the distribution. The representation Z is then sampled from this distribution.
- Probabilistic Decoder: The decoder receives Z and generates the parameters for the data probability distribution using weights and biases φ. We denote the decoder as $p_\varphi(x|z)$.

The encoder of the VAE produces two vectors that depict the mean and variance of the latent distribution from which the representation Z is randomly sampled. Nevertheless, there is a problem with the random sampling of the Z representation during training, as backpropagation gradients do not function effectively over random processes [43].

To address this issue, the sampling process can be pushed out as the input by employing a novel technique known as the reparameterization trick. Initially, a random vector ε is sampled from a Gaussian distribution with dimensions equivalent to Z. This vector is then multiplied by the variance vector σ of the latent distribution and then the mean vector μ is added to it, as shown in Fig. 3 [43].

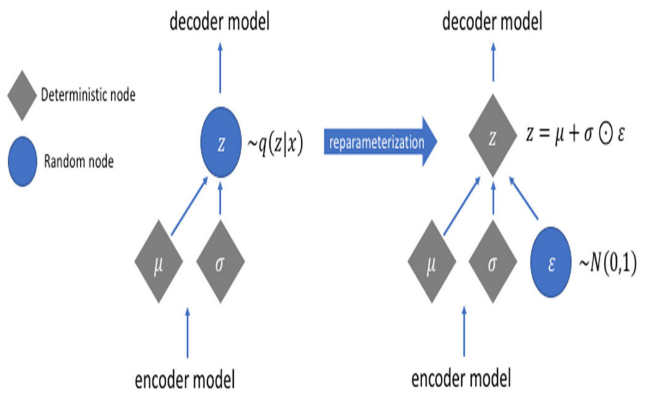


FIGURE 3. The VAE reparameterization trick.

This adjustment removes the random generator from the backward pass, ensuring that the sampled data maintains the properties of the original distribution. The improved sampling process can now be denoted as follows:

$$z = \mu + \sigma \odot \varepsilon \tag{1}$$

The VAE loss function consists of two main components: the reconstruction loss and the Kullback–Leibler (KL) divergence term.

- The reconstruction loss quantifies the VAE's ability to reconstruct the input from Z. This approach encourages the VAE to learn a meaningful representation of the data that can be used to accurately reconstruct the original input.

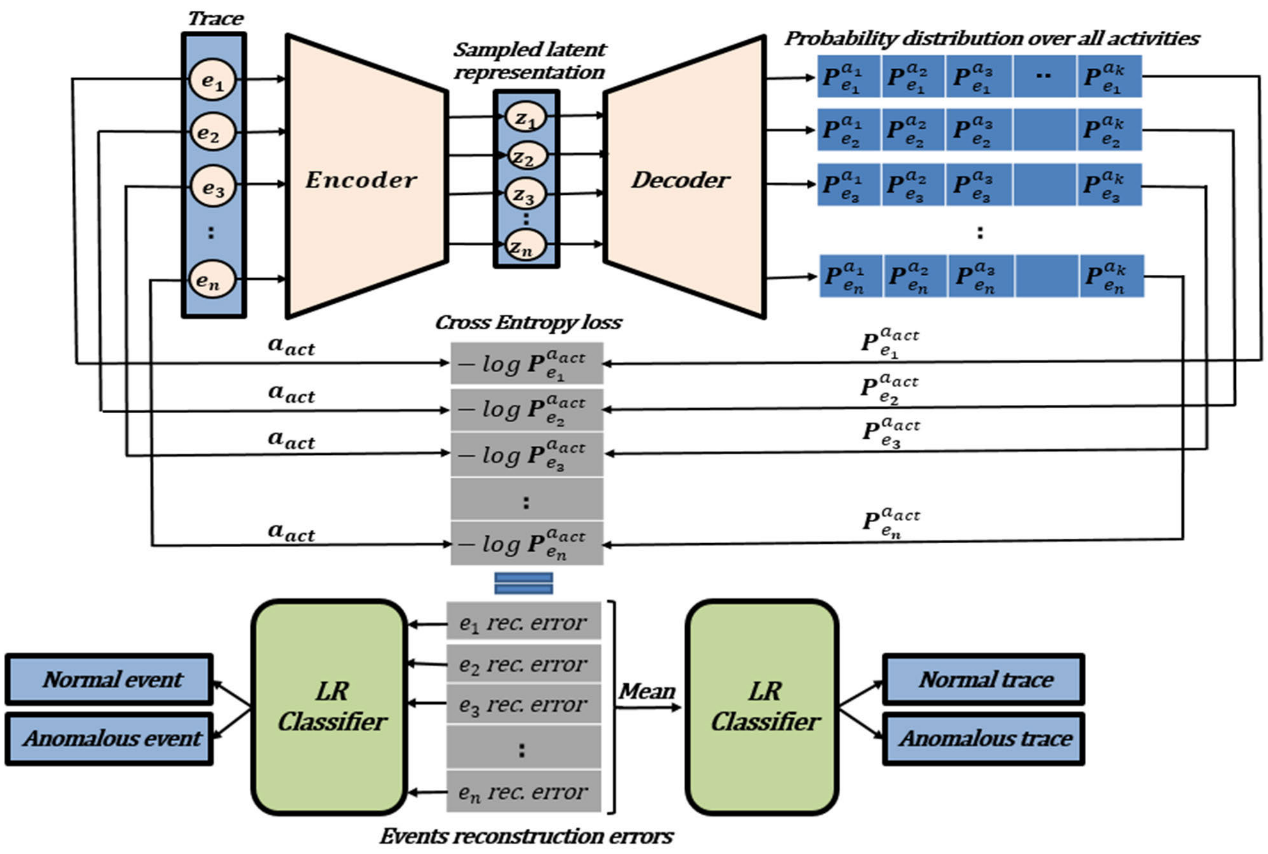

Trace
Encoder
Sampled latent representation
Decoder
Probability distribution over all activities
Cross Entropy loss
a_{act}
$-\log P^{a_{act}}_{e_1}$
$-\log P^{a_{act}}_{e_2}$
$-\log P^{a_{act}}_{e_3}$
$-\log P^{a_{act}}_{e_n}$
e_1 rec. error
e_2 rec. error
e_3 rec. error
e_n rec. error
Mean
LR Classifier
Normal event
Anomalous event
Normal trace
Anomalous trace
Events reconstruction errors

FIGURE 4. Illustration of the proposed framework.

- The KL divergence term calculates the discrepancy between the assumed prior distribution and the learned latent distribution. It is a regularization term that promotes the VAE to align the learned latent space with the desired distribution.

These components are combined to form the overall loss function that the VAE aims to minimize during the training phase. The equation for the loss function of the VAE can be defined as follows:

$$Loss = Reconstruction\ loss + KL(q_\theta\ (x_i)\ ||p(z)) \quad (2)$$

The proposed framework as shown in Fig. 4 is composed of two main components: a VAE and a LR classifier. As clarified in the figure, the VAE receives the integer-encoded trace or sequence of events $(e_1, e_2, e_3, \ldots, e_n)$, where n represents the length of the trace. This trace is fed into the first part of the VAE, which is the encoder. Then, the encoder outputs parameters for a Gaussian probability distribution, capturing the mean μ and the variance σ of the distribution. The latent representations $(z_1, z_2, z_3, \ldots, z_n)$ are then randomly sampled from this distribution. After that, the latent representations are fed into the decoder. Next, the decoder outputs a probability distribution P^a_e over all possible values of the activity attribute a of event e to identify anomalous events.

