GAMA: A Multi-graph-based Anomaly Detection Framework for Business Processes via Graph Neural Networks

Wei Guan, Jian Cao, Yang Gu, Shiyou Qian

Abstract—Anomalies in business processes are inevitable for various reasons such as system failures and operator errors. Detecting anomalies is important for the management and optimization of business processes. In this paper, we propose a multi-**G**raph based **A**nomaly detection fra**M**ework for business processes via gr**A**ph neural networks, named GAMA. GAMA makes use of structural process information and attribute information in a more integrated way. In GAMA, multiple graphs are applied to model a trace in which each attribute is modelled as a separate graph. In particular, the graph constructed for the special attribute *activity* reflects the control flow. Then GAMA employs a multi-graph encoder and a multi-sequence decoder on multiple graphs to detect anomalies in terms of the reconstruction errors. Moreover, three teacher forcing styles are designed to enhance GAMA's ability to reconstruct normal behaviours and thus improve detection performance. We conduct extensive experiments on both synthetic logs and real-life logs. The experiment results demonstrate that GAMA outperforms state-of-the-art methods for both trace-level and attribute-level anomaly detection.

Index Terms—Process mining, anomaly detection, deep learning, graph neural networks, recurrent neural networks

1 Introduction

Anomaly detection, also known as outlier detection, novelty detection, etc., focuses on identifying rare, unexpected and suspicious instances within a swarm of normal data points [1]. With a wide range of applications, including spam detection, financial fraud detection, and intrusion detection in cybersecurity [2], it has major implications for assisting practitioners and decision-makers in discovering, managing, and avoiding anomalous patterns from data. With recent developments in information technology, enterprises are increasingly relying on process-aware information systems (PAISs) to manage their operations. However, anomalies in processes are inevitable due to numerous root causes, such as system failures and operator errors. Detecting anomalies in business processes is critical to the successful operation of a business. Moreover, low-quality event logs containing anomalies hinder our ability to extract valuable information from them. For example, process mining (PM) provides techniques to comprehend and enhance processes in various application fields [3]. The output of process mining techniques using low-quality event logs may be of poor quality, potentially reducing the accuracy of any decisions based on it. Therefore, anomaly detection techniques should be used to detect and remove anomalies from the logs.

Event logs encompass multiple perspectives such as activities, resources, data, and time, with complex intrinsic dependencies between them. At the same time, there are different categories of anomalies in the event logs, such as *Skip*, *Insert*, *Rework*, *Early*, *Late* and *Attribute* [4], where the first five categories of anomalies can be referred to as control flow anomalies, which break the normal control flow dependencies. Anomalies related to resources, data, and time are categorized as *Attribute*. Capturing complex dependencies among multiple perspectives and detecting various categories of anomalies in business processes are challenging.

In recent years, a variety of approaches have been proposed for detecting anomalies in business processes, ranging from graph-agnostic to graph-based methods. Graph-agnostic approaches which treat traces as sequences have been widely studied and can be divided into several types. Machine learning-based approaches [5], [6], [7], [8], [9] transform traces into vector representations and detect anomalies using traditional machine learning anomaly detection algorithms. Information theory-based measures are applied by information theory-based approaches [10], [11], [12], [13] to detect anomalies. With the advancement of deep learning, reconstruction-based approaches [4], [14], [15], [16], [17], [18], [19], [20], [21] have recently attracted increasing attention, where traces with large reconstruction errors are treated as anomalies. It is worth noting that reconstruction-based approaches can detect low-level anomalies, such as attribute-level anomalies.

Although graph-agnostic approaches can be applied directly to the traces, they ignore the structural information of business processes which may affect the anomaly detection performance. Fig. 1 gives a toy example. It can be seen that the two traces are different in terms of their event sequences. However, both can be generated from the same process. Therefore, graph-based approaches, taking the structure of a business process into consideration, have a better generalization capability. Generally, in these approaches, normal process model-based methods are given or discovered from the clean log, and then conformance checking [22] is used to compare the differences between traces and the feasible

• *W. Guan, J. Cao, Y. Gu and S. Qian are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China. E-mail: {guan-wei, cao-jian, gu_yang, qshiyou}@sjtu.edu.cn.*

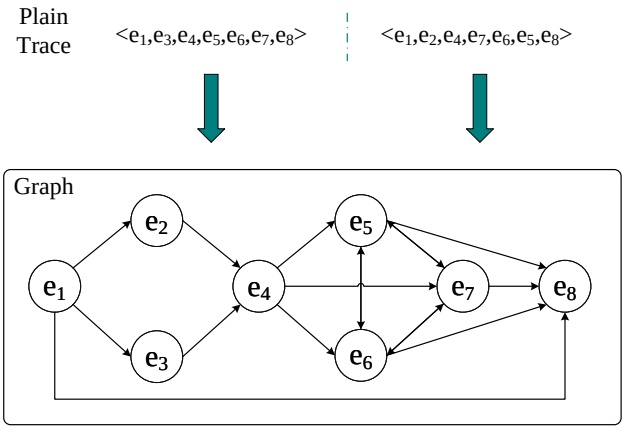


Fig. 1: A toy example of a graph-based approach

behaviors of normal process models for anomaly detection [23], [24], [25], [26]. Since anomalies can be detected from a probabilistic perspective, process models are extended with probabilistic information in the form of a likelihood graph [27], Bayesian networks [28], hidden Markov models (HMM) [29], [30] and variable order Markov models (VOMMs) [31] for anomaly detection. However, these approaches all rely on a clean dataset to construct a normal process model, which is often not feasible in practice. With the rise of graph neural networks, anomaly detection based on graph encoding has aroused researchers' interests [32]. However, the graphs derived from traces often rely solely on control flows, neglecting attribute information. While attributes can be incorporated as edge features in the graph, the underlying relationships between attributes and control flows, specifically the patterns of attribute changes with respect to control flows, are not adequately captured. As a result, existing graph neural network-based methods are ineffective in detecting anomalies at the attribute level. Additionally, the graphs extracted directly from individual traces lack crucial structural information about the underlying process.

To address the aforementioned challenges, in this paper, we propose a multi-**G**raph-based **A**nomaly detection fra**M**ework for business processes via gr**A**ph neural networks, named GAMA, to detect both trace-level and attribute-level anomalies effectively. To obtain a graph with comprehensive structural information about the process, GAMA introduces a unique approach. It derives a multi-graph for each trace by constructing a global graph using the entire event log. This global graph contains detailed structural information that encompasses the entire process. Moreover, within the multi-graph structure, GAMA adopts a meticulous modeling methodology where each attribute is represented as an independent graph. This approach ensures that attribute information is accurately captured and reflected, resulting in a comprehensive representation of the data. GAMA leverages graph neural networks (GNNs) [33] to learn the graph embeddings. These GNNs are integrated into a multi-graph encoder, enabling the extraction of hidden representations for the nodes (attribute values). These hidden representations are subsequently decoded into probabilistic maps using a multi-sequence decoder to detect anomalies. To effectively capture the intrinsic relationships between attributes, GAMA incorporates an attention mechanism that operates across the graphs for each attribute. Three innovative teacher forcing styles that take into account the unique characteristics of business processes are introduced. These customized techniques are specifically designed to accommodate the unique properties and intricacies of business processes. By leveraging these tailored teacher forcing methods, GAMA improves the accuracy and effectiveness of the decoding process.

The main contributions of our work are as follows:

- We propose a groundbreaking method to convert a trace into a multi-graph which proficiently capture both the structural information of the process and attribute information in a comprehensive manner.
- We design the framework GAMA, which uses a multi-graph encoder and a multi-sequence decoder on the multi-graph to detect anomalies in terms of reconstruction errors.
- Considering the characteristics of business processes, three teacher forcing styles are designed to enhance the capability to reconstruct normal behavior and improve detection performance.
- We perform extensive experiments on both synthetic and real-world datasets. The experiment results show that GAMA achieves the best performance in detecting trace-level and attribute-level anomalies.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 gives the preliminaries and notations used in our study. Then proposed framework GAMA is elaborated in section 4. Section 5 reports the experiment results on synthetic logs and real-life logs. Finally, we conclude our work and discuss future work in section 6.

2 Related Work

Anomaly detection approaches in business processes can be graph-agnostic or graph-based. A brief summary of the related work is given in Table 1.

2.1 Graph-agnostic Approaches

Some authors propose graph-agnostic approaches that simply treat traces as sequences. These approaches can be further divided into three categories, i.e., i) machine learning-based, ii) information theory-based, and iii) reconstruction-based approaches.

Machine Learning-based Approaches. These approaches detect anomalies using traditional machine learning anomaly detection algorithms. In [5], the normalized longest common sub-sequence (NLCS) between traces is calculated and k nearest neighbor (KNN) is applied to detect anomalies. The LOF algorithm based on k nearest neighbor imputation (KNNI) is used in [6] for the real-time business process monitoring to predict abnormal termination. Outlier-aware clustering algorithms are used in [7]. In [8], [9], activities and traces are considered as words and sentences, respectively, and traces are encoded as vectors using the word2vec [43] encoding method. After encoding, the random forest algorithm is applied in [8] to classify traces into normal and abnormal. Similarly, the authors in

TABLE 1: A brief summary of related works

Categories		Algorithms	Comments/Limitations
Graph-agnostic Approaches	Machine Learning-based	[5], [6], [7], [8], [9]	Requirs lots of calculations. Efficiency decreases with large data sizes.
	Information theory-based	[10], [11], [12], [13]	Applies statistical leverage to detect anomalies. Large detection granularity, can only detect trace-level anomalies.
	Reconstruction-based	[4], [14], [15], [16] [17], [18], [19], [20] [21], [34]	Has the ability to train incrementally when data distribution changes. Small detection granularity, can detect attribute-level anomalies.
Graph-based Approaches	Behavior Conformance-based	[23], [24], [25], [26] [35], [36], [37], [38] [39], [40], [41], [42]	A clean log or a prior knowledge of the process is required. The anomaly detection performance depends on the quality of the process model.
	Probabilistic Graphical Model-based	[27], [28], [29], [30] [31]	Converts logs to probabilistic graphical models considering the structural process information but creating models relies on clean logs.
	Graph Neural Network-based	[32]	Traces are encoded using graph neural networks that fully take into account structural process information.

