IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 22, 2021, accepted April 15, 2021, date of publication April 20, 2021, date of current version April 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3074149

Real-Time Abnormal Insider Event Detection on Enterprise Resource Planning Systems via Predictive Auto-Regression Model

JONGMIN YU[1,*], MINKYUNG KIM[2,*], HYEONTAEK OH[1], (Member, IEEE), AND JINHONG YANG[3]

[1]Institute for IT Convergence, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea
[2]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea
[3]Department of Healthcare and Information Technology, Inje University, Gimhae 50834, Republic of Korea

Corresponding author: Jinhong Yang (jinhong@inje.ac.kr)

This work was supported by the Grant from Inje University, in 2019.

*Jongmin Yu and Minkyung Kim contributed equally to this work.

ABSTRACT With the development of information and communication technology and the globalisation of enterprises, many companies are being operated through an electrical resource management system called Enterprise Resource Planning (ERP) system. An ERP system enables efficient and centralised resource management for enterprises. However, since many enterprise resources are being managed by the system, the threatening behaviour by an insider is one of the most significant risks in operating ERP systems. It is much stealthier and fatal compared with the threat from an outsider since it is considered as normal events that accessing the enterprise resource of insiders. Conventional insider threat detection methods have aimed to detect particular events manually defined by system administrators. Those approaches are not robust to the variation of event patterns, and they can not be used when the predefined cases are not given. In this paper, we present a real-time abnormal insider event detection method using the Predictive Auto-regression Model (PAM). Compared with the conventional approaches, the proposed method compiles a prediction model using normal events and identifies threat when the likelihood of prediction results is lower than a threshold. Experiments are conducted using a dataset including events defined as a sequence of ERP system logs. The logs are captured in a practical situation of an enterprise. The results demonstrate that the proposed method can successfully identify abnormal events of on ERP systems.

INDEX TERMS Enterprise resource planning system, enterprise threat prediction, auto-regression model.

I. INTRODUCTION

Enterprise Resource Planning (ERP) system is a comprehensive resource management system that helps to manage business processes across all sectors of a company [1]. In this system, as shown in Figure 1, it is possible to organically share resources and information of the overall business process (such as personnel, finance, production, sales, and operation) to improve operational efficiency. In other words, the ERP system acts as an essential business hub in most modern companies with a complex corporate structure. However, the ERP system, which handles sensitive and valuable data within the enterprise, faces the risk of an enterprise threat that breaches such critical information. In particular, enterprise threats by insiders have been attending as one of the most important issues that have to be considered. An insider includes not only a current and former employee but also a contractor or business partner, who has proper access privileges for a system and data of organisation [2]. In general, an insider threat is significantly more risky and costly for companies than a threat caused by an external attacker [3]. Because the insider already has legitimate access to the system, they typically know where the sensitive data are stored in the organisation. In addition, it is easy to consider that accessing enterprise resources by insiders is normal. According to the report [4], over about 80% of the security managers responded that insider threat effectiveness of their organisations is more than 'somewhat effective'. It shows the needs to detect insider threats and protect the organisation have been raised until recently.

The associate editor coordinating the review of this manuscript and approving it for publication was Weiping Ding.

FIGURE 1. An overview of enterprise resource planning (ERP) system. As computerisation of enterprises, many enterprise resources have been managing using ERP systems. It has been improving the efficiency for managing the enterprise.

On the rise of the need for preventing insider threats, role- and scenario-based approaches were presented to detect system insiders' abnormal activities [5]–[7]. Those approaches obtained solid performance for identifying pre-defined or known unusual or threatening behaviours. However, they have a limitation in that they can not cope with unexpected unusual behaviours. To overcome the limitation of using predefined information on unusual behaviours, and as large datasets become available, various deep learning-based approaches have been proposed [8]–[10]. Those approaches tried to detect unusual insider behaviour by analysing unusual movements of input devices or network signals leading to data leakage. However, such indirect approaches have a limitation in that they can frequently make false alarms that may not be related to the actual unusual insider behaviours on ERP systems. In addition, they did not consider the need for insider threat detection to be done in real-time.

With the consideration of those limitations, in this paper, we present a real-time abnormal insider event detection method on ERP systems, which can identify abnormal events of enterprise insiders without any predefined abnormal or unusual event cases. Straightforwardly, the key hypothesis on this paper is that a model trained by only normal events can predict normal events well. With this hypothesis, the proposed method is trained by only normal event samples, and abnormal events are identified when the predicted results are unmatched with the corresponding future events. In this way, the proposed model, particularly using the prediction results, can respond to abnormal user events in near real-time. For this methodology, it is important to derive a precise prediction model. To improve the prediction results, we propose a Predictive Auto-regression Model (PAM) that can derive a more discriminative prediction model. To demonstrate the effectiveness of the proposed method in detecting abnormal insider events, we evaluate our method exploiting a real-world event dataset. The dataset consists of ERP system logs captured from an enterprise in real-world. The experimental results shows the effectiveness of the proposed method in detecting abnormal events on ERP systems.

The key contributions of this paper are as follows:

- A novel real-time abnormal insider event detection method is proposed based on long short-term memory (LSTM). The proposed method provides a robust detection performance for abnormal insider events with real-time processing speed.
- A Predictive Auto-regression Model (PAM) is proposed to derive a accurate prediction model using given normal events, which utilises a proposed discriminative probabilistic model for identifying normal and abnormal events.

In addition to the above contribution, we provide comprehensive experimental results to find optimal hyper-parameter settings. Also, we provide a performance comparison with various methods, including conventional time-series prediction models and existing state-of-the-art methods for abnormal insider event detection (AIED) and enterprise threat detection (ETD).