The assumption is that an anomalous activity attribute will be assigned a lower probability than a normal activity attribute of the event. That is, if the dataset has 10 different activity values, then P^a_e will be a first-order tensor with a size of 10. Each dimension of P^a_e contains the probability that is assigned to one of the 10 possible activity values. To measure the reconstruction error RE of an input event, the cross-entropy loss function is used. This function takes the negative logarithm of the probability of the actual activity a_{act} encountered in that event or the probability that the z representation of that event can be reconstructed to the input event $p_\varphi(event_n|z)$, as in (3). The logarithm is applied to magnify the reconstruction error for low-probability activities.

$$RE\ (event) = -\log P^{a_{act}}_{event} = -\log p_\varphi\ (event\ |\ z) \quad (3)$$

It is important to note that the reconstruction error of the padding events will always be zero. The reconstruction error of the whole trace is computed by taking the mean of the reconstruction errors of all events within that trace. Algorithm 1 describes the VAE training process.

Once the trace reconstruction error is obtained, it needs to be mapped to a binary label $y \in \{0, 1\}$, where 0 represents a normal trace and 1 represents an anomaly. We propose using a classifier to find the best threshold value for all tested event logs to distinguish between anomalous and normal traces

Algorithm 1 VAE training

```
Input: Unlabelled normal trace set $U$
Output: Encoder parameters $\theta$, Decoder parameters $\varphi$
e = event
t = trace
l = trace length
b = batch size
Randomly initialize $\theta$, $\varphi$
  for $i = 1$ to $n\_epochs$ do
    for $j = 1$ to $n\_batches$ do
    $\{t_k\}_{k=1}^{b} \leftarrow$ Randomly sample $b$ traces from $U$
      for $k = 1$ to $b$ do
        for $n = 1$ to $l$ do
         $\mu, \sigma \leftarrow$ Encoder $(\theta, e_n)$
         $\varepsilon \leftarrow N(0,1)$
         $z = \mu + \sigma \odot \varepsilon$
         $p_{\varphi}(e_n|z) \leftarrow$ Decoder $(\varphi, z)$
         $RE(e_n) = -\log p_{\varphi}(e_n|z) + KL(q_{\theta}(z|e_n)\,||p(z))$
        end for
      $RE(t_k) = \frac{1}{l}\sum_{n=1}^{l} RE(e_n)$
      end for
    $Loss = \sum_{k=1}^{b} RE(t_k)$
    Update the parameters using gradients of loss
    end for
  end for
return $\theta$, $\varphi$
```

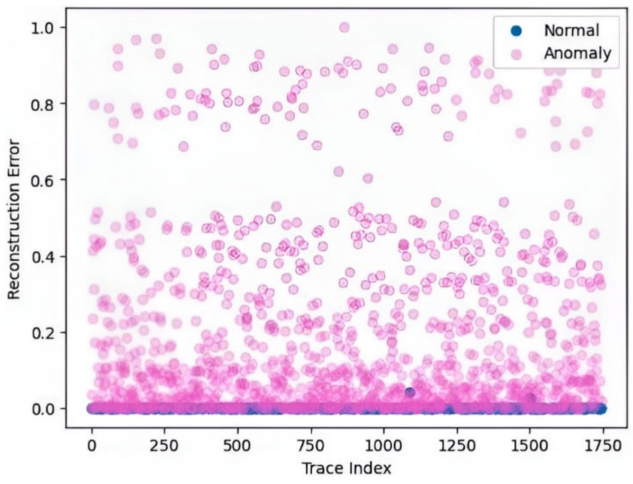


FIGURE 5. The reconstruction error distributions of the normal and anomalous traces produced by the VAE trained on PBIC12 dataset.

Algorithm 2 LR training

```
Input: Labeled set $S = \{(r_1, y_1), \ldots, (r_n, y_n)\}$
Output: Parameters $w$ and $b$
$r$ = reconstruction error
$y = \{1, 0\}$
c = batch size
Randomly initialize $w$ and $b$
for $i = 1$ to $n\_epochs$ do
  for $k = 1$ to $n\_batches$ do
    $\{r_j, y_j\}_{j=1}^{c} \leftarrow$ Randomly sample mini-batch from $S$
    for $j = 1$ to $c$ do
    Compute the linear combination of $r$ and parameters $w$ and $b$.
    $x_j = w * r_j + b$
    Apply sigmoid function to get predicted probabilities.
    $y\_pred_j = \text{Sigmoid}(x_j)$
    end for
  Compute the loss function (Binary cross-entropy loss).
  $Loss = \frac{-1}{c}\sum_{j=1}^{c} (y_j * log(y\_pred_j) + (1-y_j) * log(1 - y\_pred_j)$
  Update $w$ and $b$ using gradients of loss.
  end for
end for
return $w$ and $b$
```

automatically. The reconstruction error of the trace is fed into the classifier, which is trained to map the reconstruction error to the label.

In the same way, we detect anomalous events by feeding the event reconstruction error into the classifier to determine the cause of the detected anomalous traces. Additionally, it is essential to utilize a classifier capable of handling non-linearly separable classes, as the normal and anomaly classes in real-life business process event logs are often not linearly separable.

Fig. 5 shows the reconstruction error values of the normal, and anomalous traces produced by the VAE trained on the PBIC12 event log. It can be observed that they cannot be separated by a straight line. For this reason, we opted for the utilization of LR classifier due to its ability to effectively separate non-linearly separable classes [44].

LR classifier is a statistical method that estimates the probability that an observation belongs to a specific class [44]. It employs a logistic function, commonly referred to as the sigmoid function which is a non-linear function to map the model's output to the probability space.

The sigmoid function transforms the output into probabilities, which are then converted into binary outcomes using a decision threshold, typically set at 0.5. If the probability exceeds the threshold, the observation is classified as belonging to the positive class; otherwise, it is classified as belonging to the negative class.

The sigmoid function is defined in (4). x represents the input value, and e denotes the natural logarithm base (or Euler's number). It outputs an S-shaped curve, which can transform any real-valued number into a value ranging from 0 to 1 [45].

$$Sigmoid\ (x) = \frac{1}{1 + e^{-x}} \tag{4}$$

The sigmoid function's characteristic S-shaped curve enables it to handle data points effectively, being steeper around the decision threshold and gradually flattening towards the extremes (0 and 1). This property ensures a smooth transition of probabilities around the decision boundary.

The training and classification procedures of LR classifier are described in algorithm 1 and algorithm 2 respectively.

In our study, we present a novel architecture for VAEs that improves the ability of VAEs to accurately reconstruct business process data and effectively detect anomalies. This enhanced capability allows us to analyze and identify the specific event within a business process execution that is causing the anomaly.