[9] apply a one-class SVM to classify traces into normal and abnormal.

Information theory-based Approaches. In statistics, leverage is a widely used metric to detect anomalies. However, traditional leverage-based approaches cannot be applied to detect anomalies in business processes. In [11], [12], a new statistical leverage [44]-based approach is proposed to detect trace-level anomalies in business processes which also takes into account that traces may be of different lengths. The authors in [10], [13] apply statistical leverage to the online detection of anomalous traces in business process event streams.

Reconstruction-based Approaches. These methods first train a model that can reconstruct normal behaviors, and then detect anomalies based on the reconstruction error. An autoencoder is used in [14], [15], [16], [17], [18] to detect anomalies. GRASPED [18] introduces a well-designed teacher forcing method and the attention mechanism to improve its detection performance. Guan et al. [34] employ an autoencoder as a feature extractor and utilize a multilayer perceptron (MLP) as an anomaly score generator, effectively harnessing the potential of a limited number of labeled anomalies. Since a model obtained by an autoencoder frequently runs the risk of over-fitting, a denoising autoencoder is applied in [19], [20]. Inspired by the use of LSTM [45] in [46], [47], [48] to predict the next event in the event sequence, BINet is proposed in [4], [21], which is based on gated recurrent units (GRUs) [49] to predict the next event in the event sequence. BINet detects anomalies by predicting the probability of the attributes of the next event and if the probability is lower than the threshold, then the attribute is detected as an anomaly.

Graph-agnostic approaches can be applied directly to traces. However, a business process is a structured set of activities that have inherent relationships. These approaches do not take this information into account, which can limit the performance improvement of anomaly detection.

2.2 Graph-based Approaches

Graph-based approaches rely on a graph that models the relationships among activities to detect anomalies.

Behavior Conformance-based Approaches. A straightforward graph-based approach is to utilize conformance checking [22] to compare the differences between traces and legitimate behaviors of corresponding process models [35], [36], [37], [38], [39], [40], [41], [42], provided or discovered from the clean log using process discovery algorithms [50]. In addition to relying on process models alone, the authors in [23], [24], [25], [26] combine process models and association rule learning to detect and analyze anomalies in business processes to improve the accuracy of anomaly detection.

Probabilistic Graphical Model-based Approaches. A process model can be extended with probabilistic information, such as the likelihood of activity execution transitions. In [27], traces are mapped onto the likelihood graph. The probability of the attribute value is used to determine whether it is anomalous. In [28], Bayesian networks are automatically inferred from Petri nets, which allows the detection of non-obvious and interdependent temporal anomalies. In [29], the event logs are analyzed using the hidden Markov model (HMM). Moreover, in [30], three sequence analysis techniques based on windowing, the Markov model and the hidden Markov model are used to detect anomalies in business processes. Variable order Markov models (VOMMs) are used in [31] to predict the probability of the execution of each activity in the trace.

Graph Neural Network-based Approaches. With the successful applications of graph neural networks (GNNs), researchers have started to apply these techniques to solve the anomaly detection problem since business processes can be naturally modelled as a graph. By using GNNs, graph embedding can be obtained and anomalies can be detected in terms of reconstruction errors. For example, GAE uses an edge-conditioned convolution (ECC) to obtain a better graph encoding and then detects anomalies [32]. However, the current GNN-based approach often constructs a graph primarily based on control flow, neglecting the proper integration of attributes into the graph representation. As a consequence, it falls short in detecting attribute-level anomalies effectively. Moreover, the straightforward graph generation from individual traces further exacerbates

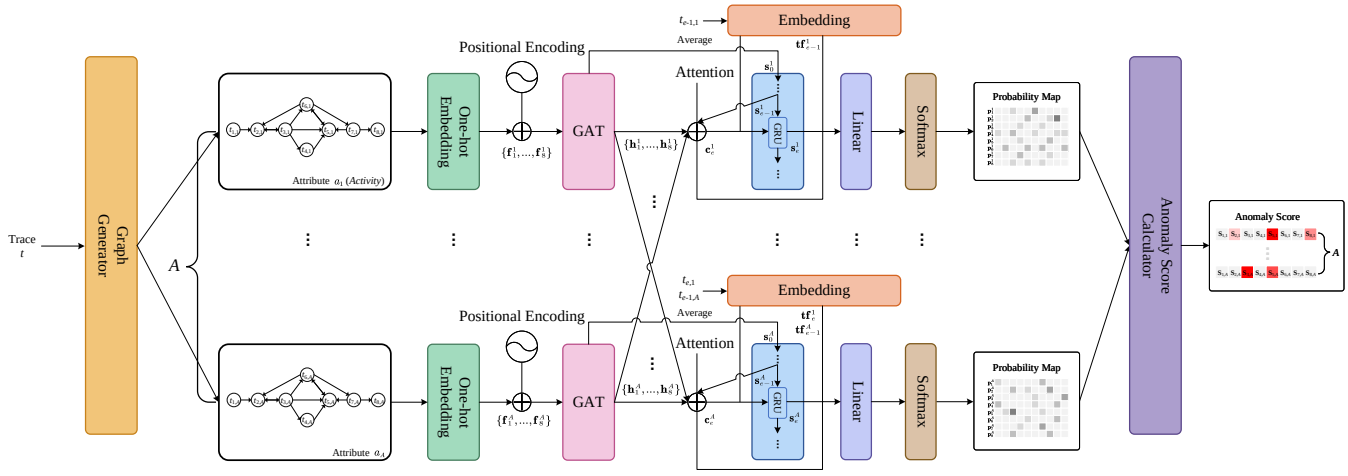


Fig. 2: The architecture of GAMA. (Using the trace involves eight events as an example).

the issue by lacking crucial structural information about the underlying process. These limitations hinder the comprehensive detection of anomalies.

GAMA is a notable example that addresses these limitations. Unlike methods that rely on a single graph to model control flows, GAMA surpasses them by constructing multiple graphs, one for each attribute, to ensure more comprehensive embeddings. A significant advantage of GAMA lies in its generation of a multi-graph derived from a global graph constructed using the entire event log. This global graph integration enables GAMA's multi-graph to accurately reflect the structural information of the process, leading to more precise and insightful analyses. Additionally, GAMA captures the intrinsic relationships between attributes through attention layers that operate across the graphs for each attribute.

3 Preliminaries

Following the widely accepted notations, we adopt bold uppercase characters (e.g., $\mathbf{A}$) to denote matrices, bold lowercase characters (e.g., $\mathbf{b}$) to indicate vectors and normal lowercase characters (e.g., c) as scalars.

In this section, we formalize the definition of log.

3.1 Log

As a foundation for the following sections, we first define the terms *log*, *trace*, *event*, and *attribute*.

Definition 3.1 (Log, Trace, Event). Let $\mathcal{A} = \{a_1, a_2, \cdots, a_A\}$ be a set of attributes, where $A = |\mathcal{A}|$ represents the number of attributes. $\mathcal{V}_a$ is the set of possible values for the attribute $a \in \mathcal{A}$. An *event* $e = [V_{a_1}, V_{a_2}, \cdots, V_{a_A}]$ is a tuple, with one value for each attribute, where $V_{a_i} \in \mathcal{V}_{a_i}$. A *trace* t is a sequence of events and an event *log* $\mathcal{L}$ is a set of traces. Note that $|t|$ is the number of events in trace t, $|\mathcal{L}|$ is the number of traces in log $\mathcal{L}$.

It should be noted that *activity* is a special attribute reflecting the control flow in attribute set $\mathcal{A}$. Let $t_{e,a}$ denote the value of attribute a of event e in trace t and e_a denote the value of attribute a of event e.

Example 1. Let attribute set $\mathcal{A} = \{Activity, Company\}$. $t = \langle[CSC, Th], [PSC, Be], [ASC, St], [CPO, Su], [RPO, Be], [PGR, Pl], [PIR, Fa], [PAY, Sl]\rangle$ represents a trace containing eight simple events taking place in chronological order. The value of the first attribute *Activity* of the second event e in trace t can be obtained with $t_{e,Activity} = PSC$ or $t_{2,1} = PSC$.

4 Methodology

In this section, we elaborate on the proposed framework GAMA.

4.1 Architecture of GAMA

The architecture of GAMA is shown in Fig. 2. The entire framework is based on an autoencoder-like unsupervised deep learning model, which consists of four essential components: i) *graph generator*: it converts a trace t into multiple graphs, where each attribute corresponds to one graph. For the sake of illustration, we default the first attribute a_1 to *Activity* (control flow perspective); ii) *multi-graph encoder*: it leverages the graph attention networks (GATs) [51] to embed the graph corresponding to each attribute, obtaining the hidden representation of each node. iii) *multi-sequence decoder*: it attempts to reconstruct the values for each attribute of each event in trace t; iv) *anomaly score calculator*: it calculates anomaly scores for each attribute of each event in trace t.