The rest of this paper is organised as follows. In Section II, we introduce related works on AIED and ETD. In Section III, we describe the details of the proposed methods, including overall architecture of PAM. We provide an explanation for the experimental details and results in Section IV. We conclude this paper in Section V.

II. RELATED WORKS

Enterprise threat detection has long been an important issue with the widespread use of ERP systems. Among them, existing studies for insider threat detection have been tried with a variety of approaches ranging from human-oriented (*e.g.,* rule- or scenario-based) to machine-oriented (*e.g.,* machine learning- or deep learning-based) detection methods. First, the rule- and scenario-based detection are performed based on prior knowledge of insider threats [5]–[7]. Sandhu *et al.* [5] proposed a role-based access control, and they analysed the user behaviour in the view of each user's role. On the other hand, Islam *et al.* [6] proposed a structure for scenario definition to identify and define user activities in ERP systems. Then, the authors analysed system log data that matches identified fraudulent activity through their scenario definition structure. Lu *et al.* [7] proposed a method to convert the event data on ERP systems to an artefact-centric process and discover causal dependencies between artefacts. Then, unusual processes were detected based on the predefined dependencies between artefacts. These methods showed outstanding performance in detecting unusual activities or processes on ERP systems. However, they have a limitation in that they need prior knowledge and they can not cope with unexpected unusual events.

To overcome the above limitations, several machine learning-based methods that are data-driven have been proposed [11]–[13]. They mainly considered that abnormal insider events can be detected based on the understands of normal events. Figure 2 illustrates a general process of AIED approaches based on machine learning. Rashid *et al.* [11] proposed a method to design a user

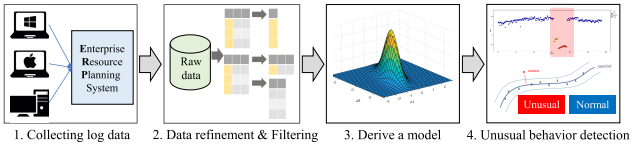


FIGURE 2. A general process for Abnormal Insider Event Detection (AIED) on ERP systems.

behaviour model from log data based on Hidden Markov Models. They trained the model using log data only corresponding normal behaviour. And then, they detected a log sequence as abnormal behaviour if the probability of the log sequence occurring in the designed model is significantly low. Sun *et al.* [13] proposed a framework for anomalous behaviour detection using an extended isolation forest method to handle data that can be categorised. Hu *et al.* [12] applied the principal component analysis to extract characteristics of a user's normal activity. The user's unusual activity is detected when the reconstruction error is high, which is obtained by using K-principal components. In the above methods, they were able to detect events that deviate from prior knowledge, but these conventional approaches have still a limitation in that they are not able to process the entire log data due to high computational complexity [9], [14].

In recent years, deep learning-based methods have been presented showing better performance [9], [10], [14], [15] compared with the machine learning-based methods. Tuor *et al.* [15] showed a method to detect abnormal network activities from system logs based on a long short-term memory (LSTM) model. However, before training the model, they periodically converted the log data into accumulated counts of each activity over a particular time range, which should be done by a system user. In other words, it needs human intervention to define the activity list from the log data. In addition, the model trained with aggregated features may overlook unusual activities. Yuan *et al.* [9] and Sharma *et al.* [10] proposed an LSTM based framework for identifying insider threats. Those methods identify abnormal events according to the reconstruction error. However, they also required an manually defined activity list to convert raw data into model inputs.

III. THE PROPOSED METHOD

A. DATA PREPROCESSING

In general, ERP system logs are represented by the combination of alphanumeric characters. However, those types of representation can not be accepted as an input of computational models. Therefore, it is necessary to change the system logs to signals that are applicable to deep learning models. In order to change the raw system logs, one-hot encoding can be regarded as the simplest approach to change any categorised logs into the signals. Du *et al.* [16] leverage a one-hot encoding for transforming raw system logs for their computational systems since it is straightforward and efficient. Unfortunately, due to the methodological simplicity of the one-hot vector encoding, it is sometimes considered as a cause of the curse of dimensionality when expanding a new dimension for a new class [17]. Since its methodological properties represent a class using one-hot vector, the size of encoding space is linearly proportional to the number of signal classes. Therefore, it is inefficient in the perspective of computational costs.

To overcome those issues, we employ dense vector embedding [18] for preprocessing the raw system logs. The data preprocessing task in this paper is conducted as follows. Initially, given raw ERP system logs are mapped to positive integers $h = \{h_t\}_{t=1:n}$, where h_t and n indicate a t^{th} log and the length of log sequence for single user session, respectively. The mapped integers are encoded to dense vectors $x = \{x_t\}_{t=1:n} \in \mathcal{X}$ (where $x_t \in \mathbb{R}^m$), where m is predefined dimensionality for the dense embedding method using an neural network based embedding network. After the encoding task is over, the vectors are utilised to train the prediction model. We conducted an ablation study on the approaches for input data encoding in Section IV-D.