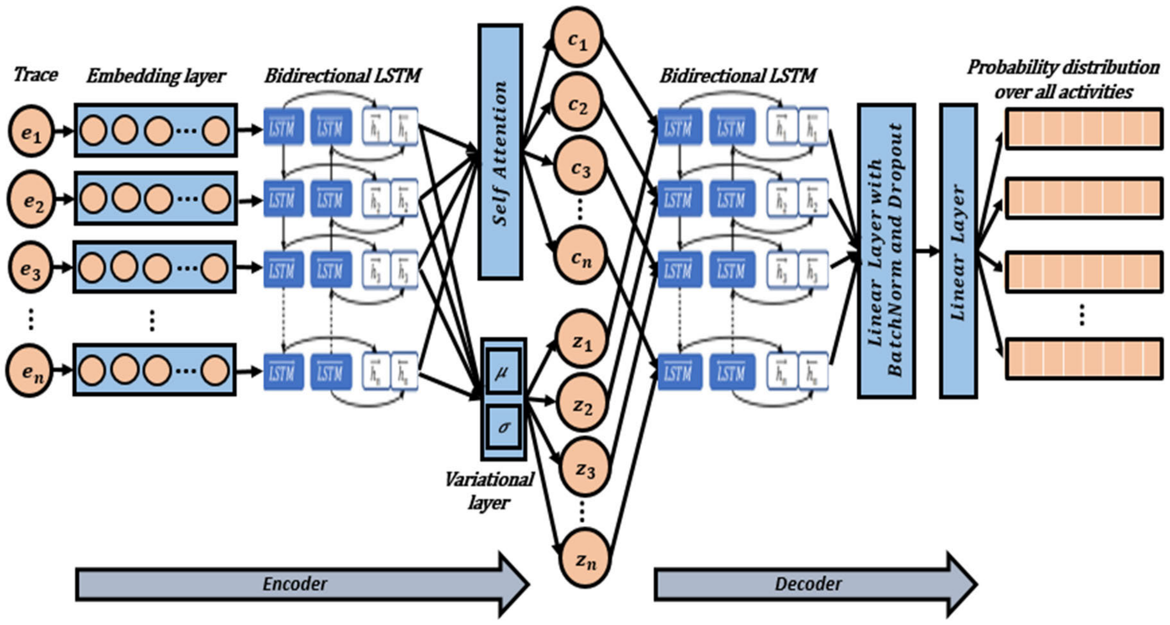


FIGURE 6. The proposed SA-BLSTM-VAE architecture.

```
Algorithm 3 LR classification label decision
Input: reconstruction error $r$
Output: $y = \{1, 0\}$
$x = w * r + b$
$y\_pred = Sigmoid(x)$
  if $y\_pred >= 0.5$
    $y = 1$ (Anomaly)
  else
    $y = 0$ (Normal)
return $y$
```

The learning of VAEs from unlabeled data is improved by leveraging different architectural components, namely, the entity embedding technique, BLSTM, and self-attention mechanism, and combining them in the VAE architecture.

We call our model the self-attention and bidirectional long short-term memory-based variational autoencoder (SA-BLSTM-VAE).

The proposed SA-BLSTM-VAE architecture is shown in Fig. 6. As clarified in the figure, the encoder of the SA-BLSTM-VAE receives the integer-encoded sequence of events or trace $t = (e_1, e_2, e_3, \ldots, e_n)$. The encoder consists of the following architectural components:

1) ENTITY EMBEDDING LAYER

To represent the events, their relations, and the corresponding business process behaviors, we utilized entity embedding. This technique allows us to capture the essential information while constraining the size of the feature vectors. The entity embedding technique mainly comes from natural language processing (NLP) and information retrieval fields to generate highly informative but low-dimensional vectors [46].

The entity embedding technique, similar to word embedding in NLP, converts words to vectors in a multidimensional space, where similar entities are placed close to each other. In our work, we apply this technique to the input trace or sequence of events. The encoder utilizes an embedding layer that generates fixed-size vectors with random numbers for each activity.

During VAE training, these random values are adjusted through backpropagation, the embedding layer is optimized to learn the intrinsic properties of the activities, and the loss function is minimized. This means that the embedding vectors are computed within the same architecture in our model, and the embedding layer weights are learned in the training phase.

Fig. 7 provides an example of how the embedding layer maps each integer-encoded activity to a five-dimensional embedding vector.

Initially, the integer-encoded activity is transformed into a one-hot encoding vector. The adjustable weights of the embedding matrix are updated in the training phase of the VAE. Subsequently, the embedding layer maps each activity to a five-dimensional vector corresponding to a row in the embedding matrix.

2) BLSTM

Considering the temporal nature of the business process data, we opted to utilize recurrent neural networks (RNNs) for modeling. RNNs excel at capturing sequential temporal data due to their ability to retain temporal information through recurrent connections. These connections allow the output

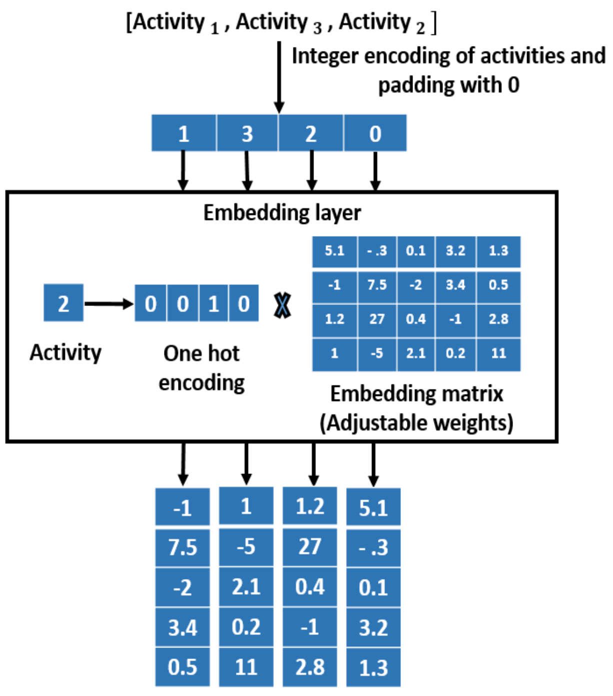


FIGURE 7. An example of how the embedding vectors of the activities are computed.

of a recurrent layer to be fed back as input, creating a causal temporal chain and preserving an internal state or "memory" [47], [48]. Among the various RNN architectures, long short-term memory (LSTM) is the most robust option [48], [49].

To further enhance the modeling capabilities, we employed a BLSTM architecture for both the encoder and decoder of our VAE. A BLSTM trains two LSTMs simultaneously: one on the original input sequence and the other on a reversed copy of the sequence. This approach provides additional context and facilitates more comprehensive learning of the sequences [50].

The BLSTM in the encoder, as shown in Fig. 6, receives the embedding vectors generated from the embedding layer and produces hidden state sequences forward and backward for these embeddings. The output sequence $\vec{h}$ of the forward LSTM layer is iteratively calculated using the input embeddings in the forward sequence, and the output sequence $\overleftarrow{h}$ of the backward LSTM layer is computed based on the reversed input embeddings.

These hidden states enable the network to maintain past information and thus be able to remember complex long-term dependencies between activities.

In this way, the input trace can be viewed from two directions, thus the extrastructural properties of this trace can be learned. The hidden states produced by the BLSTM are subsequently combined into a hidden state vector sequence $(h_1, h_2, h_3, \ldots, h_n)$ and fed into the variational layer for additional processing.