Inspired by the teacher forcing method [52], we provide the ground truth as input for the GRUs in the multi-sequence decoder. Three different teacher forcing styles are proposed to guide the reconstruction of attribute values.

Obviously, the combination of a multi-graph encoder and a multi-sequence decoder forms a straightforward autoencoder that effectively accomplishes the processes of compression and reconstruction. In other words, it compresses the input trace t into low-dimensional representations and subsequently generates the probability distribution ${}^{t}\mathbf{p}_e^a$ for each attribute a of every event e within trace t based on these condensed representations. Formally, the training procedure of the autoencoder can be characterized as minimizing the following reconstruction errors (i.e., cross-entropy loss [53]):

$$L(\mathcal{L}, \mathbf{P}) = -\sum_{t \in \mathcal{L}} \sum_{e \in t} \sum_{a \in \mathcal{A}} \log {}^{t}\mathbf{p}_e^a \left[t_{e,a} \right] \tag{1}$$

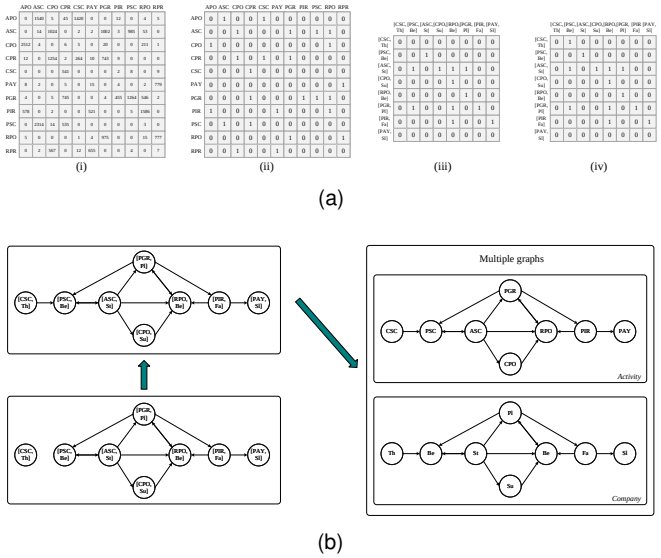


Fig. 3: Illustrations showing how multiple graphs are generated

where ${}^{t}\mathbf{p}_e^a[t_{e,a}]$ denotes the probability that the value of attribute a of event e in trace t is $t_{e,a}$.

Numerous previous studies [14], [15], [19], [54] have concluded that the magnitude of reconstruction errors is a powerful indicator of anomalies since anomalies do not follow the patterns of the majority and cannot be precisely reconstructed. Therefore, based on this indicator, anomaly scores can be computed.

For ease of reference, this section focuses solely on the reconstruction process of a trace. Therefore, we simplify the notation by omitting the trace identifiers, denoting ${}^{t}\mathbf{p}_e^a$ as $\mathbf{p}_e^a$.

4.2 Graph Generator

By utilizing existing process discovery techniques [55], [56], [57], logs can be converted to graph-based representations. However, it is non-trivial to convert a trace into a graph-based representation. Recognizing that anomalous patterns inherently exhibit distinct behavior across various attributes, we generate a graph for each attribute respectively.

We construct multiple directed graphs for trace t in log $\mathcal{L}$ to reflect the structural process information as follows. When generating the directed edges of the graphs, we only consider the control flow perspective. To begin, we compute the number of occurrences of each directly-follows relation (b, c), which is a measure employed by the α-algorithm [55]. This relation signifies the occurrence of activity b being directly followed by activity c within the log $\mathcal{L}$. Secondly, a directed global graph $G(\mathcal{L})$ is generated, containing the activities of log $\mathcal{L}$ as nodes. An edge $b \rightarrow c$ is present in directed global graph $G(\mathcal{L})$ if and only if the number of occurrences of directly-follows relation (b, c) is no less than $\beta * |\mathcal{L}|$, where β is a user-chosen threshold to filter out infrequent relations (i.e., noise). Thirdly, a directed event graph $G(t)$ is generated, containing the events of trace t as nodes. An edge $e \rightarrow e'$ is present in directed event graph $G(t)$ if $e_{Activity} \rightarrow e'_{Activity}$ is an edge in directed global graph $G(\mathcal{L})$. Fourthly, to comprehensively capture the directly-follows relations between events, an edge $e \rightarrow e'$ is present in $G(t)$ if event e directly follows event e' in trace t. Finally, a distinct graph can be derived for each attribute in $\mathcal{A}$ from the directed event graph $G(t)$. We can obtain multiple graphs by replacing the nodes in the event graph $G(t)$ from event e to attribute value e_a for every attribute $a \in \mathcal{A}$.

Example 2 (Example 1 continued). Consider attribute set $\mathcal{A}$ and trace t in Example 1. We assume that the number of occurrences of each directly-follows relation in log $\mathcal{L}$ is shown in Fig. 3a(i). Considering $\beta * |\mathcal{L}| = 50$, the adjacency matrix of the directed global graph $G(\mathcal{L})$ is shown in Fig. 3a(ii). Fig. 3a(iii) and Fig. 3a(iv) present the adjacency matrix of directed event graph $G(t)$ after step three and step four respectively. Fig. 3b illustrates the evolution of directed event graph $G(t)$. The multiple graphs of trace t are shown on the right in Fig. 3b.

4.3 Multi-graph Encoder

Given A graphs generated by trace t, where each graph contains $|t|$ nodes, we assign a distinct one-hot embedding, positional encoding and GAT to each graph. Next, we introduce these three components in detail, using attribute a as an example.

Due to the neural networks' inability to interpret language, we must transform the language to numbers. One-hot embedding [58], which transforms categorical variables into binary vectors, is one technique to accomplish this. After one-hot embedding, the value of nodes can be converted to $|\mathcal{V}_a|$-dimensional vectors, where $|\mathcal{V}_a|$ is number of possible values for the attribute a.

In our architecture, we use GATs to encode the nodes (i.e., the attribute values) to obtain hidden representations of the nodes. Compared to RNNs, which is commonly applied to solve sequence problems, GATs can better aggregate the information of events that have strong correlations in the trace. For example, assuming that the first event and the last event in a trace are strongly correlated (i.e., in the log, the activity of the first event is often directly followed by the activity of the last event), in the graph, there exists a directed edge from the first event to the last event, so that the last event is able to aggregate the information of the first event. However, from a sequence perspective, they are far apart, and RNNs will experience the vanishing gradient problem [59], and the last event cannot aggregate the information of the first event well.

The order of events is very important, so if the order of events in a trace changes, the trace will become an anomalous trace. However, GATs do not consider the order of nodes, losing the positional information, which is vital to reconstruct a complete trace. In order for the GATs to utilize the order of events, some information about the relative or absolute position of the events in the trace must be injected into one-hot embeddings. In this work, we use the positional encoding (PE) proposed in [60]:

$$\mathbf{f}^a_{pos,2i} \mathrel{+}= \sin\left(pos/10000^{2i/|\mathcal{V}_a|}\right) \tag{2}$$

$$\mathbf{f}^a_{pos,2i+1} \mathrel{+}= \cos\left(pos/10000^{2i/|\mathcal{V}_a|}\right) \tag{3}$$

where pos is the position of the event in the trace and i-th dimension of the one-hot embedding $\mathbf{f}^a_{pos}$. After positional encoding, a set of node features $\{\mathbf{f}^a_1, \mathbf{f}^a_2, \cdots, \mathbf{f}^a_{|t|}\}$ are input into the GAT and GAT outputs a set of hidden representations of nodes $\{\mathbf{h}^a_1, \mathbf{h}^a_2, \cdots, \mathbf{h}^a_{|t|}\}$. We average these vectors:

$$\mathbf{s}^a_0 = \text{mean}(\{\mathbf{h}^a_1, \mathbf{h}^a_2, \cdots, \mathbf{h}^a_{|t|}\}) \tag{4}$$

Overall, multi-graph encoder outputs $|t| * A$ hidden representations $\{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{|t|*A}\}$ and A initial hidden states $\{\mathbf{s}^1_0, \mathbf{s}^2_0, \cdots, \mathbf{s}^A_0\}$ for GRUs in multi-sequence decoder, where $|t|$ is the length of trace t and A is the number of attributes.

4.4 Multi-sequence Decoder

In ths subsection, the notation $\mathbf{s}_e = \text{GRU}(\mathbf{s}_{e-1}, \mathbf{x}_e)$ represents the process by which the input $\mathbf{x}_e$ at the current time step e and the hidden state $\mathbf{s}_{e-1}$ from the previous time step $e-1$ are fed into a GRU. This GRU operation generates the updated hidden state $\mathbf{s}_e$ at the current time step e.

The multi-sequence decoder, in contrast to the multi-graph encoder, decodes the hidden representations into a probability map. The higher the probability of the attribute value, the more likely it is to be normal.

First, the output of the multi-graph encoder $\{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{|t|*A}\}$ needs to go through the attention layer and generates $\mathbf{c}^a_e$. The attention mechanism serves as a vital link connecting the encoder and the decoder. The attention mechanism identifies which attributes of which events are relevant to the next target attribute value and gives high attention weights to those encoded attribute values.

$$eg^a_{ei} = \frac{(\mathbf{W}^a_q \mathbf{s}^a_{e-1})^T \mathbf{W}^a_k \mathbf{h}_i}{\sqrt{d}} \tag{5}$$

$$\alpha^a_{ei} = \text{Softmax}_i\left(eg^a_{ei}\right) = \frac{\exp(eg^a_{ei})}{\sum_{j=1}^{|t|*A} \exp(eg^a_{ej})} \tag{6}$$

where eg^a_{ei} represents the energy state, $\mathbf{W}^a_q$ and $\mathbf{W}^a_k$, which convert $\mathbf{s}^a_{e-1}$ and $\mathbf{h}_i$ into d-dimensional vectors (i.e., $\mathbf{W}^a_q \mathbf{s}^a_{e-1}$ and $\mathbf{W}^a_k \mathbf{h}^a_i$), are the learnable matrices. In Eq. (6), the energy states eg^a_{ei} computed at event e are normalized using softmax to obtain the corresponding attention weight α^a_{ei}, which intuitively reflects the significance of each encoded attribute value $\mathbf{h}_i$ during reconstruction. A higher value of α^a_{ei} indicates that $\mathbf{h}_i$ is more important for predicting the current attribute value.