B. METHODOLOGY OVERVIEW

As similar to various anomaly detection methods [19], [20] or outlier detection approaches [21], [22], our method also aims to derive a model for capturing normal events, and then it expects a larger error or lower likely when abnormal samples are given. Figure 3 shows an overview of the proposed AIED for ERP systems. With a given event sample x, AIED is carried out by comparing a computed error or likely with a threshold (τ), which is manually set by a system administrator, as follows:

$$\text{AIED}(x) = \begin{cases} \text{Abnormal}, & \text{if} \quad \mathcal{F}_{\mathcal{X}^{\text{nor}}}(x) \leq \tau \\ \text{Normal}, & \text{otherwise}, \end{cases}$$

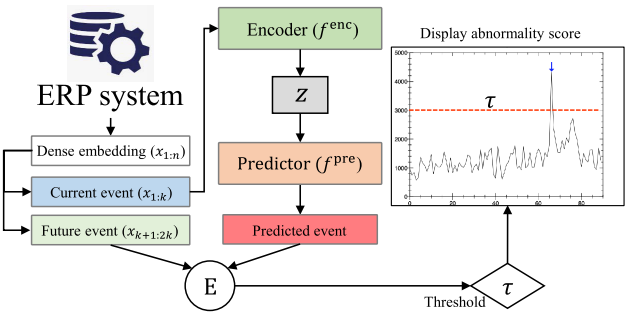


FIGURE 3. Abnormal Insider Event Detection (AIED) on an ERP system using the proposed method. Initially, the current event taken by a ERP system is encoded to the latent feature z by the encoder f^{enc}. z is applied to predictor f^{pre} to produce prediction results. Abnormal insider event detection is conducted by comparing the predicted event and the future event provided by the predictor and the ERP system, respectively.

where $\mathcal{F}$ can be defined by a discriminative model or a stochastic model on given normal event samples $\mathcal{X}^{\text{nor}}$. Depending on the types of the models (such as discriminative model or stochastic model), the outcomes can be understood

as an error or a likelihood. As this methodology, it is important to build a model which can cover all features of given normal events.

We employ the prediction setting rather than the reconstruction setting, which is the most common way to compile a probabilistic distribution from a given dataset [23]. There is an advantage in leveraging prediction setting for deriving the model. The reconstruction set requires constrained inputs and outputs, but the prediction setting is more flexible in learning the correlation between the input and output because it is changeable to decide the output corresponded to each input by changing the time intervals between them. This methodological property can improve the generalisation performance of the model by providing more diverse patterns of input samples.

The proposed method initially encodes current events to latent features via an encoder f^{enc} and produces a prediction results using the predictor f^{pre}. The encoding process for the current events is represented as follows:

$$z = f^{\text{enc}}(x^{\text{C}}), \tag{1}$$

where f^{enc} an encoder module on the propose method. x^{C} and z denote the current event samples and the extracted latent features, respectively. Using the latent feature z, the proposed method predicts future events through the predictor f^{pre} as follows:

$$\bar{x} = f^{\text{pre}}(z), \tag{2}$$

where f^{pre} denotes the predictor on PAM, and $\bar{x}$ indicates the prediction results.

A recurrent neural network (RNN) structure is exploited for the encoder f^{enc} and the predictor f^{pre} to process sequential data, and it is a commonly used approach to handle time-series data [10], [15], [19]. The encoder f^{enc} and the predictor f^{pre} are compiled using LSTM cells [24].

a LSTM consists of an input gate i, a forget gate g, and an output gate o. The input gate i_t at the t-th time step decides the weight representing the influence of the new input on the current internal state s_t at time t. The process of the input gate is represented as follows:

$$i_t = \sigma(W_{is}s_{t-1} + W_{ih}\alpha_{t-1} + W_{ix}\gamma_s^t + b_i), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function to map input into a value between 0 and 1. When the value is close to 1, the input feature γ_s^t becomes more important. W_{is} and W_{ih} are weight parameters related to the state s_{t-1} and hidden state α_{t-1}. W_{ix} and b_i are the weight matrix and the bias for the spatial feature γ_s.

The forget gate g_t adjusts the previous state s_{t-1} to control its contribution to s_t. The computation process of the forgot gate is defined as follows:

$$g_t = \sigma(W_{gs}s_{t-1} + W_{gh}\alpha_{t-1} + W_{gx}\gamma_s^t + b_g), \tag{4}$$

where W_{gs}, W_{gh}, and W_{gx} denote the weight matrices for s_{t-1}, α_{t-1}, and γ_s^t, respectively. b_g denotes the bias. By leveraging these inputs s_{t-1}, α_{t-1}, γ_s^t, g_t, and s_t of each LSTM cell is updated as follows:

$$s_t = g_t \otimes s_{t-1} + i_t \otimes \tanh(W_{sh}\alpha_{t-1} + W_{sx}\gamma_s^t + b_s), \tag{5}$$

where $\otimes$ indicates the element-wise product, and W_{sh} and W_{sx} denote the weight matrices related to the hidden state α_{t-1} and spatial feature γ_s^t, respectively.

The output gate o_t determines the influence of the current state on the future state. It is defined as

$$c_t = \sigma(W_{cs}s_t + W_{ch}\alpha_{t-1} + W_{cx}\gamma_s^t + b_c), \tag{6}$$

where W_{cx}, W_{ch}, and W_{cs} denote the weight parameters corresponding to s_t, h_{t-1}, and γ_s^t, respectively. b_c indicates a bias of the output gate. The hidden state of LSTM is computed as

$$\alpha_t = c_t \otimes \omega(s_t), \tag{7}$$

where ω is ReLu activation function. Note that a RNN strucuture used for LSTM can be replaced by a gate recurrent unit (GRU) structure [25]. We would provide ablation studies about the AIED depending on the memory cell types on our experiments.

When event prediction is over using the encoder and the predictor, detecting anomalous event is conducted by comparing a prediction error with a predefined threshold τ. As shown in Figure 3, the prediction error is obtained by comparing the predicted results and the corresponding future events. The error is formulated based on Mean Square Error (MSE), and this process is represented as follows:

$$\text{E}(x^{\text{F}}, \bar{x}) = \frac{1}{l_x}\sum_{i=0}^{l_x}(x_t^F - \bar{x}_t)^2, \tag{8}$$

where x^{F} and $\bar{x}$ indicate the future events and the predicted results by the proposed method, respectively. l_x denotes the temporal length of x^{F}.