Then, the statistical variables μ (mean) and σ (variance) are generated in the variational layer based on the hidden state vectors of the BLSTM, representing the normal distribution. From such a defined normal distribution, the latent representations of this normal distribution $(z_1, z_2, z_3, \ldots, z_n)$ are sampled.

3) SELF-ATTENTION LAYER

We incorporated an attention mechanism with BLSTM in the encoder network to achieve optimal modeling of the business process data and to enhance the learning of long-range dependencies between activities. The mechanism of neural attention was first presented in [48] for NLP. It can imitate the visual attention mechanism of humans [51]. The human eye focuses on certain objects or areas with a higher resolution than their surroundings.

In the same way, the attention mechanism enables the neural network to attend to out-of-order elements of a sequence, emphasizing the significance of one element or another in the sequence; thus, it generates a better representation of that sequence [9], [51].

The attention mechanism utilized in our model is self-attention, which is chosen for its ability to capture dependencies between elements in a sequence and maintain coherence across long-term sequences [9]. Self-attention has been applied successfully in various tasks, including sentence representation learning and textual entailment [52], [53].

In the self-attention layer, each input element interacts with the others to determine the attention it should give to different elements. The output is a weighted sum of these interactions, represented by attention scores.

As shown in Fig. 6, the self-attention layer receives the hidden state vectors $(h_1, h_2, h_3, \ldots, h_n)$ produced by the BLSTM as input and generates context vectors $(c_1, c_2, c_3, \ldots, c_n)$, where each context vector is a weighted sum of the input hidden state vectors.

To compute the context vectors, the self-attention mechanism first generates three vectors for each input vector: query q, key k, and value v; all of the dimension d. These vectors are obtained by multiplying each input vector with weight matrices specific to keys, queries, and values.

The self-attention function maps each query to a group of key-value pairs and computes a weighted sum of the values, resulting in the context attention vector C. We applied the scaled dot-product attention from [9], to calculate C as follows:

$$Attention\,(Q, K, V) = C = Softmax(\frac{QK^t}{\sqrt{d_k}})V \tag{5}$$

The q, k, and v vectors are packed into the Q, K, and V matrices, respectively. The attention weights are calculated by taking the dot product of the query Q with all the keys K^t, scaled by dividing by $\sqrt{d_k}$, i.e., the dimension of k. This scaling ensures stable gradient computation [9].

The resulting weights are then normalized using the softmax function to obtain the importance of each input vector.

Finally, the attention weights are multiplied by the V matrix to obtain the attention vector C.

In Fig. 8, we provide a more detailed explanation of the self-attention mechanism using an example from the P2P process. The self-attention layer takes the hidden state vectors of a trace containing six events as input. For each hidden state vector, q, k, $andv$ vectors are generated.

To calculate the attention vector $C\,(j)$ for the PO Created event, we want to determine its relation with the other events in the trace. To do this, we compute the scaled dot-product of the query vector for the PO Created event with the key vectors of all the input hidden state vectors.

This operation produces an attention scores vector or weights $a\,(j)$ for each input hidden state vector. Next, we obtain the attention vector $C\,(j)$ by taking the weighted sum of the attention scores vector $a\,(j)$ with the corresponding value vectors. This step combines the information from the other events in the trace to create the attention vector for the PO Created event.

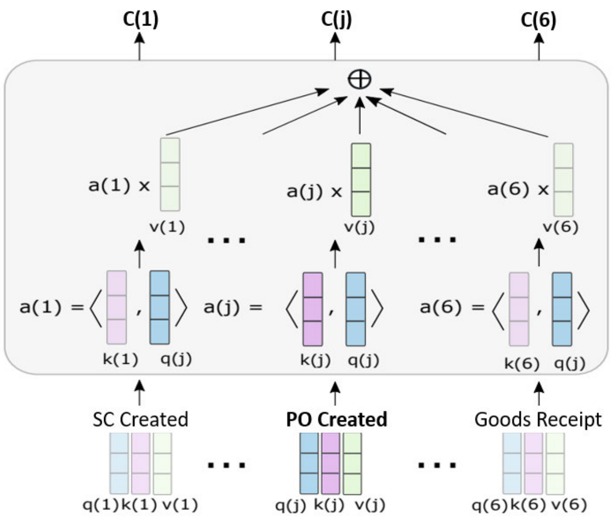


FIGURE 8. An example of how the self-attention mechanism learns the attention representation vector of the PO Created event.

Then, the context vectors generated from the self-attention layer are sent to the decoder for additional processing. In this way, the decoder is notified of the contextual information provided by the context vectors; hidden states information is important and should be considered. In other words, we pass contextual information from the hidden states of the encoder to the decoder.

Together with the context vectors, the latent representations from the variational layer are fed into the decoder. The decoder is composed of a BLSTM network and two fully connected (FC) layers. Like in the encoder case, the BLSTM produces hidden states. Then, these hidden states are passed to the first fully connected layer.

Autoencoders are intended to reconstruct the output vector from the input vector. They may learn the identity function between all the input and output vectors in the training set, leading to overfitting. To solve this problem in our VAE architecture, we utilized batch normalization [54] and dropout [55] in the first FC layer in the decoder network. After that, the second FC layer gives a probability distribution P_e^a over all possible activity values a of event e using a softmax activation function.

V. EXPERIMENTAL EVALUATION

We empirically assess the performance of the proposed SA-BLSTM-VAE model using real-life and synthetic datasets. To conduct this evaluation, we divided the datasets into three sets: 70% for training, 20% for testing, and 10% for validation. Since anomalous traces are uncommon in business processes, we add artificial anomalies to only 10% of the testing set.

A. COMPARED METHODS

To prove the effectiveness of the SA-BLSTM-VAE model, we compared it with six other anomaly detection models applied to business process data. These models include sampling [13], DAE [33], BINet [31], AE [32], VAE [32], and LAE [32]. The sampling method is based on process mining and relies on process discovery and conformance-checking techniques, whereas DAE, BINet, AE, VAE, and LAE are deep unsupervised anomaly detection methods.

B. IMPLEMENTATION SETUP

All implementations were carried out in Python utilizing the available source codes of DAE, BINet, AE, VAE, and LAE provided by their respective authors. The sampling method was implemented according to the original paper's description using the Process Mining for Python (PM4Py) library [56].

The SA-BLSTM-VAE model was trained for 100 epochs. A mini-batch size of 256 was used during training. The Adam optimizer was used with a learning rate of 0.1. The architecture and hyperparameters of the SA-BLSTM-VAE model are presented in Table 4.

The LR classifier consists of an input layer, an output layer, and two hidden layers. The number of neurons in the input and output layers is set to 1 and the number of neurons in the two hidden layers is set to 4. The activation function used with the output layer is sigmoid. All used hyperparameters were selected based on empirical experimentation with various values, and it was determined that the chosen values yielded satisfactory results.

C. RESULTS AND ANALYSIS

To assure a fair comparison, the precision, recall, and F1-score metrics were utilized. We begin by outlining a set of relevant terms that will be utilized in these metrics [57]:

- True positive (TP): represents the count of anomalous traces correctly identified as anomalies.
- True negative (TN): denotes the count of normal traces correctly identified as normal.

TABLE 4. The hyperparameter settings of the SA-BLSTM-VAE model.