Then, $\mathbf{c}^a_e$ can be obtained, which involves weighting the output of the multi-graph encoder with their respective attention weights in a direct manner.

$$\mathbf{c}^a_e = \sum_{i=1}^{|t|*A} \alpha^a_{ei} \mathbf{h}_i \tag{7}$$

Next, in order to better reconstruct the target attribute value at event e, we introduce the teacher forcing method which is widely used in the field of natural language processing.

In the case of the first attribute (*Activity*), the prediction of the probability distribution for the attribute value at the current event e is guided solely by the previous ground truth attribute value (i.e., activity name) $t_{e-1,1}$.

$$\mathbf{s}^1_e = \text{GRU}(\mathbf{s}^1_{e-1}, [\mathbf{c}^1_e \| \mathbf{tf}^1_{e-1}])) \tag{8}$$

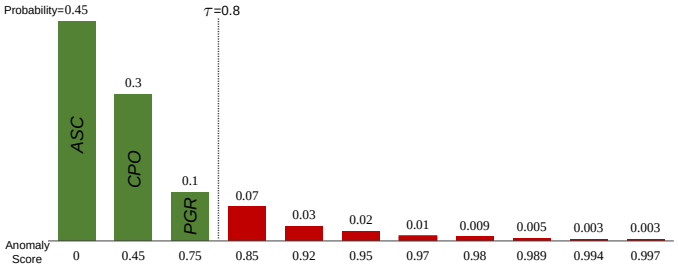


Fig. 4: Probability distribution

where $\mathbf{tf}^1_{e-1}$ is the embedding vector of $t_{e-1,1}$. $[\mathbf{c}^1_e \| \mathbf{tf}^1_{e-1}]$, which is the concatenation of $\mathbf{tf}^1_{e-1}$ and $\mathbf{c}^1_e$, is inputted into the GRU (see the blue part in Fig. 2). The initial hidden state of the GRU is the output $\mathbf{s}^1_0$ of the multi-graph encoder.

Finally, the probability distribution $\mathbf{p}^1_e$, which is the probability distribution over all possible values of the first attribute (*Activity*) at event e, can be calculated by

$$\mathbf{p}^1_e = \text{Softmax}(\mathbf{W}^1_p[\mathbf{s}^1_e \| \mathbf{c}^1_e \| \mathbf{tf}^1_{e-1}]) \tag{9}$$

which represents the linear layer and the softmax layer.

In the case of the other attribute a, three different teacher forcing styles are proposed to guide the reconstruction of attribute values.

i) Activity name (AN): We consider that the current attribute value depends mainly on the current activity name. Therefore, at current event e, the ground truth activity name $t_{e,1}$ is used to guide the prediction of the probability distribution $\mathbf{p}^a_e$ which can be calculated by

$$\mathbf{s}^a_e = \text{GRU}(\mathbf{s}^a_{e-1}, [\mathbf{c}^a_e \| \mathbf{tf}^1_e]) \tag{10}$$

$$\mathbf{p}^a_e = \text{Softmax}(\mathbf{W}^a_p[\mathbf{s}^a_e \| \mathbf{c}^a_e \| \mathbf{tf}^1_e]) \tag{11}$$

where $\mathbf{tf}^1_e$ is the embedding vector of $t_{e,1}$.

ii) Previous attribute value (PAV): We consider that the current attribute value depends mainly on the previous attribute value. Therefore, the previous ground truth attribute value $t_{e-1,a}$ is used to guide the prediction of the probability distribution $\mathbf{p}^a_e$ which can be calculated by

$$\mathbf{s}^a_e = \text{GRU}(\mathbf{s}^a_{e-1}, [\mathbf{c}^a_e \| \mathbf{tf}^a_{e-1}]) \tag{12}$$

$$\mathbf{p}^a_e = \text{Softmax}(\mathbf{W}^a_p[\mathbf{s}^a_e \| \mathbf{c}^a_e \| \mathbf{tf}^a_{e-1}]) \tag{13}$$

where $\mathbf{tf}^a_{e-1}$ is the embedding vector of $t_{e-1,a}$.

iii) Fusion of activity name and previous attribute value (FAP): We consider that the current attribute value depends both on the current activity name and the previous attribute value. Therefore, the fusion of current ground truth activity name $t_{e,1}$ and the previous ground truth attribute value $t_{e-1,a}$ is used to guide the prediction of the probability distribution $\mathbf{p}^a_e$ which can be calculated by

$$\mathbf{s}^a_e = \text{GRU}(\mathbf{s}^a_{e-1}, [\mathbf{c}^a_e \| \mathbf{tf}^1_e \| \mathbf{tf}^a_{e-1}]) \tag{14}$$

$$\mathbf{p}^a_e = \text{Softmax}(\mathbf{W}^a_p[\mathbf{s}^a_e \| \mathbf{c}^a_e \| \mathbf{tf}^1_e \| \mathbf{tf}^a_{e-1}]) \tag{15}$$

4.5 Anomaly Score Calculator

The trained model can be applied to identify anomalies after the training phase. We input trace t into the trained model to obtain the probability distribution $\mathbf{p}^a_e$ over all possible values of attribute a of event e.

Typically, compared to a normal attribute value, the probability of an anomalous attribute value is lower. Based on this idea, the anomaly score for the value of attribute a of event e in trace t is defined as the sum of all probabilities in the probability distribution $\mathbf{p}_e^a$ greater than the probability of $t_{e,a}$ (i.e., $\mathbf{p}_e^a[t_{e,a}]$). To calculate the anomaly score for attribute a of event e, the following formula can be applied:

$$\mathbf{S}_{t,e,a} = \sum_{\mathbf{p}_{e,i}^a > \mathbf{p}_e^a[t_{e,a}]} \mathbf{p}_{e,i}^a \tag{16}$$

where $\mathbf{p}_{e,i}^a$ is i-th probability in probability distribution $\mathbf{p}_e^a$.

Example 3. Consider probability distribution $\mathbf{p}_e^a = [0.07, 0.3, 0.005, 0.03, 0.01, 0.003, 0.02, 0.45, 0.003, 0.1, 0.003]$ of trace t in Fig. 4. There are eleven possible attribute values for attribute a. Anomaly scores for each possible value are calculated. Assuming that $t_{e,a}$ = *PGR*, we can calculate $\mathbf{S}_{t,e,a} = 0.45 + 0.3 = 0.75$.

4.6 Anomaly Detection

By applying a threshold τ, the anomaly scores are labeled into 0 or 1, with 0 indicating normal and 1 indicating anomalous. We label an attribute value as anomalous whenever its anomaly score exceeds τ (i.e., attribute-level). The likelihood of an attribute value being anomalous increases with its anomaly score.

$$\mathbf{L}_{t,e,a} = \begin{cases} 1 & \text{if } \mathbf{S}_{t,e,a} > \tau \\ 0 & \text{if } \mathbf{S}_{t,e,a} \leqslant \tau \end{cases} \tag{17}$$

Example 4 (Example 3 continued). Given that threshold τ is 0.8, since the anomaly score $\mathbf{S}_{t,e,a} = 0.75$ which is less than 0.8, we can infer that the corresponding label for attribute a of event e in trace t is $\mathbf{L}_{t,e,a} = 0$, indicating normal.

The following rule can be utilized to adapt our method to trace-level anomaly detection: If there exists some attribute of some event in a trace that is anomalous, the trace is anomalous. The following formal representation is offered:

$$\mathbf{L}_t^{trace} = \begin{cases} 1 & \text{if } \exists\ \mathbf{L}_{t,e,a} = 1 \text{ for } e \in t, a \in \mathcal{A} \\ 0 & \text{if } \forall\ \mathbf{L}_{t,e,a} = 0 \text{ for } e \in t, a \in \mathcal{A} \end{cases} \tag{18}$$

5 Experiments

In this section, we empirically evaluate the effectiveness of GAMA on both synthetic and real-life datasets. GAMA is implemented in Python, and the source code is accessible at https://github.com/guanwei49/GAMA.

All experiments are conducted in an unsupervised manner, meaning that no prior knowledge of the process is provided, and clean event logs are unavailable. The models are trained on event logs containing anomalies, and subsequently, anomaly detection is performed on the same event logs.

5.1 Compared Methods

To verify the superiority of the proposed method, we compare GAMA with the following state-of-the-art methods:

- *OC-SVM* [61]: It transforms traces into vector representations and applies one-class SVM [62] to find anomalies.
- *Naive* [41]: It marks all traces that are infrequent in the event log as anomalies.
- *Sampling* [41]: It selects a sample of the event log and mines the process model to detect anomalies by comparing the differences between the traces and the mined process model.
- *GAE* [32]: It transforms traces into graphs and detects anomalies based on the reconstruction errors of the edges between nodes.
- *DAE* [19]: It transforms traces into vector representations and detects anomalies based on reconstruction errors in the denoising autoencoder.
- *VAE* [14]: It transforms traces into vector representations and detects anomalies based on reconstruction errors in the variational autoencoder.
- *LAE* [14]: It transforms traces into vector representations and detects anomalies based on reconstruction errors in the LSTM-based autoencoder.
- *BINet* [4]: It detects anomalies by predicting the attribute values of the next event.