As a result, the AIED workflow with the error is defined as

$$\text{AIED}(x) = \begin{cases} \text{Abnormal}, & \text{if} \quad \text{E}(x^{\text{F}}, \bar{x}) \geq \tau \\ \text{Normal}, & \text{Otherwise}, \end{cases} \tag{9}$$

where τ is a threshold of event abnormality, and it is decided manually. Since abnormal events are detected by comparing the computed error and manually defined threshold τ, the performance of AIED is variant to the value of the threshold.

C. PREDICTIVE AUTO-REGRESSION MODEL

To obtain a robust probabilistic model from a give normal event dataset, we employ a prediction manner. The prerequisite for precise AIED is deriving an optimal prediction model to compute probability $p(x^{\text{F}}|x^{\text{C}})$ with respect to the current event x^{C} and the corresponding future event x^{F}. As the workflow of the proposed method illustrated in Figure 3, the encoder f^{enc} initially maps the current event x^{C} into the latent feature as $f^{\text{enc}} : x^{\text{C}} \rightarrow z$, and then the predictor f^{pre} outputs the prediction results as $f^{\text{pre}} : z \rightarrow \bar{x}$.

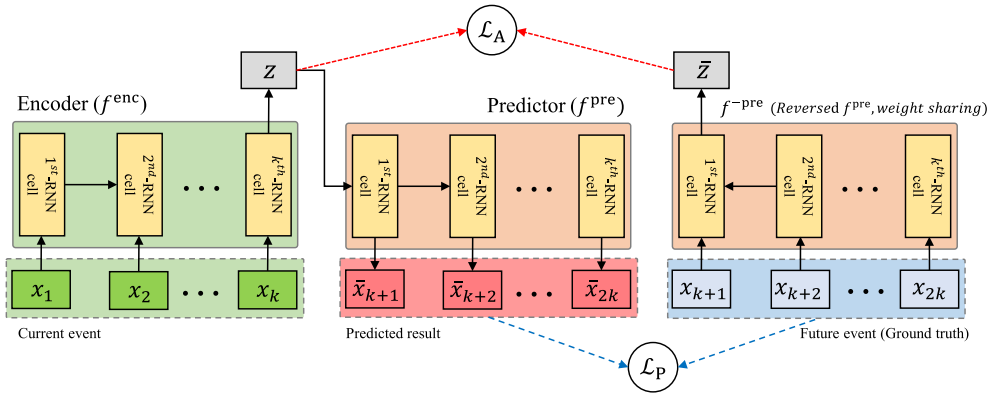


FIGURE 4. Architectural details of the proposed method. The black solid lines denote the common pipelines to compute the auto-regression loss $\mathcal{L}_A$ and the prediction error loss $\mathcal{L}_P$. The red dense dotted lines represents the workflow to compute the $\mathcal{L}_A$, and the blue sparse dotted lines define the workflow to compute $\mathcal{L}_P$.

Using above notations, $p(x^F|x^C)$ can be reformulated with $p(z|x^C)$ and $p(x^F|z, x^C)$ as follows:

$$p(x^F|x^C) \approx \int_z p(x^F|z)\frac{p(z|x^C)}{p(z)}dz, \tag{10}$$

where $p(x^C)$ and $p(z)$ denote the prior probabilities of x^C and z, respectively. $p(x^C)$ can be derived by given event samples defined as current events. Consequently, to derive optimal $p(z|x^C)$, $p(x^F|z)$ and $p(z)$ are necessary to attain the optimal $p(x^F|x^C)$, and we deal with this issue in term of the proposed Predictive Auto-regression model (PAM).

Structural details of the proposed PAM is illustrated in Figure 4. PAM is composed of two main pipelines for computing the prediction error loss $\mathcal{L}_P$ and the auto-regression loss $\mathcal{L}_A$, and these are all working under the encoder and the predictor. When a preprocessed event $x_{1:2k}$ is given (where $n = 2k$), it is divided into the current event $x_{1:k}$ (as x^C in Eq. (1)) and the future events $x_{k+1:2k}$ (as x^F in Eq. (8)). And then, the current event is encoded to the latent features though the encoder f^{enc} (Eq. (1)). The latent feature is applied to the predictor for generating the prediction result $\bar{x}_{k+1:2k}$ (Eq. (2)).

$x_{k+1:2k}$ and $\bar{x}_{k+1:2k}$ are used to compute the prediction error loss $\mathcal{L}_R$. The prediction error loss is formulated using the Euclidean distance, and it is represented by

$$\mathcal{L}_P(x_{k+1:2k}, \bar{x}_{k+1:2k}) = \|x_{k+1:2k} - \bar{x}_{k+1:2k}\|_2^2. \tag{11}$$

Minimising the $\mathcal{L}_P$ basically allows to find suitable $p(z|x^C)$ and $p(x^F|z)$.

However, using the prediction error loss only may not be enough to learn globalised feature since it can not consider abstracted representation on a latent space. Methodologically, the prediction error loss minimises an error on a sample space, which is neither abstracted nor generalised, as closely as possible. During this error minimisation process, abstracted features are mapped into a latent space, which are more globalised features than the original inputs. However, according to the previous works [10], [16], if a latent feature space is arbitrarily constructed, the randomness may cause uncertainty in reconstructing the input samples [26]. As a result, the uncertainty degenerate prediction performance, and the poorly reconstructed results can degrade the AIED performance.