Network	Layer	Output size	Activation function
Encoder	Embedding layer	Trace length*32	-
	BLSTM	Trace length*8	Sigmoid-tanh
	Self-attention layer	Trace length*8	Softmax
Decoder	BLSTM	Trace length*32	Sigmoid-tanh
	Linear layer	Trace length*32	Relu
	Linear layer	Trace length*no of activities	Softmax

- False positive (FP): refers to the count of normal traces incorrectly classified as anomalies.
- False negative (FN): specifies the count of anomalous traces incorrectly identified as normal.

Precision is a metric that evaluates the proportion of elements classified as anomalies that are indeed true anomalies. It is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Recall is a metric that measures the percentage of true anomalies correctly detected by the model. It is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

F1-score is the harmonic mean of precision and recall, so it is well-suited for assessing the model's effectiveness in accurately detecting anomalies.

$$F1 - score = 2x\frac{RecallxPrecision}{Recall + Precision} \tag{8}$$

Precision assesses the accuracy of results, while recall evaluates the completeness of results. The F1-score aims to strike a balance between precision and recall.

Table 5 displays the accuracy of the anomaly detection, in terms of the precision, recall, and F1-score, of the proposed SA-BLSTM-VAE model and other models across all datasets, considering both the trace-level and the event-level. The best results are highlighted in bold.

Based on the results of the synthetic event logs (Dataset1 and Dataset2) and real-life event logs (BPIC12 and PBIC17) for both trace-level and event-level shown in Table 5, we can categorize the proposed model and the compared models into three groups based on the comparison between precision and recall:

1. Methods with higher precision (AE, LAE, and VAE): These methods demonstrate precision > recall, indicating that they are good at identifying normal traces but may struggle to detect anomalous traces effectively.
2. Methods with higher recall (sampling, DAE, and BINET): These methods demonstrate precision < recall, showing that they are good at identifying anomalous traces but may misclassify normal traces as anomalies, leading to false positives.
3. Balanced method (SA-BLSTM-VAE): Our model exhibits a more balanced performance concerning recall and precision.

According to the precision and recall results reported in Table 5, the average of the differences between precision and recall for all the compared methods on all the datasets is around 0.23. whereas, the average of the differences on all the datasets in the case of our model is 0.03.

From our perspective, one of the reasons behind the imbalance between precision and recall in the other deep learning models (AE, LAE, VAE, DAE, and BINET) is that they have an issue with setting the anomaly threshold. When a small threshold value is set, the detection becomes more sensitive, potentially resulting in false alarms or false positives (lower precision and higher recall), as occurred in DAE and BINET. Conversely, a larger threshold value reduces false positives but may lead to missed anomaly detections (higher precision and lower recall), as occurred in AE, LAE, and VAE.

In contrast, our SA-BLSTM-VAE model achieves a balance between precision and recall because we feed the reconstruction errors obtained from the VAE into LR classifier to find the best threshold, rather than setting it manually.

For the sampling method, the main reason behind the lower precision and higher recall is the fact that it depends on mining a process model from the event log to identify anomalous traces by comparing the difference between the tested traces and the mined process model. Consequently, the sampling method may be overly strict towards the mined process model, which leads to increasing the false positive.

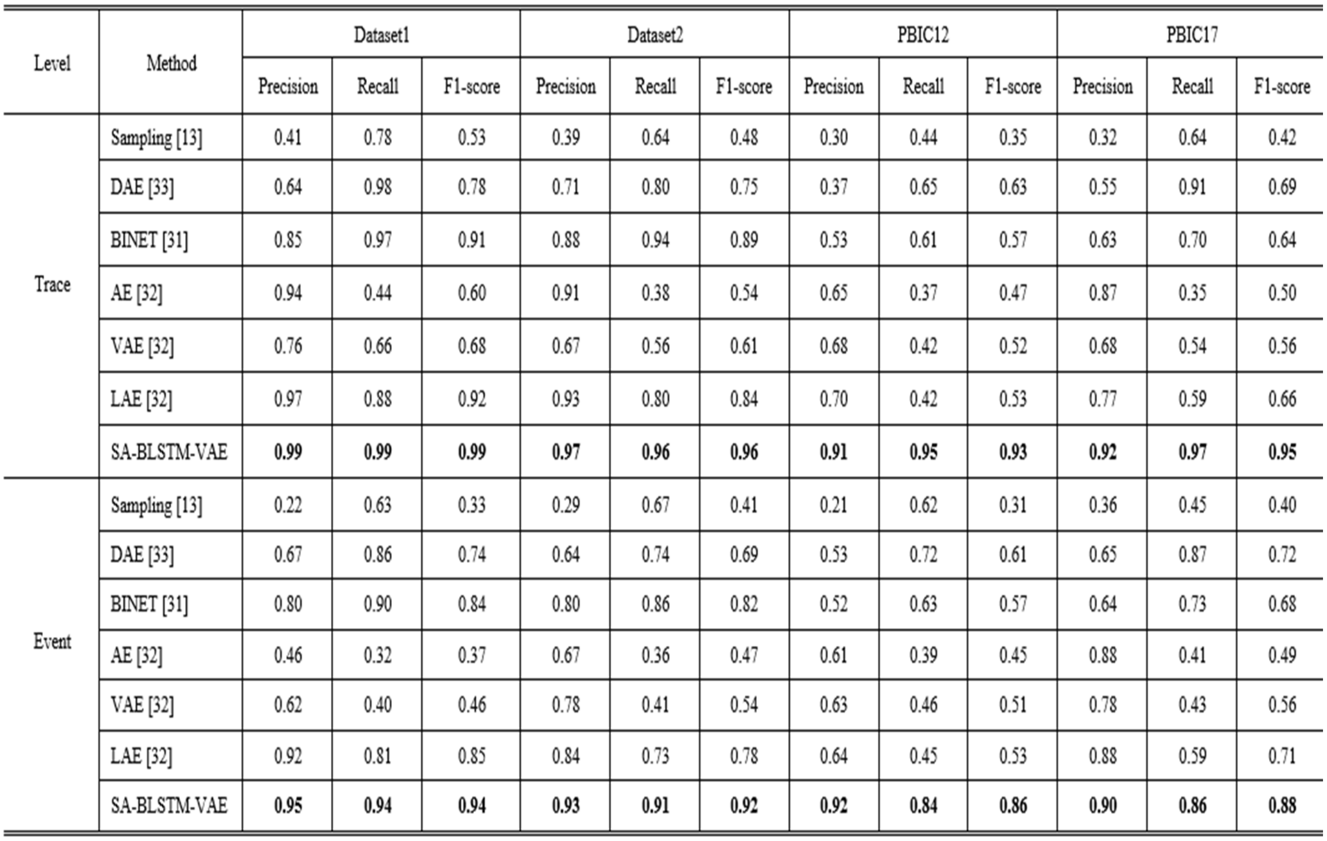
Moreover, our model demonstrates remarkably higher F1-scores compared to other models, whether for trace-level anomaly detection or event-level anomaly detection. The F1-scores of our model consistently hover around 0.9. This implies that for both synthetic and real-life event logs, our model outperforms the other models. Our model is closely followed by LAE and BINet models, whereas the remaining models exhibit significantly lower performance.

Notably, the sampling method performs the worst because the sampling method like any other process mining method utilizes signatures of known anomalies that may occur in the business process, which limits its ability to detect new types of anomalies. In contrast, our SA-BLSTM-VAE model is capable of detecting new anomalies because it is an unsupervised model and is not limited to specific classes of anomalies.