DAE, VAE, LAE and BINet support attribute-level anomaly detection and trace-level anomaly detection. But OC-SVM, Naive and Sampling only support trace-level anomaly detection.

5.2 Parameter Settings and Metrics

GAMA is implemented based on PyTorch [63]. In our experiment, we apply two-layer GATs for encoding and two-layer GRUs for decoding. The first layer of GATs has four heads and the second layer of GATs has one head. Dropout [64] is applied after each layer of GATs and GRUs to counteract overfitting. We initialize weights using the initialization introduced in [65], and train the proposed model for a maximum of 20 epochs with the Adam optimizer [66]. In the absence of special statements, the hidden layer size of the GATs and GRUs is 64, the learning rate is 0.0002, and β, which is used to control the complexity of the graph, is 0.01.

For a fair comparison, we adopt the widely used metric $F-score$, which is the harmonic mean of $Precision$ and $Recall$, to evaluate the performance of these methods. These metrics are defined as follows: $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and $F-score = \frac{2*Precision*Recall}{Precision+Recall}$, where TP, FN and FP represent true positives, false negatives and false positives respectively. To be fair, we use a grid search method to determine the optimal threshold value τ and use this threshold value to calculate the $F-score$ for each method. The method has excellent anomaly detection performance when the $F-score$ is close to 1.

5.3 Artificial Anomalies

As in previous studies [4], [12], [19], [27], [41], we inject artificial anomalies into event logs to verify the effectiveness

TABLE 2: $F-score$ over synthetic logs where 'T' and 'A' represent trace- and attribute-level anomaly detection respectively

Methods \ Logs	Paper		P2P		Small		Medium		Large		Huge		Gigantic		Wide	
	T	A	T	A	T	A	T	A	T	A	T	A	T	A	T	A
OC-SVM	0.498	-	0.480	-	0.522	-	0.446	-	0.480	-	0.446	-	0.462	-	0.460	-
Naive	0.866	-	0.850	-	0.898	-	0.691	-	0.715	-	0.690	-	0.574	-	0.779	-
Sampling	0.901	-	0.886	-	0.896	-	0.860	-	0.910	-	0.890	-	0.800	-	0.888	-
GAE	0.472	-	0.559	-	0.468	-	0.449	-	0.530	-	0.429	-	0.434	-	0.561	-
DAE	0.799	0.468	0.767	0.475	0.829	0.463	0.713	0.436	0.747	0.433	0.691	0.415	0.580	0.288	0.753	0.455
VAE	0.828	0.190	0.655	0.212	0.788	0.219	0.637	0.230	0.772	0.201	0.589	0.213	0.495	0.181	0.640	0.230
LAE	0.678	0.243	0.666	0.266	0.748	0.239	0.584	0.270	0.571	0.250	0.531	0.268	0.504	0.234	0.699	0.271
BINet	0.543	0.330	0.557	0.342	0.566	0.358	0.521	0.319	0.549	0.333	0.526	0.331	0.525	0.320	0.551	0.345
GAMA-AN	**0.949**	**0.701**	**0.950**	**0.686**	**0.962**	**0.724**	**0.873**	**0.716**	**0.945**	**0.768**	**0.916**	**0.763**	**0.821**	**0.701**	**0.921**	**0.724**
GAMA-PAV	**0.976**	**0.675**	**0.974**	**0.664**	**0.997**	**0.662**	**0.903**	**0.654**	**0.944**	**0.678**	**0.909**	**0.663**	**0.809**	**0.614**	**0.950**	**0.670**
GAMA-FAP	**0.955**	**0.699**	**0.949**	**0.683**	**0.957**	**0.712**	**0.872**	**0.700**	**0.947**	**0.752**	**0.922**	**0.750**	**0.833**	**0.691**	**0.923**	**0.712**

Fig. 5: Different anomaly types applied to a normal trace

of GAMA. Six anomaly types in [4] which frequently arise in real business processes are manually injected. Fig. 5 shows the different anomalous traces obtained by applying six anomaly types to a normal trace. These anomaly types are defined as follows:

- *Skip*: A sequence of events has been skipped.
- *Insert*: A sequence of random events has been inserted.
- *Rework*: A sequence of events has been executed a second time.
- *Early*: A sequence of events has been executed too early, and hence is skipped later.
- *Late*: A sequence of events has been executed too late, and hence is skipped earlier.
- *Attribute*: Some attribute value has been replaced by an incorrect value in some events.

5.4 Dataset

Both synthetic logs and real-life logs are applied to evaluate our method.

For synthetic logs, we adopt eight different business process models (*Paper*, *P2P*, *Small*, *Medium*, *Large*, *Huge*, *Gigantic*, *Wide*) [4] to generate synthetic logs, using simulation technology. *Paper* and *P2P* are handmade process models and the others are random process models generated by the PLG2 tool [67] with a different number of activities and varying size ranges. We also extend a likelihood graph [27] to generate causally dependent event attributes. For each process model, the number of attributes in the synthetic logs varies from 2 to 5, producing $4 * 8 = 32$ synthetic logs free of anomalies.

For real-life logs, we consider six widely used event logs: i) *Billing*: it contains events that are related to the billing of medical services that have been provided by a hospital. ii) *Receipt*: it contains the records of the execution of the receiving phase of the building permit application process in an anonymous municipality. iii) *Sepsis*: it contains events of sepsis cases from a hospital. iv) *RTFMP*: it contains the execution of the road traffic fine management process. v) *Permit*: it contains events related to travel permits (including all related events of relevant prepaid travel cost declarations and travel declarations). vi) *Declaration*: it contains events related to international travel declarations.

We inject AP percent of artificial anomalies (i.e., AP percent of the traces are anomalous) into both synthetic and real-life logs. In our experiments, we evaluate the scalability of our method by considering different values of AP, specifically 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, and 0.45. In the end, we obtain $9 * 32 = 288$ synthetic logs and $9 * 6 = 54$ real-life logs.

5.5 Experiments on Synthetic Logs

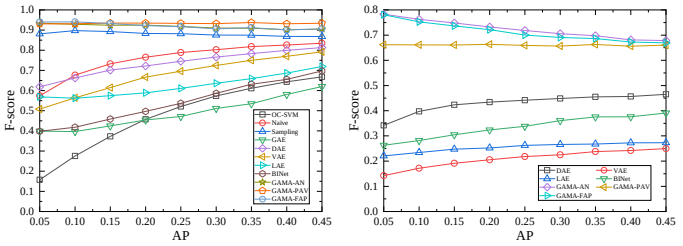
5.5.1 Performance Evaluation

Table 2 reports the anomaly detection performance for trace-level anomaly detection and attribute-level anomaly detection of various approaches on synthetic logs. We make the following observations.

In terms of trace-level anomaly detection, the proposed GAMA framework (-AN, -PAV and -FAP) achieves the best performance on all synthetic logs, which demonstrates the superiority of graph attention networks for the business process anomaly task. Teacher forcing style AN performs relatively poorly compared to PAV and FAP. OC-SVM has the lowest $F-scores$ on most of the synthetic logs. As expected, OC-SVM is not specifically designed for business process anomaly detection. Although DAE, VAE and LAE are based on the autoencoder, which is similar to our approach, they do not consider structural process information, resulting in unsatisfactory detection performance. Although GAE takes advantage of GNNs, the graphs transformed

TABLE 3: $F-score$ over real-life logs where 'T' and 'A' represent trace- and attribute-level anomaly detection respectively

Methods \ Logs	Billing		Receipt		Sepsis		RTFMP		Permit		Declaration	
	T	A	T	A	T	A	T	A	T	A	T	A
OC-SVM	0.340	-	0.464	-	0.415	-	0.507	-	0.405	-	0.449	-
Naive	0.668	-	0.638	-	0.392	-	0.776	-	0.462	-	0.495	-
Sampling	0.701	-	0.647	-	0.391	-	0.721	-	0.458	-	0.507	-
GAE	0.385	-	0.420	-	0.391	-	0.341	-	0.386	-	0.406	-
DAE	0.754	0.444	0.650	0.158	0.461	0.136	0.865	0.498	0.522	0.182	0.576	0.201
VAE	0.731	0.435	0.524	0.134	0.448	0.172	0.813	0.517	0.484	0.188	0.476	0.180
LAE	0.784	0.509	0.526	0.218	0.408	0.126	0.874	0.505	0.486	0.287	0.514	0.345
BINet	0.621	0.442	0.575	0.416	0.435	0.192	0.744	0.493	0.641	0.423	0.678	**0.499**
GAMA-AN	**0.792**	**0.545**	**0.763**	**0.548**	**0.570**	**0.457**	**0.899**	**0.534**	**0.682**	**0.428**	**0.727**	0.461
GAMA-PAV	**0.791**	**0.544**	**0.778**	**0.538**	**0.510**	**0.376**	**0.914**	**0.574**	0.634	0.369	0.669	0.406
GAMA-FAP	**0.811**	**0.548**	**0.752**	**0.535**	**0.573**	**0.450**	**0.914**	**0.576**	**0.679**	0.402	**0.718**	0.444



(a) Trace-level anomaly detection (b) Attribute-level anomaly detection

Fig. 6: $F-score$ under different AP over synthetic logs

from traces do not contain the information about the process underlying the event log. Sampling takes into account the structural process information, therefore, Sampling performs relatively well.