Therefore, we apply a auto-regression loss $\mathcal{L}_A$ on the latent feature to reduce the uncertainty. We push the distribution of the latent features used for prediction, notated by z, to be close to a true distribution embedded from true future events $x_{k+1:2k}$. As shown in Figure 4, the reverse operation of the predictor f^{pre} plays as the role of an encoder to take the latent feature from the given future event. The reverse processing on the predictor f^{pre} is represented as follows:

$$\bar{z} = f^{-pre}(x_{k+1:2k}), \tag{12}$$

where f^{-pre} denotes the reverse operation of the predictor f^{pre}, and $\bar{z}$ is the latent feature extracted from the given future events in the training step. Since this step does not require a new model, it is not affect to the model complexity. The reverse operation is conducted based on the weight sharing from the original prediction model f^{pre}, which is shown in Figure 4.

In minimising the difference between the distributions of two latent features, we measure the difference using the Kullback–Leibler (KL) divergence [27]. With given z and $\bar{z}$, the auto-regression loss for the latent features is defined as follows:

$$\begin{aligned}\mathcal{L}_A(z, \bar{z}) &= D_{KL}(z||\bar{z}) = \int_z \int_{\bar{z}} z\log(\frac{z}{\bar{z}})d\bar{z}dz \\ &\approx \sum_i \sum_j z_i \log(\frac{z_i}{\bar{z}_j}),\end{aligned} \tag{13}$$

where D_{KL} denotes the KL divergence module, z_i and $\bar{z}_j$ indicate i-th and j-th element on the latent features z and $\bar{z}$, respectively. The minimum value of the KL divergence is 0, which means that the two distributions are the exactly same.

Consequently, the total loss function is defined by the combination of the prediction error loss and the auto-regression

TABLE 1. An example of Security Audit Log (SAL) codes on the SAP-ERP systems. Those codes are randomly picked among the 146 SAL codes. The event addressed in this paper is defined by the series of those codes.

SAL code	Behaviour
AU2	Logon failed (reason=&B, typ=&A, method=&C)
AUM	User &B Locked in Client &A After Erroneous Password Checks
BUE	WS: Delayed logon failed (type &B, WP &C). Refer to Web service log &A.
BUI	WS: Delayed logon successful (type &B, WP &C). Refer to Web service log &A.
CU4	SPNego replay attack detected (UPN=&A)

loss as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{P}}(x_{k+1:2k}, \bar{x}_{k+1:2k}) + \lambda \mathcal{L}_{\text{A}}(z, \bar{z}), \tag{14}$$

where λ is a balancing weight between the two losses. In this paper, we set 0.1 as a value of the balancing weight for the best performance.

D. MODEL COMPLEXITY

According to Apaydin *et al.* [28], the model complexity of PAM using LSTM is $O(2W)$ with $W = n_c \times n_c \times 3 + n_i \times n_c \times 3 + n_c \times n_o + n_c \times 2$, where n_c, n_i, and n_o indicate the number of LSTM cells, input units, and output units, respectively. Also, the complexity of the dense vector embedding via a neural network, is $O(N^2)$, where N is the dimension of the hidden layers used for the dense vector embedding via neural networks. The model complexity of the proposed method is, therefore, $O(2W + N^2)$. However, the model complexity can be changed depending on the temporal-length of input event samples since it can increase or decrease the number of memory cells integrated into the encoder and the predictor. In addition to the theoretical model complexity, the practical execution speed is essential to demonstrate the processing speed as real-time applications for our work. We justify the ability of the real-time processing of our methods in our experiments.

IV. EXPERIMENTS

A. DATASET AND EVALUATION METRICS

We create a dataset containing the security audit logs (SAL) [29] recorded from a real-world SAP-ERP system [30]. Note that SAP occupied the largest ERP market share in 2019 according to the report [31]. A SAL code on SAP-ERP systems is originally represented by a combination of three alphanumeric characters as shown in Table 1, and there are 146 codes to represent user behaviours. An event is a sequence of the SAL from one user session (from the system log in to log out), and the events addressing in this paper are defined by the series of those codes. We replace a SAL code to a single positive integer (*e.g.,* "AU2" $\rightarrow$ 1, "AUM" $\rightarrow$ 2, etc.) for improving experimental efficiency.

Our dataset contains 32,512 normal event samples and 1,251 abnormal event samples. The dataset is split into two subsets for model training and performance evaluation. The training set only contains normal events samples of 20,621. The test set consists of 11,891 normal events and 1,251 abnormal event samples. Those event samples are recorded from a cosmetics conglomerate in Republic of Korea.

Since an abnormal event for AIED is hard to be prepared for constructing the experimental dataset in the real-world, abnormal events are generated by utilising the random sequence generations and the scenario-based sequence generations. By counting SAL codes for normal events, the random sequence generation model produces log sequences with the rarely used (or not used) logs in the system. In this case, rarely used logs can be considered abnormal events. On the other hand, the scenario-based sequence generation model creates log sequences with the well-known insider threats on SAP-ERP systems. The scenarios are referred from Cappelli *et al.* [2].

AIED can be considered as a one-class classification, and the performance of AIED varies according to the detection threshold τ. In this work, we utilise the receiver operating characteristic, the area under curve (AUC), and the equal error rate (EER). Those three measurements are computed using the true positive rate (TPR) against the false positive rate (FPR) obtained with varied thresholds. AUC and EER are computed quantitative metrics computed from ROC curves.The closer the AUC is to 1 and the EER is closer to 0, the model can be considered as better performance.

B. IMPLEMENTATION

LSTM [32] is used for the memory cell on the encoder f^{enc} and the predictor f^{pre}. The dimensionalities of hidden layers on the memory cells and the dense embedding vector for the data preprocessing are fixed to 128. To improve the generalisation performance of the proposed method, the data augmentation (through a temporal length expending or shrinking [33]) is applied in the training step. All models are optimised using Stochastic Gradient Descent (SGD) with a momentum of 0.9 for 60 epochs. At the beginning of the training, the initial learning rates of the encoder and the predictor set to 0.1, and the learning rate is decayed to a tenth for every 10 epoch. The size of the batch is 256. We implement our method using Pytorch, and all experiments are carried out with GTX2080Ti.