As expected, LAE and BINET outperform AE, VAE, and DAE because they are RNN-models and utilize the temporal dimension in the business process data, whereas DAE and VAE are not RNN-models. LAE employs LSTM to capture the temporal information of the events, whereas BINET uses a Gated Recurrent Unit (GRU), both being RNN variants.

The main advantage of our SA-BLSTM-VAE model over LAE, BINET, and the other models lies in our use of BLSTM, a superior network compared to LSTM and GRU, for capturing the temporal information and the dependency within the event sequences [50].

TABLE 5. The precision, recall, and F1-score for all datasets according to detection level and method.



Level	Method	Dataset1			Dataset2			PBIC12			PBIC17		
		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Trace	Sampling [13]	0.41	0.78	0.53	0.39	0.64	0.48	0.30	0.44	0.35	0.32	0.64	0.42
	DAE [33]	0.64	0.98	0.78	0.71	0.80	0.75	0.37	0.65	0.63	0.55	0.91	0.69
	BINET [31]	0.85	0.97	0.91	0.88	0.94	0.89	0.53	0.61	0.57	0.63	0.70	0.64
	AE [32]	0.94	0.44	0.60	0.91	0.38	0.54	0.65	0.37	0.47	0.87	0.35	0.50
	VAE [32]	0.76	0.66	0.68	0.67	0.56	0.61	0.68	0.42	0.52	0.68	0.54	0.56
	LAE [32]	0.97	0.88	0.92	0.93	0.80	0.84	0.70	0.42	0.53	0.77	0.59	0.66
	SA-BLSTM-VAE	**0.99**	**0.99**	**0.99**	**0.97**	**0.96**	**0.96**	**0.91**	**0.95**	**0.93**	**0.92**	**0.97**	**0.95**
Event	Sampling [13]	0.22	0.63	0.33	0.29	0.67	0.41	0.21	0.62	0.31	0.36	0.45	0.40
	DAE [33]	0.67	0.86	0.74	0.64	0.74	0.69	0.53	0.72	0.61	0.65	0.87	0.72
	BINET [31]	0.80	0.90	0.84	0.80	0.86	0.82	0.52	0.63	0.57	0.64	0.73	0.68
	AE [32]	0.46	0.32	0.37	0.67	0.36	0.47	0.61	0.39	0.45	0.88	0.41	0.49
	VAE [32]	0.62	0.40	0.46	0.78	0.41	0.54	0.63	0.46	0.51	0.78	0.43	0.56
	LAE [32]	0.92	0.81	0.85	0.84	0.73	0.78	0.64	0.45	0.53	0.88	0.59	0.71
	SA-BLSTM-VAE	**0.95**	**0.94**	**0.94**	**0.93**	**0.91**	**0.92**	**0.92**	**0.84**	**0.86**	**0.90**	**0.86**	**0.88**

The self-attention mechanism serves as a bridge between the encoder and the decoder in our model. When the encoder receives an input trace, the BLSTM in the encoder produces a hidden state vector for each event in that trace.

Then, the self-attention layer takes these hidden state vectors and generates a context vector for each event. These context vectors encapsulate information regarding the relation and the dependency between the events in the trace. Subsequently, the context vectors are sent to the decoder i.e. the self-attention mechanism passes the contextual information from the hidden state vectors representing the trace in the encoder to the decoder.

In this way, the decoder is notified of the importance of each event during reconstruction thus the decoder's learning is enhanced. Consequently, the ability of our model to reconstruct normal traces is improved. In other words, the self-attention mechanism empowers our model to reason about long-term dependencies and establish generic representations between input and output traces, thereby enhancing detection performance on both the trace and event levels.

Additionally, AE, LAE, VAE, DAE, and BINET which are deep learning models utilize one-hot encoding to convert the categorical event sequences or traces into numerical representations. These integer representations do not take into account the inherent relations between the events. In contrast, we opt for the entity embedding technique over one-hot encoding to enhance the event representation.

Entity embedding transforms the events into highly informative vectors, allowing our model to effectively capture the relations between the events and their corresponding process behavior. Integrating the architectural components of entity embedding, BLSTM, and self-attention in our model enables the learning of high-level representations from business process data, thereby improving the anomaly detection performance of the model.

It can be noted from Table 5 that the performance of all models on real-life event logs is relatively lower compared to their performance on synthetic event logs. This implies that training the models on real-life event logs results in a decrease in performance. It is worth noting that the performance decline in our model is minimal in comparison to the other models.

Interestingly, the decrease in performance is because real-life event logs are more likely to contain anomalies. Consequently, the models are not able to effectively learn the complex behaviors present in the real-life event logs. Additionally, the sampling method exhibits the worst performance, because the performance of the sampling method heavily relies on the availability of clean event logs.

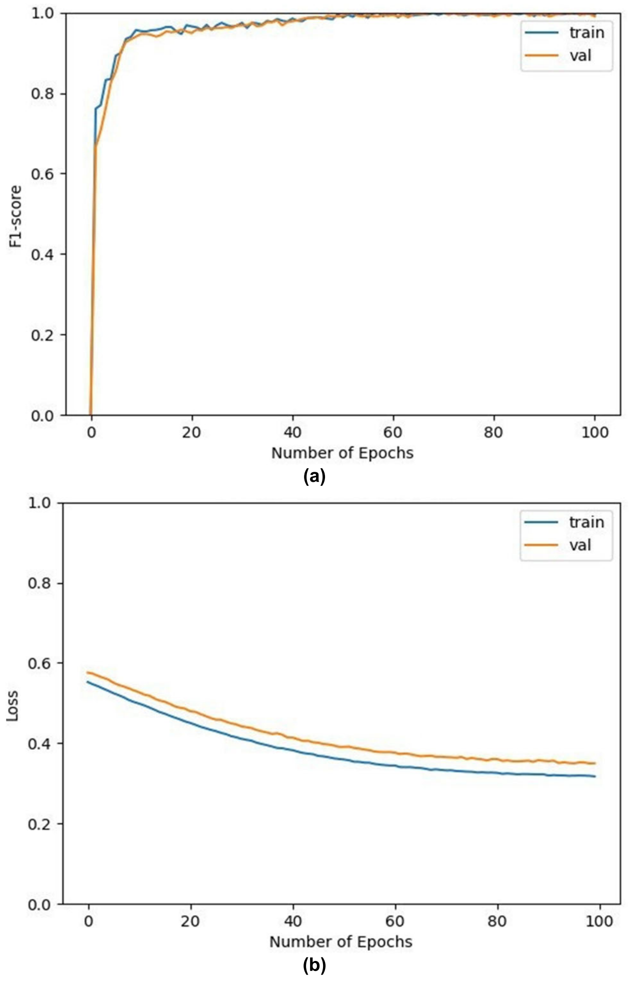


FIGURE 9. **The learning curves of the proposed model on the p2p dataset: (a) the F1-score learning curve (b) the loss learning curve.**

Moreover, real-life event logs are often produced by less structured and more flexible business processes, making it hard for the sampling method to mine the process models from these event logs. In addition, the normal behavior of these business processes is difficult for the deep learning models to learn. Another contributing factor to the decrease in performance is the presence of numerous long-term dependencies in real-life event logs.