In terms of attribute-level anomaly detection, the proposed GAMA framework (-AN, -PAV and -FAP) also achieves the best performance on all the synthetic logs, which demonstrates the superiority of GAMA. GAMA with the teacher forcing style AN achieves the best performance on all the synthetic logs, which demonstrates that the attribute value of an event greatly depends on the activity name of the previous event in synthetic logs. VAE performs the worst, the $F-scores$ of which are always lower than 0.25.

5.5.2 Impact of Anomaly Percentage

Next, we evaluate the impact of anomaly percentage. $F-score$ is averaged over all synthetic logs.

In terms of trace-level anomaly detection, from Fig. 6a, we can see that GAMA-PAV has the best performance. Its $F-score$ hovers around 0.95 and barely varies with anomaly percentage. $F-scores$ of Sampling, GAMA-AN and GAMA-FAP decrease as anomaly percentage increases, which indicates that when anomaly percentage is too high, there will be some failure of the teacher forcing style AN and PAV. Furthermore, as anomaly percentage increases, the accuracy of the process model discovered by Sampling degrades, so the detection performance becomes worse. On the contrary, the $F-scores$ of OC-SVM, Naive, GAE, DAE, VAE, LAE and BINet increase as anomaly percentage increases. As expected, when detection is weak, an increase in anomaly percentage must lead to an increase in $Precision$ (i.e., randomly select anomalous traces, $Precision$ is approximately equal to AP), and thus $F-score$ also increases.

In terms of attribute-level anomaly detection, from Fig. 6b, we can see that GAMA outperforms the other methods at any anomaly percentage. Similar to the anomaly detection results at trace level, $F-score$ of GAMA-PAV barely varies with anomaly percentage and $F-scores$ of GAMA-AN and GAMA-FAP decrease as anomaly percentage increases, which suggests that the teacher forcing methods AN and FAP are more applicable to datasets with a low anomaly percentage.

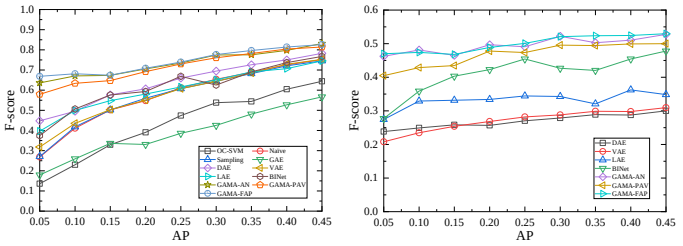
5.6 Experiments on Real-life Logs

5.6.1 Performance Evaluation

Table 3 reports the anomaly detection performance of different approaches for trace-level anomaly detection and attribute-level anomaly detection on real-life logs. We make the following observations.

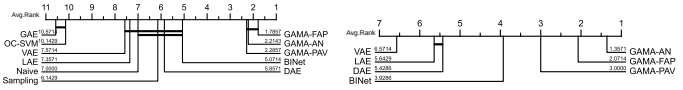
In terms of trace-level anomaly detection, we can see that the proposed GAMA framework (-AN, -PAV and -FAP) achieves the best performance on the Billing, Receipt, Sepsis and RTFMP datasets, which demonstrates the superiority of GAMA in the anomaly detection task. GAMA-AN and GAMA-FAP achieve the best performance on all real-life datasets. But due to the particularity of the Permit and Declaration datasets, BINet seems to be more effective than GAMA-PAV, although GAMA-PAV exhibits comparable results. Specifically, the activity name and attribute value of the current event in Permit and Declaration depend primarily on the previous activity name and have no long dependencies, which are carefully considered and treated by BINet. However, GATs used in GAMA are more suitable for capturing long dependencies, but for the teacher forcing method PAV, it does not input the activity name of the previous event into the network as AN and FAP do, so it does not perform better than BINet. Similar to the results shown in Table 2, OC-SVM and GAE perform the worst.

In terms of attribute-level anomaly detection, the GAMA framework (-AN, -PAV and -FAP) also achieves the best performance on the Billing, Receipt, Sepsis and RTFMP datasets. On Permit, GAMA-AN performs the best, but GAMA-PAV and GAMA-FAP do not perform better than BINet. On Declaration, BINet performs better than GAMA.



(a) Trace-level anomaly detection (b) Attribute-level anomaly detection

Fig. 7: $F-score$ under different AP over real-life logs



(a) Trace-level anomaly detection (b) Attribute-level anomaly detection

Fig. 8: Critical difference diagram over all logs.

These also demonstrate that the activity name and attribute value of the current event in Permit and Declaration depend primarily on the previous activity name and have no long dependencies, which are carefully considered and treated by BINet.

5.6.2 Impact of Anomaly Percentage

We then evaluate the impact of anomaly percentage. $F-score$ is averaged over all real-life logs.

In terms of trace-level anomaly detection, from Fig. 7a, we can see that the proposed GAMA framework (-AN, -PAV and -FAP) has the best performance, as its $F-score$ is higher than the other methods at any anomaly percentage, which further suggests that GAMA is more scalable. Furthermore, the $F-score$s of all the methods increase as the anomaly percentage increases. There could be two major reasons for this: i) the real-life dataset itself may contain some natural anomalies that are detected but not labeled. As the percentage of injected anomalies increases, these originally anomalous traces are labeled to improve the anomaly detection performance; ii) an increase in anomaly percentage must result in an increase in $Precision$ (i.e., randomly select anomalous traces, $Precision$ is roughly equal to AP), and thus $F-score$ also increases. Although Sampling performs relatively well on synthetic logs (see Fig. 6a), it performs poorly on real-life logs. This illustrates the difficulty of mining real-life process models using process discovery algorithms.

In terms of attribute-level anomaly detection, from Fig. 7b, it can be seen that the $F-score$ of GAMA is significantly larger than the other methods under any anomaly percentage.

5.7 Critical difference diagram

Fig. 8a and Fig. 8b show critical difference (CD) diagrams [68] of trace-level and attribute-level anomaly detection respectively to visualize the results with a confidence interval of 95 percent. A bold horizontal line is used to group a set of methods that do not exhibit significant differences.

In terms of trace-level anomaly detection, based on the critical difference, we recognize that GAMA, which takes full account of the structural process information within and between different attributes, performs significantly better than other methods and there was no significant difference in performance between the three teacher forcing styles. GAE, a method that also utilizes GNNs for anomaly detection, demonstrates significantly poorer performance. This discrepancy arises from the fact that GAE generates a graph directly from individual traces, resulting in a lack of structural information about the process. In contrast, GAMA derives a multi-graph for each trace from a global graph constructed using the entire event log. This global graph contains comprehensive structural information about the process, enabling GAMA to outperform other methods.

In terms of attribute-level anomaly detection, GAMA performs significantly better than other methods and teacher forcing style AN is significantly better than PAV and FAP. VAE performs significantly worse than other methods, indicating that the hidden representations of traces do not follow the normal distribution.

TABLE 4: Effectiveness of three teacher forcing styles

	Synthetic Logs		Real-life Logs	
	Trace-level	Attribute-level	Trace-level	Attribute-level
AN	0.917	**0.723**	0.733	**0.496**
PAV	**0.933**	0.660	0.716	0.468
FAP	**0.919**	**0.712**	**0.743**	**0.493**
-	0.858	0.594	0.641	0.429

5.8 Effectiveness of Three Teacher Forcing Styles

We evaluate the effectiveness of three teacher forcing styles. The $F-score$s are presented in Table 4 and the best two results are shown in bold typeface and the best results are underlined. '-' implies that the teacher forcing method is not introduced.

Compared to not introducing the teacher forcing method, incorporating any of the teacher forcing styles significantly enhances the detection performance of GAMA. This finding serves as strong evidence for the effectiveness of the teacher forcing styles specifically designed for business processes. In terms of attribute-level anomaly detection, teacher forcing style AN always has the best performance, which indicates that the current attribute value depends mainly on the current activity name. As expected, the performance of the teacher forcing style FAP is usually between AN and PAV, and is not far from the best results, which we consider to be the more worthwhile teacher forcing style.

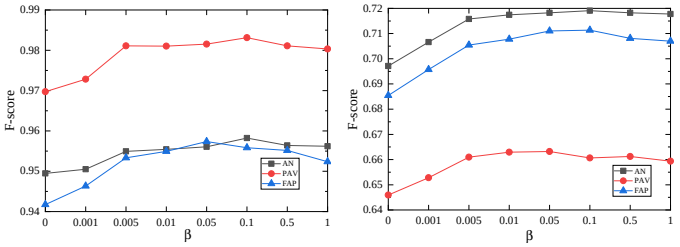
5.9 Ablation Study on Positional Encoding

It is well known that GATs do not consider the order of nodes, losing positional information, which is vital to reconstruct a complete trace. We verify whether the introduction of positional encoding can provide useful information to improve the detection performance of GAMA.