Among the above hyper-parameter setting, the LSTM can be replaced by GRU [25]. Additionally, the performance of the proposed method can be varied depending on the dimension of the hidden layers on the memory cells and the scale of dense embedding vectors. We provide insights to decide these hyper-parameters in the following section.

C. EFFECT OF THE MEMORY CELL TYPES

In this experiment, we investigate the AIED performance according to the memory cell types and the dimension of the hidden units. To this end, we compiled the two different types of PAM: PAM+LSTM [32] and PAM+GRU [25]. In addition, the number of hidden units for each memory cell type is set to 64, 128, and 256, respectively. All hyper-parameter settings for training those models are all the same (described in Section IV-A). We employ the dense embedding with a dimension of 128 as the input. The results are shown in Table 2.

TABLE 2. AUC and EER depending on the recurrent network structure and dimensionality of hidden unit. PAM using LSTM with the hidden units of 128 produces the best performance.

Network model	PAM + LSTM [32]			PAM + GRU [25]		
# of hidden units	64	128	256	64	128	256
AUC	84.15	**86.45**	85.01	83.61	84.71	82.65
EER	20.03	**19.58**	19.97	22.75	19.95	21.52

First, regarding the number of hidden units, setting 128 hidden units shows the best performance for both memory cell types. Then, AUC performances with LSTM are shown in the order of 256 and 64 hidden units. On the other hand, AUC performances with GRU are shown in the order of 64 and 256 hidden units. In the case of EER, EER performances with LSTM are shown in the same order as AUC performances. However, EER performances with GRU are shown in the order of 256 and 64 hidden units, which is opposed to the result of AUC performances.

However, in both memory cell types, the highest AUC and EER performance was obtained when the median value (i.e., 128) among the number of hidden units used, which can be explained as follows. In general, the number of hidden units is related to the capacity of model. However, when too few hidden units are used, the density of the data feature space becomes higher than necessary. It makes that the features cannot express the characteristics of data in detail, which leads to performance degradation. Conversely, even when too many hidden units are used, performance may deteriorate due to the high dimensionality of the data feature space.

Next, in all the number of hidden units used in the experiment, the performances with LSTM are higher than those with GRU. Both LSTM and GRU are memory cells are proposed to solve the gradient vanishing problem that occurs in vanilla RNN when a sequence of data is long. However, on the ERP data used in this paper, LSTM obtained higher performance than GRU with a little lower model complexity under the same experimental setting. According to the above results, in the subsequent experiments of this paper, the performance of PAM is evaluated using a LSTM having 128 hidden units as a memory cell.

D. EFFECT OF THE INPUT TYPES

Table 3 shows AIED performance according to the types of input. Here, types of input mean the vectorisation methods of ERP data. For this experiment, we implement two types of input: 1) one-hot encoding and 2) dense embedding. In the case of one-hot encoding, the dimension of input is 146, which is the same as the number of unique SAL codes in the dataset. In the case of dense embedding, as in the previous experiment, the input dimensions are set to 64, 128, and 256. In addition, a case for 146 input dimensions is added for a fair comparison with one-hot encoding. Except for the input type and input dimension, the remaining settings of the experiment are the same. According to the previous experiment results in Table 2, LSTM is used as a memory cell.

First, when one-hot encoding and dense embedding are having the same input dimension, the performance is compared as follows. For one-hot encoding, 74.52 of AUC and 25.61 of EER are obtained. For dense embedding, 86.45 of AUC and 19.58 of EER are obtained. The dense embedding shows superior performance to one-hot encoding. Furthermore, even when the dense embedding dimension (*i.e.,* 64) is smaller than half of the one-hot encoding dimension (*i.e.,* 146), dense embedding produces higher performance than one-hot encoding. Dense embedding, unlike one-hot encoding, considers the relationship between each category in the vectorisation process of data with categorical information. In other words, dense embedding generates more informative input features in training deep learning models than one-hot encoding, which leads to improved performance as shown in Table 3.

TABLE 3. AUC, EER, and the execution speed of the proposed method according to the encoding vector types. 'DEV' represents the dense embedding vector. The figures in the brackets define the dimensionality of the embedding vector space. The bolded numbers indicate the best performances.

Input type (Dim)	AUC	EER	Execution speed (ms)
One-hot vector (146)	74.52	25.61	10.2
DEV (64)	84.07	20.62	**2.4**
DEV (128)	**86.45**	**19.58**	3.4
DEV (146)	85.72	20.03	3.9
DEV (256)	83.12	21.62	5.8

Next, we evaluate the performance of dense embedding according to the input dimension. The highest performance for both AUC and EER is achieved when 128 dimensions are used. The performance is followed in the order of 128, 146, 64, and 256 dimensions from the highest to the lowest. This difference in performance can be explained in the same context as the difference in performance according to the number of hidden units shown in the previous experiment. In the case of the execution time, the shortest time was taken in the lowest dimension among dense embedding, and as the dimension increased, the execution time also increased.

E. PERFORMANCE COMPARISON

To evaluate the effectiveness of the proposed method for AIED, our method is compared with two conventional approaches and three recently published state-of-the-art methods. Compared methods include Hidden Markov Model

(HMM) [34], Dynamic Baysian Network (DBN) [35], and the following studies: Tuor *et al.* [15], Yuan *et al.* [9], and Sharma *et al.* [10]. For our method, a model is tuned by LSTM with 128 hidden units, and dense embedding input with 128 dimensions is used according to the previous results.