Despite these challenges, our model maintains good overall performance, as evidenced by its F1-score on real-life event logs, indicating its effectiveness in detecting anomalies in real-life settings. It can be trained on noisy event logs that contain anomalies without requiring a clean dataset, which is rare in real-life.

Additionally, the incorporation of BLSTM and the self-attention mechanism enables our model to better capture the long-term dependencies in real-life event logs.

The F1-scores and loss learning curves of the SA-BLSTM-VAE model on the p2p dataset at the trace level are shown in Fig. 9.

As shown in the figure, the learning curves are similar for the training and validation sets, which means that our model performs well on both the training and validation data; i.e., it can learn the patterns from the training set and generalize well on unseen new data. In other words, our model does not suffer from underfitting or overfitting, which is the ideal situation for a machine learning model.

VI. CONCLUSION

This paper presents a VAE-based model for anomaly detection and root cause analysis in business processes. The presented model addresses the limitations of the existing unsupervised methods and provides insights into the causes of the detected anomalous executions. The proposed model incorporates various architectural components, including entity embedding, BLSTM, and self-attention, to handle the sequential nature and long-term dependencies of the business process data effectively, enabling the learning of high-level representations from these data. Furthermore, the output of the VAE is fed into the LR classifier to improve the discrimination between normal and anomalous traces, thereby enhancing the detection accuracy. We evaluated our model‘s performance using real-life and synthetic datasets. The results demonstrate that our model outperforms six competing models and exhibits good accuracy in identifying the exact cause of anomalies compared to other approaches.

In future work, we plan to extend the proposed model to incorporate multiple perspectives of the business process, such as the data perspective by further inspecting other attributes of the events not just the order in which the events are executed. i.e. identifying which characteristic of the event (e.g., the executing user) is anomalous. Moreover, event logs now encompass multimedia elements like videos and images, that capture scenes upon completion of an event. These heterogeneous information sources can potentially lead to anomalous process executions. So, in the future, our upcoming investigations will explore the integration of these diverse data types to develop cross-media anomaly detection tailored for business processes.

ACKNOWLEDGMENT

The authors would like to express their gratitude to the editors and anonymous reviewers for their helpful and insightful remarks and suggestions.

REFERENCES

[1] M. Dumas, W. Van der Aalst, and A. Hofstede, *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Hoboken, NJ, USA: Wiley, 2005.
[2] A. Maaradji, M. Dumas, M. L. Rosa, and A. Ostovar, "Fast and accurate business process drift detection," in *Proc. BPM*, 2015, pp. 406–422.
[3] J. Y. Jung, W. Hur, S. H. Kang, and H. Kim, "Business process choreography for B2B collaboration," *IEEE Internet Comput.*, vol. 8, no. 1, pp. 37–45, Aug. 2004.
[4] R. Sarno, F. Sinaga, and K. R. Sungkono, "Anomaly detection in business processes using process mining and fuzzy association rule learning," *J. Big Data*, vol. 7, no. 1, p. 5, 2020.
[5] *Certified Fraud Examiners a Report to the Nations on Occupational Fraud and Abuse: 2022 Global Fraud Study*, Assoc. Certified Fraud Examiners, Austin, TX, USA, 2022.
[6] J. Ko and M. Comuzzi, "A systematic review of anomaly detection for business process event logs," *Bus. Inf. Syst. Eng.*, vol. 65, pp. 441–462, Aug. 2023.

[7] C. Doersch, "Tutorial on variational autoencoders," 2016, *arXiv:1606.05908*.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Con. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[10] W. Aalst, *Process Mining: Data Science in Action*. Berlin, Germany: Springer, 2016.

[11] W. Aalst and A. Medeiros, "Process mining and security: Detecting anomalous process executions and checking process conformance," *Electr. Notes Theor. Comput. Sci.*, vol. 121, no. 4, pp. 3–21, 2005.

[12] L. Wen, W. Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 80–145, 2007.

[13] F. de Lima Bezerra and J. Wainer, "Algorithms for anomaly detection of traces in logs of process aware information systems," *Inf. Syst.*, vol. 38, no. 1, pp. 33–44, 2013.

[14] F. Bezerra, J. Wainer, and W. Aalst, "Anomaly detection using process mining," in *Enterprise, Business-Process and Information Systems Modeling*. Heidelberg, Germany: Springer, 2009, pp. 149–161.

[15] J. Swinnen, B. Depaire, and M. Jans, "A process deviation analysis—A case study," in *Proc. Bus. Process Manag. Workshops*, vol. 99. Heidelberg, Germany: Springer, 2012, pp. 87–98.

[16] G. Li and W. Aalst, "A framework for detecting deviations in complex event logs," *Intell. Data Anal.*, vol. 21, no. 4, pp. 79–759, 2017.

[17] A. Rogge-Solti and G. Kasneci, "Temporal anomaly detection in business processes," in *Proc. 12th Int. Conf. Bus. Process Manage.*, vol. 8659. Cham, Switzerland: Springer, 2014, pp. 234–249.

[18] R. Sarno, R. D. Dewandono, T. Ahmad, M. F. Naufal, and F. Sinaga, "Hybrid association rule learning and process mining for fraud detection," *IAENG Int. J. Comput. Sci.*, vol. 42, no. 2, pp. 59–72, 2015.

[19] R. Sarno and F. P. Sinaga, "Business process anomaly detection using ontology-based process modelling and multi-level class association rule learning," in *Proc. Int. Conf. Comput., Control, Informat. Appl. (IC3INA)*, Oct. 2015, pp. 12–17.

[20] T. Zhu, Y. Guo, J. Ma, and A. Ju, "Business process mining based insider threat detection system," in *Proc. Int. Conf. Adv. P2P Parallel, Grid, Cloud Internet Comput.*, in Lecture Notes on Data Engineering and Communications Technologies, vol. 1, F. Xhafa, L. Barolli, and F. Amato, Eds. Cham, Switzerland: Springer, 2017, pp. 467–478.

[21] P. Hsu, Y. Chuang, Y. Lo, and S. He, "Using contextualized activity-level duration to discover irregular process instances in business operations," *Inf. Sci.*, vol. 391, pp. 80–98, Jun. 2017.

[22] F. Folino, G. Greco, A. Guzzo, and L. Pontieri, "Mining usage scenarios in business processes: Outlier-aware discovery and run-time prediction," *Data Knowl. Eng.*, vol. 70, no. 12, pp. 1005–1029, 2011.

[23] K. Böhmer and S. Rinderle-Ma, "Multi-perspective anomaly detection in business process execution events," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.* Cham, Switzerland: Springer, 2016, pp. 80–98.

[24] G. Calderón-Ruiz and M. Sepúlveda, "Automatic discovery of failures in business processes using process mining techniques," in *Proc. Anais 9th Simpósio Brasileiro Sistemas Informação*, 2013, pp. 439–450.

[25] S. Suriadi, C. Ouyang, W. M. van der Aalst, and A. H. T. Hofstede, "Root cause analysis with enriched process logs," in *Proc. Bus. Process Manage. Workshops*, 2013, pp. 174–186.

[26] E. Vasilyev, D. R. Ferreira, and J. Iijima, "Using inductive reasoning to find the cause of process delays," in *Proc. IEEE 15th Conf. Bus. Informat. (CBI)*, Jul. 2013, pp. 242–249.