From Table 5, we can see that the introduction of positional coding (PE) improves the detection performance of GAMA (-AN, -PAV and -FAP) both for trace- and attribute-level anomaly detection, regardless of whether it is on

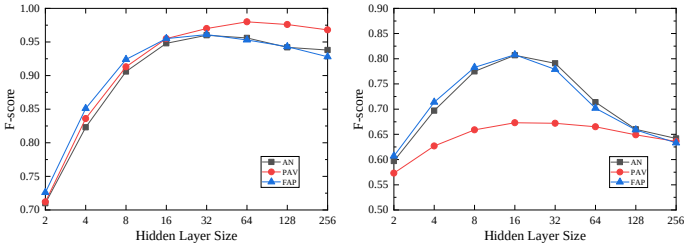
TABLE 5: Ablation study on positional encoding

	Synthetic Logs						Real-life Logs					
	Trace-level			Attribute-level			Trace-level			Attribute-level		
	AN	PAV	FAP	AN	PAV	FAP	AN	PAV	FAP	AN	PAV	FAP
With PE	**0.917**	**0.933**	**0.919**	**0.723**	**0.660**	**0.712**	**0.733**	**0.716**	**0.743**	**0.496**	**0.468**	**0.493**
Without PE	0.896	0.917	0.894	0.667	0.643	0.663	0.674	0.638	0.700	0.453	0.423	0.462



(a) Trace-level anomaly detection (b) Attribute-level anomaly detection

Fig. 9: $F-score$ under different β over *Small* logs



(a) Trace-level anomaly detection (b) Attribute-level anomaly detection

Fig. 10: $F-score$ under different different hidden layer sizes over *Small* logs

synthetic or real-life datasets. This suggests that GATs are indeed position-insensitive and the introduction of positional coding significantly enhances the ability of GATs to encode graphs transformed from sequence-like event traces.

5.10 Parameter Analysis

5.10.1 Impact of β

Note that parameter β controls the number of directed edges in the generated multiple graphs. In this section, we investigate the impact of β on the anomaly detection performance by varying the value of β. $F-score$ is averaged over all the *Small* logs.

From Fig. 9a and Fig. 9b, we can see that the $F-score$s of GAMA (-AN, -PAV and -FAP) increase and then decrease as parameter β increases both for trace- and attribute-level anomaly detection. As expected, when β is small, some directed edges between nodes without any relationship in the generated graph are retained, and these directed edges have a substantial impact on GATs' ability to encode the nodes. As β increases, some meaningless directed edges are removed, but the directed edges between nodes that have relationships are likewise incorrectly removed, resulting in GATs' inability to aggregate information about some useful nodes. Although the performance of GAMA varies with β, these variations are not significant and do not exceed 0.025, which means our proposed GAMA is not overly sensitive to β.

5.10.2 Impact of Hidden Layer Size

Next, we explore the impact of hidden layer size on detection performance. $F-score$ is averaged over all the *Small* logs.

From Fig. 10a and Fig. 10b, we can see that the $F-score$s of GAMA (-AN, -PAV and -FAP) increase and then decrease as the hidden layer size increases both for trace- and attribute-level anomaly detection. As expected, the capacity of the model is limited by the hidden layer size. The hidden layer size is set too small, resulting in too small a model capacity, and the model is unable to learn useful information. The hidden layer size is set too large, resulting in too large a model capacity, and the trained model is able to reconstruct frequent (normal) patterns well while also being able to reconstruct infrequent (anomalous) patterns.

6 Conclusions and Future Work

In this paper, we propose GAMA, a multi-graph-based anomaly detection framework for business processes via graph neural networks. Our approach comprehensively incorporates the structural process information by transforming a trace into a multi-graph with the assistance of a global graph. We utilize GNNs to effectively learn the embedding of this multi-graph. The intrinsic relationships between different attributes are captured by aggregating multiple graphs using the attention mechanism. GAMA is trained in an unsupervised manner (i.e. no data labels are required) and independent of any prior knowledge of the process, which makes it easier to employ. Inspired by the teacher forcing method in natural language processing, three teacher forcing styles are designed to enhance the capability of GAMA to reconstruct normal behaviors and thus improve detection performance. The effectiveness of GAMA is demonstrated through experiments for both trace- and attribute-level anomaly detection on both real-life and synthetic datasets. With an appropriate hidden layer size, GAMA can still capture normal patterns even when trained on a dataset containing anomalies and does not require a clean dataset, which is rarely available in the real-world.

A limited number of labeled anomalies are typically available in many real-world anomaly detection applications. These labeled anomalies may initially come from deployed detection systems, e.g., some successfully detected fraudulent transactions. A limited number of labeled anomalies can often be used as prior knowledge to train anomaly detection models. Future work will concentrate on improving the performance of anomaly detection models to detect anomalies using a limited number of labeled anomalies.

ACKNOWLEDGMENT

This work is supported by China National Science Foundation (Granted No. 62072301). This work is also partially supported by the Program of Technology Innovation of the Science and Technology Commission of Shanghai Municipality (Granted No. 21DZ1205000 and 21511104700).

REFERENCES

[1] Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng, "A deep multi-view framework for anomaly detection on attributed networks," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2539–2552, 2022. [Online]. Available: https://doi.org/10.1109/TKDE.2020.3015098

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[3] C. dos Santos Garcia, A. Meincheim, E. R. F. Junior, M. R. Dallagassa, D. M. V. Sato, D. R. Carvalho, E. A. P. Santos, and E. E. Scalabrin, "Process mining techniques and applications - A systematic mapping study," *Expert Syst. Appl.*, vol. 133, pp. 260–295, 2019. [Online]. Available: https://doi.org/10.1016/j.eswa.2019.05.003

[4] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser, "Binet: Multi-perspective business process anomaly classification," *Inf. Syst.*, vol. 103, p. 101458, 2022. [Online]. Available: https://doi.org/10.1016/j.is.2019.101458

[5] A. Sureka, "Kernel based sequential data anomaly detection in business process event logs," *CoRR*, vol. abs/1507.01168, 2015. [Online]. Available: http://arxiv.org/abs/1507.01168

[6] B. Kang, D. Kim, and S. Kang, "Real-time business process monitoring method for prediction of abnormal termination using knni-based LOF prediction," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 6061–6068, 2012. [Online]. Available: https://doi.org/10.1016/j.eswa.2011.12.007

[7] 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. [Online]. Available: https://doi.org/10.1016/j.datak.2011.07.002

[8] G. M. Tavares and S. Barbon Jr, "Analysis of language inspired trace representation for anomaly detection," in *International Conference on Theory and Practice of Digital Libraries*. Springer, 2020, pp. 296–308.

[9] S. B. Junior, P. Ceravolo, E. Damiani, N. J. Omori, and G. M. Tavares, "Anomaly detection on event logs with a scarcity of labels," in *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020*. IEEE, 2020, pp. 161–168. [Online]. Available: https://doi.org/10.1109/ICPM49681.2020.00032

[10] J. Ko and M. Comuzzi, "Online anomaly detection using statistical leverage for streaming business process events," in *Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers*, ser. Lecture Notes in Business Information Processing, vol. 406. Springer, 2020, pp. 193–205. [Online]. Available: https://doi.org/10.1007/978-3-030-72693-5_15

[11] ——, "Business process event log anomaly detection based on statistical leverage," in *Proceedings of the 1st Italian Forum on Business Process Management co-located with the 19th International Conference of Business Process Management (BPM 2021), Rome, Italy, September 10th, 2021*, ser. CEUR Workshop Proceedings, vol. 2952. CEUR-WS.org, 2021, pp. 7–12. [Online]. Available: http://ceur-ws.org/Vol-2952/paper_291a.pdf

[12] ——, "Detecting anomalies in business process event logs using statistical leverage," *Inf. Sci.*, vol. 549, pp. 53–67, 2021. [Online]. Available: https://doi.org/10.1016/j.ins.2020.11.017

[13] ——, "Keeping our rivers clean: Information-theoretic online anomaly detection for streaming business process events," *Inf. Syst.*, vol. 104, p. 101894, 2022. [Online]. Available: https://doi.org/10.1016/j.is.2021.101894

[14] H. 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, 2019. [Online]. Available: https://doi.org/10.1016/j.eswa.2019.04.052

[15] P. Krajsic and B. Franczyk, "Lambda architecture for anomaly detection in online process mining using autoencoders," in *Advances in Computational Collective Intelligence - 12th International Conference, ICCCI 2020, Da Nang, Vietnam, November 30 - December 3, 2020, Proceedings*, ser. Communications in Computer and Information Science, vol. 1287. Springer, 2020, pp. 579–589. [Online]. Available: https://doi.org/10.1007/978-3-030-63119-2_47

[16] ——, "Semi-supervised anomaly detection in business process event data using self-attention based classification," in *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES-2021, Virtual Event / Szczecin, Poland, 8-10 September 2021*, ser. Procedia Computer Science, vol. 192. Elsevier, 2021, pp. 39–48. [Online]. Available: https://doi.org/10.1016/j.procs.2021.08.005

[17] ——, "Variational autoencoder for anomaly detection in event data in online process mining," in *Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS 2021, Online Streaming, April 26-28, 2021, Volume 1*. SCITEPRESS, 2021, pp. 567–574. [Online]. Available: https://doi.org/10.5220/0010375905670574

[18] W. Guan, J. Cao, Y. Gu, and S. Qian, "Grasped: A gru-ae network based multi-perspective business process anomaly detection model," *IEEE Transactions on Services Computing*, vol. 16, no. 5, pp. 3412–3424, 2023.

[19] 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. [Online]. Available: https://doi.org/10.1007/s10994-018-5702-8

[20] T. Nolle, A. Seeliger, and M. Mühlhäuser, "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*, ser. Lecture Notes in Computer Science, vol. 9956, 2016, pp. 442–456. [Online]. Available: https://doi.org/10.1007/978-3-319-46307-0_28

[21] ——, "Binet: Multivariate business process anomaly detection using deep learning," in *Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings*, ser. Lecture Notes in Computer Science, vol. 11080. Springer, 2018, pp. 271–287. [Online]. Available: https://doi.org/10.1007/978-3-319-98648-7_16

[22] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Scalable process discovery and conformance checking," *Softw. Syst. Model.*, vol. 17, no. 2, pp. 599–631, 2018. [Online]. Available: https://doi.org/10.1007/s10270-016-0545-x

[23] R. Sarno and F. P. Sinaga, "Business process anomaly detection using ontology-based process modelling and multi-level class association rule learning," in *2015 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*. IEEE, 2015, pp. 12–17.