Table 4 shows the quantitative performances. The conventional approaches to derive probabilistic models, which are HMM and DBN, have obtained AUCs of 56.70 and 60.56, which are relatively lower than the proposed methods. The performance gaps between these methods and the proposed method can be thought that the proposed PAM can derive more discriminative probabilistic model than the others. The proposed method obtains AUC of 86.45 and EER or 19.58, which are the best performance on our experiments. The conventional machine learning approaches using HMM and DBN achieves relatively lower performance than our method and other recently proposed methods. Yuan *et al.* [9] and Sharma *et al.* [10] uses LSTMs, so those methods are methodologically similar to our methods. However, our method also outperforms these approaches. Yuan *et al.* [9] produces AUC of 80.03 and EER of 27.14, and Sharma *et al.* [10] achieves AUC of 77.98 and EER of 28.68. These experimental results show the proposed PAM is helpful in deriving the probabilistic models from a given normal event dataset. In terms of execution time, the proposed method takes only 0.003 seconds per sample, which is the shortest time among the compared methods.

TABLE 4. AIED performance comparison using AUC, EER, and execution speed on test dataset. 'Env' defines types of computational systems. Our method achieves the best performances. Our method produces AUC of 86.45 and EER of 19.58.

Method	Env	AUC	EER	Execution speed (s)
HMM	CPU	56.70	59.02	0.042
DBN	CPU	60.56	47.62	0.02
Tuor *et al.* [15]	GPU	61.52	42.36	3.26
Yuan *et al.* [9]	GPU	80.03	27.14	6.52
Sharma *et al.* [10]	GPU	77.98	28.68	0.095
Ours	GPU	**86.45**	**19.58**	**0.003**

In addition to the quantitative performance, in Figure 5, ROC curves show the performance according to varying discrimination thresholds. ROC curve of our method is positioned higher location than the compared methods. It means that the proposed method provides more discriminative AIED ability than other methods while showing a low FPR at a high TPR. Consequently, the presented experimental results show that the proposed method can provide promising performance in detecting abnormal insider events compared to other state-of-the-art methods, and is executed in a very short time.

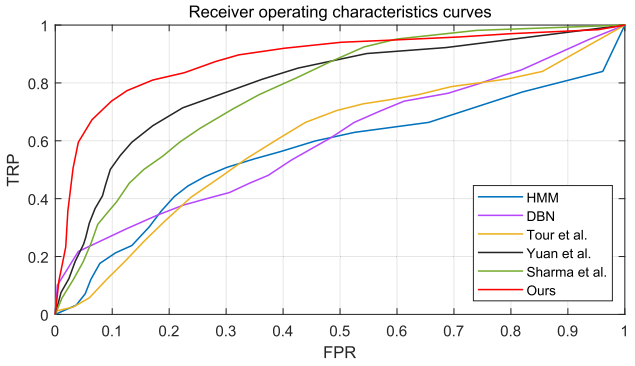


FIGURE 5. ROC curves of the proposed methods, HMM, DBN, and others [9], [10], [15] on our test dataset.

V. CONCLUSION

We have presented a real-time AIED method for ERP systems based on PAM. The proposed PAM provides a way to derive a discriminative probabilistic model for normal event samples. Based on the probabilistic model, PAM outcomes the prediction results about the current events. The proposed method detects enterprise threatening events if the predicted results show a threshold-over error compared with future events. To demonstrate the effectiveness of our method for AIED, we have constructed a dataset containing SAP-SAL codes recorded in a real-world enterprise. The experimental results demonstrate that the proposed method can identify the enterprise threat on ERP systems with real-time execution speed by producing promising performance. Even though our methods shows outstanding AIED performance, some drawbacks should be resolved in the future. Primarily, it is necessary to be provided with a great number of normal event samples to obtain a good solution that can be used in a real-world ERP system. Next, since the proposed method aims to derive a probabilistic model for a single class (*a.k.a.,* normal events), when a training dataset is polluted, which mean a training dataset contains abnormal samples unintended, the AIED performance would be significantly degraded. To overcome these issues, we have planned to employ an online learning model which a self-supervised manner.

REFERENCES

[1] L. D. Xu, "Enterprise systems: State-of-the-art and future trends," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 630–640, Nov. 2011.

[2] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*, Reading, MA, USA: Addison-Wesley, 2012.

[3] *2020 Cost of Insider Threats Global Report*, Ponemon, Boston, MA, USA, 2020.

[4] H. Schulze, "2020 insider threat survey report," GURUCUL, Maharashtra, India, Tech. Rep., 2020.

[5] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control: Towards a unified standard," in *Proc. 5th ACM Workshop Role-Based Access Control (RBAC)*, Jul. 2000, pp. 344287–344301.

[6] A. K. Islam, M. Corney, G. Mohay, A. Clark, S. Bracher, T. Raub, and U. Flegel, "Fraud detection in erp systems using scenario matching," in *Proc. IFIP Int. Inf. Secur. Conf.* New York, NY, USA: Springer, 2010, pp. 112–123.

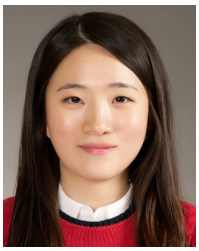
[7] X. Lu, M. Nagelkerke, D. V. D. Wiel, and D. Fahland, "Discovering interacting artifacts from ERP systems," *IEEE Trans. Services Comput.*, vol. 8, no. 6, pp. 861–873, Nov. 2015.

[8] T. Hu, W. Niu, X. Zhang, X. Liu, J. Lu, and Y. Liu, "An insider threat detection approach based on mouse dynamics and deep learning," *Secur. Commun. Netw.*, vol. 2019, pp. 1–12, Feb. 2019.

[9] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," in *Proc. Int. Conf. Comput. Sci.* New York, NY, USA: Springer, 2018, pp. 43–54.