[27] D. R. Ferreira and E. Vasilyev, "Using logical decision trees to discover the cause of process delays from event logs," *Comput. Ind.*, vol. 70, pp. 194–207, Jun. 2015.

[28] N. Gupta, K. Anand, and A. Sureka, "Pariket: Mining business process logs for root cause analysis of anomalous incidents," in *Databases in Networked Information Systems*. Cham, Switzerland: Springer, 2015, pp. 244–263.

[29] K. Böhmer and S. Rinderle-Ma, "Association rules for anomaly detection and root cause analysis in process executions," in *Proc. CAiSE*, 2018, pp. 3–18.

[30] K. Böhmer and S. Rinderle-Ma, "Mining association rules for anomaly detection in dynamic process runtime behavior and explaining the root cause to users," *Inf. Syst.*, vol. 90, 2020, Art. no. A101438.

[31] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser, "BINet: Multi-perspective business process anomaly classification," *Inf. Syst.*, vol. 103, Jan. 2022, Art. no. 101458.

[32] T. C. Nguyen, S. Lee, J. Kim, J. Ko, and M. Comuzzi, "Autoencoders for improving quality of process event logs," *Expert Syst. Appl.*, vol. 131, pp. 132–147, Oct. 2019.

[33] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser, "Analyzing business process anomalies using autoencoders," *Mach. Learn.*, vol. 107, no. 11, pp. 1875–1893, 2018.

[34] B. F. van Dongen. *BPI Challenge 2012*. Accessed: Feb. 10, 2022. [Online]. Available: https://doi.org/10.4121/UUID:3926DB30-F712-4394AEBC-75976070E91F

[35] B. F. van Dongen. *BPI Challenge 2017*. Accessed: Feb. 14, 2022. [Online]. Available: https://doi.org/10.4121/uuid:5f3067df-f10b-45dab98b-86ae4c7a310b

[36] A. Burattin, "PLG2: Multiperspective processes randomization and simulation for online and offline settings," 2015, *arXiv:1506.08415*.

[37] T. Amarbayasgalan, B. Jargalsaikhan, and K. H. Ryu, "Unsupervised novelty detection using deep autoencoders with density based clustering," *Appl. Sci.*, vol. 8, no. 9, p. 1468, 2018.

[38] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, "Detection of anomalies in large scale accounting data using deep autoencoder networks," 2017, *arXiv:1709.05254*.

[39] T. Luo and S. G. Nagarajan, "Distributed anomaly detection using autoencoder neural networks in WSN for IoT," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[40] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *Proc. 19th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2018, pp. 125–134.

[41] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining. Acad. Med.*, 2017, pp. 665–674.

[42] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture IE*, vol. 2, pp. 1–18, Dec. 2015.

[43] J. Jordan. *Variational Autoencoders*. Accessed: Jan. 20, 2023. [Online]. Available: https://www.jeremyjordan.me/variational-autoencoders

[44] D. Jurafsky and J. H. Martin, "Logistic regression," in *Speech and Language Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2019.

[45] R. K. Vinayak, W. Kong, G. Valiant, and S. Kakade, "Maximum likelihood estimation for learning populations of parameters," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 11217–11226.

[46] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," 2016, *arXiv:1604.06737*.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[48] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 551–561.

[49] L. Rahman, N. Mohammed, and A. K. Al Azad, "A new LSTM model by introducing biological cell state," in *Proc. 3rd Int. Conf. Electr. Eng. Inf. Commun. Technol. (ICEEICT)*, Sep. 2016, pp. 1–6.

[50] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[51] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[52] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," 2017, *arXiv:1703.03130*.

[53] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model," in *Proc. Empirical Methods Natural Language Process.*, 2016, pp. 2249–2255.

[54] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[56] A. Berti, V. Zelst, and V. D. Aalst, "Process mining for Python (PM4Py): Bridging the gap between process-and data science," in *Proc. CEUR*, 2019, pp. 13–16.

[57] Y. Sasaki, "The truth of the F-measure," MIBSchool Comput. Sci., Univ. Manchester, Manchester, U.K., 2007.

EMAN ABD EL-AZIZ received the B.Sc. and M.Sc. degrees in information systems from Arab Academy for Science, Technology and Maritime Transport (AASTMT), Alexandria, Egypt, in 2009 and 2015, respectively, where she is currently pursuing the Ph.D. degree in information systems. She is also a Teaching Assistant with the College of Computing and Information Technology, AASTMT. Her research interests include business process management, data mining, deep learning, and artificial intelligence.

RADWA FATHALLA received the B.Sc. and M.Sc. degrees in computer engineering from Arab Academy for Science, Technology and Maritime Transport (AASTMT), Alexandria, Egypt, in 2002 and 2007, respectively, and the Ph.D. degree in engineering and applied sciences from Aston University, U.K., in 2017. She has been an Assistant Professor with the Department of Computer Science, College of Computing and Information Technology, AASTMT, since 2018, where she was a Teaching Assistant (TA), from 2003 to 2017.

YASSER ISMAIL received the bachelor's and first master's degrees with a focus on electronics and electrical communication from Mansoura University, Egypt, and the second master's and Ph.D. degrees in computer engineering from the University of Louisiana at Lafayette.

He is an Associate Professor with the Electrical Engineering Department, Southern University and A&M College (SU). He has over 20 years of professional experience in teaching and research. Prior to joining SU, he was a Faculty Member with five international universities, such as Mansoura University, the University of Louisiana at Lafayette, Umm Al-Qura University, the University of Bahrain, and Zewail City of Science and Technology University. He has a broad background in machine learning applications; modeling and design techniques for reliable, low-power, and high-performance VLSI and FPGA systems; cloud computing; cybersecurity; the Internet of Video Things (IoVT); digital video processing algorithms/architectures levels; and wireless and digital communication systems. He has over 66 publications, including books, book chapters, and articles in high-ranked journals and conferences. He served as a PI and a Co-PI for over 12 (17) funded grants from NSF, state, and international fund agencies. He received the Partnering, Research, Innovation, Development, and Entrepreneurship award (P. R. I. D. E) by Southern University. He served as a reviewer for several peer-reviewed conferences and journals. He has been serving as an NSF Panel Reviewer, since 2019.

MOHAMED SHAHEEN received the B.Sc. degree in computer science from Alexandria University, Egypt, in 1991, the first M.Sc. degree in computer engineering from Arab Academy for Science, Technology and Maritime Transport (AASTMT), Alexandria, Egypt, in 1998, and the second M.Sc. and Ph.D. degrees in computer engineering from the University of Louisiana at Lafayette, Louisiana, USA, in 2000 and 2003, respectively. He has been a Professor and the Vice Dean for Training Affairs and Community Service with the College of Computing and Information Technology, AASTMT, since 2010. From 2003 to 2010, he was a Graduate Faculty with the University of Louisiana at Lafayette. From 1999 to 2003, he was an Adjunct Faculty with the Computer Science Department, University of Louisiana at Lafayette. His research interests include software engineering, business process management, and artificial intelligence. Funds from USA federal agents, such as DOE, NSF, and the Louisiana Governor ITI supported his research. He also secured funds from Qatar National Research Fund (QNRF); and ITIDA, Egypt.