[24] F. Sinaga and R. Sarno, "Business process anomali detection using multi-level class association rule learning," *IPTEK Journal of Proceedings Series*, vol. 2, no. 1, 2016.

[25] R. Sarno, R. D. Dewandono, T. Ahmad, M. F. Naufal, and F. Sinaga, "Hybrid association rule learning and process mining for fraud detection." *IAENG International Journal of Computer Science*, vol. 42, no. 2, 2015.

[26] 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. [Online]. Available: https://doi.org/10.1186/s40537-019-0277-1

[27] K. Böhmer and S. Rinderle-Ma, "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*, ser. Lecture Notes in Computer Science, vol. 10033, 2016, pp. 80–98. [Online]. Available: https://doi.org/10.1007/978-3-319-48472-3_5

[28] A. Rogge-Solti and G. Kasneci, "Temporal anomaly detection in business processes," in *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, ser. Lecture Notes in Computer Science, vol. 8659. Springer, 2014, pp. 234–249. [Online]. Available: https://doi.org/10.1007/978-3-319-10172-9_15

[29] D. Rahmawati, R. Sarno, C. Fatichah, and D. Sunaryono, "Fraud detection on event log of bank financial credit business process using hidden markov model algorithm," in *2017 3rd International*

Conference on Science in Information Technology (ICSITech). IEEE, 2017, pp. 35–40.

[30] C. Linn and D. Werth, "Sequential anomaly detection techniques in business processes," in *Business Information Systems Workshops - BIS 2016 International Workshops, Leipzig, Germany, July 6-8, 2016, Revised Papers*, ser. Lecture Notes in Business Information Processing, vol. 263. Springer, 2016, pp. 196–208. [Online]. Available: https://doi.org/10.1007/978-3-319-52464-1_18

[31] M. G. Armentano and A. A. Amandi, "Detection of sequences with anomalous behavior in a workflow process," in *Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 9261. Springer, 2015, pp. 111–118. [Online]. Available: https://doi.org/10.1007/978-3-319-22849-5_8

[32] S. Huo, H. Völzer, P. Reddy, P. Agarwal, V. Isahagian, and V. Muthusamy, "Graph autoencoders for business process anomaly detection," in *Business Process Management - 19th International Conference, BPM 2021, Rome, Italy, September 06-10, 2021, Proceedings*, ser. Lecture Notes in Computer Science, vol. 12875. Springer, 2021, pp. 417–433. [Online]. Available: https://doi.org/10.1007/978-3-030-85469-0_26

[33] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[34] W. Guan, J. Cao, H. Zhao, Y. Gu, and S. Qian, "Wake: A weakly supervised business process anomaly detection framework via a pre-trained autoencoder," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2023.

[35] F. de Lima Bezerra and J. Wainer, "Anomaly detection algorithms in business process logs," in *ICEIS 2008 - Proceedings of the Tenth International Conference on Enterprise Information Systems, Volume AIDSS, Barcelona, Spain, June 12-16, 2008*, 2008, pp. 11–18.

[36] ——, "A dynamic threshold algorithm for anomaly detection in logs of process aware systems," *J. Inf. Data Manag.*, vol. 3, no. 3, pp. 316–331, 2012. [Online]. Available: https://sol.sbc.org.br/journals/index.php/jidm/article/view/1456

[37] W. M. P. van der Aalst and A. K. A. de Medeiros, "Process mining and security: Detecting anomalous process executions and checking process conformance," *Electron. Notes Theor. Comput. Sci.*, vol. 121, pp. 3–21, 2005. [Online]. Available: https://doi.org/10.1016/j.entcs.2004.10.013

[38] F. Bezerra, J. Wainer, and W. M. van der Aalst, "Anomaly detection using process mining," in *International Workshop on Business Process Modeling, Development and Support*. Springer, 2009, pp. 149–161.

[39] R. Rieke, M. Zhdanova, J. Repp, R. Giot, and C. Gaber, "Fraud detection in mobile payments utilizing process behavior analysis," in *2013 International Conference on Availability, Reliability and Security, ARES 2013, Regensburg, Germany, September 2-6, 2013*. IEEE Computer Society, 2013, pp. 662–669. [Online]. Available: https://doi.org/10.1109/ARES.2013.87

[40] D. Rahmawati, M. A. Yaqin, and R. Sarno, "Fraud detection on event logs of goods and services procurement business process using heuristics miner algorithm," in *2016 International Conference on Information & Communication Technology and Systems (ICTS)*. IEEE, 2016, pp. 249–254.

[41] 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. [Online]. Available: https://doi.org/10.1016/j.is.2012.04.004

[42] M. Ebrahim and S. A. H. Golpayegani, "Anomaly detection in business processes logs using social network analysis," *J. Comput. Virol. Hacking Tech.*, vol. 18, no. 2, pp. 127–139, 2022. [Online]. Available: https://doi.org/10.1007/s11416-021-00398-8

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[44] D. C. Hoaglin and R. E. Welsch, "The hat matrix in regression and anova," *The American Statistician*, vol. 32, no. 1, pp. 17–22, 1978.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[46] J. Evermann, J. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," *Decis. Support Syst.*, vol. 100, pp. 129–140, 2017. [Online]. Available: https://doi.org/10.1016/j.dss.2017.04.003

[47] N. Tax, I. Verenich, M. L. Rosa, and M. Dumas, "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*, ser. Lecture Notes in Computer Science, vol. 10253. Springer, 2017, pp. 477–492. [Online]. Available: https://doi.org/10.1007/978-3-319-59536-8_30

[48] J. Evermann, J. Rehse, and P. Fettke, "A deep learning approach for predicting process behaviour at runtime," in *Business Process Management Workshops - BPM 2016 International Workshops, Rio de Janeiro, Brazil, September 19, 2016, Revised Papers*, ser. Lecture Notes in Business Information Processing, vol. 281, 2016, pp. 327–338. [Online]. Available: https://doi.org/10.1007/978-3-319-58457-7_24

[49] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[50] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 4, pp. 686–705, 2019. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2841877

[51] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *CoRR*, vol. abs/1710.10903, 2017. [Online]. Available: http://arxiv.org/abs/1710.10903

[52] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990. [Online]. Available: https://doi.org/10.1109/72.80202

[53] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing - Algorithms, Architectures and Applications, Proceedings of the NATO Advanced Research Workshop on Neurocomputing Algorithms, Architectures and Applications, Les Arcs, France, February 27 - March 3, 1989*, ser. NATO ASI Series, vol. 68. Springer, 1989, pp. 227–236. [Online]. Available: https://doi.org/10.1007/978-3-642-76153-9_28

[54] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 665–674. [Online]. Available: https://doi.org/10.1145/3097983.3098052

[55] W. M. P. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, 2004. [Online]. Available: https://doi.org/10.1109/TKDE.2004.47

[56] S. J. Leemans, D. Fahland, and W. M. Van Der Aalst, "Discovering block-structured process models from event logs-a constructive approach," in *Application and Theory of Petri Nets and Concurrency: 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings 34*. Springer, 2013, pp. 311–329.

[57] D. Sommers, V. Menkovski, and D. Fahland, "Process discovery using graph neural networks," in *3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021*. IEEE, 2021, pp. 40–47. [Online]. Available: https://doi.org/10.1109/ICPM53251.2021.9576849

[58] A. Y. Hussein, P. Falcarin, and A. T. Sadiq, "Enhancement performance of random forest algorithm via one hot encoding for iot ids," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 9, no. 3, pp. 579–591, 2021.

[59] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, ser. JMLR Workshop and Conference Proceedings, vol. 28. JMLR.org, 2013, pp. 1310–1318. [Online]. Available: http://proceedings.mlr.press/v28/pascanu13.html

[60] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[61] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 823–839, 2012. [Online]. Available: https://doi.org/10.1109/TKDE.2010.235

[62] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. The MIT Press, 1999, pp. 582–588. [Online]. Available: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[64] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

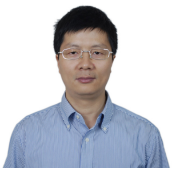
[65] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, ser. JMLR Proceedings, vol. 9. JMLR.org, 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[67] A. Burattin, "PLG2: multiperspective processes randomization and simulation for online and offline settings," *CoRR*, vol. abs/1506.08415, 2015. [Online]. Available: http://arxiv.org/abs/1506.08415

[68] P. B. Nemenyi, *Distribution-free multiple comparisons.* Princeton University, 1963.

Wei Guan is currently a Ph.D student in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests including predictive monitoring, anomaly detection and machine learning.



Jian Cao is currently a tenured professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He is also the director of research institute of network computing and service computing. Dr. Cao's research interests include intelligent data analytics, service computing, collaborative computing and software engineering. Besides national and provincial government research grants, his research is also supported by many industry partners. He has published more than 300 research papers in prestigious journals and conferences. Dr. Cao has won 10 ministerial or provincial level scientific and technological achievements rewards. Currently, he is the distinguished member of CCF and the senior member of IEEE.

Yang Gu is currently a Ph.D student in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include service computing and data mining.

Shiyou Qian received a PhD degree in computer science from Shanghai Jiao Tong University, China, in 2015. He is currently an associate researcher with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include event matching for content-based publish/subscribe systems, resource scheduling for Hybrid-Cloud, and driving recommendation with vehicular networks.