[10] B. Sharma, P. Pokharel, and B. Joshi, "User behavior analytics for anomaly detection using LSTM Autoencoder–Insider threat detection," in *Proc. 11th Int. Conf. Adv. Inf. Technol.*, Jul. 2020, pp. 1–9.
[11] T. Rashid, I. Agrafiotis, and J. R. C. Nurse, "A new take on detecting insider threats: Exploring the use of hidden Markov models," in *Proc. 8th ACM CCS Int. Workshop Manag. Insider Secur. Threats*, Oct. 2016, pp. 47–56.
[12] Q. Hu, B. Tang, and D. Lin, "Anomalous user activity detection in enterprise multi-source logs," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 797–803.
[13] L. Sun, S. Versteeg, S. Boztas, and A. Rao, "Detecting anomalous user behavior using an extended isolation forest algorithm: An enterprise case study," 2016, *arXiv:1609.06676*. [Online]. Available: https://arxiv.org/abs/1609.06676
[14] S. Yuan and X. Wu, "Deep learning for insider threat detection: Review, challenges and opportunities," 2020, *arXiv:2005.12433*. [Online]. Available: https://arxiv.org/abs/2005.12433
[15] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017, *arXiv:1710.00811*. [Online]. Available: https://arxiv.org/abs/1710.00811
[16] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1285–1298.
[17] S. Bengio and Y. Bengio, "Taking on the curse of dimensionality in joint distributions using neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 550–557, May 2000.
[18] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," *Neurocomputing*, vol. 174, pp. 806–814, Jan. 2016.
[19] J. Yu, Y. Lee, K. C. Yow, M. Jeon, and W. Pedrycz, "Abnormal event detection and localization via adversarial event prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 3, 2021, doi: 10.1109/TNNLS.2021.3053563.
[20] J. Yu, K. C. Yow, and M. Jeon, "Joint representation learning of appearance and motion for abnormal event detection," *Mach. Vis. Appl.*, vol. 29, no. 7, pp. 1157–1170, Oct. 2018.
[21] S. Pidhorskyi, R. Almohsen, D. A Adjeroh, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," 2018, *arXiv:1807.02588*. [Online]. Available: https://arxiv.org/abs/1807.02588
[22] J. Yu, D. Y. Kim, Y. Lee, and M. Jeon, "Unsupervised pixel-level road defect detection via adversarial Image-to-Frequency transform," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1708–1713.
[23] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 1–38, 2010.
[24] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Phys. D, Nonlinear Phenomena*, vol. 404, Mar. 2020, Art. no. 132306.
[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: https://arxiv.org/abs/1412.3555
[26] D. P Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*. [Online]. Available: https://arxiv.org/abs/1312.6114
[27] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, vol. 4, Apr. 2007, p. 317.
[28] H. Apaydin, H. Feizi, M. T. Sattari, M. S. Colak, S. Shamshirband, and K.-W. Chau, "Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting," *Water*, vol. 12, no. 5, p. 1500, May 2020.
[29] F. Buchholz. *Analysis and Recommended Settings of the Security Audit Log*. Accessed: Mar. 14, 2021. [Online]. Available: https://blogs.sap.com/2014/12/11/analysis-and-recommended-settings-of-t% he-security-audit-log-sm19-sm20/
[30] SAP. *ERP and Financial Management*. Accessed: Mar. 14, 2021. [Online]. Available: https://www.sap.com/products/erp-financial-management.html
[31] A. Pang, M. Markovski, and A. Micik. *Top 10 ERP Software Vendors, Market Size and Market Forecast 2019-2024*. Accessed: Mar. 14, 2021. [Online]. Available: https://www.appsruntheworld.com/top-10-erp-software-vendors-and-market-% forecast/
[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[33] J. Tu, H. Liu, F. Meng, M. Liu, and R. Ding, "Spatial-temporal data augmentation based on LSTM autoencoder network for skeleton-based human action recognition," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 3478–3482.
[34] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen, "The infinite hidden Markov model," in *Proc. Neural Inf. Process. Syst.*, vol. 14. 2002, pp. 577–584.
[35] Z. Wang, E. E. Kuruoglu, X. Yang, Y. Xu, and T. S. Huang, "Time varying dynamic Bayesian network for nonstationary events modeling and online inference," *IEEE Trans. Signal Process.*, vol. 59, no. 4, pp. 1553–1568, Apr. 2011.

JONGMIN YU received the Ph.D. degree from the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea. He was a Visiting Researcher with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia. He is currently a Research Associative with the Institute for IT Convergence, Korea Advanced Institute of Science and Technology (KAIST). His current research interests include artificial intelligence, machine learning, pattern recognition, and mathematical understanding of these.



MINKYUNG KIM received the B.E. degree from the School of Electrical and Computer Engineering, University of Seoul, in 2016, and the M.S. degree in electrical engineering from KAIST, in 2018, where she is currently pursuing the Ph.D. degree. Her research interests include time series analysis, machine learning. and anomaly detection.

HYEONTAEK OH (Member, IEEE) received the Ph.D. degree in electrical engineering from KAIST. He is currently a Team Leader with the Institute for IT Convergence, Korea Advanced Institute of Science and Technology (KAIST). His research interests include ICT environments, personal data ecosystems, the Internet of Things (IoT), and Web technologies.

JINHONG YANG received the Ph.D. degree from the Department of Information and Communications Engineering, KAIST, South Korea, in 2017. He was the Chief Technology Officer (CTO) with HECAS Inc., developed ultralow latency mobile video streaming technology. In March 2018, he joined Inje University, South Korea, as an Assistant Professor. His research interests include CPS, the IoT, and data privacy.